Laboratory Report

Group 17

Mr Tomaso Cetto

4228714

Mr Dhruva Arjun Krishnamurthy

4278450

Mr Hao Sun

4302414

Mr Mohammed Ahmed

4338335

Mr Rapee Wannasiri

4342949

Coursework 3: Support Vector Machines

29 Nov 2018

Table of Contents

Introduction

Support Vector Machines are a supervised machine learning algorithm used to analyse data for classification and regression. It attempts to find the best line which separates a dataset by assigning support vectors to define a margin between the groups of data that is as wide as possible. A caveat of this is that it can only be used for datasets that are linearly separable. In such situations, they are coupled with kernel functions which map the input space into a linearly separable feature space. Learning algorithms are then used to to generate an optimal hyperplane which maximises the distance of the closest points to the margin, splitting the data points into categories as best as possible.

In this coursework, we build SVM models for regression and binary classification using Gaussian RBF, Polynomial and Linear kernel functions. We ran an inner-fold cross-validation to select the optimum hyper parameters and tested it on unseen data, minimising generalisation error. We then compared the performance of this method to decision trees and artificial neural networks(ANNs), to determine statistical significance via t-tests.

Explanation of hyperparameter choices

The hyperparameters to be optimised are dependent on each other in all observed cases, and were chosen using grid search within a predefined range. This predefined range covers very high and very low values of each hyperparameter, to maximise the spread of values used. The hyperparameters chosen are a result of this gridsearch performed in an inner-fold validation where the highest f1 score was selected. This was then evaluated on the outerfold, reducing our generalisation error.

Unfortunately due to time constraints, linear and polynomial regression hyperparameters were treated as independent values to reduce the number of folds in which we needed to train a computationally expensive model. As a result it may not be representative of results acquired if adequate time was available.

B. Training SVMs with Gaussian RBF and Polynomial Kernels

Table 1: Binary classification results using Gaussian RBF and Polynomial kernels

	RBF Kernel		Polynomial Kernel	
Model Number	**N° of support vectors (total)**	**N° of support vectors (%)**	**N° of support vectors (total)**	**N° of support vectors (%)**
1	78	57.8%	26	19.3%
2	81	60.0%	36	26.7%

3	32	23.7%	28	20.7%
4	79	58.5%	30	22.2%
5	83	61.5%	29	21.5%
6	82	60.7%	38	28.2%
7	82	60.7%	29	21.5%
8	47	34.8%	31	23.0%
9	83	61.5%	30	22.2%
10	84	62.2%	25	18.5%
Average	**73.1**	**54.1%**	**30.2**	**22.4%**

Table 2: Regression results using Gaussian RBF and Polynomial kernels

	RBF Kernel		**Polynomial Kernel**	
Model Number	**N° of support vectors (total)**	**N° of support vectors (%)**	**N° of support vectors (total)**	**N° of support vectors (%)**
1	5703	70.76%	6213	77.1%
2	5669	70.33%	1613	20.0%
3	5657	70.19%	6203	77.0%
4	5688	70.58%	6161	76.4%
5	5723	71.01%	6167	76.5%
6	5652	70.13%	6200	76.9%
7	5702	70.75%	6209	77.0%
8	5688	70.58%	1614	20.0%
9	5658	70.21%	6193	76.9%
10	5731	71.07%	6246	77.4%
Average	**5687.1**	**70.6%**	**5281.9**	**65.5%**

C. Comparing performance between methods

Table 3: Binary Classification F1 scores

Fold Number	SVM - Linear	SVM - RBF	SVM - Polynomial	ANN	Decision Tree
1	0.9091	0.9415	0.7619	0.8235	0.7500
2	0.7500	0.8899	0.7989	1.000	0.8333
3	0.8000	0.9128	0.8674	0.3333	0.2222
4	0.6667	0.8933	0.7710	0.8889	0.6667
5	0.8000	0.9582	0.8923	0.8000	0.6667
6	1.0000	0.9196	0.8727	1.0000	0.8571
7	0.8235	0.9352	0.7444	0.8000	0.8751
8	0.8333	0.8976	0.7852	0.8889	0.6000
9	0.8889	0.9570	0.8055	0.8333	0.6667
10	0.9333	0.8897	0.8182	0.9091	1.0000
Average	**0.8404**	**0.9195**	**0.8117**	**0.8277**	**0.7120**

Table 4: Regression RMS Errors

Fold Number	SVM - Linear	SVM - RBF	SVM - Polynomial	ANN
1	3.1095	1.3850	2.5178	1.4653
2	3.2310	1.2656	2.6475	1.2606
3	3.3316	1.5140	2.7757	1.9640
4	3.3904	1.5832	2.9343	1.8147
5	3.2086	1.1438	2.5968	1.4109

6	3.3819	1.2134	2.6741	1.4745
7	3.3708	1.2491	2.7453	1.3886
8	3.0978	1.2597	2.5248	1.3375
9	3.1576	1.1466	2.5051	1.9077
10	3.2445	1.446	2.5369	1.7895
Average	**3.2523**	**1.3205**	**2.6458**	**1.5813**

Table 5: T-Test comparisons for binary classification dataset. All comparisons are statistically insignificant

	Neural Networks		**Decision Trees**	
	Statistically Significant? (Yes/No)	**P Value**	**Statistically significant? (Yes/No)**	**P Value**
RBF	No	0.0972	No	0.1842
Polynomial	No	0.1254	No	0.1999
Linear	No	0.1325	No	0.2285
Decision Trees	No	0.9834		

**Statistical significance is indicated by a P-Value < 0.05. N = 150 per comparison*

Table 6: T-Test comparisons for regression dataset. All comparisons are statistically insignificant

	Regression - Neural Networks	
	Statistically significant? (Yes/No)	**P Value**
Regression - RBF	No	0.7728
Regression - Polynomial Order	No	0.1787
Regression - Linear	No	0.1042

**Statistical significance is indicated by a P-Value < 0.05. N = 8955 per comparison*

Question 1: What does the kernel parameter of the Gaussian RBF kernel signify (sigma)? What happens when you increase its value?

To understand the role that the sigma parameter plays in the Gaussian kernel, it is important to observe its relation to the other variables in the equation defining the kernel:

$$k(x,\ x') \ = \ exp(-\ \frac{\|x-x'\|^2}{2\sigma^2})$$

As with the classical Gaussian distribution, the sigma parameter represents standard deviation, which determines the 'width' of the distribution. In the case of the Gaussian kernel, it will affect the type of decision boundary defined by our SVM. More specifically, the sigma value determines the 'influence' or 'reach' of a single training example in the setting of the decision boundary.
In the case of a **low** value of sigma, the reach will be smaller and only the points close to the decision boundary will affect its shape. In contrast, a **larger** value of sigma means that points further away from the boundary will also have an influence on the boundary itself.
The result of this is that a **large** sigma value, taking data points further away from the boundary into account, will produce a smoother, more linear boundary, giving a more general classifier. This means that for data of an overlapping nature, the misclassification rate will increase, however overfitting will be avoided.
A **smaller** sigma value, on the other hand, gives more importance to the points close to the decision boundary, and such these points will be able to 'warp' the decision boundary, resulting in a more flexible, less linear boundary. In most cases this is what we want, due to the fact that a more flexible boundary will be able to it the data more accurately, however, as with all close fits on training data, overfitting may become an issue.

Question 2: Explain what happens when a hard-margin SVM is fit to a dataset of two classes with overlapping features. What value do you need to set C (the slack-variable hyper-parameter) to attain a hard-margin SVM?

We know that a hard margin SVM maximises the margin on the condition that no training point is misclassified. Because the RBF Gaussian kernel can project data to infinite-dimension space, it is always possible to meet this condition, even if the two classes have overlapping features. However, the resulting classifier will have taken drastic measures to ensure no example is misclassified, that it will be completely specific to the training data and thus massively overfit. The goal of a soft margin SVM is to allow for some misclassification to ensure the algorithm can generate a decision boundary which is much less likely to overfit and thus generalize better.

$$\arg\min_{\xi_n, \boldsymbol{w}} C \sum_{n=1}^{N} \xi_n + \frac{1}{2}\|\boldsymbol{w}\|^2$$

In the minimization formula above for a soft margin classifier, C is a constant which gives the relative importance of the error (slack variable) term and the standard margin fitting term. In the case of a hard margin classifier, the value of C will be very large, resulting in a very high penalty for misclassified examples, as per the constraint of hard margin classifiers.

Conclusion

We successfully trained SVMs using 3 different types of kernels for both classification and regression whilst optimising hyperparameters via an inner-fold cross validation loop. We used F1 scores and RMS errors respectively as an indicator of the best hyperparameter choices. A caveat of this is that there is no weightage on the number of support vectors used which is one of the most essential components of SVMs. An apparent example is model 3 of table 1. Two sets of hyperparameters could return very similar F1 scores yet return completely different numbers of support vectors based on how they define their margins and the permutations of the training data. The selection of optimal hyperparameters therefore needs to be adjusted based on the demands and purpose of the model.

All comparisons tested were statistically insignificant. In such cases its important to consider the training time of the models. SVMs using linear and polynomial kernels for regression required a significantly longer training duration than all other methods.