University of New South Wales

COMP4121 - Final Report

Online Signature Verification System using HMM

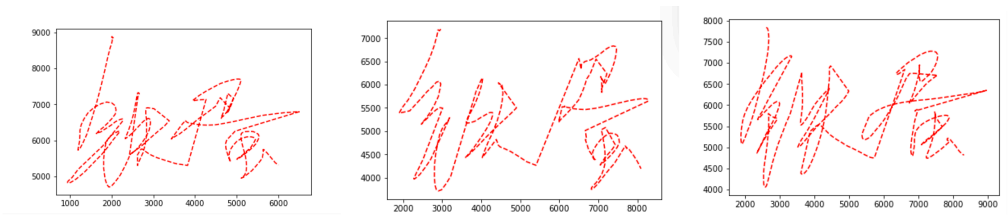
Qianrui Zhao | z5088359

November 20, 2017

Contents

1 Introduction

In recent decades, greater and greater number of people now prefer to manage their financial accounts via online applications to traditional bankbooks and cheques. While in a digital transaction, how could we make sure that the person we paying for is the one we want to pay? A simple answer is signature, where one's writing behaviors are embedded. However, it is very difficult for human-beings to verify signatures by their eyes, especially those imitated by skilled forgeries. For instance, figure 1 shows a pair of genuine and highly skilled counterfeit signatures, the difference between them is not obvious from appearance. The good new is that by using advanced machine learning technologies, our 'smart' digital devices do much better verification work than us. In this report, we introduced an accurate and efficient approach of automatically verifying online signatures using hidden markov model (HMM).



(a) valid sign (b) valid sign (c) counterfeit sign

Figure 1: A group of true and fake signatures

HMM is a statistical Markov model in which the state is not fully observable. It is widely applied in many industrial areas such as speech recognition and informatics. The following sections demonstrates how the signature problem is solved by a HMM-based method.

1.1 Dataset

The data set used for training and testing in this project comes from the first international signature verification competition, which is held for comparing various signature verification approaches systematically. According to Yeung [citation], two signature databases are provided to participants: the first dataset only has coordinate numbers and the second one contains additional information such as pressure and pen orientation.

Because the competition is already finished, we now can only access to partial the database. We used the second dataset which has 40 sets of signature data. Each set consists of 20 true signature from the same users, and 20 fake signatures which is pretended by than four other people.

In this dataset, each signature contains a sequence of samples and the number of them. One single sample has 5 elements:

- coordinate_x
- coordinate_y
- azimuth
- altitude
- pressure

1.2 Markov model

Before go to HMM, in this section we first to introduce what is a Markov model. A discrete time Markov model is called Markov chain. A finite Markov chain consists of a set of N states and a $N \times N$ size transition matrix in which element in the i^{th} row and j^{th} row represents the probability p_{ij} of changing from state i to state j. The probability p_{ij} only depends on state i, previous states have no effect on p_{ij}.

Moreover, if a finite Markov chain is irreducible and aperiodic, it will eventually converge to a unique probability distribution [1]. Based on this property, we can use the converged distribution to do the predictions.

As mentioned in previous, HMM is a partially observable Markov model. The implementation section of this paper will show how it is applied in signature verify problem.

1.3 Performance evaluation approach

Confusion matrix, as shown in table 1 is used for evaluating performance of a verification system. Based on confusion matrix, False Acceptance Rate (FAR) and False Rejection Rate (FRR) can be computed. The value of FRR and FAR can be adjusted by changing threshold, however, lowest FFR and FAR can not be achieve at the same time. They are always against each other, as in figure 2. Accuracy requirements of a biometric system are application dependent. For example, a bank verification system requires low FAR because the damage of accepting a wrong answer is huge, by the contrast, FRR are not that important; as regard a medical system, FRR is must more important than FAR, fail to response a potential user may cause medical accident.

	predict TRUE	predict FALSE
Actual TRUE	TP	FN
Actual FALSE	FP	TN

Table 1: confusion matrix

- FRR: The ratio of unaccepted genuine test signatures to the total genuine test signatures.

$$FRR = FN/(FN + TP)$$

- FAR: The ratio of accepted forgeries to the total submitted forgeries is called FRR.

$$FAR = FP/(FP + TN)$$

2 Overall design

The evaluation approach used in SVC competition is to compare Error Equal Rate (EER), which is the average of FAR and FRR. We use the same way to judge performance of our system. Using EER means the signature verification system should not be either too sensitive or too coarse.

3 Overall design

Figure 3 illustrates the overall steps of building a signature verification system. First of all, a group of features are extracted from each sample so that it can be considered as a multi-dimension variable.

In the second step, we did some pre-processing on data. so that some noises are eliminated and the calculation is simplified. The noises are eliminated by

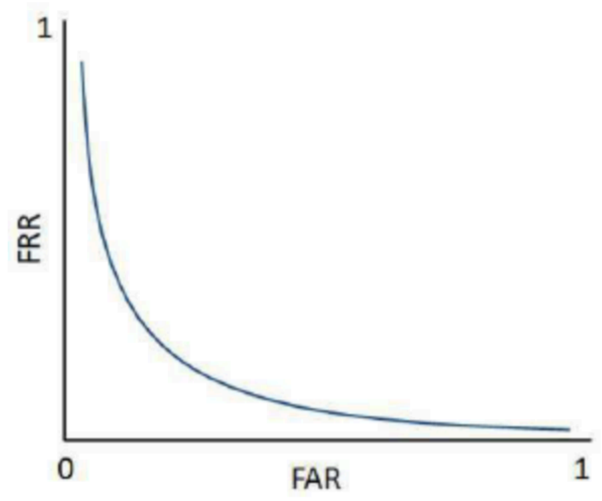


Figure 2: An example receiver operating characteristic curve (ROC)

smoothing data. Moreover, data is normalized so that in learning phase we do not need to deal with very huge values.

At the next stage, a HMM is built for every signature user on basis of training data. The HMM model is estimated by Baum–Welch algorithm.

Eventually, a similarity score between a input signature and the HMM just built is computed using Viterbi Algorithm. Here we need a threshold to determine if the signature is valid.

More details about each steps are explained in next section.

4 Implementation

4.1 Features extraction

In given dataset, one signature is represented by 5 attributes. Besides, because we want to dig more critical information about a signature, 2 more implicit features are created based on current data [2]:

Basic features:

- coordination x x

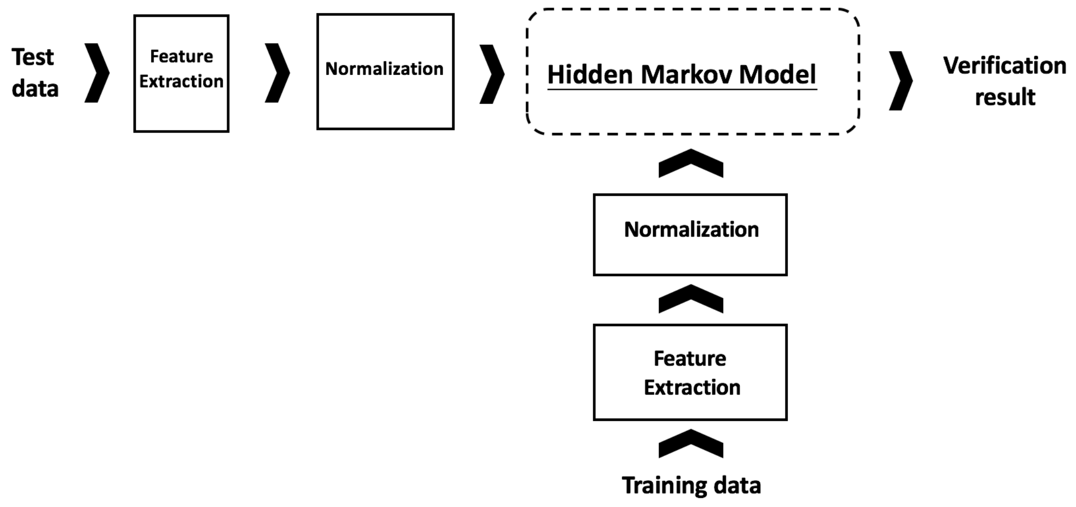


Figure 3: Overall design of HMM online signature verification system

- coordination y y
- altitude of pen ϕ
- pressure p
- azimuth λ

Extended features

- angle of path θ

$$\theta_n = arctan(y_n/x_n)$$

- velocity v

$$v_n = \sqrt[2]{x_n^2 + y_n^2}$$

However, during the evaluation phase we find that not all features have a good impact on system performance. In practice, azimuth λ and velocity v cause a decline in verification accuracy. Therefore, these 2 features are removed from our feature set. A signature in out system is represented by 5 features:

$$\{x, t, \phi, p, \theta\}$$

Figure 4 demonstrates the decomposition of a group of genuine and fake signature into 5 dimensions including coordination x, y, altitude, pressure and the angle of path.

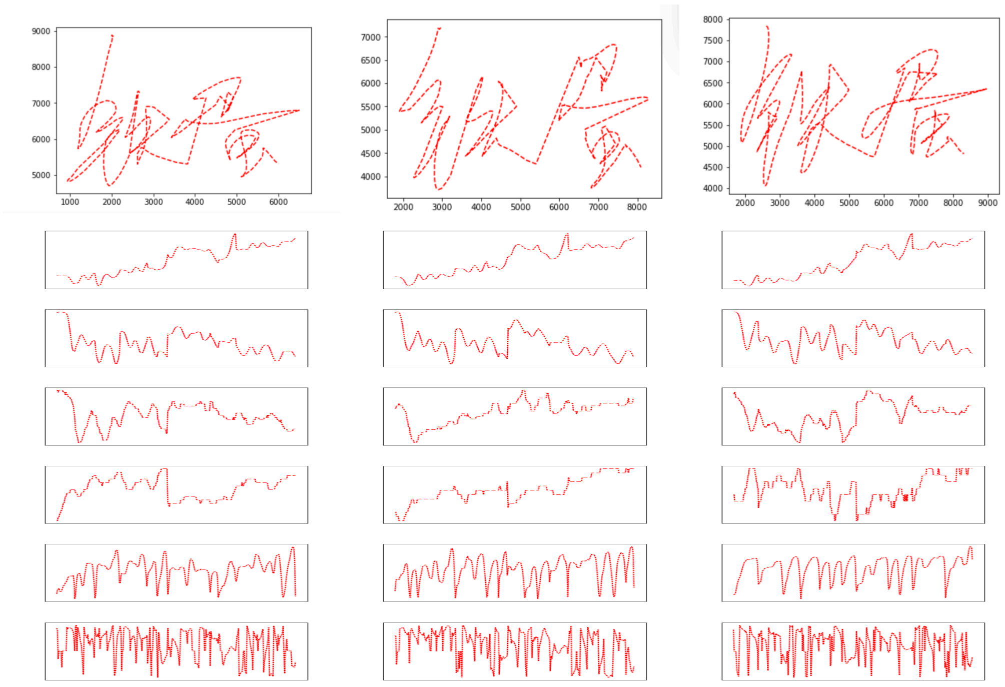


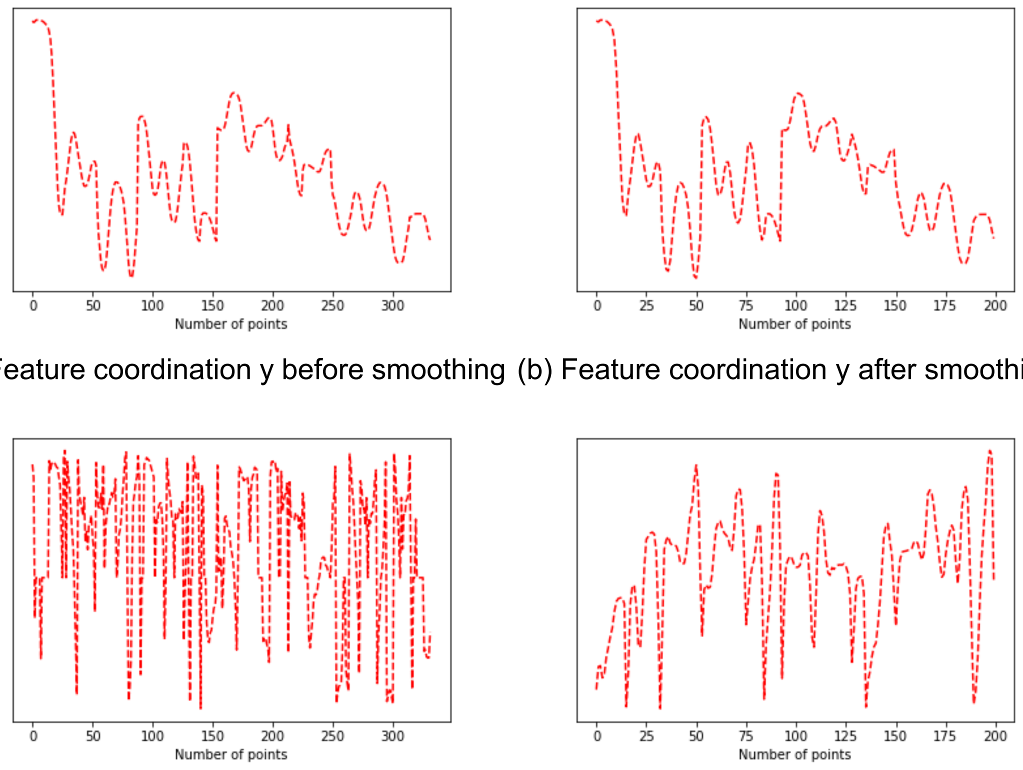
Figure 4: The group of true and fake signatures introduced in section 1 and changes of their 5 features in terms of time

4.2 Data pre-processing

Before build the model, some preprocessing need to be done on input samples. According to our observation on dataset, a signature file has a sequence of N samples, but the value N are various among different signatures. For example, the samples size of 20 valid signatures for user 1 varies from 84 to 110. For this reason, we scale input samples into a consistent size by linearly interpolating new values. In a signature, K number of new points are inserted into every two original data, then from the new sequence, which has length $K \times (N-1)$, a point is extracted for every N points. Finally we get a series of K data, in order words, we scale a sample with size of N to K:

$$\{\hat{O}_1, \hat{O}_2, \cdots, \hat{O}_K\}$$

Moreover, a branch of discrete values are used to simulate a signature, which is naturally continuous. As shown in Figure 4, original data shows sharp curves because the discrete is not precise enough to show the real tendency. The noise generated by representing continuous value by discrete samples can be reduced by data interpolation as well. Interpolating data actually reveals the real tendency by smoothing the samples, as shown in figure 5, which reflects the smoothing effect of linear data interpolation.



(a) Feature coordination y before smoothing (b) Feature coordination y after smoothing

(c) Feature path angle before smoothing (d) Feature path angle after smoothing

Figure 5: Features of a signature before and after perform smoothing operation

After interpolating and scaling data, the samples are further normalized in order to attain zero mean and unit standard deviation values, so that our calculation will not involve huge numbers. $\hat{O}_k$ is replaced with O_k. Σ is the diagonal covariance matrix and μ is the mean value. We obtain a sequence of data, which is called observations:

$$O_k = \Sigma^{-1/2}(\hat{O}_k - \mu)$$

4.3 HMM construction

The state is not fully observed in HMM, therefore, in a hidden Markov-chain, there is a sequence of observations. However, the observations can not represent the actual state. A mapping between observations and states is necessary. Figure shows a hidden Markov-chain with states and observations.

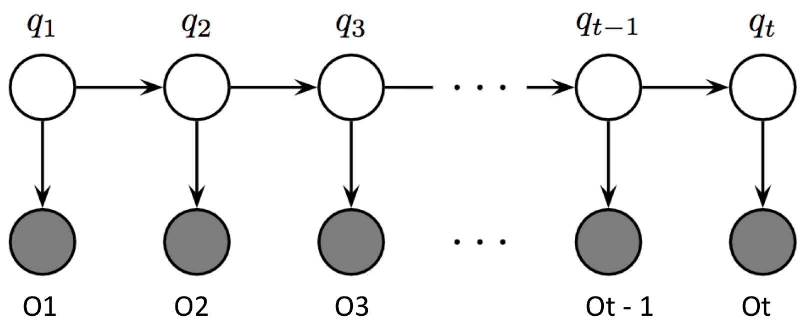


Figure 6: A hidden Markov-chain

Suppose a hidden Markov chain containing T observations $\{O_1, O_2, \cdots, O_T\}$, there are actually $T-1$ transitions from state q_1 to q_T. As introduced in Introduction section, the Hidden Markov model consists of:

- a set of H states $\{S_1, S_2, \cdots, S_H\}$
- transition probability matrix $A = \{a_{ij}\}$ where
$$a_{ij} = P(q_{n+1} = S_j | q_n = s_i),\, 1 \leq i, j \leq H$$
- emission functions, which is a sequence of H Gaussian distributed functions calculating possibility of an observation belongs to each state. The distribution function is characterized by its mean and diagonal covariance matrix $\{\mu, \Sigma\}$
- discrete emission matrix $B = \{b_{ij}\}$ where b_{ij} represents the probability density matrix of mapping training observation i to states j.
- initial distribution of H states $\{\pi_1, \pi_2, \cdots, \pi_H\}$

STEP 01: Build initial model

1. Initial states
   H states are created based on training data. The classification is done by K-means algorithm [3]:

```
cluster_means[M] = []  // Array stores means of clusters
x[N] // Array stores samples to be clustered
b[N] // Array maps a sample to a cluster

Choose arbitrary initial estimates M values for cluster_means
While (any changes among clusters in last iteration)
For i = 1 to N
     Determine the closest representative, for x[i],
        say cluster_means[m], Set b[i] = m

    For j = 1 to M
     Parameter updating: Determine cluster_means[j] as the mean
        of the vectors x[i] with b[i] = j
```

2. Initial Gaussian emission function and discrete emission matrix
   After the states are initialized, all training observations are categorized into H clusters. For each cluster we can get a distribution of data. Because there are more than one feature in the system, multivariate Gaussian distribution should be used instead of normal approach to combine distributions within several dimensions. We first compute mean μ_h and diagonal covariance Σ_h for each cluster h. The emission function is:

$$P(o_i|\mu_j, \Sigma_j) = \frac{1}{\sqrt{2\pi|\Sigma|}} \text{exp}^{-\frac{1}{2}(o_i-\mu)^T\Sigma^{-1}(o_k-\mu)}$$

   Furthermore, b_{ij} in B is calculated as:

$$b_{i,j} = P(o_i|\mu_j, \Sigma_j)$$

3. Initial transition matrix
   Yang [4] describes 5 types of It concludes that among all those topologies, model and parallel model shows the highest overall performance. The parallel model is implemented in our project, in which from current state, every state can be the next one to reach (including current state).

   At the beginning, we assumes all transitions have the same probabilities. In other words, the probabilities of jump from one state to H states are equally same. Therefore, elements in the transition matrix A are initialized as $1/H$.

$$a_{ij} = 1/H, 1 \leq i \leq H$$

4. Initial state distribution
   Because at the beginning we consider all transitions have the same probabilities, the initial state distribution is:

$$\pi_i = 1/H, 1 \leq i \leq H$$

STEP 02: Estimate model
We want to find a model that the most fit the training dataset, which means it is the model provided maximum likelihood. From the initial model, a new model is generate which offers a larger likelihood. By iterating this approach until the increase between two iterations is small enough, we can obtain the best-fit model. This method is stated as the Baum–Welch algorithm in [2]. The Baum–Welch algorithm uses the well known EM algorithm to find the maximum likelihood estimate of the parameters of a hidden Markov model given a set of observed feature vectors.

Baum–Welch algorithm:

- Algorithm outline

```
max_iter = N  // limit compute iterations
while (new_est - est) > threshold or iter > max_iter
iter ++
    preform Expectation step
    preform Maximum step
```

- Argument initialization
  $\{\pi, A, B, \{\mu, \Sigma\}\}$ are initialized in previous introduction. Besides, we create four more variables $\{\alpha, \beta, \xi, \gamma\}$ for forwarding-backwarding computation. ξ and γ are initilized as 0. For forward (α) and backward (β) probabilities:

$$\alpha_1(j) = \pi_j b_j(o_1)$$

all β are 1

- Expectation step

$$\alpha_t(j) = (\Sigma_1^H \alpha_{t-1}(j) a_{ij}) b_j(o_t)$$

$$\beta_t(j) = \Sigma_1^H a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\Sigma_{i=1}^H \Sigma_{j=1}^H \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}$$

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\Sigma_1^H \alpha_t(i)\beta_t(i)}$$

- Maximum step

$$\pi_i = \gamma^1(i)$$

$$a_{ij} = \frac{\Sigma_{t=1}^T \xi_t(i,j)}{\Sigma_{t=1}^T \gamma_t(i)} \text{, for } 1 \le i \le H$$

$$\mu_j = \frac{\Sigma_{t=1}^T o_t \gamma_t(j)}{\Sigma_{t=1}^T \gamma_t(j)}$$

$$\Sigma_j = \frac{\Sigma_{t=1}^T (o_t - \mu_j)^2 \gamma_t(j)}{\Sigma t = 1^T \gamma_t(j)}$$

$$b_j(o_j) = P(o_i | \mu_j, \Sigma_j)$$

4.4 Signature verification

After get the estimated model, we can verify a signature. Assume after feature extraction and data normalization we get a series of observations $\{y_1, y_2, \cdots, y_K\}$, which represents the signature we are going to verify. We use Viterbi Algorithm [5], which is actually a dynamic programming approach, to find out the a sequence of states such that the probability of observing $\{y_1, y_2, \cdots, y_K\}$ is the largest. This biggest probability is the similarity score of this signature.

Viterbi Algorithm is as following:

$$L(1, j) = \pi_j \times b_j(o_1)$$
$$L(t, j) = max(L(j-1, i) \times a_{ij} \times b_j(o_t)$$

Finally, $maxL(T, j)\ 1 \leq j \leq H$ is th similarity score ω.

The last step is to find a proper threshold ϵ so that:

- a signature is genuine if its similarity score $\omega \geq \epsilon$
- a signature is invalid if $\omega \leq \epsilon$

The selection of ϵ is non-trival. If the genuine signatures' scores are significantly different from the forgeries' scores, we can just put a threshold between them so that all genuine and invalid signatures located in different sides. However, in reality their distribution often overlap with each other. Figure 7 shows that placing a threshold in such case contributes to a FRR and a FAR. Recall FRR and FAR introduced in the first section, the value of FRR declines while the FAR increase, and vice versa. Therefore it is actually a trade-off: in some scenario one is preferred than the other one. But because in this system we only focus on EER – the average of FRR and FAR, we want to find a ϵ so that the average, in other words the sum, of FRR and FAR is minimal.

Assume $\omega = mean + z * variance$, the best threshold is caught by attempting different values of z.

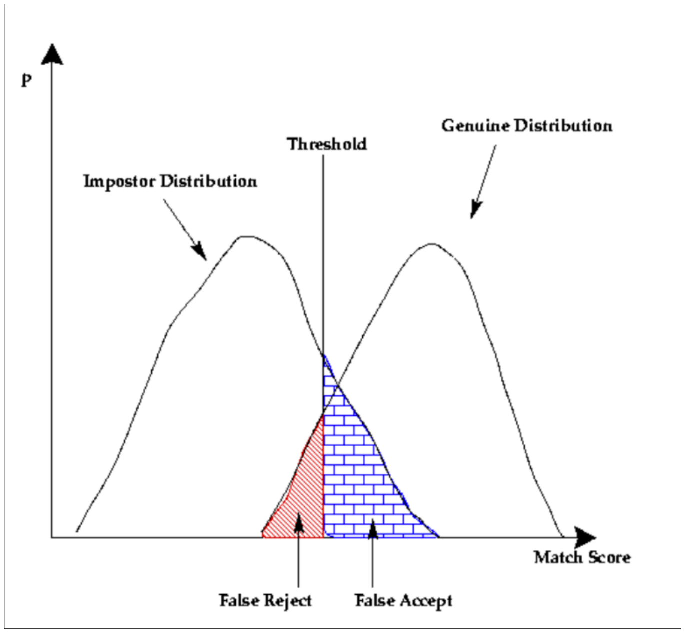


Figure 7: The relationship between threshold and FRR, FAR [6]

5 Experiment

Some parameters existing in our system such as z for computing threshold and H for number of states. In evaluation phase, we are going to find the most suitable values for H and z.

There are 40 sets of signatures in our dataset and each of them contains 20 genuine samples and 20 counterfeit samples. 20 genuine signatures have to be used for both training and testing FRR so we use 75% of them as training data, the rest of them are reserved for testing. For each set, we build a HMM based on first 15 true signatures. FRR and FAR are generated based on verification result of 5 genuine and 20 counterfeit signatures.

5.1 Threshold parameters z

The overall EER is the average of the ERRs of all the 40 sets, each set is evaluated with its own HMM but using the same parameters. EER is the average of FRR and FAR, it reaches minimum when FRR and FAR are equally same. However, it is difficult to find exactly the crossing point of two curves. Therefore, we measure values of FRR and FAR using different thresholds with interval = 0.1, as in figure 8. The threshold with minimal difference between FRR and FAR is considered as the optimal one. which are various under models with different z values. crosses at different threshold, figure 8 shows an example of their tendencies.

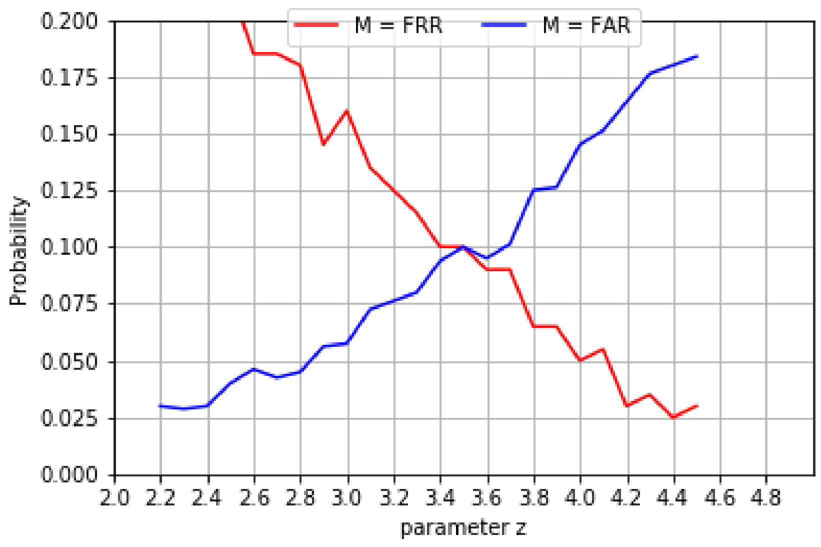
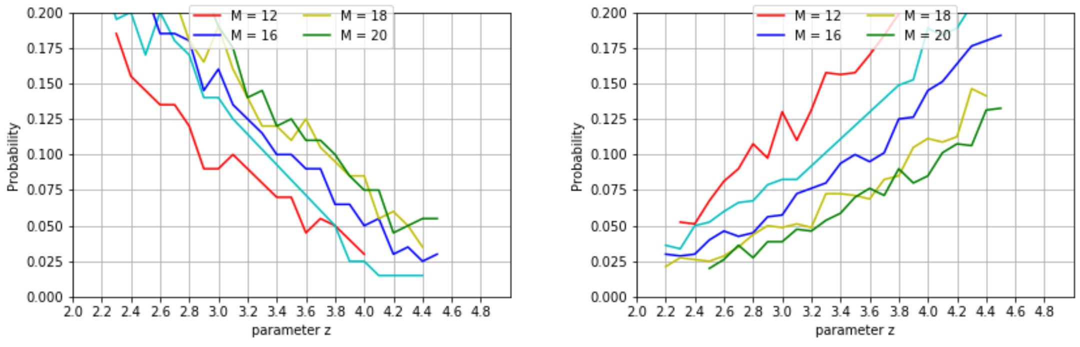


Figure 8: FRR and FAR in models (H = 16) with different thresholds. The crossing point of FRR and FAR curves is the value of actual EER

5.2 Number of states H

To find the optimal value for H, we attempted $H = \{12, 14, 16, 18, 20\}$ and get graph 9. From the figure we can see that the accuracy get improved when increasing H value, but the reduction of error rate is decreasing. When H reach 18, the improvement almost stop. More details about EER is shown in table 2. When H equals to 20, the error rate of our system is only 8.2%.



(a) FRR (b) FAR

Figure 9: FAR and FRR with different H

H	ERR
12	0.11375
14	0.10375
16	0.09999
18	0.086875
20	0.082

Table 2: EER with different H values

6 Conclusion

This report represented a Gaussian HMM-based signature verification system. It provides a high accuracy even if with small amount of training data (only 15 samples for training a model).

Final evaluation shows that our implementation achieved accuracy greater than 90%.

7 References

[1] A. Ignjatovic. (2017). The pagerank, markov chains and the perron frobenius theory, [Online]. Available: `http://www.cse.unsw.edu.au/%7Ecs4121/COMP4121_Lecture_Notes_1.pdf` (visited on 2017).

[2] J. Fierrez, J. Ortega-Garcia, D. Ramos, and J. Gonzalez-Rodriguez, "Hmm-based on-line signature verification: Feature extraction and signature modeling", *Pattern recognition letters*, vol. 28, no. 16, pp. 2325–2334, 2007.

[3] S. Theodoridis, K. Koutroumbas, *et al.*, "Pattern recognition", *IEEE Transactions on Neural Networks*, vol. 19, no. 2, p. 376, 2008.

[4] L. Yang, B. Widjaja, and R. Prasad, "Application of hidden markov models for signature verification", *Pattern recognition*, vol. 28, no. 2, pp. 161–170, 1995.

[5] A. Ignjatovic. (2017). The hidden markov models and the viterbi algorithm, [Online]. Available: `http://www.cse.unsw.edu.au/%7Ecs4121/COMP4121_Lecture_Notes_2.pdf` (visited on 2017).

[6] U. Biometrics. (2011). Pattern classification, [Online]. Available: `https://cseweb.ucsd.edu/classes/wi12/cse190-c/lec3.pdf` (visited on 2017).