Machine Learning 1
Lecture 8.1 - Supervised Learning
Neural Networks

Erik Bekkers

(Bishop 5.1)
Fixed Basis Functions

Dataset: inputs $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)^T$ and targets $\mathbf{t} = (t_1, ..., t_N)^T$

Previously:

- Fixed features: $\phi(\mathbf{x}) = (\phi_0(\mathbf{x}), ..., \phi_M(\mathbf{x}))^T$, $\phi_0(\mathbf{x}) = 1$

- Linear regression: $y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x})$, $t_n \in \mathbb{R}$

- Classification: $y(\mathbf{x}, \mathbf{w}) = f(\mathbf{w}^T \phi(\mathbf{x}))$, $t_n \in \{0, 1\}$

  $f$: nonlinear activation function (e.g., logistic sigmoid)
Neural Networks: Adaptive Basis Functions

Dataset: inputs $\mathbf{X} = (x_1, \ldots, x_N)^T$ and targets $\mathbf{t} = (t_1, \ldots, t_N)^T$

Neural networks:

- Create flexible non-linear features and learn them!

\[ \phi_m(\mathbf{x}, \mathbf{w}^{(1)}_m) = h((\mathbf{w}^{(1)}_m)^T \mathbf{x}) = h(\sum_{d=0}^{D} w_{md} x_d) \]

- Regression:

\[ y(\mathbf{x}, \mathbf{W}^{(1)}, \mathbf{w}^{(2)}) = \sum_{m=0}^{M} w_m^{(2)} h(\sum_{d=0}^{D} w_{md} x_d) = \mathbf{w}^{(2)^T} h(\mathbf{W}^{(1)^T} \mathbf{x}) \]

- Classification:

\[ y(\mathbf{x}, \mathbf{W}^{(1)}, \mathbf{w}^{(2)}) = \sigma \left( \mathbf{w}^{(2)^T} h(\mathbf{W}^{(1)^T} \mathbf{x}) \right) \]
Multilayer Perceptron (MLP): 2 layers

Model:

\[ y_k(x, W^{(1)}, W^{(2)}) = h^{(2)} \left( \sum_{m=0}^{M} w^{(2)}_{km} h^{(1)} \left( \sum_{d=0}^{D} w^{(1)}_{md} x_d \right) \right) \]

Network diagram:

Input units: \( x_d \)

Activations: \( a_m \)

Hidden units: \( z_m = h(a_m) \)

Output units: \( y_k \)

Activation functions: \( h^{(1)}, h^{(2)} \)
Activation functions

We need activation functions! Otherwise the NN is just a linear model...

Green: Logistic sigmoid

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

Red: Hyperbolic tan

\[ h(a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \]

Blue:

\[ h(a) = \text{ReLU}(a) = \max(0, a) \]

(Rectified Linear Unit)

Figure: Popular activation functions.
Linear regression & classification as NN

**Figure:** Linear regression as 1-layer NN

\[ y(X, w) = \mathbf{w}^T \mathbf{x} \]

**Figure:** Linear Classification with K classes as 1-layer NN

\[ y_k(X, w) = \text{softmax}(\mathbf{w}_k^T \mathbf{x}) = p(C_k|X) \]
Feed-Forward Networks: Multiple layers

\[ y_k(x, w) = h^{(4)}(a^{(4)}(h^{(3)}(a^{(3)}(h^{(2)}(a^{(2)}(h^{(1)}(a^{(1)}(x)))))))) \]

\[ = h^{(4)} \circ a^{(4)} \circ h^{(3)} \circ a^{(3)} \circ h^{(2)} \circ a^{(2)} \circ h^{(1)} \circ a^{(1)}(x) \]

**Figure:** 4 layer network. Number of layers = number of layers of adaptive weights.
Feed-Forward Networks: Skip Connections

**Figure:** 3 layer feed-forward net with skip connections
Feed-Forward Networks: Sparse Connections

Figure: Feed-forward architecture with sparse connections. With special weight sharing --> Convolutional Neural Nets (Le Cun et al 1989)
Feed-Forward Networks: Sparse Connections

\[ a = x \ast w \]
\[ a_i = \sum_{|i-j|<K} x_j w_{i,j} \]

**Figure:** Feed-forward architecture with sparse connections. With special weight sharing \( \rightarrow \) Convolutional Neural Nets (Le Cun et al 1989)

**Example:** 1D convolution = sparse + weight sharing:

\[
\begin{pmatrix}
  a_1 \\
  a_2 \\
  a_2 \\
  \vdots \\
  a_M
\end{pmatrix}
= \begin{pmatrix}
  w_{11} & w_{12} & w_{13} & w_{14} & w_{15} & \ldots \\
  w_{21} & w_{22} & w_{23} & w_{24} & w_{25} & \ldots \\
  w_{31} & w_{32} & w_{33} & w_{34} & w_{35} & \ldots \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\
  w_{41} & w_{42} & w_{43} & w_{44} & w_{45} & \ldots \\
\end{pmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  \vdots \\
  x_D
\end{pmatrix}
\]
Each unit (hidden & output) in feed-forward architectures computes a function of the form

\[ z_m = h \left( \sum_j w_{mj} z_j \right) \]

No closed directed cycles!