Face Recognition using Naive Bayes Classifier*

Aleksandra Stachecka[1,*,†], Tomasz Procek[1,†]

[1]Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, POLAND

Abstract

The article is about the implementation and the performance of Face Recognition using Naive Bayes Classifier. Firstly, it is explained how huge impact on today's world has the face recognition system and how it has change over past 60 years. Then the idea of PCA and eigenfaces is brought closer to a reader. Moreover, the methodology of the program is explained. Not only, the most important functions and variables but also the schema of Naive Bayes Classifier are shown on code fragments. The next part is "Experiments" where viewer can find plots and specific information about dataset, examples of generated eigenfaces and the performance and accuracy of the program which is estimated to be nearly 75%. Finally, there is a conclusion. Authors one more time remind the most important information, explain the role of PCA used in the project and look to the future in order to improve their program.

Keywords

NaiveBayes, FaceRecognition, PCA

1. Introduction

Face recognition technology has its roots in 1960s. The groundbreaking steps where made by Woody Bledsoe, Helen Chan Wolf and Charles Bisson. They were marking the most characteristic facial features such as mouth, nose or eyes. Then computing distances between them and the program was comparing values. Their pioneering efforts brought people closer to concepts and techniques that later evolved with time, leading to creation of advanced face recognition systems which everyone uses nowadays. Biometric algorithms are widely used across various sectors of life [1, 2, 3]. Law enforcement agencies use it to identify suspects and find missing persons. Airports and border control implement it to verify travelers in security process. In businesses, it secures access to buildings and confidential information. Even Smartphones use it for user authentication. Moreover, social media platforms employ face recognition for tagging individuals in photos [4, 5].

Since 1960s specialists have been constantly improving biometric algorithms. One of those improvements is called Principal Component Analysis. The approach simplifies the complexity of facial images by focusing on key features, reducing dimensions, and retaining essential information. This method revolutionized face recognition by improving accuracy and computational efficiency. PCA paved the way for more advanced algorithms, significantly influencing the development of contemporary facial recognition systems.

*IVUS2024: Information Society and University Studies 2024, May 17, Kaunas, Lithuania

[1,*] Corresponding author

[†] These author contributed equally.

as308025@student.polsl.pl (A. Stachecka); tp307987@student.polsl.pl (T. Procek)

0009-0005-5736-9073 (T. Procek)

2. Methodology

The process begins with acquiring and preprocessing the dataset. The "Labeled Faces in the Wild" (LFW) dataset is fetched using the *fetch_lfw_people* function from *sklearn.datasets*. This dataset contains labeled images of faces, and for this analysis, only inviduals with at least 50 face images are included to ensure sufficient data per class. The images are resized to 50% of their original dimensions to reduce computational complexity. The dataset is then split into a feature matrix X and a target vector y. The feature matrix X contains the pixel values of the images, while the target vector y contains the corresponding class labels for each image. Additionally, the shape parameters of the images, including the number of samples, height and width are stored for reference throughout the analysis.

Listing 1: Fetching and Preprocessing LFW dataset

```
lfw_people = fetch_lfw_people(min_faces_per_person=50, resize=0.5)
X = lfw_people.data
y = lfw_people.target
target_names = lfw_people.target_names
n_samples, h, w = lfw_people.images.shape

original_shape = (h, w)
```

The feature matrix X is standarized using StandardScaler from sklearn.preprocessing. This standarization is crucial for ensuring that the principal component analysis (PCA) operates effectively. PCA is then applied to the standarized feature matrix X to reduce its dimensionality while retaining most of the variance. This reduction in dimensionality is achieved by extracting the most significant features, known as principal components, from the data. In this analysis, 100 principal components are retained, as determined by the parameter *n_components*. The transformed data is represented in a lower-dimensional space, resulting in the matrix *X_PCA*. PCA is particularly well-suited for this task because it effectively reduces the high dimen- sionality of image data while preserving essential features that contribute to variance. This reduction is crucial for computational efficiency and helps in avoiding the curse of dimensional- ity, which can adversely affect machine learning algorithms. PCA also helps in noise reduction and improves the performance of subsequent classifiers by focusing on the most significant features.

Listing 2: Standarization and PCA on the feature matrix

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Applying PCA
n_components = 100  # Number of principal components to keep
pca = PCA(n_components=n_components, svd_solver='randomized', whiten=True)
X_pca = pca.fit_transform(X_scaled)

print(f"Original shape: {X_scaled.shape}")
print(f"Transformed shape: {X_pca.shape}")
```

To visualize the principal components, known as eigenfaces, the components are reshaped to the original image dimensions. The first 10 eigenfaces are displayed to show the main features captured by PCA. Additionally, a function is defined to visualize the original and reconstructed faces, allowing for an assessment of how well PCA captures important features. The PCA-transformed data *X_pca* is inversely transformed to reconstruct the original images, and a subset of these reconstructed images is displayed alongside their original counterparts.

Listing 3: Eigenfaces visualization and reconstruction of original faces

```
eigenfaces = pca.components_.reshape((n_components, h, w))

plt.figure(figsize=(15, 8))
for i in range(10): # displaying the first 10 eigenfaces
    plt.subplot(2, 5, i + 1)
    plt.imshow(eigenfaces[i], cmap='gray')
    plt.title(f"Eigenface {i+1}")
    plt.xticks(())
    plt.yticks(())
plt.show()

def plot_reconstructed_faces(X_original, X_reconstructed, n_faces=4):
    plt.figure(figsize=(15, 8))
    for i in range(n_faces):
        ax = plt.subplot(2, n_faces, i + 1)
        plt.imshow(X_original[i].reshape((h, w)), cmap='gray')
        plt.xticks(())
        plt.yticks(())

        ax = plt.subplot(2, n_faces, i + 1 + n_faces)
        plt.imshow(X_reconstructed[i].reshape((h, w)), cmap='gray')
        plt.xticks(())
        plt.yticks(())

    plt.show()

X_reconstructed = pca.inverse_transform(X_pca)
plot_reconstructed_faces(X, X_reconstructed)
```

For classification, a Naive Bayes classifier [6] is implemented from scratch. This involves defining methods for fitting the model to training data, calculating likelihoods and posteriors and making predictions. The dataset is split into training and testing sets using and 80-20 ratio with the *train_test_split* function from *sklearn.model_selection*. The Naive Bayes classifier is trained on the training set and used to predict labels for the test set. The accuracy of the classifier is then calculated using *accuracy_score* from *sklearn.metrics*, providing a quantitative measure of the model's performance.

Listing 4: Naive Bayes classifier implementation and prediction

```
class NaiveBayes:
    def fit(self, X, y):
        self.classes = np.unique(y)
        self.mean = {}
```

```
        self.var = {}
        self.prior = {}

        for c in self.classes:
            X_c = X[y == c]
            self.mean[c] = np.mean(X_c, axis=0)
            self.var[c] = np.var(X_c, axis=0)
            self.prior[c] = X_c.shape[0] / X.shape[0]

    def _calculate_probability(self, mean, var, x):
        exponent = np.exp(-(x - mean) ** 2 / (2 * var))
        return exponent / np.sqrt(2 * np.pi * var)

    def _calculate_posterior(self, x):
        posteriors = []

        for c in self.classes:
            prior = np.log(self.prior[c])
            likelihood = np.sum(np.log(self._calculate_probability(self.mean[c], self.
                var[c], x)))
            posterior = prior + likelihood
            posteriors.append(posterior)

        return self.classes[np.argmax(posteriors)]

    def predict(self, X):
        return [self._calculate_posterior(x) for x in X]

X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.2,
    random_state=42)

nb = NaiveBayes()
nb.fit(X_train, y_train)

predictions = nb.predict(X_test)

accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy:.2f}")
```

Finally, a single image from the test set is selected to demonstrate the classifier's prediction capability. The selected image is transformed back to its original dimensions, and its true and predicted labels are displayed. This visual representation helps in understanding the model's performance on individual instances, complementing overall accuracy metric.

Listing 5: Demonstrating classifier's prediction capability

```
index = 12
single_image = X_test[index]
single_image_original = pca.inverse_transform(single_image).reshape(original_shape)

predicted_label = nb.predict(np.array([single_image]))[0]
true_label = y_test[index]

```

```
plt.imshow(single_image_original,  cmap='gray')
plt.title(f"True label: {target_names[true_label]}\nPredicted label: {target_names[
    predicted_label]}")
plt.xticks(())
plt.yticks(())
plt.show()
```

3. Experiments

The *labeled_faces_in_the_wild_dataset* is widely used for facial recognition tasks [7]. It contains JPEG images of various famous people, and each image is labaled with the name of the person. Scikit-learn [8] provides two loaders that will automatically download, parse the metadata files, decode the JPEG and convert the slices into memmapped numpy arrays.

```
array([[[0.3150327 , 0.33333334, 0.39738563, ..., 0.22352941,
         0.2784314 , 0.30588236],
        [0.3385621 , 0.34901962, 0.40392157, ..., 0.15555556,
         0.22745098, 0.3124183 ],
        [0.36078432, 0.38039216, 0.37124184, ..., 0.17254902,
         0.18431373, 0.2496732 ],
        ...,
        [0.18692811, 0.18431373, 0.1751634 , ..., 0.6640523 ,
         0.4366013 , 0.3124183 ],
        [0.18692811, 0.18954249, 0.18169935, ..., 0.58300656,
         0.54640526, 0.3895425 ],
        [0.18692811, 0.18562092, 0.18039216, ..., 0.5254902 ,
         0.606536  , 0.46535948]],

       [[0.11764706, 0.22352941, 0.31764707, ..., 0.1751634 ,
         0.16862746, 0.15555556],
        [0.14771242, 0.26797387, 0.34117648, ..., 0.22091503,
         0.20915033, 0.1882353 ],
        [0.17254902, 0.29150328, 0.34640524, ..., 0.25882354,
         0.2379085 , 0.21176471],
        ...,
        [0.03137255, 0.1124183 , 0.4261438 , ..., 0.76862746,
         0.7385621 , 0.7176471 ],
        [0.02745098, 0.12026144, 0.43398693, ..., 0.303268  ,
         0.33202615, 0.36862746],
```

Figure 1: The exapmle of converted data

For this analysis, we use a subset of the dataset where each person has at least 50 images. This subset contains 1560 images of 12 different people.

After decreasing the size of the photos and changing them into gray, PCA algorithm [9] leads to generation of eigenfaces which create a basis set derived from all the images used to construct the matrix. This results dimensional reduction by allowing the original training images to be represented by a smaller set of basis images.

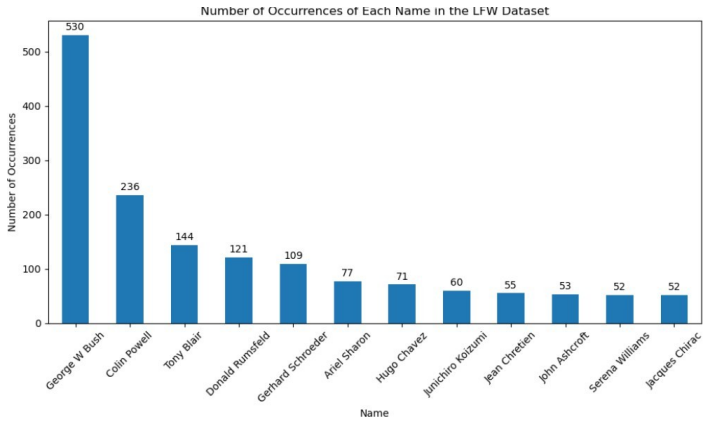

Figure 2: The plot shows the number of occurrences of each name of face in the dataset

Figure 3: Some examples of generated eigenfaces

By performing PCA we can notice a decrease in variance ratio in increasing number of components. So that the plot helps to decide how many principal components to select to retain as much information as possible while reducing the number of dimensions. We calculated that the best option for us is 100 components.

To check the performance of our Face Recognition using PCA and Bayes Classificator programe we created the confusion matrix which is fundamental tool in this field. It provides a detailed breakdown of how well the model's predictions the actual class labels.

Finally, in order to check detailed indicators as **precision**(ratio of true positive predictions to the total predicted positives), **recall**(ratio of true positive predictions to the total actual positives), **F1-Score**(average of precision and recall), **support**(the number of occurrences of each class in dataset) the classification report was made.

This report clearly shows the accuracy of our application which is estimated to be around 74%.

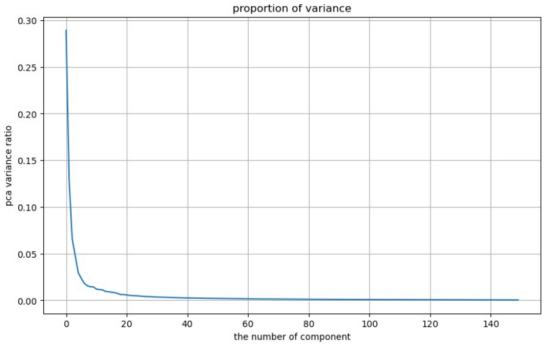

Figure 4: Proportion of variance

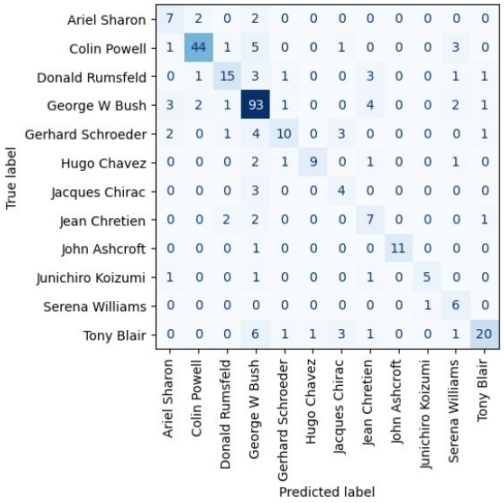

Figure 5: Confusion matrix

	precision	recall	f1-score	support
Ariel Sharon	0.50	0.64	0.56	11
Colin Powell	0.90	0.80	0.85	55
Donald Rumsfeld	0.75	0.60	0.67	25
George W Bush	0.76	0.87	0.81	107
Gerhard Schroeder	0.71	0.48	0.57	21
Hugo Chavez	0.90	0.64	0.75	14
Jacques Chirac	0.36	0.57	0.44	7
Jean Chretien	0.41	0.58	0.48	12
John Ashcroft	1.00	0.92	0.96	12
Junichiro Koizumi	0.83	0.62	0.71	8
Serena Williams	0.43	0.86	0.57	7
Tony Blair	0.83	0.61	0.70	33
accuracy			0.74	312
macro avg	0.70	0.68	0.67	312
weighted avg	0.77	0.74	0.74	312

Figure 6: Classification report

4. Conclusion

This project demonstrates the application of machine learning techniques for facial recognition using the "Labeled Faces in the Wild" (LFW) dataset. Through meticulous preprocessing and feature extraction, we prepared the dataset for analysis, ensuring its suitability for subsequent machine learning algorithms. Principal Component Analysis (PCA) played a pivotal role in reducing the dimensionality of the dataset while retaining essential variance, effectively capturing the underlying structure of the facial images. The visualization of eigenfaces provided valuable insights into the primary features captured by PCA, enhancing our understanding of the dataset's characteristics. The implementation of a Naive Bayes classifier facilitated the classification of facial images with satisfactory accuracy. By training the classifier on a subset of the dataset and evaluating its performance on unseen data, we gained valuable insights into its generalization capabilities. Furthermore, the visual representation of prediction results provided a tangible demonstration of the classifier's ability to accurately identify induviduals from facial images, showcasing the practical utility of the developed model. Overall, this project exemplifies the effectiveness of machine learning techniques in facial recognition tasks and underscores their potential for diverse real-world applications, ranging from security and surveillance to personalized user experience as beyond. As technology continues to evolve, further advancements in machine learning algorithms and datasets hold promise for even more accurate and robust facial recognition systems.

References

[1] M.-H. Le, N. Carlsson, Iddecoder: A face embedding inversion tool and its privacy and security implications on facial recognition systems, in: Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy, 2023, pp. 15–26.

[2] A. Jaszcz, D. Połap, Aimm: Artificial intelligence merged methods for flood ddos attacks detection, Journal of King Saud University-Computer and Information Sciences 34 (2022) 8090–8101.

[3] K. Prokop, D. Połap, G. Srivastava, J. C.-W. Lin, Blockchain-based federated learning with checksums to increase security in internet of things solutions, Journal of Ambient Intelligence and Humanized Computing 14 (2023) 4685–4694.

[4] M. Wieczorek, J. Siłka, M. Woźniak, S. Garg, M. M. Hassan, Lightweight convolutional neural network model for human face detection in risk situations, IEEE Transactions on Industrial Informatics 18 (2021) 4820–4829.

[5] B. U. H. Sheikh, A. Zafar, Unlocking adversarial transferability: a security threat towards deep learning-based surveillance systems via black box inference attack-a case study on face mask surveillance, Multimedia Tools and Applications 83 (2024) 24749–24775.

[6] A. Vidhya, Naive bayes explained, https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/, 2017. Accessed: 2024-05-20.

[7] A. A. Jha, Lfw people dataset, https://www.kaggle.com/datasets/atulanandjha/lfwpeople, 2023. Accessed: 2024-05-20.

[8] Scikit-learn developers, Scikit-learn: Machine learning in python, https://scikit-learn.org/stable/, 2023. Accessed: 2024-05-20.
[9] AIMonks, Principal component analysis (pca) in machine learning, https://medium.com/aimonks/principal-component-analysis-pca-in-machine-learning-407224cb4527, 2023. Accessed: 2024-05-20.