COMPUTATIONAL MODELS FOR DATA ANALYSIS

Kernel Methods

Alessandra Giordani

Department of information and communication technology
University of Trento
Email: moschitti@dit.unitn.it
**Linear Classifier**

- The equation of a hyperplane is

\[ f(\tilde{x}) = \tilde{x} \cdot \tilde{w} + b = 0, \quad \tilde{x}, \tilde{w} \in \mathbb{R}^n, b \in \mathbb{R} \]

- \( \tilde{x} \) is the vector representing the classifying example

- \( \tilde{w} \) is the gradient of the hyperplane

- The classification function is

\[ h(x) = \text{sign}(f(x)) \]
The main idea of Kernel Functions

- Mapping vectors in a space where they are linearly separable \( \vec{x} \rightarrow \phi(\vec{x}) \)
A mapping example

- Given two masses $m_1$ and $m_2$, one is constrained
- Apply a force $f_a$ to the mass $m_1$
- Experiments
  - Features $m_1$, $m_2$ and $f_a$
- We want to learn a classifier that tells when a mass $m_1$ will get far away from $m_2$
- If we consider the Gravitational Newton Law
  \[ f(m_1, m_2, r) = C \frac{m_1 m_2}{r^2} \]
- we need to find when $f(m_1, m_2, r) < f_a$
A mapping example (2)

\[ \bar{X} = (x_1, \ldots, x_n) \rightarrow \phi(\bar{X}) = (\phi_1(\bar{X}), \ldots, \phi_n(\bar{X})) \]

- The gravitational law is not linear so we need to change space

\[ (f_a, m_1, m_2, r) \rightarrow (k, x, y, z) = (\ln f_a, \ln m_1, \ln m_2, \ln r) \]

- As

\[ \ln f(m_1, m_2, r) = \ln C + \ln m_1 + \ln m_2 - 2\ln r = c + x + y - 2z \]

- We need the hyperplane

\[ \ln f_a - \ln m_1 - \ln m_2 + 2\ln r - \ln C = 0 \]

\[ (\ln m_1, \ln m_2, -2\ln r) \cdot (x, y, z) - \ln f_a + \ln C = 0 \], we can decide without error if the mass will get far away or not
A kernel-based Machine Perceptron training

\[ \tilde{w}_0 \leftarrow 0; b_0 \leftarrow 0; k \leftarrow 0; R \leftarrow \max_{1 \leq i \leq l} \| \tilde{x}_i \| \]

do

for i = 1 to \( \ell \)

if \( y_i (\tilde{w}_k \cdot \tilde{x}_i + b_k) \leq 0 \) then

\[ \tilde{w}_{k+1} = \tilde{w}_k + \eta y_i \tilde{x}_i \]

\[ b_{k+1} = b_k + \eta y_i R^2 \]

k = k + 1
endif
endfor

while an error is found

return k, (\( \tilde{w}_k, b_k \))
Kernel Function Definition

**Def. 2.26** A kernel is a function $k$, such that $\forall \vec{x}, \vec{z} \in X$

$$k(\vec{x}, \vec{z}) = \phi(\vec{x}) \cdot \phi(\vec{z})$$

where $\phi$ is a mapping from $X$ to an (inner product) feature space.

- Kernels are the product of mapping functions such as

$$\vec{x} \in \mathbb{R}^n, \quad \vec{\phi}(\vec{x}) = (\phi_1(\vec{x}), \phi_2(\vec{x}), \ldots, \phi_m(\vec{x})) \in \mathbb{R}^m$$
The Kernel Gram Matrix

- With KM-based learning, the **sole** information used from the training data set is the Kernel Gram Matrix

\[
K_{\text{training}} = \begin{bmatrix}
    k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_m) \\
    k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_m) \\
    \vdots & \vdots & \ddots & \vdots \\
    k(x_m, x_1) & k(x_m, x_2) & \cdots & k(x_m, x_m)
\end{bmatrix}
\]

- If the kernel is valid, K is symmetric definite-positive.
Valid Kernels

Def. B.11 Eigen Values
Given a matrix $A \in \mathbb{R}^{m} \times \mathbb{R}^{n}$, an eigenvalue $\lambda$ and an eigenvector $\vec{x} \in \mathbb{R}^{n} - \{\vec{0}\}$ are such that

$$A\vec{x} = \lambda \vec{x}$$

Def. B.12 Symmetric Matrix
A square matrix $A \in \mathbb{R}^{n} \times \mathbb{R}^{n}$ is symmetric iff $A_{ij} = A_{ji}$ for $i \neq j$ $i = 1, \ldots, m$ and $j = 1, \ldots, n$, i.e. iff $A = A'$.

Def. B.13 Positive (Semi-) definite Matrix
A square matrix $A \in \mathbb{R}^{n} \times \mathbb{R}^{n}$ is said to be positive (semi-) definite if its eigenvalues are all positive (non-negative).
Mercer’s condition

**Proposition 2.27 (Mercer’s conditions)**
Let $X$ be a finite input space with $K(\bar{x}, \bar{z})$ a symmetric function on $X$. Then $K(\bar{x}, \bar{z})$ is a kernel function if and only if the matrix

$$k(\bar{x}, \bar{z}) = \phi(\bar{x}) \cdot \phi(\bar{z})$$

is positive semi-definite (has non-negative eigenvalues).

- If the Gram matrix: $G = k(\bar{x}_i, \bar{x}_j)$
  is positive semi-definite there is a mapping $\phi$ that produces the target kernel function
Mercer’s Theorem (finite space)

Let us consider \( K = \left( K(\vec{x}_i, \vec{x}_j) \right)_{i,j=1}^{n} \)

\( K \) symmetric \( \Rightarrow \exists V: K = V \Lambda V' \) for Takagi factorization of a complex-symmetric matrix, where:

- \( \Lambda \) is the diagonal matrix of the eigenvalues \( \lambda_t \) of \( K \)
- \( \vec{v}_t = \left( v_{ti} \right)_{i=1}^{n} \) are the eigenvectors, i.e. the columns of \( V \)
- Let us assume lambda values non-negative

\[ \phi: \vec{x}_i \rightarrow \left( \sqrt{\lambda_t} v_{ti} \right)_{t=1}^{n} \in \mathbb{R}^n, \ i = 1,...,n \]
Mercer’s Theorem
(sufficient conditions)

Therefore

\[ \Phi(\bar{x}_i) \cdot \Phi(\bar{x}_j) = \sum_{t=1}^{n} \lambda_t v_{ti} v_{tj} = (V \Lambda V')_{ij} = K_{ij} = K(\bar{x}_i, \bar{x}_j) \]

which implies that K is a kernel function
Mercer’s Theorem (necessary conditions)

- Suppose we have negative eigenvalues \( \lambda_s \) and eigenvectors \( \vec{v}_s \) the following point

\[
\tilde{z} = \sum_{i=1}^{n} v_{si} \Phi(\tilde{x}_i) = \sum_{i=1}^{n} v_{si} (\sqrt{\lambda_t} v_{ti})_t = \sqrt{\Lambda} V' \vec{v}_s
\]

- has the following norm:

\[
\|\tilde{z}\|^2 = \tilde{z} \cdot \tilde{z} = \sqrt{\Lambda} V' \vec{v}_s \sqrt{\Lambda} V' \vec{v}_s = \vec{v}_s' V \sqrt{\Lambda} \sqrt{\Lambda} V' \vec{v}_s = \vec{v}_s' K \vec{v}_s = \vec{v}_s' \lambda_s \vec{v}_s = \lambda_s \|\vec{v}_s\|^2 < 0
\]

this contradicts the geometry of the space.
Is it a valid kernel?

- It may not be a kernel so we can use $M' \cdot M$

**Proposition B.14** Let $A$ be a symmetric matrix. Then $A$ is positive (semi-)definite iff for any vector $\vec{x} \neq 0$

$$\vec{x}' A \vec{x} > 0 \quad (\geq 0).$$

From the previous proposition it follows that: If we find a decomposition $A$ in $M'M$, then $A$ is semi-definite positive matrix as

$$\vec{x}' A \vec{x} = \vec{x}' M' M \vec{x} = (M \vec{x})' (M \vec{x}) = M \vec{x} \cdot M \vec{x} = |M \vec{x}|^2 \geq 0.$$
Valid Kernel operations

- \( k(x,z) = k_1(x,z) + k_2(x,z) \)
- \( k(x,z) = k_1(x,z) \cdot k_2(x,z) \)
- \( k(x,z) = \alpha k_1(x,z) \)
- \( k(x,z) = f(x)f(z) \)
- \( k(x,z) = k_1(\phi(x), \phi(z)) \)
- \( k(x,z) = x'Bz \)
Basic Kernels for unstructured data

- Linear Kernel
- Polynomial Kernel
- Lexical kernel
- String Kernel
Linear Kernel

- In Text Categorization documents are word vectors

\[ \Phi(d_x) = \vec{x} = (0,..,1,..,0,..,0,..,1,..,0,..,0,..,1,..,0,..,1,..,0,..,1) \]

buy acquisition stocks sell market

\[ \Phi(d_z) = \vec{z} = (0,..,1,..,0,..,1,..,0,..,0,..,0,..,1,..,0,..,0,..,0,..,0,..,0) \]

buy company stocks sell

- The dot product \[ \vec{x} \cdot \vec{z} \] counts the number of features in common

- This provides a sort of similarity
Feature Conjunction (polynomial Kernel)

The initial vectors are mapped in a higher space

\[ \Phi(<x_1, x_2>) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1) \]

More expressive, as \((x_1x_2)\) encodes \textbf{Stock+Market vs. Downtown+Market} features

We can smartly compute the scalar product as

\[ \Phi(\tilde{x}) \cdot \Phi(\tilde{z}) = \]
\[ = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1) \cdot (z_1^2, z_2^2, \sqrt{2}z_1z_2, \sqrt{2}z_1, \sqrt{2}z_2, 1) = \]
\[ = x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 + 2x_1z_1 + 2x_2z_2 + 1 = \]
\[ = (x_1z_1 + x_2z_2 + 1)^2 = (\tilde{x} \cdot \tilde{z} + 1)^2 = K_{\text{Poly}}(\tilde{x}, \tilde{z}) \]
Document Similarity

Doc 1

- industry
- telephone
- market

Doc 2

- company
- product
Lexical Semantic Kernel [CoNLL 2005]

- The document similarity is the SK function:

\[ SK(d_1, d_2) = \sum_{w_1 \in d_1, w_2 \in d_2} s(w_1, w_2) \]

- where \( s \) is any similarity function between words, e.g. WordNet [Basili et al., 2005] similarity or LSA [Cristianini et al., 2002]

- Good results when training data is small
Using character sequences

\[ \phi("bank") = \tilde{x} = (0,..,1,..,0,..,1,..,0,\ldots,1,..,0,..,1,..,0,\ldots,1,\ldots,0) \]

\[ \text{bank ank bnk bk b} \]

\[ \phi("rank") = \tilde{z} = (1,..,0,..,0,..,1,..,0,\ldots,0,\ldots,1,\ldots,0,\ldots,1,\ldots,0,\ldots,1) \]

\[ \text{rank ank rnk rk r} \]

\[ \tilde{x} \cdot \tilde{z} \text{ counts the number of common substrings} \]

\[ \tilde{x} \cdot \tilde{z} = \phi("bank") \cdot \phi("rank") = k("bank","rank") \]
String Kernel

- Given two strings, the number of matches between their substrings is evaluated.
- E.g. Bank and Rank
  - B, a, n, k, Ba, Ban, Bank, Bk, an, ank, nk,..
  - R, a, n, k, Ra, Ran, Rank, Rk, an, ank, nk,..
- String kernel over sentences and texts
- Huge space but there are efficient algorithms
Formal Definition

\[ s = s_1, \ldots, s_{|s|} \]
\[ \bar{I} = (i_1, \ldots, i_{|u|}) \quad u = s[\bar{I}] \]

\[ \phi_u(s) = \sum_{\bar{I}:u=s[\bar{I}]} \chi^{l(\bar{I})}, \quad \text{where} \quad l(\bar{I}) = i_{|u|} - i_1 + 1 \]

\[ K(s, t) = \sum_{u \in \Sigma^*} \phi_u(s) \cdot \phi_u(t) = \sum_{u \in \Sigma^*} \sum_{\bar{I}:u=s[\bar{I}]} \sum_{\bar{J}:u=t[\bar{J}]} \chi^{l(\bar{I})} \chi^{l(\bar{J})} = \]

\[ = \sum_{u \in \Sigma^*} \sum_{\bar{I}:u=s[\bar{I}]} \sum_{\bar{J}:u=t[\bar{J}]} \chi^{l(\bar{I})+l(\bar{J})}, \quad \text{where} \quad \Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n \]
Kernel between Bank and Rank

B, a, n, k, Ba, Ban, Bank, an, ank, nk, Bn, Bnk, Bk and ak are the substrings of Bank.

R, a, n, k, Ra, Ran, Rank, an, ank, nk, Rn, Rnk, Rk and ak are the substrings of Rank.
An example of string kernel computation

- $\phi_a(\text{Bank}) = \phi_a(\text{Rank}) = \lambda^{(i_1-i_1+1)} = \lambda^{(2-2+1)} = \lambda$,

- $\phi_n(\text{Bank}) = \phi_n(\text{Rank}) = \lambda^{(i_1-i_1+1)} = \lambda^{(3-3+1)} = \lambda$,

- $\phi_k(\text{Bank}) = \phi_k(\text{Rank}) = \lambda^{(i_1-i_1+1)} = \lambda^{(4-4+1)} = \lambda$,

- $\phi_{an}(\text{Bank}) = \phi_{an}(\text{Rank}) = \lambda^{(i_2-i_1+1)} = \lambda^{(3-2+1)} = \lambda^2$,

- $\phi_{ank}(\text{Bank}) = \phi_{ank}(\text{Rank}) = \lambda^{(i_3-i_1+1)} = \lambda^{(4-2+1)} = \lambda^3$,

- $\phi_{nk}(\text{Bank}) = \phi_{nk}(\text{Rank}) = \lambda^{(i_2-i_1+1)} = \lambda^{(4-3+1)} = \lambda^2$,

- $\phi_{ak}(\text{Bank}) = \phi_{ak}(\text{Rank}) = \lambda^{(i_2-i_1+1)} = \lambda^{(4-2+1)} = \lambda^3$

$K(\text{Bank, Rank}) = (\lambda, \lambda, \lambda, \lambda^2, \lambda^3, \lambda^2, \lambda^3) \cdot (\lambda, \lambda, \lambda, \lambda^2, \lambda^3, \lambda^2, \lambda^3) = 3\lambda^2 + 2\lambda^4 + 2\lambda^6$
String Kernels for OCR
Pixel Representation

Figure 6: Resampling of an image from $16 \times 16$ to $8 \times 8$ format
**Sequence of bits**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>00011100</td>
</tr>
<tr>
<td>.</td>
<td>00111100</td>
</tr>
<tr>
<td>.</td>
<td>00101100</td>
</tr>
<tr>
<td>.</td>
<td>00001100</td>
</tr>
<tr>
<td></td>
<td>00001100</td>
</tr>
<tr>
<td>L8</td>
<td>00001100</td>
</tr>
</tbody>
</table>

\[
SK(im_a, im_b) = \sum_{i=1..8} SK(L^i_a, L^i_b)
\]
## Results

- **Using columns+rows+diagonals**

<table>
<thead>
<tr>
<th>Digit</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>97.78</td>
<td>97.78</td>
<td>97.78</td>
</tr>
<tr>
<td>1</td>
<td>95.45</td>
<td>93.33</td>
<td>94.38</td>
</tr>
<tr>
<td>2</td>
<td>93.62</td>
<td>97.78</td>
<td>95.65</td>
</tr>
<tr>
<td>3</td>
<td>93.33</td>
<td>93.33</td>
<td>93.33</td>
</tr>
<tr>
<td>4</td>
<td>97.83</td>
<td>100.00</td>
<td>98.90</td>
</tr>
<tr>
<td>5</td>
<td>97.67</td>
<td>93.33</td>
<td>95.45</td>
</tr>
<tr>
<td>6</td>
<td>100.00</td>
<td>97.78</td>
<td>98.88</td>
</tr>
<tr>
<td>7</td>
<td>91.84</td>
<td>100.00</td>
<td>95.74</td>
</tr>
<tr>
<td>8</td>
<td>93.18</td>
<td>91.11</td>
<td>92.13</td>
</tr>
<tr>
<td>9</td>
<td>93.02</td>
<td>88.89</td>
<td>90.91</td>
</tr>
</tbody>
</table>

**Multiclass accuracy**: 95.33
Tree kernels

- Subtree, Subset Tree, Partial Tree kernels
- Efficient computation
Main Idea of Tree Kernels

\[ \phi(T1) = [2, 1, 1, 1, 1, 0, 0] \]
\[ \phi(T2) = [0, 0, 0, 0, 1, 1, 1] \]
\[ \text{TK}(T1, T2) = \langle \phi(T1), \phi(T2) \rangle = 1 \]
Example of a syntactic parse tree

- “John delivers a talk in Rome”

```
  S
 /   \
|     |   
N     VP
 /     /     
 John  V      PP
    /     |     /     
   NP    IN    N
     /  |  /  |
    D   N  IN  N
       /  |  /  |
      a   talk in Rome
```

S → N VP
VP → V NP PP
PP → IN N
N → Rome
The Syntactic Tree Kernel (STK)
[Collins and Duffy, 2002]
The overall fragment set
The overall fragment set

Children are not divided
Explicit kernel space

\[ \phi(T_x) = \vec{x} = (0,\ldots,1,\ldots,0,\ldots,1,\ldots,0,\ldots,1,\ldots,0,\ldots,1,\ldots,0,\ldots,0) \]

\[ \phi(T_z) = \vec{z} = (1,\ldots,0,\ldots,0,\ldots,1,\ldots,0,\ldots,1,\ldots,0,\ldots,0,\ldots,1,\ldots,0,\ldots,0) \]

- \( \vec{x} \cdot \vec{z} \) counts the number of common substructures
Efficient evaluation of the scalar product

\[ \vec{x} \cdot \vec{z} = \phi(T_x) \cdot \phi(T_z) = K(T_x, T_z) = \]

\[ = \sum_{n_x \in T_x} \sum_{n_z \in T_z} \Delta(n_x, n_z) \]
Efficient evaluation of the scalar product

\[ \vec{x} \cdot \vec{z} = \phi(T_x) \cdot \phi(T_z) = K(T_x, T_z) = \]
\[ = \sum_{n_x \in T_x} \sum_{n_z \in T_z} \Delta(n_x, n_z) \]

- [Collins and Duffy, ACL 2002] evaluate \( \Delta \) in \( O(n^2) \):

\[ \Delta(n_x, n_z) = 0, \text{ if the productions are different} \]
\[ \Delta(n_x, n_z) = 1, \text{ if pre-terminals else} \]
\[ \Delta(n_x, n_z) = \prod_{j=1}^{nc(n_x)} (1 + \Delta(ch(n_x, j), ch(n_z, j))) \]
SubTree (ST) Kernel [Vishwanathan and Smola, 2002]
Given the equation for STK

\[ \Delta(n_x, n_z) = 0, \]  
\[ \Delta(n_x, n_z) = 1, \]  
\[ \Delta(n_x, n_z) = \prod_{j=1}^{nc(n_x)} (1 + \Delta(ch(n_x, j), ch(n_z, j))) \]
SVM-light-TK Software

- Encodes ST, STK and combination kernels in SVM-light [Joachims, 1999]
- Available at http://dit.unitn.it/~moschitt/
- Tree forests, vector sets
- The new SVM-Light-TK toolkit will be released asap (email me to have the current version)
Practical Example on Question Classification

- **Definition**: What does HTML stand for?
- **Description**: What's the final line in the Edgar Allan Poe poem "The Raven"?
- **Entity**: What foods can cause allergic reaction in people?
- **Human**: Who won the Nobel Peace Prize in 1992?
- **Location**: Where is the Statue of Liberty?
- **Manner**: How did Bob Marley die?
- **Numeric**: When was Martin Luther King Jr. born?
- **Organization**: What company makes Bentley cars?
Conclusions

- Dealing with noisy and errors of NLP modules require robust approaches
- SVMs are robust to noise and Kernel methods allows for:
  - Syntactic information via STK
  - Shallow Semantic Information via PTK
  - Word/POS sequences via String Kernels
- When the IR task is complex, syntax and semantics are essential
  ⇒ Great improvement in Q/A classification
- SVM-Light-TK: an efficient tool to use them
SVM-light-TK Software

- Encodes ST, SST and combination kernels in SVM-light [Joachims, 1999]
- Available at http://dit.unitn.it/~moschitt/
- Tree forests, vector sets
- New extensions: the PT kernel will be released asap
References


An introductory book on SVMs, Kernel methods and Text Categorization

Roberto Basili
Alessandro Moschitti

Automatic Text Categorization
From Information Retrieval
to Support Vector Learning