Face Recognition using KNN, SVM and Logistic Regression

Yunwei Jiang, Chengjie Zhu

1. Introduction, Background and Motivation

Face recognition is one of the most challenging jobs of pattern recognition [1]. Many machine learning algorithms have been applied to the field because face recognition is a hotspot [2]. Under most circumstances, the dimension of original images is too high, thus many researchers applied PCA, LDA and other dimension reduction techniques. Reference [1] [2] [3] implemented 3 different algorithms in ORL which is a widely used dataset and consists of 40 individuals' 400 images, containing a high degree of variability in expression, pose, and facial details [3]. Recently, many companies and universities have developed their own algorithms and made significant progress.

2. Problem Statement

In reference [1], bagging KNN algorithm is implemented in the ORL dataset and achieves an accuracy of 92.07%, but implementation of bagging KNN is quite complex. In reference [3], the authors construct a bottom-up binary tree for classification and choose the most likely one for prediction. Fixed dataset ORL limits the application of these algorithms. In this project, we implemented a simpler KNN algorithm, adopted the one-vs-one scheme for SVM, introduced the idea of majority voting for SVM prediction and utilized logistic regression for face recognition. We built a face recognition system based on OpenCV3 to address the problem of fixed-size dataset, which can register new faces and distinguish strangers.

3. Methodology

3.1 PCA Algorithm

PCA transforms correlated variables to linearly uncorrelated by utilizing orthogonal transformation [4]. The left variables are called principal components.

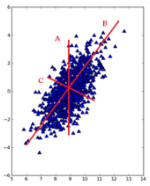


Figure 1. Three classes before PCA [5]

PCA transforms the dataset coordinate system to a new one chosen by the data itself [5]. In Figure 1, the axis covering the largest data variation is selected first, then, the axis that is orthogonal to the first one and has the second largest data variation is chosen as second axis. After PCA, three classes in Figure 1 should look like what in Figure 2.

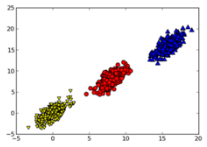


Figure 2. Three classes in two dimensions [5]

3.2 KNN Algorithm

KNN algorithm is a non-parametric method for classification [6]. KNN algorithm is intuitionistic: for each test sample, we look for K instances that are most similar to the test sample from training set. The similarity is measured by certain metrics such as Euclidean distance. The test sample is classified as the major class of the K instances.

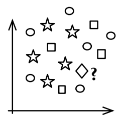


Figure 3. Principle of KNN

3.3 SVM Algorithm

Basic SVM is a binary classification algorithm maximizing the margin in the feature space. It supports kernel functions and penalty. For a simple binary classification problem, we aimed to find a hyperplane $w \cdot x + b = 0$, maximizing the minimum margin between one sample and the hyperplane, to classify these samples to positive (+1) and negative (-1).

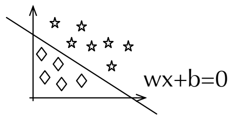


Figure 4. Binary classification of SVM

Sometimes, there are some outliers in the dataset.

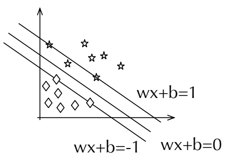


Figure 5. Dataset with outliers

In this case, slack variables $\xi_i \geq 0$ and penalty-factor C>0 are introduced. The primal problem is hard to solve, but we can try to solve it by resolving its dual problem.

By Lagrange Duality, we get the dual problem $\begin{cases} \min\limits_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N}\alpha_i \\ s.t. \sum_{i=1}^{N}\alpha_i y_i = 0 \; (0 \leq \alpha_i \leq C) \end{cases}$.

SMO algorithm is used to find the best alpha pairs. In each iteration, the algorithm only selects 2 variables α_i, α_j and update them while keeping other alphas fixed. We need to pay attention to the constraint $\begin{cases} \mathrm{L} = \max(0, \alpha_j - \alpha_i), \mathrm{H} = \min(\mathrm{C}, \mathrm{C} + \alpha_j - \alpha_i) \; if \; y_i \neq y_j \\ \mathrm{L} = \max(0, \alpha_j + \alpha_i - C), \mathrm{H} = \min(\mathrm{C}, \alpha_j + \alpha_i) \; if \; y_i = y_j \end{cases}$ [5].

After updating alphas, we need to update b following the rules $b = \begin{cases} b_1 \; if \; 0 < \alpha_i < C \\ b_2 \; if \; 0 < \alpha_j < C \\ \frac{b_1 + b_2}{2} \; otherwise \end{cases}$.

For linear non-separable problems, we can solve them with kernel functions.

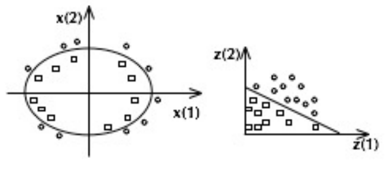


Figure 6: Linear Non-Separable Problem and Kernel Functions [5]

The basic idea of kernel functions is to map the input space to a Hilbert space through nonlinear transformation: $\phi(x): X \to H$ and for all $x, z \in X, K(x,z) = \phi(x) \cdot \phi(z)$.

3.4 Logistic Regression

Logistic regression is a kind of linear regression [2]. The key difference is that logistic regression uses a logistic function. One popular logistic function is sigmoid function $y = \frac{1}{1+e^{-w\beta x}}$ (β is a constant), which is smooth, strictly monotonic and totally differentiable.

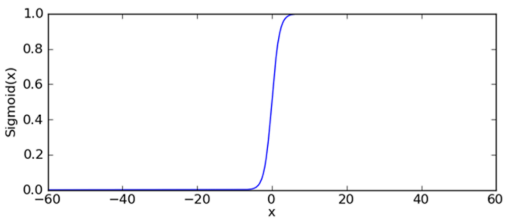


Figure 4. Sigmoid function

We use gradient descend algorithm to find the best w. The formula is $w \leftarrow w - \alpha \nabla_w f(w)$.

3.5 Progress so far

1) We finished the basic KNN algorithm.
2) We used linear kernel function to SVM.
3) We finished the logistic regression algorithm.
4) We implemented a face recognition system based on OpenCV3.

3.6 What is Left to be Done

1) The performance of KNN algorithm can be improved by using KD Tree.
2) Gaussian kernel function will be performed for SVM algorithm.
3) Kernel functions can be introduced to logistic regression for better performance.
4) The face recognition system based on OpenCV will be integrated with ORL dataset.

3.7 Problems Encountered and Alternative Solutions Applied

At first, we adopted the one-vs-rest scheme for SVM, however, it didn’t work because of unbalanced data. We changed our scheme to one-vs-one.

3.8 Changes in the Project Plan

Because of the lack of neural network design experiences, we changed our plan of implementing Back Propagation Neural Network to logistic regression.

4. Novelty

1) The project can increase the size of database by capturing new faces, and this can be a prototype of commercial software.
2) Many researchers didn’t consider strangers, but our project can distinguish strangers.
3) In reference [3], prediction is decided by the most possible model. However, in our project, we introduced an idea of majority voting for the SVM classification.

5. Analysis and Design of Project

5.1 Requirements Analysis and Uses Cases

This project consists of 2 parts: the basic part is the implementation of KNN, SVM, logistic regression on the ORL dataset; the advanced part is a face recognition system based on OpenCV which allows registering new faces, recognizing registered users and strangers. The use cases are shown in the following figure:

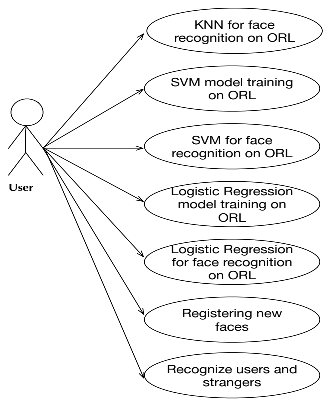

User
KNN for face recognition on ORL
SVM model training on ORL
SVM for face recognition on ORL
Logistic Regression model training on ORL
Logistic Regression for face recognition on ORL
Registering new faces
Recognize users and strangers

Figure 5. Use cases

5.2 Design Architecture

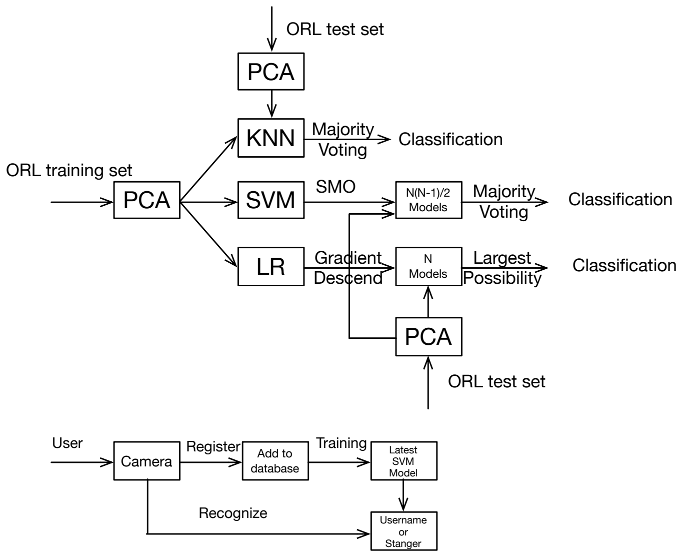
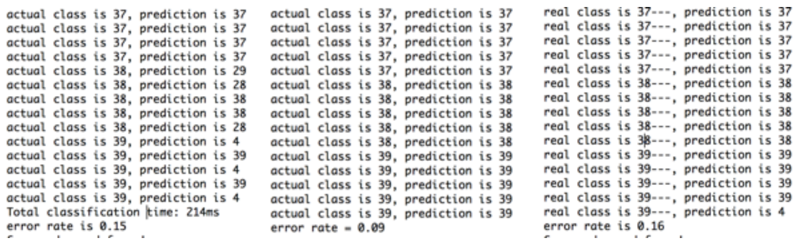


Figure 6. Overall Architecture

5.3 Testing

We only list a part of output here. Testing result of basic part:



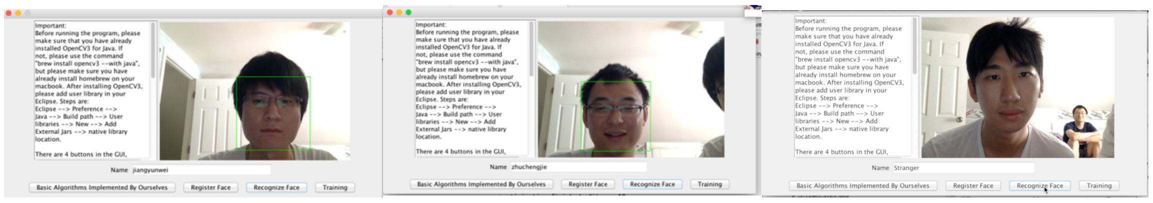
```
actual class is 37, prediction is 37
actual class is 37, prediction is 37
actual class is 37, prediction is 37
actual class is 37, prediction is 37
actual class is 38, prediction is 29
actual class is 38, prediction is 28
actual class is 38, prediction is 38
actual class is 38, prediction is 38
actual class is 38, prediction is 28
actual class is 39, prediction is 4
actual class is 39, prediction is 39
actual class is 39, prediction is 4
actual class is 39, prediction is 39
actual class is 39, prediction is 4
Total classification time: 214ms
error rate is 0.15
```

```
actual class is 37, prediction is 37
actual class is 37, prediction is 37
actual class is 37, prediction is 37
actual class is 37, prediction is 37
actual class is 37, prediction is 37
actual class is 38, prediction is 38
actual class is 38, prediction is 38
actual class is 38, prediction is 38
actual class is 38, prediction is 38
actual class is 38, prediction is 38
actual class is 39, prediction is 39
actual class is 39, prediction is 39
actual class is 39, prediction is 39
actual class is 39, prediction is 39
actual class is 39, prediction is 39
error rate = 0.09
```

```
real class is 37---, prediction is 37
real class is 37---, prediction is 37
real class is 37---, prediction is 37
real class is 37---, prediction is 37
real class is 37---, prediction is 37
real class is 38---, prediction is 38
real class is 38---, prediction is 38
real class is 38---, prediction is 38
real class is 38---, prediction is 38
real class is 38---, prediction is 38
real class is 39---, prediction is 39
real class is 39---, prediction is 39
real class is 39---, prediction is 39
real class is 39---, prediction is 39
real class is 39---, prediction is 4
error rate is 0.16
```

Figure 7. Test of KNN, SVM, Logistic Regression

The testing result of advanced part:



Important:
Before running the program, please make sure that you have already installed OpenCV3 for Java. If not, please use the command "brew install opencv3 --with java", but please make sure you have already install homebrew on your macbook. After installing OpenCV3, please add user library in your Eclipse. Steps are:
Eclipse --> Preference --> Java --> Build path --> User libraries --> New --> Add External Jars --> native library location.

There are 4 buttons in the GUI,

Name jiangyunwei

Name zhuchengjie

Name Stranger

Figure 8. Face Recognition System testing

5.4 Results and Evaluation

The accuracy of basic part is:

Algorithm	KNN	SVM	Logistic Regression
Accuracy	85%	91%	84%

The result of advanced part is showed in Figure 8. We will not mention it again.

5.5 General AI evaluation criteria

We evaluated the criteria with accuracy and confusion matrix. The accuracy of SVM is acceptable for the ORL dataset. But the other two algorithms need to be improved.

6. The AI algorithms/methods used

Work allocation:

Yunwei Jiang	**SVM, Logistic Regression, Face recognition System(OpenCV)**
Chengjie Zhu	**PCA, KNN**

6.1 PCA Algorithm

For basic part, there are 40 individuals' images. Each person has 5 images for training and 5 images for testing. The size of an image is 112*92=10304. Each image is stored at a 112*92 array whose elements are grayscale values. Each array is transformed to a 10304 x 1 vector R_i. The training set is $\{R_i | i = 1,2,\ldots,200\}$. The average vector is: $\bar{R} = \frac{1}{200}\sum_{i=1}^{200} R_i$.

After Subtracting $\bar{R}$ from R_i, the difference vector is: $A_i = R_i - \bar{R}, i = 1,2 \ldots 200$. Calculating A_i is for translating the original coordinate to the location of the average. And A_i is used to denote each training sample. Training dataset can be denoted with a matrix: $A = [A_1, A_2, \ldots, A_{200}]$.

Covariance matrix is calculated by $C = AA^T$. C is a square matrix and can be decomposed to $C = Q\Lambda Q^{-1}$ by Eigen Decomposition. Here Q is a vector whose ith column is the ith eigenvector of C and Λ is a diagonal matrix whose diagonal elements are corresponding eigenvalues. After Eigen Decomposition, there are 200 pairs of eigenvector and eigenvalue. The eigenvalues are in descending order, which means eigenvector with largest eigenvalue captures the most variation among training dataset. Because of this, 90% information of whole dataset can be reserved through first 75 pairs of eigenvectors and eigenvalues. These pairs are used to create a coordinate, whose element is eigenvalues^(-1/2). 200 original training images are projected to the base coordinate and then stored at a 200*75 array. Then the same process is done for test image. After applying PCA, dimension of image is reduced from 10304 to 75. PCA can reduce complexity of data and identify most important features, but could throw away useful information sometimes [5].

6.2 KNN Algorithm

The input for KNN is trainX, trainY, testX and testY. trainX is the training dataset processed after PCA, trainY stores labels of training dataset. testX and testY are for test dataset. For each test sample, Euclidean distances to each training images are calculated first. A TreeMap is created whose keys are the values of distances. All of the training images' labels are added into TreeMap based on the distance. Starting from the nearest distance, K train images' labels are collected at an array for following voting. The training image's label that holds majority in this array is the prediction result of KNN for the test sample.

Value of K is an odd number and crucial to accuracy of KNN algorithm. A small K can decrease approximation error and only those close training samples are considered, but estimation error increases and the prediction result is quite sensitive. Additionally, if the nearest instances are noises, the prediction result may be wrong. If K is larger, more neighbors decide the prediction. It reduces the estimation error but increases approximation error. Instances far from the test samples are considered and may result in error.

6.3 SVM Algorithm

For each 2 classes in the ORL dataset, one model is built. For example, for class 20 and class 21, we set $label_{20} = 1$, $label_{21} = -1$, thus we build $C_{40}^{2} = 780$ models. For one class pair, SMO algorithm is used to find the alphas: Traverse every alpha in the dataset first, then choose the another alpha randomly. Two alphas should be updated simultaneously due to the constraint $\sum \alpha_i y_i = 0$. To do this, a function selectJRandomly(i, m) that randomly selects one integer, which is no greater than "m" and unequal to "i", is used. The "i" is the index of the first alpha, and "m" is the number of alphas. A function clipAlpha(α,H,L) clipping alphas $(\alpha \geq H \text{ } or \text{ } \alpha \leq L)$ is created.

We first choose a class pair (target, compared) and then combine their corresponding vectors as a matrix. The function SVM(trainX, trainY, C, toler, maxIteration, target, compared) is to implement SMO algorithm. "trainX" is the training dataset processed after PCA, "trainY" is the array of corresponding classes, "C" is the penalty factor, "toler" is the tolerance, "maxIteration" is the maximum number of iterations the program should run, "target" is the class regarded as positive, "compared" is the class regarded as negative.

In each iteration, set alphaPairsChanged (used to record if any alphas optimization happens) to 0 and then visit the entire set one by one. First, compute the prediction through $fX_i = \alpha \cdot Y \cdot \emptyset(X) \cdot \emptyset(X_i)$. The error $E_i = y_i - fX_i \; (y_i = \pm 1)$. For large error, the corresponding alpha can be optimized. Both positive and negative margins should meet the "if" clause because of the constraint 0<α<C. Next, select α_j randomly and then calculate the error. After this, compute the upper and lower bound according to the SMO algorithm. "Eta" is the optimal amount to update α_j. If eta = 0, it's messy to compute the new α_j, we ignore such case because it only happens occasionally. After this step, we calculate the new α_j and then clip its value. Small changes of α_j are ignored. After updating α_j and α_i, we start updating b. Repeat the procedure above for "maxIteration" times, then, finish training models. All models vote for prediction and the majority would be selected as the prediction.

6.4 Logistic Regression Algorithm

For logistic regression, 40 models for the 40 classes are built. The function LogisticRegression(trainX, trainY, maxIteration, target) is used to train the models. "trainX" is the training dataset, "trainY" is the array of their corresponding classes, "maxIteration" is the number of iterations before quitting and "target" is the class considered as positive (+1) and other classes are considered as negative (0). For the dataset, all elements are multiplied by 256 first, then add one column whose elements are "1" to the matrix as the first column, the new matrix is called "newTrainX". After doing this, perform the gradient descend algorithm on "newTrainX": choose the sum of squared errors as cost function. Set learning rate $\alpha = 0.01$, and compute the errors between prediction and the labels (1 or 0). Make the new dataset multiply the errors.

Repeat this procedure for 40 classes and train 40 models. The label of each model is the class considered as positive. When making prediction, calculate the possibility of each class with its model that this class is regarded as positive. The prediction is the label of the model with largest possibility.

6.5 Face Recognition System based on OpenCV

Advanced part of the project is a face recognition system based on OpenCV3. First, make the system load OpenCV library. Then, call the local camera and initialize a CascadeClassifier by reading the XML file of HARR features (a file can be found in OpenCV directory) so that the system can detect faces. The system would show green square frames to highlight faces captured by camera. So far, the system can only recognize 1 face and the size of the face should be no less than 140×140. Larger images are cut to 140×140. Images are saved as grayscale images. For each new user, it is required to take no less than 10 images and the system will only keep the latest 10 images. After registering new users, new SVM model need to be trained. In this OpenCV system, we implemented SVM with libsvm. When recognizing, the system first considers the person as registered, after predicting the class, the system retrieves all images of the prediction, then compute the average cosine similarity between the person's face and the images of the prediction. If the average similarity is no less than 90%, system outputs the prediction, otherwise, classifies the person as a stranger.

7. Design and Implementation Tools

Hardware: MacBook Pro IDE: Eclipse MARS.1, Eclipse Neon

Third-party library: OpenCV3.10, Jama, libsvm(Only for face recognition system)

8. Discussion

8.1 Our Observations

1) The kernel functions can increase the accuracy of SVM: the accuracy is improved from 74% to 91% after introducing linear kernel functions;
2) Different kernel functions have different effects on the accuracy of SVM.

8.2 What are Interesting

1) The HARR features used by OpenCV to detect faces;
2) Better ideas of extending SVM and logistic regression to multiple classification;
3) Better ideas of distinguishing strangers.

8.3 Implications of the Results and Experiments:

1) KNN is not quite suitable for face recognition;
2) SVM algorithm cannot work well for unbalanced dataset;
3) Logistic regression is easy to be under-fitting.

9. Conclusion and Future Work

9.1 Deduced Conclusions

1) In PCA, it's better to decide the number of kept eigenvalues by specifying the percentage of information we want to reserve than decide the number subjectively.
2) KNN has no procedure of training. It's not feasible for large dataset because of huge demand of time and memory. SVM and logistic regression cost plenty of time to train models, but consume little classifying time, so these 2 are practical for large dataset.

9.2 Future Improvement

1) We want to find a good solution of distinguishing strangers from registered users. Our current method is a little too rough, and can be a future improvement;
2) We didn't consider the intensity of the sunlight in our face recognition system. Some papers gave some ideas to handle the problem. We can refer to these algorithms later.

3) The face recognition system can be improved to 3D face recognition.

10. User Manual

1) Before running the program, please make sure you've configured the environment by https://drive.google.com/open?id=19lsEo_ZTyLyOSgRnN_57WdajSG0HLF_sMNS7tqkg1XE (Please copy it and paste in your address blank).
2) For the implementation of KNN, SVM and logistic regression algorithms by ourselves, please click "Basic Algorithms Implemented By Ourselves" button:
   a. Click "Training Image Process" button to process training images with PCA.
   b. Click "Testing Image Process" button to process test images with PCA.
   c. Click "KNN" button to run the KNN algorithm, the outputs on the right side are the errors, the outputs in the Console window are the complete output.
   d. Click "SVM Model Training" button to train the SVM models which are saved as files and can be reused.
   e. Click the "SVM" button to run the SVM algorithm.
   f. Click the "Logistic Regression Model training" button.
   g. Click the "Logistic Regression" button.
3) For the face recognition system based on OpenCV. If you want to register new users, please click "Register Face". It is required to register 10 images for each new user. Please remember to retrain your models by clicking "Training" button. If you want to recognize a user, please click "Recognize Face" button.

11. References

[1] Ebrahimpour, Hossein, and Abbas Kouzani. "Face Recognition Using Bagging KNN." International Conference on Signal Processing and Communication Systems (ICSPCS'2007) Australia, Gold Coast. 2007.

[2] Zhou, Changjun, et al. "Face recognition based on PCA and logistic regression analysis." Optik-International Journal for Light and Electron Optics125.20 (2014): 5916-5919.

[3] Guo, Guodong, Stan Z. Li, and Kapluk Chan. "Face recognition by support vector machines." Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on. IEEE, 2000.

[4] https://en.wikipedia.org/wiki/Principal_component_analysis

[5] Harrington, Peter. Machine learning in action. Vol. 5. Greenwich, CT: Manning, 2012.

[6] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

12. Team Information, Bio and Overall Statement of Work Allocation

Yunwei Jiang: China, Master candidate of Telecommunication Systems Management, responsible for SVM, Logistic Regression, Utility class and Face recognition system based on OpenCV

Chengjie Zhu: China, Master candidate of Telecommunication Systems Management, responsible for PCA, KNN, image processing

13. Acknowledgement

PCA algorithm refers to the MATLAB code from http://wenku.baidu.com/link?url=5j80IDYdh4SR9xrgxbihQ9Bl7kf8l_DCxA_JJCrlioYbFi2bI4hmDY1_eyi88LFI5G7y7LyfYx4iaonDRwks1gojiPON17mse0TZpnmh0jW. (Please copy it and paste in your address blank).