

Classification and Neural Networks

Classification:

Model - A discriminant function is a function that takes an input vector x and assigns it to a class C_k .

Linear Discriminant Function:

$$y(x) = w_0 + w^T x \quad w_0 = \text{bias} \quad x = \text{input vector} \quad w = \text{weights}$$

For $K = 2$ if $y(x) \geq 0$ add to class C_1 else add to class C_2

We can minimise the least squares (error function) to compute the optimal parameters.

This model only works for data that is linearly separable

Logistic Functions:

A logistic function can be used to obtain a probability of being in class C_k . For example:

$$y(x) = \frac{1}{1 + e^{-w^T x}} \quad p(C_1|x) = y(x) \quad p(C_2|x) = 1 - p(C_1|x)$$

When $y \rightarrow 0$ choose class 2 and when $y \rightarrow 1$ choose class 1.

Logistic Regression - Maximum Likelihood Estimation:

$$p(t|x, w) = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1 - t_n} \quad y_n = p(C_1|x_n) \quad t_n \in \{0, 1\} \quad \text{dataset} = \{n_n, t_n\}$$

In order to find the derivative of the negative log of this model we have to use an iterative technique to find a local minimum. We start with random weights and perform gradient descent.

$$\frac{\delta \ln p(t|x, w)}{\delta w} \sum_{n=1}^N (y_n - t_n) x_n = \text{slope}$$

Neural Networks:

Perceptron:

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases} \quad \text{target: } t = \{+1, -1\} \quad \text{minimise: } - \sum_{n=1}^N w^T \phi_n t_n$$



Figure 1: Untitled

Forward Pass:

Compute Activations on hidden layer h:

$$a_j = \sum_{i=1}^D w_{ji}^{(h)} x_i + w_{j0}^{(h)}$$

Pass through a nonlinear function, typically sigmoid:

$$Z_i = \sigma(a_j)$$

Calculate activations on the output layer:

$$a_k = \sum_{i=1}^D w_{ki}^{(h)} Z_i + w_{k0}^{(h)}$$

Then compute predications using the sigmoid function.

Backward pass

Compute the error (cost) function

$$E = \frac{1}{2} \sum_{n=1}^N (y(x_n, w) - t_n)^2$$

Compute the gradients wrt. weights:

$$\text{Output Weights: } \frac{\delta E}{\delta w_{kj}} = \sigma'(y_n - t_n) Z_j$$

Input Weights: $\frac{\delta E}{\delta w_{kj}} = \sigma'(y_n - t_n)w_{kj}^T \sigma' x_i$

We can use these values to update the weights in our neural network:

$$w_{ji} = w_{ji} - \sigma'(y_n - t_n)w_{kj}^T \sigma' x_i \quad w_{kj} = w_{kj} - \sigma'(y_n - t_n)Z_i$$

