Course on Machine Learning and Data Analysis

Sommersemester 2011

Department of Machine Learning
Institute for Software und Theoretical
Computer Science
Faculty IV, Technical University Berlin
Prof. Dr. Klaus-Robert Müller
Email: klaus-robert.mueller@tu-berlin.de

Foundations of Classification

Deadline: See website.

For this problem set please hand in code as well as written solutions. The code and an electronic version of the written solutions should be submitted to PASS (see the link on the website).

Exercises

Part 1: Implementation

Exercise 1 (5 Points)

Implement cross validation as a general function, which can be used for various methods and objective functions.

```
[ C, opt_param ] = cv(X, y, classifier_handle, { param_name, value_range, ... },
                      nfolds, nrepetitions, loss_function)
```

The arguments have the following definitions:

1. `X` is the $(d \times n)$-Matrix of Data.
2. `y` is a $(1 \times n)$-Vector, which contains the labels or regression targets for every data point.
3. `classifier_handle` is the handle of the classifier to be used. The function `classifier_handle` should have the following signature,

   ```
   C = classifier_handle(X, y, param1, param2, ...)
   ```

   and should train the classifer and parameters on the given traning data `X`, `y` and return the complete information relevant to the application in the Structure `C`.

   In particular the Structure `C` should contain the field `applyfunc`, the handle to the evaluation function `y = feval(C.applyfunc, C, X)`, which applies the classifier `C` to the data `X` and calculates the labels or regression targets `y`. (N.B. The function handle can refer to a local function: implement the evaluation function in this way.)
4. A list consisting of parameters (`param_name`) and the value ranges (`value_range`). Cross validation should be carried out for all the ranges of possible parameters. The sequence of parameters in this list must correspond to their position in the handle `classifier_handle`.
5. `nfolds` is the number of partitions (m in the notes). This parameter should be optional with a standard value of 10.
6. `nrepetitions` is the number of repetitions (r in the notes). This parameter is optional with the standard value 5.
7. `loss_function` is the name of the loss function to be used. It should have the folowing signature: `l = loss_function(X, y_true, y_pred)` where `X` are the Data, `y_true` are the real labels and `y_pred` are the labels calculated. Write a loss function with the name `zero_one_loss`, which returns the classification error as a number between 0 and 1. This parameter should be optional with `'zero_one_loss'` as the default value.

The result of `cv` should be

1. `C`, the classifier trained with the best parameter values.
2. `opt_param`, a structure which contains the best parameter values. For every parameter there should be an attribute of the same name which contains its value.

The function should report the progress of the function on the command line and also give an estimate for the remaining run time. (see `tic` and `toc`).

Exercise 2 (5 Points)

Implement Kernel Ridge Regression

```
C = krr(X, y, kernel, kernelparameter, regularization)
```

The following kernels (with the accompanying parameters should be implemented):

Name	Kernel	Parameter
`linear`	$k(x,z) = \langle x, z\rangle$	(none)
`polynomial`	$k(x,z) = (\langle x, z\rangle + 1)^p$	Degree $p \in \{1,2,3,\ldots\}$
`gaussian`	$k(x,z) = \exp(-\|x-z\|^2/2dw^2)$	Kernel Width w

Here d is the dimension of X.

The Parameter `regularization` is the regularization constant, c in $\hat{\alpha} = (K + cI)^{-1}y$. If `regularization` is zero, execute Leave-One-Out cross validation on c efficiently. Use the eigenvalues of the kernel matrix K as candidates for c

Use the *noisy sinc function* (see homepage) to test the function.

Part 2: Applications

Exercise 3 (5 Points)

Write a function:

```
roc_curve(n)
```

which plots the ROC-curve of the linear classifier f_{x_0} (see script) for the one dimensional binary classification problem with Gaussian classes and identical class priors:

$$\begin{aligned} p(x|y=-1) &\sim \mathcal{N}(\mu = 0, \sigma^2 = 1) \\ p(x|y=+1) &\sim \mathcal{N}(\mu = 2, \sigma^2 = 1) \\ p(y=-1) &= 0.5 \\ p(y=+1) &= 0.5 \end{aligned}$$

where the linear classifier f_{x_0} is given as

$$f_{x_0}(x) = \begin{cases} -1 & : \quad x \leq x_0 \\ +1 & : \quad x > x_0 \end{cases}$$

The function `roc_curve` should set up 2 subplots in a figure:

1. The analytically calculated ROC-Curve. Use the probability density functions given above.
2. The ROC curve found using simulation with `n` datapoints. For this draw samples as data from the given distributions.

Plots, titles and axes labels should be supplied.

Exercise 4 (5 Points)

The various Kernel-Ridge Regression data sets can be found on our homepage. Use the algorithms on each of the datasets and fnd the best classifier using cross validation. For each result return the best classier (as per the structure `C`) and the aforementioned labels on the test set. For this construct a structure `results` which has the following structure:

```
result.krr.banana_predictor         % der gelernte Predictor für KRR
result.krr.banana_predicted_labels  % die Label auf dem Testdatensatz
result.krr.diabetis_predictor       % dasselbe für den Diabetis-Datensatz
result.krr.diabetis_predicted_labels
```

and save the structure `results` in the file `results.mat`. Plot the resulting ROC-curves for Kernel-Ridge Regression resulting from variation of the bias term.