Problem Set - CAINE, Wilbert (20584260)

1. Logistic Regression

a.

$$\begin{aligned} g(x) &= 0.2x_1 + 0.1x_2 + 0.3x_3 - 3 \\ &= 0.2 * 5 + 0.1 * 20 + 0.3 * 1 - 3 \\ &= 0.3 \end{aligned}$$

$$\begin{aligned} f(x) = P(x\ votes\ for\ party\ A) = \sigma\big(g(x)\big) &= \frac{1}{1 + e^{-g(x)}} \\ &= \frac{1}{1 + e^{-0.3}} \\ &= 0.57444251681 \end{aligned}$$

b.

$$\begin{aligned} g(x) &= 0.2x_1 + 0.1x_2 + 0.3x_3 - 3 \\ &= 0.2 * 3 + 0.1 * 12 + 0.3 * 0 - 3 \\ &= -1.2 \end{aligned}$$

$$\begin{aligned} f(x) = P(x\ votes\ for\ party\ A) = \sigma\big(g(x)\big) &= \frac{1}{1 + e^{-g(x)}} \\ &= \frac{1}{1 + e^{-(-1.2)}} \\ &= 0.2314752165 \end{aligned}$$

$$\begin{aligned} logit(p) &= \log\left[p/(1-p)\right] \\ &= \log\left[0.2314752165/(1 - 0.2314752165)\right] \\ &= -1.2 \end{aligned}$$

c. We assume that the probability threshold is 0.5.

x will vote for party A if

$$\begin{aligned} f(x) = P(x\ votes\ for\ party\ A) = \sigma\big(g(x)\big) = \frac{1}{1 + e^{-g(x)}} &> 0.5 \\ \frac{1}{1 + e^{-g(x)}} &> \frac{1}{2} \\ 1 &> e^{-g(x)} \\ \ln(1) &> \ln\big(e^{-g(x)}\big) \\ 0 &> -g(x) \\ g(x) &> 0 \end{aligned}$$

x will vote for party B if

$$\begin{aligned} f(x) = P(x\ votes\ for\ party\ A) = \sigma\big(g(x)\big) = \frac{1}{1 + e^{-g(x)}} &< 0.5 \\ \frac{1}{1 + e^{-g(x)}} &< \frac{1}{2} \\ 1 &< e^{-g(x)} \\ \ln(1) &< \ln\big(e^{-g(x)}\big) \\ 0 &< -g(x) \\ g(x) &< 0 \end{aligned}$$

x will vote for party A or B with equal probability when

$$\begin{aligned} f(x) = P(x\ votes\ for\ party\ A) = \sigma\big(g(x)\big) = \frac{1}{1 + e^{-g(x)}} &= 0.5 \\ \frac{1}{1 + e^{-g(x)}} &= \frac{1}{2} \\ 1 &= e^{-g(x)} \\ \ln(1) &= \ln\big(e^{-g(x)}\big) \end{aligned}$$

$$0 = -g(x)$$
$$g(x) = 0$$

d. In general, g(x) can defined as a dot product of the weight vector w and the input x, i.e.,

$$g(x) = w^T x$$

Following the same procedure in c, a threshold, which is usually 0.5, of the probability p must first be defined to obtain a decision boundary of a more general form. We can expand the logistic regression model into an n-class classification model by applying n different g(x) functions where each g(x) computes the probability that x belongs to the corresponding class. As for the loss function, the original loss can be replaced by the cross-entropy loss function for classification problem as described in the lecture notes. The final model would be a feedforward neural network where the output layer has n nodes instead of one node only as in logistic regression model. This network allows us to predict which class x comes from out of the n class instead of just 2 class as in logistic regression model. The following functions defines the overall new model.

$$g_i(x) = w_i^T x$$
$$f_i(x) = P(x\ votes\ for\ party\ i) = softmax\big(g_i(x)\big)$$
$$1 \le i \le n$$

In the end, to classify x into one of the n classes, we could simply pick an i that maximize $f_i(x)$ or $arg\max_i f_i(x)$.

2. Logistic Regression

a. z will approach one-hot vector such that z_j approaches 1 when $j = arg\max_j e^{x_j}$, otherwise it will approach 0

b. z will approach a uniform value, i.e., 1/K for all z_j

c. Case 1 ($i = j$):

$$\frac{\partial z_j}{\partial x_j} = \frac{\frac{e^{\frac{x_j}{\tau}}}{\tau}\sum_{k=1}^{K} e^{\frac{x_k}{\tau}} - e^{\frac{x_j}{\tau}}\frac{e^{\frac{x_j}{\tau}}}{\tau}}{\left(\sum_{k=1}^{K} e^{\frac{x_k}{\tau}}\right)^2}$$
$$= \frac{1}{\tau}\left(z_j - z_j^2\right)$$
$$= z_j\left(1 - z_j\right)/\tau$$

Case 2 ($i \neq j$):

$$\frac{\partial z_j}{\partial x_i} = \frac{-e^{\frac{x_j}{\tau}}\frac{e^{\frac{x_i}{\tau}}}{\tau}}{\left(\sum_{k=1}^{K} e^{\frac{x_k}{\tau}}\right)^2}$$
$$= \frac{1}{\tau}\left(0 - z_i z_j\right)$$
$$= z_j(0 - z_i)/\tau$$

Combining the two cases, we have

$$\frac{\partial z_j}{\partial x_i} = z_j\left(\delta_{ji} - z_i\right)/\tau$$

3. Feedforward Neural Network

We can regard this new variant of classification problem as a K binary classification problem instead of K-class classification problem. Here, we assume that the y is a zero vector when x belongs to none of the K classes.

a. While there may be many ways to modify the original network to handle this variant of the classification problem, one approach is to use the original activation function $g_j^{[1]}$ and $g_k^{[2]}$, but modify $g_l^{[3]}$ to be the logistic regression for binary classification as in the previous lecture notes in logistic regression, such as the logistic function below

$$g_l^{[3]}(x_l^{[3]}) = \frac{1}{1 + e^{-x_l^{[3]}}}$$

This modification is sufficient to solve the new variant of problem given that we use the appropriate objective function below to perform K binary classification.

b. Following the modification in a, we need to change the original loss function, which is sometimes called categorical cross-entropy loss, defined in lecture notes to the (Bernoulli or binary) cross entropy loss used in previous lecture notes in logistic regression, such as

$$L(W;S) = -\sum_{q=1}^{N}\sum_{l}\left[y_l^{(q)} \log z_l^{[3](q)} + \left(1 - y_l^{(q)}\right)\log\left(1 - z_l^{[3](q)}\right)\right]$$

c. After defining a new network for K binary classification problem, we use the $arg\max_i z_l^{[3]}$ to classify x into class i given that $\max_i z_l^{[3]}$ is greater than 0.5 (as the default threshold). Otherwise, we will indicate 'none of the above' or 'reject'.

$$class = \begin{cases} arg\max_i z_l^{[3]} & , if \ \max_i z_l^{[3]} > 0.5 \\ \text{'none of the above' or 'reject'} & , otherwise \end{cases}$$

4. Convolutional Neural Networks

a. Conv (128, 32, 7, 7, 2)

i. $h = w = \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor = \left\lfloor \frac{261+2*0-7}{2} + 1 \right\rfloor = 128$

$h \times w \times c = 128 \times 128 \times 32$

ii. $(7 \times 7 \times 128 + 1) \times 32 = 200736$

b.

i. Conv1 (128, 16, 1, 1, 1)

$$h = w = \left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor = \left\lfloor \frac{261 + 2 * 0 - 1}{1} + 1 \right\rfloor = 261$$

$h \times w \times c = 261 \times 261 \times 16$

ii. Conv1 (128, 16, 1, 1, 1)

$$(1 \times 1 \times 128 + 1) \times 16 = 2064$$

iii. Conv2 (16, 32, 7, 7, 2)

$$h = w = \left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor = \left\lfloor \frac{261 + 2 * 0 - 7}{2} + 1 \right\rfloor = 128$$

$h \times w \times c = 128 \times 128 \times 32$

iv. Conv2 (16, 32, 7, 7, 2)

$$(7 \times 7 \times 16 + 1) \times 32 = 25120$$

v. Conv1 + Conv2

$$saving = \frac{200736 - (2064 + 25120)}{200736} = 86.4578351666\%$$

5. Principal Component Analysis

a. 100%. In other words, the variance is completely preserved. By using two principal components obtained by PCA to project S linearly onto another two-dimensional space spanned by the principal components together, 100% of the total variance can be explained by the two principal components together since S also in a two-dimensional space. Thus, there is no dimensionality reduction and it is equivalent to linear transformation that maps S to the same space.

b.

$$det\left(\begin{bmatrix} 7 & \sqrt{5} \\ \sqrt{5} & 3 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = 0$$

$$det\left(\begin{bmatrix} 7-\lambda & \sqrt{5} \\ \sqrt{5} & 3-\lambda \end{bmatrix}\right) = 0$$

$$\lambda^2 - 10\lambda + 16 = 0$$

$$\lambda_1 = 8$$

$$\lambda_2 = 2$$

In PCA, the total variance is the sum of the variance of the principal component which is the eigenvalue as shown in the lecture notes. The maximum percentage of total variance that can be explained is expressed by the sum of eigenvalues of the used principal components divided by the sum of eigenvalues of all principal components. By using one principal components, the maximum percentage of total variance is $\frac{8}{8+2} \times 100\% = 80\%$.

6. Clustering – Partitional Clustering

a.

For simplicity, we partition S into k clusters or k new sets of data points

$$S_i = \left\{ x^{(l)} \middle| b_i^{(l)} = 1 \right\}$$

which contains all data points that belongs to cluster i according to b with i = 1, …, k. By rewriting the objective function, we have

$$E\left(\{m_i\}_{i=1}^k, S\right) = \sum_{i=1}^{k} \sum_{x \epsilon S_k} \|x - m_i\|^2 = \sum_{i=1}^{k} E(m_i, S_i)$$

Minimizing the original objective function is equivalent to minimizing the mean squared distance for each individual cluster given by

$$E(m_i, S_i) = \sum_{x \epsilon S_i} \|x - m_i\|^2$$

$$= \sum_{x \epsilon S_i} (x - m_i)^T (x - m_i)$$

$$= \sum_{x \epsilon S_i} (x^T x - 2m_i^T x + {m_i}^T m_i)$$

Taking the derivative, we have

$$\frac{\partial}{\partial m_i} E(m_i, S_i) = \sum_{x \epsilon S_k} (-2x + 2m_i) = 0$$

$$\sum_{x \epsilon S_k} (-2x) + |S_k| 2m_i = 0$$

Thus, $arg\min_{m_i} E(m_i, S) = \frac{\sum_{x \epsilon S_k} x}{|S_k|} = \frac{\sum_l b_i^{(l)} x^{(l)}}{\sum_l b_i^{(l)}}$ which corresponds to the cluster mean obtained after updating m_i according to the local optimization algorithm.

b.

Denote S1 and S2 as the cluster 1 and cluster 2 with mean m1 and m2 respectively.

S1 = {(x, y) | 1<x<9} S2 = {(x, y) | 11<x<17} m1 = (5, 1.5) m2 = (14, 1.5)

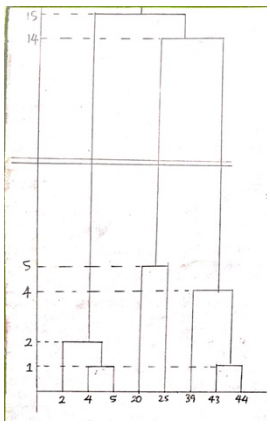
S1 = {(x, y) | 1<x<9} S2 = {(x, y) | 11<x<17} m1 = (5, 1.5) m2 = (14, 1.5)

➔ stop

S1 has 10 data points while S2 has 8 data points.

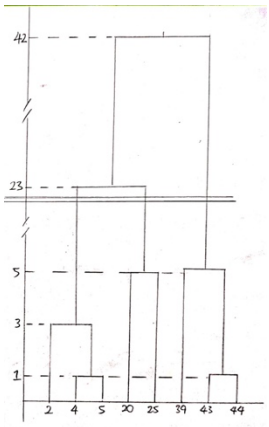
7. Clustering – Hierarchical Clustering

a.



three top-most level clusters: {2, 4, 5}, {20, 25}, and {39, 43, 44}

b.



three top-most level clusters: {2, 4, 5}, {20, 25}, and {39, 43, 44}

c. The two dendrograms have different hierarchical structure. However, the two partitions with three subsets each are the same.