1 Introduction

DISTANCE MEASURE AND VOTING METHOD FOR EVERYTHING AND DATASET IN QUESTION. (Use all metrics for all things).

In this assignment we were asked to classify instances in the Abalone Dataset into one of two ('young' or 'old') or three ('very young', 'old' and 'middle-age') categories. The classification method meant to be used was K-Nearest Neighbours.

2 Similarity Metrics

2.1 Euclidean Distance, Manhattan Distance and Minkowski Distance

The Abalone data set contains eight attributes out of which 7 are of a continuous nature, so it made sense to us to use some sort of distance metric to evaluate similarity instead of checking for equality (like in the case for Jaccard and Dice Similarity) or doing something like cosine similarity (which would compare the angles between the abalones since two abalones can be very similar in proportions, and thus have small angle and high cosine similarity but one may be much larger and older than the other). If p and q are the instances to be compared the formula for our similarity metrics are:

Euclidean Distance:

$$\sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{1}$$

Manhattan Distance:

$$\sum_{i=1}^{n}|p_i - q_i| \tag{2}$$

Minkowski Distance:

$$(\sum_{i=1}^{n}|p_i - q_i|^{pow})^{1/pow} \tag{3}$$

(We take pow = 0.5 for Minkowski distance, as it lets us makes for a nice comparison with Euclidean since Euclidean is basically Minkowski with p = 2)

Euclidean takes sum of squares before finding the root, Manhattan is plain addition of distances values and Minkowski (with p = 0.5) takes the root of each value before summing them up and squaring the sum. We especially wanted to contrast Minkowski and Euclidean since Minkowski is actually going to amplify larger distances (since most distances are ¡ 1 and root of a number with value ¡ 1 will be large than the number itself) and Euclidean would tend to supress it by squaring (square of number ¡ 1 is less than the number itself).

2.2 Experimentation with similarity metrics

Here are some results we got from testing our similarity metrics while varying other parameters in the program. The experiment shows us that that the performance of all of the similarity metrics is very closely lumped together, this is probably due to the fact that the distances are so small that it doesn't make much of a difference if we square or find the root of the values.

(Note: All measurements in the tables are averages of 10 runs and 10 fold cross validation was used for each run)

Similarity Experiment					
Classification Type	Parameters	Similarity Metric	Accuracy	Precision	Recall
Abalone-3	k=29,voting=inv. distance	euclidean	0.7614	0.6420	0.6387
Abalone-3	k=29,voting=inv. distance	minkowski	0.7606	0.6479	0.6445
Abalone-3	k=29,voting=inv. distance	manhattan	0.7592	0.6428	0.6397
Abalone-2	k=29,voting=inv. distance	euclidean	0.7847	0.7710	0.7389
Abalone-2	k=29,voting=inv. distance	minkowski	0.7826	0.7668	0.73496
Abalone-2	k=29,voting=inv. distance	manhattan	0.7830	0.7701	0.7372

3 Validation Framework

3.1 M-Fold Cross Validation

For our validation framework we used 10-Fold Cross Validation, we first divide the dataset into 10 (approximately) equal partitions and then perform K-Nearest Neighbour classification by choosing each of the 10 partitions as our test set (and the amalgamation of the remaining 9 as our training set) for one run of the K-Nearest Neighbour algorithm. This leads to very stable evaluation metrics across different runs of the program since performing the algorithm 10 times means that any positive outlier is averaged out by negative ones.

We also tested holdout, 5-Fold Cross Validation and 20-Fold Cross Validation and here is a graph of their accuracies for different runs:

4 Our choices

4.1 Representation of Data

We are using a 2-tuple to represent the data set wherein the first element of the tuple is the list of instances and the second element of the data set is the list of class labels corresponding to the instances.

4.2 Some other remarks