Name: Scott Hofman Student ID: s0943941 Exam Number:4394147 Date: 22/11/11

Introduction to Applied Machine Learning Coursework

Part A: Exploration of the Data

1) PC of Classifiers:
Naive Bayes – Percentage Correct:12%
Decision Tree (J48) – Percentage Correct: 37%
SVM (SMO) – Percentage Correct: 46%

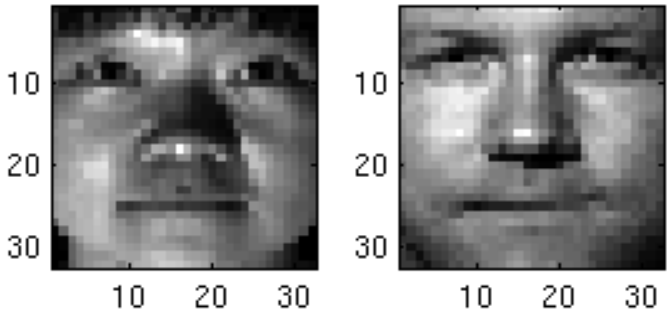
2) When using EM clustering, you should set the number of clusters to 10. This reflects the number of faces in the database – each face should have its own cluster. However, there are spikes within the cluster amounts – the values are not level across each cluster as we expect. I suspect this being a result of faulty/bad data within the dataset – the bad "faces" (typically black images) will be similar to other bad values and will be grouped as such. The high number of poor data sets (100 instances in both cluster 6 and cluster 2) will lead to low percentage calculations, as these blackened data sets contain no information to classify. The classifiers are forced to make a prediction from nothing, and will err more frequently.

3) To improve the data, and the resulting classifications, I have removed clusters 6 and 2. These had values not within a reasonable deviation of the other cluster labels (i.e. Cluster 6 had a value of 100, while clusters 4 and 5 have values of 18 and 19). The deviation between these values is too great for what should be level (aka equal number of classified faces per cluster) and often had very low or very high values across its pixel attributes. Therefore, I removed clusters 2 and 6 by applying a RemoveWithValues filter with attributeIndex set to last (selecting the clusters) and nominalIndices valued at 2,6 to remove anything classified as cluster 2 or cluster 6. This should remove the "bad values" (blackened or white images) and improve our PC upon reclassifying.

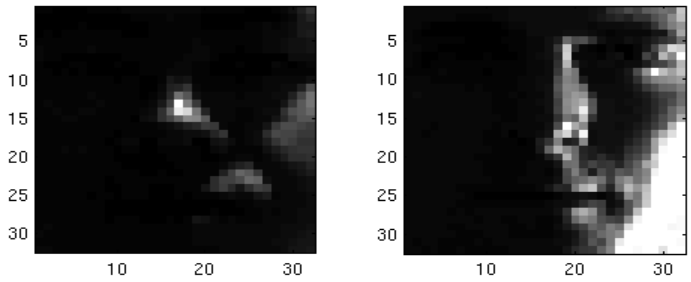
4) PC of Cleaned Data Classifiers:
Naive Bayes – Percentage Correct: 52.5 %
Decision Tree (J48) – Percentage Correct: 62%
SVM (SMO) – Percentage Correct: 88%

A simple baseline for these classifiers should be the previous trial without clean data – if the current percentages outperform question 1 of this section, then we know we have improved our results. If looking for a generalized baseline, we could assume that random guessing would correctly assign 10%. Since our percentages are better than both randomly guessing and the standard in question 1, we can safely assume that it outperforms the baseline.

5) After including the clean data and running the SMO, we achieve 92% on the validation set. We examine two correctly classified examples: image 6 and image 51:



These faces are well lit, with most features clearly visible. Their features are distinct enough to be indicative of the face. Contrast this with:



Both images 14 and 28 are misclassified, and it is not difficult to see why. It is nearly impossible for image 14 (left) to be classified by human intuition – nothing can be used to identify the face (such as chin or eye shape). Image 28 (right) is easier to understand for a human, but still lacks a half of a nose, a left eye, most of a mouth and is devoid of a chin, due to either shadows or white patches. The machine, dealing with numeric values, cannot "fill-in" the rest of the face as we do visually. These image irregularities (typically shadows) are responsible for misclassification. We have seen similar results from the darkened images removed in question 3 of this section – the dark images and shadowed images are treated by the classifier in a similar manner. That shadows cause difficulties therefore comes as no surprise.

Part B: Feature Selection & Engineering

1) Naive Bayes models its data on a Gaussian distribution. This is the standard assumption for Naive Bayes, but in this case, it is a poor assumption. Pixel intensities do not correlate to a Gaussian model, so to assume this distribution does not accurately reflect the probability distribution.

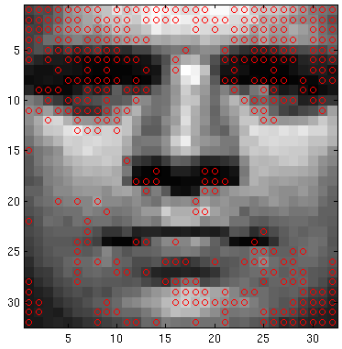
PC of Binning with Naive Bayes:

- 5 Bins – 45%
- 10 Bins – 57%
- 20 Bins – 61.5%
- 40 Bins – 62%
- 60 Bins – 62.5%
- 80 Bins – 60.5%
- 100 Bins – 59%

When choosing the number of bins, we have to consider the amount of data we are trying to bin. We need the bins to represent meaningful partitions of data. If the number of bins is too low, then too much data is included in each bin, and Naive Bayes will generalize its assumptions. If there are too many bins, then they will not have enough information for Naive Bayes to pick the right classification. Therefore, we must try to get a value between these extremes. In our example, the optimal value is 60, as it gave the best percentage of 62.5%. This increase of 10% from the previous Naive Bayes is the result of binning providing an accurate representation of a large amount of data. It overrides the Gaussian assumption, and uses the probability of the bins to adapt to the data.

2) The InfoGainAttributeEval attribute evaluator compares the information gain between two attributes – that is, the higher the information gain is, the more valuable the attribute is. In layman's terms, the information gain measures the dissimilarity between two values (attributes in this case). The higher the information gain, the more dissimilar the values are, so we can 'gain' more information from comparing these attributes. By using this evaluator, we can identify what are highest differing attributes between faces, which can be used as unique identifiers.

3) To the left is an example of draw_pixels applied to an image. The red pixels represent the values left over after applying the InfoGainAttributeEval and filtering all values having Information Gain of 0 (values from 353-1024 are filtered in my example). The locations of these red pixels make sense – we expect the areas around the eyes, nose, chin and mouth to be highly indicative of a face.



The problem with using Information Gain arises when we select multiple features at once. We expect the area surrounding a nose to be relatively unique to a person. However, when selecting a group of features, the areas around the nose will return an information gain of zero – despite our assumption that the nose should be useful to identify a face.

4) Performance on train_faces_clean_best.arff:
Naive Bayes – 60%
Decision Tree (J48) – 63%
SVM (SMO) – 91%

The PC for each of the classifiers increases from the values used in train_faces_clean.

For Naive Bayes, the improved percentage comes from the independence assumption – eliminating pixels does not affect the other pixels. Because there are less pixels per face and the remaining ones are more independent, the assumption's validity improves, correspondingly increasing the percentage.

For Decision Tree, information gain does not matter. Decision trees (J48 in this example) use information gain as a method of selecting the next node. Removing the values with a low information gain will have little effective on the percentage because the previous percentage already depended on information gain.

For SVM, the remaining values are highly indicative of the important values of the face. Because we are already dealing with clusters (Part A), these points will tend around a certain area more strongly. If we remove the weak points (outliers), the SVM will have a greater likelihood of finding an effective decision boundary which includes all the points for a class. However, since SVM is usually quite good with outliers, the percentage is minimal.

Part C: More Advanced Feature Selection

1) There are a few high valued eigenvectors, decreasing rapidly to the low value of .00000000000000115. The majority of the values contain this value, so low it can be considered negligible.

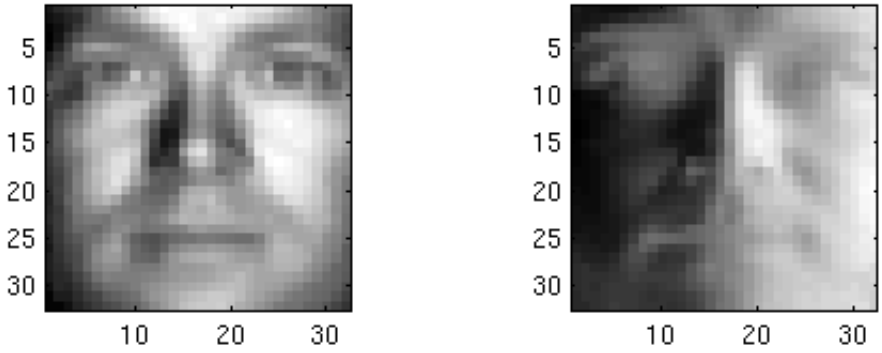
2) Applying the AttributeSelection filter leaves 32 eigenvectors. Upon classification, the performance with PCA is:

Naive Bayes: 81.5%
Decision Tree (J48): 61.5%
SVM (SMO): 82.5%

Both Naive Bayes and the SVM have increases in their PCs, while the Decision Tree suffers. This is a result of reducing the large data set of 1024 to 32. A lot of information is lost. Naive Bayes assumes independence. Since each eigenvector is independent of the others, Naive Bayes performs admirably. Some of the information the Decision Tree needs for accurate classification has been pruned away, so its performance decreases. The SVM does not increase much – the remaining eigenvectors are representative of average faces, so the clusters lack a strong enough decision

boundary to be classified better than previously.

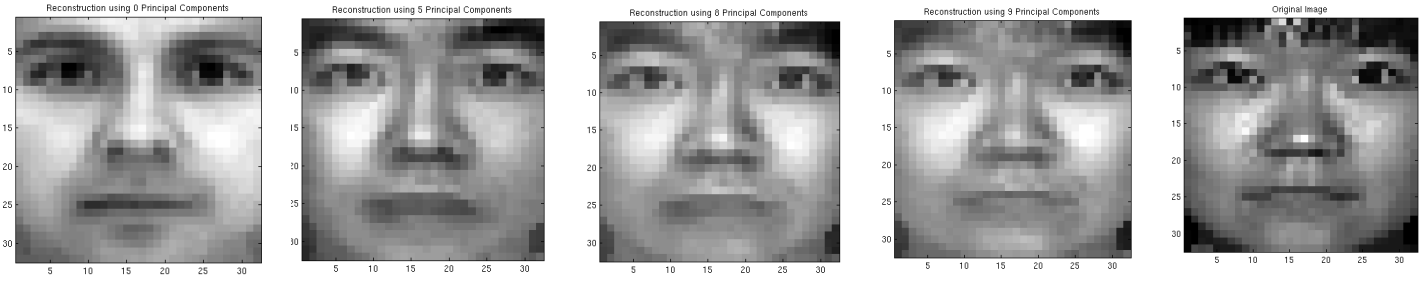
3)



Left image – Top Eigenvector. Corresponding eigenvalue: 390.78591
Right image – Second Eigenvector. Corresponding eigenvalue: 323.30464

These eigenfaces are representative of the average faces, with the eigenvector's weight providing the degree to which our instance looks like that face. The top rated eigenvector appears as a standard, if somewhat bland, face. The second rated eigenvector loses more of the distinctive features, but differs from the first in nose shape and mouth width. We assume that the trend continues, and each eigenface will have a blander, but slightly diverse representation of a face. To this extent, we should be able to classify each of our faces by using a combination of the eigenvectors, with weights representing how much of these eigenface is applicable to our face.

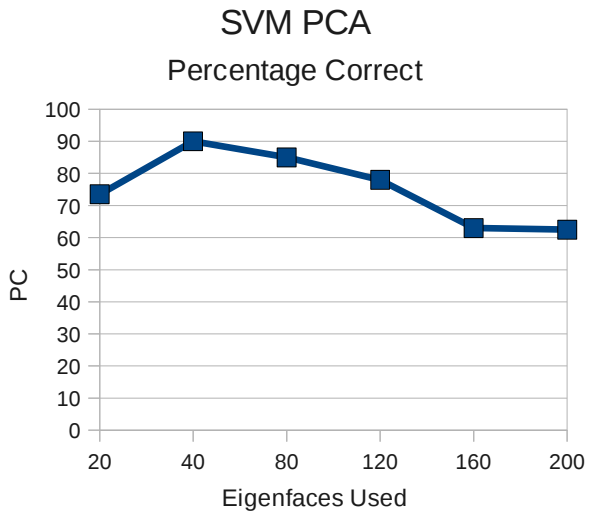
4) As stated above, any face can be composed from a set of eigenfaces with corresponding weights. Moving from left to right, we start with the first eigenface (the same as in question 3), but this time modified by a weight by how much it "looks" like the original face (far right). The second face has taken the next five eigenfaces with their respective weights and combined them to form a face with higher similarity to the original. This refinement process continues, but the degree of change decreases sharply (representing the rapid decrease in eigenvalues found in question 1 – these eigenvectors represent the subtle variations in a face). While 8 principal components (third image) looks similar to the final image, 9 principal components (fourth image) improves the nose, eye shape, and the thinness of the mouth to the point where the person becomes recognizable. Anything beyond 9 will gradually add finer details to the image, until it is undistinguishable from the original.



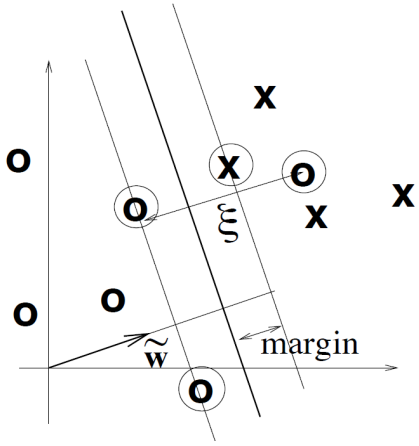
5) Naive Bayes, upon removal of the first four eigenfaces, still performs well with a PC of 82.5%. The reason that the percentage accuracy does not disappear lies with the independence assumption of Naive Bayes. Despite the four most important values being removed, the attributes are considered independent, so these values don't have as great an impact as they would on other classifiers (such as the tree, which drops to 51.5%). Therefore, I disagree.

6) SVM PCA

20: 73.5%
40: 90%
80: 85%
120: 78%
160: 63%
200: 62.5%



Initially, increasing the amount of eigenfaces used improves the classification percentage. As we add more eigenfaces, the SVM is able to refine its prediction based on the new information. However, sometime between using 40 and 80 eigenfaces, the value begins to diminish. The reason for this is because when the number of eigenvalues increases, the number of points that need classification also increases. As more points are added, it becomes more likely that not all points can be accurately classified – some will located in another group (the 'o' point located amidst those classified as 'x's is an example of such a misclassification).



1

Part D: Choosing Representation And Classifier

1) The output from the Experimenter, where the first number is the PC over five trials, and the second number is the standard deviation.

Dataset	(1) bayes.NaiveBayes	(2) trees.J48	(3) functions.SMO
train_clean	(25) 53.60(5.95)	61.90(7.19)	88.00(3.15) v
train_clean_best	(25) 63.60(6.04)	64.30(7.02)	90.10(3.19) v
train_clean_pca	(25) 79.90(5.80)	60.10(8.49) *	80.90(4.89)
	(v/ /*)	(0/2/1)	(2/1/0)

With a significance of .05, the above table indicates that SVMs outperform Naive Bayes on both the datasets 'train-clean' and 'train-best'.

Dataset	(1) train_clean	(2) train_clean_best	(3) train_clean_pca

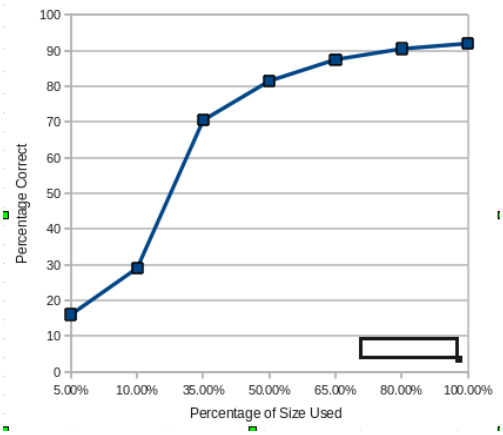
1Adapted from lecture slides by Charles Sutton and Victor Lavrenko

bayes.NaiveBayes	(25)	53.60 (5.95)	63.60 (6.04) v	79.90(5.80) v
trees.J48	(25)	61.90 (7.19)	64.30 (7.02)	60.10(8.49)
functions.SMO	(25)	88.00 (3.15)	90.10 (3.19)	80.90(4.89)
	(v/ /*)		(1/2/0)	(1/2/0)

With a significance of .01, the above table indicates that 'train_clean_pca' and 'train_best' outperform Naive Bayes, because of the 'v' beside their PC in the Naive Bayes row. The J48 tree lacks any v and *, so its performance is constant with a confidence of 99%.

I would select train_clean_best because it outperforms the other data sets on both the decision tree and the SMO, and comes in second for Naive Bayes. This is consistently better than the other datasets.

2)
Using seed 30:
5% of data used- 16%
10% of data used – 29%
35% of data used – 70.5%
50% of data used – 81.5%
65% of data used – 87.5%
80% of data used – 90.5%
100% of data used – 92%



To a certain extent, the performance would improve upon adding data points. This would peak after a set value, when the amount of data becomes too specific to 'learn'. It overfits by adjusting to include all this information, and might have difficulties classifying correctly once applied to a new data set.

3)
Performance with Filtered Classifier: 83.5%
Performance with Nested Filtered Classifier: 86.5%

Nesting the classifiers increases the PC by a small margin. I disagree with the student's assumption, because, while Naive Bayes treats the features as independent, clustering adds a dependency amongst them. When the cluster feature is used in the Naive Bayes calculation, the PC increases because this value represents dependency. It only increases because we know these features are not independent in real life. Adding a cluster feature in the calculation reduces the naivety of the Naive Bayes assumption.

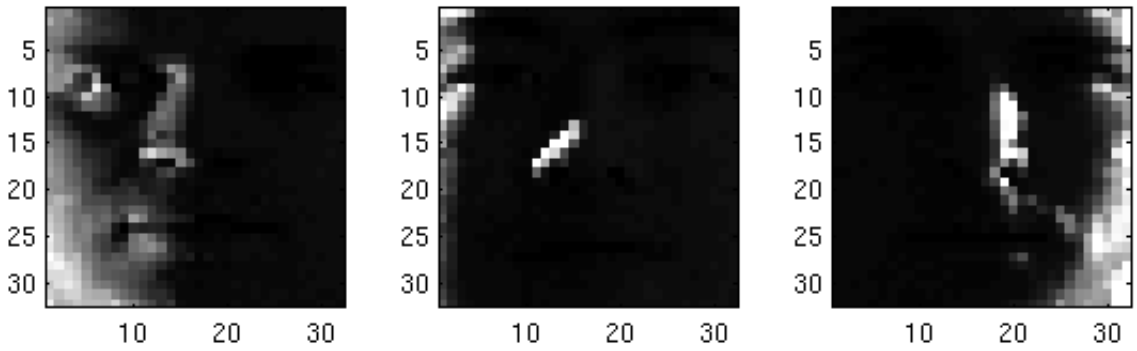
Part E: Mini-Challenge

1) Picking the Classifier

Throughout this coursework, SVM (SMO) has consistently had the best PC in comparison with the other classifiers. The highest value achieved was 92% on two different occasions (Part D, Question 2 and Part A, Question 5). The strongest prediction should be improve on this value. To improve such a percentage, we need to correct the issues that cause a misclassification. From Part A, we know shadows/darkened images are typically responsible for failed image classification. A standard method in image manipulation to eliminate the effects of shadows is to normalize the image. By using the meta function 'FilteredClassifier', the images can be normalized before they

are classified. This is done by using the “normalize” function within the unsupervised instance filters, and using the SMO function as the classifier. All values remain constant.

This resulted in PC of 98.5% when using the train_faces_clean.arff file, and the val_faces.arff. Below, I have included the faces that have been misclassified. Even after normalization, the last two do not contain enough information to make a correct prediction. The first image, however, should contain enough information to accurately classify. However, despite changing the values within the SMO, I could not improve the accuracy.



When testing the train_faces_clean_best.arff on validiation set val_faces_best.arff, the resulting PC was 95.5%. Because of the lower percentage, and that the best set has thrown away data (not a sensible practice for machine learning), I decided to use the train_faces_clean as my training data.

People Included in Discussions:

Jason Ebbin
David Eichman
Craig Innes
James McLaughlin