CSE 575 - ASSIGNMENT 2

TEJA KUNDERU

1211384158

Question 1:

Given examples are –

$x_1 = (1, 1)$, $x_2 = (1, -1)$ with label y = 1

$x_3 = (-1, 1)$, $x_4 = (-1, -1)$ with y = 0

The weight vector is initialized as $w_0 = (0, 0, 0)$

$P(Y = 1 \mid X_i) = h_w(x) = 1 / (1 + \exp(w_0 + \sum_{j=1}^{n} w_j x_j))$

Weight update function is $w_{t+1}^{j} \rightarrow w_t^j + \eta * \sum_{i=1}^{m} \left(y^i - h_w(x^i)\right) x_j^i$

Where m = number of training examples

n = number of features

i = current training example

j = current feature

t = training instance

η = learning rate

With the initial weight of (0, 0, 0)

$h_w(x^1) = 1 / 1 + \exp(0) = 1 / 2$

$h_w(x^2) = 1 / 1 + \exp(0) = 1 / 2$

$h_w(x^3) = 1 / 1 + \exp(0) = 1 / 2$

$h_w(x^4) = 1 / 1 + \exp(0) = 1 / 2$

Therefore,

$w_1^0 \rightarrow w_0^0 + \eta * \sum_{i=1}^{m} (y^i - 0.5) x_j^i$

$w_1^0 \rightarrow 0 + \eta * [(1 - 0.5) * 1 + (1 - 0.5) * 1 + (0 - 0.5) * 1 + (0 - 0.5) * 1]$

$w_1^0 \rightarrow 0$

$w_1^1 \rightarrow w_0^1 + \eta * \sum_{i=1}^{m} (y^i - 0.5) x_j^i$

$w_1^1 \rightarrow 0 + \eta$ * [(1 – 0.5) * 1 + (1 – 0.5) * 1 + (0 – 0.5) * -1 + (0 – 0.5) * -1]

$w_1^1 \rightarrow 2\ \eta$

$w_1^2 \rightarrow w_0^2 + \eta * \sum_{i=1}^{m} (y^i - 0.5) x_j^i$

$w_1^2 \rightarrow 0 + \eta$ * [(1 – 0.5) * 1 + (1 – 0.5) * -1 + (0 – 0.5) * 1 + (0 – 0.5) * -1]

$w_1^2 \rightarrow 0$

The weights are (0, 2 η, 0) after one instance of training

$h_w(x^1)$ = 1 / 1 + exp(-2 η) = exp(2 η) / 1 + exp(2 η)

$h_w(x^2)$ = 1 / 1 + exp(-2 η) = exp(2 η) / 1 + exp(2 η)

$h_w(x^3)$ = 1 / 1 + exp(2 η)

$h_w(x^4)$ = 1 / 1 + exp(2 η)

$w_2^0 \rightarrow w_1^0 + \eta * \sum_{i=1}^{m} (y^i - 0.5) x_j^i$

$w_2^0 \rightarrow 0 + \eta$ * [(1 – exp(2 η) / 1 + exp(2 η)) * 1 + (1 – exp(2 η) / 1 + exp(2 η)) * 1 + (0 – 1 / 1 + exp(2 η)) * 1 + (0 – 1 / 1 + exp(2 η)) * 1]

$w_2^0 \rightarrow 0$

$w_2^1 \rightarrow w_1^1 + \eta * \sum_{i=1}^{m} (y^i - 0.5) x_j^i$

$w_2^1 \rightarrow 0 + \eta$ * [(1 – exp(2 η) / 1 + exp(2 η)) * 1 + (1 – exp(2 η) / 1 + exp(2 η)) * 1 + (0 – 1 / 1 + exp(2 η)) * -1 + (0 – 1 / 1 + exp(2 η)) * -1]

$w_2^1 \rightarrow 2\ \eta + 4\ \eta$ / 1 - exp(2 η)

$w_2^2 \rightarrow w_1^2 + \eta * \sum_{i=1}^{m} (y^i - 0.5) x_j^i$

$w_2^2 \rightarrow 0 + \eta$ * [(1 – exp(2 η) / 1 + exp(2 η)) * 1 + (1 – exp(2 η) / 1 + exp(2 η)) * -1 + (0 – 1 / 1 + exp(2 η)) * 1 + (0 – 1 / 1 + exp(2 η)) * -1]

$w_2^2 \rightarrow 0$

The weights are (0, 2 η + 4 η / 1 - exp(2 η), 0) after two instances of training

$h_w(x^1)$ = exp(2 η + 4 η / 1 – exp(2 η)) / 1 + exp(2 η + 4 η / 1 - exp(2 η))

$h_w(x^2)$ = exp(2 η + 4 η / 1 – exp(2 η)) / 1 + exp(2 η + 4 η / 1 - exp(2 η))

$h_w(x^3)$ = 1 / 1 + exp(2 η + 4 η / 1 – exp(2 η))

$h_w(x^4)$ = 1 / 1 + exp(2 η + 4 η / 1 – exp(2 η))

We know 2 η + 4 η / 1 - exp(2 η) = w_2^1

Therefore, we get

$h_w(x^1)$ = exp(w_2^1) / 1 + exp(w_2^1)

$h_w(x^2)$ = exp(w_2^1) / 1 + exp(w_2^1)

$h_w(x^3)$ = 1 / 1 + exp(w_2^1)

$h_w(x^4)$ = 1 / 1 + exp(w_2^1)

$$w_3^0 \rightarrow w_2^0 + \eta * \sum_{i=1}^{m} (y^i - 0.5) x_j^i$$

$w_3^0 \rightarrow 0 + \eta$ * [(1 – exp(w_2^1) / 1 + exp(w_2^1)) * 1 + (1 – exp(w_2^1) / 1 + exp(w_2^1)) * 1 + (0 – 1 / 1 + exp(w_2^1)) * 1 + (0 – 1 / 1 + exp(w_2^1)) * 1]

$w_3^0 \rightarrow 0$

$$w_3^1 \rightarrow w_2^1 + \eta * \sum_{i=1}^{m} (y^i - 0.5) x_j^i$$

$w_3^1 \rightarrow 0 + \eta$ * [(1 – exp(w_2^1) / 1 + exp(w_2^1)) * 1 + (1 – exp(w_2^1) / 1 + exp(w_2^1)) * 1 + (0 – 1 / 1 + exp(w_2^1)) * -1 + (0 – 1 / 1 + exp(w_2^1)) * -1]

$w_3^1 \rightarrow w_2^1$ + 4 w_2^1 / 1 + exp(w_2^1)

$$w_3^2 \rightarrow w_2^2 + \eta * \sum_{i=1}^{m} (y^i - 0.5) x_j^i$$

$w_3^2 \rightarrow 0 + \eta * [(1 - \exp(w_2^1) \ / \ 1 + \exp(w_2^1)) * 1 + (1 - \exp(w_2^1) \ / \ 1 + \exp(w_2^1)) * -1 + (0 - 1 \ / \ 1 + \exp(w_2^1)) * 1 + (0 - 1 \ / \ 1 + \exp(w_2^1)) * -1]$

$w_3^2 \rightarrow 0$

The weights are $(0, w_2^1 + 4\ \eta \ / \ 1 + \exp(w_2^1), 0)$ after three instances of training

From this, we can deduce that

$W_{t+1} = (0, w_t^1 + 4\ \eta \ / \ 1 + \exp(w_t^1), 0)$

The final weight vector tends to ∞ as the learning rate tends to ∞

Question 2:

The class label in logistic regression is predicted based on the probability, $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

This is a function of $\theta^T x$, which is a linear function of x.

The decision boundary for the logistic regression is a set of x such that h(x) = 0.5
i.e., $\frac{1}{1+e^{-\theta^T x}} = 0.5$

=> $e^{-\theta^T x} = 1$

=> $\theta^T x = 0$

=> $\sum_{i=0}^{n} \theta_i x_i = 0$

Therefore, the decision boundary is linear.

Question 3:

1.

We have 3 positive data points {-1, 0, 1} and 3 negative data points {-3, -2, 2}

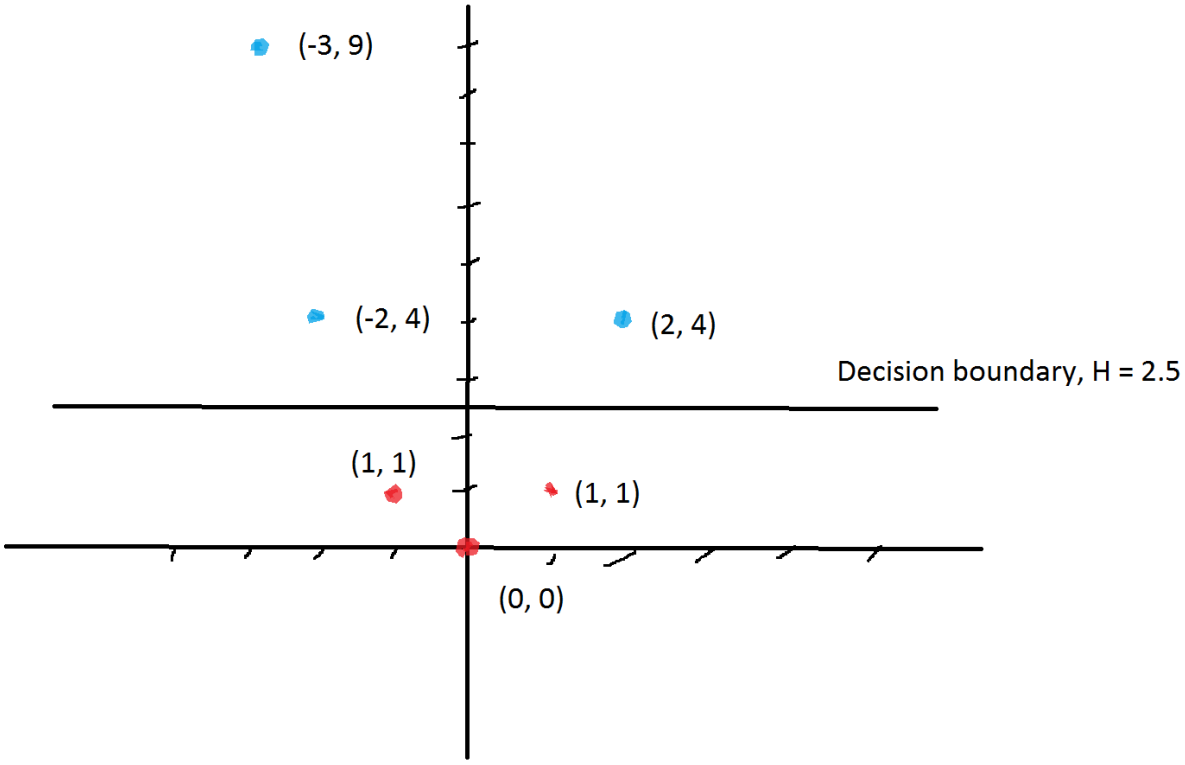
a) Feature map $\{R^1 \rightarrow R^2\}$ will be (x, x^2)

Therefore, the 2D mapping consists of –

3 positive points (-1, 1), (0, 0), (1, 1)

3 negative points (-3, 9), (-2, 4), (2, 4)

b)



Decision boundary is H = 2.5

The corresponding support vectors are (-2, 4), (2, 4), (-1, 1), (1, 1)

c) $\begin{bmatrix} x_1 \\ x_1^2 \end{bmatrix}, \begin{bmatrix} x_2 \\ x_2^2 \end{bmatrix}$

Kernel $K(x_1, x_2) = x_1 x_2 + x_1^2 x_2^2$

2.

a) $\text{argmin}_{\{w,b\}} \frac{1}{2} w^t w + C \sum_1^m e_i$

Subject to $y_i (w^t x_i + b) \geq 1 - e_i, e_i \geq 0$ for every i

=> $e_i \geq 1 - y_i (w^t x_i + b)$ and $e_i \geq 0$

Using this in the original equation, we get

$\text{argmin}_{\{w,b\}} \frac{1}{2} w^t w + C \sum_1^m (1 - yi\ (wt\ xi + b))$

=> $\text{argmin}_{\{w,b\}} \frac{1}{2} w^t w + C \sum_1^m (0)$

This depends on the maximum value of e_i.

This is an optimization problem which can be written as:

$\text{argmin}_{\{w,b\}} \frac{1}{2} w^t w + C \sum_1^m (\max(1 - yi\ (wt\ xi + b)), 0)$

$\text{argmin}_{\{w,b\}} w^t w + 2C \sum_1^m (\max(1 - yi\ (wt\ xi + b)), 0)$

Substituting 2C = ƛ, we get

$\text{argmin}_{\{w,b\}} w^t w + ƛ \sum_1^m (\max(1 - yi\ (wt\ xi + b)), 0)$

c) So, the value of ƛ as a function of C will be 2C.

3.

Given data points - x1 = (0,1), y1 = -1

x2 = (2,0), y2 = 1

x3 = (1,0), y3 = 1

x4 = (0,2), y4 = -1

We are training a soft-margin linear SVM.

A1, A2, A3, A4 are the lagrange multipliers for x1, x2, x3 and x4 respectively.

Regularization parameter C = 100.

	X1	X2	X3	X4
X1	1	0	0	2
X2	0	4	2	0
X3	0	2	1	0
X4	2	0	0	4

a) $J(A_1, A_2, A_3, A_4) = -0.5 \times (A_1^2 + 4A_1A_4 + 4A_2^2 + 4A_2A_3 + A_3^2 + 4A_4^2)$

Subject to $0 \leq A_1, A_2, A_3, A_4 \leq 100$ and $- A_1 + A_2 + A_3 + A_4 = 0$

b) $J(A_1, A_4) = (A_1 + A_4) - 0.5 \times (A_1^2 + 4A_1A_4 + 4A_4^2)$

Subject to $0 \leq A_1, A_4 \leq 100$ and $A_1 + A_2 + A_3 + A_4 = 12$

Therefore,

$J(A_1) = 12 - 0.5 (A_1 + 2A_4)^2$

$= 12 - 0.5 (A_1 + 2(12 - A_1))^2$

$= 12 - 0.5 (A_1 - 24)^2$

Taking the derivate with respect to A_1 and equating to 0, we get

$0.5 * 2 (A_1 - 24) = 0$

So, $A_1 = 24$ and $A_4 = -12$

But $0 \leq A_1 \leq 100$ and $0 \leq 12 - A_1 \leq 100$, $0 \leq A_1 \leq 12$

Therefore, $A_1 = 12$ and $A_4 = 0$

c) $J(A_2, A_3) = (A_2 + A_3) - 0.5 (A_3^2 + 4A_2A_3 + 4A_2^2)$

$J(A_2) = 12 - 0.5 (2A_2 + A_3)^2$

$= 12 - 0.5 (2A_2 + (12 - A_2))^2$

$= 12 - 0.5 (A_2 + 12)^2$

Taking the derivate with respect to A_2 and equating to 0, we get

$0.5 * 2 (A_2 + 12) = 0$

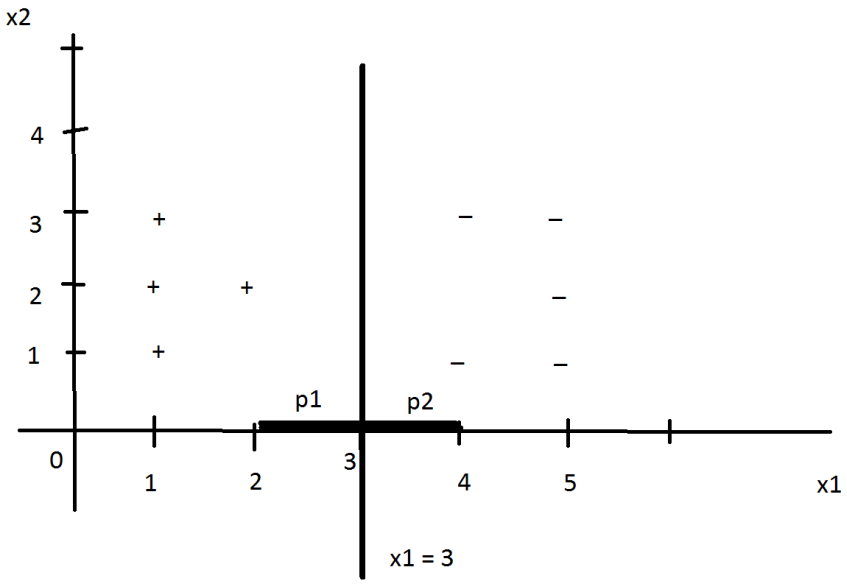
So, $A_2 = -12$ and $A_3 = 24$.

But $0 \leq A_2 \leq 100$ and $0 \leq A_3 \leq 12$

Therefore, $A_2 = 0$ and $A_3 = 12$

Question 4:

1.



Given the dataset, the value of C is large.

The decision boundary can be represented as min $\frac{1}{2}||\Theta||^2$

such that $p^{(i)}\ ||\Theta|| \geq 1$ if $Y^{(i)} = 1$

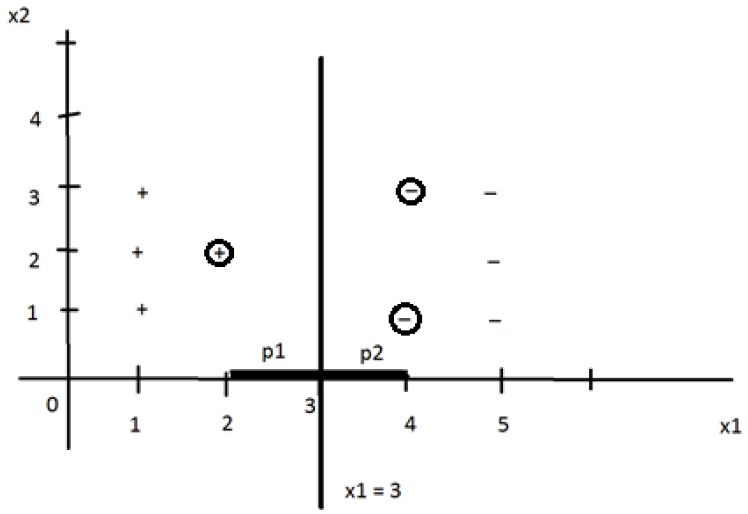
$p^{(i)}\ ||\Theta|| \leq 1$ if $Y^{(i)} = 1$

p is the projection of $x^{(i)}$ on vector Θ

For the chosen values of $p^{(1)}$, $p^{(2)}$ on the x1 = 3 boundary, the margin distance is maximum.

Therefore, x1 = 3 is the decision boundary.

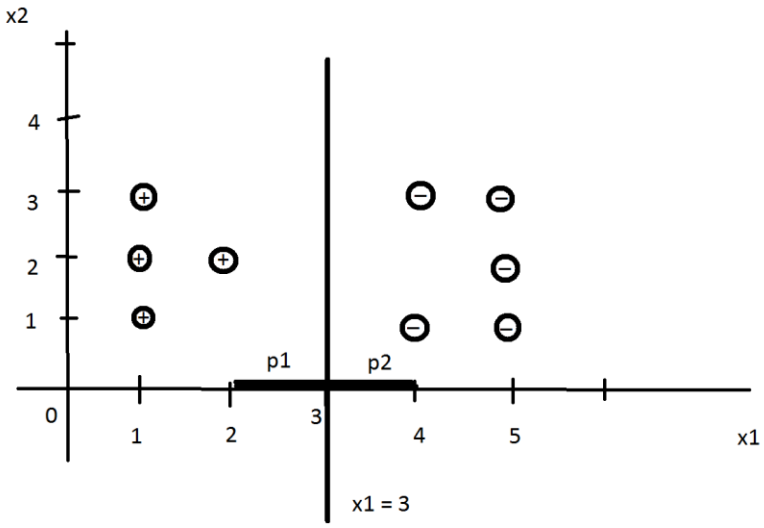
2.



The points (2,2), (4,1) and (4,3) are support vectors and lie on decision boundary.

If these points were removed, we would get a different decision boundary, x1 = 2.5

3.



In regularized logistic regression, every point contributes when we calculate the ‘w’ vector.

The ‘w’ vector affects the boundary.

Therefore, removing any of the circled points causes the boundary to change.

4.

Given high-dimensional feature map: $\phi: R^D \rightarrow R^d$

$K(x_i, x_j) = \phi(x_i) . \phi(x_j)$

The dot product in the D-dimensional space is given by:

$\phi(x_i) . \phi(x_j) = \sum_{l=1}^{D} (\phi(xi))^l (\phi(xj))^l$

Given Euclidian distance:

$$|| \phi(x_i) - \phi(x_j) || = (\sum_{l=1}^{D} (\phi(x_i)^l - \phi(x_j)^l)^2)^{1/2}$$

$$= (\sum_{l=1}^{D} (\phi(x_i)^l)^2 + (\phi(x_j)^l)^2 - 2\phi(x_i)^l \phi(x_j)^l)^{1/2}$$

$$= (\sum_{l=1}^{D} (\phi(x_i)^l)^2 + \sum_{l=1}^{D} (\phi(x_j)^l)^2 - \sum_{l=1}^{D} (2\phi(x_i)^l \phi(x_j)^l))^{1/2}$$

Using the dot product definition that we have stated above, we get

$$|| \phi(x_i) - \phi(x_j) || = (\phi(x_i).\phi(x_i) + \phi(x_j).\phi(x_j) - 2\phi(x_i).\phi(x_j))^{1/2}$$

=> $$|| \phi(x_i) - \phi(x_j) || = (k(x_i, x_i) + k(x_j, x_j) - 2(k(x_i, x_j))^{1/2}$$

5.

From the above example, we have -

$$|| \phi(x_i) - \phi(x_j) || = (k(x_i, x_i) + k(x_j, x_j) - 2(k(x_i, x_j))^{1/2}$$

Squaring on both sides -

$$|| \phi(xi) - \phi(xj) ||^2 = k(x_i, x_i) + k(x_j, x_j) - 2(k(x_i, x_j)$$

The new RBF kernel function is given by -

$K(x_i, x_j) = \exp(-1/2 || x_i - x_j ||^2)$

Substituting the kernel function in our above equation, we get

$|| \phi(xi) - \phi(xj) ||^2 = \exp(-1/2 | | x_i - x_i | |^2) + \exp(-1/2 | | x_j - x_j | |^2) - 2 \exp(-1/2 | | x_i - x_j | |^2)$

=> $|| \phi(xi) - \phi(xj) ||^2 = 2 - 2(\exp(-1/2 | | x_i - x_j | |^2))$

The exponential term is always positive. Therefore, the value on the right side is always less than or equal to 2.

Hence, $|| \phi(xi) - \phi(xj) ||^2 < 2$.

6.

Let x_i and x_j be the two neighbors in the input space.

They can be represented in Euclidian distance form as $|| x - x_i ||$ and $|| x - x_j ||$ respectively.

Assume $|| x - x_j || < || x - x_i ||$

After feature space mapping, we get

$|| \phi(x) - \phi(xj) ||^2 = 2 - 2 \exp(-1/2 | |x - x_j | |^2)$

$|| \phi(x) - \phi(xi) ||^2 = 2 - 2 \exp(-1/2 | |x - x_i | |^2)$

Since, $|| x - x_j || < || x - x_i ||$

$|| \phi(x) - \phi(xj) ||^2 < || \phi(x) - \phi(xi) ||^2$

If x_j is the nearest neighbor in R^d space (from assumption), then x_j will be the nearest neighbor also in R^D space i.e. the feature space.

Therefore, the 1-Nearest Neighbor classifier has the same performance as in the original input space, in a higher dimensional space, but not more.

Question 5:

1.

a) In Gaussian Naïve Bayes, each feature has a mean and variance to be estimated.

Number of independent parameters in Naïve Bayes can be calculated using the formula n*k + (k - 1)

Since we have two values to estimate per feature, the total number of independent parameters would be 2*m*k + (k - 1)

The output is a probability and follows Bernoulli distribution and therefore k = 2

Using this information, we can say that the total number of independent parameters required for this Gaussian Naïve Bayes classifier would be 4N + 1

b) Using Bayes rule,

P(Y = 1 | X) = P(X | Y = 1) P(Y = 1) / [P(X | Y = 0) P(Y = 0) + P(X | Y = 1) P(Y = 1)]]

= 1 / 1 + P(X | Y = 0) P(Y = 0) / P(X | Y = 1) P(Y = 1)

= 1 / 1 + exp(ln(P(X | Y = 0) P (Y = 0) / P(X | Y = 1) P (Y = 1)))

= 1 / 1 + exp(ln(P (Y = 0) / P(Y = 1)) + $\Sigma_i$ln(P(X_i | Y = 0) / P(X_i | Y = 1)))

= 1 / 1 + exp(ln(1 - θ / θ) + $\Sigma_i$ln(P(X_i | Y = 0) / P(X_i | Y = 1)))

As per the assumption, P(X_i | Y = y_k) has a Gaussian Distribution

$\Sigma_i \ln\left(\frac{P(Xi \mid Y=0)}{P(Xi \mid Y=1)}\right)$ = Σ_i ln(1 / $\sqrt{2}\pi\sigma^2$exp(- (X_i - μ_{i0})^2 / 2σ_i^2) / 1 / $\sqrt{2}\pi\sigma^2$ exp(- (X_i - μ_{i0})^2 / 2σ_i^2))

= Σ_i ln(exp((X_i - μ_{i0})^2 – (X_i - μ_{i1})^2 / 2σ_i^2))

= Σ_i ln(exp((2X_i (μ_{i0} - μ_{i1}) μ_{i0}^2 – μ_{i1}^2 / 2σ_i^2))

= Σ_i ln(exp((2X_i (μ_{i0} - μ_{i1}) + μ_{i0}^2 – μ_{i1}^2 / 2σ_i^2))

= Σ_i ln(exp(((μ_{i0} - μ_{i1}) / σ_i^2 X_i + μ_{i0}^2 – μ_{i1}^2 / 2σ_i^2))

Substituting this back into the equation, we have

P(Y = 1 | X) = 1 / 1 + exp(ln(1 - θ / θ) + Σ_i ln(exp(((μ_{i0} - μ_{i1}) / σ_i^2 X_i + μ_{i0}^2 – μ_{i1}^2 / 2σ_i^2))

= 1 / 1 + exp(w_0 + $\sum_{i=1}^{n} w_i X_i$)

Where, w_0 = ln(1 - θ / θ) + Σ_i (μ_{i0}^2 – μ_{i1}^2) / 2σ_i^2

W_i = (μ_{i0} - μ_{i1}) / σ_i^2

The Gaussian Naïve Bayes is thus expressed in the form of Logistic Regression.

2.

a) Number of independent parameters in Naïve Bayes = n*k + (k-1)

The output P (Y) has a Bernoulli distribution and hence k = 2

In Boolean Naïve Bayes classifier, for each feature X_i, we only have to estimate one value.

Therefore, the total number of independent parameters in Boolean Naïve Bayes Classifier is 2N + 1

b) Using Bayes rule,

P(Y = 1 | X) = P(X | Y = 1) P(Y = 1) / [P(X | Y = 0) P(Y = 0) + P(X | Y = 1) P(Y = 1)]]

= 1 / 1 + P(X | Y = 0) P(Y = 0) / P(X | Y = 1) P(Y = 1)

= 1 / 1 + exp(ln(P(X | Y = 0) P (Y = 0) / P(X | Y = 1) P (Y = 1)))

= 1 / 1 + exp(ln(P (Y = 0) / P(Y = 1)) + $\Sigma_i$ln(P(X_i | Y = 0) / P(X_i | Y = 1)))

= 1 / 1 + exp(ln(1 - θ / θ) + $\Sigma_i$ln(P(X_i | Y = 0) / P(X_i | Y = 1)))

X is a vector of Boolean variables such that

P (X_i | Y = k) = Θ_{ik} (k = 0,1) and P(Y=1) = Θ

Therefore,

$\sum_i \ln(\frac{P\ (X_i|Y=0)}{P\ (X_i|Y=1)}) = \sum_i \ln(\frac{\Theta_{i0}}{\Theta_{i1}})$

Using this in the above equation, we get

P (Y = 1 | X) = 1 / 1 + exp (ln ((1- Θ) / Θ) + $\sum_i \ln(\frac{\Theta_{i0}}{\Theta_{i1}})$)

= 1 / 1 + exp ($w_0 + \sum_{i=1}^{n} w_i X_i$)

Where w_0 = ln ((1- Θ) / Θ) + $\sum_i \ln(\frac{\Theta_{i0}}{\Theta_{i1}})$)

w_1 = 0

Thus, Boolean Naïve Bayes is expressed in the form of Logistic Regression.

Question 6:

1. Pseudo Code

- First, a matrix with the desired fractions, called 'fractions' is created.
- Each of the algorithms are implemented inside a loop that loops over the 'fractions' matrix.
- For every fraction, the data size 'fractionalDataSize' is found out.
- A random number is selected from the total number of training examples.
- This number is called the 'startIndex' and the following 'fractionalDataSize' numbers are stored in a new data set called 'newData'.
- If the 'endIndex' is greater than the number of training examples in the data set, then the remaining examples are taken from the top of the training data set.
- This 'newData' with random examples chosen from the original data set is used to train the algorithms.
- The process has to be repeated five times and accuracy must be averaged, therefore, a new random number is generated in each of the five iterations and is used to generate a new data set each time.
- Once the 'newData' is obtained, the algorithm can be trained and accuracy can be tested.
- The accuracy during each of the five runs and summed as 'sumAccuracyXX' and the average accuracy called 'currentAccuracyXX' is found out for each fraction,'f', in 'fractions' matrix.
- This accuracy is stored in a matrix called 'accuracyXX' and is used to plot the accuracies.
- The data sizes for each fraction are also stored in a separate matrix called 'dataSize'.
- The accracy vs data size plots for each algorithm are then generated.

Naïve Bayes Implementation:

- The probabilites P(Y = -1) and P(Y = 1) are first generated from the 'newData' data set and are stored as 'pY0' and 'pY1' respectively.

- This can be done using P(Y = a) = number of a's in the data set / num of training examples.
- The probabilities $P(X_j = k \mid Y = -1)$ and $P(X_j = k \mid Y = 1)$ are then calculated where 'j' is the feature and 'k' is the value taken by the feature in an example.
- The probabilites of P(X | Y) are calculated for each Y, feature j and value 'k'. These are stored in matrices 'pX0' and 'pX1' for P(X | Y = -1) and P(X | Y = 1) respectively.
- These can be calcualted using
  $P(X_j = k \mid Y = a)$ = number of examples with Y = a that have value 'k' in feature 'j' + 1 / total number of examples with value Y = a + number of possible values of 'k' for the feature.
- The addition of 1 in the numerator and 'k' in the denominator is for add-one smoothing.
- This causes values 'k' for features that do not appear in the training data set to be set to a very low non-zero probability.
- After finding the probabilities, for each testing examples in the data set, the probabilities P(Y = -1 | X) and P(Y = 1 | X) are calcualted. The classification is deemed correct if the algorithm returns a higher probability for the Y value given in the testing label.
- These probabilities can be calcualted using
  P(Y = a | X) = P(X | Y = a) P(Y = a) / P(X)
  where $P(X) = \sum_a p(x|y=a)p(x=a)$
  and $P(X \mid Y = a) = \prod_j P(x_j = k|y=a)$
- These probabilities can be obtained from the matrices 'pX0' and 'pX1' generated before
- The probabilities are multiplied and compared to get the total number of correct predictions.
- This value is divided by the total number of testing examples to get the accuracy.

Logistic Regression Implementation:

- First, the weights are iniliatized to 0
- The weights are then updated during each iteration according to the Weight update function $w_{t+1}^j \rightarrow w_t^j + \eta * \sum_{i=1}^{m} \left(y^i - h_w(x^i)\right) x_j^i$

  Where m = number of training examples

  n = number of features

  i = current training example

  j = current feature

  t = training instance

  η = learning rate
- Once the updated weight start to differ by less than a selected threshold, the update operation is stopped.
- The weights are then used to calculate the probabilty of $P(Y = 1 \mid X_i) = h_w(x) = 1 / (1 + \exp(w_0 + \sum_{j=1}^{n} w_j x_j))$ for each testing example.

- This probability is then used to compare with existing label in the test example and calculate the total number of correct predictions made by the algorithm.
- This is then divided by the testing data size to get the accuracy.

2.