School of Computing Sciences

Module: CMP-6002B Machine Learning
Assignment: Coursework Assignment

Set by: Anthony Bagnall (ajb@uea.ac.uk)
Checked by: Jason Lines (Jason.Lines@uea.ac.uk)
Date set: Tuesday 4th February 2020
Value: 40%

Date due: Wednesday 29th April 2020 3pm **(Week 12)**
Returned by: *date missing*
Submission: Blackboard

Learning outcomes

The student will learn about implementing variants of linear perceptron classifiers and ensemble techniques and the general issues involved with implementing any classifier. They will understand better how to evaluate and compare classifiers on a range of data and for a specific problem. They will appreciate the difficulty in presenting technical results in a way to support or refute a hypothesis. The exercises will improve the following transferable skills: programming in Java; analysis of data and results; writing clearly and concisely; presenting data coherently.

Specification

Overview

The task for this assignment is to implement variants of linear perceptron classifiers and to evaluate/compare them on a range of real world data.

Part 1: Linear Perceptron by Hand (10%)

The purpose of this exercise is to make sure you understand how the algorithm works and to provide you with a bench check for later implementations.
Use the on-line perceptron algorithm to fit a linear model to the training data shown in Table 1 below, where x_1 and x_2 are the explanatory (input) variables and $t \in \{-1, +1\}$ represents the response variable. The training examples should be presented in the order shown in the table; use a linear model without a bias term, where the initial value of the weight vector is $[1,\ 1]$, and assume a learning rate of 1.
Show all working, as discussed in the seminar.

Implement Linear Perceptron Classifiers (50%)

This task involves implementing variants of linear perceptron classifiers in Weka. Please note that this must be your own original code. We are of course aware of existing implementations

Table 1: Training data for Part 1

x_1	x_2	t
-4.9	+0.4	+1
+4.6	+4.4	-1
+7.0	-5.3	-1
+2.4	-5.1	+1
-5.7	+7.4	+1
+7.3	-4.0	-1
-0.9	-8.0	+1
+0.5	+9.3	-1

of the classifiers in Weka and other packages. Copying from existing source code will be considered possible plagiarism and will be dealt with accordingly. Please note that you can assume that all attributes are real valued for all the implementations (i.e. you do not need to handle nominal attributes in the learning algorithm).

Implement `LinearPerceptron` (15%)

Implement a standard on-line Linear Perceptron classifier by implementing the Weka `Classifier` interface (or implementing `AbstractClassifier`). Call this `LinearPerceptron`. `LinearPerceptron` should only work when all attributes are continuous (i.e. assume all the variables take real values rather than being discrete). If this is not the case it should throw an exception. You may use the Weka `Capabilities` facility if you wish. The method `buildClassifier` should create the linear model by iterating over the training data and applying the on-line Perceptron rule (called PerceptronTraining in the lecture notes). You should allow for the possible inclusion of a constant (bias) term and include a stopping condition, such as the number of iterations to perform. The method `classifyInstance` should apply the model to the new instance then apply a sensible decision rule to the resulting linear prediction. Test your classifier on the example data given in question 1.

Implement `EnhancedLinearPerceptron` (20%)

Enhance your `LinearPerceptron` class by adding the following features

1. **Standardise attributes** (default to true). Each attribute should be standardised to zero mean and unit standard deviation when the flag is set to true. This should be done in `buildClassifier`, but will also need to be performed in `classifyInstance` using the statistics calculated in `buildClassifier` (7%).

2. **Alternative learning algorithm**. A method to use an off-line update method instead of the on-line rule to update the classifier. Include a flag to determine which method to use (defaults to the on-line rule). (7%)

3. **Model selection** (default to false). Write a method to determine whether to use on-line or off-line update based on the training cross validation error. Model selection should be performed in `buildClassifier` if a flag is set to true. (6%)

Implement `LinearPerceptronEnsemble` (15%)

Implement a third classifier, `LinearPerceptronEnsemble`, that consists of an ensemble of `LinearPerceptron` (or `EnhancedLinearPerceptron`) classifiers stored as an array or List. Set the default ensemble size to 50. The method `buildClassifier` should construct a new set of instances for each element of the ensemble by selecting a random subset of the attributes. The proportion of attributes to select should be a parameter (defaults to 50%). It should then build a separate classifier on each `Instances` object. Further diversity should be injected into the ensemble by randomising the starting condition. The `LinearPerceptronEnsemble` will need to store which attributes are used with which classifier in order to recreate the attribute selections in `classifyInstance` and `distributionForInstance`. Implement `classifyInstance` so that it returns the majority vote of the ensemble. i.e., classify a new instance with each of the linear perceptrons, count how many classifiers predict each class value, then return the class that receives the largest number of votes. Implement `distributionForInstance` so that it returns the proportion of votes for each class.

Evaluation of Linear Perceptron Classifiers (40%)

Your task is to perform a series of classification experiments and write them up as a research paper. Your experiments will address the question of whether the variations you have implemented for linear perceptron improve performance over a range of problems and whether a linear perceptron is a good approach for a specific case study data set. You have been assigned a specific data set (see blackboard) and a link to further information. Please note that the for this part of the coursework we mark the paper itself, not the code used to generate the results. We advise you reserve significant time for reading and checking your paper, and recommend you ask someone else to read it before submission. Imagine you are writing a paper for a wide audience, not just for the marker. Aim for a tight focus on the specific questions the paper addresses. There are four issues to investigate:

1. Test whether tuning the parameters is better than the default linear perceptron on the 30 classification problems we have provided.
2. Test whether your ensemble is better than a single default valued linear classifier and a tuned linear classifier on the 30 classification problems.
3. Compare your ensemble against a range of built in Weka classifiers on the 30 classification problems.
4. Perform a case study on your assigned data set to propose which classifier from those you used in part 3 would be best for this particular problem.

Experiments

You should compare classifiers based on accuracy (or, equivalently, error) and any other metrics described in the evaluation lecture that you feel are appropriate. The first hypothesis requires a comparison of two classifiers over multiple data sets. There second compares three classifiers over multiple data sets. The third requires comparing multiple classifiers over multiple data sets. The final case study involves comparing multiple classifiers on a single dataset. You should think about the experimental design for each experiment, including deciding on a sampling method (train/test, cross validate or resample) and performance measures. The choice of classifiers for part 3 is up to you, but we recommend you include random forest and rotation

forest in the list. Be clear on the parameter settings of all classifiers. You should also compare the time each algorithm takes to build a model.

The Write Up

You should write a paper in Latex using the style file available on blackboard. The paper should be called "An experimental assessment of linear perceptrons". This should be around 4 pages, but there is no maximum or minimum limit. Your paper should include the following sections:

1. **Introduction**: start with the aims of the paper, a description of the hypotheses you are testing and an overview of the structure. Include in this your prior beliefs as to what you think the outcome of the test will be, with some rationalisation as to why. So, for example, do you think cross validation for k will make a difference? Can you find any published literature that claims to answer any of the questions?

2. **Data Description**: an overview describing the data, including data characteristics summarised in a table (number of attributes, number of train/test cases, number of classes, class distribution). For your case study data set you should have more detail, including a description of the source of the data and the nature of the problem the data describes. Look for references for similar types of problem.

3. **Classifier Description**: Include a description of your implementation of nearest neighbour classifiers, including details of design choices you made and data structures you employed. You should include an example of how to use the classifier with the refinements you have included. Also provide an overview of the other classifiers you use in the case study, including references.

4. **Results**: A description of the experimental procedure you used and details of the results. Remember, graphs are good. There should be a subsection for each hypothesis. The case study section should go into greater depth and remember, accuracy is not the only criteria.

5. **Conclusions**: how do you answer the questions posed? Are there any biases in your experiment and if so, how would you improve/refine them?

Presenting the output of experiments is a key transferable skill in any technical area. I stress again we mark the paper for this section, not the experimental code. Avoid writing a description of what you did. We do not need to know about any blind alleys you went down or problems you encountered, unless they are relevant to the experiments (e.g. memory constraints). Please also take care to include references where appropriate, and to check the formatting of references.

Relationship to formative work

Lectures on linear perceptron and classifier evaluation are highly relevant. Lab exercises in implementing and evaluating classifiers will help.

Deliverables

You should create a PDF for submission using the overleaf template
ttps://www.overleaf.com/read/ycgpvxbgwmkf

to include your solutions to part 1, 2 and 3. If you do Part 1 by hand, simply take a picture of it and print to pdf. If you do this, please make sure the file is small. There is a size limit on PDF submission.
You should submit a zipped Intellij/Netbeans project with your code **and** your PDF. The blackboard submission portal will go live one week before the deadline.
Plagiarism and collusion: Stackoverflow and other similar forums (such as the module discussion board) are valuable resources for learning and completing formative work. However, for assessed work such as this assignment, **you may not post questions relating to coursework solutions in such forums**. Using code from other sources/written by others without acknowledgement is plagiarism, which is not allowed (General Regulation 18).

Resources

Marking Scheme

1. **Part 1. Perceptron by hand** (10 marks)
2. **Part 2. Implement Perceptrons** (50 marks)
3. **Part 3. Evaluate Perceptrons** (40 marks)

Total: 100 Marks, 40% of the overall module marks