Today's Schedule

1. Single-layer network with a single output node (recap)
2. Single-layer network with multiple output nodes
3. Multi-layer neural network
4. Activation functions

Single-layer network with a single output node (recap)

- Activation function:
  \[ y = g(a) = g(\sum_{i=0}^{N} w_i x_i) \]
  \[ g(a) = \frac{1}{1 + \exp(-a)} \]

- Training set: \[ D = \{(x_n, t_n)\}_{n=1}^{N} \]
where \[ t_n \in \{0, 1\} \]
- Error function:
  \[ E(w) = \frac{1}{2} \sum_{n=1}^{N} (y_n - t_n)^2 \]
- Optimisation problem (training):
  \[ \min_{w} E(w) \]

Single-layer network with multiple output nodes

- Training set:
  \[ D = \{(x_n, (t_{n1}, \ldots, t_{nK}))\}_{n=1}^{N} \]
where \[ t_{nk} \in \{0, 1\} \]
- Error function:
  \[ E(w) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_{nk} - t_{nk})^2 \]
  \[ y_{nk} = g(\sum_{i=0}^{N} w_{ki} x_{ni}) \]
- Optimisation problem (training):
  \[ \min_{w_{ki}} \sum_{n=1}^{N} (y_{nk} - t_{nk})^2 \]

Multi-layer neural networks

- Hidden-to-output weights:
  \[ w^{(2)}_{kj} \leftarrow w^{(2)}_{kj} - \eta \frac{\partial E}{\partial w^{(2)}_{kj}} \]
- Input-to-hidden weights:
  \[ w^{(1)}_{ij} \leftarrow w^{(1)}_{ij} - \eta \frac{\partial E}{\partial w^{(1)}_{ij}} \]
Training of MLP

1940s Warren McCulloch and Walter Pitts: 'threshold logic'
Donald Hebb: 'Hebbian learning'

1957 Frank Rosenblatt: 'Perceptron'

1969 Marvin Minsky and Seymour Papert: limitations of neural networks

1980 Kunihiro Fukushima: 'Neocognitron'


Notes on Activation functions

Summary

Output of logistic sigmoid activation function

- Consider a single-layer network with a single output node
  logistic sigmoid activation function:
  \[ y = g(a) = \frac{1}{1 + e^{-a}} = g \left( \sum_{i=1}^{D} w_{i} x_{i} \right) \]

  \[ a = \sum_{i=1}^{D} w_{i} x_{i} \]

- Consider a two class problem, with classes C1 and C2.
  The posterior probability of C1:
  \[ P(C_1|x) = \frac{p(x|C_1)P(C_1)}{p(x)P(C_1) + p(x|C_2)P(C_2)} = \frac{p(x|C_1)P(C_1)}{1 + \exp \left( - \log \left( \frac{p(x|C_2)P(C_2)}{p(x|C_1)P(C_1)} \right) \right)} \]

Approximation of posterior probabilities

Logistic sigmoid function

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

Posterior probabilities of two classes with Gaussian distributions:

Normalisation of output nodes

Some questions on activation functions

- Outputs with sigmoid activation function:
  \[ \sum_{k=1}^{K} y_k = 1 \]
  \[ y_k = g(a_k) = \frac{1}{1 + e^{-a_k}} \]

- Softmax activation function for g():
  \[ y_k = \frac{e^{a_k}}{\sum_{i=1}^{K} e^{a_i}} \]

- Properties of the softmax function
  (i) \[ 0 \leq y_k \leq 1 \] (iii) differentiable
  (ii) \[ \sum_{k=1}^{K} y_k = 1 \] (iv) \[ y_k \approx P(C_k|x) = \frac{p(x|C_k)P(C_k)}{\sum_{i=1}^{K} p(x|C_i)P(C_i)} \]

- Is the logistic sigmoid function necessary for single-layer single-output-node network?
  No, in terms of classification. (we can replace it with \( g(a) = a \))

- What benefits are there in using the logistic sigmoid function?

- Training of single-layer network
- Training of multi-layer network with 'error back propagation'

Activation functions

- Approximation of posterior probabilities
  - Sigmoid function (for single output node)
  - Softmax function (for multiple output nodes)

A very good reference:
http://neuralnetworksanddeeplearning.com/