Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 1

QUESTION 1

Student Name: Ajita Shree
Roll Number: 20111262
Date: October 25, 2020

Solution

Absolute Loss regression with Sparsity: The absolute loss regression problem with l1 regularization is $w_{opt} = argmin_w \sum_{n \in N}(|y_n - w^T x_n|) + \lambda ||w||_1$ where $||w||_1 = \sum_{d \in D} |w_d|, |.|$ is the absolute value of the function and $\lambda > 0$ is the regularization hyper parameter.

The above equation is convex because of the following reasons:

- Property: Sum of two convex functions is also a convex function. The shape of the function L1-norm is convex in form i.e. V-shaped with one global minima. The shape of absolute loss is also V-shaped convex in nature.
- Formally, 2nd derivative of convex functions >= 0 and hence, $d^2/dw^2(\sum_{n \in N} |y_n - w^T x_n| + \lambda ||w||_1 >= 0)$, we can see from the below sub gradient equations that 2nd derivative will always result in 0 and hence it can be said that the absolute loss regression problem with l1 regularization is convex in nature.

The derivation of the (sub)gradient vector for this model is

The (sub)gradient of the loss function can be defined on the following multiple cases e.g. 3 cases for $y_n - w^T x_n$ i.e. $y_n - w^T x_n > 0, y_n - w^T x_n = 0, y_n - w^T x_n < 0$ and 3D cases for $||w||$ term corresponding to each dimension $i.e. w_d > 0, w_d = 0, w_d < 0$.

- Sub gradient of 2nd term $\lambda ||w||$ wrt w will give a vector $\mathbf{v}$ where
  1. $\mathbf{v}_d = 1$ when $w_d > 0$
  2. $\mathbf{v}_d = -1$ when $w_d < 0$
  3. $\mathbf{v}_d = c_2$ where $c2 \in [-1, +1]$ when $w_d == 0$
- Sub gradient of 1st term $\sum_{n \in N}(|y_n - w^T x_n|)$ will be a vector $\mathbf{v}' = \sum_{n \in N} \mathbf{v}''$ where $\mathbf{v}''$ is following
  1. $-x_n$ when $y_n - w^T x_n > 0$
  2. x_n when $y_n - w^T x_n < 0$
  3. $-c_1 x_n$ where $c_1 \in [-1, 1]$ when $y_n - w^T x_n == 0$

The resultant sub-gradient vector is the $\mathbf{v}' + \mathbf{v}$ based on conditions described above

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Ajita Shree
Roll Number: 20111262
Date: October 25, 2020

QUESTION 2

Solution

Feature Masking as regularization: Linear Regression Loss is defined as $\sum_{n\in N}(y_n - w^T x_n)^2$. In this scenario, each feature x_{nd} is set to 0 with probability 1-p. It is equivalent to replacing x_n by $x'_n = x_n$ o m_n, m_n is D* 1 binary mask vector from Bernoulli (p)

Prove: Minimizing the expected value of the loss function $\sum_{n\in N}(y_n - w^T x'_n)^2$ is equivalent to minimizing a regularized loss function.

- $E[\sum_{n\in N}(y_n - w^T x'_n)^2]$
- $\sum_{n\in N} E[(y_n - w^T x'_n)^2]$
- $\sum_{n\in N} E[y_n^2 + (w^T x'_n)^2 - 2y_n w^T x'_n]$
- $\sum_{n\in N}(y_n^2 +$ E$[(w^T(x_n \text{ o } m_n))^2] - 2E[y_n w^T(x_n \text{ o } m_n)])$
- Using the formula, $E[X^2] = var[X] + \mu^2$, we can write above equation as
- $\sum_{n\in N}(y_n^2 +$ var$[(w^T(x_n \text{ o } m_n))] + E[(w^T(x_n \text{ o } m_n))]^2 - 2E[y_n w^T(x_n \text{ o } m_n)])$
- We know that the Bernoulli(p) has E[x] = p and var[X] = p(1-p) and $var[a^T x + b] = a^T \sigma a$ hence, the above equation can be written as follows
- $\sum_{n\in N}(y_n^2 + \sum_{d\in D} w_d^2 x_{nd}^2 p_d(1-p_d) + (\sum_{d\in D} w_d x_{nd} p_d)^2 - 2(\sum_d y_n w_d x_{nd} p_d)$
- Rearranging term and considering w' as a new weight vector where $w'_d = w_d p_d$
- $\sum_{n\in N}(y_n - w'^T x_n)^2 + \sum_{d\in D} {w'_d}^2 \sum_{n\in N} x_{nd}^2 (1-p_d)/(pd)$
- Above term can be approximated as
- $\sum_{n\in N}(y_n - w'^T x_n)^2 + \sum_{d\in D} {w'_d}^2$
- The right side term is an L2-norm and will act as a regularize and hence it can be concluded that minimizing the expectation of the loss function in the problem got converted into the form similar to that of Ridge regression i.e. **Liner Regression with L2 regularization**

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Ajita Shree
Roll Number: 20111262
Date: October 25, 2020

QUESTION 3

Solution

Multi-output regression with reduced number of parameters:The loss function can be written as follows:

$$\sum_{n\in N}\sum_{m\in M}(y_{nm} - w_m^T x_n)^2; y_n \in R^M \tag{1}$$

To Verify: It can further be simplified as $TRACE[(Y - XW)^T(Y - XW)]$

- **Proof by Example:** The Loss function in equation 1 is the sum of the squared loss corresponding to every element in N*M Y matrix
  1. Lets take an example to understand matrix multiplication by an example of matrix A. $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$. Square of this matrix will give $\begin{bmatrix} a^2+b^2 & ac+bd \\ ca+db & c^2+d^2 \end{bmatrix}$.
  2. Hence, if we are looking to square each term in this matrix, we need to take the sum along the diagonal of the matrix multiplication i.e. $a^2 + b^2 + c^2 + d^2 = TRACE(A)$
  3. Hence, Equation 1 can be alternatively written as $TRACE[(Y - XW)^T(Y - XW)]$
- **Formal Proof**: We know $(Y - XW)^T$ is $M * N$ and $(Y - XW)$ is $N * M$ matrix
  1. We know that $Trace(A) = \sum_i A_{ii}$. Let $\mathbb{M} = (Y - XW)^T(Y - XW)$, it will be an M*M matrix
  2. Trace($\mathbb{M}$) $= \sum_m \mathbb{M}_{mm}$
  3. $\mathbb{M}_{mm} = \sum_n (Y - XW)_{mn}(Y - XW)_{nm}$
  4. Trace($\mathbb{M}$) $= \sum_m \sum_n (y_{mn} - \sum_d x_{nd}w_{dm})(y_{mn} - \sum_d x_{nd}w_{dm})$
  5. Trace($\mathbb{M}$) $= = \sum_{n\in N}\sum_{m\in M}(y_{nm} - w_m^T x_n)^2$, Hence Proved

Derive alternating optimization algorithm, when W = B*S

- Given: W = B*S, where B = D*K and S = K*M. In this case, we need to learn K*(D+M) parameters instead of D*M parameters.
- Let L, objective function is given by $TRACE[(Y - XBS)^T(Y - XBS)]$

$$(B', S') = argmin_{B,S} TRACE[(Y - XBS)^T(Y - XBS)]$$

- Expanding the above equation, we will get

$$Trace[Y^TY - Y^TXBS - S^TB^TX^TY + S^TB^TX^TXBS]$$

- **dL/dS**, Differentiating wrt S

$$-(Y^TXB)^T - B^TX^TY - B^TX^TXBS + (B^TX^TXB)^TS \quad (2)$$

- Equating (2) to 0, we have $2(B^TX^TXB)S = 2B^TX^TY$

$$S = (B^TX^TXB)^{-1}B^TX^TY = ((XB)^TXB)^{-1}(XB)^TY$$

$$dL/dS = -2B^TX^T(Y - XBS)$$

- **dL/dB**, Differentiating wrt B

$$-(Y^TX)^TS^T - X^TYS^T + (X^TX)^TBSS^T + X^TXBSS^T = 0 \quad (3)$$

- Equating (3) to 0, we have $2(X^TXBSS^T) = 2X^TYS^T$

$$B = (X^TX)^{-1}X^TYS^T(SS^T)^{-1}$$

$$dL/dB = -2X^T(Y - XBS)S^T$$

- Algorithm for ALT-OPT, first S then B
  1. Initialize S_o, B_o at t = 0
  2. Loop until $L(B_t, S_t) - L(B_{t+1}, S_{t+1}) > \epsilon$
     - Optimize for S, using the update equation $S_{t+1} = argmin_S L(S, B_t)$
     - i.e. $S_{t+1} = ((XB_t)^TXB_t)^{-1}(XB_t)^TY$
     - Optimize for B, using the update equation $B_{t+1} = argmin_B L(S_{t+1}, B)$
     - $B_{t+1} = (X^TX)^{-1}X^TYS_{t+1}^T(S_{t+1}S_{t+1}^T)^{-1}$
     - i.e. $B_{t+1} = (X^TX)^{-1}X^TY(B_t^TX^TXB_t)^{-1}B_t^TX^TY*[(B_t^TX^TXB_t)^{-1}*B_t^TX^TYY^TXB_t*(B_t^TX^TXB_t)^{-1}]^{-1}$
     - t = t + 1

- Algorithm for ALT-OPT, first B then S
  1. Initialize S_o, B_o at t = 0
  2. Loop until $L(B_t, S_t) - L(B_{t+1}, S_{t+1}) > \epsilon$
     - Optimize for B, using the update equation $B_{t+1} = argmin_B L(S_t, B)$
     - $B_{t+1} = (X^TX)^{-1}X^TYS_t^T(S_tS_t^T)^{-1}$
     - Optimize for S, using the update equation $S_{t+1} = argmin_S L(S, B_{t+1})$
     - $S_{t+1} = ((XB_{t+1})^TXB_{t+1})^{-1}(XB_{t+1})^TY$
     - i.e. $S_{t+1} = ((S_tS_t^T)^{-1}S_tY^TX(X^TX)^{-1}X^TYS_t^T(S_tS_t^T)^{-1})^{-1}*(S_tS_t^T)^{-1}S_tY^TX(X^TX)^{-1}X^TY$
     - t = t + 1

- **Conclusion:** It will be **convenient to optimize first wrt to S followed by B** as it will be involve less no of inverse operation computation compared to when we do B followed by S as shown in the above two algorithm

Alternate Approach: Derivation: We will substitute the S value obtained in Equation 2 into Loss function equation

1. Loss function is $Trace[Y^TY - Y^TXBS - S^TB^TX^TY + S^TB^TX^TXBS]$

2. After substituting $S = (B^T X^T XB)^{-1} B^T X^T Y$, the loss function will be
3. $Trace[Y^T Y - Y^T XB(B^T X^T XB)^{-1} B^T X^T Y - ((B^T X^T XB)^{-1} B^T X^T Y)^T B^T X^T Y + ((B^T X^T XB)^{-1} * B^T X^T Y)^T * B^T X^T XB(B^T X^T XB)^{-1} B^T X^T Y]$
4. On further simplifying, we have
5. $Trace[Y^T Y - 2Y^T (XB)((XB)^T XB)^{-1} (XB)^T Y + Y^T (XB)((XB)^T XB)^{-1} (XB)^T (XB) * ((XB)^T XB)^{-1} (XB)^T Y]$
6. Using the fact $(XB)^T (XB) * ((XB)^T XB)^{-1} = I$, the equation can be simplified as follows:
7. $Trace[Y^T Y - Y^T XB(B^T X^T XB)^{-1} B^T X^T Y]$
8. Using the fact that Tr(AB) = Tr(BA), we will reshuffle the $2nd$ term in the above equation
9. Loss function, $L(S', B) = Trace[Y^T Y - (B^T X^T XB)^{-1} B^T X^T Y Y^T XB]$
10. Now, differentiating it wrt B using the formula $d/dX(Trace[(A + X^T CX)^{-1}(X^T BX)) = -2CX(A + X^T CX)^{-1} X^T BX(A + X^T CX)^{-1} + 2BX(A + X^T CX)^{-1}]$
11. $dL(S', B)/dB = 0$ will be written as
12. **Update Equation for B** $-2X^T XB(I + B^T X^T XB)^{-1} B^T X^T Y Y^T XB(I + B^T X^T XB)^{-1} + 2X^T Y Y^T XB(I + B^T X^T XB)^{-1} = 0$
13. We can see from the above equation that we did not get a closed form solution for B (as we got easily for S). Hence, we need to use iterative optimization technique in order to solve this equation to learn the only variable weight vector B.

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Ajita Shree
Roll Number: 20111262
Date: October 25, 2020

QUESTION 4

Solution

Ridge Regression using Newton's Method

$$w_{opt} = argmin_w(1/2)\sum_{n\in N}(y_n - w^T x_N)^2 + (\lambda/2)w^T w$$

- In each iterations, the update equation based on Newton's method will be as follows:
- $w_{t+1} = w_t - (H(t)^{-1})g(t)$
- Gradient of the loss function can be defined as $-\sum_{n\in N} 2(y_N - w^T x_n)x_n + \lambda w$
- Gradient is a D*1 column vector with kth entry as $-\sum_{n\in N} x_{kn}y_n + \lambda w_k + \sum_{n\in N} x_{kn}^2$
- Hessian of the loss function is the D*D diagonal matrix with each kth diagonal entry as $1/\sum_{n\in N}(x_{nk}^2) + \lambda$
- $(H(t)^{-1})g(t)$ is a D*1 column vector and kth term in the expression will be $(-\sum_{n\in N} x_{kn}y_n + \lambda w_k + \sum_{n\in N} x_{kn}^2)/(\sum_{n\in N}(x_{nk}^2) + \lambda)$
- The update equation for each iteration can be written as follows. The below equation is showing the update of kth weight entry.

$$w_{t+1,k} = w_{t,k} - (-\sum_{n\in N} x_{kn}y_n + \lambda w_k + \sum_{n\in N} x_{kn}^2 w_k)/\sum_{n\in N}(x_{nk}^2) + \lambda$$

- On further solving, the w terms will get cancelled out on the right hand side and we will get,

$$w_{t+1,k} = (\sum_{n\in N} x_{kn}y_n)/(\sum_{n\in N} x_{kn}^2 + \lambda)$$

- As in the update equation, weight term is missing on the right hand side, the solution will be converged in **1 iteration**.

Introduction to ML (CS771), Autumn 2020
Indian Institute of Technology Kanpur
Homework Assignment Number 1

Student Name: Ajita Shree
Roll Number: 20111262
Date: October 25, 2020

QUESTION 5

Solution

Given: 6 face dice is rolled N times. Let N_k be the number of times kth face has come. Let Prob(kth face) = π_k for k = 1,2,3,4,5,6 and K = 6.

a) Derive the MAP estimate assuming a appropriate conjugate prior for probability vector π. In what situations MAP is better than MLE

- Each dice roll $y_n \in 1,2,3,4,5,6$ can assumed to be generated by Multinoulli distribution with parameter π vector i.e. $p(y_n/\pi) = Multinoulli(y_n/\pi) = \prod_{k\in[1,6]} \pi_k^{1:y_n==k}$

- As the samples are i.i.d, the likelihood function can be written as

$$p(y/\pi) = \prod_{n\in N} p(y_n/\pi) = \prod_{n\in N}(\prod_{k\in[1,6]} \pi_k^{1:y_n==k})$$

- The log likelihood function can be defined as follows

$$logp(y/\pi) = \sum_{n\in N} log(\prod_{k\in[1,6]} \pi_k^{1:y_n==k})$$

- The Prior for π should be Dirichlet distribution as $0 <= \pi_k <= 1$ and $\sum_{k\in[1,6]} \pi_k = 1$

- Prior distribution $p(\pi/\alpha_1, ...\alpha_6) = (\Gamma(\sum_{k\in[1,6]} \alpha_k)/\prod_{k\in[1,6]} \Gamma(\alpha_k)) * \prod_{k\in[1,6]} p_k^{\alpha_k - 1}$

- $\pi_{MAP} = argmax_\pi(p(\pi) * p(y/\pi)/p(y)) = argmax_\pi(log(p(y/\pi)) + log(p(\pi)))$

- Substituting the values, π_{MAP} can be written as

$$argmax_\pi \sum_{n\in N} log(\prod_{k\in[1,6]} \pi_k^{1:y_n==k}) + \Gamma(\sum_{k\in[1,6]} \alpha_k)/\prod_{k\in[1,6]} \Gamma(\alpha_k) * \prod_{k\in[1,6]} \pi_k^{\alpha_k - 1}$$

- Assuming $\alpha_1, ...\alpha_k$ to be constant, adding an additional term based on Lagrangian transformation to handle the constraint, we can write the above equation approximately as

$$argmax_\pi \sum_{k\in[1,6]} N_k log(\pi_k) + Const. \sum_{k\in[1,6]} log(\pi_k^{\alpha_k - 1}) + \Phi(1 - \sum_{k\in[1,6]} \pi_k)$$

- Differentiating wrt π_k will give $(N_k/\pi_k) - \Phi + (\alpha_k - 1)/\pi_k = 0 -> N_k + \alpha_k - 1 = \phi\pi_k$

- Summing over all classes will give $\sum_{k\in[1,6]} N_k + \sum_{k\in[1,6]} \alpha_k - K = \phi \sum_{k\in[1,6]} \pi_k$

- $\phi = N + \sum_{k\in K} \alpha_k - K$, where K = 6

- Substitute value of ϕ in the above equation we will get MAP estimate of π

$$\pi_k = (N_k + \alpha_k - 1)/(N + \sum_{k \in K} \alpha_k - K)$$

- **MAP is better than MLE** when very less data points are available i.e. N is very less. It will also be better if we want to incorporate some prior information about the probability of outcomes along with the current samples.

b)Derive full posterior distribution over π. Given this posterior, can we get the MLE, MAP estimate without solving the MLE/MAP optimization problem?

$$p(\pi/y) = (p(\pi) * p(y/\pi)/p(y)) = (p(\pi) * p(y/\pi)/\int p(\pi, y)d\pi)$$

- We saw that the **numerator** $(p(\pi) * p(y/\pi)$ is as follows

$$\sum_{n \in N} (\prod_{k \in [1,6]} \pi_k^{1:y_n == k}) + (\Gamma(\sum_{k \in [1,6]} \alpha_k)/\prod_{k \in [1,6]} \Gamma(\alpha_k)) * \prod_{k \in [1,6]} \pi_k^{\alpha_k - 1}$$

$$\sum_{k \in [1,6]} (\pi_k^{N_k}) + (\Gamma(\sum_{k \in [1,6]} \alpha_k)/\prod_{k \in [1,6]} \Gamma(\alpha_k)) \sum_{k \in [1,6]} (\pi_k^{\alpha_k - 1}))$$

- The **denominator** $\int (p(\pi) * p(y/\pi)\underset{\cdot}{\pi}$ can be written as

$$\int \sum_{k \in [1,6]} (\pi_k^{N_k}) + (\Gamma(\sum_{k \in [1,6]} \alpha_k)/\prod_{k \in [1,6]} \Gamma(\alpha_k)) \sum_{k \in [1,6]} (\pi_k^{\alpha_k - 1}))d\pi$$

- It can be concluded that the posterior $p(\pi/y) \propto \prod_{k \in K} \pi^{\alpha_k + N_k - 1}$
- Our posterior is Dirichlet distribution with updated parameters i.e. $Dirichlet(\pi/\alpha_1 + N_1, ..., \alpha_K + N_K)$
- **MAP estimate** can be directly calculated as there is only one peak in the shape of Dirichlet distribution. Hence, the MAP estimate will be the mode of the Dirichlet distribution. The mode **x** of this distribution is a vector of K elements where $x_i = (\alpha'_i - 1)/\sum_{k \in K}(\alpha'_k) - K$ and $\alpha'_i = \alpha_i + N_i$. It is same to the MAP estimate calculated in the previous part $\pi_k = (N_k + \alpha_k - 1)/(N + \sum_{k \in K} \alpha_k - K)$
- **MLE solution** will be obtained by taking $\alpha_k = 1 \ \forall k \in K$ as prior will become uniform and which will be similar to not using a prior and we will get the maximum likelihood estimate by the peak of the Dirichlet distribution.The mode **x** of this distribution is a vector of K elements where $x_i = (\alpha'_i - 1)/\sum_{k \in K}(\alpha'_k) - K$ and $\alpha'_i = N_i$. Alternatively, it can be written as $\pi_{MLE} = (N_k - 1)/N - K$