Lab 1 Report - f85709jw

Jonathan Walters

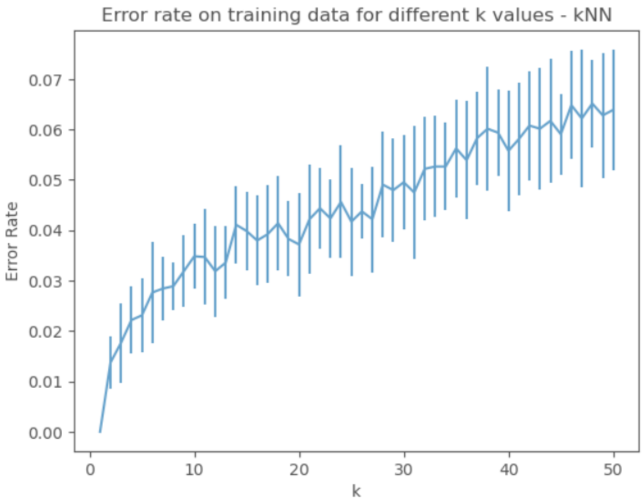
Experiment 1

The better distance measure from the experiments performed is the cosine distance. The testing accuracy measured using cosine distance was about 10% better than Euclidean distance at 96.3% and 87.5% respectively. The standard deviation was also smaller with the cosine distance compared to the Euclidean distance.

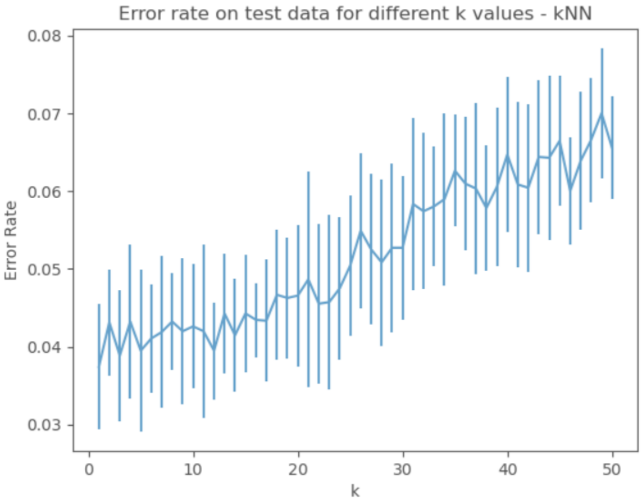
The biggest difference between Euclidean distance and cosine distance is that Euclidean accounts for the magnitude of the distance between articles, whereas cosine accounts for the angle between the articles. This is important because we might have 2 similar articles in terms of the sorts of words that are used but one article is 100 times larger in size, those articles might be quite far apart in terms of Euclidean distance but quite close in terms of cosine distance.

Experiment 2

The error in the model where we test the data on the same data used for training has a logarithmic shape to it over values of k, where at k = 1 we have 0 error. This is because the training data is simply mapped 1 to 1 to the same data it was trained on. As k increases, data points may not evaluate to the same class as it was trained on as other near data points may change the class assigned, increasing the error and bias. Because of this reason the variance tends to increase as k increases as well.



For the model that was tested using separate data from the training data we have a more linear looking shape when evaluating the error over values of k. overall the error/bias tends to be higher, and the variance is also a bit larger. One can also note that the variance stays quite consistent throughout.



This is because in the former model, each data point has at least 1 of it's closest neighbours (itself) in the correct class. The latter model has no guarantees. Of course as k increases this fact becomes less relevant, we can see that at k = 50 the error rate between the two is very close.

Experiment 3

Appropriate Classification

From reading the 5 new articles, appropriate classifications using the 4 preexisting classes might look something like:

- sp0: interest
- sp1: interest
- sp2: trade
- sp3: interest
- sp4: interest
  The classifier fails at making an appropriate prediction, assigning seemingly random classes:
- sp0: trade
- sp1: crude
- sp2: trade
- sp3: trade
- sp4: earn

Performance of the Classifiers with Sports Label

Upon adding a sports class and running tests repeatedly to evaluate the performance of the model using a confusion matrix, it can be seen that the overall performance is quite good. For all of the classes other than sports the accuracy tends to be above 95%, however the accuracy for sports is very low at 50%. This can be explained by the limited number of sport articles available for training the model, using just 3 for training.

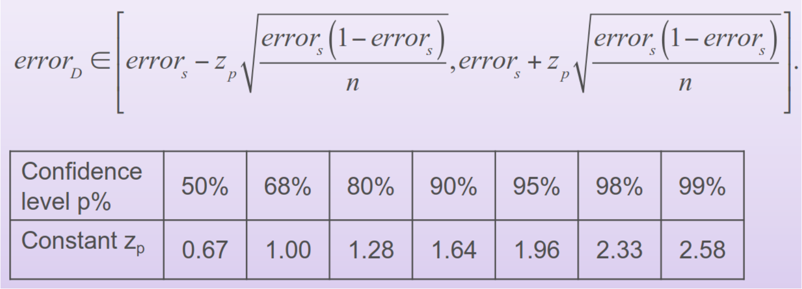
Zero-shot/Few-shot learning

The idea behind zero-shot learning is about techniques wherein a model can classify objects from an unseen class without having any training for that unseen class. Few-shot is similar but instead of having no training samples from the unseen class we now have a few.

While the model in experiment 3 is performing few-shot learning in the sense that we're needing to classify sports after only training on 3 examples, we're not incorporating any special techniques to make this effective so we're not really performing few-shot learning. We're just performing our normal learning on a very small set.

Analysis 1

To compute the interval range for where the true error lies, with a probability of 90% in experiment 2 where k = 1, using the formula



$$error_D \in \left[error_s - z_p \sqrt{\frac{error_s\left(1 - error_s\right)}{n}}, error_s + z_p \sqrt{\frac{error_s\left(1 - error_s\right)}{n}} \right].$$

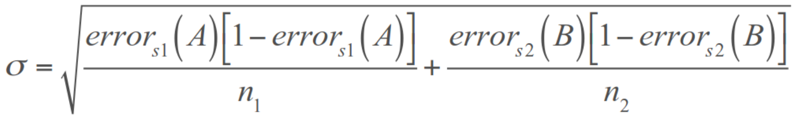
Confidence level p%	50%	68%	80%	90%	95%	98%	99%
Constant z_p	0.67	1.00	1.28	1.64	1.96	2.33	2.58

Analysis 2

To compute the probability that k=45 has a higher true error than k=1 a Z-test is preformed. A value z_p is calculated according to the formula

$$z_p = \frac{d}{\sigma}$$

where d is the absolute difference between the sample errors of the models where k=1 and k=45, and σ is calculated according to the formula:



$$\sigma = \sqrt{\frac{error_{s1}\left(A\right)\left[1 - error_{s1}\left(A\right)\right]}{n_1} + \frac{error_{s2}\left(B\right)\left[1 - error_{s2}\left(B\right)\right]}{n_2}}$$

Next a value for confidence level p is discovered by running an algorithm on z_p. Lastly the final probability is computed by:

$$C = 1 - \frac{(1 - p)}{2}$$

The result of this was that the probability that k=45 has a higher true error than k=1 is 99.5%.

Hyperparameter Selection

The splitting strategy used was a 5-fold Cross Validation strategy with 700 articles (175 from each class) used for training and validation and 100 articles kept behind for a final evaluation of the hyperparameter chosen.

This 5-fold CV was run on k values of 1 to 50, for each k value the estimated error was calculated by averaging out the errors calculated for each of the 5 folds. The lowest overall error k was selected as the best k. The final evaluation test was then run with this k using test data not used for training.

The value of k that had the lowest error was k = 1 with an estimated error of 2.86% from 5-fold CV and a final test error of 4.00% testing on data never trained on.

It's important to split the data into training, testing and validation in Machine Learning in order to ensure that we don't evaluate the model on data that it's been trained on. This is problematic because Machine Learning algorithms optimise themselves on the training data they're given so it's a bad idea to measure it's performance on what it's been optimised for. Instead it's best to train the model on certain data then to test it on different data to evaluate how well the model can generalise ideas to new data points.

In the experiment described in this section using k-fold, to pick a k value all of the data partitioned for k-fold is used in training at some point. So it's useful to have data leftover (test data) to evaluate the final choice of k.