Universitat Politècnica de Catalunya

Master in Artificial Intelligence

Computational Intelligence

Supervised classification exercice

Authors:
Miquel Perelló Nieto
Marc Albert Garcia Gonzalo

Date:
December 21, 2012

Contents

1 Abstract **2**

2 Introduction **3**

2.1 Support Vector Machine 3

2.2 Soft margin 3

2.3 Kernel 3

2.4 Primal and Dual forms 3

3 Specification of the problem and organization of the software **4**

3.1 Problem approach 4

3.1.1 Multiclass Support Vector Machine 4

3.2 Technology 5

3.3 Code structure 5

3.3.1 / 5

3.3.2 /datasets/ 6

3.3.3 /svm_train/low_level/ 6

3.3.4 /svm_train/ 7

3.3.5 /svm_test/low_level/ 7

3.3.6 /kernels/ 7

4 Experiments **8**

4.1 Wisconsin Breast Cancer 8

4.2 Pima Diabetes 9

4.3 Heart StatLog 9

5 Conclusions and future work **10**

1 Abstract

This document describes the implementation and experimentation with different kinds of Support Vector Machines. These kinds include the linearly separable version, and the soft margin version. For both cases, the primal and dual form are considered. Also, the use of different kinds of kernels is studyied.

2 Introduction

2.1 Support Vector Machine

Support Vector Machine is a linear classification model, which main goal is to maximize the margin between the separator hyperplane, and the data points.

Original Support Vector Machines were designed for data which could be linearly separable.

It has been proved, that the Support Vector Machine model has lower complexity than similar models, as the solution only depends on a reduced set of the training data. These instances are known with the name of Support Vectors.

2.2 Soft margin

In 1995, Vapnik and Cortes introduced a new version of Support Vector Machines which are able to deal with non linearly separable data. This version is known as the soft margin version, and allows violations of the margin.

The relative importance given to the maximitzation of the margin, and to the avoidance of margin violations is controlled by a C parameter of the model.

2.3 Kernel

Support Vector Machines are commonly used using kernels. Kernels allow applying transformations to the data, to a higher dimension (even infinite), but making it computational feasible. This is possible because the operations with the transformed data are never done explicitely. Operations in the calculations of the Support Vector Machine only need the result of scalar products of the transformed data among them, and the dimensionality of the scalar product does not depend on the dimensionality of the transformed data, but on the number of instances.

2.4 Primal and Dual forms

There are two different formulations of the Support Vector Machines. The primal formulation is the one derived from the Support Vector Machine theory. The dual formulation is obtained after applying Lagrangian theory, and it can be solved as a Quadratic Programming problem.

3 Specification of the problem and organization of the software

3.1 Problem approach

For this project, we have implemented Support Vector Machines sequentially, so we have seen in practice all the different formulations step by step.

Next it is listed the sequence followed for the implementation.

1. The hard margin case on its primal form
2. The soft margin case
3. Dual form of both hard and soft margin with linear kernel
4. Radial Basis Function and Polynomial kernels
5. Multiclass classification

While all the code is attached, in practice, it is not likely to be useful most of it. Specifically, the hard margin case, and the primal form, because are limiting and slower than the soft margin dual form. For the experiments, only this last version has been used.

3.1.1 Multiclass Support Vector Machine

In the project we adapted all the code to solve problems with multiple classes. First of all we create one set of labels for each class of the dataset. It is possible to create this with one classifier less than the number of classes. With one classifier less the last class is selected in the case that non of the others have predicted the point.

Our approach is to create this new labels for each class. Each new labels contains negative ones to all the other classes and positive ones for his class. Then is trained one support vector machine for each class with his respective new labels. The SVM's can be of any type of the commenteds above: without kernel or with linear, polynomial and Radial Basis Function kernels. It is possible also to select soft or hard margin. Once all SVM are trained, we can predict new instances with the one-vs-all approach. It means that in each of the SVM's the new instance is predicted with one class vs all the others. We get all the predictions and the one with greatter results is choosed.

In the implemented code first of all we can choose with internal variables if we want to solve the problem with kenels or not. Wich kernel we want and their parameters. Once done we can select some points by hand for four classes.

This program will take by default 80% of the data as training and remaining for test. Then will train the four SVM's and predict the test points. It is possible to see in the figure 1 one example with RBF kernel and sigma 0.1. The result was 0.975 of accuracy. In the figure from left to right and upper to bottom are the predictions for the first, second, third and fourth class. The higher values are representeds in red. And the selected class is the higer in the computed point.

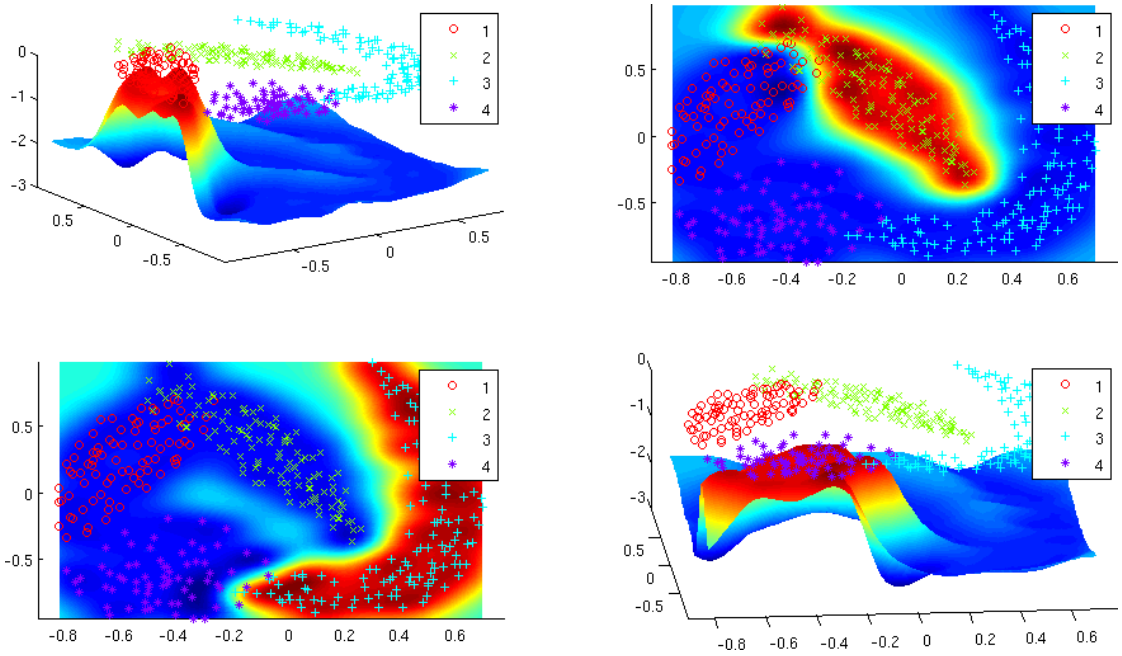

Figure 1: Multiclass support vector machine representation

3.2 Technology

This project has been developed using Matlab (version 7.12).

Only the CVX module has been used, as an external plugin.

3.3 Code structure

3.3.1 /

Main files of the project, to be executed directly.

main.m : Trains models for three different datasets, with different parameters, and calculates the accuracy of the model using 10-fold Cross Validation.

main_multiclass.m : Trains a model for the multiclass dataset problem, with customizable parameters, giving the accuracy by separating some data for testing, and plots the result.

3.3.2 /datasets/

Functions for loading different datasets.

arffload.m : Function for parsin ARFF (Weka) files

simple.m : Very simple custom dataset with only 4 linearly separable instances

picker.m : Creates a 2D dataset by selecting the points in a Matlab plot

picker_multiclass.m : Creates a 2D dataset by selecting the points in a Matlab plot, for the multiclass case

iris.m : Fisher's iris dataset, loaded from Matlab's builtin, and formatted to be used with Support Vector Machines, merging two classes to be used as a binary classification problem

breast_w.m : Loads the Wisconsin Breast Cancer dataset from an ARFF file, and formats it to be used for training and testing a Support Vector Machine

pima_diabetes.m : Loads the Pima Diabetes dataset, with the format to be used with the Support Vector Machine

heart_statlog.m : Loads the Heart StatLog dataset, with the format to be used with the Support Vector Machine

3.3.3 /svm_train/low_level/

Functions for calculating the optimal parameters of the Support Vector Machine. These functions are the low level code, and do not include the calculation of the kernel matrix. These functions use the CVX Matlab package.

train_svm_hard_margin_primal.m : Function for training the Support Vector Machine, with hard margin (linear separability assumed), using the primal definition

train_svm_soft_margin_primal.m : Function for training the Support Vector Machine, with soft margin (margin violations allowed), using the primal definition

train_svm_hard_margin_dual.m : Function for training the Support Vector Machine, with hard margin (linear separability assumed), using the primal definition

train_svm_soft_margin_dual.m : Function for training the Support Vector Machine, with soft margin (margin violations allowed), using the primal definition

3.3.4 /svm_train/

Functions that return the Support Vector Machine model, using the low level functions, and providing a higher layer interface.

train_svm_none.m : Function that resturns a SVM model, using the primal form, and without using any kernel.

train_svm_linear.m : Function that resturns a SVM model, using the dual form, using a linear kernel.

train_svm_rbf.m : Function that resturns a SVM model, using the dual form, using a Radial Basis Function kernel.

train_svm_polynomial.m : Function that resturns a SVM model, using the dual form, using a polynomial kernel.

3.3.5 /svm_test/low_level/

Low level functions to classify data using a Support Vector Machine model previously trained.

test_svm_w.m : Classifies data using the SVM primal form, without using a kernel. Uses the weights and the bias.

test_svm_svs.m : Classifies data using the SVM dual form, using the same kernel as used for training, the Support Vectors, the alphas (Lagrange multipliers) optimized in the training, and the bias.

test_svm_multiclass.m : Classifies datasets with more than two classes using the dual form approach.

3.3.6 /kernels/

Kernel functions.

kernel_linear.m : Linear kernel

kernel_rbf.m : Radial Basis Function kernel, with parametrizable Gaussian sigma

kernel_polynomial.m : Polynomial kernel with parametrizable polynomial degree

4 Experiments

For testing the performane of our custom implementation of the Support Vector Machine model, we have develop a set of experiments, in order to evaluate that it is working correctly.

We have used three different common Machine Learning datasets. All them are binary classification problems.

Classification tests have been performed using 10-fold Cross Validation. Mean accuracy and standard deviation among folds is reported.

We have also trained Matlab's implementation of the Support Vector Machine, and we used the accuracy obtained with this implementation as a benchmark.

4.1 Wisconsin Breast Cancer

This data set contains information about the results of tests on breast cancer based on visual characteristics of the result of a Fine Needle Aspiration. Data is gathered by taking measures from a microscope scan.

There are 699 samples. Attributes are visual features of the cells, such as radius, perimeter, area, symmetry, etc.

Data set contains missing values. We imputed them using Matlab's KNN function for imputation.

Table 1: Wisconsin Breast Cancer results

Kernel	**Our Accuracy**			**Matlab Accuracy**			**Standard deviation**		
	C = 0.25	C = 1	C = 4	C = 0.25	C = 1	C = 4	C = 0.25	C = 1	C = 4
Linear	0.94859	0.94996	0.96404	0.96404	0.96402	0.96549	0.040991	0.041094	0.019739
Polynomial (order 2)	0.93409	0.92686	0.93105	0.94394	0.94269	0.93694	0.020835	0.023495	0.028839
RBF (sigma 0.25)	0.62009	0.73224	0.73224	0.75424	0.85690	0.88993	0.104410	0.043951	0.043951
RBF (sigma 1)	0.85550	0.91559	0.92415	0.95132	0.95400	0.96118	0.032371	0.020555	0.018033
RBF (sigma 4)	0.96398	0.96106	0.95547	0.96831	0.96833	0.96551	0.021106	0.024122	0.018887

As we can observe, our implementation of the Support Vector Machine works fine with this dataset. Our predictions are very close to the ones obtained with Matlab implementation, and the variance of the predictions is low.

4.2 Pima Diabetes

This data set contains information about female Pima Indian individuals. Pima Indians have a higher rate of diabetes than normal, and this data set is used to study the causes.

There are 768 individuals on the data set. Attributes contain medical information such as the age, the number of times the individual was pregnant, and the results of several medical tests.

Table 2: Pima Diabetes results

Kernel	**Our Accuracy**			**Matlab Accuracy**			**Standard deviation**		
	C = 0.25	C = 1	C = 4	C = 0.25	C = 1	C = 4	C = 0.25	C = 1	C = 4
Linear	0.73438	0.73442	0.74358	0.75532	0.75537	0.75537	0.11283	0.11132	0.10840
Polynomial (order 2)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
RBF (sigma 0.25)	0.34894	0.47352	N/A	0.64843	0.64843	N/A	0.0095972	0.15706	N/A
RBF (sigma 1)	0.34894	0.38548	0.65106	0.66685	0.73205	0.70323	0.009597	0.027144	0.009597
RBF (sigma 4)	0.34894	0.63118	0.65754	0.73981	0.75002	0.74479	0.009597	0.075238	0.020859

For this dataset, we observe that there are some cases where our model is unable to give a prediction. This is caused by the optimization, which is returning NaN for the optimized variables. We could not find a reason for it.

For the cases where it was possible to predict, we can see how in the linear case the variability on the predictions gets worse, as the accuracy also drops. This is probably normal, as the dataset is harder to classify.

For the Radial Basis Function kernel, we can observe how in this case, and for some parameters, our estimation accuracy drops radically compared to the Matlab model. We do not see a reason for this, but it is probably produced on the optimization.

Considering that it is a binary problem, classifing with an accuracy lower than 50%, the model is worse than a model which classifies randomly (assuming that there are the same number of individuals per class).

4.3 Heart StatLog

This dataset contains 270 instances about patients, regarding if they have a heart disease or not.

There are 17 numerical attributes with demographical and medical data about the individuals.

For this dataset, our results are similar to the previous case. For the linear case, the model is performing almost as good as Matlab's, and with low variance.

Again, for the polynomial, the optimazer is unable to find the optimal values. And for the RBF, we can see how the predictions are performing poorly.

Table 3: Hear StatLog results

Kernel	**Our Accuracy**			**Matlab Accuracy**			**Standard deviation**		
	C = 0.25	C = 1	C = 4	C = 0.25	C = 1	C = 4	C = 0.25	C = 1	C = 4
Linear	0.82593	0.83704	0.83704	0.84074	0.83704	0.84074	0.042945	0.055761	0.055761
Polynomial (order 2)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
RBF (sigma 0.25)	0.44444	0.44444	0.55556	0.46667	0.55556	0.55556	0.000000	0.000000	0.000000
RBF (sigma 1)	0.44444	0.45556	0.55556	0.71481	0.72222	0.72593	0.000000	0.024998	0.000000
RBF (sigma 4)	0.44444	0.57778	0.58148	0.84074	0.83704	0.81481	0.000000	0.11608	0.035136

5 Conclusions and future work

In this project, he have seen the all theory inside a Support Vector Machine classifier. Its different parts, from the margin optimization, to the kernels.

We have seen the differences in the implementation between the hard and soft margin cases. Also, the differences between the different kind of kernels.

On the classification, we have seen how the dual form makes its predictions based only on information about the support vectors and the bias. This reduces the complexity of the model, as we can make the classification on a infinite dimension space, without requiring an infinite number of weights, but with a finite, and luckily reduce set of support vectors.

On the experiments, we observed how our implementation is sensitive to the datasets, not being able to predict in some cases (mostly with the polynomial kernel), and how it predicts poorly, in some cases with the RBF kernel. We believe that this is caused on the optimization part, but further research would be necessary.

Some of the possible future work regarding this project is listed next.

- Further research on the reasons why CVX does not get a optimal solution
- Refactoring of the code, so passing kernels, and possible parameters among the functions is done in a cleaner way
- Automatically detection of datasets with more than two classes, so the one-vs-all model is used when needed

References

[1] Support Vector Machines Explained, Tristan Fletcher, University College London, 2009