CMPUT 566 MINI-PROJECT: EIGENFACES FOR LABELED FACES IN THE WILD

Taivanbat Badamdorj

Department of Electrical and Computer Engineering
University of Alberta
tbadamdorj@ualberta.ca

March 25, 2020

ABSTRACT

Face recognition is a specific case of the image classification problem. Given a face, we would like to know whose face it is. In this report, we tackle the Labeled Faces in the Wild dataset [1] using the eigenfaces approach used in [2]. [2] uses Principal Component Analysis (PCA) to project each face into a "face-space". Using this projection, we learn to classify different faces using three different classifiers: a Naive Bayes classifier, Support Vector Machine, and a Neural Network. Finally, we use the student's t-test to compare the performance of the three algorithms.

1 Introduction

Face recognition is an interesting and important task in computer vision due to its problem formulation as well as its applications. The problem is a natural problem where we would like to identify whose face is in the image. However, it's a technically interesting problem as two images of the same face may have a larger variation than two images of different faces due to factors such as lighting and facial expression. This poses an interesting problem for computer vision to solve.

Facial recognition has various applications ranging from security (finding criminal suspects), as well as social (reading someone's mood from facial expressions).

Therefore, the Labeled Faces in the Wild Dataset was collected to advance the field of facial recognition. Since its introduction, it has become a common benchmark for the task.

In this project, we will test out three algorithms for the facial recognition task by using the principal components as a feature representation for the images.

2 Method

2.1 Principal Component Analysis (PCA)

Principal Component Analysis, a standard dimension reduction technique, was used in [2] as a way to classify faces.

PCA projects each image into a lower dimensional space spanned by the top N principal components of the data. The principal components are the vectors pointing in the direction of highest variance.

For our problem, we have the following definitions:

Let $\Gamma_1, \Gamma_2, \Gamma_3, ...\Gamma_M$ be the set of training images. The average face is defined as $\Psi = \frac{1}{M}\sum_{n=1}^{N}\Gamma_n$. We then use the difference $\Phi_i = \Gamma_i - \Psi$ to compute the covariance matrix $C = AA^T$ where $A = [\Phi_1, \Phi_2, ..., \Phi_M]$.

We then find the eigenvectors of this matrix, sorted in descending order by their respective eigenvalues, and pick the top N eigenvectors to be our basis.

Each image is then represented by its projection onto the subspace, i.e. $\tilde{\Gamma} = (\Gamma - \Psi) \cdot P^T$, where P^T is a matrix containing the top N principal components as column vectors.

2.2 Classifiers

Below we give a short description of each classifier that we use.

2.2.1 Naive Bayes

Naive Bayes models each feature as a Gaussian random variable with mean μ_i and variance σ_i^2. The mean and variance are computed from the dataset, and predictions use maximum likelihood e.g. we predict class 1 if $p(y = 1|x) = p(x|y = 1)p(y = 1) > p(y = 0|x)p(y = 0) = p(y = 0|x)$ for the binary classification task.

The Naive Bayes model uses a generative approach to classification. Due to its simplicity, it's worth testing for this project.

The features are assumed to be conditionally independent given the class label.

2.2.2 Support Vector Machine

A Support Vector Machine (SVM) [3] is also a linear classifier, where the objective is to learn a hyperplane that maximizes the distance from the hyperplane to the nearest object from each class. SVMs can do nonlinear classification using the kernel trick. In this project, we experiment with a nonlinear SVM using the radial basis function (RBF) kernel. SVMs have been shown to work well in classification tasks, and thus is a good algorithm to test out.

2.2.3 Neural Network

We use a neural network with up to 2 hidden layers for the classification task (the number of hidden layers is one of our hyperparameters). For each layer, we use one ReLU, tanh, or logistic as the activation. Similar to logistic regression, the final layer is also a softmax, and the predicted class is the class corresponding to the highest activation of $\hat{y}$.

Neural networks have helped achieve state of the art in many tasks, and thus we have decided to test it in our classification task - pitting it against the two more traditional approaches.

3 Dataset

We used the Labeled Faces in the Wild dataset [1]. It was created to advance the field of facial recognition, and is now a well-known benchmark in the field.

The original dataset contains 5749 distinct faces, and 13,233 samples. Each sample has dimension 5828, and the features are real values between 0 and 255.

We use a subset of this data by requiring each class to have at least 50 samples, as the dataset is highly unbalanced. We also scale the images by a factor of 0.4. Thus our final dataset has 1560 samples of dimension 1850 belonging to 12 classes. The classes are as follows (ordered from highest to lowest number of samples): Ariel Sharon, Colin Powell, Donald Rumsfeld, George W Bush, Gerhard Schroeder, Hugo Chavez, Jacques Chirac, Jean Chretien, John Ashcroft, Junichiro Koizumi, Serena Williams, Tony Blair.

We split the dataset 75/25, thus getting 1170 faces for our training set, and 390 images for our test set.

4 Experiments

All experiments were performed using the scikit-learn library [4] and the code can be found on my GitHub.

4.1 Hyperparameter tuning

We use stratified cross-validation with 5 folds on the training set to find optimal hyperparameters for each algorithm.

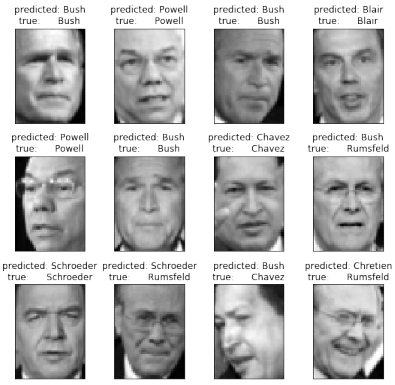

Figure 1: Naive Bayes

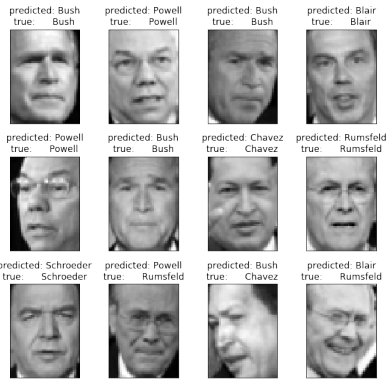

Figure 2: Support Vector Machine

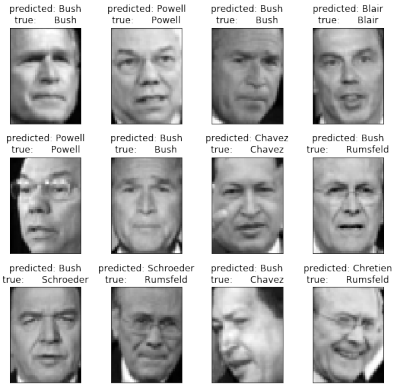

Figure 3: Neural Network

Figure 4: Qualitative classification results

4.2 Statistical Significance Testing

Then using the optimal hyperparameters that we have found using cross-validation, we compare the algorithms using the two-sided paired Student's t-test. Again we use stratified k-fold cross validation. We run this k-fold cross validation multiple times (10) as this has been shown to decrease type 1 errors.

We have the following null hypotheses, where the equality sign indicates that the performance is of equal quality according to our performance measure:

- SVM = Naive Bayes
- Naive Bayes = Neural Network
- SVM = Neural Network

By testing these hypotheses, we then state whether the difference in performance is statistically significant. We reject the null hypothesis if the performance differs with a p-value smaller than 0.05.

Finally, we train each algorithm on the entire training set and get the final average precision on the test set.

4.3 Metrics

We use mean average precision (mAP), a standard metric for object detection tasks [5]. First, the average precision is calculated for each class, and then we use the mean across all classes as our final metric. This is a good measure because it is not affected by class imbalances in the test set, as all classes contribute equally to the final measure.

In order to compute the average precision for a single class, a precision-recall curve is first obtained by using the confidence scores obtained from the model for each image. Average precision in the general case is defined as the integration of the precision over the precision-recall curve.

$$AP = \int_0^1 p(r)dr \tag{1}$$

Where $p(r)$ represents precision as a function of the recall r. The average precision is obtained using numerical integration. And finally we take the mean to get the mean average precision (mAP). A higher mAP is better.

5 Results

5.1 Classification

The main results for the best-performing hyperparameters for each model on the test set are presented in table 1. The SVM performs better than both methods, and we see that the mAP is very different from model to model. So, although

probably not required, we do t-testing simply for the sake of it. Qualititative results for the three models that were trained can be seen in figures 1. 2, and 3. The best hyperparameters can be found in table 3 and 4 in the appendix.

Table 1: mAP (mean average precision) for best performing hyperparameters chosen using stratified k-fold validation

Method	mAP
Naive Bayes	0.3952
Support Vector Machine	**0.6873**
Neural Network	0.5593

5.2 Paired t-test

The results of the repeated paired t-testing can be seen in table 2. As we can see, the p-values are very small and we reject the null hypothesis in each case i.e. no two algorithms have the same performance.

5.2.1 Checking assumptions of the paired t-test

We plotted each of our t-distributions for each hypothesis in figures 5, 6, and 7 in order to be sure that they are distributed according to the t-Distribution.

Table 2: t-testing results across 10 runs with p-value=0.05

Null Hypothesis	p-value	Reject Null Hypothesis?
SVM = Naive Bayes	8.29×10^{-24}	Yes
Naive Bayes = Neural Network	3.78×10^{-17}	Yes
SVM = Neural Network	3.31×10^{-7}	Yes

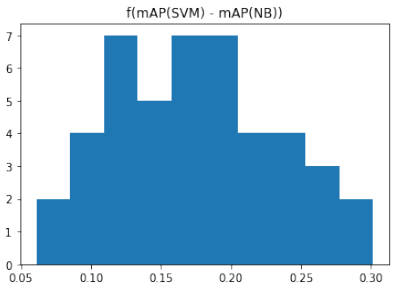

Figure 5: Distribution of SVM - NB

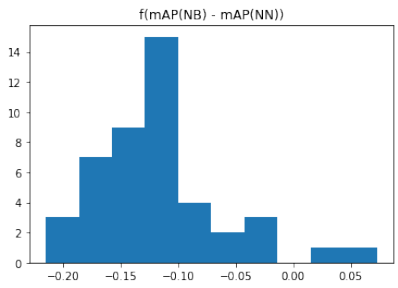

Figure 6: Distribution of NB - NN

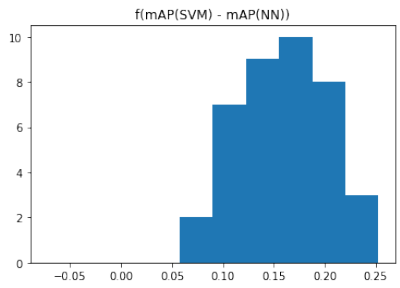

Figure 7: Distribution of SVM - NN

6 Conclusion

In this project, we compare the performance of Naive Bayes, SVM, and a Neural Network for the task of facial recognition. After finding optimal hyperparameters using k-fold cross validation, we find that the SVM outperforms the other algorithms. Finally, we run 5-fold cross validation 10 times in order to do statistical significance testing. We find that the results are significant (the difference in performance is not by chance), and reject the null hypothesis for all three of our paired tests.

References

[1] G. B. Huang, M. Mattar, H. Lee, and E. Learned-Miller, "Learning to align from scratch," in *NIPS*, 2012.

[2] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1991, pp. 586–591.

[3] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

7 Appendix

Table 3: Optimal Hyperparameters for Neural Network. For hidden layers, each number in brackets represents a hidden layer and its dimension. Multiple numbers in brackets means multiple hidden layers

Parameter	Values Tried	Optimal Value
Learning Rate	$10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}$	10^{-4}
Hidden layers	(100), (100,100), (200), (200,200)	(200)
Optimizer	SGD, Adam [6]	Adam [6]
Activation	tanh, ReLU, logistic	ReLU

Table 4: Optimal Hyperparameters for SVM

Parameter	Values Tried	Optimal Value
C	$10^3, 5 \times 10^3, 10^4, 5 \times 10^4, 10^5$	10^3
Gamma	$0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1$	0.005
Degree	$3, 4, 5$	3