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:
  \[ y = g(a) = g\left(\sum_{i=0}^{d} w_i x_i\right) \]
  \[ g(a) = \frac{1}{1 + \exp(-a)} \]

- 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)
  \[ \min_w E(w) \]
Training of single layer neural network

- Optimisation problem: \( \min_w E(w) \)
- No analytic solution (no closed form)
- Employ an iterative method (requires initial values)
e.g. Gradient descent (steepest descent), Newton’s method, Conjugate gradient methods
- Gradient descent

  (scalar rep.)
  \[
  w_i^{(\text{new})} \leftarrow w_i - \eta \frac{\partial}{\partial w_i} E(w), \quad (\eta > 0)
  \]

  (vector rep.)
  \[
  w^{(\text{new})} \leftarrow w - \eta \nabla_w E(w), \quad (\eta > 0)
  \]
Training of the single-layer neural network

\[ E(w) = \frac{1}{2} \sum_{n=1}^{N} (y_n - t_n)^2 = \frac{1}{2} \sum_{n=1}^{N} (g(a_n) - t_n)^2 \]

where \( a_n = \sum_{i=0}^{d} w_i x_{ni} \).

\[ \frac{\partial E(w)}{\partial w_i} = \frac{\partial E(w)}{\partial y_n} \frac{\partial y_n}{\partial a_n} \frac{\partial a_n}{\partial w_i} \]

\[ = \sum_{n=1}^{N} (y_n - t_n) \frac{\partial g(a_n)}{\partial a_n} \frac{\partial a_n}{\partial w_i} \]

\[ = \sum_{n=1}^{N} (y_n - t_n) g'(a_n) x_{ni} \]
Single-layer network with multiple output nodes

- $K$ output nodes: $y_1, \ldots, y_K$

\[ y_{nk} = g \left( \sum_{i=0}^{d} w_{ki} x_{ni} \right) = g(a_{nk}) \]

\[ a_{nk} = \sum_{i=0}^{d} w_{ki} x_{ni} \]
Training set: $D = \{(x_1, t_1), \ldots, (x_N, t_N)\}$

where $t_n = (t_{n1}, \ldots, t_{nK})$ and $t_{nk} \in \{0, 1\}$

Error function:

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \| y_n - t_n \|^2 = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_{nk} - t_{nk})^2$$

$$= \sum_{n=1}^{N} E_n, \quad \text{where} \quad E_n = \frac{1}{2} \sum_{k=1}^{K} (y_{nk} - t_{nk})^2$$

Training by the gradient descent:

$$w_{ki} \leftarrow w_{ki} - \eta \frac{\partial E}{\partial w_{ki}}, \quad (\eta > 0)$$
The derivatives of the error function (single-layer)

\[ E_n = \frac{1}{2} \sum_{k=1}^{K} (y_{nk} - t_{nk})^2 \]

\[ y_{nk} = g(a_{nk}) \]

\[ a_{nk} = \sum_{j=0}^{d} w_{kj} x_{nj} \]

\[ \frac{\partial E_n}{\partial w_{ki}} = \frac{\partial E_n}{\partial y_{nk}} \frac{\partial y_{nk}}{\partial a_{nk}} \frac{\partial a_{nk}}{\partial w_{ki}} \]

\[ = (y_{nk} - t_{nk}) g'(a_{nk}) x_{ni} \]
Multi-layer perceptron (MLP)

- Hidden-to-output weights:
  \[ w_{kj}^{(2)} \leftarrow w_{kj}^{(2)} - \eta \frac{\partial E}{\partial w_{kj}^{(2)}} \]

- Input-to-hidden weights:
  \[ w_{ji}^{(1)} \leftarrow w_{ji}^{(1)} - \eta \frac{\partial E}{\partial w_{ji}^{(1)}} \]
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: 'Neocognitoron'
The derivatives of the error function (two-layers)

\[ E_n = \frac{1}{2} \sum_{k=1}^{K} (y_{nk} - t_{nk})^2 \]

\[ y_{nk} = g(a_{nk}), \quad a_{nk} = \sum_{j=1}^{M} w_{kj}^{(2)} z_{nj} \]

\[ z_{nj} = h(b_{nj}), \quad b_{nj} = \sum_{i=0}^{d} w_{ji}^{(1)} x_{ni} \]

\[ \frac{\partial E_n}{\partial w_{kj}^{(2)}} = \frac{\partial E_n}{\partial y_{nk}} \frac{\partial y_{nk}}{\partial a_{nk}} \frac{\partial a_{nk}}{\partial w_{kj}^{(2)}} \]

\[ = (y_{nk} - t_{nk}) g'(a_{nk}) z_{nj} \]

\[ \frac{\partial E_n}{\partial w_{ji}^{(1)}} = \frac{\partial E_n}{\partial z_{nj}} \frac{\partial z_{nj}}{\partial b_{nj}} \frac{\partial b_{nj}}{\partial w_{ji}^{(1)}} = \left( \sum_{k=1}^{K} (y_{nk} - t_{nk}) \frac{\partial y_{nk}}{\partial z_{nj}} \right) h'(b_{nj}) x_{ni} \]

\[ = \left( \sum_{k=1}^{K} (y_{nk} - t_{nk}) g'(a_{nk}) w_{kj}^{(2)} \right) h'(b_{nj}) x_{ni} \]
Error back propagation

\[
\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \frac{\partial E_n}{\partial y_{nk}} \frac{\partial y_{nk}}{\partial a_{nk}} \frac{\partial a_{nk}}{\partial w_{kj}^{(2)}}
\]

\[
= (y_{nk} - t_{nk}) g'(a_{nk}) z_{nj}
\]

\[
= \delta_{nk}^{(2)} z_{nj}, \quad \delta_{nk}^{(2)} = \frac{\partial E_n}{\partial a_{nk}}
\]

\[
\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \frac{\partial E_n}{\partial z_{nj}} \frac{\partial z_{nj}}{\partial b_{nj}} \frac{\partial b_{nj}}{\partial w_{ji}^{(1)}}
\]

\[
= \left( \sum_{k=1}^{K} (y_{nk} - t_{nk}) g'(a_{nk}) w_{kj}^{(2)} \right) h'(b_{nj}) x_{ni}
\]

\[
= \left( \sum_{k=1}^{K} \delta_{nk}^{(2)} w_{kj}^{(2)} \right) h'(b_{nj}) x_{ni}
\]
Notes on Activation functions

- Interpretation of output values
- Normalisation of the output values
- Other activation functions
Consider a single-layer network with a single output node logistic sigmoid activation function:

$$y = g(a) = \frac{1}{1 + \exp(-a)} = g\left(\sum_{i=0}^{d} w_i x_i\right)$$

Consider a two class problem, with classes $c_1$ and $c_2$. The posterior probability of $c_1$:

$$P(c_1|x) = \frac{p(x|c_1) P(c_1)}{p(x)} = \frac{1}{p(x|c_1) P(c_1) + p(x|c_2) P(c_2)}$$

$$= \frac{1}{1 + \exp\left(-\ln\frac{p(x|c_1) P(c_1)}{p(x|c_2) P(c_2)}\right)}$$
Approximation of posterior probabilities

Logistic sigmoid function

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

Posterior probabilities of two classes with Gaussian distributions:
Normalisation of output nodes

- **Original outputs:**
  \[ y_k = g(a_k), \quad a_k = \sum_{i=0}^{d} w_{ki} x_i \]
  \[ (\sum_{k=1}^{K} y_k) \neq 1 \]

- **Softmax activation function for** \( g() \):
  \[ y_k = \frac{\exp(a_k)}{\sum_{\ell=1}^{K} \exp(a_\ell)} \]

- **Properties of the softmax**
  (i) \( 0 \leq y_k \leq 1 \)
  (ii) \( \sum_{k=1}^{K} y_k = 1 \)
  (iii) \( y_k \approx P(c_k|x) = \frac{p(x|c_k)P(c_k)}{\sum_{\ell=1}^{K} p(x|c_\ell)P(c_\ell)} \)
Some questions on activation functions

- 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?
Online gradient descent

\[ E(w) = \frac{1}{2} \sum_{n=1}^{N} \| y_n - t_n \|^2 = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_{nk} - t_{nk})^2 \]

\[ = \sum_{n=1}^{N} E_n, \quad \text{where} \quad E_n = \frac{1}{2} \sum_{k=1}^{K} (y_{nk} - t_{nk})^2 \]

- Batch gradient descent:
  \[ w_{ki} \leftarrow w_{ki} - \eta \frac{\partial E}{\partial w_{ki}} \]

- Incremental (online) gradient descent:
  Update weights for each \( x_n \)
  \[ w_{ki} \leftarrow w_{ki} - \eta \frac{\partial E_n}{\partial w_{ki}} \]

- Stochastic gradient descent:
  Update weights for randomly chosen \( x \).
Summary

- Training of single-layer network
- Training of multi-layer network with 'error back propagation'
- Activation functions (e.g. softmax)