CS 217: Artificial Intelligence and Machine Learning **Jan-Apr 2024**

Lecture 8: Logistic Regression and Multiclass Classification

Lecturer: Swaprava Nath *Scribe(s): SG15, SG16*

8.1 Logistic Regression

Continuing the introduction to the logistic regression in the last lecture, let us first jot down the problem and the notations.

Let Y be a binary output variable and we want to model the conditional probability $P(Y = 1|X = x)$ for features $X \in \mathbb{R}^d$, this can be done as

$$\begin{bmatrix} P(Y = 1|X = x, w_1, w_2) \\ P(Y = 0|X = x, w_1, w_2) \end{bmatrix} = \begin{bmatrix} \zeta e^{w_1^T x} \\ \zeta e^{w_2^T x} \end{bmatrix} ; \zeta = \frac{1}{e^{w_1^T x} + e^{w_2^T x}}$$

where w_1 and w_2 are the associated weight vectors and ζ is the normalization factor, which ensures the sum of the two probabilities to be one.
Furthermore,

$$P(Y = 1|X = x, w_1, w_2) = \frac{e^{w_1^T x}}{e^{w_1^T x} + e^{w_2^T x}}$$

$$P(Y = 1|X = x, w_1, w_2) = \frac{1}{1 + e^{(w_2 - w_1)^T x}}$$

$$P(Y = 1|X = x, w_1, w_2) = \frac{1}{1 + e^{-(w_1 - w_2)^T x}}$$

$$P(Y = 1|X = x, w) = \frac{1}{1 + e^{-w^T x}}; w = w1 - w2$$

$$P(Y = 1|X = x, w) = \sigma(w^T x)$$

where σ is the sigmoid function,

$$\sigma(z) = \frac{1}{1 + e^{-z}};\ z \in \mathbb{R}$$

One can also use the fact $\sigma(z) + \sigma(-z) = 1$ to compute $P(Y = 0|X = x, w)$.

8.2 Computing the weights vector

Let $w^*_{LR} \in \mathbb{R}^d$ denote the optimal value for the weights vector which correspond to d features of the input variable that maximizes the logistic regression objective function. We will use MLE to compute the same,

$$w^*_{LR} = \underset{w}{\text{argmax}} \prod_{i=1}^{i=n} P(Y = y_i|X = x_i, w)$$

for the training data given as n tuples of the input-output form (x_i, y_i),

$$w^*_{LR} = \underset{w}{\text{argmax}} \sum_{i=1}^{i=n} \log(P(Y = y_i|X = x_i, w))$$

$$w^*_{LR} = \underset{w}{\text{argmin}} \sum_{i=1}^{i=n} -\log(P(Y = y_i|X = x_i, w))$$

let us denote the sum to be minimized as $NLL(w)$ and each of the summands in the summation to be $NLL_i(w)$, NLL stands for the Negative Log Likelihood. As $y_i \in \{0, 1\}$ we can simplify

$$NLL_i(w) = \begin{cases} -\log((P(Y = 1|X = x_i, w))) & \text{when } y_i = 1 \\ -\log(1 - (P(Y = 1|X = x_i, w))) & \text{otherwise} \end{cases}$$

thus $NLL_i(w)$ can be rewritten as

$$NLL_i(w) = -y_i \log(\sigma(w^T x_i)) - (1 - y_i) \log(1 - \sigma(w^T x_i)))$$

The loss function so formed is the cross-entropy loss which is encountered quite frequently whenever we are trying to estimate a probability distribution for an event from a set of observations.

Simplifying further,

$$NLL_i(w) = y_i \log(1 + e^{-w^T x_i}) - (1 - y_i) \log(\frac{e^{-w^T x_i}}{1 + e^{-w^T x_i}})$$

$$NLL_i(w) = y_i \log(1 + e^{-w^T x_i}) + (1 - y_i) w^T x_i + (1 - y_i) \log(1 + e^{-w^T x_i})$$

$$NLL_i(w) = (1 - y_i) w^T x_i + \log(1 + e^{-w^T x_i})$$

taking the grads,

$$\nabla_w NLL_i(w) = x_i \frac{-e^{-w^T x_i}}{1 + e^{-w^T x_i}} + (1 - y_i) x_i$$

$$\nabla_w NLL_i(w) = -x_i(y_i - \sigma(w^T x_i))$$

we use gradient descent to approximate the weights vector

$$w_{t+1} \leftarrow w_t - \eta \sum_{i=1}^{n} \nabla_w NLL_i(w)$$

After training our weights this model classifies points as

$$\hat{y} = \begin{cases} 1 & \text{if } \frac{P(y=1|\hat{x},w)}{P(y=0|\hat{x},w)} > 1 \\ 0 & \text{otherwise} \end{cases}$$

Note: However this leads to a linear decision boundary. For a non-linear decision boundary we can, as we have done earlier make the use of basis functions. This will, as expected, increase the computational complexity of the model. Here, our prediction changes as,

$$P(Y = 1|X = x, w) = \sigma(w^T \Phi(x))$$

where w is the weights matrix of the appropriate dimensions.

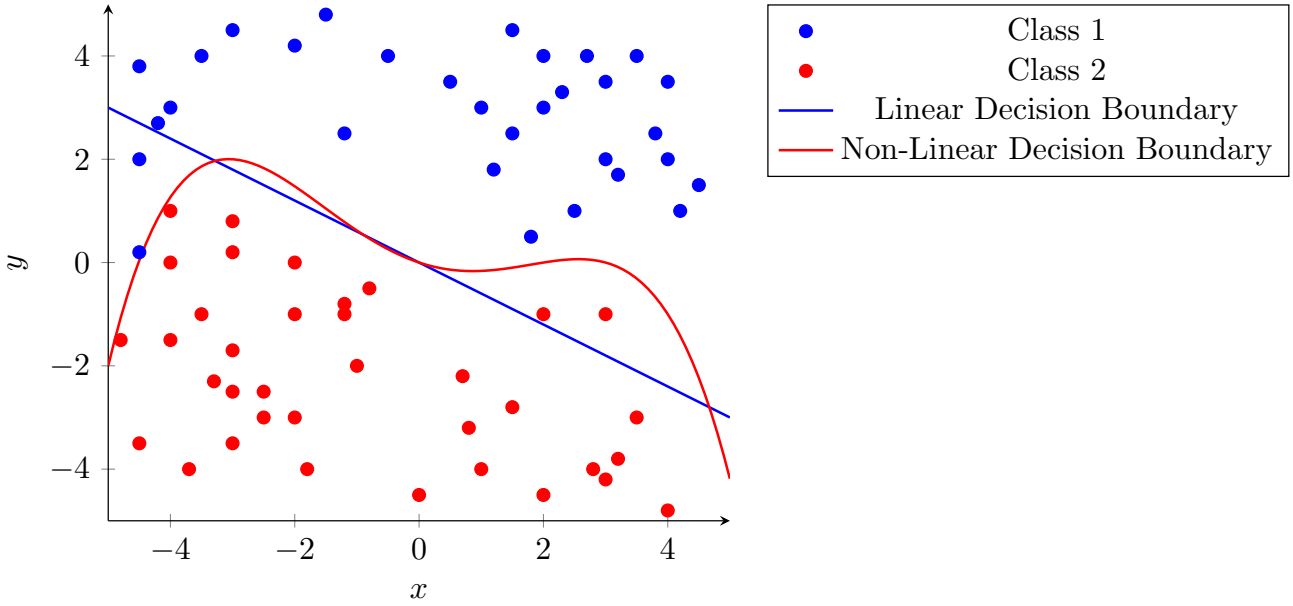

Figure 8.1: Linear and Non-Linear Decision Boundaries

8.3 Multi-class Classification

8.3.1 One vs Rest Classifier

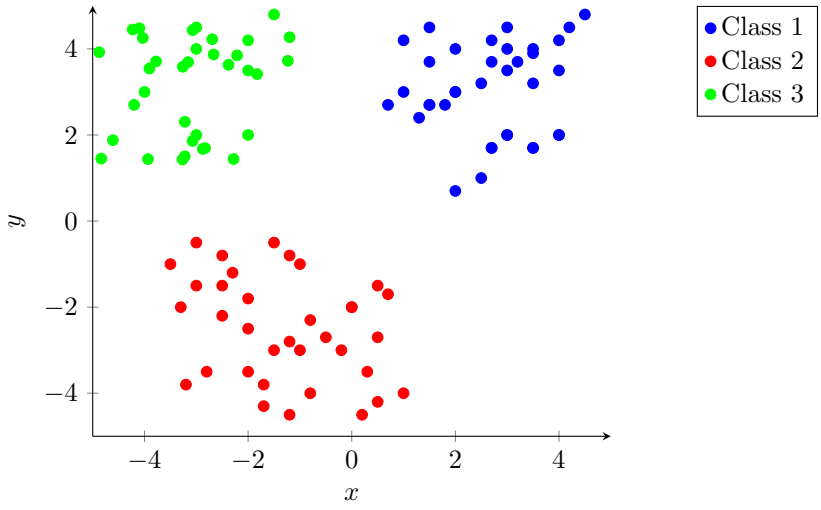

Figure 8.2: Multi-class classification problem

This method uses multiple binary regression steps to classify the data. This method seems to be a kind of hack because the final probability distribution we have is not a real one. The basic idea behind this is we consider one class as the first class and all the rest of classes as the second classes and apply binary regression on these two. We do this for every class.

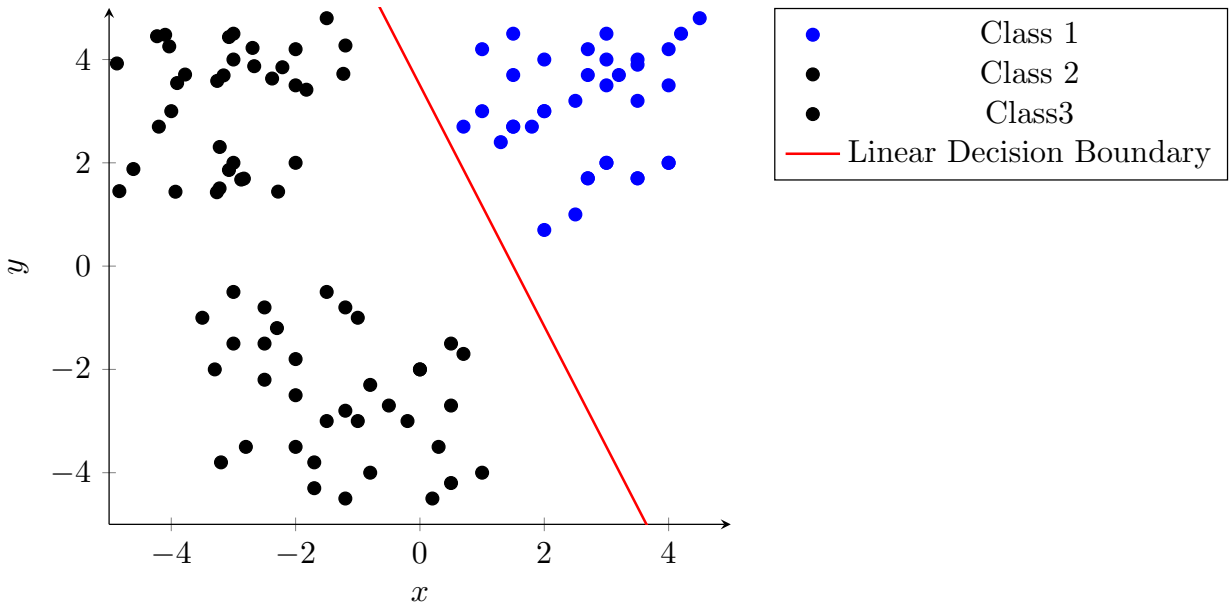

Figure 8.3: Sub-Problem 1

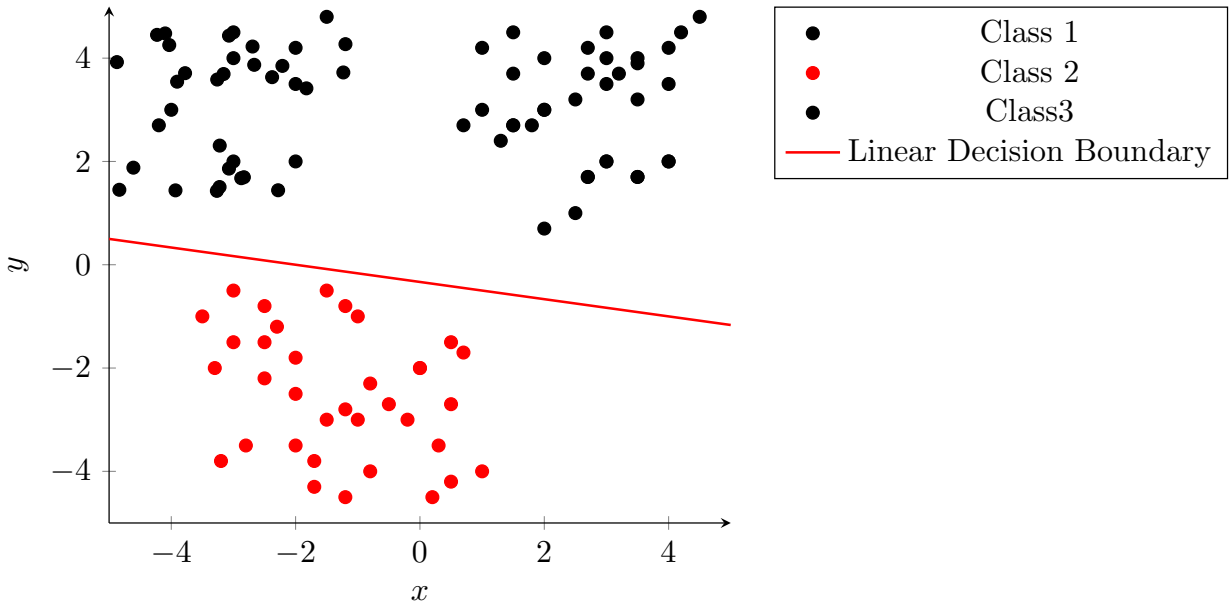

Figure 8.4: Sub-Problem 2

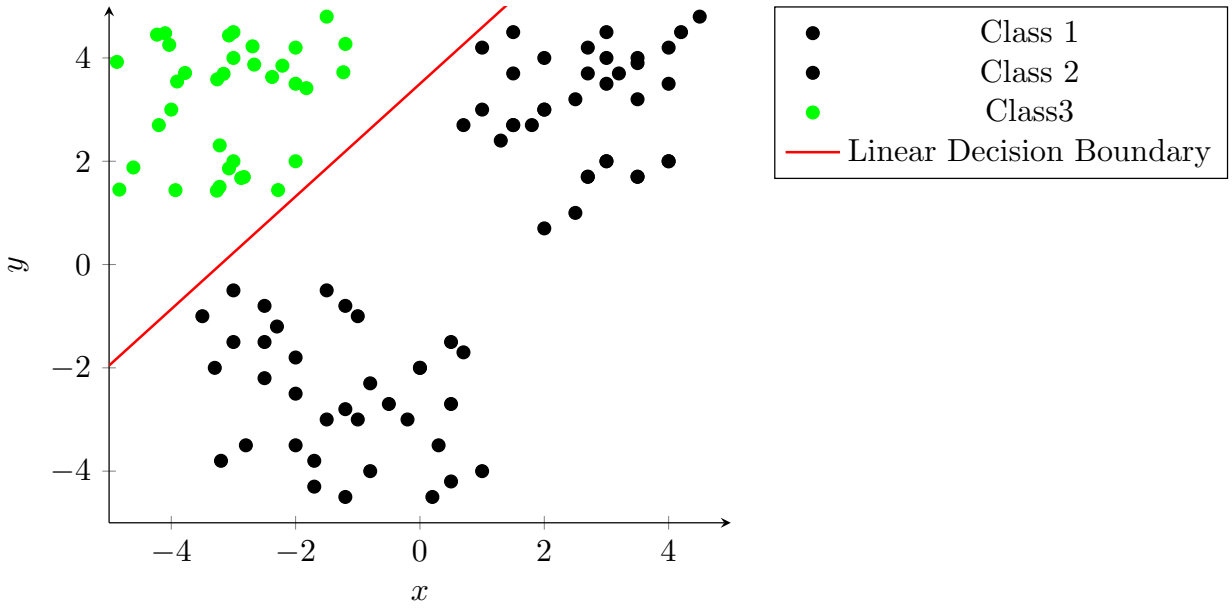

Figure 8.5: Sub-Problem 3

After perfoming all the binary regressions, in the end we have k weight vectors $w_1, w_2, \ldots, w_k$ corresponding to each class. Suppose we have a test data point $\hat{x}$

$$\sigma(w_i^T \hat{x}) \text{ is the probablity of } \hat{x} \text{ being in class } i \tag{8.1}$$

The final prediction is given by:

$$\hat{y} = \arg\max_k (\sigma(w_k^T \hat{x})) \tag{8.2}$$

i.e, after computing the probabilities for a given data point to belong to each of the three classes we choose the one for which the probability is highest.

8.3.2 Softmax Regression

The above distribution is not real. We want an actual probability distrubution. This takes us back to analysis we did earlier for binary regression. We will perform similar analysis for multiple classes(Suppose there are K classes). We will assign the score of $w_k^T \hat{x}$ to $P(y = k|x, w)$, that is

$$\begin{aligned} w_1^T \hat{x} &\rightarrow P(y = 1|x, w) \\ w_2^T \hat{x} &\rightarrow P(y = 2|x, w) \\ &\vdots \\ w_k^T \hat{x} &\rightarrow P(y = k|x, w) \end{aligned}$$

As we did before , we will exponentiate the $w_k^T \hat{x}$ and also normalise the values by addind a factor ρ to get a valid probability distribution.

$$f(\hat{x}, w) = \begin{bmatrix} P(y = 1|\hat{x}, w) \\ P(y = 2|\hat{x}, w) \\ \vdots \\ P(y = k|\hat{x}, w) \end{bmatrix} = \begin{bmatrix} \rho e^{w_1^T \hat{x}} \\ \rho e^{w_2^T \hat{x}} \\ \vdots \\ \rho e^{w_k^T \hat{x}} \end{bmatrix} \quad , w = \begin{bmatrix} \text{——} w_1^T \text{——} \\ \text{——} w_2^T \text{——} \\ \vdots \\ \text{——} w_K^T \text{——} \end{bmatrix}_{K \times d} \tag{8.3}$$

where d is dimesion of data point, K is number of classes, and

$$\rho = \frac{1}{\sum_{j=1}^{K} e^{w_j^T \hat{x}}}$$

This is also known as the **Softmax-Function**.
Finding out the value of w_i is a different story, once we have the found the w_i by regression, we use them to predict the class of $\hat{x}$ by as follows:

$$\hat{y} = \arg\max_i(P(y = i|x, w)) = \arg\max_i \left(\frac{e^{w_i^T \hat{x}}}{\sum_{j=1}^{K} e^{w_j^T \hat{x}}} \right) \tag{8.4}$$

To train our model, we will perform a regression with the same loss function as in the Binary Logistic regression:- Negative Log Likelihood function:-

$$\begin{aligned} NLL(w) &= -\sum_{i=1}^{n} log\left[P(y_i|x_i, w)\right] \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{K} \mathbb{I}\{y_i = j\} log\left[\frac{e^{w_j^T x_i}}{\sum_{m=1}^{K} e^{w_m^T x_i}}\right] \end{aligned} \tag{8.5}$$

We intend to find the gradient of $NLL(w)$, with respect to each of the w_i, so that we can perform gradient descent. But to make our life easier , lets first consider $NLL_i(w)$(loss corresponding to i^{th} data point :

$$NLL_i(w) = -\sum_{j=1}^{K} \mathbb{I}\{y_i = j\} log\left[\frac{e^{w_j^T x_i}}{\sum_{m=1}^{K} e^{w_m^T x_i}}\right] \tag{8.6}$$

Now take the gradient with repsect to w_k, we have two cases:
Case 1 : $y_i = k$

$$\begin{aligned} \nabla_{w_k} NLL_i(w) &= -\nabla_{w_k}\left(log\left[\frac{e^{w_k^T x_i}}{\sum_{m=1}^{K} e^{w_m^T x_i}}\right]\right) \\ &= -\left(1 - \frac{e^{w_k^T x_i}}{\sum_{m=1}^{K} e^{w_m^T x_i}}\right) x_i \end{aligned}$$

Case 2 : $y_i = l \neq k$

$$\begin{aligned} \nabla_{w_k} NLL_i(w) &= -\nabla_{w_k}\left(log\left[\frac{e^{w_l^T x_i}}{\sum_{m=1}^{K} e^{w_m^T x_i}}\right]\right) \\ &= -\left(-\frac{e^{w_k^T x_i}}{\sum_{m=1}^{K} e^{w_m^T x_i}}\right) x_i \end{aligned}$$

We can also write this as:

$$\nabla_{w_k} NLL_i(w) = (f_k(x_i, w) - \mathbb{I}\{y_i = k\})x_i \tag{8.7}$$

where $f_k(x_i, w)$ represents the k^{th} row of the softmax-function.Another thing to note that we want to eliminate the $\mathbb{I}\{\}$ function from our expression, in order to do that, we can represent our $y_i's$ using **one-hot representation**. For each data point, y_i is a vector whose all elements are 0, except the k^{th} , with is 1, if y_i belongs to k^{th} class.

$$y_i = [0\,0\,\ldots\underset{\substack{\uparrow \\ k^{th}\text{index}}}{1}\ldots\,0]$$

Using this we can write :

$$\nabla_{w_k} NLL_i(w) = (f_k(x_i, w) - y_i^{(k)})x_i \tag{8.8}$$

Also summing over i gives us the overall gradient

$$\nabla_{w_k} NLL(w) = \sum_{i=1}^{n} (f_k(x_i, w) - y_i^{(k)})x_i \tag{8.9}$$

Now all that is left is to do the standard GD, where each update step would be:

$$w' : \begin{bmatrix} —— w_1^T —— \\ —— w_2^T —— \\ \vdots \\ —— w_K^T —— \end{bmatrix} \leftarrow \begin{bmatrix} —— w_1^T —— \\ —— w_2^T —— \\ \vdots \\ —— w_K^T —— \end{bmatrix} - \eta \begin{bmatrix} —— \nabla_{w_1} NLL(w)^T —— \\ —— \nabla_{w_2} NLL(w)^T —— \\ \vdots \\ —— \nabla_{w_K} NLL(w)^T —— \end{bmatrix} \tag{8.10}$$

It can also be proven, that this GD converges.

8.4 Naive Bayes Vs Logistic Regression

Discriminative Models: Discriminative models tries to draw a decision boundary between classes in a classification problem.It tries to find a relation between the input data set to the classification it belongs to.

Generative Models: Generative models, in contrast to discriminative models, focus on modelling the underlying probability distribution of the entire dataset. Instead of learning a decision boundary between classes, generative models aim to learn how the data is generated.

Naive Bayes is a generative model as it focuses on learning the conditional probabilities of features of given classes rather than drawing explicit decision boundaries. It estimates how likely certain features are for each class, allowing it to make predictions based on observed data.

Logistic Regression is indeed a discriminative model as it aims to draw a decision boundary in the feature space to separate different classes. By learning the relationship between input features and the binary outcome, it directly models the posterior probability of belonging to a particular class, enabling effective classification based on the identified decision boundary

8.4.1 Equivalence of Gaussian Naive Bayes and Logistic Regression

Gaussian Naive Bayes is a machine-learning classification technique based on a probabilistic approach that assumes each feature of the class follows a normal distribution

Naive Bayes is given as follows,

$argmax_{y_k} P(Y = y_k|x) = argmax_{y_k} (\prod_{i=1}^{k} P(x_i|Y = y_k))\ P(Y = y_k)$

In Gaussian Naives $P(x = x_j|Y = y_k)$ is a normal distribution where x_1, x_2..... are all features for a given class y_k.

$P(x = x_j|Y = y_k) \sim \mathcal{N}(\mu_{jk},\ \sigma_{jk}^2)$

To proceed further, we consider 3 assumptions, Those are

1)x_i, x_j are conditionally independent.

2)let π be a different constant from the usual one.Let $P(Y = 1) = \pi$, $P(Y = 0) =$ 1- π which means we are considering that we are given only two classes .

3)We assume $\sigma_{jk}$for all k is equal to σ_j

$P(x = x_j|Y = 0) \sim \mathcal{N}(\mu_{j0}, \sigma_{j0}^2)$, $P(x = x_j|Y = 1) \sim \mathcal{N}(\mu_{j1}, \sigma_{j1}^2)$

Now lets find out the conditional probability of a Y = y_k for a given feature set x means say x represents $(x_1, x_2, x_3, \ldots)$, i.e., the combination of all features.

$$P(y_i = 1|x) = \left(\frac{P(x|y_i = 1)P(y_i = 1)}{P(x|y_i = 1)P(y_i = 1) + P(x|y_i = 0)P(y_i = 0)}\right) \tag{8.11}$$

The above equation can also be written as

$$P(y_i = 1|x) = \frac{1}{1 + \frac{P(x|y_i=0)P(y_i=0)}{P(x|y_i=1)P(y_i=1)}} \tag{8.12}$$

$$P(y_i = 1|x) = \frac{1}{1 + \exp\left(\ln\left(\frac{P(y_i=0)}{P(y_i=1)}\right) + \ln\left(\frac{P(x|y_i=0)}{P(x|y_i=1)}\right)\right)} \tag{8.13}$$

As $x_1, x_2, x_3, \ldots$ are independent features so to find joint probability we can multiply individual probabilities. Say there are total d features.

$$P(y_i = 1|x) = \frac{1}{1 + \exp\left(\ln\frac{1-\pi}{\pi} + \ln\prod_i^d \frac{P(x_i|y_i=0)}{P(x_i|y_i=1)}\right)} \tag{8.14}$$

$$P(y_i = 1|x) = \frac{1}{1 + \exp\left(\ln\frac{1-\pi}{\pi} + \sum_i^d \ln\frac{P(x_i|y_i=0)}{P(x_i|y_i=1)}\right)} \tag{8.15}$$

$$P(x_i|y_i = 1) = \frac{1}{\sqrt{2\pi\sigma_{i1}^2}} \exp\left(-\frac{1}{2\sigma_{i1}^2}(x_i - \mu_{i1})^2\right) \tag{8.16}$$

$$P(x_i|y_i = 0) = \frac{1}{\sqrt{2\pi\sigma_{i0}^2}} \exp\left(-\frac{1}{2\sigma_{i0}^2}(x_i - \mu_{i0})^2\right) \tag{8.17}$$

$$\ln\left(\frac{P(x|y_i = 0)}{P(x|y_i = 1)}\right) = \ln\left(\frac{\frac{1}{\sqrt{2\pi\sigma_{i0}^2}} \exp\left(-\frac{1}{2\sigma_{i0}^2}(x_i - \mu_{i0})^2\right)}{\cdot\frac{1}{\sqrt{2\pi\sigma_{i1}^2}} \exp\left(-\frac{1}{2\sigma_{i0}^2}(x_i - \mu_{i1})^2\right)}\right) \tag{8.18}$$

As we have assumed σ_{jk} for all k is equal to σ_j

Then above equation becomes

$$\ln\left(\frac{P(x|y_i = 0)}{P(x|y_i = 1)}\right) = \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i} + \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} x_i \tag{8.19}$$

Substituting in eq 8.15 we get

$$P(y_i = 1|x) = \frac{1}{1 + \exp\left(\ln\left(\frac{1-\pi}{\pi} + \sum_i^d \left(\ln\left(\frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2}\right) + \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} x_i\right)\right)\right)} \tag{8.20}$$

Now let us compare it with

$$P(y_i = 1|x) = \frac{1}{1 + \exp(w_0 + \sum_i^d w_i x_i)} \tag{8.21}$$

$$w_0 = \ln\left(\frac{1-\pi}{\pi}\right) + \sum_i^d \ln\left(\frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2}\right) \tag{8.22}$$

$$w_i = \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} \tag{8.23}$$

We can compare the eq 8.21 with that of logistic regression eq 8.24 and they are similar.

$$P(y_i = 1|x_i, w) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x_i})}}. \tag{8.24}$$