Perceptron (recap)

- Input-to-output function
  \[ a(x) = \mathbf{w}^T \mathbf{x} + w_0 = \mathbf{w}^T \mathbf{x} \]
  where \( \mathbf{w} = (w_0, \mathbf{w}^T)^T \), \( \mathbf{x} = (1, \mathbf{x}^T)^T \)
  \[ y(x) = g(a(x)) = g(\mathbf{w}^T \mathbf{x}) \]
  where \( g(a) = \begin{cases} 1, & \text{if } a \geq 0, \\ 0, & \text{if } a < 0 \end{cases} \)

Today’s Schedule

- Perceptron (recap)
- Problems with Perceptron
- Extensions of Perceptron
- Training of a single-layer neural network

Geometry of Perceptron’s error correction

\[ y(x_i) = g(w^T x_i) \]
\[ w^{(\text{new})} \leftarrow w + \eta (t_i - y(x_i)) x_i \quad (0 < \eta < 1) \]

Limitations of Perceptron

- Single-layer perceptron is just a linear classifier (Marvin Minsky and Seymour Papert, 1969)
- Multi-layer perceptron can form complex decision boundaries (piecewise-linear), but it is hard to train
- Training does not stop if data are linearly non-separable
- Weights \( \mathbf{w} \) are adjusted for misclassified data only (correctly classified data are not considered at all)

A limitation of Perceptron

\[ y = g(w^T x) \]
\[ z_1 = g(w_0^{(1)}) x = g(w_1^{(1)} x_1 + w_2^{(1)} x_2 + w_0^{(1)}) \]
\[ z_2 = g(w_0^{(2)}) x = g(w_1^{(2)} x_1 + w_2^{(2)} x_2 + w_0^{(2)}) \]
\[ y = g(w_0^{(3)}) z_1 + g(w_1^{(3)}) z_2 + g(w_0^{(3)}) \]

Question: Find the weights for each network
Single Layer Neural Network

Assume a single-layer neural network with a single output node with a logistic sigmoid function:

\[ y(x) = g(w^T x) = g \left( \sum_{i=0}^{d} w_i x_i \right) \]

\[ g(a) = \frac{1}{1 + \exp(-a)} \]

Output

\[ x_0 \]

\[ x_1 \]

\[ i \]

\[ + \]

\[ x_d \]

Training set: \( D = \{ (x_1, t_1), \ldots, (x_N, t_N) \} \)

where \( t_i \in \{0, 1\} \)

Error function:

\[ E(w) = \frac{1}{2} \sum_{i=1}^{N} \left( y_i - t_i \right)^2 \]

\[ = \frac{1}{2} \sum_{i=1}^{N} \left( g(w^T x_i) - t_i \right)^2 \]

\[ = \frac{1}{2} \sum_{i=1}^{N} \left( \sum_{i=0}^{d} w_i x_i - t_i \right)^2 \]

Definition of the training problem as an optimisation problem

\[ \text{min}_w E(w) \]

Optimisation problem: \( \text{min}_w E(w) \)

- No analytic solution
- Employ an iterative method (requires initial values)
  - e.g., Gradient descent (steepest decent), Newton’s method, Conjugate gradient methods
- Gradient descent

(Scalar rep.)

- \( w_i^{(new)} \leftarrow w_i - \eta \frac{\partial}{\partial w_i} E(w), \ (\eta > 0) \)

(Gradient descent)

Local minimum problem with the gradient descent

- \( w_i^{(new)} \leftarrow w_i - \eta \frac{\partial}{\partial w_i} E(w), \ (\eta > 0) \)

\[ E(w) = \frac{1}{2} \sum_{i=1}^{N} \left( y_i - t_i \right)^2 = \frac{1}{2} \sum_{i=1}^{N} \left( g \left( \sum_{i=0}^{d} w_i x_i \right) - t_i \right)^2 \]

Let \( a_i = \sum_{i=0}^{d} w_i x_i \), then \( \frac{\partial a_i}{\partial w_j} = x_j \).

\[ \frac{\partial E(w)}{\partial w_j} = \frac{\partial E(w)}{\partial y_i} \frac{\partial y_i}{\partial a_i} \frac{\partial a_i}{\partial w_j} = \sum_{i=1}^{N} (y_i - t_i) \frac{\partial g(a_i)}{\partial a_i} \frac{\partial a_i}{\partial w_j} \]

\[ = \sum_{i=1}^{N} (y_i - t_i) g'(a_i) x_j \]

\[ = \sum_{i=1}^{N} (y_i - t_i) g(a_i) (1 - g(a_i)) x_j \]
Summary

- Limitations of Perceptron
- Solutions to the problems
- Neural network with differentiable non-linear functions (e.g. logistic sigmoid function)
- Training of the network with the gradient descent algorithm
- Considered only a single-layer network with a single-output node
- A very good reference: http://neuralnetworksanddeeplearning.com/