COSC522 Machine Learning Project 2 - Parametric vs. Nonparametric, Supervised vs. Unsupervised

Yangsong Gu

September 22, 2021

1 Objective

The objectives of this project are, first of all, to get an in-depth understanding of the difference between parametric learning and non-parametric learning. Both are under the umbrella of supervised learning. The second objective is to experience the difference between supervised learning and unsupervised learning. We will use the pima datasets in this project. You've applied parametric learning (three cases of discriminant funcitons) and obtained mediocre results in Project 1. Here, you'll use the nonparametric learning scheme as well as unsupervised learning scheme and compare with the results from Project 1 (i.e., parametric learning).

2 Data Sets

Two datasets will be used from Ripley's Pattern Recognition and Neural Networks, both are 2-category classification problems. The first is a synthetic dataset with 2 features where there are about equal number of samples in each category. The second is a dataset for diabetes in Pima Indians with 7 features where the number of diabetic patients is much less than that of the normal patients.

- The synthetic dataset: synth.tr (the training set) and synth.te (the test set)

  Preprocessing you need to do: remove the first row with any text editor. No need to write Python code for that. The labels are 0 and 1.

- The Pima dataset: pima.tr (the training set) and pima.te (the test set)

  Preprocessing you need to do: 1) Remove the first row with any text editor; 2) Change the labels from "Yes" and "No" to "0" and "1" respectively indicating 'with disease' and 'without disease' - you can do this using the same text editor; and 3) Normalize the data set to make the features comparable (or with the same scale). Suppose x is a data sample, m_i is the mean of each feature i, $sigma_i$ is the standard deviation of each feature i, then normalization is conducted by $\frac{(x-m_i)}{sigma_i}$. Keep in mind that you also need to normalize the samples in the test set. Be careful which mean and standard deviation you should use. (For each sample in the test set, use the same m_i and $sigma_i$ you derived from the training set.)

Table 1 summarizes the size of each dataset.

Table 1: Descriptive statistics of two datasets

dataset	training set		total	test set		total	features
	class 0	class 1		class 0	class 1		
Synthetic	125	125	250	500	500	1000	2
Pima	68	132	200	109	223	332	7

3 Performance Metrics

The same three metrics used in Project 1, i.e., 1) overall classification accuracy, 2) classwise classification accuracy, and 3) run time.

4 Tasks

- Task 1: Implement nonparametric learning using KNN
  - (20 pts) Plot a figure of 1x2 subplots that illustrates the overall classification accuracy vs. k using the synthetic and the pima dataset. Provide your observations based on the figure.
  - (10 pts) Generate a table similar to Tables 1 2 in Project 1 but adding one row reporting the performance of kNN. Use parameters in each learning algorithm that generated the best overall accuracy.
  - (5 pts) Comment (0.5 1 page) on the pros and cons between parametric learning and non-parametric learning when applied to datasets with different characteristics.
  - (15 pts) Using the synthetic dataset, can you draw the decision boundary of kNN with an explicit function like you did for the 3 cases of parametric learning? (use the k with the best overall accuracy) If not, what is a good way to illustrate the decision "boundary" on the test set? Overlay it on the same figure as the other three explicit boundaries from parametric learning, that you generated from Project 1.
- Task 2: Implement unsupervised clustering approaches (kmeans and wta) for classification purpose.Using the pima dataset. For fair comparison, make sure you use the statistics derived from the training set to normalize the test set, like you did in Project 1. That is, the clustering approaches need to use the normalized testset. Assume the number of clusters is 2. Also assume the prior probability ratio is 1:3 wherever needed.
  - (30 pts) Implement the two clustering approaches and plot a figure of "percentage of samples changing membership" vs. "epoch" for both kmeans and wta. That is, the figure should contain two profiles, one for kmeans and one for wta.
  - (5 pts) Comment on the different behaviors of kmeans and wta.
  - (10 pts) Present another table by modifying the table you generated in Task 1 by adding two more rows from kmeans and wta and one column with "number of iterations to converge".

  - (5 pts) Comment on the different performance between using supervised approaches and unsupervised approaches for classification purpose

- Bonus: (+10 pts) Design an algorithm that would incorporate prior probability in the implementation of kNN and kmeans.

5 Task 1 Implement nonparametric learning using KNN

Here's the pseudo code of applied kNN.

Algorithm 1 k Nearest Neighbours $Xtest, ytest, Xtrain, ytrain, k$ ▷ default k is 2

```
1: set up clock $t_1$.
2: for $row$ in $Xtest$ do
3:     compute distance of $row$ in $Xtest$ to each sample in training set $Xtrain$.
4:     sort the distance array and select the nearest $k$ neighbors.
5:     label the sample by the majority vote in neighbors, using labels from $ytrain$
6: end for
7: stop timing $t_2$, compute the run time $(t_2 - t_1)$
8: compute the overall accuracy and class-wise accuracy.
```

Algorithm 2 iter_k($Xtrain, ytrain, Xtest, ytest, k$) # iterate k value

```
   for k in range $(k_1, k_2)$ do #$k_1 \geq 1$
2:     run function kNN $(Xtrain, ytrain, Xtest, ytest, k)$
       store the overall accuracy and class-wise accuracy for k.
4: end for
   sort the overall accuracy and select the corresponding k value.
```

5.1 Overall accuracy vs. K

Figure 1 presents the overall classification accuracy changing with the K value in Knn algorithm, performed on synthetic and pima dataset, respectively. From the profile of the overall accuracy, it's not hard to find that **1)** accuracy does not monotonously increase with K, there are both fluctuations at the beginning and convergence period. on the one hand, a small k makes the model sensitive to local anomalies and exceptions, leading to over fitting. on the other hand, when K becomes large, the local structure of the data distribution could be overlooked, leading to under fitting. **2)** The best overall classification varies. The overall accuracy of synthetic dataset (0.916) is much better than Pima dataset (0.780). The reason causing this difference might be because of the imbalanced training data. The size of class 0 and 1 in synthetic dataset is equal (i.e., 1:1). in contrast, samples of class 1 is approximately **twice** of class = 0 in training set (see Table 1). Thus, from the perspective of final performance, we can draw a conclusion that Knn alogrithm is sensitive to the K value and imbalanced training data.

5.2 Performance metrics

Given equal priority of class 0 and 1, **Table** 2 shows the results of three different classifiers plus best kNN algorithm performed on synthetic test set. **Table** 3 presents the results of same algorithms applied on Pima test set.

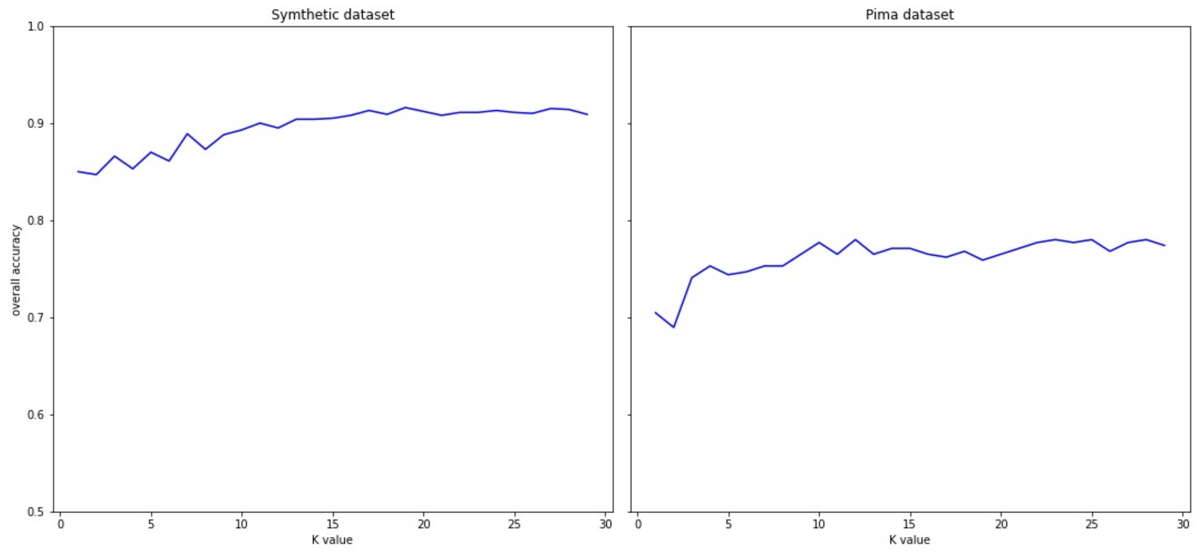


Figure 1: overall accuracy vs. k

Table 2: Model performance (synthetic dataset & Gaussian model)

	Overall acc.	Acc. of class 0	Acc. of class 1	Run time (seconds)
Euclidean dist.	71.30%	68.00%	74.60%	0.0170
Mahalanobis dist.	89.20%	90.00%	88.40%	0.0818
Quadratic	89.80%	90.80%	88.80%	0.1054
kNN (k = 19)	**91.60%**	**92.60%**	**90.60%**	**3.0017**

Table 3: Model performance (Pima dataset & Gaussian model)

	Overall acc.	Acc. of class 0	Acc. of class 1	Run time (seconds)
Euclidean dist.	74.10%	70.64%	75.78%	0.0060
Mahalanobis dist.	76.81%	72.48%	78.92%	0.0479
Quadratic	74.10%	61.47%	80.27%	0.0677
kNN (k = 12)	**78.01%**	**57.80%**	**87.89%**	**1.3741**

5.3 Comprehensive Discussion

1) **Precondition**. Parametric learning is constrained by a specified form. For instance, Gaussian probability density function (pdf) is assumed to describe the distribution of data points in three parametric learning models. The pre-selected pdf function would also influence the performance as we found that mutli-modal Gaussian improved the overall classification accuracy in synthetic data in Project. In contrast, the non-parametric learning models don't assume any distribution of data, which is more flexible than parametric learning.
2) **Speed**. Basically, the speed of algorithm is affected by the complexity of algorithm and the size of dataset. On the one hand, focusing on the same dataset, either table 2, or 3, kNN (non-parametric learning) takes much more time than three classifiers (parametric learning) to finish classification. For three classifiers, they only need to compute distance once time for each sample in test set. In contrast, kNN needs to compute the distance to each sample in training set for each sample in test set, which costs huge computation time. Therefore, the parametric learning can be used for real time classification. On the other hand, the complexity

of calculating the distance increases as number of features increases, leading to higher run time. if we expand the test set size to the same size as synthetic dataset and assume each sample would take the similar run time, then we would expect 4.1388 (i.e., (1000/332)*1.3741) seconds that is greater than the run time of synthetic test set. This suggests that the computation time would be augmented when more features involved in non-parametric learning.
3) **Accuracy**. Although non-parametric learning takes more time to do classification compared against parametric learning, non-parametric learning outperforms the non-parametric learning in terms of accuracy. For instance, the overall and class 0 accuracy generated by kNN are the highest among all approaches. Meanwhile, class 1 accuracy of kNN is still pretty close to the highest accuracy among three classifiers. For pima dataset, the overall and class 1 accuracy of kNN are best among all approaches while the accuracy of classification 0 is the lowest among the four methods. We could dive into the principle of algorithm to find out the reasons. In this regard, three classifiers measure the distance of a sample to cluster centers, while kNN search neighs locally and decide the label upon majority of votes, which reflects the local optimized decision. This local majority votes approach becomes vulnerable when the data is im-balanced, as the class 0 accuracy of kNN approach on pima dataset indicates.

5.4 Decision Boundary

Unlike the three classifiers, kNN does not have an analytic decision boundary since it non-parametric learning. Nonetheless, we could create many high-resolution points on training set space, and label them using kNN. Then, the decision boundary could be highlighted at the areas with distinct labels. if the grid resolution is high enough, we could find out a precise shape of decision boundary. Figure 2 demonstrates the three decision boundaries and one kNN boundary predicted from training set. Both generic Gaussian classifier and kNN produced the non-linear decision boundary while the latter correctly classified more samples.

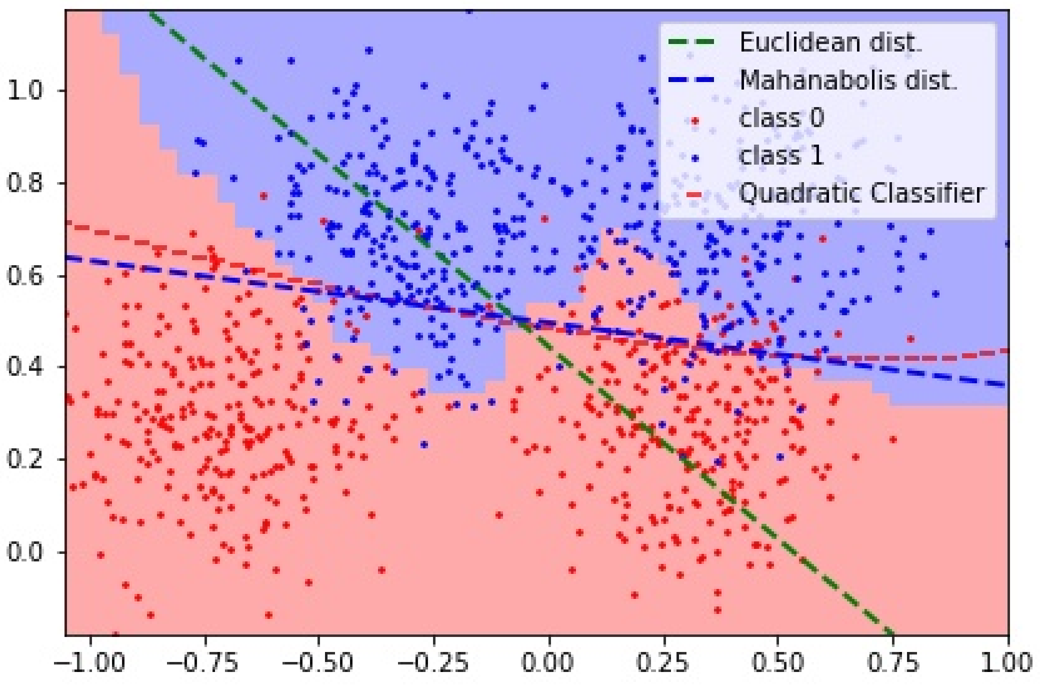


Figure 2: decision boundary visualization

6 Task 2 Implement unsupervised clustering approaches (kmeans and wta)

The pseudo code of kmeans and wta can refer to algorithm 3 and 4. Note that the pseudocode only takes k = 2 for simplicity, while the code was generalized for any k greater than 1.

Algorithm 3 Kmeans($Xtest, k$) # initial k value default k is 2

```
    Initialize:
        $diff_{label} = inf$    ▷ # $diff_{label}$ is the percent of membership changing after each epoch
        $epoch \longleftarrow 0$
    $set\ clock\ t_1$
3:  Step 1 randomly split the $Xtest$ into two groups $c_1$ and $c_2$
    Step 2 calculate the mean of each cluster $centroids_1, centroids_2$
    Step 3 Iterate to converge
6:  while $diff_{label} > 0$ do
        for row in $Xtest$ do
            compute the distance of $row$ to $centroids_1, centroids_2$.
9:          assign the sample to the nearest cluster.
        end for
        update the $centroids_1, centroids_2$ according to the clustering results.
12:     compute the $diff_{label}$
        $epoch \longleftarrow epoch + 1$
    end while
15: $end\ clock\ t_2$
    compute run time and performance
```

```
Algorithm 4 wta($Xtest$, $\varepsilon$, $k$)    ▷ winner take all algorithm, $\varepsilon$ is learning rate, default k is 2
Initialize:
    $diff_{label} = inf$ ▷ # $diff_{label}$ is the percent of membership changing after each epoch
    $epoch \longleftarrow 0$
set clock $t_1$
Step 1 randomly split the $Xtest$ into two groups $c_1$ and $c_2$
Step 2 calculate the mean of each cluster $centroids_1, centroids_2$
Step 3 Iterate to converge
while $diff_{label} > 0$ do
    for row in $Xtest$ do
        compute the distance of $row$ to $centroids_1, centroids_2$, respectively and assign the sample to the nearest cluster.
        update the $centroids_1, centroids_2$ by learning rate $\varepsilon$, $centroids_1 = centroids_1 + (row - centroids_1) \cdot \varepsilon$; $centroids_2 = centroids_2 + (row - centroids_2) \cdot \varepsilon$
    end for
    compute the $diff_{label}$
    $epoch \longleftarrow epoch + 1$
end while
end clock $t_2$
compute run time and performance
```

6.1 Algorithm convergence

Assume the number of clusters is 2, Figure 3 presents the convergence process of two unsupervised learning algorithm including kmeans and winner take all (wta) performed on pima dataset, respectively. In wta method, the learning rate is set as 0.05, 0.1, 0.2 to test its impact on algorithm convergence. x-axis represents the epoch while y-axis indicates the percentage of samples changing membership. For fair comparison, two algorithms were performed on the same initialized two clusters (i.e., the initial cluster centers are same in both experiments).

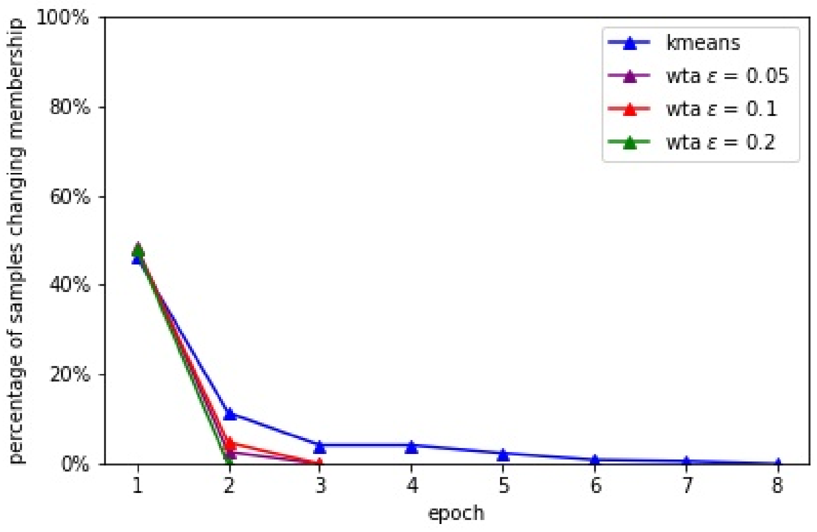


Figure 3: decision boundary visualization

6.2 Comment

1) It is evident that wta (ε = 0.05) clustering approach converges quickly which only takes three epochs. In contrast, the kmeans method converges much slower than wta, taking 8 epochs. 2) At the begining of clustering, the percentage of samples changing membership drops drastically. Taking the first two epochs as an example, wta declines more steeply than kmeans method. 3) after the first epoch, more sample changed in wta approach than in kmeans method. 4) from the algorithm perspective, the results indicate that online learning is more efficient than batch learning and online learning could mitigate the influence of anomalies on centroids.

6.3 Performance metrics

Table 4 summarized the performance of four supervised approaches (i.e., three classifiers and kNN) and two unsupervised approaches (i.e.,Kmeans and wta) for classifying the Pima test set. In this table, number of iterations is computed by the the number of epochs needed to converge and the number of samples in test set (i.e.,332). For supervised learning, algorithms learn from the data features and label, while unsupervised learning approaches classified without labels, the classification process is learned from the data features. Thus, they need some epochs of self-learning to discover the inherent structure of unlabeled data.

Table 4: Model performance (Pima dataset & Gaussian model)

	Overall acc.	Acc. of class 0	Acc. of class 1	Run time (seconds)	No. of iterations
Euclidean dist.	74.10%	70.64%	75.78%	0.0060	332
Mahalanobis dist.	76.81%	72.48%	78.92%	0.0479	332
Quadratic	74.10%	61.47%	80.27%	0.0677	332
kNN (k = 12)	78.01%	57.80%	87.89%	1.3741	332
kmeans (k=2)	71.08%	68.81%	72.20%	2.1070	2656
wta (ε = 0.05)	71.39%	72.48%	70.85%	0.9696	996

* 1 epoch = 332 iterations.

6.4 Supervised vs. unsupervised

Focusing on the overall accuracy, the supervised learning models tend to be more accurate than unsupervised learning models, with kmeans and wta producing the lowest overall accuracy among six approaches. When it comes to class-wise accuracy, we notice that kmeans and wta generate better accuracy of class 0 than other supervised learning approaches while they are stuck in a relatively low accuracy of class 1. What's more, this indicates that unsupervised learning approach can not accommodate the imbalanced data issue. When it comes to the run time and iterations of algorithms, unsupervised learning especially the kmeans approach take more time and iterations than supervised algorithms. It is worth noting that wta (learning rate

= 0.05) is much faster than the kNN approach, it takes twice iterations of kNN though. It also should be noticed that the run time of unsupervised learning approaches is highly associated with the initial status of the clusters (Na, Xumin, and Yong 2010).For instance, if the initial cluster center is close to the final converged cluster center, it would save a lot of iterations to adjust the membership of data points. In this regard, the wta approach accelerates clustering by adjusting the centroid each iteration with a small learning rate, and that's the reason why wta is much faster than kmeans in convergence.

6.5 Bonus

6.5.1 kNN with prior information

Incorporating prior information is one of the advantages of Bayesian decision rule. Log of prior probability is added to the Bayesian determinant. With the similar deal, there are several approaches incorporating prior information into kNN algorithm. Herein, the main idea is how to reconsider the "distance" and "majority voting" rule in kNN algorithm to improve the overall accuracy. Refer to Figure 5, they are:

- **approach 1:** use prior probability to weight the number of different neighbors in each search.

$$\begin{aligned} f(i=0) &= V(i=0)\cdot p_0 \\ f(i=1) &= V(i=1)\cdot p_1 \\ label &= \arg\max_i(f(i=0), f(i=1)) \end{aligned}$$

  where V(i) contains the number of nearest neighbors belong to class i.p_0 and p_1 are prior information of class 0 and 1, respectively.

- **approach 2:** use distance weighted prior probability

$$\begin{aligned} f(i=0) &= \frac{\sum_j^m d_{j0}p_0}{\sum_j^n d_{j0}p_0 + \sum_j^n d_{j1}p_1} \\ f(i=1) &= 1 - f(i=0) \\ label &= \arg\max_i(f(i=0), f(i=1)) \end{aligned}$$

  where d_{j0} denotes the distance between sample j in test set and the class 0 neighbors in a query.

- **approach 3:** consider both fraction of votes and weighted prior probability. To maximize the 'fraction', we can either take the summation or take the multiplication of them.

$$\begin{aligned} f(i=0) &= \frac{\sum_j^m d_{j0}p_0}{\sum_j^n d_{j0}p_0 + \sum_j^n d_{j1}p_1}\cdot\frac{K_0}{K} \\ f(i=1) &= \frac{\sum_j^m d_{j1}p_1}{\sum_j^n d_{j0}p_0 + \sum_j^n d_{j1}p_1}\cdot\frac{K_1}{K} \\ label &= \arg\max_i(f(i=0), f(i=1)) \end{aligned}$$

  where K is number of neighbors used in kNN, K_0 and K_1 denote the number of class 0 and class 1 in a query.

All three approaches would only change the 'label' rule, the other part is still same to common kNN algorithm 1. Figure 4 presents the overall accuracy changing with the prior probability of class 0, corresponding to three approaches described above. K is set as 19 and 12 as they generate the best overall accuracy in synthetic and pima dataset, respectively. For two classes classification, when $p_0 = 0.5$, prior probability is uninformative. The profile of three approaches displays a convex shape and overall accuracy reaches the peak around $p_0 = 0.5$.This makes sense because the percentage of two classes is equal in synthetic training set. Comparing three approaches, approach 1 and 2 generate the similar performance profile, while approach 3 significantly improved the overall accuracy across the different prior knowledge. The right hand side figure shows the overall accuracy of pima dataset. Note that the prior probability of class 0 and class 1 is about 2:1, the grey dash line visualizes p_0 in training set. Different from synthetic dataset, three approaches applied on pima test set obtained the peak performance at different prior probability. In addition, it should be noticed that when p_0 goes opposite direction with the class frequency observed in training set, the performance will decline drastically. This result suggests that the weighted prior probability and majority voting rule could make difference in improving the overall accuracy.

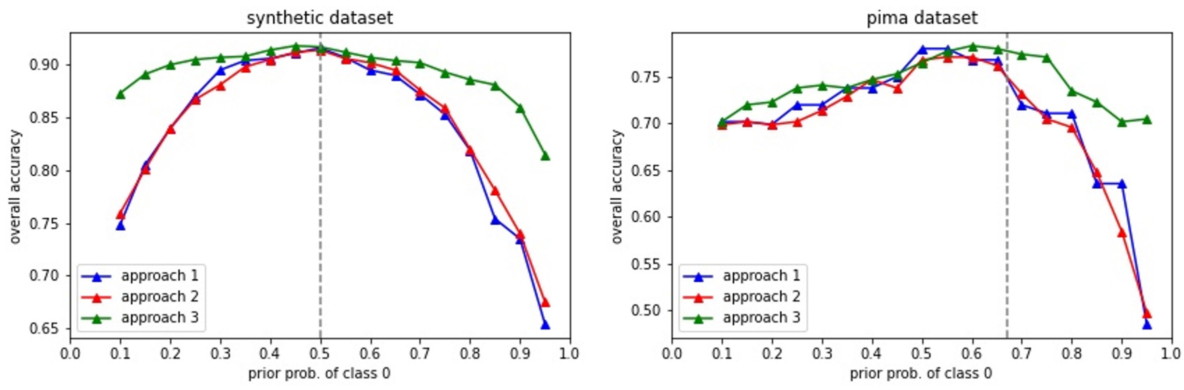


Figure 4: Overall accuracy vs. prior probability of class 0 (kNN)

6.5.2 kmeans with prior information

For kmeans algorithm, the rule of labeling data point is totally different from the kNN. The core procedure of kmeans is adjusting cluster centroids composed of samples with high degree of membership. Therefore, we can start with improving the centroids calculation considering the prior information. Note that the prior information just tells us the frequency of different classes which could be used to instruct the clustering process. Nonetheless, the prediction results may or may not reflect such frequency on test set. Refer to Figure 7, we can have such strategy on updating centroids after each epoch:

- use prior information as a threshold to filter out samples for updating centroids. In this way, anomalies and noisy samples could be eliminated and impact of them on centroids formation could be mitigated as well. Different from kNN, prior information in this method is always informative even when they are equal.

This strategy would only change the way to update centroids in kmeans algorithm 3. After trying different prior probability of cluster 0 (p_0), the result finds that when $p_0 = 0.5$, the overall accuracy is highest, which is higher than previous experiment(71.08%), as Figure 6 shows.

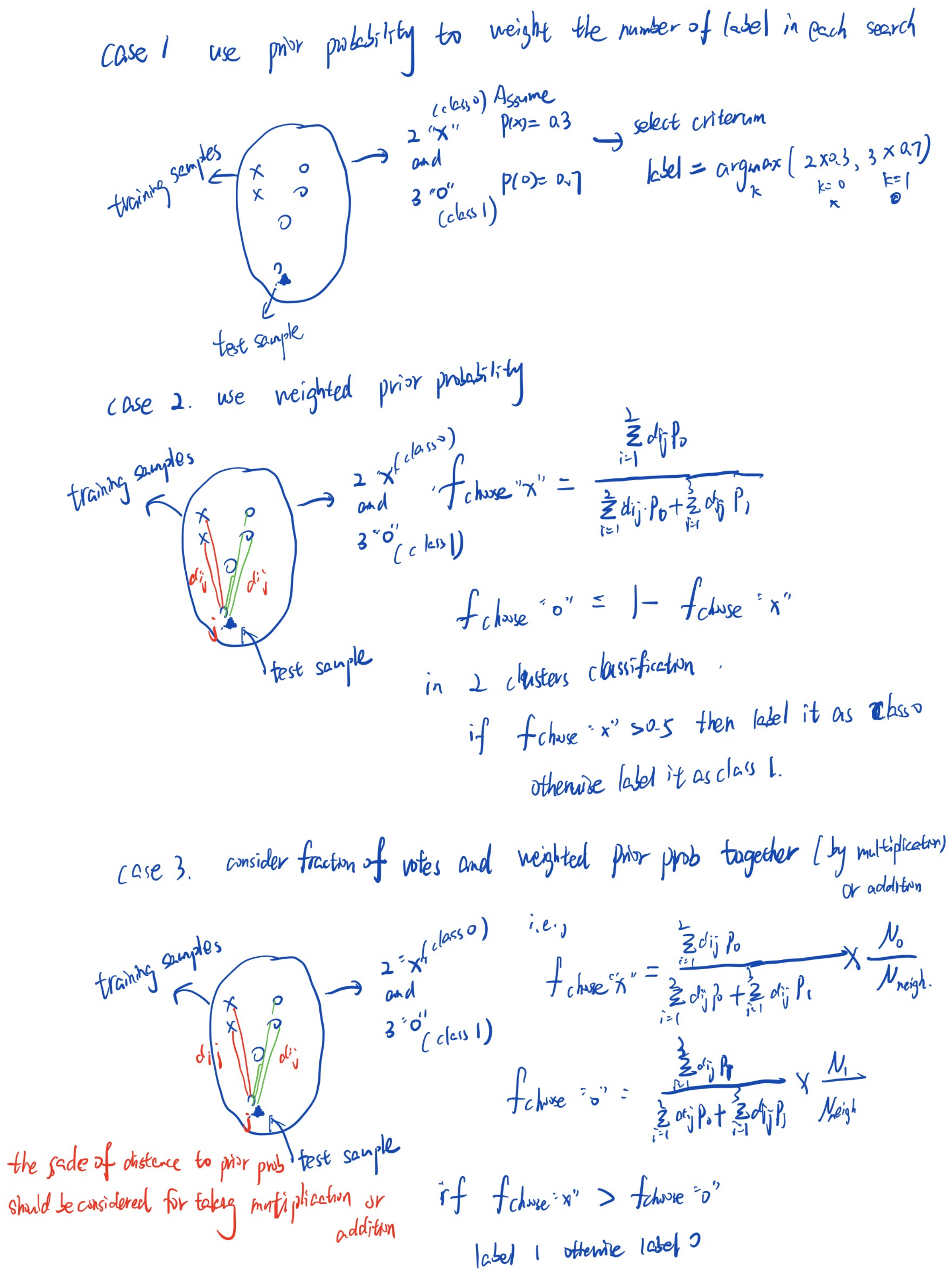

case 1 use prior probability to weight the number of label in each search
training samples
test sample
2 "x" (class 0) and 3 "o" (class 1)
Assume p(x) = 0.3, p(o) = 0.7
select criterum
label = argmax_k (2×0.3, 3×0.7), k=0 x, k=1 o
case 2. use weighted prior probability
training samples
test sample
d_ij
2 "x" (class 0) and 3 "o" (class 1)
f_chose "x" = Σ_{i=1}^{2} d_ij P_0 / (Σ_{i=1}^{2} d_ij·P_0 + Σ_{i=1}^{3} d_ij P_1)
f_chose "o" = 1 − f_chose "x"
in 2 clusters classification.
if f_chose "x" ≤ 0.5 then label it as class 0
otherwise label it as class 1.
case 3. consider fraction of votes and weighted prior prob together (by multiplication or addition)
training samples
test sample
d_ij
2 "x" (class 0) and 3 "o" (class 1)
i.e.,
f_chose "x" = Σ_{i=1}^{2} d_ij P_0 / (Σ_{i=1}^{2} d_ij P_0 + Σ_{i=1}^{3} d_ij P_1) × N_0 / N_neigh.
f_chose "o" = Σ_{i=1}^{3} d_ij P_1 / (Σ_{i=1}^{2} d_ij P_0 + Σ_{i=1}^{3} d_ij P_1) × N_1 / Neigh
if f_chose "x" > f_chose "o"
label 1 otherwise label 0
the scale of distance to prior prob should be considered for taking multiplication or addition

Figure 5: Three cases of using prior information in kNN algorithm

The enhancement is not very significant, the probable reason might be that the old centroids calculated from all labeled data are affected by the anomalies and they are already biased before computing distance between each sample and centorids. Therefore, in the future improvement, the variance of each class should should be considered to form new centroids.

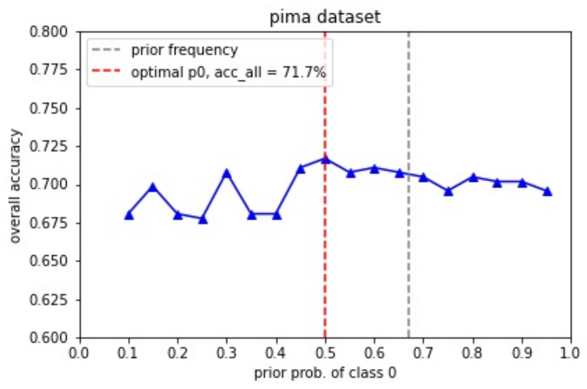
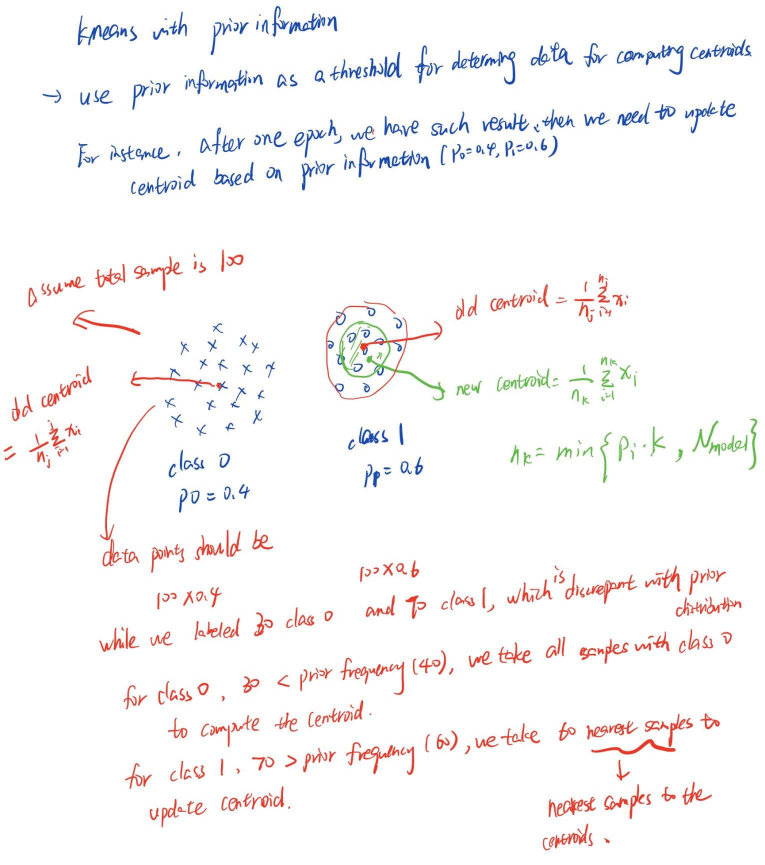


Figure 6: Three cases of using prior information in kNN algorithm



kmeans with prior information

→ use prior information as a threshold for determing data for computing centroids

For instance, after one epoch, we have such result, then we need to update centroid based on prior information ($P_0=0.4, P_1=0.6$)

Assume total sample is 100

old centroid $= \frac{1}{n_j}\sum_{i=1}^{j} x_i$

class 0 $p0 = 0.4$

class 1 $p_p = 0.6$

old centroid $= \frac{1}{n_j}\sum_{i=1}^{n_j} x_i$

new centroid $= \frac{1}{n_k}\sum_{i=1}^{n_k} x_i$

$n_k = \min\{P_i \cdot k, N_{model}\}$

data points should be 100×0.4 and 100×0.6 while we labeled 30 to class 0 and 70 to class 1, which is discrepant with prior distribution

for class 0, 30 < prior frequency (40), we take all samples with class 0 to compute the centroid.

for class 1, 70 > prior frequency (60), we take 60 $\underbrace{\text{nearest samples}}_{\downarrow}$ to update centroid.

nearest samples to the centroids.

Figure 7: apply prior information to kmeans algorithm

References

[1] Shi Na, Liu Xumin, and Guan Yong. “Research on k-means clustering algorithm: An improved k-means clustering algorithm.” In: *2010 Third International Symposium on intelligent information technology and security informatics*. Ieee. 2010, pp. 63–67.