Advances in Neural Networks
Course Project - Terrain Classification

Vicente Bosch Campos †
viboscam@posgrado.upv.es

May 25, 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.

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.

Snns: The Snns class can be found in the *./lib/snns.rb* file. It is an object wrapper for the Batchman snns interface and allows us to perform the following tasks from the framework:

- The class allows as input an experiment object, it automatically generates an specific batchman script file and runs it.
- Permits us to programmatically access to the results of the .res files when reviewed with the analyze tool.

SnnsFileWriter: The SnnsFileWriter class can be found in the *./lib/snns.rb* file. The file writer is in charge of performing header and footer special formatting for the toolset chosen. In this case it allows us to call mkhead from the framework to prepare the pattern files.

SnnsPatternWriter: It can be found in *./lib/snns.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 it ensures there is a space between each dimension value and ensures there is a line space between the vector and the class label.

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 train neural networks for the chromosome classification task:

Listing 1: chromosome.rb

```ruby
#! /usr/bin/env ruby

require 'ap'
require_relative '../lib/data_set'
require_relative '../lib/svm'
require_relative '../lib/study'
require_relative '../lib/blank_formatters'

data_folder="../test/theory_data/"
results_folder="./svm_combinations_rbf"


set = MachineLearning::DataSet.new(data_folder+"sat6c.tra",36,:float)

set.file_writer= MachineLearning::SvmFileWriter.new
set.line_writer = MachineLearning::SvmPatternWriter.new
set.line_formatter = MachineLearning::BlankSVMPatternFormatter.new

set.read_data

study_params = Array.new



kernel_type=MachineLearning::Parameter.new(:kernel_type)
kernel_type.fix_set([2])
study_params << kernel_type

algo_param = MachineLearning::Parameter.new(:algo)
algo_param.fix_set(["SVM_Class"])#"RadialBasisLearning"])
study_params << algo_param

gamma_rbf_factor = MachineLearning::Parameter.new(:gamma_rbf_factor)

gamma_rbf_factor.numeric_range(0.5,3,0.5)

study_params << gamma_rbf_factor

#d_polynomial_factor = MachineLearning::Parameter.new(:d_polynomial_factor)

#d_polynomial_factor.numeric_range(1,3,1)

#study_params << d_polynomial_factor

#ap study_params

study = MachineLearning::Study.new(results_folder,:SvmCombinations,study_params,set,5,"results.txt","st

study.generate_experiments

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:

```
------------------------------------------------------------------------------------
1
{:algo=>"Quickprop", :in_model=>"../test/theory_data/trr_36_20_6.net", :learning=>0.15000000000000002, :max_growth=>0.
:id=>1, :test_pat=>"./QP30_20_6/20110209-000607_test.pat", :val_pat=>"./QP30_20_6/20110209-000607_val.pat",
:train_pat=>"./QP30_20_6/20110209-000607_train.pat", :out_model=>"./QP30_20_6/20110209-000607Quickprop.net",
:test_res=>"./QP30_20_6/20110209-000607test.res", :val_res=>"./QP30_20_6/20110209-000607val.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.

Log files: Contains an execution log of the Batchman execution with the training and validation MSE.

Bat files: Automatically generated Batchman script for the experiment.

Snns standard files: It also generates and saves the following SNNS files:

- Result files of the validation and test sets.
- Pattern files used in the training.
- Resulting neural network.

3 Experimental Results

3.1 Neural Networks

4 Conclusions

As we have seen in the previous tasks the best results have been obtained for the Radial based learning. The best results for each algorithm in the private test set has been as follows:

Table 1: Private test set - best error results for training algorithms

Back-propagation	Quick-propagation	Radial Network
10.40	15.50	9.60

We have obtained the best results for Back-propagation and Radial based learning, being our radial network the network with least classification task in the board.

In future work it would be interesting to use the SVM software that has been integrated in the framework to study this task.