THEORY CLUB

Introduction to Machine Learning Theory

Diptodip Deb

March 6, 2017

Theory Club, Georgia Tech

Overview

What is Machine Learning?

General Definitions

- Figures out a program from data

General Definitions

- Figures out a program from data
- Can do this automatically

General Definitions

- Figures out a program from data
- Can do this automatically
- Can do this even if it is too complex for a human to do so

Three Categories of ML

Supervised Learning

What we call function approximation, $y = f(x)$.

Unsupervised Learning

What we call data description, or clustering data.

Reinforcement Learning

Learning optimal (ish) policies – often with just a reward function and no model!

What This Talk Is/Isn't

We're going to focus on supervised learning for the majority of this talk. At the end (if time permits), we will discuss the challenge of high dimensionality and some basic unsupervised learning techniques (brought to you by linear algebra!).

We are *not* going to be discussing specific algorithms for machine learning (except [maybe] the two at the end) and some toy benchmark algorithms we use for analysis. We are also not going to be talking about deep learning (neural networks) too much (because there's not much we can prove about them).

Formalizing Machine Learning

Definitions

We say a machine **learns** on task T if its performance P improves with experience E.

We restrict ourselves to binary classification tasks unless otherwise noted. Note also that we typically talk about our data existing in some instance space X.

Definition

We define the true concept $c(x) : X \to \{0, 1\}$ as the absolutely true classification for some observation/data $x \in X$.

Formalizing Machine Learning

Definition

We define a **hypothesis** $h(x) : X \to \{0, 1\}$ as the learner's classification for some observation/data $x \in X$.

Definition

We define the **hypothesis space** H as the set of all possible functions that the learner could output. Note that the true concept need not necessarily be in this hypothesis space (though ideally it will be).

Definition

We define a consistent hypothesis as one that classifies all of the given training data $D \subseteq X$ correctly.

Definition

We define the version space $VS_{H,D}$ as the space of hypothesis in some given H that are consistent with some given set of data D.

Bayesian Learning

Why Bayesian?

A Bayesian outlook lets us quantitatively weigh evidence for hypotheses to choose the best one. It also lets us design algorithms that manipulate probability distributions (which are useful for benchmarks) as well as analyze algorithms that do not explicitly manipulate probability distributions (i.e. many popular algorithms).

Characteristics of Bayesian Learning

- Each new observation can change the estimated likelihood of some hypothesis being correct.

Characteristics of Bayesian Learning

- Each new observation can change the estimated likelihood of some hypothesis being correct.
- On that note, Bayesian Learning allows for probabilities of correctness, rather than hard elimination of inconsistent learners.

Characteristics of Bayesian Learning

- Each new observation can change the estimated likelihood of some hypothesis being correct.
- On that note, Bayesian Learning allows for probabilities of correctness, rather than hard elimination of inconsistent learners.
- Because we take a Bayesian (as opposed to frequentist) viewpoint, we can incorporate prior information (domain knowledge) into our learning.

Characteristics of Bayesian Learning

- Each new observation can change the estimated likelihood of some hypothesis being correct.
- On that note, Bayesian Learning allows for probabilities of correctness, rather than hard elimination of inconsistent learners.
- Because we take a Bayesian (as opposed to frequentist) viewpoint, we can incorporate prior information (domain knowledge) into our learning.
- Toy Bayesian learners can often provide useful benchmarks to compare results of (often more practical) algorithms.

The Bayesian Learning Problem

Problem Statement

Given some observations $D \subseteq X$ over some instance space X and some hypothesis space H, pick the hypothesis $h \in H$ which maximizes $\mathbb{P}(h|D)$, i.e. find the optimal hypothesis $h^* = \text{argmax}_h \mathbb{P}(h|D)$.

The Bayesian Learning Problem

Problem Statement

Given some observations $D \subseteq X$ over some instance space X and some hypothesis space H, pick the hypothesis $h \in H$ which maximizes $\mathbb{P}(h|D)$, i.e. find the optimal hypothesis $h^* = \text{argmax}_h \mathbb{P}(h|D)$.

What do we know from Bayes' Rule?

Bayesian Learning

MAP

MAP (maximum a posteriori) Hypothesis

Definition

We define

$$h_{MAP} = \text{argmax}_h \mathbb{P}(h|D) = \text{argmax}_h \frac{\mathbb{P}(D|h)\mathbb{P}(h)}{\mathbb{P}(D)}.$$

This lends itself to an obvious brute-force MAP algorithm.

Small Results of MAP

Assumptions

- The data is not noisy (i.e. $\forall i : c(x_i) = d_i$).
- The true concept is contained in the hypothesis space, $c \in H$.
- We have no a priori bias towards any hypothesis (all $h \in H$ are equally likely without looking at data).

Small Results of MAP

If we have no a priori bias, what is $\mathbb{P}(h)$ for any $h \in H$?

Small Results of MAP

$$\mathbb{P}(h) = \frac{1}{|H|}$$

Small Results of MAP

If we have no noise in our data, what shall we make $\mathbb{P}(D|h)$?

Small Results of MAP

$$\mathbb{P}(D|h) = \begin{cases} 1 & \text{if } h(x_i) = d_i \text{ for all } d_i \text{ in } D \\ 0 & \text{otherwise} \end{cases}$$

Small Results of MAP

Considering the previous results, what can we determine about $\mathbb{P}(h|D)$?

Small Results of MAP

Clearly, if h is inconsistent, we have that $\mathbb{P}(h|D) = 0$.

What if we have a consistent h?

Small Results of MAP

$$\mathbb{P}(h|D) = \frac{\mathbb{P}(D|h)\mathbb{P}(h)}{\mathbb{P}(D)} \tag{1}$$

$$= \frac{\frac{1}{|H|}(1)}{\frac{|VS_{H,D}|}{|H|}} \tag{2}$$

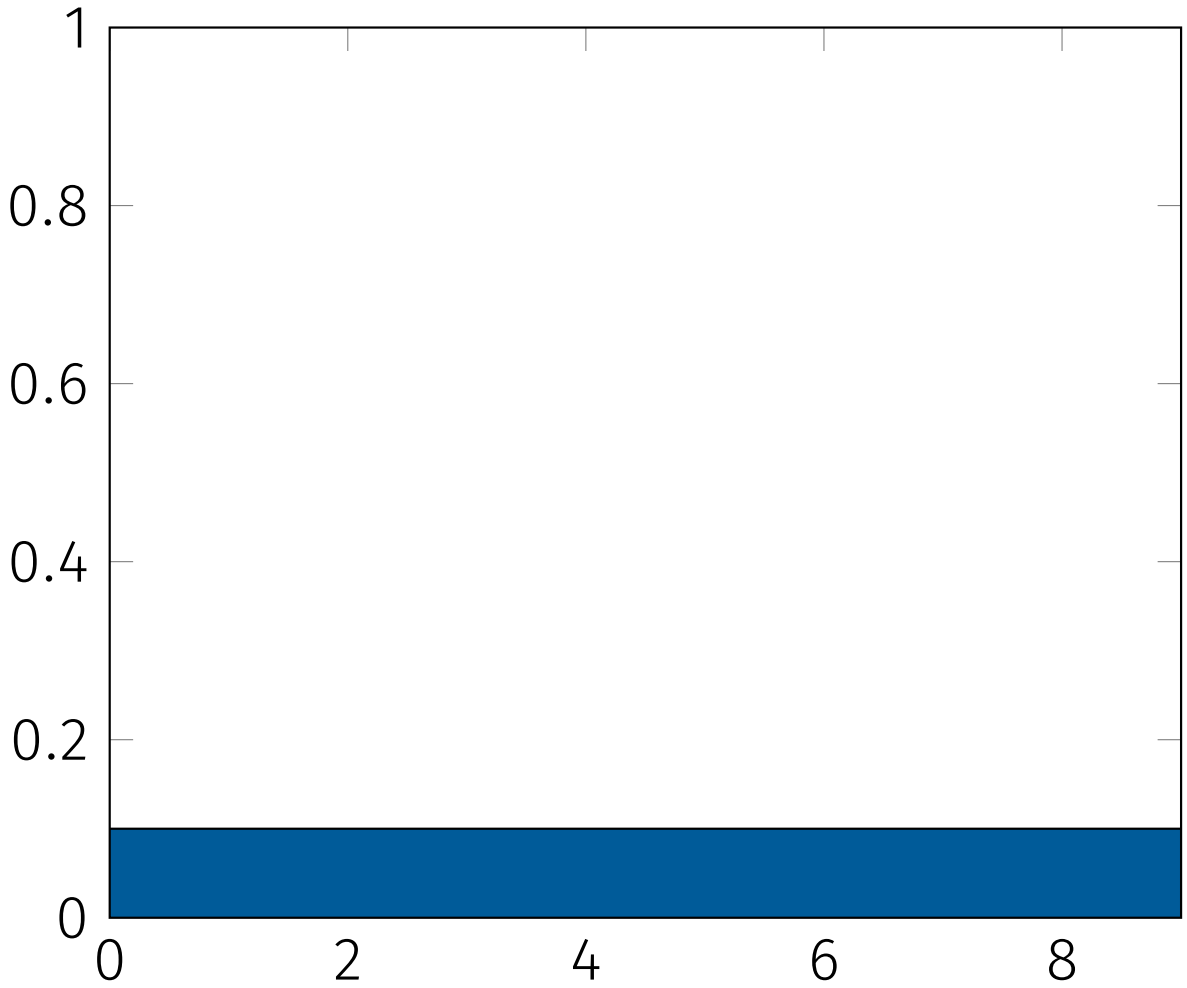
$$= \frac{1}{|VS_{H,D}|} \tag{3}$$

Small Results of MAP

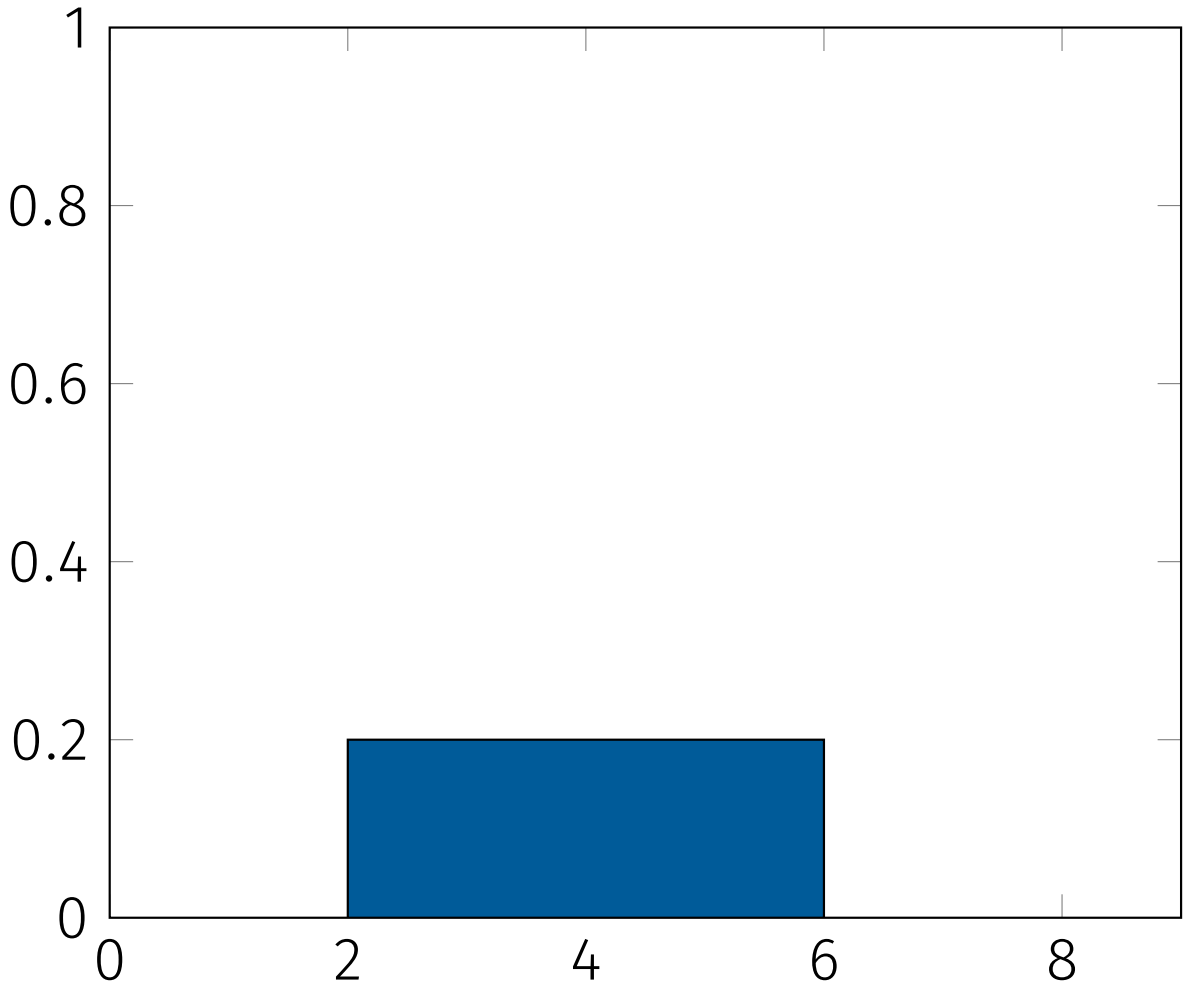
In summary, we found that

$$\mathbb{P}(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h(x_i) = d_i \text{ for all } d_i \text{ in } D \\ 0 & \text{otherwise} \end{cases}$$

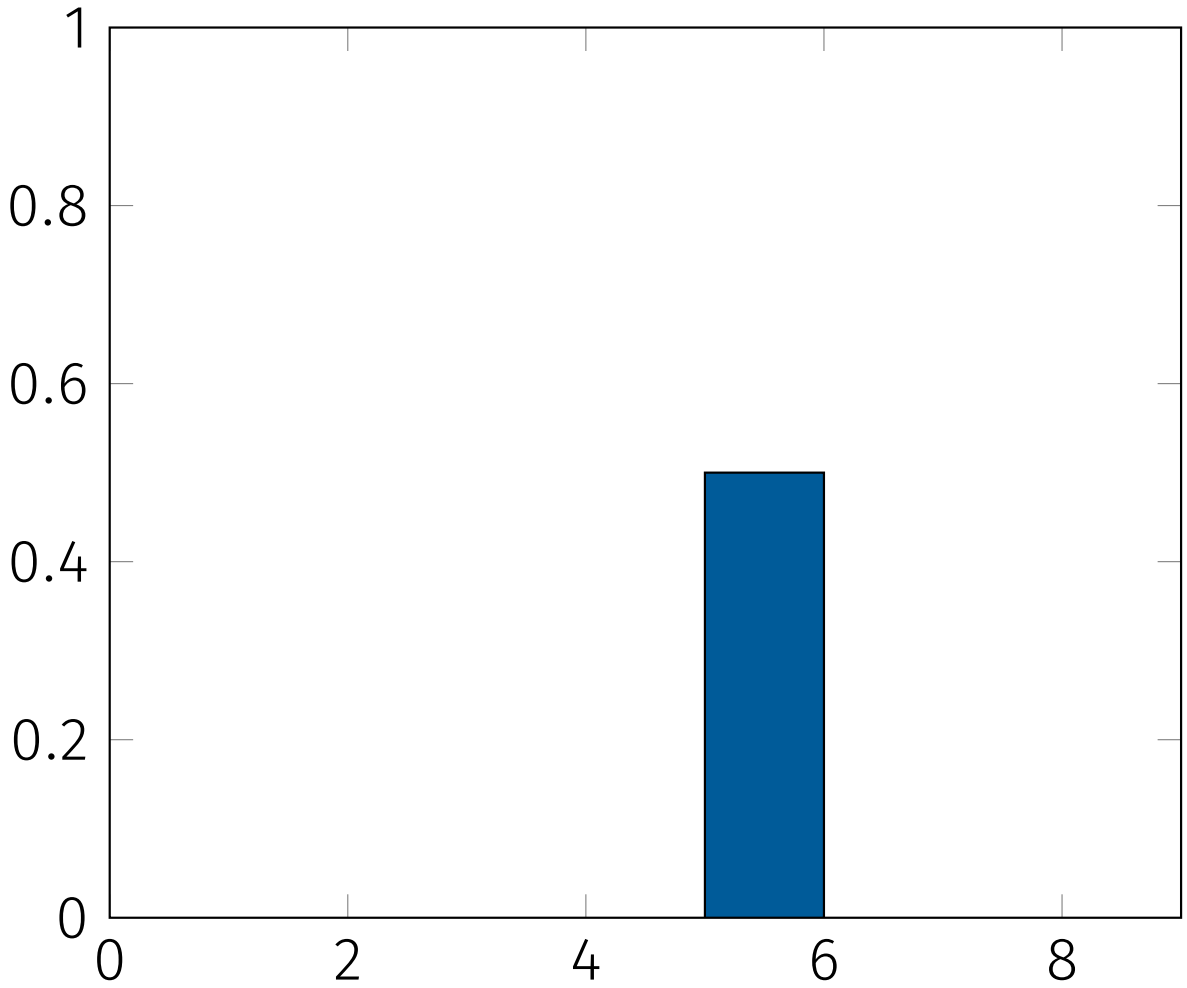
MAP Evolution



MAP Evolution



MAP Evolution



Bayesian Learning

ML

ML (maximum likelihood) Hypothesis

In the previous analysis, we operated under the assumption that our prior for hypotheses was $\mathbb{P}(h) = \frac{1}{|H|}$, i.e. uniform.

ML (maximum likelihood) Hypothesis

In the previous analysis, we operated under the assumption that our prior for hypotheses was $\mathbb{P}(h) = \frac{1}{|H|}$, i.e. uniform.

This is actually a special case of MAP which we call ML, for maximum likelihood.

ML (maximum likelihood) Hypothesis

We're going to use ML hypotheses to justify mean squared error (under certain conditions).

We're also going to be moving from Boolean functions (classification) to real-valued functions.

ML (maximum likelihood) Hypothesis

We now have some instance space X but with a hypothesis space H where for any $h \in H$ we have $h : X \to \mathbb{R}$. We also assume that $c(x) \in H$. Given that, we model each data observation as the following $< x_i, d_i >$. Each pair represents the observation (input) x_i and the given target value d_i.

However, d_i is no longer noise free! We know have that $d_i = c(x_i) + e_i$ where e_i is some random error drawn from a Normal (Gaussian) distribution with 0 mean ($\mu = 0$).

Justifying Mean-Squared Error

Assumptions

- Target concept $c \in H$.
- Target concept is deterministic and observations are only perturbed by some noise.
- Noise is drawn independently and from a Normal distribution with 0 mean.
- We again have no a priori bias towards any hypotheses.
- The number of examples in our training data is given by m.

Justifying Mean-Squared Error

We use the p.d.f. (probability density function) to represent $\mathbb{P}(D|h)$ from now on. This is simply $p(x') = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \mathbb{P}(x' \leq x \leq x' + \epsilon)$.

Justifying Mean-Squared Error

We use the p.d.f. (probability density function) to represent $\mathbb{P}(D|h)$ from now on. This is simply $p(x') = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \mathbb{P}(x' \leq x \leq x' + \epsilon)$.

Now we have

$$h_{ML} = \text{argmax}_h p(D|h).$$

Justifying Mean-Squared Error

Via independence of the errors, we have

$$h_{ML} = \text{argmax}_h \prod_{i=1}^{m} p(d_i|h).$$

Justifying Mean-Squared Error

$$h_{ML} = \text{argmax}_h \prod_{i=1}^{m} p(d_i|h) \tag{1}$$

Justifying Mean-Squared Error

$$h_{ML} = \text{argmax}_h \prod_{i=1}^{m} p(d_i|h) \tag{1}$$

$$= \text{argmax}_h \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i-\mu)^2} \tag{2}$$

Justifying Mean-Squared Error

$$h_{ML} = \text{argmax}_h \prod_{i=1}^{m} p(d_i|h) \tag{1}$$

$$= \text{argmax}_h \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i-\mu)^2} \tag{2}$$

$$= \text{argmax}_h \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i-h(x_i))^2} \tag{3}$$

Justifying Mean-Squared Error

$$h_{ML} = \text{argmax}_h \prod_{i=1}^{m} p(d_i|h) \tag{1}$$

$$= \text{argmax}_h \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i-\mu)^2} \tag{2}$$

$$= \text{argmax}_h \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i-h(x_i))^2} \tag{3}$$

$$= \text{argmax}_h \sum_{i=1}^{m} \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(d_i - h(x_i))^2 \tag{4}$$

Justifying Mean-Squared Error

$$h_{ML} = \text{argmax}_h \sum_{i=1}^{m} -\frac{1}{2\sigma^2}(d_i - h(x_i))^2 \tag{5}$$

$$= \text{argmin}_h \sum_{i=1}^{m} \frac{1}{2\sigma^2}(d_i - h(x_i))^2 \tag{6}$$

$$= \text{argmin}_h \sum_{i=1}^{m} (d_i - h(x_i))^2 \tag{7}$$

Error Function for Probabilistic Functions

What if instead of a real-valued function, we returned to the setting of Boolean functions – but made them nondeterministic?

We still have $c(x) : X \rightarrow \{0, 1\}$, but now each x_i may now map to multiple values of $\{0, 1\}$. This unpredictability is often because we do not observe all the necessary features, so some instances look identical (even though they should not).

Error Function for Probabilistic Functions

Usually, in these cases, we want to use a neural network to learn the probability that some x_i will map to 1.

Let us define $c'(x_i) : X \to [0, 1]$ which returns the probability that $c(x_i) = 1$.

Error Function for Probabilistic Functions

This time, we won't assume that anything is fixed. That is, we assume that both the features that we observe (x_i) and the classifications we receive (d_i) are random variables.

We still assume independence between samples. What does this give us?

Error Function for Probabilistic Functions

$$\mathbb{P}(D|h) = \prod_{i=1}^{m} \mathbb{P}(x_i, d_i|h) \tag{1}$$

Error Function for Probabilistic Functions

$$\mathbb{P}(D|h) = \prod_{i=1}^{m} \mathbb{P}(x_i, d_i|h) \tag{1}$$

$$= \prod_{i=1}^{m} \mathbb{P}(d_i|h, x_i)\mathbb{P}(x_i) \tag{2}$$

Error Function for Probabilistic Functions

What is the probability of obtaining a $d_i = 1$ for some single event x_i, given that h_i is true? That is, what is $\mathbb{P}(d_i|h, x_i)$?

Error Function for Probabilistic Functions

$$\mathbb{P}(d_i|h, x_i) = \begin{cases} h(x_i) & \text{if } d_i = 1 \\ (1 - h(x_i)) & \text{otherwise} \end{cases}$$

Error Function for Probabilistic Functions

Let’s rewrite that into something we can actually work with.

Error Function for Probabilistic Functions

Let's rewrite that into something we can actually work with.

$$\mathbb{P}(d_i|h, x_i) = h(x_i)^{d_i}(1 - h(x_i))^{1-d_i}$$

Error Function for Probabilistic Functions

If we substitute this back into our previous expression for $\mathbb{P}(D|h)$, we have

$$\mathbb{P}(D|h) = \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i}\mathbb{P}(x_i).$$

Error Function for Probabilistic Functions

But we want to maximize this, so we stick an argmax in the front:

$$h_{ML} = \text{argmax}_h \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i}\mathbb{P}(x_i).$$

Error Function for Probabilistic Functions

Removing constants (in terms of h):

$$h_{ML} = \text{argmax}_h \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i}.$$

Error Function for Probabilistic Functions

We take the log again:

$$h_{ML} = \text{argmax}_h \sum_{i=1}^{m} d_i \log(h(x_i) + (1 - d_i) \log(1 - h(x_i)).$$

Error Function for Probabilistic Functions

Switching max and min, we get cross entropy:

$$h_{ML} = \text{argmin}_h - \sum_{i=1}^{m} d_i \log(h(x_i) + (1 - d_i) \log(1 - h(x_i)).$$

Error Function for Probabilistic Functions

Switching max and min, we get cross entropy:

$$h_{ML} = \text{argmin}_h - \sum_{i=1}^{m} d_i \log(h(x_i) + (1 - d_i) \log(1 - h(x_i)).$$

This connection to entropy seems very suspicious.

Bayesian Learning

MDL

Occam’s Razor

“Among competing hypotheses, the one that makes the fewest assumptions should be selected.”

Occam's Razor

"Among competing hypotheses, the one that makes the fewest assumptions should be selected." Alternatively, pick the shortest hypothesis.

We've seen this meme a lot, but it has always seemed wishy washy.

Occam's Razor

Let's consider MAP again. We know that

$$h_{MAP} = \text{argmax}_h \mathbb{P}(D|h)\mathbb{P}(h).$$

Occam's Razor

Taking the log gives

$$h_{MAP} = \text{argmax}_h \log \mathbb{P}(D|h) + \log \mathbb{P}(h).$$

Occam's Razor

Switching max and min gives

$$h_{MAP} = \text{argmin}_h - \log \mathbb{P}(D|h) - \log \mathbb{P}(h).$$

Occam's Razor

Rewriting in terms of lengths, we have

$$h_{MAP} = \text{argmin}_h L_{C_{D|h}}(D|h) - L_{C_h}(h).$$

Occam's Razor

Rewriting in terms of lengths, we have

$$h_{MAP} = \text{argmin}_h L_{C_{D|h}}(D|h) - L_{C_h}(h).$$

That is, the most likely hypothesis is the one that has the shortest description!

Occam’s Razor

Of course, in reality that’s still not quite true.

Bayesian Learning

Bayesian Optimal

Intuition

Suppose we have $H = \{h_1, h_2, h_3\}$. Then, suppose these hypotheses have posterior probabilities (given training data) of $0.4, 0.3, 0.3$ respectively. Clearly, h_1 is our MAP hypothesis. If we obtain a new instance x which is classified as follows: $h_1(x) = 1, h_2(x) = 0, h_3(x) = 0$, what happens?

Bayesian Optimal Learner

Intuition

Suppose we have $H = \{h_1, h_2, h_3\}$. Then, suppose these hypotheses have posterior probabilities (given training data) of $0.4, 0.3, 0.3$ respectively. Clearly, h_1 is our MAP hypothesis. If we obtain a new instance x which is classified as follows: $h_1(x) = 1, h_2(x) = 0, h_3(x) = 0$, we see that the probability that x is negative is actually 0. and the probability that it is a positive instance is actually 0.4. The MAP hypothesis is wrong!

Bayesian Optimal Learner

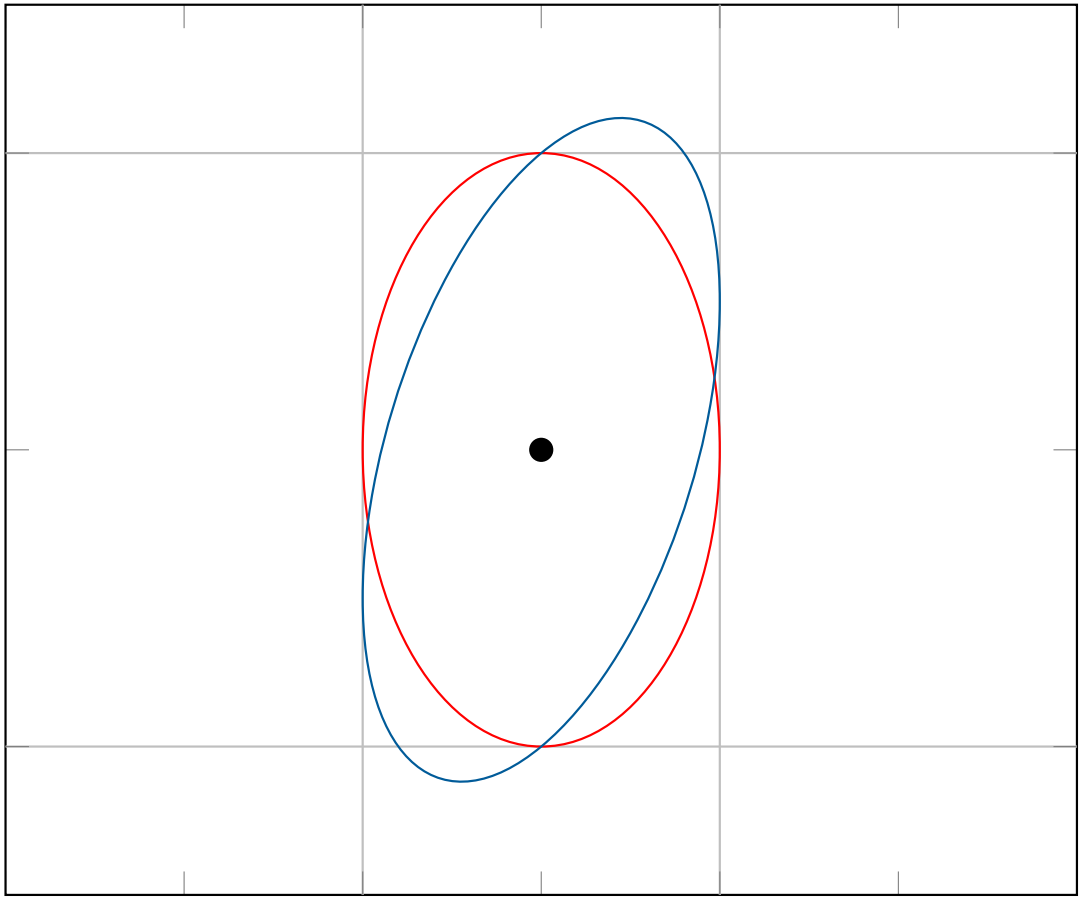
Definition

We define the Bayesian Optimal classification as

$$\text{argmax}_{v_j \in V} \sum_{h_i \in H} \mathbb{P}(v_j|h_i)\mathbb{P}(h_i|D).$$

Computational Learning Theory

Visualizing The Classification Problem



True Error

The last slide showed our model of what a hypothesis is. We are easily able to find things such as training error, but what about the error on the full instance set X?

True Error

Definition

We define **true error** as $e_{\mathcal{D}}(h) = \mathbb{P}_{x \in \mathcal{D}}(c(x) \neq h(x))$.

True Error

But we can't observe true error!

We don't know the underlying distribution (if we did, we would be done!). All we have are our learning examples D. We focus, therefore, on "how probable is it that the observed training error for $h \in H$ gives a misleading estimate of $e_{\mathcal{D}}(h)$?"

Computational Learning Theory

PAC Learning

Probably Approximately Correct Model

We might try characterizing how many examples we would need to observe to obtain a true error of 0. After all, the more data we have, the better. Surely.

Probably Approximately Correct Model

Unfortunately, this is impossible.

Probably Approximately Correct Model

Unfortunately, this is impossible.

- Without looking at all of X, we might have multiple consistent hypotheses and no way to pick the target concept from amongst these.

Probably Approximately Correct Model

Unfortunately, this is impossible.

- Without looking at all of X, we might have multiple consistent hypotheses and no way to pick the target concept from amongst these.
- Because we choose training examples randomly, there is a nonzero chance that we get a misleading set of examples.

Probably Approximately Correct Model

How can we solve these two problems?

Probably Approximately Correct Model

How can we solve these two problems?

First, let's not require that learner be perfectly accurate. We say it can have some error ϵ. Second, let's say that it doesn't have to succeed with this error rate all the time (i.e. for any randomly drawn sequence of examples). It can fail with some probability δ.

Probably Approximately Correct Model

What we have then, is a learner that probably (δ) learns an approximately correct (ϵ) hypothesis.

Probably Approximately Correct Model

Definition

Suppose we have some concept class C over some set of instances X. We say that this class of concepts C is PAC-learnable by a learner L with hypothesis space H if for all $c \in C$, distributions $\mathcal{D}$ over X, ϵ such that $0 \leq \epsilon \leq \frac{1}{2}$, and δ such that $0 \leq \delta \leq \frac{1}{2}$: L learns with probability $1 - \delta$ to output a hypothesis $h \in H$ with true error $e_{\mathcal{D}}(h) \leq \epsilon$ *in time polynomial* with respect to $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, n, and $size(c)$.

Computational Learning Theory

Sample Complexity for Finite Hypothesis Spaces

Characterizing Sample Complexity Using PAC

How is PAC useful?

Characterizing Sample Complexity Using PAC

How is PAC useful?

Recall the definition of version space.

Characterizing Sample Complexity Using PAC

S

uppose we have hypothesis space H, target concept c, instance distribution $\mathcal{D}$ and some training examples D. The version space $VS_{H,D}$ is $\epsilon - exhausted$ with respect to c and $\mathcal{D}$ if for all $h \in VS_{H,D}$, we have that $e_{\mathcal{D}}(h) < \epsilon$.

Characterizing Sample Complexity Using PAC

We fail to ϵ-exhaust a version space if any single hypothesis in a hypothesis space where all hypotheses have true error greater than ϵ are consistent with all of the data.

What is the probability that we fail to ϵ-exhaust the version space?

Characterizing Sample Complexity Using PAC

Say we have m independently drawn examples. For any example, the probability that any single hypothesis succeeds is at most $(1 - \epsilon)$ if our hypothesis space only has hypotheses with true error greater than ϵ.

Characterizing Sample Complexity Using PAC

Over m examples, the probability that this bad hypothesis succeeds on all of them is $(1 - \epsilon)^m$.

Characterizing Sample Complexity Using PAC

Over m examples, the probability that this bad hypothesis succeeds on all of them is $(1-\epsilon)^m$. With $k = |H|$, we see that the probability that at least one of these bad hypotheses is consistent with our training data is $k(1-\epsilon)^m$.

Characterizing Sample Complexity Using PAC

We know that $k \leq |H|$. Therefore, we see that $k(1-\epsilon)^m \leq |H|(1-\epsilon)^m$.

Characterizing Sample Complexity Using PAC

We know that $k \leq |H|$. Therefore, we see that $k(1-\epsilon)^m \leq |H|(1-\epsilon)^m$. Because $(1-\epsilon) \leq e^{-\epsilon}$, this expression is less than or equal to $|H|e^{-\epsilon m}$.

Characterizing Sample Complexity Using PAC

We recall that we bound this probability of failure by δ. Thus, we have the probability of failure is less than or equal to the previous expression which, in turn: $|H|e^{-\epsilon m} \leq \delta$.

Characterizing Sample Complexity Using PAC

Solving for m gives: $m \geq \frac{1}{\epsilon}(\ln|H| + \ln(\frac{1}{\delta}))$.

Characterizing Sample Complexity Using PAC

What happens if we have an infinite hypothesis space?

Computational Learning Theory

VC Dimension

Our Good Friends Vapnik and Chervonenkis

Definition

We say that a hypothesis space H over instance space X has a VC-dimension of d if the size of the largest finite subset of X shattered by H is d.

The Wondrous World of High Dimensionality

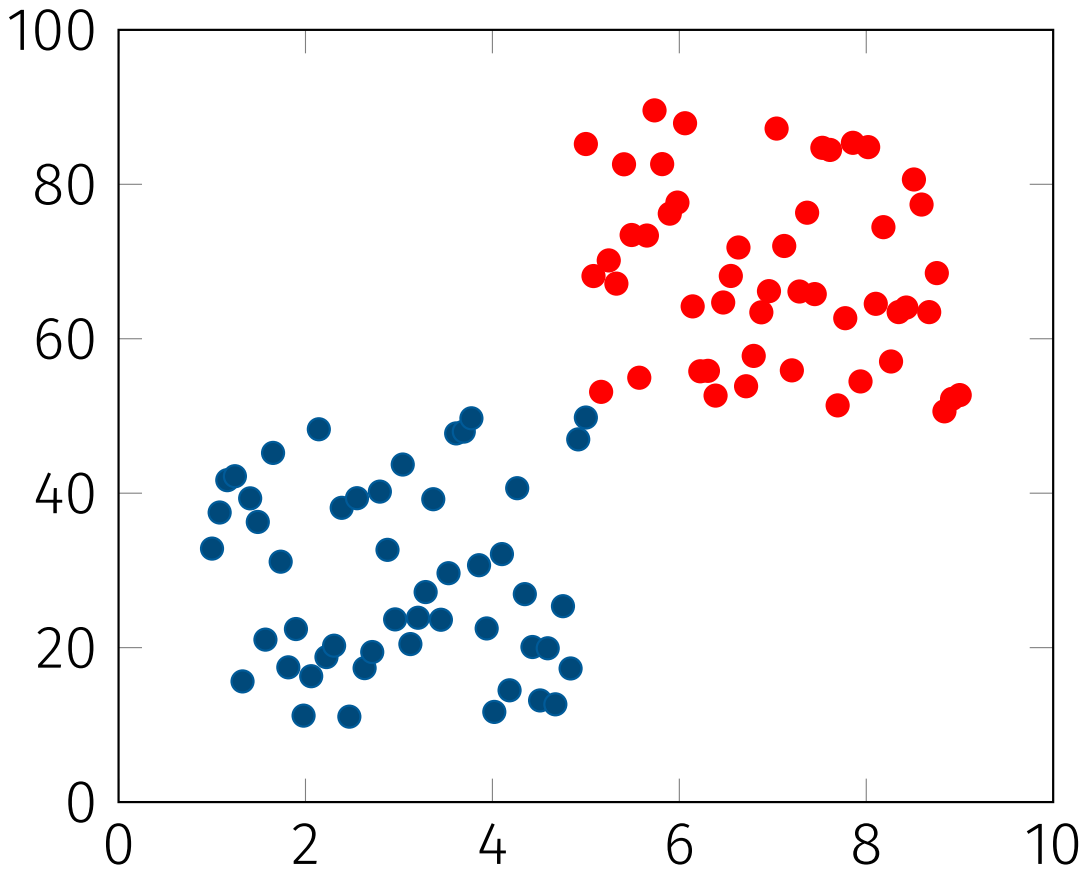
Nonintuitivity

When we try to think about learning in higher dimensions, our intuition often fails us.

The Wondrous World of High Dimensionality

k-Means Clustering

Visualion of k-Means/Clusters



The Wondrous World of High Dimensionality

Spectral Clustering