Advances in Neural Networks
Course Project - Terrain Classification

Vicente Bosch Campos †
viboscam@posgrado.upv.es

June 13, 2011

Contents

List of Figures

1 Introduction

The application of Support Vector Machines to uni - dimensionally multi-spectral representation of a 3x3 pixel matrix of an image section to recognize terrain type of the central pixel is considered in this course project.

The use of Pattern Recognition and Image Analysis is quite established for the task of satellite image classification. In this course project we consider the classification of terrain types.

1.1 Data

A Landsat MSS image consists of four digital images of the same zone using different spectral bands. Two of the bands are in the visible region (corresponding approximately to green and red regions of the visible spectrum) while the other two are in the (near) infra-red. Each pixel is represented as a 8-bit binary word, with 0 corresponding to black and 255 to white. The spatial resolution of a pixel is approximately 80m x 80m. Each image contains 2340 x 3380 such pixels.

The database used in the course project is a sub-area of a scene, consisting of 82 x 100 pixels. Each line of data corresponds to a 3x3 square neighborhood of pixels completely contained within the 82x100 sub-area. Each line contains the pixel values in the four spectral bands (converted to ASCII) of each of the 9 pixels in the 3x3 neighborhood and a number indicating the classification label of the central pixel. As per the original database the number is a code for the following classes:

1. red soil
2. cotton crop
3. grey soil
4. damp grey soil

5. soil with vegetation stubble
6. mixture class (all types present)
7. very damp grey soil

Further preprocessing has been performed to the data provided to us. It has been normalized and as there are no records of the 6th type the classes label have been moved so that type 7 is represented by the label 6.

We have been provided with two data partitions:

- sat6c.tra: Training set formed by 2000 records with the following composition:
  - Class 1: 466
  - Class 2: 216
  - Class 3: 429
  - Class 4: 195
  - Class 5: 226
  - Class 6: 468
- sat6c.public.tst: Public test set containing of 1000 records with the following composition:
  - Class 1: 234
  - Class 2: 106
  - Class 3: 208
  - Class 4: 98
  - Class 5: 118
  - Class 6: 236

1.2 Scope

As part of the course project we will train classifiers for the landstat task described in the above sections. To be more precise we will use the following techniques:

- Support Vector Machines:
  - Multi class algorithm
  - Combination of two class classifier
- For both SVM techniques we will use the following Kernels:
  - Polynomial Kernel: $(\ s \ a * b + c)^d$
  - Radial Basis Function Kernel: $exp(\ -gamma \ ||a - b||^2)$
  - Sigmoid Kernel: $tanh(\ s \ a * b + c)$

All training classification will be performed with the sat6c.tra data and the best obtained classifiers will be evaluated against the sat6c.public.tst data set, which is not used for training nor for partial evaluation/guidance of the performance during the experimentation.

2 Development

In this section we will describe the software framework developed in order to perform the experimentations for the task. The framework developed allows us to load, format and perform transformations with specific data sets and also launch classifier training with different classifiers and obtain mean test set classification error with cross validation.

For the development of the framework we have used:

- SVM Light and SVM Multiclass developed by Thorsten Joachims.
- Ruby scripting language 1.9.2
- Ruby libraries:
  - Awesome print: to print outputs in a more human readable manner.
  - Open 3: to allow us to connect to the input, output and error streams of an executed command
  - Date: to retrieve the current date and time in order to time stamp files.

2.1 Software Structure

The software is structured into the following classes:

DataPattern: The DataPattern class is defined in the *./lib/data_pattern.rb* file. It encapsulates a sample record of class containing the actual data (can be in numeric or string representations) and the class label. It allows us to perform the following operations:

- Order to records by label value.
- Iterate over the different values of the vector.
- Duplicate the record.

DataSet: The DataSet class can be found in the *./lib/data_set.rb* file. It represents a whole set of records of a task allowing us to perform the following tasks over them:

- Read the original data from a file and apply specific format to them to prepare them for the ML tool. e.g. Uniform the chain length for a Chromosome classification task.
- Write them to file using an specific format for a Machine Learning toolset.
- Partition the set into blocks ensuring correct representation and number of each class in the block.
- Combine different sets in an additive manner.
- Sort the set by class label.
- Shuffle the set.
- Filter the set to a sub set of labels/classes.
- Binarize labels so that they can be sent to 2-class limited classifiers.

Svm: The Svm class can be found in the *./lib/svm.rb* file. It is an object wrapper for the svm light and svm multiclass developed by Thorsten Joachims and allows us to perform the following tasks from the framework:

- The class allows as input an experiment object, and performs a classification with the svm light or svm multiclass algorithms.
- Permits the evaluation of a test set and returns us the classification error obtained.
- Permits the classification of a test pattern by saving it to a temporary file and returns us the estimated class.

SvmFileWriter: The class can be found in the *./lib/svm.rb* file. The file writer is in charge of performing header and footer special formatting for the SVM toolset. In this case it creates the pattern file and just adds a #EOF at the end of the file as the SVM commands used do not require any other file formatting.

SvmPatternWriter: Can be found in *./lib/svm.rb*. The pattern writer in in charge of performing special formatting to the patterns required for correct usage on the desired toolset. In this case the label is printed at the beginning and each of the dimensions value is preceded by the relative position of the dimension inside the feature space as the selected SVM toolset enforces sparse vector representation.

SvmCombinations Coded in *./lib/svm_combinations.rb*. The class represents a multiclass classifier that is made up from the combination of two class classifiers in a directed acyclic graph (Plat, Cristianini & Shawe-Taylor 2000). The class allows us to:

- Generate the specific data sets required to train each of the 2-class SVMs to implement the DAGSVM.
- Train and store the 2-class classifiers of the DAGSVM.
- Permits the evaluation of a test set and returns us the classification error obtained.

Parameter: Coded in *./lib/parameter.rb*. The class allows us to define a parameter and a range of values of variation. The values can be a set of strings, if for example we want to vary between different neural network files or a numeric value, to study an specific range of the momentum parameter.

Experiment: Present in the file *./lib/experiment.rb*. It is an specific configuration, value set, of the parameters that will be executed on the chosen ML tool set and the value will be averaged over cross experimentation.

Study: contained in *./lib/study.rb*. It generates the set of experiments by performing combinations of each of the values for each parameter described and runs them writing the results and studies performed in specific files.

2.2 Library Options

With the defined software structure we can easily perform studies on an indicated task by just having to write the specific read and write formatters for the task. Next we will show a short example of the framework scripts used to DAGSVMs for the terrain classification task:

Listing 1: chromosome.rb

```
1  #! /usr/bin/env ruby
2
3  require 'ap'
4  require_relative '../lib/data_set'
5  require_relative '../lib/svm'
6  require_relative '../lib/study'
7  require_relative '../lib/blank_formatters'
8
9  data_folder="../test/theory_data/"
10 results_folder="./svm_combinations_sigmoid_3"
11
12
13 set = MachineLearning::DataSet.new(data_folder+"sat6c.tra",36,:float)
14
15 set.file_writer= MachineLearning::SvmFileWriter.new
16 set.line_writer = MachineLearning::SvmPatternWriter.new
17 set.line_formatter = MachineLearning::BlankSVMPatternFormatter.new
18
19 set.read_data
20
21 study_params = Array.new
22
```

```
23
24
25 kernel_type=MachineLearning::Parameter.new(:kernel_type)
26 kernel_type.fix_set([3])
27 study_params << kernel_type
28
29 algo_param = MachineLearning::Parameter.new(:algo)
30 algo_param.fix_set(["DAGSVM"])
31 study_params << algo_param
32
33 trade_off = MachineLearning::Parameter.new(:trade_off)
34
35 trade_off.fix_set([10.0])
36
37 study_params << trade_off
38
39 s_sigmoid_factor = MachineLearning::Parameter.new(:s_sigmoid_factor)
40
41 s_sigmoid_factor.fix_set([0.00001])
42
43 study_params << s_sigmoid_factor
44
45 c_sigmoid_factor = MachineLearning::Parameter.new(:c_sigmoid_factor)
46
47 c_sigmoid_factor.fix_set([1])
48
49 study_params << c_sigmoid_factor
50
51 study = MachineLearning::Study.new(results_folder,:SvmCombinations,study_params,set,5,"results.txt","st
52
53 study.generate_experiments
54
55 study.run
```

2.3 Software Output

The software generates the following output files that will help us out for the review of the classification training:

Study List: Is a detailed print out indicating the parameters used in the experiment as well as the pattern files and output files in case the experiment wants to be reviewed in detail:

```
--------------------------------------------------------------------------------
0
{:kernel_type=>2, :algo=>"SVM_Class", :gamma_rbf_factor=>0.5, :id=>0,
 :test_pat=>"./svm_combinations_rbf/20110524-230325_test.pat",
 :val_pat=>"./svm_combinations_rbf/20110524-230325_val.pat",
 :train_pat=>{
"1-6"=>"./svm_combinations_rbf/20110524-230325_train.pat_1-6.pat",
"1-5"=>"./svm_combinations_rbf/20110524-230325_train.pat_1-5.pat",
"1-4"=>"./svm_combinations_rbf/20110524-230325_train.pat_1-4.pat",
"1-3"=>"./svm_combinations_rbf/20110524-230325_train.pat_1-3.pat",
"1-2"=>"./svm_combinations_rbf/20110524-230325_train.pat_1-2.pat",
"2-6"=>"./svm_combinations_rbf/20110524-230325_train.pat_2-6.pat",
"2-5"=>"./svm_combinations_rbf/20110524-230325_train.pat_2-5.pat",
"2-4"=>"./svm_combinations_rbf/20110524-230325_train.pat_2-4.pat",
"2-3"=>"./svm_combinations_rbf/20110524-230325_train.pat_2-3.pat",
"3-6"=>"./svm_combinations_rbf/20110524-230325_train.pat_3-6.pat",
"3-5"=>"./svm_combinations_rbf/20110524-230325_train.pat_3-5.pat",
"3-4"=>"./svm_combinations_rbf/20110524-230325_train.pat_3-4.pat",
"4-6"=>"./svm_combinations_rbf/20110524-230325_train.pat_4-6.pat",
"4-5"=>"./svm_combinations_rbf/20110524-230325_train.pat_4-5.pat",
"5-6"=>"./svm_combinations_rbf/20110524-230325_train.pat_5-6.pat"},
:out_model=>"./svm_combinations_rbf/20110524-230325SVM_Class.net",
:test_res=>"./svm_combinations_rbf/20110524-230325test.res",
:val_res=>"./svm_combinations_rbf/20110524-230325val.res"}
--------------------------------------------------------------------------------
```

Result File: For each of the parameter combinations a result is printed out containing the mean error, wrong and right classification % for the validation and test set over the cross training.

SVM model: The trained model file generated by svm_learn or svm_multiclass_learn.

3 Experimental Results

3.1 Experimentation Methodology

For the SVMs experiments a cross validation over 5 blocks will be performed for each experiment. The data will be composed of the public training data: sat6c.tra

Once selected the best resulting parameters for each of the SVM algorithms and Kernels we will perform a final evaluation by performing training a SVM with the whole of sat6c.tra and test it against the public test set sat6c.public.tst.

We will not use sat6c.public.tst other than for final evaluation to ensure that we not indirectly over fit the set by using it when exploring the solution space.

Due to time constraints on the experimentation we will perform the parameter search with the DAGs SVM and then compare that best result with the same error we get with the SVM Multiclass.

3.2 Polynomial Kernel

We initially performed a base DAGs SVM for the polynomial kernel of grades 1, 2 and 3 with the default values for all other variables:

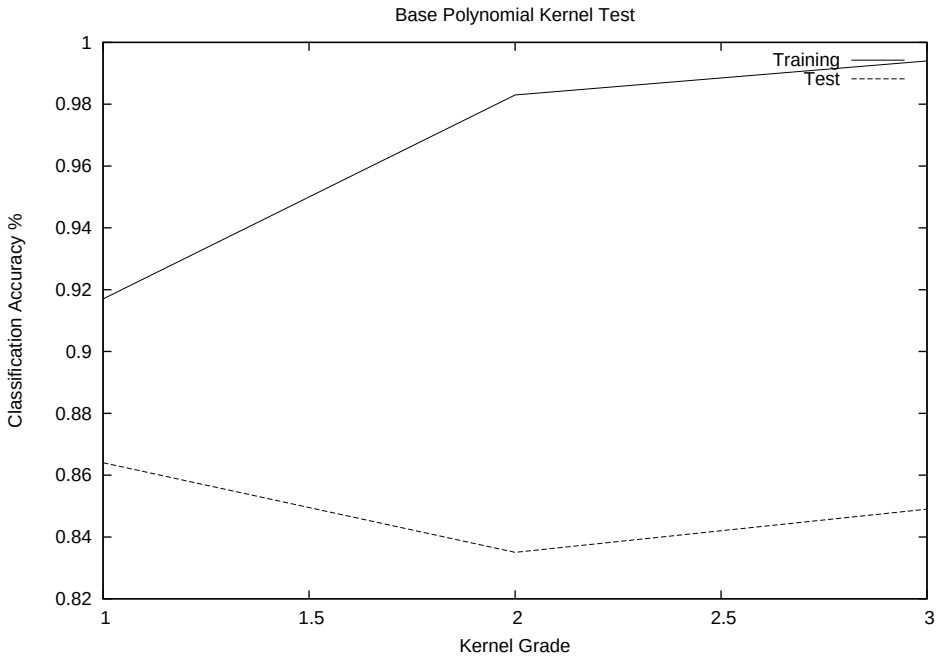

Figure 1: Plot shows the classification accuracy (%) as a function of the grade of the polynomial kernel used to train the SVM for the Statlog task. Each plotted point is an average computed with 5-block cross validation over the public training set sat6c.tra.

Next we perform a full range search varying the tradeoff parameter from 0.00001 up to 10.0 and the "s" parameter with the following set: 1,2,4,8,16 for a total of 90 experiments (if we consider the 5 block cross validation that means a total of 450 trainings).

Polynomial Degree	trade off	s parameter	c parameter	training accuracy	test accuracy
1	0.00001	1	1	0.844	0.839
1	0.00001	16	1	0.887	0.871
1	0.00001	2	1	0.855	0.849
1	0.00001	4	1	0.866	0.858
1	0.00001	8	1	0.879	0.862
1	0.0001	1 1	0	.882 0	.866
1	0.0001	16	1	0.907	0.872
1	0.0001	2	1	0.89	0.869
1	0.0001	4	1	0.894	0.88
1	0.0001	8	1	0.903	0.875
1	0.001	1	1	0.905	0.872
1	0.001	16	1	0.919	0.863
1	0.001	2	1	0.907	0.874
1	0.001	4	1	0.911	0.871
1	0.001	8	1	0.917	0.861
1	0.1	1	1	0.925	0.854
1	0.1	16	1	0.926	0.846
1	0.1	2	1	0.924	0.853
1	0.1	4	1	0.924	0.855
1	0.1	8	1	0.924	0.854
1	1.0	1	1	0.924	0.85
1	1.0	16	1	0.804	0.743
1	1.0	2	1	0.926	0.856
1	1.0	4	1	0.92	0.852
1	1.0	8	1	0.887	0.821
1	10.0	1	1	0.831	0.755
1	10.0	16	1	0.758	0.703
1	10.0	2	1	0.794	0.732
1	10.0	4	1	0.688	0.641
1	10.0	8	1	0.685	0.642

Polynomial Degree	trade off	s parameter	c parameter	training accuracy	test accuracy
2	0.00001	1	1	0.985	0.857
2	0.00001	16	1	0.972	0.836
2	0.00001	2	1	0.977	0.849
2	0.00001	4	1	0.976	0.828
2	0.00001	8	1	0.978	0.826
2	0.0001	1	1	0.981	0.839
2	0.0001	16	1	0.982	0.85
2	0.0001	2	1	0.98	0.841
2	0.0001	4	1	0.977	0.835
2	0.0001	8	1	0.968	0.847
2	0.001	1	1	0.977	0.841
2	0.001	16	1	0.98	0.85
2	0.001	2	1	0.985	0.85
2	0.001	4	1	0.978	0.848
2	0.001	8	1	0.977	0.829
2	0.1	1	1	0.976	0.845
2	0.1	16	1	0.98	0.856
2	0.1	2	1	0.973	0.834
2	0.1	4	1	0.979	0.841
2	0.1	8	1	0.982	0.854
2	1.0	1	1	0.979	0.849
2	1.0	16	1	0.981	0.843
2	1.0	2	1	0.973	0.846
2	1.0	4	1	0.981	0.846
2	1.0	8	1	0.976	0.84
2	10.0	1	1	0.977	0.841
2	10.0	16	1	0.984	0.837
2	10.0	2	1	0.978	0.84
2	10.0	4	1	0.975	0.834
2	10.0	8	1	0.974	0.849

Polynomial Degree	trade off	s parameter	c parameter	training accuracy	test accuracy
3	0.00001	1	1	0.991 0.834	
3	0.00001	16	1	0.926 0.801	
3	0.00001	2	1 0.985 0.846		
3	0.00001	4	1 0.983 0.832		
3	0.00001	8	1 0.946 0.821		
3	0.0001	1	1	0.996	0.831
3	0.0001	16	1	0.931	0.797
3	0.0001	2	1	0.973	0.835
3	0.0001	4	1	0.981	0.845
3	0.0001	8	1	0.931	0.805
3	0.001	1	1	0.992	0.844
3	0.001	16	1	0.953	0.813
3	0.001	2	1	0.986	0.845
3	0.001	4	1	0.973	0.831
3	0.001	8	1	0.924	0.788
3	0.1	1	1	0.996	0.85
3	0.1	16	1	0.91	0.779
3	0.1	2	1	0.987	0.834
3	0.1	4	1	0.981	0.853
3	0.1	8	1	0.92	0.788
3	1.0	1	1	0.997	0.85
3	1.0	16	1	0.979	0.848
3	1.0	2	1	0.994	0.85
3	1.0	4	1	0.922	0.796
3	1.0	8	1	0.95	0.816
3	10.0	1	1	0.993	0.846
3	10.0	16	1	0.92	0.79
3	10.0	2	1	0.991	0.84
3	10.0	4	1	0.963	0.825
3	10.0	8	1	0.94	0.815

The best result accuracy result 0.88 was obtained with a polynomial degree of 1, trade off set to 0.0001 and s parameter to 4.

3.3 Radial Basis Function Kernel

Usage of the RBF was more complicated as the tradeoff factor and the gamma factor required a certain configuration otherwise the accuracy results where very low.

We performed an initial experimentation where we:

- Varied the trade-off between: 1.0, 0.1 , 0.01, 0.001, 0.0001
- Varied the gamma factor: 1.0 , 0.1 , 0.01, 0.001, 0.0001, 0.00001, 0.000001

35 experiments with 5 block cross validation for a total of 175 experiments.

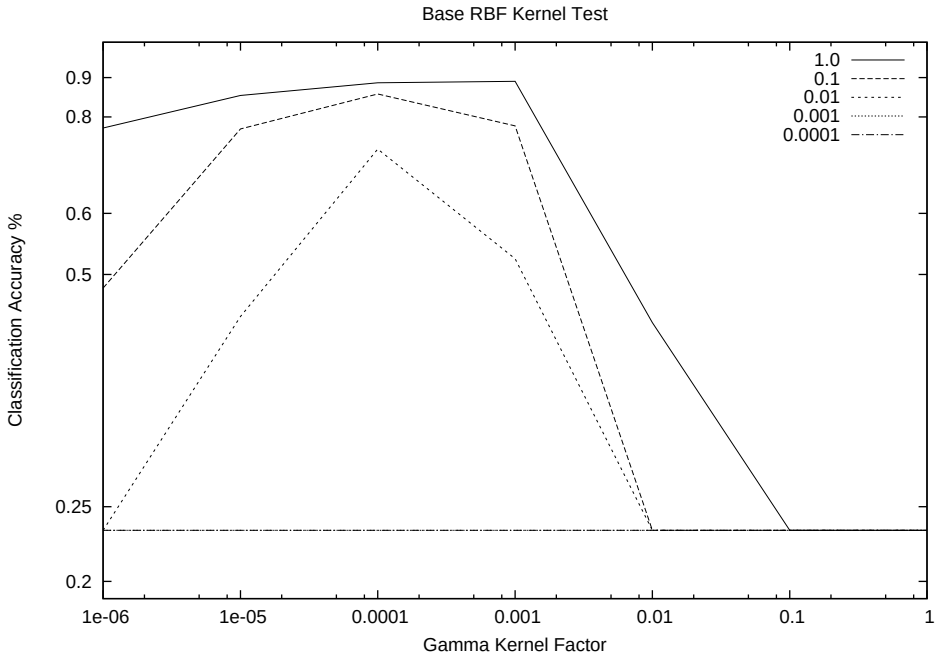

Figure 2: Plot shows the classification accuracy (%) as a function of the gamma factor used to train the SVM for the Statlog task for different trade off values with logarithmic axis. Each plotted point is an average computed with 5-block cross validation over the public training set sat6c.tra.

As we saw that the best results occurred for large trade off values with small gamma factors, in fact for the lowest trade offs of 0.001 and 0.0001 classification rate was very bad. We performed a more precise experimentation to review large trade offs with low gamma factors where we:

- Varied the trade-off between: 1.0 , 2.0 , 5.0 , 10.0
- Varied the gamma factor: 0.001, 0.0001, 0.00001

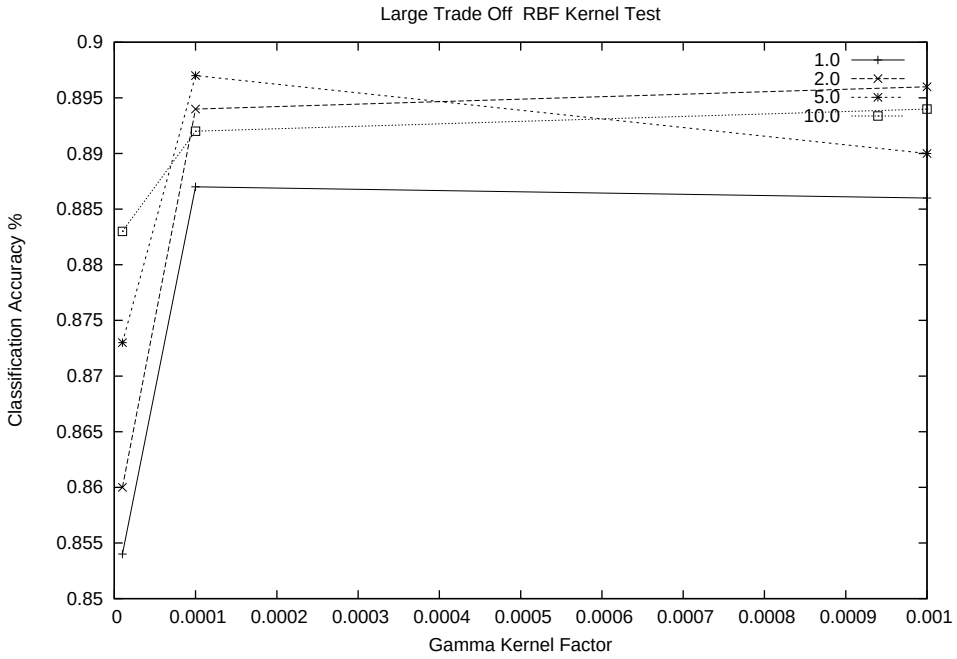

Figure 3: Plot shows the classification accuracy (%) as a function of the gamma factor used to train the SVM for the Statlog task for different trade off values. Each plotted point is an average computed with 5-block cross validation over the public training set sat6c.tra.

Finally as we saw the best results where obtained for 0.0001 gamma factor value we performed another batch to see if there was any gain when varying the trade off between 3.0 and 7.0.

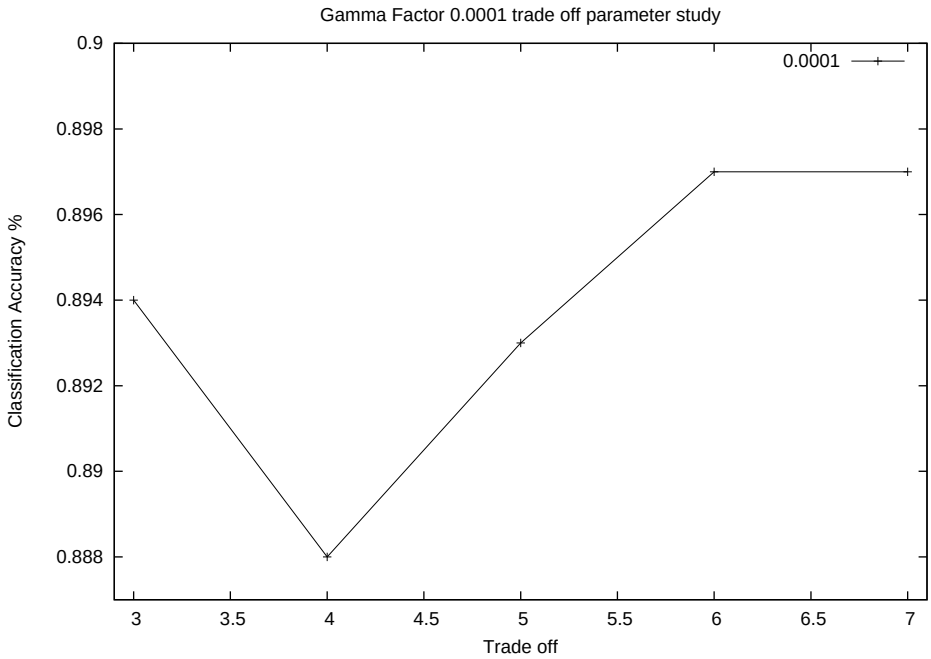

Figure 4: Plot shows the classification accuracy (%) as a function of the trade off parameter used to train the SVM for the Statlog task. Each plotted point is an average computed with 5-block cross validation over the public training set sat6c.tra.

As we have seen the best RBF kernel accuracy for the DAGS SVM has been obtained with gamma factor RBF 0.0001 and trade off 6.0.

3.4 Sigmoid Kernel

For the sigmoid kernel we performed an experiment varying the following:

- Trade off was reviewed for: 10.0, 1.0, 0.1, 0.001, 0.0001 and 0.00001
- S sigmoid factor for: 10.0, 2.0 and 1.0

The results where quite bad all models trained obtained just 0.234 accuracy over the train set and 0.233 over the cross validation.

None of the resulting 24 experiments (120 trainings in total) gave good performance. As there was no more time to review other parameter ranges this kernel type is initially discarded due to its bad performance.

4 Conclusions

We now perform a separate test using the best parameters for each of the kernel types and test it agains the sat6c.public.tst separate data set not used will performing the experimentation:

Table 1: Public test set - best error results for kernels

Model	Polynomial Kernel	RBF Kernel	Sigmoid Kernel
DAG SVM	0.8621	0.8971	0.233
Multiclass SVM	0.77	0.515	0.106

As we can see from the results above:

- The error obtained during the classification seems to represent adequately de real error we will get on unseen data.
- On the whole 2 class SVM combined by means of the DAG technique worked better than Multiclass SVM for this task.
- Training time for DAG SVM was much lower than for Multiclass SVM.
- Classification time for DAG SVM was higher than for Multiclass SVM due to havint to perform so many classifications to get a final result.
- The best result for the task was 0.897 which is a couple of points over the NN classifier result of 0.885 reported for the task.