Exercise Session 1: Classification

Dries Hendrikx* Michaël Fanuel† Francesco Tonin‡ Yingyi Chen§
Johan Suykens

Academic Year 2019-2020

This document describes a step-by-step guide for the application of support vector machine methods. The goal of the exercise sessions is to provide you with experience and teach you how to tackle future application problems. The MATLAB scripts and toolboxes, the course documents, the referred papers and all the datasets used in the exercise sessions are available for academic purposes on `https://toledo.kuleuven.be`.

- The exam consists of an oral discussion on the basis of the reports written for the 3 exercise sessions. It is important to show that you have understanding about the problem and that you can work in a constructive manner towards getting good solutions to given problems.
- The reports have to be written *individually*.
- The reports contain the solutions to the exercises *and* the homework problems. All the questions that need to be answered in the report are indicated by the grey box, which reads 'Questions'. We do not expect minimal answers to the questions but rather 'comprehensive' answers.
- A good report finds a good balance between *visuals* on the one hand and to-the-point *explanations* on the other hand. Use figures and tables to explain things, rather than only long and elaborate sentences. Make sure to include a few key equations in the textual part.
- Both the text and figures should not be too small. They have to be readable.
- Write a *separate* report for every exercise session. At the end of the semester, you have to submit a pdf document of **30 pages maximum** consisting of 3 separate reports in single column (max 10 pages for each session).

*dries.hendrikx@esat.kuleuven.be
†michael.fanuel@esat.kuleuven.be
‡francesco.tonin@esat.kuleuven.be
§yingyi.chen@esat.kuleuven.be

1 Exercises

1.1 A simple example: two Gaussians

To get an intuitive idea what classification is about, the exercises start with a simple toy example. An artificial dataset is constructed from two classes of Gaussians with the same covariance matrices:

```
>> X1 = randn(50,2) + 1;
>> X2 = randn(51,2) - 1;
```

Note that ';' avoids the content of the variables to be written out on the screen. The corresponding class labels are defined:

```
>> Y1 = ones(50,1);
>> Y2 = -ones(51,1);
```

To see the content of the variables, type:

```
>> X1
```

Leaving out the ';' at the end of the statement allows for the variable to written out on the screen. Alternatively, you can type:

```
>> disp(X1);
```

The dataset can be visualized using the following series of commands:

```
>> figure;
>> hold on;
>> plot(X1(:,1), X1(:,2), 'ro');
>> plot(X2(:,1), X2(:,2), 'bo');
>> hold off;
```

where `'ro'` and `'bo'` define respectively the labels of the data of the positive class to be red circles and the labels of the data of the negative class to be blue circles. The `hold on` command allows to display multiple items within one figure.

Questions

- Obtain a line to classify the data by using what you know about the distributions of the data. In which sense is it optimal?

1.2 Support vector machine classifier

Go to https://cs.stanford.edu/people/karpathy/svmjs/demo/. On this website, you find an online application which lets you experiment with support vector machine (SVM) classification. Create a dataset by adding more data points to the existing dataset (unfortunately, you can not remove these 'starting points'). Make sure to have at least 10 data points for each class. Try out both the linear and the RBF kernel for classification.

- For each kernel, answer the following questions:
  - What do you observe when you add more data points to the dataset - both on the right and on the wrong side of the hyperplane. How does it affect the classification hyperplane?

  - Try out different values of the regularization hyperparameter `C` and the kernel parameter `sigma`. What is the role of the parameters? How do these parameters affect the classification outcome?

- Compare classification using the linear kernel with classification using the RBF kernel. Which performs better? Why?

Questions

- What is a support vector? When does a particular datapoint become a support vector? When does the importance of the support vector change? Illustrate visually. Note that a support vector is indicated by a large circle with bold-lined circumference and that its importance is proportional to the size in the online application.
- What is the role of parameters `C` and `sigma`? What happens to the classification boundary if you change these parameters. Illustrate visually.
- What happens to the classification boundary when `sigma` is taken very large? Why?

1.3 Least-squares support vector machine classifier

We proceed with the least squares based variant of the support vector machine (LS-SVM), using the LS-SVMlab toolbox. More information on the use of the toolbox can be found in the introduction guide on Toledo. Before we start experimenting with the toolbox ourselves, let's first have a look at a demo by typing

```
>> democlass
```

In this exercise, we investigate the `iris` dataset (available as `iris.mat` on Toledo). The data points are two-dimensional and indicate the length and the width of the sepals of the iris flower. Since the data points are two-dimensional, we can visualize our resulting classifier. In addition, the classification problem is binary: two different types of iris plants. We start by loading the dataset:

```
>> load iris.mat
```

Four variables appear in the workspace: `Xtrain`, `Ytrain`, `Xtest` and `Ytest`. `X` variables are used to indicate the data points, while `Y` variables denote class labels.

1.3.1 Influence of hyperparameters and kernel parameters

The hyperparameter (regularization constant γ) and kernel parameters highly influence the classification model.

Questions

- Try out a polynomial kernel with `degree` $= 1, 2, 3, \ldots$ and `t` $= 1$ (fix `gam` $= 1$). Assess the performance on the test set. What happens when you change the degree of the polynomial kernel?
- Let's now focus on the RBF kernel with squared kernel bandwidth σ^2.

  - Try out a good range of different `sig2` values as kernel parameters (fix `gam` = 1). Assess the performance on the test set. What is a good range for `sig2`?
  - Fix a reasonable choice for the `sig2` parameter and compare the performance using a range of `gam`. What is a good range for `gam`?

- Compare your results with `SampleScript_iris.m`, which is available on Toledo.

1.3.2 Tuning parameters using validation

The intuition developed in the previous section is now used to motivate automated tuning algorithms. In general, automatic tuning algorithms further split up the training data into a *training* and a *validation* part. This can be done in multiple ways:

- **Random split**. One way of splitting up the training dataset into a training and a validation part is by randomly taking some data points for training. The remaining data points are consequently used for validation. This can be implemented as:

```
>> perf = rsplitvalidate({Xtrain, Ytrain, 'c', gam, sig2,
    'RBF_kernel'}, 0.80, 'misclass');
```

  where the parameter value `0.80` indicates that 80 percent of the training data is used for *training* (thus, 20 percent of the training data is used for *validation*).

- ***k*-fold crossvalidation**. Another validation method is k-fold crossvalidation. In general, k is often taken to be equal to 10. This can be implemented as:

```
>> perf = crossvalidate({Xtrain, Ytrain, 'c', gam, sig2,
    'RBF_kernel'}, 10, 'misclass');
```

  where the parameter value `10` indicates that 10 folds are used in the crossvalidation procedure.

- **Leave-one-out validation**. Leave-one-out validation is a special case of k-fold crossvalidation, where k is taken equal as the number of data points. This can be implemented as:

```
>> perf = leaveoneout({Xtrain, Ytrain, 'c', gam, sig2,
    'RBF_kernel'}, 'misclass');
```

Questions

- Compute the performance for a range of `gam` and `sig2` values (e.g., $\gamma, \sigma^2 = 10^{-3}$, $\ldots$, 10^3). Use the random split method, 10-fold crossvalidation and leave-one-out validation. Visualize the results of each method: do you observe differences? Interpret the results: which values of `gam` and `sig2` would you choose?

- Why should one prefer crossvalidation over simple validation (random split)? How to choose the value of k in k-fold crossvalidation?

1.3.3 Automatic parameter tuning

The tuning procedure is fully automized in the LS-SVMlab toolbox. Type:

```
>> [gam,sig2,cost] = tunelssvm({Xtrain, Ytrain, 'c', [], [],
    'RBF_kernel'}, 'algorithm', 'crossvalidatelssvm',{10, 'misclass'});
```

where `'algorithm'` can be chosen as `'simplex'` (Nelder-Mead method) or `'gridsearch'` (brute force gridsearch).

Questions

- Try out the different `'algorithm'`. What differences do you observe? Why do the obtained hyperparameters differ a lot in different runs? What about the `cost`? Computational speed? Explain the results.

1.3.4 Using ROC curves

In practice, an alternative way to judge a classifier is by using the Receiver Operating Characteristic (ROC) curve of a binary classifier (see `>> help roc`). The main performance measure derived from such a curve is the area under the curve: the better the classifier, the higher the area under the curve. A ROC plot can be generated as:

```
>> % Train the classification model.
>> [alpha, b] = trainlssvm({Xtrain, Ytrain, 'c', gam, sig2,
    'RBF_kernel'});

>> % Classification of the test data.
>> [Yest, Ylatent] = simlssvm({Xtrain, Ytrain, 'c', gam, sig2,
    'RBF_kernel'}, {alpha, b}, Xtest);

>> % Generating the ROC curve.
>> roc(Ylatent, Ytest);
```

Questions

- In practice, we compute the ROC curve on the test set, rather than on the training set. Why?
- Generate the ROC curve for the `iris.mat` dataset (use tuned `gam` and `sig2` values). Interpret the result.

1.3.5 Bayesian framework

Using the Bayesian framework, it is possible to get probability estimates. Use a tuned pair of parameters (`gam`, `sig2`). Now, the probabilities that a certain data point belongs to the positive class given the model is computed by:

```
>> bay_modoutClass({Xtrain, Ytrain, 'c', gam, sig2}, 'figure');
```

Questions

- How do you interpret the colors of the plot? Hint: activate the colorbar of the figure.

- Change the values of `gam` and `sig2`. Visualize and discuss the influence of the parameters on the figure.

2 Homework problems

Illustrate your skills on support vector machine learning of synthetic and real-life datasets:

1. the Ripley dataset (`ripley.mat`),
2. the Wisconsin Breast Cancer dataset (`breast.mat`) and
3. the Diabetes dataset (`diabetes.mat`).

All datasets are available on Toledo. More information regarding these datasets can be found on the UCI Machine Learning Repository: `https://archive.ics.uci.edu/ml/index.php`. For each of these datasets, answer the following questions related to LS-SVM classification.

Questions

- Visualize the data. Inspect the data structure: what seems to be important properties of the data? Which classification model do you think you need, based on the complexity of the data?
- Try out different models (linear, polynomial, RBF kernel) with tuned hyperparameter and kernel parameters. Compute the ROC curves. Which model performs best? Which model would you choose?
- Are you satisfied with the performance of your model? Would you advise another methodology?

Exercise Session 2: Function Estimation and Time Series Prediction

Dries Hendrikx* Michaël Fanuel† Francesco Tonin‡ Yingyi Chen§
Johan Suykens

Academic Year 2019-2020

This document describes a step-by-step guide for the application of support vector machine methods. The goal of the exercise sessions is to provide you with experience and teach you how to tackle future application problems. The MATLAB scripts and toolboxes, the course documents, the referred papers and all the datasets used in the exercise sessions are available for academic purposes on `https://toledo.kuleuven.be`.

- The exam consists of an oral discussion on the basis of the reports written for the 3 exercise sessions. It is important to show that you have understanding about the problem and that you can work in a constructive manner towards getting good solutions to given problems.
- The reports have to be written *individually*.
- The reports contain the solutions to the exercises *and* the homework problems. All the questions that need to be answered in the report are indicated by the grey box, which reads 'Questions'. We do not expect minimal answers to the questions but rather 'comprehensive' answers.
- A good report finds a good balance between *visuals* on the one hand and to-the-point *explanations* on the other hand. Use figures and tables to explain things, rather than only long and elaborate sentences. Make sure to include a few key equations in the textual part.
- Both the text and figures should not be too small. They have to be readable.
- Write a *separate* report for every exercise session. At the end of the semester, you have to submit a pdf document of **30 pages maximum** consisting of 3 separate reports in single column (max 10 pages for each session).

*dries.hendrikx@esat.kuleuven.be
†michael.fanuel@esat.kuleuven.be
‡francesco.tonin@esat.kuleuven.be
§yingyi.chen@esat.kuleuven.be

1 Exercises

1.1 Support vector machine for function estimation

In this exercise the support vector machine (SVM) toolbox is used for regression. You can find it on Toledo (`svm_toolbox_1.zip`). Download this toolbox and add it to your MATLAB path. Consequently, execute:

```
>> uiregress
```

Executing this script opens a GUI, which allows you to create datasets and do SVM regression on them. In order to add data points, click `Data`, which creates a cursor that can be used to define the locations of the data points. A left mouse click adds a data point, a right mouse click removes the data cursor. Remove the data cursor before clicking `Regress`, otherwise the GUI will add an additional data point under the `Regress` button.

Questions

- Construct a dataset where a linear kernel is better than any other kernel (around 20 data points). What is the influence of `e` (try small values such as $0.10, 0.25, 0.50, \ldots$) and of `Bound` (try larger increments such as $0.01, 0.10, 1, 10, 100$). Where does the sparsity property come in?
- Construct a more challenging dataset (around 20 data points). Which kernel is best suited for your dataset? Motivate why.
- In what respect is SVM regression different from a classical least squares fit?

1.2 A simple example: the sinc function

We proceed with the least squares based variant of the support vector machine (LS-SVM), using the LS-SVMlab toolbox. More information on the use of the toolbox can be found in the general introduction on Toledo. Before we start experimenting with the toolbox ourselves, let's first have a look at a demo by typing

```
>> demofun
```

Go through this introductory example in order to gain more feeling with the syntax and the possibilities of the LS-SVMlab toolbox.

1.2.1 Regression of the sinc function

To get an intuitive idea what function estimation is about, the exercises start with a simple toy example. An artificial dataset is constructed from the sinc function (with white noise):

```
>> X = (-3:0.01:3)';
>> Y = sinc(X) + 0.1.*randn(length(X), 1);
```

Consequently, we create the training and the test sets:

```
>> Xtrain = X(1:2:end);
>> Ytrain = Y(1:2:end);
>> Xtest = X(2:2:end);
>> Ytest = Y(2:2:end);
```

In this exercise, we construct a LS-SVM regression model with the RBF kernel.

Questions

- Try out a range of different `gam` and `sig2` parameter values (e.g., `gam` $= 10, 10^3, 10^6$ and `sig2` $= 0.01, 1, 100$) and visualize the resulting function estimation on the test set data points. Discuss the resulting function estimation. Report the mean squared error for every combination (`gam`, `sig2`).
- Do you think there is one optimal pair of hyperparameters? Argument why (not).
- Tune the `gam` and `sig2` parameters using the `tunelssvm` procedure. Use multiple runs: what can you say about the hyperparameters and the results? Use both the `simplex` and `gridsearch` algorithms and report differences.

1.2.2 Application of the Bayesian framework

In addition to the approach outlined above, the Bayesian framework can also be used to tune and to analyze the LS-SVM regressor. The basic result from the Bayesian framework for the LS-SVM is the derivation of the probability that the data points are generated by the given model. This is called the posterior probability. This probability criterion is expressed as a number. There are 3 variants: the posterior with respect to the model parameters `alpha` and `b`, the posterior with respect to the regularization constant `gam` and the posterior with respect to the choice of the kernel and its parameter `sig2`. The cost (negative logarithm of the posteriors) is computed by the function call

```
>> sig2     = 0.4;
>> gam      = 10;
>> crit_L1 = bay_lssvm({Xtrain, Ytrain, 'f', gam, sig2}, 1);
>> crit_L2 = bay_lssvm({Xtrain, Ytrain, 'f', gam, sig2}, 2);
>> crit_L3 = bay_lssvm({Xtrain, Ytrain, 'f', gam, sig2}, 3);
```

The model can be optimized with respect to these criteria:

```
>> [~,alpha,b] = bay_optimize({Xtrain, Ytrain, 'f', gam, sig2}, 1);
>> [~,gam]     = bay_optimize({Xtrain, Ytrain, 'f', gam, sig2}, 2);
>> [~,sig2]    = bay_optimize({Xtrain, Ytrain, 'f', gam, sig2}, 3);
```

For regression, the error bars can be computed using Bayesian inference using

```
>> sig2e = bay_errorbar({Xtrain, Ytrain, 'f', gam, sig2}, 'figure');
```

Questions

- Discuss in a schematic way how parameter tuning works using the Bayesian framework. Illustrate this scheme by interpreting the function calls denoted above.

1.3 Automatic Relevance Determination

In addition to parameter tuning, the Bayesian framework can also be used to select the most relevant inputs by Automatic Relevance Determination (ARD). The following procedure uses this criterion for backward selection for a three dimensional input selection task, constructed as (use tuned `gam` and `sig2` parameter values):

```
>> X = 6.*rand(100, 3) - 3;
>> Y = sinc(X(:,1)) + 0.1.*randn(100,1);
>> [selected, ranking] = bay_lssvmARD({X, Y, 'f', gam, sig2});
```

Questions

- Visualize the results in a simple figure.
- How can you do input selection in a similar way using the `crossvalidate` function instead of the Bayesian framework?

1.4 Robust regression

In situations where the data is corrupted with non-Gaussian noise or outliers, it becomes important to incorporate robustness into the estimation. Consider the following simple example:

```
>> X = (-6:0.2:6)';
>> Y = sinc(X) + 0.1.*rand(size(X));
```

Outliers can be added via:

```
>> out = [15 17 19];
>> Y(out) = 0.7+0.3*rand(size(out));
>> out = [41 44 46];
>> Y(out) = 1.5+0.2*rand(size(out));
```

Let's say we first train a LS-SVM regressor model, without giving special attention to the outliers. We can implement this as:

```
>> model   = initlssvm(X, Y, 'f', [], [], 'RBF_kernel');
>> costFun = 'crossvalidatelssvm';
>> model   = tunelssvm(model, 'simplex', costFun, {10, 'mse';});
>> plotlssvm(model);
```

We explicitly write out the code here, since the object oriented notation is used.

If we were to train a robust LS-SVM model, using robust crossvalidation, we can implement this as:

```
>> model   = initlssvm(X, Y, 'f', [], [], 'RBF_kernel');
>> costFun = 'rcrossvalidatelssvm';
>> wFun    = 'whuber';
>> model   = tunelssvm(model, 'simplex', costFun, {10, 'mae';}, wFun);
>> model   = robustlssvm(model);
>> plotlssvm(model);
```

Train and plot a LS-SVM regression model, without giving special attention to the outliers: tune the parameters (using `tunelssvm`), train the regressor model (using `trainlssvm`) and plot the result (using `plotlssvm`). What is the influence of the outlying points?

Questions

- Visualize and discuss the results. Compare the non-robust version with the robust version. Do you spot any differences?

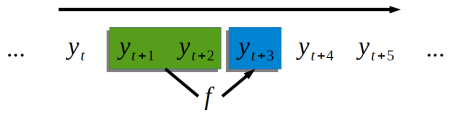

Figure 1: Schematic representation of the nonlinear auto-regressive model.

- Why in this case is the mean absolute error (`'mae'`) preferred over the classical mean squared error (`'mse'`)?
- Try alternatives to the weighting function `wFun` (e.g., `'whampel'`, `'wlogistic'` and `'wmyriad'`. Report on differences. Check the user's guide of LS-SVMlab for more information.

2 Homework problems

2.1 Introduction: time series prediction

The linear autoregressive (AR) model of a process Z_t with $t = 1, 2, \ldots$ is given by

$$\hat{z}_t = a_1 z_{t-1} + a_2 z_{t-2} + \cdots + a_n z_{t-n} \tag{1}$$

with $a_i \in \mathbb{R}$ for $i = 1, \ldots, n$ and n the model order. At the same time, the nonlinear variant (NAR) is desscribed as:

$$\hat{z}_t = f(z_{t-1}, z_{t-2}, \ldots, z_{t-n}), \tag{2}$$

A depiction of these processes can be found in figure 1. Time series identification can be written as a classical black-box regression modeling problem:

$$\hat{y}_t = f(x_t) \tag{3}$$

with $y_t = z_t$ and $x_t = [z_{t-1}, z_{t-2}, \ldots, z_{t-n}]^\top$. In a nutshell, any provided sequence Z_t can be mapped into a regression problem.

2.2 Logmap dataset

Let's now apply time series prediction on the `logmap` dataset. As always, we first have to load the data:

```
>> load logmap.mat
```

Two variables are loaded into the workspace: `Z` (training data) and `Ztest` (test data). First, we have to map our sequence `Z` into a regression problem. This can be done using the command `windowize`:

```
>> order = 10;
>> X = windowize(Z, 1:(order + 1));
>> Y = X(:, end);
>> X = X(:, 1:order);
```

Now, a model can be built using these data points:

```
>> gam = 10;
>> sig2 = 10;
>> [alpha, b] = trainlssvm({X, Y, 'f', gam, sig2});
```

It is straightforward to predict the next data points using the `predict` function of the LS-SVMlab toolbox. In order to call the function, we first have to define the starting point of the prediction:

```
>> Xs = Z(end-order+1:end, 1);
```

Naturally, this is the last point of the training set. The test set `Ztest` presents data points after this point, which we will try to predict. This can be implemented as follows:

```
>> nb = 50;
>> prediction = predict({X, Y, 'f', gam, sig2}, Xs, nb);
```

where `nb` indicates how many time points we want to predict. Here, we define this number equal to the number of data points in the test set. Finally, the performance of the predictor can be checked visually:

```
>> figure;
>> hold on;
>> plot(Ztest, 'k');
>> plot(prediction, 'r');
>> hold off;
```

In this figure, the data points of the test set, that is, the actual data points that we want to predict are depicted in black, while the prediction is presented in red. Try to assess the performance of the prediction: do you think it's good? Is there still some room for improvements? As indicated numerous times before, the parameters `gam` and `sig2` can be optimized using crossvalidation. In the same way, one can optimize `order` as a parameter.

Questions

- As indicated numerous times before, the parameters `gam` and `sig2` can be optimized using crossvalidation. In the same way, one can optimize `order` as a parameter. Define a strategy to tune these 3 parameters.
- Do time series prediction using the optimized parameter settings. Visualize your results. Discuss.

2.3 Santa Fe dataset

In this exercise, we apply time series prediction on the Santa Fe laser dataset.

Questions

- Does `order` = 50 for the utilized auto-reressive model sounds like a good choice?
- Would it be sensible to use the performance of this recurrent prediction on the validation set to optimize hyperparameters and the model order?
- Tune the parameters (`order`, `gam` and `sig2`) and do time series prediction. Visualize your results. Discuss.

Exercise Session 3: Unsupervised Learning and Large Scale Problems

Dries Hendrikx* Michaël Fanuel† Francesco Tonin‡ Yingyi Chen§
Johan Suykens

Academic Year 2019-2020

This document describes a step-by-step guide for the application of support vector machine methods. The goal of the exercise sessions is to provide you with experience and teach you how to tackle future application problems. The MATLAB scripts and toolboxes, the course documents, the referred papers and all the datasets used in the exercise sessions are available for academic purposes on `https://toledo.kuleuven.be`.

- The exam consists of an oral discussion on the basis of the reports written for the 3 exercise sessions. It is important to show that you have understanding about the problem and that you can work in a constructive manner towards getting good solutions to given problems.
- The reports have to be written *individually*.
- The reports contain the solutions to the exercises *and* the homework problems. All the questions that need to be answered in the report are indicated by the grey box, which reads 'Questions'. We do not expect minimal answers to the questions but rather 'comprehensive' answers.
- A good report finds a good balance between *visuals* on the one hand and to-the-point *explanations* on the other hand. Use figures and tables to explain things, rather than only long and elaborate sentences. Make sure to include a few key equations in the textual part.
- Both the text and figures should not be too small. They have to be readable.
- Write a *separate* report for every exercise session. At the end of the semester, you have to submit a pdf document of **30 pages maximum** consisting of 3 separate reports in single column (max 10 pages for each session).

*dries.hendrikx@esat.kuleuven.be
†michael.fanuel@esat.kuleuven.be
‡francesco.tonin@esat.kuleuven.be
§yingyi.chen@esat.kuleuven.be

1 Exercises

1.1 Kernel principal component analysis

Kernel principal component analysis (KPCA) corresponds to linear PCA in a kernel-induced feature space, which is non-linearly related to the original input space. Thus, nonlinearities can be included via the kernel function and the corresponding problem keeps the form of an eigenvalue problem. Kernel PCA has numerous applications: it can be used for feature extraction, denoising, dimensionality reduction and density estimation, among others. In this section, we will explore the use of kernel PCA for denoising.

Download the files `kpca_script.m` and `pca.m` from Toledo. In order to illustrate the power of kernel PCA for denoising, this exercise focusses on an artificial example: a dataset which resembles the yin yang pattern. In order to get insight in the number of principal components, the choice of the kernel (and influence of kernel parameters) and the regularization hyperparameter, execute the script:

```
>> kpca_script
```

In order to provide uniformity across results, choose the number of data points equal to 400 and the data point dispersion equal to 0.3. Answer the following questions.

Questions

- Describe how you can do denoising using PCA. Describe what happens with the denoising if you increase the number of principal components.
- Compare linear PCA with kernel PCA. What are the main differences? How many principal components can you obtain?
- For the dataset at hand, propose a technique to tune the number of components, the hyperparameter and the kernel parameters.

1.2 Spectral clustering

Spectral clustering techniques make use of the eigenvectors of a Laplacian matrix derived from the data to create groups of data points that are similar. In this context, the kernel function acts as a similarity measure between two data points. The Laplacian matrix is consequently obtained by rescaling the kernel matrix. Note that these techniques can be interpreted as a form of kernel PCA.

Download the files `sclustering_script.m` and `two3drings.mat` from Toledo. Try the script:

```
>> sclustering_script
```

Questions

- Explain briefly how spectral clustering works.
- What are the differences between spectral clustering and classification?
- Edit the script and try different values of `sig2` (e.g., 0.001, 0.005, 0.01, 0.02). What is the influence of the `sig2` parameter on the clustering results?

1.3 Fixed-size LS-SVM

Based on the Nyström approximation, an approximation to the feature map can be obtained. This map can consequently be used to construct parametric models in the primal representation of the LS-SVM model. The approximation of the feature space is based on a fixed subset of data-points. One way to select this fixed-size set is to optimize the entropy criterion (`kentropy`) of the subset.

In some cases, we are interested in a sparser solution that we can attain using a predefined number of representative points. We can achieve this by applying ℓ_0-type of a penalty in an iterative fashion to an initial fixed-size LS-SVM solution.

For this exercise, we need the files `fixedSizeScipts.zip` (which contains two scripts) and `fixed-size` (which can be found in `SVM course scripts`). Download these files and add them to your directory.

Questions

- In which setting would one be interested in solving a model in the primal? In which cases is a solution in the dual more advantageous?
- What is the effect of the chosen kernel parameter `sig2` on the resulting fixed-size subset of data points (see `fixedsize_script1.m`)? Can you intuitively describe to what subset the algorithm converges?
- Run `fslssvm_script.m`. Use the Wisconsin Breast Cancer dataset for this exercise. Compare the results of fixed-size LS-SVM to ℓ_0-approximation in terms of test errors, number of support vectors and computational time.

2 Homework problems

2.1 Kernel principal component analysis

Download `digitsdn.m`. In this homework problem, we do image denoising using kernel PCA. The dataset used in this exercise consists of images of handwritten numerals (0 to 9), extracted from a collection of Dutch utility maps. Approximately 20 patterns per class (digit 9 has 18 images, total of 198 patterns) have been digitized in binary images. Execute the script:

```
>> digitsdn
```

As a rule of thumb, `sig2` is calculated as the mean of the variances of each dimension times the dimension (number of features) of the training data.

Questions

- Illustrate the difference between linear and kernel PCA by giving an example of digit denoising for `noisefactor` $= 1.0$. Give your comments on the results (based on visual inspection).
- What happens when the `sig2` parameter is much bigger than the suggested estimate? What if the parameter value is much smaller? In order to investigate this, change the `sigmafactor` parameter for equispaced values in logarithmic scale.

- Investigate the reconstruction error on training (`Xtest`) and validation sets (`Xtest1` and `Xtest2`), as a function of the kernel PCA denoising parameters. Select parameter values such that the error on the validation sets is minimal. Can you observe any improvements in denoising using these optimized parameter settings?

2.2 Fixed-size LS-SVM

In the second homework problem, we investigate the use of fixed-size LS-SVM for two additional datasets: the Shuttle dataset (classification) and the California housing dataset (regression).

2.2.1 Shuttle (statlog)

Adjust `fslssvm_script.m` and proceed with classification on the Shuttle dataset. Additional information on the Shuttle dataset can be found at `https://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle)`.

Questions

- Explore and visualize (part of) the dataset. How many datapoints? How many and meaning of attributes? How many classes? What is to be expected about classification performance?
- Visualize and explain the obtained results.

2.2.2 California

Adjust `fslssvm_script.m` and proceed with regression on the California dataset. Additional information on the California dataset can be found at `http://www.dcc.fc.up.pt/%7Eltorgo/Regression/cal_housing.html`.

Questions

- Explore and visualize (part of) the dataset. How many datapoints? How many and meaning of attributes?
- Visualize and explain the obtained results.