Lecture 6:
- Learning theory
- Probability Review
Last time... Regularization, Cross-Validation

Validation error

Training error

number of base functions

error

Underfitting
- large training error
- large validation error

Just Right
- small training error
- small validation error

Overfitting
- small training error
- large validation error

The data

NN classifier

5-NN classifier

Figure credit: Fei-Fei Li, Andrej Karpathy, Justin Johnson
Today

• Learning Theory

• Probability Review
Learning Theory: Why ML Works
Computational Learning Theory

• Entire subfield devoted to the mathematical analysis of machine learning algorithms

• Has led to several practical methods:
  - PAC (probably approximately correct) learning → boosting
  - VC (Vapnik–Chervonenkis) theory → support vector machines

Annual conference: Conference on Learning Theory (COLT)
Computational Learning Theory

• Is learning always possible?
• How many training examples will I need to do a good job learning?
• Is my test performance going to be much worse than my training performance?

The key idea that underlies all these answers is that simple functions generalize well.
The Role of Theory

- Theory can serve two roles:
  - It can justify and help understand why common practice works.
  - It can also serve to suggest new algorithms and approaches that turn out to work well in practice.

Often, it turns out to be a mix!

adapted from Hal Daume III
The Role of Theory

• Practitioners discover something that works surprisingly well.

• Theorists figure out why it works and prove something about it.
  – In the process, they make it better or find new algorithms.

• Theory can also help you understand what’s possible and what’s not possible.
Induction is Impossible

• From an algorithmic perspective, a natural question is
  - whether there is an “ultimate” learning algorithm, $A^{\text{awesome}}$, that solves the Binary Classification problem.

• Have you been wasting your time learning about KNN and other methods Perceptron and decision trees, when $A^{\text{awesome}}$ is out there?

• What would such an ultimate learning algorithm do?
  - Take in a data set $D$ and produce a function $f$.
  - No matter what $D$ looks like, this function $f$ should get perfect classification on all future examples drawn from the same distribution that produced $D$.

adapted from Hal Daume III
Induction is Impossible

• From an algorithmic perspective, a natural question is
  − whether there is an “ultimate” learning algorithm, $A_{\text{awesome}}$, that solves the Binary Classification problem.

• Have you been wasting your time learning about KNN and other methods Perceptron and decision trees, when $A_{\text{awesome}}$ is out there?

• What would such an ultimate learning algorithm do?
  − Take in a data set $D$ and produce a function $f$.
  − No matter what $D$ looks like, this function $f$ should get perfect classification on all future examples drawn from the same distribution that produced $D$. 

adapted from Hal Daume III
Label Noise

• Let $X = \{-1, +1\}$ (i.e., a one-dimensional, binary distribution)

  $D = ⟨+1⟩, +1⟩ = 0.4 \quad D = ⟨-1⟩, -1⟩ = 0.4$
  $D = ⟨+1⟩, -1⟩ = 0.1 \quad D = ⟨-1⟩, +1⟩ = 0.1$

  – 80% of data points in this distribution have $x = y$ and 20% don’t.

• No matter what function your learning algorithm produces, there’s no way that it can do better than 20% error on this data.

  – No $A^{\text{awesome}}$ exists that always achieves an error rate of zero.
  – The best that we can hope is that the error rate is not “too large.”
Sampling

• Another source of difficulty comes from the fact that the only access we have to the data distribution is through sampling.
  - When trying to learn about a distribution, you only get to see data points drawn from that distribution.
  - You know that “eventually” you will see enough data points that your sample is representative of the distribution, but it might not happen immediately.

• For instance, even though a fair coin will come up heads only with probability 1/2, it’s completely plausible that in a sequence of four coin flips you never see a tails, or perhaps only see one tails.
Induction is Impossible

• We need to understand that $A_{\text{awesome}}$ will not always work.
  – In particular, if we happen to get a lousy sample of data from $D$, we need to allow $A_{\text{awesome}}$ to do something completely unreasonable.

• We cannot hope that $A_{\text{awesome}}$ will do perfectly, every time.

The best we can reasonably hope of $A_{\text{awesome}}$ is that it will do pretty well, most of the time.
Probably Approximately Correct (PAC) Learning

- A formalism based on the realization that the best we can hope of an algorithm is that
  - It does a good job most of the time (probably approximately correct)
Consider a hypothetical learning algorithm

- We have 10 different binary classification data sets.
- For each one, it comes back with functions $f_1, f_2, \ldots, f_{10}$.
  - For some reason, whenever you run $f_4$ on a test point, it crashes your computer. For the other learned functions, their performance on test data is always at most 5% error.
  - If this situation is guaranteed to happen, then this hypothetical learning algorithm is a PAC learning algorithm.

It satisfies probably because it only failed in one out of ten cases, and it’s approximate because it achieved low, but non-zero, error on the remainder of the cases.
PAC Learning

Definitions 1. An algorithm $A$ is an $(\epsilon, \delta)$-PAC learning algorithm if, for all distributions $D$: given samples from $D$, the probability that it returns a “bad function” is at most $\delta$; where a “bad” function is one with test error rate more than $\epsilon$ on $D$. 

adapted from Hal Daume III
PAC Learning

- Two notions of efficiency
  - **Computational complexity**: Prefer an algorithm that runs quickly to one that takes forever
  - **Sample complexity**: The number of examples required for your algorithm to achieve its goals

**Definition**: An algorithm $A$ is an efficient $(\epsilon, \delta)$-PAC learning algorithm if it is an $(\epsilon, \delta)$-PAC learning algorithm whose runtime is polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$.

In other words, to let your algorithm to achieve 4% error rather than 5%, the runtime required to do so should not go up by an exponential factor!
Example: PAC Learning of Conjunctions

- Data points are binary vectors, for instance \( \mathbf{x} = \langle 0, 1, 1, 0, 1 \rangle \)
- Some Boolean conjunction defines the true labeling of this data (e.g. \( x_1 \land x_2 \land x_5 \))
- There is some distribution \( D_X \) over binary data points (vectors) \( \mathbf{x} = \langle x_1, x_2, \ldots, x_D \rangle \).
- There is a fixed concept conjunction \( c \) that we are trying to learn.
- There is no noise, so for any example \( x \), its true label is simply \( y = c(x) \)

**Example:**
- Clearly, the true formula cannot include the terms \( x_1, x_2, \neg x_3, \neg x_4 \)

<table>
<thead>
<tr>
<th>( y )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>+1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

adapted from Hal Daume III
Example: PAC Learning of Conjunctions

<table>
<thead>
<tr>
<th>y</th>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
<th>x₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>+1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
f_0(x) = x_1 \land \neg x_1 \land x_2 \land \neg x_2 \land x_3 \land \neg x_3 \land x_4 \land \neg x_4
\]

\[
f_1(x) = \neg x_1 \land \neg x_2 \land x_3 \land x_4
\]

\[
f_2(x) = \neg x_1 \land x_3 \land x_4
\]

\[
f_3(x) = \neg x_1 \land x_3 \land x_4
\]

- After processing an example, it is guaranteed to classify that example correctly (provided that there is no noise)
- Computationally very efficient
  - Given a data set of N examples in D dimensions, it takes \(O(ND)\) time to process the data. This is linear in the size of the data set.

```
Algorithm 30 BinaryConjunctionTrain(D)
1. f ← x₁ \land \neg x₁ \land x₂ \land \neg x₂ \land \cdots \land x_D \land \neg x_D
2. for all positive examples (x,+1) in D do
3.   for d = 1 \ldots D do
4.     if x_d = 0 then
5.       f ← f without term “x_d”
6.     else
7.       f ← f without term “\neg x_d”
8.   end if
9. end for
10. end for
11. return f
```

“Throw Out Bad Terms”
Example: PAC Learning of Conjunctions

<table>
<thead>
<tr>
<th>$y$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>+1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

• Is this an efficient $(\varepsilon, \delta)$-PAC learning algorithm?

• What about sample complexity?
  - How many examples $N$ do you need to see in order to guarantee that it achieves an error rate of at most $\varepsilon$ (in all but $\delta$-many cases)?
  - Perhaps $N$ has to be gigantic (like $2^{2^D/\varepsilon}$) to (probably) guarantee a small error.

Algorithm 30 BinaryConjunctionTrain(D)

```
1: $f \leftarrow x_1 \land \neg x_1 \land x_2 \land \neg x_2 \land \cdots \land x_D \land \neg x_D$ // initialize function
2: for all positive examples $(x_i, +1)$ in D do
3:   for $d = 1 \ldots D$ do
4:     if $x_d = 0$ then
5:       $f \leftarrow f$ without term “$x_d$”
6:     else
7:       $f \leftarrow f$ without term “$\neg x_d$”
8:     end if
9:   end for
10: end for
11: return $f$
```

"Throw Out Bad Terms"
Occam’s Razor

“If one can explain a phenomenon without assuming this or that hypothetical entity, then there is no ground for assuming it i.e. that one should always opt for an explanation in terms of the fewest possible number of causes, factors, or variables.”

- Simple solutions generalize well
- The hypothesis class $H$, is the set of all boolean formulae over $D$-many variables.
  - The hypothesis class for Boolean conjunctions is finite; the hypothesis class for linear classifiers is infinite.
  - For Occam’s razor, we can only work with finite hypothesis classes.

**Theorem 14** (Occam’s Bound). Suppose $A$ is an algorithm that learns a function $f$ from some finite hypothesis class $H$. Suppose the learned function always gets zero error on the training data. Then, the sample complexity of $f$ is at most $\log |H|$.
Complexity of Infinite Hypothesis Spaces

- Occam’s Bound is is completely useless when $|H| = \infty$

- In example, instead of representing your hypothesis as a Boolean conjunction, represent it as a conjunction of inequalities.
  - Instead of having $x_1 \land \neg x_2 \land x_5$, you have $[x_1 > 0.2] \land [x_2 < 0.77] \land [x_5 < \pi/4]$
  - In this representation, for each feature, you need to choose an inequality ($<$ or $>$) and a threshold.
  - Since the thresholds can be arbitrary real values, there are now infinitely many possibilities: $|H| = 2^{D \times \infty} = \infty$
Vapnik-Chervonenkis (VC) Dimension

- A classic measure of complexity of infinite hypothesis classes based on this intuition.

- The VC dimension is a very classification-oriented notion of complexity
  - The idea is to look at a finite set of unlabeled examples
  - no matter how these points were labeled, would we be able to find a hypothesis that correctly classifies them

- The idea is that as you add more points, being able to represent an arbitrary labeling becomes harder and harder.

**Definitions 2.** For data drawn from some space $\mathcal{X}$, the VC dimension of a hypothesis space $\mathcal{H}$ over $\mathcal{X}$ is the maximal $K$ such that: there exists a set $X \subseteq \mathcal{X}$ of size $|X| = K$, such that for all binary labelings of $X$, there exists a function $f \in \mathcal{H}$ that matches this labeling.
How many points can a linear boundary classify exactly? (1-D)

- 2 points:
  Yes!

- 3 points:
  No!

VC-dimension = 2

e tc (8 total)
How many points can a linear boundary classify exactly? (2-D)

• 3 points:
  Yes!

• 4 points:
  No!

VC-dimension = 3
Basic Probability Review
Probability

• A is non-deterministic event
  – Can think of A as a boolean-valued variable

• Examples
  – A = your next patient has cancer
  – A = Rafael Nadal wins French Open 2019
Interpreting Probabilities

If I flip this coin, the probability that it will come up heads is 0.5

- **Frequentist Interpretation:** If we flip this coin many times, it will come up heads about half the time. *Probabilities are the expected frequencies of events over repeated trials.*

- **Bayesian Interpretation:** I believe that my next toss of this coin is equally likely to come up heads or tails. *Probabilities quantify subjective beliefs about single events.*

- Viewpoints play complementary roles in **machine learning:**
  - Bayesian view used to build models based on domain knowledge, and automatically derive learning algorithms
  - Frequentist view used to analyze worst case behavior of learning algorithms, in limit of large datasets

- From either view, basic mathematics is the same!
The Axioms Of Probability
Axioms of Probability

• \(0 \leq P(A) \leq 1\)
• \(P(\text{empty-set}) = 0\)
• \(P(\text{everything}) = 1\)
• \(P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)\)
Interpreting the Axioms

- $0 \leq P(A) \leq 1$
- $P(\text{empty-set}) = 0$
- $P(\text{everything}) = 1$
- $P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$
Interpreting the Axioms

• $0 \leq P(A) \leq 1$
• $P(\text{empty-set}) = 0$
• $P(\text{everything}) = 1$
• $P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$

The area of A cannot get any smaller than 0

And a zero area would mean no world could ever have A true
Interpreting the Axioms

- \(0 \leq P(A) \leq 1\)
- \(P(\text{empty-set}) = 0\)
- \(P(\text{everything}) = 1\)
- \(P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)\)

The area of A can't get any bigger than 1

And an area of 1 would mean all worlds will have A true
Interpreting the Axioms

- $0 \leq P(A) \leq 1$
- $P(\text{empty-set}) = 0$
- $P(\text{everything}) = 1$
- $P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$
Discrete Random Variables

\[ X \rightarrow \text{discrete random variable} \]
\[ \mathcal{X} \rightarrow \text{sample space of possible outcomes, which may be finite or countably infinite} \]
\[ x \in \mathcal{X} \rightarrow \text{outcome of sample of discrete random variable} \]
Discrete Random Variables

$X$ $\rightarrow$ discrete random variable

$\mathcal{X}$ $\rightarrow$ sample space of possible outcomes, which may be finite or countably infinite

$x \in \mathcal{X}$ $\rightarrow$ outcome of sample of discrete random variable

$p(X = x)$ $\rightarrow$ probability distribution (probability mass function)

$p(x)$ $\rightarrow$ shorthand used when no ambiguity

$0 \leq p(x) \leq 1$ for all $x \in \mathcal{X}$

$\sum_{x \in \mathcal{X}} p(x) = 1$

$\mathcal{X} = \{1, 2, 3, 4\}$

uniform distribution

degenerate distribution
Joint Distribution
Marginalization

- **Marginalization**
  - Events: \( P(A) = P(A \text{ and } B) + P(A \text{ and } \neg B) \)
  
- Random variables \( P(X = x) = \sum_{y} P(X = x, Y = y) \)
Marginal Distributions

\[ p(x, y) = \sum_{z \in \mathcal{Z}} p(x, y, z) \]

\[ p(x) = \sum_{y \in \mathcal{Y}} p(x, y) \]
Conditional Probabilities

- $P(Y=y \mid X=x)$
- What do you believe about $Y=y$, if I tell you $X=x$?
- $P($Rafael Nadal wins French Open 2019$)$?
- What if I tell you:
  - He has won the French Open 11/13 he has played there
  - Rafael Nadal is ranked 1
Conditional Probabilities

- \( P(A \mid B) = \) In worlds that where \( B \) is true, fraction where \( A \) is true

- Example
  - \( H \): “Have a headache”
  - \( F \): “Coming down with Flu”

  \[
  \begin{align*}
  P(H) &= 1/10 \\
  P(F) &= 1/40 \\
  P(H\mid F) &= 1/2 \\
  \end{align*}
  \]

  Headaches are rare and flu is rarer, but if you’re coming down with flu there’s a 50-50 chance you’ll have a headache.
Conditional Distributions

\[ p(x, y \mid Z = z) = \frac{p(x, y, z)}{p(z)} \]
Independent Random Variables

\[ p(x, y) = p(x) \cdot p(y) \]

for all \( x \in \mathcal{X}, y \in \mathcal{Y} \)

Equivalent conditions on conditional probabilities:

\[ p(x \mid Y = y) = p(x) \quad \text{and} \quad p(y) > 0 \quad \text{for all} \quad y \in \mathcal{Y} \]

\[ p(y \mid X = x) = p(y) \quad \text{and} \quad p(x) > 0 \quad \text{for all} \quad x \in \mathcal{X} \]
Bayes Rule (Bayes Theorem)

\[ p(x, y) = p(x)p(y \mid x) = p(y)p(x \mid y) \]

\[ p(y \mid x) = \frac{p(x, y)}{p(x)} = \frac{p(x \mid y)p(y)}{\sum_{y' \in Y} p(y')p(x \mid y')} \]

\[ \propto p(x \mid y)p(y) \]

• A basic identity from the definition of conditional probability
• Used in ways that have no thing to do with Bayesian statistics!
• Typical application to learning and data analysis:

\[ Y \rightarrow \text{unknown parameters we would like to infer} \]
\[ X = x \rightarrow \text{observed data available for learning} \]
\[ p(y) \rightarrow \text{prior distribution (domain knowledge)} \]
\[ p(x \mid y) \rightarrow \text{likelihood function (measurement model)} \]
\[ p(y \mid x) \rightarrow \text{posterior distribution (learned information)} \]
Binary Random Variables

- **Bernoulli Distribution**: Single toss of a (possibly biased) coin

  \[ X = \{0, 1\} \]
  \[ 0 \leq \theta \leq 1 \]
  \[ \text{Ber}(x \mid \theta) = \theta^{\delta(x,1)}(1 - \theta)^{\delta(x,0)} \]

- **Binomial Distribution**: Toss a single (possibly biased) coin \( n \) times, and report the number \( k \) of times it comes up

  \[ K = \{0, 1, 2, \ldots, n\} \]
  \[ 0 \leq \theta \leq 1 \]
  \[ \text{Bin}(k \mid n, \theta) = \binom{n}{k} \theta^k (1 - \theta)^{n-k} \]
  \[ \binom{n}{k} = \frac{n!}{(n-k)!k!} \]
Binomial Distributions

\[
\theta = 0.250
\]

\[
\theta = 0.500
\]

\[
\theta = 0.900
\]
Bean Machine (Sir Francis Galton)

http://en.wikipedia.org/wiki/Bean_machine
Categorical Random Variables

- **Multinoulli Distribution:** Single roll of a (possibly biased) die

\[ \mathcal{X} = \{0, 1\}^K, \sum_{k=1}^{K} x_k = 1 \]

\[ \theta = (\theta_1, \theta_2, \ldots, \theta_K), \theta_k \geq 0, \sum_{k=1}^{K} \theta_k = 1 \]

\[ \text{Cat}(x \mid \theta) = \prod_{k=1}^{K} \theta_k^{x_k} \]

- **Multinomial Distribution:** Roll a single (possibly biased) die \( n \) times, and report the number \( n_k \) of each possible outcome

\[ \text{Mu}(x \mid n, \theta) = \binom{n}{n_1 \ldots n_K} \prod_{k=1}^{K} \theta_k^{n_k} \]

\[ n_k = \sum_{i=1}^{n} x_{ik} \]
Aligned DNA Sequences

c g a t a c g g g g t c g a a
c a a t c c g a g a t c g c a
c a a t c c g t g t t g g g a
c a a t c g g c a t g c g g g
c g a g c c g c g t a c g a a
c a t a c g g g a g c a c g a a
t a a t c c g g g g c a t g t a
c g a g c c g a g t a c a g a
c c a t c c g c g t a a g c a
g g a t a c g g a t g a c a
Multinomial Model of DNA

Sequence Position

Bits
Next Lecture:
Maximum Likelihood Estimation (MLE)