Administration

- Mid-term

- Projects
  - We are late with responses
  - But, don’t wait – **start working.**
  - Intermediate project reports are due on **Thursday, November 19.**
  - Tentative:
    - Final Reports: December 16; Presentations: ???
## Midterm

<table>
<thead>
<tr>
<th>Stat</th>
<th>Overall Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>1</td>
</tr>
<tr>
<td>max</td>
<td>98</td>
</tr>
<tr>
<td>median</td>
<td>58</td>
</tr>
<tr>
<td>mean</td>
<td>58.02201258</td>
</tr>
<tr>
<td>STD</td>
<td>16.46845334</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stat</th>
<th>Grads/Undergrads</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean (G)</td>
<td>62.51829268</td>
</tr>
<tr>
<td>mean (U)</td>
<td>53.23376623</td>
</tr>
<tr>
<td>STD (G)</td>
<td>15.65216924</td>
</tr>
<tr>
<td>STD (U)</td>
<td>16.05246014</td>
</tr>
</tbody>
</table>

Histogram (Overall scores)
Midterm

Overall Scores (Grads)

Overall Score (Undergrads)
• Exams are available at Dawn Cheek’s office (3318sc)

• Questions/Regrades: Send me email with title: "CS446 midterm grading username".

• Which question/what problems

Histogram (Q1: Short Answer Questions)

Histogram (Q2: Kernels)

Histogram (Q3: Online Learning)

Histogram (Q4: Decision Trees)
Consider a distribution $D$ over space $X \times Y$

- $X$ - the instance space;
- $Y$ - set of labels. (e.g. +/-1)

Can think about the data generation process as governed by $D(x)$, and the labeling process as governed by $D(y|x)$, such that

$$D(x,y) = D(x) \cdot D(y|x)$$

This can be used to model both the case where labels are generated by a function $y = f(x)$, as well as noisy cases and probabilistic generation of the label.

- If the distribution $D$ is known, there is no learning. We can simply predict
  $$y = \arg\max_y D(y|x)$$

- If we are looking for a hypothesis, we can simply find the one that minimizes the probability of mislabeling:
  $$h = \arg\min_h E_{(x,y) \sim D} [[h(x) \neq y]]$$
Inductive learning comes into play when the distribution is not known.

Then, there are two basic approaches to take.

- Discriminative (direct) learning

and

- Bayesian Learning (Generative)

Running example: Text Correction:

“I saw the girl it the park” → I saw the girl in the park
1: Direct Learning

- Model the problem of text correction as a problem of learning from examples.
- Goal: learn directly how to make predictions.

PARADIGM

- Look at many (positive/negative) examples.
- Discover some regularities in the data.
- Use these to construct a prediction policy.
- A policy (a function, a predictor) needs to be specific.
  
  [it/in] rule: if the occurs after the target ⇒ in

- Assumptions comes in the form of a hypothesis class.

Bottom line: approximating h : X → Y, is estimating P(Y|X).
Direct Learning (2)

- Consider a distribution $D$ over space $X \times Y$
- $X$ - the instance space; $Y$ - set of labels. (e.g. +/-1)
- Given a sample $\{(x,y)\}_{1}^{m}$, and a loss function $L(x,y)$
- Find $h \in H$ that minimizes
  $$\sum_{i=1}^{m} D(x_i,y_i) L(h(x_i),y_i) + \text{Reg}$$

- $L$ can be:
  - $L(h(x),y)=1, h(x) \neq y, \text{ o/w } L(h(x),y) = 0$ (0-1 loss)
  - $L(h(x),y)=(h(x)-y)^2, \quad (L_2)$
  - $L(h(x),y)=\max\{0,1-y\ h(x)\}$ (hinge loss)
  - $L(h(x),y)=\exp\{-y\ h(x)\}$ (exponential loss)

- Guarantees: If we find an algorithm that minimizes loss on the observed data. Then, learning theory guarantees good future behavior (as a function of $H$).
Model the problem of text correction as that of generating correct sentences.

Goal: learn a model of the language; use it to predict.

**PARADIGM**

Learn a probability distribution over all sentences
- In practice: make assumptions on the distribution’s type

Use it to estimate which sentence is more likely.
- \( \Pr(\text{I saw the girl it the park}) \not\Rightarrow \Pr(\text{I saw the girl in the park}) \)
- In practice: a decision policy depends on the assumptions

Bottom line: the generating paradigm approximates
\[
P(X,Y) = P(X|Y) P(Y).
\]

**Guarantees:** We need to assume the “right” probability distribution
There are actually two different notions.

Learning probabilistic concepts
- The learned concept is a function $c : X \rightarrow [0, 1]$
- $c(x)$ may be interpreted as the probability that the label 1 is assigned to $x$
- The learning theory that we have studied before is applicable (with some extensions).

Bayesian Learning: Use of a probabilistic criterion in selecting a hypothesis
- The hypothesis can be deterministic, a Boolean function.

It’s not the hypothesis – it’s the process.
Goal: find the best hypothesis from some space $H$ of hypotheses, given the observed data $D$.

Define best to be: most probable hypothesis in $H$.

In order to do that, we need to assume a probability distribution over the class $H$.

In addition, we need to know something about the relation between the data observed and the hypotheses (E.g., a coin problem.)

As we will see, we will be Bayesian about other things, e.g., the parameters of the model.
Basics of Bayesian Learning

- **P(h)** - the **prior probability** of a hypothesis **h**
  Reflects background knowledge; before data is observed. If no information - uniform distribution.

- **P(D)** - The probability that **this sample** of the Data is observed. (No knowledge of the hypothesis)

- **P(D|h)**: The probability of observing the sample **D**, given that hypothesis **h** is the target

- **P(h|D)**: The **posterior probability** of **h**. The probability that **h** is the target, given that **D** has been observed.
Bayes Theorem

\[ P(h \mid D) = \frac{P(D \mid h) P(h)}{P(D)} \]

- \( P(h \mid D) \) increases with \( P(h) \) and with \( P(D \mid h) \)
- \( P(h \mid D) \) decreases with \( P(D) \)
Basic Probability

- **Product Rule:** \( P(A,B) = P(A|B)P(B) = P(B|A)P(A) \)

- If \( A \) and \( B \) are independent:
  - \( P(A,B) = P(A)P(B) \); \( P(A|B) = P(A) \), \( P(A|B,C) = P(A|C) \)

- **Sum Rule:** \( P(A \lor B) = P(A) + P(B) - P(A,B) \)

- **Bayes Rule:** \( P(A|B) = P(B|A)P(A)/P(B) \)

- **Total Probability:**
  - If events \( A_1, A_2, \ldots, A_n \) are mutually exclusive: \( A_i \cap A_j = \emptyset \), \( \sum_i P(A_i) = 1 \)
  - \( P(B) = \sum_i P(B, A_i) = \sum_i P(B|A_i)P(A_i) \)

- **Total Conditional Probability:**
  - If events \( A_1, A_2, \ldots, A_n \) are mutually exclusive: \( A_i \cap A_j = \emptyset \), \( \sum_i P(A_i) = 1 \)
  - \( P(B|C) = \sum_i P(B, A_i|C) = \sum_i P(B|A_i,C)P(A_i|C) \)
The learner considers a set of candidate hypotheses $H$ (models), and attempts to find the most probable one $h \in H$, given the observed data.

Such maximally probable hypothesis is called maximum a posteriori hypothesis (MAP); Bayes theorem is used to compute it:

$$h_{\text{MAP}} = \arg\max_{h \in H} P(h|D) = \arg\max_{h \in H} \frac{P(D|h) P(h)}{P(D)} = \arg\max_{h \in H} P(D|h) P(h)$$
Learning Scenario (2)

\[ h_{\text{MAP}} = \arg\max_{h \in \mathcal{H}} P(h \mid D) = \arg\max_{h \in \mathcal{H}} P(D \mid h) P(h) \]

- We may assume that a priori, hypotheses are equally probable:
  \[ P(h_i) = P(h_j) \quad \forall h_i, h_j \in \mathcal{H} \]

- We get the **Maximum Likelihood hypothesis**:
  \[ h_{\text{ML}} = \arg\max_{h \in \mathcal{H}} P(D \mid h) \]

- Here we just look for the hypothesis that best explains the data
A given coin is either **fair** or has a 60% bias in favor of Head. Decide what is the bias of the coin [This is a learning problem!]

Two hypotheses: $h_1$: P(H)=0.5; $h_2$: P(H)=0.6
- **Prior**: P(h): P(h₁)=0.75  P(h₂)=0.25
- Now we need Data. 1\textsuperscript{st} Experiment: coin toss is H.
- **P(D|h)**:
  - P(D|h₁)=0.5 ; P(D|h₂) =0.6
- **P(D)**:
  - $P(D)=P(D|h_1)P(h_1) + P(D|h_2)P(h_2)$
  - $= 0.5 \times 0.75 + 0.6 \times 0.25 = 0.525$
- **P(h|D)**:
  - P(h₁|D) = P(D|h₁)P(h₁)/P(D) = 0.5\times0.75/0.525 = 0.714
  - P(h₂|D) = P(D|h₂)P(h₂)/P(D) = 0.6\times0.25/0.525 = 0.286
Examples (2)

\[ h_{\text{MAP}} = \underset{h \in \mathcal{H}}{\text{argmax}} \quad P(h|D) = \underset{h \in \mathcal{H}}{\text{argmax}} \quad P(D|h) \cdot P(h) \]

A given coin is either fair or has a 60% bias in favor of Head.

**Decide** what is the bias of the coin [This is a learning problem!]

Two hypotheses: \( h_1: P(H) = 0.5; \quad h_2: P(H) = 0.6 \)
- Prior: \( P(h): P(h_1) = 0.75 \quad P(h_2) = 0.25 \)

After 1\(^{st}\) coin toss is H we still think that the coin is more likely to be fair.

If we were to use Maximum Likelihood approach (i.e., assume equal priors) we would think otherwise. The data supports the biased coin better.

Try: 100 coin tosses; 70 heads.

You will believe that the coins is biased.
A given coin is either **fair** or has a 60% bias in favor of Head. **Decide** what is the bias of the coin *[This is a learning problem!]*

Two hypotheses: $h_1$: $P(H)=0.5$; $h_2$: $P(H)=0.6$

- Prior: $P(h)$: $P(h_1)=0.75$ $P(h_2)=0.25$

Case of 100 coin tosses; 70 heads.

$$P(D) = P(D|h_1) P(h_1) + P(D|h_2) P(h_2) =$$

$$= 0.5^{100} \cdot 0.75 + 0.6^{70} \cdot 0.4^{30} \cdot 0.25 =$$

$$= 7.9 \cdot 10^{-31} \cdot 0.75 + 3.4 \cdot 10^{-28} \cdot 0.25$$

0.0057 = $P(h_1|D) = P(D|h_1) P(h_1)/P(D) \ll P(D|h_2) P(h_2) /P(D) = P(h_2|D) = 0.9943$
Example: Learning a Concept Class

- Assume that we are given a concept class $C$.

- Given a collection of examples $(x,f(x))$,

- For $f \in C$, we try to identify $h$ that is consistent with $f$ on the training data. We showed that it will do well in the future.

- What will the Bayesian approach tell us?
Learning a Concept Class

- **P(h):** the prior probability of a hypothesis h: 
  \[ p(h) = \frac{1}{|H|} \text{ for all } h \text{ in } H \]

- **P(D|h):** Let d=(x,\ell) be an observed labeled example 
  \[ P(\{(x,\ell)\}^m_1 | h) = 1, \text{ if } \forall x \ h(x)=\ell; \quad P(\{(x,\ell)\}^m_1 | h) = 0 \text{ otherwise} \]

- **P(D):** 
  For a set D of m examples: 
  \[ P(D) = \sum_{h_i \in H} P(D|h_i)P(h_i) = \frac{|H_{\text{CON}}|}{|H|} \]
  where \( H_{\text{CON}} \) is the set of hypotheses in H which are consistent with the sample D

- **P(h|D):** via Bayes rule 
  \[ P(h | D) = \frac{P(D | h)P(h)}{P(D)} = \begin{cases} \frac{1}{|H_{\text{CON}}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases} \]
Example: A Model of Language

- **Model 1:** There are 5 characters, A, B, C, D, E, and space.
- At any point can generate any of them, according to:
  
  \[ P(A) = p_1; \quad P(B) = p_2; \quad P(C) = p_3; \quad P(D) = p_4; \quad P(E) = p_5; \quad P(SP) = p_6 \]
  
  \[ \sum_i p_i = 1 \]
  
  E.g., \( P(A) = 0.3; \quad P(B) = 0.1; \quad P(C) = 0.2; \quad P(D) = 0.2; \quad P(E) = 0.1 \quad P(SP) = 0.1 \)

- We assume a **generative model** of independent characters:
  
  \[ P(U) = P(x_1, x_2, \ldots, x_k) = \prod_{i=1,k} P(x_i \mid x_{i+1}, x_{i+2}, \ldots, x_k) = \prod_{i=1,k} P(x_i) \]

- The **parameters of the model** are the character generation probabilities (**Unigram**).
- **Goal:** to determine which of two strings \( U, V \) is more likely.
- **The Bayesian way:** compute the probability of each string, and decide which is more likely.

Consider Strings: AABBC & ABBBA

- **Learning here is:** learning the parameters of a known model family
- **How?**

You observe a string; use it to learn the language model.
E.g., \( S = \text{AABBABC}; \quad \text{Compute } P(A) \)
Assume that you toss a \((p, 1-p)\) coin \(m\) times and get \(k\) Heads, \(m-k\) Tails. What is \(p\)?

If \(p\) is the probability of Head, the probability of the data observed is:

\[
P(D|p) = p^k (1-p)^{m-k}
\]

The log Likelihood:

\[
L(p) = \log P(D|p) = k \log(p) + (m-k)\log(1-p)
\]

To maximize, set the derivative w.r.t. \(p\) equal to 0:

\[
dL(p)/dp = k/p - (m-k)/(1-p)
\]

Solving this for \(p\), gives:

\[
p = k/m
\]

1. The model we assumed is binomial. You could assume a different model! Next we will consider other models and see how to learn their parameters.

2. In practice, smoothing is advisable – deriving the right smoothing can be done by assuming a prior.
Bernoulli Distribution:
- Random Variable $X$ takes values $\{0, 1\}$ s.t. $P(X=1) = p = 1 - P(X=0)$

Binomial Distribution:
- Random Variable $X$ takes values $\{1, 2, \ldots, n\}$ representing the number of successes $(X=1)$ in $n$ Bernoulli trials.
- $P(X=k) = f(n, p, k) = C_n^k p^k (1-p)^{n-k}$
- Note that if $X \sim \text{Binom}(n, p)$ and $Y \sim \text{Bernoulli}(p)$, $X = \sum_{i=1}^n Y$
Categorical Distribution:
- Random Variable $X$ takes on values in $\{1, 2, \ldots, k\}$ s.t. $P(X=i) = p_i$ and $\sum_{1}^{k} p_i = 1$

Multinomial Distribution: is to Categorical what Binomial is to Bernoulli
- Let the random variables $X_i$ ($i=1, 2, \ldots, k$) indicates the number of times outcome $i$ was observed over the $n$ trials.
- The vector $X = (X_1, \ldots, X_k)$ follows a multinomial distribution $(n,p)$ where $p = (p_1, \ldots, p_k)$ and $\sum_{1}^{k} p_i = 1$
- $f(x_1, x_2, \ldots, x_k, n, p) = P(X_1 = x_1, \ldots, X_k = x_k) = \frac{n!}{x_1! \cdots x_k!} p_1^{x_1} \cdots p_k^{x_k}$, when $\sum_{i=1}^{k} x_i = n$
Our eventual goal will be: Given a document, predict whether it’s “good” or “bad”

A Multinomial Bag of Words

- We are given a collection of documents written in a three word language \{a, b, c\}. All the documents have exactly \(n\) words (each word can be either a, b or c).
- We are given a labeled document collection \(\{D_1, D_2 \ldots , D_m\}\). The label \(y_i\) of document \(D_i\) is 1 or 0, indicating whether \(D_i\) is “good” or “bad”.
- This model uses the multinomial distribution. That is, \(a_i (b_i, c_i, \text{resp.})\) is the number of times word a (b, c, resp.) appears in document \(D_i\).
- Therefore: \(a_i + b_i + c_i = |D_i| = n\).
- In this generative model, we have:
  \[
P(D_i | y = 1) = n!/(a_i! \ b_i! \ c_i!) \ \alpha^a_1 \ \beta^b_1 \ \gamma^c_1
  \]
  where \(\alpha_1 (\beta_1, \gamma_1 \text{ resp.})\) is the probability that a (b, c) appears in a “good” document.
- Similarly,
  \[
P(D_i | y = 0) = n!/(a_i! \ b_i! \ c_i!) \ \alpha^a_0 \ \beta^b_0 \ \gamma^c_0
  \]
- Note that: \(\alpha_0 + \beta_0 + \gamma_0 = \alpha_1 + \beta_1 + \gamma_1 = 1\)

Unlike the discriminative case, the “game” here is different:
- We make an assumption on how the data is being generated.
  - (multinomial, with \(\alpha_i, \beta_i, \gamma_i\))
- Now, we observe documents, and estimate these parameters.
- Once we have the parameters, we can predict the corresponding label.
We are given a collection of documents written in a three word language \{a, b, c\}. All the documents have exactly \(n\) words (each word can be either \(a\), \(b\) or \(c\)).

We are given a labeled document collection \(\{D_1, D_2 \ldots, D_m\}\). The label \(y_i\) of document \(D_i\) is \(1\) or \(0\), indicating whether \(D_i\) is “good” or “bad”.

The classification problem: given a document \(D\), determine if it is good or bad; that is, determine \(P(y|D)\).

This can be determined via Bayes rule: \(P(y|D) = P(D|y) \frac{P(y)}{P(D)}\)

But, we need to know the parameters of the model to compute that.
How do we estimate the parameters?

We derive the most likely value of the parameters defined above, by maximizing the log likelihood of the observed data.

\[
PD = \prod_i P(y_i, D_i) = \prod_i P(D_i | y_i) P(y_i) =
\]

We denote by \( P(y_i) = \eta \) the probability that an example is “good” \((y_i=1); \text{ otherwise } y_i=0\). Then:

\[
\prod_i P(y, D_i) = \prod_i \left( \eta \frac{n!}{(a_i! b_i! c_i!)} \alpha_1^{a_i} \beta_1^{b_i} \gamma_1^{c_i} \right)^{y_i} \cdot \left(1 - \eta \right)^{n!/(a_i! b_i! c_i!)} \alpha_0^{a_i} \beta_0^{b_i} \gamma_0^{c_i} \right)^{1-y_i}
\]

We want to maximize it with respect to each of the parameters. We first compute \( \log(PD) \) and then differentiate:

\[
\log(PD) = \sum_i y_i \left[ \log(\eta) + C + a_i \log(\alpha_1) + b_i \log(\beta_1) + c_i \log(\gamma_1) + (1- y_i) \left[ \log(1-\eta) + C' + a_i \log(\alpha_0) + b_i \log(\beta_0) + c_i \log(\gamma_0) \right] \right]
\]

\[
d\log PD/ d \eta = \sum_i \left[ y_i /\eta - (1-y_i)/(1-\eta) \right] = 0 \quad \Rightarrow \quad \sum_i (y_i - \eta) = 0 \quad \Rightarrow \quad \eta = \sum_i y_i /m
\]

The same can be done for the other 6 parameters. However, notice that they are not independent: \( \alpha_0 + \beta_0 + \gamma_0 = \alpha_1 + \beta_1 + \gamma_1 = 1 \) and also \( a_i + b_i + c_i = |D_i| = n \).
Consider data over 5 characters, x=a, b, c, d, e, and 2 states s=B, I.

We generate characters according to:
- Initial state prob: p(B)=0.5; p(I)=0.5
- State transition prob:
  \[ p(B \rightarrow B)=0.8 \quad p(B \rightarrow I)=0.2 \]
  \[ p(I \rightarrow B)=0.5 \quad p(I \rightarrow I)=0.5 \]
- Output prob:
  \[ p(a|B)=0.25, p(b|B)=0.10, p(c|B)=0.10, \ldots \]
  \[ p(a|I)=0.25, p(b|I)=0, \ldots \]

Can follow the generation process to get the observed sequence.

Other Examples (HMMs)
- We can do the same exercise we did before.
- Data: \{\{(x_1, x_2, \ldots, x_m, s_1, s_2, \ldots, s_m)\}\}_{1}^{n}
- Find the most likely parameters of the model:
  \[ P(x_i | s_i), P(s_{i+1} | s_i), p(s_1) \]
- Given an unlabeled example
  \[ x = (x_1, x_2, \ldots, x_m) \]
  use Bayes rule to predict the label \( \ell=(s_1, s_2, \ldots, s_m) \):
  \[ \ell^* = \arg \max \ell \; P(\ell | x) = \arg \max_i P(x | \ell) \; P(\ell) / P(x) \]
- The only issue is computational: there are \( 2^m \) possible values of \( \ell \)
- (This is an HMM model, but nothing was hidden)
Bayes Optimal Classifier

- How should we use the general formalism?
- What should $H$ be?

- **$H$ can be a collection of functions.** Given the training data, choose an optimal function. Then, given new data, evaluate the selected function on it.

- **$H$ can be a collection of possible predictions.** Given the data, try to directly choose the optimal prediction.

- Could be different!
The first formalism suggests to learn a good hypothesis and use it.
(Language modeling, grammar learning, etc. are here)

\[ h_{\text{MAP}} = \arg\max_{h \in H} P(h \mid D) = \arg\max_{h \in H} P(D \mid h) P(h) \]

The second one suggests to directly choose a decision.
This is the issue of “thresholding” vs. entertaining all options until the last minute. (Computational Issues)
Bayesian Learning

Bayes Optimal Classifier: Example

- Assume a space of 3 hypotheses:
  - \( P(h_1|D) = 0.4; P(h_2|D) = 0.3; P(h_3|D) = 0.3 \Rightarrow h_{MAP} = h_1 \)

- Given a new instance, assume that
  - \( h_1(x) = 1 \); \( h_2(x) = 0 \); \( h_3(x) = 0 \)

- In this case,
  - \( P(f(x) = 1) = 0.4 \); \( P(f(x) = 0) = 0.6 \) but \( h_{MAP}(x) = 1 \)

- We want to determine the most probable classification by combining the prediction of all hypotheses, weighted by their posterior probabilities.
Bayes Optimal Classifier: Example(2)

- Let \( V \) be a set of possible classifications

\[
P(v_j | D) = \sum_{h_i \in H} P(v_j | h_i, D)P(h_i | D) = \sum_{h_i \in H} P(v_j | h_i)P(h_i | D)
\]

- Bayes Optimal Classification:

\[
v = \arg\max_{v_j \in V} P(v_j | D) = \arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i)P(h_i | D)
\]

- In the example:

\[
P(1 | D) = \sum_{h_i \in H} P(1 | h_i)P(h_i | D) = 1 \cdot 0.4 + 0 \cdot 0.3 + 0 \cdot 0.3 = 0.4
\]

\[
P(0 | D) = \sum_{h_i \in H} P(0 | h_i)P(h_i | D) = 0 \cdot 0.4 + 1 \cdot 0.3 + 1 \cdot 0.3 = 0.6
\]

- and the optimal prediction is indeed 0.

- The key example of using a “Bayes optimal Classifier” is that of the naïve Bayes algorithm.
The Bayes optimal function is

\[ f_B(x) = \arg\max_y D(x; y) \]

That is, given input \( x \), return the most likely label.

It can be shown that \( f_B \) has the lowest possible value for \( \text{Err}(f) \).

Caveat: we can never construct this function: it is a function of \( D \), which is unknown.

But, it is a useful theoretical construct, and drives attempts to make assumptions on \( D \).
Maximum-Likelihood Estimates

- We attempt to model the underlying distribution
  \[ D(x, y) \text{ or } D(y \mid x) \]

- To do that, we assume a model
  \[ P(x, y \mid \theta) \text{ or } P(y \mid x, \theta), \]
  where \( \theta \) is the set of parameters of the model

- Example: Probabilistic Language Model (Markov Model):
  - We assume a model of language generation. Therefore, \( P(x, y \mid \theta) \) was written as a function of symbol & state probabilities (the parameters).

- We typically look at the log-likelihood

- Given training samples \( (x_i; y_i) \), maximize the log-likelihood

- \( L(\theta) = \sum_i \log P(x_i; y_i \mid \theta) \) or \( L(\theta) = \sum_i \log P(y_i \mid x_i, \theta) \)
Assumption: Our selection of the model is good; there is some parameter setting $\theta^*$ such that the true distribution is really represented by our model

$$D(x, y) = P(x, y | \theta^*)$$

Define the maximum-likelihood estimates:

$$\theta_{ML} = \arg\max_\theta L(\theta)$$

As the training sample size goes to $\infty$, then

$$P(x, y | \theta_{ML}) \text{ converges to } D(x, y)$$

Given the assumption above, and the availability of enough data

$$\arg\max_y P(x, y | \theta_{ML})$$

converges to the Bayes-optimal function

$$f_B(x) = \arg\max_y D(x; y)$$