Support Vector Machines

ICS3206 – Machine Learning, Expert Systems & Fuzzy Logic

247696M

Edward Fleri Soler

Table of Contents

Introduction

This project shall focus on Support Vector Machines (SVMs), a supervised learning algorithm for the classification of data. The primary objective is to design, construct and implement an SVM capable of learning from a training set, and classifying an unseen test set. Three data sets have been made available for this project, each designed to test the capabilities of the proposed system. MATLAB has been chosen as the platform on which to develop this system as it is well suited for such a task and the availability of the VLFeat open source library [1] for the implementation of the SVM classifier.

We shall proceed by first discussing and understanding what SVMs are, and how they work. Following this background research, we shall elaborate upon the datasets provided and the experimental setup proposed, together with the reasons backing this setup. The configuration and fine tuning of the system will then be discussed, followed by the comparison of the expected and actual results obtained and some concluding thoughts.

Background Research

The earliest implementations of Support Vector Machines date back to 1963, when Vapnik and Chernvonenkis first devised the algorithm. SVMs are mathematical models which perform categorisation similar to K-Nearest Neighbour, Artificial Neural Networks and Decision Trees. However, SVMs only perform binary classification, classifying a given input as one outcome or another. They are provided with a training set, which is used to learn the classification model so as to classify unseen test examples.

Most classification algorithms approach the problem in a similar work flow. Input data is treated to exist as points in a search space; for ease of understanding we shall consider this search space to be 2-dimensional. We start by considering a training search space, consisting of a number of points (examples) spread across the domain. Being the training set, each point within the domain is labelled as being part of one category or another. The concept is to find a line across the search space which separates all points belonging to one category on one side, and all points belonging to the other category on the other side. Future unlabelled examples are then plotted in this same search space, and labelled according to the side of the line on which they fall.

The motivation behind SVMs is the search for the best possible hyper-plane dividing the two subsets (classifications). The approach is to find a division which yields the greatest marginal gap to either side. This means that we would like to find the hyper-plane which correctly separates the labelled data while achieving the greatest distance (margin) from every point on either side. This ‘perfect’ division is achieved through a mathematical model based upon vector calculus.

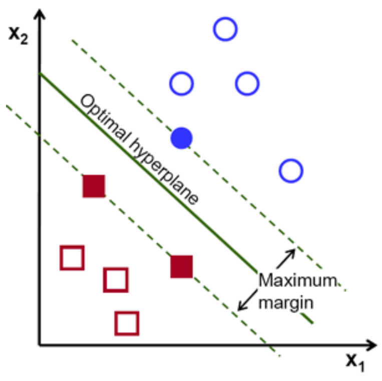

Figure 1 Depiction of best possible division of training data
Source: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

This division is then retained for the categorisation of unlabelled test data. This data is plotted onto the search space and, according to the side of the line on which it falls, is assigned a label. One limitation of this algorithm when restricted to a 2-dimensional domain is the possibility of overfitting. This occurs when the training data, together with its possible errors, are learnt perfectly, resulting in incorrect classification of unlabelled data. This could occur with due to the complex line division required to separate chaotic search spaces. A solution to this problem is kernelling.

Kernelling involves mapping all examples in the current search space onto a new transformed space known as the feature space. The idea is that the mapping of these values to their new space, may result in new configurations of points which are easier to separate and allow for a division with a greater margin to be achieved. With the application of kernelling, the mapping of unlabelled data is transformed in the same way, yielding the predicted classification without actually transforming the example value. For this reason, in literature the division of the search space into two separate parts is called a hyper-plane, which holds one dimension less than the search space within which it exists.

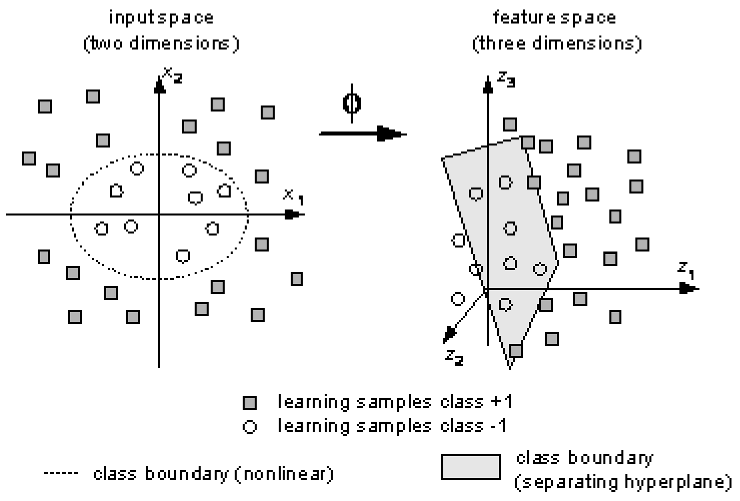

Figure 2 Kernel mapping from a 2-dimensional domain to a 3-dimensional feature space Source: http://forecasting-courses.com/support_vector_machines.htm

Different kernel mappings exist, each suited for different scenarios. Some of the most common include quadratic, polynomial or RBF kernels. We shall explore the use and success of these kernels on the different datasets in this project.

One final drawback of SVMs is the restriction of binary classification. This means that the classification of an X-category problem cannot be completed by one SVM. Instead, X SVMs are trained, with each one focusing on one specific category. When training each SVM, all labelled examples which are not the focus category of the current SVM, are labelled as negative, while all examples falling under the current focus category are labelled positive. Therefore, each of the X SVMs is capable of specifying whether an unlabelled example is of the current category or not. Test examples are passed through all X SVMs and are labelled according to which SVM provides a positive score with the greatest confidence.

Experimental Setup

In this section we shall go through and discuss the various components of the system and their respective requirements and functionality.

Dataset

Three separate datasets were provided for the validation and testing of this system, each one with different characteristics. The first data set is an Occupancy Detection Data Set [2]. This dataset is based upon a binary classification of occupancy. Environment variables within a room, such as temperature, humidity and CO_2 levels are used to predict whether an individual is currently within the room or not. This dataset was readily split up into three files; one training file holding over 8000 labelled examples and two testing files. For the purpose of evaluation and fine tuning, one of the two test files holding roughly 3000 examples was designated as a validation set used to improve the SVM training, while the other holding just under 1000 examples was reserved to test the system.

Table 1 List of data attributes in Dataset 1

Attribute Title	**Description**	**Considered for training/classification**
Number	Example number, simply acting as a label	No
Date	Date & Time stamp of recording	No
Temperature	Room temperature in °C	Yes
Humidity	Relative Humidity in %	Yes
Light	Light readings in Lux	Yes
CO_2	CO_2 levels in parts per million	Yes
Humidity Ratio	Derived quantity from temperature and relative humidity, in kg-water-vapour/kg-air	Yes
Occupancy	Binary value (0/1)	Target attribute

The second dataset relates to the medical field, consisting of 22 binary feature pattern values extracted from a Single Proton Computed Tomography (SPECT) image [3]. The target attribute is a binary value, relating to a normal or abnormal diagnosis. This dataset is comprised of two files; a training dataset of 80 instances and a test dataset of 187 instances. The minimal number of training examples provided with this problem shall test the effectiveness of the setup and the system's ability to classify input with little learning.

The third and final dataset is by far the most complex, relating to human activity recognition using smartphones [4]. This dataset is readily split into two files pertaining to the training and testing sets. Both sets involve an attribute space of 561 attributes, relating to measurements taken by the smartphone devices, with the training set containing roughly 2900 examples, and the testing set containing over 7000 examples. Each example relates to one of six activities: walking, walking up stairs, walking down stairs, sitting, standing and laying. It is apparent that 6 separate SVMs must be trained and used during the classification process of this data set.

Data Importing

The first step in the training and testing of the SVMs involved importing the datasets from text files into MATLAB. This process was completed by means of the 'Import Data' function offered by MATLAB, which, through a GUI, allows the specification and organisation of the various attributes to be imported. Only salient attributes which would have a positive outcome on the training of the SVMs were imported from the three datasets. Since each dataset provided was readily split into training a testing sets, no separation of data was required at this stage. Once all the required data was imported and stored in matrix variables, these variables were saved as '*.mat*' files within the workspace directory. This step was included to improve upon the efficiency of the system, skipping the importation step whenever it was possible to load the variables from file.

Cross Validation

A standard step in the development and testing of all machine learning systems is cross validation. This step involves producing separate datasets for the validation of the system. The first of the three datasets was provided with a validation set readily separated, however, the remaining datasets required this validation set to be created. A simple function was therefore defined to take a training set as well as a ratio, and return two new sets. Examples from the training set are randomly selected and assigned to one of two new sets at the ratio defined. This results in the creation of a new validation set from examples extracted from the training set.

During the testing and improvement stage of the system, the validation set will be used to gauge the accuracy. Once the system is tuned to the validation set, the real test data can be input to the system and categorised. This simulates the case of the real-world implementation of this system, where the test data would not be labelled.

SVM Implementation

The design and implementation of the SVM algorithm from scratch was beyond the scope of this project. Therefore, the SVM implementation found within the VLFeat open source MATLAB library was made use of. This implementation, consisting of a simple function call, takes as arguments the dataset for training as well as their target labels and a decimal value relating to sigma, which regularizes the kernel used to map the input data onto the feature space. This function call returns two values, which are used to compute the SVM score of each unlabelled input. The label assigned to the test input is based upon the positivity or negativity of the score value computed. Another function from the VLFeat library is the *vl_svmdataset* function, which is used for kernel mapping of data prior to the training of the SVM. This takes a number of arguments including the training data and the name of the kernel to implement.

Upon execution, the datasets are loaded into matrix variables and cross validation is performed, creating training, validation and testing variables for each of the tree datasets. We then proceed by training an SVM classifier upon the first dataset. The classifier is trained, and the SVM score for each validation example is computed, by multiplying ratings produced by the SVM with the attribute values of each example to produce a classification score. As was mentioned above, a label is then defined for each test example based upon this score. Finally, we must predict the accuracy of the classification. This is done by comparing the label assigned to each test example, with the true label of the category to which it belongs. This results in the computation of a percentile score for the accuracy of the classification upon the current dataset.

The same exact process is followed for the second dataset, however due to the multiple target categories in the third dataset, the same procedure is followed 6 times, training 6 different classifiers, each one tailored to test for a specific target attribute. Each test example is scored by each SVM classifier, and assigned a label relating to the classifier which yields the greatest score.

Configuration and Parameters

For the purpose of testing and improvements, each of the SVM classifiers were trained with different sigma values and different kernels. Therefore, the above process was repeated 25 times upon the validation set, to accommodate 5 sigma values (0.00001, 0.0001, 0.001, 0.01, 0.1) and 5 different kernels (linear, quadratic, polynomial, RBF and MLP). This, in turn, was repeated 100 times over across the three datasets. This yielded 7500 accuracy scores, 2500 for each dataset, with each combination of sigma and kernel being repeated 100 times each. These 7500 accuracy ratings were then reduced to 75 by averaging out the accuracy of each combination of sigma and kernel for the three datasets. This was done so as to overcome outlying precision ratings.

The final set of accuracy ratings were then plotted on three figures, comparing the accuracy of classification of each dataset with different kernels and sigma values. Finally, this data was used to select the best combination of sigma value and kernel for each dataset, to be implemented upon the test set.

Observing figures 3-5 on the following pages, the datasets may be better understood and a prediction of expected results may be made. Considering figure 3, we may see that the classifiers with sigma values set to 0.00001 achieved a high level of accuracy upon the validation set, with the RBF kernel yielding the best result. However, for other sigma values the data is more chaotic, and accounts for a large distribution of accuracies across the 100 tests conducted. We may therefore predict that as long as a miniscule sigma value is applied during the training of the classifier, accuracy ratings within the upper 90 percentiles should be expected on the test set.

Figure 4 depicts a slightly less chaotic outcome for the SVM classifier on the second dataset. A correlation between the sigma value and the accuracy may be observed, with an increase in sigma relating to a higher precision measure. According to the plot, the best suited parameters include an MLP kernel with a sigma value of 0.1. Considering that the training and validation sets of this dataset are so small, it is hard to predict the expected outcome of the classifier on the test set. The 80 examples provided are insufficient to achieve high ratings as are expected in the other two datasets. A somewhat modest prediction would be a precision of between 70% and 80%.

Out of the three figures, it is immediately apparent that figure 5 is the most organised and the easiest to interpret. The choice of kernel has little effect on the accuracy of the system, while the sigma value does. This is most likely due to the large number of attributes involved, and the size of the training set, making it easy to distinguish amongst the different unlabelled examples. While classifying the test examples, a small sigma value shall be set, in the hope that the case depicted below is mirrored. It is predicted that a classification accuracy close to 100% should be achieved when applying the trained SVM to the test data.

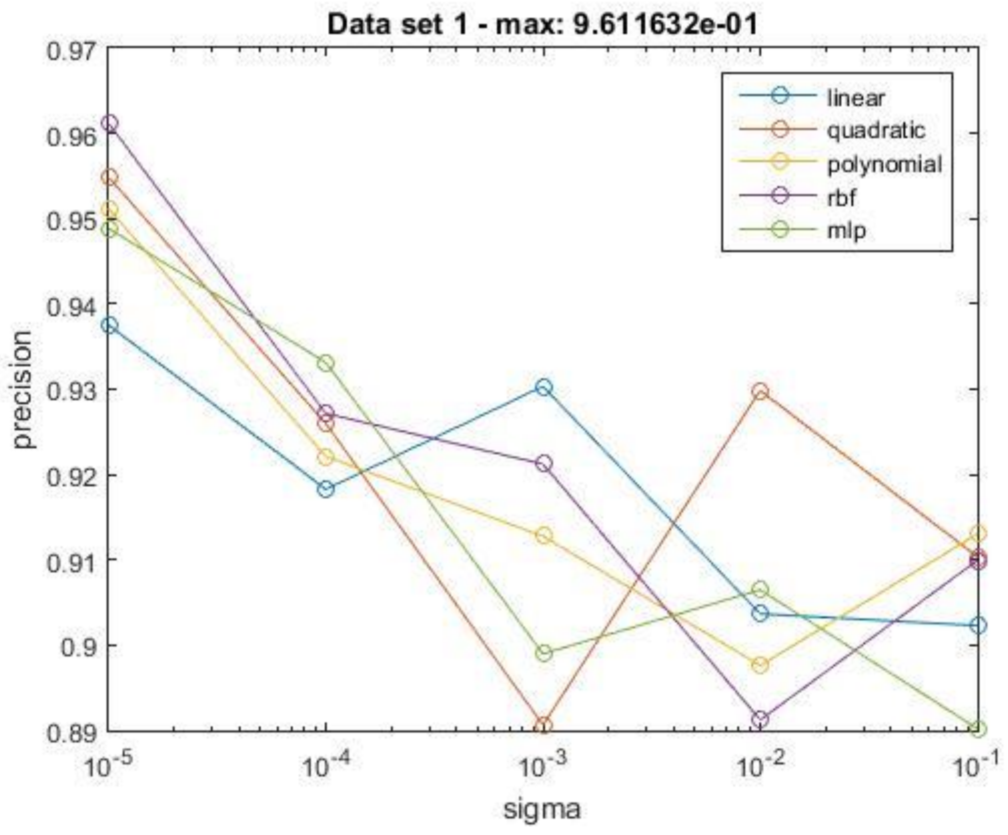

Figure 3 Comparison of kernels and sigma values with respect to accuracy for dataset 1

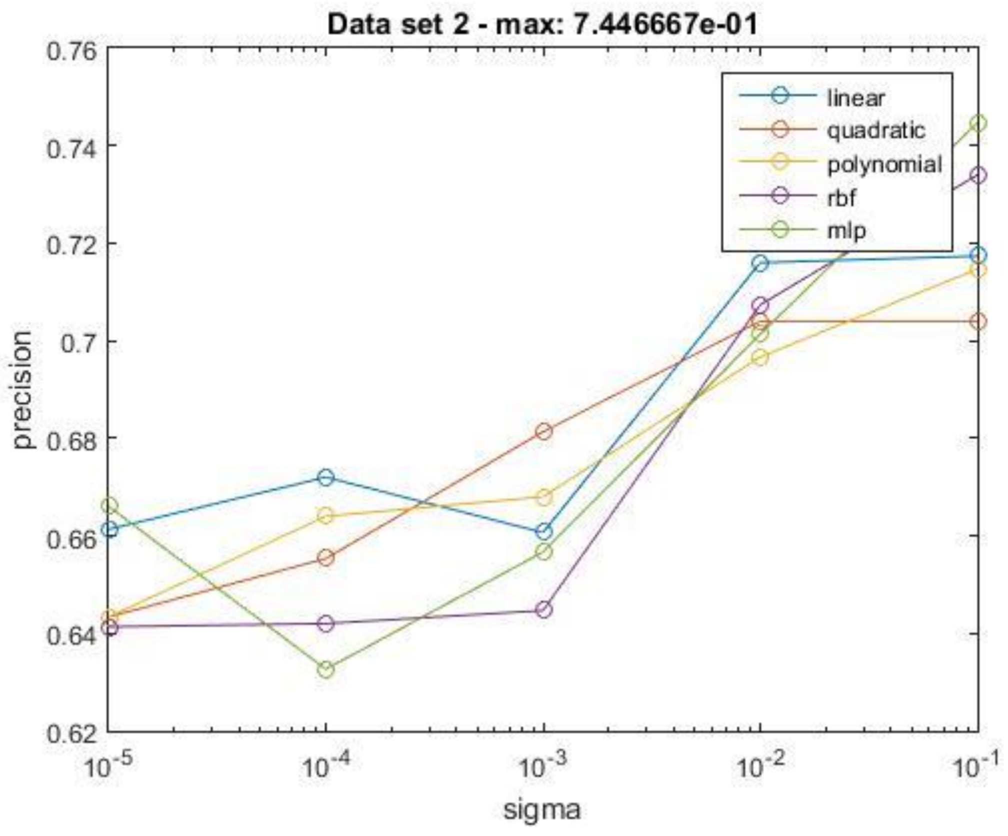

Figure 4 Comparison of kernels and sigma values with respect to accuracy for dataset 2

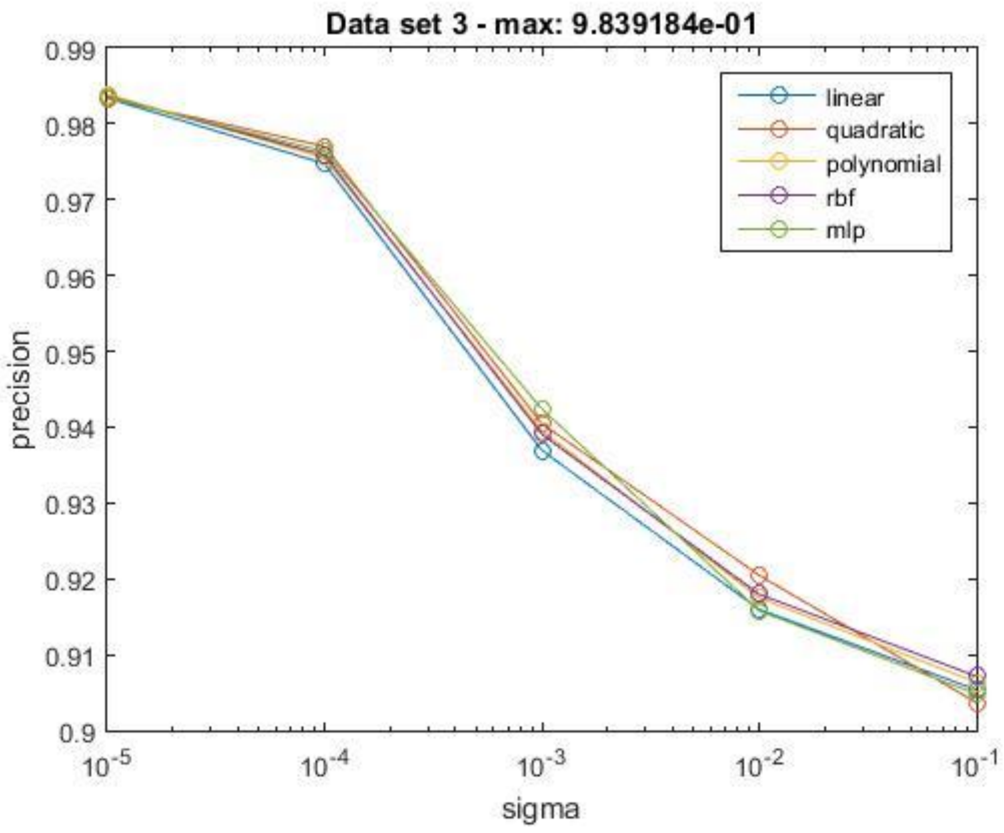

Figure 5 Comparison of kernels and sigma values with respect to accuracy for dataset 3

Results

The MATLAB code for the system was slightly modified so as to run the SVM classification upon the test datasets and not on the validation sets. The following was output on the MATLAB command window, specifying the best parameters for the tuning of the classifiers:

```
Maximum performance of 9.612e-01 for data set 1 achieved with rbf kernel at
1.00000e-05 lambda
Maximum performance of 7.447e-01 for data set 2 achieved with mlp kernel at
1.00000e-01 lambda
Maximum performance of 9.839e-01 for data set 3 achieved with polynomial kernel at
1.00000e-05 lambda
```

The following table outlines the results achieved when training the classifiers with the above set parameters, and running them on the test data:

Table 2 Test set classification results

Dataset	Sigma	Kernel	Accuracy
1	0.00001	RBF	99.3%
2	0.1	MLP	74.9%
3	0.00001	Polynomial	96.4%

Conclusion

Following the design, implementation and testing of the SVM classifiers, the effectiveness of this classification model has been appreciated. A better understanding of the tuning of such classifiers, and their flexibility to handle datasets of different specifications has been achieved. I believe that the approach taken towards the evaluation of the system was comprehensive and we may conclude that the classifier has achieved satisfactory results in the classification of all three test sets, with the consideration for the lack of training examples in the second dataset.

References

[1] *http://www.vlfeat.org/install-matlab.html*

[2] Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. Luis M. Candanedo, VÃ©ronique Feldheim. Energy and Buildings. Volume 112, 15 January 2016, Pages 28-39.

[3] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[4] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. A Public Domain Dataset for Human Activity Recognition Using Smartphones. 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013. Bruges, Belgium 24-26 April 2013.