CS440 : Introduction to Artificial Intelligence **Spring 2014**

Problem Set 4 - Problem 2

Dan McQuillan *Handed In: April 8, 2014*

1. **Solution to problem 2**

   (a) Explain how kernel functions affect the SVM

   The degree of the polynomial kernel and the phi parameter of the Gaussian kernel control the flexibility of the resulting SVM in fitting the data. This can affect the speed of convergence of the SVM's training as well.

   (b) Why would some kernel functions perform better than others?

   If the complexity is too large (degree too high, wrong phi value for Gaussian) then over fitting will occur since the decision boundary will only perform well on the training data and will lose generality since the decision boundary could eventually become the dataset with a high enough complexity.

   (c) Train the classifier for each of the following kernel functions

Kernel Function	Time taken(s)	Accuracy
i Linear or string kernel	0.826	60.2803738317757%
ii Homogeneous quadratic kernel	0.325	61.44859813084112%
iii Nonhomogeneous quadratic kernel	0.293	62.77258566978193%
iiv Homogeneous cubic kernel	0.353	63.43457943925234%

1 Source code:

1.1 MainP2.java

```java
package hw4.weka;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Random;

import weka.classifiers.evaluation.Evaluation;
import weka.classifiers.evaluation.output.prediction.XML;
import weka.classifiers.functions.SMO;
import weka.classifiers.functions.supportVector.Kernel;
import weka.classifiers.functions.supportVector.PolyKernel;
import weka.core.Instances;

```

```
public class MainP2 {

  public static PolyKernel buildKernel(final Integer cacheSize, final Double
      exponent, final Boolean useLowerOrder, final Instances instances )
      throws Exception {
    PolyKernel polyKernel = new PolyKernel();

    // Linear Kernel
    polyKernel.setExponent( exponent );
    polyKernel.setCacheSize( cacheSize );
    polyKernel.setUseLowerOrder( useLowerOrder );
    polyKernel.buildKernel( instances );

    return polyKernel;
  }

  public static void main(String[] args)  {
    try {
      File file = new File("glass.arff");
      BufferedReader bufferedReader = new BufferedReader(new FileReader(file))
          ;



      Instances instances = new Instances(bufferedReader);

      bufferedReader.close();

      final int numInstances = instances.numAttributes();
      instances.setClassIndex(numInstances - 1);

      // Part c
      Instances trainInstances = instances;
      final Integer folds = 10;

      StringBuilder applicationOutput = new StringBuilder();

      applicationOutput.append("Begin problem 2 part c\n");
      applicationOutput.append("folds: " + folds + "\n\n");


      StringBuffer internalStringBuffer = new StringBuffer();
      XML internalOutput = new XML();
      internalOutput.setBuffer(internalStringBuffer);
      internalOutput.setHeader(trainInstances);
      internalOutput.setOutputDistribution(true);

      Evaluation evaluation = new Evaluation(trainInstances);

      PolyKernel linearKernel = buildKernel(0, 1.0, false, trainInstances);
      PolyKernel homogeneousQuadraticKernel = buildKernel(0, 2.0, false,
          trainInstances);
      PolyKernel nonhomogeneousQuadraticKernel = buildKernel(0, 2.0, true,
          trainInstances);
```

```java
      PolyKernel homogeneousCubickernel = buildKernel(0, 3.0, false, trainInstances);

      List<PolyKernel> kernels = new ArrayList<PolyKernel>();
      kernels.add(linearKernel);
      kernels.add(homogeneousQuadraticKernel);
      kernels.add(nonhomogeneousQuadraticKernel);
      kernels.add(homogeneousCubickernel);

      for( Kernel kernel : kernels ) {

        applicationOutput.append( "Evaluating using kernel function: " + kernel.toString() + "\n" );

        SMO classifier = new SMO();
        classifier.setKernel(kernel);
        Date beforeCrossValidate = new Date();
        evaluation.crossValidateModel(classifier, trainInstances, folds, new Random( new Date().getTime() ), internalOutput );
        Date afterCrossValidate = new Date();

        Double timeTakenCrossValidation =  ((double)( afterCrossValidate.getTime() - beforeCrossValidate.getTime())) / 1000.0;
        applicationOutput.append("Time taken: " + timeTakenCrossValidation + "\n");
        applicationOutput.append( "Accurracy: " + evaluation.pctCorrect() + "%\n\n");
      }

      System.out.println(applicationOutput.toString());
    } catch (IOException e) {
      System.err.println( e );
    } catch (Exception e) {
      e.printStackTrace();
    }

  }
}
```