PROJECT 2
NEAREST NEIGHBORS CLASSIFICATION AND REGRESSION

DAVID ATLAS

Abstract. This paper will introduce the K Nearest-Neighbors (KNN) algorithm for supervised learning. It will also introduce two variants - Edited KNN and Condensed KNN - that provide more efficient predictions (and potentially better performance). Heuristics on when to use each of the algorithms are also discussed. In several experiments on real world classification problems, the algorithms perform comparably, with the standard KNN outperforming the variants. On regression problems, KNN does not perform particularly well compared to an OLS baseline, for the datasets investigated.

1. Problem Statement & Hypothesis

In this paper, a non-parametric density estimation technique, K-Nearest Neighbors, is adapted for supervised learning, and applied to real world classification and regression problems. Additionally, two variants are introduced that seek to make prediction more efficient by reducing the set of points included in calculated the nearest neighbors. These variants are only valid for classifications problems, and so are not used on regression problems.

Because these algorithms rely on density estimation, it is expected that they will not perform well in high dimensional spaces. Additionally, the algorithms cannot detect interactions between features, and so would not be expected to perform well if there are interactions.

The density estimation technique (explained in more detail below) treats all features equally. This means that given two features, one of which is highly indicative of the label, and one of which is not, the distance between two datapoints equally weights both features, and so a point may be close in the space of the relevant feature, and far in the space of the irrelevent feature. This datapoint might get classified incorrectly as a result. Therefore, performance may be poor if many unnecessary features are included in the model, as the important features may be drowned out by the distances of the unimportant features.

Out of the algorithms in this paper, the standard KNN should outperform in situations where the data is not too noisy, and the classes are reasonably seperated. An Edited KNN model might perform better on noisy data, as points that lie in the space of a different class are deleted from the prediction set. Additionally, if there are irrelevant features included in the algorithm, an Edited KNN may perform well, as it may drop points in a space that are far from the true class center only along irrelevant attribute dimensions.

A Condensed KNN might perform poorly on noisy data, as points far from their class center will be added back into the prediction set.

Both the Edited KNN and the Condensed KNN will be far more efficient at prediction time, as the standard KNN must calculate the distance across all points in the training set.

Holistically, it is worth noting that all of the algorithms presented have similar inductive learning biases. They do not assume any specific parametric form, but are all predicated on features being relevant, and distances being important across the features. Therefore, it would be unexpected if performance were not roughly similar across the algorithms on any given classification problem.

2. Description of Algorithms

K Nearest Neighbors. The K Nearest-Neighbors algorithm[1] relies on the general hypothesis that points that are near each other in feature space are also near each other in the target space.

The K Nearest-Neighbors algorithm is a lazy algorithm, in that it does not do anything on fitting, but does all the work at prediction time. It calculates the distance between the instance to label, x_0, and the training instances, $X = (x_1, \cdots, x_n)$. The training points with the k shortest distances are then selected.

In a classification problem, the class with the most elements of the k is chosen for x_0, and ties are broken randomly.

In a regression problem, the mean target value of the k nearest training points is used for prediction. Note that the mean could be substituted for any aggregation function, and should be chosen based on the problem

There are two choices to make regarding the algorithm. One is the choice of k, or the number of neighbors that participate in the majority vote. This value can be chosen via a holdout validation set.

The other choice is that of the distance metric to be used in finding the "nearest" neighbors. This is commonly the 2-norm (Euclidean distance), but other p-norms can be chosen. There are obviously many other distances that can be used, and this should be chosen based on the problem, and/or via a holdout validation set.

One other important note on nearest-neighbor schemes is the importance of standardizing the data before running the algorithm. Because the distance between data points is used, if one feature is of a larger scale, it can easily drown out others, and if it is of a smaller scale, it can easily be drowned out by others.

Edited KNN. The motivation behind Edited KNN is twofold. Consider that with the standard KNN, at prediction time, given n training points, n distance calculations must be made in order to make a single prediction. If the set of training points can be edited such that there are fewer points, but the predictions are roughly the same, prediction becomes more computationally efficient. Additionally, if there are many points in the training set that are not near their class centers, prediction quality may suffer, as an instance near them may be misclassified.

Edited KNNs[7] are very similar to the standard KNN. However, at fitting time, each of the training points x_i are iteratively classified using all of the other data points, X_{-i}. If the prediction $\hat{y}_i \neq y_i$, then x_i is dropped from the training set. In [7], all points are considered for removal once. Alternatively, this process can be repeated until no further changes are made, or until performance on a validation set starts to decrease.

Alternatively, the editing procedure can remove points that are correctly classified, with the hypothesis that they are likely not needed for correct classification going forward. In the experiments below, the points that are incorrectly classified will be dropped.

Condensed KNN. The motivation for the Condensed KNN[3] is similar to that of the Edited KNN. It is desirable to have fewer points in the training set, and to retain only those points that are needed to avoid an increase in error.

The condensing algorithm begins with an empty set $Z = \emptyset$. The first data point considered is added to Z. Then, for each of the remaining data points x_i, the distances between x_i and all points in Z are calculated, and the nearest data point z_* is found. If the class of x_i is not equal to the class of z_*, x_i is added to Z. This process is repeated until no changes are made.

Note that at prediction time, the class of the single nearest neighbor is used. This is because within the training set, all training points are already correctly classified by the nearest point.

3. Description of Experimental Approach

The 3 algorithms were applied to 2 real life classification problems:

(1) The Ecoli dataset for classifying the localization site of protein.
(2) The Image Segmentation dataset for classifying the part of the image of a given pixel.

Standard KNN was applied to 2 additional regression problems.

(1) The CPU Performance dataset for predicting the performance of a given CPU (regression problem).
(2) The Forest Fires dataset in which the area burned is predicted (regression problem).

Discrete inputs were one-hot encoded. A 5-Fold cross validation scheme was used to estimate the expected out-of-sample error. For classification problems, a stratefied cross-validation approach was used to accurately represent the various classes. Within the cross-validation scheme, k was tuned by choosing

$$k^* = \underset{1 \leq k \leq 5}{\arg\max}\, \bar{f(k)},$$

where $f(k)$ was accuracy (proportion of correctly labeled instances) for classification problems, and negative square root mean squared error (RMSE) for regression problems. Note that negative RMSE was used to make the selection of k consistent as a maximization problem. k was chosen by running 5-fold CV over the first training set of the overall 5-fold CV, and taking the mean of $f(k)$ across the folds, giving $\bar{f(k)}$.

Each dataset was standardized dividing by the column-wise mean and standard deviation. This ensures that the scale of each feature is the same (a distance of 1 indicates a 1 standard deviation difference), and all columns are centered at zero.

The KNN structure naturally handles multi-class classification problems, and regression problems were solved by taking the mean across the target values of the k neighbors.

4. Experimental Results

Ecoli Data. The first experimental dataset is the Ecoli data[4], a classification problem. All 3 algorithms above are applied, and the results are shown in Table 4.

Algorithm	k	Accuracy
Standard KNN	4	81.9%
Edited KNN	4	81.0%
Condensed KNN	1	75.3%

Table 1. The results of various KNN variants on the Ecoli dataset for classification.

As noted above, k was chosen using cross-validation on the training set of the first of the 5-folds. The highest accuracy across the cross-validation folds (within the first fold) was selected. k was set to one for the condensed algorithm.

Image Segmentation. The second experimental dataset is the Image Segmentation dataset[2], a classification problem. The goal is to classify pixel to the portion of the image from which they came. All 3 algorithms above are applied, and the results are shown in Table 4. Note that the "REGION-PIXEL-COUNT" class is dropped, as it does not have any variance.

Algorithm	k	Accuracy
Standard KNN	5	85.2%
Edited KNN	1	78.1%
Condensed KNN	1	82.3%

Table 2. The results of various KNN variants on the Image Segmentation dataset for classification.

As noted above, k was chosen using cross-validation on the training set of the first of the 5-folds. The highest accuracy across the cross-validation folds (within the first fold) was selected. k was set to one for the condensed algorithm.

CPU Performance. The third experimental dataset is the CPU performance dataset[6]. The goal is to predict the relative performance given a set of characteristics of a CPU. Only the standard KNN is applied to the regression problem, as the edited and condensed algorithms rely on class labels. The results are shown in Table 4.

Algorithm	k	RMSE
Standard KNN	1	63.4

Table 3. The results of a KNN on the CPU performance regression problem.

Forest Fires. The fourth experimental dataset is a regression problem in which the area of a given forest fire is predicted based on features about the time and weather of the fire[5]. In preprocessing the data, features describing the month and day are one-hot encoded. For this dataset, two experiments were run - one contained all of the inputs in the design matrix, a total of 27 features, and one contained only the 4 weather-related inputs mentioned in[5] as part of the best performing model. There are temperature, relative humidity, wind and rain. The results of the experiments are shown in Table 4. As noted in [5], this is a very difficult regression problem, and the performance of the KNN is well below that of the naive mean prediction.

Algorithm	Features	k	RMSE
Standard KNN	All features	4	91.8
Standard KNN	Direct weather features	1	80.2

TABLE 4. The results of a KNN on the Forest Fire Area regression problem.

The KNN performs better when fewer features are included, especially as those features are known to be more relevant to producing good predictions. This aligns with the hypothesis above, as KNNs are expected to struggle in high dimensional feature spaces with many potentially irrelevant features. Removing them is a good way to boost performance.

5. Algorithm Behavior and Conclusions

The performance of the algorithms on classification problems was not unexpected, as the standard KNN outperformed the edited and condensed algorithms. The hypothesis was that the edited and condensed versions might outperform if the data were noisy or if irrelevant features were included, but absent that, the larger samples should provide better estimation. All of the performances were also roughly in-line with each other, indicating that the alogrithms all present similar inductive biases, which is as expected.

On the regression problems, the KNN did not perform particularly well. For the CPU performance dataset, the linear regression model cited[6] outperformed the KNN, although the KNN outperformed the naive mean prediction model. On the Forest Fires dataset[5], the KNN did not outperform a linear regression model fit on the same set of features, nor did either of them outperform the naive mean prediction model. This was not unexpected, as the authors referenced the difficulty of the problem, and noted that in terms of RMSE, none of the battery of learning methods applied outperformed the naive mean prediction model.

It is difficult to draw generalizations about the performance of learning algorithms from only a few datasets, but one structural limitation is worth mentioning. KNNs struggle in high dimensional spaces. This was evidenced by the forest fires data, which has a feature space in $\mathbb{R}^{27}$, once the categorical variables are one-hot encoded. This is exacerbated by the inclusion of features that are irrelevant, as the algorithm has little mechanism for determine what is and what is not important. It was shown that performance improves when fewer, more relevant features are included.

Additionally, if parametric forms are appropriate for a problem, they may outperform KNN - this occurs in the forest fires regression problem, where some transformations and a linear regression model outperform the KNN. This may contain a valuable detail about the KNN - it should be applied in cases where the parametric form is unknown (e.g. not easily linearly seperable), and the flexible Voronoi diagram created by KNN is a desirable quality.

6. Summary

This paper introduced the K Nearest-Neighbors algorithm for non-parametric supervised learning, as well as two variants of the classification algorithm designed for efficient inference, and potentially improved performance, called Edited and Condensed KNN. The algorithms select training points that are close in feature space to make predictions about the inference point.

The KNN was applied to 4 problems, two classification and two regression, while the variants are only applicable to classification, and therefore were only applied to those problems.

It was found that KNN outperforms its variants on the classification problems, and performs inline with other learning algorithms on the regression problems.

References

[1] Naomi Altman. An introduction to kernel and nearest neighbor nonparametric regression. 1992.
[2] Vision Group. Image segmentation data. URL: `https://archive.ics.uci.edu/ml/machine-learning-databases/image/segmentation.data`.
[3] Baba-Ali Ahmed Riadh Miloud-Aouidate Amal. Survey of nearest neighbor condensing techniques. *International Journal of Advanced Computer Science and Application*, pages 59–64, 2001.
[4] Kenta Nakai. Protein localization sites. URL: `https://archive.ics.uci.edu/ml/machine-learning-databases/ecoli/ecoli.data`.
[5] Anibal Morais Paulo Corez. Forest fires. URL: `https://archive.ics.uci.edu/ml/machine-learning-databases/forest-fires/forestfires.csv`.

[6] Jacob Feldmesser Phillip Ein-Dor. Relative cpu performance data. URL: `https://archive.ics.uci.edu/ml/machine-learning-databases/cpu-performance/machine.data`.

[7] Dennis L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 408–421, 1972.