Today

- Neural Nets -- wrap

- Formalizing Learning
  - Consistency
  - Simplicity

- Decision Trees
  - Expressiveness
  - Information Gain
  - Overfitting
Deep Neural Network

\[ z_i^{(k)} = g(\sum_j W^{(k-1),k}_{i,j} z_j^{(k-1)}) \]
\[ g = \text{nonlinear activation function} \]

Deep Neural Network: Also Learn the Features!

- Training the deep neural network is just like logistic regression:

\[ \max_w \; ll(w) = \max_w \sum_i \log P(y^{(i)} | x^{(i)}; w) \]

just \( w \) tends to be a much, much larger vector 😊

- just run gradient ascent
  + stop when log likelihood of hold-out data starts to decrease
Neural Networks Properties

- Theorem (Universal Function Approximators). A two-layer neural network with a sufficient number of neurons can approximate any continuous function to any desired accuracy.

- Practical considerations
  - Can be seen as learning the features
  - Large number of neurons
    - Danger for overfitting
    - (hence early stopping!)

How well does it work?
Computer Vision

Object Detection
Manual Feature Design

Features and Generalization

[Hog: Dalal and Triggs, 2005]
Features and Generalization

Image

HoG

Performance

ImageNet Error Rate 2010-2014

graph credit Matt Zeiler, Clarifai
Performance

ImageNet Error Rate 2010-2014

graph credit Matt Zeiler, Clarifai

Performance

ImageNet Error Rate 2010-2014

graph credit Matt Zeiler, Clarifai

AlexNet
MS COCO Image Captioning Challenge

Karpathy & Fei-Fei, 2015; Donahue et al., 2015; Xu et al, 2015; many more

Visual QA Challenge
Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh
Speech Recognition

TIMIT Speech Recognition

Graph credit Matt Zeiler, Clarifai

Machine Translation

Google Neural Machine Translation (in production)
Today

- Neural Nets -- wrap
- **Formalizing Learning**
  - Consistency
  - Simplicity
- Decision Trees
  - Expressiveness
  - Information Gain
  - Overfitting
- Clustering

Inductive Learning
Simplest form: learn a function from examples
- A target function: $g$
- Examples: input-output pairs $(x, g(x))$
  - E.g. $x$ is an email and $g(x)$ is spam / ham
  - E.g. $x$ is a house and $g(x)$ is its selling price

Problem:
- Given a hypothesis space $H$
- Given a training set of examples $x_j$
- Find a hypothesis $h(x)$ such that $h \approx g$

Includes:
- Classification (outputs = class labels)
- Regression (outputs = real numbers)

How do perceptron and naive Bayes fit in? ($H$, $h$, $g$, etc.)

Curve fitting (regression, function approximation):
- Consistency vs. simplicity
- Ockham’s razor
Consistency vs. Simplicity

- Fundamental tradeoff: bias vs. variance

- Usually algorithms prefer consistency by default (why?)

- Several ways to operationalize “simplicity”
  - Reduce the hypothesis space
    - Assume more: e.g. independence assumptions, as in naïve Bayes
    - Have fewer, better features / attributes: feature selection
    - Other structural limitations (decision lists vs trees)
  - Regularization
    - Smoothing: cautious use of small counts
    - Many other generalization parameters (pruning cutoffs today)
    - Hypothesis space stays big, but harder to get to the outskirts

Decision Trees
Reminder: Features

- **Features, aka attributes**
  - Sometimes: TYPE=French
  - Sometimes: $f_{\text{TYPE=French}}(x) = 1$

<table>
<thead>
<tr>
<th>Example</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>WillWait</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X_2</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>30–60</td>
<td>F</td>
</tr>
<tr>
<td>X_3</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X_4</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>10–30</td>
<td>T</td>
</tr>
<tr>
<td>X_5</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X_6</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X_7</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>X_8</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X_9</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X_{10}</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>Italian</td>
<td>10–30</td>
<td>F</td>
</tr>
<tr>
<td>X_{11}</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>X_{12}</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30–60</td>
<td>T</td>
</tr>
</tbody>
</table>

**Decision Trees**

- **Compact representation of a function:**
  - Truth table
  - Conditional probability table
  - Regression values

- **True function**
  - Realizable: in $\mathcal{H}$
Expressiveness of DTs

- Can express any function of the features

```
A   B   A xor B
F   F   F
F   T   T
T   F   T
T   T   F
```

$$P(C|A, B)$$

- However, we hope for compact trees

Comparison: Perceptrons

- What is the expressiveness of a perceptron over these features?

<table>
<thead>
<tr>
<th>Example</th>
<th>Attrs</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>T F F T Same SSS F T French 0-10 T</td>
<td></td>
</tr>
<tr>
<td>X2</td>
<td>T F F T Full $ F F Thai 30-60 F</td>
<td></td>
</tr>
</tbody>
</table>

- For a perceptron, a feature’s contribution is either positive or negative
  - If you want one feature’s effect to depend on another, you have to add a new conjunction feature
  - E.g. adding “PATRONS=full ∧ WAIT = 60” allows a perceptron to model the interaction between the two atomic features

- DTs automatically conjoin features / attributes
  - Features can have different effects in different branches of the tree!

- Difference between modeling relative evidence weighting (NB) and complex evidence interaction (DTs)
  - Though if the interactions are too complex, may not find the DT greedily
Hypothesis Spaces

- How many distinct decision trees with $n$ Boolean attributes?
  - number of Boolean functions over $n$ attributes
  - number of distinct truth tables with $2^n$ rows
  - $2^{2^n}$
    - E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

- How many trees of depth 1 (decision stumps)?
  - number of Boolean functions over 1 attribute
  - number of truth tables with 2 rows, times $n$
  - $4n$
    - E.g. with 6 Boolean attributes, there are 24 decision stumps

- More expressive hypothesis space:
  - Increases chance that target function can be expressed (good)
  - Increases number of hypotheses consistent with training set (bad, why?)
  - Means we can get better predictions (lower bias)
  - But we may get worse predictions (higher variance)

Decision Tree Learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose “most significant” attribute as root of (sub)tree

```plaintext
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value $v_i$ of best do
            examples ← {elements of examples with best = $v_i$}
            subtree ← DTL(examples, attributes - best, MODE(examples))
            add a branch to tree with label $v_i$ and subtree subtree
        return tree
```
Choosing an Attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) “all positive” or “all negative”

So: we need a measure of how “good” a split is, even if the results aren’t perfectly separated out

Entropy and Information

- Information answers questions
  - The more uncertain about the answer initially, the more information in the answer
  - Scale: bits
    - Answer to Boolean question with prior <1/2, 1/2>?
    - Answer to 4-way question with prior <1/4, 1/4, 1/4, 1/4>?
    - Answer to 4-way question with prior <0, 0, 0, 1>?
    - Answer to 3-way question with prior <1/2, 1/4, 1/4>?

- A probability p is typical of:
  - A uniform distribution of size 1/p
  - A code of length log 1/p
Entropy

- General answer: if prior is $<p_1, \ldots, p_n>$:
  - Information is the expected code length

$$H(\langle p_1, \ldots, p_n \rangle) = E_p \log_2 1/p_i$$

$$= \sum_{i=1}^{n} -p_i \log_2 p_i$$

- Also called the entropy of the distribution
  - More uniform = higher entropy
  - More values = higher entropy
  - More peaked = lower entropy
  - Rare values almost “don’t count”

Information Gain

- Back to decision trees!
- For each split, compare entropy before and after
  - Difference is the information gain
  - Problem: there’s more than one distribution after split!

- Solution: use expected entropy, weighted by the number of examples
Next Step: Recurse

- Now we need to keep growing the tree!
- Two branches are done (why?)
- What to do under “full”?
  - See what examples are there...

### Example: Learned Tree

- Decision tree learned from these 12 examples:
  - Substantially simpler than “true” tree
  - A more complex hypothesis isn’t justified by data
  - Also: it’s reasonable, but wrong
Find the First Split

- Look at information gain for each attribute
- Note that each attribute is correlated with the target!
- What do we split on?
Result: Decision Stump

Second Level
Reminder: Overfitting

- **Overfitting:**
  - When you stop modeling the patterns in the training data (which generalize)
  - And start modeling the noise (which doesn’t)

- **We had this before:**
  - Naïve Bayes: needed to smooth
  - Perceptron: early stopping
The test set error is much worse than the training set error...

...why?

Consider this split
Significance of a Split

- Starting with:
  - Three cars with 4 cylinders, from Asia, with medium HP
  - 2 bad MPG
  - 1 good MPG
- What do we expect from a three-way split?
  - Maybe each example in its own subset?
  - Maybe just what we saw in the last slide?
- Probably shouldn’t split if the counts are so small they could be due to chance
- A chi-squared test can tell us how likely it is that deviations from a perfect split are due to chance
- Each split will have a significance value, $p_{\text{CHANCE}}$

Keeping it General

- Pruning:
  - Build the full decision tree
  - Begin at the bottom of the tree
  - Delete splits in which $p_{\text{CHANCE}} > \text{Max}P_{\text{CHANCE}}$
  - Continue working upward until there are no more prunable nodes
  - Note: some chance nodes may not get pruned because they were “redeemed” later
Pruning example

- With MaxP\(_{\text{CHANCE}} = 0.1\):

```
<table>
<thead>
<tr>
<th>mpg values:</th>
<th>bad</th>
<th>good</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td>cylinders = 3</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>cylinders = 4</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>cylinders = 5</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>cylinders = 6</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>Predict bad</td>
<td>Predict good</td>
<td>Predict bad</td>
</tr>
</tbody>
</table>
```

Note the improved test set accuracy compared with the unpruned tree.

<table>
<thead>
<tr>
<th>Num Errors</th>
<th>Set Size</th>
<th>Percent Wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>Test Set</td>
<td>56</td>
<td>352</td>
</tr>
</tbody>
</table>

Regularization

- MaxP\(_{\text{CHANCE}}\) is a regularization parameter
- Generally, set it using held-out data (as usual)
Two Ways of Controlling Overfitting

- **Limit the hypothesis space**
  - E.g. limit the max depth of trees
  - Easier to analyze

- **Regularize the hypothesis selection**
  - E.g. chance cutoff
  - Disprefer most of the hypotheses unless data is clear
  - Usually done in practice

Next Lecture: Applications!