**Today’s Schedule**

- Single-layer network with a single output node (recap)
- Single-layer network with multiple output nodes
- Multi-layer neural network

**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: \( \min E(w) \)
- 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.)
  \[ w_{i}^{(\text{new})} \leftarrow w_{i} - \eta \frac{\partial E(w)}{\partial w_{i}}, \quad \eta > 0 \]

- Gradient descent (vector rep.)
  \[ w^{(\text{new})} \leftarrow w - \eta \nabla_{w} E(w), \quad \eta > 0 \]

**The derivatives of the error function (single-layer)**

- Error function:
  \[ E(w) = \frac{1}{2} \sum_{n=1}^{N} (y_{n} - t_{n})^{2} \]
  where \( a_{n} = \sum_{i=0}^{d} w_{i} x_{i} \)
  \[ \frac{\partial E(w)}{\partial w_{i}} = \frac{\partial E(w)}{\partial a_{n}} \frac{\partial a_{n}}{\partial w_{i}} = \frac{1}{2} \sum_{n=1}^{N} (y_{n} - t_{n}) \frac{\partial g(a_{n})}{\partial a_{n}} \]

- Error function:
  \[ E(w) = \frac{1}{2} \sum_{n=1}^{N} (y_{n} - t_{n})^{2} \]
  \[ \frac{\partial E(w)}{\partial w_{i}} = \frac{\partial E(w)}{\partial a_{n}} \frac{\partial a_{n}}{\partial w_{i}} = \frac{1}{2} \sum_{n=1}^{N} (y_{n} - t_{n}) g'(a_{n}) x_{i} \]

**Multi-layer neural networks**

- Multi-layer perceptron (MLP)
  \[ w_{ij}^{(2)} \leftarrow w_{ij}^{(2)} - \eta \frac{\partial E(w)}{\partial w_{ij}^{(2)}} \]

- Input-to-hidden weights:
  \[ w_{ij}^{(1)} \leftarrow w_{ij}^{(1)} - \eta \frac{\partial E(w)}{\partial w_{ij}^{(1)}} \]
### 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
- Interpretation of output values
- Normalisation of the output values
- Other activation functions

### Output of logistic sigmoid activation function
- Consider a single-layer network with a single output node logistic sigmoid activation function:
  \[ y = g(a) = \frac{1}{1 + \exp(-a)} = g \left( \sum_{j=1}^{d} w_{j} x_{j} \right) \]
- Consider a two class problem, with classes \( c_1 \) and \( c_2 \). The posterior probability of \( c_1 \):
  \[ P(c_1|x) = \frac{p(x|c_1) P(c_1)}{p(x)} = \frac{p(x|c_1) P(c_1)}{p(x|c_1) P(c_1) + p(x|c_2) P(c_2)} = \frac{1}{1 + \exp(-\ln \frac{p(x|c_1) P(c_1)}{p(x|c_2) P(c_2)})} \]

### Normalisation of output nodes
- Original outputs:
  \[ y_k = g(a_k), a_k = \sum_{i=0}^{d} w_{ki} x_i \]
- Softmax activation function for \( g() \):
  \[ y_k = \frac{\exp(a_k)}{\sum_{i=1}^{K} \exp(a_i)} \]
- Properties of the softmax
  (i) \( 0 \leq y_k \leq 1 \)
  (ii) \( \sum_{k=1}^{K} y_k = 1 \)
  (iii) \( y_k \approx P(c_k|x) = \frac{p(x|c_k) P(c_k)}{\sum_{k=1}^{K} p(x|c_k) P(c_k)} \)

### 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
- \[ E(w) = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_{nk} - t_{nk})^2 = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_{nk} - t_{nk})^2 \]
- Batch gradient descent:
  \[ w_{ki} \leftarrow w_{ki} - \frac{\partial E}{\partial w_{ki}} \]
- Incremental (online) gradient descent:
  Update weights for each \( x_n \)
  \[ w_{ki} \leftarrow w_{ki} - \frac{\partial E}{\partial w_{ki}} \]
- Stochastic gradient descent:
  Update weights for randomly chosen \( x \).
<table>
<thead>
<tr>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Training of single-layer network</td>
</tr>
<tr>
<td>- Training of multi-layer network with 'error back propagation'</td>
</tr>
<tr>
<td>- Activation functions (e.g. softmax)</td>
</tr>
</tbody>
</table>