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

QUESTION 1

Student Name: Nidhi Hegde
Roll Number: 180472
Date: October 30, 2020

We know that $L1$ loss is convex. Similarly, it is also known that the $L1$ regularization function is convex. Since the sum of 2 convex functions is also convex, the above objective function is convex.
This is because by defn of a convex function F, we have $F(xt + y(1-t)) \leq F(xt) + F(y(1-t))$.
Let F and H be 2 convex functions, so since this property holds for both the functions, we can say that:

$$(F+H)(xt+y(1-t)) = F(xt+y(1-t))+H(xt+y(1-t)) \leq F(xt)+F(y(1-t))+H(xt)+H(y(1-t))$$

The RHS above can also be written as: $(F + H)(xt) + (F + H)(y(1 - t))$, proving that $F + H$ is also convex Hence, we prove the sum property.
However, the given function is non-differentiable at a few points.
If we consider the given objective to be a function $L(w_1, w_2, w_3...w_D)$ with a domain $\mathbb{R}^D$ and range $\mathbb{R}$, then it is non-differentiable at the following points:

- points where either of the w_d's become 0
- point where $\boldsymbol{w = X^{-1}}Y$

Thus, we need to define a sub-gradient at these points.
We know that for a mod function $|w_d|$, where $|w_d|$ is an element of w

$$\partial|w_d| = \begin{cases} 1 & \text{if } w_d > 0 \\ -1 & \text{if } w_d < 0 \\ C_d & \text{if } w_d = 0 \;\; where \;\; C_d \in [-1, 1] \end{cases}$$

Similarly, for $f(w) = |y_n - w^T x_n|$, we have:

$$\partial f = \begin{cases} x_n & \text{if } y_n < w^T x_n \\ -x_n & \text{if } y_n > w^T x_n \\ A.x_n & \text{if } y_n = w^T x_n \;\; where \;\; A \in [-1, 1] \end{cases}$$

Using these 2 results, we proceed to calculate the expression for sub-gradient vector of the given loss function:
Here the loss function has 2 components: $L(w) = \Sigma_{n=1}^{N} |y_n - w^T x_n| + \lambda||w||_1$
The sum of sub-gradients of these 2 components is the sub-gradient of the entire function (using Sum Rule of Sub-Gradients).
The sub-gradient of this loss function will be a D-dimensional vector, as shown in the equation below, where the individual terms can be calculated using the two results above.

$$\partial L = \partial f + \lambda \mathbf{G} \tag{1}$$

$\mathbf{G}$ above is of the form $[\partial|w_1| \;\; \partial|w_2| \;\; ....\partial|w_D|]$, hence a D-dimensional vector whose individual entries are $\partial|w_d| \; for \; d = 1 \; to \; D$. The individual entries of this vector can be computed using the result for $\partial|w_d|$ above.
Thus the result for sub-gradient of L can be computed by summing up the two components dimension-wise.

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

QUESTION 2

Student Name: Nidhi Hegde
Roll Number: 180472
Date: October 30, 2020

The new loss, being defined as $\Sigma_{n=1}^{N}\left(y_n - \mathbf{w}^T(x_n \odot m_n)\right)^2$. We need to calculate the expected value of this loss. By linearity of expectation, we can write it as:

$$E\left[\Sigma_{n=1}^{N}\left(y_n - \mathbf{w}^T(x_n \odot m_n)\right)^2\right] = E\left[\left(y_1 - \mathbf{w}^T(x_1 \odot m_1)\right)^2\right] \; + ... \; E\left[\left(y_N - \mathbf{w}^T(x_N \odot m_N)\right)^2\right] \tag{2}$$

We also observe that the form is similar for each of the N training examples. So, it is sufficient to obtain the form of expectation for any one training example.
Since m_{nd} is a Bernoulli RV, we already know its expectation, i.e. $E[m_{nd}] = p$. Now, we need to translate this information to calculate the value of $E[\left(y_1 - \mathbf{w}^T(x_1 \odot m_1)\right)^2]$.

$$\begin{aligned} E\left[\left(y_1 - \mathbf{w}^T(x_1 \odot m_1)\right)^2\right] &= E\left[\left(y_1^2 + (\mathbf{w}^T(x_1 \odot m_1))^2 - 2y_1.(\mathbf{w}^T(x_1 \odot m_1))\right)\right] \\ &= E\left[\left(y_1^2 + (\Sigma_{d=1}^{D} w_d.x_{1d}.m_{1d})^2 - 2y_1.(\Sigma_{d=1}^{D} w_d.x_{1d}.m_{1d})\right)\right] \end{aligned}$$

Using linearity of expectation again,

$$\begin{aligned} &= E\left[y_1^2\right] + E\left[(\Sigma_{d=1}^{D} w_d.x_{1d}.m_{1d})^2\right] - E\left[2y_1.(\Sigma_{d=1}^{D} w_d.x_{1d}.m_{1d})\right] \\ &= y_1^2 - 2y_1.(\Sigma_{d=1}^{D} w_d.x_{1d}.E\left[.m_{1d}\right]) + \Sigma_{d=1}^{D}(w_d.x_{1d})^2.E\left[m_{1d}^2\right] + \Sigma_{d,d'}(w_d.x_{1d}).(w_{d'}.x_{1d'}).E\left[m_{1d}.m_{1d'}\right] \end{aligned}$$

Since $y1, w_d, w_{d'}, x1$ are all constants, they can be taken out. Also, we know the value of $E[m_{1d}] = E[m_{1d'}] = p$, and we know that since m_{1d} and $m_{1d'}$ are independent, $E\left[m_{1d}.m_{1d'}\right] = E\left[m_{1d}\right].E\left[m_{1d'}\right]$. Lastly, we also get $E\left[m_{1d}^2\right] = p$ on computation.
Putting these results in the above eqn gives us:

$$\begin{aligned} &= y_1^2 - 2y_1.(\Sigma_{d=1}^{D} w_d.x_{1d}.p) + \Sigma_{d=1}^{D}(w_d.x_{1d})^2.p + \Sigma_{d,d'}(w_d.x_{1d}).(w_{d'}.x_{1d'}).p \\ &= y_1^2 - 2y_1.(\Sigma_{d=1}^{D} w_d.x_{1d}.p) + p.\left(\Sigma_{d=1}^{D}(w_d.x_{1d})^2 + \Sigma_{d,d'}(w_d.x_{1d}).(w_{d'}.x_{1d'})\right) \\ &= y_1^2 - 2y_1.p.(\Sigma_{d=1}^{D} w_d.x_{1d}) + p.\left(\Sigma_{d=1}^{D} w_d.x_{1d}\right)^2 \\ &= \left(y_1 - p.\Sigma_{d=1}^{D} w_d.x_{1d}\right)^2 + p.\left(\Sigma_{d=1}^{D} w_d.x_{1d}\right)^2 - p^2.\left(\Sigma_{d=1}^{D} w_d.x_{1d}\right)^2 \\ &= \left(y_1 - p.\mathbf{w}^T x_1\right)^2 + p.(1-p).\left(\mathbf{w}^T x_1\right)^2 \end{aligned}$$

So, on combining all the training examples, we get the expected value of loss to be:

$$E\left[\Sigma_{n=1}^{N}\left(y_n - \mathbf{w}^T(x_n \odot m_n)\right)^2\right] = \Sigma_{n=1}^{N}\left(\left(y_n - p.\mathbf{w}^T x_n\right)^2 + p.(1-p).\left(\mathbf{w}^T x_n\right)^2\right)$$

Now, we need to minimize this loss, so the optimization problem becomes:

$$\arg\min_{\mathbf{w}} \Sigma_{n=1}^{N}\left(\left(y_n - p.\mathbf{w}^T x_n\right)^2 + p.(1-p).\left(\mathbf{w}^T x_n\right)^2\right)$$

This is equivalent to a regularized loss function since it contains a minimization term for a component $w^T x_n$, which prevents w from exploding. Minimizing the function above leads to the solution of the masked loss, which also acts as a regularizer.

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

QUESTION 3

Student Name: Nidhi Hegde
Roll Number: 180472
Date: October 30, 2020

We need to prove that $TRACE\left[(Y-XW)^T(Y-XW)\right]$ is equivalent to $\Sigma_{n=1}^{N}\Sigma_{m=1}^{M}(y_{nm}-w_m^Tx_n)^2$. Firstly, we know that trace is the sum of diagonals of a matrix. Multiplying the matrix transpose by itself results in the sum of squares of columns of that matrix, since the i^{th} row of matrix transpose is same as i^{th} column of the matrix.

$$
\begin{aligned}
TRACE\left[(Y-XW)^T(Y-XW)\right] &= \Sigma_{m=1}^{M}[(Y-XW)^T(Y-XW)]_{mm} \\
We\ know\ that\ \ [(Y-XW)^T(Y-XW)]_{mm} &= \Sigma_{n=1}^{N}(Y-XW)_{mn}^T(Y-XW)_{nm} \\
&= \Sigma_{n=1}^{N}(Y-XW)_{nm}^2 = \Sigma_{n=1}^{N}(Y_{nm}-(XW)_{nm})^2
\end{aligned}
$$

$(XW)_{nm}$ represents n^{th} row of X multiplied by m^{th} column of W. If we represent x_n as a column vector containing n^{th} row of X and w_m^T as m^{th} row of W^T(which is same as m^{th} column of W), we can represent $(XW)_{nm}$ as $w_m^Tx_n$.

$$
\begin{aligned}
\therefore\ \ (XW)_{nm}) &= w_m^Tx_n \\
\implies TRACE\left[(Y-XW)^T(Y-XW)\right] &= \Sigma_{m=1}^{M}\Sigma_{n=1}^{N}(Y_{nm}-w_m^Tx_n)^2
\end{aligned}
$$

It is a double summation. And we know that the order of the two summations can be swapped, since they are independent of each other here. Hence, we get:

$$
TRACE\left[(Y-XW)^T(Y-XW)\right] = \Sigma_{n=1}^{N}\Sigma_{m=1}^{M}(Y_{nm}-w_m^Tx_n)^2 \tag{3}
$$

Now, we replace W by BS for the second part.

An ALTOPT Algorithm for B and S:

1. Initialize $t=0$ and S to $S^{(t)}$
2. Solve $B^{(t+1)} = \arg\min_B TRACE\left[(Y-XBS^{(t)})^T(Y-XBS^{(t)})\right]$, keeping S fixed
3. Fix B to $B^{(t+1)}$
4. Solve $S^{(t+1)} = \arg\min_S TRACE\left[(Y-XB^{(t+1)}S)^T(Y-XB^{(t+1)}S)\right]$, keeping B fixed
5. $t=t+1$. Go to Step 2 if not converged yet

Let's try to run 1 iteration of the above algorithm. Fixing S as $S_{(0)}$, we solve for B. Taking derivative wrt B, we get:

$$
\begin{aligned}
Solving\ for\ :\ \ &\arg\min_B TRACE\left[(Y-XBS_{(0)})^T(Y-XBS_{(0)})\right] \\
= \arg\min_B TRACE&\left[Y^TY - S_{(0)}^TB^TX^TY - Y^TXBS_{(0)} + S_{(0)}^TB^TX^TXBS_{(0)}\right] \\
\implies \frac{\partial L}{\partial B} &= 2(-X^TYS_{(0)}^T + X^TXBS_{(0)}S_{(0)}^T) = 0
\end{aligned}
$$

Solving for B, We get:

$$B_{(1)} = (X^T X)^{-1}(X^T Y S_{(0)}^T)(S_{(0)} S_{(0)}^T)^{-1}$$

Moving to step 4, we fix B as $B_{(1)}$ in this step and minimize wrt S. Using (101) and (117) of Matrix cookbook, we get:

$$\begin{aligned}
Solving\ for\ :\ \arg\min_S TRACE\left[(Y - XB_{(1)}S)^T(Y - XB_{(1)}S)\right] \\
= \arg\min_S TRACE\left[Y^TY - S^TB_{(1)}^TX^TY - Y^TXB_{(1)}S + S^TB_{(1)}^TX^TXB_{(1)}S\right] \\
\implies \frac{\partial L}{\partial S} = -2B_{(1)}^TX^TY + 2(B_{(1)}^TX^TXB_{(1)})S = 0
\end{aligned}$$

Solving for S, We get:

$$S_{(1)} = (B_{(1)}^T X^T X B_{(1)})^{-1}(B_{(1)}^T X^T Y)$$

As we can see, B is more difficult to compute since there are 2 inversion terms in its expression, as opposed to S, which just contains one inversion term. This is because computing an inverse takes time, and slows down the computation of B more than S. Hence, both sub-problems above are not equally easy to compute.

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

QUESTION 4

Student Name: Nidhi Hegde
Roll Number: 180472

Date: October 30, 2020

Newton's method works by minimizing the second-order approximation of the function at each step. In class, we derived the update for Newton's Method as:

$$w^{(t+1)} = w^{(t)} - (H^{(t)})^{-1} G^{(t)} \tag{4}$$

Hence, we calculate Hessian, Gradient for the given loss function.Given function is:

$$L(w) = \frac{1}{2}.\left((y - Xw)^T(y - Xw) + \lambda w^T w\right)$$

Differentiating w.r.t w, we get G as:

$$\begin{aligned}\frac{\partial L(w)}{\partial w} &= \frac{1}{2}.\left(\frac{\partial((y - Xw)^T(y - Xw))}{\partial w} + \frac{\partial(\lambda w^T w)}{\partial w}\right)\\ &= \frac{1}{2}.\left(-2X^T(y - Xw) + 2\lambda I w\right)\end{aligned}$$

The last line is obtained by using Eqn (84) and (81) of Matrix cookbook.
Now, as we have the gradient. So, Hessian is calculated by differentiating the gradient again w.r.t. w

$$\frac{\partial^2 L(w)}{\partial w^2} = X^T X + \lambda I$$

We observe that Hessian is independent of w. The above equation is obtained using the differentiation rule that $\frac{\partial(Aw)}{\partial w} = A^T$, where A is a constant matrix. Now, we have G and H, we use eqn 4 to obtain $w^{(t+1)}$, starting from $w^{(t)}$. We get:

$$\begin{aligned}w^{(t+1)} &= w^{(t)} - (X^T X + \lambda I)^{-1}\frac{1}{2}.\left(-2X^T(y - Xw^{(t)}) + 2\lambda I w^{(t)}\right)\\ &= w^{(t)} - (X^T X + \lambda I)^{-1}\left(-X^T(y - Xw^{(t)}) + \lambda I w^{(t)}\right)\\ &= w^{(t)}(I - (X^T X + \lambda I)^{-1}(X^T X + \lambda I)) + (X^T X + \lambda I)^{-1} X^T y\\ &= w^{(t)}(I - I) + (X^T X + \lambda I)^{-1} X^T y\\ &= (X^T X + \lambda I)^{-1} X^T y \quad \therefore \textit{ Converges as independent of } w^{(t)}\end{aligned}$$

Thus, if we start with $w^{(0)}$, we converge in 1 iteration of Newton's Method.

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

Student Name: Nidhi Hegde
Roll Number: 180472
Date: October 30, 2020

QUESTION 5

This event can be modelled by using a Multinomial distribution as the likelihood. The likelihood expression can be written as:

$$p(\mathbf{N}|\pi) = \binom{N}{N1, N2, ...N6} \prod_{k=1}^{6} \pi_k^{N_k} \tag{5}$$

Here, $\mathbf{N}$ represents the vector $[N1, N2, N3, N4, N5, N6]$ and N is a scalar that represents their sum.
An appropriate prior for this problem will be Dirichlet prior.

$$p(\pi|\alpha) = \frac{\Gamma\left(\Sigma_{k=1}^{6}\alpha_k\right)}{\prod_{k=1}^{6}\Gamma\left(\alpha_k\right)}.\prod_{k=1}^{6}\pi_k^{\alpha_k - 1} \tag{6}$$

We know that the MAP estimate can be expressed as:

$$\theta_{MAP} = \arg\min_{\theta}(NLL(\theta) - \log p(\theta)) \tag{7}$$

where $NLL(\theta)$ is the negative log likelihood term and $p(\theta)$ is the prior.
For this problem, it becomes:

$$\begin{aligned} &\arg\min_{\pi}(-\log p(\mathbf{N}|\pi) - \log p(\pi|\alpha)) \\ &= \arg\min_{\pi}(A - \Sigma_{k=1}^{6} N_k \log \pi_k - \Sigma_{k=1}^{6}(\alpha_k - 1)\log \pi_k) \\ &= \arg\min_{\pi}(-\Sigma_{k=1}^{6} N_k \log \pi_k - \Sigma_{k=1}^{6}(\alpha_k - 1)\log \pi_k) \end{aligned}$$

All the multiplied constant terms are subsumed in A. This further gives us the final optimization problem as:

$$\arg\min_{\pi}(-\Sigma_{k=1}^{6}(N_k + \alpha_k - 1)\log \pi_k)$$
$$\textit{constrained by}: \ \Sigma_{k=1}^{6}\pi_k = 1$$

We can express this constrained optimization problem as a dual by introducing Lagrange's Multiplier β. Thus, it becomes:

$$\arg\max_{\beta}\arg\min_{\pi}(-\Sigma_{k=1}^{6}(N_k + \alpha_k - 1)\log \pi_k + \beta.(\Sigma_{k=1}^{6}\pi_k - 1))$$

Differentiating wrt π_k, we get:

$$-\frac{(N_k + \alpha_k - 1)}{\pi_k} + \beta = 0 \tag{8}$$

Thus,

$$\pi_k^{opt} = \frac{(N_k + \alpha_k - 1)}{\beta} \tag{9}$$

. Putting optimal value of π_k for each k and differentiating wrt β, we get:

$$\frac{\Sigma_{k=1}^{6}(N_k + \alpha_k - 1)}{\beta} - 1 = 0 \tag{10}$$

Thus, we get:

$$\beta = \Sigma_{k=1}^{6}(N_k + \alpha_k - 1) \tag{11}$$

On putting the value of in Equation 9, we get the MAP estimate as:

$$\pi_k^{MAP} = \frac{(N_k + \alpha_k - 1)}{\Sigma_{k=1}^{6}(N_k + \alpha_k - 1)} \qquad \forall k \in [1, 6] \tag{12}$$

MAP solution is better than MLE solution if N is very less, because of lack of sufficient data. Here, the prior term implies that we carried out $\Sigma_{k=1}^{6}(\alpha_k - 1)$ extra virtual observations and got a value k on dice $\alpha_k - 1$ times for every k. Thus, if $\alpha_k = 1 \forall k$, our MAP solution will be only as good as MLE estimate. Only for $\alpha_k > 1$ will MAP give a better solution than MLE.
Now, to calculate the posterior distribution, we need to multiply prior and likelihood as the formula for posterior is:

$$p(\pi|\mathbf{N}) = \frac{p(\pi|\alpha).p(\mathbf{N}|\pi)}{p(\mathbf{N})} \tag{13}$$

We get:

$$p(\pi|\mathbf{N}) = B.\binom{N}{N1, N2, ...N6}(\prod_{k=1}^{6} \pi_k^{N_k}) \ . \ p(\pi|\alpha).\frac{\Gamma\left(\Sigma_{k=1}^{6}\alpha_k\right)}{\prod_{k=1}^{6}\Gamma\left(\alpha_k\right)}.(\prod_{k=1}^{6} \pi_k^{\alpha_k - 1})$$

where B is a proportionality constant, since the marginal likelihood has no term of π. On rearranging:

$$p(\pi|\mathbf{N}) = Const \ . \ (\prod_{k=1}^{6} \pi_k^{N_k + \alpha_k - 1})$$

This is essentially another Multinomial Distribution where parameters N_k are replaced by $N_k + \alpha_k - 1$.
Given this distribution, we can find the MAP estimate as the mode of this posterior distribution. MLE estimate can be found by setting $\alpha_k = 1$ in the MAP estimate. But with the posterior distribution alone, we can't compute MLE as the value of α_k can't be deduced by us from the given posterior distribution.