CSE250B Homework 3

Qiao Zhang

February 3, 2016

1 Bivariate Gaussian

1.1

$$\mu = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 1 & -0.25 \\ -0.25 & 0.25 \end{bmatrix}$$

1.2

$$\mu = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

2 More bivariate Gaussian

2.1

```
In [6]: x, y = np.random.multivariate_normal(mean, covariance, 100).T
        plt.scatter(x, y)
        plt.axis('equal')
        plt.show()
```

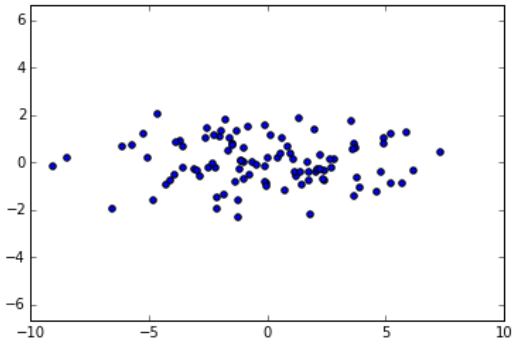

Figure 1: random sample from Gaussian

2.2

```
In [7]: mean = [0,0]
        covariance = [[1,-0.75],[-0.75,1]]
        x, y = np.random.multivariate_normal(mean, covariance, 100).T
        plt.scatter(x, y)
        plt.axis('equal')
        plt.show()
```

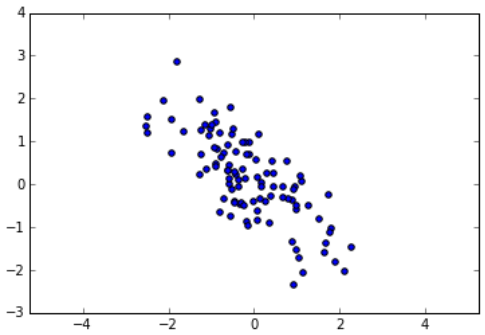

Figure 2: random sample from Gaussian

3 Linear classification

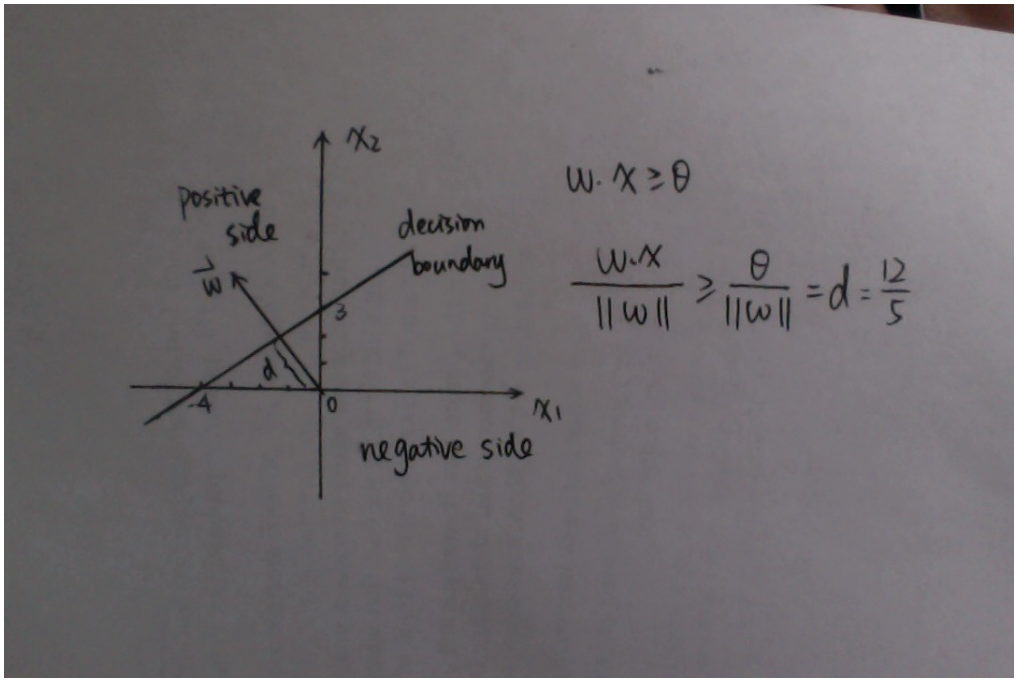

Figure 3: decision boundary plot

4 Eigen-decomposition of a covariance matrix

4.1

According to Invertible Matrix Theorem, $\forall$ p $\times$ p matrix A is invertible $\iff$ 0 is not an eigenvalue of A.
Therefore, Σ is invertible $\iff \forall i \quad \lambda_i \neq 0$
Since covariance matrices should be P.S.D, Σ is invertible $\iff \forall i \quad \lambda_i > 0$
We can tell whether Σ is invertible simply by inspecting the eigenvalues.

4.2

$\because \forall i \quad \Sigma u_i = \lambda_i u_i$
$\therefore \forall i \quad (\Sigma + cI)u_i = (\lambda_i + c)u_i$
Thus the eigenvalues of $\Sigma + cI$ are $\lambda_1 + c, ..., \lambda_p + c$ and the eigenvectors are $u_1, ..., u_p$.

4.3

$$Q = \begin{bmatrix} u_1 & u_2 & ... & u_p \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & \lambda_2 & ... & 0 \\ 0 & 0 & ... & 0 \\ 0 & 0 & ... & \lambda_p \end{bmatrix}$$

Given $\Sigma = Q\Lambda Q^T$, we can tell that $\Sigma^{-1} = Q\Lambda^{-1}Q^T$, which is proved in class.
Thus the eigenvalues of Λ^{-1} are $\lambda_1^{-1}, ..., \lambda_p^{-1}$ and the eigenvectors are $u_1, ..., u_p$.

4.4 reference

http://www.math.harvard.edu/archive/20_spring_05/handouts/ch05_notes.pdf

5 Handwritten digit recognition using a Gaussian generative model

5.1 Pseudo code

Randomly choose 10000 data from train set to split into a smaller train set and a validation set
Separate the training data into 10 categories according to the label
For label = 0:9
    calculate the average μ and covariance matrix Σ
c = designed value
$\Sigma = \Sigma + cI$
calculate the norm and inverse of Σ for each category
For each data in test set
    max_prob = $-\infty$
    pred = -1
    For label = 0:9
        tmp = $log\pi_j + logP_j(x)$
        if tmp $>$ max_prob
            max_prob = tmp
            predict = label

5.2 error rate

The validation error corresponding to different setting of c is shown in the table below to help us determine a perfect model.

c	0	0.1	0.3	0.5	1	5	10
error	0.507	0.037	0.034	0.035	0.041	0.083	0.116

Table 1: Validation error corresponding to c

According to the table above, we choose 0.3 as the value of c for the best model. The error rate on the MNIST test set with c = 3 is 0.041.

5.3

The posterior probabilities are shown in the table below. As a result of underflow, we can not calculate the actual posterior probabilities which is $\frac{\pi_j P_j(x)}{\sum_j \pi_j P_j(x)}$ at one time. Thus we applied exponential to $log\pi_j + logP_j(x)$ and normalize the probability vector instead.
The corresponding test images are also shown respectively below.

k-index	80	1101	2648	5922	9982
0	1.39e-15	2.51e-24	8.19e-1	2.79e-7	8.20e-7
1	1.27e-25	2.74e-43	1.81e-23	3.65e-40	3.09e-19
2	6.74e-13	10.00e-1	1.77e-4	1.08e-9	2.08e-3
3	1.15e-10	4.74e-6	1.15e-2	8.46e-1	9.99e-4
4	2.77e-4	5.96e-11	1.05e-7	7.47e-29	3.15e-13
5	5.85e-9	2.58e-10	1.56e-1	1.54e-1	1.40e-5
6	7.44e-20	3.63e-12	4.49e-13	2.32e-18	9.97e-1
7	1.28e-1	1.79e-21	4.93e-6	2.13e-25	3.87e-19
8	1.30e-1	2.65e-5	2.86e-3	4.24e-12	1.55e-8
9	8.72e-1	1.27e-15	1.04e-2	1.28e-24	3.18e-17

Table 2: posterior probabilities

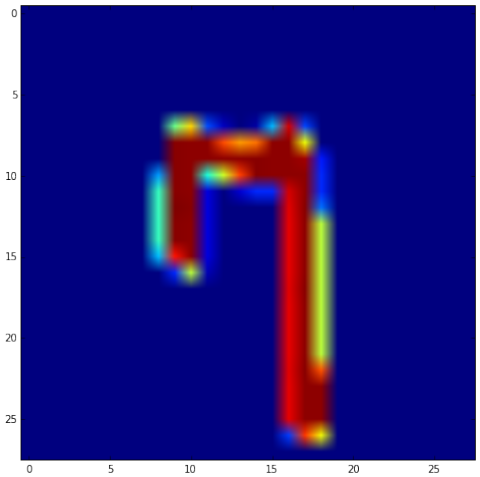

Figure 4: index: 80, label: 7, predict: 9

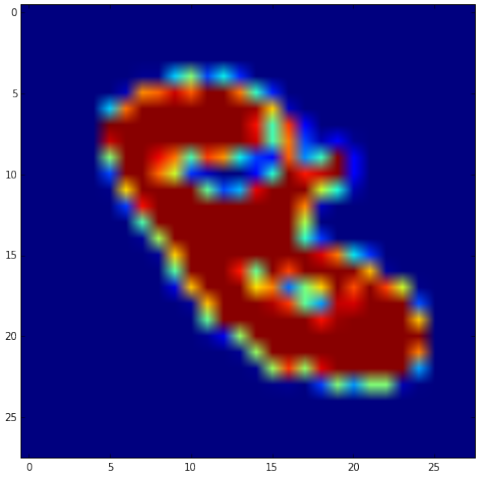

Figure 5: index: 1101, label: 8, predict: 2

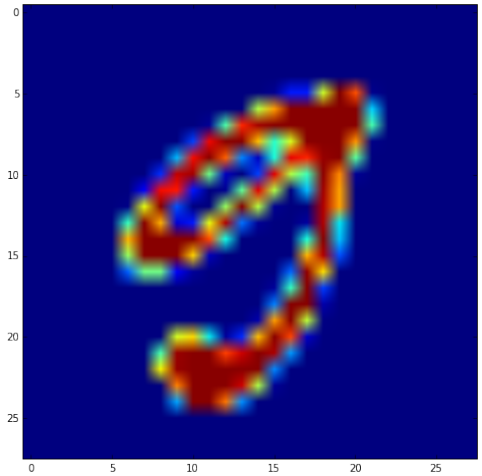

Figure 6: index: 2648, label: 9, predict: 0

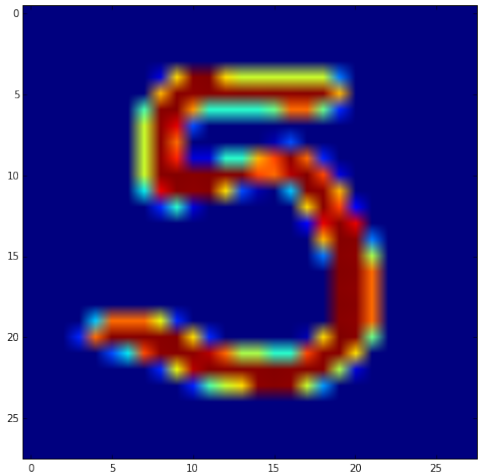

Figure 7: index: 5922, label: 5, predict: 3

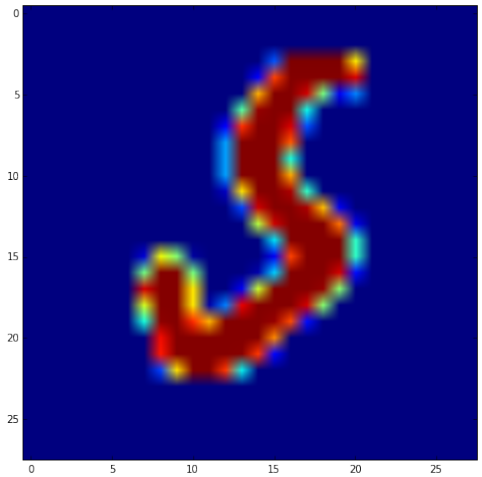

Figure 8: index: 9982, label: 5, predict: 6

6 A classifier for MNIST that occasionally abstains

6.1 prediction description

The strategy I adopt is that (1) to choose a threshold elaborately based on the performance on validation set and given f, (2) to abstain when the largest and the second largest posterior probability are rather close, or their difference is smaller than the threshold, (3) to predict otherwise. This strategy gives privilege to the prediction of easy instances which are far from the decision boundary.
Suppose $a = argmax_j\ (log\pi_j + logP_j(x))\ j \in \{0, 1, ..., 9\}$, $b = argmax_j\ (log\pi_j + logP_j(x))\ j \in \{0, 1, ..., 9\} - \{a\}$.
The equation for prediction should be

$$h(x) = \begin{cases} argmax_j\ (log\pi_j + logP_j(x)) & if\ log\pi_a + logP_a(x) - [log\pi_b + logP_b(x)] > threshold \\ abstain & otherwise \end{cases}$$

6.2 Pseudo code for choosing parameters

From the strategy above, it is important to choose an appropriate threshold. The pseudo code for this threshold calculation is as follows:
diff = []
For each data in validation set
    calculate the difference between the largest two posterior probabilities δ
    diff += [δ]
sort diff in increasing order
threshold = diff[f * len(diff)]

6.3 Performance

The two curves are shown below.

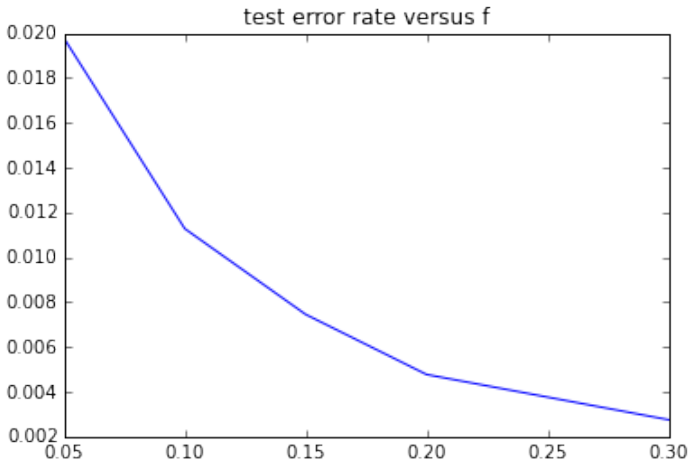

Figure 9: test error rate - f

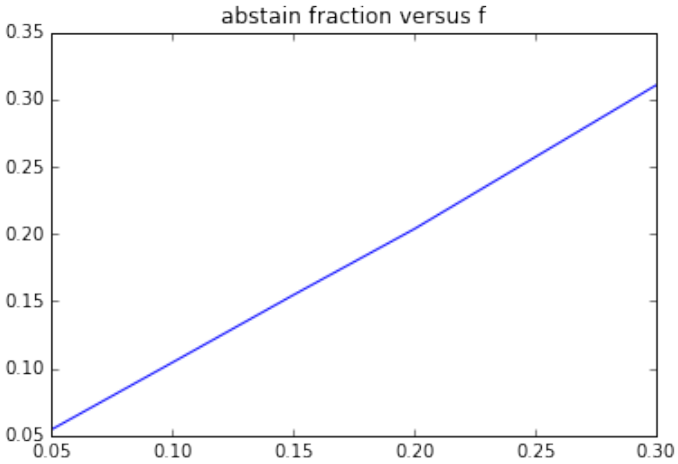

Figure 10: abstain fraction - f