Single Gaussian vs. Mixture-of-Gaussians Models for Pattern Recognition

by Khoa Do

Abstract: Single-Gaussian and Gaussian-mixture models are utilized in various pattern recognition task including image classification. We implemented a vanilla single-Gaussian and a vanilla mixture-of Gaussian model to classify face images using the images themselves as features. In this report, we compare the ability of the two classifiers to classify face and non-face images using the Face Detection Data Set and Benchmark (FDDB) dataset. Their abilities are evaluated in terms of false-negative rate, false-positive rate, misclassification rate, and Receiver Operating Curve (ROC). We found that the Gaussian-mixture model outperformed the single-Gaussian model.

1. Introduction

Face detection is one of many important tasks in computer vision. The discrete labels are inferred in the dataset to indicate whether a face is presented in each image or not (1 for face and 0 for no face). Through proper training on the observed image data, the computer vision model will be able to perform such task fast, accurately, and efficiently on a large scale compared to a human. This face detection problem boils down to a typical binary classification task where a model classifies a given image as face or non-face. In this paper, we want to explore and experiment on the classic probabilistic normal-distribution classification model, which is foundational to state-of-the-art deep neural networks.

To achieve such goal, we implemented a vanilla single-Gaussian and a vanilla mixture-of Gaussians model to classify face images using the images themselves as features. We then

- Visualized the estimated means and covariance matrices for face and non-face images
- Evaluated the learned models on the test images using a predefined threshold for the posterior by computing false-positive rate, false-negative rate, and misclassification rate
- Plotted the ROC to evaluate the diagnostic ability of the binary classifier

Following along the report, we will introduce the relevant information used in implementing the overall software in section 2, further discuss how section 2 helped implement the software and set up the experiment in section 3, discuss the obtained results from the experiment in section 4, and conclude our work in section 5

2. Related study

2.1 FDDB dataset

The Face Detection Data Set and Benchmark (FDDB), a data set of face regions designed for studying the problem of unconstrained face detection. This data set contains the annotations for 5171 faces in a set of 2845 images taken from the Faces in the Wild data set. The dataset contains the original unannotated set of images and face annotations. The original set of images can be downloaded from http://tamaraberg.com/faceDataset/originalPics.tar.gz. Uncompressing this file organizes the images as **originalPics/year/month/day/big/*.jpg**. The face annotations are split into ten folds. Uncompressing the **FDDB-folds.tgz** file creates a directory **FDDB-folds**, which contains files with names: **FDDB-fold-xx.txt and FDDB-fold-xx-ellipseList.txt**, where xx = {01, 02, ..., 10} represents the fold-index. Each line in the **FDDB-fold-xx.txt** file specifies a path to an image in the above-mentioned data set. For instance, the entry **2002/07/19/big/img_130** corresponds to **originalPics/2002/07/19/big/img_130.jpg**. The corresponding annotations are included in the file **FDDB-fold-xx-ellipseList.txt** in the following format:

<image name i>

<number of faces in this image =im>

The output is the representation of a face depends on the specifics of the shape of the hypothesized image region (rectangular or elliptical) [1].

2.2 K-means clustering

K-means clustering is a distance-based algorithm. It tries to group the closest points together to form a cluster. We first define the number of groups that we want to the divide the population. Based on the desired number of clusters, we then randomly initialize ***k*** centroids. The data points are then assigned to the closest centroid and a cluster is formed. We continue updating the centroids and the data points until the location of centroids no longer changes.

The drawback of k-means clustering is that it will not work out well if the data points does not form a circular shape. The solution to this is using the distribution-based approach instead of the distance-based approach for the k-means clustering [2].

2.3 Expectation maximization

The EM algorithm is a general-purpose tool for finding the right model parameter $\boldsymbol{\vartheta}$. It is used when the data is incomplete (data has missing values). The missing variables are called latent variables and are optimally estimated using the existing data and then finds the model parameter. Based on the parameter, we update the values for the latent variables. The algorithm has two steps: ***E-step*** and ***M-step*** that are repeated for a certain number of times (iterations). In the ***E-step***, we use the available data to estimate the values of latent variables. In the ***M-step***, we update the parameters based on the estimated values [2][3].

2.4 Gaussian distribution

The posterior probability followed by a Gaussian is given by:

$$Pr(\mathbf{x}|w) = \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w] = \frac{1}{\sqrt{2\pi|\Sigma|}}\exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^t\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] \qquad (1)$$

where $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$ are the mean and covariance for class $\boldsymbol{w}$ [2][3].

3. Experimental set-up

To set up the experiment, we followed this software pipeline: data preparation, models implementation, and visualization. The software was developed in Python programming language using Jupyter Notebook.

3.1 Data preparation

We wrote the **data_extracter.ipynb** to extract 10x10 images of face and non-face from the FDDB dataset. For face images, the annotated elliptical coordinates were extracted and transformed into x-y coordinates of the top-left and bottom-right corner of rectangular bounding boxes. The bounding boxes were then cropped out of the images to form the **posImages** folder that contained all face images. For non-face images, the bottom-right x-y coordinate was kept and the top-left x-y coordinate was randomly initialized to formed bounding boxes that contained 30% less than the face's ground truth. The bounding boxes were then cropped out of the images to form the **negImages** folder that contained all non-face images. The extraction-from-annotation algorithm is quite popular. A lot of open-source codes are available online. However, we referred to the general algorithm in [4] to our purpose-specific FDDB dataset.

To form the dataset for training and testing our models, we carefully and manually picked 1000 images each of face and non-face images for training set and 100 images each of face and non-face for test set to ensure the best image quality. No images in the training set appeared in the test set. In total, there were about 2,200 images used for the experiment and 10x10 images saved a tremendous amount of time in training the models.

Figure 1. Sample of an original face image.

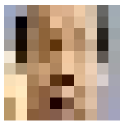

Figure 2. Sample of a processed face image.

3.2 Models implementation

The **models.ipynb** was implemented from scratch containing several cells including the dataset loader cell, single-Gaussian model cell, MoG model cell, and rate calculation & ROC plotting cell (which is a part of the visualization in section 3.3). The data loader converts the 10x10 image to grayscale and flattened it into a 1-D array of 100 elements (10 pixels). Therefore, the image's pixels (intensities) themselves are the features to be fed to the models.

3.2.1 SG model

This model assumes the entire training data follows a single Gaussian distribution. The parameters can be learned from the underlying data itself without the use of the EM algorithm. We used the **parameters** function to calculate the mean and covariance of the entire face and non-face images in the training set following the formulation shown in equation 2 [3].

$$\begin{aligned} Pr(\mathbf{x}|w=0) &= \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0] \\ Pr(\mathbf{x}|w=1) &= \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1]. \end{aligned} \quad (2)$$

where 1 is face and 0 is non-face.

The mean and covariance were then used to calculate the normal distribution for face and non-face returned log-likelihood PDF with the **Norm** function on the test set. The log-likelihood PDF was then used to assign labels to test images using the formulation for the **Posterior** function shown in equation 3 [3].

$$Pr(w=1|\mathbf{x}) = \frac{Pr(\mathbf{x}|w=1)Pr(w=1)}{\sum_{k=0}^{1} Pr(\mathbf{x}|w=k)Pr(w=k)}. \quad (3)$$

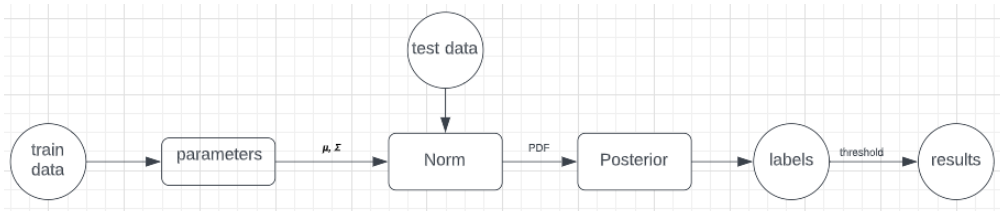

Figure 3. SG algorithm flowchart.

3.2.2 MoG model

The mixture of Gaussians (MoG) is a prototypical example for the EM algorithm, where the data are described as a weighted sum of *K* normal distributions (or ***K*** clusters of normal distribution that we had mentioned in section 2.2):

$$Pr(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \lambda_k \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k], \qquad (4)$$

where $\boldsymbol{\mu}_{1\ldots K}$ and $\boldsymbol{\Sigma}_{1\ldots K}$ are the means and covariances of the normal distributions and $\boldsymbol{\lambda}_{1\ldots K}$ are positive valued weights that sum to one [3].

Like the single-Gaussian model, we defined the **parameters**, **Norm**, and **Posterior** functions. However, we initialized the parameters $\boldsymbol{\lambda}$, mean, and covariance based on ***K*** clusters (we chose 4 clusters) and assigned random values to them instead of calculating the values from the training data in the **parameters** function. For the **Norm** function, we followed the formulation shown in equation 4. It was very similar to one of the single-Gaussian model except that we needed to multiply the normal distribution by $\boldsymbol{\lambda}_k$.

We are now in the ***E-step***. From the initialized λ, mean, and covariance, for each cluster, we calculate the likelihood using the training set for face and non-face and the PDF following equation 4. We then calculated the probability of a sample point $\boldsymbol{r}_{ik}$ that belongs to a cluster/distribution using equation 5 [3].

$$\frac{\lambda_k \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]}{\sum_{j=1}^{K} \lambda_j \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j]} = r_{ik}. \qquad (5)$$

With the estimated $\boldsymbol{r}_{ik}$, new $\boldsymbol{\lambda}$, mean, and covariance of that cluster/distribution were updated following equation 6 [3] in the ***M-step***.

$$\begin{aligned}
\lambda_k^{[t+1]} &= \frac{\sum_{i=1}^{I} r_{ik}}{\sum_{j=1}^{K}\sum_{i=1}^{I} r_{ij}} \\
\boldsymbol{\mu}_k^{[t+1]} &= \frac{\sum_{i=1}^{I} r_{ik}\mathbf{x}_i}{\sum_{i=1}^{I} r_{ik}} \\
\boldsymbol{\Sigma}_k^{[t+1]} &= \frac{\sum_{i=1}^{I} r_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k^{[t+1]})(\mathbf{x}_i - \boldsymbol{\mu}_k^{[t+1]})^T}{\sum_{i=1}^{I} r_{ik}}
\end{aligned} \qquad (6)$$

The EM algorithm was repeated for eight iterations before we moved on to calculating the normal distribution for face and non-face and returned log-likelihood PDF with the **Norm** function on the test set. The log-likelihood PDF was then used to assign labels to test images using the formulation for the **Posterior** function. The logic was very similar to the single-Gaussian model.

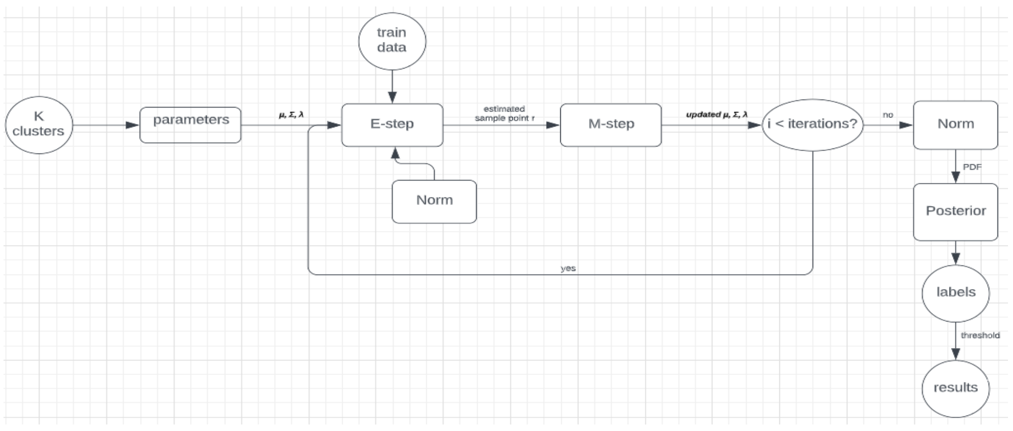

Figure 4. MoG algorithm flowchart.

3.3 Visualization

The visualization part of coding was built in **models.ipynb** due to its convenience. In this part, we computed the true-positive (TP) rate, true-negative (TN) rate, false-positive (FP) rate, false-negative (FN) rate, and plotted the ROC for the discussion in section 4. TP and TN followed a simple likelihood ratio test, whereas a threshold of 0.55 was used to determine FP and FN.

4. Results and discussion

Upon completing the experiment in section 3. We obtained the following:

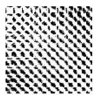

Figure 5. SG cov., face.

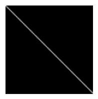

Figure 7. MoG cov., face.

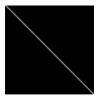

Figure 6. SG cov., non-face.

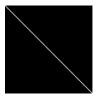

Figure 8. MoG cov., non-face.

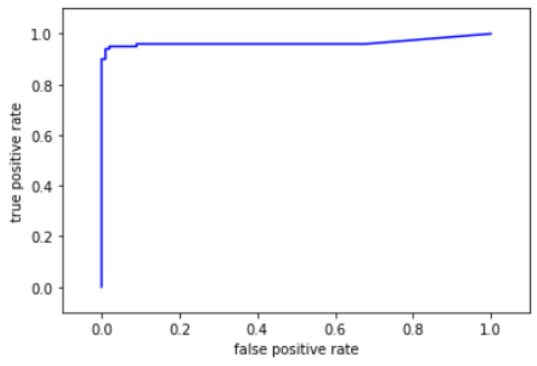

Figure 9. SG ROC.

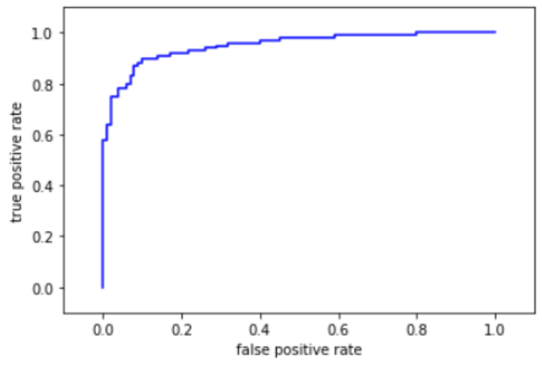

Figure 10. MoG ROC.

--------------------------	SG	MG
True-Positive Rate	0.03	0.32
True-Negative Rate	0.88	0.68
False-Positive Rate	0.56	0.13
False-Negative Rate	0.18	0.21
Misclassification Rate	0.37	0.17

Table 1. Model Performance Comparison

Figure 4 was supposed to show a diagonal covariance matrix, but it did not. For this reason, the SG model did not learn well for face images. As a result, the TP rate was only 3% while it assumed pretty much all images as non-face (88%) and the misclassification rate was very high (37%). Or it may just predict all correct for non-face images. On the other hand, the MoG model performed way better in terms of misclassification rate (17%). Although its TN rate went down to 68%, it generalized better for face images (32% in TP rate) due to multiple Gaussian distributions that could handle outliers. In the experiment in section 3, we tried different combinations of the number of clusters, number of iterations, and threshold values but the relative combinations of 4, 9, and 0.55 gave the lowest misclassification rate for the two models.

5. Conclusion

In this report, we showed that the performance of the MoG model was better than the SG model on the 10x10 grayscale images in the FDDB dataset for face and non-face. The MoG model had the lower misclassification rate and seemed to generalize and handle outliers better between the two models while the MG model was biased toward the non-face images. The combinations of 4, 9, and 0.55 of the number of clusters, number of iterations, and threshold values relatively gave the lowest misclassification rate for the two models. There were several ways of implementing the EM algorithm such as assuming all covariance matrices were same and/or diagonal. For this work, we assume all covariance matrices are different. For future improvement, we will experiment the two models on larger images such as 20x20 or 60x60 RGB instead of 10x10 grayscale images and think of other criteria for the features used other than just the pixels (intensities) of the images.

6. References

[1] Vidit Jain and Erik Learned-Miller. FDDB: A Benchmark for Face Detection in Unconstrained Settings. *Technical Report UM-CS-2010-009, Dept. of Computer Science, University of Massachusetts, Amherst*. 2010.

[2] A. Singh, “Build Better and Accurate Clusters with Gaussian Mixture Models,” Analytics Vidhya, 31-Oct-2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/. [Accessed: 11-Apr-2022].

[3] Prince, S.J., 2012. Computer vision: models, learning, and inference. Cambridge University Press.

[4] A. Rosebrock, “Intersection over union (IOU) for object detection,” PyImageSearch, 07-Nov-2016. [Online]. Available: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. [Accessed: 11-Apr-2022].