Assignment 3

Name: Vuong Minh Phu

Student ID: 202050798

Class Computer Engineering

Instructor: Prof. Seung-Hoon Na

Problem 4.6

Problem description: Let $x \sim N(\mu, \Sigma)$ where $x \in \mathrm{R}^2$ and

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

where ρ is the correlation coefficient. Show that the pdf is given by:

$$p(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \left(\exp(-\frac{1}{2(1-\rho^2)}(\frac{(x_1-\mu_1)^2}{\sigma_1^2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2} - 2\rho\frac{(x_1-\mu_1)}{\sigma_1}\frac{(x_2-\mu_2)}{\sigma_2})) \right)$$

Solution:

The general expression for a multivariate Gaussian is given by:

$$\frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$$

We have: $|\Sigma| = (\sigma_1\sigma_2)^2 - (\rho\sigma_1\sigma_2)^2 = (\sigma_1\sigma_2)^2(1-\rho^2)$ (2)

Now we need to inverse Σ:

$$\Sigma^{-1} = \frac{1}{(\sigma_1\sigma_2)^2(1-\rho^2)} \begin{bmatrix} \sigma_2^2 & -\rho\sigma_1\sigma_2 \\ -\rho\sigma_1\sigma_2 & \sigma_1^2 \end{bmatrix}$$

Substitude (3) and (2) into (1) and we get the result:

$$\frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \left(\exp(-\frac{1}{2(det(\Sigma))}(\sigma_2^2(x_1-\mu_1)^2 - 2\rho\sigma_1\sigma_2(x_1-\mu_1)(x_2-\mu_2) + \sigma_1^2(x_2-\mu_2))) \right) =$$

$$\frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \left(\exp(-\frac{1}{2(1-\rho^2)}(\frac{(x_1-\mu_1)^2}{\sigma_1^2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2} - 2\rho\frac{(x_1-\mu_1)}{\sigma_1}\frac{(x_2-\mu_2)}{\sigma_2})) \right)$$

Problem 4.15

Problem description: The unbiased estimates for the covariance of a d-dimensional Gaussian based on n samples is given by

$$\hat{\Sigma} = C_n = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - m_n)(x_i - m_n)^T$$

a. Show that the covariance can be sequentially updated as follows

$$C_{n+1} = \frac{n-1}{n}C_n + \frac{1}{n+1}(x_{n+1} - m_n)(x_{n+1} - m_n)^T$$

b. How much time does it take per sequential update?

c. Show that we can sequentially update the precision matrix using

$$C_{n+1}^{-1} = \frac{n}{n-1}\left[C_n^{-1} - \frac{C_n^{-1}(x_{n+1}-m_n)(x_{n+1}-m_n)^T C_n^{-1}}{\frac{n^2-1}{n} + (x_{n+1}-m_n)^T C_n^{-1}(x_{n+1}-m_n)} \right]$$

d. What is the time complexity per update?

Solution:

a. We can rewrite (4.278) as follow:

$$nC_{n+1} - (n-1)C_n = \frac{n}{n-1} + (x_{n+1} - m_n)(x_{n+1} - m_n)^T$$

we can turn the two terms on the left-hand-side into:

$$nC_{n+1} = \sum_{j=1}^{n+1}(x_i - m_{n+1})(x_i - m_{n+1})^T = \sum_{j=1}^{n+1} x_j x_j^T - (n+1)m_{n+1}m_{n+1}^T (n-1)C_n = \sum_{j=1}^{n}(x_i - m_n)(x_i - m_n)^T = \sum_{i=1}^{n} x_j x_j^T - nm_n m_n^T$$

Substitude (2) into LHS of (1) we get:

$$nC_{n+1} - (n-1)C_n = x_{n+1}x_{n+1}^T - (n+1)m_{n+1}m_{n+1}^T + nm_n m_n^T$$

we also have:

$$m_{n+1} = \frac{x_{x+1} + nm_n}{n+1}$$

Substitude (4) into (3):

$$nC_{n+1} - (n-1)C_n = x_{n+1}x_{n+1}^T - \frac{(n+1)}{(n+1)^2}(nm_n + x_{n+1})(nm_n + x_{n+1})^T + nm_n m_n = x_{n+1}x_{n+1}^T - \frac{1}{n+1}(n^2 m_n m_n^T + nm_n x_{n+1}^T + nx_{n+1}m_n^T + x_{n+1}x_{n+1}^T) + nn$$

b.

$$C_{n+1} = \frac{n-1}{n}C_n + \frac{1}{n+1}(x_{n+1} - m_n)(x_{n+1} - m_n)^T$$

base on this we can see that computing the mean $m_{n+1} = \frac{x_n + x_{n+1}}{n+1}$ took $O(1)$

outer product $(x_{n+1} - m_n)(x_{n+1} - m_n)^T$ took $O(d^2)$

the rest of the operations took $O(1)$

so the total time complexity: $O(d^2)$

c. First we need to calculate the inverse of C_{n+1}

$$C_{n+1}^{-1} = [\frac{n-1}{n}C_n + \frac{1}{n+1}(x_{n+1} - m_n)(x_{n+1} - m_n)^T]^{-1} =$$

$$\frac{n}{n-1}C_n^{-1} - \frac{\frac{n}{n-1}C_n^{-1}\frac{1}{n+1}(x_{n+1} - m_n)(x_{n+1} - m_n)^T\frac{n}{n-1}C_n^{-1}}{1 + \frac{1}{n+1}(x_{n+1} - m_n)^T\frac{n}{n-1}C_n^{-1}(x_{n+1} - m_n)} =$$

$$\frac{n}{n-1}[C_n^{-1} - \frac{nC_n^{-1}(x_{n+1} - m_n)(x_{n+1} - m_n)^T C_n^{-1}}{n^2 - 1 + n(x_{n+1} - m_n)^T C_n^{-1}(x_{n+1} - m_n)}] =$$

$$\frac{n}{n-1}[C_n^{-1} - \frac{C_n^{-1}(x_{n+1} - m_n)(x_{n+1} - m_n)^T C_n^{-1}}{\frac{n^2-1}{n} + (x_{n+1} - m_n)^T C_n^{-1}(x_{n+1} - m_n)}]$$

d. We have:

$$C_n^{-1}(x_{n+1} - m_n)(x_{n+1} - m_n)^T C_n^{-1} A = C_n^{-1}(x_{n+1} - m_n) : O(d^2) B = (x_{n+1} - m_n)^T C_n^{-1} : O(d^2)$$

finally:

$$C_n^{-1}(x_{n+1} - m_n)(x_{n+1} - m_n)^T C_n^{-1} = AB : O(d^2)$$

Problem 4.20

Problem description: Given:

a) GaussI: A generative classifier where the class conditional densities are Gaussian, with both covariance matrices set to **I** and $p(y)$ is uniform

b) GaussX: as for GaussI, but the covariance matrices are unconstrained

c) LinLog: A logistic regression model with linear features

d) QuadLog: A logistic regression model, using linear and quadratic features

For each of the model pairs state whether $L(M) \leq L(M')$, $L(M) \geq L(M')$ or whether no such statement can be made

Decision creiteria:

$$L(M) = \frac{1}{n}\sum_{i=1}^{n} log\, p(y_i | x_i, \hat{\theta}, M)$$

a. GaussI, LinLog

$$M_1 = GaussI M_2 = LinLog \Rightarrow L(M_1) \leq L(M_2)$$

This is because when we calculate the cost function during training for Gaussian classifier, we need the maxmimize the joint log-likelihood instead of strainght forward like the linear Log case:

$$l(w, w_0) = \frac{1}{n} log\, p(x_i, y_i | w, w_0) = \frac{1}{n}\sum_i log\, p(x_i | w, w_0) p(y_i | x_i, w, w_0) = \frac{1}{n}\sum_i log\, p(x_i | w, w_0) + L$$

where (w, w_0) is the weight vector which define the decision boundary plan on the input spacce x and L is the same as the cost function of LogReg. Therefore $L(M_1) \leq L(M_2)$

b. GaussX, QuadLog

$$M_1 = GaussX M_2 = QuadLog \Rightarrow L(M_1) \leq L(M_2)$$

The reason is the same as a., but we have an extra quadratic function of the input x inside the sigmoid both for GaussX and QuadLog

c. LinLog, QuadLog

$$M_1 = LinLog M_2 = QuadLog \Rightarrow L(M_1) \leq L(M_2)$$

This is because LinLog is the special case of QuadLog just by setting the weights of the quadratic terms to 0. Therefore QuadLog outperforms LinLog

d. GaussI, QuadLog

$$M_1 = GaussI M_2 = QuadLog \Rightarrow L(M_1) \leq L(M_2)$$

This is due to $L(M_{GaussI}) \leq L(M_{LinLog}) \leq L(M_{QuadLog})$

e. It is true because all the models are based on decision boundary and a sigmoid and the scores L gets better as the instances are forced into the correct region of the boundary. Due to the shape of the sigmoid function, its derivative is bigger around 0.5 than at the extrmes where the function is satureated. Therefore when we maximize the cost function we will decrease R.