UNIVERSITÀ DEGLI STUDI DI FIRENZE
Dipartimento di Sistemi e Informatica

Dottorato di Ricerca in
Ingegneria Informatica e dell'Automazione
XVI Ciclo

Kernel Methods, Multiclass Classification and Applications to Computational Molecular Biology

Andrea Passerini

Dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Computer and Control Engineering

Ph.D. Coordinator
Prof. Edoardo Mosca

Advisors
Prof. Paolo Frasconi

Prof. Giovanni Soda

Anno Accademico 2003-2004

Abstract

Support Vector Machines for pattern recognition were initially conceived for the binary classification case. A common approach to address multi-classification problems with binary classifiers is that of reducing the multiclass problem to a set of binary sub-problems, and combine their predictions in order to obtain a multiclass prediction. Reduction schemes can be represented by Error Correcting Output Codes, while binary predictions can be combined using a decoding function which outputs a score for each possible class. We propose a novel decoding function which computes the conditional probability of the class given the binary predictions, and present an extensive set of experiments showing that it outperforms all the decoding functions commonly used in practice. An alternative approach for solving multiclass problems is that of directly extending binary classification algorithms to the multiclass case. Various multicategory extensions for SVM have been proposed so far. We review most of them showing their similarities and differences, as well as the connections to the ECOC schemes. We report a series of experiments comparing different multiclass methods under various conditions, showing that while they perform similarly at the optimum, interesting differences emerge when forced to produce degenerate solutions. Moreover, we present a novel bound on the leave-one-out error of ECOC of kernel machines, showing that it can be successfully employed for model selection.

In the second part of this thesis, we present applications of kernel machines to problems in computational molecular biology. We address the problem of predicting disulphide bridges between cysteines, which represent strong constraints on proteins 3D structure, and whose correct location can significantly help the overall folding prediction. The problem of disulphide bridges prediction can be divided in two successive steps: firstly, for each cysteine in a given protein, predict whether it is involved or not in a disulphide bond; secondly, given the subset of disulphide bonded cysteines in the protein, predict their connectivity pattern, by coupling each cysteine with the correct partner. We focus on the first step, and develop state-of-the-art learning algorithms combining kernel machines and connectionist models. Disulphide bridges are not the only type of binding a cysteine can be involved in, and many cysteines actually bind different types of ligands, usually including metal ions, forming complexes which play very important roles in biological systems. We employed kernel machine algorithms to learn to discriminate between ligand bound and disulphide bound cysteines, in order to get deeper insights into the role of each cysteine in a given protein. We developed ad-hoc kernels able to exploit information on residues similarity, showing that they obtain performances similar to standard kernels with much simpler models.

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would really like to thank my advisors Paolo Frasconi and Giovanni Soda for encouraging me in the decision to undertake this rewarding path, and all people in the Machine Learning and Neural Network Group for being good fellow travellers. Various researchers contributed to the results in different parts of this thesis, like Alessio Ceroni, Paolo Frasconi, Massimiliano Pontil and Alessandro Vullo, and I should also thank Fabrizio Costa and Sauro Menchetti for fruitful discussions.

Huge thanks go to my family, my friends and especially my girlfriend, for bearing me in the zombi condition I assumed while writing most of this thesis. Special thanks should finally go to good old Chianti wine for its contribution in inspiring some of the ideas contained in this thesis.

Chapter 1

Introduction

Learning from examples can be seen as the problem of approximating an unknown function given a finite number of possibly noisy input output pairs. A learning algorithm is characterized by the set of candidate functions, also called the hypothesis space, and the search strategy within this space. The sparseness and finiteness of training data poses the problem of generalization to unseen cases, that is the capacity of the learned function to predict the correct output for an unseen input. A learning algorithm which simply outputs the function which best fits training data, choosing from the set of all possible functions going from the input to the output space, would simply memorize training examples without really developing a model of the underlying target function, and fail to generalize to unseen cases. Moreover, the problem is ill-posed, as there is no unique solution. The problem of avoiding *overfitting* [MP92] training data is typically addressed by restricting the set of allowed candidate functions, either by directly reducing the hypothesis space, by acting on the search strategy, or both, and is termed in different ways, such as bias-variance dilemma [GBD92], inductive bias [Mit97] or capacity control [GVB$^+$92]. In regularization theory, turning an ill-posed problem into a well-posed one is done by adding a regularization term to the objective function [Tik63], which corresponds in the learning framework to modifying the search strategy by trading off between training data fitting and complexity of the learned function. This concepts were investigated by Vapnik in the late Seventies [Vap79] and led to the development of the Support Vector Machine [Vap95] learning algorithm, later generalized to Kernel Machines. In the case of classification tasks, kernel machine algorithms learn a decision function which separates examples with a large margin, possibly accounting for training errors, thus actually trading off between function complexity and fitting the training data. Versions of the algorithm have been developed for tasks different from classification, such as regression [Vap95], clustering [BHHSV01] and ranking [FSS98, CD02a], and have been successfully applied to a vast range of learning tasks, from handwritten digit recognition [CV95] to text categorization [Joa98b].

This thesis is divided in two parts: the first one deals with kernel machines in a theoretical way, providing an extensive review of kernel methods, and presenting results for multiclass classification learning. The second part deals with applications of kernel methods to protein structure predictions, presenting results of applications for prediction of cysteine bonding state.

1.1 Kernel Machines and Multiclass Classification

Support Vector Machines for pattern recognition have been originally conceived for binary classification tasks. A common approach to address multi-classification problems with binary classification algorithms is that of dividing the multiclass problem into several binary subproblems, and combining the predictions for such subproblems into a multiclass prediction. Common examples of this approach include the one-vs-all [Nil65] strategy and the all-pairs [Fri96] one. A general framework for such approaches is represented by Error Correcting Output Codes [DB95, ASS00]. They consist of a coding matrix of values in $\{-1, 0, +1\}$ with a number of rows equal to the number of classes. Each row is a codeword for the corresponding class, while each column induces a binary classification task, which consists of discriminating the subset of classes having $+1$ in the column from that of classes having -1, while classes having 0 are not considered. In order to classify an unseen example, the vector of predictions from all binary classifiers is computed, and the example is assigned to the class whose codeword is closest to such vector, where closeness is measured by a given decoding function. We propose a novel decoding function which computes the conditional probability of the class given the vector of predictions, and present an extensive set of experiments showing that it outperforms all the decoding functions commonly used in practice.

An alternative approach to address multi-classification problems is that of directly extending binary classification algorithms to the multiclass case. Various multicategory extensions of support vector machines have been proposed in the literature [Vap98, WW98, BB99, GEPM00, CS00]. We extensively review most of them, focusing on their similarities and differences, and highlight the connections to ECOC schemes basing on the works on continuous codes learning [CS00]. We present experimental results comparing various methods for multiclass classification and discuss their performances, showing that interesting differences emerge when methods are forced to produce a degenerate solution.

Finally, we propose a novel bound on the leave-one-out error of ECOC of kernel machines, which is valid for any monotonic non-increasing margin based decoding function, and show that it can be successfully employed for model selection.

In chapter 2 we present an extensive introduction to kernel methods, while chapter 3 focuses on multiclass categorization. The proposed results on ECOC of kernel machines are based on [PPF02, PPF04], while the section on multicategory SVM and the comparisons between different multiclass methods are not published elsewhere.

1.2 Applications to Bioinformatics

The explosion of genomic scale sequencing projects has provided us with a huge amount of data from a wide variety of different organisms. However, in order to fully exploit the information contained in such data, increasingly difficult analyses are necessary, identifying single genes within a genome, proteins synthesized from such genes, and three dimensional structure of proteins, which is crucial to derive information on their biological function. The gap between available information at different steps is dramatically increasing, urging for automatic algorithms to fill it up. The determination of a protein fold, that is its 3D-structure, given the sequence of its residues is a challenging task in this context. No simple rule, as those discovered for transcription and translation, is available to map sequences of residues to three dimensional coordinates. Experimental methods such as X-Ray crystallography [Dre94] and NMR spectroscopy [W8̈6] are highly expensive and time consuming, and cannot be applied in all situations. However, resolved three dimensional structures provide a valuable source of information which can be employed to predict the structure of new proteins. Given a new protein, the problem of predicting its three dimensional structure depends how the target protein resembles already resolved ones. If resolved proteins with enough sequence similarity with the target protein are available, its fold can be predicted by *comparative modeling* [KNV03] techniques, relying on the observation that the 3D structure of proteins is more conserved than their residue composition during evolution. However, when such similar proteins are not available, different methods have to be used. Difference in primary structure does not always imply difference in fold, as many examples of so called *remote homologues* exist.*Threading techniques* [God03] search for remote homologues of a target protein by trying to fit its sequence into known folds. Machine learning algorithms, included kernel machines, have been successfully employed for remote homology modeling tasks [Nob04]. For proteins with new folds, however, these methods fail, and a *de novo* protein structure prediction [CRBB03] is necessary. This complex task is typically addressed by solving a number of simpler sub-problems and combining their predictions in order to provide an overall fold prediction [Ros98]. Such sub-problems include 1D predictions such as secondary structure and solvent accessibility as well as 2D predictions such as contact maps and disulphide bridges, and many machine learning algorithms have been employed to address them.

In this thesis we address the problem of predicting the bonding state of cysteines, which are the residues involved in the formation of disulphide bridges. Such residues are very reactive, and under favorable conditions pairs of cysteines within a given protein bind together forming a disulphide bridge, which helps stabilizing the three dimensional structure of the protein [WWNS00, Pai00]. By correctly predicting the connectivity pattern of bridges in a protein, strong constraints on the protein fold are obtained, which can be employed to help the overall three dimensional structure prediction. The problem of disulphide bridges prediction is usually divided in two successive steps: firstly, for each cysteine in a protein, predict if it is in an oxidized state, and thus involved in a disulphide bond, or reduced; secondly, given the subset of oxidized cysteines in a protein, their connectivity pattern is predicted, by pairing each cysteine with its correct partner. We focused on the first task, and developed learning algorithms employing kernel machines combined with connectionist models, while the second task was addressed in a parallel project within our lab [?].

Disulphide bridges are not the only type of bond a cysteine can be involved in, as they can also bind different ligands, usually containing metal ions, forming complexes which play very important roles in biological systems [GWG+03, KH04]. We employed kernel machines to learn to discriminate between ligand bound and disulphide bound cysteines, thus obtaining a finer prediction on the role of each cysteine in a given protein. We developed ad-hoc kernels able to exploit information on residues similarity as contained in substitution matrices [McL72, HH92], showing that they obtain performances similar to standard kernels with much simpler models.

In chapter 4 we provide an overview of protein structure, describing both experimental resolution methods and prediction methods. In chapter 5 we describe disulphide bridges prediction, and present our methods for disulphide bonding state prediction described in [FPV02, CFPV03b, CFPV04]. Finally, in chapter 6 we address the problem of discriminating between ligand bound and disulphide bound cysteines, reporting results in [?].

Part I

Kernel Machines and Multiclass Classification

Chapter 2

Kernel Methods

The theory underlying kernel machines was developed by Vapnik [Vap79] in the late Seventies, but it received increasing attention quite recently, with the introduction of Support Vector Machines [Vap95, Vap98] for pattern recognition, at first in their hard margin implementation for linearly separable binary classification problems, and successively extended to accommodate non-linear separations and the possibility of committing errors in the training set. Support Vector Machines have been successfully applied to a wide range of applications, from handwritten digit recognition [CV95] to text categorization [Joa98b]. Versions of the algorithm have been developed to manage learning tasks other than binary classification, such as novelty detection [SWS$^+$00], unsupervised clustering [BHHSV01], regression [Vap95], feature extraction [SSM99, WMC$^+$01, GWBV02] and multiclass classification, which will be discussed in detail in chapter 3. Many tutorials and books have been written on kernel machines (see for example [Bur98a, SS02, CST00]).

In this chapter we will provide a general introduction to kernel machines. In section 2.1 we review the most important results in statistical learning theory justifying the effectiveness of kernel machines in terms of generalization. In section 2.2 we introduce Support Vector Machines, from the initial formulation for linearly separable binary classification problems, to the non-separable and non-linear extensions, together to results aiming to efficiently estimate the generalization performance of the algorithm. In section 2.3 we present different machines for learning tasks other than binary classification, such as regression and clustering. Section 2.4 more formally describes kernel functions, while section 2.5 addresses the problem of designing appropriate kernels to treat different types of data.

2.1 Statistical Learning Theory

The problem of generalization of learning algorithms has been addressed in many different ways, from bias variance tradeoff [GBD92] to overfitting [MP92], and is crucial to the problem of learning from examples. Given a learning task and a finite number of training examples, a too complex machine will perfectly memorize all training data, without being able to make predictions on unseen examples, while a too simple one won't have the power to learn the given task. A balance between machine capacity and performance on the training set must be achieved in order to obtain good generalization [Vap79]. A detailed discussion on statistical learning theory and its implications in kernel machines can be found in [Vap95, Vap98, EPP00, CS01, HTF01].

2.1.1 Loss Function and Risk Minimization

Let $D_m = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^m$ be a training set, whose data are independently drawn and identically distributed with respect to an unknown probability distribution $P(x, y)$. Suppose we have a set of functions $\mathcal{F}_\alpha$ with parameters α, that is for each value of α we have a function $f_\alpha : \mathcal{X} \to \mathcal{Y}$. Let us give a formal definition of the loss incurred by the function f_α at example (x, y).

Definition 2.1.1 (Loss Function) *Given a triplet $(x, y, f_\alpha(x))$ containing a pattern x , its observation y and a prediction $f_\alpha(x)$, we define loss function any map $\ell : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to [0, \infty]$ such that $\ell(x, y, y) = 0$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.*

Common examples of loss functions in the case of binary classification, where $\mathcal{Y} = \{-1, 1\}$, are the misclassification error:

$$\ell(x, y, f_\alpha(x)) = \begin{cases} 0 & \text{if } y = f_\alpha(x) \\ 1 & \text{otherwise} \end{cases} \tag{2.1}$$

and the *soft margin* loss function [BM92] (see fig.2.1(a)):

$$\ell(x, y, f_\alpha(x)) = |1 - yf_\alpha(x)|_+ = \begin{cases} 0 & \text{if } yf_\alpha(x) \geq 1 \\ 1 - yf_\alpha(x) & \text{otherwise} \end{cases} \tag{2.2}$$

which takes into account the confidence of the prediction.

For regression tasks ($\mathcal{Y} = \mathbb{R}$)), common losses are the square error:

$$\ell(x, y, f_\alpha(x)) = (y - f_\alpha(x))^2 \tag{2.3}$$

and the extension of soft margin loss called ϵ-*insensitive* loss (see fig.2.1(b)):

$$\ell(x, y, f_\alpha(x)) = |y - f_\alpha(x)|_\epsilon = \begin{cases} 0 & \text{if } |y - f_\alpha(x)| \leq \epsilon \\ |y - f_\alpha(x)| - \epsilon & \text{otherwise} \end{cases} \tag{2.4}$$

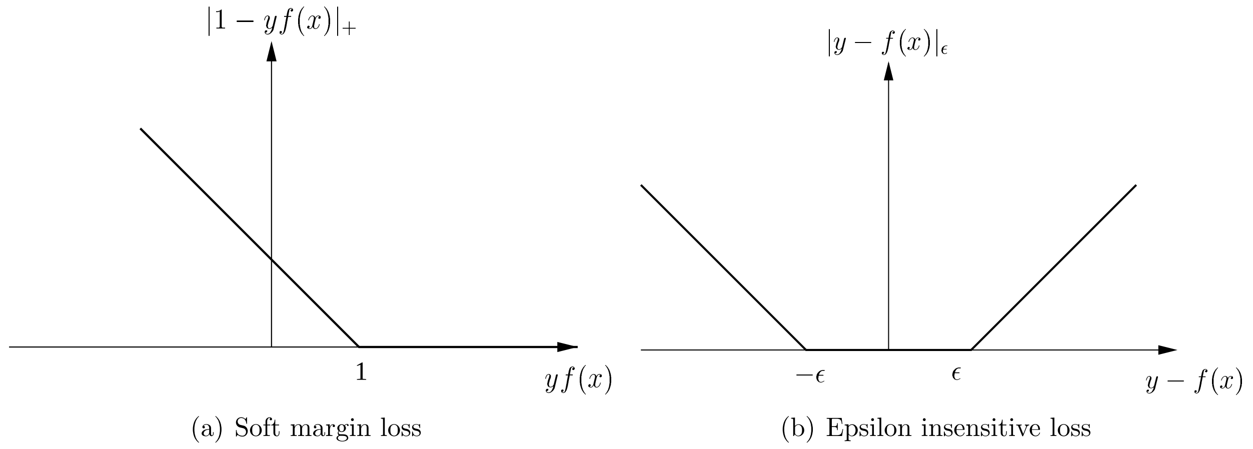


Figure 2.1. Confidence-based losses for binary classification (a) and regression (b).

which doesn't penalize deviations up to ϵ from the target value, and gives a linear penalty to further deviations. Note that all these losses only depend on x by $f_\alpha(x)$, while definition 2.1.1 is more general.

Given a loss function to weight errors on individual patterns, we can define the expectation of the test error for a trained function on the entire set of possible patterns.

Definition 2.1.2 (Expected Risk) *Given a probability distribution $P(x, y)$ of patterns and observations, a trained function $f_\alpha : \mathcal{X} \to \mathcal{Y}$ and a loss function $\ell : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$, the expected risk for f_α is defined as*

$$R[f_\alpha] = \int_{\mathcal{X} \times \mathcal{Y}} \ell(x, y, f_\alpha(x)) dP(x, y). \tag{2.5}$$

The expected risk is also known as generalization or true error. We cannot directly minimize such risk, as the probability distribution $P(x, y)$ is unknown. The only error we can actually measure is the mean error rate on the training set D_l, called the empirical risk.

Definition 2.1.3 (Empirical Risk) *Given a training set $D_m = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^m$ of patterns and observations a trained function $f_\alpha : \mathcal{X} \to \mathcal{Y}$ and a loss function $\ell : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$, the empirical risk for f_α is defined as*

$$R_{emp}[f_\alpha] = \frac{1}{m} \sum_{i=1}^{m} \ell(x_i, y_i, f_\alpha(x_i)). \tag{2.6}$$

Minimizing this risk alone, however, doesn't give any guarantee on the value of the expected risk itself. If we choose the set of functions $\mathcal{F}_\alpha$ to be the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$, we can always find a function which has zero empirical error, mapping each training pattern x_i to its observation y_i, and maps each other pattern $x_j, j > m$ to a fixed value, thus achieving no learning at all.

In order to generalize to unseen patterns, we have to restrict the set of possible learning functions, taking into account the *complexity* or *capacity* of such set with respect to the learning task and the number of training examples available.

2.1.2 VC Dimension and Bounds on Expected Risk

A well known measure of complexity for a set of functions $\mathcal{F}_\alpha$ is the Vapnik Chervonenkis (VC) dimension [VC71]. In the case of binary classification, a set of points m is *shattered* by the set of functions $\mathcal{F}_\alpha$ if for each of the 2^m possible labellings of the points, there exist a function in $\mathcal{F}_\alpha$ which correctly assigns all labels. The *VC dimension* of $\mathcal{F}_\alpha$ is given by the maximum m for which a set of m points shattered by $\mathcal{F}_\alpha$ exists. If the maximum does not exist, the VC dimension is said to be infinite. It can be proved [Bur98a] that the VC dimension of a set of oriented hyperplanes in $\mathbb{R}^n$ is $n+1$.

Vapnik [Vap95] derived a bound on the expected risk which holds with probability $1-\eta$ for $\eta \in [0,1]$, and depends only on the empirical risk and the so called *VC confidence*:

$$R[f_\alpha] \leq R_{emp}[f_\alpha] + \sqrt{\left(\frac{h(\log 2m/h + 1) - \log \eta/4}{m}\right)} \tag{2.7}$$

Fixing the probability η and the training set size m, the VC confidence is a monotonic increasing function of the VC dimension h. That is, given two learning functions with equal empirical risk, the one associated to the set of functions with smaller VC dimension will result in a better upper bound on the expected risk.

2.1.3 Structural Risk Minimization

The bound on the expected risk in equation 2.7 leads to the induction principle called *structural risk minimization* [Vap79]. The VC confidence term of the bound depends on the set of functions under investigation, while the empirical risk depends on the particular function chosen by the training procedure. In order to minimize the bound with respect to these two terms, we first divide the entire class of possible functions into nested subsets with decreasing VC dimension. Within each subset, we obtain a trained function by minimizing the empirical risk only. Finally we choose the trained machine for which the sum of empirical risk and VC confidence is minimal (see fig. 2.2).

2.1.4 Empirical Estimates of the Expected Risk

The expected risk of a learned function can be empirically estimated by a *hold out* procedure, which consists of training on a subset of the available data, and testing the learned function on the remaining unseen cases. When available data are few, however, more complex procedures are usually employed, in order to reduce the bias induced by the choice of the train/test split. A very common procedure is the so called *k-folds cross validation.*

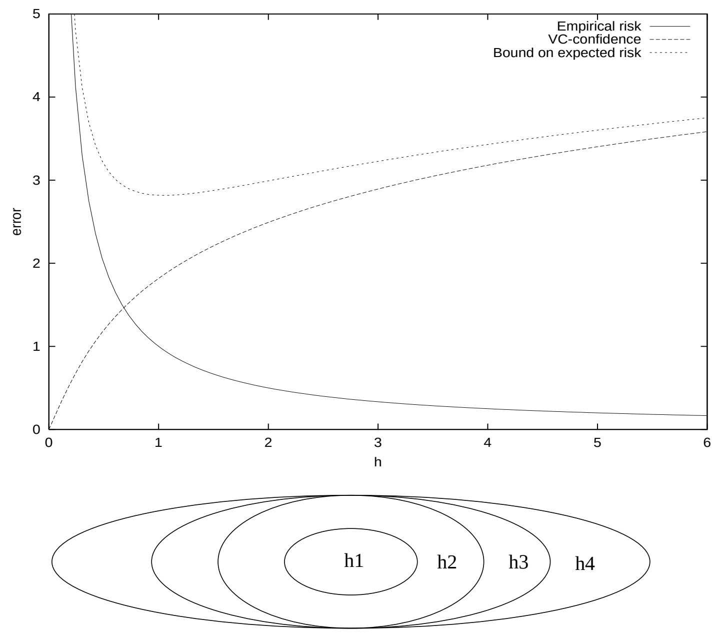


Figure 2.2. Structural risk minimization (SRM) induction principle: the entire class of functions is divided into nested subsets with decreasing VC dimension. Within each subset, a trained function is obtained by minimizing the empirical risk only. Finally, we choose the training function which minimizes the bound on the expected risk as given by the sum of the empirical risk and the VC confidence.

The training set $D_m = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^m$ is randomly split into k mutually exclusive subsets $D_m^1, \dots, D_m^k$ of approximately equal size. The learning function is trained k times, each time t training on $D_m \backslash D_m^t$ and testing on D_m^t. The cross validation error estimate of the function trained on the full dataset is given by the number of errors committed in all the tests of the procedure, divided by the size of the dataset:

$$R_{kcv}^m[f_\alpha] = \frac{1}{m} \sum_{t=1}^{k} \sum_{i \in D_m^t} \ell(x_i, y_i, f_\alpha^{D_m^t}(x_i)), \tag{2.8}$$

where $f_\alpha^{D_m^t}$ is the function trained on the subset $D_m \backslash D_m^t$, and ℓ is the loss function (see section 2.1.1).

A similar approach is followed in the *leave one out* (LOO) procedure, where the learning function f_α^i is iteratively trained on $D_m \backslash \{(x_i, y_i)\}$ and tested on the remaining example (x_i, y_i), for all i in $[1, m]$. The LOO error of the function f_α trained over the entire set D_m is defined as

$$R_{loo}^m[f_\alpha] = \frac{1}{m} \sum_{i=1}^{m} \ell(x_i, y_i, f_\alpha^i(x_i)). \tag{2.9}$$

The popularity of the LOO error estimate is mostly due to the fact that it is an almost unbiased estimate of the generalization error, as showed in the following theorem [LB67].

Theorem 2.1.1 (Bias of Leave One Out Estimator) *The leave one out estimator is almost unbiased, that is*

$$E[R^m_{loo}[f_\alpha]] = E[R^{m-1}[f_\alpha]], \tag{2.10}$$

where the expectation on the left hand site is over training sets of size m, while the one on the right hand side is over training sets of size $m-1$.

On the other hand, the variance of the LOO error estimator can be large [Bur89]. A bound on the variability of such estimator, based on the VC dimension of the class of learning functions, is presented in the following theorem [KR97].

Theorem 2.1.2 (Variability of Leave One Out Estimator) *Let f_α be a function learned by empirical error minimization over a class of functions with VC dimension h. Then for every $\nu > 0$, with probability $1-\nu$,*

$$|R^m_{loo}[f_\alpha] - R^m[f_\alpha]| \leq \frac{8\sqrt{\frac{(h+1)(\log(9m/h)+2)}{m}}}{\nu} \tag{2.11}$$

For a detailed review on estimators of generalization error see [Koh95], while [EP03] studies the advantages of using the LOO error estimator in terms of stability of learning algorithms.

2.2 Support Vector Machines

Support Vector Machines [CV95, Vap95, Vap98] (SVM) are the first effective application of the principles of structural risk minimization, by providing a learning procedure which tries to fit the training data while keeping the VC dimension of the class of candidate functions as low as possible. While the simplest formulation of SVM deals with linearly separable data, extensions have been developed [CV95] to allow violations of the separability constraint, and to implement nonlinear separations [BGV92].

2.2.1 Hard Margin Hyperplanes

Let $\mathcal{X}$ be a space endowed with a dot product $< \cdot, \cdot >$. Any hyperplane in such space can be represented as

$$\{\mathbf{x} \in \mathcal{X} | < \mathbf{w}, \mathbf{x} > + b \ = \ 0\}, \mathbf{w} \in \mathcal{X}, b \in \mathbb{R}, \tag{2.12}$$

where $\mathbf{w}$ is normal to the hyperplane and $|b|/||\mathbf{w}||$ is the distance of the hyperplane from the origin. This formulation still allows to obtain an infinite number of equivalent hyperplanes by multiplying both $\mathbf{w}$ and b by the same non-zero constant. We thus define the pair $(\mathbf{w}, b) \in \mathcal{X} \times \mathbb{R}$ a *canonical* form for the hyperplane (2.12) with respect to the set $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathcal{X}$ if it is scaled such that

$$\min_{i=1,\ldots,m} | < \mathbf{w}, \mathbf{x}_i > + b \,| = 1. \tag{2.13}$$

This implies that the minimal distance from the hyperplane is equal to $1/||\mathbf{w}||$.

Let $D_m = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \{\pm 1\}\}_{i=1}^m$ be a training set. We want to find a hyperplane which separates the positive from the negative examples, that is a decision function

$$f_{\mathbf{w},b}(\mathbf{x}) = \mathrm{sgn}(< \mathbf{w}, \mathbf{x} > + b \,) \tag{2.14}$$

satisfying

$$f_{\mathbf{w},b}(\mathbf{x}_i) = y_i \quad \forall\, (\mathbf{x}_i, y_i) \in D_m.$$

Suppose such hyperplane exists, that is the training set D_m is linearly separable. The canonical form (2.13) for the hyperplane implies

$$y_i(< \mathbf{w}, \mathbf{x}_i > + b \,) \geq 1 \quad \forall\, (\mathbf{x}_i, y_i) \in D_m.$$

The sum of the minimal distances on both sides of the separating hyperplane is called the (geometric) *margin* and is equal to $2/||\mathbf{w}||$, while the hyperplane itself is called a *hard margin* hyperplane. A slightly different definition of margin is that indicating the confidence of the predictions of the decision function. We define the (confidence) margin of $f_{\mathbf{w},b}$ on D_m as

$$\min_{D_m} y_i(< \mathbf{w}, \mathbf{x}_i > + b \,),$$

which is always equal to one for canonical hyperplanes and linearly separable sets. Given an example $(\mathbf{x}_i, y_i)$, we also say it's classified with confidence margin equal to

$$y_i(< \mathbf{w}, \mathbf{x}_i > + b \,),$$

which can be negative if the example is incorrectly classified. In the following, unless otherwise specified, we will always write margin referring to the geometric margin.

It's rather intuitive to see that, in order to generalize well, we should choose a separating hyperplane with the largest possible margin. A more formal justification of large margin algorithms is given in the following theorem [SS02, BST99].

Theorem 2.2.1 (Margin Error Bound) *Consider the set of decision functions $f(\mathbf{x}) = \mathrm{sign} < \mathbf{w}, \mathbf{x} >$ with $||\mathbf{w}|| \leq \Lambda$ and $||\mathbf{x}|| \leq R$, for some $R, \Lambda > 0$. Moreover, let $\rho > 0$ and ν denote the fraction of training examples with margin smaller than $\rho/||\mathbf{w}||$, referred to as the* margin error*.*

For all distributions P generating the data, with probability at least $1 - \delta$ over the drawing of the m training patterns, and for any $\rho > 0$ and $\delta \in (0, 1)$, the probability that a test pattern drawn from P will be misclassified is bound from above by

$$\nu + \sqrt{\frac{c}{m}\left(\frac{R^2\Lambda^2}{\rho^2} \ln^2 m + \ln(1/\delta)\right)}. \tag{2.15}$$

Here, c is a universal constant.

It's interesting to note the similarity between this bound and the bound on the expected risk of equation (2.7). The margin error bound is still the sum of a training error (the margin error ν) and a capacity term. This second term tends to zero as m tends to infinity, and is proportional to R and Λ. With such values fixed, the capacity term is inversely proportional to ρ. However, increasing ρ leads to an increase of the margin error ν, as there will be more training patterns with margin smaller than $\rho/||\mathbf{w}||$. By finding a balance between these two opposite behaviours we can minimize the test error probability and achieve good generalization. Finally, note that maximizing ρ equals to minimizing $||\mathbf{w}||$ for a fixed value of ρ (i.e. $\rho = 1$ in canonical hyperplanes).

We can now construct a large margin separating hyperplane by solving the following problem:

$$\min_{\mathbf{w}\in\mathcal{X}, b\in\mathbb{R}} \quad \frac{1}{2}||\mathbf{w}||^2 \tag{2.16}$$

$$\text{subject to} \quad y_i(<\mathbf{w}, \mathbf{x}_i> + \, b\,) \geq 1 \quad \forall\, (\mathbf{x}_i, y_i) \in D_m. \tag{2.17}$$

Note that this is a convex quadratic optimization problem, since the objective function is convex, and the points satisfying the constraints form a convex set. Therefore it admits a unique global minimum $(\mathbf{w}^*, b^*)$.

The constraints (2.17) can be included by means of a set of Lagrange multipliers $\alpha_i \geq 0$, giving the Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{m} \alpha_i(y_i(<\mathbf{w}, \mathbf{x}_i> + \, b\,) - 1), \tag{2.18}$$

which must be maximized with respect to α_i and minimized with respect to $\mathbf{w}$ and b (the solution is termed a *saddle point*). According to the KKT theorem [Fle87], at the saddle point either $\alpha_i = 0$ or $\alpha_i > 0$ and $y_i(<\mathbf{w}, \mathbf{x}_i> + \, b\,) - 1 = 0$, the latter meaning that pattern $\mathbf{x}_i$ is at the minimal distance of the separating hyperplane. Such points are called *Support Vectors*, and are the only critical points of the training set, staying on the two closest hyperplanes H_1 and H_{-1} on both sides of the separating one (see fig 2.3). Removing all other points would lead to the same separating hyperplane.

The optimization problem can be turned to its dual formulation, called the Wolfe dual [Fle87]. which has some interesting properties, both for efficiency and for subsequent expansions to non linear cases. By vanishing the derivatives with respect to the primal variables we obtain:

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \quad \Rightarrow \quad \sum_{i=1}^{m} \alpha_i y_i = 0, \tag{2.19}$$

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i. \tag{2.20}$$

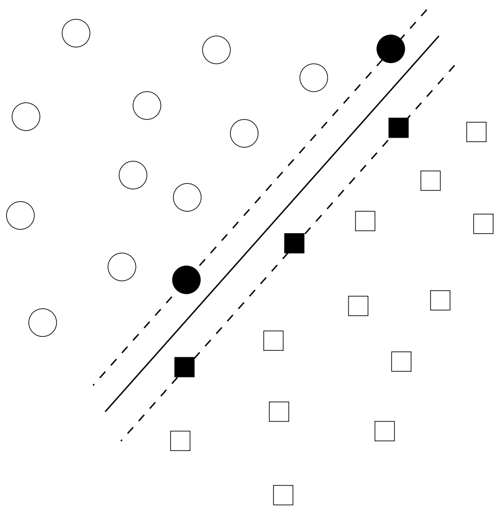


Figure 2.3. Separable classification problem solved by support vector machines. The solid line represents the separating hyperplane. Support vectors (in black) are the points lying on the two closest hyperplanes on both sides of the separating one, corresponding to a confidence margin of one. All other points (in white) do not contribute to the decision function.

Substituting the conditions (2.19) and (2.20) in the Lagrangian (2.18), we obtain the dual form of the optimization problem:

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^m} \quad \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j < \mathbf{x}_i, \mathbf{x}_j >, \tag{2.21}$$

$$\text{subject to} \quad \alpha_i \geq 0 \quad i = 1, \dots, m, \tag{2.22}$$

$$\sum_{i=1}^{m}\alpha_i y_i = 0. \tag{2.23}$$

Note that the dual form (2.21) has a number of variables equal to the number of training patterns m , while in the primal one (2.16) they are equal to the dimension of the pattern space $\mathcal{X}$. The proportion between these two numbers can suggest which of the two optimization problems is the more efficient to solve.

Substituting the expansion of $\mathbf{w}$ (2.20) into the decision function (2.14) we obtain its formulation in terms of the training patterns $\mathbf{x}_i$

$$f_{\boldsymbol{\alpha},b}(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i\alpha_i < \mathbf{x}, \mathbf{x}_i > + b\right), \tag{2.24}$$

where it is clear that only support vectors, for which $\alpha_i > 0$, contribute to the decision function.

2.2.2 Soft Margin Hyperplanes

Hard margin hyperplanes can only be found for linearly separable problems. Moreover, in order to have a robust classifier in terms of generalization, it is necessary to balance performance on the training set with model complexity, thus tolerating a certain fraction of outliers in the training patterns. This can be achieved [CV95] by introducing *slack* variables relaxing the separation constraints (2.17):

$$y_i(<\mathbf{w}, \mathbf{x}_i> + \, b\,) \geq 1 - \xi_i \quad i = 1, \dots, m, \tag{2.25}$$

with

$$\xi_i \geq 0 \quad i = 1, \dots, m. \tag{2.26}$$

In order to penalize such relaxations we add an error penalty term to the objective function (2.16), which becomes:

$$\min_{\mathbf{w} \in \mathcal{X}, b \in \mathbb{R}, \boldsymbol{\xi} \in \mathbb{R}^m} \quad \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{m}\sum_{i=1}^{m} \xi_i, \tag{2.27}$$

subject to (2.25) and (2.26). Here $C > 0$ is a cost parameter balancing model complexity versus margin errors, given by all nonzero ξ, and can be therefore used to achieve a minimum of the margin error bound of theorem 2.2.1.

It can be shown [CV95] that the Wolfe dual problem has the same formulation as in the separable case (2.21) except for an additional upper constraint C on the α values:

$$0 \leq \alpha_i \leq C/m, \quad i = 1, \dots, m. \tag{2.28}$$

All the *unbound* support vectors (those with $\alpha_i < C/m$) are on the hyperplanes H_1 and H_{-1} closest to the separating one. Conversely, the *bound* ones (for $\alpha_i = C/m$) correspond to violations $\xi_i > 0$ to the separation constraint, and have a distance $\xi_i/||\mathbf{w}||$ from their respective hyperplane. When $\xi_i > 1$ they are also training errors (see fig. 2.4).

2.2.3 Nonlinear Support Vector Machines

Linear support vector machines can deal with linearly separable data, possibly tolerating a certain quantity of outliers in the training patterns. In order to learn decision functions which are not linear in the data, a more general approach is required [BGV92].

The basic idea is that of transforming the input data $\mathbf{x} \in \mathcal{X}$ into a higher dimensional *feature space* $\mathcal{H}$ by a nonlinear map $\Phi : \mathcal{X} \to \mathcal{H}$, and find a separating hyperplane in such space:

$$\{\Phi(\mathbf{x}) \in \mathcal{H} | <\mathbf{w}, \Phi(\mathbf{x})> + \, b \; = \; 0\}, \mathbf{w} \in \mathcal{H}, b \in \mathbb{R}, \tag{2.29}$$

corresponding to a decision function

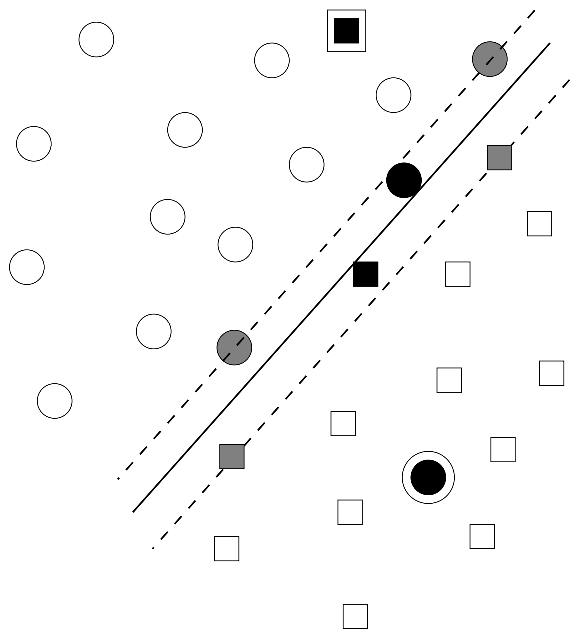
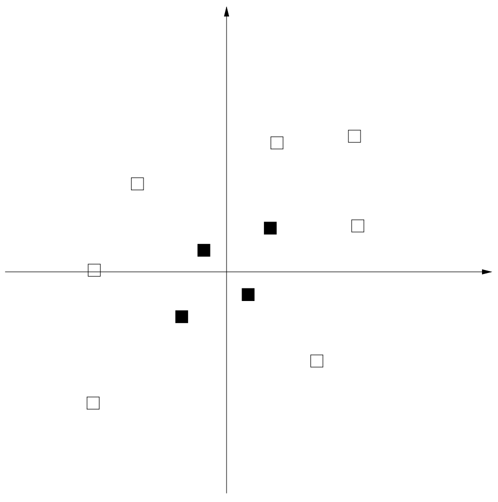


Figure 2.4. Non separable classification problem solved by support vector machines. The solid line represents the separating hyperplane, while dotted lines are hyperplanes with confidence margin equal to one. Grey points are unbound SVs, black points are bound SVs and extra borders indicate bound SVs which are also training errors. All other points do not contribute to the decision function.

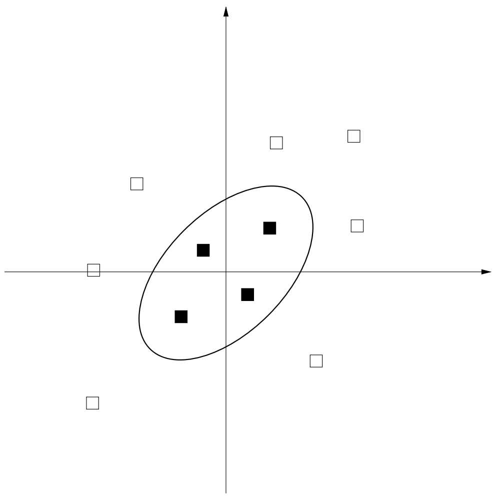
$$f_{\mathbf{w},b}(\Phi(\mathbf{x})) = \mathrm{sgn}(< \mathbf{w}, \Phi(\mathbf{x}) > + b). \tag{2.30}$$

As an example, consider the problem in figure 2.5(a), for which no separating hyperplane in $\mathbb{R}^2$ exists. By the nonlinear map $\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ such that

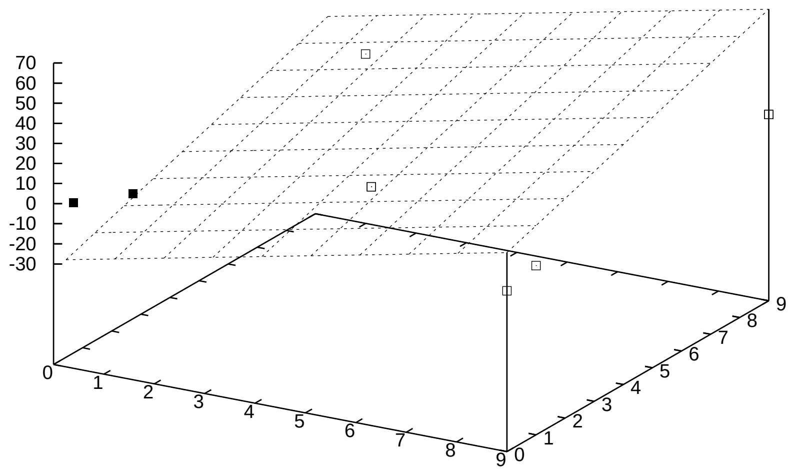
$$\Phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ \sqrt{2}\, x_1 x_2 \\ x_2^2 \end{pmatrix} \tag{2.31}$$



(a)



(b)



(c)

Figure 2.5. (a) No linear decision function can separate black points from white ones. (b) A non linear decision surface correctly separating points. (c) By nonlinearly mapping points into a higher dimensional feature space we can find a separating hyperplane, which corresponds to the nonlinear decision surface (b) in the input space.

we can transform the input data into a higher dimensional feature space where a separating hyperplane $((w_1, w_2, w_3)^T, b)$ does exist (see fig 2.5(c)). Note that the linear decision function in $\mathbb{R}^3$ corresponds to a nonlinear decision surface in $\mathbb{R}^2$ (see fig 2.5(b)):

$$f_{\mathbf{w},b}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \operatorname{sgn}(w_1 x_1^2 + w_2\sqrt{2}x_1x_2 + w_3x_2^2 + b). \tag{2.32}$$

In terms of statistical learning theory it amounts to increasing the capacity of the class of learning functions in order to account for the difficulty of the learning task. The feature space $\mathcal{H}$ can have very high dimension, even infinite, thus apparently contradicting the principle of structural risk minimization (see section 2.1.3). The trick is in the idea of *large margin* hyperplanes, which can have VC dimension much smaller than that of the entire class of hyperplanes. In the following theorem [Vap79], we employed $\hat{\mathbf{x}}$ instead of $\mathbf{x}$ in order to stress their being in a possibly infinite dimensional space $\mathcal{H}$.

Theorem 2.2.2 (VC Dimension of Margin Hyperplanes) *Let $<\mathbf{w}, \hat{\mathbf{x}}>$ be a set of canonical hyperplanes with respect to a set of points $\mathcal{H}^* = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_m\}$.*

Let R be the radius of the smallest sphere centered at the origin and containing $\mathcal{H}^*$.

The set of decision functions $f_{\mathbf{w}}(\hat{\mathbf{x}}) = sign <\mathbf{w}, \hat{\mathbf{x}}>$ defined on $\mathcal{H}^*$, and satisfying the constraint $||\mathbf{w}|| \leq \Lambda$, has a VC dimension satisfying

$$h \leq R^2\Lambda^2. \tag{2.33}$$

The VC dimension can therefore be controlled by increasing the margin of the set of admissible hyperplanes. Extensions of the theorem exist for hyperplanes with a nonzero offset b [Vap98], and for the general case of functions defined on the entire domain $\mathcal{H}$ [BST99].

Considered that the feature space $\mathcal{H}$ can have very high, even infinite dimension, it becomes difficult and computationally expensive to directly compute the transformation Φ. However, this is all but necessary, as in both the dual objective function (2.21) and the expanded decision function (2.24), the inputs $\mathbf{x}$ only appear in the form of dot products $<\mathbf{x}_i, \mathbf{x}_j>$. We can therefore use the so-called *kernel trick*, by employing a *kernel* function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which satisfies

$$k(\mathbf{x}_i, \mathbf{x}_j) = <\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)>, \quad \forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}, \tag{2.34}$$

obtaining the following optimization problem:

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^m} \quad \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m \alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{2.35}$$

with constraints (2.23) and (2.28), which is still quadratic provided k is positive definite (see section 2.4.1). The corresponding decision function is given by:

$$f_{\boldsymbol{\alpha},b}(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^m y_i\alpha_i k(\mathbf{x}, \mathbf{x}_i) + b\right). \tag{2.36}$$

Note that we don't even need to know the true mapping Φ in order to employ kernel k, provided that such a mapping exists.

Generally speaking, the input space $\mathcal{X}$ can now be any set, we just need to find a suitable positive definite kernel for it. This allows to treat non vectorial data such as strings or graphs, as will be discussed in section 2.5.3. From now on, we will write a generic input pattern as x, turning to $\mathbf{x}$ whenever we will need to stress its vectorial nature. Kernels will be more formally defined in section 2.4, where we will also show how to verify if a candidate function is actually a valid positive definite kernel.

2.2.4 Bounds on the LOO Error of SVM

In order to estimate the generalization performance of support vector machines, the LOO error procedure is a desirable approach (see section 2.1.4). However, it is very time consuming, especially if the number of training examples is high. To obtain an efficient estimate of the generalization error, upper bounds on the LOO error have been proposed, which can be computed with less or no effort given the learned machine. Let ℓ be the misclassification loss (eq. (2.1)) for binary classification, which can be equivalently written as [1]

$$\ell(x, y, f(x)) = \theta(-yf(x)), \tag{2.37}$$

where θ is the Heavyside function. The LOO error of f can be written as

$$R^m_{loo}[f] = \frac{1}{m}\sum_{i=1}^{m}\theta(-y_i f^i(x_i)) = \frac{1}{m}\sum_{i=1}^{m}\theta(-y_i f(x_i) + y_i(f(x_i) - f^i(x_i))). \tag{2.38}$$

Thus, by providing an upper bound U_i for $y_i(f(x_i) - f^i(x_i))$, we get the following upper bound on the LOO error:

$$R^m_{loo}[f] \leq \frac{1}{m}\sum_{i=1}^{m}\theta(-y_i f(x_i) + U_i), \tag{2.39}$$

since θ is monotonically increasing.

The simplest bound [Vap95] arises from the fact that removing non support vectors does not change the solution computed by the SVM. The number of support vectors N_{SV} is thus an upper bound for the number of possible errors ($U_i = 2$ if x_i is a support vector), giving the bound

$$R^m_{loo}[f] \leq \frac{N_{SV}}{m}. \tag{2.40}$$

A few bounds have been found for unbiased SVM. Jaakkola and Haussler [JH98] proved the inequality

[1] to simplify notation from henceforth we will drop the explicit dependence of f on its parameters.

$$y_i(f(x_i) - f^i(x_i)) \leq \alpha_i k(x_i, x_i) = U_i, \tag{2.41}$$

leading to the upper bound

$$R^m_{loo}[f] \leq \frac{1}{m}\sum_{i=1}^{m} \theta(-y_i f(x_i) + \alpha_i k(x_i, x_i)). \tag{2.42}$$

Let R be the radius of the smallest sphere in feature space containing all training points x_i for all $i \in [1, m]$, computed as

$$R = \min_{\mathbf{o}\in\mathcal{H}, x_i} ||\Phi(x_i) + \mathbf{o}||, \tag{2.43}$$

where $\mathbf{o}$ is the center of this minimal enclosing sphere. Then Vapnik [Vap98] derived a bound for unbiased hard margin SVM given by

$$R^m_{loo}[f] \leq \frac{1}{m}\frac{R^2}{\gamma^2}, \tag{2.44}$$

where $\gamma^2 = 1/||\mathbf{w}||^2$ is the margin of the separating hyperplane. Note that this bound justifies the idea of constructing large margin hyperplanes, as maximizing the margin corresponds to minimizing the bound, for fixed values of the sphere radius.

Assume that the set of support vectors remains unchanged during the LOO procedure. Then Chapelle et al. [CVBM02] proved the following equality:

$$y_i(f(x_i) - f^i(x_i)) = \alpha_i S_i^2, \tag{2.45}$$

where S_i is called *span* of the point $\Phi(x_i)$ and is its distance from the set Λ_i, where

$$\Lambda_i = \left\{ \sum_{j\neq i, \alpha_j > 0} \lambda_j \Phi(x_j) \mid \sum_{j\neq i} \lambda_j = 1 \right\}. \tag{2.46}$$

We can thus compute the exact number of errors made by the LOO procedure, giving

$$R^m_{loo}[f] = \frac{1}{m}\sum_{i=1}^{m} \theta(-y_i f(x_i) + \alpha_i S_i^2). \tag{2.47}$$

Finally, Joachims [Joa99, Joa00] derived a bound which is valid for general biased soft margin SVM, provided they are *stable*, that is they have at least one unbound support vector. The bound is given by

$$R^m_{loo}[f] \leq |\{i : (2\alpha_i R^2 + \xi_i) \geq 1\}|, \tag{2.48}$$

where ξ_i are the slack variables (see eq. (2.26)) for the non separable case, R^2 is an upper bound on $k(x, x)$ for all x.

2.3 Other Support Vector Methods

The idea of large margin classifiers, as well as that of decision functions characterized by a subset of the training patterns (the support vectors) have been successfully extended to different learning tasks. In regression, large margins correspond to flat estimation functions, while in clustering the task is to find the smallest sphere enclosing the data in the feature space, which gives clusters boundaries when mapped back to the input space. Other learning tasks for which support vector methods have been conceived include ranking [FSS98, CD02a] and multiclass classification, which will be discussed in detail in chapter 3.

2.3.1 Support Vector Regression

The basic ideas of SVM have been later [Vap95] extended to regression estimation, where the task is to learn a real valued function of the data given a finite number of examples $D_m = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathbb{R}\}_{i=1}^m$. In its first formulation, the SV regression algorithm seeks to estimate a linear function of the data:

$$f(\mathbf{x}) \ = < \mathbf{w}, \mathbf{x} > + b, \tag{2.49}$$

where $b \in \mathbb{R}$ and $\mathbf{w} \in \mathcal{X}$.

In order to trade off between function complexity and fitting of the training data, we will introduce an $\epsilon - insensitive$ region [Vap95] around the target function, that is a tolerance on the distance between the target output and the function estimation (see fig 2.6(a)), below which the pattern is considered correctly assigned. Such patterns correspond to those which are separated with a confidence margin greater than one in SVM, and the introduction of the $\epsilon - insensitive$ region permits to obtain a sparse solution in a fashion analogous to that of SVM. The notion of large margin is here replaced by that of *flatness*: by minimizing $||w||$ we search for the flattest estimation function which fits the training data (see fig. 2.6(b)). For a detailed treatment on the connections between large margins and the ϵ margin of regression estimation see [PRE98]. By further introducing a set of slack variables ξ_i,ξ_i^* to allow for errors in the training set, we obtain the following constrained optimization problem:

$$\min_{\mathbf{w}\in\mathcal{X}, b\in\mathbb{R}, \boldsymbol{\xi},\boldsymbol{\xi}^*\in\mathbb{R}^m} \quad \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{m}\sum_{i=1}^m(\xi_i + \xi_i^*), \tag{2.50}$$

$$\text{subject to} \quad < \mathbf{w}, \mathbf{x}_i > +b - y_i \leq \epsilon + \xi_i, \tag{2.51}$$

$$y_i - (< \mathbf{w}, \mathbf{x}_i > +b\ ) \leq \epsilon + \xi_i^*, \tag{2.52}$$

$$\xi_i, \xi_i^* \geq 0, \tag{2.53}$$

where i goes from 1 to m.

In a similar fashion as for SVM, we consider the Lagrangian

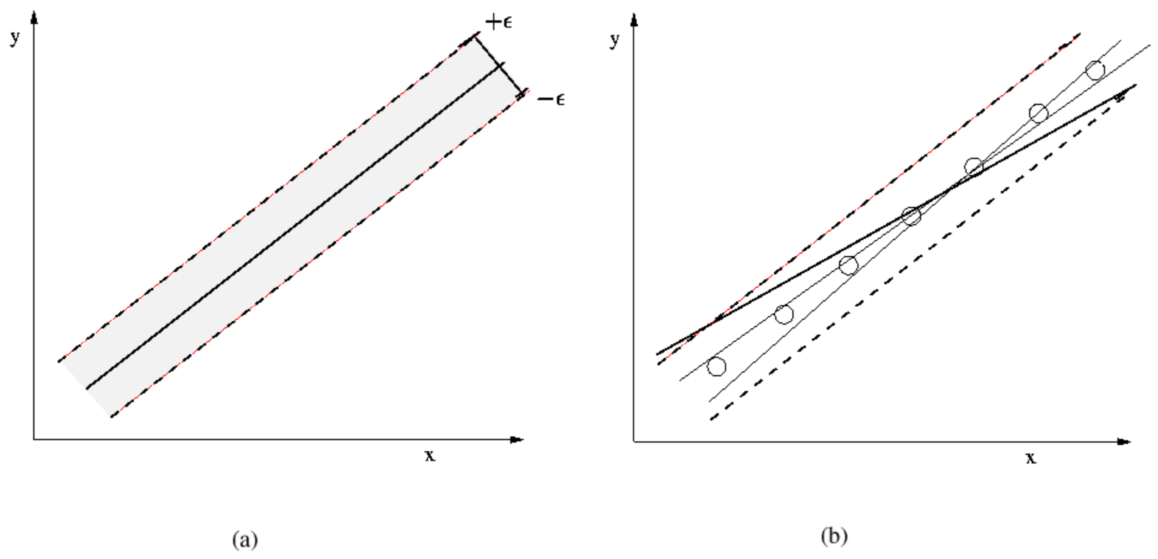


Figure 2.6. (a) ϵ-*insensitive* region around the target function. Points lying within the region are considered as correctly predicted. (b) Large margin separation corresponds in regression to *flat* estimation functions. Circles are training points, and dotted lines enclose the ϵ-insensitive region. Solid lines are candidate regression functions, the bold one being the flattest which still fits training data with an approximation up to ϵ.

$$
\begin{aligned}
L = \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{m}\sum_{i=1}^{m}(\xi_i + \xi_i^*) - \sum_{i=1}^{m}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\
- \sum_{i=1}^{m}\alpha_i(\epsilon + \xi_i + y_i - < \mathbf{w}, \mathbf{x}_i > - b) \\
- \sum_{i=1}^{m}\alpha_i^*(\epsilon + \xi_i^* - y_i + < \mathbf{w}, \mathbf{x}_i > + b),
\end{aligned}
\tag{2.54}
$$

with $\alpha_i, \alpha_i*, \eta_i, \eta_i^* \geq 0$, $\forall i \in [1, m]$. By vanishing the derivatives of L with respect to the primal variables we obtain:

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) = 0, \tag{2.55}$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{m} (\alpha_i - \alpha_i^*)\mathbf{x}_i = 0, \tag{2.56}$$

$$\frac{\partial L}{\partial \xi_i} = C/m - \alpha_i - \eta_i = 0, \tag{2.57}$$

$$\frac{\partial L}{\partial \xi_i^*} = C/m - \alpha_i^* - \eta_i^* = 0. \tag{2.58}$$

Finally, substituting (2.55),(2.56),(2.57) and (2.58) into (2.54) we derive the dual problem

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^m} \quad -\frac{1}{2} \sum_{i,j=1}^{m} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) < \mathbf{x}_i, \mathbf{x}_j >$$

$$-\epsilon \sum_{i=1}^{m} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{m} y_i(\alpha_i^* - \alpha_i), \tag{2.59}$$

$$\text{subject to} \quad \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) = 0, \tag{2.60}$$

$$\alpha_i, \alpha_i^* \in [0, C/m], \quad \forall i \in [1, m]. \tag{2.61}$$

Note that in virtue of (2.56), we can rewrite the estimated function (2.49) as a linear combination of the training patterns:

$$f(\mathbf{x}) = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) < \mathbf{x}_i, \mathbf{x} > +b. \tag{2.62}$$

Again, both the dual optimization problem (2.59) and the learned function (2.62) only contain input patterns $\mathbf{x}_i$ in the form of dot products $< \mathbf{x}_i, \mathbf{x}_j >$, thus allowing to use a kernel $k(x_i, x_j)$ corresponding to a dot product in a higher dimensional feature space.

It follows from the KKT theorem [Fle87] that at the saddle point of the Lagrangian we have

$$\alpha_i(\epsilon + \xi_i + y_i - < \mathbf{w}, \mathbf{x}_i > -b) = 0, \tag{2.63}$$

$$\alpha_i^*(\epsilon + \xi_i^* - y_i + < \mathbf{w}, \mathbf{x}_i > +b) = 0, \tag{2.64}$$

$$(C/m - \alpha_i)\xi_i = 0, \tag{2.65}$$

$$(C/m - \alpha_i^*)\xi_i^* = 0. \tag{2.66}$$

These last conditions enlighten some interesting analogies to the SVM case (see also fig. 2.7). Let us define the $\epsilon - tube$ of f as $\{(\mathbf{x}, y) \in \mathcal{X} \times \mathbb{R} : |f(\mathbf{x}) - y| < \epsilon\}$.

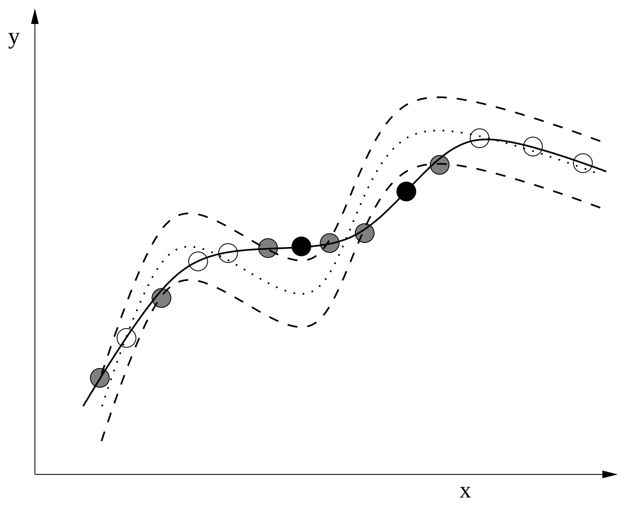


Figure 2.7. Regression problem solved by support vector machines. The dotted line is the true function, and the dashed lines enclose the ϵ-insensitive region. The solid line is the learned regression function. Grey points are unbound SVs, and black points are bound SVs. All other points do not contribute to the estimation function. Flatness in feature space implies smoothness of the regression function in input space.

- All patterns within the ϵ-tube, for which $|f(\mathbf{x}_i) - y_i| < \epsilon$, have $\alpha_i, \alpha_i^* = 0$ and thus don't contribute to the estimated function f.
- Patterns for which either $0 < \alpha_i < C/m$ or $0 < \alpha_i^* < C/m$ are on the border of the ϵ-tube, that is $|f(\mathbf{x}_i) - y_i| = \epsilon$. They are the unbound support vectors.
- The remaining training patterns are margin errors (either $\xi_i > 0$ or $\xi_i^* > 0$), and reside out of the ϵ-insensitive region. They are bound support vectors, with corresponding $\alpha_i = C/m$ or $\alpha_i^* = C/m$.

2.3.2 Support Vector Clustering

A support vector algorithm has been used in [TD99, SPST$^+$01] in order to characterize a set of data in terms of support vectors, thus allowing to compute a set of contours which enclose the data points. These contours were interpreted as cluster boundaries in [BHHSV01], resulting in the so called *support vector clustering* (SVC) algorithm.

In SVC, data are mapped to a high dimensional feature space using a Gaussian kernel (see section 2.5.1). In such space, the smallest sphere containing the data is searched for. When mapped back to the input space, the sphere forms a set of contours which enclose the data points, thus determining a clustering of the input space. By decreasing the width of the Gaussian kernel it is possible to increase the number of clusters in the solution. Outliers can be dealt with by relaxing the enclosing constraint and allowing some points to stay out of the sphere in feature space. In the following, we will briefly describe the algorithm, showing its connections to the general SVM case.

Given a set of m points $x_i \in \mathcal{X}, i = 1, \ldots, m$ and a nonlinear transformation Φ from $\mathcal{X}$ to some higher dimensional feature space $\mathcal{H}$, we can define the problem of finding the smallest enclosing sphere of radius R in the feature space as follows:

$$\min_{R \in \mathbb{R}, \mathbf{o} \in \mathcal{H}, \Xi \in \mathbb{R}^m} \quad R^2 + \frac{C}{m} \sum_{i=1}^{m} \xi_i \tag{2.67}$$

$$\text{subject to} \quad ||\Phi(x_i) - \mathbf{o}||^2 \leq R^2 + \xi_i, \quad \forall i \in [1, m] \tag{2.68}$$

$$\xi_i \geq 0, \quad \forall i \in [1, m], \tag{2.69}$$

where $\mathbf{o}$ is the center of the sphere, ξ_i are slack variables allowing for soft constraints and C is a cost parameter balancing the radius of the sphere versus the number of outliers. We consider the Lagrangian

$$L = R^2 + \frac{C}{m} \sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i (R^2 + \xi_i - ||\Phi(x_i) - \mathbf{o}||^2) - \sum_{i=1}^{m} \eta_i \xi_i, \tag{2.70}$$

with $\alpha_i \geq 0$ and $\eta_i \geq 0$ for all $i \in [1, m]$, and by vanishing the derivatives with respect to the primal variables R, $\mathbf{o}$ and ξ_i we obtain

$$\frac{\partial L}{\partial R} = \sum_{i=1}^{m} \alpha_i - 1 = 0, \tag{2.71}$$

$$\frac{\partial L}{\partial \mathbf{o}} = \mathbf{o} - \sum_{i=1}^{m} \alpha_i \Phi(x_i) = 0, \tag{2.72}$$

$$\frac{\partial L}{\partial \xi_i} = \alpha_i - C/m + \eta_i = 0, \quad \forall i \in [1, m]. \tag{2.73}$$

By substituting (2.71),(2.72) and (2.73) into (2.70) we derive the Wolf dual problem

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^m} \quad \sum_{i=1}^{m} \alpha_i \Phi(x_i)^2 - \sum_{i,j=1}^{m} \alpha_i \alpha_j \Phi(x_i) \Phi(x_j), \tag{2.74}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C/m, \quad \forall i \in [1, m]. \tag{2.75}$$

The distance of a given point x from the center of the sphere

$$R^2(x) = ||\Phi(x) - \mathbf{o}||^2 \tag{2.76}$$

can be written using (2.72) as

$$R^2(x) = \Phi(x)\Phi(x) - 2\sum_{i=1}^{m}\alpha_i\Phi(x)\Phi(x_i) + \sum_{i,j=1}^{m}\alpha_i\alpha_j\Phi(x_i)\Phi(x_j). \qquad (2.77)$$

As for the other SV algorithms, both (2.74) and (2.77) contain only dot products in the feature space, which can be substituted by a kernel function k. It has been noted [TD99] that polynomial kernels do not yield tight contours for clusters. Gaussian kernels (see section 2.5)

$$k(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2}) \qquad (2.78)$$

with width parameter σ were successfully employed [BHHSV01] and are assumed in the rest of the section.

The KKT conditions [Fle87] imply that at the saddle point of the Lagrangian

$$\eta_i\xi_i = 0, \qquad (2.79)$$
$$\alpha_i(R^2 + \xi_i - ||\Phi(x_i) - \mathbf{o}||^2) = 0, \qquad (2.80)$$

showing the presence of (see fig. 2.8):

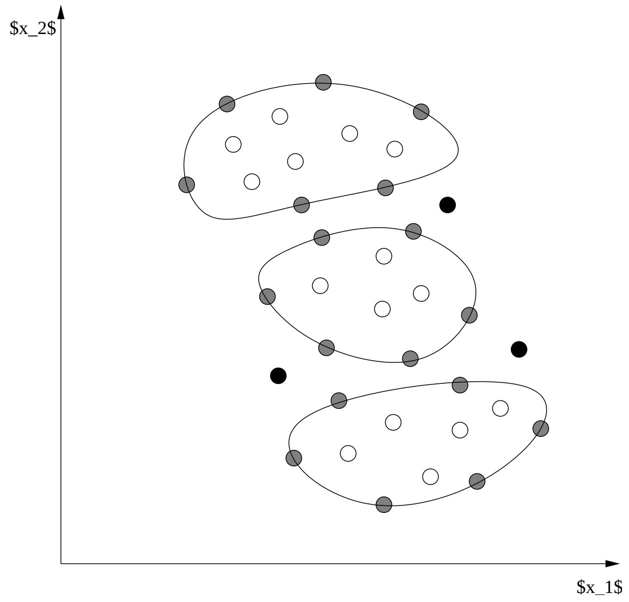


Figure 2.8. Support vector clustering. Lines represent cluster boundaries, formed by unbound SVs (grey points). Bound support vectors (black points) are outliers and do not belong to any cluster. All other points stay within the boundaries of their respective cluster.

- Unbound support vectors ($0 < \alpha_i < C/m$), whose images lie on the surface of the enclosing sphere, corresponding to cluster boundaries in the input space.

- Bound support vectors ($\alpha_i = C/m$), whose images lie outside of the enclosing sphere, which correspond to outliers, staying out of the clusters boundaries.
- All other points ($\alpha = 0$) which stay within clusters boundaries, with images inside the enclosing sphere.

The radius R of the enclosing sphere can be computed by (2.77) provided x is a bound support vector.

In order to assign points to clusters, note that given a pair of points belonging to different clusters, any path connecting them must exit from the sphere in feature space, that is contain a segment of points x such that $R(x) > R$. We can therefore define an adjacency matrix A_{ij},such that

$$A_{ij} = \begin{cases} 1 & \text{if, for all } x \text{ in the segment connecting } x_i \text{ and } x_j, R(x) \leq R \\ 0 & \text{otherwise.} \end{cases} \tag{2.81}$$

Clusters are defined as the connected components of the graph induced by A. Outliers are not classified by this procedure, and can be possibly assigned to the cluster they are closest to. In [BHHSV01], an iterative version of the algorithm is also proposed: start by a Gaussian kernel with variance big enough to assure a single cluster solution. Then systematically decrease it together to the cost parameter C along a direction that guarantees a minimal number of unbound support vectors. This allows to find a range of possible solutions of increasing complexity, stopping when the fraction of support vectors exceeds a certain threshold.

2.4 Kernel Theory

The kernel trick introduced in section 2.2.3 allows to implicitly compute a dot product between instances in a possibly infinite feature space. In this section we will treat in more detail the theory underlying kernel functions, showing how to verify if a given function is actually a valid kernel, and how to generate given a valid kernel a feature space such that the kernel computes a dot product in that space. We will then highlight the connections between kernel machines and regularization theory, showing how kernel machines can be seen as instances of regularized empirical risk minimization.

2.4.1 Positive Definite and Mercer Kernels

We start by introducing the concept of positive definite kernels and showing how to build a space where they act as a dot product. We will treat the case of real valued matrices and functions, but results can be extended to complex matrices as well. Moreover, we will give some necessary and sufficient condition for a function k to be a positive definite kernel. Details and proofs of the reported results can be found in [Aro50, BCR84, Sai88, SS02].

Definition 2.4.1 (Gram Matrix) *Given a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and patterns $x_1, \dots, x_m$, the $m \times m$ matrix K such that*

$$K_{ij} = k(x_i, x_j) \tag{2.82}$$

is called the Gram matrix *of k with respect to $x_1, \dots, x_m$.*

Definition 2.4.2 (Positive Definite Matrix) *A symmetric $m \times m$ matrix K is* positive definite *if*

$$\sum_{i,j=1}^{m} c_i c_j K_{ij} \geq 0, \quad \forall \mathbf{c} \in \mathbb{R}^m \tag{2.83}$$

Definition 2.4.3 (Positive Definite Kernel) *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $\forall\, m \in \mathbb{N}$ and $\forall\, x_1, \dots, x_m \in \mathcal{X}$ gives rise to a positive definite gram matrix is called a* positive definite kernel.

We will now build a so called *reproducing kernel map* for the positive definite kernel k, that is a map from $\mathcal{X}$ into a space where k acts as a dot product. Let

$$\Phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}} \mid \Phi(x) = k(\,\cdot\,, x) \tag{2.84}$$

be a map from $\mathcal{X}$ to the space $\mathbb{R}^{\mathcal{X}}$ of functions mapping $\mathcal{X}$ into $\mathbb{R}$. We need to turn $\mathbb{R}^{\mathcal{X}}$ into a vector space endowed with a dot product $< \cdot, \cdot >$ satisfying

$$k(x, x') \; = < \Phi(x), \Phi(x') > . \tag{2.85}$$

In order to make it a vector space we just need to take the span of kernel k, that is all functions

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\,\cdot\,, x_i) \tag{2.86}$$

for all $m \in \mathbb{N}$, $\alpha_i \in \mathbb{R}$, $x_i \in \mathcal{X}$. A dot product in such space between f and another function

$$g(\cdot) = \sum_{j=1}^{m'} \beta_j k(\,\cdot\,, x'_j) \tag{2.87}$$

can be defined as

$$< f, g > = \sum_{i=1}^{m} \sum_{j=1}^{m'} \alpha_i \beta_j k(x_i, x'_j). \tag{2.88}$$

For each given function f, it holds that

$$< k(\,\cdot\,, x), f(\cdot) > = \; f(x). \tag{2.89}$$

In particular, for $f = k(\,\cdot\,, x')$ we have

$$< k(\,\cdot\,, x), k(\,\cdot\,, x') > = \; k(x, x'). \tag{2.90}$$

By satisfying equation (2.85) we showed that each positive definite kernel can be seen as a dot product in another space. In order to show that the converse is also true, it suffices to prove that given a map Φ from $\mathcal{X}$ to a product space, the corresponding function $k(x, x') \; = < \Phi(x), \Phi(x') >$ is a positive definite kernel. This can be proved by noting that for all $m \in \mathbb{N}$, $\mathbf{c} \in \mathbb{R}^m$ and $x_1, \dots, x_m \in \mathcal{X}$ we have

$$\sum_{i,j=1}^{m} c_i c_j k(x_i, x_j) = \left\langle \sum_{i=1}^{m} c_i \Phi(x_i), \sum_{j=1}^{m} c_j \Phi(x_j) \right\rangle = \left\| \sum_{i=1}^{m} c_i \Phi(x_i) \right\|^2 \geq 0. \tag{2.91}$$

The existence of a map to a dot product space satisfying (2.85) is therefore an alternative definition for a positive definite kernel. The completion of the dot product space $\mathbb{R}^{\mathcal{X}}$, obtained by adding all limit functions of Cauchy sequences in that space, is called a *reproducing kernel Hilbert space* (*RKHS*).

Definition 2.4.4 (Reproducing Kernel Hilbert Space) *Let $\mathcal{X}$ be a non empty set and $\mathcal{H}$ a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$ with dot product $< \,\cdot\,, \cdot\, >$. $\mathcal{H}$ is called a* reproducing kernel Hilbert space *if there exists a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that:*

1. *k has the* reproducing property

$$< k(\,\cdot\,, x), f > = \; f(x), \quad \forall f \in \mathcal{H}, \; \forall x \in \mathcal{X}. \tag{2.92}$$

2. *k spans $\mathcal{H}$ ($\bar{\cdot}$ denotes completion)*

$$\mathcal{H} = \overline{span\{k(\,\cdot\,, x) \,|\, x \in \mathcal{X}\}}. \tag{2.93}$$

An alternative way to build a space with a dot product corresponding to $k(x, x')$ is given by *Mercer's theorem* [Mer09, Kon86].

Theorem 2.4.1 (Mercer's Theorem) *Let $\mathcal{X}$ be a compact set with Borel measure μ, $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a symmetric real valued function determining an integral operator $T_k : L_2(\mathcal{X}) \to L_2(\mathcal{X})$, such that*

$$(T_k f)(x) \;:= \int_{\mathcal{X}} k(x, x') f(x') d\mu(x'), \tag{2.94}$$

which is positive definite*; that is, for all $f \in L_2(\mathcal{X})$, we have*

$$\int_{\mathcal{X}^2} k(x,x')f(x)f(x')d\mu(x)\mu(x') \geq 0. \tag{2.95}$$

Let λ_k be the k^{th} eigenvalue of T_k, with corresponding eigenfunction ψ_k. For all $x, x' \in \mathcal{X}$

$$k(x,x') = \sum_{k=1}^{\infty} \lambda_k \psi_k(x) \psi_k(x'), \tag{2.96}$$

where the convergence is absolute and uniform.

The *Mercer kernel* k in eq.(2.96) corresponds to a dot product in $L_2^{\infty}(\mathcal{X})$ thanks to the map $\Phi : \mathcal{X} \rightarrow L_2^{\infty}(\mathcal{X})$ such that

$$\Phi(x) = (\sqrt(\lambda_k)\psi_k(x))_{k=1}^{\infty}, \tag{2.97}$$

proving that k is also a positive definite kernel. Mercer's condition (eq. (2.95)) is therefore an alternative way to verify if a given function k is actually a positive definite kernel.

It can be shown [Bur98b, SSM98] that such condition is satisfied for any function k which can be expressed as

$$k(x,x') = \sum_{i=1}^{\infty} c_i (xx')^i, \tag{2.98}$$

where c_i are positive real coefficients and the series converges uniformly.

It's interesting to note how to find a reproducing kernel Hilbert space corresponding to Mercer kernel k, with functions

$$f(x) = \sum_{i=1}^{\infty} \alpha_i k(x, x_i) = \sum_{i,j=1}^{\infty} \alpha_i \lambda_j \psi_j(x) \psi_j(x_i). \tag{2.99}$$

Given a dot product $< \cdot, \cdot >$, by linearity we have

$$< f, k(\cdot, x') > = \sum_{i,j,k=1}^{\infty} \alpha_i \lambda_j \psi_j(x_i) < \psi_j, \psi_k > \lambda_k \psi_k(x'). \tag{2.100}$$

Being the eigenfunctions ψ_i orthogonal in $L_2(\mathcal{X})$, we can choose $< \cdot, \cdot >$ such that

$$< \psi_j, \psi_k > = \frac{\delta_{jk}}{\lambda_j} \tag{2.101}$$

with δ_{jk} the Kronecker symbol [2], showing that (2.99) satisfies the reproducing kernel property (2.92).

[2]
$$\delta_{jk} = \begin{cases} 1 & \text{if} \quad j = k \\ 0 & \text{otherwise} \end{cases}$$

2.4.2 Regularization Theory

Learning from examples can be seen as the problem of approximating a target function with a finite number of training data. The problem of approximating a function from sparse data is ill-posed, and regularization theory is a classical way to turn the problem into a well-posed one. Recall the problem of empirical risk minimization (see section 2.1.1), where we have a training set $D_m = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^m$, a set of functions $\mathcal{F}$ from $\mathcal{X}$ to $\mathcal{Y}$, and a loss function $\ell : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to [0, \infty]$. Assume for simplicity that $\mathcal{Y} \equiv \mathrm{I\!R}$. Assume also that $R_{emp}[f]$ is continuous in f [3]. Regularization theory deals with the problem of restricting $\mathcal{F}$ in order to make it a compact set, thus obtaining a well-posed minimization problem [Tik63, Vap98]. Instead of directly specifying a compact set for $\mathcal{F}$, which would cast the problem into a complex constrained optimization task, we add a regularization term $\Omega[f]$ to the objective functional, such that $\Omega[f] \geq 0$ for all f, and the sets

$$\mathcal{F}_c = \{f : \Omega[f] \leq c\}, \quad c \geq 0,$$

are all compact. This results in a regularized risk functional

$$R_{reg}[f] = R_{emp}[f] + \lambda\Omega[f]. \tag{2.102}$$

giving a well-posed minimization problem [Tik63, Vap98]. Here the regularization parameter $\lambda > 0$ trades the effect of training errors with the complexity of the function, thus providing a mean to control overfitting. By choosing Ω to be convex, and provided $R_{emp}[f]$ is also convex, the problem has a unique global minimum.

When $\mathcal{F}$ is a reproducing kernel Hilbert space $\mathcal{H}$ (see def. 2.4.4) associated to a kernel k, the *representer theorem* [KW71] gives an explicit form of the minimizers of $R_{reg}[f]$. We present a more general version [SHSW01, SS02] of the theorem with respect to the original one.

Theorem 2.4.2 (Representer Theorem) *Let $D_m = \{(x_i, y_i) \in \mathcal{X} \times \mathrm{I\!R}\}_{i=1}^m$ be a training set, $\ell : (\mathcal{X} \times \mathrm{I\!R} \times \mathrm{I\!R})^m \to \mathrm{I\!R} \cup \{\infty\}$ a cumulative loss function, $\mathcal{H}$ a RKHS with norm $||\cdot||_{\mathcal{H}}$, and $\Omega : [0, \infty] \to \mathrm{I\!R}$ a strictly monotonic increasing function. Then each minimizer $f \in \mathcal{H}$ of the regularized risk*

$$\ell((x_1, y_1, f(x_1)), \ldots, (x_m, y_m, f(x_m))) + \Omega(||f||_{\mathcal{H}}) \tag{2.103}$$

admits a representation of the form

$$f(x) = \sum_{i=1}^{m} \alpha_i k(x_i, x). \tag{2.104}$$

[3]This doesn't hold for the misclassification loss (eq. (2.1)), which should be replaced by a continuous approximation such as the soft margin loss (eq. (2.2)).

The theorem states that regardless of the dimension of the RKHS $\mathcal{H}$, the solution lies on the span of the m kernels centered on the training points. If both Ω and ℓ are also convex, such solution is unique.

A semiparametric extension of the representer theorem is given by the following theorem [SS02].

Theorem 2.4.3 (Semiparametric Representer Theorem) *Lets add to the assumptions of the representer theorem a set of M functions $\{\psi_p\}_{p=1}^M : \mathcal{X} \rightarrow \mathrm{I\!R}$ such that the m x M matrix $(\psi_p(x_i))_{ip}$ has rank M. Then any $\tilde{f} := f + g$, with $f \in \mathcal{H}$ and $g \in$ span $\{\psi_p\}$, minimizing the functional*

$$\ell((x_1, y_1, \tilde{f}(x_1)), \ldots, (x_m, y_m, \tilde{f}(x_m))) + \Omega(||f||_{\mathcal{H}}) \tag{2.105}$$

admits a representation of the form

$$\tilde{f}(x) = \sum_{i=1}^{m} \alpha_i k(x_i, x) + \sum_{p=1}^{M} \beta_p \psi_p(x), \tag{2.106}$$

with $\beta_p \in \mathrm{I\!R}, \forall p \in [1, M]$.

We are now able to enlighten the connections between regularization theory and kernel machines. Given the regularized risk (2.105), different choices of the loss function ℓ, the regularization function Ω and the real valued functions ψ_p give rise to different types of kernel machines. Support vector machines for binary classification (see section 2.2), for example, employ the soft margin loss function (eq. 2.2), which as a cumulative loss becomes

$$\ell((x_1, y_1, f(x_1)), \ldots, (x_m, y_m, f(x_m))) = \frac{1}{m} \sum_{i=1}^{m} |1 - y_i f(x_i)|_+, \tag{2.107}$$

and a regularizer of the form $\Omega(||f||_{\mathcal{X}}) = \frac{\lambda}{2}||f||^2$, where the regularization parameter λ corresponds to the inverse of the cost C. A single additional constant function $\psi_1(x) = 1$ is used as an offset in biased support vector machines, those with hyperplanes not passing for the origin (that is $b \neq 0$, see eq. (2.12)). For $\lambda \rightarrow 0$ we have hard margin SVM where all training patterns have to be correctly classified. Note that the decision function for SVM is actually *sign(f)*.

Support vector regression is obtained using the $\epsilon -$ *insensitive* loss (eq. 2.4)

$$\ell((x_1, y_1, f(x_1)), \ldots, (x_m, y_m, f(x_m))) = \frac{1}{m} \sum_{i=1}^{m} |1 - y_i f(x_i)|_\epsilon, \tag{2.108}$$

with same regularization Ω and constant function ψ_1 as for the SVM for binary classification.

2.5 Kernel Design

Kernel design deals with the problem of choosing an appropriate kernel for the task at hand, that is a similarity measure of the data capable of best capturing the available information. We start by introducing a few basic kernels commonly used in practice, and show how to realize complex kernels by combination of simpler kernels, allowing to treat different parts or characteristics of the input data in different ways. We will introduce the notion of kernels on discrete structures, providing examples of kernels for strings, trees and graphs, and that of generative kernels, aiming at combining the expressive power of generative models and the separation capabilities of discriminative models. Finally, we will discuss the problem of choosing kernel hyperparameters, a task which is part of the general problem of model selection. Unless diversely specified, in the rest of the chapter we will refer to positive definite kernels simply with the term kernels.

2.5.1 Basic Kernels

A simple example of effective kernels commonly used in practice is that of the *polynomial* kernels, both homogeneous

$$k(x,x') = < x, x' >^d \tag{2.109}$$

and inhomogeneous

$$k(x,x') = (< x, x' > + c)^d, \tag{2.110}$$

with $d \in \mathbb{N}$ and $c \in \mathbb{R}_0^+$. They correspond respectively to the feature space of all possible monomials of degree d (eq. (2.109)), and that of all possible monomials of degree up to d (eq. (2.110)). Another interesting class of kernels is that of the Radial basis function (RBF) ones, satisfying

$$k(x,x') = f(d(x,x')), \tag{2.111}$$

where d is a metric in $\mathcal{X}$ and f a function on $\mathbb{R}_0^+$. Gaussian kernels [BGV92, GBV93]

$$k(x,x') = \exp\left(-\frac{||x-x'||^2}{2\sigma^2}\right) \tag{2.112}$$

with $\sigma > 0$ are a popular example of these translation invariant kernels. Recalling the decision function for support vector machines (eq. (2.36)), we can see that the smallest the variance σ^2, the least support vectors will be involved in decision on a test pattern, namely those nearest to it according to the distance induced by the norm $||\cdot||$. Roughly speaking, a smaller variance usually corresponds to a greater number of support vectors, and thus to a more complex model. Tuning this *hyperparameter* according to the complexity of the problem at hand is another mean to perform risk minimization and prevent overfitting (see section 2.5.5).

Finally, sigmoid kernels

$$k(x, x') = \tanh(\kappa < x, x' > +\vartheta), \tag{2.113}$$

where $\kappa > 0$ and $\vartheta > 0$, have been successfully used in practice, even if they are not actually positive definite (see [Sch97] for details).

2.5.2 Kernel Combination

The class of kernels has a few interesting closure properties useful for combinations. It is closed under addition, multiplication by a positive constant and pointwise limits [BCR84], that is they form a convex cone. Moreover it's close under product [PS72]: if $k_1(x, x')$ and $k_2(x, x')$ are kernels, also $k(x, x') = k_1(x, x')k_2(x, x')$ is a kernel. Such properties are still valid in the case that the two kernels are defined on different domains [Hau99]:

Proposition 2.5.1 (Direct Sum) *If k_1 and k_2 are kernels defined respectively on $\mathcal{X}_1 \times \mathcal{X}_1$ and $\mathcal{X}_2 \times \mathcal{X}_2$, then their direct sum,*

$$(k_1 \oplus k_2)((x_1, x_2), (x'_1, x'_2)) = k_1(x_1, x'_1) + k_2(x_2, x'_2) \tag{2.114}$$

is a kernel on $(\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2)$, with $x_1, x'_1 \in \mathcal{X}_1$ and $x_2, x'_2 \in \mathcal{X}_2$.

Proposition 2.5.2 (Tensor Product) *If k_1 and k_2 are kernels defined respectively on $\mathcal{X}_1 \times \mathcal{X}_1$ and $\mathcal{X}_2 \times \mathcal{X}_2$, then their tensor product,*

$$(k_1 \otimes k_2)((x_1, x_2), (x'_1, x'_2)) = k_1(x_1, x'_1)k_2(x_2, x'_2) \tag{2.115}$$

is a kernel on $(\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2)$, with $x_1, x'_1 \in \mathcal{X}_1$ and $x_2, x'_2 \in \mathcal{X}_2$.

These combinations allow to treat in a diverse way parts of an individual which have different meanings. A general extension of these concepts is at the basis of the so called *convolution* kernels [Hau99, Wat00] for discrete structures. Suppose $x \in \mathcal{X}$ is a composite structure made of "parts" $x_1, \ldots, x_D$ such that $x_d \in \mathcal{X}_d$ for all $i \in [1, D]$. This can be formally represented by a relation R on $\mathcal{X}_1 \times \cdots \times \mathcal{X}_D \times \mathcal{X}$ such that $R(x_1, \ldots, x_D, x)$ is true iff $x_1, \ldots, x_D$ are the parts of x. For example if $X_1 = \cdots = X_D = X$ are sets containing all finite strings over a finite alphabet $\mathcal{A}$, we can define a relation $R(x_1, \ldots, x_D, x)$ which is true iff $x = x_1 \circ \cdots \circ x_D$, with $\circ$ denoting concatenation of strings. Note that in this example x can be decomposed in multiple ways. If their number is finite, the relation is said to be finite. Given a set of kernels $k_d : \mathcal{X}_d \times \mathcal{X}_d \to \mathrm{I\!R}$, one for each of the parts of x, the *R-convolution* kernel is defined as

$$(k_1 \star \cdots \star k_D)(x, x') = \sum_R \prod_{d=1}^{D} k_d(x_d, x'_d), \tag{2.116}$$

where the sum runs over all the possible decompositions of x and x'. For finite relations R, this can be shown to be a valid kernel [Hau99].

It's easy to verify [Hau99] that Gaussian kernels are a particular case of R-convolution kernels, as well as simple exponential kernels

$$(k_1 \star \cdots \star k_D)(x, x') = \exp(\sum_{d=1}^{D} \frac{f_d(x_d) f_d(x'_d)}{\sigma_d^2}) = \prod_{d=1}^{D} k_d(x_d, x'_d), \tag{2.117}$$

where

$$k_d(x, x') = \exp(\frac{f_d(x) f_d(x')}{\sigma_d^2}) \tag{2.118}$$

and $f_d : \mathcal{X}_d \to \mathbb{R}$, $\sigma_d > 0$ for each $d \in [1, D]$. In both cases there is only one way to decompose each x.

A more complex type of R-convolution kernel is the so called *analysis of variance* (ANOVA) kernel [Wah90, SGV98, Vap98, SGV$^+$99]. Let $\mathcal{X} = \mathcal{S}^n$ for some set S, and $k_i : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, $i \in [1, n]$ a set of kernels, which will typically be the same function. For $D \in [1, n]$, the ANOVA kernel of order D is defined by

$$k(x, y) = \sum_{1 \leq i_1 < \cdots < i_D \leq n} \sum_{d=1}^{D} k_{i_d}(x_{i_d}, x'_{i_d}). \tag{2.119}$$

Note that for $D = n$, the sum consists only of the term for which $(i_1 = 1, \ldots, i_D = n)$, and k becomes the tensor product $k_1 \otimes \cdots \otimes k_n$. Conversely, for $D = 1$, each product collapses to a single factor, while i_1 ranges from 1 to n, giving the direct sum $k_1 \oplus \cdots \oplus k_n$. By varying D we can run between these two extremes. In order to reduce the computational cost of kernel evaluations, recursive procedures are usually employed [Vap98].

Finally, note that whenever the kernel procedure k explicitly computes the mapping $\Phi : \mathcal{X} \to \mathcal{H}$ into the feature space, as for absolute convergent graph kernels (see section 2.5.3, eq. 2.134) or Fisher kernels (see section 2.5.4, eq. 2.147), it's always possible to apply a new kernel

$$k' : \mathcal{H} \times \mathcal{H} \to \mathbb{R}. \tag{2.120}$$

This amounts to searching for a nonlinear separation in feature space or, equivalently, to mapping data from the feature space $\mathcal{H}$ to an even higher dimensional feature space $\mathcal{H}'$ by the feature map $\Phi' : \mathcal{H} \to \mathcal{H}'$ corresponding to k'. The resulting kernel

$$k^*(x, y) = k'(\Phi(x), \Phi(y)) = < \Phi'(\Phi(x)), \Phi'(\Phi(y)) > = < \Phi^*(x), \Phi^*(y) > \tag{2.121}$$

corresponds to a dot product in the feature space with map

$$\Phi^* : \mathcal{X} \to \mathcal{H}' \mid \Phi^* = \Phi' \circ \Phi \tag{2.122}$$

given by the composition of Φ' and Φ.

2.5.3 Kernels on Discrete Structures

R-convolution kernels are a very general class of kernels, which can be used to model similarity between objects with discrete structures, such as strings, trees and graphs. In order for a kernel to be of practical utility, however, it must be computed in reasonable time. Therefore, a kernel can also be thought of as a procedure efficiently implementing a given dot product in feature space. In the following, we will report a series of kernels developed for efficiently treating objects with discrete structure. A different problem is that of treating objects that are related to each other by a graph structure. For a description of kernels on structured domains see [KL02, GLF03, GFKS02, SW03].

Strings

The so called *string subsequence kernel* (SSK) [LSTCW00] is an interesting example of kernel on strings. Consider a finite alphabet $\mathcal{A}$. A string s is a finite sequence of (possibly zero) characters from $\mathcal{A}$. We define by $|s|$ the length of string s, $\mathcal{A}^n$ the set of all strings of length n, and

$$\mathcal{A}^* = \bigcup_{n=0}^{\infty} \mathcal{A}^n \tag{2.123}$$

the set of all strings. Concatenation between strings s and t is simply represented as st. A subsequence u of s is defined as $u := s(\mathbf{i}) := s(i_1) \dots s(i_{|u|})$, with $1 \leq i_1 < \cdots < i_{|u|} \leq |n|$ and $s(i)$ the i^{th} element of s. The length $l(\mathbf{i})$ of the subsequence u in s is $i_{|u|} - i_1 + 1$. Note that if $\mathbf{i}$ is not contiguous, $l(\mathbf{i}) > |u|$. We can now define the feature space $\mathcal{H}_n := \mathbb{R}^{(\mathcal{A}^n)}$ as the space whose coordinates are all strings of length n. The feature map Φ for a given string s and coordinate $u \in \mathcal{A}^n$ is defined as

$$[\Phi_n(s)]_u = \sum_{\mathbf{i}:s(\mathbf{i})=u} \lambda^{l(\mathbf{i})}, \tag{2.124}$$

where $0 < \lambda \leq 1$ is a weight decay penalizing gaps. Such feature measures the number of occurrences of u in s, weighted according to their lengths. The inner product between strings s and t according to such mapping is

$$k_n(s,t) = \sum_{u \in \mathcal{A}^n} [\Phi_n(s)]_u [\Phi_n(t)]_u = \sum_{u \in \mathcal{A}^n} \sum_{\mathbf{i}:s(\mathbf{i})=u} \sum_{\mathbf{j}:t(\mathbf{j})=u} \lambda^{l(\mathbf{i})+l(\mathbf{j})}. \tag{2.125}$$

Note that k_n is a valid kernel as it explicitly computes the inner product in feature space. In order to make this product computationally efficient, we first introduce the auxiliary function

$$k'_i(s,t) = \sum_{u \in \mathcal{A}^i} \sum_{\mathbf{i}:s(\mathbf{i})=u} \sum_{\mathbf{j}:t(\mathbf{j})=u} \lambda^{|s|+|t|-i_1-j_1+2} \tag{2.126}$$

for $i = 1, \dots, n-1$, counting the length from the beginning of the substring match to the end of s and t instead of $l(\mathbf{i})$ and $l(\mathbf{j})$. The SSK can now be computed by the following recursive procedure, where $a \in \mathcal{A}$:

$$\begin{aligned}
k_0(s,t) &= 1 \quad \forall\, s,t \in \mathcal{A}^* \\
k'_i(s,t) &= 0 \quad \text{if } \min(|s|,|t|) < i \\
k_i(s,t) &= 0 \quad \text{if } \min(|s|,|t|) < i \\
k'_i(sa,t) &= \lambda k'_i(s,t) + \sum_{j:t(j)=a} k'_{i-1}(s,t[1,\dots,j-1])\lambda^{|t|-j+2}, \forall\, i \in [1,n-1] \\
k_n(sa,t) &= k_n(s,t) + \sum_{j:t(j)=a} k'_{n-1}(s,t[1,\dots,j-1])\lambda^2. \qquad (2.127)
\end{aligned}$$

To prove the correctness of the procedure note that $k_n(sa,t)$ is computed by adding to $k_n(s,t)$ all the terms resulting by the occurrences of substrings terminated by a, matching t anywhere and sa on its right terminal part. In fact, in the second term of the recursion step for k_n, k'_{n-1} will count any matching substring found in s as if it finished at $|s|$, and the missing λ for the last element a is added for both s and t.

This kernel can be readily expanded to consider substrings of different lengths, i.e. by using a linear combination like

$$k(s,t) = \sum_n c_n k_n(s,t), \qquad (2.128)$$

with $c_n \geq 0$. In such case, we simply compute k'_i for all i up to one less than the largest n required, and then apply the last recursion in (2.127) for each n such that $c_n > 0$, using the stored values of k'_i.

Note that in order to remove the bias introduced by string length, it's convenient to employ a normalized version of the kernel:

$$\hat{k}(s,t) = \frac{k(s,t)}{\sqrt{k(s,s)k(t,t)}} \qquad (2.129)$$

Alternative types of string kernels will be discussed in sections 2.5.4 and 5.2.2.

Trees

A similar approach for trees was proposed in [CD02a, CD02b] in the field of natural language processing. Here we present the procedure in a slightly mode general case, without entering in details of computational linguistics. Let $\mathcal{A}$ be a set of labels, and $\mathcal{T}$ a set of labelled trees, where each node is associated with a single label in $\mathcal{A}$, and there is a total order relation in the children of a node. Given $t \in \mathcal{T}$, we consider a subtree t' of t to be valid if each node in it has either all or none of the children of the corresponding node in t (see fig. 2.9) . We will refer to these valid subtrees as tree fragments or simply subtrees.

Given two nodes n_1 and n_2, we say that they *match* if they have the same label, same number of children if any, and each child of n_1 has the same label of the corresponding child of n_2 in the total ordering (see fig. 2.10).

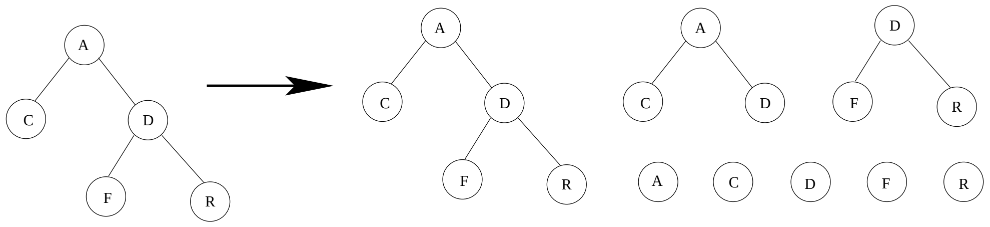


Figure 2.9. Example of extraction of all valid subtrees of a given tree.

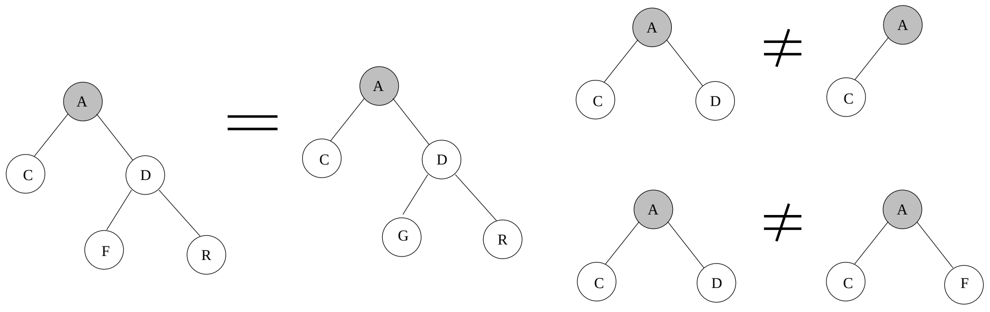


Figure 2.10. Examples of matching and non matching nodes between two trees, where the nodes to be compared are the root nodes (shaded) of the respective trees.

The feature space is built by fixing a coordinate for each possible tree fragment in $\mathcal{T}$. Given by M the number of these fragments, a tree t is mapped to a vector $[\Phi_1(t), \ldots, \Phi_M(t)]$, where each element i counts the number of times the corresponding tree fragment is found in t. The kernel k between two trees t_1 and t_2 is simply given by the inner product in such space. We define the set of nodes in t_1 and t_2 as N_1 and N_2 respectively. We further define an indicator function $I_i(n)$ to be 1 if subtree i is seen rooted at node n and 0 otherwise. The kernel between t_1 and t_2 can now be written as

$$k(t_1, t_2) = \sum_{i=1}^{M} \Phi_i(t_1)\Phi_i(t_2) = \sum_{i=1}^{M} \sum_{n_1 \in N_1} I_i(n_1) \sum_{n_2 \in N_2} I_i(n_2) = \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} C(n_1, n_2) \quad (2.130)$$

where we define $C(n_1, n_2) = \sum_{i=1}^{M} I_i(n_1) I_i(n_2)$, that is the number of common subtrees rooted at both n_1 and n_2. The following recursive definition permits to compute $C(n_1, n_2)$ in polynomial time:

- If n_1 and n_2 don't match $C(n_1, n_2) = 0$.

- if n_1 and n_2 match, and they are both leaves $C(n_1, n_2) = 1$.
- Else

$$C(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + C(ch(n_1, j), ch(n_2, j))), \tag{2.131}$$

  where $nc(n_1)$ is the number of children of n_1 (equal to that of n_2 for the definition of match) and $ch(n_1, j)$ is the j^{th} child of n_1.

To prove the correctness of (2.131), note that each child of n_1 contributes exactly $1 + C(ch(n_1, j), ch(n_2, j))$ common subtrees for n_1,n_2, the first with the child alone, and the other $C(ch(n_1, j), ch(n_2, j))$ with the common subtrees rooted at the child itself. The product in (2.131) considers all possible combinations of subtrees contributed by different children.

As for the string kernel described before, it's usually convenient to employ a normalized version of the kernel. Moreover, the kernel tends to produce extremely large values for very similar trees, thus making the algorithm behave like a nearest neighbour rule. This effect can be reduced by restricting the depth of the allowed subtrees to a fixed value d, or by scaling their relative importance with their size. To this extent we can introduce a parameter $0 < \lambda \leq 1$, turning the last two points of the definition of C into:

- if n_1 and n_2 match, and they are both leaves $C(n_1, n_2) = \lambda$.
- Else

$$C(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + C(ch(n_1, j), ch(n_2, j))). \tag{2.132}$$

This corresponds to a modified inner product

$$k(t_1, t_2) = \sum_{i=1}^{M} \lambda^{size_i} \Phi_i(t_1) \Phi_i(t_2), \tag{2.133}$$

where $size_i$ is the number of nodes of the corresponding tree fragment.

Graphs

A class of kernels on labelled graphs has been introduced in [GÖ2], exploiting the idea of common paths.

Let $\mathcal{G}$ be a set of labelled directed graphs, and $\mathcal{L} = \{l_r\}_{r \in \mathbb{N}}$ some enumeration of all possible labels. A given graph g is represented by the tuple $g = (\mathcal{V}, \mathcal{E}, L)$, where $\mathcal{V} = \{v_i\}_{i=1}^{|\mathcal{V}|}$ is the vertex set of the graph, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ its edge set, $(v_i, v_j) \in \mathcal{E}$ iff there exists a (directed) edge from v_i to v_j, and L the $|\mathcal{L}| \times |\mathcal{V}|$ sparse matrix representing its labels. That is $[L]_{r,i} = 1$ if l_r is the label of v_i, and zero otherwise, where $[M]_{i,j}$ indicates the element in the i^{th} row and j^{th} column of matrix M. Lets represent the neighbourhood of a vertex v in g with $\delta^+(v) = \{(v, v') \in \mathcal{E}\}$ and $\delta^-(v) = \{(v', v) \in \mathcal{E}\}$, and the maximum in and out degree of a graph as $\Delta^+(g) = \max_{v \in \mathcal{V}}(|\delta^+(v)|)$ and $\Delta^-(g) = \max_{v \in \mathcal{V}}(|\delta^-(v)|)$.

We will equivalently represent a graph g with the matrices $g = (A, L)$, where L is the aforementioned label matrix, and A is the adjacency matrix of the graph, that is $[A]_{ij} = 1 \Leftrightarrow (v_i, v_j) \in \mathcal{E}$, $[A]_{ij} = 0$ otherwise. The adjacency matrix has the useful property that its n power A^n extends the adjacency concept to paths of length n, that is $[A^n]_{i,j}$ is the number of walks [4] of length n from v_i to v_j. Such structural properties can be extended to labelled graphs using LA^nL^T. There exists a walk of length n between two nodes labelled l_i, l_j iff $[LA^nL^T]_{ij} \geq 1$.

We can now represent the similarity between two graphs $g = (A, L), g' = (A', L')$ with the inner products $\{< LA^nL^T, L'A'^nL'^T >\}_n$ for different path lengths n, where the inner product between two $m \times n$ matrices M, M' is defined as $< M, M' >= \sum_{i,j}[M]_{i,j}[M']_{i,j}$. In order to account for insertions or deletions of vertices along a walk, we can consider $\{< LA^iL^T, L'A'^jL'^T >\}_{i,j}$.

The kernels k_n between labelled directed graphs are defined as

$$k_n(g, g') = \sum_{i,j=0}^{n} \lambda_i \lambda_j < LA^iL^T, L'A'^jL'^T >, \tag{2.134}$$

with $\lambda_i \geq 0, \quad \forall i \in [0, n]$. The kernel k_∞ is defined as

$$k_\infty(g, g') = \lim_{n \to \infty} k_n(g, g') \tag{2.135}$$

if such limit exists. It can be proved [GÖ2] that given $a \in \mathbb{N}$ such that

$$a \geq \max_{g \in \mathcal{G}} \min\{\Delta^+(g), \Delta^-(g)\}, \tag{2.136}$$

the following proposition holds, thus providing a sufficient condition for *absolute convergent graph kernels*, while also proving they are valid positive definite kernels.

Proposition 2.5.3 (Absolute Convergent Graph Kernel) *If $\sum_{i=0}^{n} \lambda_i a^i$ is absolute convergent for $n \to \infty$, and $\lim_{i \to \infty} \lambda_i i^2 a^i = 0$ then k_n is absolute convergent for $n \to \infty$.*

The feature map corresponding to such kernels is given by

$$\Phi(g) = \lim_{n \to \infty} \sum_{i=0}^{n} \lambda_i LA^iL^T. \tag{2.137}$$

In the following we report two examples of absolute convergent graph kernels.

[4] A walk is a sequence of vertices and edges $v_1, e_1, v_2, e_2, \ldots, e_n, v_{n+1}$, $e_i \in \mathcal{E}$, $v_i \in \mathcal{V}$, such that $e_i = (v_i, v_{i+1})$.

Definition 2.5.1 (Exponential Graph Kernel) *The exponential graph kernel k_{exp} between labelled directed graphs is defined as*

$$k_{exp}(g,g') = \left\langle Le^{\beta A}L^T, L'e^{\beta A'}L'^T \right\rangle, \tag{2.138}$$

where $\beta \in \mathbb{R}$ is a parameter and the exponential of a matrix M is defined as

$$e^{\beta M} = \lim_{n\to\infty}\sum_{i=0}^{n}\frac{(\beta M)^i}{i!}. \tag{2.139}$$

Definition 2.5.2 (Geometric Graph Kernel) *The geometric graph kernel k_{geom} between labelled directed graphs in a set $\mathcal{G}$ is defined as*

$$k_{geom}(g,g') = \left\langle L\left(\lim_{n\to\infty}\sum_{i=0}^{n}\gamma^i A^i\right)L^T, L'\left(\lim_{n\to\infty}\sum_{i=0}^{n}\gamma^i A'^i\right)L'^T \right\rangle \tag{2.140}$$

for $\gamma < 1/\max_{g\in\mathcal{G}}\min\{\Delta^+(g), \Delta^-(g)\}$.

It's straightforward to extend these kernels to the case of unordered graphs, by treating every undirected edge as consisting of two directed edges, and graphs with weighted edges, by setting $[A]_{i,j} = weight(v_i, v_j)$.

Feasible matrix exponentiation usually requires diagonalizing the matrix. If we can diagonalize A such that $A = T^{-1}DT$, we can easily compute any power of A as $A^n = (T^{-1}DT)^n = T^{-1}D^nT$, where the power of the diagonal matrix D is calculated component-wise $[D^n]_{i,j} = [D_{i,j}]^n$. Therefore we have

$$e^{\beta A} = T^{-1}e^{\beta D}T \tag{2.141}$$

where $e^{\beta D}$ is calculated component-wise.

2.5.4 Kernels from Generative Models

Generative models such as Hidden Markov Models [Rab89] are a principled way to represent the probability distribution underlying the generation of data, and allow to treat missing information, uncertainty and variable length sequences. On the other hand, discriminative methods such as kernel machines are an effective way to build decision boundaries, and often outperform generative models in prediction tasks. It would be thus desirable to have a learning method able to combine these complementary approaches. In the following we will present two different examples of kernels derived from generative models, either by directly representing similarity between sequences by their joint probability distribution [Wat00, Hau99], or by defining a similarity measure between the models underlying two examples [JH99, JH98].

2.5.4.1 Dynamic Alignment Kernels

Joint probability distributions are a natural way of representing relationships between objects. The similarity of two objects can be modeled as a joint probability distribution that assigns high probabilities to pairs of related objects and low probabilities to pairs of unrelated objects. These considerations have been used in [Wat00] to propose a kernel based on joint probability distributions. An analogous kernel was independently presented in [Hau99] as a special case of convolution kernel (see section 2.5.2).

Definition 2.5.3 *A joint probability distribution is* conditionally symmetrically independent *(CSI) if it is a mixture of a finite or countable number of symmetric conditionally independent distributions.*

In order to show that a CSI joint p.d. is a positive definite kernel, let's write it as a dot product. Let X, Z, C be three discrete random variables such that

$$p(x,z) = P(X = x, Z = z) = p(z,x) \tag{2.142}$$

and

$$p(x,z|c) = P(X = x, Z = z|C = c) = p(x|c)p(z|c) \tag{2.143}$$

for all possible realizations of X, Z, C. We can thus write

$$p(x,z) = \sum_c p(x|c)p(z|c)p(c) = \sum_c \left(p(x|c)\sqrt{p(c)}\right)\left(p(z|c)\sqrt{p(c)}\right) \tag{2.144}$$

where the sum is over all possible realizations $c \in \mathcal{C}$ of C. This corresponds to a dot product with feature map

$$\Phi(x) = \{p(x|c)\sqrt{p(c)} \mid c \in \mathcal{C}\}. \tag{2.145}$$

For a more general proof see [Wat00].

A joint p.d. for a finite symbol sequence can be defined with a pair Hidden Markov Model. Such models generate two symbol sequences simultaneously, and are used in bioinformatics to align pairs of protein or DNA sequences [DEKM98]. A PHMM can be defined as follows, where A, B represent the two sequences modeled.

- A finite set S of states, given by the disjoint union of:

  S^{AB} - states that emit one symbol for A and one for B,

  S^{A} - states that emit one symbol only for A,

  S^{B} - states that emit one symbol only for B,

  a starting state START and an ending state END, which don't emit symbols.

- An $|S| \times |S|$ state transition probability matrix T.

- An alphabet $\mathcal{A}$.
- For states emitting symbols:
  - for $s \in S^{AB}$ a probability distribution over $\mathcal{A} \times \mathcal{A}$,
  - for $s \in S^{A}$ or $s \in S^{B}$ a probability distribution over $\mathcal{A}$.

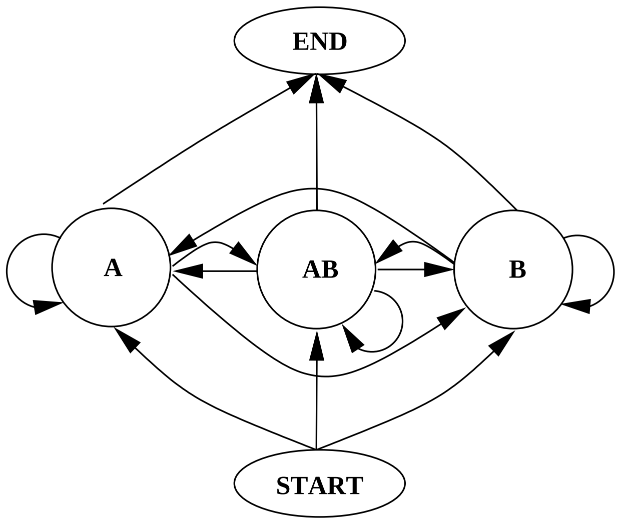


Figure 2.11. State diagram for PHMM modeling pairs of sequences AB. The state AB emits common or similar symbols for both sequences, while the states A and B model insertions in sequence A and B respectively.

The state diagram for this PHMM is represented in figure 2.11. The state AB emits matching or nearly matching symbols for both sequences, while states A an B model insertions, that is symbols found in one sequence but not in the other. The joint p.d. for two sequences is given by the combination of all possible paths from START to END, weighted by their probabilities. This can be efficiently computed by well known dynamic programming algorithms [Rab89].

We will now present sufficient conditions for a PHMM h to be CSI. Let T^{AB*} be the transition probability restricted to $S^{AB*} = S^{AB} \cup \{START\} \cup \{END\}$. That is, for $s, t \in S^{AB*}$, $[T^{AB*}]_{s,t}$ is the probability that, starting from s, the next state in S^{AB*} is t. Let $\mathtt{A}(s,t)$ be the random variable denoting the possibly empty sequence of states in S^{A} passed, starting from $s \in S^{AB*}$, given that t is the next state in S^{AB*} reached. Let $\mathtt{B}(s,t)$ be the analogous random variable for S^{B}.

Proposition 2.5.4 *Let h be a PHMM such that:*

1. *The joint p.d. induced by h is unchanged if S^{A} and S^{B} are swapped.*

2. For all states $s \in S^{AB}$, the symbol emission joint p.d. over $\mathcal{A} \times \mathcal{A}$ is CSI.

3. h has the independent insertion property, that is for all $s, t \in S^{AB*}$ $\mathtt{A}(s,t)$ and $\mathtt{B}(s,t)$ are independent.

Then the joint p.d. induced by h over pairs of sequences of symbols is CSI.

The proof can be found in [Wat00].

2.5.4.2 Fisher Kernel

The basic idea of the Fisher Kernel [JH99, JH98] is that of representing the generative processes underlying two examples into a metric space, and compute a similarity measure in such space. Given a generative probability model $P(X|\theta)$ parametrized by $\theta = (\theta^1, \dots, \theta^r)$, the gradient of its loglikelihood with respect to θ, $V_\theta(X) := \nabla_\theta \log P(X|\theta)$, is called *Fisher score*. It indicates how much each parameter θ^i contributes to the generative process of a particular example. The gradient is directly related to the expected *sufficient statistics* for the parameters. In the case that the generative model is an HMM, such statistics come as a by product of the forward backward algorithm [Rab89] used to compute $P(X|\theta)$, without any additional cost. The derivation of the gradient for HMM and its relation to sufficient statistics is described in [JDH00].

A class of models $P(X|\theta)$, $\theta \in \Theta$ defines a Riemannian manifold M_Θ (see [Ama90, Ama98]), with metric tensor given by the covariance of the Fisher score, called *Fisher information matrix* and computed as

$$F := E_p[V_\theta(X)V_\theta(X)^T], \tag{2.146}$$

where the expectation is over $P(X|\theta)$. The direction of steepest ascent of the loglikelihood along the manifold is given by the *natural gradient* $\tilde{V}_\theta(X) = F^{-1}V_\theta(X)$ (see [Ama98] for a proof). The inner product between such natural gradients relative to the Riemannian metric,

$$k(X, X') = \tilde{V}_\theta(X)^T F \tilde{V}_\theta(X) = V_\theta(X)^T F^{-1} V_\theta(X) \tag{2.147}$$

is called *Fisher kernel.*

When the Fisher information matrix is too difficult to compute, it can be approximated by $F \approx \sigma^2 I$, where I is the identity matrix and σ a scaling parameter.

Moreover, as $V_\theta(X)$ maps X to a vectorial feature space, we can simply use the dot product in such space, giving rise to the *plain* kernel

$$k(X, X') = V_\theta(X)^T V_\theta(X). \tag{2.148}$$

The Fisher kernel has been employed for detecting remote protein homologies [JDH00], where the generative model is chosen to be an HMM representing a given protein family. This approach will be described in section 4.3.2.

2.5.5 Hyper Parameter Tuning

Many kernel functions depend on a few fixed *hyperparameters*, which are not optimized by the training procedure, such as the degree of polynomial kernels (eq. 2.109) or the width of Gaussian kernels (eq. 2.112). Even the regularization parameter C, trading off between training data fitting and model complexity, can be seen as a kernel parameter, by employing a modified kernel [CV95, CST00]

$$K \leftarrow K + \frac{1}{C} I, \tag{2.149}$$

where I is the identity matrix.

Hyperparameter tuning is thus a critical step in order to obtain good generalization performances. The problem is also known as *model selection* in statistics and machine learning, where several early criteria have been proposed (see, e.g., [Aka73, CW79, Vap79]). Model selection usually consists in determining the value of a very small set of hyperparameters. In this case, it can be carried out by calling as a subroutine a learning algorithm that receives hyperparameters as constant input arguments. Therefore, a typical approach for model selection is that of sampling the space of possible hyperparameters, and for each given sample, compute an estimate of the generalization error of the corresponding machine. Such estimate can be computed by a k-fold cross validation procedure as described in section (2.1.4), but it can be very time consuming for high values of k. A more efficient solution is that of using error bounds, such as those on the LOO error described in section (2.2.4) for support vector machines. Note that loose bounds are not a problem for this purpose, provided that their variation as a function of the hyperparameters is similar to that of the generalization error itself. Furthermore, the sampling procedure can be optimized by exploiting dependencies of the bounds on the hyperparameters to tune. For example, Cristianini et al [CCST99] proved that the radius margin bound (eq. 2.44) is smooth in the width of Gaussian kernels, thus deriving an optimized procedure for width tuning in such kernels. Yoshua Bengio [Ben00] proposed a more general method which consists in choosing a differentiable model selection criterion and searching a global optimum in the joint space of parameters and hyperparameters. This approach was applied in [CVBM02] to the case of support vector machines, giving a procedure that alternates the SVM optimization with a gradient step towards the minimization of the generalization error estimate. Let $\boldsymbol{\theta} = (\theta_1, \dots, \theta_n)$ be the vector of hyperparameters of a given kernel k, and $\boldsymbol{\alpha}$ be the set of parameters of the decision function f. Let $E(\boldsymbol{\theta}, \boldsymbol{\alpha})$ be the chosen generalization error estimate, and $W(\boldsymbol{\theta}, \boldsymbol{\alpha})$ the dual optimization problem of the SVM (eq. 2.35). The optimization procedure is defined as:

1. Initialize $\boldsymbol{\theta}$.

2. Keeping $\boldsymbol{\theta}$ fixed, find the maximum of the dual optimization problem:

$$\text{argmax}_{\boldsymbol{\alpha}} W(\boldsymbol{\theta}, \boldsymbol{\alpha})$$

3. Update $\boldsymbol{\theta}$ by a gradient step $\epsilon > 0$ towards the minimization of E:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \epsilon \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}, \boldsymbol{\alpha}).$$

4. If the minimum of E is reached STOP, else goto 2.

The (approximate) gradient of the radius margin (eq. 2.44) and the span (eq. 2.47) error bounds were derived in [CVBM02], provided that the kernel function itself is derivable with respect to the hyperparameters.

Chapter 3

Multiclass Classification

Many machine learning algorithms are intrinsically conceived for binary classification. However, in general, real world learning problems require that inputs are mapped into one of several possible categories. The extension of a binary algorithm to its multiclass counterpart is not always possible or easy to conceive (examples where this is straightforward are decision trees or prototypes methods such as $k-$nearest neighbours). A common alternative consists in *reducing* a multiclass problem into several binary sub-problems. A general reduction scheme is the information theoretic method based on error correcting output codes (ECOC), introduced by Dietterich and Bakiri [DB95] and more recently extended in [ASS00]. The simplest coding strategy, sometimes called "one-hot" or "one-vs-all", consists in defining as many dichotomies of the instance space as the number of classes, where each class is considered as "positive" in one and only one dichotomy. For example, one-hot encoding is the method of choice when using neural networks trained to predict conditional probabilities of the class given the input [Bis95].

Dichotomies can be learned in different ways. Here we are interested in the case of margin-based binary classifiers, as induced by a fairly large class of algorithms that include most kernel machines (see previous chapter), but also classic methods such as the perceptron [Ros58] and its variants [FS98], or boosting algorithms such as AdaBoost [FS97]. They all learn a real-valued function $f(x)$ of an instance x, called the margin of x, and then take the sign of $f(x)$ to obtain classification.

When using margin-based classifiers to implement a set of dichotomies for multiclass problems, the input instance is first mapped to a real vector of margins formed by the outputs of the binary classifiers. A target class is then computed from this vector by means of a decoding function [DB95]. Different strategies have been proposed to choose the decoding function. A simple approach is that of using the Hamming distance, treating the output of each binary classifier as a Boolean variable. However, in the case that the binary learners are margin-based classifiers, Allwein et al. [ASS00] shown the advantage of using the same loss-based function of the margin used to train the binary classifiers. In this chapter, we suggest a different approach which is based on decoding via conditional probabilities of the outputs of the classifiers. The advantages offered by our approach are twofold. Firstly, the use of conditional probabilities allows to combine the margins of each classifier in a principled way. Secondly, the decoding function is itself a class conditional probability which can give an estimate of multiclassification confidence. We report experiments using support vector machines as the base binary classifiers, showing the advantage of the proposed decoding function over other functions of the margin commonly used in practice.

In the case of support vector machine algorithms, a number of recent works [Vap98, WW98, BB99, GEPM00, CS00] addressed the problem of directly extending the algorithm to the multiclass case. Resulting optimization problems have the advantage of simultaneously learning all decision functions, but are computationally very intensive. The first wide comparison of these methods and simple ECOC schemes showed no significant difference for optimal values of the hyperparameters [HL02a] when using Gaussian kernels, while a simple all-pairs scheme achieved the best performances with linear kernels. In this chapter we review the different multiclass methods developed, focusing on their similarities and differences, and show how they can be viewed in the framework of ECOC, basing on the work on continuous codes by Crammer and Singer [CS02b]. Moreover, we run a set of experiments comparing different multiclass methods under various conditions. While agreeing with [HL02a] regarding performances at the optimum, our experiments highlight some interesting differences emerging when methods are forced to produce degenerate solutions.

Finally, we show how to study the generalization error of ECOC and use the results to estimate kernel hyperparameters. Concerning this aspect, we begin by recalling that the problem of estimating the generalization performances of a learning algorithm can be efficiently addressed by mean of a bound on the expected risk, such as a LOO error bound (see section 2.2.4). Moreover, such estimate can be employed for model selection, that is tuning of the hyperparameters of the learning function (see section 2.5.5), provided that its behaviour is similar to that of the expected risk itself, and they are minimized for close values of the hyperparameters. In this chapter we present a general bound on the LOO error for ECOC of kernel machines. The novelty of this analysis is that it allows multiclass parameters optimization even though the binary classifiers are trained independently. We report experiments showing that the bound leads to good estimates of kernel parameters.

The rest of the chapter is organized as follows. In section 3.1 we shortly review the theory of ECOC and the associated loss-based decoding methods. In section 3.1.1 we introduce a new decoding function based on conditional probabilities and give a theoretical justification of its validity. In section 3.2 we review the different approaches to multiclass classification with kernel machines, highlighting their similarities and differences. In section 3.3 we present the bound on the LOO error for general ECOC of kernel machines. Finally, in section 3.4 we empirically validate the usefulness of the theoretical results presented.

The probabilistic decoding function and the LOO error bound are based on the works in [PPF02, PPF04], while the section and results on different multiclass methods are not published elsewhere.

3.1 Error Correcting Output Codes

ECOC work in two steps: training and classification. During the first step, S binary classifiers are trained on S dichotomies of the instance space, formed by joining non overlapping subsets of classes. Assuming Q classes, let us introduce a "coding matrix" $\mathbf{M} \in \{-1, 0, 1\}^{Q \times S}$ which specifies a relation between classes and dichotomies. $m_{qs} = 1$ ($m_{qs} = -1$) means that points belonging to class q are used as positive (negative) examples to train the $s-$th classifier f_s. When $m_{qs} = 0$, points in class q are not used to train the $s-$th classifier. Thus each class q is encoded by the $q-$th row of matrix $\mathbf{M}$ which we denoted by $\mathbf{m}_q$. During prediction a new input x is classified by computing the vector formed by the outputs of the classifiers, $\mathbf{f}(x) = (f_1(x), \ldots, f_S(x))$ and choosing the class whose corresponding row is closest to $\mathbf{f}(x)$. This process is the same as the decoding step for error correcting codes (ECC) in communication [BRC60]. We can therefore see a learning task as the problem of transmitting the correct class of a new example over a noisy channel, given by input features, training data and learning algorithm. In order to recover possible errors in the transmission, the class is encoded in an ECC and each bit is transmitted separately (via a separate run of the learning algorithm). Error recovering depends on the quality of the ECC, which can be measured by the minimum Hamming distance between any pair of code words. A minimum distance of d allows to correct at least $\frac{d-1}{2}$ bit errors, as the nearest codeword would still be the correct one. However, error correction is possible only if bits are independent one another, which is usually not true in the learning context as classifiers are trained on overlapping data. In order to reduce dependence between classifiers, a good ECC should have also a high column separation, both for columns and their complements. Different approaches for ECOC generation have been proposed in [DB95], depending on the number of classes.

Once the matrix is chosen and the binary learners are trained, classification can be seen as a decoding operation, and the class of input x is computed as

$$\arg\min_{q=1}^{Q} d(\mathbf{m}_q, \mathbf{f}(x)) \tag{3.1}$$

where d is the decoding function. A simple choice for d is given by the *Hamming decoding* [ASS00]:

$$d(\mathbf{m}_q, \mathbf{f}) = \sum_{s=1}^{S} \frac{|m_{qs} - \text{sign}(f_s)|}{2}, \tag{3.2}$$

which is equal to the Hamming distance for binary valued matrices $\mathbf{M}$, while in the general case zero entries contribute $1/2$ to the sum. When the binary learners are margin-based classifiers, [ASS00] shown the advantage of using a loss-based function of the margin

$$d_L(\mathbf{m}_q, \mathbf{f}) = \sum_{s=1}^{S} L(m_{qs} f_s)$$

where L is a loss function (see section 2.1.1) which depends on the confidence margin (see section 2.2.1). L is typically a non-decreasing function of the margin and, thus, weights the confidence of each classifier according to the margin. The simplest loss function one can use is the linear loss for which $L(m_{qs}f_s) = -m_{qs}f_s$. When all binary classifiers are computed by the same learning algorithm, Allwein et al. [ASS00] proposed to set L to be the same loss function used by that algorithm.

It is worthwhile noting that the ECOC framework includes two multiclass classification approaches often used in practice: one-vs-all [Nil65] and all-pairs [Fri96]. In the former approach there is one classifier per class, which separates it from all the others. A new input is assigned to the class whose associated classifier has the maximum output. In the ECOC framework one-vs-all is equivalent to linear decoding with a $Q \times Q$ coding matrix whose entries are always -1 except diagonal entries which are equal to 1. Note that one-vs-all is not actually a correcting code, as its minimum Hamming distance is two, and thus cannot correct any error. Nevertheless, it turns out to be often quite effective with SVM as binary learners, as showed in section 3.4, where we will also present some reasons for this behaviour. In the all-pairs approach, also known as *round robin classification* [F02], there are $Q(Q-1)/2$ classifiers, each separating a pair of classes. Classification is decided by majority voting. This scheme is equivalent to Hamming decoding with the appropriate coding matrix.

In the next section we propose a new probabilistic approach, which is based on decoding via conditional probabilities of the outputs of the classifiers.

3.1.1 Decoding Functions Based on Conditional Probabilities

We mentioned before that a loss function of the margin may have some advantages over the standard Hamming distance because it can encode the confidence of each classifier in the ECOC. This confidence is, however, a relative quantity, i.e. the range of the values of the margin may vary with the classifier used. Thus, just using a linear loss function may introduce some bias in the final classification in the sense that classifiers with a larger output range will receive a higher weight. Not surprisingly, we will see in the experiments in section 3.4 that the Hamming decoding usually works better than the linear one in the case of pairwise schemes. A straightforward normalization in some interval, e.g. $[-1, 1]$, can also introduce bias since it does not fully take into account the margin distribution. A more principled approach is to estimate the conditional probability of each class q given the input x. Given the S trained classifiers, we assume that all the information about x that is relevant for determining the class is contained in the margin vector $\mathbf{f}(x)$ (or $\mathbf{f}$ for short), i.e. $P(Y = q|x) = P(Y = q|\mathbf{f})$. Let us now introduce the set of all possible codewords $\mathbf{o}_k$, $k = 1, \ldots, 2^S$ and let $\mathbf{O}$ be a random vector of binary variables. A realization of $\mathbf{O}$ will be a codeword. For simplicity, we shall use the symbols -1 and $+1$ to denote codebits.

The probability of Y given the margin vector can thus be rewritten by marginalizing out codewords and decomposing using the chain rule:

$$P(Y=q|\mathbf{f}) = \sum_{k=1}^{2^S} P(Y=q|\mathbf{O}=\mathbf{o}_k,\mathbf{f})P(\mathbf{O}=\mathbf{o}_k|\mathbf{f}).$$

The above model can be simplified by assuming the class to be independent of $\mathbf{f}$ given the codeword $\mathbf{o}_k$. This assumption essentially means that $\mathbf{f}$ has a direct causal impact on $\mathbf{O}$, and in turn $\mathbf{O}$ has a direct causal impact on Y. As a result:

$$P(Y=q|\mathbf{f}) = \sum_{k=1}^{2^S} P(Y=q|\mathbf{O}=\mathbf{o}_k)P(\mathbf{O}=\mathbf{o}_k|\mathbf{f}).$$

We choose a simple model for the probability of class q given the codeword $\mathbf{o}_k$ by looking at the corresponding row $\mathbf{m}_q$ in the coding matrix. A zero entry in the row is treated as "don't care", i.e. replacing it with a a value of 1 or -1 results in an equally correct codeword for the class. Thus each class q has a number of valid codes $\mathcal{C}_q$ given by all possible substitutions of 1 or -1 in the zero entries of $\mathbf{m}_q$. Invalid codes $\bar{\mathcal{C}}$ are those which are not valid for any class $q \in Q$. We then define

$$P(Y=q|\mathbf{O}=\mathbf{o}_k) = \begin{cases} 1 & \text{if } \mathbf{o}_k \in \mathcal{C}_q \\ 0 & \text{if } \mathbf{o}_k \in \mathcal{C}_{q'} \text{ with } q' \neq q \\ \frac{1}{Q} & \text{otherwise (i.e. } \mathbf{o}_k \in \bar{\mathcal{C}} \text{ is not a valid class code)} \end{cases}$$

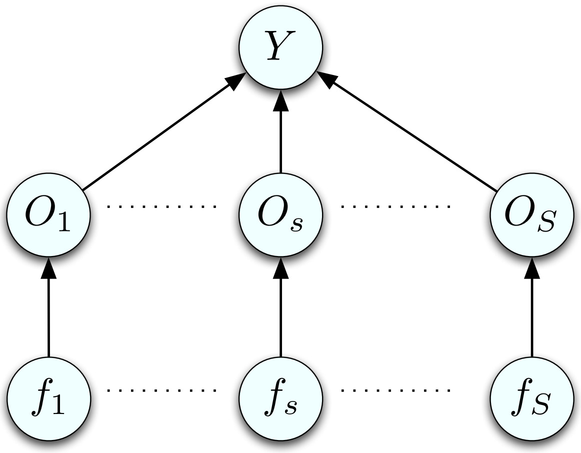


Figure 3.1. Bayesian network describing the probabilistic relationships amongst margins, codewords, and class.

Under this model,

$$P(Y=q|\mathbf{f}) = \sum_{\mathbf{o}_k \in \mathcal{C}_q} P(\mathbf{O}=\mathbf{o}_k|\mathbf{f}) + \alpha$$

where

$$\alpha = \frac{1}{Q} \sum_{\mathbf{o}_k \in \bar{\mathcal{C}}} P(\mathbf{O} = \mathbf{o}_k | \mathbf{f})$$

collects the probability mass dispersed on the invalid codes. We further assume that each individual codebit O_s is conditionally independent of the others given $\mathbf{f}$, and that it is also independent of the other outputs $f_{s'}$ given f_s (in other words, we are assuming that the only cause for O_s is f_s). Our conditional independence assumptions can be graphically described by the Bayesian network depicted in figure 3.1. As a result, we can write the conditional probability of the class q as

$$P(Y = q \,| \mathbf{f}) = \sum_{\mathbf{o}_k \in \mathcal{C}_q} \prod_{s=1}^{S} P(O_s = o_{ks} \,| f_s) + \alpha. \tag{3.3}$$

We further note that the probability of a bit corresponding to a zero value in the coding matrix is independent of the output of the classifier (it is a “don’t care” bit). Moreover, it should be equally distributed between the possible realizations $\{-1, 1\}$. All valid codes $\mathbf{o}_k \in \mathcal{C}_q$ for a given class q have then the same probability

$$\prod_{s=1}^{S} P(O_s = o_{ks} \,| f_s) \;=\; \frac{1}{2^Z} \prod_{s \in S : m_{qs} \neq 0} P(O_s = m_{qs} \,| f_s)$$

where Z is the number of zero entries in the row corresponding to the class. By noting that there are exactly 2^Z of such valid codes, we can simplify equation (3.3) to

$$P(Y = q \,| \mathbf{f}) = \prod_{s \in S : m_{qs} \neq 0} P(O_s = m_{qs} \,| f_s) + \alpha. \tag{3.4}$$

In this case the decoding function will be:

$$d(\mathbf{m}_q, \mathbf{f}) = -\log P(Y = q \,| \mathbf{f}). \tag{3.5}$$

The problem boils down to estimating the individual conditional probabilities in eq. (3.4), a problem that has been addressed also in [Kwo99, Pla00]. Our solution consists of fitting the following set of parametric models:

$$P(O_s = m_{qs} \,| f_s) = \frac{1}{1 + \exp(m_{qs}(A_s f_s + B_s))}$$

where A_s and B_s are adjustable real parameters that reflect the slope and the offset of the cumulative distribution of the margins. A_s and B_s can be estimated independently by maximizing the following set of Bernoulli log-likelihoods:

$$\sum_{i : m_{y_i s} \neq 0} \log \left(\frac{1}{1 + \exp(m_{y_i s}(A_s f_s(x_i) + B_s))} \right). \tag{3.6}$$

The index i in equation (3.6) runs over the training examples (x_i, y_i). It must be observed that fitting the sigmoid parameters using the same examples used to train the margin classifiers would unavoidably lead to poor estimates since the distribution of $f_s(x_i)$ is very different for training and for testing instances (for example, in the case of separable SVM, all the support vectors contribute a confidence margin that is exactly +1 or -1). To address this, in our experiments we used a 3-fold cross validation procedure to fit A_s and B_s, as suggested in [Pla00].

We remark that an additional advantage of the proposed decoding algorithm is that the multiclass classifier outputs a conditional probability rather than a mere class decision.

3.2 Multiclass Classification with Kernel Machines

In this section we will discuss the most common methods for extending kernel machines for binary classification to the multiclass case. A first approach is that of employing ECOC as those described in the previous section, using kernel machines as the underlying binary classifier. However, different approaches have been proposed [Vap98, WW98, BB99, GEPM00, CS00], which directly extend the optimization problem of SVM to the multiclass case. Basing on the work of Crammer and Singer [CS00] on continuous codes, we will show how these approaches are related to the encoding/decoding process of ECOC algorithms.

3.2.1 ECOC of Kernel Machines

Margin based kernel machines for binary classification can be defined according to regularization theory (see section 2.4.2) as the minimizers of functionals of the form

$$F[f; D_n] = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i f(x_i)) + \lambda ||f||_K^2 \tag{3.7}$$

where $D_n = \{(x_i, y_i) \in \mathcal{X} \times \{-1, 1\}\}_{i=1}^n$ is a training set, $\ell : \mathbb{R} \to \mathbb{R}$ is a monotonic non decreasing margin based loss, and λ a regularization parameter. Assuming ℓ is convex, the minimizer of the functional in (3.7) is unique (see theorem 2.4.2) and has the form [1]

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x). \tag{3.8}$$

ECOC of kernel machines are obtained by combining such binary classifiers as described in section 3.1. The decoding function can be made the same as the binary loss ℓ as suggested in [ASS00], but different choices are possible. In the case of support vector machines as binary classifiers, our experimental results (see section 3.4.1) show that the probabilistic decoding function we introduced in section (3.1.1) is often better than other common loss-based schemes, especially for non optimal choices of the hyperparameters.

[1] We assume that the bias term is incorporated into the kernel k (see [MM99, FCC98, HL02b]).

An advantage of ECOC over other multiclassification methods such as those described in the following section, is that it allows to employ different learning functions for different bits of the codewords, thus giving great flexibility in adapting the learning algorithm.

3.2.2 Multicategory Support Vector Machines

The first extensions of support vector machines to the multiclass case were independently developed by Vapnik [Vap98] and Weston and Watkins [WW98]. As they differ in the choice of the loss function, we will briefly review both methods, starting with the one by Vapnik and pointing out the differences.

Given a training set $D_m = \{\{(\mathbf{x}_i, y_i) \in \mathcal{X} \times [1, Q]\}_{i=1}^m$ of examples belonging to one of Q possible classes, the optimization problem is

$$\min_{\mathbf{w}\in\mathcal{X}^Q,\, b\in\mathbb{R}^Q,\, \boldsymbol{\xi}\in\mathbb{R}^m} \quad \frac{1}{2}\sum_{q=1}^{Q}||\mathbf{w}_q||^2 + C\sum_{i=1}^{m}\xi_i \tag{3.9}$$

subject to

$$< \mathbf{w}_{y_i}, \mathbf{x}_i > +b_{y_i} - < \mathbf{w}_p, \mathbf{x}_i > -b_p \geq 1 - \xi_i, \quad i \in [1, m], p \in [1, Q]\backslash\{y_i\} \tag{3.10}$$

$$\xi_i \geq 0, \quad i \in [1, m]. \tag{3.11}$$

The constraints (3.10) amount to stating that each example must be correctly classified with a gap of at least one with respect to the highest other possible assignment, suffering a linear loss for any violation. The left hand side of such inequality is the multiclass extension of the concept of confidence margin for binary classification (see sec. 2.2.1), and is negative in case of errors. If the point is correctly classified, such value is equivalent to the confidence measure of the class with highest activity in one-per-class neural networks [LBD+89]. The loss induced by these constraints gives a linear penalty to examples classified with multiclass confidence margin less than one. The corresponding decision function is given by

$$f(\mathbf{x}) = \text{argmax}_{q\in Q}\left(< \mathbf{w}_q, \mathbf{x} > +b_q\right). \tag{3.12}$$

The optimization problem can be solved by finding the saddle point of the Lagrangian

$$L = \frac{1}{2}\sum_{q=1}^{Q}||\mathbf{w}_q||^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\sum_{p\neq y_i}\alpha_i^p[< (\mathbf{w}_{y_i} - \mathbf{w}_p), \mathbf{x}_i > +b_{y_i} - b_p - 1 + \xi_i] - \sum_{i=1}^{m}\beta_i\xi_i \tag{3.13}$$

with constraints

$$\alpha_i^p \geq 0, \beta_i \geq 0, \xi_i \geq 0, \quad i \in [1, m], p \in [1, Q]\backslash\{y_i\}. \tag{3.14}$$

By vanishing the derivatives of the primal variables we obtain

$$\frac{\partial L}{\partial \mathbf{w}_q} = 0 \Rightarrow \mathbf{w}_q = \sum_{i:y_i=q} \sum_{p \neq q} \alpha_i^p \mathbf{x}_i - \sum_{i:y_i \neq q} \alpha_i^q \mathbf{x}_i \tag{3.15}$$

$$\frac{\partial L}{\partial b_q} = 0 \Rightarrow \sum_{i:y_i=q} \sum_{p \neq q} \alpha_i^p = \sum_{i:y_i \neq q} \alpha_i^q \tag{3.16}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow \beta_i + \sum_{p \neq y_i} \alpha_i^p = C \text{ and } 0 \leq \sum_{p \neq y_i} \alpha_i^p \leq C. \tag{3.17}$$

By substituting (3.15), (3.16) and (3.17) into the Lagrangian (3.13) and rearranging we obtain the dual formulation, quadratic in α, where both the slack variables and the bias terms cancel:

$$\begin{aligned} \max_{\boldsymbol{\alpha}^q \in \mathbb{R}^m,\, q \in [1,Q]} \quad & \sum_{i=1}^{m} \sum_{p \neq y_i} \alpha_i^p - \frac{1}{2} \sum_{q=1}^{Q} \sum_{i:y_i=q, j:y_j=q} \sum_{p \neq q, r \neq q} \alpha_i^p \alpha_j^q < \mathbf{x}_i, \mathbf{x}_j > \\ & -\frac{1}{2} \sum_{q=1}^{Q} \sum_{i:y_i \neq q, j:y_j \neq q} \alpha_i^q \alpha_j^q < \mathbf{x}_i, \mathbf{x}_j > \\ & + \sum_{q=1}^{Q} \sum_{i:y_i=q} \sum_{p \neq q} \sum_{j:y_j \neq p} \alpha_i^p \alpha_j^p < \mathbf{x}_i, \mathbf{x}_j > \end{aligned} \tag{3.18}$$

with constraints (3.16) and (3.17). The corresponding decision function is given by:

$$f(\mathbf{x}) = \text{argmax}_{q \in Q} \left(\sum_{i:y_i=q} \sum_{p \neq q} \alpha_i^p < \mathbf{x}_i, \mathbf{x} > - \sum_{i:y_i \neq q} \alpha_i^q < \mathbf{x}_i, \mathbf{x} > + b_q \right). \tag{3.19}$$

As for the binary case, both in the dual optimization problem and in the resulting decision function the examples only appear in inner products, thus allowing to apply the kernel trick and replace $< \mathbf{x}_i, \mathbf{x}_j >$ with $k(x_i, x_j)$, being k a valid kernel function.

In this formulation, each training example $(\mathbf{x}_i, y_i)$ is associated with a vector of $Q-1$ alphas $\boldsymbol{\alpha}_i$. By the KKT conditions [Fle87], we can enlighten the connections between the values of the alphas and role of the example in the learned model (see fig. 3.2).

- If all alphas are zero, the multiclass confidence margin of $\mathbf{x}_i$ is greater than one, and the example is not a support vector.
- if $\sum_{q \neq y} \alpha_i^q < C$, the example is an unbound support vector, it has multiclass confidence margin equal to one, and the corresponding slack variable is zero.
- if $\sum_{q \neq y} \alpha_i^q = C$, the example is a bound support vector, with confidence margin less than one and positive slack variable. If $\xi_i > 1$, the example is also a training error.

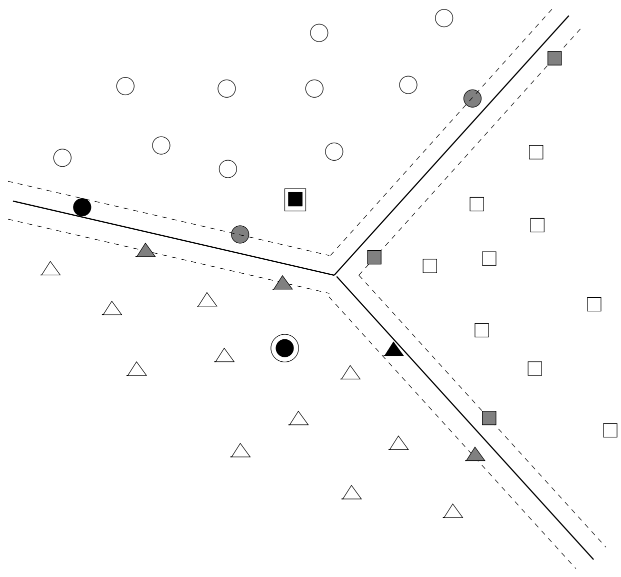


Figure 3.2. Multiclass classification problem solved by multicategory extension of support vector machines. Solid lines represent separating hyperplanes, while dotted lines are hyperplanes with confidence margin equal to one. Grey points are unbound SVs, black points are bound SVs and extra borders indicate bound SVs which are also training errors. All other points do not contribute to the decision function.

A slightly different formulation was developed in [WW98], where they suggest the following constrained minimization problem:

$$\min_{\mathbf{w}\in\mathcal{X}^Q,\, b\in\mathbb{R}^Q,\, \xi_i^q\in\mathbb{R}, i\in[1,m], q\in[1,Q]\setminus\{y_i\}} \quad \frac{1}{2}\sum_{q=1}^{Q}||\mathbf{w}_q||^2 + C\sum_{i=1}^{m}\sum_{q\neq y_i}\xi_i^q \tag{3.20}$$

$$\text{subject to} \quad <\mathbf{w}_{y_i},\mathbf{x}_i> + b_{y_i} \geq <\mathbf{w}_q,\mathbf{x}_i> + b_q + 1 - \xi_i^q, \tag{3.21}$$

$$\xi_i^q \geq 0, \tag{3.22}$$

$$i\in[1,m], q\in[1,Q]\setminus\{y_i\}.$$

The constraints amount to stating that each example must be correctly classified with a confidence of at least one more than *each* other possible assignment, suffering a linear loss for any violation. Here we have a distinct penalty for each assignment different from the correct one, increasing the number of variables in the optimization problem.

The Lagrangian is given by

$$L = \frac{1}{2}\sum_{q=1}^{Q}||\mathbf{w}_q||^2 + C\sum_{i=1}^{m}\sum_{q=1}^{Q}\xi_i^q - \sum_{i=1}^{m}\sum_{q=1}^{Q}\alpha_i^q[<(\mathbf{w}_{y_i}-\mathbf{w}_q),\mathbf{x}_i> + b_{y_i} - b_q - 1 + \xi_i^q] - \sum_{i=1}^{m}\sum_{q=1}^{Q}\beta_i^q\xi_i^q \tag{3.23}$$

with constraints

$$\alpha_i^q \geq 0, \beta_i^q \geq 0, \xi_i^q \geq 0, \quad i \in [1,m], q \in [1,Q]\backslash\{y_i\}. \tag{3.24}$$

and dummy variables

$$\alpha_i^{y_i} = 0, \xi_i^{y_i} = 1, \beta_i^{y_i} = 0, \quad i \in [1,m]. \tag{3.25}$$

We can reach a more compact form using the following notation

$$c_i^n = \begin{cases} 1 & \text{if } y_i = n \\ 0 & \text{if } y_i \neq n \end{cases} \tag{3.26}$$

and

$$A_i = \sum_{q=1}^{Q} \alpha_i^q. \tag{3.27}$$

By vanishing the derivatives of the primal variables we obtain

$$\frac{\partial L}{\partial \mathbf{w}_q} = 0 \Rightarrow \mathbf{w}_q = \sum_{i=1}^{m} (c_i^q A_i - \alpha_i^q)\mathbf{x}_i \tag{3.28}$$

$$\frac{\partial L}{\partial b_q} = 0 \Rightarrow \sum_{i=1}^{m} \alpha_i^q = \sum_{i=1}^{m} c_i^q A_i \tag{3.29}$$

$$\frac{\partial L}{\partial \xi_i^q} = 0 \Rightarrow \beta_i^q + \alpha_i^q = C \text{ and } 0 \leq \alpha_i^q \leq C. \tag{3.30}$$

Substituting (3.28) (3.29) (3.30) into the Lagrangian (3.23) and rearranging we obtain the dual formulation

$$\max_{\boldsymbol{\alpha}^q \in \mathbb{R}^m,\, q \in [1,Q]} \quad \sum_{i,q} \alpha_i^q + \sum_{i,j,q} (-\frac{1}{2} c_j^{y_i} A_i A_j + \alpha_i^q \alpha_j^{y_i} - \frac{1}{2} \alpha_i^q \alpha_j^q) < \mathbf{x}_i, \mathbf{x}_j > \tag{3.31}$$

which is a quadratic function of α with linear constraints (3.29), (3.30) and $\alpha_i^{y_i} = 0$ for all $i \in [1,m]$.

The resulting decision function is

$$f(\mathbf{x}) = \text{argmax}_{q \in Q} \left(\sum_{i=1}^{m} (c_i^q A_i - \alpha_i^q) < \mathbf{x}_i, \mathbf{x} > + b_q \right). \tag{3.32}$$

Again, we can substitute $< \mathbf{x}_i, \mathbf{x}_j >$ with $k(x_i, x_j)$, being k a valid kernel function.

Using the definitions of c_i^q and A_i and the constraints $\alpha_i^{y_i} = 0$ for all $i \in [1,m]$, it's easy to show that (3.32) can be rewritten as (3.19). However, the role of a support vector in terms of its alpha values is different. By the KKT conditions [Fle87], for a training example $(\mathbf{x}_i, y_i)$ with $Q-1$ alpha values $\boldsymbol{\alpha}_i$ (recall that $\alpha_i^{y_i} = 0$ for all i) we have the following situations.

- If $\alpha_i^q = 0$, the confidence with which the correct class y_i is preferred over class q is greater than one. If this holds for each q, the example has multiclass confidence margin greater than one, and is not a support vector.
- If $0 < \alpha_i^q < C$ the confidence with which the correct class y_i is preferred over class q is equal to one and the corresponding slack variable ξ_i^q is zero. If this holds for all q for which the corresponding alpha is greater than zero, the example is an unbound support vector, with multiclass confidence margin equal to one.
- if $\alpha_i^q = C$, the confidence of y_i versus q is less that one, and the corresponding slack variable ξ_i^q is greater than zero. Thus, the example is a bound support vector. If for some q $\xi_i^q > 1$, the example is also a training error.

Other kinds of multiclass support vector machines were proposed in [BB99] and [GEPM00], with constraints (3.21) and (3.22), but different regularization term, given by

$$\frac{1}{2}\sum_{q<p}^{Q} ||\mathbf{w}_p - \mathbf{w}_q||^2 + \sum_{q=1}^{Q} ||\mathbf{w}_q||^2 \tag{3.33}$$

and

$$\frac{1}{2}\sum_{q<p}^{Q} ||\mathbf{w}_p - \mathbf{w}_q||^2, \tag{3.34}$$

respectively, where for the latter an additional sum to zero constraint $\sum_{q=1}^{Q} \mathbf{w}_q = 0$ is necessary in order to have a unique global minimum. However, it was soon proved [CS02b, Gue02] that all these objective functions are equivalent modulo a multiplicative factor, as it holds

$$\sum_{q<p}^{Q} ||\mathbf{w}_p - \mathbf{w}_q||^2 = Q\sum_{q=1}^{Q} ||\mathbf{w}_q||^2 \tag{3.35}$$

and the sum to zero constraint is always satisfied at the optimum. This equivalence also shows that multiclass support vector machines (MSVM) actually search separating hyperplanes with large geometric margins. Consider for simplicity the case of linearly separable MSVM. Given two classes p and q, the separating hyperplane between them is given by $< \mathbf{w}_q - \mathbf{w}_p, \mathbf{x} > +b_q - b_p = 0$ (see fig. 3.3), and the hyperplanes for a confidence margin equal to one are $< \mathbf{w}_q - \mathbf{w}_p, \mathbf{x} > +b_q - b_p = 1$ and $< \mathbf{w}_q - \mathbf{w}_p, \mathbf{x} > +b_q - b_p = -1$ for class q and class p respectively. The distance between the last two hyperplanes is the geometric margin of the separation between q and p, given by

$$\frac{2}{||\mathbf{w}_q - \mathbf{w}_p||}.$$

Therefore, by minimizing (3.35), we actually maximize the sum of all the (squared) biclass geometric margins. The multiclass geometric margin for the MSVM is given by the minimum of its biclass geometric margins. Soft margin MSVM relax the separability condition by introducing slack variables to penalize violations. For a study on bounds on the generalization capabilities of MSVM see [EGPM99, GEZ02].

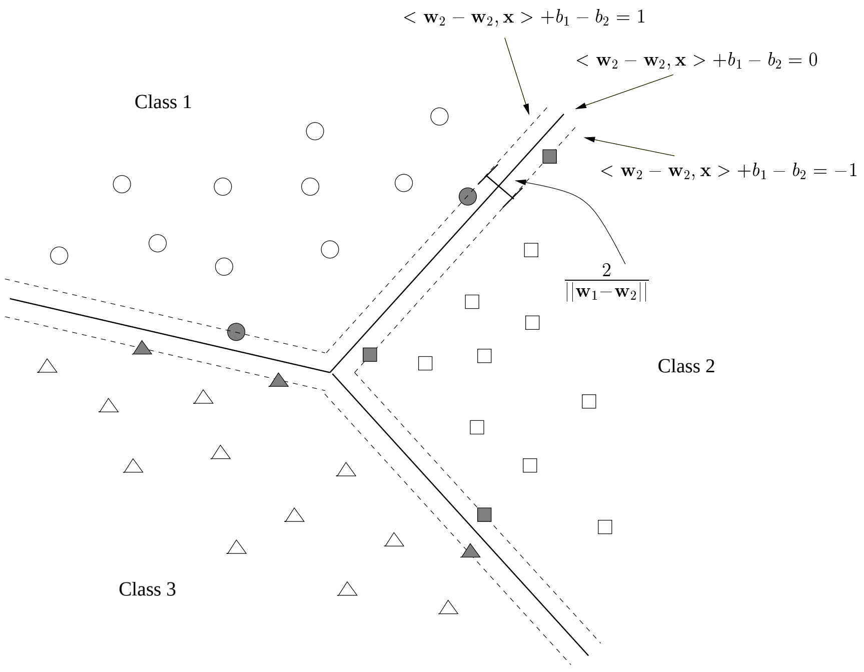

$< \mathbf{w}_2 - \mathbf{w}_2, \mathbf{x} > +b_1 - b_2 = 1$
$< \mathbf{w}_2 - \mathbf{w}_2, \mathbf{x} > +b_1 - b_2 = 0$
$< \mathbf{w}_2 - \mathbf{w}_2, \mathbf{x} > +b_1 - b_2 = -1$
Class 1
$\frac{2}{||\mathbf{w}_1 - \mathbf{w}_2||}$
Class 2
Class 3

Figure 3.3. Multiclass classification problem solved by multicategory extension of support vector machines. Solid lines represent separating hyperplanes, while dotted lines are hyperplanes with confidence margin equal to one. Grey points are unbound SVs. The multiclass geometric margin is given by the minimum of the biclass geometric margins.

3.2.3 Connections between ECOC and MSVM

Training ECOC algorithms amounts to training each binary function associated with each single bit of the code, according to the dichotomy induced by the corresponding column. A complementary approach investigated by Crammer and Singer [CS00] consists of fixing the binary functions, and learning the ECOC matrix. However, they proved that optimal learning of discrete coding matrices is NP-complete. Therefore, they turned to continuous codes learning, and formulated the task as a constrained optimization problem.

$$\begin{aligned} \min_{\mathbf{M}\in \mathbb{R}^{Q\times L}, \boldsymbol{\xi}\in \mathbb{R}^m} \quad & ||(\mathbf{M}_1, \ldots, \mathbf{M}_Q)||_p + C \sum_{i=1}^{m} \xi_i \qquad (3.36) \\ \text{subject to} \quad & \Psi(\mathbf{g}(\mathbf{x}_i), \mathbf{M}_{y_i}) - \Psi(\mathbf{g}(\mathbf{x}_i), \mathbf{M}_r) \geq 1 - \delta_{y_i r} - \xi_i \quad \forall i, r. \end{aligned}$$

Here, $\mathbf{M}$ is a $Q \times L$ real coding matrix, $\mathbf{M}_r$ denotes the r^{th} row of $\mathbf{M}$, $\mathbf{g} = [g_1, \ldots, g_L]$ is a vector of real encoding functions ($g_i : \mathcal{X} \to \mathrm{I\!R}$), $\Psi : \mathrm{I\!R}^L \times \mathrm{I\!R}^L \to \mathrm{I\!R}$ computes the similarity between output vectors and codewords, and δ_{ij} is the Kronecker symbol ($\delta_{ij} = 1$ if $i = j$, 0 otherwise). Note that for $r = y_i$ we obtain the non negativity constraints on ξ_i.

A problem analogous to MSVM can be obtain as a special case of (3.36), by setting $L = m$, $p = 2$ (and using the square norm), $g_i(\mathbf{x}) = \mathbf{x}$ for all g_i and $\mathbf{x}$, and the similarity S to be the dot product $\Psi(\mathbf{x}, \mathbf{M}_r) :=< \mathbf{x}, \mathbf{M}_r >$. The resulting optimization problem is:

$$\min_{\mathbf{M} \in \mathrm{I\!R}^{Q \times m}, \boldsymbol{\xi} \in \mathrm{I\!R}^m} \quad \sum_{q=1}^{Q} ||\mathbf{M}_q||^2 + C \sum_{i=1}^{m} \xi_i \tag{3.37}$$

$$\text{subject to} \quad < \mathbf{x}_i, \mathbf{M}_{y_i} > - < \mathbf{x}_i, \mathbf{M}_r > \geq 1 - \delta_{y_i r} - \xi_i \quad \forall i, r. \tag{3.38}$$

The only difference with respect to Vapnik's MSVM formulation (3.9) is the absence of the bias terms, leading to fewer constraints on the dual problem, which can be decomposed more easily. A compact representation of the dual problem and an efficient algorithm for solving it are developed in [CS00, CS02a]. Note that the use of the Kronecker symbol results in a Lagrangian where multipliers β_i (see eq.(3.13)) are replaced by $\alpha_i^{y_i}$, and the inequality constraint on the alphas (see eq.(3.17)) present in the dual formulation is replaced by an equality one, while the decision function can be written

$$f(\mathbf{x}) = \mathrm{argmax}_{q \in Q} \left(\sum_{i:y_i = q} (C - \alpha_i^q) < \mathbf{x}_i, \mathbf{x} > - \sum_{i:y_i \neq q} \alpha_i^q < \mathbf{x}_i, \mathbf{x} > \right). \tag{3.39}$$

By looking directly at the general formulation (3.36), and simply setting $p = 2$ (and using the square norm) and Ψ as the dot product, we can more clearly enlighten the connections between the encoding/decoding process of ECOC algorithms and MSVM. It turns out that the encoding function $\mathbf{g}$ is actually the feature map Φ induced by the kernel, L is the dimension of the feature space, Ψ is the inner product in such space, and the codeword M_r is the vector of parameters of the hyperplane separating r from the other classes in feature space. Therefore, the encoding process consists in mapping the input into a (possibly) higher dimensional feature/codeword space, and code learning consists in searching hyperplanes/class-codewords in the code space which give the maximum separation between codewords corresponding to examples of different classes. The encoding process can be done only implicitly, as the kernel function directly computes the similarity between inputs corresponding to the inner product of their encodings. Moreover, by choosing the kernel we implicitly choose the encoding function, and by tuning kernel hyperparameters we can actually combine encoding function and class codewords learning.

3.3 Bounds on the LOO Error

The LOO error is an almost unbiased estimate of the expected generalization error (see section 2.1.4), but it can be very expensive to compute if the training set is large. In order to make algorithm evaluation feasible, upper bounds on the LOO error have been proposed for different learning tasks, included SVM for binary classification (see section 2.2.4). Moreover, such estimates can be employed for model selection, that is tuning of the hyperparameters of the learning function (see section 2.5.5). It suffices that the bound behaviour is similar to that of the expected risk, and that they are minimized for close values of the hyperparameters. In the following section we present a general bound on the LOO error for ECOC of kernel machines. The novelty of this analysis is that it allows multiclass parameters optimization even though the binary classifiers are trained independently. We report experiments (see sec. 3.4.3) showing that the bound leads to good estimates of kernel parameters.

3.3.1 LOO Error Bounds for ECOC of Kernel Machines

We will now derive a general bound on the LOO error of error correcting output codes, in the case that the binary classifiers are kernel machines. We start by defining the multiclass confidence margin [ASS00] of point $(x, y) \in \mathcal{X} \times \{1, \ldots, Q\}$ as

$$g(x,y) = d_L(\mathbf{m}_p, \mathbf{f}(x)) - d_L(\mathbf{m}_y, \mathbf{f}(x))$$

with

$$p = \text{argmin}_{q \neq y} d_L(\mathbf{m}_q, \mathbf{f}(x)).$$

Considered that the predicted class for x is the one whose row is closest to $\mathbf{f}(x)$ (see eq. (3.1)), this margin represents the confidence of the correct prediction. If $g(x, y)$ is negative, point x is misclassified. When L is the linear loss, $g(x, y)$ reduces to the definition of multiclass confidence margin for unbiased MSVM. In the following we will always refer to the confidence margin simply with margin.

The empirical misclassification error can be written in terms of the multiclass margin as:

$$\frac{1}{n}\sum_{i=1}^{n} \theta\left(-g(x_i, y_i)\right)$$

where $\theta(\cdot)$ is the Heavyside function: $\theta(x) = 1$ if $x > 0$ and zero otherwise. Similarly, the LOO error can be written as

$$\frac{1}{n}\sum_{i=1}^{n} \theta\left(-g^i(x_i, y_i)\right) \tag{3.40}$$

where we have denoted by $g^i(x_i, y_i)$ the margin of example x_i when the ECOC is trained on the data set $D_n \backslash \{(x_i, y_i)\}$. As described in section 2.1.4, the LOO error is an almost unbiased estimator for the generalization error. Unfortunately, computing the LOO error is time demanding when n is large. Moreover, if the purpose is to use it for model selection (see section 2.5.5), the estimate has to be computed for several values of the hyperparameters to tune. In the case of binary SVM, bounds on the LOO error have been proposed (see section 2.2.4), which can be computed with little or no additional effort once the machine is trained. In the following we will derive a bound on the LOO error of general ECOC of kernel machines, which only depends on the solution of the machines trained on the full data set (so training the machines once will suffice). To this end we first need the following lemma.

Lemma 3.3.1 *Let f be the kernel machine as defined in equations (3.8) obtained by solving (3.7). Let f^i be the solution of (3.7) found when the data point (x_i, y_i) is removed from the training set. We have*

$$y_i f(x_i) - \alpha_i G_{ii} \leq y_i f^i(x_i) \leq y_i f(x_i). \tag{3.41}$$

Proof: The l.h.s part of Inequality (3.41) was proved in [JH98]. Let's prove the r.h.s part. Note that, if x_i is not a support vector, $\alpha_i = 0$ and $f = f^i$, so both inequalities are trivial in this case. Thus suppose that x_i is a support vector. We observe that $F[f; D_n] \leq F[f^i; D_n]$, because otherwise f would not be the minimizer of $F[f; D_n]$. Similarly $F[f^i; D_n^i] \leq F[f; D_n^i]$, as f^i is the minimizer of $F[f^i; D_n^i]$. By combining these two inequalities we have

$$F[f; D_n] - F[f; D_n^i] \leq F[f^i; D_n] - F[f^i; D_n^i].$$

By substituting the definition of F (eq. 3.7) and simplifying we obtain

$$\ell(y_i f(x_i)) \leq \ell(y_i f^i(x_i)).$$

Then, the result follows from the fact that ℓ is monotonic non increasing. $\Box$

We are now able to present the LOO bound, starting from the case of linear decoding. Below we denote by f_s the $s-$machine, $f_s(x) = \sum_{i=1}^n \alpha_i^s m_{y_i s} k^s(x_i, x)$, and let $G_{ij}^s = k^s(x_i, x_j)$.

Theorem 3.3.1 *Suppose the linear decoding function $d_L(\mathbf{m}_q, \mathbf{f}) = -\mathbf{m}_q \cdot \mathbf{f}$ is used, where $\cdot$ denotes the inner product. Then, the LOO error of the ECOC of kernel machines is bound by*

$$\frac{1}{n} \sum_{i=1}^n \theta \left(-g(x_i, y_i) + \max_{q \neq y_i} U_q(x_i) \right) \tag{3.42}$$

where we have defined the function

$$U_q(x_i) = (\mathbf{m}_q - \mathbf{m}_p) \cdot \mathbf{f}(x_i) + \sum_{s=1}^S m_{y_i s}(m_{y_i s} - m_{qs}) \alpha_i^s G_{ii}^s \tag{3.43}$$

with $p = \mathrm{argmax}_{q \neq y_i} \mathbf{m}_q \cdot \mathbf{f}(x_i)$.

Proof: Note first that the LOO error can be equally written as

$$\frac{1}{n}\sum_{i=1}^{n}\theta\left(-g^i(x_i,y_i)\right) = \frac{1}{n}\sum_{i=1}^{n}\theta\left(-g(x_i,y_i) + g(x_i,y_i) - g^i(x_i,y_i)\right).$$

Therefore we have to prove that

$$\max_{q\neq y_i} U_q(x_i) \geq g(x_i,y_i) - g^i(x_i,y_i).$$

By definition of the multiclass margin we have

$$g^i(x_i,y_i) = \mathbf{m}_{y_i}\cdot\mathbf{f}^i(x_i) - \mathbf{m}_{p^i}\cdot\mathbf{f}^i(x_i)$$

where we have defined

$$p^i = \text{argmax}_{q\neq y_i}\mathbf{m}_q\cdot\mathbf{f}^i(x_i).$$

By applying Lemma 3.3.1 simultaneously to each kernel machine used in the ECOC procedure, inequality (3.41) can be rewritten as

$$f_s^i(x_i) = f_s(x_i) - \lambda_s m_{y_i s}, \quad s\in\{1,\dots,S\} \tag{3.44}$$

where λ_s is a parameter in $[0,\alpha_i^s G_{ii}^s]$ [2]. Using the above equation we have:

$$\begin{aligned} g^i(x_i,y_i) &= \sum_{s=1}^{S}(m_{y_i s} - m_{p^i s})f^i(x_i) \\ &= \sum_{s=1}^{S}[(m_{y_i s} - m_{p^i s})f_s(x_i) - m_{y_i s}(m_{y_i s} - m_{p^i s})\lambda_s] \\ &\geq \sum_{s=1}^{S}[(m_{y_i s} - m_{p^i s})f_s(x_i) - m_{y_i s}(m_{y_i s} - m_{p^i s})\alpha_i^s G_{ii}^s]. \end{aligned}$$

Last inequality follows from the observation that $m_{y_i s}(m_{y_i s} - m_{p^i s})$ is always non-negative. From the same inequality, and applying the definition of margin for $g(x_i,y_i)$, we have

$$g(x_i,y_i) - g^i(x_i,y_i) \leq \sum_{s=1}^{S}[(m_{p^i s} - m_{ps})f_s(x_i) + m_{y_i s}(m_{y_i s} - m_{p^i s})\alpha_i^s G_{ii}^s].$$

Finally, as we cannot obtain p^i without explicitly training f^i, we choose it to be the one maximizing the right hand side of the inequality, thus proving the result. □

[2]Note that for (x_i,y_i) and s such that $m_{y_i s}=0$ it simply holds $f_s^i(x_i)=f_s(x_i)$, as (x_i,y_i) is not used as a training example.

The theorem says that point x_i is counted as a leave-one-out error when its multiclass margin is smaller than $\max_{q \neq y_i} U_q(x_i)$. This function is always larger or equal than the positive value

$$\sum_{s=1}^{S} m_{y_i s}(m_{y_i s} - m_{ps})\alpha_i^s G_{ii}^s$$

Roughly speaking, this value is controlled by two factors: the parameters α_i^s, $s = 1, \ldots, S$ (where each parameter indicates if point x_i is a support vector for the $s-$th kernel machine) and the Hamming distance between the correct codeword, $\mathbf{m}_{y_i}$, and the closest codeword to it, $\mathbf{m}_p$. Moreover, the final part of the theorem proof shows that such value is actually an upper bound on $g(x_i, y_i) - g^i(x_i, y_i)$, if we assume that the incorrect codeword nearest to $\mathbf{f}^i(x_i)$ is the same as the one nearest to $\mathbf{f}(x_i)$ for all i. We could therefore derive a tighter approximated upper bound on the LOO error, given by

$$\frac{1}{n}\sum_{i=1}^{n} \theta\left(-g(x_i, y_i) + \sum_{s=1}^{S} m_{y_i s}(m_{y_i s} - m_{ps})\alpha_i^s G_{ii}^s\right) \tag{3.45}$$

which could occasionally slightly underestimate the LOO error, whenever the assumption is violated.

Theorem 4.1 also enlightens some interesting properties of the ECOC of kernel machines which we briefly summarized in the following.

- **Stability of the ECOC schemes**

  One-vs-all schemes are more stable than other ECOC schemes, meaning that their multiclass margin is less affected by removing one point in that case. In fact, note that in the one-vs-all scheme each pair of rows has only two different elements, so when one point is removed, the bound in Theorem 3.3.1 implies that the margin will not change of more than $2C$. For pairwise schemes, instead, the worst change is $(Q-1)C$. For dense codes the situation is even worse: the worst case is $(S-1)C$. This observation provides some insights on why the simple one-vs-all SVM works well in practice.

- **One-vs-all schemes**

  For one-vs-all schemes we easily see that the multiclass margin $g^i(x_i, y_i)$ can be rewritten as

$$g^i(x_i, y_i) \geq 2\left[(f_{y_i}(x_i) - \alpha_i^{y_i} G_{ii}^{y_i}) - \max_{q \neq y_i}(f_q(x_i) + \alpha_i^q G_{ii}^q)\right].$$

  This has a simple interpretation: when x_i is removed from the training set, its margin is bound by the margin obtained if the classifier of that point, f_{y_i}, was penalized by $\alpha_i^{y_i} G_{ii}^{y_i}$ while the remaining classifiers f_q, $q \neq y_i$, increased their margin of $\alpha_i^q G_{ii}^q$).

Theorem 3.3.1 can be extended to deal with other decoding functions provided that they are monotonic non-increasing. This is formalized in the next corollary.

Corollary 3.3.1 *Suppose the loss function L is monotonic non increasing. Then, the LOO error of the ECOC of kernel machines is bound by*

$$\frac{1}{n}\sum_{i=1}^{n}\theta\left[L\left(m_{y_i s}f_s(x_i)-\alpha_i^s G_{ii}^s m_{y_i s}^2\right)-\min_{q\neq y_i}\sum_{s=1}^{S}L\left(m_{qs}f_s(x_i)-\alpha_i^s G_{ii}^s m_{y_i s}m_{qs}\right)\right]. \quad (3.46)$$

Proof: Following the main argument in the proof of Theorem 3.3.1, the multiclass margin of point x_i when this is removed from the training set is defined as

$$g^i(x_i,y_i)=\sum_{s=1}^{S}L(m_{p^i s}f_s^i(x_i))-\sum_{s=1}^{S}L(m_{y^i s}f_s^i(x_i)),$$

where

$$p^i=\text{argmin}_{q\neq y_i}L(\mathbf{m}_q,\mathbf{f}^i(x_i)).$$

By substituting equations (3.44) for each learning function f_s we obtain

$$g^i(x_i,y_i)=\sum_{s=1}^{S}L\left(m_{p^i s}f_s(x_i)-\lambda_s m_{y_i s}m_{p^i s}\right)-L\left(m_{y_i s}f_s(x_i)-\lambda_s m_{y_i s}^2\right)$$

with $\lambda_s\in[0,\alpha_i^s G_{ii}^s]$ for all s. If L is monotonic non increasing, the right hand side of the equation is minimized when all λ_s are at their right border. To see this, note that if $m_{p^i s}=m_{y^i s}$, the two loss functions get identical values, and the term is zero regardless of the value of λ_s. On the other hand, if $m_{p^i s}\neq m_{y^i s}$, the loss on the left hand side is minimized for maximal value of λ_s, while the loss on the right hand side has the opposite behaviour but contributes with a minus sign to the sum. The multiclass margin is thus bound by

$$g^i(x_i,y_i)\geq\sum_{s=1}^{S}L\left(m_{p^i s}f_s(x_i)-\alpha_i^s G_{ii}^s m_{y_i s}m_{p^i s}\right)-L\left(m_{y_i s}f_s(x_i)-\alpha_i^s G_{ii}^s m_{y_i s}^2\right)$$

We finally approximate the unknown value p^i with the one that minimizes the l.h.s. of the inequality, giving

$$g^i(x_i,y_i)\geq\min_{q\neq y_i}\sum_{s=1}^{S}L\left(m_{qs}f_s(x_i)-\alpha_i^s G_{ii}^s m_{y_i s}m_{qs}\right)-L\left(m_{y_i s}f_s(x_i)-\alpha_i^s G_{ii}^s m_{y_i s}^2\right).$$

which plugged into the definition of the LOO error (3.40) gives the result. □

Note that the corollary applies to all decoding functions used in this chapter. Moreover, as for the linear decoding case, we can approximate the LOO bound, by assuming that for any point x_i, the incorrect codeword nearest to $\mathbf{f}(x_i)$ does not change when removing the point from the training set, giving

$$\frac{1}{n}\sum_{i=1}^{n}\theta\left[L\left(m_{y_i s}f_s(x_i)-\alpha_i^s G_{ii}^s m_{y_i s}^2\right)-\sum_{s=1}^{S}L\left(m_{ps}f_s(x_i)-\alpha_i^s G_{ii}^s m_{y_i s}m_{ps}\right)\right], \tag{3.47}$$

where

$$p=\mathrm{argmin}_{q\neq y_i}L(\mathbf{m}_q,\mathbf{f}(x_i)). \tag{3.48}$$

3.4 Experiments

The proposed methods are validated on ten data sets from the UCI repository [BM98]. Their characteristics are shortly summarized in Table 3.1. Continuous attributes were linearly normalized between zero and one, while categorical attributes where "one-hot" encoded, i.e. if there are D categories, the $d-$th category is represented by a $D-$dimensional binary vector having the $d-$th coordinate equal to 1 and all remaining coordinates equal to zero.

Name	Classes	Train	Test	Inputs
Anneal	5	898	-	38
Ecoli	8	336	-	7
Glass	6	214	-	9
Letter	26	15000	5000	16
Optdigits	10	3823	1797	64
Pendigits	10	7494	3498	16
Satimage	6	4435	2000	36
Segment	7	1540	770	19
Soybean	19	683	-	35
Yeast	10	1484	-	8

Table 3.1. Characteristics of the Data Sets used

3.4.1 Comparison between Different Decoding Functions

We trained ECOC using SVM as the base binary classifier[3] with a fixed value for the regularization parameter given by the inverse of the training set average of $k(x,x)$. In our experiments we compared our decoding strategy to Hamming and other common loss-based

[3]Our experiments were carried out using SVM^{Light} [Joa98a].

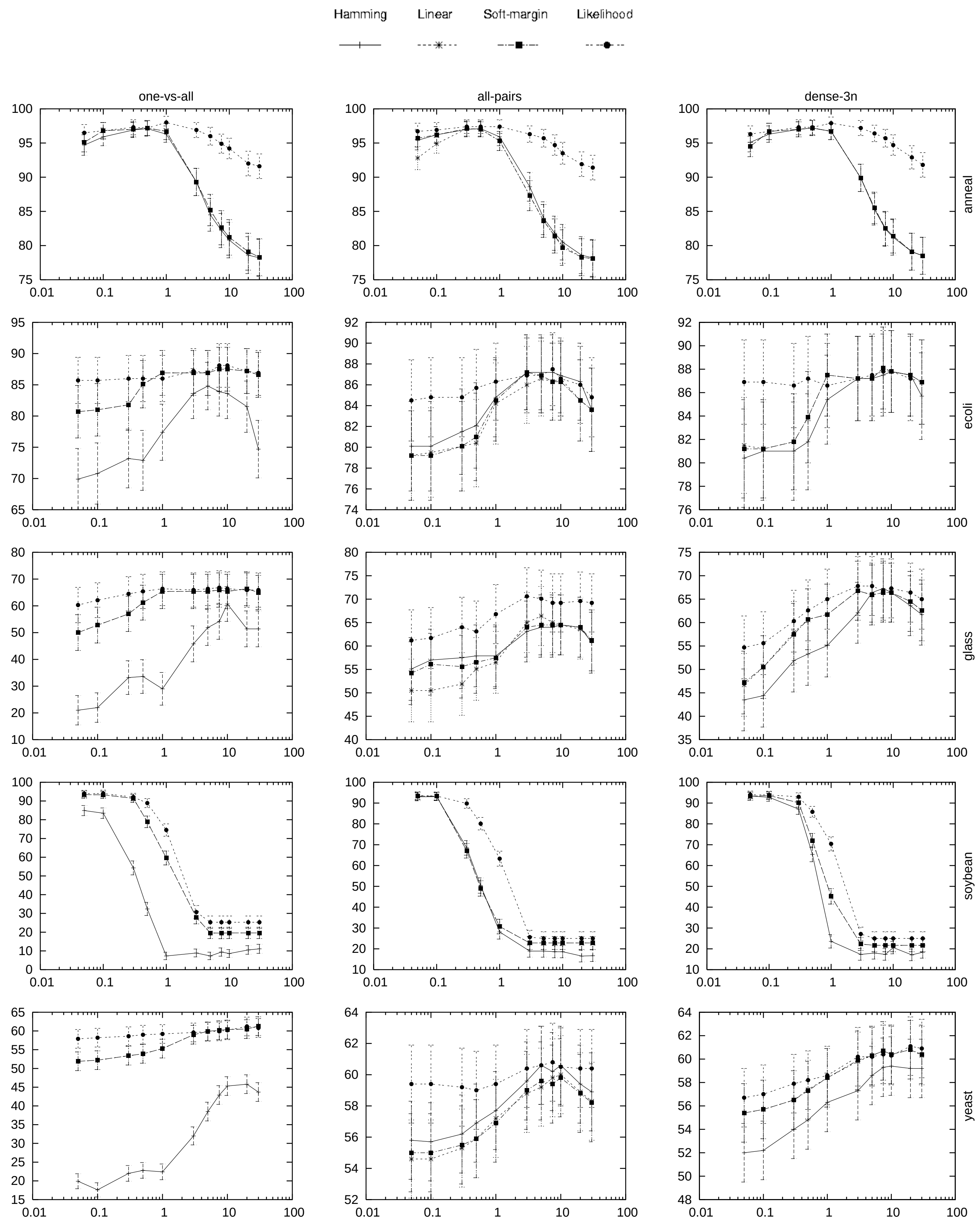


Figure 3.4. Test accuracy plotted against kernel hyperparameter γ. Data sets anneal, ecoli, glass, soybean, yeast.

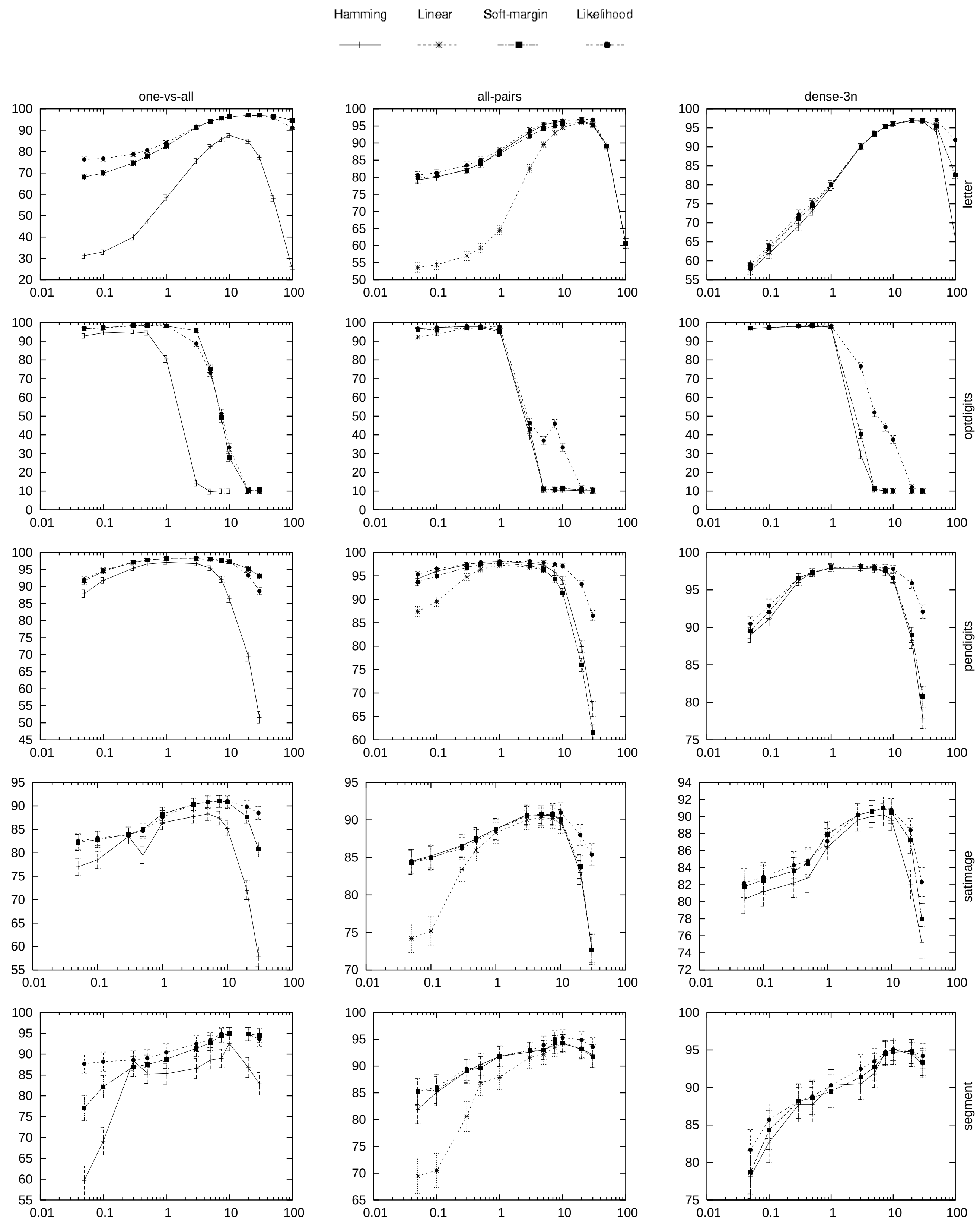


Figure 3.5. Test accuracy plotted against kernel hyperparameter γ. Data sets letter, optdigits, pendigits, satimage, segment.

decoding schemes (linear, and the soft-margin loss used to train SVM) for three different types of ECOC schemes: one-vs-all, all-pairs, and dense matrices consisting of $3Q$ columns of $\{-1, 1\}$ entries[4]. SVM were trained on a Gaussian kernel, $k(x, t) = \exp\{-\gamma\|x - t\|^2\}$. In order to avoid the possibility that a fortuitous choice of the parameter γ could affect our results we carried out an extensive series of experiments where we compared the test error of the four decoding schemes considered for 11 different values of γ.

Results are summarized in figures 3.4 and 3.5. For data sets with less than 2,000 instances (figure 3.4) we estimated prediction accuracy by a twenty-fold cross-validation procedure. For the larger data sets (figure 3.5) we used the original split defined in the UCI repository except in the case of letter, where we used the split of 15000–5000. All accuracies are reported together to error bars for 95% confidence intervals.

Our likelihood decoding works better for all ECOC schemes and for most values of γ, and is often less sensitive to the choice of the kernel hyperparameter.

Another interesting observation is that the Hamming distance works well in the case of pairwise classification, while it performs poorly with one-vs-all classifiers. Both results are not surprising: the Hamming distance corresponds to the majority vote, which is known to work well for pairwise classifiers [Fri96] but does not make much sense for one-vs-all because in this case ties may occur often.

3.4.2 Comparison between Different Multiclass Methods

We ran a series of experiments in order to compare the performances of different multiclass methods, choosing a subset of the datasets in table 3.1, namely anneal, ecoli, optdigits, satimage, segment and yeast. We compared the ECOC schemes discussed in the previous section (one-vs-all, all-pairs and dense codes) using our probabilistic decoding function and the two types of direct multiclass methods discussed in section 3.2.2: the one by Vapnik [Vap98], in the formulation by Crammer and Singer [CS02a] (indicated as "C&S"), and the one independently developed in [WW98, BB99, GEPM00] (indicated as "msvm"). We employed SVM^{Light} [Joa98a] for the binary SVM in ECOC schemes, and `bsvm`[5] [HL02a] for the two direct multiclass methods.

In this setting, the regularization parameter C cannot be kept fixed as in the previous section, as it has a different effect for different methods. Therefore, for each multiclass method independently and for each value of γ, we employed a validation procedure in order to choose the best regularization parameter in a uniformly distributed set of values ranging from 0.00001 to 1000. For datasets with more than 2,000 instances, we divided the training set in 2/3 for train and 1/3 for validation, and chose the regularization parameter which gave the best results on the validation set, retraining the algorithm on the full training set for such value. The test set was then employed to compute accuracy as in the previous section. For datasets with less than 2,000 instances, we run a 20-fold cross

[4]Dense matrices were generated using a variant of the BCH algorithm [BRC60] realized by T. G. Dietterich [DB95].

[5]`http://www.csie.ntu.edu.tw/~cjlin/bsvm`

validation procedure for each value of the regularization parameter, and reported the best cross-validation accuracy obtained. Results are shown in figure 3.6, with γ values versus accuracy and error bars for 95% confidence intervals. Note that for ECOC schemes we employed the same γ for all binary classifiers, while a more accurate optimization would require to search the best value for each of them, resulting in a much more complex hyperparameter tuning problem.

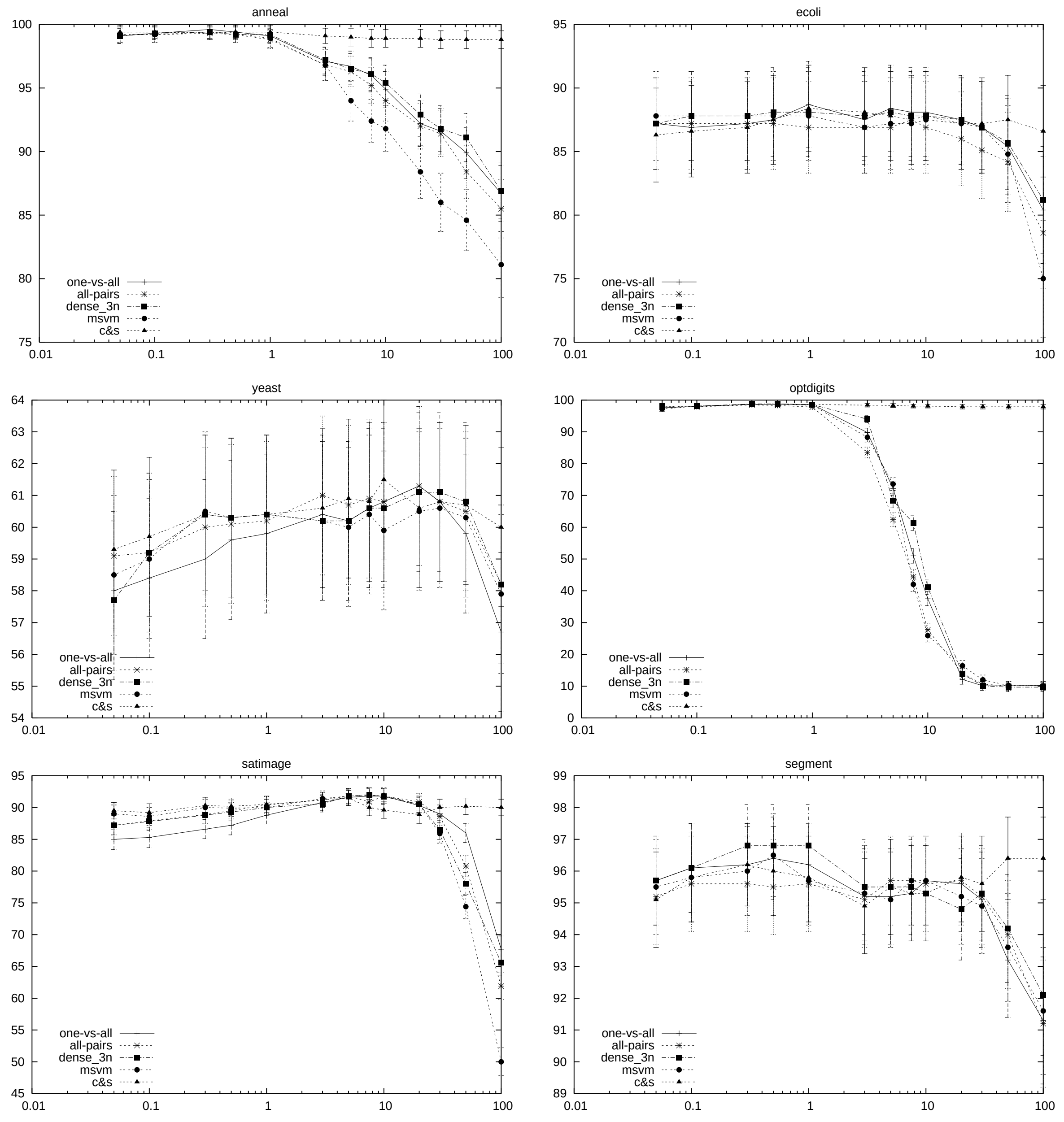


Figure 3.6. Test accuracy plotted against kernel hyperparameter γ for different multiclass methods and datasets. Regularization parameter C is optimized for each value of γ and for each method independently by a validation procedure. All ECOC schemes employ the likelihood decoding function.

At the optimum, all methods perform equally well, a result also noted in other works [RR01, HL02a]. However, for different values of γ, the methods have a different sensibility to overfitting. Between different ECOC, no significant difference can be observed, except for an overall tendency of dense codes for performing at least as well as the better of the other two ECOC. The most significant result of the experiments however, is the performance of the "C&S" method for very high values of the γ parameter, corresponding to small values of the Gaussian variance, a condition which produces a severe overfitting with possible degenerate solutions in all other methods. Conversely, the solution discovered by the "C&S" method achieves results similar to the ones obtained for the best value of the parameter. A deeper investigation of the models learned by the method for high values of γ provided some insights on the reason for this behaviour. The "C&S" method actually outputs a degenerate model in such situations, where all training examples are support vectors, and all alphas assume the same value given by C/Q. Thus the decision function simply reduces to

$$f(x) = \mathrm{argmax}_{q\in[1,Q]} \left((Q-1) \sum_{i:y_i=q} k(x_i,x) - \sum_{i:y_i\neq q} k(x_i,x) \right).$$

Each support vector thus contributes to the confidence margin for a given class simply by its similarity with the example to classify (as measured by the kernel value $k(x_i, x)$), where this contribution is positive if the SV belongs to the class under investigation, and negative otherwise. The multiplicative term for positive contributions $(Q-1)$ tends to re-balance the number of positive and negative contributions, assuming a uniform distribution of support vectors between the classes.

Note that the "msvm" method actually achieves the same degenerate solution, but in the `bsvm` implementation [HL02a] the bias term is included in the kernel, producing the following decision function

$$f(x) = \mathrm{argmax}_{q\in[1,Q]} \left((Q-1) \sum_{i:y_i=q} [k(x_i,x)+1] - \sum_{i:y_i\neq q} [k(x_i,x)+1] \right).$$

In the above equation, $k(x_i, x)$ is neglectable compared to 1 for very small values of the Gaussian variance, thus the decision function always outputs a default class independently of the input pattern. By eliminating the bias term from the kernel, we obtain the same results as for the "C&S" method.

3.4.3 Hyperparameter Tuning

We now show experiments where we use the bound presented in section 3.3 to select optimal kernel parameters. We focused on the datasets with more than 2,000 instances, and searched for the best value of the γ hyperparameter of the Gaussian kernel. To simplify the problem we searched for a common value for all binary classifiers among a set of possible values, as discussed in the previous section.

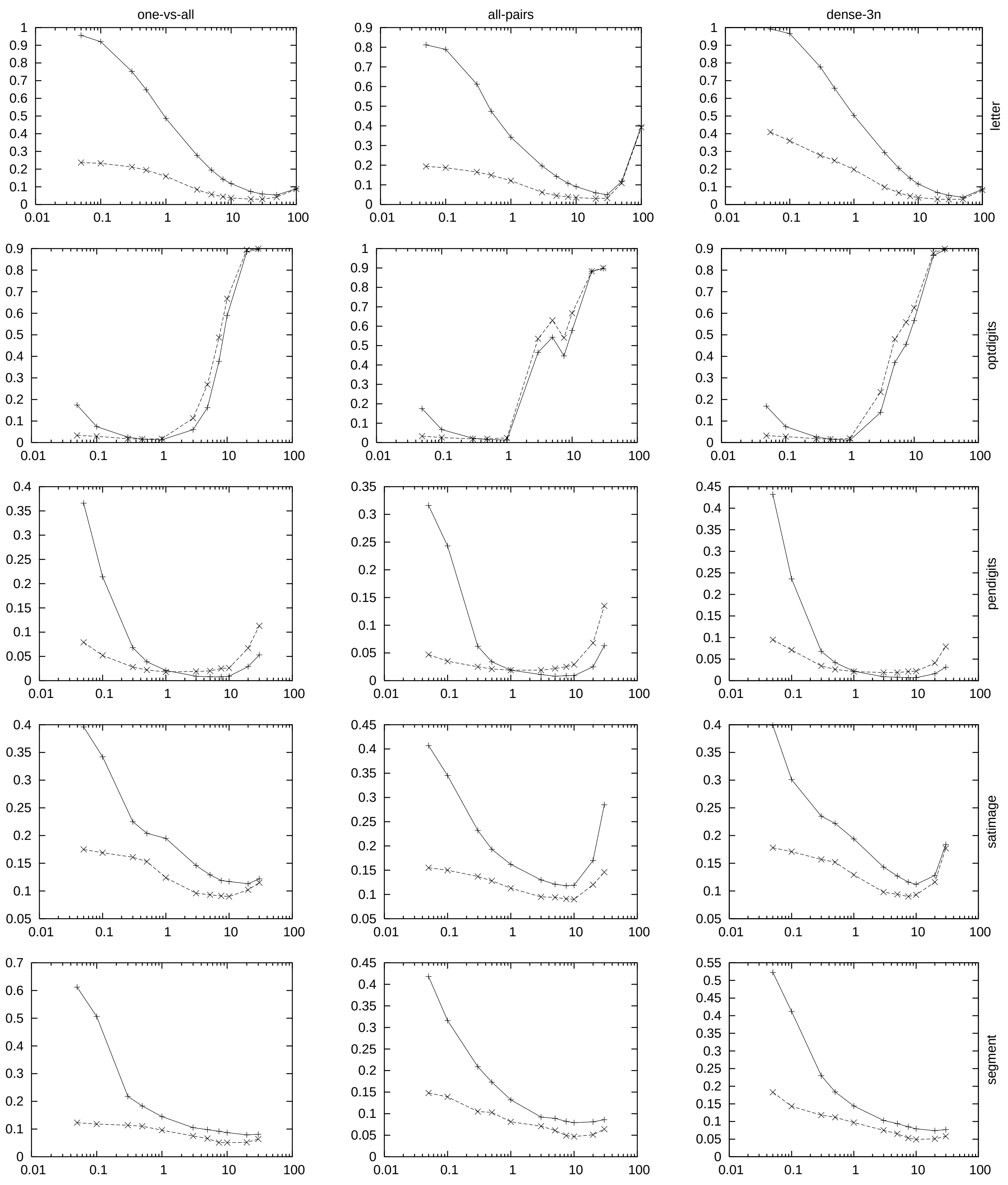


Figure 3.7. Empirical comparison between test error (dashed line) and the leave-one-out (solid line) bound of Corollary 4.1. The likelihood decoding function is used in all the experiments.

Plots in figure 3.7 show the test error and our LOO estimate for different values of γ for the three ECOC schemes one-vs-all, all-pairs and dense codes. Note that the minimum of the LOO estimate is very close to the minimum of the test error, although we often observed a slight bias towards smaller values of the variance. Note also that the test error is itself an approximation of the LOO error, computed on the test set of each dataset. This explains why the bound sometimes underestimates the test error, as can be seen in the graphs for the pendigits dataset (second row in fig. 3.7). Finally, we observed that in all experiments the approximate version of the bound (see eq. (3.47)) was always identical to the true bound (eq. (3.46)). Thus, the assumption that the incorrect codeword nearest to the predicted output for a point does not change when removing the point from the training set was never violated.

3.5 Conclusions

Multiclass categorization problems have been typically addressed in two complementary ways, either by reducing the multiclass problem in several binary subproblems, or by directly extending binary algorithms to the multiclass case when such extension is possible. Concerning the first approach, we proposed a novel decoding function to combine output of the binary classifiers in the general framework of Error Correcting Output Codes. Such function outputs the conditional probability of a codeword given the outputs of the binary classifiers. Our experiments, using SVM as the underlying binary classifiers, showed that the function actually helps recalibrating the outputs of the classifiers, thus improving the overall multiclass classification accuracy in comparison to other loss-based decoding schemes. This is particularly evident for non-optimal values of kernel hyperparameters, making the corresponding algorithm less sensitive to their choice.

Individual conditional probabilities for each bit of the codeword where estimated using a simple logistic function, with slope and offset learned from examples. However, margin distributions do not necessarily have a symmetric behaviour around zero, especially for unbalanced classification tasks. Thus, a more complex function could be possibly employed to better approximate the margin distributions.

Concerning direct extensions of SVM to the multiclass case, we extensively reviewed most of the methods proposed so far, focusing on similarities and differences between them, and highlighted the connections between multiclass SVM and ECOC of kernel machines, basing on the works of Crammer and Singer [CS00] on continuous codes.

We conducted an extensive set of experiments aimed at comparing the different approaches to multiclass classification. While results agreed with previous findings [RR01, HL02a] in showing analogous performances for the different methods at their optimum, differences emerged for non-optimal values of kernel parameters. Between different ECOC schemes, no significant difference could be observed, except for an overall tendency of dense codes for performing at least as well as the better of the other two ECOC. The most significant finding of this set of experiments was the strong resistance to overfitting of the multiclass method proposed by Vapnik [Vap98] in the implementation by Crammer and

Singer [CS00]. For small values of the Gaussian variance, such method actually learns a degenerate solution, where all examples are support vectors and all alphas share the same value, but it still achieves results similar to those obtained for optimal values of γ. Same results were obtained with an unbiased version of the other direct multiclass method [WW98, BB99, GEPM00].

Finally, we derived a novel bound on the leave-one-out error of ECOC of kernel machines, which is valid for generic monotonic non increasing decoding functions. We run a set of experiments aimed at verifying the effectiveness of the bound for choosing kernel hyperparameters, focusing on the γ of Gaussian kernels. Bound curves and test error curves for different values of γ actually showed a similar behaviour, being minimized for very close values of the hyperparameter.

The leave-one-out bound we presented could be smoothed into a differentiable function, enabling the application to the optimization of several hyperparameters simultaneously. An interesting future study in this sense is to use the derived leave-one-out bound to perform feature selection. From a theoretical viewpoint it will be also interesting to study generalization error bounds of the ECOC of kernel machines. It should be possible to use our result within the framework of stability and generalization introduced in [BE02].

Part II

Cysteine Bonding State Prediction

Chapter 4

Protein Structure

Shortly after synthesis, proteins *fold* into a stable three-dimensional structure which is strongly tied to their function within an organism. Knowing the three-dimensional structure of a given protein is thus an important step in understanding its biological function. Genomic scale sequencing projects are dramatically widening the gap between available nucleotide sequences, annotated proteins, and proteins with known three dimensional structure, urging for automatic algorithms to fill it up. No simple rule, as those discovered for transcription and translation, is available to map sequences of residues to three dimensional coordinates. Experimental methods for structure determination, such as X-Ray crystallography [Dre94] and NMR spectroscopy [W8̈6], are highly expensive and time consuming, and cannot be applied in all situations. Developing methods for the prediction of a protein three dimensional structure given the sequence of its residues is thus one of the greatest challenges in computational molecular biology.

4.1 Overview

The basic units of proteins are amino acids, which are made of a main common part, and a side chain which characterizes each of the twenty different amino acids (see fig.4.1(a)). Amino acids are often grouped, according to the chemical nature of their side chain, into hydrophobic (A,V,L,I,F,P,M), charged (D,E,K,R) and polar (S,T,C,N,Q,H,Y,W). During protein synthesis, amino acids are joined by *peptide bonds*, where the carboxyl group of an amino condenses with the amino group of the next eliminating water (fig. 4.1(b)), and form polypeptide chains of so called residues.

Proteins are composed of one or more of these polypeptide chains. The sequence of

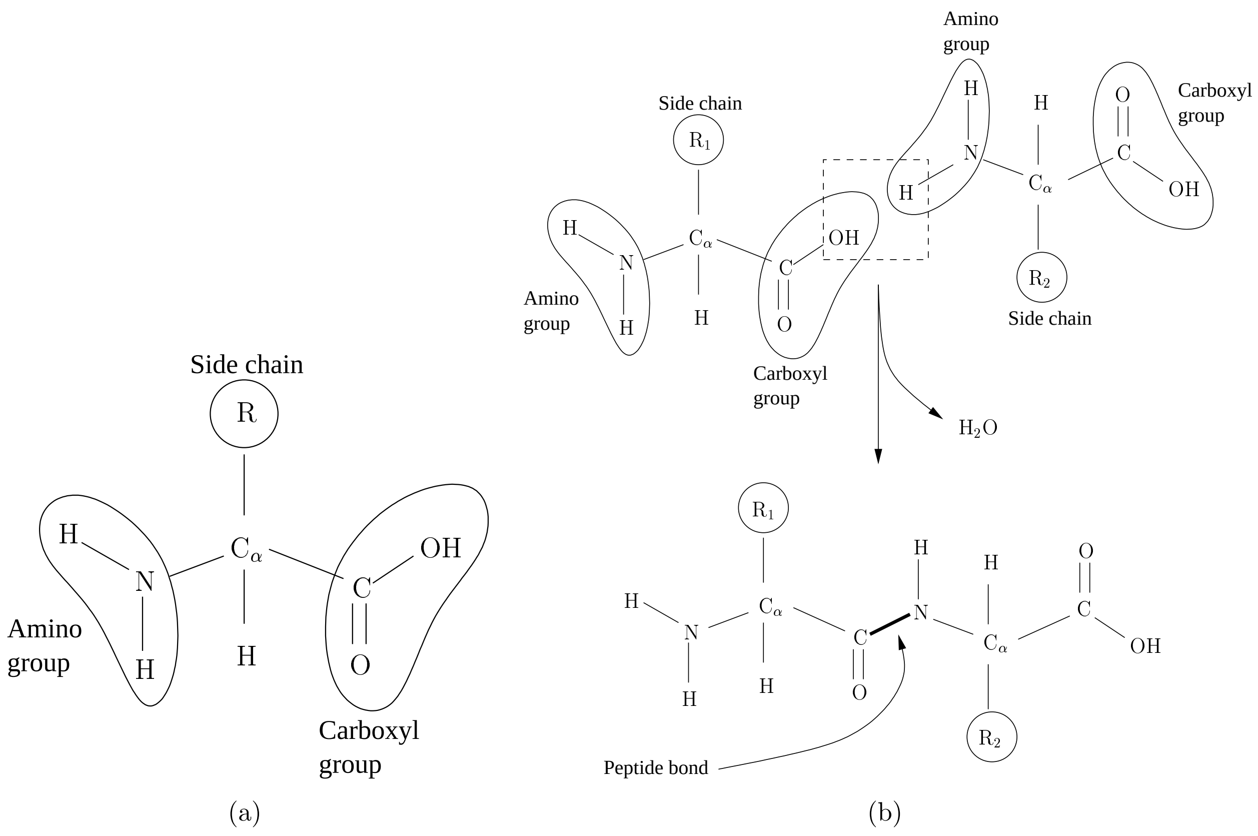
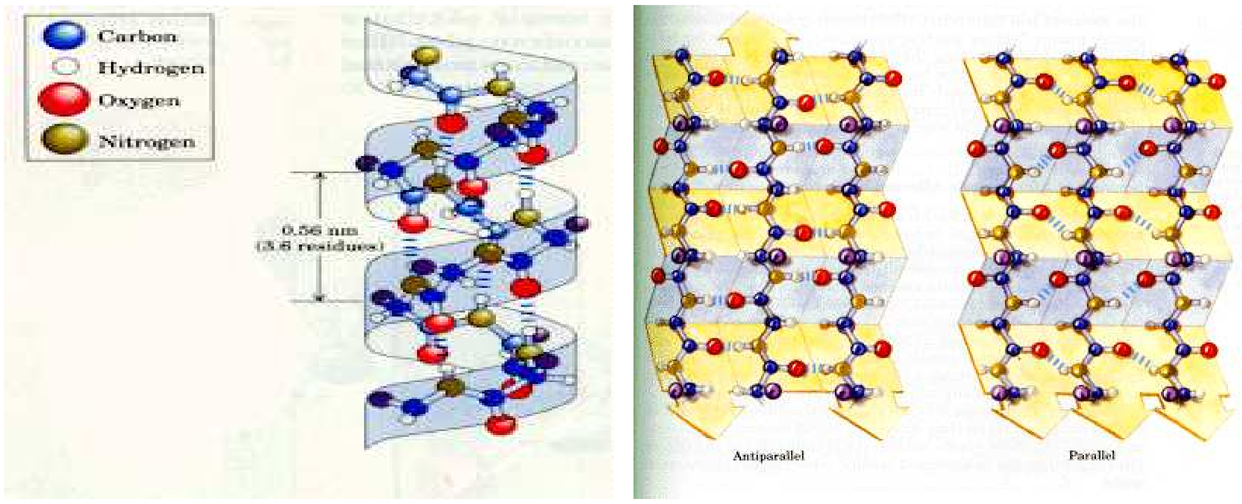


Figure 4.1. (a) All amino acids share a common part, containing a central carbon atom known as *carbon alpha* (C_α), bonded to a hydrogen atom, an amino group and a carboxyl group. The fourth valence of the C_α is involved in binding a side chain which is peculiar to each amino acid. (b) Peptide bonds form by condensation of the carboxyl and amino group of two successive amino acids, eliminating a molecule of water.

amino acids forming a polypeptide chain is the protein *primary structure* (fig. 4.3(a)). The function of a given protein is determined by its three dimensional structure, which arises from the folding of its polypeptide chains (fig. 4.3). The experimental determination of the structure of a wide range of proteins (see section 4.2) allowed to discover common patterns in proteins structure. Local regularities involving segments of adjacent residues represent the *secondary structure* (fig. 4.3(b)) of a protein, and are determined by hydrogen bonds between `C'=O` parts of residues, which have a slight negative charge, and `NH` parts which are slightly positively charged.

Alpha helices (H) are sequences of residues with hydrogen bonds between `C'=O` of residue n and `NH` of residue $n+4$, forming helices with 3.6 residues per turn (see fig. 4.2(a)). A *beta strand* is a sequence of residues which form hydrogen bonds with the residues of another aligned beta strand, giving a planar conformation called *beta sheet* (see fig. 4.2(b)). Beta sheets are made of two or more aligned beta strands, and can be *parallel* if the strands share the same direction (i.e. from the `C`-terminus to the `N`-terminus), or *antiparallel* if they are made of pairs of strands with alternate direction (see fig. 4.2(b)). Mixed beta



(a) Alpha helix (b) Beta sheet

Figure 4.2. Regular elements of secondary structure. (a) An *alpha helix* is a sequence of residues with hydrogen bonds between `C'=O` of residue n and `NH` of residue $n+4$, forming an helix of 3.6 residues per turn. (b) A *beta sheet* is a planar conformation made by two or more aligned *beta strands* connected by hydrogen bonds. Beta sheets can be either *parallel* or *antiparallel* depending on the direction of the aligned strands. Mixed sheets containing both parallel and antiparallel strands also exist but are far less common. Figures are taken from `http://www.agsci.ubc.ca/courses/fnh/301/protein/protprin.htm`.

sheets containing both parallel and antiparallel strands also exist but are far less common. Alpha helices and beta sheets usually form the hydrofobic core of the protein structure, and are connected by irregular *loop regions* called *coil* (C), that primarily reside on the surface of the protein.

The protein *tertiary structure* is the way a single chain folds into its three-dimensional structure. Such structure is usually composed of common combinations of elements of secondary structure, such as alpha or beta *hairpins*, called *motifs*, and more complex arrangements of motifs into *domains*, which also represent functional units within the chain. In order to offer a comprehensive view of a chain in terms of its secondary structure elements, schematic pictures are often employed, where alpha helices are represented as cylinders or helices, beta sheets as arrows oriented from the amino to the carboxyl end, and the remaining coil as ribbons (see fig. 4.3(c)). *Multimeric* proteins composed of several polypeptide chains also have a *quaternary structure* given by the overall arrangement of the chains (see fig. 4.3(d)). A different constraint on protein structure is given by a particular amino acid, the *cysteine* (C), which can be oxidized to form a bond with another cysteine residue, called a *disulphide bridge*. Such bonds form in oxidative environments, and are thus common in extracellular proteins, and help stabilizing the protein structure. We will treat cysteines and disulphide bridges in detail in chapter 5. For a comprehensive treatment of protein structure see [BT99] and [Les01].

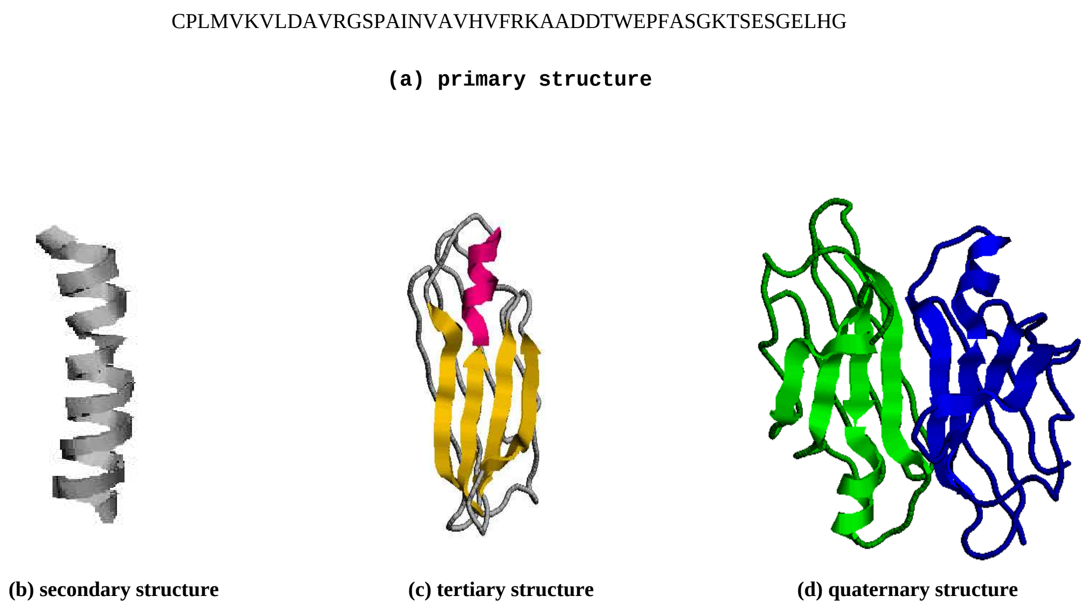


Figure 4.3. The *primary structure* (a) of a protein is the sequence of its residues. Local regularities such as alpha helices and beta strands form the protein *secondary structure* (b). The combination of such elements within a given polypeptide chain determines its *tertiary structure* (c). Multimeric proteins made of several chains also have a *quaternary structure* (d) given by the overall arrangement of the chains.

4.2 Protein Structure Determination

Experimental protein structure determination has been a major step in structural genomics. The first and most widely employed experimental procedure is X-Ray crystallography, pioneered by W.L Bragg, and led to the structural resolution of the first globular protein in 1958. Nuclear magnetic resonance (NMR) spectroscopy, studied by K. Wüthrich since the Sixties, led to the first de novo globular protein structure determination in 1984, and is assuming increasing importance as a resolution method, even if it is still limited to proteins of around 300 residues. The Protein Data Bank [BBB$^+$02], which collects all resolved structures, around the end of March 2004 contained 21,111 sequences resolved by X-Ray crystallography, and 3,674 sequences resolved by NMR spectroscopy [1]. In the following we give a brief overview of the two resolution methods, focusing on the reliability of their results and their limitations. Details on X-Ray crystallography can be found in [Dre94], and NMR spectroscopy is extensively treated in [Ẅ86].

[1]PDB Holdings List: 23-Mar-2004, current holdings list available at `http://www.rcsb.org/pdb/holdings.html`

4.2.1 X-Ray Crystallography

The first step to resolve a protein structure by X-Ray crystallography is that of growing a crystal from a pure and homogeneous sample of the protein to determine. Crystallization is a complex and very long step, which can take months in order to achieve a crystal of the necessary size and resolution. Moreover, certain proteins such as the integral membrane ones are very difficult to crystallize. Once the crystal is obtained, it is hit by a beam of x-rays, a part of which is diffracted by the electrons inside the crystal giving a set of diffraction spots. In order to determine which atom gives rise to each individual spot, different techniques are employed, either by comparing the resulting spots with those obtained with other crystals previously embedded with heavy metal complexes, or by employing polychromatic x-rays, which are partially absorbed by heavy atoms (still they have to be plugged into the protein if it lacks them, usually by substituting methionine with selenomethionine). The procedure results in an electron-density map, whose *resolution*, measured in Å, depends on the quality of the crystals. The polypeptide chain is thus fitted into the electron-density map with a trial-and-error process, aiming at minimizing the so called *R-factor*, that is the residual disagreement between the experimentally observed diffraction amplitudes, and those that would be obtained with an hypothetical crystal corresponding to the fitted model. At low resolution (5.0 Å), only the most evident features can be detected. At medium resolution (3.0 Å) it's still possible to make serious errors, especially in the irregular coil regions, which can be refined by a proper minimization of the R-factor (between 0.15 and 0.20 for well-determined structures). An unbiased estimate of the agreement between the model and the experimental data, called *free R-factor*, can be calculated by comparing the model with a subset of the experimental data withheld during the refinement.

4.2.2 NMR Spectroscopy

The nuclei of atoms like $^{1}\mathtt{H}$, $^{13}\mathtt{C}$ and $^{15}\mathtt{N}$ have a magnetic momentum which can be measured by nuclear magnetic resonance. The protein is placed into a strong magnetic field, which forces the spins of the nuclei to align along it. The nuclei are then excited by applying radio frequency pulses, and when returning to the equilibrium condition, each nucleus emits a radiation whose frequency depends on the type of atom and its molecular environment. The values of these frequencies relative to a reference signal are called *chemical shifts*, and each of them has to be assigned to the corresponding nucleus. $^{1}\mathtt{H}$ nuclei, which are naturally abundant in proteins, have a fairly limited range of chemical shifts, and overlapping signals make the assignment task difficult. This problem have been addressed by Wüthrich with a strategy based upon homonuclear two dimensional experiments. *Correlation spectroscopy* (COSY) experiments give peaks among hydrogen atoms covalently connected through at most two other atoms, while *nuclear Overhauser effect* spectra give peaks of pairs of hydrogen atoms close in space, regardless of their position in the primary sequence. More recent extensions include three and four dimensional spectra, obtained by producing proteins in environments enriched with $^{13}\mathtt{C}$ and $^{15}\mathtt{N}$ isotopes. A *sequential assignment* is conducted using the available spectra, where each single assignment determines a set of constraints on the subsequent possible assignments. The result of the procedure is a set of *distance constraints*, which are processed to obtain a set of protein structure models which satisfy the constraints. Several algorithms have been developed to iterate this procedure and automate the whole process (see e.g. [HGW02]), but a lot of manual work is still usually necessary. Common measures of quality of an NMR experiment are the number of violated distance constraints, and the root mean square deviation (RMSD) of the set of derived models, which has been recently revisited in [SNB$^+$03]. The greatest current limitation of NMR spectroscopy is given by the size of the proteins that can be processed, which is of around 300 residues.

4.3 Protein Structure Prediction

The explosion of large scale genome sequencing projects allowed to obtain the entire genomes of several organisms in all three terrestrial kingdoms, included the first draft of the human genome [Con01, Gen01]. This led to an exponential growth of sequence databases such as GenBank [BKML$^+$04], widening the gap with the number of annotated sequences as recorded in SwissProt [BBA$^+$03], and that of the sequences with resolved three dimensional structure, stored in PDB [BBB$^+$02]. Automatic methods for protein structure prediction are thus becoming increasingly important in order to exploit the amount of biological information being accumulated. Predictive methods can be roughly divided in three groups, depending on the amount of similarity of the target protein to proteins with known structure.

4.3.1 Comparative Modeling

Comparative or *homology* modeling relies on the observation that structure is more conserved than sequence. That is, in order for a protein to maintain its biological function during evolution, various changes in its residues composition are possible, provided they don't imply a significant modification of its three dimensional structure. Levels of 25%-30% of pairwise sequence identity (percentage of identical residues between the aligned sequences) are sufficient to assure that the two proteins have similar folds [SS91], that is regular secondary structure elements are identical, while irregular loop regions may vary. Given a target protein, homology modeling thus essentially consists of finding out the best aligning protein with known 3D structure, if any with sufficient similarity exists, and correctly modeling irregular loop regions and residues side chains given the conserved structural core of the protein. The alignment problem is addressed by dynamic programming algorithms like the Smith-Waterman algorithm [SW81], or less accurate but faster heuristic algorithms such as BLAST [AGM$^+$90] and FASTA [Pea85]. These algorithms deal with the problem of allowing gaps in alignments by adding gap open penalties to the alignment score, and that of scoring residues similarity based on amino acid biochemical properties, employing substitution matrices such as the BLOSUM [HH92] ones. Loop regions modeling is usually done using databases of loop regions from proteins with known structure. Examples with similar residues length, and which connect the same type of secondary structure elements (α-α, β-β, α-β, β-α) as for the target region, are attached to the model, trying to obtain a structure similar to that of the homologous protein. Side chains conformation is predicted by free energy minimization over the set of naturally observed conformations, called *rotamers*. Even if loop region prediction can be far from accurate for decreasing levels of pairwise sequence identity, biologically important regions in proteins are usually well conserved, and therefore often accurately predicted by comparative modeling.

4.3.2 Fold Recognition

Comparative modeling algorithms are limited to the case when proteins with known structure homologous to the target protein can be found. However, most similar protein structures found in PDB are *remote homologous*, with pairwise sequence identity less than 25% [Ros97]. This *remote homology modeling* task is also addressed as *fold recognition*, as it amounts at identifying the most reasonable fold for a given protein having low sequence similarity to known proteins with such fold. The first *threading* techniques employed mean-force potentials [Sip95, SW92] to evaluate the fitness of the target sequence to a given known structure. More recent approaches [RSS97, KKB$^+$01, MJ03] employ a 1D prediction (see section (4.3.3.1)) of the target sequence, which is aligned to that of known structures.

Substantial improvements in the detection of remote homologues have been obtained by new generations of sequence based methods like PSI-BLAST [AMS$^+$97], and hidden Markov model (HMM) based models [DEKM98]. Starting from a BLAST alignment, PSI-Blast iteratively builds a position specific scoring matrix based on the currently aligned sequences, and repeats the search for homologues with the updated matrix. *Profile* and *motif* HMM are models especially conceived for representing multiple alignments of sequences. They allow probabilistic modeling of residues at each match in the alignment, and the presence of gaps either all along the alignment or between given motifs. For a review on profile and motif HMM see [Edd98].

The SCOP database [HMBC97] is a database of structural classification of proteins. It contains proteins structures hierarchically grouped into domains, families, superfamilies and folds. Such database can be used to assess performances of remote homology detection algorithms, as proteins into different families of the same superfamily are likely to be remote homologues. Therefore, a method able to detect a protein of a given family when trained on proteins of the other families of its superfamily is actually recognizing a remote homologue. This framework have been cast into a discriminative problem by Jaakkola et al. [JDH00], who paved the way for the use of kernel methods with excellent results. They employed state-of-art HMM methods [KBH98] to generate models of a given protein superfamily, and used them to train a Fisher kernel (see section 2.5.4.2) in order to discriminate between examples belonging to the given superfamily and examples belonging to all other superfamilies. A wide range of kernels have been developed to the scope after their work. The spectrum kernel [LEN02] (see section 5.2.2.3) compares two sequences by counting the occurrences of all common substrings of size k. Later variants included allowing mismatches in the common substrings [LEWN03] as well as deletions [LKE04]. Weight matrix [HH91] or regular expression [SSB03] motif databases derived from multiple alignments have been employed in [LMS$^+$01, BHB03]: a sequence is mapped to the feature space of all the comparisons with a motif of the database, and the inner product is computed in such space. Most of these methods employ efficient data structures such as suffix trees [Ukk95] or tries in order to be computationally feasible. A natural way to represent an object belonging to a given set is by its similarity to other elements of the set. This idea is implemented in the kernel framework by the *empirical* feature map [Tsu99], where each example is mapped to the vector of the similarities with all other reference examples. Liao and Noble [LN03] employed pairwise sequence similarity scores obtained by the Smith-Waterman algorithm [SW81]. The feature map for a sequence is thus represented by a vector of pairwise sequence similarities with positive examples (from the superfamily to be modeled) and negative examples (from the other superfamilies), and the size of this vectorization set heavily affects the efficiency of the algorithm. For a detailed treatment of kernel methods for remote protein homology, as well as for other tasks in computational biology, see [Nob04].

4.3.3 *De Novo* Protein Structure Prediction

When no remote homologue for a target protein can be identified, and no existing fold is found to be compatible with the protein sequence, similarity methods fail and a de novo prediction has to be made. The experiments conducted by Anfinsen forty years ago [AHSW61, Anf73] led him to formulate the so-called *thermodynamic hypothesis*, which states that a protein three dimensional structure is determined solely by its residue sequence and the environmental conditions. While it's now known that particular proteins (the so-called chaperones) are often involved in the correct folding of proteins, it's still generally assumed that the protein final folding is at a free energy minimum. However, finding such a minimum is computationally infeasible, and molecular dynamics methods with full atom representations are thus of no practical utility. Considered the difficulty of the prediction task, the problem is usually divided into smaller subproblems, whose approximate solutions give inter-atomic distance constraints that can be exploited by distance geometry algorithms. A potential architecture of such a method is shown in figure 4.4. In the following we give an overview of the proposed methods for specific 1D and 2D prediction tasks, and for the ultimate 3D prediction. For a general review of protein structure prediction see [Ros98].

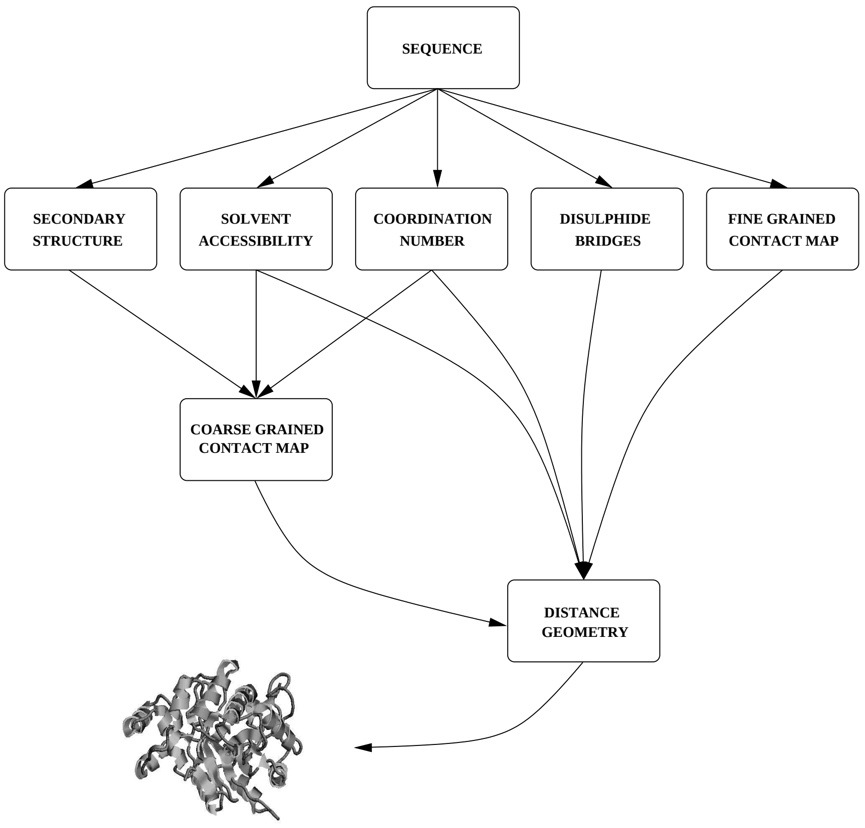


Figure 4.4. Potential architecture for protein structure prediction by combination of predictions for different subproblems.

4.3.3.1 Predictions in 1D

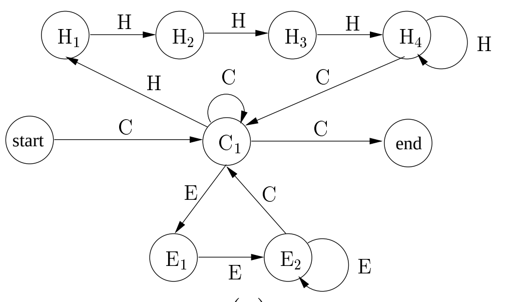
Protein structure prediction in 1D involves tasks like secondary structure and solvent accessibility prediction. Such predictions are often employed in threading methods for fold recognition as described in section 4.3.2, making the distinction between prediction by similarity and de novo approaches less clear. Moreover, they can give insights into the protein folding family [HMBC97, OMJ$^+$97] and consequently its biological function. Finally, 1D predictions can be employed for 2D prediction tasks such as coarse contact maps (see fig. 4.4), as described in section 4.3.3.2.

Secondary Structure Secondary structure (SS) prediction amounts at predicting for each residue of a target protein its secondary structure category. Training data can be extracted from experimentally resolved protein structures, using programs like DSSP [KS83a]. DSSP assigns each residue to one of eight classes, which are typically collapsed into three standard classes associated with helices, beta strands and coils. Different associations lead to tasks of different complexity [CB99].

The first machine learning algorithm for SS prediction [QS88] employed a multi-layer perceptron (MLP) taking as input a window of residues centered around the residue to be predicted, with one-hot encoding for residues. The first major improvement [RS93] resulted from using *multiple alignment profiles*. The multiple alignment is obtained aligning the target sequence with other sequences from a large database of proteins, using programs such as the Smith-Waterman [SW81] algorithm or PSI-BLAST [AMS$^+$97]. For each position in the sequence, the profile is then computed as the frequency of each residue in the corresponding column of the alignment, and replaces the one-hot encoding as input to the learning algorithm. Multiple alignment profiles allow to plug evolutionary information into the input, and have become a standard ingredient of most algorithms for SS prediction, as well as for other prediction tasks such as coordination number (see next paragraph) or disulphide bridge prediction (see chapter 5). For a study of the effect of increasing powerful alignment algorithms (such as PSI-BLAST with respect to BLAST) on prediction performances see [PR02]. In order to obtain stronger correlations between SS predictions of close residues within the sequence, a filtering stage has been often added [RS93, Jon99], taking as input the predictions of the algorithm for a window of residues, and refining them in a structure to structure mapping. A different approach was proposed in [BBF$^+$99] and refined in [PPRB02], employing bidirectional recurrent neural networks (BRNN) to handle non-causal dependencies within the sequence, and developing ensembles of networks in order to increase performances.

Kernel machines have entered quite recently [HS01] the arena of secondary structure prediction, and the necessary multiclass extension was either realized with combinations of binary classifiers [HS01, WMBJ03, CFPV03a] or multiclass support vector machines [NR03b, GPE$^+$04] (see section 3.2.2). No clear evidence emerged in favour of SVM compared to NN, taking into account the much higher computational time required. This can be partially explained by the fact the both are *local* classifiers fed with the same inputs, and the great amount of data available make the usual SVM advantages less evident. However, a few recent works [NR03b, GPE$^+$04] underlined the effectiveness of multiclass SVM as a filtering stage, to be fed with the output of other predictive methods (which can be MSVM themselves). A simple and effective refinement stage was proposed in [CFPV03a] in order to remove *inconsistencies* in the predicted sequences, that is violations of the constraints that can be imposed on consecutive secondary structure labels (see fig. 4.5). The method builds a finite state automata (FSA) representing all allowed sequences, and turns a predicted labelling sequence into the maximum likelihood sequence given the grammar and the predictions.

All the SVM approaches proposed so far employ standard kernels, such as polynomial or Gaussian ones. The first attempt to develop a kernel especially modeled for secondary structure prediction is presented in [GLV04]. Firstly, it employs a dot product between residues (or profiles) mediated by a substitution matrix, which compares residues according to their biochemical similarity. The substitution matrix was derived from [LRG86] and is especially designed for secondary structure prediction tasks (a similar approach will be discussed in section 6.3 for binding site prediction). Secondly, it introduces an adaptive



(a)

Target	..CEEEEECHHHHC..	..CHHHHHC
Predicted	..CE*H*EEE*H*H*C*HHC..	..CH*CC*HH*H*
Corrected	..CEEEEECHHHHC..	..CHHHHHC

(b)

Figure 4.5. (a) Finite state automata correcting inconsistencies in secondary structure predictions. Constraints imposed by the automata are: alpha helices at least four residues long, beta strands at least two residues long, presence of a coil between a beta strand and an alpha helix (and viceversa) as well as at the beginning and end of a chain. (b) Example of errors corrected by the automata. Note that such FSA is designed for a particular mapping from eight to three classes of secondary structure.

weighting of the window around the target residue, with weights learned by a version of kernel target alignment [CSTEK02] extended to the multiclass case [Ver02]. The proposed kernel is proved superior to a standard MLP, but given its computational overhead, its usage as a module in big architectures, such as those currently employed for secondary structure prediction [Jon99, PPRB02, PLN+00], is not straightforward.

Coordination Number and Solvent Accessibility The coordination number of a residue inside a sequence represents the number of its contacts, computed as the number of residues falling within a sphere of a certain radius centered on the residue [FBL+95]. Different radii (measured in Å) give correlated but different pictures of the residue local environment [PBFC02].

Solvent accessibility measures the degree with which a given residue inside a sequence interacts with the solvent molecules. Such measure is usually given relative to the maximum exposed surface area obtainable for a particular amino acid type. Solvent accessibility is highly (inversely) correlated to coordination number, but the two measures show different distributions [FC00], and individual predictors have been often developed.

Both regression tasks are usually cast into a simpler binary classification problem, by thresholding on the average coordination number (for a given radius) and the relative solvent accessibility respectively. Common predictors for both tasks employ neural networks fed by a window of profiles (see i.e. [RB99, FC00, RS94]), but also Bayesian methods [TG96] and algorithms based on information theory [Mov01]. State-of-art performances were obtained with ensembles of BRNN [PBFC02] similar to those employed for secondary structure prediction (see previous paragraph).

4.3.3.2 Predictions in 2D

Predictions in 2D consist in predicting structural interactions between pairs of residues within a protein sequence. Common examples of 2D prediction tasks are contact maps (described below) and disulphide bridges, which will be discussed in detail in chapter 5

Contact Maps A distance map for a sequence of n residues is an $n \times n$ matrix of distances between pairs of residues. The distance between two residues can be computed as the distance between their C_α atoms [MD96], that between their C_β atoms [TCS96] or as the minimal distance between atoms in the backbone or side chain of the residues [FC99]. Distance maps are the result of NMR spectroscopy experiments (see sec. 4.2.2), and distance geometry algorithms are employed in order to compute the three-dimensional structure [Nil96].

A contact map is a matrix of 0,1 entries obtained from a distance matrix using a threshold on the distance values, and it represents the adjacency matrix of the contact graph at the given resolution. Distance geometry or stochastic optimization algorithms can be employed to reconstruct the 3D atom coordinates from even partial or corrupted contact maps [VKD97]. Recent methods for contact maps predictions include neural networks [FOVC01] and HMM combined with mining techniques [ZJB00], which are trained to predict if a given pair of residues in a sequence is actually in contact. Useful inputs for the task include multiple alignment profiles, 1D predictions such as secondary structure and solvent accessibility, and *correlated mutations* [ALBK87]. Correlated mutations are co-variations of residues in different positions of a given sequence, that can occur during evolution and can be detected by sequence alignments. The rationale behind their usage in contact maps prediction is that correlated mutations should involve pairs of residues which are close in space in the folded protein.

In order to exploit the information contained in the graph structure of contact maps, specific algorithms for discrete structured data [FGS98] have been recently proposed. A GIOHMM [PB02] is a generalization of IOHMM (or recurrent neural networks) which allows to propagate information from all four cardinal corners. It's employed to model the spatial context of the pair of residues under investigation. A bi-recursive neural network [VF03] is a generalization of recursive neural networks able to treat undirected graphs, such as those induced by contact maps. The network is trained to learn a scoring function for candidate maps, and used to guide a search in the space of possible contact maps. Given the computational overhead of this last method, it has been currently applied to a reduced 2D prediction task, namely *coarse contact map* prediction [VF03]. Coarse contact maps are maps between segments of secondary structure, representing their 3D spatial proximity. Such maps are by far smaller than fine grained contact maps, and more complex algorithms can thus be employed on them.

4.3.3.3 Predictions in 3D

Ab initio protein structure prediction by free energy minimization at atomic level is computationally not feasible, and several simplification techniques have been proposed to address the problem.

Firstly, reduced representations both of the protein to be modeled and the conformational space to be searched are employed. For example, side chains can be collapsed in either their centroid or their beta carbon [SKHB97], and a limited set of naturally observed conformations (the rotamers [JK94]) are searched. 1D and 2D predictions can be employed to obtain restraints on tertiary structure [OKR$^+$99], as well as sets of possible representations of sequence segments, inducing a reduced search space given by the combinations of such segments [SKHB97, BSB01].

Secondly, in order to assign a score to candidate representations of the protein structure, approximations of their free energy have to be computed. Molecular mechanics potentials are based on physical and chemical interactions [vG93], while mean force potentials [Sip95, SW92] are empirically derived from known structures. Energy functions combining both kinds of potentials have been also developed [ON97], providing an interesting research direction for the improvement of free energy function approximations.

Finally, different search methods have been developed in order to allow a broad and fast search of the conformational space. Search moves involving large perturbations of the candidate structure were implemented with Monte Carlo simulated annealing [SKHB97] as well as genetic algorithms [PM97]. Rosetta [SKHB97, BSB01] employs a simulated annealing procedure to assembly structures from fragments, thus moving between different local structures by inserting different fragments.

For a review of current ab initio methods and their prospects in protein structure prediction see [SSB01, CRBB03].

Chapter 5

Disulphide Bonding State Prediction

Cysteine is the most reactive of all natural amino-acids, and it performs a wide variety of roles in biological systems, mostly related to its capacity to covalently bind another cysteine residue forming a *disulphide bridge*. Recent researches enlightened the role of this residue in determining both prokaryotes and eukaryotes response to oxidative stress (see [LJ03] for a review), which is a major factor of ageing [FH00] as well as of various diseases including cancer [KJ01]. However, the best-known property of cysteines is that of stabilizing protein structure [WWNS00, Pai00, Bet93, ea89] by forming intra or inter chain disulphide bridges which lower the free energy of folded proteins. Therefore, by correctly predicting the pattern of disulphide bridges in a protein with unknown structure, strong constraints over its three dimensional structure are obtained, which can significantly help the overall folding prediction. Disulphide bridges prediction can be divided in two steps. Firstly, the bonding state of each cysteine in a given sequence is predicted, as either *reduced* or *oxidized*, the latter meaning that it is involved in a disulphide bridge. Secondly, the connectivity pattern between oxidized cysteines is predicted, pairing each bonded cysteine with its correct partner. We will see in chapter 6 that some cysteines not forming disulphide bridges actually are non-free, and bind different ligands usually containing metals. While this particular situation will be addressed there, in the present chapter we are only interested in disulphide bonded cysteines, and will include ligand bonded cysteines within the class of free ones.

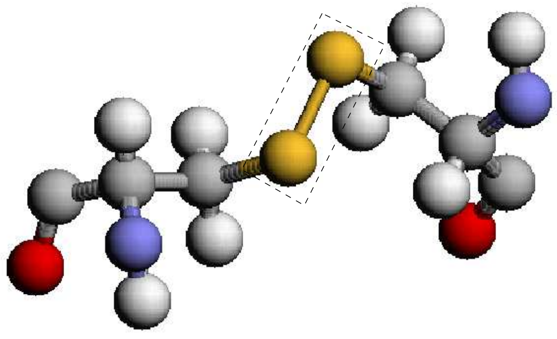


Figure 5.1. Ball-&-stick representation of a disulphide bridge: the bridge is drawn as a cylinder connecting the sulphur atoms of two oxidized cysteines.

5.1 Disulphide Bonds Formation

Disulphide bridges are covalent bonds formed between pairs of oxidized cysteines (see fig 5.1), according to the following hypothetical reaction scheme

$$2\mathrm{CH_2SH} + 1/2\,\mathrm{O_2} \rightleftharpoons \mathrm{CH_2SSCH_2} + \mathrm{H_2O} \tag{5.1}$$

Such reaction requires an oxidative environment, and Anfinsen and his group actually observed [AHSW61] in vitro disulphide bond formation in presence of oxygen. However, such reaction requires hours or days of incubation, while in vivo disulphide bridge formation takes seconds or minutes. More recent researches (see [FG03, RB98, KKB03, RB01] for reviews) have showed that such reactions occur within pathways involving enzymes acting as catalysts for oxidation, reduction and isomerization reactions (see fig. 5.2). Most of these enzymes share a conserved `CXXC` motif (two cysteines separated by two other residues) and a common three dimensional fold called *thioredoxin fold.* The proportion between the reduced and oxidized forms of such oxidoreductases in a particular location of the cell determines its environmental conditions.

Cytosolic proteins usually do not have disulphide bridges, suggesting that the cytosol should have a reducing environment. Two reducing systems have been actually detected in the cytosol of *E.coli* [RB01]: the thioredoxin system, composed of two thioredoxins with their thioredoxin reductase, and the glutaredoxin system, composed of three glutaredoxins, a glutathione and the glutathione reductase (see fig. 5.3). Thioredoxins reduce disulphide bridges in cytosolic proteins, and are in turn reduced by thioredoxin reductase, which gets the necessary electron from `NADPH` (see fig 5.3(a)). In a similar but slightly more complex fashion (see fig. 5.3(b)), glutathione attacks a protein disulphide bond forming a protein-`Glu` mixed disulphide, which is in turn reduced by glutaredoxins forming a `Grx-Glu` intermediate, subsequently reduced by a second molecule of glutathione. The resulting `Glu-Glu` complex is finally reduced by glutathione reductase. The ratio of reduced to oxidized glutathione in the cytosol is about 200 to 1.

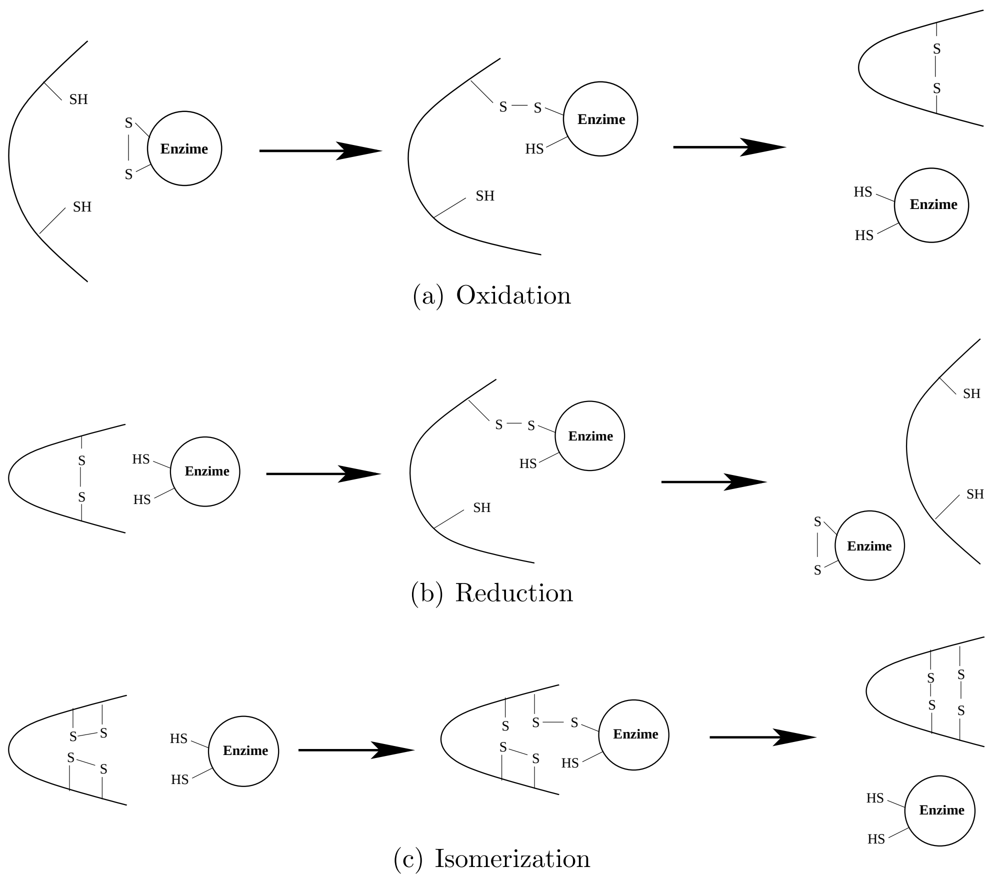
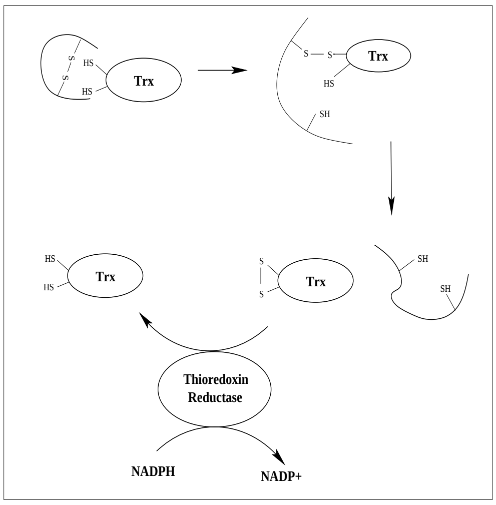
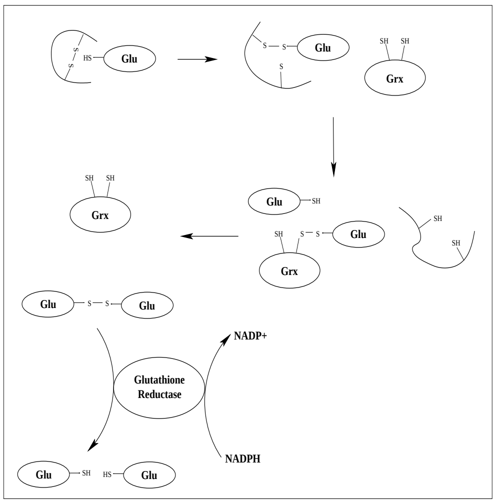


Figure 5.2. Schematic reactions between enzyme oxidoreductase and chain in oxidation (a), reduction (b) and isomerization (c), involving the formation of a temporary mixed enzyme-protein disulphide bond.



(a) Thioredoxin System



(b) Glutaredoxin System

Figure 5.3. Reducing systems in cytosol of *E.coli*: the thioredoxin system (a) employs thioredoxin enzymes (Trx) to reduce disulphide bonded cysteines in a chain, and thioredoxin reductase to reduce the resulting oxidized enzyme, taking an electron from `NADPH`. The glutaredoxin system (b) employs the molecule glutathione (Glu) to form a protein-`Glu` mixed disulphide, which is reduced by glutaredoxin enzymes (Grx). The resulting `Grx-Glu` intermediate is in turn reduced by a second molecule of glutathione, and the `Glu-Glu` complex is finally reduced by glutathione reductase, again taking an electron from `NADPH`.

Conversely to cytosol, disulphide bridge formation usually takes place in the periplasm of prokaryotes and in the endoplasmic reticulum (ER) of eukaryotes. While the process has been roughly determined in the former case for *E.coli* (see fig. 5.4), some missing steps have to be discovered in the latter (see fig. 5.5). Both processes involve protein disulphide bond formation catalysed by an oxidoreductase (DsbA in periplasm and PDI in ER) which is in turn re-oxidized by a recharging mechanism, which has not yet been entirely determined in the case of eukaryotes. Moreover, both processes contain a mechanism to correct misfolded proteins, either by rearranging the incorrect bridge (the so-called isomerization) or by reducing it, a step which again has not been completely explained in the ER case.

As a consequence of the environment dependent mechanisms of disulphide bridge formation, cytosolic and nuclear proteins usually do not contain disulphide bridges, while prokaryotic periplasmic proteins as well as eukaryotic proteins belonging to the secretory pathway (passing through the ER) are typically stabilized by disulphide bridges.

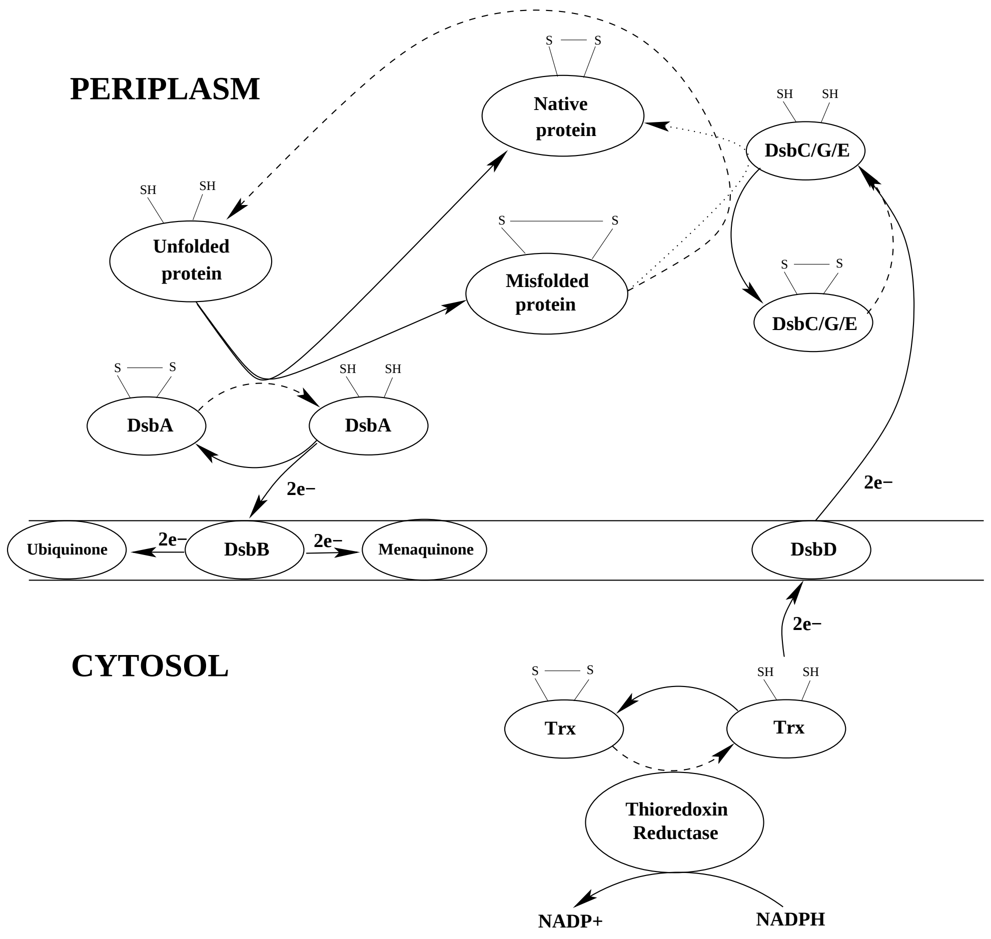


Figure 5.4. Model for disulphide bridge formation in the periplasm of *E.coli*[FG03]. A solid line indicates oxidation of the molecule originating the line, while a dashed line stands for reduction. The dotted line indicates an isomerization reaction, which does not involve a net change in redox state. Unfolded proteins are oxidized by the enzyme DsbA, which is in turn reoxidized by transferring electrons to dsbB within a respiratory chain. Misfolded proteins are either refolded by isomerization or unfolded by reduction, both tasks being accomplished by DsbC,DsbG and DsbE, which are in turn reduced by the thioredoxin-system dependent membrane reductase DsbD.

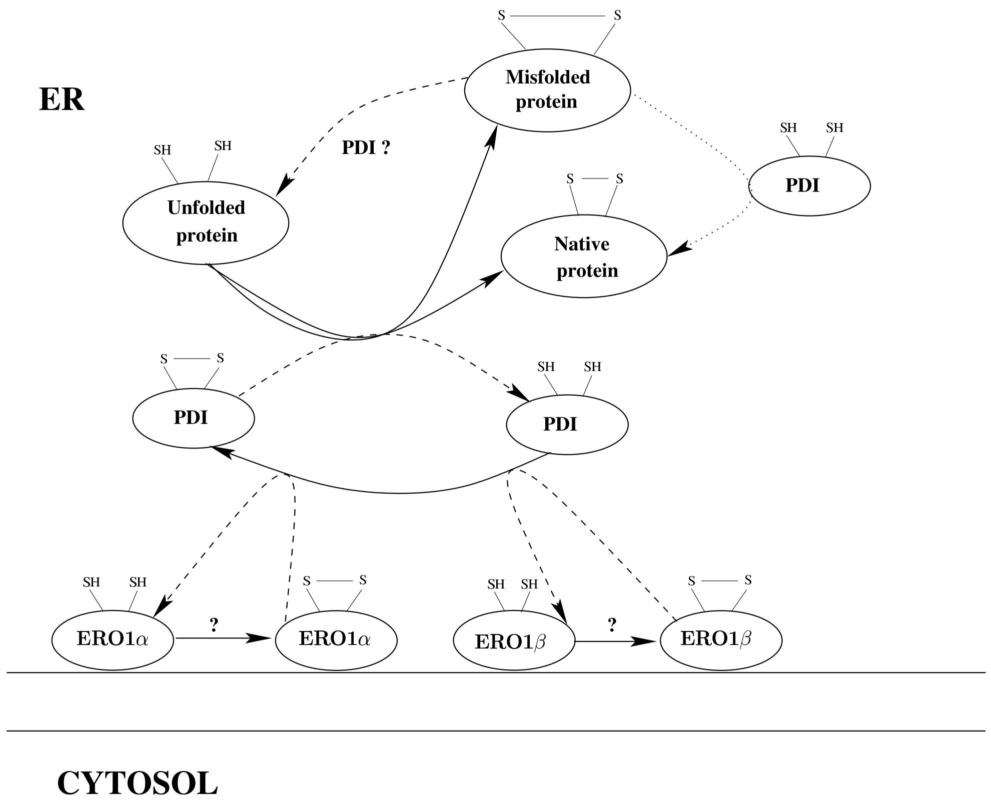


Figure 5.5. Model for disulphide bridge formation in the endoplasmic reticulum of *S.cerevisiae* [FG03]. A solid line indicates oxidation of the molecule originating the line, while a dashed line stands for reduction. The dotted line indicates an isomerization reaction, which does not involve a net change in redox state. Unfolded proteins are oxidized by the enzyme PDI, which is in turn reoxidized by peripheral membrane proteins ERO1α and ERO1β, whose recharging mechanism has not been completely explained. PDI is also involved in correcting misfolded proteins, and is supposed to be implied in their possible unfolding, but this last capacity has not been proved.

5.2 Cysteine Bonding State Prediction

The first step in predicting disulphide bridges in a given protein is that of identifying oxidized cysteines which are involved in disulphide bridge formation. Due to the environmental conditions discussed in the previous section, most cysteines within a given protein usually share the same redox state, either reduced (ab. 64%) or oxidized (ab. 26%), while only a small fraction (ab. 10%) of known protein structures contain both cases together. In the following paragraph we will give an overview of the different approaches proposed for bonding state prediction, while the rest of the section will discuss the algorithms we developed for the task (see [FPV02, CFPV03b, CFPV04]).

5.2.1 Overview of Current Methods

The program CYSPRED developed by Fariselli et al. [FRC99] (accessible at `http://gpcr.biocomp.unibo.it/predictors/cyspred/`), uses a neural network with no hidden units, fed by a window of $2k + 1$ residues centered around the target cysteine. Each element of the window is a vector of 20 components (one for each amino acid) obtained from multiple alignment profiles. This method achieved 79% accuracy (correct assignment of the bonding state) measured by 20-fold cross validation and using a non-redundant set of 640 high quality proteins from PDB Select [HS94] of October 1997. Accuracy was boosted to 81% using a jury of six networks. Still, the bonding state of each cysteine is assigned independently.

Fiser & Simon [FS00] later proposed an improvement based on the observation that cysteines and half cystines [1] rarely co-occur in the same protein. In their algorithm, if a larger fraction of cysteines are classified as belonging to one class, then all the remaining cysteines are predicted in the same state. The accuracy of this method is as high as 82%, measured by a jack-knife procedure (leave-one-out applied at the level of proteins) on a set of 81 protein alignments. This result suggests that a good method for classifying proteins in two classes is also a good method for predicting the bonding state of each cysteine, even though in this way the accuracy for proteins containing both cysteines and half cystines is sacrificed. The program, called CYSREDOX, is accessible at `http://pipe.rockefeller.edu/cysredox/cysredox.html`.

Later, Mucchielli-Giorgi et al. [MGHT02] have proposed a predictor that exploits both local context (a window centered around the target cysteine) and global protein descriptors. Interestingly, they found that in absence of evolutionary information, prediction of covalent state based on global descriptors was more accurate (77.7%) than prediction based on local descriptors alone (67.3%). Their best predictor, based on a multiple classifier reaches almost 84% accuracy measured by 5-fold cross-validation on a set of 559 proteins from Culled PDB.

We developed a different approach [FPV02, CFPV03b] for exploiting the key fact that cysteines and half cystines rarely co-occur. Prediction in this case is achieved by using two cascaded classifiers. The first classifier predicts the type of protein based on the whole sequence. Classes in this case are "all", "none", or "mix", depending whether all, none, or some of the cysteines in the protein are involved in disulphide bridges. The second binary classifier is then trained to selectively predict the state of cysteines for proteins assigned to class "mix", using as input a local window with multiple alignment profiles. The method achieves an accuracy of 85% as measured by 5-fold cross validation, on a set of 716 proteins from the September 2001 PDB Select dataset [CFPV03b].

Shortly after, Malaguti et al. [MFMC02] have proposed yet another approach where the disulphide bonding state is predicted as in CYSPRED but predictions are then refined using a Hidden Markov Model [Rab89] trained to recognize the stochastic language that describes the alternate presence of bonding and non-bonding cysteines along the sequence. This improved method achieved the performance level of 88% correct prediction measured

[1] a cystine is the dimer formed by a pair of disulphide-bonded cysteines.

by 20-fold cross validation on a non redundant dataset.

In the following we describe the architecture we developed in [FPV02, CFPV03b] together to a global refinement stage [CFPV04], either by BRNN or by the HMM employed in [MFMC02] [2]. In the case of BRNN refinement, a final stage with a finite state automata (FSA) was also employed to force consistent predictions, reaching the best performances to date.

5.2.2 Output-Local Predictor

We denote by $Y_{i,t}$ a binary random variable associated with the bonding state of cysteine at position t in protein i. For each protein, the available features consist of:

- a vector W_t^k representing the input-local context of cysteine t; this is a window of size $2k+1$ centered around position t, enriched with evolutionary information in the form of multiple alignment profiles;
- a vector D_i representing a global set of attributes (or descriptors) for protein i; in the simplest case this may consist of amino acid frequencies but more complex descriptors will be used in the following (see section 5.2.2.3);

By comparison, note that traditional feedforward neural network approaches predict the bonding state using input-local information W_t^k only [FRC99].

We are interested in building a model for $P(Y_{i,t} = 1|D_i, W_t^k)$. For each protein, let C_i be a three-state variable that represents the propensity of the protein to form disulphide bridges. The possible states for C_i are “all”, “none”, and “mix”, depending whether all, none, or some of the cysteines in the protein are involved in disulphide bridges. After introducing C_i, the model can be decomposed as follows:

$$P(Y_{i,t}|D_i, W_t^k) = \sum_{C_i} P(Y_{i,t}|D_i, W_t^k, C_i)P(C_i|D_i, W_t^k). \tag{5.2}$$

We can simplify the above model by introducing some conditional independence assumptions. First, we assume that the type of protein C_i depends only on its descriptor: $P(C_i|D_i, W_t^k) = P(C_i|D_i)$. Second, we simplify equation (5.2) by remembering the semantics of C_i:

$$\begin{aligned} P(Y_{i,t} = 1|D_i, W_t^k, C_i = \text{all}) = 1 \\ P(Y_{i,t} = 1|D_i, W_t^k, C_i = \text{none}) = 0 \end{aligned} \tag{5.3}$$

(this can be seen as a particular form of context-specific independence [BFGK96]). As a result, the model in equation (5.2) can be implemented by a cascade of two classifiers. Intuitively, we start with a multiclass classifier for computing $P(C_i|D_i)$. If this classifier

[2]Results for HMM filtering are obtained in collaboration with the laboratory of biocomputing, CIRB, University of Bologna, Italy

predicts one of the classes "all" or "none", then all the cysteines of the protein should be classified as disulphide-bonded or non-disulphide-bonded, respectively. If instead the protein is in class "mix", we refine the prediction using a second (binary) classifier for computing $P(Y_{i,t}|D_i, W_t^k, C_i = \text{mix})$. Thus the prediction is obtained as follows (see also fig. 5.6):

$$\begin{aligned} P(Y_{i,t} = 1|D_i, W_t^k) &= P(Y_{i,t} = 1|D_i, W_t^k, C_i = \text{mix})P(C_i = \text{mix}|D_i) \\ &\quad + P(C_i = \text{all}|D_i) \end{aligned} \tag{5.4}$$

By comparison, note that the method in [FS00] cannot assign different bonding states to cysteine residues in the same sequence.

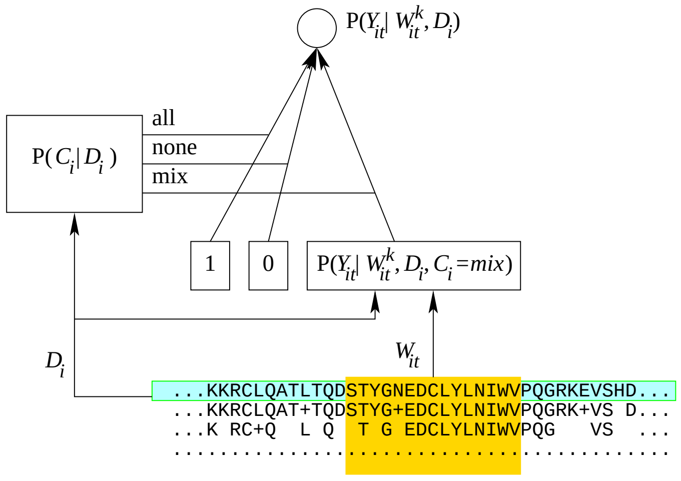

$P(Y_{it}| W_{it}^k, D_i)$
$P(C_i|D_i)$
all
none
mix
1
0
$P(Y_{it}| W_{it}^k, D_i, C_i = mix)$
D_i
W_{it}
...KKRCLQATLTQDSTYGNEDCLYLNIWVPQGRKEVSHD...
...KKRCLQAT+TQDSTYG+EDCLYLNIWVPQGRK+VS D...
...K RC+Q L Q T G EDCLYLNIWVPQG   VS ...

Figure 5.6. The two-stage system. The protein classifier on the left uses a global descriptor based on amino acid frequencies. The local context classifier is fed by profiles derived from multiple alignments together with protein global descriptor.

5.2.2.1 Implementation using probabilistic SVM

The architecture in fig. 5.6 is implemented using combinations of support vector machines. In their standard formulation SVM output hard decisions rather than conditional probabilities. However, margins can be converted into conditional probabilities in different ways both in the case of binary classification [Kwo99, Pla00] and in the case of multiclass classification [PPF02] (see section 3.1.1). The method used here extends the algorithm presented in [Pla00], where svm margins are mapped into conditional probabilities using a logistic function, parameterized by an offset B and a slope A:

$$P(C_i = 1|x) = \frac{1}{1 + \exp(-Af(x) - B)} \tag{5.5}$$

In [Pla00], parameters A and B are adjusted according to the maximum likelihood principle, assuming a Bernoulli model for the class variable. This is extended here to the multiclass case by assuming a multinomial model and replacing the logistic function by a softmax function [Bri89]. More precisely, assuming Q classes, we train Q binary classifiers, according to the one-against-all output coding strategy. In this way, for each point x, we obtain a vector $[f_1(x), \cdots, f_Q(x)]$ of margins, that can be transformed into a vector of probabilities using the softmax function as follows:

$$g_q(x) = P(C = q|x) = \frac{e^{A_q f_q(x)+B_q}}{\sum_{r=1}^{Q} e^{A_r f_r(x)+B_r}} \tag{5.6}$$

The softmax parameters A_q, B_q are determined as follows. First, we introduce a new dataset $\{(f_1(x_i), \ldots, f_Q(x_i), \mathbf{z}_i), i = 1, \ldots, m\}$ of examples whose input portion is a vector of Q margins and output portion is a vector $\mathbf{z}$ of indicator variables encoding (in one hot) one of Q classes. As suggested in [Pla00] for the two classes case, this dataset should be obtained either using a hold-out strategy, or a k-fold cross validation procedure. Second, we derive the (log) likelihood function under a multinomial model, and search the parameters A_q and B_q that maximize

$$\ell = \sum_i \sum_{q=1}^{Q} z_{q,i} \log g_q(x_i) \tag{5.7}$$

where $z_{q,i} = 1$ if the i-th training example belongs to class q and $z_{q,i} = 0$ otherwise.

5.2.2.2 A Fully-Observed Mixture of SVM Experts

While the above method yields multiclass conditional probabilities it does not yet implement the model specified by equation (5.4). We now discuss the following general model, that can be seen as a variant of the mixture-of-experts architecture [JJNH91]:

$$P(Y = 1|x) = \sum_{q=1}^{Q} P(C = q|x) P(Y = 1|C = q, x) \tag{5.8}$$

In the above equation, $P(C = q|x)$ is the probability that q is the expert for data point x, and $P(Y = 1|C = q, x)$ is the probability that x is a positive instance, according to the q-th expert. Collobert *et al.* [CBB02] have recently proposed a different SVM embodiment of the mixture-of-experts architecture, the main focus in their case being on the computational efficiency gained by problem decomposition. Our present proposal for cysteines is actually a simplified case since the discrete variable C associated with the gating network is not hidden[3]. Under this assumption there is no credit assignment problem and a simplified training procedure for the model in equation (5.8) can be derived as follows.

[3]Actually the architecture in fig. 5.6 for cysteines is even simpler since two of the experts output a constant prediction.

Let $f'_q(x)$ denote the margin associated with the q-th expert. We may obtain estimates of $P(Y = 1|C = q, x)$ using a logistic function as follows:

$$p_q(x) = P(Y = 1|C = q, x) = \frac{1}{1 + \exp(A'_q f'_q(x) + B'_q)}. \tag{5.9}$$

Plugging equations (5.6) and (5.9) into equation (5.8), we obtain the overall output probability as a function of $4Q$ parameters: A_q, B_q, A'_q, and B'_q. These parameters can be estimated by maximizing the following likelihood function

$$\ell = \sum_{i=1}^{m} \frac{1 - y_i}{2} \log \left(\sum_{q=1}^{Q} g_q(x_i) p_q(x_i) \right) \tag{5.10}$$

The margins to be used for maximum likelihood estimation are collected by partitioning the training set into k subsets. On each iteration all the $2Q$ SVMs are trained on $k - 1$ subsets and the margins computed on the held-out subset. Repeating k times we obtain as many margins vectors $(f_1(x), \cdots, f_Q(x), f'_1(\mathbf{x}), \cdots, f'_Q(\mathbf{x}))$ as training examples. These vectors are used to fit the parameters A_q, B_q, A'_q, and B'_q. Finally, the $2Q$ machines are re-trained on the whole training set.

5.2.2.3 Spectrum Kernel

The ultimate goal of predicting the bonding state of cysteines is the location of disulphide bonds, a structural feature which depends on the properties of possibly very distant portions of the sequence. Therefore, it might be useful to adopt computational approaches which can exploit the whole sequence as input. Besides standard kernels, which are naturally limited to process fixed sized numerical inputs, convolution kernels (see sec. 2.5.2) allow to process structured data. The spectrum kernel [LEN02] is a convolution kernel specialized for string comparison problems, which has been successfully employed in remote homology detection (see sec. 4.3.2).

Given all the possible strings of size k in a certain alphabet $\mathcal{A}$, the k-spectrum of a sequence s is the vector $\Phi_k(s)$ of the number of times each string of length k is contained in the sequence. The k-spectrum kernel is then defined as the scalar product of the feature vectors Φ_k corresponding to the two sequences. While the feature space has a very large dimension these feature vectors are extremely sparse. In effect there exists a very efficient method to compute the kernel that makes use of suffix trees.

Suffix trees [Gus97] are particular data structures that can be extremely useful and efficient in solving various problems of string matching. A suffix tree for a given string s of size m is a tree with exactly m leaves, where each path from the root to a leaf is a suffix of the string s. The suffix tree for the string s can be constructed in $O(m)$ time using Ukkonen's algorithm [Ukk95]. In our case, a suffix tree can be used to identify all the substrings of size k contained in the given sequence, simply following all the possible paths of size k starting from the root of the tree. Moreover, the problem of calculating the number of occurrences of each substring can be solved just counting the number of leaves in the subtree that starts at the end of the corresponding path. Given that the number of leaves of the tree is simply the size of the represented string, we have a linear-time method to calculate the k-spectrum of a string.

Further modifications are needed to avoid the need of directly calculating the scalar product of the two feature vectors for the computation of the kernel. A generalized suffix tree is a suffix tree constructed using more than one string [Gus97]. Given a set of strings there exists a variant of Ukkonen's algorithm that can build the corresponding generalized suffix tree in a time linear in the sum of the sizes of all the strings. A generalized suffix tree can be used to calculate the k-spectrum kernel at once, just traveling the tree in a depth first manner and counting the number of occurrences of every substring of size k in all the strings.

Interestingly, descriptors based on amino acid frequencies as defined in [MGHT02], basically correspond to the use of a spectrum with $k = 1$. Augmenting the feature space by incorporating short subsequences increases the expressive power of the model and may improve prediction accuracy, if k is carefully chosen and enough training sequences are available. A more general form of the spectrum kernel can also be constructed summing the values of some k-spectrum kernels for certain values of k. No modifications of the presented algorithm are needed, given that all the k-spectrum kernels with different k can be calculated at once with a single visit of the tree.

5.2.3 Output-Global Refiners

Local predictions were refined by a global stage, taking as input the vector of probabilistic predictions for all cysteines of a given sequence. Global refinement has been successfully employed in protein structure prediction tasks such as secondary structure prediction (see section 4.3.3.1), and proved to be very effective in bonding state predictions refinement [MFMC02]. The problem can be seen as that of translating an input vector (the vector of local predictions) into an equal size output one (that of globally refined predictions). Such task is rather hard to address by kernel machines, and no general approach has been developed so far. However, connectionist models [FGS98] offer a powerful methodology to treat both discrete structured data and input output isomorph problems. We implemented different refinement architectures, either by bi-recurrent neural networks or by hidden Markov models as proposed in [MFMC02].

5.2.3.1 Hidden Markov Models

The HMM proposed in [MFMC02] is a modified version of HMM able to handle probability vectors as those provided by the local output architecture. Consider a sequence of T vectors of size A. We refer to this sequence vector with the notation:

$$I(1), \ldots, I(T) = (I_1(1), \ldots, I_A(1)), \ldots, (I_1(T), \ldots, I_A(T))). \tag{5.11}$$

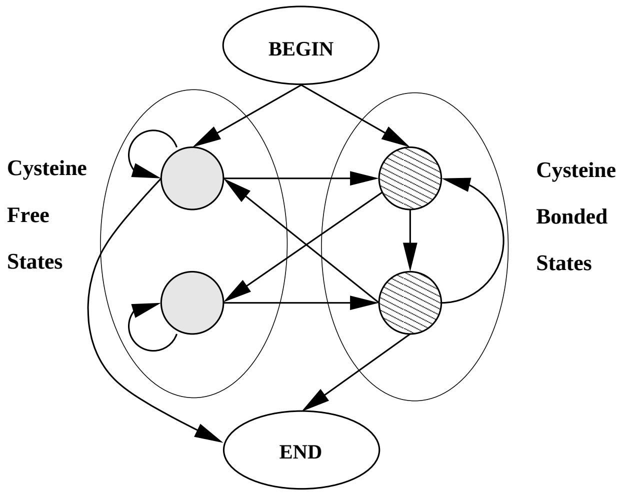


Figure 5.7. The four-state HMM constraining bonded cysteines in a sequence to an even number. States on the left are non-bonding cysteine states, while states on the right are bonded cysteine ones. The end state can only be reached by paths containing an even number of bonded cysteines.

The components of each vector $I(t)$ are positive and sum to a constant value M (independent of the position t). A sequence vector based HMM is composed of S states connected by means of the transition probabilities a_{ij}, where each state emits a vector of size A. The probability density function for the emission of a vector from each state is determined by a number A of parameters that are peculiar for each state k and are indicated with the symbols $e_{k,h}$ (with $h = 1, \ldots, A$):

$$P(I(t)|s(t) = k) = \frac{1}{Z} \sum_h I_h(t) e_{k,h} \tag{5.12}$$

where $s(t)$ is the t-th state in the path, and the normalization parameter Z is independent of the state and of the values of the sequence vector (see [MFKC02a] for a proof).

In our refinement task, the sequence vector is the vector of local predictions for a given protein chain, T is the number of cysteines in the chain, and each vector has two components (A=2) corresponding to the predicted probabilities of free and bonding state, respectively. The HMM is composed of a Markov model with four states connected as shown in figure 5.7. The four states are the minimum number of states required to constrain to an even number the paired cysteines in a chain, and the model actually allows to reach an end state only if the path contains an even number of bonded states/cysteines.

Training the HMM parameters is accomplished by using a modified Expectation-Maximization algorithm [MFKC02a]. Globally refined predictions for all cysteines within a given chain are obtained by the maximum likelihood path in the HMM as computed by the Viterbi algorithm [Rab89].

5.2.3.2 Bidirectional Recurrent Neural Networks

A bidirectional recurrent neural network (BRNN) is a noncausal dynamical system with a factorized hidden state space. The model was first proposed in the context of prediction of protein secondary structure [BBF$^+$99] and is described by the following set of equations

$$\begin{aligned} F(t) &= \phi(F(t-1), I(t), \theta_\phi) \\ B(t) &= \beta(B(t+1), I(t), \theta_\beta) \\ O(t) &= \eta(F(t), B(t), I(t), \theta_\eta) \end{aligned} \tag{5.13}$$

where $F(t) \in \mathbb{R}^n$ and $B(t) \in \mathbb{R}^m$ are two real vectors, $\phi(\cdot)$ and $\beta(\cdot)$ are nonlinear transition functions (realized by two feedforward neural networks with shared weights θ_ϕ and θ_β, respectively) and $I(t)$ is a real vector that encodes the input at sequence position t. $O(t)$, the output at position t is an adaptive function $\eta(\cdot)$ of the state variables and the current input, realized by a feedforward neural network with weights θ_η.

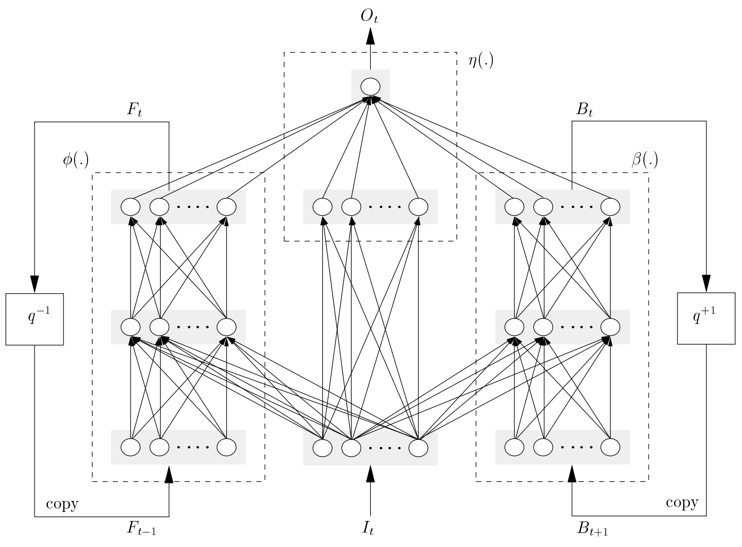


Figure 5.8. A BRNN architecture realizing a noncausal IO-isomorph transduction.

An example of architecture implementing such equations is shown in figure 5.8, where all feed forward neural networks have a single hidden layer, and the output at position t is represented by a single value O_t. The forward hidden state $F(t)$ is copied back to the input of $\phi(\cdot)$, as graphically represented by the *causal* shift operator q^{-1}. Therefore, $F(t)$ can be interpreted as an adaptive memory that summarizes information about the inputs $I(1), \ldots, I(t)$. Similarly, the backward hidden state $B(t)$ is copied to the input of $\beta(\cdot)$, as shown by the *noncausal* shift operator q^{+1}, and it summarizes information about $I(t), \ldots, I(T)$, being T the total length of the input sequence. As a result, each output $O(t)$ depends on the entire input sequence. The above equations are completed by the two boundary conditions $F(0) = B(T+1) = 0$. In the same way as traditional feedforward neural networks can be interpreted as a model of the conditional distribution of outputs given the inputs (both real vectors), BRNN can be interpreted as a model of the conditional distribution of output sequences given input sequences. Under this interpretation, the network is trained by a maximum likelihood procedure. The optimization problem can be solved by gradient descent where the gradient of the likelihood with respect to all the free parameters $\theta_\phi, \theta_\beta$, and θ_η is computed by a generalization of the backpropagation through time algorithm.

In our application to global refinement of cysteine bonding state predictions, sequence positions t correspond to the ordinal numbers assigned to each cysteine in a given chain, starting from its C-terminus, and T is the total number of cysteines in the chain. The input vector $I(t)$ for the t-th cysteine is represented by the local prediction for the cysteine, possibly enriched with contextual information.

5.2.4 Data Preparation

All the experiments were performed using 4136 protein segments containing cysteines (free and disulphide-bonded, or half cystines). Sequences were taken from the crystallographic data of the Brookhaven Protein Data Bank [BBB+02]. Disulphide bond assignment was based on the Define Secondary Structure of Proteins (DSSP) program [KS83b]. Non-homologous proteins (with an identity value $<$ 25 % and without chain breaks) were selected using the PAPIA system (`www.cbrc.jp/papia/papia.html`). Cysteines forming inter-chain disulphide bonds are included as 'free' cysteines in the database (30 cysteines extracted from 23 monomeric chains and amounting to 0.76% of the total number of cysteines), in order to obtain an even number of bonded cysteines for chain and also simplify the subsequent connectivity prediction stage (see section 5.3). After this filtering procedure, and after removing trivial segments containing a single cysteine, the total number of proteins is 780, with 3935 cysteines, 1446 of which were in the disulphide-bonded state and 2489 of which were in the non disulphide-bonded state. For each protein in our database, a profile based on a multiple sequence alignment was created using the PSIBLAST [AMS+97] program on the non-redundant Swiss-Prot+TrEMBL dataset of sequences.

5.2.4.1 Input Encoding

The descriptor D_i described in [FPV02] is a real vector with 24 components, similar to the one used in [MGHT02]. The first 20 components are $\log(N_i^j/N^j)$, where N_i^j is the number of occurrences of amino acid type j in protein i and N^j is the number of occurrences of amino acid type j in the whole training set. The 21st component is $\log(N_i/N_{avg})$ where N_i is the length in residues of sequence i and N_{avg} is the average length of the proteins in the training set. The next two components are N_i^{cys}/N_{max}^{cys} and N_i^{cys}/N_i where N_i^{cys} and N_{max}^{cys} are respectively the number of cysteines in protein i and the maximum number of observed cysteines in the training set. The last component is a flag indicating whether the cysteine count is odd.

The local input window W_t^k is represented as the set of multiple sequence profile vectors of the residues flanking cysteine at position t. In the experiments, we used a symmetrical window centered at each cysteine varying the window size parameter k from 7 to 9. Note that although the central residue is always a cysteine, the corresponding feature is still taken into account since the profile in this case indicates the degree of conservation of the cysteine. For each of the $2k + 1$ positions we used a vector of 22 components, enriching the 20-components profile with relative entropy and conservation weight.

5.2.4.2 Cysteines Conservation

Cysteines tends to be conserved in multiple alignments when they form disulphide bridges. In the experiments reported below, we made available this information to the output-local stage in two ways. First, we defined an extended descriptor with H additional components related to the conservation of cysteines. For $h = 0, \ldots, H-1$, the h-th extra component is the fraction of cysteines in the sequence whose multiple alignment conservation falls in the bin $[h/H, (h+1)/H]$. This global descriptor is fed to the binary classifier together with the local window. Second, we defined a special sequential representation of the proteins that incorporates evolutionary information. In this representation a protein is a string in an extended alphabet having $19{+}Z$ symbols, where occurrences of C (cysteine) are replaced by a special symbol that indicates the degree of conservation of the cysteines in the corresponding positions of multiple alignments. For example if $Z = 2$, one symbol encodes highly ($> 50\%$) conserved cysteines and another one encodes lowly conserved ones. We employed such extended alphabet for the none-all-mix classifier implemented with the Spectrum kernel.

5.2.5 Results

Table 5.1 shows comparative experimental results for the various implementations. In all the experiments we estimated prediction accuracy by a 20-fold cross-validation procedure. The quantities of interest for measuring the performance of the classifier are:

- Q_2 the classification accuracy measured as the fraction of cysteines correctly assigned to their disulphide bonding state
- Q_p the fraction of sequences for which each cysteine is assigned to the correct disulphide bonding state.

Results are reported for different sizes of the local input window k, ranging from 7 to 9. All the values are enriched by the 95% confidence interval computed on the results of the 20-fold cross-validation.

5.2.5.1 Output-local experts

A preliminary model selection phase was conducted in order to choose the appropriate kernel type (for the binary classifier) and hyperparameters.

Table 5.1. Summary of the experimental results.

	$k = 7$		$k = 8$		$k = 9$	
Method	Q_2	Q_p	Q_2	Q_p	Q_2	Q_p
Loc29	86±1	69±3	87±1	70±3	86±1	70±3
S24	86.5±1	71±3	87±1	72±3	86.5±1	71±3
HMM Loc29	88±1	82±3	88±1	82±3	88±1	82±3
HMM S24	88±1	83±3	88±1	82.5±3	88±1	83±3
BRNN Loc29	86±1	73±3	86.5±1	73±3	87±1	73±3
BRNN S24	87±1	75±3	87±1	75±3	87±1	74±3
BRNN Loc29+f	89±1	79±3	89±1	80±3	89±1	79±3
BRNN S24+f	89±1	80±3	89±1	80±3.1	89±1	78±3
BRNN S24+f FSA	89±1	82±3	89±1	83±3	89±1	82±3

We first report the results obtained by using only a single stage method, named *Loc29* (first row in Table 1), constituted by the SVM classifier with third degree polynomial kernel, taking as input the window and the 29 symbols protein descriptors.

In *S24* we added the second stage classifier which uses the spectrum kernel operating on sequences. The spectrum kernel is calculated summing over all the subsequence lengths from 2 to 5. Softmax parameters (see equations (5.6) and (5.9)) were estimated by 3-fold cross validation (inside each fold of the outer 20-fold cross-validation), *after* kernel parameter estimation.

The accuracy of these methods at the single cysteine level, Q_2, is encouragingly high. In particular, SVM with a spectrum kernel outperforms all previously published methods, with the exception of the one reported in [MFMC02], which uses HMM to refine neural network predictions. Not surprisingly, however, accuracy at the whole protein level, Q_p is not particularly high since SVM predict each cysteine independently. The following experiments show that HMM and BRNN used to refine SVM prediction produce a significant improvement of Q_p and also improve Q_2.

5.2.5.2 Filtering with HMM

The vector-based HMM described in section 5.2.3 analyses the local outputs, constraining the overall system to predict an even number of cysteines in the bonding state in each given chain, independently of the number of cysteines in the protein. The HMM thus outputs the most probable sequence of cysteine labels consistent with the regular grammar induced by the trained model, given the local predictions. The advantage of this refinement procedure is particularly evident when comparing performance at the protein level, where accuracy is increased of more than 10 percentage points regardless of the type of algorithm employed for local predictions (see rows three and four in table 5.1).

5.2.5.3 Filtering with BRNN

BRNN can develop complex nonlinear and noncausal dynamics that can be used to correct output-local prediction by trying to capture globally valid sequences of cysteine bonding states. Input sequences $I(t)$ are relatively short (from 2 to 20 elements) since they are obtained by looking at cysteines only. In this setting, vanishing gradients do not represent a problem for training.

In a preliminary tuning phase, we ran the model varying the architectural parameters. The results showed in table 5.1 are for a BRNN having forward and backward state vectors of dimension $n = m = 2$. The feedforward neural networks implementing transition functions $\phi(\cdot)$ and $\beta(\cdot)$ and the output function $\eta(\cdot)$ have no hidden units. In order to control overfitting and to stop the learning process, we added a penalty (regularization) term to the cross-entropy function used for binary classification problems. Rows labeled as 'BRNN Loc29' and 'BRNN S24' in table 5.1 show prediction accuracies obtained using the prediction of the corresponding SVM as a single scalar input $I(t)$ to the BRNN.

In a second set of experiments, we provided the BRNN with a richer input $I(t)$ comprising SVM prediction and a vector of local features encoding cysteine conservation in multiple alignments (encoded in a 5-dimensional vector whose positions correspond to five degrees of conservation). Results (reported in rows labeled as 'BRNN Loc29+f' and 'BRNN S24+f') show a significant improvement of accuracy at the protein level. Using this setting we obtained the best accuracy (89%) at the cysteine level.

5.2.5.4 Combining BRNN and HMM Advantages

The two refinement methods proposed obtain significant improvements over local predictions. However, they provide different advantages, as the HMM results in the best predictions at a protein level (Q_p), while the BRNN obtains the best performances at a cysteine level (Q_2). Moreover, the architecture employing BRNN can still produce inconsistent labelling, that is chains containing an odd number of bonded cysteines. In order to combine the advantages of the two filtering procedures, we could add the HMM filter to the output of the BRNN one. However, in order to control the parameters of the overall architecture and avoid overfitting, and considered that the BRNN already implements a global refinement, an easier filter can be realized, which does not need to be trained, and simply ensures the consistence of the final output. Such *finite state automaton* (FSA) has the same states and allowed transitions of the HMM in figure 5.7, but all transitions have the same probability, and emission probabilities for a given state are a delta Dirac over the bonding/non-bonding emission for bonding and non-bonding states respectively (see fig. 5.7). The most probable path for a given vector of predictions can then be computed by the Viterbi algorithm [Rab89]. A similar approach for ensuring consistencies in secondary structure predictions was proposed in [CFPV03a] (see section 4.3.3.1). Results for such postprocessing applied to the output of the best BRNN architecture are reported in the last row of table 5.1, and obtain the best performances both in cysteine (Q_2) and protein (Q_p) accuracy. The best overall results are obtained for $k = 8$, and confusion matrices at protein class level (none/all/mix) for the architecture both before (BRNN S24+f) and after (BRNN S24+f FSA) the post-processing are reported in tables 5.2(a) and 5.2(b) respectively.

	None	All	Mix	
None	463	21	18	True
All	25	160	18	True
Mix	40	22	13	True
	Predicted			

(a)

	None	All	Mix	
None	471	24	7	True
All	32	164	7	True
Mix	42	6	27	True
	Predicted			

(b)

Table 5.2. Confusion matrices at protein class level (none/all/mix) for the 'BRNN S24+f' architecture with $k = 8$ both before (a) and after (b) post-processing by FSA.

The effect of the FSA post-processing is entirely related to mix predictions: it allows to turn part of the false mix proteins into none or all, and to correct inconsistent all predictions (with an odd number of bonded cysteines) into mix by swapping the least confident prediction. Note that while a correct all/none assignment implies correct prediction for all the cysteines in the chain, a correct mix assignment gives no information on predictions at cysteine level for the chain. Actually, only two of the true mix predicted by 'BRNN S24+f FSA' also have correct assignments at cysteine level, and thus contribute to the overall protein accuracy (Q_p). On the other hand, the post-processing allows to correctly predict cysteines in eleven mix chains, all of them being chains with an odd number of cysteines and just one free cysteine. A detailed inspection of such cases shows two main reasons explaining why the error is actually the least confident prediction of the BRNN (thus allowing the FSA to correct it):

- The cysteine was predicted as bonded with the smallest confidence already by the local-output stage, and most of the times it is actually poorly conserved.
- For long chains, there is a preference in swapping the label of one of the two extreme cysteines, that is the ones nearest to the N or the C terminus.

Finally, by looking at the prediction of the BRNN over the 30 cysteines involved in interchain bonds (here treated as 'free' cysteines), we note that 16 of them are actually predicted as bonded, showing that they often share the same characteristics of intrachain bonded ones. The advantages of forcing an even number of bonded cysteines in a chain, both in allowing constraint application and in simplifying the subsequent connectivity prediction stage, amply compensate such systematic errors. However, a finer architecture would be required in order to take into consideration this additional degree of complexity.

5.3 Connectivity Prediction

Once the set of disulphide bonded cysteines within a chain has been identified, the correct connectivity pattern has to be predicted, by pairing each oxidized cysteine with its bonded counterpart.

The problem has been addressed in [FC01] by representing a given chain by a fully connected weighted graph $G = (V, E)$, with a number of vertices $|V|$ equal to the number of oxidized cysteines in the chain. The weight on the edge E_{ij} connecting two oxidized cysteines i and j represents their interaction potential. The task is that of finding the maximum weight perfect matching in G, that is the subset of edges $E^* \subset E$ such that each pair of vertices in G is connected by one and only one edge, and which maximizes the sum of the weights of the edges. The perfect matching problem can be solved in polynomial time using linear programming, while the computation of the interaction potentials has been initially addressed by simulated annealing [FC01] and subsequently improved by using neural networks [FMC02].

State-of-art performances have been obtained in [?], with an algorithm analogous to the one employed for coarse contact maps prediction (see sec. 4.3.3.2). The problem is cast into the task of learning a scoring function computing the similarity between the correct connectivity graph and all possible candidate graphs, and using it to recover the maximal score graph for a given chain. The scoring function is represented by a bi-recursive neural network, taking as input for each vertex the context of the corresponding cysteine in the form of a multiple alignment profile, and information on the length of the chain and the relative position of the cysteine within it.

The available methods for connectivity prediction are limited to chains containing a number of bridges B not greater than five, as the search space dimension, given by $(2B - 1)!!$, is too huge for a higher number of bridges. However, the vast majority of the chains respect this limitation, as less then 10% of the disulphide-bonded chains in Swiss-Prot have more than five bridges.

5.4 Conclusions

Prediction of disulphide bridges is an important step towards overall protein structure prediction, as disulphide bridges impose strong constraints on the chains three-dimensional structure. The task has been typically divided into two subsequent steps. Firstly, predict the bonding state of each cysteine within a given chain as either reduced or oxidized. Secondly, given the subset of oxidized cysteines, predict their connectivity pattern, that is the correct pairing between a cysteine and its disulphide bonded partner. Concerning bonding state prediction, we have presented state-of-art algorithms reaching an accuracy as high as 89% at a cysteine level, and 83% at a protein level, where a protein is considered correctly predicted if all its cysteines have been predicted in the correct bonding state. Such methods consist of a local prediction stage, and a global refinement stage. The former takes as input a window of multiple alignment profiles around a target cysteine, as well as a global descriptor containing information on the entire sequence, and outputs a probability estimate for the cysteine being in a bonding state. The latter takes as inputs the local predictions for all cysteines in a given chain and outputs refined predictions by propagating information all along the chain. The number of bonded cysteines in a chain can be constrained to be even, either directly in the refinement stage or by post-processing the outputs with a finite state automata, the latter method resulting in slightly better overall performances. However, systematic errors on small but important sets of proteins are still committed. The only cases in which the architecture is able to correctly predict a mix protein, that is a protein with both bonded and free cysteines, are proteins containing an odd number of cysteines, only one of which being non-bonded. Such correct mix predictions are typically obtained thanks to the constraints which force an even number of bonded cysteines in the predictions, as the non-bonded cysteine is usually predicted as the one with the lowest probability of being bonded. However such constraints, together to the necessity to simplify the successive connectivity prediction task, force the method to consider the rare interchain bonded cysteines as free, even if they contribute to overall protein structure stability, and often share similar characteristics with intrachain bonded ones, as highlighted by the fact that they are often predicted as bonded by the unconstrained algorithm.

Anyway, it seems that no significant further improvement is possible using the same type and quantity of inputs as those employed so far. More sources of information have to be sought and more accurate biological knowledge of cysteine function should be plugged into the models, in order to obtain a significant improvement over the state-of-art. Recent researches enlightened the variety of roles that cysteines play in biological systems (see [GWG+03] for a review), the most important being driven by disulphide bridge formation and ligand binding, that is formation of bonds between cysteines and various ligands typically containing metal groups, such as heme groups and iron-sulfur clusters. In the next chapter, we will address the problem of discriminating between disulphide bond and ligand bond cysteines, in order to provide additional and more accurate information useful to help protein structure prediction as well as experimental structure determination methods.

Concerning disulphide connectivity prediction, recent analyses seem to suggest that the most important features for the task are the position of the cysteines within the sequence, their relative distance as well as the overall length of the chain, which tend to be more informative with respect to cysteine context and evolutionary information. We are investigating the possibility to develop ad hoc kernels able to fully exploit this kind of information in order to improve predictive performances in this task.

Chapter
6

Cysteine Binding Types Prediction

Non-free cysteines that are not involved in the formation of disulphide bridges usually bind prosthetic groups that include a metal ion and that play an important role in the function of the protein. The discrimination between the presence of a disulphide bridge (DB) or a metal binding site (MBS) in correspondence of a bound cysteine is often a necessary step during the NMR structural determination process of metalloproteins, and its automation may significantly help toward speeding up the overall process. Several proteins are known where both situations are in principle plausible and it is not always possible to assign a precise function to each cysteine. For example the mitochondrial copper metallochaperone Cox17 contains six cysteines but it is not precisely known which ones are actually involved in metal binding [HGGW01]. Another example is the inner mitochondrial membrane protein sco1p that contains a `CXXXC` motif and experimental evidence that leads us to believe it is involved in copper transport, but whose fold similarity to a thioredoxin fold would conversely suggest a catalytic activity [Chi00, BBB+03].

In addition to the above motivations, the discrimination between DB and MBS may help toward a more accurate prediction of the disulphide bonding state of cysteines (see chapter 5). A potential direction that we are currently investigating is that of firstly predicting the subcellular location [NBvH99, ENBvH00, NR03a] of a given protein. For proteins (or parts of proteins in the case in integral membrane ones [RCFS95, CE99, MFKC02b, CR02]) staying in an oxidizing environment, well conserved cysteines are mostly involved in some type of binding, and the problem can be cast into that of discriminating between disulphide bond and ligand bond cysteines.

Many different post-translational modifications have been observed or supposed for cysteines [GWG+03], and various biological mechanisms exist in order to incorporate metal-containing groups in proteins [KH04]. Metallochaperones are metal-binding proteins which deliver the appropriate metal cofactor to the target ligand. They often contain flexible metal-bindings, which can possibly form disulphide bonds in the oxidized apo (metal-free) form, as has been observed in vitro for Atx1 [RHH+99]. Cytochrome *c* in *E.coli* binds a heme group by covalent bonds between two cysteine residues in a `CXXCH` motif (where `X` stands for any residue) and two vinyl groups in the heme. However, when the early synthesized apoprotein emerges into the oxidizing environment of the periplasm, dsbA oxidizes the two cysteines which form a disulphide bridge (see fig. 5.4), successively reduced in order to allow them to react with the heme. Therefore, the presence of a metal group binding site involving a cysteine does not imply that the cysteine cannot be involved in a disulphide bridge in the apo form of the protein.

In some well known cases, the presence of a binding site for a prosthetic groups can be detected by inspecting the consensus pattern that matches the portion of the protein sequence containing the target cysteine. For example the 4Fe-4S ferredoxin group is associated with the pattern `C-X(2)-C-X(2)-C-X(3)-C-[PEG]` [OO87]. Many of these patterns are highly specific but there are well known examples of false positives. For example, the `C-{CPWHF}-{CPWR}-C-H-{CFYW}` consensus pattern is often correctly associated with a cytochrome c family heme binding site [Mat85], but the rate of false positives for this pattern is very high. In addition, there are cases of cysteines involved in a MBS and having no associated pattern.

To our knowledge, no predictive method has been proposed to directly address the problem of discriminating between ligand bonded and disulphide bonded cysteines. The methods and results presented in this chapter are based on [?]. We formulate the prediction task as a binary classification problem: given a non-free cysteine and information about flanking residues, predict whether the cysteine can bind to a prosthetic group containing a metal ion (positive class) or it is always bound to another cysteine forming a disulphide bridge (negative class).

Firstly, we suggest a nontrivial baseline predictor based on PROSITE [FPB+02] pattern hits and the induction of decision trees using the program C4.5 [Qui93]. Secondly, we introduce a classifier fed by multiple alignment profiles and based on support vector machines. We show that the latter classifier is capable of discovering the large majority of the relevant PROSITE patterns, but is also sensitive to signal in the profile sequence that cannot be detected by regular expressions and therefore outperforms the baseline predictor.

6.1 Data Preparation

The data for cysteines involved in disulphide bridge formation were extracted from PDB [BBB+02]. We excluded chains if:

- the protein was shorter than 30 residues;
- it had less than two cysteines;
- it had cysteines marked as unresolved residues in the PDB file;

The data for metal binding sites were extracted from Swiss-Prot version 41.23 [BBA+03], since PDB does not contain enough examples of metal ligands. In this case we included all entries containing at least one cysteine involved in a metal binding, regardless of the annotation confidence. In this way examples of bindings with iron-sulfur clusters (2FE2S,3FE4S,4FE4S), copper, heme groups, iron, manganese, mercury, nickel and zinc were obtained.

Intra-set redundancy due to sequence similarity was avoided by running the UniqueProt program [MR03] with hssp distance set to zero. Inter-set redundancy however was kept in order to handle proteins with both disulphide bridges and metal bindings. It must be remarked that while inter-set redundancy can help the learning algorithm by providing additional data for training, it cannot favorably bias accuracy estimation since redundant cases should be assigned to opposite classes. The number of non-homologous sequences remaining in the datasets are shown in table 6.1, together with the number of cysteines involved in disulphide bridges or metal bindings. Free cysteines were ignored.

	# Sequences	# Cysteines
Disulphide Bridges	529	2860
Metal Bindings	202	758

Table 6.1. Non homologous sequences obtained by running *uniqueprot* [MR03] with hssp distance set to zero. Sequences containing disulphide bridges were obtained from resolved proteins in PDB [BBB+02], while those with metal bindings were recovered from Swiss-Prot version 41.23 [BBA+03] by keyword matching.

6.2 PROSITE Patterns as a Baseline

In this section we establish a procedure to compute a baseline accuracy for the addressed prediction task. In general, the base accuracy of a binary classifier is the frequency of the most common class. For the data set described above the base accuracy is 84.4%. In total absence of prior knowledge a predictor that performs better than the base accuracy is generally considered as successful. However:

- it must be remarked that, especially for highly unbalanced datasets, precision/recall rates are also needed in order to have a correct view of the classifier performance.
- for the specific task under investigation, several well known consensus patterns exist that are associated with disulphide bridges and with metal binding sites. These patterns partially encode expert knowledge and it seems reasonable, when possible, to make use of them as a rudimentary prediction tool.

Thus, in order to compare our prediction method with respect to a more interesting baseline than the mere base accuracy, we extracted features that consist of PROSITE [FPB+02] pattern hits.

A PROSITE pattern is an annotated regular expression that describes a relatively short portion of a protein sequence that has a biological meaning. We run the program `ScanProsite` [GGB02] on the data set described above searching for patterns listed in the release 18.18 (December 2003) of PROSITE. In this way we found 199 patterns whose matches with the sequences in our data set contain the position of at least one bound cysteine. About 56% of the cysteines bound to a metal group and about 41% of the cysteines forming disulphide bridges matched at least one PROSITE pattern. Many of the patterns associated with DB have perfect (100%) specificity but each one only covers a very small set of cases. Overall, the fraction of disulphide-bond cysteines matched by a perfectly specific pattern is about 26%. Patterns associated with MBS have a significantly higher coverage, although their specificity is perfect only about 18% of the times and sometimes is low. We remark that in our context a metal binding pattern is perfectly specific if every match is actually a metal binding site, regardless of the metal group involved in the bond. Thus, examples of perfectly specific patterns associated with MBS include PS00198 (4Fe-4S ferredoxins iron-sulfur binding region), PS00190 (Cytochrome c family heme-binding site), and PS00463 (Fungal Zn(2)-Cys(6) binuclear cluster domain). To further complicate the scenario, 12% of the bound cysteines match more than one pattern. For these reasons, a prediction rule based on pattern matches is difficult to craft by hand and we used the program C4.5 to create rules automatically from data. C4.5 induces a decision trees from labeled examples by recursively partitioning the instance space, using a greedy heuristic driven by information theoretic considerations [Qui86]. Rules are then obtained by visiting the tree from the root to a leaf. When using C4.5, each bound cysteine was simply represented by the bag of its matching patterns.

6.3 Prediction by Support Vector Machines

The SVM algorithm is capable of handling extremely numerous and sparse features, thus allowing us to exploit a wide local context surrounding the cysteine under investigation. In particular, we provided information in the form of a symmetric window of $2k+1$ residues, centered around the target cysteine, with k varying from 1 to 25. In order to include evolutionary information, we coded each element of the window by its multiple alignment profile computed with PSIBLAST [AMS$^+$97].

Preliminary model selection experiments were conducted in order to choose the appropriate kernel, together with its hyperparameters. In subsequent experiments we employed third degree polynomial kernels, with offset equal to one, fixed regularization parameter given by the inverse of the average of $K(x, x)$ with respect to the training set, and distinct penalties [Joa98a] for errors on positive or negative examples, in order to rebalance the different proportion of examples in the two classes (see table 6.1).

Different amino acid exchanges within sequences are more or less neutral in terms of structural and functional preservation. This observation led to the creation of various exchange matrices which are commonly employed in sequence alignment algorithms. Such matrices can be directly derived from chemico-physical properties of the amino acids, as for the McLachlan similarity matrix [McL72], or extracted from sequence databases, as for the Blosum substitution matrices [HH92]. These are based on log odds ratios of observed to expected alignment probabilities between pairs of residues, and are derived from blocks of aligned sequence segments. Different identity percentages in clustering sequences within a block produce different matrices, and the Blosum62 matrix showed the best results in sequence alignment [HH93].

In order to include this similarity information in the learning algorithm, we implemented a new kernel in which the dot product between x_i and x_j is mediated by a matrix M:

$$K(x_i, x_j) = (x_i^T M x_j + 1)^3. \tag{6.1}$$

Note that the standard polynomial kernel amounts at choosing M to be the identity matrix. In order for equation (6.1) to be a valid kernel, matrix M has to be symmetric positive definite (see section 2.4.1). We tried the McLachlan matrix, , which already satisfies such condition, and the Blosum62 one, that turned out to be positive definite after normalizing each element as $(M_{rc} - min)/(max - min)$ where max and min are computed over the entire matrix. A similar approach [GLV04] was recently applied to secondary structure prediction (see section 4.3.3.1).

6.4 Results and Discussion

Test performances were calculated by three fold *cross validation*: proteins were divided in three groups, maintaining in each group approximately the same distribution of disulphide bridges and different kinds of metal binding sites.

	Precision (%)	Recall (%)	Bridge	Metal	
Bridge	84	99	2845	15	True
Metal	93	27	556	202	
Accuracy	84.2		Predicted		

Table 6.2. Disulphide bridge vs metal binding prediction by decision rules learned by c4.5 from patterns extracted from PROSITE. Precision and recall for both classes, confusion matrix and overall accuracy.

The confusion matrix for PROSITE induced rules is shown in table 6.2. It must be observed that the accuracy in table 6.2 is an optimistic upper bound of the true predictive power of this method with respect to future sequences. This is because PROSITE patterns have been designed to minimize the number of false hits and missed hits by actually inspecting the available sequence databases.

Results for the polynomial kernel are reported in figure 6.1. Train and test accuracies are plotted for growing size of the context window, with error bars for 95% confidence intervals, together to the fraction of support vectors over the train examples in the learned

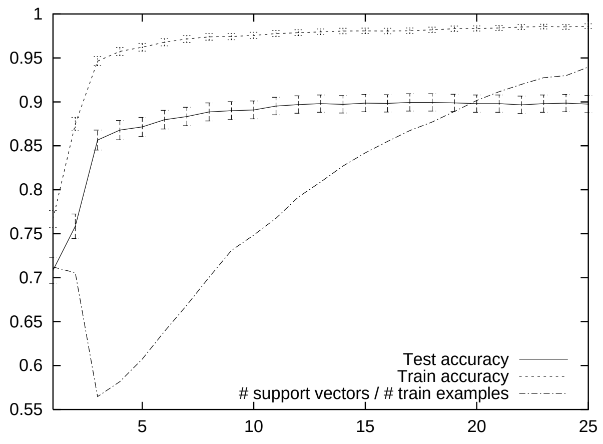


Figure 6.1. Disulphide bridge vs metal binding prediction by SVM with 3^{rd} degree polynomial kernel. Test and train accuracies with 95% confidence intervals are plotted, together to the fraction of support vectors over the number of training examples, for growing sizes of the window of $2k+1$ residues profiles around the target cysteine, with k going from 1 to 25. Results are averaged over a three fold cross validation procedure.

	Precision (%)	Recall (%)	Bridge	Metal	
Bridge	91	90	2588	272	True
Metal	65	67	247	511	
Accuracy	86		Predicted		

Table 6.3. Disulphide bridge vs metal binding prediction by 3^{rd} degree polynomial kernel SVM. Window of 3 residues profiles on both sides of the target cysteine. Precision and recall for both classes, confusion matrix and overall accuracy.

models, which roughly indicates the complexity of the learned models and is a loose upper bound on the leave one out error (see sec. 2.2.4). The most evident improvement in test accuracy is obtained for a window of size k=3, and corresponds to the global minimum in the model complexity curve with about 56% of training examples as support vectors. Detailed results for such window are reported in table 6.3, showing they are obtained for an approximate break-even point in the precision recall curve. A deeper analysis of individual predictions showed that the vast majority of predictions were driven by the presence of a well conserved `CXXC` pattern, taken as the indicator of a metal binding. This explains the high rate of false MBS compared to the total number of MBS examples, being most of them cysteines containing the pattern but involved in disulphide bridges, while most of the false DB are metal bindings missing it. The learned pattern is actually very common for most bindings involving iron-sulfur, iron-nickel and heme groups, and these kinds of metal binding are actually predicted with the highest recall.

The best accuracy is obtained for a window of size k=17, corresponding to about 87% of examples as support vectors. Detailed results are reported in tables 6.4 and 6.5, showing a strong reduction of false MBS at the cost of a slight increase in the number of metal

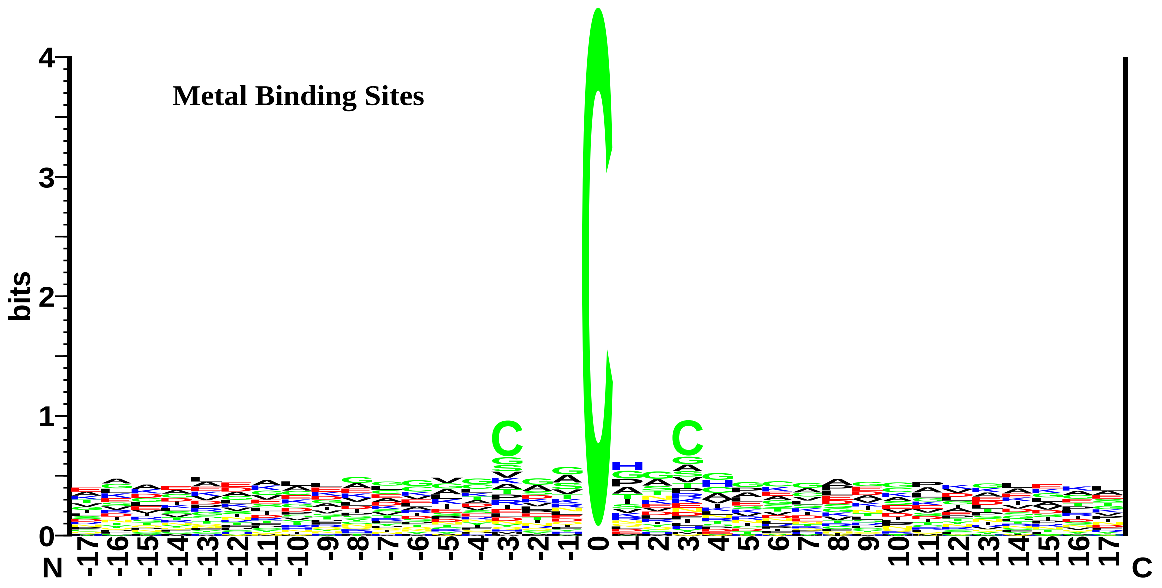
	Precision (%)	Recall (%)	Bridge	Metal	
Bridge	91	97	2788	72	True
Metal	87	61	292	466	
Accuracy	90		Predicted		

Table 6.4. Disulphide bridge vs metal binding prediction by 3^{rd} degree polynomial kernel SVM. Window of 17 residues profiles on both sides of the target cysteine. Precision and recall for both classes, confusion matrix and overall accuracy.

Ligand	**# examples**	**Recall** (%)
Cysteine	2860	97.5
Metal	758	61.4
4FE4S	250	71.6
Zinc	156	38.5
Heme	139	87.8
2FE2S	91	58.2
Copper	50	38.0
Iron	26	42.3
3FE4S	25	48.0
Nickel	13	76.9
Mercury	7	00.0
Manganese	1	00.0

Table 6.5. Disulphide bridge vs metal binding prediction by 3^{rd} degree polynomial kernel SVM. Window of 17 residues profiles on both sides of the target cysteine. Recall and number of examples for cysteine (disulphide bridge) or metal ligand, and details for different kinds of metal binding.

bindings predicted as disulphide bridges. Sequence logos [SS90] for cysteine context in the case of metal bindings (figure 6.2(upper)) show a well conserved `CXXCXXC` pattern, which is common in 4FE4S clusters, and the `CXXCH` pattern typical of heme groups, but also the presence of `CG` patterns in positions as distant as nine residues from the target cysteine. Disulphide bridge contexts are much more uniform (see figure 6.2(lower)), but show a strong tendency for polar aminos, especially glycine, cysteine and serine all along the window. This finding agrees with previous observations for smaller sizes of the cysteine context [FRC99], as well as with observations relating bridges to secondary structure [FS00], showing that bridges tend to connect coil regions together or a coil region to a more regular region such as an alpha helix or a beta strand.



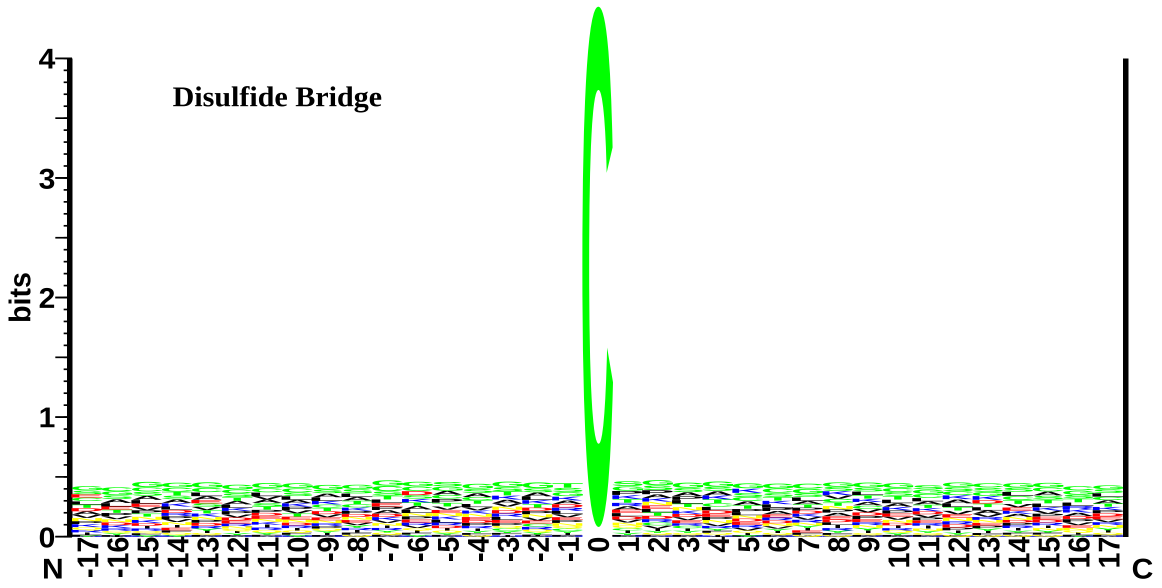
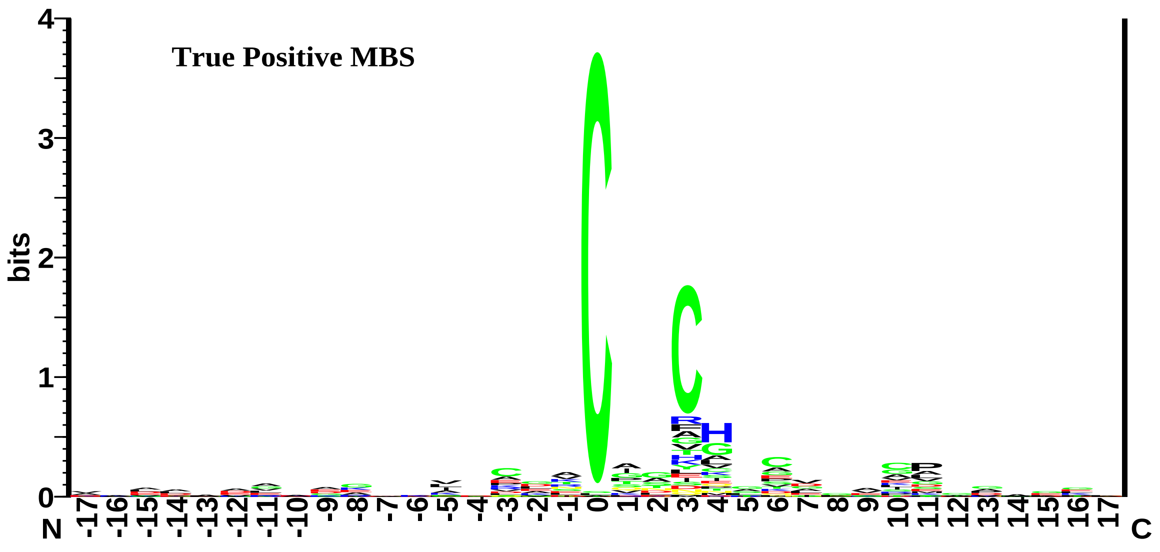


Figure 6.2. Sequence logos [SS90] of context of cysteines involved in metal bindings (upper) and disulphide bridges (lower) respectively, with a window of 17 residues on each side of the bond cysteine. Hydrophobic residues are shown in black, positively charged residues are blue and negatively charged are red, while uncharged polar residues are green.



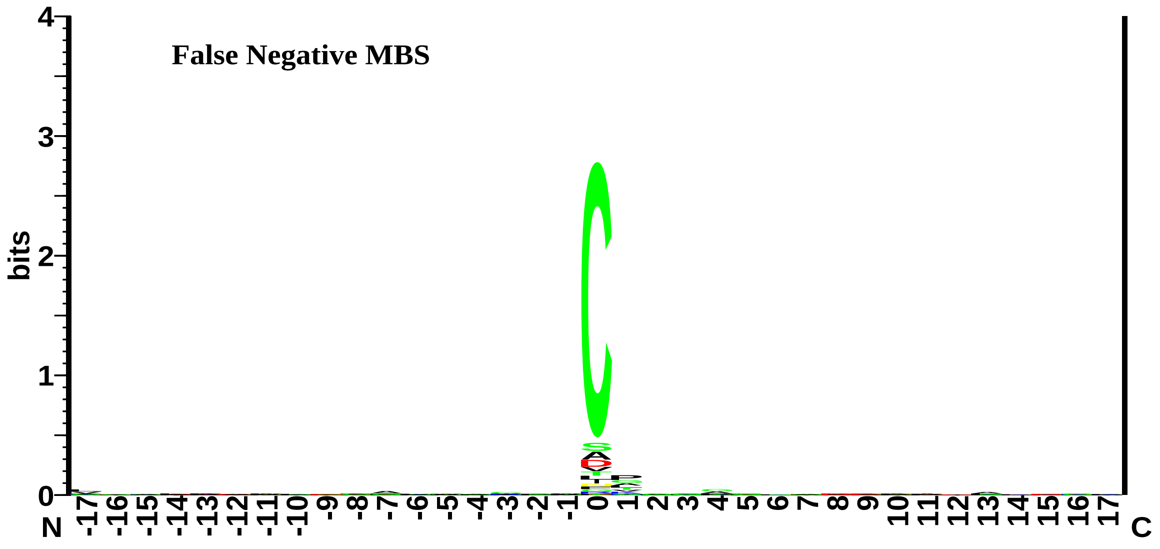


Figure 6.3. Sequence logos [SS90] of 17 aminos context of cysteines that don't match any PROSITE pattern, and are either truly predicted as MBS (upper) or mistakenly predicted as DB (lower) by an SVM with 3^{rd} degree polynomial kernel. Hydrophobic residues are shown in black, positively charged residues are blue and negatively charged are red, while uncharged polar residues are green.

The model seems capable of exploiting very distant information, and it discovers almost all rules induced by PROSITE patterns (see table 6.2), as only 13 over 202 metal bindings are lost, while it also corrects 8 over 15 false metal bindings. Moreover, the SVM is capable of discovering metal bindings sites where no pattern is available. In order to get some insights on the reasons for these predictions, we collected all the MBS that do not contain any pattern, and divided them into examples actually predicted as MBS by the SVM (true positives) and examples wrongly predicted as DB (false negatives). Figures 6.3(upper) and 6.3(lower) represent sequence logos for true positives and false negatives respectively, where the logos are computed using the average of the PSI-BLAST profiles of all the examples. Part of the correct predictions could be explained by the ability of the SVM to actually recover continuous-valued patterns in the profiles. We conjecture that in some other cases the predictor could have discovered potential consensus patterns that are still not known.

Reported results are quite robust with respect to model regularization, controlled by the cost parameter "C" in SVM. Figure 6.4 shows a test accuracy maximization obtained by varying the regularization parameter: accuracies remain mostly within the confidence interval from the maximum.

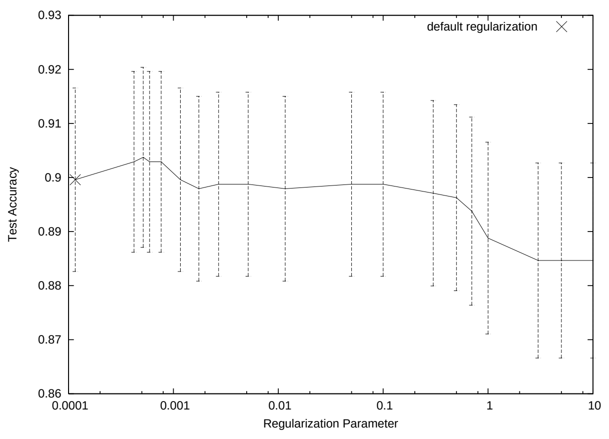


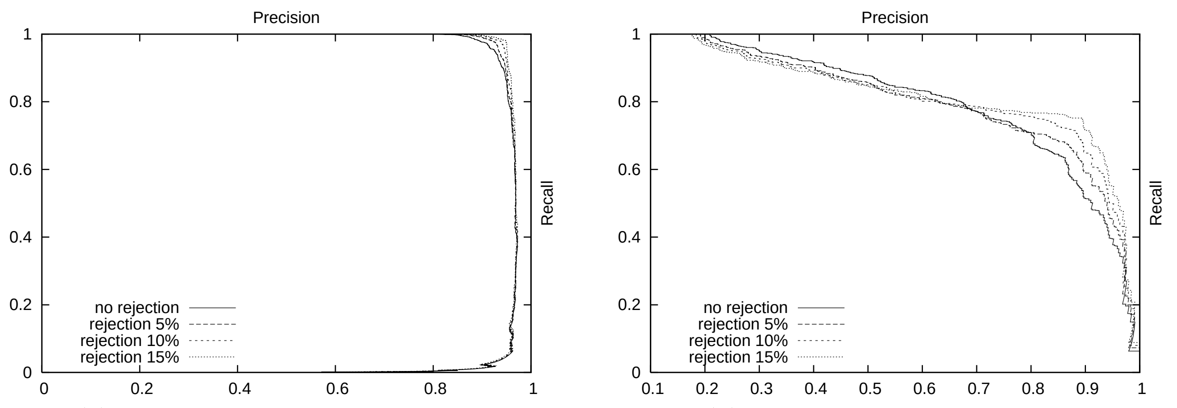
Figure 6.4. Disulphide bridge vs metal binding prediction by 3^{rd} degree polynomial kernel SVM. Window of 17 residues profiles on both sides of the target cysteine. Test accuracies with 95% confidence intervals are plotted versus "C" regularization parameter of SVM. Default regularization is computed as the inverse of the average of $K(x, x)$ with respect to the training set.

Table 6.6 shows test accuracies for different rejection rates, when predictions are to be made only if they are over a certain confidence. A rejection of 15% of both positive and negative test examples results in about 94% test accuracy, and predictions on disulphide bridges tend to be much more confident than those on metal bindings, having a higher threshold for equal rejection rate. Furthermore, metal bindings are rejected more frequently with respect to their total number in the test set. A finer analysis shows that rejected metal bindings are mostly those for which the model has low recall (see table 6.5), such as 3FE4S, iron, mercury and zinc, while 4FE4S, heme and nickel are seldom rejected.

Figures 6.5(a) and 6.5(b) show precision/recall curves for disulphide bridges and metal

	Threshold			Bridge		Metal		Rejected (%)	
Rejection (%)	Bridge	Metal	Acc	Pre	Rec	Pre	Rec	Bridge	Metal
00	0	0	89.9	90.5	97.5	86.6	61.5	0.0	0.0
05	0.20845	0.0470436	91.8	92.7	97.6	86.9	67.4	02.8	12.9
10	0.340927	0.113824	93.0	93.8	97.9	88.5	71.5	07.1	20.8
15	0.436391	0.159688	94.0	94.8	98.1	89.3	75.2	11.4	28.2
20	0.506477	0.225416	94.4	95.2	98.2	90.3	76.6	16.5	33.0
25	0.56122	0.267722	95.2	95.9	98.4	91.1	79.5	21.3	38.8
30	0.609002	0.337221	95.7	96.1	98.8	93.4	80.8	26.6	42.3

Table 6.6. Disulphide bridge vs metal binding prediction by 3^{rd} degree polynomial kernel SVM. Window of 17 residues profiles on both sides of the target cysteine. Performances for different rejection rates, where rejection percentage is computed separately for examples predicted as disulphide bridges or metal bindings, in order to consider the unbalanced distribution of examples between the two classes (i.e. 5% rejection rate indicates that 5% of examples predicted as disulphide bridges are to be rejected, as well as 5% of examples predicted as metal bindings). Reported results include rejection thresholds, accuracies, precision and recall, and percentage of rejected examples belonging to each of the two classes.

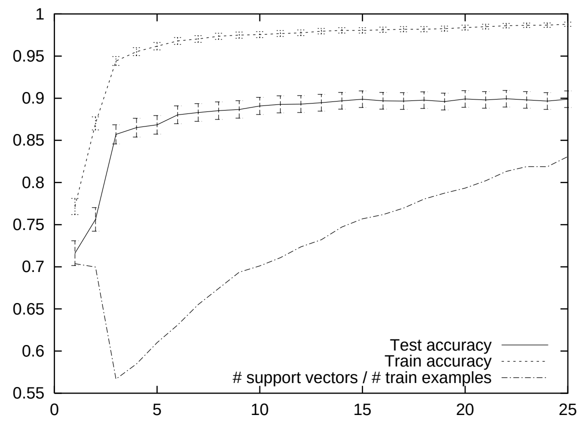


(a) Disulphide bridge prediction (b) Metal binding prediction

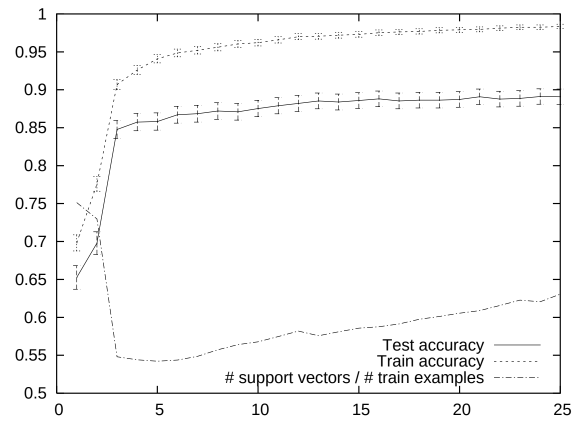
Figure 6.5. Disulphide bridge vs metal binding prediction by 3^{rd} degree polynomial kernel SVM. Window of 17 residues profiles on both sides of the target cysteine. Precision/recall curves over the test set for different rejection rates for disulphide bridge (a) and metal binding (b) prediction.

bindings respectively, for different rejection rates. The former tends slightly towards the optimal curve at growing rejection rates, while the latter has a complementary behaviour with respect to the break-even point: at higher precisions growing rejection rates result in better performance, while at lower precisions performance for growing rejection rates is affected by the greater quantity of metal bindings rejected with respect to disulphide bridges.

Figure 6.6(a) shows results for growing size of the context window, for a third de-



(a) Polynomial Kernel with McLachlan Matrix



(b) Polynomial Kernel with Blosum62 Matrix

Figure 6.6. Disulphide bridge vs metal binding prediction by SVM with 3^{rd} degree polynomial kernel with McLachlan (a) or Blosum62 (b) similarity matrix. Test and train accuracies with 95% confidence intervals are plotted, together to the fraction of support vectors over the number of training examples, for growing sizes of the window of $2k{+}1$ residues profiles around the target cysteine, with k going from 1 to 25. Results are averaged over a three fold cross validation procedure.

gree polynomial kernel with McLachlan similarity matrix (eq. 6.1). While train and test accuracies are similar to those obtained without the similarity matrix (figure 6.1), the corresponding models have less support vectors, with reductions up to 11% of the training set. This behaviour is even more evident for the Blosum62 substitution matrix (figure 6.6(b)) where a slight decrease in test accuracy, still within the confidence interval, corresponds to a reduction up to 30% of the training set. Note that the fraction of support vectors over training examples is a loose upper bound on the leave one out error (see sec. 2.2.4), which is an almost unbiased estimate of the true generalization error of the learning algorithm (see sec. 2.1.4) . These kernels are able to better exploit information on residue similarity, thus obtaining similar performances with simpler models. Moreover, the precision/recall rate of the kernel with the Blosum62 matrix is much more balanced with respect to the other two, meaning that it suffers less from the unbalancing in the training set.

6.5 Conclusions

Recent researches have shown that cysteines can undergo numerous post-translational modifications, apart from the most common thiol and disulfide oxidation states (see [GWG+03] for a review). In particular, cysteines are well suited to bind various types of ligands, usually containing metal ions [KH04]. Learning to discriminate between ligand bond and disulfide bond cysteines is an important step in understanding the function and structure of a given protein, and can help NMR structural determination of metalloproteins as well as disulfide bridges prediction, when combined with algorithms predicting subcellular localization of proteins.

We have proposed learning algorithms for predicting the type of binding in which non-free cysteines are involved. The experimental results indicate that learning from multiple alignment profiles data outperforms even non-trivial approaches based on pattern matching, suggesting that relevant information is contained not only in the residues flanking the cysteine but also in their conservation. In some cases the learning algorithm could be actually exploiting in a probabilistic way patterns that are not yet listed in PROSITE, although detecting them with few data points is difficult. In addition, we found that using ad hoc kernels exploiting residue similarity matrices effectively reduces the complexity of the model, measured by the number of support vectors, which is also an indication of the expected generalization error. We expect that similar kernel functions could be also useful for other predictive tasks both in 1D (e.g. solvent accessibility) and in 2D (e.g. contact maps), especially if task-dependent similarity matrices could be devised.

Bibliography

[AGM$^+$90] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. A basic local alignment search tool. *J Mol. Biol.*, 215:403–410, 1990.

[AHSW61] C.B. Anfinsen, E. Haber, M. Sela, and F.W Jr White. The kinetics of the formation of native ribonuclease during oxidation of the reduced polypeptide domain. *Proc. Natl. Acad. Sci. USA*, 47:1309–1314, 1961.

[Aka73] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Proc. Second International Symposium on Information Theory*, pages 267–281, 1973.

[ALBK87] D. Altschuh, A.M. Lesk, A.C. Bloomer, and A. Klug. Correlation of co-ordinated amino acid substitutions with function in viruses related to tobacco mosaic virus. *J Mol. Biol.*, 193:693–707, 1987.

[Ama90] S.I. Amari. Mathematical foundations of neurocomputing. *Proceedings of the IEEE*, 78(9):1443–1463, 1990.

[Ama98] S.I. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.

[AMS$^+$97] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–3402, 1997.

[Anf73] C.B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181:223–230, 1973.

[Aro50] N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 686:337–404, 1950.

[ASS00] E. L. Allwein, R.E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proc. 17th International Conf. on Machine Learning*, pages 9–16. Morgan Kaufmann, San Francisco, CA, 2000.

[BB99] E.J. Bredensteiner and K.P. Bennet. Multicategory classification by support vector machines. *Computational Optimizations and Applications*, 12:53–79, 1999.

[BBA$^+$03] B. Boeckmann, A. Bairoch, R. Apweiler, M. C. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The swiss-prot protein knowledgebase and its supplement trembl in 2003. *Nucleic Acids Res*, 31(1):365–370, 2003.

[BBB$^+$02] H. M. Berman, T. Battistuz, T. N. Bhat, W. F. Bluhm, P. E. Bourne, K. Burkhardt, Z. Feng, G. L. Gilliland, L. Iype, S. Jain, P. Fagan, J. Marvin, D. Padilla, V. Ravichandran, B. Schneider, N. Thanki, H. Weissig, J. D. Westbrook, and C. Zardecki. The protein data bank. *Acta Cryst.*, D58:899–907, 2002.

[BBB$^+$03] E. Balatri, L. Banci, I. Bertini, F. Cantini, and S. Ciofi-Baffoni. Solution structure of sco1: a thioredoxin-like protein involved in cytochrome c oxidase assembly. *Structure*, 11(11):1431–1443, November 2003.

[BBF$^+$99] P. Baldi, S. Brunak, P. Frasconi, G. Soda, and G. Pollastri. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15(11):937–946, 1999.

[BCR84] C. Berg, J.P.R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag, New York, 1984.

[BE02] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

[Ben00] Y. Bengio. Gradient-based optimization of hyper-parameters. *Neural Computation*, 12(8):1889–1900, 2000.

[Bet93] S.F. Betz. Disulfide bonds and the stability of globular proteins. *Protein Sci.*, 1993.

[BFGK96] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in bayesian networks. In *Prof. 12th Conf. on Uncertainty in Artificial Intelligence*, pages 115–123. Morgan Kaufmann, 1996.

[BGV92] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifier. In *Proc. 5th ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992.

[BHB03] A. Ben-Hur and D. Brutlag. Remote homology detection: a motif based approach. *Bioinformatics*, 19:26–33, 2003.

[BHHSV01] A. Ben-Hur, D. Horn, H.T. Siegelmann, and V.N. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[Bis95] C.M. Bishop. *Neural networks for pattern recognition.* Oxford University Press, Oxford, 1995.

[BKML$^+$04] D.A. Benson, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, and D.L. Wheele. Genbank: update. *Nucleic Acids Res.*, 32(Database issue):D23–D26, 2004.

[BM92] K.P. Bennett and O.L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[BM98] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[BRC60] R.C. Bose and D.K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3:68–79, March 1960.

[Bri89] J. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fogelman-Soulie and J. Hérault, editors, *Neuro-computing: Algorithms, Architectures, and Applications*. Springer-Verlag, 1989.

[BSB01] R. Bonneau, C.E. Strauss, and D. Baker. Improving the performance of rosetta using multiple sequence alignment information and global measures of hydrophobic core formation. *Proteins*, 43(1):1–1, 2001.

[BST99] P.L. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Schölkopf, C Burges, and A.J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 43–54. MIT Press, Cambridge, MA, 1999.

[BT99] C. Branden and J. Tooze. *Introduction to Protein Structure.* Garland Publishing, New York, 1999.

[Bur89] P. Burman. A comparative study of ordinary cross validation, 5-fold cross validation, and the repeated learning testing methods. *Biometrica*, 76(3):503–514, 1989.

[Bur98a] C. Burges. A tutorial on support vector machines for pattern recognition. In *Data Mining and Knowledge Discovery.* Kluwer Academic Publishers, Boston, 1998. (Volume 2).

[Bur98b] C. Burges. A tutorial on support vector machines for pattern recognition. In *Data Mining and Knowledge Discovery.* Kluwer Academic Publishers, Boston, 1998. (Volume 2).

[CB99] J.A. Cuff and G.J. Barton. valuation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins*, 34(4):508–519, 1999.

[CBB02] R. Collobert, S. Bengio, and Y. Bengio. A parallel mixture of SVMs for very large scale problems. *Neural Computation*, 14(5), 2002.

[CCST99] N. Cristianini, C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 11. MIT Press, 1999.

[CD02a] M. Collins and N. Duffy. Convolution kernels for natural language. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[CD02b] M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 263–270, Philadelphia, PA, USA, 2002.

[CE99] K.-C. Chou and D.W. Elrod. Prediction of membrane protein types and subcellular locations. *Proteins*, 34:137–153, 1999.

[CFPV03a] A. Ceroni, P. Frasconi, A. Passerini, and A. Vullo. A combination of support vector machines and bidirectional recurrent neural networks for protein secondary structure prediction. In A. Cappelli and F. Turini, editors, *AI*IA 2003: Advances in Artificial Intelligence*, pages 142–153, 2003.

[CFPV03b] A. Ceroni, P. Frasconi, A. Passerini, and A. Vullo. Predicting the disulfide bonding state of cysteines with combinations of kernel machines. *Journal of VLSI Signal Processing*, 35(3):287–295, 2003.

[CFPV04] A. Ceroni, P. Frasconi, A. Passerini, and A. Vullo. Cysteine bonding state: Local prediction and global refinment using a combination of kernel machines and bi-directional recurrent neural networks. In preparation, 2004.

[Chi00] Y.V. Chinenov. Cytochrome c oxidase assembly factors with a thioredoxin fold are conserved among prokaryotes and eukaryotes. *J. Mol. Med.*, pages 239–242, 2000.

[Con01] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 15 February 2001.

[CR02] C.P. Chen and B. Rost. State-of-the-art in membrane protein prediction. *Applied Bioinformatics*, 1(1):21–35, 2002.

[CRBB03] D. Chivian, T. Robertson, R. Bonneau, and D. Baker. Ab initio methods. In *Structural Bioinformatics*, pages 547–557. Wiley-Liss, 2003.

[CS00] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learning Theory*, pages 35–46, 2000.

[CS01] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin (New Series) of the American Mathematical Society*, 39(1):1–49, 2001.

[CS02a] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, 2002.

[CS02b] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2–3):201–233, 2002.

[CST00] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.

[CSTEK02] N. Cristianini, J. Shawe-Taylor, A. Elisseef, and J. Kandola. On kernel-target alignment. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*. MIT Press, Cambridge, MA, 2002.

[CV95] C. Cortes and V.N. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.

[CVBM02] O. Chapelle, V.N. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1–3):131–159, 2002.

[CW79] P. Craven and G. Wahba. Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross validation. *Numer. Math*, 31:377–403, 1979.

[DB95] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[DEKM98] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998.

[Dre94] J. Drenth. *Principles of protein x-ray cristallography.* Springer-Verlag, Berlin, 1994.

[ea89] M. Matsumura et al. Substantial increase in protein stability by multiple disulfide bonds. *Nature*, 342:291–293, 1989.

[Edd98] S.R. Eddy. Profile hidden markov models. *Bioinformatics*, 14(9):755–763, 1998.

[EGPM99] A. Elisseeff, Y. Guermeur, and H. Paugam-Mousy. Margin error and generalization capabilities of multi-class discriminant systems. Technical Report NC2-TR-1999-051, NeuroCOLT2 Technical Report, 1999.

[ENBvH00] O. Emanuelsson, H. Nielsen, S. Brunak, and G. von Heijne. Predicting subcellular localization of proteins based on their n-terminal amino acid sequence. *J Mol. Biol.*, 300:1005–1016, 2000.

[EP03] A. Elisseeff and M. Pontil. Leave-one-out error and stability of learning algorithms with applications. In J.A.K. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, editors, *Advances in Learning Theory: Methods, Models and Applications*, volume 190 of *NATO Science Series III: Computer & Systems Sciences*, pages 111–130. IOS Press Amsterdam, 2003.

[EPP00] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

[F0̈2] J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

[FBL$^+$95] H. Flöckner, M. Braxenthaler, P. Lackner, M. Jaritz, M. Ortner, and M.J. Sippl. Progress in fold recognition. *Proteins*, 23(3):376–386, 1995.

[FC99] P. Fariselli and R. Casadio. A neural network based predictor of residue contacts in proteins. *Protein Eng.*, 12(1):15–21, 1999.

[FC00] P. Fariselli and R. Casadio. Prediction of the number of residue contacts in proteins. In *Proc. Int. Conf. Intell. Syst. Mol. Biol. (ISMB00)*, volume 8, pages 146–151, La Jolla, CA, 2000. AAAI Press.

[FC01] P. Fariselli and R. Casadio. Prediction of disulfide connectivity in proteins. *Bioinformatics*, 17:957–964, 2001.

[FCC98] T.-T. Frieß, N. Cristianini, and C. Campbell. The kernel-adatron algorithm: a fast and simple learning procedure for support vector machines. In *Proc. of the 15th Int. Conf. on Machine Learning.* Morgan Kaufmann, 1998.

[FG03] D.E. Fomenko and V.M. Gladyshev. Genomics perspective on disulfide bond formation. *Antioxid. Redox Signal.*, 5(4):397–402, 2003.

[FGS98] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE TNN*, 9(5):768–786, 1998.

[FH00] T. Finkel and N.J. Holbrook. Oxidants, oxidative stress and the biology of ageing. *Nature*, 408(6809):239–247, 2000.

[Fle87] R. Fletcher. *Practical Methods of Optimization, Second Edition.* John Wiley & Sons, 1987.

[FMC02] P. Fariselli, P. Martelli, and R. Casadio. A neural network based method for predicting the disulfide connectivity in proteins. In E. Damiani et al., editor, *Knowledge based intelligent information engineering systems and allied technologies (KES 2002)*, volume 1, pages 464–468. IOS Press, 2002.

[FOVC01] P. Fariselli, O. Olmea, A. Valencia, and R. Casadio. Prediction of contact maps with neural networks and correlated mutations. *Protein Eng.*, 14(11):835–843, 2001.

[FPB$^+$02] L. Falquet, M. Pagni, P. Bucher, N. Hulo, C.J.A. Sigrist, K. Hofmann, and A. Bairoch. The PROSITE database, its status in 2002. *Nucleic Acids Res.*, 30(1):235–238, 2002.

[FPV02] P. Frasconi, A. Passerini, and A. Vullo. A two-stage SVM architecture for predicting the disulfide bonding state of cysteines. In *Proc. of the IEEE Workshop on Neural Networks for Signal Processing*, 2002.

[FRC99] P. Fariselli, P. Riccobelli, and R. Casadio. Role of evolutionary information in predicting the disulfide-bonding state of cysteine in proteins. *Proteins*, 36, 340–346 1999.

[Fri96] J.H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1996.

[FS97] Y. Freund and R. Shapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[FS98] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. In *Computational Learning Theory*, pages 209–217, 1998.

[FS00] A. Fiser and I. Simon. Predicting the oxidation state of cysteines by multiple sequence alignment. *Bioinformatics*, 16(3):251–256, 2000.

[FSS98] Y. Freund, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proc. of the* 15*th* *International Conference on Machine Learning*, San Francisco, 1998. Morgan Kaufmann.

[GÖ2] T. Gärtner. Exponential and geometric kernels for graphs. In *NIPS Workshop on Unreal Data: Principles of Modeling Nonvectorial Data*, 2002.

[GBD92] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[GBV93] I. Guyon, B. Boser, and V.N. Vapnik. Automatic capacity tuning of very large VC-dimension classifiers. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 147–155. Morgan Kaufmann, San Mateo, CA, 1993.

[Gen01] Celera Genomics. The human genome. *Science*, 291(5507), 16 February 2001.

[GEPM00] Y. Guermeur, A. Elisseeff, and H. Paugam-Mousy. A new multi-class svm based on a uniform convergence result. In *Proceedings of IJCNN - International Joint Conference on Neural Networks*, volume IV, pages 183–188. IEEE, 2000.

[GEZ02] Y. Guermeur, A. Elisseeff, and D. Zelus. Bounding the capacity measure of multi-class discriminant models. Technical Report NC2-TR-2002-123, Neuro-COLT2 Technical Report, 2002.

[GFKS02] T. Gärtner, P. Flach, A. Kowalczyk, and A.J. Smola. Multi-instance kernels. In C.Sammut and A. Hoffmann, editors, *Proceedings of the* 19^{th} *International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann, 2002.

[GGB02] A. Gattiker, E. Gasteiger, and A Bairoch. ScanProsite: a reference implementation of a PROSITE scanning tool. *Applied Bioinformatics*, 1:107–108, 2002.

[GLF03] T. Gärtner, J.W. Lloyd, and P.A. Flach. Kernels for structured data. In S. Matwin and C. Sammut, editors, *Proceedings of the 12th International Conference on Inductive Logic Programming*, volume 2583 of *Lecture Notes in Artificial Intelligence*, pages 66–83. Springer-Verlag, 2003.

[GLV04] Y. Guermeur, A. Lifchitz, and R. Vert. A kernel for protein secondary structure prediction. In B. Schölkopf, K. Tsuda, and J. P. Vert, editors, *Kernel Methods in Computational Biology.* The MIT Press, Cambridge, MA, 2004. In press.

[God03] A. Godzik. Fold recognition methods. In *Structural Bioinformatics*, pages 525–546. Wiley-Liss, 2003.

[GPE$^+$04] Y. Guermeur, G. Pollastri, A. Elisseeff, D. Zelus, H. Paugam-Moisy, and P. Baldi. Combining protein secondary structure prediction models with ensemble methods of optimal complexity. *Neurocomputing*, 56C:305–327, 2004.

[Gue02] Y. Guermeur. Combining discriminant models with new multi-class svms. *Pattern Analysis and Applications (PAA)*, 5(2):168–179, 2002.

[Gus97] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

[GVB+92] I. Guyon, V.N. Vapnik, B. Boser, L. Bottou, and S. Solla. Structural risk minimization for character recognition. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems IV*, San Mateo, CA, 1992. Morgan Kaufmann Publishers.

[GWBV02] I. Guyon, J. Weston, S. Barnhill, and V.N. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1–3):389–422, 2002.

[GWG+03] N.M. Giles, A.B. Watts, G.I. Giles, F.H. Fry, J.A. Littlechild, and C. Jacob. Metal and redox modulation of cysteine protein function. *Chemistry & Biology*, 10:677–693, 2003.

[Hau99] D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California, Santa Cruz, 1999.

[HGGW01] D.N. Heaton, G.N. George, G. Garrison, and D.R. Winge. The mitochondrial copper metallochaperone cox17 exists as an oligomeric, polycopper complex. *Biochemistry*, 40(3):743–751, 2001.

[HGW02] T. Herrmann, P. Güntert, and K. Wüthrich. Protein nmr structure determination with automated noe assignment using the new software candid and the torsion angle dynamics algorithm dyana. *J Mol. Biol.*, 319:209–227, 2002.

[HH91] S. Henikoff and J.G. Henikoff. Automated assembly of proteins blocks for database searching. *Nucleic Acids Res.*, 19(23):6565–6572, 1991.

[HH92] S. Henikoff and J.G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, 89:10915–10919, 1992.

[HH93] S. Henikoff and J.G. Henikoff. Performance evaluation of amino acid substitution matrices. *Proteins*, 17(1):49–61, 1993.

[HL02a] C. W. Hsu and C. J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, Mar 2002.

[HL02b] C.-W. Hsu and C.-J. Lin. A simple decomposition method for support vector machines. *Machine Learning*, 46:291–314, 2002.

[HMBC97] T. Hubbard, A. Murzin, S. Brenner, and C. Chothia. Scop: a structural classification of proteins database. *Nucleic Acids Res.*, 25(1):236–9, January 1997.

[HS94] U. Hobohm and C. Sander. Enlarged representative set of protein structures. *Protein Science*, 3:522–524, 1994.

[HS01] S. Hua and Z. Sun. A novel method of protein secondary structure prediction with high segment overlap measure: Support vector machine approach. *J Mol. Biol.*, 308(2):397–407, 2001.

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer-Verlag, 2001.

[JDH00] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1–2):95–114, 2000.

[JH98] T. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proc. of Neural Information Processing Conference*, 1998.

[JH99] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 487–493, Cambridge, MA, USA, 1999. MIT Press.

[JJNH91] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[JK94] R.L. Dunbrack Jr. and M. Karplus. Conformational analysis of the backbone-dependent rotamer preferences of protein sidechains. *Nature Struct. Biol.*, 1:334–340, 1994.

[Joa98a] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, chapter 11, pages 169–185. MIT Press, 1998.

[Joa98b] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, 1998.

[Joa99] T. Joachims. Estimating the generalization performance of a svm efficiently. Technical Report LS VIII Report 25, Universität Dortmund, 1999.

[Joa00] T. Joachims. Estimating the generalization performance of a SVM efficiently. In Pat Langley, editor, *Proceedings of the 17th International Conference on Machine Learning (ICML00)*, pages 431–438, Stanford, US, 2000. Morgan Kaufmann Publishers, San Francisco, US.

[Jon99] D.T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *J Mol. Biol.*, 292:195–202, 1999.

[KBH98] K. Karplus, C. Barrett, and R. Hughey. Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856, 1998.

[KH04] J. Kuchar and R.P. Hausinger. Biosynthesis of metal sites. *Chem. Rev.*, 104:509–525, 2004.

[KJ01] P. Kovacic and J.D. Jacintho. Mechanisms of carcinogenesis: focus on oxidative stress and electron transfer. *Curr Med Chem 2001; 8: 773*, 8:773–796, 2001.

[KKB$^+$01] K. Karplus, R. Karchin, C.Tu S. Barnett, M. Cline, M. Diekhans, L. Grate, J. Casper, and R. Hughey. What is the value added by human intervention in protein structure prediction? *Proteins*, 45(suppl 5):86–91, 2001.

[KKB03] H. Kadokura, F. Katzen, and J. Beckwith. Protein disulfide bond formation in prokaryotes. *Annu Rev Biochem.*, 72:111–135, 2003.

[KL02] R.I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In C.Sammut and A. Hoffmann, editors, *Proc. of the* 19^{th} *International Conference on Machine Learning*, pages 315–322. Morgan Kaufmann, 2002.

[KNV03] E. Krieger, S.B. Nabuurs, and G. Vriend. Homology modeling. In *Structural Bioinformatics*, pages 509–524. Wiley-Liss, 2003.

[Koh95] R. Kohavi. *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. PhD thesis, Computer Science department, Stanford University, 1995.

[Kon86] H. Konig. *Eigenvalue Distributions of Compact Operators*. Birkhauser, Basel, 1986.

[KR97] M. Kearns and D. Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross validation. In *Proceedings of the 10th Annual Conference on Computational Learning Theory*, pages 152–162, 1997.

[KS83a] W. Kabsch and C. Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22:2577–2637, 1983.

[KS83b] W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22:2577–2637, 1983.

[KW71] G. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.

[Kwo99] J. Kwok. Moderating the outputs of support vector machine classifiers. *IEEE Transactions on Neural Networks*, 10(5):1018–1031, 1999.

[LB67] A. Lunts and V. Brailovskiy. Evaluation of attributes obtained in statistical decision rules. *Engineering Cybernetics*, 3:98–109, 1967.

[LBD$^+$89] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L.J. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[LEN02] C. Leslie, E. Eskin, and W.S. Noble. The spectrum kernel: a string kernel for svm protein classification. In *Proc. of the Pacific Symposium on Biocomputing*, pages 564–575, 2002.

[Les01] A.M. Lesk. *Introduction to Protein Architecture*. Oxford University Press, New York, 2001.

[LEWN03] C. Leslie, E. Eskin, J. Weston, and W.S. Noble. Mismatch string kernels for svm protein classification. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1417–1424. MIT Press, Cambridge, MA, 2003.

[LJ03] K. Linke and U. Jakob. Not every disulfide lasts forever: Disulfide bond formation as a redox switch. *Antioxid. Redox Signal.*, 5(4):425–434, 2003.

[LKE04] C. Leslie, R. Kuang, and E. Eskin. Inexact matching string kernels for protein classificatio. In B. Schölkopf, K. Tsuda, and J.-P. Vert, editors, *Kernel Methods in Computational Biology*. MIT Press, 2004. In press.

[LMS$^+$01] B. Logan, P. Moreno, B. Suzek, Z. Weng, and S. Kasif. A study of remote homology detection. Technical report, Cambridge Research Laboratory, June 2001.

[LN03] L. Liao and W.S. Noble. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *Journal of Computational Biology*, 10(6):857–868, 2003.

[LRG86] J.M. Levin, B. Robson, and J. Garnier. An algorithm for secondary structure determination in proteins based on sequence similarity. *FEBS*, 205(2):303–308, 1986.

[LSTCW00] H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. In *Advances in Neural Information Processing Systems*, pages 563–569, 2000.

[Mat85] F.S. Mathews. The structure, function and evolution of cytochromes. *Prog. Biophys. Mol. Biol.*, 45(1):1–56, 1985.

[McL72] A.D. McLachlan. Repeating sequences and gene duplication in proteins. *J Mol. Biol.*, 64:417–437, 1972.

[MD96] L. Mirny and E. Domany. Protein fold recognition and dynamics in the space of contact maps. *Proteins*, 26(4):391–410, 1996.

[Mer09] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446, 1909.

[MFKC02a] P.L. Martelli, P. Fariselli, A. Krogh, and R. Casadio. A sequence-profile-based hmm for predicting and discriminating β barrel membrane proteins. *Bioinformatics*, 18:S46–S53, 2002.

[MFKC02b] P.L. Martelli, P. Fariselli, A. Krogh, and R. Casadio. A sequence-profile-based hmm for predicting and discriminating β barrel membrane proteins. *Bioinformatics*, 18:S46–S53, 2002.

[MFMC02] P.L. Martelli, P. Fariselli, L. Malaguti, and R. Casadio. Prediction of the disulfide-bonding state of cysteines in proteins at 88% accuracy. *Protein Sci.*, 11(2735–2739), 2002.

[MGHT02] M.H. Mucchielli-Giorgi, S. Hazout, and P. Tuffèry. Predicting the disulfide bonding state of cysteines using protein descriptors. *Proteins*, 46:243–249, 2002.

[Mit97] T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[MJ03] L.J. McGuffin and D.T. Jones. Benchworking secondary structure prediction for fold recognition. *Proteins*, 52:166–175, 2003.

[MM99] O.L. Mangasarian and D.R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 1999.

[Mov01] H. Naderi-Manesh M. Sadeghi S. Arab A.A. Moosavi Movahedi. Prediction of protein surface accessibility with information theory. *Proteins*, 42(4):452–459, 2001.

[MP92] D.C. Montgomery and E.A. Peck. *Introduction to Linear Regression Analysis*. John Wiley and Sons, Inc., 2nd edition edition, 1992.

[MR03] S. Mika and B. Rost. Uniqueprot: creating sequence-unique protein data sets. *Nucleic Acids Res.*, 31(13):3789–3791, 2003.

[NBvH99] H. Nielsen, S. Brunak, and G. von Heijne. Machine learning approaches for the prediction of signal peptides and other protein sorting signals. *Protein Eng.*, 12(1):3–9, 1999.

[Nil65] N.J. Nilsson. *Learning Machines.* McGraw-Hill, New York, 1965.

[Nil96] M. Nilges. Structure calculation from nmr data. *Curr. Opin. Str. Biol.*, 6:617–623, 1996.

[Nob04] W.S. Noble. Support vector machine applications in computational biology. In B. Schölkopf, K. Tsuda, and J.-P. Vert, editors, *Kernel Methods in Computational Biology.* MIT Press, 2004. In press.

[NR03a] R. Nair and B. Rost. Loc3d: annotate sub-cellular localization for protein structures. *Nucleic Acids Res.*, 31(13):3337–3340, 2003.

[NR03b] M.N. Nguyen and J.C. Rajapakse. Multi-class support vector machines for protein secondary structure prediction. *Genome Informatics*, 14:218–227, 2003.

[OKR$^+$99] A.R. Ortiz, A. Kolinski, P. Rotkiewicz, B. Ilkowski, and J. Skolnick. Ab initio folding of proteins using restraints derived from evolutionary information. *Proteins*, Suppl 3:177–185, 1999.

[OMJ$^+$97] C.A. Orengo, A.D. Michie, S. Jones, D.T. Jones, M.B. Swindells, and J.M. Thornton. Cath - a hierarchic classification of protein domain structures. *Structure, 5, 1093-1108*, 5:1093–1108, 1997.

[ON97] S.I. O'Donoghue and M. Nilges. Tertiary structure prediction using mean-force potentials and internal energy functions: successful prediction for coiled-coil geometries. *Folding & Design*, 2:S47–S52, 1997.

[OO87] E. Otaka and T. Ooi. Examination of protein sequence homologies: Iv. twenty-seven bacterial ferredoxins. *J. Mol. Evol.*, 26(3):257–67, 1987.

[Pai00] R.H. Pain, editor. *Mechanisms of Protein Folding.* Frontiers in Molecular Biology Series. Oxford University Press, 2000.

[PB02] G. Pollastri and P. Baldi. Prediction of contact maps by recurrent neural network architectures and hidden context propagation from all four cardinal corners. *Bioinformatics*, 1(1):1–9, 2002.

[PBFC02] G. Pollastri, P. Baldi, P. Fariselli, and R. Casadio. Prediction of coordination number and relative solvent accessibility in proteins. *Proteins*, 47(2):142–153, 2002.

[Pea85] W.R. Pearson. Rapid and sensitive sequence comparisons with fastp and fasta. *Methods in Enzymology*, 183:63–98, 1985.

[Pla00] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers.* MIT Press, 2000.

[PLN$^+$00] T.N. Petersen, C. Lundegaard, M. Nielsen, H. Bohr, J. Bohr, S. Brunak, G.P. Gippert, and O. Lund. Prediction of protein secondary structure at 80% accuracy. *Proteins*, 41(1):17–20, October 2000.

[PM97] J.T. Pedersen and J. Moult. Protein folding simulations with genetic algorithms and a detailed molecular description. *J Mol. Biol.*, 269(2):240–259, 1997.

[PPF02] A. Passerini, M. Pontil, and P. Frasconi. From margins to probabilities in multiclass learning problems. In F. van Harmelen, editor, *Proc. 15th European Conf. on Artificial Intelligence*, 2002.

[PPF04] A. Passerini, M. Pontil, and P. Frasconi. New results on error correcting output codes of kernel machines. *IEEE Transactions on Neural Networks*, 15(1):45–54, 2004.

[PPRB02] G. Pollastri, D. Przybylski, B. Rost, and P. Baldi. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. *Proteins*, 47(2):228–235, 2002.

[PR02] D. Przybylski and B. Rost. Alignments grow, secondary structure prediction improves. *Proteins*, 46(2):197–205, 2002.

[PRE98] M. Pontil, R. Rifkin, and T. Evgeniou. From regression to classification in support vector machines. A.I. Memo 1649, MIT Artificial Intelligence Lab., 1998.

[PS72] K.R. Parthasarathy and K. Schmidt. Positive definite kernels, continuous tensor products, and central limit theorems of probability theory. In *Lecture Notes in Math.*, volume 272. Springer, Berlin, 1972.

[QS88] N. Qian and T.J. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *J Mol. Biol.*, 202(4):865–884, 1988.

[Qui86] J.Ross. Quinlan. Inductive learning of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui93] J.Ross. Quinlan. *C4.5: Programs for machine learning.* Morgan Kaufmann Publishers, 1993.

[Rab89] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RB98] A. Rietsch and J. Beckwith. The genetics of disulfide bond metabolism. *Annu Rev Genet.*, 32:163–184, 1998.

[RB99] C.J. Richardson and D.J. Barlow. The bottom line for prediction of residue solvent accessibility. *Protein Eng.*, 12:1051–1054, 1999.

[RB01] D. Ritz and J. Beckwith. Roles of thiol-redox pathways in bacteria. *Annu. Rev. Microbiol.*, 55:21–48, 2001.

[RCFS95] B. Rost, R. Casadio, P. Fariselli, and C. Sander. Transmembrane helices predicted at 95% accuracy. *Protein Sci.*, 4:521–533, 1995.

[RHH$^+$99] A.C. Rosenzweig, D.L. Huffman, M.Y. Hou, A.K. Wernimont, R.A. Pufahl, and T.V. O'Halloran. Crystal structure of the atx1 metallochaperone protein at 1.02 Åresolution. *Structure*, 7:605–617, 1999.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[Ros97] B. Rost. Protein structures sustain evolutionary drift. *Folding & Design*, 2:S19–S24, 1997.

[Ros98] B. Rost. Protein structure prediction in 1d, 2d, and 3d. In P. von Rague-Schleyer, N.L. Allinger, T. CClark, J. Gasteiger, P.A. Kollman, and H.F. Schaefer, editors, *Encyclopedia of Computational Chemistry*, pages 2242–2255. John Wiley, Sussex, 1998.

[RR01] J.D.M. Rennie and R. Rifkin. Improving multiclass text classification with the support vector machine. Technical Report 2001-026, Massachusetts Institute of Technology, 2001.

[RS93] B. Rost and C. Sander. Prediction of protein secondary structure at better than 70% accuracy. *J Mol. Biol.*, 232:584–599, 1993.

[RS94] B. Rost and C. Sander. Conservation and prediction of solvent accessibility in protein families. *Proteins*, 20(216–226), 1994.

[RSS97] B. Rost, R. Schneider, and C. Sander. Protein fold recognition by prediction-based threading. *J Mol. Biol.*, 270:471–480, 1997.

[Sai88] S. Saitoh. *Theory of Reproducing Kernels and its Applications.* Longman Scientific Technical, Harlow, England, 1988.

[Sch97] B. Schölkopf. *Support Vector Learning.* PhD thesis, Technische Universität Berlin, 1997.

[SGV98] C. Suanders, A. Gammerman, and V. Vovk. Ridge regression learning algortihm in dual variables. In *Proc. of the* 15*th* *International Conference on Machine Learning (ICML98)*, 1998.

[SGV+99] M. Stitson, A. Gammerman, V.N. Vapnik, V. Vovk, C. Watkins, and J. Weston. Support vector regression with anova decomposition kernels. In B. Schölkopf, C Burges, and A.J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning.* MIT Press, Cambridge, MA, 1999.

[SHSW01] B. Schölkopf, R. Herbrich, A.J. Smola, and R. Williamson. A generalized representer theorem. In *Proceedings COLT 2001*, Lecture Notes in Artificial Intelligence. Springer, 2001.

[Sip95] M.J. Sippl. Knowledge-based potentials for proteins. *Curr. Opin. Struct. Biol.*, 5:229–235, 1995.

[SKHB97] K.T. Simons, C. Kooperberg, E. Huang, and D. Baker. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and bayesian scoring functions. *J Mol. Biol.*, 268(1):209–225, 1997.

[SNB+03] C.A.E.M. Spronk, S.B. Nabuurs, A.M.J.J. Bonvin, E. Krieger, G.W. Vuister, and G. Vriend. The precision of nmr structure ensembles revisited. *Journal of Biomolecular NMR*, 25:225–234, 2003.

[SPST+01] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson. Estimating the support of a high dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.

[SS90] T. D. Schneider and R. M. Stephens. Sequence logos: A new way to display consensus sequences. *Nucleic Acids Res.*, 18:6097–6100, 1990.

[SS91] C. Sander and R. Schneider. Database of homology-derived structures and the structural meaning of sequence alignment. *Proteins*, 9:56–68, 1991.

[SS02] B. Schölkopf and A.J. Smola. *Learning with Kernels.* The MIT Press, Cambridge, MA, 2002.

[SSB01] K.T. Simons, C. Strauss, and D. Baker. Prospects for ab initio protein structural genomics. *J Mol. Biol.*, 306(5):1191–1199, 2001.

[SSB03] Q. Su, S. Saxonov, and D. Brutlag. eblocks: an automated database of protein conserved regions maximizing sensitivity and specificity, 2003. http://fold.stanford.edu/eblocks/.

[SSM98] A. Smola, B. Schölkopf, and K. Muller. General cost functions for support vector regression. In T. Downs, M. Frean, and M. Gallagher, editors, *Proc. of the Ninth Australian Conf. on Neural Networks*, pages 79–83, Brisbane, Australia, 1998.

[SSM99] B. Schölkopf, A.J. Smola, and K.-R. Müller. Kernel principal component analysis. In B. Schölkopf, C Burges, and A.J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 327–352. MIT Press, Cambridge, MA, 1999.

[SW81] T. Smith and M. Waterman. Identification of common molecular subsequences. *J Mol. Biol.*, 147:195–197, 1981.

[SW92] M.J. Sippl and S. Weitckus. Detection of native like models for amino acid sequences of unknown three dimensional structure in a data base of known protein conformations. *Proteins*, 13:258–271, 1992.

[SW03] B. Schölkopf and M.K. Warmuth, editors. *Kernels and Regularization on Graphs*, volume 2777 of *Lecture Notes in Computer Science.* Springer, 2003.

[SWS$^+$00] B. Schölkopf, R.C. Williamson, A.J. Smola, J. Shawe-Taylor, and J.C. Platt. Support vector method for novelty detection. In S.A. Solla, T.K. Leen, and K.-R. Muller, editors, *Advances in Neural Information Processing Systems 12*, Proc. of the 1999 Conference, 2000.

[TCS96] D.J. Thomas, G. Casari, and C. Sander. The prediction of protein contacts from multiple sequence alignments. *Protein Eng.*, 9(11):941–948, 1996.

[TD99] D.M.J. Tax and R.P.W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20:1991–1999, 1999.

[TG96] M.J. Thompson and R.A. Goldstein. Predicting solvent accessibility: higher accuracy using bayesian statistics and optimized residue substitution classes. *Proteins*, 25(1):38–47, 1996.

[Tik63] A.N. Tikhonov. On solving ill-posed problem and method of regularization. *Dokl. Akad. Nauk USSR*, 153:501–504, 1963.

[Tsu99] K. Tsuda. Support vector classification with asymmetric kernel function. In M. Verleysen, editor, *Proc. of ESANN*, pages 183–188, 1999.

[Ukk95] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14:249–260, 1995.

[Vap79] V.N. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian].* Springer-Verlag, Nauka, Moscow, 1979. (English translation: Springer-Verlag, New York, 1982).

[Vap95] V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[Vap98] V.N. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[VC71] V.N. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[Ver02] R. Vert. Designing a m-svm kernel for protein secondary structure prediction. Master's thesis, DEA informatique de Lorraine, 2002.

[VF03] A. Vullo and P. Frasconi. Prediction of protein coarse contact maps. *Journal of Bioinformatics and Computational Biology*, 1(2):411–431, 2003.

[vG93] W.F. van Gunsteren. Molecular dynamics studies of proteins. *Curr. Opin. Struct. Biol.*, 3:277–281, 1993.

[VKD97] M. Vendruscolo, E. Kussell, and E. Domany. Recovery of protein structure from contact maps. *Fold Des.*, 2(5):295–306, 1997.

[Ẅ86] K. Wüthrich. *NMR of Proteins and Nucleic Acids.* Wiley, New York, 1986.

[Wah90] G. Wahba. *Splines Models for Observational Data.* Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.

[Wat00] C. Watkins. Dynamic alignment kernels. In A.J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classiers*, pages 39–50. MIT Press, 2000.

[WMBJ03] J.J. Ward, L.J McGuffin, B.F. Buxton, and D.T. Jones. Secondary structure prediction with support vector machines. *Bioinformatics*, 19(13):1650–1655, 2003.

[WMC$^+$01] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V.N. Vapnik. Feature selection for support vector machines. In *NIPS-13*, 2001.

[WW98] J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science, 1998.

[WWNS00] W.J. Wedemeyer, E. Welker, M. Narayan, and H.A. Scheraga. Disulfide bonds and protein folding. *Biochemistry*, 39:4207–4216, 2000.

[ZJB00] M.J. Zaki, S. Jin, and C. Bystroff. Mining residue contacts in proteins using local structure predictions. In *IEEE International Symposium on Bio-Informatics and Biomedical Engineering (BIBE'00)*, Arlilngton, Virginia, November 08 - 10 2000.