02-750 Week 6

Automation of Scientific Research

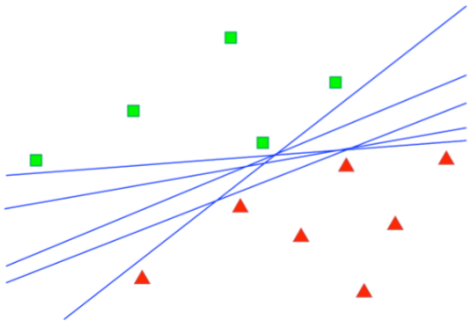
Aidan Jan

February 19, 2026

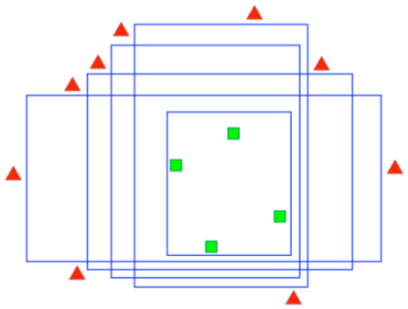
Version Space

Given labeled data $D_L = \{(x_1, y_1), \cdots, (x_n, y_n)\}$

- **Definition:** The **version space** is the set of hypotheses that are consistent with the current labeled training data D_L



Version space for linear functions on D_L



Version space for axis-parallel box classifiers on D'_L

Background

Probably Approximately Correct (PAC) Learning

- In general, it is **impossible** to **guarantee** that any particular learning algorithm will **always** find the best model in the hypothesis space.
  - i.e., the one with the lowest **generalization error** (not the lowest training error)
- Informally, PAC learning considers scenarios where it is possible to provide a probabilistic guarantee that the learning algorithm will find a "good" model (i.e., close to optimal).
- Concept class $\mathcal{C}$ is **PAC-learnable** by learning algorithm $A : L, \mathcal{H} \to h$ if, $\forall c \in \mathcal{C}$ and $0 < \epsilon, \delta < 1/2$ and finite $m = |L|$

  $$P(|R(h) - \hat{R}(h)| \leq \epsilon) \geq 1 - \delta$$

  or equivalently, $P(|R(h) - \hat{R}(h)| > \epsilon) < \delta$
- Concept class $\mathcal{C}$ is **efficiently PAC-learnable** if the above inequality holds and m is polynomial in $1/\epsilon$ and $1/\delta$, and the algorithm returns hypothesis h in time and space polynomial in $1/\epsilon$ and $1/\delta$
- The 'probably' in PAC refers to the $(1 - \delta)$; the 'approximately' refers to the ϵ.

PAC learning results let us do two useful things

1. Derive formulas to bound the sample complexity (i.e., select ϵ and δ and determine m - the minimum number of samples needed to guarantee that $P(|R(h) - \hat{R}(h)| \leq \epsilon) \geq 1 - \delta)$
2. Derive formulas to bound the **generalization error** with respect to the **training error**. (i.e., select m and δ and determine ϵ (the error bound))

Note the precise form of the formulas will depend on any assumptions we make about $\mathcal{C} and \mathcal{H}$

The CAL Algorithm

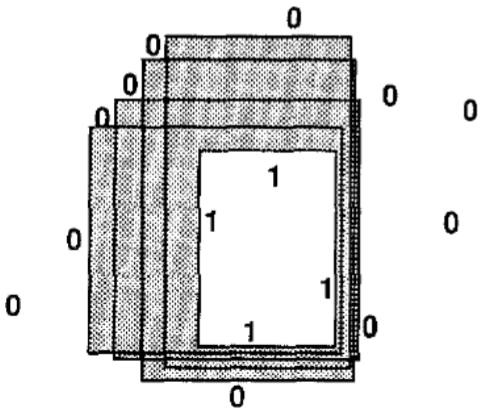
Key features:

- It is a Type 1 algorithm (i.e., hypothesis elimination, picks points near decision boundary)
- CAL is an active learning algorithm for binary classifiers
- It can be used with any data access model
- Guarantees
  - Under the assumptions of separable data and suitable choice of hypothesis class, CAL will converge to a perfect model
  - The label complexity is $\Omega\left(\log \frac{1}{\epsilon}\right)$
- The algorithms we will study next time are 'fancy' versions of CAL that come with better guarantees for label complexity.

Notation and Definitions:

- Let $\mathcal{X}$ be an arbitrary domain
- Let $\mathcal{Y} = \{0, 1\}$ be the set of binary labels
- Let $\mathcal{D}$ be a probability distribution on $\mathcal{X}$
- Let $\mathcal{H}$ be a hypothesis* class, $h \in \mathcal{H}, h : \mathcal{X} \to \mathcal{Y}$
- Let $\mathcal{S}$ be a set of labeled points in $\mathcal{X}$
- Hypothesis $h \in \mathcal{H}$ is **consistent** with S if it correctly labels **every** point in S.
- The **version space** is the set of hypotehses consistent with S.

In the example below, $\mathcal{H}$ is the space of axis-aligned rectangles. The illustrated rectangles are *some* of the hypotheses in the version space



$$\mathcal{R}(S) = \{x \,:\, \exists h_1, h_2 \in \mathcal{H},\ \text{consistent with } S \text{ and } h_1(x) \neq h_2(x)\}$$

Notice that if we get the label for *any* point in the disagreement region, we are **guaranteed** to be able to eliminate at least one hypothesis (and possibly many more)

```
Let H_0 = H be the initial set of hypothesis
for i = 1, 2, ...
    Get unlabeled point x_i in X (MQS, Pool-based, or Stream-based)
    if (x_i in disagreement region):
        y_i = oracle(x_i)
        update version space: H_i = {h in H_{i-1}: h(x_i) = y_i}
    else
        H_i = H_{i - 1}
```

Note that the version space shrinks over time (since updating version space is $\mathcal{H}_i = \mathcal{H}_{i-1}$)

- Unfortunately, we can't implement the conceptual version of CAL (unless $\mathcal{H}$ is finite.)

Problems with Conceptual CAL

1. How do we represent and update the version space? The version space will, in general, contain an infinite number of models.
2. How do we detect whether x_i is in the disagreement region? That is, how do we determine whether there is *any* pair of models in the version space disagree on x_i's label?

The 'real' version of CAL answers these questions using a clever idea: **Maintain the version space using two models.**

- Use S_i to construct two **consistent** models h_s and h_G, that are the "most specific" (h_s) and "most general" (h_g) models in the version space, $V(S_i)$
  - h_s is a model that assigns label 1 to as few points in $x \in \mathcal{X}$ as possible (i.e., $h_s(x) = 1$), subject to being consistent with the points in S_i
  - h_G is a model that assigns label 1 to as many points in $x \in \mathcal{X}$ as possible (i.e., $h_G(x) = 1$), subject to being consistent with the points in S_i
- Notice that **if** we can construct and maintain h_S and h_G, then it is easy to detect whether any given $x_i \in \mathcal{X}$ is in the disagreement
- We simply test whether $h_S(x_i) \neq h_G(x_i)$.

Digression on Inductive Bias

- An inductive bias is a predisposition of a learning algorithm for some solutions over others.
  - Most learning algorithms inherently have at least some form of an inductive bias.
    - Preference for simple solutions over complex ones (Occam's razor)
    - A tendency to choose solutions where the absolute values of the parameters remain small
- A most specific network, h_S, for a set of examples S is one that classifies as positive those example points that are in fact positive and classifies as negative as much as possible of the rest of the domain
  - How? Explicitly add a new inductive bias to the backpropagation algorithm by penalizing the network for any part of the domain that it classifies as positive.
  - That is, they add a bias that prefers specific concepts (hypotheses) over general ones.

The Implicit Version of the CAL Algorithm

The 'Two Faces' paper points out that there is an implicit version of the CAL algorithm that doesn't require neural nets or inductive biases

- Suppose that when we are given $x \in \mathcal{X}$, we learn two **consistent** models
  - $h_s = \text{Learn}(S \cup \{(x, 0)\})$
  - $h_G = \text{Learn}(S \cup \{(x, 1)\})$
- Key observations
  - If h_G is consistent with $S \cup \{(x, 1)\}$, and h_S is consistent with $S \cup \{(x, 0)\}$, then x must be in the disagreement region.
  - If h_G is not consistent with $S \cup \{(x, 1)\}$, then x's true label must be 0.
  - If h_S is not consistent with $S \cup \{(x, 0)\}$, then x's true label must be 1.

Note that this works because CAL assumes all data is separable, and therefore true labels can be *inferred* if they are not consistent with either h_G or h_S. If the point is consistent with both, then it means the point is in the disagreement region, and therefore we ask the oracle for the label.

Pseudocode of the Implicit Version

```
Let S = {}
For i = 1, 2, ...:
    Get unlabeled point x_i in X.  (MQS, Pool-based, or Stream-based)
    Let h_S = Learn(S + {(x, 0)})
    Let h_G = Learn(S + {(x, 1)})
    if h_S and h_G are both consistent with their respective training sets:
        y_i = oracle(x_i)
    else:
        if h_S is not consistent with (S + {(x, 0)}):
            y_i = 1
        elif h_G is not consistent with (S + {(x, 1)}):
            y_i = 0
    S = S + {(x_i, y_i)}
h = Learn(S)
```

CAL Algorithm Guarantee

- The volume of h_S is monotonically increasing
- The volume of h_G is monotonically decreasing
- Thus, the algorithm is guaranteed to terminate when $h_S = h_G$ or, equivalently, when the volume of the disagreement region shrinks to zero
  - When that happens, you know that you have found the optimal model!
- Unfortunately, we can't prove an upper bound on how many labeled samples it will take to converge upon the optimal model (yet)

Aside: CAL vs Uncertainty Sampling vs QBC

- Notice that the disagreement region is defined with respect to a set of hypotheses
  - It is **not** defined with respect to any single hypothesis/model
  - Do not confuse the disagreement region with uncertainty sampling
- Also, recall that Query by committee (QBC) selects points based on disagreement, but
  - QBC quantifies disagreement (i.e., QBC is 'agressive')
  - CAL doesn't quantify disagreement (i.e., CAL is 'mellow')
  - The size of CAL's 'committee' (i.e., the version space), is generally infinite

CAL Summary

- CAL is an active learning algorithm for binary classifiers
- It is a Type 1 algorithm (i.e., hypothesis elimination)
  - The *version space* is maintained by learning two models, h_S and h_G
  - CAL queries the oracle if x_i is in the region of disagreement, otherwise it can correctly infer the true label
- It is a mellow method (i.e., it doesn't look for maximally informative points)
- CAL can be used with any data access model
- Guarantee
  - Under the assumptions of separable data and suitable choice of hypothesis space/class, CAL will converge to a perfect model
  - It can be shown that the label complexity is $\Omega(-\log \epsilon)$

Handling Non-Separable Data

- The CAL algorithm was noteworthy because it has some clever ideas and comes with a consistency guarantee
  - It will find the optimal model
  - But it can only handle separable data
- Can **any** active learning algorithm guarantee consistency on non-separable data?
  - Interestingly, the answer to that question was unknown until 2006 when the **Agnostic Active** (A^2) **Learning Algorithm** was published
  - In this context, "agnostic" means that the algorithm does not assume that $\mathcal{H}$ contains a perfect classifier, nor does it make any assumptions about the nature of any noise in the data.

The A^2 Algorithm

Key features

- A^2 is an active learning algorithm for **binary classifiers**
- It is a Type I algorithm (i.e., hypothesis elimination)
- It can be used with any data access model

- **It handles non-separable data**
- Guarantees
  - A^2 will converge to the optimal model
  - A^2 bounds the number of samples needed to reach the optimal model

Intuition

- Consider an unknown binary function of the form $f : [0, 1] \rightarrow \{-1, +1\}$. f maps a point on the unit interval to $+1$ or -1, but that is all we know about it.
- Let's approximate f by learning a **threshold function** of the form:

$$h_\theta(x) = \begin{cases} -1 & x \leq \theta \\ +1 & x > \theta \end{cases}$$

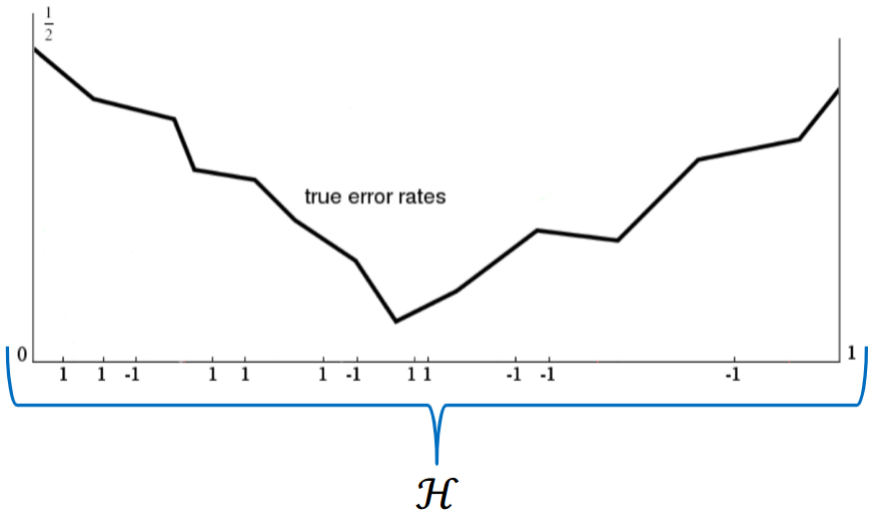
  where θ is the **threshold parameter**
- Our goal is to find the value of θ that minimizes the **true error**
  - The true error of model h_θ is:

$$err(h_\theta) = P_{x \sim D}(h_\theta(x) \neq f(x))$$

    where D is some distribution over $[0, 1]$

Our initial hypothesis space is $\mathcal{H} = [0, 1]$.

- The black curve illustrates the (unknown) true error rates for every possible choice of threshold, θ.
- We cannot compute the exact value of the true error using a finite set of labeled examples, for any choice of θ!
- The best we can do is to estimate the error and use that estimate to select the "best" choice of threshold, θ.



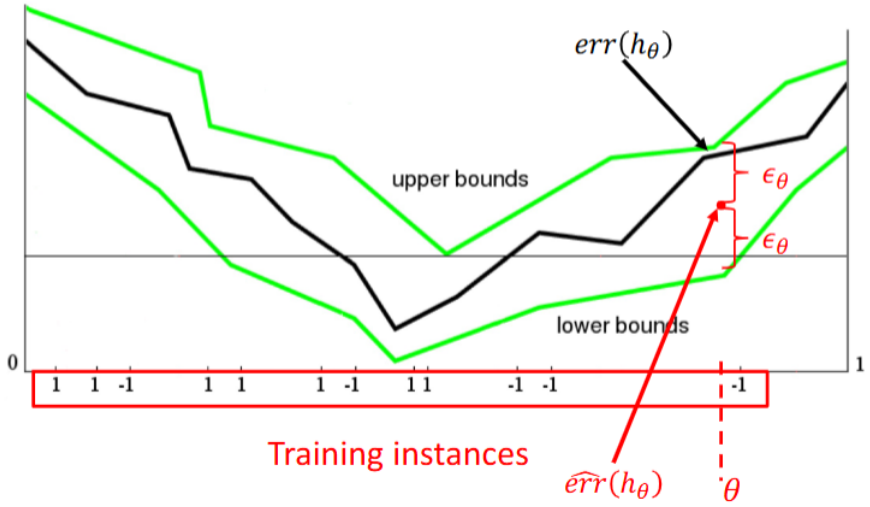
Suppose that we have an algorithm that, given a set of m training instances and a choice of threshold, θ, returns the **empirical error**

$$\hat{err}(h_\theta) = \frac{1}{m}\sum_{i=0}^{m}\mathbb{I}(h_\theta(x_i) \neq f(x_i))$$

and a bound on the difference between the **true error** and the **empirical error**

$$|err(h_\theta) - \hat{err}(h_\theta)| \leq \epsilon_\theta$$

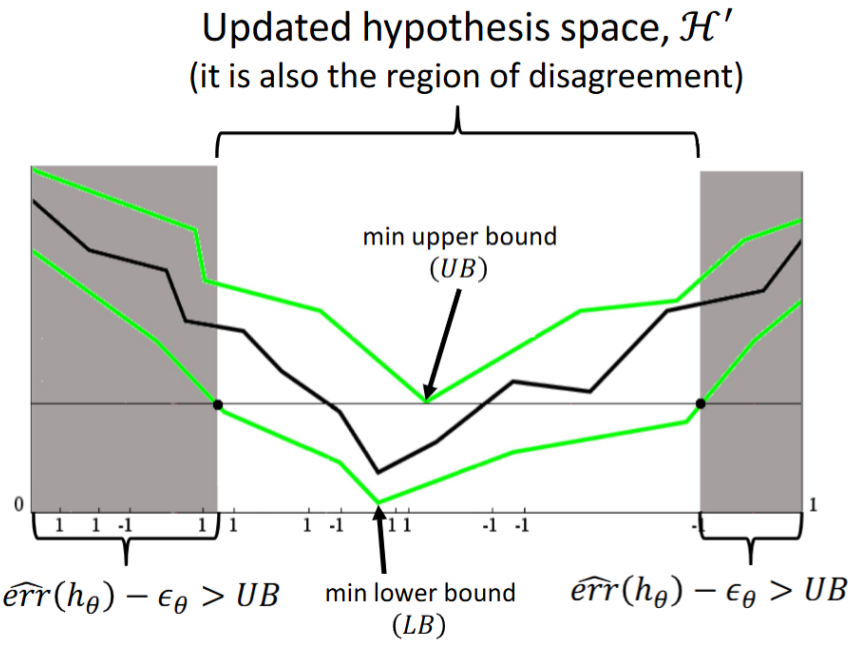
We still cannot compute the true error, for any given choice of θ, but we do know that it must between the green lines.



Intuitively, the bound on the distance between the **true error** and the **empirical error** should decrease as we obtain more samples. Thus, if we obtain more samples and re-estimate the empirical errors of the models in **version space**, we will also obtain tighter bounds, which can be used to eliminate even more hypotheses, in a recursive fashion

- Let $\theta^{UB} = \arg\min_\theta \hat{err}(h_\theta) + \epsilon_\theta$
- Let $\theta^{LB} = \arg\min_\theta \hat{err}(h_\theta) - \epsilon_\theta$
- Let $UB = \hat{err}(h_{\theta^{UB}}) + \epsilon_{\theta^{UB}}$ be the least upper bound in the current hypothesis space
- Let $LB = \hat{err}(h_{\theta^{LB}}) + \epsilon_{\theta^{LB}}$ be the least lower bound in the current hypothesis space

Notice that if $UB - LB$ ever becomes really small, we can stop and return θ^{UB}, because we know that θ^{UB} must be close to optimal, and it has the best worst-case behavior. Similarly, if the disagreement region ever becomes really small, we can stop and return θ^{UB}, because we know the probability of a random point being in the disagreement region is very small, and θ^{UB} has the best worst-case behavior.



Aside: Computing Upper and Lower Bounds

- Unfortunately, we cannot compute the **exact** upper and lower error bounds for an arbitrary model, but we can do something almost as good
- We can compute **generalization bounds**
  - It is possible to guarantee that:

$$|err(h_\theta) - \hat{err}(h_\theta)| \leq \epsilon_\theta, \quad \text{with probability at least } (1-\delta)$$

    if h is trained on at least m independent and identically distributed samples
  - m is a function of the hypothesis class and the values of ϵ and δ (but we get to select the values of ϵ and δ!)

A^2 Algorithm: Simplified Version

- Let $LB(m, h, \delta)$ be the lower bound on the true error of model h trained on m samples and with user-specified $0 < \delta < 0.5$
- Let $UB(m, h, \delta)$ be the upper bound on the true error of model h trained on m samples and with user-specified $0 < \delta < 0.5$
- Let $Disagree(\mathcal{H}_i, D) = P_{x\sim D}(\exists h, h' \in \mathcal{H} : h(x) \neq h'(x))$ be the volume of the **disagreement region**, measured as the probability that there are at least two models that would give different labels to an instance x drawn from sampling distribution D.
- Let $Bound(\mathcal{H}_i, D) = Disagree(\mathcal{H}_i, D) \cdot (\min_{h\in\mathcal{H}_i} UB(m, h, \delta) - \min_{h\in\mathcal{H}_i} LB(m, h, \delta))$ be a bound on the error rate of the current version, $\mathcal{H}_i$

A^2 Guarantees

- Convergence: It will return an ϵ-optimal model. That is, if the optimal model in $\mathcal{H}$ achieves error η, then A^2 will return a model h with true error $err(h)$ such that $\eta \leq err(h_\theta) \leq \eta + \epsilon$ with probability at least $1 - \delta$

- Sample Complexity:
  - If the amount of noise is "small", then it is possible to learn an ϵ-optimal model using only $O(\log \frac{1}{\epsilon})$ samples
  - If the amount of noise is "large" the algorithm's label complexity is never much worse than standard supervised learning (i.e., passive learning)

Limitations of A^2 Algorithm

- If $\mathcal{H}$ is infinite, then you can't compute any of the following, exactly:
  - $Disagree(\mathcal{H}, D)$
  - $Bound(\mathcal{H}, D)$
  - $\mathcal{H}' = \{h \in \mathcal{H} \ : \ LB(m, h, \delta) \leq \min_{h' \in \mathcal{H}} UB(m, h', \delta)\}$
- In practice, you need to resort to sampling models from $\mathcal{H}$ to approximate these things, which can be expensive

The DHM Algorithm

- Key Features
  - DHM is an active learning algorithm for binary classifiers
  - It is a Type I algorithm (i.e., hypothesis elimination)
  - It can be used with any data access model
  - It handles non-separable data
- Guarantees
  - DHM will converge to the optimal model
  - DHM bounds the number of samples needed to reach the optimal model

Architecture

- DHM maintains two sets, S and T
  - T contains points labeled by the oracle
  - S contains points with inferred labels
    - Note: Unlike CAL, the points in S **may be mislabeled**
- DHM uses two techniques for inferring labels
  - The first technique is almost identical to CAL
  - The second technique estimates the difference in the **generalization bounds** between the two models (a bit like A^2)
    - If this difference is "big enough", we can infer the label.