9 Some issues for the projects

You will be occupied with doing your projects during the second half of this course, and you will probably learn the major part of the course at the same time.

This lecture will cover the practical aspects of the project, i.e. what you are required to do and how you should do it.

9.1 The general structure of the project

All the projects are of the following form: You are given a data set $\mathcal{X} = \{\mathbf{x}(n), \mathbf{y}(n)\}_{n=1,\ldots,N}$ and told to construct a model to solve the problem. The problem is typically:

- Create a nonlinear regression model $\hat{y}(\mathbf{x}) = f(\mathbf{x})$, trying and comparing a set of different techniques.
- Create a nonlinear classifier, which classifies the data you have been given.

Your final model (which will be ANN based) will be tested using data that I have kept for myself. You will be required to estimate how well your model generalizes to new data, before I test your model, and I will compare your estimate to the actual result.

Your project is examined in two steps: (1) An oral seminar where you present (in about 20 minutes) the results of your project for the others in the class. (2) A brief written report where you summarize the project, and include all relevant figures.

Your work with the project can be divided into three phases

1. Data exploration: Exploring the relationships in the data, understanding the nature of the problem, exploring different transforms of the input data. (25%)
2. Model construction: Trying different models on the data (linear and nonlinear), model selection, model validation. (55%)
3. Model testing, report writing and project presentation. (20%)

9.1.1 Sample project 1: Estimating meat fat content

Task: Estimate the fat content of a meat sample on the basis of its near infrared absorbance spectrum. For each meat sample the data consists of a 100 channel spectrum of absorbances and the measured contents of moisture (water), fat and protein. The absorbance is the negative logarithm (base 10) of the transmittance measured by the spectrometer. The three contents, measured in %, have been determined by analytic chemistry. The data was recorded on a Tecator Infratec Food and Feed Analyzer working in the wavelength range 850 - 1050 nm by the "Near Infrared Transmission" principle (Tecator AB, Höganäs, Sweden).

Data: There are a total of 215 observations. You will receive 172 of these for training and validation, and 43 are kept for out-of-sample testing. You are given the file *meatfatTrain.mat*, which contains two matrices: *IRspectraTrain*, which is 172×100, and *MoistFatProteinTrain*, which is 172×3. The first matrix contains the IR-spectrum for each meat sample, one row per sample, and the second matrix contains the output values % moisture, % fat, and % protein for each meat sample, one row per sample. You are only required to model the fat content. The spectrum has 100 channels, and the data must first be reduced in dimensionality by transforming to a principal component basis (which is how it is done in the commercial application). This is a linear transformation, and is described e.g. in chapter 8.6 of Bishop's (1995) book and in chapter 8.3 in Haykin's (1999) book.

Steps and subgoals:

1. Work out how to do the principal component analysis (PCA).

2. Get acquainted with the data. Plot it and try different transformations. Try to get a feel for the possible relationship between input and output.

3. Construct a linear model and see how well the fat content can be estimated. That is, estimate the generalization error with a linear model. Optimize the number of principal components, i.e. determine how many and which components you need to get good generalization performance. The function *lsselect* is quite handy for this purpose. It makes sense to report both an MSE and an RMS error. Do a residual analysis (this comment goes for all items).

4. If you think it can be done, construct a polynomial model and see how well this describes the system. This involves trying different orders (up to some reasonable number) and cross-terms.

5. Construct an ANN model (try MLP). The inputs to the ANN model should be the principal components you found to be the optimal for the linear model in item 3. Optimize the number of hidden units/layers with respect to the generalization error.

6. See if you can improve the performance by removing one or more of the principal components.

7. If the answer from item 6 is no, then see if you can improve the result by adding more principal components.

8. Hand in your best ANN model to me (M-file), together with your estimate of the generalization error, so that I can test it on the 43 sample hold-out data set.

Report and presentation of results: You will present the results from your project in two ways: (1) A written report where the main conlusions are presented together with figures and tables supporting your conclusions. (2) An oral presentation, of about 20 minutes, to your course colleagues. The report should be not more than 10 pages, including figures and tables, and should contain the elementary report constituents:

- Introduction (brief presentation of problem, 1 page)
- Methodology (brief listing of methods, 1 page)
- Data (presentation of your data set with important observations, 1-2 pages)
- Results (4-5 pages)
- Discussion (your results and comparison to other researchers' results, 1 page) The report writing should not take much more than one full day, since you are two persons sharing the work.

When you are done with your report, and it has been accepted, then you should produce a postscript file with it, and pack it together with your dataset and other important parts of your project (like MATLAB M-files). The idea being that someone else could unpack it and repeat the main steps in your analysis without rewriting everything.

9.1.2 Sample project 2: Diagnosing Thyroid disease

Task: Tell if a particular set of measurements (test results) comes from a person who is normal, or suffers from being hypothyroid or hyperthyroid (i.e. 3 output categories).

Data: There are 7,200 observations representing patients. You are given 5,000 of these, and 2200 are witheld for out-of-sample testing. There are 21 variables (there is no information on what these represent). You are given the file *thytrain.mat*, which contains the matrices *Ptr* and *Ttr*. The first matrix, *Ptr* (21×5000), contains the input patterns. The second matrix, *Ttr* (3×5000), contains the outputs coded in a "1-out-of-3" fashion. That is, the outputs are coded as (1,0,0), (0,1,0), or (0,0,1).

Steps and subgoals:

1. Get acquainted with the data. Plot it and try to get a feel for the possible relationships between input and output.

2. Construct a quadratic Gaussian classifier for the problem, using all the variables, and estimate the generalization error. Note that the Gaussian classifier assumes that the variables are normally distributed. Thus, you must transform the variables that don't look normally distributed. How normal an empirical distribution is can be determined with e.g. d'Agostinos test, which I will provide to you.

3. Prune the Gaussian classifier by successively removing the variable that results in the least degradation of the generalization error, until the degradation is significant. Note the classification error (generalization).

4. Compute the "Fisher index" (FI) for each input variable and determine which transformations that maximize the FI (this is individual for each variable and does not coincide with the normality transformations above). Exclude all variables that do not give an FI that is significantly larger than that of a purely random variable.

5. Construct a k-nearest-neighbor (kNN) classifier for the problem, using the remaining variables and a Euclidean distance metric, and estimate the generalization error. Optimize k with respect to the generalization error. (Note: You can find a nearest neighbor classifier in e.g. the

NETLAB toolbox, or you can ask me for one, if you don't feel like constructing your own.)

6. Prune the kNN classifier by successively removing variables that don't make the model worse when they are removed, until you get a significant deterioration. Note the classification error (generalization).

7. Construct an ANN model using the remaining inputs. Optimize the number of hidden units (one hidden layer) with respect to the generalization error.

8. Try to prune the ANN model by successively removing the variable that results in the least degradation of the generalization performance, until the degradation is significant. Optimize the number of hidden units for the final model. Note the classification error.

9. Train a few networks with your optimal number of parameters. Combine these into a committee.

10. Hand in your best Gaussian classifier (M-file), your best kNN classifier (M-file), and your ANN committee (M-file), together with your estimate of the generalization classification error, so that I can test it on the hold-out data set.

Report and presentation of results: You will present the results from your project in two ways: (1) A written report where the main conlusions are presented together with figures and tables supporting your conclusions. (2) An oral presentation, of about 20 minutes, to your course colleagues. The report should not be more than 10 pages, including figures and tables, and should contain the elementary report constituents:

- Introduction (brief presentation of problem, 1 page)
- Methodology (brief listing of methods, 1 page)
- Data (presentation of your data set with important observations, 1-2 pages)
- Results (4-5 pages)
- Discussion (your results and comparison to other researchers' results, 1 page) The report writing should not take much more than one full day, since you are three persons sharing the work.

When you are finished with your report, and it has been accepted, then you should produce a postscript file with it, and pack it together with your dataset and other important parts of your project (like MATLAB M-files). The idea being that someone else could unpack it and repeat the main steps in your analysis without rewriting everything.

9.2 Exploring relationships in the data

9.2.1 Scatter plots

A scatter plot is a very useful tool to see structures in data, especially in the case of nonlinear regression problems. You should do scatter plots of:

- The input variable x_k versus the output y.
- The input variable x_k versus the input variable x_j.

The former will tell you if there are strong relationships between a single input x_k and the output. If so, then the input x_k is useful to include in your model (it can of course be useful to include x_k even if there is no strong relationship with y, since it can interact with other input variables). The latter will tell you if two inputs are correlated with each other. If so, then only one of the inputs should be (needs to be) included in the model.

Scatter plots can also tell you what type of relationship there is between e.g. x_k and y. That is, if it is linear, logarithmic, exponential, and so on.

However, it is important to realize that

$$\text{Relationship in scatter plot} \Rightarrow \text{Relationship exists}$$

and not the other way around. A relationship may very well exist between x_k and y without showing up in a scatter plot. For example, suppose that $y = x_1 x_2$, then a scatterplot of x_1 versus y, or of x_2 versus y, may not show any relationship (but a scatterplot of $\log x_1$ versus $\log y$ would show a linear relationship).

Scatter plots can also be done in 3 dimensions and not just in 2 dimensions. It is often useful to color code the data, especially if the problem is a classification problem, to discern structure.

9.2.2 Correlation coefficients

A simple way to find out if there is an approximate linear relationship between x and y is to compute Pearson's correlation coefficient (Cohen 1995). The correlation coefficient[1] is defined as

$$r_{xy} = \frac{\sigma_{xy}}{\sqrt{\sigma_{xx}\sigma_{yy}}}, \tag{9.1}$$

where

$$\sigma_{xx} = \sigma_x^2 = \frac{1}{N-1}\sum_{n=1}^{N}\left[x(n) - \langle x\rangle\right]^2, \tag{9.2}$$

$$\sigma_{yy} = \sigma_y^2 = \frac{1}{N-1}\sum_{n=1}^{N}\left[y(n) - \langle y\rangle\right]^2, \tag{9.3}$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{n=1}^{N}\left[x(n) - \langle x\rangle\right]\left[y(n) - \langle y\rangle\right]. \tag{9.4}$$

We have that

$$\begin{aligned} y = a + bx &\rightarrow r_{xy} = 1, \\ y = a - bx &\rightarrow r_{xy} = -1, \\ y \text{ and } x \text{ linearly independent} &\rightarrow r_{xy} = 0. \end{aligned} \tag{9.5}$$

Thus, if $|r_{xy}|$ is close to one then we can conclude that there is a strong linear relationship between x and y.

The standard deviation for the correlation coefficient between two random series of length N is $1/\sqrt{N}$. This means that if

$$|r_{xy}| > \frac{1.96}{\sqrt{N}} \tag{9.6}$$

then the correlation between x and y is significant at the 95% level.

Pearson's correlation coefficient is available in MATLAB as *corrcoef*.

9.2.3 Spearman's correlation coefficient

In cases where the dependence is not linear, then it is more appropriate to use a so-called non-parametric test (i.e. a test that does not assume the

[1] The name Pearson is often left out since it is the most common form for the correlation coefficient

parameterized linear functional form). Spearman's correlation coefficient is such a non-parametric measure of dependence (Conover 1999). It is very similar to Pearson's correlation coefficient, with the difference that the observations ranks are used instead of their numerical values. Suppose we have N observations of the variable x matched with N observations of the output y. The lowest value of x is then given the rank 1, and the highest value is given the rank N. If two or more values are equal, then they are all given the average rank of the ranks they would have been given if they were not equal. The same goes for y. Consider the following example:

$$\begin{aligned} x(1) &= 0.76 \quad \text{and} \quad y(1) = -0.15 \\ x(2) &= 0.46 \quad \text{and} \quad y(2) = -0.54 \\ x(3) &= 0.46 \quad \text{and} \quad y(3) = -0.54 \\ x(4) &= 0.82 \quad \text{and} \quad y(4) = -0.11 \\ x(5) &= 0.44 \quad \text{and} \quad y(5) = -0.56 \end{aligned}$$

The ranks of these numbers are

$$\begin{aligned} R[x(1)] &= 4 \quad \text{and} \quad R[y(1)] = 4 \\ R[x(2)] &= 2.5 \quad \text{and} \quad R[y(2)] = 2.5 \\ R[x(3)] &= 2.5 \quad \text{and} \quad R[y(3)] = 2.5 \\ R[x(4)] &= 5 \quad \text{and} \quad R[y(4)] = 5 \\ R[x(5)] &= 1 \quad \text{and} \quad R[y(5)] = 1 \end{aligned}$$

The Spearman's correlation coefficient is

$$s_{xy} = \frac{\sum_{n=1}^{N} R[x(n)]R[y(n)] - N\left(\frac{N+1}{2}\right)^2}{\sqrt{\sum_{n=1}^{N} R^2[x(n)] - N\left(\frac{N+1}{2}\right)^2}\sqrt{\sum_{n=1}^{N} R^2[y(n)] - N\left(\frac{N+1}{2}\right)^2}}, \quad (9.7)$$

which equals the result if we would compute Pearson's correlation coefficient r_{xy} using the ranks $R[x]$ and $R[y]$ instead of the values themselves. (It is left as an exercise to the student to show this.)

Spearman's correlation coefficient is available in the MATLAB Toolbox STIXBOX, and called *spearman*.

The standard deviation for Spearman's correlation coefficient between two random series of length N is $1/\sqrt{N}$. This means that if

$$|s_{xy}| > \frac{1.96}{\sqrt{N}} \quad (9.8)$$

then the (non-parametric) correlation between x and y is significant at the 95% level.

9.2.4 Histograms and density plots

Many modeling methods work best if the data is distributed evenly, e.g. normally. If the data is distributed in some other fashion, e.g. following a very skew distribution like the log-normal distribution, then it is often beneficial to transform the data so that it is more normally distributed. This goes for both input and output.

Histograms and density plots are very useful tools to explore the distribution of the data, and also to discover "outliers" in the data (see below).

Other important information that comes from the histogram is, for instance, if the data follows a unimodal or multimodal distribution. Multimodality is important when doing classification, since it may indicate that the problem should be divided into subproblems.

A histogram is drawn in MATLAB using the function *hist*. The STIXBOX Toolbox contains some alternative ways to draw distributions, e.g. *plotdens*, which is a Parzen windows density estimate, and the function *histo*.

9.2.5 The Fisher index

One way to estimate if a variable is useful for classification or not is to use the "Fisher index". The Fisher index (FI) is defined as

$$FI(k) = \frac{(\mu_{k,1} - \mu_{k,2})^2}{(N_1 - 1)\sigma_{k,1}^2 + (N_2 - 1)\sigma_{k,2}^2} \tag{9.9}$$

where the indices 1 and 2 refer to the two categories, respectively, and

$$\mu_{k,1} = \frac{1}{N_1} \sum_{\mathbf{x}(n) \in c_1} x_k(n) \tag{9.10}$$

$$\mu_{k,2} = \frac{1}{N_2} \sum_{\mathbf{x}(n) \in c_2} x_k(n) \tag{9.11}$$

$$\sigma_{k,1}^2 = \frac{1}{N_1 - 1} \sum_{\mathbf{x}(n) \in c_1} [x_k(n) - \mu_{k,1}]^2 \tag{9.12}$$

$$\sigma_{k,2}^2 = \frac{1}{N_2 - 1} \sum_{\mathbf{x}(n) \in c_2} [x_k(n) - \mu_{k,2}]^2 \tag{9.13}$$

are the class means and variances for variable x_k. The FI is a measure of between class spread $(\mu_{k,1} - \mu_{k,2})^2$ in relation to the within class spread

$[(N_1-1)\sigma_{k,1}^2 + (N_2-1)\sigma_{k,2}^2]$. A variable x_k with a high Fisher index is a good variable for classification, because the classes are well separated along x_k.

Again, there is the caveat that

$$\text{High Fisher index} \Rightarrow \text{Good for classification},$$

and that this does not imply that a variable that is good for classification necessarily has a high Fisher index.

9.2.6 Outliers

"Outliers" are observations that deviate significantly from the general pattern of the other observations. This could be due to observations having been miscoded (the "human factor") or a faulty sensor. They could also be perfectly good measurements but examples of something that occurs so rarely that there is only one or two examples of it in the data. In the former case, it is directly harmful to include the wrong observations, because they represent misinformation. In the latter case, it is not wrong to include them but the model may become poorer from including them. This is because there are not enough examples of the effect to build a good model of it, and including the effect may mean a poor compromise for some other measurements.

In general, it is always wise to exclude mysterious observations from the data set. It is also wise to include in your model a technique for detecting observations that are very different from the training data. This is because you will probably do very poorly on observations that are very different from your training data.

9.3 Transforming the data

It can make a huge difference in model performance, and in training time, if the input variables have been properly transformed before being presented to the model.

9.3.1 Regression

The easiest regression models to build are those where there is an approximate linear relationship between the input and the output. One should therefore try to make the problem as linear as possible. One way to go about this is to require that the input variable x_k approximately follows the same distribution as the output variable y. For example, if the output variable is normally distributed then one should also try to make the input variable normally distributed. (It is, in principle, always wise to try make the variables symmetrically distributed, i.e. that the distributions are not skewed.)

A good measure to monitor when trying different transformations, is the correlation coefficient (9.1) between the output and the transformed input. The transformation that gives the maximum absolute correlation coefficient is the transformation that achieves the most linear relationship between input and output. (However, this must be used with caution since outliers may corrupt the situation.)

"Generalized linear models", discussed in lecture 2, are based on an extreme form of this. The idea there is to transform the input variables so that the remaining problem becomes linear. ANN models take an intermediate position. The inputs are nonlinearly transformed into the hidden layer representation, from which the problem is treated linearly (in the regression case). Nevertheless, it is often very useful to help the network model by transforming the variables before they are entered into the neural network.

9.3.2 Classification

For classification, the situation is a little bit different. Here the crucial thing is that the classifier sees the decision boundary region well. One should therefore choose the transformation of input variable x_k that maximizes the separation of the categories, measured e.g. with the Fisher index.

Exceptions to this rule are methods that assume a specific parametric form for the input distribution, like e.g. the Gaussian classifier. In these cases, one should transform the variables so that they follow the assumed distributions (if possible), typically the normal distribution.

One method to test the normality of a distribution, without actually plotting it, is D'Agostino's test (Wetherill 1986).

9.3.3 The Box-Cox family of transformations

The "Box-Cox" transformations are:

$$z_k = \begin{cases} (x_k + \kappa_k)^{\lambda_k} & \text{if } \lambda_k > 0 \\ \ln(x_k + \kappa_k) & \text{if } \lambda_k = 0 \end{cases} \tag{9.14}$$

Here, κ_k is set so that $x_k + \kappa_k > 0$ and λ_k set so that the desired effect is achieved. If $\lambda_k < 1$ then small values of x_k are emphasized over large values, and vice versa if $\lambda_k > 1$.

This family of transformations is sufficient for most purposes. A search for exactly the exponent λ_k that gives e.g. the most normal distribution is often not very efficient. It is sufficient to find a rough value of λ_k that gives a reasonable result.

Note that many measures of dependence, like Pearson's correlation coefficient and the Fisher index, change when the variable is transformed. (However, Spearman's rank correlation coefficient is invariant under all monotonic transformations.)

9.3.4 Normalization

It is, for practical purposes (due to how computers compute), very important that the numbers entering the model during the training are approximately of the same size. This is achieved by appropriate normalization.

Standardization: The simplest form for normalization is "standardization", where all variables (usually also the output y) are standardized to have zero mean and unit variance. That is

$$z_k(n) = \frac{x_k(n) - \mu_k}{\sigma_k} \tag{9.15}$$

where μ_k is the mean value of x_k, and σ_k is the standard deviation of x_k.

This standardization is often sufficient to achieve good results.

For binary variables, the proper standardization is to translate them to $\{-1, +1\}$.

Note: Expression (9.15) can be expressed in matrix language as

$$\mathbf{Z} = \left(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T\right)\boldsymbol{\Lambda}^{-1/2}, \tag{9.16}$$

where $\mathbf{X}$ is the observation matrix introduced earlier, $\mathbf{1}$ is a $(N \times 1)$ vector with ones in all positions

$$\mathbf{1} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \tag{9.17}$$

$\boldsymbol{\mu}$ is a $(D \times 1)$ vector with the mean values of the variables, i.e.

$$\mu_k = \frac{1}{N}\sum_{n=1}^{N} x_k(n), \tag{9.18}$$

and $\boldsymbol{\Lambda}$ is a diagonal matrix with the variances of the variables on the diagonal

$$\boldsymbol{\Lambda} = \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_D^2 \end{pmatrix}. \tag{9.19}$$

Whitening: A little more complicated normalization is the "whitening" procedure. Here, the idea is to decorrelate the input variables by rotating the basis vectors and also standardize them.

The underlying idea is that the covariance matrix $\boldsymbol{\Sigma} = \left(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T\right)^T\left(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T\right)$ is symmetric and can thus be diagonalized using an orthonormal transformation $\mathbf{Q}$ (the spectral theorem). That is, there exists a matrix $\mathbf{Q}$ such that

$$\begin{aligned} \mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T &= \mathbf{I} \\ \mathbf{Q}^T\boldsymbol{\Sigma}\mathbf{Q} &= \boldsymbol{\Lambda} \end{aligned} \tag{9.20}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix (not exactly the same matrix as in the standardization case, but close). From this follows that

$$\left[\left(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T\right)\mathbf{Q}\boldsymbol{\Lambda}^{-1/2}\right]^T\left[\left(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T\right)\mathbf{Q}\boldsymbol{\Lambda}^{-1/2}\right] = \mathbf{I}. \tag{9.21}$$

Thus, if we choose

$$\mathbf{Z} = \left(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T\right)\mathbf{Q}\boldsymbol{\Lambda}^{-1/2} \tag{9.22}$$

then our new variables z_k will be zero mean (we have subtracted the mean), be uncorrelated (their covariance matrix is diagonal), and have unit standard deviation (the diagonal elements are all equal to one). The transformation should be read in the following order: Subtract the mean, rotate the coordinate system, and then rescale.

The transformation (9.22) is the whitening transformation. The name "whitening" comes from the fact that the variables are uncorrelated and with equal "power" in all directions, similar to white noise being uncorrelated and with equal power in all spectral bands.

9.3.5 Principal components

Principal components, see (Bishop 1995) or (Haykin 1999), are close in spirit to the whitening transformation. The idea with principal components is to use only a small subset of the variables z_k, and select a subset of the ... so that as much as possible of the variation in $\mathbf{X}$ is preserved.

The columns $\mathbf{q}_i$ of the matrix $\mathbf{Q}$ in (9.20) are eigenvectors to the covariance matrix $\mathbf{\Sigma}$ and the eigenvalues are the diagonal elements of the matrix $\mathbf{\Lambda}$. Let us suppose that we have sorted the values such that

$$\begin{aligned} \mathbf{\Sigma}\mathbf{q}_i &= \lambda_i \mathbf{q}_i \\ \lambda_1 \geq \lambda_2 \geq \lambda_3 \geq &\cdots \geq \lambda_D \end{aligned} \tag{9.23}$$

(the matrix $\mathbf{\Sigma}$, being a covariance matrix, is positive semidefinite, which means that all $\lambda_i \geq 0$). We then pick the first M eigenvectors and use only them in the transformation from $\mathbf{X}$ to $\mathbf{Z}$. This corresponds to using the first M columns of $\mathbf{Z}$ in (9.22). (The standard principal component transformation does not include the zero mean and unit variance step, but this is anyway done in most cases before the principal components are entered into the model so the result is the same as taking the first columns of $\mathbf{Z}$.)

9.4 Model selection

A central part of the model building process is how to select the best model. This question arises when the ANN architecture is selected, when hyperparameters (e.g. regularization parameters) are set, when the number of input variables are selected, and so on.

The key element in the model selection process is the cross-validation technique for estimating the generalization error.

A typical experiment for selecting the optimal number of hidden units in an ANN model would go as follows:

- The training data is split into K parts (e.g. $K = 10$).
- A set of different ANN architectures are, with e.g. 2, 4, 6, 8 and 10 hidden units, are trained. K networks are trained for each number of hidden units.
- The mean error values (the estimated generalization error) are compared, considering also the standard deviations of the errors, and the smallest network (smallest number of hidden units) that is not significantly worse than the best network architecture is selected.

The cross-validation errors are typically not normally distributed (this is not so surprising since the error is bounded by zero from below but not bounded upwards). Instead, it is often wise to take the logarithm of the errors before the means are compared.

In the example, we end up with 5×10 networks and corresponding test errors $E_{test}(i,k)$. The k index runs from 1 to 10 and denote different training subsets in the cross-validation procedure. The i index runs from 1 to 5 and denotes the different ANN architectures (2, 4, 6, 8 or 10 hidden units). We take the logarithm of the error and compute the mean (since we are dealing with logarithms it will be the geometric mean)

$$\mu_i = \langle \log E_{test}(i) \rangle = \frac{1}{K} \sum_{k=1}^{K} \log E_{test}(i,k) \tag{9.24}$$

and the variance

$$\sigma_i^2 = \langle [\log E_{test}(i)]^2 \rangle - \langle \log E_{test}(i) \rangle^2 = \frac{1}{K-1} \sum_{k=1}^{K} [\log E_{test}(i,k) - \mu_i]^2. \tag{9.25}$$

If we assume that the errors are approximately log-normally distributed then we can use the t-test (Cohen 1995) to see if two architectures are significantly different. The two architectures are significantly different with 95% confidence if

$$|\mu_i - \mu_j| > 1.96 \frac{\sigma_{ij}}{\sqrt{K}}, \tag{9.26}$$

where

$$\sigma_{ij}^2 = \sigma_i^2 + \sigma_j^2. \tag{9.27}$$

It is actually better to use a non-parametric test (the t-test is parametric since it assumes that the values are normally distributed). An example of such a test is the Mann-Whitney/Wilcoxon test (Conover 1999). We will not cover this test in detail here, but an approximate form of it is achieved by doing a t-test on the ranks of the test errors, discussed earlier in connection with Spearman's correlation coefficient.

9.5 Interpretation of the model output

9.5.1 Regression

If we work with a regression model, then the output should be interpreted as the conditional mean

$$\hat{y}(\mathbf{x}) = \int y(\mathbf{x}, \varepsilon) d\varepsilon. \tag{9.28}$$

We assume that the noise process ε has zero mean when we use the summed square error. If this is true, which we can check by looking at the residuals $e(n)$, then we have the following relation between the mean square error (SSE/N) and the noise variance

$$\sigma_\varepsilon^2 = \frac{1}{N-1}\sum_{n=1}^{N} \varepsilon^2(n) \approx \frac{1}{N}\sum_{n=1}^{N} \varepsilon^2(n) = \frac{1}{N}\sum_{n=1}^{N} e^2(n) = \text{MSE}. \tag{9.29}$$

The distribution of the true value: From a maximum likelihood perspective, using the summed square error is equivalent to assuming a Gaussian noise distribution. Thus, if the residuals are normally distributed with zero mean (which we want them to be) then we also have an idea about the distribution of the true value y given the prediction $\hat{y}$.

$$\begin{aligned} p(e) &= \frac{1}{\text{RMS}\sqrt{2\pi}} \exp\left[\frac{-e^2}{2\text{RMS}}\right] \\ \Rightarrow p(y|\hat{y}) &= \frac{1}{\text{RMS}\sqrt{2\pi}} \exp\left[\frac{-(y-\hat{y})^2}{2\text{RMS}}\right] \end{aligned} \tag{9.30}$$

where $\text{RMS} = \sqrt{\text{MSE}}$. That is, if we were able to collect a large quantity of experimental data and compared the prediction to the true value, then we

would expect the true value to be distributed normally around the prediction value. This allows us to give a 95% confidence interval for our prediction, i.e.

$$y(\mathbf{x}) = \hat{y}(\mathbf{x}) \pm 1.96\text{RMS}. \tag{9.31}$$

A consequence of this is that if we have several (two or more) independent models that give different predictions $\hat{y}_k$, and with corresponding MSE values (MSE_k), then we could combine the predictions into a committee

$$\hat{y}_{comm} = \frac{\sum_{k=1}^{K} \hat{y}_k \text{MSE}_k^{-1}}{\sum_{k=1}^{K} \text{MSE}_k^{-1}} \tag{9.32}$$

(it is left to the reader as an exercise to work out that this is indeed the weighting). In the special case that all MSE values are the same (i.e. we have no preference among the models) then we get what is sometimes called a "flat" committee

$$\hat{y}_{comm} = \frac{1}{K} \sum_{k=1}^{K} \hat{y}_k. \tag{9.33}$$

Error bars and confidence tests for the RMS value: Once we have constructed a model that we believe in, then the RMS value we get on our out-of-sample test data will only be a sample estimate of the true RMS value for our model. If the residuals are normally distributed and we have N samples in our out-of-sample test data, then the approximate 95% confidence band for our RMS estimate is, see e.g. (Blom 1980),

$$\left(1 - \frac{1.96}{\sqrt{N}}\right) \cdot \text{RMS} < \text{RMS}_{true} < \left(1 + \frac{1.96}{\sqrt{N}}\right) \cdot \text{RMS}. \tag{9.34}$$

If, on the other hand, we are comparing two models A and B, which have been tested on the same test data, then it is appropriate to use the paired t-test, see e.g. (Conover 1999). The paired t-test relates the statistic

$$t = \frac{\langle d \rangle}{\sqrt{\frac{1}{N(N-1)} \sum_n \left[d(n) - \langle d \rangle\right]^2}} \tag{9.35}$$

to a normal distribution with zero mean and unit variance. Here $d(n) = e_A(n) - e_B(n)$, i.e. the difference between the two models' residuals on the observation n, and $\langle d \rangle$ denotes the average $d(n)$. Thus, if $t > 1.96$ then the two models are significantly different, at the 95% confidence level.

Error bars from cross-validation: An alternative method to compute error bars is to use the cross-validation technique. If we train K models using different training data sets, and test them on K different test data sets, then the estimated MSE error is the average of the individual test errors

$$\mathrm{MSE}_{CV} = \frac{1}{K}\sum_{k=1}^{K}\mathrm{MSE}_k. \tag{9.36}$$

The cross-validation technique also provides a standard deviation

$$\sigma^2_{\mathrm{MSE}} = \frac{1}{K-1}\sum_{k=1}^{K}\left[\mathrm{MSE}_k - \mathrm{MSE}_{CV}\right]^2, \tag{9.37}$$

which also includes the model variation due to different starting points and training sets. This could also be used as an error bar for the MSE value.

9.5.2 Classification

If we work with a classification model, then the output should be interpreted as the conditional probability for one of the classes (which one depends on how we code our target values):

$$\hat{y}(\mathbf{x}) = \hat{p}(c_1|\mathbf{x}). \tag{9.38}$$

Error bars and confidence tests for the classification error: If we have a model that produces the classification error $\hat{R}$ on out-of-sample test data, then this corresponds to a sample estimate of the model's true classification error R. (Note that the classification error is the fraction of patterns that are misclassified by the model and not the MSE.) The situation is analogous to a bowl containing a fraction R red balls, and the rest are white balls. If we pick N balls out of the bowl, i.e. our test data set, what fraction of the balls will be red? The answer is given by the Binomial distribution (provided that there are many more balls in the bowl than what we pick out). On average, R balls will be red, but the standard devation is

$$\sigma_R = \sqrt{\frac{R(1-R)}{N}}. \tag{9.39}$$

Thus, we can say that the true classification error of our model is

$$\hat{R} - 1.96\sqrt{\frac{\hat{R}(1-\hat{R})}{N}} < R < \hat{R} + 1.96\sqrt{\frac{\hat{R}(1-\hat{R})}{N}}. \tag{9.40}$$

Note: This expression is true if we know the standard deviation exactly, which we don't. A correct expression gives a factor larger than 1.96. We ignore this fact here.

If we are comparing two models A and B that have been tested on the same test data, then it is appropriate to use McNemar's test (Conover 1999). If n_A is the number of misclassifications made by model A and not by model B, and n_B is the number of misclassifications made by model B and not by model A, then McNemar's test relates the statistic

$$Z = \frac{n_A - n_B}{\sqrt{n_A + n_B}} \tag{9.41}$$

to a normal distribution with zero mean and unit variance. An alternative form, with so-called "continuity correction", is given by Ripley (Ripley 1996) as

$$Z = \frac{|n_A - n_B| - 1}{\sqrt{n_A + n_B}}. \tag{9.42}$$

Thus, if $Z > 1.96$ then there is a significant difference, with 95% confidence, between model A and model B.

Error bars from cross-validation: It is also possible to use cross-validation to compute error bars for the classification error, similar to the regression case.

References

Bishop, C. M. (1995), *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford.

Blom, G. (1980), *Sannolikhetsteori och statistikteori med tillämpningar*, Studentlitteratur, Lund.

Cohen, P. R. (1995), *Empirical Methods for Artificial Intelligence*, MIT Press, Cambridge, Massachusetts.

Conover, W. J. (1999), *Practical Nonparametric Statistics,* 3^{rd}*ed.*, John Wiley & Sons, Inc., New York.

Haykin, S. (1999), *Neural Networks – A Comprehensive Foundation*, second edn, Prentice Hall Inc., Upper Saddle River, New Jersey.

Ripley, B. D. (1996), *Pattern Recognition and Neural Networks*, Cambridge University Press, Cambridge.

Wetherill, G. B. (1986), *Regression Analysis with Applications*, Monographs on Statistics and Applied Probability, Chapman and Hall, London.