1. Prove Bayes' Theorem. Briefly explain why it is useful for machine learning problems, i.e., by converting posterior probability to likelihood and prior probability.

Bayes' theorem is a formula that describes how to update the probabilities of hypotheses when given evidence. If A and B denote two events, P(A|B) denotes the conditional probability of A occurring, given that B occurs. The two conditional probabilities P(A|B) and P(B|A) are in general different. Bayes theorem gives a relation between P(A|B) and P(B|A).

$$P(A \mid B) = P(A \cap B) / P(B)$$

Similarly,

$$P(B \mid A) = P(A \cap B) / P(A)$$

Thus,

$$P(A \cap B) = P(A \mid B) * P(A) = P(B \mid A) * P(B)$$

Equating the two yields:

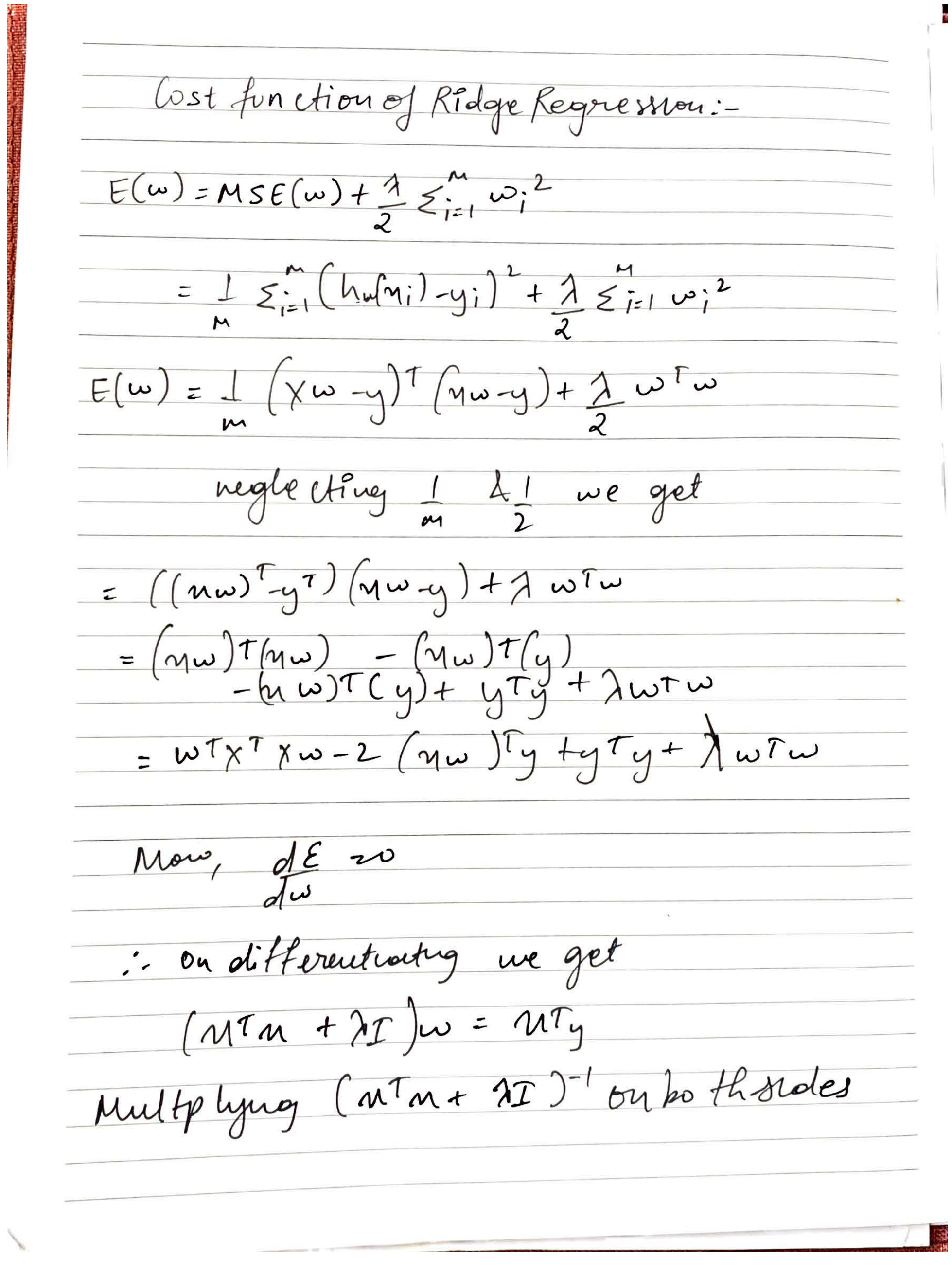
$$P(A|B) = P(A)\ P(B|A)\ P(B)$$

This equation, is known as Bayes Theorem is the basis of statistical inference

P(A) and P(B) are probabilities of A and B which are mutually exclusive. Hence, P(B|A) and P(A|B) are conditional probabilities. In the same lines, P(B|A) is Likelihood and P(A|B) is Posterior probability.

In machine learning, we have attributes, response variables and predictions or classifications. Using this algorithm, we will deal with the probability distributions of the variables in the dataset and predict the probability of the response variable belonging to a particular value, given the attributes of a new instance. We are interested in defining the best hypothesis from some space H, given the observed training data D. One way to specify what we mean by the best hypothesis is to say that we demand the most probable hypothesis, given the data D plus any initial knowledge about the prior probabilities of the various hypotheses in H. Bayes theorem provides a direct method for calculating such probabilities.

2. In Lecture 3-1, we gave the normal equation (i.e., closed-form solution) for linear regression using MSE as the cost function. Prove that the closed-form solution for Ridge Regression is $\boldsymbol{w} = (\lambda I + XT \cdot X)-1 \cdot XT \cdot \boldsymbol{y}$, where I is the identity matrix, $X = (x(1), x(2), \ldots, x(m))T$ is the input data matrix, $x(i) = (1, x1, x2, \ldots, xn)$ is the i^{th} data sample, and $y = (y(1), y(2), \ldots, ym)$. Assume the hypothesis function $\text{h}w(x) = w0 + w1x1 + w2x2 + \cdots + wnxn$, and $y(j)$ is the measurement of $\text{h}w(x)$ for the jth training sample. The cost function of the Ridge Regression is $E(\boldsymbol{w}) = MSE(\boldsymbol{w}) + \lambda$ 2∑m i=1 wi2. [Hint: please refer to the proof of the normal equation of linear regression].



Cost function of Ridge Regression:-

$$E(w) = MSE(w) + \frac{\lambda}{2}\sum_{i=1}^{m} w_i^2$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_w(x_i)-y_i)^2 + \frac{\lambda}{2}\sum_{i=1}^{m} w_i^2$$

$$E(w) = \frac{1}{m}(Xw-y)^T(xw-y) + \frac{\lambda}{2} w^Tw$$

neglecting $\frac{1}{m}$ & $\frac{1}{2}$ we get

$$= ((xw)^T - y^T)(xw-y) + \lambda w^Tw$$

$$= (xw)^T(xw) - (xw)^T(y) - (xw)^T(y) + y^Ty + \lambda w^Tw$$

$$= w^TX^TXw - 2(xw)^Ty + y^Ty + \lambda w^Tw$$

Now, $\frac{dE}{dw} = 0$

∴ on differentiating we get

$$(x^Tx + \lambda I)w = x^Ty$$

Multplying $(x^Tx + \lambda I)^{-1}$ on both sides

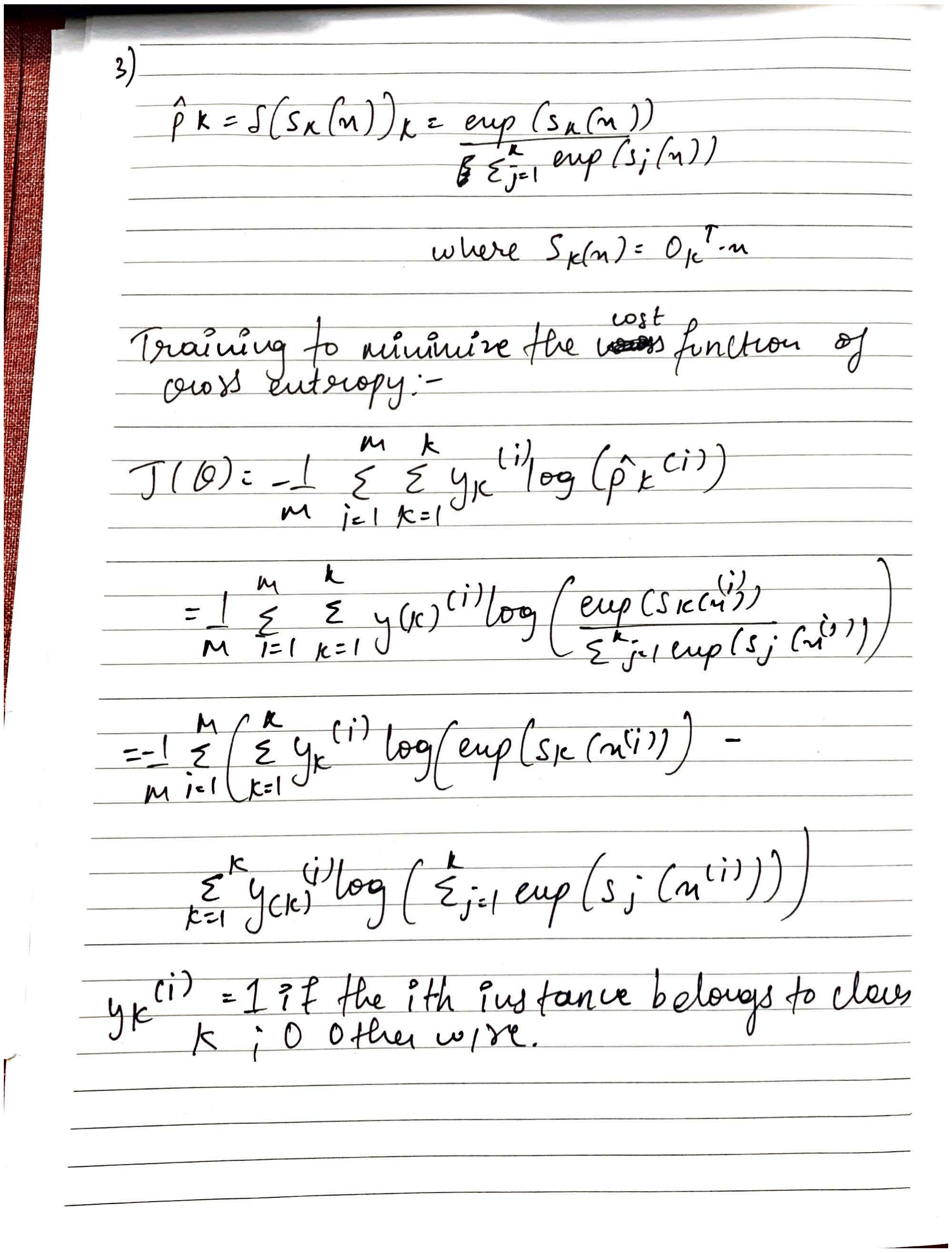
$w = (x^T x + \lambda I)^{-1} x^T y$

$w = (\lambda I + x^T . x)^{-1} . x^T y$

Hence Proved.

3. [10 points] Recall the multi-class Softmax Regression model on page 16 of Lecture 3-3. Assume we have K different classes. The posterior probability is $\hat{p}_k = \delta(s_k(x))_k = \frac{\exp(s_k(x))}{\sum_{j=1}^{K}\exp(s_j(x))}$ for $k = 1, 2, \dots, K$, where $s_k(x) = \theta_k{}^T \cdot x$, and input x is an n-dimension vector.
   1) To learn this Softmax Regression model, how many parameters we need to estimate? What are these parameters?
   2) Consider the cross-entropy cost function $J(\Theta)$ (see page 16 of Lecture 3-3) of m training samples $\{(x_i, y_i)\}_{i=1,2,\dots,m}$. Derive the gradient of $J(\Theta)$ regarding to θ_k as shown in page 17 of Lecture 3-3

To learn Softmax Regression Model, we need to estimate (n+1) parameters and the parameters are Θ_0, Θ_1, Θ_2, Θ_3.... Θ_n



3)

$$\hat{p}_k = \delta(s_k(x))_k = \frac{\exp(s_k(x))}{\sum_{j=1}^{k}\exp(s_j(x))}$$

where $S_k(x) = \theta_k^T \cdot x$

Training to minimize the ~~cross~~ cost function of cross entropy:-

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{k} y_k^{(i)}\log(\hat{p}_k^{(i)})$$

$$= \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{k} y_{(k)}^{(i)}\log\left(\frac{\exp(s_k(x^{(i)}))}{\sum_{j=1}^{k}\exp(s_j(x^{(i)}))}\right)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(\sum_{k=1}^{k} y_k^{(i)}\log(\exp(s_k(x^{(i)}))) - \sum_{k=1}^{k} y_{(k)}^{(i)}\log\left(\sum_{j=1}^{k}\exp(s_j(x^{(i)}))\right)\right)$$

$y_k^{(i)} = 1$ if the ith instance belongs to class k ; 0 other wise.

Softmax Regression gradient for cross-entropy cost function:

$$\nabla_{\theta_k} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{p}_k^{(i)} - y_k^{(i)} \right) x^{(i)}$$

$$= \frac{-1}{m} \sum_{i=1}^{m} x^{(i)} - \frac{1}{\sum_{j=1}^{K} \exp(\theta_j^T x^{(i)})} \exp(\theta_k^T x^{(i)}) x^{(i)}$$

$$= \frac{-1}{m} \sum_{i=1}^{m} \left(1 - \hat{p}_k^{(i)} \right) x^{(i)}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(\hat{p}_k^{(i)} - 1 \right) x^{(i)}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(\hat{p}_k^{(i)} - y_k^{(i)} \right) x^{(i)} = \nabla_{\theta_k} J(\theta)$$

Hence Proved //