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” \( \rightarrow \) ?

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

Smoothing

- Class 1: 3, class 2: 0, class 3: 1
- Original probability:
  - \( P(y=1)=3/4, \ p(y=2)=0/4, \ p(y=3)=1/4 \)
- Add-1 smoothing:
  - Class 1: 3+1, class 2: 0+1, class 3: 1+1
  - \( P(y=1)=4/7, \ p(y=2)=1/7, \ p(y=3)=2/7 \)

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

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

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

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

Overfitting

- Feature weights 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

Accuracy

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

Confusion Matrix

\[
\begin{array}{c|c|c}
\text{Predicted 1} & \text{Predicted 0} \\
\hline
\text{True 1} & a & b \\
\hline
\text{True 0} & c & d \\
\end{array}
\]

\[
\text{accuracy} = \frac{a+d}{a+b+c+d}
\]

Prediction Threshold

- \( \text{threshold} > \text{MAX}(f(x)) \)
- \( \text{all cases predicted 0} \)
- \( \text{accuracy} = \%\text{False} = \%0's \)

- \( \text{threshold} < \text{MIN}(f(x)) \)
- \( \text{all cases predicted 1} \)
- \( \text{accuracy} = \%\text{True} = \%1's \)

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

(Some slides are borrowed from Rich Caruana)
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

\[
\text{Precision} = \frac{a}{a + c} \\
\text{Recall} = \frac{a}{a + b}
\]
**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

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

**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?

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

Top-Down Induction of Decision Trees

```
node = 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>$X_1$</th>
<th>$X_2$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$Y = 4$</th>
<th>$Y = 1$</th>
<th>$Y = 3$</th>
<th>$Y = 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</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

- Gain($S$, $A$) = expected reduction in entropy due to sorting on $A$
  $$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

Information Gain

Day | Outlook | Temperature | Humidity | Wind | PlayTennis
--- | --- | --- | --- | --- | ---
D1 | Sunny | Hot | High | Weak | No
D2 | Sunny | Hot | High | Strong | No
D3 | Overcast | Hot | High | Weak | Yes
D4 | Rain | Mild | High | Weak | Yes
D5 | Rain | Cool | Normal | Weak | Yes
D6 | Rain | Cool | Normal | Strong | No
D7 | Overcast | Cool | Normal | Strong | Yes
D8 | Sunny | Mild | High | Weak | No
D9 | Sunny | Cool | Normal | Weak | Yes
D10 | Rain | Mild | Normal | Weak | Yes
D11 | Sunny | Mild | Normal | Strong | Yes
D12 | Overcast | Mild | High | Strong | Yes
D13 | Overcast | Hot | Normal | Weak | Yes
D14 | Rain | Mild | High | Strong | No
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

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
  - …

Some of the slides are borrowed from Kevin Murphy’s Lectures

Number Game

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

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 even numbers</td>
</tr>
<tr>
<td>60 80 10 30</td>
<td>multiples of 10 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
  - \( p(X|h) \) ["likelihood"]: statistical information in examples.
  - \( p(h|X) \) ["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.

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

- Occam’s razor
  - The model favors the simplest or smallest hypothesis consistent with the data
  - \( D=\{16\} \)
  - \( h_1 \): powers of two under 100
  - \( h_2 \): even 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

Posterior

\[ p(h|D) = \frac{p(D|h)p(h)}{\sum_{h' \in H} p(D|h')p(h')} = \frac{p(h) \prod_{i=1}^{N} p(x_i|D \in h)}{\sum_{h' \in H} \prod_{i=1}^{N} p(x_i|D \in h')} \]

Posterior predictive distribution

- Bayesian model averaging

\[ p(\hat{y} \in C|D) = \sum_{h} p(y = 1|\hat{y}, h)p(h|D) \]
Posterior predictive distribution

• Maximum a posteriori (MAP)
  – Or plug-in approximation
  \[ p(\hat{z} \in C | \mathcal{D}) = \sum_{h} p(\hat{z} | h) \delta_h(h) = p(\hat{z} | h) \]

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 i.i.d.: \( D = \{x_i \}_{i=1}^{n} \), \( P(\mathcal{D} | \theta) = \Pi P(x_i | \theta) \)
  – Independent events
  – Identically distributed according to Bernoulli distribution
• Sequence \( D \) of \( \alpha_H \) Heads and \( \alpha_T \) Tails
  \[ P(\mathcal{D} | \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T} \]

Called the “likelihood” of the data under the model

Parameter Estimation: Maximum Likelihood Estimation

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

\[ \hat{\theta} = \arg \max_{\theta} \ln P(\mathcal{D} | \theta) \]

\[ \hat{\theta} = \arg \max_{\theta} \ln \theta^{\alpha_H}(1-\theta)^{\alpha_T} \]

• Set derivative to zero, and solve!
  \[ \frac{d}{d\theta} \ln P(\mathcal{D} | \theta) = \frac{d}{d\theta} [\ln \theta^{\alpha_H}(1-\theta)^{\alpha_T}] \]
  \[ = \frac{d}{d\theta} [\alpha_H \ln \theta + \alpha_T \ln(1-\theta)] \]
  \[ = \alpha_H \frac{d}{d\theta} \ln \theta + \alpha_T \frac{d}{d\theta} \ln(1-\theta) \]
  \[ = \frac{\alpha_H}{\theta} - \frac{\alpha_T}{1-\theta} = 0 \]

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

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

In the beginning

- Observe flips e.g. (tails, tails)

After observations

\[
P(\theta \mid D) \propto P(D \mid \theta) P(\theta)
\]

Likelihood:

\[
P(D \mid \theta) = \theta^{x_1}(1 - \theta)^{y_1}\]

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

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

\[
P(\theta) = \frac{\theta^{y_1-1}(1 - \theta)^{y_2-1}}{B(y_1, y_2)} \sim \text{Beta}(y_1, y_2)
\]

Bayesian Inference for Prediction

- We now have a distribution over parameters.
- For any function $f$, compute the expected value of $f$:

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

- 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

\[
\hat{\theta} = \arg \max_{\theta} P(\theta \mid D)
\]

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

Bayesian Classification

- Problem statement:
  - Given features $X_1, X_2, \ldots, X_n$
  - Predict a label $Y$

Beta Prior Distribution

\[
P(\theta) = \frac{\theta^{y_1-1}(1 - \theta)^{y_2-1}}{B(y_1, y_2)} \sim \text{Beta}(y_1, y_2)
\]

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

\[
P(\theta \mid D) \propto P(D \mid \theta) P(\theta)
\]

\[
\propto \theta^{x_1+y_1-1}(1 - \theta)^{y_2+y_2-1}
\]

\[
= \frac{\theta^{x_1+y_1-1}}{(1 - \theta)^{y_2+y_2-1}}
\]

\[
= \text{Beta}(\alpha_{y_1}, \beta_{y_2})
\]
Example Application

- Digit Recognition
  - $x_{i,j} \in \{0,1\}$ (Black vs. White pixels)
  - $y \in \{0,1,2,3,4,5,6,7,8,9\}$

$\text{Classifier \,\, 5}$

The Bayes Classifier

- If we had the joint distribution on $x_{i,j}, \ldots, x_n$ 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:
  $$P(Y|X_1, \ldots, X_n) = \frac{P(X_1, \ldots, X_n|Y)P(Y)}{P(X_1, \ldots, X_n)}$$
- Why did this help? Well, we think that we might be able to specify how features are "generated" by the class label

Model Parameters

- How many parameters are required to specify the likelihood, $P(X_1, \ldots, X_n|y)$?
  - (Supposing that each image is 30x30 pixels)
- The problem with explicitly modeling $P(X_1, \ldots, X_n|y)$ 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)

- Naive 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 \( X_1 \ldots X_n \), given the class \( Y \)
  - For each feature \( i \), we specify \( P(X_i|Y) \)

- Classification decision rule:
  \[
  y^* = h_{Naive}(x) = \arg \max_y P(y) P(x_1, \ldots, x_n | y) = \arg \max_y P(y) \prod_i P(x_i|y)
  \]

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

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
  - \( \text{Count}(A=a, B=b) \) ← 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:
    \[
    P(X_i = x | Y = y) = \frac{\text{Count}(X_i = x, Y = y) + a}{\sum_{x'} \text{Count}(X_i = x', Y = y) + |X_i| a}
    \]
- 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

• \( P(Y|X) = \frac{P(X,Y)}{P(X)} \)

• 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 \( y \)

Logistic Regression

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

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

\[
P(Y = 1|X) = \frac{1}{1 + \exp(-\eta)}
\]

\[
P(Y = 0|X) = \frac{\exp(\eta)}{1 + \exp(\eta)}
\]

Sigmoid function

• Definition

\[
sign(\eta) \triangleq \frac{1}{1 + \exp(-\eta)} = \frac{e^\eta}{e^\eta + 1}
\]

Logistic Regression
Logistic Regression

\[ P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_j w_j x_j)} \]
\[ P(Y = 0|X) = \frac{\exp(w_0 + \sum_j w_j x_j)}{1 + \exp(w_0 + \sum_j w_j x_j)} \]

- **Prediction**: Output the Y with highest \( P(Y|X) \)
  - For binary Y, output \( Y = 0 \) if
    \[ 1 < \exp(w_0 + \sum_j w_j x_j) \]
    \[ 0 < w_0 + \sum_j w_j x_j \]
  - For binary Y, output \( Y = 1 \) if
    \[ 0 < \exp(w_0 + \sum_j w_j x_j) \]
    \[ 1 < w_0 + \sum_j w_j x_j \]

A Linear Classifier!

Maximizing Log Likelihood

\[ l(w) = \ln \prod_j P(y^j|\xi^j, w) \]
\[ = \sum_j y^j (w_0 + \sum_i w_i x_i^j) - \ln(1 + \exp(w_0 + \sum_i w_i x_i^j)) \]

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

Update rule:
\[ \Delta w = \eta \nabla_w l(w) \]
\[ w_i^{(t+1)} = w_i^{(t)} + \eta \frac{\partial l(w)}{\partial w_i} \]

Gradient ascent algorithm: (learning rate \( \eta > 0 \))
do:
\[ w_0^{(t+1)} = w_0^{(t)} + \eta \sum_j [y^j - P(Y = 1|\xi^j, w)] \]

For \( i = 1 \) to \( n \): (iterate over features)
\[ w_i^{(t+1)} = w_i^{(t)} + \eta \sum_j x_i^j [y^j - P(Y = 1|\xi^j, w)] \]
until “change” < \( \epsilon \)

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

- Regularization
  - Helps avoid very large weights and overfitting

- MAP estimate:
  \[ w^* = \arg \max_w \prod_i P(y_i \mid x_i, w) \]

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

- 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.