Machine Learning as Search & as Continuous Optimization

CSE 573

DANIEL WELD
Acknowledgements

Some of the material in the decision trees presentation is courtesy of Andrew Moore, from his excellent collection of ML tutorials:

◦ [http://www.cs.cmu.edu/~awm/tutorials](http://www.cs.cmu.edu/~awm/tutorials)

Improved by

◦ Carlos Guestrin, Luke Zettlemoyer, Dan Weld
Logistics

PS1 due Thurs 1/19
PS2 due Thurs 1/26
Machine Learning

Study of algorithms that improve their performance at some task with experience.
## Space of ML Problems

### Type of Supervision
(eg, Experience, Feedback)

<table>
<thead>
<tr>
<th>What is Being Learned?</th>
<th>Labeled Examples</th>
<th>Reward</th>
<th>Examples w/o labels</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discrete Function</strong></td>
<td>Classification</td>
<td></td>
<td>Clustering</td>
</tr>
<tr>
<td><strong>Continuous Function</strong></td>
<td>Regression</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Policy</strong></td>
<td>Apprenticeship Learning</td>
<td>Reinforcement Learning</td>
<td></td>
</tr>
</tbody>
</table>
Classification
from data to discrete classes

Task: Predicting class membership (e.g., spam or not?)
  Output = F: messages → T/F

Performance: Accuracy of prediction

Experience: Labeled examples
  \{ … <message_i, T>… \}
# Training Data for Spam Filtering

"Features"

<table>
<thead>
<tr>
<th>a</th>
<th>...</th>
<th>homework</th>
<th>...</th>
<th>viagra</th>
<th>...</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>F</td>
</tr>
</tbody>
</table>
Weather prediction
Object detection

(Prof. H. Schneiderman)

Example training images for each orientation
The classification pipeline
Hypothesis:
Function for labeling examples

Prediction: +
Prediction: -
Key Concepts
Generalization

Hypotheses must *generalize* to correctly classify instances not in the training data.

Simply memorizing training examples is a consistent hypothesis *that does not generalize*. 
ML = Function Approximation

May not be any perfect fit

Classification $\sim$ discrete functions

$$h(x) = \text{contains}(`\text{nigeria}', x) \quad \land \quad \text{contains}(`\text{wire-transfer}', x)$$
Why is Learning Possible?

Experience alone never justifies any conclusion about any unseen instance.

Learning occurs when PREJUDICE meets DATA!
Bias

The nice word for prejudice is “bias”.
○ Different from “Bias” in statistics

What kind of hypotheses will you consider?
○ What is allowable range of functions you use when approximating?
○ E.g., pure conjunctions, linear separators, ...

What kind of hypotheses do you prefer?
○ E.g., simple with few parameters

“It is needless to do more when less will suffice”
– William of Occam,
died 1349 of the Black plague
ML as Optimization

Specify Preference Bias
◦ aka “Loss Function”

Solve using optimization
◦ Combinatorial
◦ Convex
◦ Linear
◦ Nasty
Overfitting

Hypothesis $H$ is **overfit** when $\exists H'$ and

- $H$ has *smaller* error on training examples, but
- $H$ has *bigger* error on test examples
Overfitting

Hypothesis H is *overfit* when ∃ H’ and
- H has *smaller* error on training examples, but
- H has *bigger* error on test examples

Causes of overfitting
- Training set is too small
- Large number of features

Some solutions
- Validation set
- Regularization
Overfitting

Accuracy

On training data
On test data

Model complexity (e.g., number of nodes in decision tree)
A learning problem: predict fuel efficiency
From the UCI repository (thanks to Ross Quinlan)

- 40 Records
- Discrete data (for now)
- Predict MPG

<table>
<thead>
<tr>
<th>mpg</th>
<th>cylinders</th>
<th>displacement</th>
<th>horsepower</th>
<th>weight</th>
<th>acceleration</th>
<th>modelyear</th>
<th>maker</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>75to78</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>70to74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>75to78</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>low</td>
<td>70to74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>70to74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
<td>70to74</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>low</td>
<td>70to74</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>low</td>
<td>75to78</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>75to78</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
<td>75to78</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>5</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>75to78</td>
<td>europe</td>
</tr>
</tbody>
</table>

Need to find “Hypothesis”: \( f : X \rightarrow Y \)
How Represent Function?

\[ f \left( \begin{array}{cccccc}
\text{cylinders} & \text{displacement} & \text{horsepower} & \text{weight} & \text{acceleration} & \text{modelyear} \\
4 & \text{low} & \text{low} & \text{high} & 75\text{to}78 & \text{asia}
\end{array} \right) \rightarrow \text{mpg} \]

Good

General Propositional Logic?

\[ \text{maker} = \text{asia} \lor \text{weight} = \text{low} \]

Need to find “Hypothesis”:

\[ f : X \rightarrow Y \]
Hypotheses: decision trees $f : X \rightarrow Y$

- Each internal node tests an attribute $x_i$
- Each branch assigns an attribute value $x_i=v$
- Each leaf assigns a class $y$
- To classify input $x$? traverse the tree from root to leaf, output the labeled $y$
What functions can be represented?

cyl=3 ∨ (cyl=4 ∧ (maker=asia ∨ maker= europe)) ∨ …
Are all decision trees equal?

Many trees can represent the same concept

But, not all trees will have the same size!

e.g., $\phi = (A \land B) \lor (\neg A \land C)$

How to find the best tree?
Learning decision trees is hard!!!

Finding the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest ’76]

What to do?
Learning as Search

Nodes?
Operators?
Start State?
Goal?
Search Algorithm?
Heuristic?
The Starting Node: What is the Simplest Tree?

```
predict mpg=bad
```

Is this a good tree?

\[ 22+, 18- \]

Means: correct on 22 examples incorrect on 18 examples
Operators: Improving the Tree

predict
mpg=bad
Recursive Step

Take the Original Dataset.

And partition it according to the value of the attribute we split on.

Records in which cylinders = 4
Records in which cylinders = 5
Records in which cylinders = 6
Records in which cylinders = 8
Recursive Step

- mpg values: bad good
- root: 22 18
- pchance = 0.001

- cylinders = 3
  - Predict bad: 0 0
  - Predict good: 4 17

- cylinders = 4
  - Predict good: 1 0

- cylinders = 5
  - Predict bad: 8 0

- cylinders = 6
  - Predict bad: 9 1

- Build tree from these records..
Second level of tree

Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)
A full tree
Two Questions

1. Which attribute gives the best split?
2. When to stop recursion?

Hill Climbing Algorithm:
- Start from empty decision tree
- Split on the **best attribute** (feature)
- Recurse
Splitting: choosing a good attribute

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

Idea: use counts at leaves to define probability distributions so we can measure uncertainty!
Measuring uncertainty

Good split if we are more certain about classification after split

- Deterministic good (all true or all false)
- Uniform distribution? Bad
- What about distributions in between?

<table>
<thead>
<tr>
<th>P(Y=A)</th>
<th>P(Y=B)</th>
<th>P(Y=C)</th>
<th>P(Y=D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>1/4</td>
<td>1/8</td>
<td>1/8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P(Y=A)</th>
<th>P(Y=B)</th>
<th>P(Y=C)</th>
<th>P(Y=D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/3</td>
<td>1/4</td>
<td>1/4</td>
<td>1/6</td>
</tr>
</tbody>
</table>
Which attribute gives the best split?

A\textsubscript{1}: The one with the highest \textit{information gain}

Defined in terms of \textit{entropy}

A\textsubscript{2}: Actually many alternatives, eg, \textit{accuracy}

Seeks to reduce the \textit{misclassification rate}
Entropy

Entropy $H(Y)$ of a random variable $Y$

$$H(Y) = - \sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

More uncertainty, more entropy!

Information Theory interpretation:

$H(Y)$ is the expected number of bits needed to encode a randomly drawn value of $Y$ (under most efficient code)
Entropy Example

\[ H(Y) = - \sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i) \]

\[
P(Y=t) = \frac{5}{6}
\]
\[
P(Y=f) = \frac{1}{6}
\]

\[
H(Y) = - \frac{5}{6} \log_2 \frac{5}{6} - \frac{1}{6} \log_2 \frac{1}{6}
\]
\[
= 0.65
\]
Conditional Entropy

Conditional Entropy $H(Y \mid X)$ of a random variable $Y$ conditioned on a random variable $X$

$$H(Y \mid X) = - \sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j)$$

Example:

<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>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>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

$P(X_1=t) = 4/6$

$P(X_1=f) = 2/6$

$H(Y \mid X_1) = - \frac{4}{6} (1 \log_2 1 + 0 \log_2 0)$

$= \frac{2}{6} (1/2 \log_2 1/2 + 1/2 \log_2 1/2)$

$= 0.33$
Information Gain

Advantage of attribute – decrease in entropy (uncertainty) after splitting

\[ IG(X) = H(Y) - H(Y | X) \]

In our running example:

\[ IG(X_1) = H(Y) - H(Y | X_1) \]
\[ = 0.65 - 0.33 \]

\[ IG(X_1) > 0 \rightarrow \text{we prefer the split!} \]
Learning Decision Trees

Start from empty decision tree

Split on **next best attribute (feature)**

- Use information gain (or...?) to select attribute:

\[
\text{arg max}_i IG(X_i) = \text{arg max}_i H(Y) - H(Y | X_i)
\]

Recurse
Suppose we want to predict MPG

Now, Look at all the information gains...

predict \( \text{mpg}=\text{bad} \)
Tree After One Iteration

mpg values: bad good

root
22 18
pchance = 0.001

cylinders = 3
0 0
Predict bad

cylinders = 4
4 17
Predict good

cylinders = 5
1 0
Predict bad

cylinders = 6
8 0
Predict bad

cylinders = 8
9 1
Predict bad
When to Terminate?
Don’t split a node if all matching records have the same output value.
Don’t split a node if none of the attributes can create multiple [non-empty] children.
Base Cases: An idea

Base Case One: If all records in current data subset have the same output then don’t recurse

Base Case Two: If all records have exactly the same set of input attributes then don’t recurse

Proposed Base Case 3: If all attributes have zero information gain then don’t recurse

Is this a good idea?
The problem with Base Case 3

\[ y = a \text{ XOR } b \]

The information gains:

<table>
<thead>
<tr>
<th>Input</th>
<th>Value</th>
<th>Distribution</th>
<th>Info Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The resulting decision tree:

y values: 0 1

root

2 2

Predict 0
But *Without* Base Case 3:

\[ y = a \text{ XOR } b \]

The resulting decision tree:

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

So: Base Case 3? Include or Omit?
General View of a Classifier

Hypothesis:
Decision Boundary for labeling function

Label: +
Label: -
Decision Tree Decision Boundaries

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the $K$ classes.
Ok, so how does it perform?
mpg values: bad good

Num Errors Set Size Percent Wrong
Training Set 1 40 2.50
Test Set 74 352 21.02

horsepower = low
0 4
Predict good
pchance = 0.894

horsepower = medium
2 1

horsepower = high
0 0
Predict bad

acceleration = low
1 0
Predict bad

acceleration = medium
1 1
Predict good
pchance = 0.717

acceleration = high
0 1

modelyear = 70to74
0 0
Predict bad

modelyear = 75to78
1 0
Predict bad

modelyear = 79to83
0 0
Predict bad
The test set error is much worse than the training set error…

...why?
Decision trees will overfit

Our decision trees have no learning bias

◦ Training set error is always zero!
  ◦ (If there is no label noise)
◦ Lots of variance
◦ Will definitely overfit!!!
◦ Must introduce some bias towards simpler trees

Why might one pick simpler trees?
Occam’s Razor

Why Favor Short Hypotheses?

Arguments for:
- Fewer short hypotheses than long ones
  - A short hyp. less likely to fit data by coincidence
  - Longer hyp. that fit data may might be coincidence

Arguments against:
- Argument above uses fact that hypothesis space is small!
- What is so special about small sets based on the complexity of each hypothesis?
How to Build Small Trees

Several reasonable approaches:

Stop growing tree before overfit

- Bound depth or # leaves
- Base Case 3
- *Doesn’t work well in practice*

Grow full tree; then prune

- **Optimize on a held-out (development set)**
  - If growing the tree hurts performance, then cut back
  - Con: Requires a larger amount of data...
- **Use statistical significance testing**
  - Test if the improvement for any split is likely due to noise
  - If so, then prune the split!
- **Convert to logical rules**
  - Then simplify rules
Reduced Error Pruning

Split data into *training* & *validation* sets (10-33%)

Train on training set (overfitting)

Do until further pruning is harmful:

1) Evaluate effect on validation set of pruning *each* possible node (and tree below it)

2) Greedily remove the node that *most improves accuracy of validation set*
Alternatively

Chi-squared pruning
  ◦ Grow tree fully
  ◦ Consider leaves in turn
    ◦ Is parent split worth it?

Compared to Base-Case 3?
Consider this split
A chi-square test

Suppose that mpg was completely *uncorrelated* with maker.

What is the chance we’d have seen data of at least this apparent level of association anyway?

By using a particular kind of chi-square test, the answer is 13.5%.

Such hypothesis tests are relatively easy to compute, but involved...
Using Chi-squared to avoid overfitting

Build the full decision tree as before

But when you can grow it no more, start to prune:

° Beginning at the bottom of the tree, delete splits in which $p_{\text{chance}} > \text{MaxPchance}$
° Continue working you way up until there are no more prunable nodes

$\text{MaxPchance}$ is a magic parameter you must specify to the decision tree, indicating your willingness to risk fitting noise
Regularization

Note for Future: **MaxPchance** is a regularization parameter that helps us bias towards simpler models.

We’ll learn to choose the value of magic parameters like this one later!
ML as Optimization

Greedy search for best \textit{scoring} hypothesis

Where $score =$

\begin{itemize}
  \item Fits training data most accurately?
  \item Sum: \textit{training accuracy} $-$ \textit{complexity penalty}
\end{itemize}

\{ regularization \}
Advanced Decision Trees

Attributes with:

- Numerous Possible Values
- Continuous (Ordered) Values
- Missing Values
Decision trees are one of the most popular ML tools
- Easy to understand, implement, and use
- Computationally cheap (to solve heuristically)

Information gain to select attributes (ID3, C4.5,...)

Presented for classification, can be used for regression and density estimation too

Decision trees will overfit!!!
- Must use tricks to find “simple trees”, e.g.,
  - Fixed depth/Early stopping
  - Pruning
  - Hypothesis testing
Loss Functions

How measure quality of hypothesis?
Loss Functions

How measure quality of hypothesis?

\[ L(x, y, \hat{y}) = \text{utility} \left( \text{result of using } y \text{ given input of } x \right) - \text{utility} \left( \text{result of using } \hat{y} \text{ given input of } x \right) \]

\[ L(\text{edible, poison}) \]

\[ L(\text{poison, edible}) \]
Common Loss Functions

0/1 loss

0 if \( y = \hat{y} \) else 1

Absolute value loss

\( |y - \hat{y}| \)

Squared error loss

\( |y - \hat{y}|^2 \)
Overview of Learning

Type of Supervision
(eg, Experience, Feedback)

<table>
<thead>
<tr>
<th>What is Being Learned?</th>
<th>Labeled Examples</th>
<th>Reward</th>
<th>Nothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Function</td>
<td>Classification</td>
<td></td>
<td>Clustering</td>
</tr>
<tr>
<td>Continuous Function</td>
<td><strong>Regression</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Policy</td>
<td>Apprenticeship Learning</td>
<td>Reinforcement Learning</td>
<td></td>
</tr>
</tbody>
</table>
Polynomial Curve Fitting

Hypothesis Space

\[ y(x, w) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \]
Sum-of-Squares Error Function

\[ E(w) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2 \]
1\textsuperscript{st} Order Polynomial

\begin{center}
\begin{tikzpicture}
\begin{axis}[
width=\textwidth, height=\textwidth, xlabel={x}, ylabel={t},
]
\addplot[red] coordinates {
(0,1)
(0.5,0.5)
(1,1)
};
\addplot[green] coordinates {
(0,0)
(0.5,1)
(1,0)
};
\addplot[blue] coordinates {
(0.2,1)
(0.4,0.5)
(0.6,1)
(0.7,0.5)
(0.8,1)
(0.9,0.5)
};
\node at (axis cs:1,1) {$M = 1$};
\end{axis}
\end{tikzpicture}
\end{center}
$3^{rd}$ Order Polynomial

$M = 3$
9\textsuperscript{th} Order Polynomial

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{9th_order_polynomial.png}
\caption{9th Order Polynomial}
\end{figure}
Over-fitting

Root-Mean-Square (RMS) Error: \( E_{RMS} = \sqrt{2E(w^*)/N} \)
## Polynomial Coefficients

<table>
<thead>
<tr>
<th></th>
<th>(M = 0)</th>
<th>(M = 1)</th>
<th>(M = 3)</th>
<th>(M = 9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(w_0^*)</td>
<td>0.19</td>
<td>0.82</td>
<td>0.31</td>
<td>0.35</td>
</tr>
<tr>
<td>(w_1^*)</td>
<td></td>
<td>-1.27</td>
<td>7.99</td>
<td>232.37</td>
</tr>
<tr>
<td>(w_2^*)</td>
<td></td>
<td></td>
<td>-25.43</td>
<td>-5321.83</td>
</tr>
<tr>
<td>(w_3^*)</td>
<td></td>
<td></td>
<td>17.37</td>
<td>48568.31</td>
</tr>
<tr>
<td>(w_4^*)</td>
<td></td>
<td></td>
<td></td>
<td>-231639.30</td>
</tr>
<tr>
<td>(w_5^*)</td>
<td></td>
<td></td>
<td></td>
<td>640042.26</td>
</tr>
<tr>
<td>(w_6^*)</td>
<td></td>
<td></td>
<td></td>
<td>-1061800.52</td>
</tr>
<tr>
<td>(w_7^*)</td>
<td></td>
<td></td>
<td></td>
<td>1042400.18</td>
</tr>
<tr>
<td>(w_8^*)</td>
<td></td>
<td></td>
<td></td>
<td>-557682.99</td>
</tr>
<tr>
<td>(w_9^*)</td>
<td></td>
<td></td>
<td></td>
<td>125201.43</td>
</tr>
</tbody>
</table>
Data Set Size: $N = 15$

9\textsuperscript{th} Order Polynomial
Data Set Size: $N = 100$

$9^{th}$ Order Polynomial
Regularization

\[ \tilde{E}(w) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2} \|w\|^2 \]

Penalize large coefficient values
Regularization:

\[ \ln \lambda = -18 \]
Regularization:

\[ \ln \lambda = 0 \]
Regularization: $E_{\text{RMS}}$ vs. $\ln \lambda$
## Polynomial Coefficients

<table>
<thead>
<tr>
<th></th>
<th>$\ln \lambda = -\infty$</th>
<th>$\ln \lambda = -18$</th>
<th>$\ln \lambda = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0^*$</td>
<td>0.35</td>
<td>0.35</td>
<td>0.13</td>
</tr>
<tr>
<td>$w_1^*$</td>
<td>232.37</td>
<td>4.74</td>
<td>-0.05</td>
</tr>
<tr>
<td>$w_2^*$</td>
<td>-5321.83</td>
<td>-0.77</td>
<td>-0.06</td>
</tr>
<tr>
<td>$w_3^*$</td>
<td>48568.31</td>
<td>-31.97</td>
<td>-0.05</td>
</tr>
<tr>
<td>$w_4^*$</td>
<td>-231639.30</td>
<td>-3.89</td>
<td>-0.03</td>
</tr>
<tr>
<td>$w_5^*$</td>
<td>640042.26</td>
<td>55.28</td>
<td>-0.02</td>
</tr>
<tr>
<td>$w_6^*$</td>
<td>-1061800.52</td>
<td>41.32</td>
<td>-0.01</td>
</tr>
<tr>
<td>$w_7^*$</td>
<td>1042400.18</td>
<td>-45.95</td>
<td>-0.00</td>
</tr>
<tr>
<td>$w_8^*$</td>
<td>-557682.99</td>
<td>-91.53</td>
<td>0.00</td>
</tr>
<tr>
<td>$w_9^*$</td>
<td>125201.43</td>
<td>72.68</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Part 2

Continuous Optimization
Machine Learning

Supervised Learning

Parametric
- Gaussians
  - Learned in closed form
- Linear Functions
  - 1. Learned in closed form
  - 2. Using gradient descent

Non-parametric

Y Continuous

Decision Trees
- Greedy search; pruning

Y Discrete

Unsupervised Learning

Reinforcement Learning

Parametric

Non-parametric

Non-probabilistic Linear Classifier
- Perceptron – w/ gradient descent

Probability of Class | Features
1. Learn $P(Y)$, $P(X|Y)$; apply Bayes
2. Learn $P(Y|X)$ w/ gradient descent
Hypothesis Expressiveness

**LINEAR**
- Naïve Bayes
- Logistic Regression
- Perceptron
- Support Vector Machines

**NONLINEAR**
- Decision Trees
- Neural Networks
- Ensembles
- Kernel Methods
- Nearest Neighbor
- Graphical Models
Logistic Regression

Want to Learn: \( h: X \mapsto Y \)
- \( X \) – features
- \( Y \) – target classes

Probabilistic Discriminative Classifier
- Assume some \textbf{functional form for} \( P(Y|X) \)
  - \textbf{Logistic Function}
  - Accepts both discrete & continuous features
- Estimate parameters of \( P(Y|X) \) directly from training data
- This is the \textit{‘discriminative’ model}
  - Directly learn \( P(Y|X) \)
  - But \textbf{cannot generate a sample of the data},
  - No way to compute \( P(X) \)
Earthquake or Nuclear Test?

\[ P(Y = 1 | X = \langle X_1, \ldots, X_n \rangle) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)} \]

implies

\[ \ln \frac{P(Y = 0 | X)}{P(Y = 1 | X)} = w_0 + \sum_i w_i X_i \]

linear classification rule!
Logistic w/ Initial Weights

\[ w_0 = 20 \quad w_1 = -5 \quad w_2 = 10 \]

Loss(\(H_w\)) = Error(\(H_w, \text{data}\))
Minimize Error \(\rightarrow\) Maximize \(l(w) = \ln P(D_Y | D_x, H_w)\)

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

Step size
Gradient Ascent

\[ w_0 = 40 \quad w_1 = -10 \quad w_2 = 5 \]

Maximize \( l(\mathbf{w}) = \ln P(D_Y | D_x, H_w) \)

Update rule:
\[ \Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w}) \]
\[ w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(\mathbf{w})}{\partial w_i} \]
Root Finding

![Diagram showing minimum, maximum, and saddle point](http://www.deeplearningbook.org/contents/numerical.html)

Fig from “Deep Learning” by Goodfellow et al.
http://www.deeplearningbook.org/contents/numerical.html
Gradient Descent

Assume we have a continuous function: \( f(x_1, x_2, \ldots, x_N) \) and we want to minimize over continuous variables \( X_1, X_2, \ldots, X_n \)

1. Compute the gradients for all \( i \): \( \frac{\partial f(x_1, x_2, \ldots, x_N)}{\partial x_i} \)

2. Take a small step downhill in the direction of the gradient:

   \[ x_i \leftarrow x_i - \lambda \frac{\partial f(x_1, x_2, \ldots, x_N)}{\partial x_i} \]

3. Repeat.

- How to select step size, \( \lambda \)
  - Line search: successively double
    - until \( f \) starts to increase again
Higher Order Derivatives

Fig from “Deep Learning” by Goodfellow et al.
http://www.deeplearningbook.org/contents/numerical.html
Newton’s Method

Assume function can be locally approximated with quadratic
Use both first & second derivatives
Newton’s Method
Newton’s Method
Newton’s Method

At each step:

\[ x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \]

Requires 1\textsuperscript{st} and 2\textsuperscript{nd} derivatives

Quadratic convergence
Newton’s Method in Multiple Dimensions

Replace 1\textsuperscript{st} derivative with gradient, 2\textsuperscript{nd} derivative with Hessian

\[ f(x, y) \]

\[ \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} \]

\[ H = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} \]
Newton’s Method in Multiple Dimensions

Replace 1\textsuperscript{st} derivative with gradient, 2\textsuperscript{nd} derivative with Hessian

So,

\[
\bar{x}_{k+1} = \bar{x}_k - H^{-1}(\bar{x}_k) \nabla f(\bar{x}_k)
\]

Tends to be extremely fragile unless function very smooth and starting close to minimum
Problem With Steepest Descent
Conjugate Gradient Methods

Idea: avoid “undoing” minimization that’s already been done

Walk along direction

\[ d_{k+1} = -g_{k+1} + \beta_k d_k \]

Polak and Ribiere formula:

\[ \beta_k = \frac{g^T_{k+1} (g_{k+1} - g_k)}{g^T_k g_k} \]
Conjugate Gradient Methods

Conjugate gradient implicitly obtains information about Hessian

For quadratic function in $n$ dimensions, gets exact solution in $n$ steps (ignoring roundoff error)

Works well in practice...