IAML Assignment
Using Weka 3.6.2

Chris Swetenham (s1149322)

November 22, 2011

This assignment was discussed with Dan Mankowitz, Dino Vougioukas, Kostas Kanellis, Facundo Bellosi, Carlos García.

Part A

Question i

Training on the dataset `train_faces.arff` and evaluating with 5-fold cross-validation, we evaluate three classifiers using their default settings:

Method	Accuracy
`NaiveBayes`	12%
`J48`	37%
`SMO`	46%

Question ii

We use the `AddCluster` unsupervised attribute filter, using the EM algorithm, to add cluster labels to each training instance. We set the number of clusters to 10, corresponding to the 10 subjects in the dataset. In the histogram for the resulting cluster attribute, shown in Figure 1, we notice that there are two dominating clusters which with exactly 100 instances each. We were hoping to see 10 roughly equal clusters corresponding to the 10 subjects in the dataset. Additionally, labelling them by class, we see that there is very little correspondance between the two class and cluster attributes.

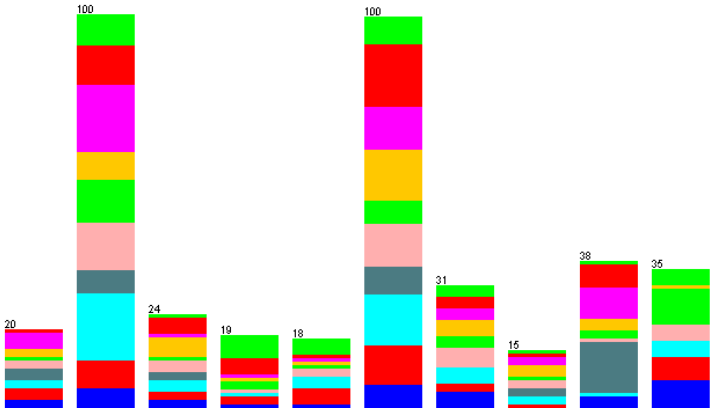

Figure 1: Cluster Histogram

Checking the histogram of the class labels, shown in Figure 2, we notice that these are unevenly distributed, and further we have 400 instances rather than the 20*10 we were promised. The two dominating classes together sum to 200, the discrepancy between the expected and actual size of the dataset.

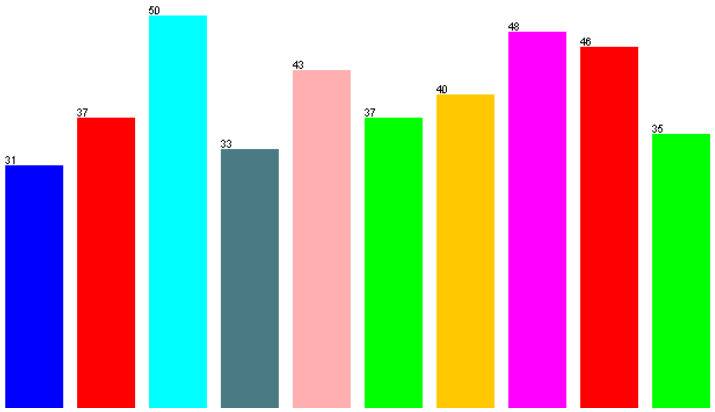

Figure 2: Class Histogram

The dominating clusters are *cluster2* and *cluster6*. The first three images from each cluster are shown in Figure 3 and Figure 4.

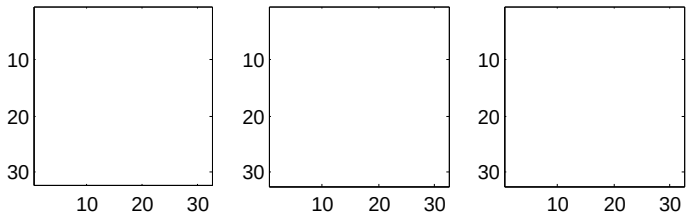

Figure 3: cluster2, images 6, 12 and 20

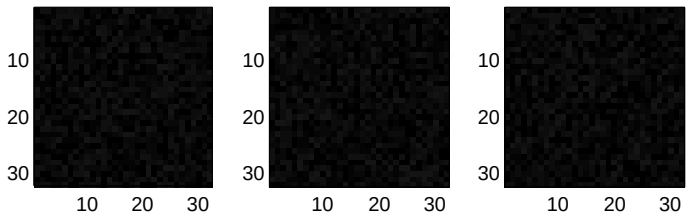

Figure 4: cluster6, images 2, 15 and 29

The first three images from *cluster1* and *cluster3* are shown in Figure 5 and Figure 6:

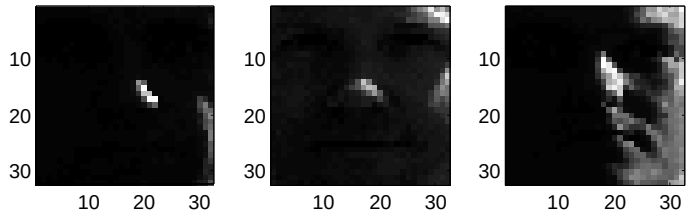

Figure 5: cluster1, images 1, 7 and 8

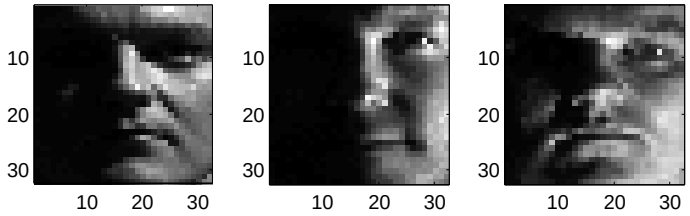

Figure 6: cluster3, images 13, 17 and 19

We see that the two dominating clusters seem to be pure white and noisy dark images which are not relevant to our task.

Question iii

Using the filter `RemoveWithAttribute`, we removed instances in *cluster2* and *cluster6*. As shown in Figure 7, the resulting dataset has 20 instances of each class, as expected from the task description.

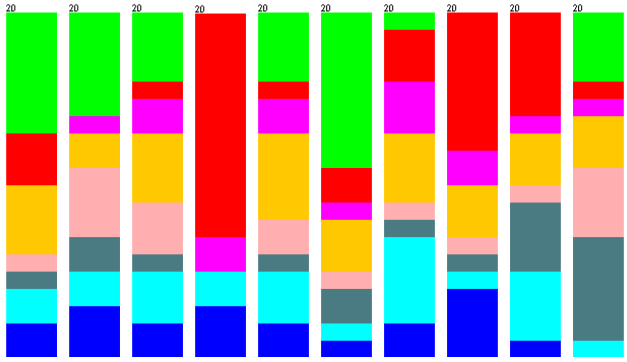

Figure 7: Class Histogram after `RemoveWithAttribute`

Question iv

We remove the cluster attribute from the instances in the dataset, and as in part i) evaluate different methods with 5-fold CV:

Method	Accuracy	Baseline
`NaiveBayes`	52.5%	12%
`J48`	62%	37%
`SMO`	88%	46%

We can compare this performance with the results from part i) as baseline. After removing the erroneous instances, the accuracy of all three classifiers has greatly improved. Since the dataset was smaller, the time taken to train and evaluate the classfiers was also shorter.

Question v

We train the `SMO` classifier from `train_faces_clean.arff`, using `val_faces.arff` as test set.

Method	Accuracy
`SMO`	92%

This is comparable to the performance on the training data earlier. Two example correctly classified instances are shown in Figure 8.

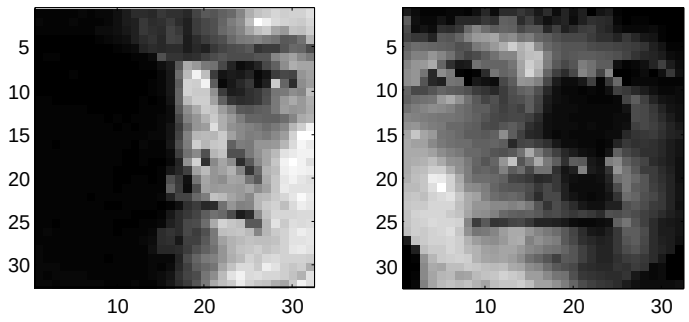

Figure 8: Correctly classified instances 32, 85

Two example incorrectly classified instances are shown in Figure 9. The classifier seems to make mistakes on instances where the subject's eyes are entirely in shadow.

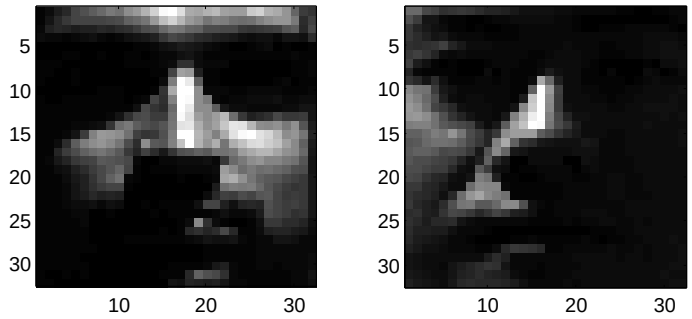

Figure 9: Incorrectly classified instances 15, 42

Part B

Question i

In the Naive Bayes model, each of the attributes are independently distributed given the class label. This is not a very good assumption for our data set, since nearby pixels will usually have similar values, and lighting conditions affect pixels in similar ways across large regions of the image.

For continuous attributes, `NaiveBayes` will build a gaussian distribution of the values for each pixel for each subject's images. This is a poor choice, because the many lighting conditions mean that the variation of any pixel within one class is much greater than the variation of that pixel between classes.

Discretization allows for more arbitrary distributions to be learned, for example we might have a bimodal distribution for a pixel with one mode when that pixel is in shadow and one when it is well lit. Choosing the number of bins is a parameterization problem; the more bins we allow, the closer fit to the data we allow, but the more parameters we have to learn. On one hand, we want to choose a small enough number of bins that we have some bins with multiple instances in them. On the other hand, too few bins will give us very little discriminative power.

After applying the unsupervised attribute filter `Discretize` using different numbers of bins, we train `NaiveBayes` and evaluate using 5-fold CV.

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

Out of the settings tested, using 60 bins for discretization gives the best performance. We see a moderate gain in performance over the original NaiveBayes result (62.5% vs 52.5%); this is about what we would expect, since we improved on the original gaussian model, but `NaiveBayes` remains less than ideal for the

task at hand since its assumptions about the structure of the data do not hold well for our dataset.

Question ii

`InfoGainAttributeEval` works by computing the information gain of each attribute, which is the difference between the entropy of the marginal class distribution and the entropy of the class distribution conditioned on that attribute. The attribute with the highest information gain tells us the most about the class, and one with 0 information gain is independent of the class.

After removing the attributes with 0 information gain, 353 attributes were left - we note that the provided `train_faces_clean_best.arff` has 350 attributes.

Question iii

As shown in Figure 10, the selected pixels with highest information gain cluster around the areas of the eyes, the corner of the chin, and the mouth and nose, ignoring the cheeks and the nose bridge. This corresponds with the most distinguishing features of a human face.

A potential problem with the information gain criterion is that if we use it to select more than one feature at once, these features might be correlated, and so the second we select would tell us a lot by itself but very little if we already know the first.

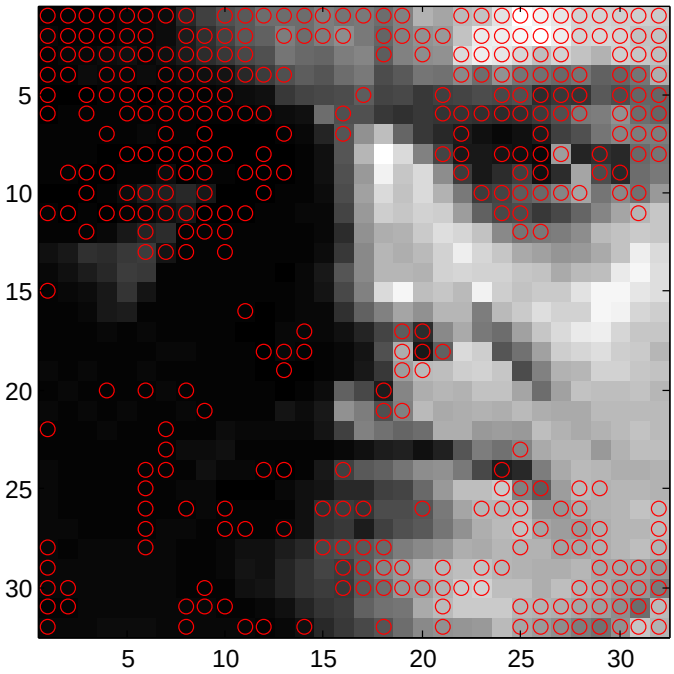

Figure 10: Pixels selected by `InfoGainAttributeVal`

Question iv

We evaluate the same three classifiers as before on the new reduced dataset, again evaluating using 5-fold CV. We take the results from A iv) as baseline for comparaison.

Method	Accuracy	Baseline
`NaiveBayes`	60%	52.5%
`J48`	63.5%	62%
`SMO`	91%	88%

We argued in part i) that the Naive Bayes algorithm has trouble with the original dataset due to the independence assumption being a poor fit to the data. Keeping only the attributes with the highest information gain should have removed a lot of attributes which were poor predictors and strongly correlated with each other, giving us better performance as observed. The decision tree algorithm already works by selecting the attributes with the highest information gain, so as expected we see very little difference.

The SVM algorithm performs about the same as before. SVM should not be affected by additional attributes which are poor predictors.

Part C

Question i

The eigenvalues in descending order are plotted in Figure 11. As we can see there is a sharp fall-off after the first few eigenvalues which contribute most of the variation in the dataset.

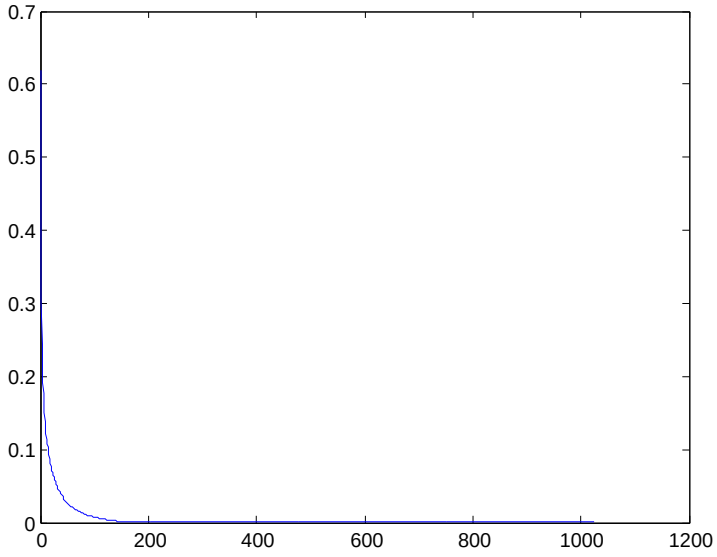

Figure 11: Eigenvalues

Question ii

After using the `PrincipalComponents` attribute filter to retain 95% of the variation in the dataset, we are left with only 32 attributes.

We evaluate the same three classifiers as before on the new reduced dataset, again evaluating using 5-fold CV. We take the results from A iv) as baseline for comparaison.

Method	Accuracy	Baseline
`NaiveBayes`	81.5%	52.5%
`J48`	61.5%	62%
`SMO`	82.5%	88%

PCA gives us components which are uncorrelated and so suit `NaiveBayes` well, and this is reflected in the improved performance of Naive Bayes on this version of the dataset. The decision tree algorithm tries to find binary splits in the values of each attribute to classify instances. It is not affected by correlated attributes, since after it splits on one of them the inormation gain of the other would be small; so we would not expect an improvement. It would still be possible for the attributes we have thrown away to contain a lot of the information; the fact that we did not get worse performance suggests that most of the information is in fact contained in the principal components.

Since PCA performs a rotation of the attribute space and SVM is a linear classifier, we would expect `SMO` to perform as well on this dataset as on the original; if we perform the above evaluation but retain 100% of the variation, we get an accuracy of 87.5% using `SMO` on the processed dataset. Therefore the attributes which had low variance and were discarded must have been useful for linear classification.

Question iii

The eigenfaces corresponding to the largest eigenvalues capture gross detail, with successively finer features as one proceeds to smaller eigenvalues.

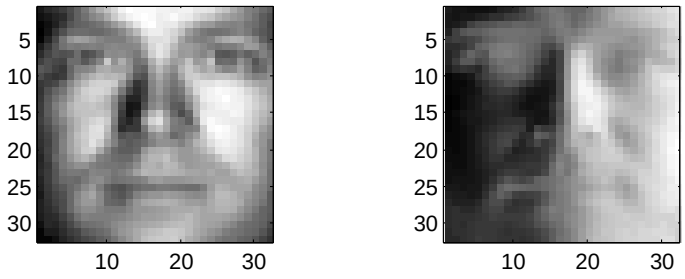

Figure 12: The first two eigenfaces

The largest two eigenvalues are 390.8 and 323.3. The corresponding eigenfaces are show in Figure 12. These eigenfaces are an average face, corresponding to the variation in level of global illumination in the dataset, and a face which looks lit from the side, corresponding to the variation in left-right illumination

in the images. Only when we get some number of eigenfaces in do we start seeing ones which distinguish between facial features as opposed to illumination; this makes sense since in the dataset the difference between images with different illumination directions is much more dramatic than the difference between two faces under the same illumination.

Question iv

Around 10 components are required to produce a face which is clearly recognisable as the original. As we increase the number of components, the face moves away from the average of the underlying faces and gains specific details from the face we are reconstructing.

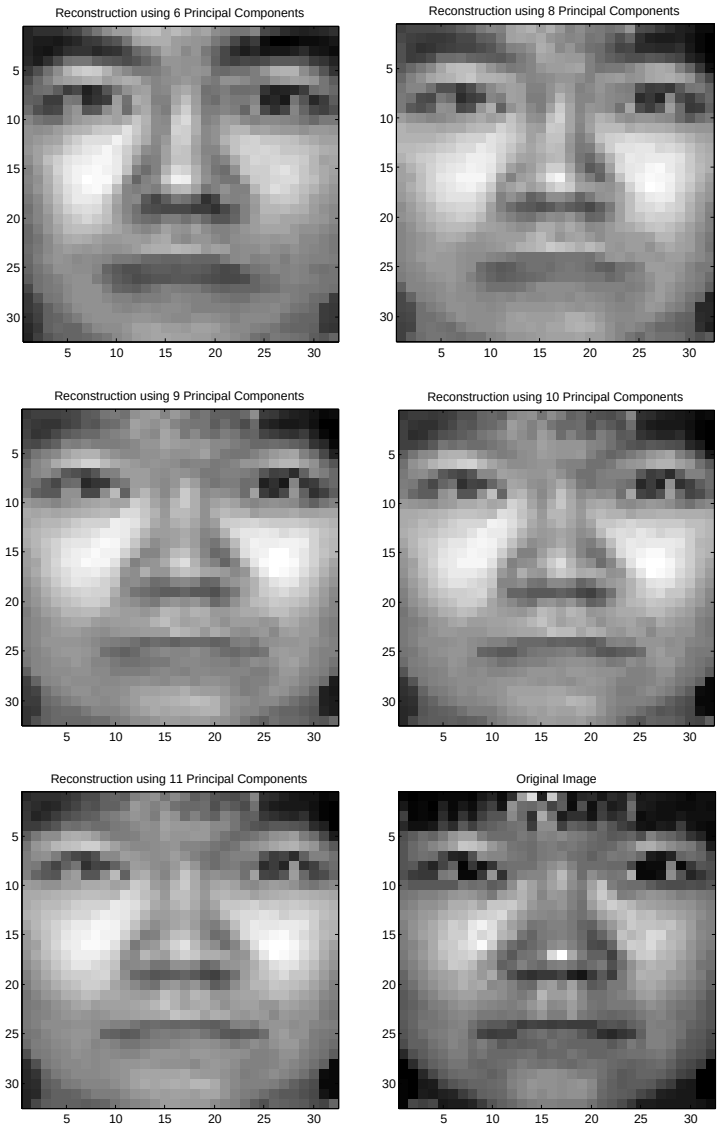

Figure 13: Reconstructing a face

Question v

We remove the four attributes with the largest eigenvalues, and evaluate the `NaiveBayes` classifier using 5-fold CV, using the results from C ii) as baseline.

Method	Accuracy	Baseline
`NaiveBayes`	82.5%	81.5%

The accuracy has increased slightly. As stated before the greatest part of the variation in the dataset is in the lighting, which is not significant to our classification task; so the attributes we have left after removing the four with largest variation serve well for classifying the data.

Question vi

We try running `SMO` retaining different numbers of the top eigenfaces, evaluating the results using 5-fold CV. We include results obtained in part ii).

Eigenfaces	Accuracy
20	73.5%
32	82.5%
40	90%
80	85%
120	78%
160	63%
200	62.5%
1024	87.5%

We notice that there is a peak in performance around 40 attributes, after which performance falls off; yet we know it climbs back up somewhere between 200 and 1024 attributes. This is difficult to explain. The peak at 40 attributes does not correspond to the 10 attributes after which a face becomes distinguishable by a human. As successively less important eigenfaces are kept, the level of detail we can distinguish increases, but the noise does as well.

Part D

Question i

Results at 99% confidence (95% is identical), by dataset:

Dataset	`NaiveBayes`	`J48`		`SMO`	
`clean`	53.60	61.90		88.00	v
`clean_best`	63.60	64.30		90.10	v
`clean_pca`	79.90	60.10	*	80.90	
	(v/ /*)	(0/2/1)		(2/1/0)	

We see that for two datasets, `train_faces_clean` and `train_faces_clean_best`, `SMO` does significantly better than `NaiveBayes` with 99% confidence.

Results at 99% confidence, by algorithm:

Dataset	clean	clean_best		clean_pca	
NaiveBayes	53.60	63.60	v	79.90	v
J48	61.90	64.30		60.10	
SMO	88.00	90.10		80.90	
	(v/ /*)	(1/2/0)		(1/2/0)	

We see that for NaiveBayes, train_faces_clean_best and train_faces_clean_pca are both significantly better than train_faces_clean with 99% confidence. For J48, none of the datasets are significantly better than the others.

Based on this, I would select train_faces_clean_best for further experiments.

Question ii

We use the Randomize instance filter to shuffle the dataset, using the default seed of 42. We then produce nested datasets by using the RemovePercentage instance filter, setting invertSelection true, with percentage values of 100 to 5. We evaluate the SMO classifier on the resulting datasets, evaluating using val_faces rather than cross-validation.

Percentage	Instances	Accuracy
100%	200	92%
80%	160	87%
65%	130	86.5%
50%	100	84%
35%	70	76.5%
10%	20	40%
5%	10	21.5%

As we can see from the plot above, the performance is tailing off somewhat as we add more data, but the last 20% of the dataset does provide a reasonable improvement and so a significantly larger dataset could still improve the performance of the classifier.

Question iii

We use the FilteredClassifer to allow us to use a validation set while performing filtering on the data. We train using train_faces_clean and evaluate using val_faces. In one trial we apply PCA to the dataset and evaluate using NaiveBayes. In the second trial we apply cluster lbels using the EM algorithm with the default seed of 100 and using 10 clusters, then apply PCA.

Filtering	Accuracy
PCA	83.5%
EM and PCA	86.5%

Adding cluster labels has slightly helped. Although they don't contain any data independent of the original dataset, NaiveBayes assumes all attributes are independent given the class labels, and so doesn't make use of all the information in the original dataset; by adding new features which are distinct from any of the original ones and which are useful in themselves for classification, we can improve the performance of the classifier.

Part E

Using both the `train_faces_clean` and `train_faces_clean_best`, we systematically explored combinations of the techniques and classifiers used in previous parts, trying to improve on the 92.5% rate obtained with `SMO` on the `best` dataset. The only success was found when applying PCA retaining 100% of variation, ranking the resulting attributes by information gain, removing all those with 0 information gain, and adding a cluster label using EM with 10 clusters before classifying with `SMO`; this gave 93% accuracy. Unfortunately applying clustering after PCA gives an error in WEKA, even when telling it to ignore the class label using the ignoredAttributes field, so this technique could not be applied to the test set. Finally, we tried different classifiers with cool-sounding names until we hit upon `MultilayerPerceptron` which gives 96.5% on `val_faces_best`. These results and the test set predictions are included in the file `iaml_assignment.res` submitted alongside this report.