Coursework 4

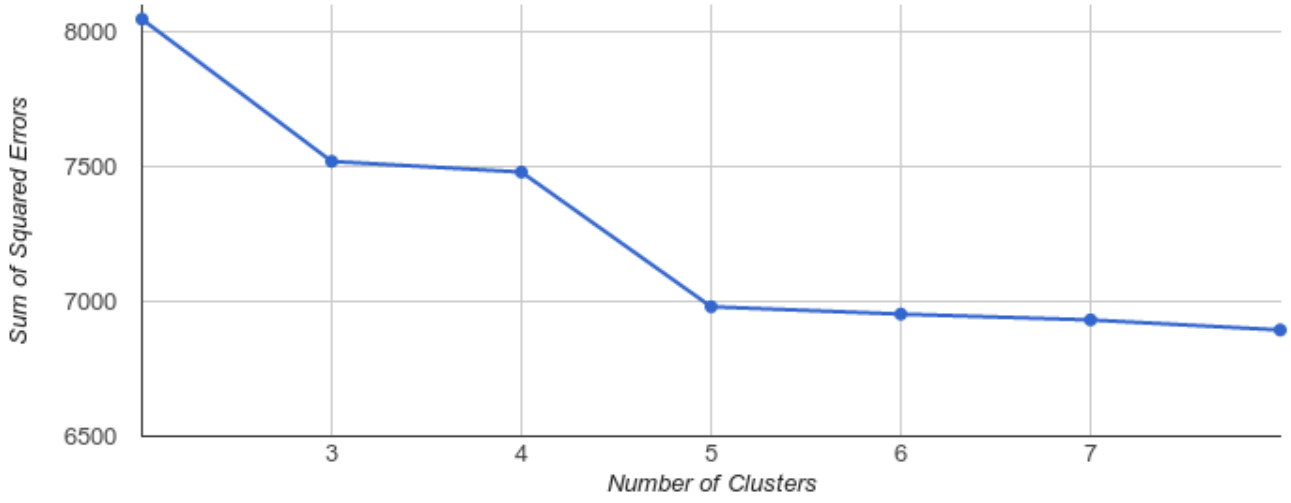
1. a

The clusters do not correspond to the classes very well. We can observe that for *Cluster* 2, we do not have any class assigned to it. Therefore, there are only ⅘ classes assigned to clusters. Class 1 is generally fairly accurate with a low number of misclassifications. Class 2 is highly misclassified with 432 assigned to class 2 while belonging to class 3. Class 3 is not present at all. Class 4 is fairly accurate at classification with a slight error when classified as class 4 while being class 5. Class 5 is highly accurate with only 9 instances misclassified belonging to class 4 while classified as 5.

Classes 2 and 3 appear to confused with each other. This is due to class 2 being comp.sys.ibm.pc.hardware and class 3 comp.sys.mac.hardware. One would imagine the terminology is fairly similar and the two classes are quite close to each other. Additionally, classes 4 and 5 face the same issue, they both relate to sports and will have similar terminology which puts them closer to each other in terms of spatial distance.

1. b

The graph below shows the number of clusters from 2 to 8 as a function of sum of squared errors computed by weka.



Given the graph, one would select k = 8. This is a contradiction, however, as we only have 5 classes to cluster. A large number of clusters will result in smaller error rate as give more freedom to the clustering algorithm. Ultimately, when k = size of the dataset we get 0 error but gain no useful information. Using k = 5 seems reasonable as the error drops rapidly from k = 4 to k = 5 but decreases at a lower pace as we increase k.

The seed should be selected based on the data as it depends on the domain. Ideally, our seed would be the most frequent words in the dataset.

1. c

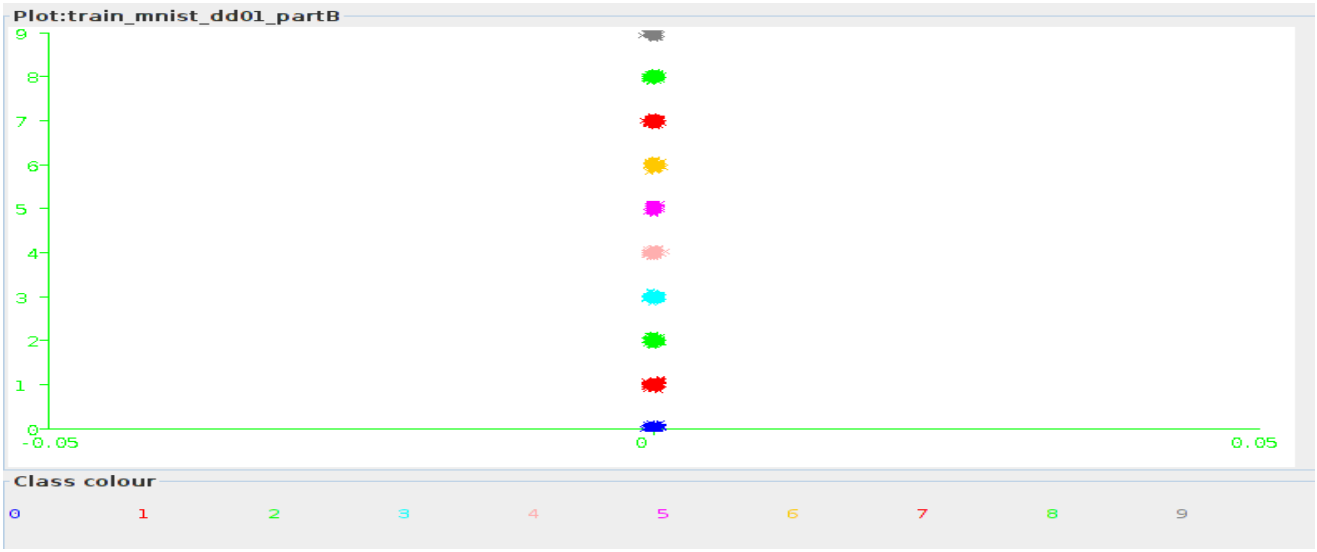
	Column 1	Column 2	Column 3	Column 4	Column 5
Class	*3:comp.sys.mac.hardware*	*2:comp.sys.ibm.pc.hardware*	*5:rec.sport.hockey*	*4:rec.sport.baseball*	*1:alt.atheism*
Reason	*Occurrence of mac and general computer keywords*	*Occurrence of mostly server related computer keywords such as motherboard, memory, speed*	*Occurrence of the words hockey, nhl, playoffs*	*Occurrence of baseball, hit, runs and pitching*	*Occurrence of god, atheism, religion, morality*

Columns 1 and 2 appear to be the closes by finding the minimum of the difference of the sum of probabilities in each column.

2. a

Classifier	Naive Bayes	SMO
PC	*71.15 %*	*88.5%*

Looking at the histograms for class against pixel data, we can observer there are many attributes that do not contribute with any data. Majority of the histograms look very similar to the one presented below. All classes have a value of 0 for this attribute and do not give any useful information.



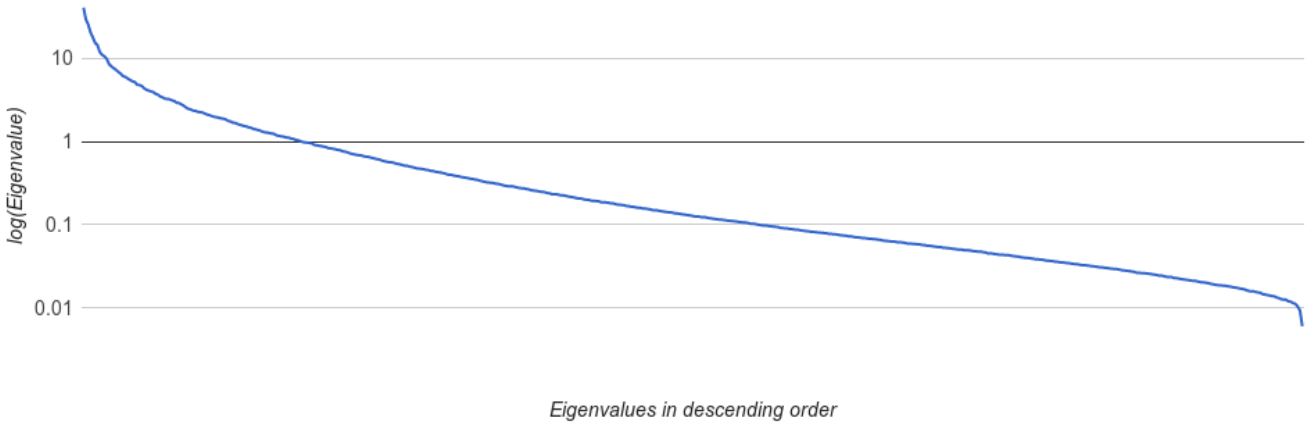
Useless attributes can be removed with RemoveUseless -M 99.0 function in the pre process tab. In total,145 attributes have been removed this way. We are left with only attributes that do not have equal values across all classes.

Retraining the models we obtain:

Classifier	Naive Bayes	SMO
PC	*71.15 %*	*88.5%*

We can observe that neither classifier has gained any increase in PC.

2. b



We can notice that considerable proportion of the attributes have eigen values below zero and negative. Eigenvalues below 0 *(log(1))* indicate that there is not a sufficient amount of variance for the attribute which results in little to no usefulness to the classifier.

2. c

Classifier	Naive Bayes	SMO
PC	*82.8 %*	*88.75%*

The reason NB classifier improves while SMO stays the same could be due to the way SMO picks the direction with the highest variance. The variance in the data has not changed through the *PrincipalComponent* function and SMO simply performs the same due to that. Naive bayes, on the other hand, is not based on variance and simply applies the base rule to determine the probability of each class. This results in improved performance for NB.

3. a

Dataset	**train_mnist_binary_pairConj_partC.arff, e = 1**	**train_mnist_binary_partC.arff, e = 2**
SMO PC	*90.2%*	*89.55%*

Looking at the PC for both datasets, there is not much difference.
Given a kernel function $K(x,y) = (x^T y + c)^d$, Yoav Goldberg and Michael Elhadad (2008) state that *"[w]hen using binary valued features [], this kernel function essentially implies that the classifier considers not only the explicitly specified features, but also all available sets of size d of features"*. This implies that raising the polynomial degree will implicitly give more freedom to the classifier to consider values around the value in question to allow for changes in nearby attributes.

The two methods are almost identical in this manner. Goldberg and Elhadad also suggest using d = 2 for the best results. Larger values of d give too much freedom to the model and overfit the data.

References:

- **Yoav Goldberg, Michael Elhadad: splitSVM: Fast, Space-Efficient, non-Heuristic, Polynomial Kernel Computation for NLP Applications**
  [PDF] *https://docs.google.com/viewer?url=http%3A%2F%2Fwww.aclweb.org%2Fanthology%2FP08-2060.pdf* [Visited Nov 19, 2013]