Active Learning & Beyond

Stefano Teso

University of Trento, Italy

Preliminaries

Strategies

Extensions

Conclusion and Further Reading

Preliminaries

"Imagine that you are the leader of a colonial expedition from Earth to an extrasolar planet. Luckily, this planet is habitable and has a fair amount of vegetation suitable for feeding your group. Impor- tantly, the most abundant source of food comes from a plant whose fruits are sometimes smooth and round, but sometimes bumpy and irregular."

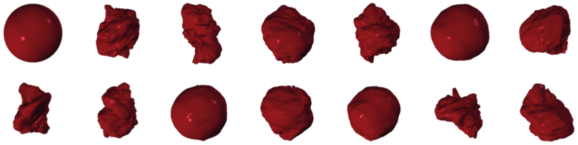


Figure 1.1: Several alien fruits, which vary in shape from round to irregular.

"The physicians assure you that the shape of a fruit is the only feature that seems related to its safety. The problem, though, is that a wide variety of fruit shapes from these plants exist: almost a continuous range from round to irregular. Since the colony has essential uses for both safe and noxious fruits, you want to be able to classify them as accurately as possible. "

Source: [Settles, 2012].

- We know that *smoother* fruits are (monotonically) *safer*, but we don't know where to set the **threshold**.

- We know that *smoother* fruits are (monotonically) *safer*, but we don't know where to set the **threshold**.

- In other words, we want to learn a **threshold function**:

$$f_\theta(\mathbf{x}) = \begin{cases} 1 & \text{if } x_{\text{irreg}} < \theta \\ -1 & \text{otherwise} \end{cases}$$

where $\mathbf{x}$ are measurements of fruit features and x_{irreg} captures its shape "irregularity".

Idea: use regular supervised learning

- Collect a large enough training set $S = \{(\mathbf{x}, y)\}$, fit threshold classifier f_θ on S
- If maximum % errors is $\epsilon \in (0, 1)$, enough to collect $O(\frac{1}{\epsilon})$ examples [Shalev-Shwartz and Ben-David, 2014]

Figure 1.2: Supervised learning for the alien fruits example. Given a set of $\langle x, y\rangle$ instance-label pairs, we want to choose the threshold θ^* that classifies them most accurately.

Idea: use regular supervised learning

- Collect a large enough training set $S = \{(\mathrm{x}, y)\}$, fit threshold classifier f_θ on S
- If maximum % errors is $\epsilon \in (0, 1)$, enough to collect $O(\frac{1}{\epsilon})$ examples [Shalev-Shwartz and Ben-David, 2014]

Figure 1.2: Supervised learning for the alien fruits example. Given a set of $\langle x, y\rangle$ instance-label pairs, we want to choose the threshold θ^* that classifies them most accurately.

■ We want to find θ as quickly and as economically as possible, by requiring fewer tests.

■ Can we do better?

Key features:

- Fruits $\mathbf{x}$ are plentiful and easy to harvest and measure
- Obtaining y incurs a **cost**: person that eats the fruit may get sick

So we definitely want to **minimize the number of needed labels**.

Idea: gather large set of **unlabeled** fruits $U = \{\mathrm{x}_i\}$ and arrange them by roughness.

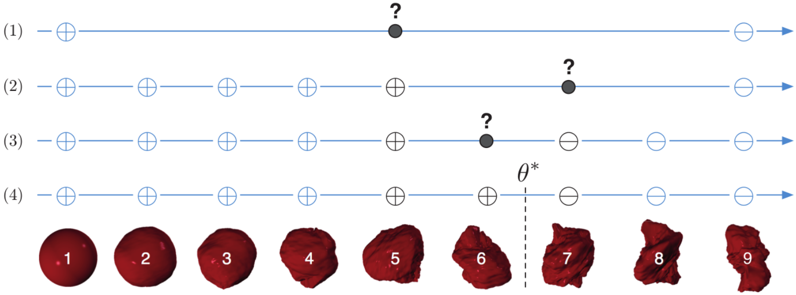


Figure 1.3: A binary search through the set of ordered, untested alien fruits. By only testing this subset of fruits, we can exponentially reduce costs while achieving the same result. The labels shown in light blue can be inferred, and therefore do not need to be tested.

Then use **binary search** (as in the illustration) to figure out the threshold θ. This only requires $O(\log_2 \frac{1}{\epsilon})$ tests!

Idea: gather large set of **unlabeled** fruits $U = \{\mathbf{x}_i\}$ and arrange them by roughness, then use **binary search**:

ϵ	$\frac{1}{\epsilon}$	$\log_2 \frac{1}{\epsilon}$
0.1	10	3.321
0.001	1000	9.966
0.00001	100000	16.610

- In this (cleverly designed illustrative) scenario, there is an **exponential** improvement in sample complexity

Active vs Passive

"*The key hypothesis is that if the learner is allowed to choose the data from which it learns — to be active, curious, or exploratory, if you will — it can perform better with less training*." [Settles, 2012]

Preconditions:

- Collecting unlabelled instances $\mathbf{x}$ is **cheap**
- Obtaining their labels y is **expensive**

Preconditions:

- Collecting unlabelled instances $\mathbf{x}$ is **cheap**
- Obtaining their labels y is **expensive**

Example: Citizen Science

There are tons of images of celestial bodies (think sky surveys). However, in order to undestand what's in an image (is it a spiral galaxy? is it a gravitational lensing effect?) you have to ask a human expert.

Preconditions:

- Collecting unlabelled instances $\mathbf{x}$ is **cheap**
- Obtaining their labels y is **expensive**

Example: Citizen Science

There are tons of images of celestial bodies (think sky surveys). However, in order to undestand what's in an image (is it a spiral galaxy? is it a gravitational lensing effect?) you have to ask a human expert.

Example: Recommendation

There are millions of products on online catalogues (think Amazon), but in order to discover what are the tastes of a user, you have to actually convince them to score the items. This information is personalized, so this is the only way to obtain supervision.

Example: Scientific Discovery

- Adam, the “robot scientist” [King et al., 2009]

The Automation of Science

Ross D. King,[1]* Jem Rowland,[1] Stephen G. Oliver,[2] Michael Young,[3] Wayne Aubrey,[1] Emma Byrne,[1] Maria Liakata,[1] Magdalena Markham,[1] Pınar Pir,[2] Larisa N. Soldatova,[1] Andrew Sparkes,[1] Kenneth E. Whelan,[1] Amanda Clare[1]

The basis of science is the hypothetico-deductive method and the recording of experiments in sufficient detail to enable reproducibility. We report the development of Robot Scientist “Adam,” which advances the automation of both. Adam has autonomously generated functional genomics hypotheses about the yeast *Saccharomyces cerevisiae* and experimentally tested these hypotheses by using laboratory automation. We have confirmed Adam’s conclusions through manual experiments. To describe Adam’s research, we have developed an ontology and logical language. The resulting formalization involves over 10,000 different research units in a nested treelike structure, 10 levels deep, that relates the 6.6 million biomass measurements to their logical description. This formalization describes how a machine contributed to scientific knowledge.

- The learner obtains labels by operating an automated testing machine.

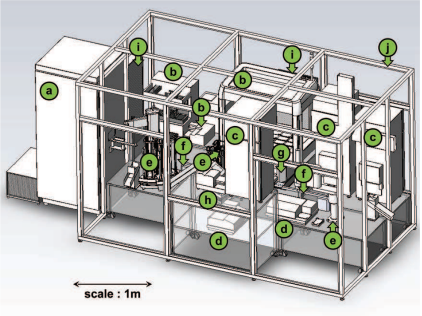


Fig. 1. The Robot Scientist Adam. The advances that distinguish Adam from other complex laboratory systems are the individual design of the experiments to test hypotheses and the utilization of complex internal cycles. Adam's basic operations are selection of specified yeast strains from a library held in a freezer, inoculation of these strains into microtiter plate wells containing rich medium, measurement of growth curves on rich medium, harvesting of a defined quantity of cells from each well, inoculation of these cells into wells containing defined media (minimal synthetic dextrose medium plus up to four added metabolites from a choice of six), and measurement of growth curves on the specified media. To achieve this functionality, Adam has the following components: a, an automated –20°C freezer; b, three liquid handlers (one of which can separately control 96 fluid channels simultaneously); c, three automated +30°C incubators; d, two automated plate readers; e, three robot arms; f, two automated plate slides; g, an automated plate centrifuge; h, an automated plate washer; i, two high-efficiency particulate air filters; and j, a rigid transparent plastic enclosure. There are also two bar code readers, seven cameras, 20 environment sensors, and four personal computers, as well as the software. Adam is capable of designing and initiating over a thousand new strain and defined-growth-medium experiments each day (from a selection of thousands of yeast strains), with each experiment lasting up to 5 days. The design enables measurement of OD_{595nm} for each experiment at least once every 30 min (more often if running at less than full capacity), allowing accurate growth curves to be recorded (typically we take over a hundred measurements a day per well), plus associated metadata. See the supporting online material for pictures and a video of Adam in action.

- Similar strategies used in chemical engineering, material engineering, *etc.*

Notation

A summary of frequently used terms:

- **Instances** $\mathbf{x} \in \mathbb{R}^d$ are *unlabelled* d-dimensional vectors of observations
- **Examples** $z = (\mathbf{x}, y)$ are instances *annotated* by a label $y \in \{0, 1\}$ or $y \in \{1, \ldots, c\}$
- A **classifier** $f : \mathbb{R}^d \rightarrow \{0, 1\}$ maps instances to labels
- $\mathcal{F} = \{f_\theta\}$ is a **family of classifiers** parameterized by θ

■ The meaning of θ depends on the model class, e.g., for neural nets with a fixed architecture, θ represents their weights; for random forests, θ represents the structure and leaves of all trees.

Assumptions

- We assume the data to be distributed according to a **ground-truth data generating process** $p^*(Y, \mathbf{X})$, and use $p^*(Y \mid \mathbf{X})$ and $p^*(\mathbf{X})$ to indicate the prior and class likelihood of this process, i.e.,

$$p^*(Y, \mathbf{X}) \equiv p^*(Y \mid \mathbf{X}) \cdot p^*(\mathbf{X}) \tag{1}$$

The data may be corrupted by label noise, but we explicitly avoid adversarial settings.

- We also assume to deal with **probabilistic classifiers**:

$$p_\theta(Y = y \mid \mathbf{X} = \mathbf{x}) \tag{2}$$

Predictions are computed as:

$$f_\theta(\mathbf{x}) := \underset{y=1,\dots,c}{\operatorname{argmax}} \; p_\theta(Y = y \mid \mathbf{X} = \mathbf{x}) \tag{3}$$

This includes, e.g., logistic regression, Gaussian processes, and neural nets with a top $\mathrm{softmax}$ activation.

- Predictors that only output a **score** (e.g., Support Vector Machines output the distance to the separating hyperplane) can often be adapted (via, e.g., calibration or Platt scaling). This is not necessary to leverage AL.

The Labeling Oracle

- The **labeling oracle** $\mathrm{label} : \mathbb{R}^d \to \{0, 1\}$ returns:

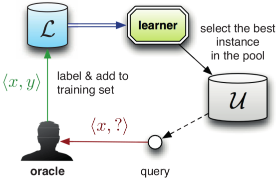
$$\mathrm{label}(\mathbf{x}) := \underset{y \in \{0,1\}}{\operatorname{argmax}} \; p^*(Y = y \mid \mathbf{X} = \mathbf{x}) \tag{4}$$

where $p^*(Y \mid \mathbf{X})$ is the ground-truth distribution.

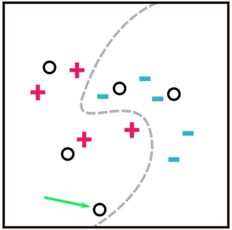
- Invoking the oracle comes at a **cost**, which is usually non-negligible, instance- and class-dependent, and possibly unknown [Herde et al., 2021].[1]

For simplicity, we assume the cost to be **unitary**: querying any instance $\mathbf{x} \in \mathbb{R}^d$ costs the same.

[1] Although it could be estimated from interaction data.



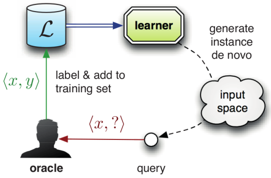
(b) pool-based sampling



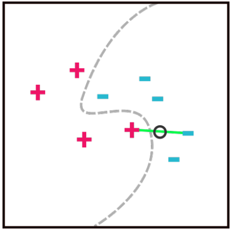
Active Learning (Pool-based). Given:

- a family of classifiers $\mathcal{F}$,
- a set of unlabelled instances $U = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathbb{R}^d$ sampled i.i.d. from $p^*(\mathbf{X})$,
- a (costly) labeling oracle $\text{label} : \mathbb{R}^d \to \{0, 1\}$,

Find a classifier $\widehat{f} \in \mathcal{F}$ that achieves low risk on $p^*(\mathbf{X}, Y)$ while keeping annot. cost low



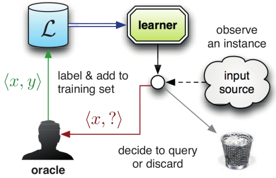
(a) query synthesis



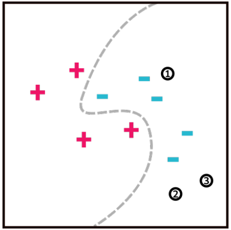
Active Learning (Query Synthesis). Given:

- a family of classifiers $\mathcal{F}$,
- a generator of instances $\mathrm{synthesize}(\text{region}) \mapsto \mathbf{x}$,
- a (costly) labeling oracle $\mathrm{label} : \mathbb{R}^d \to \{0, 1\}$,

Find a classifier $\widehat{f} \in \mathcal{F}$ that achieves low risk on $p^*(\mathbf{X}, Y)$ while keeping annot. cost low



(a) selective sampling

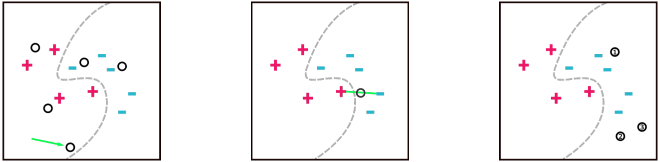


Active Learning (Selective Sampling). **Given**:

- a family of classifiers $\mathcal{F}$,
- a sequence of unlabelled instances $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots,$
- a (costly) labeling oracle $\text{label} : \mathbb{R}^d \to \{0, 1\}$

Find a classifier $\widehat{f}_t \in \mathcal{F}$ that achieves low risk on future data $\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \ldots$ while keeping annot. cost low

Query Sampling vs. Query Synthesis



- Left to right:
  - **Pool-based**: moderate control over queries, requires memory to store U
  - **Query synthesis**: maximum control over queries, can generate uninterpretable queries [Baum and Lang, 1992], although deep generative models can help somehow [Nguyen et al., 2016].
  - **Selective sampling**: little control over the distribution of queries, often solved under tight memory constraints (online learning)
- We will focus on **pool-based AL**.

Strategies

Template

Input: models $\mathcal{F}$, examples L, pool U, query budget $T \geq 1$
Output: selected model $f \in \mathcal{F}$

$f \leftarrow \mathrm{fit}(\mathcal{F}, L)$ ▷ initialize the model
for $t = 1, 2, \ldots, T$ **do** ▷ until the budget is exhausted
  $\mathbf{x} \leftarrow \mathrm{argmax}_{x \in U}\, \mathrm{acq}(f, \mathbf{x})$ ▷ select a query instance
  obtain label y of $\mathbf{x}$ from annotator
  $U \leftarrow U \setminus \{\mathbf{x}\}$ ▷ remove unlabeled instance from pool
  $L \leftarrow L \cup \{(\mathbf{x}, y)\}$ ▷ update training set
  $f \leftarrow \mathrm{fit}(\mathcal{F}, L)$ ▷ update the model
return f

- fit performs training (e.g., trains for a fixed # of epochs)
- acq scores instances based on their "informativeness"
- What instance $\mathbf{x} \in U$ should be selected so to convey **as much information as possible** to f?

Uncertainty Sampling

What's the point of asking the label of instances on which the classifier is already certain?[2]

- **Left**: two Gaussians (40 points each)
- **Middle**: picking points completely **at random** (ignoring the class label!)
- **Right**: picking points based on **uncertainty**

[2] There is a point to doing so, as we will see later.

Uncertainty Sampling

- **Idea**: pick $\mathbf{x} \in U$ on which the classifier is **most uncertain**.

- **Idea**: pick $\mathbf{x} \in U$ on which the classifier is **most uncertain**.

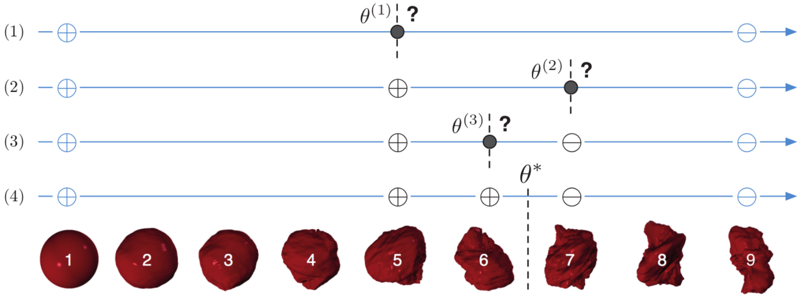


Figure 2.1: The binary search from Figure 1.3, re-interpreted as an uncertainty sampling approach. The best instance to query is deemed to be the one closest to the threshold θ.

Uncertainty Sampling

- **Idea**: pick $\mathbf{x} \in U$ on which the classifier is **most uncertain**.

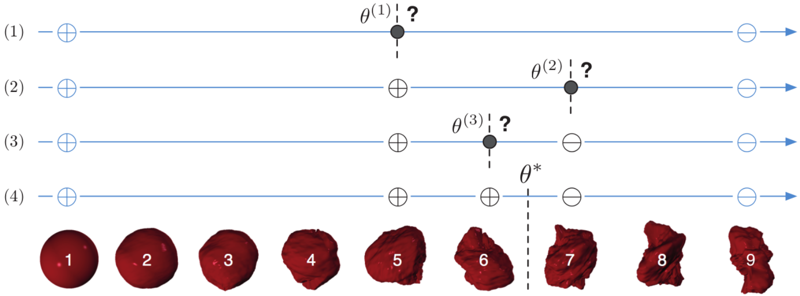


Figure 2.1: The binary search from Figure 1.3, re-interpreted as an uncertainty sampling approach. The best instance to query is deemed to be the one closest to the threshold θ.

- How should uncertainty be defined?

Uncertainty Sampling

- Define uncertainty using the **confidence**, i.e., *distance from certainty*:

$$\mathrm{acq}(\theta, \mathbf{x}) := 1 - p_\theta(\hat{y} \mid \mathbf{x}) \tag{5}$$

where $\hat{y}$ is the predicted label:

$$\hat{y} := f_\theta(\mathbf{x}) = \underset{y}{\mathrm{argmax}}\ p_\theta(y \mid \mathbf{x}) \tag{6}$$

Uncertainty Sampling

- Define uncertainty using the **margin**, i.e., difference in (conditional) likelihood:

$$\mathrm{acq}(\theta, \mathbf{x}) := p_\theta(\hat{y}' \mid \mathbf{x}) - p_\theta(\hat{y} \mid \mathbf{x}) \tag{7}$$

where $\hat{y}$ is the predicted label and $\hat{y}'$ is the **2nd best** label:

$$\hat{y} = \operatorname*{argmax}_{y} p_\theta(y \mid \mathbf{x}) \tag{8}$$

$$\hat{y}' := \operatorname*{argmax}_{y \neq \hat{y}} p_\theta(y \mid \mathbf{x}) \tag{9}$$

Uncertainty Sampling

- Define uncertainty using the Shannon **entropy** of the label:

$$\mathrm{acq}(\theta, \mathbf{x}) := H_\theta(Y \mid \mathbf{X} = \mathbf{x}) \tag{10}$$

where H_θ is defined as:

$$H_\theta(Y \mid \mathbf{X} = \mathbf{x}) := -\sum_{y \in [c]} p_\theta(y \mid \mathbf{x}) \log_2 p_\theta(y \mid \mathbf{x}) \tag{11}$$

  - It achieves a **minimum** on dead certain distributions:

$$p_\theta(Y \mid \mathbf{x}) = (0, 1, 0, \ldots, 0)$$

  - and a **maximum** on the uniform distribution:

$$p_\theta(Y \mid \mathbf{x}) = (\frac{1}{c}, \cdots, \frac{1}{c})$$

Confidence vs. Margin vs. Entropy

- **Left**: confidence considers prob. of top class only
- **Middle**: margin considers prob. of top & runner up classes
- **Right**: entropy considers prob. of all classes

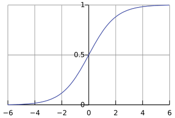
If $c = 2$, they are equivalent. If $c > 2$, no obvious best choice, it really depends on the task and loss (e.g., cross-entropy vs.accuracy)

Confidence vs. Margin vs. Entropy

- **Left**: confidence considers prob. of top class only
- **Middle**: margin considers prob. of top & runner up classes
- **Right**: entropy considers prob. of all classes

If $c = 2$, they are equivalent. If $c > 2$, no obvious best choice, it really depends on the task and loss (e.g., cross-entropy vs.accuracy)

Example: for classifiers with a sigmoid top layer:



uncertainty depends on distance from separating hyperplane of predicted vs. top two vs. all classes

Uncertainty Sampling

- Uncertainty sampling is very **easy to implement**.
- Margin & Confidence can be defined even in terms of unnormalized scores.
- Usually performs reasonably well (though not optimally) in practice: a useful baseline/starting point.

Example: Structured Output

Consider an LSTM that takes a sequence of MNIST images $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ that composes a word and outputs the word itself $\mathbf{y} = (y_1, \dots, y_n)$.

- Computing the most likely output $\hat{\mathbf{y}}$ can be done efficiently.
- Computing the entropy amounts to:

$$H_\theta(Y \mid \mathbf{X} = \mathbf{x}) := - \sum_{\mathbf{y} \in \{1,\dots,26\}^n} p_\theta(\mathbf{y} \mid X) \log_2 p_\theta(\mathbf{y} \mid X) \tag{12}$$

This involves summing over 26^n possible outputs, which takes time **exponential** in n.

Computing the most likely output can be NP-hard. For instance, if $\mathbf{y}$ is molecular structure that mast satisfy specific hard constraints (chemical validity), then finding the best structure amounts to solving a hard combinatorial problem.

Hence, the **confidence** and **margin** can also be very hard.

Uncertainty in Regression

- When considering regression models with $Y \in \mathbb{R}$, uncertainty at $\mathbf{x}$ can be implemented as **differential entropy**:

$$H_\theta(Y \mid \mathbf{X} = \mathbf{x}) := \mathbb{E}[-\log_2 p_\theta(y \mid \mathbf{x}) \mid \mathbf{x}] \tag{13}$$

$$= -\int_{\mathbb{R}} p_\theta(y \mid \mathbf{x}) \log_2 p_\theta(y \mid \mathbf{x}) \tag{14}$$

- When considering regression models with $Y \in \mathbb{R}$, uncertainty at $\mathbf{x}$ can be implemented as **differential entropy**:

$$H_\theta(Y \mid \mathbf{X} = \mathbf{x}) := \mathbb{E}[-\log_2 p_\theta(y \mid \mathbf{x}) \mid \mathbf{x}] \tag{13}$$

$$= -\int_{\mathbb{R}} p_\theta(y \mid \mathbf{x}) \log_2 p_\theta(y \mid \mathbf{x}) \tag{14}$$

- As an alternative heuristic, use the **variance**:

$$\mathrm{Var}_\theta(Y \mid \mathbf{x}) := \mathbb{E}[(Y - \underbrace{\mathbb{E}[Y \mid \mathbf{x}]}_{\mu_\theta(Y \mid \mathbf{x}):=})^2 \mid \mathbf{x}] \tag{15}$$

$$= \int_{\mathbb{R}} (y - \mu_\theta(Y \mid \mathbf{x}))^2 p_\theta(y \mid \mathbf{x}) dy \tag{16}$$

$$\mu_\theta(Y \mid \mathbf{x}) = \int_{\mathbb{R}} y \; p_\theta(y \mid \mathbf{x}) dy \tag{17}$$

Uncertainty in Regression

- When considering regression models with $Y \in \mathbb{R}$, uncertainty at $\mathbf{x}$ can be implemented as **differential entropy**:

$$H_\theta(Y \mid \mathbf{X} = \mathbf{x}) := \mathbb{E}[-\log_2 p_\theta(y \mid \mathbf{x}) \mid \mathbf{x}] \tag{13}$$

$$= -\int_{\mathbb{R}} p_\theta(y \mid \mathbf{x}) \log_2 p_\theta(y \mid \mathbf{x}) \tag{14}$$

- As an alternative heuristic, use the **variance**:

$$\mathrm{Var}_\theta(Y \mid \mathbf{x}) := \mathbb{E}[(Y - \underbrace{\mathbb{E}[Y \mid \mathbf{x}]}_{\mu_\theta(Y|\mathbf{x}):=})^2 \mid \mathbf{x}] \tag{15}$$

$$= \int_{\mathbb{R}} (y - \mu_\theta(Y \mid \mathbf{x}))^2 p_\theta(y \mid \mathbf{x}) dy \tag{16}$$

$$\mu_\theta(Y \mid \mathbf{x}) = \int_{\mathbb{R}} y \; p_\theta(y \mid \mathbf{x}) dy \tag{17}$$

- How to compute them?

Uncertainty in Regression

- Differential entropy and variance:

$$H_\theta(Y \mid \mathbf{X} = \mathbf{x}) = -\int_{\mathbb{R}} p_\theta(y \mid \mathbf{x}) \log_2 p_\theta(y \mid \mathbf{x}) \qquad \mathrm{Var}_\theta(Y \mid \mathbf{x}) = \int_{\mathbb{R}} (p_\theta(y \mid \mathbf{x}) - \mu_\theta(Y \mid \mathbf{x})) dy \tag{18}$$

[3] See https://en.wikipedia.org/wiki/Normal_distribution.

Uncertainty in Regression

- Differential entropy and variance:

$$H_\theta(Y \mid \mathbf{X} = \mathbf{x}) = -\int_{\mathbb{R}} p_\theta(y \mid \mathbf{x}) \log_2 p_\theta(y \mid \mathbf{x}) \qquad \mathrm{Var}_\theta(Y \mid \mathbf{x}) = \int_{\mathbb{R}} (p_\theta(y \mid \mathbf{x}) - \mu_\theta(Y \mid \mathbf{x}))dy \tag{18}$$

- Both expensive to compute for general models, can **approximate** via quadrature or sampling, but **closed-form solutions** exist for some models (e.g., Gaussian Processes and NNs with a Gaussian output)

[3] See `https://en.wikipedia.org/wiki/Normal_distribution`.

Uncertainty in Regression

- Differential entropy and variance:

$$H_\theta(Y \mid \mathbf{X} = \mathbf{x}) = -\int_{\mathbb{R}} p_\theta(y \mid \mathbf{x}) \log_2 p_\theta(y \mid \mathbf{x}) \qquad \mathrm{Var}_\theta(Y \mid \mathbf{x}) = \int_{\mathbb{R}} (p_\theta(y \mid \mathbf{x}) - \mu_\theta(Y \mid \mathbf{x}))dy \tag{18}$$

- Both expensive to compute for general models, can **approximate** via quadrature or sampling, but **closed-form solutions** exist for some models (e.g., Gaussian Processes and NNs with a Gaussian output)

Example: 1-dimensional Gaussian Output

Consider one-dimensional output $y \in \mathbb{R}$ and a neural net:

$$nn : \mathbf{x} \mapsto (\mu, \sigma), \qquad y \sim \mathcal{N}(\mu, \sigma) \tag{19}$$

In this case, it is well known[3] that:

$$\mathrm{Var}_\theta(Y \mid \mathbf{x}) = \sigma^2, \qquad H_\theta(Y \mid \mathbf{x}) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2} \tag{20}$$

Notice that $\mathrm{Var}_\theta(Y \mid \mathbf{x}) \propto \exp H_\theta(Y \mid \mathbf{x})$, so they change monotonically.

[3] See https://en.wikipedia.org/wiki/Normal_distribution.

- Differential entropy and variance:

$$H_\theta(Y \mid \mathbf{X} = \mathbf{x}) = -\int_{\mathbb{R}} p_\theta(y \mid \mathbf{x}) \log_2 p_\theta(y \mid \mathbf{x}) \qquad \mathrm{Var}_\theta(Y \mid \mathbf{x}) = \int_{\mathbb{R}} (p_\theta(y \mid \mathbf{x}) - \mu_\theta(Y \mid \mathbf{x}))dy \tag{21}$$

- Both expensive to compute for general models, can **approximate** via quadrature or sampling, but **closed-form solutions** exist for some models (e.g., Gaussian Processes and NNs with a Gaussian output)

Example: k-dimensional Gaussian Output

Consider one-dimensional output $y \in \mathbb{R}^k$ and a neural net:

$$nn : \mathbf{x} \mapsto (\boldsymbol{\mu}, S), \qquad \Sigma \leftarrow SS^T, \qquad \mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma) \tag{22}$$

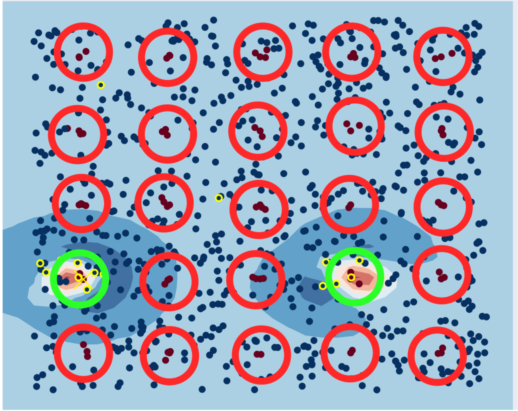
with Σ PSD by construction. In this case, it is well known[4] that:

$$\mathrm{Var}_\theta(Y \mid \mathbf{x}) \propto \mathrm{tr}\Sigma \qquad H_\theta(Y \mid \mathbf{x}) \propto \log\det\Sigma \tag{23}$$

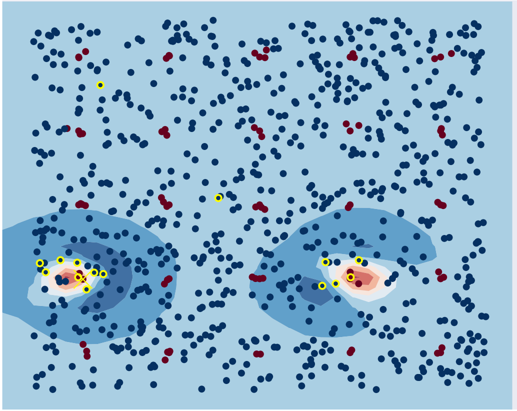
where the trace is cheap to compute but the determinant is more challenging.

[4] See `https://en.wikipedia.org/wiki/Multivariate_normal_distribution`.

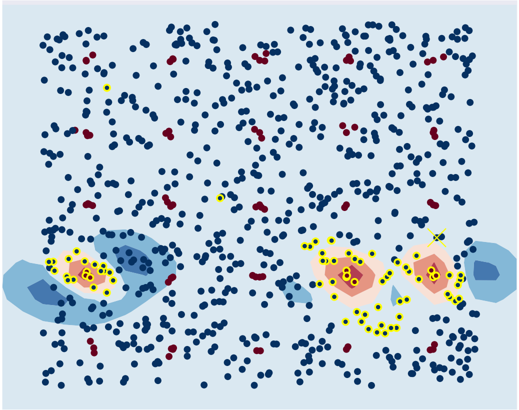
Illustration



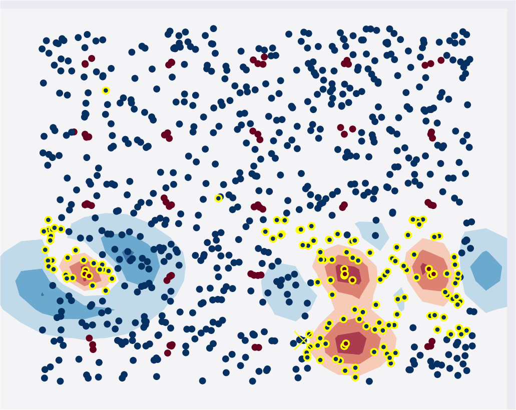
- Synthetic dataset: 25 clusters of **red** points arranged in a 5 $\times$ 5 grid, surrounded by a sea of **blue** points



- After **10** iterations of uncertainty sampling.



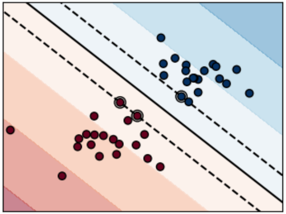
- After **70** iterations of uncertainty sampling.



- After **140** iterations of uncertainty sampling. Not nice!

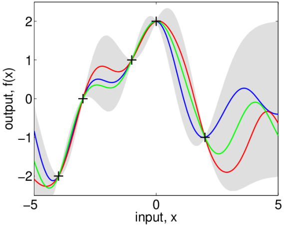
Over-confidence

- **Discriminative** models are **over-confident**:



Uncertainty does **not** decrease with distance from the training set.

- **Bayesian generative models** not so much:



(a), posterior

Uncertainty **does** decrease with distance from the training set.

Aleatoric vs Epistemic [Hüllermeier and Waegeman, 2021]

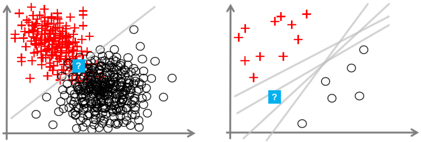


Figure 5: Left: Even with precise knowledge about the optimal hypothesis, the prediction at the query point (indicated by a question mark) is aleatorically uncertain, because the two classes are overlapping in that region. Right: A case of epistemic uncertainty due to a lack of knowledge about the right hypothesis, which is in turn caused by a lack of data.

Aleatoric vs Epistemic [Hüllermeier and Waegeman, 2021]

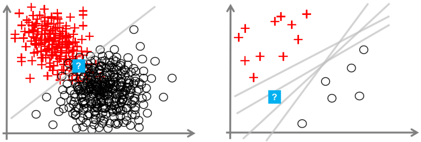


Figure 5: Left: Even with precise knowledge about the optimal hypothesis, the prediction at the query point (indicated by a question mark) is aleatorically uncertain, because the two classes are overlapping in that region. Right: A case of epistemic uncertainty due to a lack of knowledge about the right hypothesis, which is in turn caused by a lack of data.

- **Aleatoric** uncertainty ("random") captures how much we can trust the supervision itself. It cannot be decreased. (**left**)

Aleatoric vs Epistemic [Hüllermeier and Waegeman, 2021]

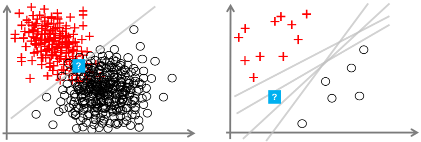


Figure 5: Left: Even with precise knowledge about the optimal hypothesis, the prediction at the query point (indicated by a question mark) is aleatorically uncertain, because the two classes are overlapping in that region. Right: A case of epistemic uncertainty due to a lack of knowledge about the right hypothesis, which is in turn caused by a lack of data.

- **Aleatoric** uncertainty ("random") captures how much we can trust the supervision itself. It cannot be decreased. (**left**)

- **Epistemic** uncertainty ("relating to knowledge") captures how little we know about the world. This reflects on uncertainty on the choice of θ. It decreases by acquiring **more data**. (**right**)

Aleatoric vs Epistemic [Hüllermeier and Waegeman, 2021]

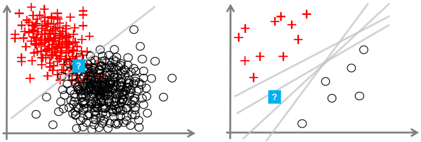


Figure 5: Left: Even with precise knowledge about the optimal hypothesis, the prediction at the query point (indicated by a question mark) is aleatorically uncertain, because the two classes are overlapping in that region. Right: A case of epistemic uncertainty due to a lack of knowledge about the right hypothesis, which is in turn caused by a lack of data.

- **Aleatoric** uncertainty ("random") captures how much we can trust the supervision itself. It cannot be decreased. (**left**)
- **Epistemic** uncertainty ("relating to knowledge") captures how little we know about the world. This reflects on uncertainty on the choice of θ. It decreases by acquiring **more data**. (**right**)
- There isn't much point in trying to reduce **aleatoric** uncertainty in AL [Sharma and Bilgic, 2017]

Uncertainty Sampling for Streaming Data

Input: models $\mathcal{F}$, bootstrap training set L, threshold τ
Output: selected model $f \in \mathcal{F}$

```
f ← fit(F, L)                          ▷ initialize the model
for t = 1, 2, 3, . . . , do
    receive instance x
    if unc(f, x) > τ then              ▷ if f is uncertain about x
        obtain label y of x from annotator
        L ← L ∪ {(x, y)}               ▷ update training set
        f ← fit(F, L)                  ▷ update the model
return f
```

- The tricky bit is setting τ. Many algorithms update it dynamically by, e.g, starting from a large τ and lowering it as new data is received and the model improves

- For some problems, US **converges** to the right thing – because it is uncertain enough

- If you are unluckly, US becomes **over-confident**: in this example, the model becomes confident that the regions inside the black blob cannot be white, so it does not sample them and converges to the wrong shape.

- Uncertainty sampling is quite heuristic. Are there more **principled** approaches?

Version Space

- Consider a **hypothesis space** $\mathcal{F} = \{f_\theta : \mathbb{R}^d \to [c]\}$ and a data set $S = \{(\mathbf{x}_i, y_i)\}$

Version Space

- Consider a **hypothesis space** $\mathcal{F} = \{f_\theta : \mathbb{R}^d \to [c]\}$ and a data set $S = \{(\mathbf{x}_i, y_i)\}$

Consistency

A hypothesis $f \in \mathcal{F}$ is **consistent** with S, written $f \models S$, iff it makes *zero mistakes* on it, that is:

$$(f \models S) \iff \left(\sum_{(\mathbf{x},y)\in S} \mathbb{1}\{f(\mathbf{x}) \neq y\}\right) = 0 \tag{24}$$

Version Space

- Consider a **hypothesis space** $\mathcal{F} = \{f_\theta : \mathbb{R}^d \to [c]\}$ and a data set $S = \{(\mathbf{x}_i, y_i)\}$

Consistency

A hypothesis $f \in \mathcal{F}$ is **consistent** with S, written $f \models S$, iff it makes *zero mistakes* on it, that is:

$$(f \models S) \iff \left(\sum_{(\mathbf{x},y)\in S} \mathbb{1}\{f(\mathbf{x}) \neq y\}\right) = 0 \tag{24}$$

Version Space

The **version space** $VS_{\mathcal{F}}(S)$ of $\mathcal{F}$ given S is the set of hypotheses $f \in \mathcal{F}$ that is *consistent* with S, that is:

$$VS_{\mathcal{F}}(S) = \{f \in \mathcal{F} \;:\; f \models S\} \tag{25}$$

- Consider a **hypothesis space** $\mathcal{F} = \{f_\theta : \mathbb{R}^d \to [c]\}$ and a data set $S = \{(\mathbf{x}_i, y_i)\}$

Consistency

A hypothesis $f \in \mathcal{F}$ is **consistent** with S, written $f \models S$, iff it makes *zero mistakes* on it, that is:

$$(f \models S) \iff \left(\sum_{(\mathbf{x},y)\in S} \mathbb{1}\{f(\mathbf{x}) \neq y\}\right) = 0 \tag{24}$$

Version Space

The **version space** $VS_{\mathcal{F}}(S)$ of $\mathcal{F}$ given S is the set of hypotheses $f \in \mathcal{F}$ that is *consistent* with S, that is:

$$VS_{\mathcal{F}}(S) = \{f \in \mathcal{F} \ : \ f \models S\} \tag{25}$$

- $VS_{\mathcal{F}}(L)$ contains those hypotheses that have not yet been ruled out by the acquired examples L: **they are all just as good w.r.t.** S

Version Space

Version Space

The **version space** $VS_{\mathcal{F}}(S)$ of $\mathcal{F}$ given S is the set of hypotheses $f \in \mathcal{F}$ that is *consistent* with S, that is:

$$VS_{\mathcal{F}}(S) = \{f \in \mathcal{F} \ : \ f \models S\} \tag{26}$$

- If S is not **separable** w.r.t. $\mathcal{F}$, i.e., if there is no hypothesis $f \in \mathcal{F}$ that is constent with it, then the version space is empty ($VS_{\mathcal{F}}(S) = \varnothing$). This happens because:

Version Space

Version Space

The **version space** $VS_{\mathcal{F}}(S)$ of $\mathcal{F}$ given S is the set of hypotheses $f \in \mathcal{F}$ that is *consistent* with S, that is:

$$VS_{\mathcal{F}}(S) = \{f \in \mathcal{F} \ : \ f \models S\} \tag{26}$$

- If S is not **separable** w.r.t. $\mathcal{F}$, i.e., if there is no hypothesis $f \in \mathcal{F}$ that is constent with it, then the version space is empty ($VS_{\mathcal{F}}(S) = \varnothing$). This happens because:
  - $\mathcal{F}$ is **not expressive enough**. Example: *no* $f^* \in \mathcal{F}$ *that correctly labels the data*.

Version Space

The **version space** $VS_{\mathcal{F}}(S)$ of $\mathcal{F}$ given S is the set of hypotheses $f \in \mathcal{F}$ that is *consistent* with S, that is:

$$VS_{\mathcal{F}}(S) = \{f \in \mathcal{F} \ : \ f \models S\} \tag{26}$$

- If S is not **separable** w.r.t. $\mathcal{F}$, i.e., if there is no hypothesis $f \in \mathcal{F}$ that is constent with it, then the version space is empty ($VS_{\mathcal{F}}(S) = \varnothing$). This happens because:
  - $\mathcal{F}$ is **not expressive enough**. Example: *no* $f^* \in \mathcal{F}$ *that correctly labels the data*.
  - S is **noisy**. Example: *S contains the same instance twice but annotated with different labels – e.g.,* $(\mathrm{x}, 1)$ *and* $(\mathrm{x}, 3)$ *– so no* $f \in \mathcal{F}$ *can classify both correctly.*

Version Space

The **version space** $VS_{\mathcal{F}}(S)$ of $\mathcal{F}$ given S is the set of hypotheses $f \in \mathcal{F}$ that is *consistent* with S, that is:

$$VS_{\mathcal{F}}(S) = \{f \in \mathcal{F} \ : \ f \models S\} \tag{26}$$

- If S is not **separable** w.r.t. $\mathcal{F}$, i.e., if there is no hypothesis $f \in \mathcal{F}$ that is constent with it, then the version space is empty ($VS_{\mathcal{F}}(S) = \varnothing$). This happens because:
  - $\mathcal{F}$ is **not expressive enough**. Example: *no* $f^* \in \mathcal{F}$ *that correctly labels the data.*
  - S is **noisy**. Example: *S contains the same instance twice but annotated with different labels – e.g.,* $(\mathbf{x}, 1)$ *and* $(\mathbf{x}, 3)$ *– so no* $f \in \mathcal{F}$ *can classify both correctly.*

- We assume the **realizable case**: $\exists f^* \in \mathcal{F}$ s.t. $y = f^*(\mathbf{x})$ for all $\mathbf{x}$ and no noise.

This implies that $f^* \in VS_{\mathcal{F}}(L)$ for all choices of labeled examples L, because the supervision $(\mathbf{x}, y)$ is always consistent with f^*. In addition, $VS_{\mathcal{F}}(L) \neq \varnothing$.

Version Space $\leftrightarrow$ Disagreement Region

Disagreement Region

Given $\mathcal{F}$ and S, the **disagreement region** is the set of points $\mathbf{x} \in \mathbb{R}^d$ such that there exist two classifiers f, f' in the version space $VS_{\mathcal{F}}(S)$ that produce different predictions for them:

$$DIS_{\mathcal{F}}(S) = \{\mathbf{x} \in \mathbb{R}^d \ : \ \exists f, f' \in VS_{\mathcal{F}}(S) \ . \ f(\mathbf{x}) \neq f'(\mathbf{x})\} \tag{27}$$

Version Space $\leftrightarrow$ Disagreement Region

Disagreement Region

Given $\mathcal{F}$ and S, the **disagreement region** is the set of points $\mathbf{x} \in \mathbb{R}^d$ such that there exist two classifiers f, f' in the version space $VS_{\mathcal{F}}(S)$ that produce different predictions for them:

$$DIS_{\mathcal{F}}(S) = \{\mathbf{x} \in \mathbb{R}^d \ : \ \exists f, f' \in VS_{\mathcal{F}}(S) \ . \ f(\mathbf{x}) \neq f'(\mathbf{x})\} \tag{27}$$

- If $\mathbf{x} \notin DIS_{\mathcal{F}}(S)$, then all candidate classifiers f in the version space classify it the same: acquiring its label is **pointless**.

Version Space $\leftrightarrow$ Disagreement Region

Disagreement Region

Given $\mathcal{F}$ and S, the **disagreement region** is the set of points $\mathbf{x} \in \mathbb{R}^d$ such that there exist two classifiers f, f' in the version space $VS_{\mathcal{F}}(S)$ that produce different predictions for them:

$$DIS_{\mathcal{F}}(S) = \{\mathbf{x} \in \mathbb{R}^d \; : \; \exists f, f' \in VS_{\mathcal{F}}(S) \; . \; f(\mathbf{x}) \neq f'(\mathbf{x})\} \tag{27}$$

- If $\mathbf{x} \notin DIS_{\mathcal{F}}(S)$, then all candidate classifiers f in the version space classify it the same: acquiring its label is **pointless**.
- If $\mathbf{x} \in DIS_{\mathcal{F}}(S)$, then at least one f in the version space classifies it differently: acquiring its label is **useful**.

Left: **input space** $\mathbb{R}^d$, data set S of red crosses vs blue circles. **Right**: **hypothesis space** $\mathcal{F}$, each f is a point; the ground-truth f^* is in red.

Left: **input space** $\mathbb{R}^d$, data set S of red crosses vs blue circles. **Right**: **hypothesis space** $\mathcal{F}$, each f is a point; the ground-truth f^* is in red.

$\mathcal{F}$ is the set of $2D$ rectangles. Rectangles in instance space (left) are points in hypothesis space (right), as shown by the arrows.

Left: **input space** $\mathbb{R}^d$, data set S of red crosses vs blue circles. **Right**: **hypothesis space** $\mathcal{F}$, each f is a point; the ground-truth f^* is in red.

$\mathcal{F}$ is the set of $2D$ rectangles. Rectangles in instance space (left) are points in hypothesis space (right), as shown by the arrows.

The **version space** $VS_{\mathcal{F}}(S)$ contains all the rectangles (pale gray) between inner & outer rectangles (darker gray)

Left: **input space** $\mathbb{R}^d$, data set S of red crosses vs blue circles. **Right**: **hypothesis space** $\mathcal{F}$, each f is a point; the ground-truth f^* is in red.

$\mathcal{F}$ is the set of $2D$ rectangles. Rectangles in instance space (left) are points in hypothesis space (right), as shown by the arrows.

The **version space** $VS_{\mathcal{F}}(S)$ contains all the rectangles (pale gray) between inner & outer rectangles (darker gray)

The **disagreement region** $DIS_{\mathcal{F}}(S)$ is the space enclosed between these two rectangles.

Streaming AL

Input: models $\mathcal{F}$
Output: selected model $f \in \mathcal{F}$

$L \leftarrow \varnothing$
$\mathcal{V} \leftarrow \mathcal{F}$ ▷ implements the version space $VS_{\mathcal{F}}(L)$
for $t = 1, 2, 3, \ldots,$ **do**
  receive instance $\mathbf{x}$
  if $\mathbf{x} \in DIS(\mathcal{V})$ **then** ▷ if $\mathbf{x}$ falls in the disagreement region
    obtain label y of $\mathbf{x}$
    update $\mathcal{V} \leftarrow \{f \in \mathcal{V} \ : \ f(\mathbf{x}) = y\}$ ▷ update version space
return any $f \in \mathcal{V}$

Streaming AL

Input: models $\mathcal{F}$

Output: selected model $f \in \mathcal{F}$

$L \leftarrow \varnothing$

$\mathcal{V} \leftarrow \mathcal{F}$ ▷ implements the version space $VS_{\mathcal{F}}(L)$

for $t = 1, 2, 3, \dots,$ **do**

  receive instance $\mathbf{x}$

  if $\mathbf{x} \in DIS(\mathcal{V})$ **then** ▷ if $\mathbf{x}$ falls in the disagreement region

    obtain label y of $\mathbf{x}$

    update $\mathcal{V} \leftarrow \{f \in \mathcal{V} \ : \ f(\mathbf{x}) = y\}$ ▷ update version space

return any $f \in \mathcal{V}$

- If $\mathbf{x} \in DIS_{\mathcal{F}}(L)$, then there are two $f, f' \in VS_{\mathcal{F}}(L)$ that disagree on how $\mathbf{x}$ should be labeled. Getting its label allows us to get rid of either f or f', so $VS_{\mathcal{F}}(S)$ *and* $DIS_{\mathcal{F}}(S)$ both shrink.

Streaming AL

Input: models $\mathcal{F}$
Output: selected model $f \in \mathcal{F}$

$L \leftarrow \varnothing$
$\mathcal{V} \leftarrow \mathcal{F}$ ▷ implements the version space $VS_{\mathcal{F}}(L)$
for $t = 1, 2, 3, \ldots,$ **do**
  receive instance $\mathbf{x}$
  if $\mathbf{x} \in DIS(\mathcal{V})$ **then** ▷ if $\mathbf{x}$ falls in the disagreement region
    obtain label y of $\mathbf{x}$
    update $\mathcal{V} \leftarrow \{f \in \mathcal{V} \ : \ f(\mathbf{x}) = y\}$ ▷ update version space
return any $f \in \mathcal{V}$

- If $\mathbf{x} \in DIS_{\mathcal{F}}(L)$, then there are two $f, f' \in VS_{\mathcal{F}}(L)$ that disagree on how $\mathbf{x}$ should be labeled. Getting its label allows us to get rid of either f or f', so $VS_{\mathcal{F}}(S)$ *and* $DIS_{\mathcal{F}}(S)$ both shrink.

- Recall that f^* is always compatible with examples $(\mathbf{x}, y)$, so it is always in $\mathcal{V}$ $\rightarrow$ algorithm *zooms* into it!

Streaming AL

Input: models $\mathcal{F}$
Output: selected model $f \in \mathcal{F}$

$L \leftarrow \varnothing$
$\mathcal{V} \leftarrow \mathcal{F}$ ▷ implements the version space $VS_{\mathcal{F}}(L)$
for $t = 1, 2, 3, \ldots,$ **do**
    receive instance $\mathbf{x}$
    if $\mathbf{x} \in DIS(\mathcal{V})$ **then** ▷ if $\mathbf{x}$ falls in the disagreement region
        obtain label y of $\mathbf{x}$
        update $\mathcal{V} \leftarrow \{f \in \mathcal{V} \ : \ f(\mathbf{x}) = y\}$ ▷ update version space
return any $f \in \mathcal{V}$

- If $\mathbf{x} \in DIS_{\mathcal{F}}(L)$, then there are two $f, f' \in VS_{\mathcal{F}}(L)$ that disagree on how $\mathbf{x}$ should be labeled. Getting its label allows us to get rid of either f or f', so $VS_{\mathcal{F}}(S)$ *and* $DIS_{\mathcal{F}}(S)$ both shrink.

- Recall that f^* is always compatible with examples $(\mathbf{x}, y)$, so it is always in $\mathcal{V}$ $\rightarrow$ algorithm *zooms* into it!

- This algorithm makes **no useless queries**!

Streaming AL

Input: models $\mathcal{F}$
Output: selected model $f \in \mathcal{F}$

$L \leftarrow \varnothing$
$\mathcal{V} \leftarrow \mathcal{F}$ ▷ implements the version space $VS_{\mathcal{F}}(L)$
for $t = 1, 2, 3, \ldots,$ **do**
  receive instance $\mathbf{x}$
  if $\mathbf{x} \in DIS(\mathcal{V})$ **then** ▷ if $\mathbf{x}$ falls in the disagreement region
    obtain label y of $\mathbf{x}$
    update $\mathcal{V} \leftarrow \{f \in \mathcal{V} \;:\; f(\mathbf{x}) = y\}$ ▷ update version space
return any $f \in \mathcal{V}$

- If $\mathbf{x} \in DIS_{\mathcal{F}}(L)$, then there are two $f, f' \in VS_{\mathcal{F}}(L)$ that disagree on how $\mathbf{x}$ should be labeled. Getting its label allows us to get rid of either f or f', so $VS_{\mathcal{F}}(S)$ *and* $DIS_{\mathcal{F}}(S)$ both shrink.

- Recall that f^* is always compatible with examples $(\mathbf{x}, y)$, so it is always in $\mathcal{V}$ $\rightarrow$ algorithm *zooms* into it!

- This algorithm makes **no useless queries**! **. . . can we do better if we can choose $\mathbf{x}$?**

Pool-based AL

Input: models $\mathcal{F}$
Output: selected model $f \in \mathcal{F}$

1: $L \leftarrow \varnothing$
2: $\mathcal{V} \leftarrow \mathcal{F}$ ▷ implements the version space $VS_{\mathcal{F}}(L)$
3: **for** $t = 1, 2, \ldots, T$ **do**
4: $\quad \mathbf{x} \leftarrow \text{argmax}_{\mathbf{x} \in U} \ \mathrm{acq}_{VS}(\mathcal{V}, \mathcal{F}, \mathbf{x})$
5: $\quad$ obtain label y of $\mathbf{x}$
6: $\quad$ update $\mathcal{V} \leftarrow \{f \in \mathcal{V} \ : \ f(\mathbf{x}) = y\}$ ▷ update version space
7: **return** any $f \in \mathcal{V}$

- We can always ensure that $\mathbf{x} \in DIS_{\mathcal{F}}(L)$ – as long as $VS_{\mathcal{F}}(S) \neq \varnothing$, in which case we can simply terminate.

Problem: how do we define the acquisition function?

Consider the *homogeneous linear classifiers*:

$$\mathcal{F} = \left\{ f_\theta(\mathbf{x}) = \mathbb{1}\left\{ \boldsymbol{\theta}^\top \mathbf{x} > 0 \right\} \ : \ \boldsymbol{\theta} \in \mathbb{R}^d, \ \|\boldsymbol{\theta}\|_2 = 1 \right\} \tag{28}$$

The version space of S is essentially the set of direction vectors $\boldsymbol{\theta}$ that classify all points correctly.

- Classifiers are hyperplanes in instance space and instances are hyperplanes in hypothesis space (**duality**)

- Pick the point $\mathbf{x} \in U$ that (greedily) restricts the version space as much as possible. In this special case, $\mathbf{x}$ it passes close to the **center** of $VS_{\mathcal{F}}(S)$.

Idea: pick $\mathbf{x} \in U$ that reduces the *volume* of the version space $VS_{\mathcal{F}}(S)$ as much as possible.

Idea: pick $\mathbf{x} \in U$ that reduces the *volume* of the version space $VS_{\mathcal{F}}(S)$ as much as possible.

- The **volume** of a region $A \subseteq \mathcal{F}$ is:

$$\mathrm{Vol}\,(A) = \int_A d\boldsymbol{\theta}$$

Idea: pick $\mathrm{x} \in U$ that reduces the *volume* of the version space $VS_{\mathcal{F}}(S)$ as much as possible.

- The **volume** of a region $A \subseteq \mathcal{F}$ is:

$$\mathrm{Vol}\,(A) = \int_A d\boldsymbol{\theta} = \int_{\boldsymbol{\theta} \in \mathbb{R}^{|\boldsymbol{\theta}|}} \delta\{\boldsymbol{\theta} \in A\} d\boldsymbol{\theta} \tag{29}$$

So computing a volume in general requires **integration**.

Idea: pick $\mathbf{x} \in U$ that reduces the *volume* of the version space $VS_{\mathcal{F}}(S)$ as much as possible.

- The **volume** of a region $A \subseteq \mathcal{F}$ is:

$$\mathrm{Vol}\,(A) = \int_A d\boldsymbol{\theta} = \int_{\boldsymbol{\theta} \in \mathbb{R}^{|\boldsymbol{\theta}|}} \delta\{\boldsymbol{\theta} \in A\} d\boldsymbol{\theta} \tag{29}$$

So computing a volume in general requires **integration**.

- Pick $\mathbf{x}$ that minimizes $\mathrm{Vol}\,(VS_{\mathcal{F}}(S \cup \{(\mathbf{x}, y)\}))$. However, we don't *know* the label y of $\mathbf{x}$

Idea: pick $\mathbf{x} \in U$ that reduces the *volume* of the version space $VS_{\mathcal{F}}(S)$ as much as possible.

- The **volume** of a region $A \subseteq \mathcal{F}$ is:

$$\mathrm{Vol}\,(A) = \int_A d\boldsymbol{\theta} = \int_{\boldsymbol{\theta} \in \mathbb{R}^{|\boldsymbol{\theta}|}} \delta\{\boldsymbol{\theta} \in A\} d\boldsymbol{\theta} \tag{29}$$

So computing a volume in general requires **integration**.

- Pick $\mathbf{x}$ that minimizes $\mathrm{Vol}\,(VS_{\mathcal{F}}(S \cup \{(\mathbf{x}, y)\}))$. However, we don't *know* the label y of $\mathbf{x}$ → **marginalize it**:

$$\operatorname*{argmin}_{\mathbf{x} \in U} \mathbb{E}_{y \sim p_\theta(Y|\mathbf{x})} \left[\mathrm{Vol}\,(VS_{\mathcal{F}}(S \cup \{(\mathbf{x}, y)\}))\right] \tag{30}$$

Question: how to **encode** the version space?

Question: how to **encode** the version space?

- If $\mathcal{F}$ is **finite**, can explicitly store $f \models S$. **Bonus**: computing expected volume is doable (integral becomes sum).

Question: how to **encode** the version space?

- If $\mathcal{F}$ is **finite**, can explicitly store $f \models S$. **Bonus**: computing expected volume is doable (integral becomes sum).

- If $\mathcal{F}$ is **infinite**, cannot store explicitly. However, we only need to compute its **volume**:

$$\mathbb{E}_{y \sim p_\theta(Y \mid \mathbf{x})}\left[\mathrm{Vol}\left(VS_{\mathcal{F}}(S \cup \{(\mathbf{x}, y)\})\right)\right] = \sum_{y \in [c]} p_\theta(y \mid \mathbf{x}) \cdot \underbrace{\mathrm{Vol}\left(VS_{\mathcal{F}}(S \cup \{(\mathbf{x}, y)\})\right)}_{\text{this is the difficult bit}} \tag{31}$$

Question: how to **encode** the version space?

- If $\mathcal{F}$ is **finite**, can explicitly store $f \models S$. **Bonus**: computing expected volume is doable (integral becomes sum).

- If $\mathcal{F}$ is **infinite**, cannot store explicitly. However, we only need to compute its **volume**:

$$\mathbb{E}_{y \sim p_\theta(Y \mid \mathbf{x})} \left[\mathrm{Vol}\left(VS_{\mathcal{F}}(S \cup \{(\mathbf{x}, y)\}) \right) \right] = \sum_{y \in [c]} p_\theta(y \mid \mathbf{x}) \cdot \underbrace{\mathrm{Vol}\left(VS_{\mathcal{F}}(S \cup \{(\mathbf{x}, y)\}) \right)}_{\text{this is the difficult bit}} \tag{31}$$

  - If **$\mathcal{F}$ is "simple" and/or S is small**, volume can be approximated cheaply using Monte Carlo techniques. For instance with *rejection sampling*, let $B \subseteq VS_{\mathcal{F}}(S)$ of known volume:

$$\{\tilde{\boldsymbol{\theta}}_i \sim \mathrm{Uniform}(B) \ : \ i \in [s]\}, \qquad \mathrm{Vol}\left(VS_{\mathcal{F}}(S')\right) \approx \frac{1}{\mathrm{Vol}(B)} \cdot \frac{1}{s} \sum_{i \in [s]} \mathbb{1}\left\{ \tilde{\boldsymbol{\theta}}_i \in VS_{\mathcal{F}}(S') \right\} \tag{32}$$

  To check, $\mathbb{1}\{\boldsymbol{\theta}_i \in VS_{\mathcal{F}}(S')\}$, check that $f_{\boldsymbol{\theta}}$ classifies all examples in S correctly.

Question: how to **encode** the version space?

- If $\mathcal{F}$ is **finite**, can explicitly store $f \models S$. **Bonus**: computing expected volume is doable (integral becomes sum).

- If $\mathcal{F}$ is **infinite**, cannot store explicitly. However, we only need to compute its **volume**:

$$\mathbb{E}_{y \sim p_\theta(Y|\mathbf{x})}\left[\mathrm{Vol}\left(VS_{\mathcal{F}}(S \cup \{(\mathbf{x}, y)\})\right)\right] \quad = \quad \sum_{y \in [c]} p_\theta(y \mid \mathbf{x}) \cdot \underbrace{\mathrm{Vol}\left(VS_{\mathcal{F}}(S \cup \{(\mathbf{x}, y)\})\right)}_{\text{this is the difficult bit}} \tag{31}$$

  - If **$\mathcal{F}$ is "simple" and/or S is small**, volume can be approximated cheaply using Monte Carlo techniques. For instance with *rejection sampling*, let $B \subseteq VS_{\mathcal{F}}(S)$ of known volume:

$$\{\widetilde{\boldsymbol{\theta}}_i \sim \mathrm{Uniform}(B) \; : \; i \in [s]\}, \qquad \mathrm{Vol}\left(VS_{\mathcal{F}}(S')\right) \approx \frac{1}{\mathrm{Vol}\left(B\right)} \cdot \frac{1}{s} \sum_{i \in [s]} \mathbb{1}\left\{\widetilde{\boldsymbol{\theta}}_i \in VS_{\mathcal{F}}(S')\right\} \tag{32}$$

    To check, $\mathbb{1}\{\boldsymbol{\theta}_i \in VS_{\mathcal{F}}(S')\}$, check that $f_{\boldsymbol{\theta}}$ classifies all examples in S correctly.

  - Otherwise (think CNN on ImageNet), can be **extremely challenging**

- These approaches make two **assumptions**:

- These approaches make two **assumptions**:
  - Disagreement is measures using *all* hypotheses in the version space $VS_{\mathcal{F}}(S)$.

- These approaches make two **assumptions**:
  - Disagreement is measures using *all* hypotheses in the version space $VS_{\mathcal{F}}(S)$.
  - Disagreement is binary: it is only 0 if all hypotheses fully agree on $\mathbf{x} \in U$.
- Let's **relax** both of them $\rightarrow$ **speed-up**!

- These approaches make two **assumptions**:
  - Disagreement is measures using *all* hypotheses in the version space $VS_{\mathcal{F}}(S)$.
  - Disagreement is binary: it is only 0 if all hypotheses fully agree on $\mathbf{x} \in U$.
- Let's **relax** both of them $\rightarrow$ **speed-up**!

- Moreover, version space is only non-empty in the realizable case. How do we deal with this?

Query By Committee (QBC)

Idea:

- Subsample k representatives $C \subset VS_{\mathcal{F}}(S)$, $k \gg 1$
- Measure disagreement efficiently between representatives in C

Query By Committee (QBC)

Idea:

- Subsample k representatives $C \subset VS_{\mathcal{F}}(S)$, $k \gg 1$
- Measure disagreement efficiently between representatives in C

■ How to generate C?

- Sample $\mathcal{C} := \{c_j : j \in [k]\} \overset{k}{\sim} VS_{\mathcal{F}}(S)$, as done previously (i.e., we don't *gain* much)

Query By Committee (QBC)

Idea:

- Subsample k representatives $C \subset VS_{\mathcal{F}}(S)$, $k \gg 1$
- Measure disagreement efficiently between representatives in C

■ How to generate C?

- Sample $\mathcal{C} := \{c_j : j \in [k]\} \overset{k}{\sim} VS_{\mathcal{F}}(S)$, as done previously (i.e., we don't *gain* much)
- Define a distribution $p(\theta \mid L)$, very Bayesian $\rightarrow$ can be challenging to sample from the posterior

Query By Committee (QBC)

Idea:

- Subsample k representatives $C \subset VS_{\mathcal{F}}(S)$, $k \gg 1$
- Measure disagreement efficiently between representatives in C

How to generate C?

- Sample $\mathcal{C} := \{c_j : j \in [k]\} \overset{k}{\sim} VS_{\mathcal{F}}(S)$, as done previously (i.e., we don't *gain* much)
- Define a distribution $p(\theta \mid L)$, very Bayesian $\rightarrow$ can be challenging to sample from the posterior
- **Bagging**: *sample* k subsets of L, train one classifier c_j on each.

Idea:

- Subsample k representatives $C \subset VS_{\mathcal{F}}(S)$, $k \gg 1$
- Measure disagreement efficiently between representatives in C

How to generate C?

- Sample $\mathcal{C} := \{c_j : j \in [k]\} \overset{k}{\sim} VS_{\mathcal{F}}(S)$, as done previously (i.e., we don't *gain* much)
- Define a distribution $p(\theta \mid L)$, very Bayesian $\rightarrow$ can be challenging to sample from the posterior
- **Bagging**: *sample* k subsets of L, train one classifier c_j on each.
- **Boosting**: *randomly reweight* L, sequentially train k classifiers by repeatedly reweighting examples by mistakes made by previous classifier.

- How to measure disagreement of $\mathcal{C}$ on $\mathbf{x} \in U$?

- How to measure disagreement of $\mathcal{C}$ on $\mathbf{x} \in U$?

- **“Hard” Voting + Entropy**:

$$\underset{\mathbf{x}\in U}{\text{argmax}} \; -\sum_{y} \frac{n(y,\mathbf{x})}{k} \log \frac{n(y,\mathbf{x})}{k}, \qquad n(y,\mathbf{x}) := \sum_{c\in\mathcal{C}} \mathbb{1}\{c(\mathbf{x}) = y\} \tag{33}$$

Each classifiers votes either 0 or 1.

- How to measure disagreement of $\mathcal{C}$ on $\mathbf{x} \in U$?

- **"Hard" Voting + Entropy**:

$$\operatorname*{argmax}_{\mathbf{x} \in U} -\sum_{y} \frac{n(y,\mathbf{x})}{k} \log \frac{n(y,\mathbf{x})}{k}, \qquad n(y,\mathbf{x}) := \sum_{c \in \mathcal{C}} \mathbb{1}\{c(\mathbf{x}) = y\} \tag{33}$$

Each classifiers votes either 0 or 1.

- **"Soft" Voting + Entropy**:

$$\operatorname*{argmax}_{\mathbf{x} \in U} -\sum_{y} p_{\mathcal{C}}(y \mid \mathbf{x}) \log p_{\mathcal{C}}(y \mid \mathbf{x}), \qquad p_{\mathcal{C}}(y \mid \mathbf{x}) := \frac{1}{k} \sum_{c \in \mathcal{C}} p_c(y \mid \mathbf{x}) \tag{34}$$

Output probabilities of each $c \in \mathcal{C}$ taken into account.

- How to measure disagreement of $\mathcal{C}$ on $\mathbf{x} \in U$?

- **"Hard" Voting + Entropy**:

$$\underset{\mathbf{x} \in U}{\operatorname{argmax}} \; -\sum_{y} \frac{n(y, \mathbf{x})}{k} \log \frac{n(y, \mathbf{x})}{k}, \qquad n(y, \mathbf{x}) := \sum_{c \in \mathcal{C}} \mathbb{1}\{c(\mathbf{x}) = y\} \tag{33}$$

Each classifiers votes either 0 or 1.

- **"Soft" Voting + Entropy**:

$$\underset{\mathbf{x} \in U}{\operatorname{argmax}} \; -\sum_{y} p_{\mathcal{C}}(y \mid \mathbf{x}) \log p_{\mathcal{C}}(y \mid \mathbf{x}), \qquad p_{\mathcal{C}}(y \mid \mathbf{x}) := \frac{1}{k} \sum_{c \in \mathcal{C}} p_c(y \mid \mathbf{x}) \tag{34}$$

Output probabilities of each $c \in \mathcal{C}$ taken into account.

- Average Kullback-Liebler divergence:

$$\underset{\mathbf{x} \in U}{\operatorname{argmax}} \; \frac{1}{k} \sum_{c \in \mathcal{C}} \mathrm{KL}(p_c(Y \mid \mathbf{x}) \| p_{\mathcal{C}}(Y \mid \mathbf{x})) \tag{35}$$

$$\mathrm{KL}(p(Y \mid \mathbf{x}) \| q(Y \mid \mathbf{x})) := \sum_{y} p(y \mid \mathbf{x}) \log \frac{p(y \mid \mathbf{x})}{q(y \mid \mathbf{x})} \tag{36}$$

Very expressive, measures difference between **whole distributions**.

Model Improvement

Idea: pick the point that gives the maximal **improvement in model quality**

Model Improvement

Idea: pick the point that gives the maximal **improvement in model quality**

Useful Concepts

The **loss** of p_θ on example $z = (\mathbf{x}, y)$ is denoted $\ell(\theta, z)$. For instance, cross-entropy loss:

$$\ell(\theta, z) := -\sum_j \mathbb{1}\{j = y\} \log p_\theta(j \mid \mathbf{x}) = -\log p_\theta(y \mid \mathbf{x}) \tag{37}$$

Model Improvement

Idea: pick the point that gives the maximal **improvement in model quality**

Useful Concepts

The **loss** of p_θ on example $z = (\mathbf{x}, y)$ is denoted $\ell(\theta, z)$. For instance, cross-entropy loss:

$$\ell(\theta, z) := -\sum_j \mathbb{1}\{j = y\} \log p_\theta(j \mid \mathbf{x}) = -\log p_\theta(y \mid \mathbf{x}) \tag{37}$$

The **true risk** $\mathcal{L}^*$ of θ w.r.t. the *ground-truth distribution* $p^*(\mathbf{X}, Y)$ is:

$$\mathcal{L}^*(\theta) := \mathbb{E}_{z \sim p^*}[\ell(\theta, z)] \tag{38}$$

It measures the **true** quality of the model, unobserved.

Model Improvement

Idea: pick the point that gives the maximal **improvement in model quality**

Useful Concepts

The **loss** of p_θ on example $z = (\mathbf{x}, y)$ is denoted $\ell(\theta, z)$. For instance, cross-entropy loss:

$$\ell(\theta, z) := -\sum_j \mathbb{1}\{j = y\} \log p_\theta(j \mid \mathbf{x}) = -\log p_\theta(y \mid \mathbf{x}) \tag{37}$$

The **true risk** $\mathcal{L}^*$ of θ w.r.t. the *ground-truth distribution* $p^*(\mathbf{X}, Y)$ is:

$$\mathcal{L}^*(\theta) := \mathbb{E}_{z \sim p^*}[\ell(\theta, z)] \tag{38}$$

It measures the **true** quality of the model, unobserved.

The **empirical risk** $\widehat{\mathcal{L}}_S$ of θ w.r.t. data set $S = \{z_1, \ldots, z_m\}$ sampled i.i.d. from p^* is:

$$\widehat{\mathcal{L}}_S(\theta) := \frac{1}{|S|} \sum_{z \in S} \ell(\theta, z) \tag{39}$$

It **estimates** the quality of the model from a sample S, optimized during training.

Let $\widehat{\theta}$ be the parameters obtained by training on S and $\widehat{\theta}^{+z}$ those obtained by training on $S \cup \{z\}$, i.e.,

$$\widehat{\theta} := \operatorname*{argmin}_{\theta} \widehat{\mathcal{L}}_S(\theta) \qquad \widehat{\theta}^{+z} := \operatorname*{argmin}_{\theta} \widehat{\mathcal{L}}_{S \cup \{z\}}(\theta) \tag{40}$$

where optimization is possibly *approximate*, e.g., based on SGD.

Let $\widehat{\theta}$ be the parameters obtained by training on S and $\widehat{\theta}^{+z}$ those obtained by training on $S \cup \{z\}$, i.e.,

$$\widehat{\theta} := \underset{\theta}{\text{argmin}} \; \widehat{\mathcal{L}}_S(\theta) \qquad \widehat{\theta}^{+z} := \underset{\theta}{\text{argmin}} \; \widehat{\mathcal{L}}_{S\cup\{z\}}(\theta) \tag{40}$$

where optimization is possibly *approximate*, e.g., based on SGD.

Model Improvement

The **model improvement** (MI) given by a new example $\mathbf{z} \notin S$ is the decrease in true risk:

$$\mathrm{acq}(\mathbf{x}) := \mathcal{L}^*(\widehat{\theta}) - \mathcal{L}^*(\widehat{\theta}^{+\mathbf{z}}) \tag{41}$$

The higher, the better $\longrightarrow$ pick the $\mathbf{x} \in U$ that **maximizes** the improvement.

Model Improvement as Greedy Optimization

- MI amounts to solving:

$$\underset{\mathbf{x}\in U}{\text{argmax}} \; \mathcal{L}^*(\widehat{\theta}) - \mathcal{L}^*(\widehat{\theta}^{+\mathbf{z}}) \quad = \quad \underset{\mathbf{x}\in U}{\text{argmin}} \; \mathcal{L}^*(\widehat{\theta}^{+\mathbf{z}}) \tag{42}$$

It is *guaranteed* to find the best next candidate!

[5]**Note**: MI is *greedy*, *not* optimal! Non-greedy alternatives are conceptually better, but they also computationally infeasible and for this reason they are ignored in the AL literature.

Model Improvement as Greedy Optimization

- MI amounts to solving:

$$\underset{\mathbf{x}\in U}{\text{argmax}}\ \mathcal{L}^*(\widehat{\theta}) - \mathcal{L}^*(\widehat{\theta}^{+\mathbf{z}}) \quad = \quad \underset{\mathbf{x}\in U}{\text{argmin}}\ \mathcal{L}^*(\widehat{\theta}^{+\mathbf{z}}) \tag{42}$$

It is *guaranteed* to find the best next candidate!

- MI is essentially a **greedy strategy** for solving:[5]

$$\underset{S\subseteq U}{\text{argmin}} \quad \mathcal{L}^*(\widehat{\theta}) \tag{43}$$

$$\text{s.t.} \quad |S| \leq \text{query budget} \tag{44}$$

In this view, **AL is a subset optimization problem**, and MI solves it directly.

[5] **Note**: MI is *greedy*, *not* optimal! Non-greedy alternatives are conceptually better, but they also computationally infeasible and for this reason they are ignored in the AL literature.

Model Improvement as Greedy Optimization

- MI amounts to solving:

$$\underset{\mathbf{x}\in U}{\text{argmax}}\ \mathcal{L}^*(\widehat{\theta}) - \mathcal{L}^*(\widehat{\theta}^{+\mathbf{z}}) \quad = \quad \underset{\mathbf{x}\in U}{\text{argmin}}\ \mathcal{L}^*(\widehat{\theta}^{+\mathbf{z}}) \tag{42}$$

It is *guaranteed* to find the best next candidate!

- MI is essentially a **greedy strategy** for solving:[5]

$$\underset{S\subseteq U}{\text{argmin}} \quad \mathcal{L}^*(\widehat{\theta}) \tag{43}$$

$$\text{s.t.} \quad |S| \leq \text{query budget} \tag{44}$$

In this view, **AL is a subset optimization problem**, and MI solves it directly.

- Compare this to *uncertainty sampling*, which is not as sound

[5] **Note**: MI is *greedy*, *not* optimal! Non-greedy alternatives are conceptually better, but they also computationally infeasible and for this reason they are ignored in the AL literature.

- We want to solve:

$$\underset{\mathbf{x} \in U}{\operatorname{argmin}} \; \mathcal{L}^*(\widehat{\theta}^{+z}) \tag{45}$$

[6] The unlabeled set U is ideally pretty large, so the approximation is reasonable.

- We want to solve:

$$\underset{\mathbf{x} \in U}{\operatorname{argmin}} \; \mathcal{L}^*(\widehat{\theta}^{+z}) \tag{45}$$

Problem: $\mathcal{L}^*(\cdot)$ is an **integral over $\mathbf{x}' \in \mathbb{R}^d$**:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) = \mathbb{E}_{z' \sim p^*}[\ell(\widehat{\theta}^{+z}, z')] = \int_{\mathbb{R}^d} \ell(\widehat{\theta}^{+z}, (\mathbf{x}', y'))d\mathbf{x}' \tag{46}$$

which is intractable

[6] The unlabeled set U is ideally pretty large, so the approximation is reasonable.

- We want to solve:

$$\underset{\mathbf{x} \in U}{\text{argmin}} \ \mathcal{L}^*(\widehat{\theta}^{+z}) \tag{45}$$

Problem: $\mathcal{L}^*(\cdot)$ is an **integral over $\mathbf{x}' \in \mathbb{R}^d$**:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) = \mathbb{E}_{z' \sim p^*}[\ell(\widehat{\theta}^{+z}, z')] = \int_{\mathbb{R}^d} \ell(\widehat{\theta}^{+z}, (\mathbf{x}', y'))d\mathbf{x}' \tag{46}$$

which is intractable → **approximate using empirical average over U**:[6]

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \approx \widehat{\mathcal{L}}_U(\widehat{\theta}^{+z}) = \frac{1}{|U|} \sum_{\mathbf{x}' \in U} \ell(\widehat{\theta}^{+z}, (\mathbf{x}', y')) \tag{47}$$

[6] The unlabeled set U is ideally pretty large, so the approximation is reasonable.

- We want to solve:

$$\operatorname*{argmin}_{\mathbf{x} \in U} \; \mathcal{L}^*(\widehat{\theta}^{+z}) \tag{45}$$

Problem: $\mathcal{L}^*(\cdot)$ is an **integral over $\mathbf{x}' \in \mathbb{R}^d$**:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) = \mathbb{E}_{z' \sim p^*}[\ell(\widehat{\theta}^{+z}, z')] = \int_{\mathbb{R}^d} \ell(\widehat{\theta}^{+z}, (\mathbf{x}', y'))d\mathbf{x}' \tag{46}$$

which is intractable $\rightarrow$ **approximate using empirical average over U**:[6]

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \approx \widehat{\mathcal{L}}_U(\widehat{\theta}^{+z}) = \frac{1}{|U|} \sum_{\mathbf{x}' \in U} \ell(\widehat{\theta}^{+z}, (\mathbf{x}', y')) \tag{47}$$

Example: if ℓ is the 0–1 loss, then this amounts to:

$$\frac{1}{|U|} \sum_{\mathbf{x}' \in U} \mathbb{1}\{f_{\widehat{\theta}^{+z}}(\mathbf{x}') \neq y'\} \tag{48}$$

[6] The unlabeled set U is ideally pretty large, so the approximation is reasonable.

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \widehat{\mathcal{L}}_U(\widehat{\theta}^{+z}) = \frac{1}{|U|} \sum_{\mathbf{x}' \in U} \ell(\widehat{\theta}^{+z}, (\mathbf{x}', y')) \tag{49}$$

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \widehat{\mathcal{L}}_U(\widehat{\theta}^{+z}) = \frac{1}{|U|} \sum_{\mathbf{x}' \in U} \ell(\widehat{\theta}^{+z}, (\mathbf{x}', y')) \tag{49}$$

Problem: we don't have access to the ground-truth label $z = (\mathbf{x}, y)$

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \widehat{\mathcal{L}}_U(\widehat{\theta}^{+z}) = \frac{1}{|U|} \sum_{\mathbf{x}' \in U} \ell(\widehat{\theta}^{+z}, (\mathbf{x}', y')) \tag{49}$$

Problem: we don't have access to the ground-truth label $z = (\mathbf{x}, y) \to$ **marginalize w.r.t.** p^*:

$$\mathbb{E}_{y \sim p^*(Y|\mathbf{x})} \left[\frac{1}{|U|} \sum_{\mathbf{x}' \in U} \ell(\widehat{\theta}^{+(\mathbf{x},y)}, (\mathbf{x}', y')) \right] \tag{50}$$

This averages over alternative **future models** $\widehat{\theta}^{+(\mathbf{x},y)}$ obtained after retraining on $L \cup (\mathbf{x}, y)$.

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \widehat{\mathcal{L}}_U(\widehat{\theta}^{+z}) = \frac{1}{|U|} \sum_{\mathbf{x}' \in U} \ell(\widehat{\theta}^{+z}, (\mathbf{x}', y')) \tag{49}$$

Problem: we don't have access to the ground-truth label $z = (\mathbf{x}, y) \rightarrow$ **marginalize w.r.t.** p^*:

$$\mathbb{E}_{y \sim p^*(Y|\mathbf{x})} \left[\frac{1}{|U|} \sum_{\mathbf{x}' \in U} \ell(\widehat{\theta}^{+(\mathbf{x},y)}, (\mathbf{x}', y')) \right] \tag{50}$$

This averages over alternative **future models** $\widehat{\theta}^{+(\mathbf{x},y)}$ obtained after retraining on $L \cup (\mathbf{x}, y)$.

Problem: we don't have access to the ground-truth label y' either

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \longrightarrow \widehat{\mathcal{L}}_U(\widehat{\theta}^{+z}) = \frac{1}{|U|}\sum_{\mathbf{x}'\in U} \ell(\widehat{\theta}^{+z}, (\mathbf{x}', y')) \tag{49}$$

Problem: we don't have access to the ground-truth label $z = (\mathbf{x}, y) \rightarrow$ **marginalize w.r.t.** p^*:

$$\mathbb{E}_{y\sim p^*(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}'\in U} \ell(\widehat{\theta}^{+(\mathbf{x},y)}, (\mathbf{x}', y'))\right] \tag{50}$$

This averages over alternative **future models** $\widehat{\theta}^{+(\mathbf{x},y)}$ obtained after retraining on $L \cup (\mathbf{x}, y)$.

Problem: we don't have access to the ground-truth label y' either $\rightarrow$ **marginalize w.r.t.** p^*:

$$\mathbb{E}_{y\sim p^*(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}'\in U} \mathbb{E}_{y'\sim p^*(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+(\mathbf{x},y)}, (\mathbf{x}', y'))\right]\right] \tag{51}$$

This averages over the **unknown labels** y' of the instances in $\mathbf{x}' \in U$.

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \quad \longrightarrow \quad \mathbb{E}_{y \sim p^*(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \mathbb{E}_{y' \sim p^*(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+(\mathbf{x},y)}, (\mathbf{x}', y'))\right]\right] \tag{52}$$

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \quad \longrightarrow \quad \mathbb{E}_{y\sim p^*(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}'\in U}\mathbb{E}_{y'\sim p^*(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+(\mathbf{x},y)}, (\mathbf{x}', y'))\right]\right] \tag{52}$$

Problem: we don't have access to p^* at all

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y\sim p^*(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}'\in U}\mathbb{E}_{y'\sim p^*(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+(\mathbf{x},y)},(\mathbf{x}',y'))\right]\right] \tag{52}$$

Problem: we don't have access to p^* at all → **estimate using model's distribution**:

$$\underbrace{\mathbb{E}_{y\sim p_{\widehat{\theta}}(Y|\mathbf{x})}\underbrace{\left[\frac{1}{|U|}\sum_{\mathbf{x}'\in U}\underbrace{\mathbb{E}_{y'\sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+},(\mathbf{x}',y'))\right]}_{(a)}\right]}_{(b)}}_{(c)} \tag{53}$$

where $\widehat{\theta}^+ := \widehat{\theta}^{+(\mathbf{x},y)}$. If p_θ is "good enough", then the approximation is good.

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y\sim p^*(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}'\in U}\mathbb{E}_{y'\sim p^*(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+(\mathbf{x},y)},(\mathbf{x}',y'))\right]\right] \tag{52}$$

Problem: we don't have access to p^* at all → **estimate using model's distribution**:

$$\underbrace{\mathbb{E}_{y\sim p_{\widehat{\theta}}(Y|\mathbf{x})}\underbrace{\left[\frac{1}{|U|}\sum_{\mathbf{x}'\in U}\underbrace{\mathbb{E}_{y'\sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^+,(\mathbf{x}',y'))\right]}_{(a)}\right]}_{(b)}}_{(c)} \tag{53}$$

where $\widehat{\theta}^+ := \widehat{\theta}^{+(\mathbf{x},y)}$. If p_θ is "good enough", then the approximation is good.

(a) Is the **expected** loss of the updated model on $\mathbf{x}' \in U$,

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \longrightarrow \mathbb{E}_{y \sim p^*(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \mathbb{E}_{y' \sim p^*(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+(\mathbf{x},y)}, (\mathbf{x}', y'))\right]\right] \tag{52}$$

Problem: we don't have access to p^* at all $\rightarrow$ **estimate using model's distribution**:

$$\underbrace{\mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\Big[\underbrace{\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \underbrace{\mathbb{E}_{y' \sim p_{\widehat{\theta}+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+}, (\mathbf{x}', y'))\right]}_{(a)}}_{(b)}\Big]}_{(c)} \tag{53}$$

where $\widehat{\theta}^{+} := \widehat{\theta}^{+(\mathbf{x},y)}$. If p_θ is "good enough", then the approximation is good.

(a) Is the **expected** loss of the updated model on $\mathbf{x}' \in U$,

(b) Is the **average** expected oss of the updated model on *all* of U,

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \quad \longrightarrow \quad \mathbb{E}_{y \sim p^*(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \mathbb{E}_{y' \sim p^*(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+(\mathbf{x},y)}, (\mathbf{x}', y'))\right]\right] \tag{52}$$

Problem: we don't have access to p^* at all $\rightarrow$ **estimate using model's distribution**:

$$\underbrace{\mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\Bigg[\underbrace{\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \underbrace{\mathbb{E}_{y' \sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^+, (\mathbf{x}', y'))\right]}_{(a)}}_{(b)}\Bigg]}_{(c)} \tag{53}$$

where $\widehat{\theta}^+ := \widehat{\theta}^{+(\mathbf{x},y)}$. If p_θ is "good enough", then the approximation is good.

(a) Is the **expected** loss of the updated model on $\mathbf{x}' \in U$,

(b) Is the **average** expected oss of the updated model on *all* of U,

(c) Is the above **averaged over the possible updated models** $\widehat{\theta}^{+(\mathbf{x},y)}$.

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y\sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}'\in U}\mathbb{E}_{y'\sim p_{\widehat{\theta}^{+}}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+},(\mathbf{x}',y'))\right]\right] \tag{54}$$

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y\sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}'\in U}\mathbb{E}_{y'\sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^+,(\mathbf{x}',y'))\right]\right] \tag{54}$$

Example: consider the 0–1 loss $\ell(\theta,(\mathbf{x},y)) = \mathbb{1}\{f_\theta(\mathbf{x}) \neq y\}$.

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \mathbb{E}_{y' \sim p_{\widehat{\theta}+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+}, (\mathbf{x}', y'))\right]\right] \tag{54}$$

Example: consider the 0–1 loss $\ell(\theta, (\mathbf{x}, y)) = \mathbb{1}\{f_\theta(\mathbf{x}) \neq y\}$. Then:

$$\mathbb{E}_{y' \sim p_{\widehat{\theta}+}(Y|\mathbf{x}')}\left[\mathbb{1}\{f_{\widehat{\theta}+}(\mathbf{x}') \neq y'\}\right] = p_{\widehat{\theta}+}(\hat{y}' \neq y' \mid \mathbf{x}'), \qquad \hat{y}' := f_{\widehat{\theta}+}(\mathbf{x}') \tag{55}$$

$$= 1 - p_{\widehat{\theta}+}(\hat{y}' \mid \mathbf{x}') \tag{56}$$

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y\sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}'\in U}\mathbb{E}_{y'\sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^+,(\mathbf{x}',y'))\right]\right] \tag{54}$$

Example: consider the <u>0–1 loss</u> $\ell(\theta,(\mathbf{x},y)) = \mathbb{1}\{f_\theta(\mathbf{x}) \neq y\}$. Then:

$$\mathbb{E}_{y'\sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\mathbb{1}\{f_{\widehat{\theta}^+}(\mathbf{x}') \neq y'\}\right] = p_{\widehat{\theta}^+}(\hat{y}' \neq y' \mid \mathbf{x}'), \qquad \hat{y}' := f_{\widehat{\theta}^+}(\mathbf{x}') \tag{55}$$

$$= 1 - p_{\widehat{\theta}^+}(\hat{y}' \mid \mathbf{x}') \tag{56}$$

Hence, the above can be rewritten as ($\frac{1}{|U|}$ doesn't matter because it is independent of $\mathbf{x}$):

$$\mathbb{E}_{y\sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}'\in U}\left(1 - p_{\widehat{\theta}^+}(\hat{y}' \mid \mathbf{x}')\right)\right] \quad \propto \quad \sum_{y\in[c]} p_{\widehat{\theta}}(y \mid \mathbf{x}) \sum_{\mathbf{x}'\in U}\left(1 - p_{\widehat{\theta}^+}(\hat{y}' \mid \mathbf{x}')\right) \tag{57}$$

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \mathbb{E}_{y' \sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^+, (\mathbf{x}', y'))\right]\right] \tag{54}$$

Example: consider the 0–1 loss $\ell(\theta, (\mathbf{x}, y)) = \mathbb{1}\{f_\theta(\mathbf{x}) \neq y\}$. Then:

$$\mathbb{E}_{y' \sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\mathbb{1}\left\{f_{\widehat{\theta}^+}(\mathbf{x}') \neq y'\right\}\right] = p_{\widehat{\theta}^+}(\hat{y}' \neq y' \mid \mathbf{x}'), \qquad \hat{y}' := f_{\widehat{\theta}^+}(\mathbf{x}') \tag{55}$$

$$= 1 - p_{\widehat{\theta}^+}(\hat{y}' \mid \mathbf{x}') \tag{56}$$

Hence, the above can be rewritten as ($\frac{1}{|U|}$ doesn't matter because it is independent of $\mathbf{x}$):

$$\mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U}\left(1 - p_{\widehat{\theta}^+}(\hat{y}' \mid \mathbf{x}')\right)\right] \quad \propto \quad \sum_{y \in [c]} p_{\widehat{\theta}}(y \mid \mathbf{x}) \sum_{\mathbf{x}' \in U}\left(1 - p_{\widehat{\theta}^+}(\hat{y}' \mid \mathbf{x}')\right) \tag{57}$$

- We pick $\mathbf{x} \in U$ that minimizes the above $\rightarrow$ **minimizes expected future confidence on** U

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \mathbb{E}_{y' \sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^+, (\mathbf{x}', y'))\right]\right] \tag{58}$$

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \mathbb{E}_{y' \sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^+, (\mathbf{x}', y'))\right]\right] \tag{58}$$

Example: consider the negative log-likelihood $\ell(\theta, (\mathbf{x}, y)) = -\log p_\theta(y \mid \mathbf{x})$.

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \mathbb{E}_{y' \sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^+, (\mathbf{x}', y'))\right]\right] \tag{58}$$

Example: consider the negative log-likelihood $\ell(\theta, (\mathbf{x}, y)) = -\log p_\theta(y \mid \mathbf{x})$. Then:

$$\mathbb{E}_{y' \sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[-\log p_{\widehat{\theta}^+}(y' \mid \mathbf{x}')\right] = -\sum_{y' \in [c]} p_{\widehat{\theta}^+}(y' \mid \mathbf{x}') \log p_{\widehat{\theta}^+}(y' \mid \mathbf{x}') \tag{59}$$

$$= H_{\widehat{\theta}^+}(Y \mid \mathbf{x}) \tag{60}$$

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \mathbb{E}_{y' \sim p_{\widehat{\theta}+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+}, (\mathbf{x}', y'))\right]\right] \tag{58}$$

Example: consider the negative log-likelihood $\ell(\theta, (\mathbf{x}, y)) = -\log p_\theta(y \mid \mathbf{x})$. Then:

$$\mathbb{E}_{y' \sim p_{\widehat{\theta}+}(Y|\mathbf{x}')}\left[-\log p_{\widehat{\theta}+}(y' \mid \mathbf{x}')\right] = -\sum_{y' \in [c]} p_{\widehat{\theta}+}(y' \mid \mathbf{x}') \log p_{\widehat{\theta}+}(y' \mid \mathbf{x}') \tag{59}$$

$$= H_{\widehat{\theta}+}(Y \mid \mathbf{x}) \tag{60}$$

Hence, the above can be rewritten as:

$$\mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U}\left(H_{\widehat{\theta}+}(Y \mid \mathbf{x})\right)\right] \quad \propto \quad \sum_{y \in [c]} p_{\widehat{\theta}}(y \mid \mathbf{x}) \sum_{\mathbf{x}' \in U}\left(H_{\widehat{\theta}+}(Y \mid \mathbf{x})\right) \tag{61}$$

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \mathbb{E}_{y' \sim p_{\widehat{\theta}+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+}, (\mathbf{x}', y'))\right]\right] \tag{58}$$

Example: consider the negative log-likelihood $\ell(\theta, (\mathbf{x}, y)) = -\log p_\theta(y \mid \mathbf{x})$. Then:

$$\mathbb{E}_{y' \sim p_{\widehat{\theta}+}(Y|\mathbf{x}')}\left[-\log p_{\widehat{\theta}+}(y' \mid \mathbf{x}')\right] = -\sum_{y' \in [c]} p_{\widehat{\theta}+}(y' \mid \mathbf{x}') \log p_{\widehat{\theta}+}(y' \mid \mathbf{x}') \tag{59}$$

$$= H_{\widehat{\theta}+}(Y \mid \mathbf{x}) \tag{60}$$

Hence, the above can be rewritten as:

$$\mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U}\left(H_{\widehat{\theta}+}(Y \mid \mathbf{x})\right)\right] \quad \propto \quad \sum_{y \in [c]} p_{\widehat{\theta}}(y \mid \mathbf{x}) \sum_{\mathbf{x}' \in U}\left(H_{\widehat{\theta}+}(Y \mid \mathbf{x})\right) \tag{61}$$

- We pick $\mathbf{x} \in U$ that minimizes the above $\rightarrow$ **minimizes expected future entropy on** U

- In uncertainty sampling, we pick $\mathbf{x}$ that minimizes model's estimate of current uncertainty w.r.t. itself, this is **myopic**

- In expected model improvement, we pick $\mathbf{x}$ that minimizes model's estimate of expected future uncertainty w.r.t. unlabeled set, this is **less myopic**

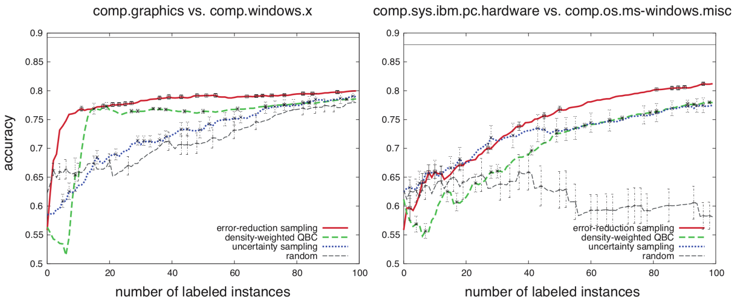


Figure 4.1: Learning curves showing that expected error reduction can outperform QBC and uncertainty sampling for two binary text classification tasks. *Source*: Adapted from Roy and McCallum (2001), with kind permission of the authors.

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \mathbb{E}_{y' \sim p_{\widehat{\theta}^{+}}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^{+}, (\mathbf{x}', y'))\right]\right] \tag{62}$$

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \quad \longrightarrow \quad \mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \mathbb{E}_{y' \sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^+, (\mathbf{x}', y'))\right]\right] \tag{62}$$

Problem: computing $\widehat{\theta}^+$ requires to **fit model on** $L \cup \{(\mathbf{x}, y)\}$ (**slow**)

We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathrm{x})}\left[\frac{1}{|U|}\sum_{\mathrm{x}' \in U} \mathbb{E}_{y' \sim p_{\widehat{\theta}^+}(Y|\mathrm{x}')}\left[\ell(\widehat{\theta}^+, (\mathrm{x}', y'))\right]\right] \tag{62}$$

Problem: computing $\widehat{\theta}^+$ requires to **fit model on** $L \cup \{(\mathrm{x}, y)\}$ (**slow**)

Problem: this has to be done $|U| \times [c]$ times.

- We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \qquad \longrightarrow \qquad \mathbb{E}_{y\sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}'\in U}\mathbb{E}_{y'\sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^+,(\mathbf{x}',y'))\right]\right] \tag{62}$$

Problem: computing $\widehat{\theta}^+$ requires to **fit model on** $L \cup \{(\mathbf{x}, y)\}$ (**slow**)

Problem: this has to be done $|U| \times [c]$ times.

Problem: this has to be done in each iteration of active learning.

■ We already decided on this approximation:

$$\mathcal{L}^*(\widehat{\theta}^{+z}) \quad \longrightarrow \quad \mathbb{E}_{y \sim p_{\widehat{\theta}}(Y|\mathbf{x})}\left[\frac{1}{|U|}\sum_{\mathbf{x}' \in U} \mathbb{E}_{y' \sim p_{\widehat{\theta}^+}(Y|\mathbf{x}')}\left[\ell(\widehat{\theta}^+, (\mathbf{x}', y'))\right]\right] \tag{62}$$

Problem: computing $\widehat{\theta}^+$ requires to **fit model on** $L \cup \{(\mathbf{x}, y)\}$ (**slow**)

Problem: this has to be done $|U| \times [c]$ times.

Problem: this has to be done in each iteration of active learning.

■ Only **practical** for classes of models that support **closed-form updates** (e.g., Gaussian Processes) or stable **incremental learning** (e.g., perceptron-like learning algorithms).

Expected Model Change

- Unless a candidate $(\mathbf{x}, y)$ induces a large change in the model $\widehat{\theta}$ upon retraining, then it cannot reduce the model's risk by much: **change is a prerequisite for improvement**.

Expected Model Change

- Unless a candidate (x, y) induces a large change in the model $\widehat{\theta}$ upon retraining, then it cannot reduce the model's risk by much: **change is a prerequisite for improvement**.

Inituition:

$$\ell(\widehat{\theta}, z') - \ell(\widehat{\theta}^{+z}, z') \quad \leq \quad |\ell(\widehat{\theta}, z') - \ell(\widehat{\theta}^{+z}, z')| \quad \leq \quad c \cdot \|\widehat{\theta} - \widehat{\theta}^{+z}\|, \qquad c > 0 \tag{63}$$

where $\|\cdot\|$ is, e.g., the Euclidean norm. This formally holds for all c-Lipshitz loss functions ℓ.

- Unless a candidate $(\mathbf{x}, y)$ induces a large change in the model $\widehat{\theta}$ upon retraining, then it cannot reduce the model's risk by much: **change is a prerequisite for improvement**.

Inituition:

$$\ell(\widehat{\theta}, z') - \ell(\widehat{\theta}^{+z}, z') \quad \leq \quad |\ell(\widehat{\theta}, z') - \ell(\widehat{\theta}^{+z}, z')| \quad \leq \quad c \cdot \|\widehat{\theta} - \widehat{\theta}^{+z}\|, \qquad c > 0 \tag{63}$$

where $\|\cdot\|$ is, e.g., the Euclidean norm. This formally holds for all c-Lipshitz loss functions ℓ.

- Large change also occurs when the loss *increases* – hence the absolute value in the second step of Eq. 63.

All in all, EMC looks for examples $\mathbf{x} \in U$ that "make a difference" one way or the other.

But once $(\mathbf{x}, y)$ is acquried it is added to the training set L on which $\widehat{\theta}$ is fit, so loss is likely to go *down* rather than *up*.

Expected Model Change

- The trick is that if $\widehat{\theta}$ is obtained via **gradient descent**, the difference $\widehat{\theta} - \widehat{\theta}^{+z}$ is easy to compute:

$$\widehat{\theta} - \widehat{\theta}^{+z} = \eta \cdot \nabla_{\theta} \ell(\theta, z) \tag{64}$$

where η is the learning rate. This gives **expected gradient length**:

$$\mathrm{acq}_{\mathsf{EGL}}(\mathbf{x}) := \mathbb{E}_{y \sim p_{\theta}(Y|\mathbf{x})} \left[\|\nabla_{\theta} \ell(\widehat{\theta}, (\mathbf{x}, y))\|^2 \right] \tag{65}$$

The square does not change ranking of examples & avoids computing a square root.

- Quite cheap to compute using automatic differentiation packages (using Jacobian to parallelize over U)
- Assuming η is constant across examples and GD, the computation is exact. For other optimizers, it is an approximation

Are Uncertain Points Representative?

Diversity-based Selection

Idea: pick instances $\mathbf{x} \in U$ that are *both* locally informative and also similar to as many other unlabeled points as possible:

$$\underset{\mathbf{x} \in U}{\text{argmax}} \; \mathrm{acq}(f, \mathbf{x}) \cdot \left(\frac{1}{|U|} \sum_{\mathbf{x}' \in U} \mathrm{sim}(\mathbf{x}, \mathbf{x}') \right)^{\beta} \tag{66}$$

where:

- $\mathrm{acq}(f, \mathbf{x})$ is a "standard" acquisition function based on, e.g., pointwise uncertainty.
- $\mathrm{sim}(\mathbf{x}, \mathbf{x}')$ measures the similarity between $\mathbf{x}$ and $\mathbf{x}'$, e.g., a Gaussian kernel, Pearson's correlation coefficient, Spearman's rank correlation. **Application specific**.
- $\beta > 0$ is a hyper-parameter

Intuitively, $\mathbf{x}$'s label conveys information about the label on the other points in U

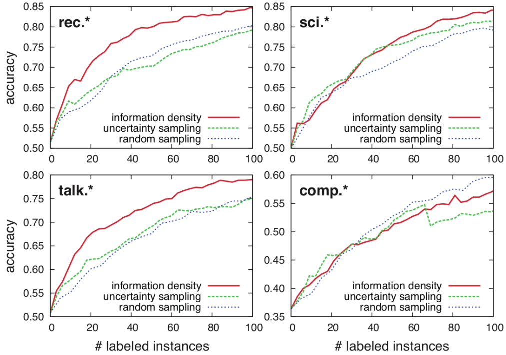


Figure 5.2: Learning curves showing that, by explicitly weighting queries by their representativeness among the input instances, information density can yield better results than the base uncertainty sampling heuristic by itself.

- We optimize:

$$\underset{\mathbf{x} \in U}{\text{argmax}} \; \mathrm{acq}(f, \mathbf{x}) \cdot \left(\frac{1}{|U|} \sum_{\mathbf{x}' \in U} \mathrm{sim}(\mathbf{x}, \mathbf{x}') \right)^{\beta} \tag{67}$$

Properties:

- Tends to work better than pure more "local" acquisition functions [Settles, 2012]
- Even when uncertainty sampling is worse than random, information density performs well

- We optimize:

$$\underset{\mathbf{x} \in U}{\text{argmax}} \ \mathrm{acq}(f, \mathbf{x}) \cdot \left(\frac{1}{|U|} \sum_{\mathbf{x}' \in U} \mathrm{sim}(\mathbf{x}, \mathbf{x}') \right)^{\beta} \tag{67}$$

Properties:

- Tends to work better than pure more “local” acquisition functions [Settles, 2012]
- Even when uncertainty sampling is worse than random, information density performs well
- Similarity computation can be sped-up using **caching**: “simply” store similarity matrix $S_{ij} = [\mathrm{sim}(\mathbf{x}_i, \mathbf{x}_j)]$ for all $\mathbf{x}_i, \mathbf{x}_j \in U$ (only needs to be done **once**)

- We optimize:

$$\underset{\mathbf{x}\in U}{\text{argmax}}\ \mathrm{acq}(f,\mathbf{x})\cdot\left(\frac{1}{|U|}\sum_{\mathbf{x}'\in U}\mathrm{sim}(\mathbf{x},\mathbf{x}')\right)^{\beta} \quad (67)$$

Properties:

- Tends to work better than pure more “local” acquisition functions [Settles, 2012]
- Even when uncertainty sampling is worse than random, information density performs well
- Similarity computation can be sped-up using **caching**: “simply” store similarity matrix $S_{ij} = [\mathrm{sim}(\mathbf{x}_i,\mathbf{x}_j)]$ for all $\mathbf{x}_i,\mathbf{x}_j \in U$ (only needs to be done **once**)
- Approximate using **clustering**: cluster U so that points within cluster are similar and points across clusters are not $\rightarrow$ block-diagonal similarity matrix, lowers storage requirement from $O(|U|^2)$ to $O(\sum_i |\mathrm{cluster}_i|^2)$

- Do we gain anything by “summarizing” the data using **clustering**?

- Do we gain anything by "summarizing" the data using **clustering**?

Idea:

- Cluster unlabeled data set $U \to \{C_i \subset U \ : \ i \in [k]\}$
- Treat each C_i as a separate problem, e.g., **query cluster centroids**

■ Do we gain anything by "summarizing" the data using **clustering**?

Idea:

- Cluster unlabeled data set $U \to \{C_i \subset U \ : \ i \in [k]\}$
- Treat each C_i as a separate problem, e.g., **query cluster centroids**

Problems:

- U may not have a good clustering structure or $\mathrm{sim}(\cdot, \cdot)$ may not be able to capture it
- How many clusters and at what granularity?
- Clusters of x's may not correlate well with label y.

Figure: the swiss roll dataset has no obvious clustering structure.

Unified Derivation

WRITEME

Extensions

- Consider a neural network $f_\theta : \mathbb{R}^d \rightarrow [c]$:

$$f_\theta(\mathbf{x}) = \underset{y \in [c]}{\text{argmax}} \; p_\theta(y \mid \mathbf{x})$$

$$p_\theta(y \mid \mathbf{x}) = \mathrm{softmax}(W\phi_\omega(\mathbf{x}))_y$$

where:

- $\theta = \{W, \omega\}$ are parameters
- $\phi_\omega : \mathbb{R}^d \rightarrow \mathbb{R}^k$ is an embedding function (e.g., convolutions + pooling layers)
- $W \in \mathbb{R}^{c \times k}$ are the parameters of the top dense layer

- Deep NNs have a number of **quirks**:

- Deep NNs have a number of **quirks**:
  - Very **overconfident** even away from the training set: their *uncertainty* cannot be trusted $\rightarrow$ strategies based on confidence, margin, entropy will underperform (including uncertainty sampling, model improvement, density-aware sampling, etc.)

- Deep NNs have a number of **quirks**:
  - Very **overconfident** even away from the training set: their *uncertainty* cannot be trusted $\rightarrow$ strategies based on confidence, margin, entropy will underperform (including uncertainty sampling, model improvement, density-aware sampling, etc.)
  - **Expensive** to fit on data: training a ResNet on a realistic data set can take minutes to days $\rightarrow$ hard to ensure responsivity

- Deep NNs have a number of **quirks**:
  - Very **overconfident** even away from the training set: their *uncertainty* cannot be trusted → strategies based on confidence, margin, entropy will underperform (including uncertainty sampling, model improvement, density-aware sampling, etc.)
  - **Expensive** to fit on data: training a ResNet on a realistic data set can take minutes to days → hard to ensure responsivity
  - Quite **insensitive** to the addition of a single example → what's the point of querying individual instances?

- Deep NNs have a number of **quirks**:
  - Very **overconfident** even away from the training set: their *uncertainty* cannot be trusted → strategies based on confidence, margin, entropy will underperform (including uncertainty sampling, model improvement, density-aware sampling, etc.)
  - **Expensive** to fit on data: training a ResNet on a realistic data set can take minutes to days → hard to ensure responsivity
  - Quite **insensitive** to the addition of a single example → what's the point of querying individual instances?
  - Training is **stochastic** (i.e., not 100% stable) → changes in performance can depend on factors other than new labeled examples, high variance

Overconfidence

Problem: Deep NNs tend to be very **overconfident** even away from the training set → their *uncertainty* cannot be trusted

Overconfidence

Problem: Deep NNs tend to be very **overconfident** even away from the training set → their *uncertainty* cannot be trusted

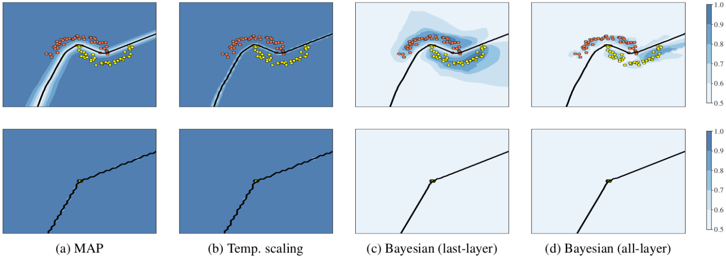


Figure 1. Binary classification on a toy dataset using a MAP estimate, temperature scaling, and both last-layer and all-layer Gaussian approximations over the weights which are obtained via Laplace approximations. Background color and black line represent confidence and decision boundary, respectively. Bottom row shows a zoomed-out view of the top row. The Bayesian approximations—even in the last-layer case—give desirable uncertainty estimates: confident close to the training data and uncertain otherwise. MAP and temperature scaling yield overconfident predictions. The optimal temperature is picked as in Guo et al. (2017).

Credit: [Kristiadi et al., 2020].

Aleatoric vs Epistemic [Hüllermeier and Waegeman, 2021]

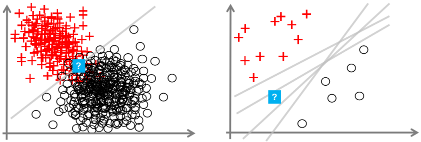


Figure 5: Left: Even with precise knowledge about the optimal hypothesis, the prediction at the query point (indicated by a question mark) is aleatorically uncertain, because the two classes are overlapping in that region. Right: A case of epistemic uncertainty due to a lack of knowledge about the right hypothesis, which is in turn caused by a lack of data.

Aleatoric vs Epistemic [Hüllermeier and Waegeman, 2021]

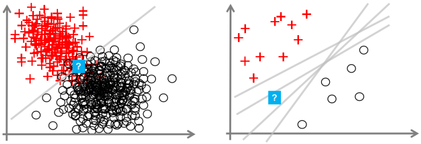


Figure 5: Left: Even with precise knowledge about the optimal hypothesis, the prediction at the query point (indicated by a question mark) is aleatorically uncertain, because the two classes are overlapping in that region. Right: A case of epistemic uncertainty due to a lack of knowledge about the right hypothesis, which is in turn caused by a lack of data.

- **Aleatoric** uncertainty (“random”) captures how much we can trust the supervision itself. It cannot be decreased. (**left**)

Aleatoric vs Epistemic [Hüllermeier and Waegeman, 2021]

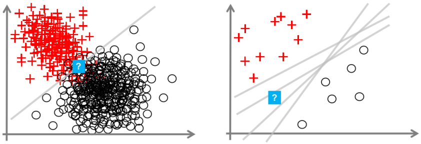


Figure 5: Left: Even with precise knowledge about the optimal hypothesis, the prediction at the query point (indicated by a question mark) is aleatorically uncertain, because the two classes are overlapping in that region. Right: A case of epistemic uncertainty due to a lack of knowledge about the right hypothesis, which is in turn caused by a lack of data.

- **Aleatoric** uncertainty ("random") captures how much we can trust the supervision itself. It cannot be decreased. (**left**)

- **Epistemic** uncertainty ("relating to knowledge") captures how little we know about the world. This reflects on uncertainty on the choice of θ. It decreases by acquiring **more data**. (**right**)

Aleatoric vs Epistemic [Hüllermeier and Waegeman, 2021]

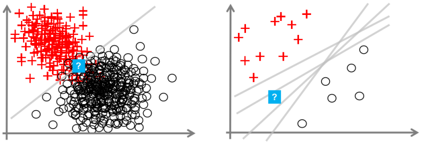


Figure 5: Left: Even with precise knowledge about the optimal hypothesis, the prediction at the query point (indicated by a question mark) is aleatorically uncertain, because the two classes are overlapping in that region. Right: A case of epistemic uncertainty due to a lack of knowledge about the right hypothesis, which is in turn caused by a lack of data.

- **Aleatoric** uncertainty ("random") captures how much we can trust the supervision itself. It cannot be decreased. (**left**)

- **Epistemic** uncertainty ("relating to knowledge") captures how little we know about the world. This reflects on uncertainty on the choice of θ. It decreases by acquiring **more data**. (**right**)

- There isn't much point in trying to reduce **aleatoric** uncertainty in AL [Sharma and Bilgic, 2017]

Bayesian NNs

- The problem with NNs is that uncertainty depends on a *single model*:
  - This gives poor *epistemic* uncertainty

Bayesian NNs

- The problem with NNs is that uncertainty depends on a *single model*:
  - This gives poor *epistemic* uncertainty
  - Using **ensambles of NNs** is computationally challenging: training one NN is expensive, training k even more so

Bayesian NNs

- The problem with NNs is that uncertainty depends on a *single model*:
  - This gives poor *epistemic* uncertainty
  - Using **ensambles of NNs** is computationally challenging: training one NN is expensive, training k even more so
  - Using **Bayesian** techniques – i.e., maintaining a distribution over alternative NNs – is also challenging.

Bayesian NNs

- The problem with NNs is that uncertainty depends on a *single model*:
  - This gives poor *epistemic* uncertainty
  - Using **ensambles of NNs** is computationally challenging: training one NN is expensive, training k even more so
  - Using **Bayesian** techniques – i.e., maintaining a distribution over alternative NNs – is also challenging.

Idea of **Bayesian NNs**:

- Replace parameters θ with distribution over alternative parameters $p(\theta \mid L)$

- The problem with NNs is that uncertainty depends on a *single model*:
  - This gives poor *epistemic* uncertainty
  - Using **ensambles of NNs** is computationally challenging: training one NN is expensive, training k even more so
  - Using **Bayesian** techniques – i.e., maintaining a distribution over alternative NNs – is also challenging.

Idea of **Bayesian NNs**:

- Replace parameters θ with distribution over alternative parameters $p(\theta \mid L)$
- Compute predictions by marginalizing over θ:

$$p(y \mid \mathbf{x}) = \int \underbrace{p(y \mid \mathbf{x}, \theta)}_{\text{NN with params } \theta} \cdot \underbrace{p(\theta \mid L)}_{\text{posterior over params}} d\theta \tag{68}$$

Bayesian NNs

- The problem with NNs is that uncertainty depends on a *single model*:
  - This gives poor *epistemic* uncertainty
  - Using **ensambles of NNs** is computationally challenging: training one NN is expensive, training k even more so
  - Using **Bayesian** techniques – i.e., maintaining a distribution over alternative NNs – is also challenging.

Idea of **Bayesian NNs**:

- Replace parameters θ with distribution over alternative parameters $p(\theta \mid L)$
- Compute predictions by marginalizing over θ:

$$p(y \mid \mathbf{x}) = \int \underbrace{p(y \mid \mathbf{x}, \theta)}_{\text{NN with params } \theta} \cdot \underbrace{p(\theta \mid L)}_{\text{posterior over params}} d\theta \quad (68)$$

- Learn by updating distribution:

$$p(\theta \mid L) \quad \rightarrow \quad p(\theta \mid L \cup \{(\mathbf{x}, y)\}) \quad (69)$$

Not trivial! Is there an **efficient approximation**?

Dropout

- Randomly set nodes to 0 with a fixed probability.

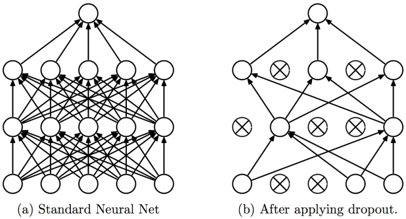


Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

- Used as a **regularization technique**: by randomly removing neurons, prevents them from relying on each other "too much"

- Computing class probabilities:

$$p(y \mid \mathrm{x}, L) = \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid L) d\boldsymbol{\theta} \tag{70}$$

$$\approx \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p_{\mathrm{dropout}}(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{71}$$

$$\approx \frac{1}{R} \sum_{r=1}^{R} p(y \mid \mathrm{x}, \widehat{\boldsymbol{\theta}}_r), \qquad \widehat{\boldsymbol{\theta}}_r \sim p_{\mathrm{dropout}}(\boldsymbol{\theta}) \tag{72}$$

In other words, run NN R times with **dropout enabled** (during inference!) then average the R vectors of class probabilities.

Dropout as Bayesian Approximation

- Computing class probabilities:

$$p(y \mid \mathrm{x}, L) = \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid L) d\boldsymbol{\theta} \tag{70}$$

$$\approx \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p_{\mathrm{dropout}}(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{71}$$

$$\approx \frac{1}{R} \sum_{r=1}^{R} p(y \mid \mathrm{x}, \widehat{\boldsymbol{\theta}}_r), \qquad \widehat{\boldsymbol{\theta}}_r \sim p_{\mathrm{dropout}}(\boldsymbol{\theta}) \tag{72}$$

In other words, run NN R times with **dropout enabled** (during inference!) then average the R vectors of class probabilities.

- Dropout can be viewed as variational Bayesian approximation where the approximating distribution is a mixture of two Gaussians [Gal and Ghahramani, 2016].

Dropout as Bayesian Approximation

- Computing class probabilities:

$$p(y \mid \mathrm{x}, L) = \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid L) d\boldsymbol{\theta} \tag{70}$$

$$\approx \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p_{\mathrm{dropout}}(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{71}$$

$$\approx \frac{1}{R} \sum_{r=1}^{R} p(y \mid \mathrm{x}, \widehat{\boldsymbol{\theta}}_r), \qquad \widehat{\boldsymbol{\theta}}_r \sim p_{\mathrm{dropout}}(\boldsymbol{\theta}) \tag{72}$$

In other words, run NN R times with **dropout enabled** (during inference!) then average the R vectors of class probabilities.

- Dropout can be viewed as variational Bayesian approximation where the approximating distribution is a mixture of two Gaussians [Gal and Ghahramani, 2016]. The approximation is **independent** of $L \rightarrow$ no training required.

- Computing class probabilities:

$$
\begin{aligned}
p(y \mid \mathrm{x}, L) &= \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid L) d\boldsymbol{\theta} && (70)\\
&\approx \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p_{\mathrm{dropout}}(\boldsymbol{\theta}) d\boldsymbol{\theta} && (71)\\
&\approx \frac{1}{R} \sum_{r=1}^{R} p(y \mid \mathrm{x}, \widehat{\boldsymbol{\theta}}_r), \qquad \widehat{\boldsymbol{\theta}}_r \sim p_{\mathrm{dropout}}(\boldsymbol{\theta}) && (72)
\end{aligned}
$$

In other words, run NN R times with **dropout enabled** (during inference!) then average the R vectors of class probabilities.

- Dropout can be viewed as variational Bayesian approximation where the approximating distribution is a mixture of two Gaussians [Gal and Ghahramani, 2016]. The approximation is **independent** of $L \rightarrow$ no training required.

- Immediately leads to **more calibrated output probabilities**!

BALD [Gal et al., 2017]

Question: does dropout help with query selection too?

BALD [Gal et al., 2017]

Question: does dropout help with query selection too? **Yes.**

- Uncertainty sampling:

$$\mathrm{acq}_{UNC}(\mathbf{x}) = -\sum_{y\in[c]} p(Y = y \mid \mathbf{x}, L) \log p(Y = y \mid \mathbf{x}, L) \tag{73}$$

Application is immediate.

BALD [Gal et al., 2017]

Question: does dropout help with query selection too? **Yes.**

- Uncertainty sampling:

$$\mathrm{acq}_{UNC}(\mathbf{x}) = -\sum_{y \in [c]} p(Y = y \mid \mathbf{x}, L) \log p(Y = y \mid \mathbf{x}, L) \tag{73}$$

Application is immediate.

- Mutual information between predictions and model posterior (**BALD**):

$$\mathrm{acq}_{BALD}(\mathbf{x}) = H(Y \mid \mathbf{x}, L) - \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid L)}[H(Y \mid \mathbf{x}, \boldsymbol{\theta})] \tag{74}$$

BALD [Gal et al., 2017]

Question: does dropout help with query selection too? **Yes.**

- Uncertainty sampling:

$$\mathrm{acq}_{UNC}(\mathbf{x}) = -\sum_{y \in [c]} p(Y = y \mid \mathbf{x}, L) \log p(Y = y \mid \mathbf{x}, L) \tag{73}$$

Application is immediate.

- Mutual information between predictions and model posterior (**BALD**):

$$\mathrm{acq}_{BALD}(\mathbf{x}) = H(Y \mid \mathbf{x}, L) - \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid L)}[H(Y \mid \mathbf{x}, \boldsymbol{\theta})] \tag{74}$$

- Left: entropy of the prediction $\rightarrow$ high when the model's prediction is uncertain

BALD [Gal et al., 2017]

Question: does dropout help with query selection too? **Yes.**

- Uncertainty sampling:

$$\mathrm{acq}_{UNC}(\mathbf{x}) = -\sum_{y \in [c]} p(Y = y \mid \mathbf{x}, L) \log p(Y = y \mid \mathbf{x}, L) \tag{73}$$

Application is immediate.

- Mutual information between predictions and model posterior (**BALD**):

$$\mathrm{acq}_{BALD}(\mathbf{x}) = H(Y \mid \mathbf{x}, L) - \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid L)}[H(Y \mid \mathbf{x}, \boldsymbol{\theta})] \tag{74}$$

- Left: entropy of the prediction $\rightarrow$ high when the model's prediction is uncertain
- Right: expected entropy of the prediction over the posterior of the model parameters $\rightarrow$ low when the model is overall certain for each draw of model parameters from the posterior.

BALD [Gal et al., 2017]

Question: does dropout help with query selection too? **Yes.**

- Uncertainty sampling:

$$\mathrm{acq}_{UNC}(\mathbf{x}) = -\sum_{y \in [c]} p(Y = y \mid \mathbf{x}, L) \log p(Y = y \mid \mathbf{x}, L) \tag{73}$$

Application is immediate.

- Mutual information between predictions and model posterior (**BALD**):

$$\mathrm{acq}_{BALD}(\mathbf{x}) = H(Y \mid \mathbf{x}, L) - \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid L)}[H(Y|\mathbf{x}, \boldsymbol{\theta})] \tag{74}$$

- Left: entropy of the prediction → high when the model's prediction is uncertain
- Right: expected entropy of the prediction over the posterior of the model parameters → low when the model is overall certain for each draw of model parameters from the posterior.

High when model has many possible ways to explain the data, i.e., **the posterior draws are disagreeing among themselves**.

- **BALD** can be computed as:

$$
\begin{aligned}
H(Y \mid \mathrm{x}, L) &- \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid L)}[H(Y \mid \mathrm{x}, \boldsymbol{\theta})] \\
&= -\sum_{y \in [c]} p(y \mid \mathrm{x}, L) \log p(y \mid \mathrm{x}, L) + \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid L)}[\sum_{y \in [c]} p(y \mid \mathrm{x}, \boldsymbol{\theta}) \log p(y \mid \mathrm{x}, \boldsymbol{\theta})] \\
&= -\sum_{y \in [c]} \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid L) d\boldsymbol{\theta} \cdot \log \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid L) d\boldsymbol{\theta} + \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid L)}[\sum_{y \in [c]} p(y \mid \mathrm{x}, \boldsymbol{\theta}) \log p(y \mid \mathrm{x}, \boldsymbol{\theta})] \\
&\approx -\sum_{y \in [c]} \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p_{drop}(\boldsymbol{\theta}) d\boldsymbol{\theta} \cdot \log \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p_{drop}(\boldsymbol{\theta}) d\boldsymbol{\theta} + \mathbb{E}_{\boldsymbol{\theta} \sim p_{drop}(\boldsymbol{\theta})}[\sum_{y \in [c]} p(y \mid \mathrm{x}, \boldsymbol{\theta}) \log p(y \mid \mathrm{x}, \boldsymbol{\theta})] \\
&\approx -\sum_{y \in [c]} \left(\frac{1}{R} \sum_r \hat{p}_y^r \right) \log \left(\frac{1}{R} \sum_r \hat{p}_y^r \right) + \frac{1}{R} \sum_{y,r} \hat{p}_y^r \log \hat{p}_y^r,
\end{aligned}
$$

where $\hat{p}_y^r = p(y \mid \mathrm{x}, \boldsymbol{\theta}_r)$ is the output of the NN's softmax for class y and randomized parameters $\boldsymbol{\theta}_r \sim p_{drop}(\boldsymbol{\theta})$.

- **BALD** can be computed as:

$$
\begin{aligned}
H(Y \mid \mathrm{x}, L) &- \mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|L)}[H(Y|\mathrm{x}, \boldsymbol{\theta})] \\
&= -\sum_{y\in[c]} p(y \mid \mathrm{x}, L) \log p(y \mid \mathrm{x}, L) + \mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|L)}[\sum_{y\in[c]} p(y \mid \mathrm{x}, \boldsymbol{\theta}) \log p(y \mid \mathrm{x}, \boldsymbol{\theta})] \\
&= -\sum_{y\in[c]} \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid L) d\boldsymbol{\theta} \cdot \log \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid L) d\boldsymbol{\theta} + \mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|L)}[\sum_{y\in[c]} p(y \mid \mathrm{x}, \boldsymbol{\theta}) \log p(y \mid \mathrm{x}, \boldsymbol{\theta})] \\
&\approx -\sum_{y\in[c]} \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p_{drop}(\boldsymbol{\theta}) d\boldsymbol{\theta} \cdot \log \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p_{drop}(\boldsymbol{\theta}) d\boldsymbol{\theta} + \mathbb{E}_{\boldsymbol{\theta}\sim p_{drop}(\boldsymbol{\theta})}[\sum_{y\in[c]} p(y \mid \mathrm{x}, \boldsymbol{\theta}) \log p(y \mid \mathrm{x}, \boldsymbol{\theta})] \\
&\approx -\sum_{y\in[c]} \left(\frac{1}{R}\sum_{r} \hat{p}_y^r\right) \log \left(\frac{1}{R}\sum_{r} \hat{p}_y^r\right) + \frac{1}{R}\sum_{y,r} \hat{p}_y^r \log \hat{p}_y^r,
\end{aligned}
$$

where $\hat{p}_y^r = p(y \mid \mathrm{x}, \boldsymbol{\theta}_r)$ is the output of the NN's softmax for class y and randomized parameters $\boldsymbol{\theta}_r \sim p_{drop}(\boldsymbol{\theta})$.

- Only need to compute $\hat{\mathbf{p}}^r$ once per $r \in [R]$, for a modest R times increase in runtime; could be parallelized.

- **BALD** can be computed as:

$$H(Y \mid \mathrm{x}, L) - \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid L)}[H(Y \mid \mathrm{x}, \boldsymbol{\theta})]$$

$$= - \sum_{y \in [c]} p(y \mid \mathrm{x}, L) \log p(y \mid \mathrm{x}, L) + \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid L)}[\sum_{y \in [c]} p(y \mid \mathrm{x}, \boldsymbol{\theta}) \log p(y \mid \mathrm{x}, \boldsymbol{\theta})]$$

$$= - \sum_{y \in [c]} \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid L) d\boldsymbol{\theta} \cdot \log \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid L) d\boldsymbol{\theta} + \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid L)}[\sum_{y \in [c]} p(y \mid \mathrm{x}, \boldsymbol{\theta}) \log p(y \mid \mathrm{x}, \boldsymbol{\theta})]$$

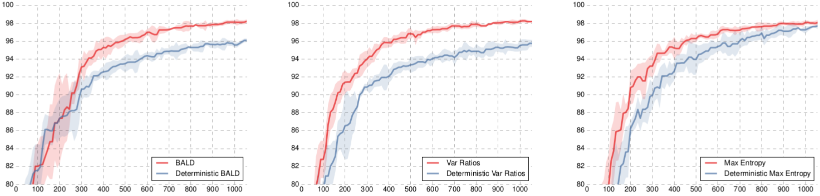
$$\approx - \sum_{y \in [c]} \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p_{drop}(\boldsymbol{\theta}) d\boldsymbol{\theta} \cdot \log \int p(y \mid \mathrm{x}, \boldsymbol{\theta}) p_{drop}(\boldsymbol{\theta}) d\boldsymbol{\theta} + \mathbb{E}_{\boldsymbol{\theta} \sim p_{drop}(\boldsymbol{\theta})}[\sum_{y \in [c]} p(y \mid \mathrm{x}, \boldsymbol{\theta}) \log p(y \mid \mathrm{x}, \boldsymbol{\theta})]$$

$$\approx - \sum_{y \in [c]} \left(\frac{1}{R} \sum_{r} \hat{p}_y^r \right) \log \left(\frac{1}{R} \sum_{r} \hat{p}_y^r \right) + \frac{1}{R} \sum_{y,r} \hat{p}_y^r \log \hat{p}_y^r,$$

where $\hat{p}_y^r = p(y \mid \mathrm{x}, \boldsymbol{\theta}_r)$ is the output of the NN's softmax for class y and randomized parameters $\boldsymbol{\theta}_r \sim p_{drop}(\boldsymbol{\theta})$.

- Only need to compute $\hat{\mathbf{p}}^r$ once per $r \in [R]$, for a modest R times increase in runtime; could be parallelized.

- As $R \to \infty$, this approximates the original mutual information.



(a) BALD (b) Var Ratios (c) Max Entropy

Figure 2. Test accuracy as a function of number of acquired images for various acquisition functions, using both a **Bayesian CNN (red) and a deterministic CNN (blue)**.

- For all choices of acquisition function, the dropout-based uncertainty helps!

- Let us look at **batch-based** active learning.

Batch Selection

Given L, U and a classifier $f \in \mathcal{F}$ trained on L, find a batch $B \subseteq U$ of $b \gg 1$ unlabeled instances that brings maximal information to the model:

$$\underset{B \subseteq U}{\text{argmax}}\ \mathrm{acq}_{BALD}(f, B) \tag{75}$$

$$\text{s.t. } |B| = b \tag{76}$$

Let us look at **batch-based** active learning.

Batch Selection

Given L, U and a classifier $f \in \mathcal{F}$ trained on L, find a batch $B \subseteq U$ of $b \gg 1$ unlabeled instances that brings maximal information to the model:

$$\underset{B \subseteq U}{\text{argmax}} \ \mathrm{acq}_{BALD}(f, B) \tag{75}$$

$$\text{s.t. } |B| = b \tag{76}$$

Advantages:

- Only retrain the model after ever b examples, meaning that supervision has an effect.
- Retraining is less frequent, leading to faster overall execution (at the expense of possibly instance selection, because b examples depend on a fixed f).
- Supports parallel annotation for, e.g., crowd-sourcing scenarios.

Question: can regular acquisition function (like BALD) be extended to this setting?

- Natural generalization of instance-level strategies:

$$\mathrm{acq}(f, B) = \sum_{\mathbf{x} \in B} \mathrm{acq}(f, \mathbf{x}) \tag{77}$$

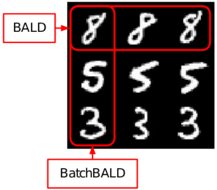
How well does this work?

- Natural generalization of instance-level strategies:

$$\mathrm{acq}(f, B) = \sum_{\mathbf{x} \in B} \mathrm{acq}(f, \mathbf{x}) \tag{77}$$

How well does this work?

- This ignores *correlation* between instances in $\mathbf{x}$:
  - Even if all of them are informative, they may **carry the same information**
  - We want B to be informative **as a whole**!



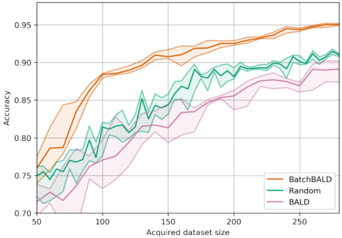


Figure 1: *Idealised acquisitions of BALD and BatchBALD.* If a dataset were to contain many (near) replicas for each data point, then BALD would select all replicas of a single informative data point at the expense of other informative data points, wasting data efficiency.

Figure 2: *Performance on* Repeated MNIST *with acquisition size 10.* See section 4.1 for further details. BatchBALD outperforms BALD while BALD performs worse than random acquisition due to the replications in the dataset.

(Credit: [Kirsch et al., 2019].)

- The problem with the "natural generalization":

$$\mathrm{acq}(f, B) = \sum_{\mathbf{x} \in B} \mathrm{acq}(f, \mathbf{x}) \tag{78}$$

is that the sum doesn't consider the *overlap* between the information carried by different $\mathbf{x} \in b$.

BatchBALD

- The problem with the "natural generalization":

$$\mathrm{acq}(f, B) = \sum_{\mathbf{x} \in B} \mathrm{acq}(f, \mathbf{x}) \tag{78}$$

is that the sum doesn't consider the *overlap* between the information carried by different $\mathbf{x} \in b$.

Idea: don't break the acquisition function into a sum! For BALD, this means replacing:

$$\sum_{\mathbf{x} \in B} \left\{ \underbrace{H(Y \mid \mathbf{x}, L) - \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid L)}[H(Y \mid \mathbf{x}, \boldsymbol{\theta})]}_{MI(Y, \Theta \mid \mathbf{x}, L)} \right\} \tag{79}$$

with

$$MI(\{Y_1, \ldots, Y_b\}, \Theta \mid \{\mathbf{x}_1, \ldots, \mathbf{x}_b\}, L) \tag{80}$$

- The problem with the "natural generalization":

$$\mathrm{acq}(f, B) = \sum_{\mathbf{x} \in B} \mathrm{acq}(f, \mathbf{x}) \tag{78}$$

is that the sum doesn't consider the *overlap* between the information carried by different $\mathbf{x} \in b$.

Idea: don't break the acquisition function into a sum! For BALD, this means replacing:

$$\sum_{\mathbf{x} \in B} \Big\{ \underbrace{H(Y \mid \mathbf{x}, L) - \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid L)}[H(Y \mid \mathbf{x}, \boldsymbol{\theta})]}_{MI(Y, \Theta \mid \mathbf{x}, L)} \Big\} \tag{79}$$

with

$$MI(\{Y_1, \ldots, Y_b\}, \Theta \mid \{\mathbf{x}_1, \ldots, \mathbf{x}_b\}, L) \tag{80}$$

- In other words, **don't assume independence** between the elements of B!

$$\sum_i \mathbb{I}(y_i\,;\,\boldsymbol{\omega} \mid \boldsymbol{x}_i, \mathcal{D}_{\textbf{train}}) = \sum_i \mu^*(y_i \cap \boldsymbol{\omega})$$

$$\mathbb{I}(y_1, ..., y_b\,;\,\boldsymbol{\omega} \mid \boldsymbol{x}_1, ..., \boldsymbol{x}_b, \mathcal{D}_{\textbf{train}}) = \mu^*\left(\bigcup_i y_i \cap \boldsymbol{\omega}\right)$$

(a) BALD **(b) BatchBALD**

Figure 3: *Intuition behind* BALD *and* BatchBALD *using I-diagrams [30]. BALD* overestimates the joint mutual information. *BatchBALD*, however, takes the overlap between variables into account and will strive to acquire a better cover of $\boldsymbol{\omega}$. Areas contributing to the respective score are shown in grey, and areas that are double-counted in dark grey.

(Credit: [Kirsch et al., 2019].)

Problem: batch selection amounts to solving

$$\underset{B \subseteq U : |B| = b}{\operatorname{argmax}} \quad MI(\{Y_1, \ldots, Y_b\}, \Theta \mid \{\mathbf{x}_1, \ldots, \mathbf{x}_b\}, L) \tag{81}$$

How can we solve this?

Problem: batch selection amounts to solving

$$\underset{B \subseteq U : |B| = b}{\text{argmax}} \quad MI(\{Y_1, \ldots, Y_b\}, \Theta \mid \{\mathbf{x}_1, \ldots, \mathbf{x}_b\}, L) \tag{81}$$

How can we solve this?

Submodular Function [Krause and Guestrin, 2008]

Let S be a set. A function f that maps subsets of S to real values is **submodular** if for every $B \subset A \subseteq S$ and any $x \in S \setminus A$ it holds that:

$$f(A \cup \{x\}) - f(A) \leq f(B \cup \{x\}) - f(B) \tag{82}$$

f enjoys a **diminishing returns** property: *adding an element x to a smaller set B "adds more" than adding the same element to a superset $A \supset B$.*

Problem: batch selection amounts to solving

$$\underset{B \subseteq U : |B| = b}{\operatorname{argmax}} \; MI(\{Y_1, \ldots, Y_b\}, \Theta \mid \{\mathbf{x}_1, \ldots, \mathbf{x}_b\}, L) \tag{81}$$

How can we solve this?

Submodular Function [Krause and Guestrin, 2008]

Let S be a set. A function f that maps subsets of S to real values is **submodular** if for every $B \subset A \subseteq S$ and any $x \in S \setminus A$ it holds that:

$$f(A \cup \{x\}) - f(A) \leq f(B \cup \{x\}) - f(B) \tag{82}$$

f enjoys a **diminishing returns** property: *adding an element x to a smaller set B "adds more" than adding the same element to a superset $A \supset B$.*

Maximizing a Submodular Functions

Let $f(A)$ be submodular and S the domain. Then, the greedy algorithm:

- $A_1 \leftarrow \varnothing$.
- $A_{t+1} \leftarrow \operatorname{argmax}_{x \in (S \setminus A_t)} f(A_t \cup \{x\})$.
- Stop when budget T is exhausted.

finds $A_T \subseteq S$ that has score $(1 - \frac{1}{e}) \approx 67\%$ as good as the score of the global optimum $A^* \subseteq S$ of f.

Problem: batch selection amounts to solving

$$\underset{B \subseteq U : |B| = b}{\text{argmax}} \; MI(\{Y_1, \ldots, Y_b\}, \Theta \mid \{\mathbf{x}_1, \ldots, \mathbf{x}_b\}, L) \tag{83}$$

How can we solve this?

Problem: batch selection amounts to solving

$$\underset{B \subseteq U : |B| = b}{\text{argmax}} \; MI(\{Y_1, \dots, Y_b\}, \Theta \mid \{\mathbf{x}_1, \dots, \mathbf{x}_b\}, L) \tag{83}$$

How can we solve this?

- The mutual information is **submodular**! $\rightarrow$ apply greedy optimization:
  - Pick x_1 to optimize $MI(Y_1, \Theta \mid \mathbf{x}_1, L)$ (BALD)
  - Pick x_{t+1} to optimize $MI(Y_{t+1} \cup Y_{1:t}, \Theta \mid \mathbf{x}_{t+1} \cup \mathbf{x}_{1:t}, L)$ (BALD over updated MI)

where $B = \{\mathbf{x}_1, \dots, \mathbf{x}_b\}$.

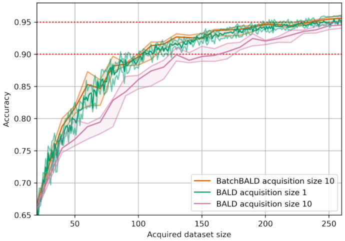


Figure 5: *Performance on* MNIST. BatchBALD outperforms BALD with acquisition size 10 and performs close to the optimum of acquisition size 1.

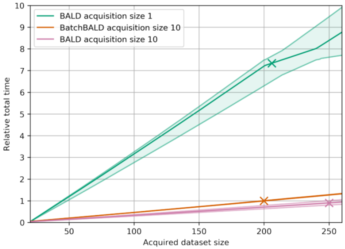


Figure 6: *Relative total time on* MNIST. Normalized to training BatchBALD with acquisition size 10 to 95% accuracy. The stars mark when 95% accuracy is reached for each method.

(Credit: [Kirsch et al., 2019].)

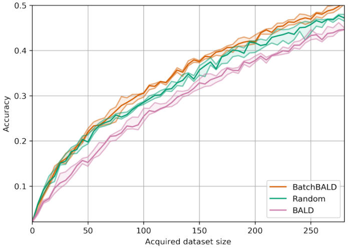


Figure 7: *Performance on* EMNIST. BatchBALD consistently outperforms both random acquisition and BALD while BALD is unable to beat random acquisition.

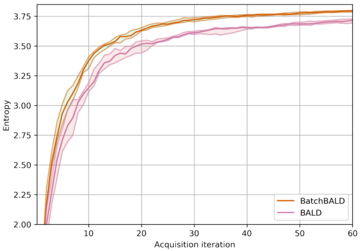


Figure 8: *Entropy of acquired class labels over acquisition steps on* EMNIST. BatchBALD steadily acquires a more diverse set of data points.

(Credit: [Kirsch et al., 2019].)

Idea: we want instances in L to be distributed as ground-truth

- This involves getting $p^*(\mathbf{X})$ right
- It also involves getting all the *modes* in it right

Idea: we want instances in L to be distributed as ground-truth

- This involves getting $p^*(\mathbf{X})$ right
- It also involves getting all the *modes* in it right

■ **Highly non-trivial**: we could train a generative model $\hat{p}_\theta(\mathbf{X}, Y)$ using **density estimation** and use that to guide query selection

Idea: we want instances in L to be distributed as ground-truth

- This involves getting $p^*(\mathbf{X})$ right
- It also involves getting all the *modes* in it right

■ **Highly non-trivial**: we could train a generative model $\hat{p}_\theta(\mathbf{X}, Y)$ using **density estimation** and use that to guide query selection $\rightarrow$ hard to train, break down in high dimension (in general)

Discriminative Active Learning

Idea: we want instances in L to be distributed as ground-truth

- This involves getting $p^*(\mathbf{X})$ right
- It also involves getting all the *modes* in it right

Highly non-trivial: we could train a generative model $\hat{p}_\theta(\mathbf{X}, Y)$ using **density estimation** and use that to guide query selection $\rightarrow$ hard to train, break down in high dimension (in general)

Is there any alternative?

Idea: if we **cannot** find a classifier $f \in \mathcal{F}$ that tells L from U apart, and the latter is large enough (i.e., it can be used to approximate the ground-truth distribution $p^*(\mathbf{X})$), then L is high-quality.

Idea: if we **cannot** find a classifier $f \in \mathcal{F}$ that tells L from U apart, and the latter is large enough (i.e., it can be used to approximate the ground-truth distribution $p^*(\mathbf{X})$), then L is high-quality.

- Given $\mathbf{x} \in U$, how certain are we that it comes from U rather than from L?
  - if indistinguishable, we represented the true distribution using L
  - if distinguishable, it looks different from L so labeling it should be informative

$\mathcal{F}$-divergence

Given two distribution $p_S(X)$ and $p_T(X)$ on $X \in \mathcal{X}$ and a hypothesis class $\mathcal{F}$ also on $\mathcal{X}$, the $\mathcal{F}$-divergence between p_S and p_T is:

$$d_{\mathcal{F}}(p_S, p_T) = 2 \cdot \sup_{f \in \mathcal{F}} |p_S(\{x : f(x) = 1\}) - p_T(\{x : f(x) = 1\})| \tag{84}$$

$\mathcal{F}$-divergence

Given two distribution $p_S(X)$ and $p_T(X)$ on $X \in \mathcal{X}$ and a hypothesis class $\mathcal{F}$ also on $\mathcal{X}$, the $\mathcal{F}$-divergence between p_S and p_T is:

$$d_{\mathcal{F}}(p_S, p_T) = 2 \cdot \sup_{f \in \mathcal{F}} |p_S(\{x : f(x) = 1\}) - p_T(\{x : f(x) = 1\})| \tag{84}$$

- Measures how different two domains p_S and p_T are from the perspective of a model class $\mathcal{F}$: the larger the difference, the more different they look.

- $\mathcal{F}$-divergence used to identify **concept drift** and – symmetrically – to estimate how well a classifier trained on one task will perform on a different, related task

Idea: use $\mathcal{F}$-divergence to discriminate between L and U

Idea: use $\mathcal{F}$-divergence to discriminate between L and U

- The $\mathcal{F}$-divergence is:

$$d_{\mathcal{F}}(p_S, p_T) = 2 \cdot \sup_{f \in \mathcal{F}} |p_S(\{x : f(x) = 1\}) - p_T(\{x : f(x) = 1\})| \tag{85}$$

Idea: use $\mathcal{F}$-divergence to discriminate between L and U

- The $\mathcal{F}$-divergence is:

$$d_{\mathcal{F}}(p_S, p_T) = 2 \cdot \sup_{f \in \mathcal{F}} |p_S(\{x : f(x) = 1\}) - p_T(\{x : f(x) = 1\})| \tag{85}$$

- Approximate as follows:
  - $p_S(\mathbf{x}) := \frac{1}{|L|} \sum_{\mathbf{x}' \in L} \delta\{\mathbf{x}' = \mathbf{x}\}$
  - $p_T(\mathbf{x}) := \frac{1}{|U|} \sum_{\mathbf{x}' inU} \delta\{\mathbf{x}' = \mathbf{x}\}$.

Then $\mathcal{F}$-divergence becomes:

$$2 \sup_{f \in \mathcal{F}} \left| \frac{1}{|L|} \sum_{\mathbf{x} \in L} f(\mathbf{x}) - \frac{1}{|U|} \sum_{\mathbf{x} \in U} f(\mathbf{x}) \right| \tag{86}$$

Idea: use $\mathcal{F}$-divergence to discriminate between L and U

- The $\mathcal{F}$-divergence is:

$$d_{\mathcal{F}}(p_S, p_T) = 2 \cdot \sup_{f \in \mathcal{F}} |p_S(\{x : f(x) = 1\}) - p_T(\{x : f(x) = 1\})| \tag{85}$$

- Approximate as follows:
  - $p_S(\mathbf{x}) := \frac{1}{|L|} \sum_{\mathbf{x}' \in L} \delta\{\mathbf{x}' = \mathbf{x}\}$
  - $p_T(\mathbf{x}) := \frac{1}{|U|} \sum_{\mathbf{x}' inU} \delta\{\mathbf{x}' = \mathbf{x}\}$.

Then $\mathcal{F}$-divergence becomes:

$$2 \sup_{f \in \mathcal{F}} \left| \frac{1}{|L|} \sum_{\mathbf{x} \in L} f(\mathbf{x}) - \frac{1}{|U|} \sum_{\mathbf{x} \in U} f(\mathbf{x}) \right| \tag{86}$$

- How to compute this?

Idea: use $\mathcal{F}$-divergence to discriminate between L and U

- The $\mathcal{F}$-divergence is:

$$d_{\mathcal{F}}(p_S, p_T) = 2 \cdot \sup_{f \in \mathcal{F}} |p_S(\{x : f(x) = 1\}) - p_T(\{x : f(x) = 1\})| \tag{85}$$

- Approximate as follows:
  - $p_S(\mathbf{x}) := \frac{1}{|L|} \sum_{\mathbf{x}' \in L} \delta\{\mathbf{x}' = \mathbf{x}\}$
  - $p_T(\mathbf{x}) := \frac{1}{|U|} \sum_{\mathbf{x}' inU} \delta\{\mathbf{x}' = \mathbf{x}\}$.

Then $\mathcal{F}$-divergence becomes:

$$2 \sup_{f \in \mathcal{F}} \left| \frac{1}{|L|} \sum_{\mathbf{x} \in L} f(\mathbf{x}) - \frac{1}{|U|} \sum_{\mathbf{x} \in U} f(\mathbf{x}) \right| \tag{86}$$

- How to compute this? $\sup_{f \in \mathcal{F}}$ can be implemented by learning f from data set:

$\mathcal{F}$-divergence as learning

- Given L and U, define binary classification task:

$\mathcal{F}$-divergence as learning

- Given L and U, define binary classification task:
  - For all $\mathbf{x} \in L$, add $(\phi(\mathbf{x}), \text{labeled})$ to dataset

$\mathcal{F}$-divergence as learning

- Given L and U, define binary classification task:
  - For all $\mathbf{x} \in L$, add $(\phi(\mathbf{x}), \text{labeled})$ to dataset
  - For all $\mathbf{x} \in U$, add $(\phi(\mathbf{x}), \text{unlabeled})$ to dataset

$\mathcal{F}$-divergence as learning

- Given L and U, define binary classification task:
  - For all $\mathbf{x} \in L$, add $(\phi(\mathbf{x}), \text{labeled})$ to dataset
  - For all $\mathbf{x} \in U$, add $(\phi(\mathbf{x}), \text{unlabeled})$ to dataset
  - Train (simple) classifier $p_\psi(\cdot \mid \phi(\mathbf{x}))$ to distinguish between the two sources by optimizing **cross-entropy loss**

$\mathcal{F}$-divergence as learning

- Given L and U, define binary classification task:
  - For all $\mathbf{x} \in L$, add $(\phi(\mathbf{x}), \text{labeled})$ to dataset
  - For all $\mathbf{x} \in U$, add $(\phi(\mathbf{x}), \text{unlabeled})$ to dataset
  - Train (simple) classifier $p_\psi(\cdot \mid \phi(\mathbf{x}))$ to distinguish between the two sources by optimizing **cross-entropy loss**

This implicitly makes use of the labels y in L through $\phi(\mathbf{x})$.

$\mathcal{F}$-divergence as learning

- Given L and U, define binary classification task:
  - For all $\mathbf{x} \in L$, add $(\phi(\mathbf{x}), \text{labeled})$ to dataset
  - For all $\mathbf{x} \in U$, add $(\phi(\mathbf{x}), \text{unlabeled})$ to dataset
  - Train (simple) classifier $p_\psi(\cdot \mid \phi(\mathbf{x}))$ to distinguish between the two sources by optimizing **cross-entropy loss**

This implicitly makes use of the labels y in L through $\phi(\mathbf{x})$.

- Pick those instances $\mathbf{x} \in U$ that have lowest probability $p_\psi(\text{labeled} \mid \phi(\mathbf{x}))$

$\mathcal{F}$-divergence as learning

- Given L and U, define binary classification task:
  - For all $\mathbf{x} \in L$, add $(\phi(\mathbf{x}), \text{labeled})$ to dataset
  - For all $\mathbf{x} \in U$, add $(\phi(\mathbf{x}), \text{unlabeled})$ to dataset
  - Train (simple) classifier $p_\psi(\cdot \mid \phi(\mathbf{x}))$ to distinguish between the two sources by optimizing **cross-entropy loss**

This implicitly makes use of the labels y in L through $\phi(\mathbf{x})$.

- Pick those instances $\mathbf{x} \in U$ that have lowest probability $p_\psi(\text{labeled} \mid \phi(\mathbf{x}))$

Remark: this is closely related to GANs!

Conclusion and Further Reading

Take-away

- AL useful when supervision is expensive high $\rightarrow$ choose it wisely
- Many variants: pool-based, streaming, and query synthesis
- Many practical approaches: **uncertainty**-based (uncertainty sampling, QBC, expected gradient length), **diversity**-based (information density).

Some can be derived from *version spaces* and *model improvement*.

- Deep variants select entire **batches** and often rely on Bayesian techniques
- Critique & realistic annotators, costs, *etc.*: [Herde et al., 2021] [Settles, 2011]
- Plenty of room for new research ;-)

Baum, E. B. and Lang, K. (1992).
Query learning can work poorly when a human oracle is used.
In *International joint conference on neural networks*, volume 8, page 8.

Gal, Y. and Ghahramani, Z. (2016).
Dropout as a bayesian approximation: Representing model uncertainty in deep learning.
In *international conference on machine learning*, pages 1050–1059. PMLR.

Gal, Y., Islam, R., and Ghahramani, Z. (2017).
Deep bayesian active learning with image data.
In *International Conference on Machine Learning*, pages 1183–1192. PMLR.

Herde, M., Huseljic, D., Sick, B., and Calma, A. (2021).
A survey on cost types, interaction schemes, and annotator performance models in selection algorithms for active learning in classification.
arXiv preprint arXiv:2109.11301.

Hüllermeier, E. and Waegeman, W. (2021).
Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods.
Machine Learning, 110(3):457–506.

King, R. D., Rowland, J., Oliver, S. G., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L. N., et al. (2009).
The automation of science.

Science, 324(5923):85–89.

Kirsch, A., Van Amersfoort, J., and Gal, Y. (2019).
Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning.
Advances in neural information processing systems, 32:7026–7037.

Krause, A. and Guestrin, C. (2008).
Beyond convexity: Submodularity in machine learning.
ICML Tutorials.

Kristiadi, A., Hein, M., and Hennig, P. (2020).
Being bayesian, even just a bit, fixes overconfidence in relu networks.
In *International Conference on Machine Learning*, pages 5436–5446. PMLR.

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., and Clune, J. (2016).
Synthesizing the preferred inputs for neurons in neural networks via deep generator networks.
Advances in neural information processing systems, 29:3387–3395.

Settles, B. (2011).
From theories to queries: Active learning in practice.
In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 1–18. JMLR Workshop and Conference Proceedings.

Settles, B. (2012).
Active learning.

Shalev-Shwartz, S. and Ben-David, S. (2014).
Understanding machine learning: From theory to algorithms.
Cambridge university press.

Sharma, M. and Bilgic, M. (2017).
Evidence-based uncertainty sampling for active learning.
Data Mining and Knowledge Discovery, 31(1):164–202.