Logistic regression

- $P(Y|X)$ represented by:

$$P(Y = 1 \mid x, W) = \frac{1}{1 + e^{-(w_0 + \sum_i w_i x_i)}}$$

- Learning rule – MLE:

$$\frac{\partial \ell(W)}{\partial w_i} = \sum_j x_i^j [y^j - P(Y^j = 1 \mid x^j, W)]$$

$$= \sum_j x_i^j [y^j - g(w_0 + \sum_i w_i x_i^j)]$$

$$w_i \leftarrow w_i + \eta \sum_j x_i^j \delta^j$$

$$\delta^j = y^j - g(w_0 + \sum_i w_i x_i^j)$$
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

- Can learn majority
- Can perceptrons do everything?
Going beyond linear classification

- Solving the XOR problem

Hidden layer

- Perceptron: \( \text{out}(x) = g(w_0 + \sum_i w_ix_i) \)

- 1-hidden layer: 
  \[
  \text{out}(x) = g \left( w_0 + \sum_k w_k g(w_0^k + \sum_i w_i^k x_i) \right)
  \]
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 → 10000000</td>
<td></td>
</tr>
<tr>
<td>01000000 → 01000000</td>
<td></td>
</tr>
<tr>
<td>00100000 → 00100000</td>
<td></td>
</tr>
<tr>
<td>00010000 → 00010000</td>
<td></td>
</tr>
<tr>
<td>00001000 → 00001000</td>
<td></td>
</tr>
<tr>
<td>00000100 → 00000100</td>
<td></td>
</tr>
<tr>
<td>00000010 → 00000010</td>
<td></td>
</tr>
<tr>
<td>00000001 → 00000001</td>
<td></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>Values</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10000000 → .89 .04 .08 → 10000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01000000 → .01 .11 .88 → 01000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00100000 → .01 .97 .27 → 00100000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00010000 → .99 .97 .71 → 00010000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00001000 → .03 .05 .02 → 00001000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000100 → .22 .99 .99 → 00000100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000010 → .80 .01 .98 → 00000010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000001 → .60 .94 .01 → 00000001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
NN for images

Weights in NN for images

90% accurate learning head pose, and recognizing 1-of-20 faces
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 \( \frac{\partial \ell(W)}{\partial w_k} \)

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

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

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

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

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

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

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

Multilayer neural networks
Forward propagation – prediction

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

$$V_k = g \left( \sum_i w_i^k U_i \right)$$

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 $w_i^k$
Many possible response/link functions

- Sigmoid
- Linear
- Exponential
- Gaussian
- Hinge
- Max
- …

Poster Session

- Thursday Dec 4, 2:30-4:30pm
  - Please arrive 15mins early to set up
- Everyone is expected to attend
- Prepare a poster
  - We provide poster board (32"x40") and pins
  - Both one large poster and several pinned pages are OK
- Capture
  - Problem you are solving
  - Data you used
  - ML methodology
  - Results
- Prepare a 2-minute speech about your project
- Two instructors will visit your poster separately
- You’ll be graded on 3 criteria:
  - Scope: how much stuff you did
  - Technical depth: how challenging it was to do your project (and whether your methodology was correct)
  - Presentation: how you share what you did
Convolutional Neural Networks & Application to Computer Vision

Machine Learning – CSEP546
Carlos Guestrin
University of Washington
December 2, 2014

Contains slides from…
- LeCun & Ranzato
- Russ Salakhutdinov
- Honglak Lee
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

Slide Credit: Honglak Lee
### Scanning an image with a detector

- Detector = Classifier from image patches:

- Typically scan image with detector:

![Detector example](image)

### Using neural nets to learn non-linear features

**Deep Learning = Learning Hierarchical Representations**

- It's deep if it has more than one stage of non-linear feature transformation

**Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]**
But, many tricks needed to work well...

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

Sample results

**Traffic Sign Recognition (GTSRB)**
- German Traffic Sign Recognition 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

Results by Krizhevsky, Sutskever, Hinton 2012

ImageNet Large Scale Visual Recognition Challenge
1000 categories, 1.5 Million labeled training samples
Application to scene parsing

Semantic Labeling: Labeling every pixel with the object it belongs to

- Would help identify obstacles, targets, landing sites, dangerous areas
- Would help line up depth map with edge maps

Learning challenges for neural nets

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

- Standard backprop:

\[ w_i \leftarrow w_i + \eta \sum_j x_i^j \delta^j \]

- 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