INFO-F-528 Machine learning methods for bioinformatics
Exercise 3: supervised learning

Gianluca Bontempi
Machine Learning Group, Computer Science Department,
Université Libre de Bruxelles

In this exercice we will practice how to build models for regression and classification.

1 Supervised learning

Supervised learning aims to model phenomena characterized by a stochastic dependency between a variable $\mathbf{y}$ called **output** and a set $x = \{x_{\cdot 1}, \ldots, x_{\cdot n}\}$ of real variables called **inputs** on the basis of a training set $D_N = \{\langle x_i, y_i \rangle\}, i = 1, \ldots, N$. The learned model is expected to predict the output using the input variables.

According to the type of output we can distinguish between two types of prediction tasks:

Regression: $\mathbf{y} \in \mathbb{R}$ is a continuous real variable and the dependency is described by

$$\mathbf{y} = f(x_{\cdot 1}, \ldots, x_{\cdot n}) + \mathbf{w}$$

where $\mathbf{w}$ is a random noise with zero mean. Note that this implies that $E[\mathbf{y}|x] = f(x)$.

Classification (or pattern recognition) $\mathbf{y} \in \{c_1, \ldots, c_K\}$ is a categorical random output variable that takes values in a finite set of classes. In this case the dependency is expressed by a conditional probability function

$$\text{Prob}\{\mathbf{y}|x\}$$

Note that in the case of a binary class $\mathbf{y} \in \{0, 1\}$ we have $\text{Prob}\{\mathbf{y} = 1|x\} = E[\mathbf{y}|x]$

Exercise

- Let us consider a regression problem with an input variable $\mathbf{x} \in \mathbb{R}$ and an output $\mathbf{y} \in \mathbb{R}$. By using R, generate a dataset of $N = 100$ samples which is a realization of the dependency

$$\mathbf{y} = \sin x + \mathbf{w}$$

  where $\mathbf{x} \sim \mathcal{U}(-4, 4)$ is uniformly distributed and $\mathbf{w} \sim \mathcal{N}(0, 1)$ is distributed like a standard normal variable. Plot the dataset and the regression function on a bivariate plot.

- Let us consider a regression problem with an input variable $\mathbf{x} \in \mathbb{R}$ and an output $\mathbf{y} \in \mathbb{R}$. By using R, generate a dataset of $N = 200$ samples which is a realization of the dependency

$$\mathbf{y} = \sin^2 x + \mathbf{w}$$

  where $\mathbf{x} \sim \mathcal{N}(1, 2)$ is Gaussian distributed and $\mathbf{w} \sim \mathcal{U}(-1, 1)$ is distributed like a uniform variable. Plot the dataset and the regression function on a bivariate plot.

- Let us consider a regression problem with an input variable $\mathbf{x} \in \mathbb{R}$ and a binary class output $\mathbf{y} \in \{0, 1\}$. By using R, generate a dataset of $N = 200$ samples which is a realization of the dependency

$$\text{Prob}\{\mathbf{y} = 0|x\} = \sin^2 x$$

  where $\mathbf{x} \sim \mathcal{N}(1, 2)$ is Gaussian distributed. Plot the dataset on a bivariate plot.

2 Parametric identification

R uses a special syntax to specify statistical models of the following form

```
model<-model.algo(response ~ predictor, data= learningSet,
structPar, numPar)
```

where `model.algo` designs the algorithm used for fitting a predictive model, `predictor` is the input and `response` is the output. The observations of these two variables are contained in two columns named `predictor` and `response` of the dataframe `learningSet`. Each model algorithm is also characterised by a specific set `structPar` of structural parameters (e.g. related to the complexity, the number of parameters) and a set `numPar` of additional parameters (e.g. related to the initialisation, stopping criteria, maximum number of iterations). The learned model `model` can be used for prediction purposes by using typically the command

```
response.hat <-predict(model, data=testInputSet)
```

where the dataframe `testInputSet` must at least contain a column named `predictor` (i.e. the predictor variable). The output is the vector of predictions `response.hat` returned by the fitted model for the input values specified in `testInputSet`.

For instance the notation

```
model <-lm(y ~x, data= learningSet)
```

is used to fit a linear regression model to a set of observations of the pair of random variables $\mathbf{x}, \mathbf{y}$ stored in the dataframe `learningSet` and to return the parametric model in the variable `model`.

Other commonly used commands for model fitting are:

- `nnet` in `library(nnet)` fits single-hidden-layer neural network
- `rpart` in `library(rpart)` fits a regression tree
- `svm` in `library(e1071)` fits a a support vector machine for classification and regression
- `randomForest` in `library(randomForest)` implements a Random Forest learner
- `lazy` in `library(lazy)` implements a local learning technique
- `naiveBayes` in `library(e1071)` implements a Naive Bayes classifier

2.1 Model formula notation

The symbol `~` is used in R to define the formula of the statistical model or in other terms which variables of the dataframes should be taken as input and outputs. Suppose a dataframe `D` contains the numeric variables `y,x1,x2`. The following formulae can be used to specify different configurations

- `y ~x1` a model with input `x1` and output `y`
- `y ~1+x1+I(x1^2)` a model with output `y` and inputs `x1, x1^2`
- `y ~x1+x2` a multiple regression model with inputs `x1, x2` and output `y`
- `y ~x1*x2` a multiple regression model with inputs `x1, x2, x1*x2` and output `y`
- `y ~.` a multiple regression model of `y` on all the other variables in the dataframe.

2.2 Linear models

A model is called linear if the dependency between the parameters and the output variable is linear though the relation between the input variables and the output variable does not necessarily have to be linear. The command `lm` is used to fit linear models.

Example

The dataset `mtcars` was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models). It contains continuous variables `mpg`, `hp`, `wt` and factors like `cyl`, `am`, and `gear`. If we want to compute the linear regression of `mpg` on `hp` we can use the following code

```
data(mtcars)
names(mtcars)
plot(mtcars$hp,mtcars$mpg)
model<-lm(mpg~hp,data=mtcars)
coeff<-coefficients(model)
mpg.hat<-predict(model,data=mtcars)
plot(mtcars$hp,mtcars$mpg,xlab="input",ylab="output")
lines(mtcars$hp,mpg.hat)
```

This portion of code creates a dataframe, plots the bivariate data, fits a linear model, extracts the coefficients, computes the prediction and shows the linear fitting together with the data plot.

A comprehensive summary of the results of the regression (in terms of model parameters, significativity and quality of the fitting) can be obtained by the command

```
summary(model)
```

Exercises

The script `linPolynomial.R` fits polynomials

$$y = b_0 + b_1 x + \cdots + b_m x^m$$

of different degrees to a dataset of observations of the stochastic dependency

$$\mathbf{y} = \sin(2\pi x) + \mathbf{w}$$

where $\mathbf{w} \sim \mathcal{N}(0, 0.04)$ is normally distributed.

1. For which degree m does the model provide the best generalisation?
2. Modify the script such that the model is a sum of sinusoids

$$y = b_0 + \sum_{i=1}^{m} b_i sin(2\pi i x)$$

where m varies from 1 to 10. Which value of m provides the best generalization capacity?

2.3 Neural networks

The command `nnet` (package `nnet`) fits a feedforward neural network model (non-linear model) with a single hidden layer. The parametric identification is done via the BFGS optimization method of `optim`. Once the neural model has been created, it can be used for predicting the output for new input data by employing the function `predict` (see scripts). Type `help(nnet)` in R for more details. The parameters `x` and `y` correspond to the inputs and the outputs respectively (from the learning base used for creating the model). The other parameters we use are `size` for the number of units in the hidden layer, `maxit` for the maximal number of iterations and `linout` to specify if a neuron's output function is linear or logistic (regression or classification).

Exercises

Test the script `neuralNet.R`. The script shows the model using the `nnet` function for different sizes of hidden layers in the neural network.

1. For how many neurons in the hidden layer do you obtain the best generalization accuracy?
2. Note that the convergence of a neural network depends on the initialization of the network's weights. What happens if the generator's random number seed is changed?

3 Model assessment and selection by cross-validation

The K-fold cross-validation allows the assessment of the MSE estimate in a less biased way than the empirical error (see Section 6.7 of the handbook). It is in particular used for detecting overfitting and to perform model selection.

Exercises

Test the script `neuralNetCV.R`. This script displays the 10-fold cross validation for a given number of neurons in the hidden layer.

- Which model seems to be the best one using cross-validation?
- Does this match the error obtained on the test set?
- Adapt the script in order to compute the leave-one-out error and plot in a diagram the l-o-o error vs. the number of neurons. Can you see an overfitting/underfitting trade-off.