CS 188: Artificial Intelligence
Neural Nets

Instructors: Brijen Thananjeyan and Aditya Baradwaj--- University of California, Berkeley

[These slides were created by Dan Klein, Pieter Abbeel, Sergey Levine. All CS188 materials are at http://ai.berkeley.edu.]
Announcements

- MT2 Self Assessment: on Gradescope + due Sunday
- Q5: clustering and Q7: decision trees: now optional on HW6 written component
- Tomorrow: Guest lecture canceled, math/ML review
Neural Networks
Multi-class Logistic Regression

- special case of neural network

$\mathbf{z}_1 = f_1(x)$

$\mathbf{z}_2 = f_2(x)$

$\mathbf{z}_3 = f_3(x)$

\[ P(y_1|x; w) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}} \]

\[ P(y_2|x; w) = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}} \]

\[ P(y_3|x; w) = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}} \]

3 $\mathbf{z}_i$'s, so 3 classes
Deep Neural Network = Also learn the features!

\[ P(y_1|x; w) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}} \]
\[ P(y_2|x; w) = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}} \]
\[ P(y_3|x; w) = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}} \]
Deep Neural Network = Also learn the features!

\[ f_1(x) \]
\[ f_2(x) \]
\[ f_K(x) \]

\[ P(y_1|x; w) = e^{z_1^{(1)}} + e^{z_2^{(1)}} + e^{z_3^{(1)}} \]
\[ P(y_2|x; w) = e^{z_2^{(1)}} + e^{z_2^{(2)}} + e^{z_3^{(2)}} \]
\[ P(y_3|x; w) = e^{z_3^{(1)}} + e^{z_2^{(2)}} + e^{z_3^{(3)}} \]

\[ g = \text{nonlinear activation function} \]
Deep Neural Network = Also learn the features!

\[
z_i^{(k)} = g\left( \sum_j W_{i,j}^{(k-1,k)} z_j^{(k-1)} \right)
\]

\(g = \text{nonlinear activation function}\)
Common Activation Functions

**Sigmoid Function**

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

**Hyperbolic Tangent**

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

**Rectified Linear Unit (ReLU)**

$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

[Source: MIT 6.891 Intro to Deep Learning.com]
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 about computing all the derivatives?

- Derivatives tables:

\[
\frac{d}{dx} (a) = 0
\]
\[
\frac{d}{dx} (x) = 1
\]
\[
\frac{d}{dx} (au) = a \frac{du}{dx}
\]
\[
\frac{d}{dx} (u + v - w) = \frac{du}{dx} + \frac{dv}{dx} - \frac{dw}{dx}
\]
\[
\frac{d}{dx} (uv) = u \frac{dv}{dx} + v \frac{du}{dx}
\]
\[
\frac{d}{dx} \left( \frac{u}{v} \right) = \frac{1}{v} \frac{du}{dx} - \frac{u}{v^2} \frac{dv}{dx}
\]
\[
\frac{d}{dx} (u^n) = nu^{n-1} \frac{du}{dx}
\]
\[
\frac{d}{dx} (\sqrt{u}) = \frac{1}{2\sqrt{u}} \frac{du}{dx}
\]
\[
\frac{d}{dx} \left( \frac{1}{u} \right) = -\frac{1}{u^2} \frac{du}{dx}
\]
\[
\frac{d}{dx} \left( \frac{1}{u^2} \right) = -\frac{n}{u^{n+1}} \frac{du}{dx}
\]
\[
\frac{d}{dx} [f(u)] = \frac{df}{du} \frac{du}{dx}
\]
\[
\frac{d}{dx} [\ln u] = \frac{d}{dx} [\log_e u] = \frac{1}{u} \frac{du}{dx}
\]
\[
\frac{d}{dx} [\log_a u] = \frac{d}{dx} \log_e \frac{1}{u} \frac{du}{dx}
\]
\[
\frac{d}{dx} e^u = e^u \frac{du}{dx}
\]
\[
\frac{d}{dx} a^u = a^u \ln a \frac{du}{dx}
\]
\[
\frac{d}{dx} \left( u^v \right) = v u^{v-1} \frac{du}{dx} + \ln u \ u^v \frac{dv}{dx}
\]
\[
\frac{d}{dx} \left( \sin u \right) = \cos u \frac{du}{dx}
\]
\[
\frac{d}{dx} \left( \cos u \right) = -\sin u \frac{du}{dx}
\]
\[
\frac{d}{dx} \left( \tan u \right) = \sec^2 u \frac{du}{dx}
\]
\[
\frac{d}{dx} \left( \cot u \right) = -\csc^2 u \frac{du}{dx}
\]
\[
\frac{d}{dx} \left( \sec u \right) = \sec u \tan u \frac{du}{dx}
\]
\[
\frac{d}{dx} \left( \csc u \right) = -\csc u \cot u \frac{du}{dx}
\]

[source: http://hyperphysics.phy-astr.gsu.edu/hbase/Math/derfunc.html]
How about computing all the derivatives?

- But neural net \( f \) is never one of those?
  - No problem: **CHAIN RULE:**

\[
\text{If } \quad f(x) = g(h(x))
\]
\[
\text{Then } \quad f'(x) = g'(h(x))h'(x)
\]

→ Derivatives can be computed by following well-defined procedures
Automatic Differentiation

- **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
Summary of Key Ideas

- Optimize probability of label given input
  \[ \max_w ll(w) = \max_w \sum_i \log P(y^{(i)}|x^{(i)}; w) \]

- Continuous optimization
  - Gradient ascent:
    - Compute steepest uphill direction = gradient (= just vector of partial derivatives)
    - Take step in the gradient direction
    - Repeat (until held-out data accuracy starts to drop = “early stopping”)

- Deep neural nets
  - Last layer = still logistic regression
  - Now also many more layers before this last layer
    - = computing the features
    - \( \rightarrow \) the features are learned rather than hand-designed
  - Universal function approximation theorem
    - If neural net is large enough
    - Then neural net can represent any continuous mapping from input to output with arbitrary accuracy
    - But remember: need to avoid overfitting / memorizing the training data \( \rightarrow \) early stopping!
  - Automatic differentiation gives the derivatives efficiently (how? = outside of scope of 188)
Computer Vision
Object Detection
Features and Generalization

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

Image

HoG
Performance

ImageNet Error Rate 2010-2014

Error Rate

79%
60%
40%
20%
7%

2010  2011  2012  2013  2014

Traditional CV

graph credit Matt Zeiler, Clarifai
Performance

ImageNet Error Rate 2010-2014

graph credit Matt Zeiler, Clarifai
Performance

ImageNet Error Rate 2010-2014

- Traditional CV
- Deep Learning

Error Rate
- 79%
- 60%
- 40%
- 20%
- 7%

2010 2011 2012 2013 2014

AlexNet

graph credit Matt Zeiler, Clarifai
ImageNet Error Rate 2010-2014

Performance

graph credit Matt Zeiler, Clarifai
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

- What vegetable is on the plate?
  - Neural Net: broccoli
  - Ground Truth: broccoli

- What color are the shoes on the person's feet?
  - Neural Net: brown
  - Ground Truth: brown

- How many school buses are there?
  - Neural Net: 2
  - Ground Truth: 2

- What sport is this?
  - Neural Net: baseball
  - Ground Truth: baseball

- What is on top of the refrigerator?
  - Neural Net: magnets
  - Ground Truth: cereal

- What uniform is she wearing?
  - Neural Net: shorts
  - Ground Truth: girl scout

- What is the table number?
  - Neural Net: 4
  - Ground Truth: 40

- What are people sitting under in the back?
  - Neural Net: bench
  - Ground Truth: tent
Semantic Segmentation/Object Detection
Speech Recognition

TIMIT Speech Recognition

- Traditional
- Deep Learning

Error Rate


Graph credit: Matt Zeiler, Clarifai