Lab 2
Experiments in Machine Learning

Overview: This week we continue our experimentation with learning algorithms at a high level. That is, we are fine for now with using them while understanding *what* they do, though we haven't yet learned *how* they do it. The algorithm in question for today's lab is called *nearest neighbor*, implemented by the *IBk* classifier in Weka. We will learn what this algorithm does, and discover by experimentation what impact is had by three different modifications to the algorithm.

1. Instance-Based Learning

Between lecture and lab we have seen several different types of *structural descriptions*, or *models* that result from data mining. These include sets of *rules* and *decision trees*, which we saw first-hand in lab last week, and *regression formulas*, which we've seen examples of in lecture. Instance-based learning is different in that the dataset used for training is itself the structural description. The algorithm that is employed is called *nearest neighbor*, and is straightforward to explain.

Given a new instance to classify, the nearest neighbor algorithm simply measures the instance against all the training data, using some *distance function* – that is, some way of measuring how similar the new instance is to the training instances. When the "closest" neighbor (training instance) is found, the classification for that instance is used as the prediction for the new instance. Simple!

The performance of the algorithm can be improved by using more than one nearest neighbor, particularly when you have a large dataset. For example, given the new instance to classify, we might find the *5* nearest neighbors, then predict the classification of the new instance by the majority classification of the neighbors.

The beauty of data science is that no one algorithm, or no one algorithm setting, is always right. Weka allows us to experiment with different nearest neighbor clusters to find the best model for the problem we happen to be working on.

2. Experiments

Nearest-neighbor instance-based learning is both intuitive and effective. But its performance is affected by disproportionate influence of irrelevant attributes, and by high sensitivity to *noise* – training examples with erroneous or missing data. (Can you reason out why these might be particular problems? See if you can by the end of the lab.)

Today we'll run some experiments to determine how much these factors influence the predictive ability of the nearest-neighbor algorithm. Additionally, we'll experiment with training datasets of increasing size, to see what effect the *learning curve* has.

Next, turn to the next page and get started. The tutorial exercises are clearly labeled in bold. Most of them require some sort of answer or observation based on your experiment. Enter your answers on the lab report on the last page, and hand in the sheet when you're done with lab.

For this lab, a real-world forensic glass classification dataset is used. We begin by taking a preliminary look at the dataset. Then we examine the effect of selecting different attributes for nearest-neighbor classification. Next we study class noise and its impact on predictive performance for the nearest-neighbor method. Following that we vary the training set size, both for nearest-neighbor classification and for decision tree learning. Finally, you are asked to interactively construct a decision tree for an image segmentation dataset.

Before continuing you should review in your mind some aspects of the classification task:

- How is the accuracy of a classifier measured?
- To make a good classifier, are all the attributes necessary?
- What is class noise, and how would you measure its effect on learning?
- What is a learning curve in the context of machine learning?

3. The Glass Dataset

The dataset *glass.arff* from the U.S. Forensic Science Service contains data on six types of glass. Glass is described by its refractive index and the chemical elements that it contains. The aim is to classify different types of glass based on these features. This dataset is taken from a repository at the University of California Irvine, which is freely available on the Web (*https://archive.ics.uci.edu/ml/datasets.html*). We will be exploring this repository later. This particular dataset is often used as a benchmark for comparing data mining algorithms, which is why we're using it today.

Find the dataset *glass.arff* and load it into the Explorer interface, For your own information, complete the following exercises, which review material covered in Lab #1.

Exercise 1 How many attributes are there in the dataset? What are their names? What is the class attribute? Run the classification algorithm *IBk* (*weka.classifiers.lazy.IBk*). Use cross-validation to test its performance, leaving the number of folds at the default value of 10. Recall that you can examine the classifier options in the Generic Object Editor window that pops up when you click the text beside the *Choose* button. The default value of the KNN field is 1. This sets the number of neighboring instances to use when classifying. Be sure you understand what that sentence means before continuing. Ask a question if you're not sure.

Exercise 2 What is the accuracy of *IBk* (given in the Classifier Output box)? Run *IBk* again, but increase the number of neighboring instances to $k = 5$ by entering this value in the KNN field. Here and throughout this lab, continue to use cross-validation as the evaluation method. Remember, lab activities aren't just about pushing buttons. Make sure you understand what it means to increase the KNN field from 1 to 5.

Exercise 3 What is the accuracy of *IBk* with five neighboring instances (k=5)?

4. Attribute Selection

Now we investigate which subset of attributes produces the best cross-validated classification accuracy for the *IBk* algorithm on the glass dataset. This is a data cleaning process called *data reduction*. Weka contains automated attribute selection facilities, which are examined in a later lab, but it is instructive to do this manually.

Performing an exhaustive search over <u>all</u> possible subsets of the attributes is infeasible (Why? How many subsets are there?), so we apply a *backward elimination procedure*. To do this, first experiment with dropping each attribute individually from the full dataset, and run a cross-validation for each reduced version of the dataset. Once you have determined the best eight-attribute dataset, repeat the procedure with this reduced dataset to find the best seven-attribute dataset, and so on.

Exercise 4 Set the KNN value *k* back to 1. Record in the table in the lab report the best attribute set and the greatest accuracy obtained in each iteration of the backward elimination procedure. Note that the best accuracy obtained in this process is quite a bit higher than the accuracy obtained on the full dataset. (Yes, this activity is tedious and time-consuming. You are doing it correctly! It will serve as an example in class several times in the future and is worth the time.)

Exercise 5 Is this best accuracy an unbiased estimate of accuracy on future data? Be sure to explain your answer. (*Hint*: To obtain an unbiased estimate of accuracy on future data, we must not look at the test data <u>at all</u> when producing the classification model for which the estimate is being obtained. *Another hint*: I'm basically re-asking a question about training data and test data from Lab #1, just a little bit more nuanced.)

5. Class Noise and Nearest-Neighbor Learning

Nearest-neighbor learning, like other techniques, is sensitive to noise in the training data. In this section we inject varying amounts of class noise into the data and observe the effect on classification performance.

You can flip a certain percentage of class attribute values in the data to a randomly chosen other value using an unsupervised attribute filter called *AddNoise*, in *weka.filters.unsupervised.attribute*. However, for this experiment it is important that the test data remains unaffected by class noise. <u>Filtering the training data without filtering the test data is a common requirement</u>, and is achieved using a *metalearner* called *FilteredClassifier*, in *weka.classifers.meta*. This metalearner should be configured to use *IBk* as the classifier and *AddNoise* as the filter. *FilteredClassifier* applies the filter to the data before running the learning algorithm. This is done in two batches: first the training data and then the test data. The *AddNoise* filter only adds noise to the first batch of data it encounters, <u>which means that the test data passes through unchanged</u>.

Exercise 6. Record in the table in the lab report the cross-validated accuracy estimate of *IBk* for 10 different percentages of class noise and neighborhood sizes *k*= 1, *k* = 3, *k* = 5 (determined by the value of *k* in the *k*-nearest-neighbor classifier).
Exercise 7. What is the effect of increasing the amount of noise in the class attribute? (Remember that the class attribute is the "*last*" attribute in the dataset.)
Exercise 8. What is the effect of altering the value of *k*?

6. Varying the Amount of Training Data

Next we examine learning curves, which show the effect of gradually increasing the amount of training data. Again, we use the glass data, but this time with both *IBk* and the *C4.5* decision tree learners (implemented in Weka as *J48*, remember).

To obtain learning curves, use *FilteredClassifier* again, this time in conjunction with *weka.filters.unsupervised.instance.Resample*, which extracts a certain specified percentage of a given dataset and returns the reduced dataset. Again, this is done only for the first batch to which the filter is applied, so the test data passes unmodified through the *FilteredClassifier* before it reaches the classifier. Make sure you understand this, as it very well may prove useful later in your term project experiments.

Exercise 9. Record in the table in the lab report the data for learning curves for both the one-nearest-neighbor classifier (i.e. *IBk* with $k = 1$) and *J48*.
Exercise 10 What is the effect of increasing the amount of training data?
Exercise 11 Is this effect more pronounced for *IBk* or *J48*?

Lab 2: Experiments in Machine Learning

Name: ___________________________________

Exercise 1:

Exercise 2:

Exercise 3: What explanation can you think of for this result?

Exercise 4: ***Be sure you have set the value of K back to 1.***

Subset Size (Num. of input attributes)	**Input attributes in "Best" Subset**	**Classification Accuracy**
9 (all but the class att.)	Ri, Na, Mg, Al, Si, K, Ca, Ba, Fe	70.5607%
8		
7		
6		
5		
4		
3		
2		
1		
0 (only the class att. left)	None	35.514%

In conclusion, which subset of attributes would this experiment suggest gives the best accuracy?

We've used a *backward elimination procedure* here. Why is performing an exhaustive search over all possible subsets of the attributes infeasible?

Exercise 5:

Exercise 6: (Remember that the class attribute is the "***last***" attribute in this dataset.)

Percentage Noise	**k = 1**	**k = 3**	**k = 5**
0%	70.5607	71.9626	67.757
10%			
20%			
30%			
40%			
50%			

Exercise 7: Consider how the results change moving down each column.

Exercise 8: Consider how the results change moving across each row, *and* moving down each column.

Exercise 9:

Percentage of Training Set	***IBk***	***J48***
10%	52.8037%	45.3271
20%		
30%		
40%		
50%		
60%		
70%		
80%		

Exercise 10:

Exercise 11: Consider the range of lowest to highest accuracy for each algorithm.