CMPT459 Spring 2020
Data Mining
Martin Ester
TAs: Ruijia Mao and Ruchita Rozario

Solution of Assignment 2

Solution posted and discussed in class: February 13, 2020

Assignment 2.1

In this assignment, we consider binary classification (two classes, „+“ and „-“) and compare the ability of different classifiers to deal with certain types of training datasets. In each of the following cases, draw a two-dimensional (two numerical features) dataset of four training examples that has the required properties.

a) A dataset on which a decision tree achieves 100% training accuracy and a linear SVM achieves 75% training accuracy:

b) A dataset on which a linear SVM achieves 100% training accuracy and a 1-nearest neighbor classifier using Euclidean distance achieves 0% training accuracy:

Assignment 2.2

Labeled data is typically much harder to obtain than unlabelled data, because labelling requires a domain expert and is therefore expensive and time-consuming. For example, you may have a large dataset of patients with their demographic and clinical attributes, but only a small dataset of patients with all of these attributes and a diagnosis, i.e. a class label. Consider a situation where you want to learn a classifier and have a small labeled dataset L (with features and class labels) and a large unlabeled dataset U (only features). The goal is to predict the class labels of all examples in U. How can you employ the datasets L and U to train a classifier that is more accurate than a classifier trained using only dataset L?

Hint: learn a sequence of classifiers with increasingly better accuracy.

a) Explain your idea in plain English.

Apply the classifier to U. Label the examples from U whose class label can be confidently predicted and add them to L. Retrain the classifier on the extended L. Iterate until there are no more examples in U whose class labels can be predicted confidently or until the classification accuracy on some held-out validation dataset does no longer improve.

b) Present the pseudo-code of your solution, i.e. a function **Classification (labeled dataset L, unlabeled dataset U)**, which assigns class labels to all examples in U.
Your pseudo-code can use the following functions (which you do not have to implement): **TrainClassifier (training dataset T),** which returns a classifier (trained on labelled dataset T), i.e. a function **Classifier (test case t),** which returns the class label of the test case and the confidence of the prediction.

```
Classification (labeled dataset L, unlabeled dataset U)
Training := Sample 80% from L;
Validation := L – Training;
Unlabeled := U;
BestClassifier := TrainClassifier(Training);
BestAccuracy := accuracy of BestClassifier on Validation;
AddedExample := TRUE;
While AddedExample
        AddedExample := FALSE;
        For all examples u in Unlabeled do
                (label,conf):=BestClassifier(u);
                if conf >= 95% then
                        add u with its label to Training;
                        Unlabeled := Unlabeled – {u};
                        AddedExample := TRUE;
        Classifier := TrainClassifier(Training);
        Accuracy := accuracy of Classifier on Validation;
        If Accuracy > BestAccuracy then
                BestClassifier := Classifier;
                BestAccuracy := Accuracy;
        Else
                For each example u in U do
                        (label,conf):= BestClassifier(u);
                        add label to u;
```

Assignment 2.3

In many training datasets, the class distribution is skewed, i.e. one class is very frequent (e.g. 95%), while another is rare (e.g. 5%). In such applications, the Naive classifier that assigns all test cases to the majority class achieves very high accuracy. However, the Naive classifier does not predict any cases of the minority class, although it is often more important than the majority class.

Propose a method that will improve the accuracy on the minority class for any given classification algorithm.

Solution 1
Undersample the majority class or oversample the minority class, such that the numbers of training examples from both classes will be (roughly) equal. This solution does not require any modification of the classification algorithm.

Solution 2
Assign higher weights to the training examples from the minority class, such that the weighted number of minority class examples is equal to the number of majority class examples. This solution requires the following modification of the classification algorithm: The objective function needs to be modified to consider the weights of the training examples. For a decision tree classifier, e.g., the computation of the gini-index needs to multiply the term corresponding to a training example by its weight.