Artificial Neural Networks

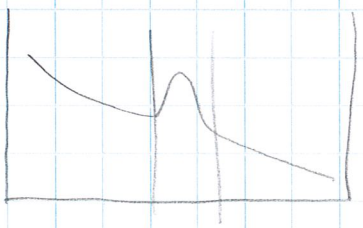
Recall the Linear Fisher Discriminant:

$$f(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

$\vec{w}$ defines a hyperplane that separates signal from background.

$$f(\vec{x}_s) > 0 \qquad f(\vec{x}_b) < 0 .$$

But what if the signal is bounded by two hyperplanes?



Example in one dimension.

Require $f_1(\vec{x}) = \vec{w} \cdot \vec{x} + b_1 > 0$

and $f_2(\vec{x}) = \vec{w} \cdot \vec{x} + b_2 < 0$

or, in general, $f_2(\vec{x}) = \vec{w}_2 \cdot \vec{x} + b_2 > 0$

($\vec{w}_2 = -\vec{w}_1$ for example)

These are combined by forming another function

$g(f_1(\vec{x}), f_2(\vec{x})) > 0$ when

$f_1 > 0$ and $f_2 > 0$

and < 0 otherwise.

This behavior can be facilitated by means of an activation function

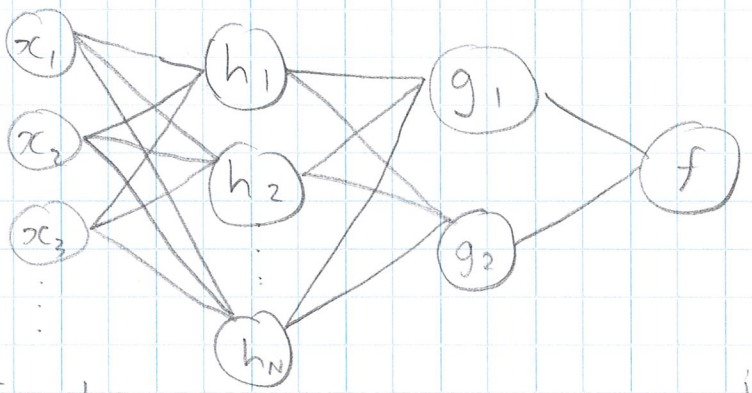
$$K(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

then $$g(K(\vec{w}_1 \cdot \vec{x} + b_1), K(\vec{w}_2 \cdot \vec{x} + b_2))$$

$$= g(y_1, y_2) = y_1 + y_2 - 1$$

This will be 1 when $y_1 + y_2 = 2$ and ≤ 0 otherwise.

In general, an arbitrary set of cuts on multiple hyperplanes can be represented by a network:



$$h_i(\vec{x}) = K(\vec{w}_i \cdot \vec{x} + b_i)$$

each h_i implements one cut on either side of hyperplane $\vec{w}_i$.

K makes the output yes/no.

The hidden layer g_i combine the cuts on individual hyperplanes into a single output.

$$g_i(\vec{y}) = K(\vec{v}_i \cdot \vec{y} + c_i)$$

again, a linear function of the inputs.

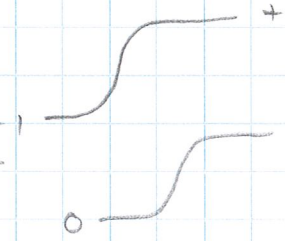
The output node can also be a linear function of its inputs and can produce an output that selects signal and background.

Tuning the weights is a complicated, high-dimensional minimization problem. It is complicated by non-continuous derivatives. Therefore, a smooth activation function is preferable:

$$f(x) = wx + w_0 \qquad \text{linear}$$

$$f(x) = \tanh(wx + w_0)$$

$$f(x) = \frac{1}{1 + \exp(wx + w_0)}$$



These make all derivatives smooth so that a small change in $\vec{w}_i$ gives a small change in the final output.

One training algorithm is backpropagation in which

$$\Delta w_{ij} = r x_i (t_j - y_j)$$

where t_j is the target output for signal / background training sample and y_j is the observed output.

In this case, r is a parameter that controls the 'strength of the "learning".

Overtraining

It is possible to tune the weights to optimally describe the training sample but this might not describe a statistically independent sample.

Check whether this is happening by dividing the training sample into training and validation samples.

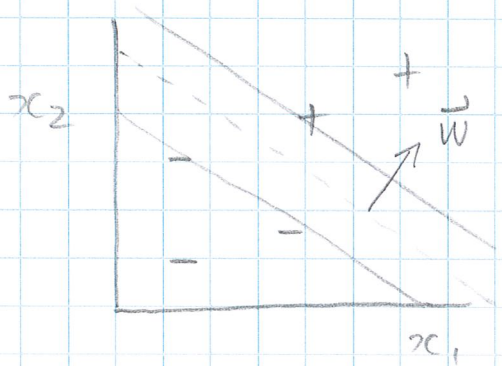
When the separation on the validation samples stops improving, terminate the training process.

Alternatively, regularize the function being minimized by smoothing the input or some other penalization function for overtraining.

Sometimes it can be of some concern whether the solution is a local minimum of the optimization function rather than a global minimum. Various algorithms try to accommodate this (eg simulated annealing).

Support Vector Machines

Separable data:



Many hyperplanes can divide + from − but only one will maximize the margin.

$$y_i(\vec{x}_i \cdot \vec{w} + b) - 1 \geq 0$$

where $y_i = +1$ for +
and $y_i = -1$ for −.

$\vec{w}$ is normalized so that $\vec{w} \cdot \vec{x} + b = +1$ for the + points on the margin and $\vec{w} \cdot \vec{x} + b = -1$ for the − points on the other margin.

The points on the margin are called the support vectors.

This is a constrained quadratic minimization problem.

$$L = \frac{1}{2}|\vec{w}|^2 - \sum_i \alpha_i \left(y_i(\vec{x}_i \cdot \vec{w} + b) - 1 \right)$$

where α_i are Lagrange multipliers.

Minimized by $\alpha_i = 0$ when $\vec{x}_i$ is not a support vector.

$$\frac{\partial L}{\partial \vec{w}} = 0 \quad , \quad \frac{\partial L}{\partial b} = 0$$

Then, maximize

$$L(\vec{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j$$

This is a quadratic minimization problem. A unique minimum exists.

$$\vec{w} = \sum_i \alpha_i y_i \vec{x}_i$$

When the data is not separable we introduce a "slack variable" ξ_i so that $\xi_i = 0$ when classification is correct, and otherwise

$$y_i (\vec{x}_i \cdot \vec{w} + b) - 1 + \xi_i \geq 0$$

$$\xi_i \geq 0.$$

Then introduce the constraint

$$L = \frac{1}{2} |\vec{w}|^2 + C \sum_i \xi_i$$

So there is now a tradeoff between maximizing the margin and increasing misclassification.