Neural Networks

Multi-class Logistic Regression

* = special case of neural network
Deep Neural Network = Also learn the features!

$z_j^{(k)} = g\left( \sum_i W_{ij}^{(k-1,k)} z_i^{(k-1)} \right)$
$g$: nonlinear activation function

Training the deep neural network is just like logistic regression:

$\max_w \ell(w) = \max_w \sum_i \log P(y_i|x_i; w)$

just $w$ tends to be a much, much larger vector $\otimes$

$\rightarrow$ just run gradient ascent
+ stop when log likelihood of hold-out data starts to decrease

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!)

But neural net $f$ is never one of those?

No problem: CHAIN RULE:

If $f(x) = g(h(x))$
Then $f'(x) = g'(h(x)) \cdot h'(x)$

Derivatives can be computed by following well-defined procedures
Think of the function as a composition of many functions.  

Automatic Differentiation software  
- e.g. Theano, TensorFlow, PyTorch, Chainer  
- Only need to program the function \( g(x,y,w) \)  
- Can automatically compute all derivatives w.r.t. all entries in \( w \)  
- This is typically done by caching info during forward computation pass of \( f \), and then doing a backward pass = “backpropagation”  
- Autodiff / Backpropagation can often be done at computational cost comparable to the forward pass  
- Need to know this exists  
- How this is done? -- outside of scope of CS188

How this is done?  
- Automatic differentiation software  
- e.g. Theano, TensorFlow, PyTorch, Chainer  
- Only need to program the function \( g(x,y,w) \)  
- Can automatically compute all derivatives w.r.t. all entries in \( w \)  
- This is typically done by caching info during forward computation pass of \( f \), and then doing a backward pass = “backpropagation”  
- Autodiff / Backpropagation can often be done at computational cost comparable to the forward pass  
- Need to know this exists  
- How this is done? -- outside of scope of CS188

Back Propagation: \( g(w) = w_1^2w_2 + 3w_1 \)  
- Suppose we have \( g(w) = w_1^2w_2 + 3w_1 \) and want the gradient at \( w = [2, 3] \)  
- Think of the function as a composition of many functions.  
  - Can use derivative chain rule to compute \( \partial g / \partial w_1 \) and \( \partial g / \partial w_2 \).  
  - \( g = b + c \)  
  - \( b = \alpha \cdot w_2 \)  
  - \( c = \beta \cdot w_3 \)  
  - \( \alpha = 1 \), \( \beta = 1 \)  
  - \( z_1 = g(\sum_j \frac{w_{2j} - 1}{j}) \)  
  - Backpropagation can often be done at computational cost comparable to the forward pass  
  - Need to know this exists  
  - How this is done? -- outside of scope of CS188

Back Propagation: \( g(w) = w_1^2w_2 + 3w_1 \)  
- Suppose we have \( g(w) = w_1^2w_2 + 3w_1 \) and want the gradient at \( w = [2, 3] \)  
- Think of the function as a composition of many functions.  
  - Can use derivative chain rule to compute \( \partial g / \partial w_1 \) and \( \partial g / \partial w_2 \).  
  - \( g = b + c \)  
  - \( b = \alpha \cdot w_2 \)  
  - \( c = \beta \cdot w_3 \)  
  - \( \alpha = 1 \), \( \beta = 1 \)  
  - \( z_1 = g(\sum_j \frac{w_{2j} - 1}{j}) \)  
  - Backpropagation can often be done at computational cost comparable to the forward pass  
  - Need to know this exists  
  - How this is done? -- outside of scope of CS188
Think of the function as a composition of many functions.

Suppose we have $g(w) = w_1^3 w_2 + 3w_1$ and want the gradient at $w = [2, 3]$. Think of the function as a composition of many functions.

- Can use derivative chain rule to compute $\frac{dg}{dw_1}$ and $\frac{dg}{dw_2}$.
- $\frac{dg}{dw_1} = 3w_1^2 w_2 + 3$.
- $\frac{dg}{dw_2} = w_1^3$.

Interpretation: A tiny increase in $w_2$ will result in an approximately linear increase in $g$ due to this cubic function.

$\frac{\partial y}{\partial y} = \frac{3w_1^2 w_2 + 3}{w_1^3}$.

Hint: $b$ may be useful.

How do we reconcile the seeming contradiction? The partial derivative means cube function contributes $3w_1^2$, and bottom p.d. means product contribution $w_1$, so add them.
Suppose we have $g(w) = w_1^2 w_2 + 3w_3$ and want the gradient at $w = [2, 3]$

- Think of the function as a composition of many functions, use chain rule.

- $g = b + c$
  - $b = w_2$
  - $c = w_1^2 w_3$

- $a = w_1^2$
  - $a_1 = 2w_1$
  - $a_2 = w_1 w_3$

- $c = 3w_3$
  - $c_1 = 3$ $w_3$
  - $c_2 = 3w_1 w_3$

- $w_1 = 2$
  - $w_{1,1} = 1$
  - $w_{1,2} = 1$

- $w_2 = 3$
  - $w_{2,1} = 1$

\[ \frac{\partial g}{\partial w} = [24, 36] \]

\[ \frac{\partial g}{\partial b} = 1 \]

\[ \frac{\partial g}{\partial c} = 1 \]

\[ \frac{\partial g}{\partial a} = 2w_1 \]

\[ \frac{\partial g}{\partial a} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w_1} = 2w_1 w_2 + 3w_3 \]

\[ \frac{\partial g}{\partial w_2} = w_1^2 \]

\[ \frac{\partial g}{\partial w_3} = 3w_3 \]

\[ \frac{\partial g}{\partial w_{1,1}} = 2w_1 \]

\[ \frac{\partial g}{\partial w_{1,2}} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w_{2,1}} = w_1^2 \]

\[ \frac{\partial g}{\partial w_{2,2}} = 2w_1 w_3 \]

\[ \frac{\partial g}{\partial w_{3,1}} = 3w_3 \]

\[ \frac{\partial g}{\partial w_{3,2}} = 3w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = [24, 36] \]

\[ \frac{\partial g}{\partial w} = 1 \]

\[ \frac{\partial g}{\partial w} = 1 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 w_2 + 3w_3 \]

\[ \frac{\partial g}{\partial w} = w_1^2 \]

\[ \frac{\partial g}{\partial w} = 3w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]

\[ \frac{\partial g}{\partial w} = w_1 w_3 \]

\[ \frac{\partial g}{\partial w} = 2w_1 \]
Performance

ImageNet Error Rate 2010-2014

Speech Recognition

TIMIT Speech Recognition

What’s still missing? – correlation ≠ causation

What’s still missing? – covariate shift

What’s still missing? – covariate shift

Decision Trees
Reminder: Features

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A xor B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Decision Trees

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

Expressiveness of DTs

- Can express any function of the features

\[
P(C \mid A, B)
\]

- However, we hope for compact trees

Decision Tree Learning

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

Comparison: Perceptrons

- What is the expressiveness of a perceptron over these features?
- 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 \( \land \) \( \text{WAIT}=60 \)" 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

Choosing an Attribute

- Idea: a good attribute splits the examples into subsets that are "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 2-way question with prior \( \frac{1}{2}, \frac{1}{2} \)
    - Answer to 4-way question with prior \( \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4} \)
    - Answer to 4-way question with prior \( 0, 0, 0, 1 \)
    - Answer to 3-way question with prior \( \frac{1}{2}, \frac{1}{4}, \frac{1}{4} \)
- A probability \( p \) is typical of:
  - A uniform distribution of size \( \frac{1}{p} \)
  - A code of length \( \log \frac{1}{p} \)

Entropy

- General answer: if prior is \( p_1, \ldots, p_n \):
  - Information is the expected code length
  - Also called the entropy of the distribution
    - More uniform = higher entropy
    - More values = higher entropy
    - More peaked = lower entropy

\[
H(p_1, \ldots, p_n) = \sum_{i=1}^{n} -p_i \log_2 p_i
\]

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
  - More complex hypothesis isn't justified by data
  - Also: it's reasonable, but wrong

Example: Miles Per Gallon
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

Final Tree

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

MPG Training Error

The test set error is much worse than the training set error... why?
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 \( p \text{CHANCE} \geq \text{MaxPCHANCE} \)
  - Continue working upward until there are no more prunable nodes

Pruning example

- With MaxPCHANCE = 0.1:

Regularization

- MaxPCHANCE is a regularization parameter
- Generally, set it using held-out data (as usual)

A few important points about learning

- Ensemble method of learning trees
- Very effective, used in industry and more
- Related to Boosting

Forward Pointer: Random Forests

- Data, labeled instances, e.g. emails marked spam/ham
- Training set
- Held-out set
- Test set
- Features: attribute-value pairs which characterize each x
- Experimentation cycle
  - Hyperparameters (e.g. model parameters) on training set
  - Compute accuracy of held-out set
- Very important to keep “eyes” on the test set
- Evaluation:
  - Accuracy: fraction of instances predicted correctly
  - Overfitting and generalization:
    - Want a classifier which does well on both data
    - Overfitting: fits the training data very closely, but not generalizing well
    - Underfitting: fits the training set poorly
A few important points about learning

- What should we learn where?
  - Learn parameters from training data
  - Tune hyperparameters on different data
  - Why?
  - For each value of the hyperparameters, train and test on the held-out data
  - Choose the best value and do a final test on the test data

- What are examples of hyperparameters?

Inductive Learning

Inductive Learning (Science)

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

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

- Includes:
  - Classification (outputs = class labels)
  - Regression (outputs = real numbers)
  - How do perceptron and naive Bayes fit in? ($H, h, g$, etc.)

Consistency vs. Simplicity

- Fundamental tradeoff: bias vs. variance
  - Usually algorithms prefer consistency by default (why?)

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