Announcements:

- HW4 posted
- Poster Session Thurs, Dec 8

Today:
- Review: EM
- Neural nets and deep learning
Poster Session

- Thursday Dec 8, 9-11:30am
  - Please arrive 20 mins early to set up
- Everyone is expected to attend
- Prepare a poster
  - We provide poster board and pins
  - Both one large poster (recommended) and several pinned pages are OK.
- Capture
  - Problem you are solving
  - Data you used
  - ML methodology
  - Results

**Prepare a 1-minute speech about your project**
- Two instructors will visit your poster separately
- Project Grading: scope, depth, data

---

Logistic regression

- \( P(Y|X) \) represented by:
  \[
P(Y = 1 \mid x, W) = \frac{1}{1 + e^{-(w_0 + \sum_i w_i x_i)}}
  = g(w_0 + \sum_i w_i x_i)
\]

- Learning rule – MLE:
  \[
  \frac{\partial \ell(W)}{\partial w_i} = \sum_j x^j_i [y^j - P(Y^j = 1 \mid x^j, W)] \\
  = \sum_j x^j_i [y^j - g(w_0 + \sum_i w_i x^j_i)]
  \]
  \[
  w_i \leftarrow w_i + \eta \sum_j x^j_i \delta^j \\
  \delta^j = y^j - g(w_0 + \sum_i w_i x^j_i)
  \]
Perceptron as a graph

\[ g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{-(w_0 + \sum_i w_i x_i)}} \]

Linear perceptron classification region

\[ g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{-(w_0 + \sum_i w_i x_i)}} \]
Perceptron, linear classification, Boolean functions

- Can learn \( x_1 \text{ AND } x_2 \)

- Can learn \( x_1 \text{ OR } x_2 \)

- Can learn any conjunction or disjunction

---

Perceptron, linear classification, Boolean functions

- Can learn majority

- Can perceptrons do everything?
Going beyond linear classification

- Solving the XOR problem

Hidden layer

- Perceptron: \( out(x) = g(w_0 + \sum_i w_i x_i) \)

- 1-hidden layer:
  \[
  out(x) = g \left( w_0 + \sum_k w_k g(w_0^k + \sum_i w_i^k x_i) \right)
  \]
Example data for NN with hidden layer

A target function:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000000</td>
<td>10000000</td>
</tr>
<tr>
<td>01000000</td>
<td>01000000</td>
</tr>
<tr>
<td>00100000</td>
<td>00100000</td>
</tr>
<tr>
<td>00010000</td>
<td>00010000</td>
</tr>
<tr>
<td>00001000</td>
<td>00001000</td>
</tr>
<tr>
<td>00000100</td>
<td>00000100</td>
</tr>
<tr>
<td>00000010</td>
<td>00000010</td>
</tr>
<tr>
<td>00000001</td>
<td>00000001</td>
</tr>
</tbody>
</table>

Can this be learned??

Learned weights for hidden layer

A network:

Learned hidden layer representation:

<table>
<thead>
<tr>
<th>Input</th>
<th>Hidden</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Values</td>
<td></td>
</tr>
<tr>
<td>10000000</td>
<td>.89</td>
<td>.04</td>
</tr>
<tr>
<td>01000000</td>
<td>.01</td>
<td>.11</td>
</tr>
<tr>
<td>00100000</td>
<td>.01</td>
<td>.97</td>
</tr>
<tr>
<td>00010000</td>
<td>.99</td>
<td>.97</td>
</tr>
<tr>
<td>00001000</td>
<td>.03</td>
<td>.05</td>
</tr>
<tr>
<td>00000100</td>
<td>.22</td>
<td>.99</td>
</tr>
<tr>
<td>00000010</td>
<td>.80</td>
<td>.01</td>
</tr>
<tr>
<td>00000001</td>
<td>.60</td>
<td>.94</td>
</tr>
</tbody>
</table>
NN for images

90% accurate learning head pose, and recognizing 1-of-20 faces

Weights in NN for images

Typical input images
Forward propagation for 1-hidden layer - Prediction

- 1-hidden layer:

\[ \text{out}(x) = g \left( w_0 + \sum_k w_k g \left( w_0^k + \sum_i w_i^k x_i \right) \right) \]

Gradient descent for 1-hidden layer – Back-propagation: Computing \( \frac{\partial \ell(W)}{\partial w_k} \)

\[ \ell(W) = \frac{1}{2} \sum_j \left[ y^j - \text{out}(x^j) \right]^2 \]

\[ \text{out}(x) = g \left( \sum_{k'} w_{k'} g \left( \sum_{i'} w_{i'}^{k'} x_{i'} \right) \right) \]

\[ \frac{\partial \ell(W)}{\partial w_k} = \sum_{j=1}^m \left[ y^j - \text{out}(x^j) \right] \frac{\partial \text{out}(x^j)}{\partial w_k} \]

Dropped \( w_0 \) to make derivation simpler
Gradient descent for 1-hidden layer – Back-propagation: Computing $\frac{\partial \ell(W)}{\partial w^k_i}$

$\ell(W) = \frac{1}{2} \sum_j (y^j - out(x^j))^2$

$out(x) = g \left( \sum_{k'} w_{k'k} g(\sum_{i'} w_{i'k} x_{i'}) \right)$

$\frac{\partial \ell(W)}{\partial w^k_i} = \sum_{j=1}^{m} -[y - out(x^j)] \frac{\partial out(x^j)}{\partial w^k_i}$

Multilayer neural networks
Forward propagation – prediction

- Recursive algorithm
- Start from input layer
- Output of node $V_k$ with parents $U_1, U_2, \ldots$:

$$V_k = g \left( \sum_i w^k_i U_i \right)$$

Back-propagation – learning

- Just stochastic gradient descent!!!
- Recursive algorithm for computing gradient
- For each example
  - Perform forward propagation
  - Start from output layer
  - Compute gradient of node $V_k$ with parents $U_1, U_2, \ldots$
  - Update weight $w^k_i$
Many possible response/link functions

- Sigmoid
- Linear
- Exponential
- Gaussian
- Hinge
- Max
- ...

Convolutional Neural Networks & Application to Computer Vision

Machine Learning – CSE4546
Sham Kakade
University of Washington
November 29, 2016
Neural Networks in Computer Vision

- Neural nets have made an amazing come back
  - Used to engineer high-level features of images

- Image features:
Some hand-created image features

- SIFT
- Spin image
- HoG
- RIFT
- Textons
- GLOH

Scanning an image with a detector

- Detector = Classifier from image patches:

- Typically scan image with detector:
Using neural nets to learn non-linear features

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

But, many tricks needed to work well…

Convoluted Models for Image Recognition

• Learning multiple layers of representation. (LeCun, 1992)
Convolution Layer

- Example: 200x200 image
  - Fully-connected, 400,000 hidden units = 16 billion parameters
  - Locally-connected, 400,000 hidden units 10x10 fields = 40 million params
  - Local connections capture local dependencies

Parameter sharing

- Fundamental technique used throughout ML
- Neural net without parameter sharing:
  - Sharing parameters:
Pooling/Subsampling

- Convolutions act like detectors:
- But we don’t expect true detections in every patch
- Pooling/subsampling nodes:

Example neural net architecture

- Input: 83x83
- Layer 1: 64x75x75
- Layer 2: 64x14x14
- Layer 3: 256x6x6
- Layer 4: 256x1x1
- Output: 101

- 9x9 convolution (64 kernels)
- 10x10 pooling, 5x5 subsampling
- 9x9 convolution (4096 kernels)
- 6x6 pooling, 4x4 subsamp
Sample results

**Traffic Sign Recognition (GTSRB)**
- German Traffic Sign Recon Bench
- 99.2% accuracy

**House Number Recognition (Google)**
- Street View House Numbers
- 94.3% accuracy

---

Example from Krizhevsky, Sutskever, Hinton 2012

Won the 2012 ImageNet LSVRC. 60 Million parameters, 832M MAC ops

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Parameters</th>
<th>MAC Ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL CONNECT</td>
<td>4M</td>
<td>4Mflop</td>
</tr>
<tr>
<td>FULL 4096/ReLU</td>
<td>16M</td>
<td>16M</td>
</tr>
<tr>
<td>FULL 4096/ReLU</td>
<td>37M</td>
<td>37M</td>
</tr>
<tr>
<td>MAX POOLING</td>
<td>442K</td>
<td>74M</td>
</tr>
<tr>
<td>CONV 3x3/ReLU 256fm</td>
<td>1.3M</td>
<td>224M</td>
</tr>
<tr>
<td>CONV 3x3/ReLU 384fm</td>
<td>884K</td>
<td>149M</td>
</tr>
<tr>
<td>MAX POOLING 2x2sub</td>
<td>307K</td>
<td>223M</td>
</tr>
<tr>
<td>LOCAL CONTRAST NORM</td>
<td>35K</td>
<td>105M</td>
</tr>
<tr>
<td>CONV 11x11/ReLU 256fm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONV 11x11/ReLU 384fm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONV 11x11/ReLU 96fm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ImageNet Large Scale Visual Recognition Challenge
1000 categories, 1.5 Million labeled training samples

Object Recognition
[Krizhevsky, Sutskever, Hinton 2012]
TEST IMAGE

RETRIEVED IMAGES

Application to scene parsing

[Farabet et al. ICML 2012, PAMI 2013]
Learning challenges for neural nets

- Choosing architecture
- Slow per iteration and convergence
- Gradient “diffusion” across layers
- Many local optima

Random dropouts

- Standard backprop:

\[ w_i \leftarrow w_i + \eta \sum_j x_j^i \delta_j \]

- Random dropouts: randomly choose edges not to update:

- Functions as a type of “regularization”… helps avoid “diffusion” of gradient
Revival of neural networks

- Neural networks fell into disfavor in mid 90s - early 2000s
  - Many methods have now been rediscovered 😊
- Exciting new results using modifications to optimization techniques and GPUs

- Challenges still remain:
  - Architecture selection feels like a black art
  - Optimization can be very sensitive to parameters
  - Requires a significant amount of expertise to get good results