Introduction to Data Mining

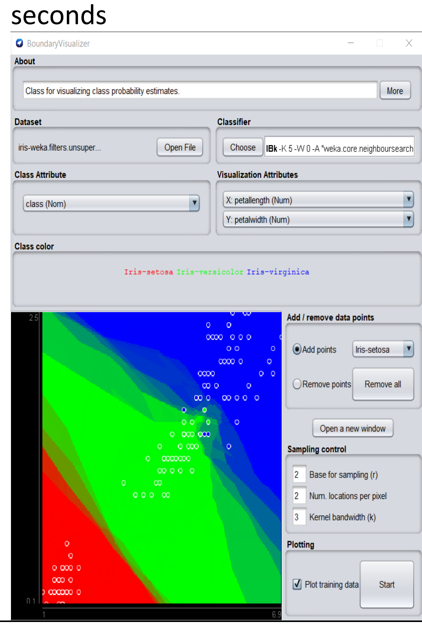
Lab 4: More Classifiers

Name: Phạm Đức Đạt
ID: ITITIU20184

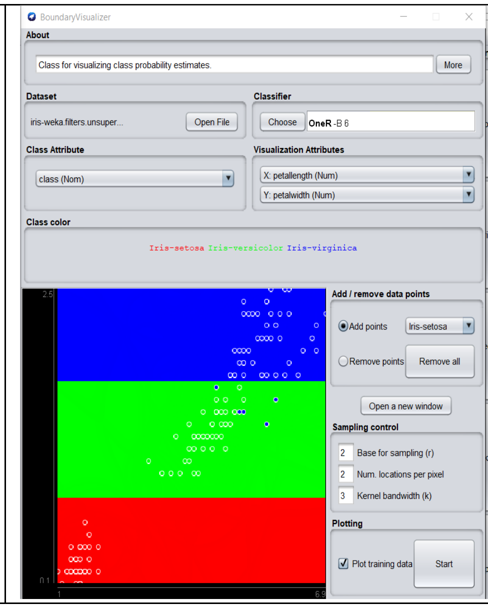
4.1. Classification boundaries

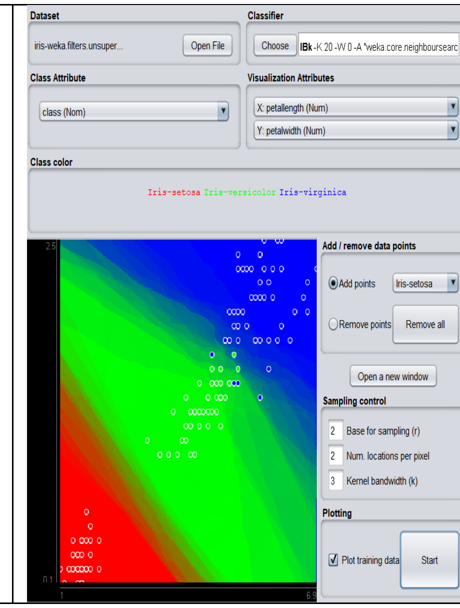
In the fifth class, we are going to look at some machine learning methods used to classify datasets in Weka. (See the lecture of class 4 by Ian H. Witten, [1][1]). We are going to learn about linear regression, classification by regression, and support vector machines.

In this section, we are going to start by looking at classification boundaries for different machine learning methods. We are going to use Weka's **Boundary Visualizer**, and a 2-dimensional datasets. Follow the instructions in [1] to do some experiments, and then fill in the following table with the **classifier models**.

Dataset	**Rules → OneR**	**Lazy → IBk**	
		K=5	K=20
Iris.2D.arff	=== Classifier model (full training set) === petalwidth: < 0.8 -> Iris-setosa < 1.75 -> Iris-versicolor >= 1.75 -> Iris-virginica (144/150 instances correct)	IB1 instance-based classifier using 5 nearest neighbour(s) for classification Time taken to build model: 0 seconds 	=== Classifier model (full training set) === IB1 instance-based classifier using 20 nearest neighbour(s) for classification

[1] http://www.cs.waikato.ac.nz/ml/weka/mooc/dataminingwithweka/





Try other learning methods, e.g NaiveBayes using SupervisedDiscretization, i.e. supervised discretization is to take the classes into account when discretizing numeric attributes into ranges... [Refer to Text [2]. Chapter 7 for discretization part]

<table>
<tr><th rowspan="2">Dataset</th><th rowspan="2">Bayes > NaiveBayes</th><th colspan="2">Trees > J48</th></tr>
<tr><th>minNumbObj = 5</th><th>minNumbObj = 10</th></tr>
<tr>
<td>Iris.2D.arff</td>
<td>=== Classifier model (full training set) ===

Naive Bayes Classifier

Class
Attribute Iris-setosa Iris-versicolor Iris-virgi
(0.33) (0.33) (0
==
petallength
mean 1.4694 4.2452 5
std. dev. 0.1782 0.4712 0
weight sum 50 50
precision 0.1405 0.1405 0

petalwidth
mean 0.2743 1.3097 2
std. dev. 0.1096 0.1915 0
weight sum 50 50
precision 0.1143 0.1143 0

Time taken to build model: 0 seconds</td>
<td>J48 pruned tree
----

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
| petalwidth <= 1.7
| | petallength <= 4.9: Iris-versicolor (48.0/
| | petallength > 4.9: Iris-virginica (6.0/2.0
| petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves : 4

Size of the tree : 7</td>
<td>J48 pruned tree
----

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
| petalwidth <= 1.7: Iris-versicolor (54.0
| petalwidth > 1.7: Iris-virginica (46.0/1

Number of Leaves : 3

Size of the tree : 5</td>
</tr>
</table>

4.2. Linear regression

In this section, we are going to deal with numeric classes using a classical statistical method.

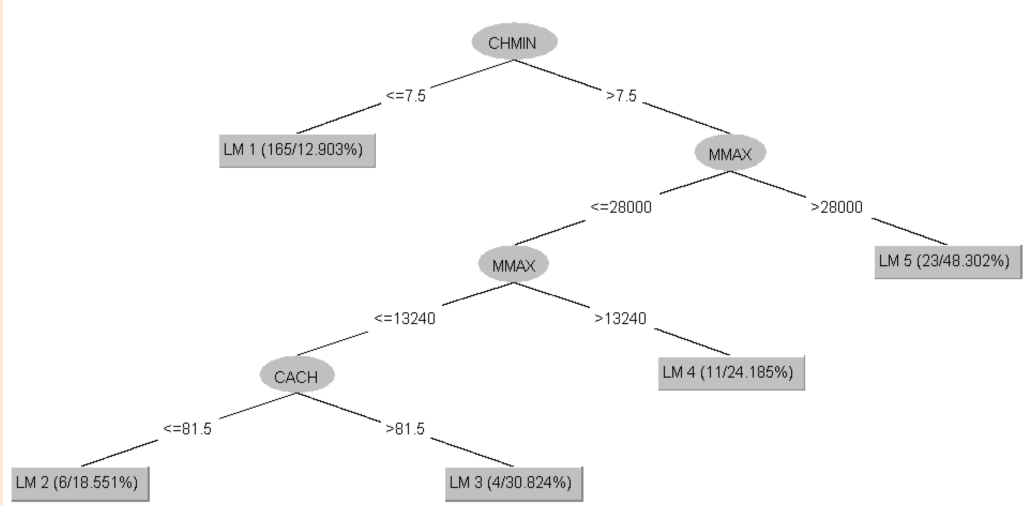
Follow the lecture of linear regression in [1] to learn how to calculate weights of attributes from training data, and make predictions. [Refer to Text [2]. Chapter 4.6 for linear regression part]

Follow the instructions in [1] to examine the model of **linear regression** on the **cpu** dataset.

Write down the results in the following table:

Dataset	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
Cpu	0.9012	41.0886	69.556	42.6943 %	43.2421 %
Linear Regression Model	class = 0.0491 * MYCT + 0.0152 * MMIN + 0.0056 * MMAX + 0.6298 * CACH + 1.4599 * CHMAX + -56.075				

Do again to examine **M5P** on the **cpu** dataset, and then write down the results in the following table:

Dataset	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
Cpu	0.9274	29.8309	60.7112	30.9967	37.7434
Classifier model	M5 pruned model tree: 				

<table>
<tr><td></td><td>Linear regression models:

Linear Regression Model

class =

      0.0491 * MYCT +
      0.0152 * MMIN +
      0.0056 * MMAX +
      0.6298 * CACH +
      1.4599 * CHMAX +
    -56.075</td></tr>
</table>

Is **M5P** non-linear regression? - – No, M5P is a tree model

4.3. Classification by regression

Follow the instructions in [1] to investigate two-class classification by regression, using the **diabetes** dataset.

We are going to convert the nominal class to the numeric class so that the linear regression model is applicable.

Write down the results in the following table:

<table>
<tr><th>Classifier model</th><th>Evaluation</th></tr>
<tr><td>Linear Regression Model

class=tested_positive =

      0.0209 * preg +
      0.0057 * plas +
     -0.0024 * pres +
      0.0131 * mass +
      0.1403 * pedi +
      0.0028 * age +
     -0.8363</td><td></td></tr>
</table>

Correlation coefficient 0.5322 Mean absolute error 0.3366 Root mean squared error 0.4036 Relative absolute error 74.0119 % Root relative squared error 84.6013 % Total Number of Instances 768	

4.4. Support vector machines

Learn about logistic regression in [2]. Chapter 4.6

Follow the lecture of support vector machines (SVMs) in [1], …

Support vector machines (SVMs, also **support vector networks** [1]) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

Follow the instructions in [1] to examine **SMO** and **LibSVM**, and fill in the following table:

Dataset	SMO's classifier model and performance	LibSVM's classifier model and performance

diabetes	SMO
	
	Kernel used:
	  Linear Kernel: K(x,y) = <x,y>
	
	Classifier for classes: tested_negative, tested_positive
	
	BinarySMO
	
	Machine linear: showing attribute weights, not support vectors.
	
	        1.3614 * (normalized) preg
	 + 4.8764 * (normalized) plas
	 + -0.8118 * (normalized) pres
	 + -0.1158 * (normalized) skin
	 + -0.1776 * (normalized) insu
	 + 3.0745 * (normalized) mass
	 + 1.4242 * (normalized) pedi
	 + 0.2601 * (normalized) age
	 - 5.1761
	
	Number of kernel evaluations: 19131 (69.279% cached)
	
	Correctly Classified Instances 594 77.3438 %
	Incorrectly Classified Instances 174 22.6563 %
	Kappa statistic 0.4682
	Mean absolute error 0.2266
	Root mean squared error 0.476
	Relative absolute error 49.848 %
	Root relative squared error 99.862 %
	Total Number of Instances 768
		=== Classifier model (full training set) ===
		
		LibSVM wrapper, original code by Yasser EL-Manzalawy (= WLSVM)
		
		Time taken to build model: 0.14 seconds
		
		=== Stratified cross-validation ===
		=== Summary ===
		
		Correctly Classified Instances 500 65.1042 %
		Incorrectly Classified Instances 268 34.8958 %
		Kappa statistic 0
		Mean absolute error 0.349
		Root mean squared error 0.5907
		Relative absolute error 76.7774 %
		Root relative squared error 123.9347 %
		Total Number of Instances 768

Notice: A wrapper class for the libsvm tools (the libsvm classes, typically the jar file, need to be in the classpath to use this classifier) >> see http://weka.wikispaces.com/LibSVM