Implementation of K-Nearest Neighbor in Multi-Label Learning

Mark Battle battlema@udel.edu

Thomas Huber thuber@udel.edu

Tyler Rust trust@udel.edu

Abstract

Most data can be categorized with more than one label; multi-label learning is used in such situations to find the most probable label set to assign the new data point. This paper implements a lazy K-Nearest Neighbor (KNN) approach to multi-Label learning. For any unseen data point, the algorithm will gather information from the label sets of neighboring instances to determine the most probable label set. Experiments are performed on a generated synthetic dataset and demonstrates the effectiveness of KNN as a multi-label learning algorithm

1 Introduction

Multi-label learning is a common problem in the real world. Oftentimes, data can simultaneously fall under many categories, each of which needs to be accurately represented to ensure that all the important aspects are captured. For example, movie recommender systems account for different movie labels, action, comedy, romance, tearjerkers, etc. With those labels in mind, it is important to analyze and differentiate a user who likes action and comedy from somebody who likes comedy and romance; to recommend movies with a label set that more closely aligns with a user's label set.

K-Nearest Neighbors is a simple algorithm and assumes that similar things are in proximity; they belong to the same class. The distance between points is a hyperparameter as many different methods can be used, examples are Manhattan, Euclidean, and Jaccard. By taking the neighbors of an unknown data point, it is then possible to build a label set for the unknown data point under the same assumption that the data points are similar. K is a hyperparameter that the user initializes. It is important to adjust K to reach an optimal output, as it is possible to underfit and overfit the data. As the value of K is decreased toward 1, predictions become less accurate. Fewer data is available for inference, and it is entirely possible that the nearest neighbor belongs to a different class than the unknown data point. Oppositely, as K approaches the size of a dataset, the predictions become more accurate to a point, before eventually falling off and becoming less accurate. This is because "majority voting" will lead to every new data point being assigned to the majority class.

Advantages of KNN include, but are not limited to, ease of implementation, intuitive, and no time spent training (all the work happens during the prediction). Though, as the number of data points increases, KNN becomes significantly slower as it must calculate the distance between the unknown data point and every other data point every time new data is introduced. This is an O(N) time complexity, and as the number of data points increases, so too should the value for K.

The next sections are organized as follows. Problem Statement, an explanation of the problem with multi-label classification that KNN addresses. Proposed method, containing details of study design and how the experiment will be conducted. Numerical experiments, the results of our implemented method. And finally, a conclusion of evidence obtained from the experiment.

2 Problem Statement

There are a few options that exist to handle the problem of multi-label data. The two main methods that exist are problem transformation and algorithm adaptation methods. Problem transformation methods cast the multi-label problem into a set of binary classification problems and can be handled by binary classifiers. Algorithm adaptation takes existing binary classification algorithms and directly modifies them to handle multi-label classification. Such is the case in multi-label K-Nearest Neighbor, ML-KNN. By implementing ML-KNN and showing the effectiveness of such an algorithm in determining a label set for unknown data, the algorithm can then be applied to many real-world problems including recommender systems, text categorization, and sentiment analysis.

3 Proposed Method

The method implemented for Multi-Label Learning is multi-Label k-Nearest Neighbors (ML-KNN). This is an Algorithm Adaptation method where the basic idea is to adapt k-nearest neighbor techniques to deal with multi-label data. ML-KNN is a first-order approach in which the relevance of each label is weighed separately. ML-KNN has the advantage of inheriting the merits of both lazy learning and Bayesian reasoning. In terms of lazy learning, decision boundaries can be adaptively adjusted due to the varying neighbors identified for each unseen instance. As for Bayesian reasoning, the class-imbalance issue can be largely mitigated due to the prior probabilities estimated for each class label.

ML-KNN can be split up into training and testing a dataset. To train the dataset, ML-KNN finds the prior probability P(H$_j$) and P(-H$_j$). These probabilities are found using the frequency counting strategy. Counting the number of data samples associated with each label then smoothing them using the following equation (where s=1 is Laplacian Smoothing):

$$\mathbb{P}(H_j) = \frac{s + \sum_{i=1}^{m} [\![y_j \in Y_i]\!]}{s \times 2 + m}; \quad \mathbb{P}(\neg H_j) = 1 - \mathbb{P}(H_j)$$

From there it is necessary to create two frequency matrices k$_j$ and k$_j$bar. K$_j$ keeps track of the number of training samples which have label y$_j$ and have exactly r neighbors with label y$_j$ where r is from 0 to the k number of neighbors chosen. Similarly, k$_j$bar keeps track of the number of training samples which do not have label y$_j$ and have exactly r neighbors with label y$_j$.

The multi-label classifier is now trained. From here, the classifier is ready to test new points. The classifier labels new points with the following:

$$Y = \{ y_j \mid \mathbb{P}(H_j \mid C_j) / \mathbb{P}(\neg H_j \mid C_j) > 1, \ 1 \le j \le q \}$$

The classifier checks if P(H$_j$|C$_j$) is larger than P(-H$_j$|C$_j$). If it is, that is the most likely label that the new point will have. Those probabilities utilize the information obtained in the training process. Using Bayes Theorem, we know the following:

$$\frac{\mathbb{P}(H_j \mid C_j)}{\mathbb{P}(\neg H_j \mid C_j)} = \frac{\mathbb{P}(H_j) \cdot \mathbb{P}(C_j \mid H_j)}{\mathbb{P}(\neg H_j) \cdot \mathbb{P}(C_j \mid \neg H_j)}$$

The value C$_j$ counts the number of neighbors with label y$_j$. If one of the 0-K neighbors has label [0,1], then C$_j$ will be 1 for that label for that point. P(C$_j$|H$_j$) and P(C$_j$|-H$_j$) equal the following:

$$\begin{aligned} \mathbb{P}(C_j \mid H_j) &= \frac{s + \kappa_j[C_j]}{s \times (k+1) + \sum_{r=0}^{k} \kappa_j[r]} \quad (1 \le j \le q, \ 0 \le C_j \le k) \\ \mathbb{P}(C_j \mid \neg H_j) &= \frac{s + \tilde{\kappa}_j[C_j]}{s \times (k+1) + \sum_{r=0}^{k} \tilde{\kappa}_j[r]} \quad (1 \le j \le q, \ 0 \le C_j \le k) \end{aligned}$$

These represent the likelihood that new point x has exactly C$_j$ neighbors with label y$_j$ when H$_j$ holds or doesn't hold for the case of -H$_j$. This is how ML-KNN utilizes both prior probabilities (P(H$_j$), P(-H$_j$)) as well as likelihood (P(C$_j$|H$_j$), P(C$_j$|-H$_j$)) to determine labels for multi-label learning.

4 Numerical Experiments

For this experiment, a multi-label problem is solved where there exist two classes with binary labels [0] and [1].

4.1 Labeling

The data that we use for our experiment has four sets in the training and test dataset ([0, 0], [0, 1], [1, 0], [1, 1]). The multiple label classification is an array Y where Y = [class1 label, class2 label,...]. Each entry in the list Y corresponds to a probability of how likely that set of [class 1 label, and class 2 label] is to occur.

Label Sets: Red circle = [1,1], Red X = [1,0], Green Circle = [0,1], Green X = [0,0].

4.2 Testing Algorithm with a Simple Generated Dataset

The Multi Label Learning KNN Algorithm is trained using a generated synthetic dataset. The dataset is generated by randomly assigning the [x,y] position values for each data point with float values range from -5.0 to 10.0 i.e. [-5.0 : 10.0, -5.0 : 10.0]. Data points for each label set ([0,0], [0,1], [1,0], [1,1]) are placed in a specific quadrant. Each quadrant is generated with an equal number of data points. The test dataset is generated in the same way, and the ML-KNN algorithm predicted with an accuracy of 100%.

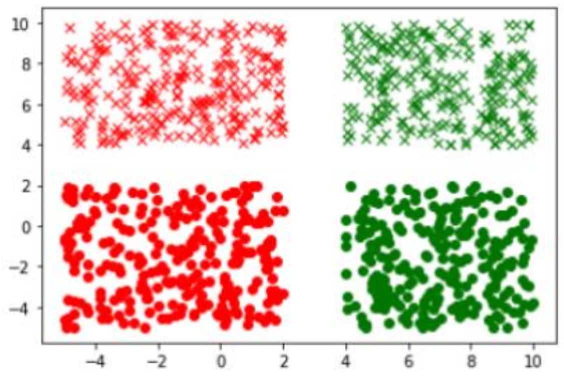

Figure 1: Visual representation of 1000 datapoints in the training set.

4.3 Testing Algorithm with a more Challenging Dataset

New, more challenging, test and training datasets are generated. Our second dataset shown in Figure 2 below is more challenging than the original dataset, as the data points are now overlapping. The ML-KNN algorithm has a more difficult decision to make, especially in the center where data points from each of the four sets are clustered together with high density. Upon training and testing using the dataset represented in Figure 2, the ML-KNN classifier Accuracy is 96%.

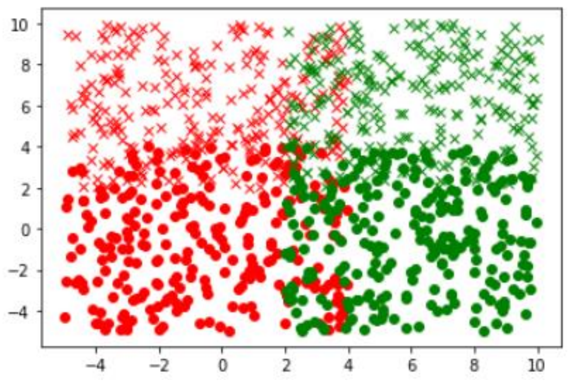

Figure 2: Visual Representation of 1000 datapoints in the new training set.

5.3 - Deep-dive into ML-KNN algorithm

During testing of ML-KNN, the hyperparameter k is set to 5. Reducing the number of k's or increasing the number of k's (k>=2) did not make a difference in the accuracy of predictions. The training dataset gives positions of the distributed points within the 2-D problem space. For this training dataset, the probability of each Output Label is seen below.

```
P([1, 1]) = 0.25149700598802394      P(-[1, 1]) = 0.7485029940119761
P([1, 0]) = 0.250499001996008        P(-[1, 0]) = 0.749500998003992
P([0, 1]) = 0.250499001996008        P(-[0, 1]) = 0.749500998003992
P([0, 0]) = 0.249500998003992        P(-[0, 0]) = 0.750499001996008
```

Data is collected for frequency arrays k_j and $k_j$bar which hold a count for the number of neighbors for each point that has y_j as their label where k_j holds the count when that point is also labeled y_j and $k_j$bar holds the count for when that point is not labeled y_j.

```
kj: [[14, 83, 151, 131, 78, 543], [14, 83, 151, 131, 78, 543], [14, 83, 151, 131, 78, 543], [14, 83, 151, 131, 78, 543], [14, 83, 151, 131, 78, 543]]
kjBar: [[2481, 131, 172, 151, 58, 7], [2481, 131, 172, 151, 58, 7], [2481, 131, 172, 151, 58, 7], [2481, 131, 172, 151, 58, 7], [2481, 131, 172, 151, 58, 7]]
```

For each point, get a count of k nearest neighbors. Then, grab a count (C_j) of how many neighbors have each label. Below is an example count of neighbor's labels for a single point.

```
count of neighbors with label [1, 1]: 2
count of neighbors with label [1, 0]: 3
count of neighbors with label [0, 1]: 0
count of neighbors with label [0, 0]: 0
```

Then ML-KNN can be used to determine which set of labels this point should have based on the probabilities P(Hj|Cj) and P(-Hj|Cj) described above. Using the same point above, the classifier can classify this label (correctly) as [1,1]. Even though there are more neighbors near it labeled [1,0] or red 'x's, the output is determined to be a red circle due to the prior probability of label [1,1] being more likely in combination with the number of points labeled [1,1] with exactly r = 2 neighbors labeled [1,1]. Figure 3 shows that the datapoint is labeled as a red circle (the data point is painted blue to make it more obvious). The accuracy of the ML-KNN classifier on this synthetic dataset is 96.7%.

```
New Datapoint Location: [0.03450240786113401, 1.980664249286587, [1, 1]]
```

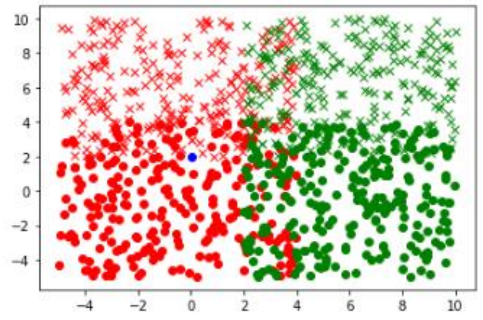

Figure 3: Data point correctly classified as [1,1] instead of [1,0] despite neighbor majority.

5 Conclusion

Multi-label classification is a very common real-world problem. There are many everyday tasks that utilize multi-label data, and the importance of accurately labeling such data extends far beyond these Xs and dots. Through this paper, it has been demonstrated that the adapted algorithm ML-KNN is an effective means of lazy multi-label classification in small datasets. It is an intuitive, resilient algorithm that works quickly and efficiently on small datasets. In the future, it would be interesting to compare the effectiveness of ML-KNN using different hyperparameters, such as different distance measures.

References

[1] Zhang, Min-Ling, and Zhi-Hua Zhou. ML-KNN: A Lazy Learning Approach to Multi-Label Learning NJU.EDU.CN. https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/pr07.pdf.