Ch.6 Feed-forward neural network models

6.6 Multi-layer perceptron classifier [Book, Sect. 8.1]

The multi-layer perceptron (MLP) NN model can easily be modified for classification problems.

If only 2 classes C_1 and C_2, take the target data y_{d} to be a binary variable, with $y_{\mathrm{d}} = 1$ denoting C_1 and $y_{\mathrm{d}} = 0$ denoting C_2.

Since the output is bounded in classification, instead of using a linear activation function in the output layer, use the logistic sigmoidal function.

With s denoting the logistic sigmoidal activation function, the MLP network with one layer of hidden neurons h_j has the output y given by

$$y = s(\sum_j \tilde{w}_j h_j + \tilde{b}), \quad \text{with } h_j = s(\mathbf{w}_j \cdot \mathbf{x} + b_j)\,, \tag{1}$$

where $\tilde{w}_j$ and $\mathbf{w}_j$ are weights and $\tilde{b}$ and b_j are offset or bias parameters.

The output from a logistic sigmoidal function can be interpreted as a posterior probability $P(C_1|\mathbf{x})$. [Book, Eq.(4.15)]
The logistic regression model is basically the MLP classifier without a hidden layer.

In MLP regression problems, the objective function J minimizes the mean squared error (MSE), i.e.

$$J = \frac{1}{2N} \sum_{n=1}^{N} (y_n - y_{\mathrm{d}n})^2 \,. \tag{2}$$

This objective function can still be used in classification problems, though there is an alternative objective function (the cross entropy function) [Book, Sect. 8.1.1].

To classify the MLP output y as either 0 or 1, we invoke the *indicator function I*, where

$$I(x) = \begin{cases} 1 & \text{if} \quad x > 0\,, \\ 0 & \text{if} \quad x \leq 0\,. \end{cases} \tag{3}$$

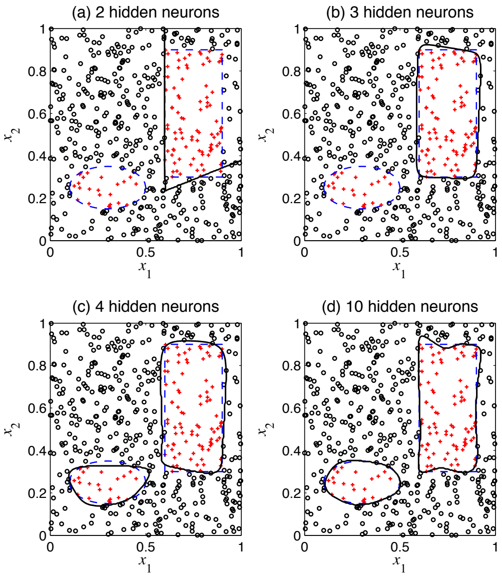
The classification is then given by

$$f(\mathbf{x}) = I[y(\mathbf{x}) - 0.5]. \tag{4}$$

The classification error can be defined by

$$E = \frac{1}{2N} \sum_{n=1}^{N} (f(\mathbf{x}_n) - y_{\mathrm{d}n})^2 \,. \tag{5}$$

Classification of *noiseless* data by MLP with no. of hidden neurons being (a) 2, (b) 3, (c) 4 and (d) 10. The 2 classes of data points are indicated by the pluses and the circles, with the MLP decision boundaries (solid curves) & theoretical boundaries (dashed):


(a) 2 hidden neurons
(b) 3 hidden neurons
(c) 4 hidden neurons
(d) 10 hidden neurons
x_1
x_2
0
0.2
0.4
0.6
0.8
1
0.5

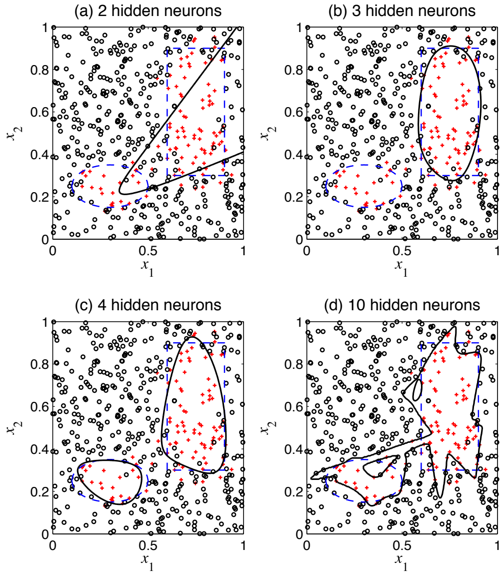


Figure : Classification of *noisy* data by MLP: Overfitting occurred in (d).

6.7 Multi-class classification [Book, Sect. 8.2]

There are now c classes (c an integer > 2). E.g., the temperature is to be classified as warm, normal and cold.

The target data typically use a 1-of-c coding scheme, e.g. warm is $(1, 0, 0)$, normal is $(0, 1, 0)$ and cold is $(0, 0, 1)$.

Use three model outputs, one for each class. If interested in the outputs giving the posterior probability of each class, we will need a generalization of the logistic sigmoidal function.

Since posterior probabilities are non-negative, model non-negative outputs by exponential functions like $\exp(a_k)$ for the kth model output.

Require the posterior probabilities to sum to 1, so the kth model output y_k is

$$y_k = \frac{\exp(a_k)}{\sum_{k'} \exp(a_{k'})}\,, \tag{6}$$

which satisfies $\sum_k y_k = 1$.

This normalized exponential activation function is called the *softmax* activation function. The name softmax comes about because the function is a smooth form of a "maximum" function — e.g. if $a_j \gg a_k$, for all $k \neq j$, then $y_j \approx 1$ and all other $y_k \approx 0$.

Q3: Given three outputs, $a_1 = -1$, $a_2 = 1$ and $a_3 = 5$. What does Eq.(6) give for the values y_1, y_2 and y_3?
Note: Matlab provides a function `softmax`, see
`http://www.mathworks.com/help/nnet/ref/softmax.html`

6.8 Radial basis functions (RBF) [Book, Sect. 4.6]

Besides sigmoidal-shaped activation functions, *radial basis functions (RBF)* involving Gaussian-shaped functions are also commonly used in NN models.

RBF methods originated in the problem of *exact interpolation*, where every input vector is required to be mapped exactly to the corresponding target vector.

First, consider a 1-D target space. The output of the mapping f is a linear combination of basis functions g

$$f(\mathbf{x}) = \sum_{j=1}^{k} w_j \, g(\|\mathbf{x} - \mathbf{c}_j\|, \sigma)\,, \tag{7}$$

where each basis function is specified by its centre $\mathbf{c}_j$ and a width parameter σ.

For exact interpolation, with n observations, there are $k = n$ basis functions to allow for the exact interpolation, and each $\mathbf{c}_j$ corresponds to one of the input data vectors.

Choices for the basis functions — the most common being the Gaussian form

$$g(r, \sigma) = \exp\left(-\frac{r^2}{2\sigma^2}\right) . \qquad (8)$$

In NN applications, exact interpolation is undesirable, as it means an exact fit to noisy data.

Choose $k < n$, i.e. use fewer (often far fewer) basis functions than the number of observations. This prevents an exact fit, but allows a smooth interpolation of the noisy data. By adjusting the number of basis functions used, one can obtain the desired level of closeness of fit. The mapping is now

$$f(\mathbf{x}) = \sum_{j=1}^{k} w_j \, g(\|\mathbf{x} - \mathbf{c}_j\|, \sigma_j) + w_0 \,, \tag{9}$$

where (i) the centres $\mathbf{c}_j$ are no longer given by the input data vectors but are determined during training,
(ii) instead of using a uniform σ for all the basis functions, each basis function has its own width σ_j, determined from training, and
(iii) an offset parameter w_0 has been added.

If the output is multivariate, the mapping generalizes to

$$f_i(\mathbf{x}) = \sum_{j=1}^{k} w_{ji} \, g(\|\mathbf{x} - \mathbf{c}_j\|, \sigma_j) + w_{0i} \,, \tag{10}$$

for the *i*th output variable.

Also common to use *renormalized* (or simply *normalized*) radial basis functions, where the RBF $g(\|\mathbf{x} - \mathbf{c}_j\|, \sigma_j)$ in (9) is replaced by the renormalized version

$$\frac{g(\|\mathbf{x} - \mathbf{c}_j\|, \sigma_j)}{\sum_{m=1}^{k} g(\|\mathbf{x} - \mathbf{c}_m\|, \sigma_m)} . \tag{11}$$

Using RBF can lead to holes, i.e. regions where the basis functions all give little support:

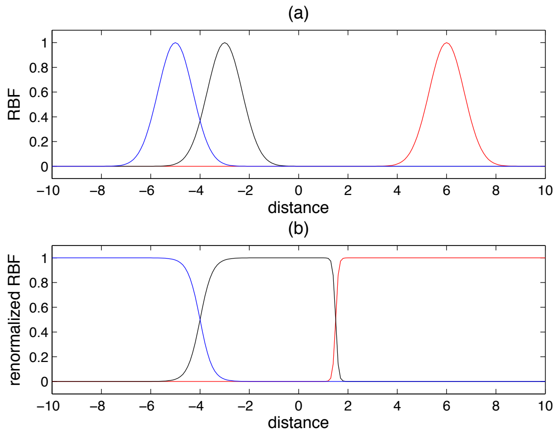


Figure : (a) Radial basis functions (RBFs) and (b) renormalized RBFs. Holes are present in (a), where RBFs with fixed width σ are used. This problem is avoided in (b) with the renormalized RBFs.

While RBF neural networks can be trained like MLP networks by back-propagation (termed adaptive RBFs), RBFs are most commonly used in a non-adaptive manner, i.e. the training is performed in two separate stages:
Stage 1 uses unsupervised learning to find the centres $\mathbf{c}_j$ and widths σ_j of the RBFs;
Stage 2 uses supervised learning via linear least squares to minimize the MSE between the network output and the target data to solve for w_{ji} and w_{0i}.

Procedure of the non-adaptive RBF:
First choose k, the number of RBFs to be used. To find the centres of the RBFs, one commonly uses *K-means clustering*, or self-organizing maps (SOMs).

Next, estimate the width parameters σ_j. For the jth centre $\mathbf{c}_j$, find the distance r_j to the closest neighbouring centre, then set $\sigma_j = \alpha r_j$, with the factor α typically chosen in the range $1 \le \alpha \le 3$.

With the basis functions $g(\|\mathbf{x} - \mathbf{c}_j\|, \sigma_j)$ now determined, only need to find the weights w_{ji} in the equation

$$f_i(\mathbf{x}) = \sum_{j=0}^{k} w_{ji}\, g_j(\mathbf{x})\,, \tag{12}$$

which is the same as (10), with $g_j(\mathbf{x}) = g(\|\mathbf{x} - \mathbf{c}_j\|, \sigma_j)$ $(j = 1, \ldots, k)$, $g_0(\mathbf{x}) = 1$, and the summation starting from $j = 0$ to incorporate the offset parameter w_{0i}.

The network output $f_i(\mathbf{x})$ is to approximate the target data $y_i(\mathbf{x})$ by minimizing the MSE, which is simply a linear least squares problem. In matrix notation, this can be written as

$$\mathbf{Y} = \mathbf{G}\mathbf{W} + \mathbf{E}\,, \tag{13}$$

where $(\mathbf{Y})_{li} = y_i(\mathbf{x}^{(l)})$, (with $l = 1, \dots n$), $(\mathbf{G})_{lj} = g_j(\mathbf{x}^{(l)})$, $(\mathbf{W})_{ji} = w_{ji}$, and $\mathbf{E}$ is the error or residual in the least squares fit.

The linear least squares solution is given by

$$\mathbf{W} = (\mathbf{G}^{\mathrm{T}}\mathbf{G})^{-1}\mathbf{G}^{\mathrm{T}}\mathbf{Y}\,. \tag{14}$$

In summary, the RBF NN is most commonly trained in two distinct stages:

Stage 1 finds the centres and widths of radial basis functions by unsupervised learning of the input data (with no consideration of the output target data).
Stage 2 finds the best linear least squares fit to the output target data (supervised learning).

Supervised learning in MLP requires nonlinear optimization $=>$ multiple minima in the objective function.
The supervised learning in RBF needs only optimization of *linear* least squares $=>$ no multiple minima – advantage of RBF over MLP.

However, basis functions are computed in the RBF NN without considering the output target data. This can be a major drawback, especially when the input dimension is large:

Many of the input variables may have significant variance but have no influence on the output target, yet these irrelevant inputs introduce a large number of basis functions.
The second stage training may involve a very large, poorly conditioned matrix problem — computationally very expensive or even intractable.

6.9 Conditional probability distributions [Book, Sect. 4.7]

In many applications, one is less interested in a single predicted value for y given by $f(\mathbf{x})$ than in $p(y|\mathbf{x})$, a conditional probability distribution of y given $\mathbf{x}$.

With $p(y|\mathbf{x})$, one can easily obtain a single predicted value for y by taking the mean, the median or the mode (i.e. the location of the peak) of the distribution $p(y|\mathbf{x})$. In addition, the distribution provides an estimate of the uncertainty in the predicted value for y.

For managers of utility companies, the forecast that tomorrow's air temperature will be 25°C is far less useful than the same forecast accompanied by the additional information that there will be a 10% chance that the temperature will be higher than 30°C and 10% chance lower than 22°C.

Many types of non-Gaussian distributions are encountered in the environment. E.g. precipitation and wind speed have distributions which are skewed to the right, since one cannot have negative values for precipitation and wind speed.

The gamma distribution and the Weibull distributions have been commonly used to model precipitation and wind speed respectively (Wilks, 1995).

Suppose we have selected an appropriate distribution function, which is governed by some parameters $\boldsymbol{\theta}$. For instance, the gamma distribution is governed by two parameters ($\boldsymbol{\theta} = [c, s]^{\mathrm{T}}$):

$$g(y|c, s) = \frac{1}{Z}\left(\frac{y}{s}\right)^{c-1} \exp\left(-\frac{y}{s}\right), \quad 0 \le y < \infty, \tag{15}$$

where $c > 0$, $s > 0$ and $Z = \Gamma(c)s$, with Γ denoting the gamma function, an extension of the factorial function to a real or complex variable.

The parameters are functions of the inputs $\mathbf{x}$, i.e. $\boldsymbol{\theta} = \boldsymbol{\theta}(\mathbf{x})$. The conditional distribution $p(y|\mathbf{x})$ is now replaced by $p(y|\boldsymbol{\theta}(\mathbf{x}))$.

The functions $\boldsymbol{\theta}(\mathbf{x})$ can be approximated by an NN (e.g. an MLP or an RBF) model, i.e. inputs of the NN are $\mathbf{x}$ while the outputs are $\boldsymbol{\theta}$.

Using NN to model the parameters of a conditional probability density distribution is called a *conditional density network* (CDN) model.

To obtain an objective function, we turn to the principle of *maximum likelihood*: If we have a probability distribution $p(\mathbf{y}|\boldsymbol{\theta})$, and we have observed values $\mathbf{y}_{\mathrm{d}}$ given by the dataset D, then the parameters $\boldsymbol{\theta}$ can be found by maximizing the likelihood function $p(D|\boldsymbol{\theta})$, i.e. the parameters $\boldsymbol{\theta}$ should be chosen so that the likelihood of observing D is maximized.

$p(D|\boldsymbol{\theta})$ is a function of $\boldsymbol{\theta}$ as D is known, and the output $\mathbf{y}$ can be multivariate.

Since the observed data have $n = 1, \ldots, N$ observations, and if we assume independent observations so we can multiply their probabilities together, the likelihood function is then

$$L = p(D|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{y}^{(n)}|\boldsymbol{\theta}^{(n)}) = \prod_{n=1}^{N} p(\mathbf{y}^{(n)}|\boldsymbol{\theta}(\mathbf{x}^{(n)}))\,, \tag{16}$$

where the observed data $\mathbf{y}_{\mathrm{d}}^{(n)}$ are simply written as $\mathbf{y}^{(n)}$, and $\boldsymbol{\theta}(\mathbf{x}^{(n)})$ are determined by the weights $\mathbf{w}$ of the NN model. Hence

$$L = \prod_{n=1}^{N} p(\mathbf{y}^{(n)}|\mathbf{w}, \mathbf{x}^{(n)}). \tag{17}$$

Mathematically, maximizing the likelihood function is equivalent to minimizing the negative logarithm of the likelihood function, since log is a monotonically increasing function.

Choose the objective function to be

$$J = -\ln L = -\sum_{n=1}^{N} \ln p(\mathbf{y}^{(n)}|\mathbf{w}, \mathbf{x}^{(n)}), \tag{18}$$

where we have converted the (natural) logarithm of a product of N terms to a sum of N logarithmic terms.

Since $\mathbf{x}^{(n)}$ and $\mathbf{y}^{(n)}$ are known from the given data, the unknowns $\mathbf{w}$ are optimized to get the minimum J.

Once the weights of the NN model are solved, then for any input $\mathbf{x}$, the NN model outputs $\boldsymbol{\theta}(\mathbf{x})$, which gives the conditional distribution $p(\mathbf{y}|\mathbf{x})$ via $p(\mathbf{y}|\boldsymbol{\theta}(\mathbf{x}))$.

E.g. where $p(y|\boldsymbol{\theta})$ is the gamma distribution (15), need to ensure that the outputs of the NN model satisfy the restriction that both parameters (c and s) of the gamma distribution are positive. Let

$$c = \exp(z_1), \quad s = \exp(z_2), \tag{19}$$

where z_1 and z_2 are the NN model outputs.

6.9.1 Mixture models

The disadvantage of specifying a parametric form for the conditional distribution is that even adjusting the parameters may not lead to a good fit to the observed data.

One way to produce an extremely flexible distribution function is to use a mixture (i.e. a weighted sum) of simple distribution functions to produce a *mixture model*:

$$p(\mathbf{y}|\mathbf{x}) = \sum_{k=1}^{K} a_k(\mathbf{x})\phi_k(\mathbf{y}|\mathbf{x}), \tag{20}$$

where K is the number of components (also called *kernels*) in the mixture, $a_k(\mathbf{x})$ is the (non-negative) *mixing coefficient* and $\phi_k(\mathbf{y}|\mathbf{x})$ the conditional distribution from the kth kernel.

There are many choices for the kernel distribution function ϕ, the most popular choice being the Gaussian function

$$\phi_k(\mathbf{y}|\mathbf{x}) = \frac{1}{(2\pi)^{M/2}\,\sigma_k^M(\mathbf{x})} \exp\left(-\frac{\|\mathbf{y} - \boldsymbol{\mu}_k(\mathbf{x})\|^2}{2\sigma_k^2(\mathbf{x})}\right) , \qquad (21)$$

where the Gaussian kernel function is centred at $\boldsymbol{\mu}_k(\mathbf{x})$ with variance $\sigma_k^2(\mathbf{x})$, and M is the dimension of the output vector $\mathbf{y}$.

With large enough K, and with properly chosen $\boldsymbol{\mu}_k(\mathbf{x})$ and $\sigma_k(\mathbf{x})$, $p(\mathbf{y}|\mathbf{x})$ of any form can be approximated to arbitrary accuracy by the Gaussian mixture model.

An NN model approximates the parameters of the Gaussian mixture model, i.e. $\boldsymbol{\mu}_k(\mathbf{x})$, $\sigma_k(\mathbf{x})$ and $a_k(\mathbf{x})$.

A total of $M \times K$ parameters in $\boldsymbol{\mu}_k(\mathbf{x})$, and K parameters in each of $\sigma_k(\mathbf{x})$ and $a_k(\mathbf{x})$, hence a total of $(M+2)K$ parameters.

Let $\mathbf{z}$ represents the $(M+2)K$ outputs of the NN model. Since there are constraints on $\sigma_k(\mathbf{x})$ and $a_k(\mathbf{x})$, they cannot simply be the direct outputs from the NN model.

As $\sigma_k(\mathbf{x}) > 0$, we need to represent them as

$$\sigma_k = \exp\left(z_k^{(\sigma)}\right) \,, \tag{22}$$

where $z_k^{(\sigma)}$ are the NN model outputs related to the σ parameters.

From the normalization condition

$$\int p(\mathbf{y}|\mathbf{x}) d\mathbf{y} = 1, \tag{23}$$

we obtain, via (20) and (21), the constraints

$$\sum_{k=1}^{K} a_k(\mathbf{x}) = 1, \quad 0 \leq a_k(\mathbf{x}) \leq 1. \tag{24}$$

To satisfy these constraints, $a_k(\mathbf{x})$ is related to the NN output $z_k^{(a)}$ by a *softmax function*, i.e.

$$a_k = \frac{\exp\left(z_k^{(a)}\right)}{\sum_{k'=1}^{K} \exp\left(z_{k'}^{(a)}\right)} \, . \tag{25}$$

Since there are no constraints on $\boldsymbol{\mu}_k(\mathbf{x})$, they can simply be the NN model outputs directly, i.e.

$$\mu_{jk} = z_{jk}^{(\mu)} \, . \tag{26}$$

The objective function for the NN model is again obtained via the likelihood as in (18), with

$$J = -\sum_{n} \ln \left(\sum_{k=1}^{K} a_k(\mathbf{x}^{(n)}) \phi_k(\mathbf{y}^{(n)} | \mathbf{x}^{(n)}) \right) . \tag{27}$$

Once the NN model weights **w** are solved from minimizing J, we get the mixture model parameters $\boldsymbol{\mu}_k(\mathbf{x})$, $\sigma_k(\mathbf{x})$ and $a_k(\mathbf{x})$ from the NN model outputs, and the conditional distribution $p(\mathbf{y}|\mathbf{x})$ via (20).

To get a specific **y** value given **x**, we calculate the mean of the conditional distribution using (20) and (21), yielding

$$\mathrm{E}[\mathbf{y}|\mathbf{x}] = \int \mathbf{y}\, p(\mathbf{y}|\mathbf{x}) d\mathbf{y} = \sum_{k} a_k(\mathbf{x}) \int \mathbf{y}\, \phi_k(\mathbf{y}|\mathbf{x}) d\mathbf{y} = \sum_{k} a_k(\mathbf{x}) \boldsymbol{\mu}_k(\mathbf{x}). \tag{28}$$

Also get the variance of the conditional distribution about the conditional mean using (20), (21) and (28):

$$\begin{aligned} s^2(\mathbf{x}) &= \mathrm{E}\left[\, \|\mathbf{y} - \mathrm{E}[\mathbf{y}|\mathbf{x}]\,\|^2 \mid \mathbf{x}\,\right] \\ &= \sum_k a_k(\mathbf{x}) \left[\sigma_k(\mathbf{x})^2 + \left\| \boldsymbol{\mu}_k(\mathbf{x}) - \sum_{k=1}^{K} a_k(\mathbf{x}) \boldsymbol{\mu}_k(\mathbf{x}) \right\|^2 \right] \end{aligned} \tag{29}$$

Hence the Gaussian mixture model not only gives the conditional distribution $p(\mathbf{y}|\mathbf{x})$, but also conveniently provides, for a given $\mathbf{x}$, a specific estimate for $\mathbf{y}$ and a measure of its uncertainty via the conditional mean (28) and variance (29).

References:

Wilks, D. S. (1995). *Statistical Methods in the Atmospheric Sciences*. Academic Pr., San Diego.