Exercises for the lecture „Moderne Methoden der Datenanalyse”

Prof. Dr. S. Hansmann-Menzemer, M. Schiller
Physikalisches Institut der Universität Heidelberg

July 13 2010

Exercise 11: Classification

A problem one often encounters in particle physics is the discrimination of signal and background. Is a certain event signal, of background type 1 or background type 2? What kind of particle is it; pion, kaon or proton? Etc. There are different possibilities to base the decision on: e.g. cut based analysis, the fisher discriminant method, neural networks.

Exercise 11.1

In an experiment two types of events, signal (S) and background (B), are observed. The measured quantity x of signal events follows a Gaussian distribution N with a mean of 1 and a sigma of 1: $x_i^S \in \mathcal{N}(1,1)$. The distribution of background events is given by a Gaussian distribution with mean of 0 and a sigma of 1: $x_i^B \in \mathcal{N}(0,1)$.

Simulate a large number of signal events and the same number of background events and plot their x distribution. Make a decision whether a simulated event is signal or background based on a cut at x_c. Plot the significance α, the power β, the signal efficiency ϵ, the signal purity p and the fraction of wrong decisions as a function of the cut value x_c. Also plot the purity versus the efficiency. Repeat the simulation and the plots with 10 times more background.

Exercise 11.2

The experiment described in exercise 11.1 is extended by the measurement of an additional variable. Each measured event is now a pair $\vec{x} = (x_1, x_2)$.

Simulate n signal events $\vec{x}_i^S \in \mathcal{N}(1,1) \times \mathcal{N}(1,1)$ and n background events $\vec{x}_i^B \in \mathcal{N}(0,1) \times \mathcal{N}(0,1)$. Apply the Fisher discriminant method to separate both classes of events. Plot the Fisher discriminant value t for signal and background and choose a cut value. Make a two dimensional scatter plot of the signal and background events in different colors together with a line indicating the chosen cut.

Exercise 11.3

In a further experiment the background distribution is changed with respect to exercise 11.2. Simulate n signal events $\vec{x}_i^S \in \mathcal{N}(1,1) \times \mathcal{N}(1,1)$, $n/2$ background events $\vec{x}_i^{B1} \in \mathcal{N}(0,1) \times \mathcal{N}(0,1)$, $n/2$ background events $\vec{x}_i^{B2} \in \mathcal{N}(2,1) \times \mathcal{N}(2,1)$ and write the measurement pairs together with a flag for the type of event to a ntuple. You may skip this part of the exercise and take the ntuple from the web page.

Train a neural network to distinguish between both classes. Use the root class `TMultiLayerPerceptron` which is available in root after loading the appropriate library with `gSystem->Load("libMLP.so")`.

Plot the net output value o for signal and background. Make two dimensional scatter plots of the signal and background events and add a contour plot of the neural net output to each. Why is the Fisher discriminant method not suitable for a classification in this case?

Fisher discriminant method

Given is a set of events $\vec{x}^{(1)}$ and $\vec{x}^{(2)}$ of class 1 and class 2, respectively. The covariance matrix of class j is estimated by

$$V_{km}^{(j)} = \frac{1}{N} \sum_N (x_m^{(j)} - \bar{x}_m^{(j)})(x_k^{(j)} - \bar{x}_k^{(j)})$$

with $\bar{\vec{x}}$ being the mean value and N the number of events. Then the Fisher discriminant value for a measurement $\vec{x}$ is defined as:

$$t = \sum_{i=1}^{n} f_i x_i - \frac{1}{2} \sum_{i=1}^{n} f_i (\bar{x}_i^{(1)} + \bar{x}_i^{(2)})$$

with

$$f_i = \sum_k (V^{-1})_{ik} (\bar{x}_k^{(1)} - \bar{x}_k^{(2)}) \quad \text{and} \quad V_{mk} = \frac{1}{2} \left(V_{mk}^{(1)} + V_{mk}^{(2)} \right)$$

Here n is the dimension of the measurement vector.

Artificial Neural Network

Articficial neural networks exploit all the existing information and correlations. They take the possible correlations between variables into account. The most important type is the *multi-layer feed forward network* which we will also use in the exercises.

A multi-layer feed forward network consists of an input layer, an output layer and an arbitrary number of hidden layers (see figure 1). Each input node is assigned to an input variable. The nodes of two consecutive layers are catenated with each other by weighted connections. A weighted sum is calculated from the values of the previous layer. It is given to the transfer function whose value determines the output. Usually, a sigmoid function[1] is used as transfer function. The following output results for the output node i with the input vector $\vec{x}$ for a network with one hidden layer:

$$O_i(\vec{x}) \quad = \quad g\left(\sum_j W_{ij} g\left(\sum_k w_{jk} x_k + \Theta_j\right) + \Theta_i\right) \quad . \tag{1}$$

Here, w_{jk} denotes the weights from the input layer to the hidden one, Θ_j the threshold for the j-th node in the hidden layer, W_{ij} the weights from the hidden to the output layer, Θ_i the threshold of the output node i and g the transfer function. (The thresholds can be implemented by an additional node with a fixed value connected to each node in the consecutive layer, the bias node.) The output values represent discriminators for the different classes. For two classes, one output node is sufficient. The actual information is contained in the weights of the connections. Both these and the thresholds have to be determined in a training. This is made by the use of patterns for which it is known to which class an event belongs to (e.g. in Monte Carlo simulations). The target values of each output node are defined for each class. The weights are adjusted by minimisation of an error function. Widely used functions are the sum-of-squares error function, which is essentially given by summing up the square of the difference between the output and the target value assigned to the pattern considered, and the entropy error function, which is essentially given by the sum of the logarithms of the output values. The exact argument of the logarithm is a function of the target values assigned, but this function always has the advantage that the contribution of a totally wrong classification[2] of a pattern leads to an infinite contribution to the sum. To update the weights in every connection the contribution of each node to the total error has to be calculated. This is done with two formulae consolidated as the *general δ-rule*. From that follows the change in the weights to minimise the total error. This algorithm is called *backpropagation algorithm*.[3]

If the neural network is well trained the output is a measure for the probability that a pattern belongs to a certain class. It can be shown[4] that under

[1] A sigmoid function is kind of a smooth step function with values in the interval $[-1, 1]$ or $[0, 1]$.

[2] e.g. target value is 0 but 1 is assigned.

[3] See D. E. Rumelhart, G. E. Hinton, R. J. Williams, *Learning Internal Representations by Error Propagation*, in D. E. Rumelhart, J. L. McClelland (Hrgb.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition (Vol 1)* , MIT Press (1986). for the exact functionality.

[4] M. D. Richard, R. P. Lippmann, Neural Comp. **3** (1991) 461.

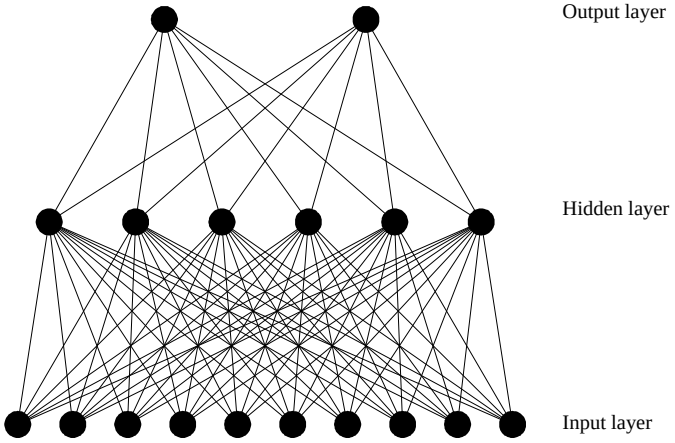

Figure 1: *Draft of a multi-layer feed forward network with ten nodes in the input layer, six nodes in the hidden layer and two output nodes.*

certain conditions the network output corresponds to the a posteriori probability $f(C_i|x)$ that a pattern belongs to the class i for which follows with *Bayes' Rule*:

$$f(C_i|x) = \frac{f(x|C_i) \cdot f(C_i)}{f(x)}. \tag{2}$$

Here, $f(C_i)$ denotes the a priori probability for the class i, $f(x)$ is the class independent probability for the occurance of the pattern x and $f(x|C_i)$ the appropriate conditional probabilities for the class i. This means that the available information contained in the used variables have been optimally utilised. Usually, the target values chosen for the output node are $0/1$ or $-1/1$ for the class $1/2$. Does one get a value near 0 (-1) or 1 then the probability is high that the pattern belongs to the adequate class.

For more information look e.g. in:

- A. Scherer, *Neuronale Netze: Grundlagen und Anwendungen* (in German), Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, Germany, 1997.
- http://www.statsoftinc.com/textbook/stathome.html