Project 2: *k* Nearest Neighbors

Sarah Wilson SWI1S117@JHU.EDU
303-921-7225
Engineering Professionals Computer Science
Johns Hopkins University
Baltimore, MD 21218, USA

1. Introduction

Regression and classification are both common tasks in the realm of Machine Learning. Regression and classification are both supervised learning problems. Supervised learning is where the system is given an input and output and then asked to learn or predict the mapping of input to output. The algorithm explored in this paper is K-Nearest Neighbor (KNN). KNN is an example of a nonparametric algorithm, nonparametic algorithms do not make any strong assumptions about the form of the mapping function from input data to output predictions. The advantage offered by this approach is that not a lot of prior knowledge on the data or its features is required to build a predictor. KNN is nonparametric as it makes predictions for new data based upon training data by looking at the k closest neighbors to the new data. The primary philosophy behind the kNN algorithm is "In nonparametric estimation, all we assume is that similar inputs have similar outputs. This is a reasonable assumption: the world is smooth, and functions, whether they are densities, discriminants, or regression functions change slowly. Similar instances mean similar things. We all love our neighbors because they are so much like us."(1. Alpaydin)

The problem statement presented in this paper is to understand and implement a kNN classifier and regressor on 6 different and unique data sets. The experimentation will first be tuned using a validation subset of the overall data set under experimentation, to determine the most optimal value of k to use. Then the optimal value of k will be run through the full k-fold cross validation process. The experimentation will examine: kNN, edited kNN and condensed kNN The results presented will be the classification error and the regression mean squared error on each of the 6 unique data sets and across the 3 variations of the kNN algorithm.

The hypothesis of this report is that for the Classification tasks, normal KNN will do better at classifying an instance when compared with Edited or Condensed KNN. For Regression tasks it is hypothesized, that Edited or Condensed KNN will be better at regressing a new instance than normal KNN and that results from Edited and Condensed will be reality similar. This hypothesis will be tested by looking at 6 unique data sets, 3 of which that are classification task, 3 of which that are regression tasks. These data sets will be optimally tuned for the value of K nearest neighbors to use, the σ value to use in the Gaussian Kernel for regression, and the ϵ to use when running Condensed or Edited KNN. The results of these experiments will be discussed and presented against the outlined hypothesis.

Section 1 has provided the introduction, problem statement and hypothesis in regards to the kNN algorithm. Section 2 will provide an in-depth explanation of the kNN algorithm, how the algorithm will be tuned and specifics on each of the 6 data sets used. Section 3 will present the results obtained by variations of kNN, edited kNN and condensed kNN and the values that were chosen as part of the tuning process. Section 4 will discuss the results that were obtained and compare them to the hypothesis that was outlined in the introduction. This report will conclude in Section 5 with a discussion of lessons learned and areas of possible future work.

2. Algorithms and Experimental Methods

The experimental method used in this report is k-Fold Cross Validation. k-Fold Cross Validation is used when the data sets that an algorithm is being experimented on is small, the goal of k-Fold Cross validation is to maximize the amount of data that is used for training of the algorithm. The experiment will use 5-Fold Cross Validation ($k = 5$).
For the experiment in this report a Validation / Tuning set is first used to determine the optimal value of k neighbors in the kNN, σ in the case of regression and ϵ in the case of Condensed or Edited KNN.

General *k*NN
For General KNN the Entire Data set is loaded, and randomly shuffled.
The validation and tune process is started by removing 20% of the observations from Entire data set, and assigning them to new Validation / Tune data set.
A list of k nearest neighbors is assigned, and the 5 fold cross validation process will be run on each value of k to determine the optimal k. In regression the Gaussian Kernel is also applied so the bandwidth of that kernel σ is also optimized during this time. (Consider k and σ to running in a nested for loop during the tuning process.) For classification there is no σ, so only k needs to be tuned.
The Validation / Tune Data set is broken into 5 folds. 1 of those folds is assigned as the Test fold, while the other folds are concatenated together to form the Train folds. The fold that is assigned as the test fold rotates through all 5 fold positions.
For each observation in the Test Set, the distance is calculated to each observation in the Train Set using the euclidean Distance Equation. $d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$
Where p is the observation in the test set or the Query Point, and q is the observation in the train set. These values are also vectors, so the full form of this equation is scaled to the number of attributes that are in each data set. The predictor attribute in each of the data sets is dropped from this distance calculation, as we only want to use attributes that determine the predictor, not the predictor itself.
Sort the distances in an overall list for the Query Point sorted from smallest to largest.
Take the K defined smallest values from that list
If running Classification return the most commonly occurring predictor value based on those K nearest neighbors.
If running regression, take these K nearest distance values and apply the Gaussian Kernel based on the σ that we are running. The Gaussian Kernel is described by the equation:

$\exp \frac{-1}{2\sigma} * d(p,q)$
Each of the K nearest distances values has this Kernel applied. This Kernel is then used to determine the weighted average to come up with Regression prediction value. What specifically is meant by that is that each closest predictor value is then multiplied by its corresponding Gaussian Kernel value and then each one is summed, and divided by the sum of the Gaussian Kernel distances. This can be expressed as:
$WeightedPrediction = \frac{(Predictor1 * GK(dist1) + Predictor2 * GK(dist2)) ... PredictorK * GK(distK)}{GK(dist1) + GK(dist2) .... GK(distK)}$

Where K represents up to the Kth nearest neighbor. This weighted average provides a predicted value, this predicted value is then compared to the actual predicted value as determined by the Test Set or Query point. The error reported between the predicted and actual value is the Root Mean Squared Error. Root Mean Squared Error was used, since it provides how far off a measurement was in the units of the measurement that was taken. The higher the Root Mean Squared Error, the worse the prediction was to the actual value. Low Root Mean Squared Error is the objective.
Once every observation in the Test Set is Compared to the Train Set, the Fold is completed. This process then repeats for all 5 folds. The Classification Error and Regression (RMSE) are then used to determine what the optimal values of K and σ are to run the full experiment.

The full experiment for Normal KNN is completed using the optimized K and σ, and on the other 80% of the data that was not used as part of tuning. The process is exactly the same as what was outlined for tuning, with instead an optimized K and σ. And again the 5 Fold Cross Validation is performed.

Edited *k*NN

Edited KNN follows the same steps that were outlined above in regards to the 5 fold cross validation, classification and application of the Gaussian Kernel for regression. The major difference is in the generation of the Training set. Instead of just taking the Training Set as it is, first we edit out the outliers from the Training Set based on their distance from the K nearest data points. When the Training Set is created, we loop over every observation INSIDE the training set, not the test set, essentially the Query Point has changed. The process then still remains the same, the distance to all other observations in the Train Set is calculated with respect to the Query Inside the Test set.
If the task is Classification, we get the closest K neighbors, determine the majority predictor value. If the Train Set Query point does not mach the majority predictor, then it is dropped completely from the data set. This dropping is the editing. If the task is Regression, we get the closest K neighbors, determine the predictor based on the application of the Gaussian Kernel / Weighted Average. We define an ϵ, this ϵ is then used in the equation $|CurrentQueryPredictor - WeightedAveragePrediction| \leq \epsilon$.
If the delta between the Current observation in the Train Set and the Weighted Average provided by the K nearest neighbors, is less than ϵ, this data point is dropped from the Train Set.

Once the values have been dropped from the Train Set, then the Normal KNN Classification / Regression process takes place. (Here the Query Point is derived from the Test

Set, and we run against our Edited Train Set and all other Error parameters are reported exactly as described in the Normal KNN description.)

ϵ is tuned for using 20% of the entire data set that was set aside for validation. Once an optimal ϵ is determined then the full experiment is run using the other 80% of the data.

Condensed *k*NN

Edited KNN follows the same steps that were outlined above in regards to the 5 fold cross validation, classification and application of the Gaussian Kernel for regression. The major difference is in the generation of the Training set, much liked Edited KNN. In the case of Condensed nearest neighbors, the Training Set is built. Each observation point in the Training Set is examined, if running classification we get the closest K neighbors to the Query from the Training Set, determine the majority predictor value. If the Train Set Query point does mach the majority predictor, then it is added to the Condensed Training Set from the data set. The addition of these observations is what makes this condensed KNN. If the task is Regression, we get the closest K neighbors, determine the predictor based on the application of the Gaussian Kernel / Weighted Average. We define an ϵ, this ϵ is then used in the equation $|CurrentQueryPredictor - WeightedAveragePrediction| \leq \epsilon$. If the delta between the Current observation in the Train Set and the Weighted Average provided by the K nearest neighbors, is less than epsilon this data point is added to the newly formed Condensed Train Set.

Once the values have been added from the Train Set, then the Normal KNN Classification / Regression process takes place. (Here the Query Point is derived from the Test Set, and we run against our Edited Train Set and all other Error parameters are reported exactly as described in the Normal KNN description.)

ϵ is tuned for using 20% of the entire data set that was set aside for validation. Once an optimal ϵ is determined then the full experiment is run using the other 80% of the data.

Data Sets

The following data sets were used during the classification and regression tasks for this project.

Breast Cancer

Description:
Task: Classification
Predictor: Diagnosis (Malignant or Benign)
Link:
https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29

Car Evaluation

Description:
Task: Classification
Predictor: Car Evaluation (Unacceptable, Acceptable, Good, Very Good)
Link:
https://archive.ics.uci.edu/ml/datasets/Car+Evaluation

Congressional Vote

Description: 1984 United Stated Congressional Voting Records
Task: Classification
Predictor: Party (Republican / Democrat)
Link:
https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records

Albalone

Description: Physical measurements of Albalone
Task: Regression
Predictor: Rings (int)
Link:
https://archive.ics.uci.edu/ml/datasets/Abalone

Computer Hardware

Description: Relative CPU performance data.
Task: Regression
Predictor: PRP
Link:
https://archive.ics.uci.edu/ml/datasets/Computer+Hardware

Forest Fires

Description: Forest Fire burn area data
Task: Regression
Predictor: Area (float)
Link:
https://archive.ics.uci.edu/ml/datasets/Forest+Fires

3. Results

Tables 1-9 display the results from Classification Tasks: the Breast Cancer, Car Evaluation and Congressional Vote data sets while running Normal KNN, Edited KNN and Condensed KNN. These tables show the results from the train set and the test set during each fold of the k-fold validation process. The results of the tuning process were omitted from this report for brevity but can be found in the project submission, under Results Output. Each KNN variation was run using the optimal values as determined by the parameter tuning. Discussion of the tuning process will occur in the discussion section. The error used is classification error and can be described as the Number of Times Prediction was Wrong / Total Number of Comparisons. This error was averaged across the 5 folds to provide the Average Classification Error against each data set.
Tables 10-18 display the results from the Regression Tasks: the Albalone, Computer Hardware and Forest Fire data sets. These tables show the result from the train set and the test set during each fold of the k-fold validation process.The results of the tuning process were omitted from this report for brevity but can be found in the project submission, under Results Output. Each KNN variation was run using the optimal values as determined by the parameter tuning. Discussion of the tuning process will occur in the discussion section. The error used is Root Mean Square Error. Root Mean Squared Error was used, since it provides how far off a measurement was in the units of the measurement that was taken. The higher the Root Mean Squared Error, the worse the prediction was to the actual value. Low Root Mean Squared Error is the objective.

Table 1: Car Evaluation: KNN - Experimental Results

Car Eval: Normal KNN	
	kVal = 3
k-fold[1]	0.1047
k-fold[2]	0.0939
k-fold[3]	0.1123
k-fold[4]	0.1775
k-fold[5]	0.1848
Average Classification Error Across Folds	0.1346

Car Evaluation: Edited KNN	
	kVal = 3
k-fold[1]	0.2310
k-fold[2]	0.2779
k-fold[3]	0.4311
k-fold[4]	0.4963
k-fold[5]	0.5543
Average Classification Error Across Folds	0.3981

Car Evaluation: Condensed KNN	
	kVal = 3
k-fold[1]	0.2310
k-fold[2]	0.2779
k-fold[3]	0.4311
k-fold[4]	0.4963
k-fold[5]	0.5543
Average Classification Error Across Folds	0.3981

Table 2: Breast Cancer: KNN - Experimental Results

Breast Cancer: Normal KNN	
	kVal = 7
k-fold[1]	0.4196
k-fold[2]	0.4643
k-fold[3]	0.4286
k-fold[4]	0.3304
k-fold[5]	0.2703
Average Classification Error Across Folds	0.3826

Breast Cancer: Edited KNN	
	kVal = 7
k-fold[1]	0.4285
k-fold[2]	0.4821
k-fold[3]	0.4107
k-fold[4]	0.5892
k-fold[5]	0.3693
Average Classification Error Across Folds	0.4560

Breast Cancer: Condensed KNN	
	kVal = 7
k-fold[1]	0.4285
k-fold[2]	0.4821
k-fold[3]	0.4107
k-fold[4]	0.5892
k-fold[5]	0.3693
Average Classification Error Across Folds	0.4560

Table 3: Congressional Vote: KNN - Experimental Results

Congressional Vote: Normal KNN	
	kVal = 3
k-fold[1]	0.0429
k-fold[2]	0.0571
k-fold[3]	0.0571
k-fold[4]	0.0290
k-fold[5]	0.1449
Average Classification Error Across Folds	0.0662

Congressional Vote: Edited KNN	
	kVal = 3
k-fold[1]	0.4428
k-fold[2]	0.4142
k-fold[3]	0.5000
k-fold[4]	0.4782
k-fold[5]	0.5072
Average Classification Error Across Folds	0.4685

Congressional Vote: Condensed KNN	
	kVal = 3
k-fold[1]	0.4428
k-fold[2]	0.4142
k-fold[3]	0.5000
k-fold[4]	0.4782
k-fold[5]	0.5072
Average Classification Error Across Folds	0.4685

Table 4: Albalone: KNN - Experimental Results

Alablone: Normal KNN	
	kVal = 7 Sigma = 0.1
k-fold[1]	3.31
k-fold[2]	1.59
k-fold[3]	2.43
k-fold[4]	1.99
k-fold[5]	1.82
Average RMSE Across Folds	2.23

Alablone: Edited KNN	
	kVal = 7 Sigma = 0.1 Epsilon = 100
k-fold[1]	3.31
k-fold[2]	1.59
k-fold[3]	2.44
k-fold[4]	1.99
k-fold[5]	1.82
Average RMSE Across Folds	2.23

Alablone: Condensed KNN	
	kVal = 7 Sigma = 0.1 Epsilon = 100
k-fold[1]	3.31
k-fold[2]	1.59
k-fold[3]	2.44
k-fold[4]	1.99
k-fold[5]	1.82
Average RMSE Across Folds	2.23

Table 5: Computer Hardware: KNN - Experimental Results

Computer Hardware: Normal KNN	
	kVal = 7 Sigma = 1
k-fold[1]	5.78
k-fold[2]	10.35
k-fold[3]	10.19
k-fold[4]	10.11
k-fold[5]	9.78
Average RMSE Across Folds	9.25

Computer Hardware: Edited KNN	
	kVal = 7 Sigma = 0.1 Epsilon = 100
k-fold[1]	131.60
k-fold[2]	18.35
k-fold[3]	636.18
k-fold[4]	29.10
k-fold[5]	28.69
Average RMSE Across Folds	168.78

Computer Hardware: Condensed KNN	
	kVal = 7 Sigma = 0.1 Epsilon = 100
k-fold[1]	131.60
k-fold[2]	18.35
k-fold[3]	636.18
k-fold[4]	29.10
k-fold[5]	28.69
Average RMSE Across Folds	168.78

Table 6: Forest Fires: KNN - Experimental Results

Forest Fires: Normal KNN	
	kVal = 7 Sigma = 0.01
k-fold[1]	28.94
k-fold[2]	171.31
k-fold[3]	129.77
k-fold[4]	24.34
k-fold[5]	49.27
Average RMSE Across Folds	80.72

Forest Fires: Edited KNN	
	kVal = 7 Sigma = 0.01 Epsilon = 0.1
k-fold[1]	4.65
k-fold[2]	16.86
k-fold[3]	129.56
k-fold[4]	24.66
k-fold[5]	34.09
Average RMSE Across Folds	41.97

Forest Fires: Condensed KNN	
	kVal = 7 Sigma = 0.01 Epsilon =
k-fold[1]	4.65
k-fold[2]	16.86
k-fold[3]	129.56
k-fold[4]	24.66
k-fold[5]	34.09
Average RMSE Across Folds	41.97

4. Discussion

The hypothesis presented at the start of this report was that for the Classification tasks, normal KNN will do better at classifying an instance when compared with Edited or Condensed KNN. For Regression tasks it is hypothesized, that Edited or Condensed KNN will be better at regressing a new instance than normal KNN and that results from Edited and Condensed will be reality similar.

Looking at Classification results from the Congressional Vote, Breast Cancer and Car Evaluation data sets, it can be seen that the average classification error across the 5 folds, is lower than when compared with the average classification error from the edited and condensed experiments. These results seem to support the hypothesis. It may also be possible to see that based on the implementation details of the condensed and edited KNN, for classification a query is removed from the train set based solely on if it matches the classification of it's neighbors. If one predictor was dominate in train set compared to other, then outliers would get toss quickly. This same adding / removal process is not applied to the test set. Meaning that outliers that are naturally occurring in the Test Set, will be counted against the correct prediction count, leading to higher error. It can be noted that looking across all three data sets, the Edited KNN and Condensed KNN when looking at a specific data set are exactly equal. This seems suspect. It can be suspected that this is an error in the implementation code with the shuffling. The Entire Data Sets were not being shuffled prior to insertion into the KNN algorithm. If the random shuffling was occurring it would be likely that the results obtained for Edited and Condensed KNN in a data set would be similar, but not identical.

Looking at the Regression results from the Computer Hardware, Alablone and Forest Fire data sets, in all three cases the RMSE values obtained from the Edited and Condensed are lower than the values obtained for Normal KNN. For regression, based on implementation this would seem to make sense, you're defining a range ϵ that predictions are allowed to match to in order to be added or removed as outliers or inliers. This tuning seems to be less susceptible to the fault identified for classification, since you're defined a range of values ϵ that can be let into the Train Set, there are more chances to not over edit the data. Again the Condensed and Edited results were exactly the same, this is most likely due to the shuffling error identified as par of classification. This is my best guess assumption as

to what may be happening there, a full root cause investigation would need to be had in order to determine if that is true.

5. Conclusion

The problem statement presented in this paper is to understand and implement a kNN classifier and regressor on 6 different and unique data sets. That objective was met, through the explanation provided in Section 2 on the different variations of KNN, normal KNN, edited KNN and condensed KNN. During the implementation, tuning and review of the results, a few areas of improvement could have been applied. The results obtained from the Edited and Condensed nearest neighbors was the same inside compared unique data sets, the root cause for this discrepancy is most likely caused by and error in the implementation code where the Entire Data Sets were not being shuffled prior to insertion into the KNN algorithm. If the random shuffling was occurring it would be likely that the results obtained for Edited and Condensed KNN in a data set would be similar, but not identical. During the course of this project, tuning was introduced and it appears that different values of k, σ, and ϵ do impact the accuracy of the results that are seen out of the prediction algorithms. An area of future work would be to explore the different types of distance calculations that could be applied to the K Nearest Neighbors problem, such as Hamming or Manhattan, and seeing if one is better suited to a particular type of Learning problem.

6. References

1. Alpaydin, E. (2004). Introduction to machine learning (Oip). Mit Press.