Symbols, Patterns & Signals - CW2: A Memory of Wine

Saif Anwar & Tommy Van Aalst

Side Note

In the report we have used the features in the range of 1-13.

Choosing the best features

To choose the best features we observed all the plots of the pairwise combinations of the features and tried to find the ones which separated the data points into the three classes the best. There are some which clearly do not work, because the data points are overlapping and there is clearly no way to separate them into their three classes.

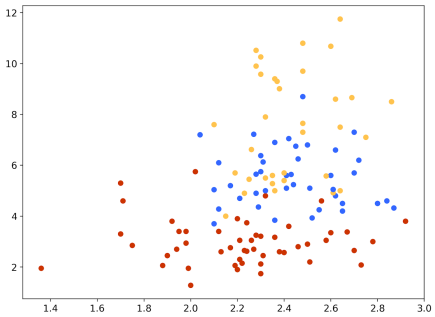

Fig 1: Plot of features 3 and 10

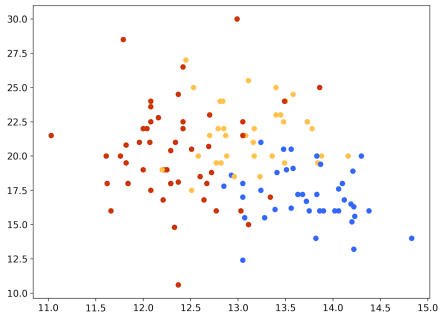

Fig 2: Plot of features 1 and 4

Here you can see in Fig 1 that although the red class is somewhat separable from the other two, the blue and yellow completely mix. It would be impossible to separate them in any way. In Fig 2 each class is in its own part of the plot, however in the middle they all overlap, again meaning there is no way to clearly separate the points into their three classes.

There are also a quite a few which do all work well, like these ones below:

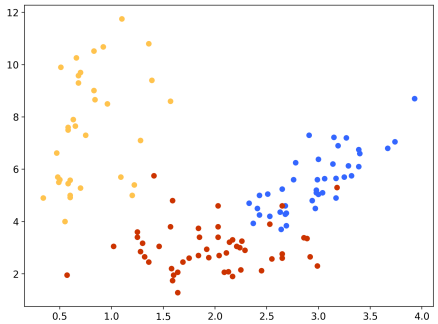

Fig 3: Plot of features 7 and 10

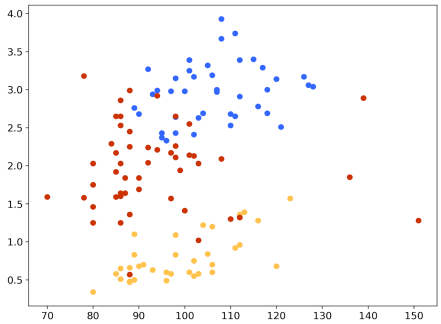

Fig 4: Plot of features 5 and 7

Here you can see clearly in Fig 3 that you can separate each of the classes well. There is some slight overlap on the edges but not much. Also, each class only has a boundary to one other class. All three classes don't overlap together in the middle so the boundary data could be only one of two classes, not one of three, therefore resulting in more accurate predictions. Fig 4 is quite good however there is a little bit of overlap between the blue and red classes and the red class has some outliers spread out. We concluded to choose features 7 and 10.

We also ran a script to calculate the accuracies of each pairwise combination. We found that the features we chose gave relatively good accuracy in each of our classifiers.

K-Nearest Neighbours

We implemented the K-Nearest Neighbours (K-NN) algorithm using functions to represent the steps. The first step is to calculate the Euclidean Distance between the wine in question from the wine_test set and every wine from the wine_train set. For a two-dimensional K-NN this will only be for the two selected features using the following formula:

$$d(x,y) = \sqrt{\sum_{n=1}^{k} (x_i - y_i)}$$

From all of these distances we can now find the k smallest distances to get the nearest neighbours when the test wine is plotted onto the scatter of the train sets selected features.

Now we have to assign a class to each of the neighbours by reading the wine_train_labels file to get an idea of what classes are closest to our test wine graphically. From this we can assume and assign a class to our test wine. Depending on the selected number of nearest neighbours, k, we may or may not have a modal class amongst our neighbours. If k=1 then we have an obvious mode of the single class and our test wine can be assumed to be of this class. Any k higher than this invites the possibility for multiple mode classes in the nearest neighbours. In this situation we recalculate the modal classes for k-1 neighbours which eliminates the furthest neighbour. This is repeated until we successfully find a mode which is then assigned as our test wines class.

We can repeat this process for each of the wines in the wine_test set and create our list of predictions for the test set. By comparing our predictions with the provided wine_test_labels, we can calculate an accuracy for our algorithm using the following formula:

$$Accuracy = \frac{\text{Number of Correctly Predicted Wines}}{\text{Total Number of Wines}}$$

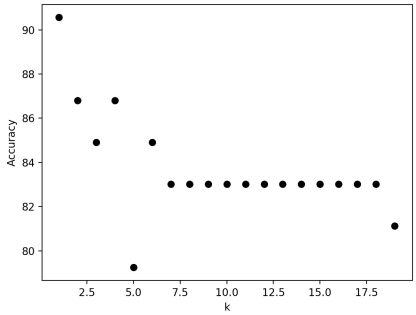

Fig 5: Plot of k and accuracies

Fig 5 shows the effect increasing the number of neighbours, k, has on the accuracy of the classifier whilst the selected features remain the same. As can be observed from the figure, features (7 and 10) would get a reasonably high accuracy of 90.57% at k=1. As the number of nearest neighbours is increased, we observe a lower accuracy until k=5 where after that it begins to increase again. This would be due to an increase in incorrect classification based on closest neighbours. The figure shows the accuracies for k in the range 0-20. It can also be seen that at a certain value of k, the accuracy plateaus. This can be assumed to be due to the wine in question being in a region where the modal class is the mode by a significant number.

Here are the confusion matrices for each of the values of $k \in (1,2,3,4,5,6,7)$

$$k=1\begin{bmatrix} 1 & 0 & 0 \\ 0.19 & 0.81 & 0 \\ 0 & 0.07 & 0.93 \end{bmatrix} \quad k=2\begin{bmatrix} 1 & 0 & 0 \\ 0.24 & 0.76 & 0 \\ 0 & 0.14 & 0.86 \end{bmatrix} \quad k=3\begin{bmatrix} 0.89 & 0.11 & 0 \\ 0.19 & 0.81 & 0 \\ 0 & 0.14 & 0.86 \end{bmatrix} \quad k=4\begin{bmatrix} 1 & 0 & 0 \\ 0.24 & 0.76 & 0 \\ 0 & 0.14 & 0.86 \end{bmatrix}$$

$$k=5\begin{bmatrix}0.78 & 0.22 & 0\\ 0.24 & 0.76 & 0\\ 0 & 0.14 & 0.86\end{bmatrix} \quad k=6\begin{bmatrix}0.94 & 0.06 & 0\\ 0.24 & 0.76 & 0\\ 0 & 0.14 & 0.86\end{bmatrix} \quad k=5\begin{bmatrix}0.89 & 0.11 & 0\\ 0.24 & 0.76 & 0\\ 0 & 0.14 & 0.86\end{bmatrix}$$

It can be observed that there are no wrong predictions between 1 and 3 (1 being classed as three and vice versa). This was expected as there is not an approximate boundary, in the scatter plot of the features, that is shared by both classes 1 (Blue) and 3 (Yellow). Also we can see that there is the most discrepancy between the classification of wines that are actually class 2 (Red) but being classed as 1. This was to be expected from the visible overlap between the two classes in the plot of the features (Fig 3).

Alternate Classifier - Naïve Bayes Classifier

We implemented Naive Bayes Classifier as our alternative classifier. With the same features used for K-NN, (7 and 10), we got an accuracy of 62.3%. This is a fair bit worse than K-NN with one neighbour which has an accuracy of 90.6%. This could be due to the fact that K-NN is a discriminative classifier, whereas Naive Bayes is a generative classifier, therefore K-NN looks at just classifying the given data with a boundary between the two datasets by only looking at the data but Naive Bayes tries to assume where to classify the data based on how the previous data was generated. Most of the time discriminative classifiers are better than generative ones and our data supports this claim.

K-NN has running time $\mathcal{O}(n)$ whereas Naive Bayes is almost $\mathcal{O}(1)$ time. This would mean in a larger dataset, K-NN would be less efficient, but since our data set is fairly small, there is no need to worry about run time and we can choose the best classifier. With K-NN, the decision boundaries can take on any form, since its non-parametric, which when we have good feature selection, means accuracy can be very high whereas Bayes does not do this.

Because of the fact that K-NN optimises locally by viewing an area around the given datapoint, outliers would seriously affect the accuracy since they could be in a cluster of different data points. When there are more outliers in the data, Naive Bayes would be a better choice of classifier since it doesn't have this problem. However, we have chosen good features with very few outliers and so in this case, with these features, K-NN would be preferable and the results support this.

K-Nearest Neighbours with three features

Constructing the K-NN Classifier with 3 features adds a new dimension to calculating the Euclidean Distance between the points. It incorporates all the possible combinations of the three features. For example, we selected features 2, 7 and 10; proceeded to plot them on a 3-dimensional axis to observe the boundaries between the classes. Observing the 3d scatter plot perpendicular to each of the axis will give you the 2d plot of the other two features. Doing this in turn will give you an idea of the relationship between each of the features individually.

Here are some plots of features 2, 7 and 10 in 3d space from each axis:

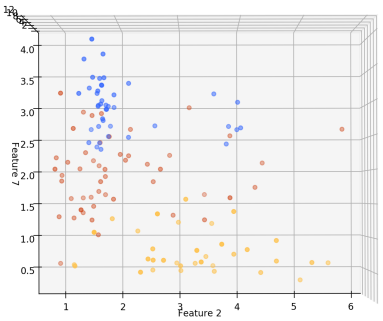

Fig 5: Plot of features 2 and 7 in 3d space

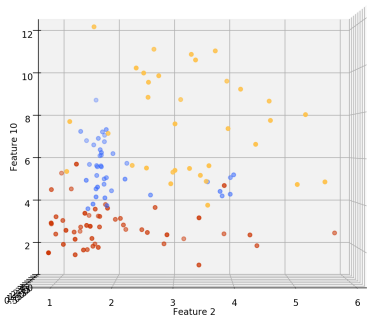

Fig 6: Plot of features 2 and 10 in 3d space

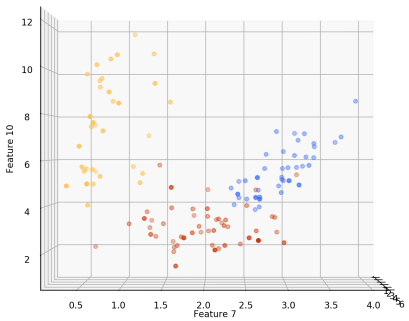

Fig 7: Plot of features 7 and 10 in 3d space

Using three features can either be beneficial by picking features that compliment each other in their individual plots against the other features, or can in fact make the classifier worse overall due to a single pair of the combinations not

showing a strong distinctive relationship between the classes. To select our 3 features, we kept our 2 features from our original K-NN classifier so we could see what difference a single new dimension would make without changing our other features. We used a script to calculate the accuracy from each selection and we deemed feature 2 to be a fitting addition to the K-NN classifier. As can be seen in the figures above, each of the pairwise combinations of features show reasonable boundaries between each of the classes confirming that (2,7,10) is a suitable set of features to use for our 3d K-NN classifier.

We can compare the accuracy between using 3 features and using 2 features by comparing the confusion matrices for $k \in (1,2,3,4,5,6,7)$:

$$k=1\begin{bmatrix}1 & 0 & 0\\ 0.1 & 0.9 & 0\\ 0 & 0.07 & 0.93\end{bmatrix} \quad k=2\begin{bmatrix}1 & 0 & 0\\ 0.24 & 0.76 & 0\\ 0 & 0.07 & 0.93\end{bmatrix} \quad k=3\begin{bmatrix}0.94 & 0.06 & 0\\ 0.19 & 0.81 & 0\\ 0 & 0.07 & 0.93\end{bmatrix} \quad k=4\begin{bmatrix}0.94 & 0.06 & 0\\ 0.24 & 0.76 & 0\\ 0 & 0.07 & 0.93\end{bmatrix}$$

$$k=5\begin{bmatrix}0.89 & 0.11 & 0\\ 0.19 & 0.81 & 0\\ 0 & 0.07 & 0.93\end{bmatrix} \quad k=6\begin{bmatrix}0.89 & 0.11 & 0\\ 0.24 & 0.76 & 0\\ 0 & 0.07 & 0.93\end{bmatrix} \quad k=7\begin{bmatrix}0.89 & 0.11 & 0\\ 0.24 & 0.76 & 0\\ 0 & 0.07 & 0.93\end{bmatrix}$$

From the confusion matrices of K-NN3d we can deduce that the accuracies are generally better than the K-NN with 2 features. However The plot of features 2 & 10 shows multiple crossovers the classes and doesn't show distinct boundaries. This would negatively affect our classifier when taking more neighbours into account as shown by the trend where accuracy decreases as k increases. This can be seen to possibly be due to the pairwise combination of features 2 & 7 where there is significant crossover between class 1 (Blue) and class 2 (Red).

K-Nearest Neighbours Using Principle Component Analysis (PCA)

K-NN is a technique that is generally likely to overfit the data which is why there is a need to optimise the value of k and carefully select features which represent the train data and future data in a fair way. One way to reduce the effect of overfitting is to increase the size of the training data but with K-NN being $\mathcal{O}(n)$ it would lead to the program becoming slow. Principal Component Analysis (PCA) is a method to reduce the dimensionality of the data to change a large data set with a large number of variables to a small set that still contains most of the data from the large set. It extracts the most useful parts of the data whilst aiming to maintain the same level of accuracy within the classifier. Using the PCA implementation from the Scipy library, we reduced our training data.

The following is a scatter plot of the PCA-reduced data and the original scatter plot of the features we selected:

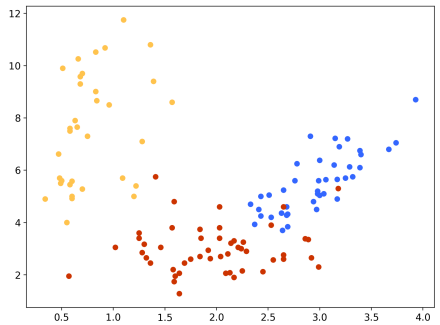

Fig 8: Plot of features 7 and 10

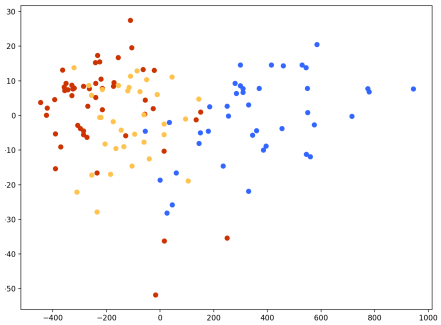

Fig 9: Plot of PCA-reduced training data

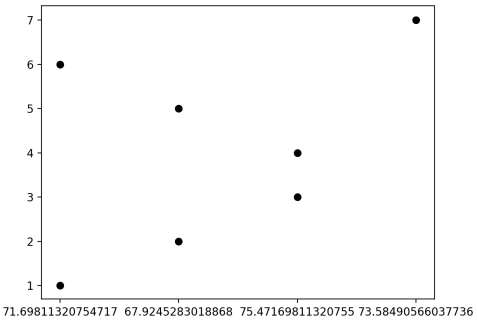

Fig 10: Plot of Accuracy of K-NN-PCA with k

From the PCA scatter plot in Fig 8 we can see that although there is crossover between the different classes which shows in the accuracy. Observing, Fig 10, the accuracy, using the PCA data, in general is lower than the accuracy of using K-NN with our two manually selected features, which is expected. As described above, PCA focuses on making the program faster which has a tradeoff between speed and accuracy. However using PCA has reduced the chance of us overfitting to the training data by using collating all the features to create a new data set rather than excluding all the data except for our selected features.