CSC411: Assignment 3

# 1. Introduction

The objective of this assignment is to test out and apply the different machine learning methods learned throughout the semester. There are 2925 labeled images and these are used for both training as well as cross-validation. The labels for the 2925 images are of faces with facial expressions from seven categories. The seven categories include: Anger, Disgust, Fear, Happy, Sad, Surprise and Neutral, labeled with integers between 1 to 7 respectively. The goal is to train a model that is the most capable at predicting labels to any of the public test images with high accuracy.

The high level approach to this assignment includes testing out each of the methods such as SVM, Mixture of Gaussians and KNN separately at first. Once each of these methods have been tuned correctly to maximize the accuracy of each method, an attempt to use an ensemble method composing of all of the proven methods will be tested and used. Throughout the testing and tuning of each of these methods, 10-fold cross-validation will be used to check the performance of each of these methods.

# 2. Description

After testing each of the methods, the method that provide the highest average accuracy across 10-fold validation is Support Vector Machines (**SVM**). To implement SVMs in MATLAB, we used **fitcecoc**, the multiclass SVM classifier function built into MATLAB.

For the parameters, we used:

* **one-vs-one** design to decompose the multiclass classification into binary ones
* **Default** kernels was used for the classification
* The images vectors were preprocessed using **Standardize** in templateSVM

The effect of changing the parameters is included under the Empirical Results section. Including this section there better demonstrates that the modelled trained is sound.

For training, we used the entire labelled images set given to us. We did not use the unlabeled images as it would require unsupervised learning and a very successful model. As our model barely peaks at 71%, there will high confirmation bias when training with no labels.

# 3. Empirical Results

This section compares the best method as described in Section 2 to the other methods and techniques used in attempting to create the most accurate prediction possible.

## 3.1 Design

We tested changing the coding design, the method by which the multiclass problem is reduced into binary problem. The result is summarized below.

|  |  |  |
| --- | --- | --- |
| **Design** | **# of Learners** | **Performance** |
| one-vs-all | K | 66.7% |
| one-vs-one | K(K – 1)/2 | 71% |
| binary complete | 2K – 1 – 1 | 67.2% |

The other designs were not tested as the number of binary learners created were either significantly lower than one-vs-all, or significantly higher than binary-complete. This means they are unlikely to do better than one-vs-all or binary-complete when both failed to beat one-vs-one

## 3.2 Kernels

We tested different kernel functions, which transforms the inputs into kernel space for easier classification. We set *KernelFunction* to ‘polynomial’ and tested the following *PolynomialOrder* values.

|  |  |
| --- | --- |
| **Order** | **Performance** |
| 1 | 71% |
| 3 | 65% |
| 5 | 47% |

As it can be seen, linear or 1st order polynomial gave the best result. Order 2 was also tested, but failed to converge.

## 3.3 Mixture of Gaussians

The first thing we tried was the Mixture of Gaussians, as the means and variances produced by the gaussians give us an intuitive result of how the training is progressing.

The basis behind this method is to predict the generative functions, assumed to be gaussians, which generated the data. The training is done by using EM to iteratively move the gaussian to match the input data. The gaussians returned from the training can then be used to estimate the likelihood of new sample belonging to the trained class of the input. By repeating this for each class, the probabilities of the new sample being from each class can be found. The highest probability is assumed to the correct class.

This method was implemented using code from Assignment 2, generalized to seven classes instead of binary classes. The major change is finding all the training images which belong to each emotion/class so the seven models can be trained separately.

The hyper-parameter K, the number of gaussians to use, is a significant factor in the performance of the method. To pick this hyper-parameter, we used cross-validation to test different Ks. At first, we started with very spread out values, going in multiples of eight, as testing every single value was not very feasible. After each cross-validation we took the half portion which the best K resided in and interpolated to find new values in between the old values. This gives a new set with half the range, but the same number of values. This was repeated until every value was consecutive to the next. The best value found was **32** gaussians.

After training the 32 gaussian on the entire training set, the prediction for the test set was around 60% accurate. As the MoG was initialized randomly, the performance will have some variance. We found that overall, not using the processing from the baseline improved the performance by 1-2%.

## 3.4 KNN

The baseline provided to us was a k-Nearest-Neighbor. KNN was likely picked as it is deterministic; showing no variation in performance as no training or random processes is used.

The code provided came with a 10-fold cross-validation function which, when executed, iterates numerous values between 3 and 30, for the hyper-parameter K. The result of cross-validation was that 5 showed the best performance, obtaining 57% accuracy for the validation set.

The result of training the 5-NN on the entirety of the training set was 58.612% classification of the online test set. This is the baseline which acts as the minimum.

## 3.5 Ensemble Method - Popular Vote

The ensemble method tested included a combination of SVM, Mixture of Gaussians and KNN. Our ensemble method took a weighted popular vote from the three methods for the best result. The best result that we got amongst the three methods was SVM followed by Mixture of Gaussians which is followed by KNN.

The accuracy for Mixture of Gaussians and KNN was around the same during 10-fold cross-validation, around 60%. However, the accuracy of SVM during 10-fold cross-validation was slightly under 72%, and thus SVM has a cross-validation accuracy that is approximately 10% higher than both Mixture of Gaussians and KNN.

For the implementation of popular vote, ensemble method, the prediction would be based on the results of the three different methods. Based on the results from cross-validation, a good approach for the ensemble method would be to select the prediction that is agreed upon by the majority of methods (2 methods). However, if none of the methods agree with each other, then the prediction from SVM would be selected over the others.

The theory behind the implementation of this approach is that there are features that cannot be seen/predicted by SVM, but may be seen/predicted with KNN or Mixture of Gaussians. Thus, this can be seen as an approach to prevent overfitting and consequently it is a method of regularization.

During the testing of this ensemble method via 10-fold cross-validation, the accuracy yielded from numerous 10-fold cross-validation runs was worse than the accuracy of the plain SVM method tested earlier. The accuracy for this ensemble method was around 68% which is approximately 3% worse than just SVM. This lower accuracy was later confirmed when this method was just on the public test set with the results uploaded to Kaggle. The accuracy on the public set was around the same as that from 10-fold cross-validation.

A hypothesis behind the drop in accuracy using the ensemble method over using just SVM is probably related to the inaccuracies of both KNN and Mixture of Gaussians. It appears that in certain test cases, both the KNN and the Mixture of Gaussians appears to predict that the face is neutral, while the prediction from SVM states the expression on the face is something else. Going through these images with the human eye was able to help determine that these faces are not neutral. This is an indication that there are certain features that both KNN and Mixture of Gaussians cannot detect and they indicate images with these features as neutral. Both of these happen to provide the same incorrect prediction and thus the ensemble method would take this majority vote with the incorrect prediction as the final prediction for this image. As a result, the final prediction for certain images would be incorrect.

The inaccuracies and features not detected by the Mixture of Gaussians method and the KNN method lead to a drop in performance for the ensemble method compared to SVM. A future improvement for the ensemble method may be to pair up another method that is capable at achieving more than 70% on 10-fold cross-validation and use that along with SVM. If the prediction of these two methods do not agree, then the tie can be broken with either the current KNN method or the current Mixture of Gaussians method.

## 3.6 Pre-processing Images

Pre-processing images was a technique that was used during the testing of both the Mixture of Gaussians and SVM. The inspiration to use this technique was from the baseline KNN starter code provided. The mean of each image was subtracted and the variance was divided. The purpose of this is to attempt to reduce the amount of variation amongst the similar features and lighting in the images as much as possible.

When 10-fold cross-validation was ran for Mixture of Gaussians and SVM, the average accuracy across multiple runs decreased. A possible reason behind this decrease in performance may be related to the method in which pre-processing the image affected the image vector. Subtracting the mean and dividing by the variance may have made the distinct features that separate each facial expression less distinct. Thus, less distinct information is available to the training set and the accuracy across 10-fold cross-validation decreases. This decrease has also been confirmed with a third of the public test set on Kaggle. The performance of Pre-processing compared to no processing and the MATLAB standardize command is shown below

|  |  |
| --- | --- |
| **Method** | **Performance** |
| None | 70.8% |
| Pre-proccessing | 68.9% |
| MATLAB Standardize | 71% |

# 4.0 Conclusion

The model got around 71% performance. The SVM model has inherently a higher accuracy than KNN and MoG. However, the accuracy was not higher due to the lack of proper preprocessing of the data. Preprocessing the images in a certain way, such as using a Gabor Filter would have made the distinguishing features in the images more distinct. This would have increased the separation between the feature distinguishing data-points in the SVM model. Consequently, the increase in separation between these images would have created a model with higher accuracy.

Our model performed well for face with very distinct features like happy, as there will be a large separation between classes. This is ideal for the SVM as it can create a very large hyper-plane between the classes. The hardest ones for it to classify was the faces very close to neutral; one with little unique features. This is further compounded by the fact neutral images occupy the largest amount of training examples. This leads to the problem where ambiguous faces are always classified as neutral, introducing a bias.

In conclusion, SVM was pretty successful, but it was difficult to improve upon initial results. The kernels we tried performed poorly and changing the number of learners decreased the accuracy. Standardizing using the built-in function only improved the results slightly. This could be simply the limit of the SVM model for faces.

# 5.1 References

* The built-in MATLAB SVM function
* The Neural Net code from Assignment 2
* The MoG code from Assignment 2
* The kNN code from Assignment 1, 2 and 3