CS 6140: Machine Learning
Spring 2017

Instructor: Lu Wang
College of Computer and Information Science
Northeastern University
Webpage: www.ccs.neu.edu/home/luwang
Email: luwang@ccs.neu.edu
Logistics

• Sign up at Piazza:
  – http://piazza.com/northeastern/spring2017/cs614002

• Assignment 1 is out
  – Due 2/9/2017
  – Start early!
Logistics

• Course project
  – 2 to 3 students for each team
Logistics

• Project Proposal
  – Due 1/26/2017
  – 1 page

• Content
  – Problem definition: what do you want to do?
  – Related work: put the work into context
  – Datasets: is there any data available or do you need to collect new data?
  – Evaluation metrics
  – Potential model and algorithms (optional)
What We Learned Last Week

• Basic Concept
  – Supervised learning vs. unsupervised learning
  – Parametric vs. non-parametric
  – Classification vs. regression
  – Overfitting vs. underfitting

• K-Nearest Neighbors

• Linear Regression

• Ridge Regression
A non-parametric classifier: K-nearest neighbors (KNN)
A non-parametric classifier: K-nearest neighbors (KNN)

• Basic idea: memorize all the training samples
  – The more you have in training data, the more the model has to remember

• K-Nearest neighbor:
  – Testing phase: find the K nearest neighbors, and return the majority vote of their labels
Problems of kNN

• Can be slow when training data is big
  – Searching for the neighbors takes time

• Needs lots of memory to store training data

• Needs to tune k and distance function

• Not a probability distribution
Probabilistic kNN

• We prefer a probabilistic output because sometimes we may get an “uncertain” result
  – 1 samples as “yes”, 199 samples as “no” \( \rightarrow ? \)
  – 99 samples as “yes”, 101 samples as “no” \( \rightarrow ? \)

• Probabilistic kNN:

\[
p(y|x, D) = \frac{1}{K} \sum_{j \in \text{nbr}(x, K, D)} I(y = y_j)
\]
Smoothing

• Class 1: 3, class 2: 0, class 3: 1

• Original probability:
  – $P(y=1) = \frac{3}{4}$, $p(y=2) = \frac{0}{4}$, $p(y=3) = \frac{1}{4}$

• Add-1 smoothing:
  – Class 1: $3+1$, class 2: $0+1$, class 3: $1+1$
  – $P(y=1) = \frac{4}{7}$, $p(y=2) = \frac{1}{7}$, $p(y=3) = \frac{2}{7}$
Softmax

- Class 1: 3, class 2: 0, class 3: 1
- Original probability:
  - \( P(y=1)=3/4, \ p(y=2)=0/4, \ p(y=3)=1/4 \)

- Redistribute probability mass into different classes
  - Define a softmax as

\[
S(x, \beta)_i = \frac{\exp(\beta x_i)}{\sum_j \exp(\beta x_j)}
\]
A parametric classifier: linear regression

• Assumption: the response is a linear function of the inputs

\[ y(x) = w^T x + \epsilon = \sum_{j=1}^{D} w_j x_j + \epsilon \]

- Inner product between input sample \( X \) and weight vector \( W \)
- Residual error: difference between prediction and true label
Learning with Maximum Likelihood Estimation (MLE)

• Log-likelihood

\[ \ell(\theta) \triangleq \log p(\mathcal{D}|\theta) = \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \theta) \]

• Maximize log-likelihood is equivalent to minimize negative log-likelihood (NLL)

\[ \text{NLL}(\theta) \triangleq -\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \theta) \]
Overfitting

Feature weights $\mathbf{w}$:

A Prior on the Weight

- Zero-mean Gaussian prior

\[ p(w) = \prod_j \mathcal{N}(w_j|0, \tau^2) \]

- New objective function

\[
\arg\max_w \sum_{i=1}^N \log \mathcal{N}(y_i|w_0 + w^T x_i, \sigma^2) + \sum_{j=1}^D \log \mathcal{N}(w_j|0, \tau^2)
\]
Ridge Regression

- We want to minimize

\[ J(w) = \frac{1}{N} \sum_{i=1}^{N} (y_i - (w_0 + w^T x_i))^2 + \lambda \|w\|_2^2 \]
Today’s Outline

• Evaluation metrics

• Decision Tree

• Generative Models

• Generative Model and Discriminative Model

• Logistic Regression
Evaluation Measures

• Accuracy

• Precision/recall/f-measure

• ROC

[Some slides are borrowed from Rich Caruana]
Accuracy

- Target: 0/1, -1/+1, True/False, ...
- Accuracy := #correct/#total prediction
Confusion Matrix

<table>
<thead>
<tr>
<th></th>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>True 1</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>True 0</td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

accuracy = \( \frac{a+d}{a+b+c+d} \)
# Prediction Threshold

<table>
<thead>
<tr>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True 1</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>True 0</strong></td>
<td>0</td>
</tr>
</tbody>
</table>

- threshold > MAX(f(x))
- all cases predicted 0
- (b+d) = total
- accuracy = %False = %0’s

<table>
<thead>
<tr>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True 1</strong></td>
<td>a</td>
</tr>
<tr>
<td><strong>True 0</strong></td>
<td>c</td>
</tr>
</tbody>
</table>

- threshold < MIN(f(x))
- all cases predicted 1
- (a+c) = total
- accuracy = %True = %1’s
Problems with Accuracy

• Assumes equal cost for both kinds of errors
  – Medical domain: cold v.s. cancer

• is 99% accuracy good?

• is 10% accuracy bad?

• BaseRate = accuracy of predicting predominant class
Problems with Accuracy

• Assumes equal cost for both kinds of errors
  – Medical domain: cold v.s. cancer

• is 99% accuracy good?
  – can be excellent, good, mediocre, poor, terrible
• is 10% accuracy bad?
  – information retrieval

• BaseRate = accuracy of predicting predominant class
Percent Reduction in Error

- 80% accuracy = 20% error
- Suppose learning increases accuracy from 80% to 90%
- Error reduced from 20% to 10%
- 50% reduction in error

- 99.90% to 99.99% = 90% reduction in error
- 50% to 75% = 50% reduction in error
- Can be applied to many other measures
Precision and Recall

• typically used in document retrieval

• Precision:
  – how many of the returned documents are correct
  – precision(threshold)

• Recall:
  – how many of the positives does the model return
  – recall(threshold)

• Precision/Recall Curve: sweep thresholds
Precision/Recall

<table>
<thead>
<tr>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>True 1</td>
<td>a</td>
</tr>
<tr>
<td>True 0</td>
<td>c</td>
</tr>
</tbody>
</table>

\[
\text{PRECISION} = \frac{a}{a + c} \\
\text{RECALL} = \frac{a}{a + b}
\]
\[ \text{PRECISION} = \frac{a}{a + c} \]

\[ \text{RECALL} = \frac{a}{a + b} \]

\[ F = \frac{2 \times (\text{PRECISION} \times \text{RECALL})}{(\text{PRECISION} + \text{RECALL})} \]

harmonic average of precision and recall
ROC Plot and ROC Area

- Receiver Operator Characteristic
- Developed in WWII to statistically model false positive and false negative detections of radar operators
- Better statistical foundations than most other measures
- Standard measure in medicine and biology
- Becoming more popular in ML
ROC Plot

• Sensitivity = $a/(a+b) = \text{Recall}$
  – True positive rate
• $1 - \text{Specificity} = 1 - d/(c+d)=c/(c+d)$
  – False positive rate
Sensitivity = True Positive Rate = P(pred_true|true)

ROC Area = 0.9049

diagonal line is random prediction
Properties of ROC

- Slope is non-increasing
- Each point on ROC represents different tradeoff (cost ratio) between false positives and false negatives
- Slope of line tangent to curve defines the cost ratio
- ROC Area represents performance averaged over all possible cost ratios
- If two ROC curves do not intersect, one method dominates the other
- If two ROC curves intersect, one method is better for some cost ratios, and other method is better for other cost ratios
Properties of ROC

- ROC Area:
  - 1.0: perfect prediction
  - 0.9: excellent prediction
  - 0.8: good prediction
  - 0.7: mediocre prediction
  - 0.6: poor prediction
  - 0.5: random prediction
  - <0.5: something wrong!
Summary

• the measure you optimize to makes a difference
• the measure you report makes a difference
• use measure appropriate for problem/community
• accuracy often is not sufficient/appropriate
• ROC is gaining popularity in the ML community
• only accuracy generalizes to $>2$ classes!
Today’s Outline

• Evaluation metrics
• Decision Tree
• Generative Models
• Generative Model and Discriminative Model
• Logistic Regression
Decision Tree

[some of the slides are borrowed from Tom Mitchell’s lecture and David Sontag’s lecture]
Decision Tree

• Play tennis?
Decision Tree

- Play tennis?

- Each internal node: test one feature $X_i$
- Each branch from a node: selects one value for $X_i$
- Each leaf node: predict $Y$ (or $P(Y|X \in \text{leaf})$)
Top-Down Induction of Decision Trees

$node = \text{Root}$

Main loop:

1. $A \leftarrow$ the “best” decision attribute for next $node$
2. Assign $A$ as decision attribute for $node$
3. For each value of $A$, create new descendant of $node$
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes
Top-Down Induction of Decision Trees

• Which attribute to use for split?

Would we prefer to split on $X_1$ or $X_2$?

<table>
<thead>
<tr>
<th></th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Top-Down Induction of Decision Trees

• Which attribute to use for split?
• Good split if we are more certain about classification after split
  – Deterministic good (all true or all false)
  – Uniform distribution bad
  – What about distribution in between?
Entropy

• Entropy $H(X)$ of a random variable $X$

$$H(X) = - \sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

• More uncertainty, more entropy!

• From information theory: $H(X)$ is the expected number of bits needed to encode a randomly drawn value of $X$ (under most efficient code)
Entropy

$S$ is a sample of training examples

$p_\oplus$ is the proportion of positive examples in $S$

$p_\ominus$ is the proportion of negative examples in $S$

Entropy measures the impurity of $S$

$$Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$
Information Gain

• Gain(S,A)=expected reduction in entropy due to sorting on A

\[ Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \]
<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Information Gain

\[
Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)
\]

\[
S: [9+, 5-] \\
E = 0.940
\]

\[
\text{Humidity} \\
\text{High} \rightarrow [3+, 4-] \\
E = 0.985 \\
\text{Normal} \rightarrow [6+, 1-] \\
E = 0.592
\]

\[
\text{Gain} (S, \text{Humidity}) \\
= .940 - (7/14).985 - (7/14).592 \\
= .151
\]

\[
S: [9+, 5-] \\
E = 0.940
\]

\[
\text{Wind} \\
\text{Weak} \rightarrow [6+, 2-] \\
E = 0.811 \\
\text{Strong} \rightarrow [3+, 3+] \\
E = 1.00
\]

\[
\text{Gain} (S, \text{Wind}) \\
= .940 - (8/14).811 - (6/14)1.0 \\
= .048
\]
$S_{\text{Sunny}} = \{D_1, D_2, D_8, D_9, D_{11}\}$

$\text{Gain} (S_{\text{Sunny}}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$

$\text{Gain} (S_{\text{Sunny}}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$

$\text{Gain} (S_{\text{Sunny}}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$
Avoid Overfitting

• Stop growing when data split is not statistically significant
Avoid Overfitting

• Stop growing when data split is not statistically significant

• Grow a full-tree, then prune
Reduce-Error Pruning

Split data into training and validation set

Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (plus those below it)

2. Greedily remove the one that most improves validation set accuracy
Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use
IF  
\((Outlook = Sunny) \land (Humidity = High)\)
THEN  \(PlayTennis = No\)

IF  
\((Outlook = Sunny) \land (Humidity = Normal)\)
THEN  \(PlayTennis = Yes\)
Today’s Outline

• Evaluation metrics

• Decision Tree

• Generative Models

• Generative Model and Discriminative Model

• Logistic Regression
Bayesian Concept Learning

• How do human beings learn from everyday life?
  – Meanings of words
  – Causes of a person’s action
  – Future outcomes of a dynamic process
  – ....
Number Game

- Observe one or more examples
- Judge whether other numbers are yes or no

1 random "yes" example:

4 random "yes" examples:
Hypothesis space

• Mathematical properties
  – odd, even, square, cube, prime, ...
  – multiples of small integers
  – powers of small integers
  – same first (or last) digit
## Number Game

<table>
<thead>
<tr>
<th>Examples of “yes” numbers</th>
<th>Hypotheses</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>multiples of 10</td>
</tr>
<tr>
<td></td>
<td>even numbers</td>
</tr>
<tr>
<td></td>
<td>???</td>
</tr>
<tr>
<td>60 80 10 30</td>
<td>multiples of 10</td>
</tr>
<tr>
<td></td>
<td>even numbers</td>
</tr>
<tr>
<td>60 63 56 59</td>
<td>numbers “near” 60</td>
</tr>
</tbody>
</table>
Generalization from positive samples
Bayesian model

- H: Hypothesis space of possible concepts
- X: n examples of a concept C
- Evaluate hypotheses given data using Bayes’ rule:

\[ p(h \mid X) = \frac{p(X \mid h) p(h)}{\sum_{h' \in H} p(X \mid h') p(h')} \]

- \( p(h) \) [“prior”]: domain knowledge, pre-existing biases
- \( p(X \mid h) \) [“likelihood”]: statistical information in examples.
- \( p(h \mid X) \) [“posterior”]: degree of belief that \( h \) is the true extension of \( C \).
Likelihood

- Size principle: Smaller hypotheses receive greater likelihood, and exponentially more so as $n$ increases.

$$p(D|h) = \left[ \frac{1}{\text{size}(h)} \right]^N = \left[ \frac{1}{|h|} \right]^N$$
Likelihood

\[ p(D|h) = \left[ \frac{1}{\text{size}(h)} \right]^N = \left[ \frac{1}{|h|} \right]^N \]

- Occam’s razor
  - The model favors the simplest or smallest hypothesis consistent with the data
- \( D=\{16\} \)
- \( h_1: \) powers of two under 100
- \( h_2: \) event numbers under 100
- \( P(D|h_1)=1/6 \)
- \( P(D|h_2)=1/50 \)
Prior

- $X = \{60, 80, 10, 30\}$
Prior

• X={60,80,10,30}
  – Why prefer “multiples of 10” over “even numbers”? 
  – Why prefer “multiples of 10” over “multiples of 10 except 50 and 20”? 
  – Cannot learn efficiently if we have a uniform prior over all $2^{100}$ logically possible hypotheses
\[
p(h | \mathcal{D}) = \frac{p(\mathcal{D} | h)p(h)}{\sum_{h' \in \mathcal{H}} p(\mathcal{D}, h')} = \frac{p(h) \mathbb{I}(\mathcal{D} \in h) / |h|^N}{\sum_{h' \in \mathcal{H}} p(h') \mathbb{I}(\mathcal{D} \in h') / |h'|^N}
\]
Posterior predictive distribution

• Bayesian model averaging

\[ p(\tilde{x} \in C|\mathcal{D}) = \sum_h p(y = 1|\tilde{x}, h)p(h|\mathcal{D}) \]
Posterior predictive distribution

• Maximum a posteriori (MAP)
  – Or plug-in approximation

\[ p(\tilde{x} \in C|\mathcal{D}) = \sum_h p(\tilde{x}|h) \delta_{\hat{h}}(h) = p(\tilde{x}|\hat{h}) \]
Today’s Outline

• Evaluation metrics
• Decision Tree
• Generative Models
• Generative Model and Discriminative Model
• Logistic Regression
Naïve Bayes

- \( P(\text{Heads}) = \theta, \ P(\text{Tails}) = 1-\theta \)

- Flips are \textit{i.i.d.}: \( D = \{x_i \mid i=1...n\}, \ P(D \mid \theta) = \Pi_i P(x_i \mid \theta) \)
  - Independent events
  - Identically distributed according to Bernoulli distribution

- Sequence \( D \) of \( \alpha_H \) Heads and \( \alpha_T \) Tails

\[
P(D \mid \theta) = \theta^{\alpha_H} (1 - \theta)^{\alpha_T}
\]

Called the “likelihood” of the data under the model
Parameter Estimation: Maximum Likelihood Estimation

- **Data:** Observed set $D$ of $\alpha_H$ Heads and $\alpha_T$ Tails
- **Hypothesis:** Bernoulli distribution
- **Learning:** finding $\theta$ is an optimization problem
  - What’s the objective function?
    \[ P(D \mid \theta) = \theta^{\alpha_H} (1 - \theta)^{\alpha_T} \]
- **MLE:** Choose $\theta$ to maximize probability of $D$
  \[
  \hat{\theta} = \underset{\theta}{\arg \max} \ P(D \mid \theta) \\
  = \underset{\theta}{\arg \max} \ \ln P(D \mid \theta)
  \]
\[ \hat{\theta} = \arg \max_\theta \ln P(\mathcal{D} \mid \theta) \]
\[ = \arg \max_\theta \ln \theta^{\alpha_H} (1 - \theta)^{\alpha_T} \]

- Set derivative to zero, and solve!

\[ \frac{d}{d\theta} \ln P(\mathcal{D} \mid \theta) = \frac{d}{d\theta} [\ln \theta^{\alpha_H} (1 - \theta)^{\alpha_T}] \]
\[ = \frac{d}{d\theta} [\alpha_H \ln \theta + \alpha_T \ln(1 - \theta)] \]
\[ = \alpha_H \frac{d}{d\theta} \ln \theta + \alpha_T \frac{d}{d\theta} \ln(1 - \theta) \]
\[ = \frac{\alpha_H}{\theta} - \frac{\alpha_T}{1 - \theta} = 0 \]

\[ \hat{\theta}_{MLE} = \frac{\alpha_H}{\alpha_H + \alpha_T} \]
Data

\[ L(\theta; \mathcal{D}) = \ln P(\mathcal{D}|\theta) \]

\[ \hat{\theta}_{MLE} = \frac{\alpha_H}{\alpha_H + \alpha_T} \]
Adding Prior

- Rather than estimating a single $\theta$, we obtain a distribution over possible values of $\theta$.

**In the beginning**

\[
\Pr(\theta) \sim \text{Beta}(2, 2)
\]

**After observations**

\[
\Pr(\theta \mid D) \sim \text{Beta}(3, 2)
\]

Observe flips e.g.: \{tails, tails\}
• Use Bayes’ rule!

\[ P(\theta | \mathcal{D}) = \frac{P(\mathcal{D} | \theta)P(\theta)}{P(\mathcal{D})} \]

Prior

Data Likelihood

Normalization

• Or equivalently:

\[ P(\theta | \mathcal{D}) \propto P(\mathcal{D} | \theta)P(\theta) \]

• For uniform priors, this reduces to maximum likelihood estimation!

\[ P(\theta) \propto 1 \quad P(\theta | \mathcal{D}) \propto P(\mathcal{D} | \theta) \]
\[ P(\theta \mid D) \propto P(D \mid \theta)P(\theta) \]

Likelihood: \[ P(D \mid \theta) = \theta^{\alpha_H}(1 - \theta)^{\alpha_T} \]

- What should the prior be?
  - Represent expert knowledge
  - Simple posterior form

- For binary variables, commonly used prior is the Beta distribution:

\[ P(\theta) = \frac{\theta^{\beta_H-1}(1 - \theta)^{\beta_T-1}}{B(\beta_H, \beta_T)} \sim \text{Beta}(\beta_H, \beta_T) \]
Beta Prior Distribution

\[ P(\theta) = \frac{\theta^{\beta_H-1}(1 - \theta)^{\beta_T-1}}{B(\beta_H, \beta_T)} \sim Beta(\beta_H, \beta_T) \]

- Since the Beta distribution is *conjugate* to the Bernoulli distribution, the posterior distribution has a particularly simple form:

\[
P(\theta \mid D) \propto P(D \mid \theta)P(\theta) \\
\propto \theta^{\alpha_H} (1 - \theta)^{\alpha_T} \theta^{\beta_H-1}(1 - \theta)^{\beta_T-1} \\
= \theta^{\alpha_H + \beta_H - 1} (1 - \theta)^{\alpha_T + \beta_T - 1} \\
= Beta(\alpha_H + \beta_H, \alpha_T + \beta_T)
\]
Bayesian Inference for Prediction

• We now have a **distribution** over parameters

• For any specific $f$, a function of interest, compute the expected value of $f$:

$$E[f(\theta)] = \int_0^1 f(\theta) P(\theta | D) d\theta$$

• Integral is often hard to compute

• *As more data is observed, posterior is more concentrated*

• MAP (Maximum a posteriori approximation): use most likely parameter to approximate the expectation

$$\hat{\theta} = \text{arg max}_\theta P(\theta | D)$$

$$E[f(\theta)] \approx f(\hat{\theta})$$
Bayesian Classification

- Problem statement:
  - Given features $X_1, X_2, ..., X_n$
  - Predict a label $Y$
Example Application

- Digit Recognition

\[ X \rightarrow \text{Classifier} \rightarrow Y \]

- \( X_1, \ldots, X_n \in \{0,1\} \) (Black vs. White pixels)
- \( Y \in \{0,1,2,3,4,5,6,7,8,9\} \)
The Bayes Classifier

• If we had the joint distribution on $X_1, \ldots, X_n$ and $Y$, could predict using:

$$\arg \max_Y P(Y|X_1, \ldots, X_n)$$

  (for example: what is the probability that the image represents a 5 given its pixels?)

• So ... How do we compute that?
The Bayes Classifier

- Use Bayes Rule!

\[ P(Y | X_1, \ldots, X_n) = \frac{P(X_1, \ldots, X_n | Y)P(Y)}{P(X_1, \ldots, X_n)} \]

Likelihood \quad Prior

Normalization Constant

- Why did this help? Well, we think that we might be able to specify how features are “generated” by the class label.
The Bayes Classifier

- Let’s expand this for our digit recognition task:

\[
P(Y = 5 | X_1, \ldots, X_n) = \frac{P(X_1, \ldots, X_n | Y = 5) P(Y = 5)}{P(X_1, \ldots, X_n | Y = 5) P(Y = 5) + P(X_1, \ldots, X_n | Y = 6) P(Y = 6)}
\]

\[
P(Y = 6 | X_1, \ldots, X_n) = \frac{P(X_1, \ldots, X_n | Y = 6) P(Y = 6)}{P(X_1, \ldots, X_n | Y = 5) P(Y = 5) + P(X_1, \ldots, X_n | Y = 6) P(Y = 6)}
\]

- To classify, we’ll simply compute these probabilities, one per class, and predict based on which one is largest.
Model Parameters

• How many parameters are required to specify the likelihood, $P(X_1,\ldots,X_n \mid Y)$?
  - (Supposing that each image is 30x30 pixels)

• The problem with explicitly modeling $P(X_1,\ldots,X_n \mid Y)$ is that there are usually way too many parameters:
  - We’ll run out of space
  - We’ll run out of time
  - And we’ll need tons of training data (which is usually not available)
• Naïve Bayes assumption:
  – Features are independent given class:

\[
P(X_1, X_2|Y) = P(X_1|X_2, Y)P(X_2|Y) = P(X_1|Y)P(X_2|Y)
\]

  – More generally:

\[
P(X_1 \ldots X_n|Y) = \prod_{i} P(X_i|Y)
\]

• How many parameters now?
  • Suppose \( X \) is composed of \( n \) binary features
Naïve Bayes

• **Given:**
  - Prior $P(Y)$
  - $n$ conditionally independent features $X_1, ..., X_n$, given the class $Y$
  - For each feature $i$, we specify $P(X_i | Y)$

• **Classification decision rule:**
  
  $y^* = h_{NB}(x) = \arg \max_y P(y)P(x_1, \ldots, x_n | y)$
  
  $= \arg \max_y P(y) \prod_i P(x_i | y)$

If certain assumption holds, NB is optimal classifier!
(they typically don’t)
Digit Recognition

- **Input:** pixel grids

- **Output:** a digit 0-9

Are the naïve Bayes assumptions realistic here?
What to be Learned

$P(Y)$

$P(F_{3,1} = on|Y)$

$P(F_{5,5} = on|Y)$
Maximum Likelihood Estimation

- Given dataset
  - $\text{Count}(A=a,B=b)$ ← number of examples where $A=a$ and $B=b$

- MLE for discrete NB, simply:
  - Prior:
    $$P(Y = y) = \frac{\text{Count}(Y = y)}{\sum_{y'} \text{Count}(Y = y')}$$
  - Observation distribution:
    $$P(X_i = x|Y = y) = \frac{\text{Count}(X_i = x, Y = y)}{\sum_{x'} \text{Count}(X_i = x', Y = y)}$$
Maximum Likelihood Estimation

- Training amounts to, for each of the classes, averaging all of the examples together:
Maximum Likelihood Estimation

• Given dataset
  – Count(A=a, B=b) ← number of examples where A=a and B=b

• MAP estimation for discrete NB, simply:
  – Prior:
    \[ P(Y = y) = \frac{\text{Count}(Y = y)}{\sum_{y'} \text{Count}(Y = y')} \]
  – Observation distribution:
    \[ P(X_i = x | Y = y) = \frac{\text{Count}(X_i = x, Y = y)}{\sum_{x'} \text{Count}(X_i = x', Y = y)} \frac{+ a}{+ |X_i| * a} \]

• Called “smoothing”. Corresponds to Dirichlet prior!
Today’s Outline

• Evaluation metrics

• Decision Tree

• Generative Models

• Generative Model and Discriminative Model

• Logistic Regression
Generative VS. Discriminative Model

• $P(Y|X) = p(X,Y)/P(X)$

• Generative model
  – Learn $P(X, Y)$ from training sample
  – $P(X, Y) = P(Y)P(X|Y)$
  – Specifies how to generate the observed features $x$ for $y$

• Discriminative model
  – Learn $P(Y|X)$ from training sample
  – Directly models the mapping from features $x$ to $y$
Today’s Outline

• Evaluation metrics

• Decision Tree

• Generative Models

• Generative Model and Discriminative Model

• Logistic Regression
Logistic Regression

Learn \( P(Y|X) \) directly!

- Assume a particular functional form
- Sigmoid applied to a linear function of the data:

\[
P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}
\]

\[
P(Y = 0|X) = \frac{\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}
\]
Sigmoid function

• Definition

\[ \text{sigm}(\eta) \triangleq \frac{1}{1 + \exp(-\eta)} = \frac{e^\eta}{e^\eta + 1} \]
Logistic Regression

\[ P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)} \]

\[ P(Y = 0|X) = \frac{\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)} \]

- **Prediction:** Output the Y with highest \( P(Y|X) \)
  - For binary Y, output \( Y=0 \) if

\[ 1 < \frac{P(Y = 0|X)}{P(Y = 1|X)} \]

\[ 1 < \exp(w_0 + \sum_{i=1}^{n} w_i X_i) \]

\[ 0 < w_0 + \sum_{i=1}^{n} w_i X_i \]

A Linear Classifier!
Maximizing Log Likelihood

\[ l(w) \equiv \ln \prod_j P(y^j | x^j, w) \]

\[ = \sum_j y^j (w_0 + \sum_i^n w_i x_i^j) - \ln (1 + \exp (w_0 + \sum_i^n w_i x_i^j)) \]

0 or 1!

**Bad news:** no closed-form solution to maximize \( l(w) \)

**Good news:** \( l(w) \) is concave function of \( w \) →

No local maxima

Concave functions easy to optimize
Conditional likelihood for Logistic Regression is concave →

**Gradient:**
\[ \nabla_w l(w) = \left[ \frac{\partial l(w)}{\partial w_0}, \ldots, \frac{\partial l(w)}{\partial w_n} \right]^T \]

**Update rule:**
\[ \Delta w = \eta \nabla_w l(w) \]
\[ w_i(t+1) \leftarrow w_i(t) + \eta \frac{\partial l(w)}{\partial w_i} \]
\[ l(w) = \sum_j y^j (w_0 + \sum_{i=1}^{n} w_i x_i^j) - \ln(1 + \exp(w_0 + \sum_{i=1}^{n} w_i x_i^j)) \]

\[
\frac{\partial l(w)}{\partial w_i} = \sum_j \left[ \frac{\partial}{\partial w_i} y^j (w_0 + \sum_{i} w_i x_i^j) - \frac{\partial}{\partial w_i} \ln \left( 1 + \exp(w_0 + \sum_{i} w_i x_i^j) \right) \right]
\]

\[
= \sum_j \left[ y^j x_i^j - \frac{x_i^j \exp(w_0 + \sum_i w_i x_i^j)}{1 + \exp(w_0 + \sum_i w_i x_i^j)} \right]
\]

\[
= \sum_j x_i^j \left[ y^j - \frac{\exp(w_0 + \sum_i w_i x_i^j)}{1 + \exp(w_0 + \sum_i w_i x_i^j)} \right]
\]

\[
\frac{\partial l(w)}{\partial w_i} = \sum_j x_i^j \left( y^j - P(Y^j = 1|x^j, w) \right)
\]
Gradient ascent algorithm: (learning rate $\eta > 0$)

do:

$$w^{(t+1)}_0 \leftarrow w^{(t)}_0 + \eta \sum_j [y^j - \hat{P}(Y^j = 1 \mid x^j, w)]$$

For $i=1$ to $n$: (iterate over features)

$$w^{(t+1)}_i \leftarrow w^{(t)}_i + \eta \sum_j x^j_i [y^j - \hat{P}(Y^j = 1 \mid x^j, w)]$$

until “change” < $\varepsilon$

Loop over training examples (could also do stochastic GD)
Gradient Descent

• Example

\[ f(\theta) = 0.5(\theta_1^2 - \theta_2)^2 + 0.5(\theta_1 - 1)^2 \]

\[ \eta = 0.1 \]

\[ \eta = 0.6 \]
Changing Step Size
Adding Prior

\[ p(w \mid Y, X) \propto P(Y \mid X, w)p(w) \]

• One common approach is to define priors on \( w \)
  – Normal distribution, zero mean, identity covariance
  – “Pushes” parameters towards zero
    \[ p(w) = \prod_i \frac{1}{\kappa \sqrt{2\pi}} \frac{-w_i^2}{e^{2\kappa^2}} \]

• **Regularization**
  – Helps avoid very large weights and overfitting

• **MAP estimate:**
  \[ w^* = \arg \max_w \ln \left[ p(w) \prod_{j=1}^N P(y^j \mid x^j, w) \right] \]
\[ w^* = \arg \max_w \ln \left[ p(w) \prod_{j=1}^{N} P(y^j | x^j, w) \right] \]

\[ p(w) = \prod_i \frac{1}{\kappa \sqrt{2\pi}} \frac{-w_i^2}{e^{2\kappa^2}} \]

- Adds \( \log p(w) \) to objective:

\[ \ln p(w) \propto -\frac{\lambda}{2} \sum_i w_i^2 \]

\[ \frac{\partial \ln p(w)}{\partial w_i} = -\lambda w_i \]

- Quadratic penalty: drives weights towards zero
- Adds a negative linear term to the gradients
Generative VS. Discriminative Model

• Generative model
  – Learn $P(X, Y)$ from training sample
  – $P(X, Y) = P(Y)P(X|Y)$
  – Specifies how to generate the observed features $x$ for $y$

• Discriminative model
  – Learn $P(Y|X)$ from training sample
  – Directly models the mapping from features $x$ to $y$
Generative VS. Discriminative Model

- Easy to fit the model
  - Generative model: Sometimes counting is good enough
  - Discriminative model: optimization
Generative VS. Discriminative Model

• Fit classes separately
  – Generative model: train separately for different classes
  – Discriminative model: parameters interact
Generative VS. Discriminative Model

• Symmetric in inputs and outputs
  – Generative model: we’ve modeled \( p(x,y) \)
  – Discriminative model: just \( p(y|x) \)
Generative VS. Discriminative Model

• Handle feature preprocessing
  – Generative model: hard to generalize
  – Discriminative model: replace x with other forms
Generative VS. Discriminative Model

• Handle missing values
  – Generative model: easy to do
  – Discriminative model: unclear
Generative VS. Discriminative Model

• Handle missing values
  – Generative model: easy to do
  – Discriminative model: unclear

Naïve Bayes problem in assignment 1!
What We Learned Today

• Evaluation Metrics

• Decision Tree

• Generative Models

• Generative Model and Discriminative Model

• Logistic Regression
Homework

• Reading Murphy Ch 3, 8.1-8.3, 8.6
• Project proposal due next Thursday.
• First assignment is out.