

Yizhen Ma - Assignment 1

by Yizhen Ma

FILE	25400_YIZHEN_MA_YIZHEN_MA_- _ASSIGNMENT_1_309231_1613885255.PDF (1.35M)		
TIME SUBMITTED	28-NOV-2020 12:28PM (UTC+0000)	WORD COUNT	4153
SUBMISSION ID	137666837	CHARACTER COUNT	22003

Face Recognition Using SVM Classifier

Yizhen Ma^{1,2}

1.University of Wales Trinity St David, Swansea, Britain; 2.Wuhan University of Technology, Wuhan, China

Abstract: We present a method that using SVM classifier for face recognition task. We build a new face dataset based on Voxceleb2 and also represent each face image using Eigenface. In order to remove variation in face images irrelevant for the recognition task, we utilize some preprocessing methods. To evaluate the performance of our model, we employ some common evaluation metrics for classification task, including classification accuracy and confusion matrix. Our method achieves 76.4% top-1 accuracy in identifying specific person, and achieves 46% top-1 accuracy on average. Further experimental results demonstrate that our method can solve the problem of face recognition to a certain extent.

Key words: face recognition; SVC ; Eigenface

1. Introduction

Face recognition refers to a biometric technology that uniquely identifies or verifies a person by comparing and analyzing patterns based on an individual's facial contour. The main challenges of face recognition are to solve both intra-personal variations and inter-personal variations, the former means to eliminate variations in different images of the same person, such as expressions, head poses and backgrounds, while the latter refers to the ability to identify the difference among images of different identities. Face recognition technology mostly includes face feature extraction, feature selection and classifier design stages, a typical pipeline preparing for face recognition[23] is showed in Figure 1.

In this assignment, we utilize SVM[5][17] classifier to complete the task of face recognition. To obtain a decent dataset for training, we employ some preprocessing methods on the audio-visual dataset Voxceleb2[14] which was designed for the task of speaker recognition. In addition, after we acquire our subset of VoxCeleb2, we further utilize a few data augment methods to improve the quality of our dataset. Since the dataset contains lots of videos of the same identity, it is convenient to extract multiple face images of the same person under different conditions (head pose, appearance etc.). We use some evaluation methods on our classifier, and the results indicate that our model can complete the face recognition task.

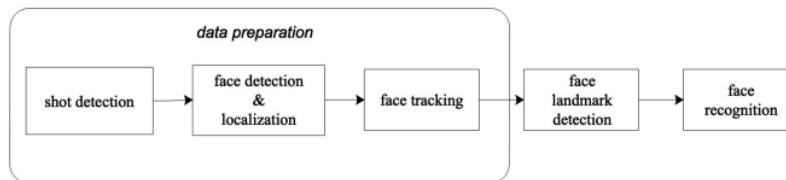


Figure 1. a typical pipeline for face recognition

2. Related work

2.1 classifying methods

KNN. K-Nearest Neighbor(KNN) algorithm is one of the simplest classification algorithms. It predicts the classification of new sample points by identifying data points that are divided into several classes. KNN is a non-parametric algorithm and a well-known representative of “lazy

learning". KNN can handle a small number of input variables very well, but problems will arise when the input is very large.

LR. Logistic regression(LR) is a discriminative model, it was first used for binary classification problem, due to its effectiveness, some extensions generalize it to multi-class problem which is called MLR(multinomial logistic regression). MLR utilize softmax function to replace logistic function in traditional binary logistic regression.

SVM. Support Vector Machine(SVM) was first proposed by Vapnik et al. [17]. SVM is a generalized method that classifier data in a supervised way, its decision boundary is the maximum-margin hyperplane for all samples. SVM is a sparse and robust classifier that calculate empirical risk using a hinge loss, it utilize a regularization term in the decision-making system to optimize. Kernel functions introduce nonlinearity into SVM, which also makes it one common technique of kernel learning[18].

Since the 1990s, SVM has experienced a rapid growth and has derived a variety of improved algorithms, which have been widely applied in academic problems, the most famous of them may be face recognition[19] and web classification[20].

In this assignment, we focus on explore the classification ability of SVM, i.e. we applied SVM to face recognition task, unlike previous methods which process with conditioned frontal face images, we innovatively explore using SVM in non-frontal face images, as well as other variations such as illumination and background.

2.2 Eigenface

PCA(principal components analysis) is a method of statistical analysis and simplification of dataset, it is often used to reduce the dimension of the dataset, while retaining the features in the dataset that contribute the most to the variance. PCA replace high dimensional principal components with the low dimensional principal components while keep most important information of the original dataset.

PCA was first proposed by Pearson[21] to analyze data and build mathematical models, After that, many derivative technologies based on PCA were born, one example is Eigenface[22], generally speaking, it conduct PCA on a large group of images describing different faces. Any new face image can be considered as a linear combination of these standard faces, while these standard faces are called Eigenface. Concretely, to compute Eigenface, we need a large dataset of human faces and resize all of them to a certain resolution, then just like PCA we compute the mean value which is also called mean face and the covariance matrix of the dataset. Finally we compute the eigen values and eigen vectors of the covariance matrix, the eigen vectors contain the information of shape variation, we select K eigen vectors with the largest eigen values, those K principal components are called eigenface. For any new face image, we can represent it by eigen faces through optimizing the coefficients of principal components.

In this assignment, we extract eigenfaces on our dataset, and every new faces are represented as combination of coefficients and corresponding components(eigenface).

2.3 face recognition

Bledsoe et al.[1] published their technical report about automatic face recognition(AFR), which is the earliest academic dissertation on face recognition. During that period, face recognition was mainly based on the facial geometric structure characteristics. These methods only describes the basic shape and structural relationship of the facial organs features, while ignoring the more detailed information in the entire image, which lead to poor classification results.

After that, Sirovich et al.[2] introduce PCA into their research of face recognition, Turk and Pentland[3] further introduce eigenface to extract statistical feature of face images, which lead to a new era in the area of face recognition, but eigenface method require high correlation between the training set and test set images, which lead to a low recognition rate when testing.

Another extraordinary work based on hidden Markov model was proposed by Samaria et al.[4], They believed that the face image should include the five most important feature areas of the face from top to bottom: forehead, eyes, nose, mouth and chin. Each organ of the face can be represented by a corresponding set of feature values, the drawback is their method require very large computational resources. SVM based method face recognition method was first proposed by Osuna et al.[6], Their method directly uses the nonlinear SVM classifier to classify real and fake faces.

With the development of deep neural networks[7][8], a large number of face recognition models based on deep learning algorithms emerged. Deepface[9] utilized convolution neural network and softmax activation function to obtain discriminative representation of faces, which can be used for face identification. DeepID and DeepID2[10][11] introduce contrastive loss to ensure the L2 distance of the same type of features as small as possible, and the L2 distance of different types of features is greater than a certain threshold value. Recently, Arcface[12] use an additive angle interval loss, the features and weights are also normalized. In order to stabilize the performance, ArcFace does not need to add other loss functions. Curricularface[13] adopt an adaptive curriculum learning loss function to solve the inherent defects of mining loss functions based on interval and difficult samples.



Figure 2. demo faces of our dataset

3. Method

3.1 dataset

We collect a small face dataset of different people, The procedures of collecting the dataset are as follows:

collect raw video data. we download raw videos in Voxceleb2. Voxceleb2 contains 145569 videos of 5994 identities, the videos have been cropped so each video only contain one person who is speaking. The structure of Voxceleb2 dataset is as follows. The first-level directories are named according to different person, and the second-level directories are different scenes, each scene may contain one or more short video segments, depends on the length of the original video. We only use the first video segment for each scene, in order to ensure the diversity of our data. For the same reason, we only extract single face image of each video segment. Since the original

dataset is too large for our task, we select a subset of 400 videos of 10 persons, so the problem becomes a 10-way classification, notably we discarded audio information in all videos, since our task is to recognize different identities only by information in image modality, while audio will introduce extra information of the person.

collect face images. for each video in our subset, we use only the first frame of each video. The reason is that different frames in a video share many similarities in pixel level, such similarities in low dimension will affect the judgment of the classifier, reflected in the results, the classification accuracy will be very high when two different frames of the same video appear respectively during training and testing, which contrary to our purpose that enable the classifier learn to identify based on high-level features related to human faces.

preprocessing face images. for each frame, we resize resolution of the original image from 224×224 to 64×64 and convert each image to grayscale, for the purpose of reducing computation complexity and the variation of input, there exists other preprocessing methods that can make our model work better, for example face frontalization[15][16] which has been proven that can improve performance of classification tasks, but due to they all involve some deep learning techniques to some extent, so we do not use them in this assignment.

3.2 implementation details

As mentioned in section 3.1, we first collect 400 face images of 10 different people, each person owns about 40 images. we put all face images of the same identity in one directory which was name after a list of consecutive numbers as labels(0,1,...9). For each face images, we resize it to 64×64 and convert it to grayscale. After that, we load all images into our program and we split all face images into training set and test set at the ratio of 3:1, resulting in a training set of 300 images and a test set of 100 images, some example faces can be seen in figure 2.

Normalization and data augment. Since our dataset is kind of small for face recognition, if the training set is too small, we won't get a stable model. we have adopt some data augment methods to improve the stability of our model. For every training face images, we flip it 3 times(horizontally,vertically,horizontally and vertically) and each time we get a new face image of the same person but from a different angle.Our experiment shows that this kind of rotation to some extend reduce the artifacts caused by different poses in our original dataset.

For normalization, we utilize the standard scaler in sklearn preprocessing package, which will normalize our training data into a normal distribution which the mean is 0 and the variance is 1.

Eigenface representation. We use PCA to compute 10 eigenfaces of the training set. The reason behind that is the original dimension of face images is too large for computation, so we use PCA to reduce the dimension. Since we have obtained the eigenfaces of our training set, we use the same transformation to our test set, the result of our eigenfaces can be seen in figure 3.

Training and testing. We use SVM with RBF kernel as our classifier, and the eigenfaces representation face images to train our classifier. Since finding the best hyper-parameters('c' and 'gamma') of SVM classifier is tricky, to find the best parameters of the classifier, we employ grid search to select the global optimal of these parameters. After that, we train our classifier with the parameters computed by grid search. The training stage takes about 80s for 1200 face images. We use 100 face images of 10 different identities for testing. Some testing results can be seen in figure 4.

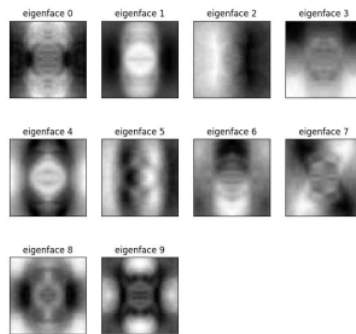


Figure 3. eigenfaces computed from our training set



Figure 4. some testing results. We test our model for 4 times, every time we use 10 images in the testing set to see the matching accuracy of our model

4. Evaluation

We evaluate our model using sklearn metrics packages. We select two methods to evaluate our model.

4.1 classification report

The first method is using classification report in sklearn. It contains four metrics to evaluate the classification result of our model.

precision: precision means the ratio $tp / (tp + fp)$ where tp is the number of true positives and fp is the number of false positives. The precision metric indicates the classification ability of our model that not to label negative samples as positive samples.

recall: the recall is the ratio $tp / (tp + fn)$ where tp is the number of true positives and fn the number of false negatives. When the recall metric is higher, it means our classifier is more able to find all positive samples.

f1-score: the f1-score metric is the weighted harmonic mean of the precision and recall, f1-score reaches its best when the value is 1 and worst when the value is 0.

support: the support metric indicates the frequency of appearance of each class in y-true.

Table 1: classification report

person id	precision	recall	f1-score	support
person 0	0.764	0.764	0.764	17
person 1	0.4	0.182	0.25	11
person 2	0.563	0.6	0.581	15
person 3	0.2	0.333	0.25	3
person 4	0.5	0.714	0.588	7
person 5	0.4	0.5	0.444	8
person 6	0.5	0.556	0.526	9
person 7	0.375	0.3	0.333	10
person 8	0.182	0.286	0.222	7
person 9	0.25	0.154	0.190	13
accuracy	0.46	0.46	0.46	0.46
macro avg	0.413	0.439	0.4162	100
weighted avg	0.459	0.46	0.45	100

From Table 1, we can draw the conclusion that, for person whose id is 0, we get the highest classification accuracy, while for person whose id is 3 and 8, our model fail to predict the true label of their face images, and images whose true label is not one of them are easily been incorrectly classify into these categories.

4.2 confusion matrix

confusion matrix usually used as a method to visualize the accuracy of classification tasks, it describe the classification result with the ground-truth value and predicted value.

Each column of the confusion matrix represents a specific prediction label while the sum of each column represents the amount of data predicted to be labeled as the column represents; each row represents the true label of the data while the total number of data in each row represents the how many samples to be of that category in the original dataset.

As show in figure 5, the horizontal axis indicate labels predicated by our model, while the vertical axis represent the true label of the image, we use color tensity to representation how many samples are classified into the grid, the accuracy of model can be seen roughly from the diagonal.

Well, Although for some identities, our model emerge some mismatches, the failure cases include people whose id is 1,7 and 9, but from the whole picture, our model can classify face images correctly,especially for people whose id are 0 and 4.

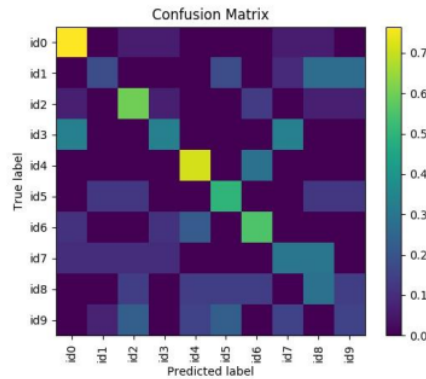


Figure 5. confusion matrix of testing

5. Conclusion

We introduced a face recognition method based on SVM. We did some preprocessing on the audio-visual dataset Voxceleb2 to obtain a dataset that can be used for face recognition. Some

evaluation methods show that our model can complete the task of face recognition to some extent.

Although our method has some effect on the problem of face recognition, there are still some problems. First, there are significant differences in classification accuracy for different people, the major reason may be the quality of training images, which means the images of single person are too different (appearance, pose and wearing), due to the fact that our face images are randomly select and it is kind of hard to remove the influence of these factors, so this is inevitable. Perhaps some further preprocessing methods can eliminate these variations, for example, frontalize all the pictures to remove the difference in pose among people. Since our original image contains clear background, this is also one of the key factors that may cause the classification accuracy to be low. Except for the quality of dataset, the expressive ability of SVM may be insufficient, which means it still lack of the ability to generalize to random wild face images.

References

- [1] W. W. Bledsoe. The model method in facial recognition. Technical report pri 15, Panoramic Research, Inc., Palo Alto, California, 1964
- [2] Sirovich, Lawrence, and Michael Kirby. "Low-dimensional procedure for the characterization of human faces." *Josa* a 4.3 (1987): 519-524.
- [3] Turk, Matthew, and Alex Pentland. "Eigenfaces for recognition." *Journal of cognitive neuroscience* 3.1 (1991): 71-86.
- [4] Samaria, F. S., & Harter, A. C. (1994, December). Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE workshop on applications of computer vision* (pp. 138-142). IEEE.
- [5] Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992, July). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144-152).
- [6] Osuna, E., Freund, R., & Girosit, F. (1997, June). Training support vector machines: an application to face detection. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition* (pp. 130-136). IEEE.
- [7] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
- [8] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84-90.
- [9] Taigman, Y., Yang, M., Ranzato, M. A., & Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1701-1708).
- [10] Sun, Y., Wang, X., & Tang, X. (2014). Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1891-1898).
- [11] Sun, Y., Chen, Y., Wang, X., & Tang, X. (2014). Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems* (pp. 1988-1996).
- [12] Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and*

- Pattern Recognition (pp. 4690-4699).
- [13] Huang, Y., Wang, Y., Tai, Y., Liu, X., Shen, P., Li, S., ... & Huang, F. (2020). Curricularface: adaptive curriculum learning loss for deep face recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5901-5910).
 - [14] Chung, J. S., Nagrani, A., & Zisserman, A. (2018). Voxceleb2: Deep speaker recognition. arXiv preprint arXiv:1806.05622.
 - [15] Cao, J., Hu, Y., Zhang, H. et al. Towards High Fidelity Face Frontalization in the Wild. Int J Comput Vis 128, 1485 – 1504 (2020). <https://doi.org/10.1007/s11263-019-01229-6>
 - [16] Banerjee, S., Brogan, J., Krizaj, J., Bharati, A., Webster, B. R., Struc, V., ... & Scheirer, W. J. (2018, March). To frontalize or not to frontalize: Do we really need elaborate pre-processing to improve face recognition?. In 2018 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 20-29). IEEE.
 - [17] Cortes, C. & Vapnik, V., (1995). Support-vector networks. Machine learning, 20(3), pp.273-297.
 - [18] Hsieh, W.W.. Machine learning methods in the environmental sciences: Neural networks and kernels: Cambridge university press,2009:Chapter 7, pp.157-169
 - [19] Qin, J. and He, Z.S., 2005, August. A SVM face recognition method based on Gabor-featured key points. In Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on (Vol. 8, pp. 5144-5149). IEEE.
 - [20] Sun, A., Lim, E.P. and Ng, W.K., 2002, November. Web classification using support vector machine. In Proceedings of the 4th international workshop on Web information and data management (pp. 96-99). ACM.
 - [21] Pearson, K. On Lines and Planes of Closest Fit to Systems of Points in Space (PDF). Philosophical Magazine. 1901, 2 (6): 559 – 572.
 - [22] Sirovich, L. and Kirby, M. (1987) Low-Dimensional Procedure for the Characterization of Human Faces. Journal of the Optical Society of America A, 4, 519-524.
 - [23] Joon Son Chung, Andrew Zisserman: Lip Reading in the Wild. ACCV (2) 2016: 87-103

Appendix

In this part, we will show the implementation of our code

extract_image.py: it is for extracting face images from Voxceleb2 videos.

main.py: which is used to do data preprocessing, training, testing and evaluation.

1. extract_image.py

```
"""
extract face images from Voxceleb2 raw videos
for each identity, we have several videos of the same person
for each video, we only use the first segment and save the first images of the segment
"""

def extract_images():
    data_root = "./data/raw"
    save_root = "./data/input"

    # for each segment, how many frames we are going to save
    frame_per_segment = 1

    ids = sorted(glob.glob(data_root + '/*'))
    id_count = len(ids)
    total_frames = 0

    for current_id in range(id_count):
        frame_count = 0

        if not os.path.exists(os.path.join(save_root, str(current_id))):
            os.makedirs(os.path.join(save_root, str(current_id)))
        videos = sorted(glob.glob(ids[current_id] + '/*'))

        for video in videos:
            segments = sorted(glob.glob(video + '/*'))
            # for segments of one video, we only use the first segment
            segment = segments[0]
            reader = imageio.get_reader(segment)

            for (i, image) in enumerate(reader):
                if i >= frame_per_segment:
                    break
                frame_count += 1
                save_path = os.path.join(save_root, str(current_id), 'frame' + str(frame_count)
+ '.jpg')
```

```
result_image = process_img(image)
imageio.imwrite(save_path, result_image)
```

```
total_frames += frame_count
```

```
print('total frames:'+str(total_frames))
```

2. main.py

```
### code for data preprocessing, training, testing and evaluation.
```

```
DATA_ROOT = "./data/input"
```

```
id_count = 10
```

```
total_frames = 400
```

```
def load_image(id_count):
```

```
    h, w = 0, 0
```

```
    total_count = 0
```

```
    for i in range(id_count):
```

```
        img_path = glob.glob(os.path.join(DATA_ROOT, str(i) + '/*'))
```

```
        for j in range(len(img_path)):
```

```
            img = mpimg.imread(img_path[j])
```

```
            h, w = img.shape
```

```
            img_flatten = img.reshape(h * w)
```

```
            all_data_set.append(img_flatten)
```

```
            all_data_label.append(i)
```

```
            total_count += 1
```

```
    print(total_count)
```

```
    return h, w
```

```
def augment(x_train: np.ndarray, y_train: np.ndarray):
```

```
    x_train_list = x_train.tolist()
```

```
    y_train_list = y_train.tolist()
```

```
    for (i, image) in enumerate(x_train):
```

```
        image = np.reshape(image, (64, 64))
```

```
        h_flip = cv2.flip(image, 1) # Flipped Horizontally
```

```
        x_train_list.append(np.reshape(h_flip, (64 * 64)))
```

```
        y_train_list.append(y_train[i])
```

```
        v_flip = cv2.flip(image, 0) # Flipped Vertically
```

```
        x_train_list.append(np.reshape(v_flip, (64 * 64)))
```

```

        y_train_list.append(y_train[i])

        hv_flip = cv2.flip(image, -1) # Flipped Horizontally & Vertically
        x_train_list.append(np.reshape(hv_flip, (64 * 64)))
        y_train_list.append(y_train[i])

    x_train = np.array(x_train_list)
    y_train = np.array(y_train_list)

    return x_train, y_train

all_data_set = []
all_data_label = []
h, w = load_image(id_count)

x = np.array(all_data_set)
x = preprocessing.StandardScaler().fit(x).transform(x)

y = np.array(all_data_label)

n_samples_ = x.shape
n_classes = len(np.unique(y))
labels = []
for i in range(id_count):
    names = "person" + str(i)
    labels.append(names)

# statistic dataset
print("n_samples: %d" % n_samples)
print("n_classes: %d" % n_classes)

# train/test split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25) # print(x_train.shape,
y_train.shape)
x_train, y_train = augment(x_train, y_train)
print('after augment, the length of training set:', len(x_train))

# PCA
n_components = 10
pca = PCA(n_components=n_components, svd_solver="randomized", whiten=True).fit(x_train)

# eigenfaces
eigenfaces = pca.components_.reshape((n_components, h, w))
x_train_pca = pca.transform(x_train)

```

```

x_test_pca = pca.transform(x_test)

# train
ckpt_path = 'checkpoints/model_ckpt_1.m'

if os.path.exists(ckpt_path):
    clf = joblib.load(ckpt_path)
    print('load model from checkpoint %s' % ckpt_path)
else:
    t0 = time()
    param_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5],
                  'gamma': [1e-4, 5e-4, 1e-3, 5e-3, 1e-2, 1e-1]}
    clf = GridSearchCV(SVC(kernel="rbf", class_weight="balanced"), param_grid)
    clf = clf.fit(x_train_pca, y_train)

    print('train done in %0.3fs' % (time() - t0))
    joblib.dump(clf, ckpt_path)
    print('grid search result:')
    print(clf.best_estimator_)

# test
t0 = time()
y_pred = clf.predict(x_test_pca)
print('test done in %0.3fs' % (time() - t0))

# outputs
report = classification_report(y_test, y_pred, target_names=labels, output_dict=True)
print(report)

df = pandas.DataFrame(report).transpose()
df.to_excel('./output/report/report1.xlsx', sheet_name='Sheet1')

cm = confusion_matrix(y_test, y_pred, labels=range(n_classes))
plot_confusion_matrix(cm, labels_name=['id' + str(i) for i in range(n_classes)], title='Confusion
Matrix')
plt.savefig('output/confusion_matrix/cm1.jpg')

pred_titles = []
for i in range(y_pred.shape[0]):
    pred_titles.append(plot_title(y_pred, y_test, labels, i))

# plot some prediction result
plot_gallery(x_test, pred_titles, h, w, savepath='output/predict_visualize_13.jpg')

```

```
titles = ['eigenface %d' % i for i in range(eigenfaces.shape[0])]
```

```
# plot eigenface
```

```
plot_gallery(eigenfaces, titles, h, w, savepath='output/eigenfaces.jpg')
```

```
plt.show()
```

Yizhen Ma - Assignment 1

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

68/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

GRADING FORM: SMU GENERIC_GJ

YIZHEN MA

68

RESEARCH-INFORMED LITERATURE



A very good review of related literature (given the limited word count). In the end you focused on only one algorithm (SVM) but place it in the context of a working face recognition application. PCA also used here as part of the method which helps, however I was expecting you to continue onto a second method and provide two distinct classification or regression methods (see the brief). Hence the grade - though I have taken into consideration the context and application of PCA here.

6.5

KNOWLEDGE AND UNDERSTANDING



A very good explanation of the overall approach though a little more detail on the SVM method itself would help. PCA and its role in the process well documented. Appropriate methods of evaluation selected and applied.

13.5

ANALYSIS



Good analysis of the SVM and you do incorporate grid search to identify best parameters. Did you separate your dataset into two test and train sets? You need to do this and discuss it here in your report. Some detailed pre-processing attempted.

23.25

PRACTICAL APPLICATION AND DEP.



Some very good work here. Pre-processing good.
Evaluation and production of visualisations good.
Well done.

21

SKILLS FOR PROFESSIONAL PRAC.



Well structured. References appropriate. Diagrams
helpful.

3.75