Logistic regression

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

- Learning rule – MLE:
  \[
  \frac{\partial \ell(W)}{\partial w_i} = \sum_j x_i^j [y^j - P(Y^j = 1 | x^j, W)]
  = \sum_j x_i^j [y^j - g(w_0 + \sum_i w_i x_i^j)]
  \]
  \[
  w_i \leftarrow w_i + \eta \sum_j x_i^j \delta^j
  \]
  \[
  \delta^j = y^j - g(w_0 + \sum_i w_i x_i^j)
  \]
Sigmoid

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

- \( w_0 = 2, \ w_i = 1 \)
- \( w_0 = 0, \ w_i = 1 \)
- \( w_0 = 0, \ w_i = 0.5 \)

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

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

Optimizing the perceptron

- Trained to minimize sum-squared error

\[ \ell(W) = \frac{1}{2} \sum_j [y_j - g(w_0 + \sum_i w_ix_i)]^2 \]
Derivative of sigmoid

$$\frac{\partial l(W)}{\partial w_i} = - \sum_j [y^j - g(w_0 + \sum_i w_ix_i^j)] x_i^j g'(w_0 + \sum_i w_ix_i^j)$$

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

The perceptron learning rule

$$w_i \leftarrow w_i + \eta \sum_j x_j^i \delta^j$$

$$\delta^j = [y^j - g(w_0 + \sum_i w_ix_i^j)]g'(w_0 + \sum_i w_ix_i^j)$$

$$g^j = g(w_0 + \sum_i w_ix_i^j)$$

- Compare to MLE:

$$w_i \leftarrow w_i + \eta \sum_j x_j^i \delta^j$$

$$\delta^j = [y^j - g(w_0 + \sum_i w_ix_i^j)]$$
Perceptron, linear classification, Boolean functions

- Can learn $x_1 \lor x_2$
- Can learn $x_1 \land x_2$
- 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: \( \text{out}(x) = g(w_0 + \sum_i w_i x_i) \)
- 1-hidden layer:
  \[
  \text{out}(x) = g \left( w_0 + \sum_k w^k g(w^k_0 + \sum_i w^k_i x_i) \right)
  \]
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 Values</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<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

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

Weights in NN for images
Forward propagation for 1-hidden layer - Prediction

1-hidden layer:
\[ out(x) = g \left( w_0 + \sum_k w_k g \left( \sum_i w_i x_i \right) \right) \]

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

\[ \ell(W) = \frac{1}{2} \sum_j [y^j - out(x^j)]^2 \]
\[ out(x) = g \left( \sum_k w_k g \left( \sum_i w_i x_i \right) \right) \]
\[ \frac{\partial \ell(W)}{\partial w_k} = \sum_{j=1}^m -[y^j - out(x^j)] \frac{\partial out(x^j)}{\partial w_k} \]

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

$\epsilon(W) = \frac{1}{2} \sum_i [y^i - out(x^i)]^2$

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

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

Dropped $w_0$ to make derivation simpler

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 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 functions

- Sigmoid
- Linear
- Exponential
- Gaussian
- ...

Convergence of backprop

- Perceptron leads to convex optimization
  - Gradient descent reaches global minima

- Multilayer neural nets not convex
  - Gradient descent gets stuck in local minima
  - Hard to set learning rate
  - Selecting number of hidden units and layers = fuzzy process
  - NNs falling in disfavor in last few years
  - We’ll see later in semester, kernel trick is a good alternative
  - Nonetheless, neural nets are one of the most used ML approaches
    - Plus, neural nets are back with a new name!!!!
      - Deep belief networks
        - (and a probabilistic interpretation & slightly different learning procedure)
Overfitting?

- Neural nets represent complex functions
  - Output becomes more complex with gradient steps

Overfitting

- Output fits training data “too well”
  - Poor test set accuracy
- Overfitting the training data
  - Related to bias-variance tradeoff
  - One of central problems of ML
- Avoiding overfitting?
  - More training data
  - Regularization
  - Early stopping
What you need to know about neural networks

- Perceptron:
  - Representation
  - Perceptron learning rule
  - Derivation

- Multilayer neural nets
  - Representation
  - Derivation of backprop
  - Learning rule

- Overfitting
  - Definition
  - Training set versus test set
  - Learning curve