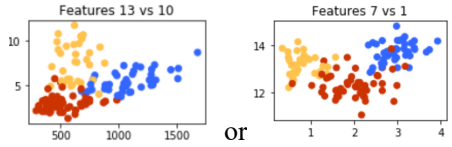
COMS21202: Symbols, Patterns and Signals
Coursework 2: A Memory of Wine Report

By: Yang Li (yl15893) & Haoyang Wang (hw17183)

1 Feature selection.

After all combinations are plotted, the first pair we selected are 10 and 13 which looks best to us. We then tried to run K-NN for using k from 1 to 7 for all combinations and we were surprised to find that the pair 1 and 7 delivers the best result (accuracy table in section 2).



Best looking Best result

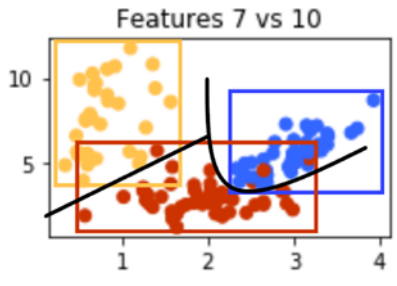
We didn't want to run K-NN on all combinations to get the 'best result', because when more than 2 features are needed, the compute complexity will increase exponentially. So, we started thinking about the how 'pretty looking' pair delivers 'pretty results' and put some efforts on writing a low complexity algorithm to help us select a pair rather than by just observing or run K-NN on everything.

The first idea is to rank all features. We create a measurement that adds together number of overlapping points of three class. Overlapping points here are defined as points of one class locating between the maximum and minimum of another class. This measurement tells us how pure each class is in its range and we can rank all the feature by this number. As a result, the best one is 7th (78), following by 13th (100), 12th (104) and 11th (105) etc.. So, if we need two features, we will pick feature 7 and 13.

But that is still quite different from K-NN result. The first measurement reflects total purity of classes, but when features are combined, if the lower ranked features could complement each other, the combined pair may get better result than the combination of "best features" in previous ranking. Then we came up with another measurement. This time we broke down three purity into six factors, that are how many class A data in range of class B. And the new measurement is calculated in this formula:

$$\mathrm{M}_{F_1 \ldots F_n} = \sum_{i=1}^{6} \prod_{j=1}^{n} F_j[i] \text{ where } \mathrm{F} = [R \text{ in } B, Y \text{ in } B, Y \text{ in } R, B \text{ in } R, B \text{ in } Y, R \text{ in } Y]$$

Which n means the number of combined features, in our case n=2. We multiple all factor i first means factor in other features will be the weight of i. For example, if $F_1[0] = 0$, then $\prod_{j=1}^{n} F_j[0] = 0$. And this will fix the inaccuracy of the old algorithm during feature combination.

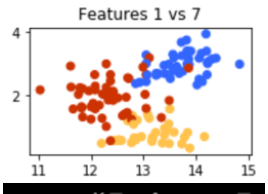


The new algorithm will run all the combination (but much lower complexity than K-NN), calculate measurements and select the smallest one. This time our result is 7 and 10. In K-NN result, pair [7,10] is better than [7,13] in the first version, but still not the best. We find there is a defect existing in both two version: in the following picture, the colouring box represents the range we calculated purity, but like the overlapping rectangular area of red and yellow, it can be separated by a diagonal black line nicely

but our algorithm will think the data in that area as “overlapping points”. After considering we gave up fixing this and use the latest [7,10] and final algorithm as result. Anyway, this algorithm gives us a good result that close to the best. On the other hand, we thought [1,7] is too perfect (accuracy 1.0 in both K-NN and naïve Bayes) which may contain an overfitting problem.

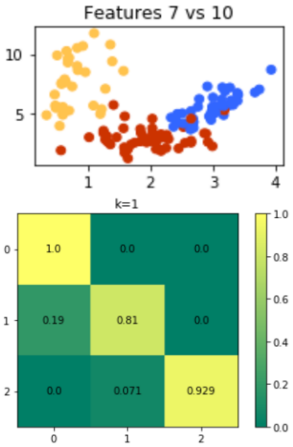
2 K-Nearest Neighbours

The K-Nearest Neighbours classifier is a method to predict the class of a given unknown sample. By finding k points in training data nearest to the unknown sample point, and finding their corresponding classes, the class of the sample point is determined by the majority class in these k points. In a 2-D scenario, it can be understood as drawing a circle around the sample point, containing k training points in the circle, the class holds the most places gives its name to the sample point. It is simple, intuitive and has fairly reasonable accuracy. It is a non-parametric algorithm, so it makes no assumptions, making it usable with any data set. Evolving K-NN is also simple, as it does not build any model, adding new training points will not require the whole system to recalculate, so it adapts quickly to changes.



K	Accuracy
1	0.89
2	0.91
3	0.94
4	0.94
5	1
7	1

Outliers will have minor influence on the result, especially when using a reasonably larger k. The figure on the left shows the scatter plot of features 1vs7 of the training set, we can see clearly that there are a few red outliers mixed into blue and yellow points. The table shows accuracy of K-NN classifier for different Ks. There’s a rising trend of accuracy along with the increment of K. This is because when using a small K, only training points near the sample point influences the result, it makes the model more complicating and causes overfitting. However, it is not the larger K is the better. Consider an extreme situation where we take K=N (N=number of training data), then the classifier will always return the same result. Large K simplifies the model but taking training points far away from the sample point also causes error. We think the best way to choose K is to use cross validation: set multiple test sets, test them with different K values and use K with the best accuracy.



On the left shows the confusion matrix of the result calculated by K-NN using features 7 and 10 and K=1. Labels 0, 1, 2 are represented by blue, red and yellow on the scatter plot respectively. According to the confusion matrix, there’s no confusion between yellow and blue points, as blue and yellow are well separated on the scatter plot. The biggest error is recognizing 19% red points as blue, which can also be explained on the scatter plot. Some red points are located within the high-blue-density region, so if any test points are distributed the same way, they will be recognised as blue, especially for a larger K.

3 Alternative Classifier (Naïve Bayes)

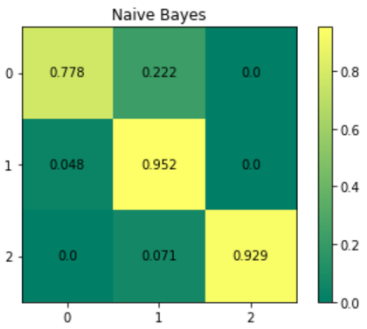
rank	f1	f2	accuracy
1	1	7	1.000
2	1	12	0.925
3	1	6	0.868
4	1	2	0.830
5	1	11	0.830
6	7	10	0.830
7	1	8	0.811
8	3	7	0.811
9	2	9	0.792
10	6	10	0.792

K-NN accuracy ranking

rank	f1	f2	accuracy
1	1	7	1.000
2	7	13	0.943
3	12	13	0.925
4	1	12	0.925
5	10	13	0.906
6	1	11	0.906
7	1	6	0.887
8	1	13	0.887
9	7	10	0.887
10	2	13	0.868

Naïve Bayes accuracy ranking

We are using Naïve Bayes algorithm as our alternative classifier. Since the data points are continuous, Gaussian Naïve Bayes is needed. We make the assumption that the data is parametric – it follows a 2-D normal distribution, and the features are independent within each class. Then by finding the mean and covariance of the training set, we can calculate the possibilities of a sample point belonging to each class and the highest one as decision. It takes more pre-treatment on training data but also perform better. This table shows the 10 best results carried out by Naïve Bayes, by comparing it to the one we achieved for K-NN, we can tell it has a higher overall accuracy than K-NN.



The confusion matrix of using Naïve Bayes on features 7 and 10 is very different from using K-NN: we can see the accuracy of recognizing blue points decreased significantly. On the scatter plot, we can see blue points does not exactly follow 2-D normal distribution. There's a dense area in the centre and another dense area at the bottom-left side of the blue area, making the blue mean shift to the bottom-left side and interferes with the red area. Meanwhile, the red and yellow points validate our assumption a lot better than blue. The overall accuracy of using Naïve Bayes is still higher than K-NN (NB=0.887, K-NN=0.830, shown in the table above), but apparently Naïve Bayes is highly sensitive to the data model, thus can only be applied on limited occasions.

Naïve Bayes also works better on smaller data sets as we are building a parametric model to determine the classes of sample points. If we can make an accurate training set, even if the size is small, it can build a complete classify model, and produce accurate results. In situations like classifying spam emails where we can make a well fabricated but small training set, we can make most use out of Naïve Bayes.

4 K-NN three features

In K-NN_3d, the first challenge is to pick the third feature. We simply run other 11 features with [7,10] in K-NN algorithm and evaluate how them perform in different k. By observing the results, we manually picked feature 1 and run K-NN again on [1,7,10] and result as follow:

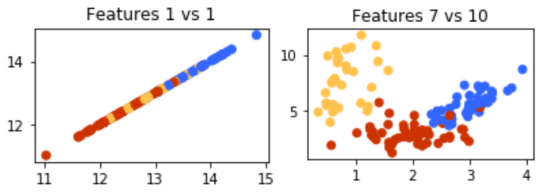
K	Accuracy
1	0.91
2	0.87
3	0.85
4	0.87
5	0.79
7	0.83

feature [7,10]

K	accuracy
1	0.92
2	0.91
3	0.92
4	0.89
5	0.89
7	0.85

feature [1,7,10]

By comparing with K-NN feature [7,10], obviously the accuracy of [1,7,10] increased 0.01~0.1 in different k. The reason of feature 1 contribute positive result can be observed by these two graphs:

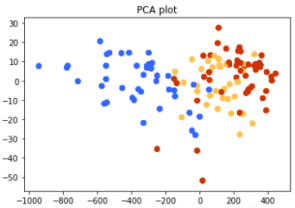


The blue and yellow points are separated perfectly, which could be proved by confusion matrix, but the blue and red points are mixed in the boundary. On the other hand, for only feature 1, we can see that red and blue points are nearly not overlapped. So the feature 1 complement the short of feature [7,10] and the 3 feature combination will be better.

f	measurement
1	[8 22 32 24 32 30]
2	[32 25 34 41 41 38]
3	[34 34 34 41 37 34]
4	[36 22 34 41 16 38]
5	[21 24 34 41 38 40]
6	[20 2 33 39 9 40]
7	[11 0 27 30 0 10]
8	[29 14 34 41 40 46]
9	[41 12 34 40 34 42]
10	[9 23 13 26 38 5]
11	[38 3 11 41 3 9]
12	[35 0 20 36 0 13]
13	[8 9 34 11 6 32]

In more detail, the “complementation” can be shown in practical way – the 6-factors measurements we introduced in the feature selection part. The double coloured box on the numbers represent the feature f present good separation on those colours. We can see feature 7 separate blue and yellow perfectly, 10 do well on red and yellow and the third feature 1 contributes lower red and blue numbers to whole result. Conversely, our analysis can prove the meaning of our 6-factor measurement.

5 PCA



1 0.7358490566037735
2 0.7358490566037735
3 0.7169811320754716
4 0.6792452830188679
5 0.7358490566037735
7 0.7169811320754716

We implemented the PCA method with different k and the result as follow. The average of PCA accuracy is around 72, which is much lower than K-NN or naïve Bayes. We think there are two reasons. The first one is PCA merged all features includes some with bad performance of scattering data, but in previous algorithm we are only using selected features that are considered as ‘above average’ in all 13 features. Besides, different features have different distribution of three classes. When they are projected together, such as conflict distribution will make the clustering result even worse.