Neural Nets for Dummies

Training:
- Choose connection weights that minimize error

Prediction:
- Propagate input feature values through the network of "artificial neurons"

Advantages:
- Fast prediction
- Does feature weighting
- Very generally applicable

Disadvantages:
- Very slow training
- Overfitting is easy

Perceptron Unit

\[ y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i > 0 \\ 0 & \text{otherwise} \end{cases} \]

Creates a decision plane (line) in feature space

Sigmoid Unit

\[ y = \frac{1}{1 + e^{-z}} \]

\[ z = \sum_{i=1}^{n} w_i x_i \]

Creates a "soft" decision plane (line) in feature space

The simplest two-layer sigmoid Neural Net

\[ E = \frac{1}{2} \left( y^* - F(x, \mathbf{w}) \right)^2 \]

\[ y = F(x, \mathbf{w}) \]

\[ y^* = \text{Desired output} \]

Goal: find the weight vector that minimizes the error

Approach: Gradient Descent

\[ \delta y = \delta E / \delta w_j \]

\[ \delta E / \delta w_j = -(y^* - y) \delta y / \delta w_j \]

\[ y = F(x, \mathbf{w}) = s(w_1 x_1) = s(w_2 x_2) \]

Descent rule:

\[ w_{j, \text{new}} = w_{j, \text{old}} - r \delta_j y_i \]

Backpropagation rule:

\[ \delta_j = \frac{dE}{dz_j} \sum_{k} \delta_{j, k} w_{j, k} \]

Adapted from Tomas Lozano Perez's 6.034 Recitation Notes
Example of Backpropagation

Initial Conditions: all weights are zero, learning rate is 8. Input: (x₁, x₂) = (0, 1)

\[
\begin{align*}
y' &= 1 \\
z_i &= 0 \\
y_i &= 1/2 \\
y &= 1/2 \\
z &= 0 \\
y &= 1/2
\end{align*}
\]

Example of Backpropagation

Initial Conditions: all weights are zero, learning rate is 8. Input: (x₁, x₂) = (0, 1)

\[
\begin{align*}
y' &= 1 \\
z_i &= 0 \\
y_i &= 1/2 \\
y &= 1/2 \\
z &= 0 \\
y &= 1/2
\end{align*}
\]