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

Student Name: Utkarsh Gupta
Roll Number: 180836
Date: October 20, 2020

QUESTION 1

The required Loss Function of ℓ_1 to optimize is given as:

$$L(\mathbf{w}) = \textstyle\sum_{n=1}^{N} |y_n - \mathbf{w}^T\mathbf{x}_n| + \lambda||\mathbf{w}||_1$$

We know that ℓ_1 norms as well as the absolute function $|\mathbf{w}|$ are convex function. If $f(x)$ is convex, $f(ax+b)$ is also convex $\implies$ $|y_n - \mathbf{w}^T\mathbf{x}_n|$ is convex. Moreover, non-negative weighted sums of convex functions are also convex. Therefore, $L(\mathbf{w})$ is a convex function.

Let $\partial L_n(\mathbf{w})$ and $\partial\,\ell_1(\mathbf{w})$ be the sub-differential of $|y_n - \mathbf{w}^T\mathbf{x}_n|$ and $||\mathbf{w}||_1$ respectively. By using the sum property of sub-differentials, we have $\partial L(\mathbf{w}) = \sum_{n=1}^{N} \partial L_n(\mathbf{w}) + \lambda\partial\,\ell_1(\mathbf{w})$.

Using the affine transformation rule, $\partial\, L_n(\mathbf{w}) = -\mathbf{x}_n\partial\,|\mathbf{t}|$ where $\mathbf{t} = y_n - \mathbf{w}^T\mathbf{x}_n$, which gives:

$$\partial L_n(\mathbf{w}) = \begin{cases} -\mathbf{x}_n & \text{if } y_n - \mathbf{w}^T\mathbf{x}_n > 0 \\ \mathbf{x}_n & \text{if } y_n - \mathbf{w}^T\mathbf{x}_n < 0 \\ k\mathbf{x}_n & \text{if } y_n - \mathbf{w}^T\mathbf{x}_n = 0, \text{ where k} \in [-1, +1] \end{cases}$$

The ℓ_1 norm of a $D - \mathit{dimensional}$ vector is the cartesian product of D-independent absolute functions (B_d). In other words, B_d corresponds to the d^{th} entry of $\partial\ell_1$. Thus, $\partial\,\ell_1(\mathbf{w}) = \prod_{d=1}^{D} B_d$ such that $B_d = \partial\,|w_d|$.

$$B_d = \begin{cases} \{-1\} & \text{if } w_d > 0 \\ \{1\} & \text{if } w_d < 0 \\ \{k\} & \text{if } w_d = 0, \text{ where k} \in [-1, +1] \end{cases}$$

Therefore, we now get the required sub-differential $\partial L(\mathbf{w})$ which gives the general expression of the sub-gradients. Setting the constants (randomly or otherwise) gives us a sub-gradient.

$$\boxed{\partial L(\mathbf{w}) = \left\{ \sum_{n=1}^{N} a_n\mathbf{x}_n + \lambda\mathbf{b} \,|\, a_n \in \partial\,|y_n - \mathbf{w}^T\mathbf{x}_n| \;\&\; \mathbf{b} \in \partial\,\ell_1(\mathbf{w}) \right\}}$$

Algorithm 1: Sub-gradient for absolute loss regression with ℓ_1 regularization

```
Function abs_grad($x$):
    if $x > 0$ then
        return 1
    else if $x < 0$ then
        return $-1$
    else
        return $k$     // $k \in [-1, 1]$
    end
return
Function sub_grad($\mathbf{w}$, $\mathbf{x}$, $\mathbf{y}$, $\lambda$):
    $\mathbf{g}_{sub}(\mathbf{w})$ := zero $D$ x 1 vector
    for $n = 1$ to $N$ do
        $\mathbf{g}_{sub}(\mathbf{w})$ += $-\mathbf{x}_n \cdot$ abs_grad($y_n - \mathbf{w}^T\mathbf{x}_n$)
    end
    $\mathbf{b}$ := empty $D$ x 1 vector
    for $d = 1$ to $D$ do
        $\mathbf{b}[d] = \lambda \cdot$ abs_grad($w_d$)
    end
    $\mathbf{g}_{sub}(\mathbf{w})$ += $\mathbf{b}$
return $\mathbf{g}_{sub}(\mathbf{w})$
```

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

Student Name: Utkarsh Gupta
Roll Number: 180836
Date: October 20, 2020

QUESTION 2

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathbb{E}\left[\sum_{n=1}^{N}\left(y_n - \mathbf{w}^T\tilde{\mathbf{x}}_n\right)^2\right]$$

Simplifying the objective and applying Linearity of Expectations, we get:

$$\tilde{L}(\mathbf{w}) = \sum_{n=1}^{N}\mathbb{E}\left[y_n^2 + \left(\mathbf{w}^T\tilde{\mathbf{x}}_n\right)^2 - 2y_n\mathbf{w}^T\tilde{\mathbf{x}}_n\right]$$

Also, $\mathbb{E}\left[y_n^2 + \left(\mathbf{w}^T\tilde{\mathbf{x}}_n\right)^2 - 2y_n\mathbf{w}^T\tilde{\mathbf{x}}_n\right] = y_n^2 - 2y_n\mathbb{E}\left[\mathbf{w}^T\tilde{\mathbf{x}}_n\right] + \mathbb{E}\left[\left(\mathbf{w}^T\tilde{\mathbf{x}}_n\right)^2\right]$.

Now, $\mathbf{w}^T\tilde{\mathbf{x}}_n = \sum_{d=1}^{D} w_d\tilde{x}_{nd} = \sum_{d=1}^{D} w_d x_{nd} r_{nd}$, where $r_{nd} \sim \textsc{Bernoulli}(p)$ is the mask bit.

So, $\mathbb{E}\left[\mathbf{w}^T\tilde{\mathbf{x}}_n\right] = \sum_{d=1}^{D} w_d x_{nd}\mathbb{E}\left[r_{nd}\right] = p\sum_{d=1}^{D} w_d x_{nd} = p\mathbf{w}^T\mathbf{x}_n$.

All mask bits are independent so, $\mathbb{E}\left[r_{ni}r_{nj}\right] = \mathbb{E}\left[r_{ni}\right]\mathbb{E}\left[r_{nj}\right] = p^2$ and $\mathbb{E}\left[r_{nd}^2\right] = p$.

$\therefore \mathbb{E}\left[\left(\mathbf{w}^T\tilde{\mathbf{x}}_n\right)^2\right] = \sum_{d=1}^{D} w_d^2 x_{nd}^2 p + 2\sum_{i=1}^{D}\sum_{j=1}^{D} w_i w_j x_{ni} x_{nj} p^2 = p^2(\mathbf{w}^T\mathbf{x}_n)^2 + p(1-p)\sum_{d=1}^{D} w_d^2 x_{nd}^2$.

Hence, we obtain $\mathbb{E}\left[\left(y_n - \mathbf{w}^T\tilde{\mathbf{x}}_n\right)^2\right] = y_n^2 - 2y_n p\mathbf{w}^T\mathbf{x}_n + p^2(\mathbf{w}^T\mathbf{x}_n)^2 + p(1-p)\sum_{d=1}^{D} w_d^2 x_{nd}^2$.

The required regularised Loss Function $\tilde{L}(\mathbf{w})$:

$$\tilde{L}(\mathbf{w}) = \sum_{n=1}^{N}\left(y_n - p\mathbf{w}^T\mathbf{x}_n\right)^2 + p(1-p)\sum_{n=1}^{N}\sum_{d=1}^{D} w_d^2 x_{nd}^2$$

Since, $\mathbf{w} \in \mathbb{R}^D$ and $\arg\min_{\mathbf{w}} \tilde{L}(\mathbf{w}) = \arg\min_{\mathbf{w}} \tilde{L}(\frac{\mathbf{w}}{p})$ we can replace $\mathbf{w}$ by $\frac{\mathbf{w}}{p}$, without the loss of generality. Also, $\sum_{d=1}^{D} w_d^2 x_{nd}^2 = \mathbf{w}^T\text{diag}\left(\mathbf{x}_n\mathbf{x}_n^T\right)\mathbf{w}$. $\text{diag}\left(\mathbf{x}_n\mathbf{x}_n^T\right)$ is a diagonal matrix with each diagonal entry as x_{nd}^2.

$$\boxed{\tilde{L}(\mathbf{w}) = \sum_{n=1}^{N}\left(y_n - \mathbf{w}^T\mathbf{x}_n\right)^2 + \frac{1-p}{p}\sum_{n=1}^{N}\mathbf{w}^T\text{diag}\left(\mathbf{x}_n\mathbf{x}_n^T\right)\mathbf{w}}$$

In matrix form, the same function can be written as:

$$\boxed{\tilde{L}(\mathbf{w}) = ||\mathbf{y} - \mathbf{X}\mathbf{w}||^2 + \left(\frac{1-p}{p}\right)\mathbf{w}^T\text{diag}\left(\mathbf{X}^T\mathbf{X}\right)\mathbf{w}}$$

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

Student Name: Utkarsh Gupta
Roll Number: 180836
Date: October 20, 2020

QUESTION 3

To Verify: $\sum_{n=1}^{N}\sum_{m=1}^{M}\left(y_{nm}-\mathbf{w}_m^T\mathbf{x}_n\right)^2 = \text{TRACE}\left[\left(\mathbf{Y}-\mathbf{XW}\right)^T\left(\mathbf{Y}-\mathbf{XW}\right)\right]$.

$$[\mathbf{Y}-\mathbf{XW}]_{ij} = y_{ij} - \sum_{d=1}^{D} x_{id}w_{dj} \qquad [\mathbf{Y}-\mathbf{XW}]_{ij}^T = y_{ji} - \sum_{d=1}^{D} x_{di}w_{jd}$$

$$\left[\left(\mathbf{Y}-\mathbf{XW}\right)^T\left(\mathbf{Y}-\mathbf{XW}\right)\right]_{ij} = \sum_{n=1}^{N}\left[\left(y_{ni}-\sum_{d=1}^{D}x_{nd}w_{di}\right)\left(y_{nj}-\sum_{d=1}^{D}x_{nd}w_{dj}\right)\right]$$

$$\implies \text{TRACE}\left[\left(\mathbf{Y}-\mathbf{XW}\right)^T\left(\mathbf{Y}-\mathbf{XW}\right)\right] = \sum_{m=1}^{M}\left[\left(\mathbf{Y}-\mathbf{XW}\right)^T\left(\mathbf{Y}-\mathbf{XW}\right)\right]_{mm}$$

$$= \sum_{n=1}^{N}\sum_{m=1}^{M}(y_{nm}-\sum_{d=1}^{D}x_{nd}w_{dm})^2 = \sum_{n=1}^{N}\sum_{m=1}^{M}(y_{nm}-\mathbf{w}_m^T\mathbf{x}_n)^2$$

HENCE VERIFIED.

Now, as described in the question, we need to apply alternate optimisation algorithm on the objective $L(\mathbf{B},\mathbf{S}) = \text{TRACE}\left[\left(\mathbf{Y}-\mathbf{XBS}\right)^T\left(\mathbf{Y}-\mathbf{XBS}\right)\right]$.

$$\left\{\hat{\mathbf{B}},\hat{\mathbf{S}}\right\} = \arg\min_{\mathbf{B},\mathbf{S}} \text{TRACE}\left[\left(\mathbf{Y}-\mathbf{XBS}\right)^T\left(\mathbf{Y}-\mathbf{XBS}\right)\right]$$

We know that $\text{TRACE}\left[\left(\mathbf{Y}-\mathbf{XBS}\right)^T\left(\mathbf{Y}-\mathbf{XBS}\right)\right] = \text{TRACE}\left[\left(\mathbf{Y}-\mathbf{XBS}\right)\left(\mathbf{Y}-\mathbf{XBS}\right)^T\right]$. Using this, and equation 119 from Matrix Cookbook, we get:

$$\nabla_{\mathbf{B}}L(\mathbf{B},\mathbf{S}) = -2\mathbf{X}^T\left(\mathbf{Y}-\mathbf{XBS}\right)\mathbf{S}^T$$

$$\nabla_{\mathbf{S}}L(\mathbf{B},\mathbf{S}) = -2\mathbf{B}^T\mathbf{X}^T\left(\mathbf{Y}-\mathbf{XBS}\right)$$

The objective $L_{\mathbf{B}}(\mathbf{S})$ corresponding to $\nabla_{\mathbf{S}}L(\mathbf{B},\mathbf{S}) = \nabla L_{\mathbf{B}}(\mathbf{S})$ is clearly convex as it reduces to single-output regression if we assume $\mathbf{XB}=\mathbf{G}$. So, we can set the derivative to 0 and obtain $\mathcal{S}$.

$$\boxed{\mathcal{S} = \left[\left(\mathbf{XB}\right)^T\left(\mathbf{XB}\right)\right]^{-1}\left(\mathbf{XB}\right)^T\mathbf{Y}}$$

In order to get $L_{\mathbf{Z}}(\mathbf{S})$, we can try to write $\mathbf{Y}=\mathbf{SB}$ and $\mathbf{E}=\mathbf{Z}-\mathbf{XB}$, which yields:

$$L_{\mathbf{S}}(\mathbf{B}) = \text{TRACE}\left[\mathbf{S}^T\left(\mathbf{Z}-\mathbf{XB}\right)^T\left(\mathbf{Z}-\mathbf{XB}\right)\mathbf{S}\right] = \text{TRACE}\left[\mathbf{SS}^T\mathbf{E}^T\mathbf{E}\right]$$

Let's look at the $k^{th}D$xD Hessian for $k=1,2,\ldots,K$.
$e_{ij} = z_{ij}-\sum_{d=1}^{D}x_{id}b_{dj}$, $\left[\mathbf{E}^T\mathbf{E}\right]_{ij} = \sum_n e_{ni}e_{nj}$ and $\left[\mathbf{SS}^T\right]_{ij} = \sum_m s_{im}s_{jm}$.

$$L^k_{\mathbf{S}}(\mathbf{B}) = \sum_{k'} \sum_m s_{km} s_{k'm} \sum_n e_{nk'} e_{nk}$$

$$\nabla^2 L^k_{\mathbf{S}}(\mathbf{B})_{ij} = \sum_{k'} \sum_m s_{km} s_{k'm} \sum_n \frac{\partial^2}{\partial b_{ik} \partial b_{jk}} e_{nk'} e_{nk}$$

$$\frac{\partial^2}{\partial b_{ik} \partial b_{jk}} e_{nk'} e_{nk} = 2 x_{ni} x_{nj} \delta_{k'k}$$

$$\nabla^2 L^k_{\mathbf{S}}(\mathbf{B})_{ij} = 2 \sum_m s_{km} s_{km} \sum_n x_{ni} x_{nj} = 2 \left[\mathbf{X}^T \mathbf{X}\right]_{ij} \left[\mathbf{S}\mathbf{S}^T\right]_{kk}$$

So, $\nabla^2 L^k_{\mathbf{S}}(\mathbf{B}) = 2 \left[\mathbf{S}\mathbf{S}^T\right]_{kk} \mathbf{X}^T \mathbf{X}$. Since, a vector valued function is convex only when each of it's individual components are convex, $L_{\mathbf{S}}(\mathbf{B})$ will be convex iff all the diagonal elements of $\mathbf{S}\mathbf{S}^T$ are non-negative. This is indeed the case as, $\mathbf{S}\mathbf{S}^T$ is PSD $\implies \nabla^2 L^k_{\mathbf{S}}(\mathbf{B})$ is also PSD.

Thus, $L_{\mathbf{S}}(\mathbf{B})$ is convex and $\mathcal{B}$ can be obtained by setting $\nabla_{\mathbf{B}} L(\mathbf{B}, \mathbf{S}) = \nabla L_{\mathbf{S}}(\mathbf{B})$ to 0.

$$\mathbf{X}^T (\mathbf{Y} - \mathbf{X}\mathcal{B}\mathbf{S}) \mathbf{S}^T = 0 \implies \mathbf{X}^T \mathbf{Y} \mathbf{S}^T = \mathbf{X}^T \mathbf{X} \mathcal{B} \mathbf{S} \mathbf{S}^T$$

$$\boxed{\mathcal{B} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{Y} \mathbf{S}^T \left(\mathbf{S}\mathbf{S}^T\right)^{-1}}$$

After multiple iterations, $\mathcal{B}$ & $\mathcal{S}$ will converge to give $\hat{\mathbf{B}}$ & $\hat{\mathbf{S}}$.

Algorithm 2: Alternating Optimisation to get $\hat{\mathbf{B}}$ & $\hat{\mathbf{S}}$

```
t := 0
Initialise $\mathbf{B}^{(0)}$ & $\mathbf{S}^{(0)}$
repeat
    $\mathbf{B}^{(t+1)} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{Y} \mathbf{S}^{(t)T} \left(\mathbf{S}^{(t)} \mathbf{S}^{(t)T}\right)^{-1}$
    $\mathbf{S}^{(t+1)} = \left[\left(\mathbf{X}\mathbf{B}^{(t+1)}\right)^T \left(\mathbf{X}\mathbf{B}^{(t+1)}\right)\right]^{-1} \left(\mathbf{X}\mathbf{B}^{(t+1)}\right)^T \mathbf{Y}$
    t := t+1
until convergence
return $\mathbf{B}^{(t)}$ & $\mathbf{S}^{(t)}$
```

As far as complexity is concerned, calculating $\hat{\mathbf{S}}$ is easier than calculating $\hat{\mathbf{B}}$. This is because, the size of matrix inverted in $\hat{\mathbf{S}}$ can be small if K is chosen to be small. On the other hand, the size of matrix inverted in $\hat{\mathbf{B}}$ will be large if D is large.

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

Student Name: Utkarsh Gupta
Roll Number: 180836
Date: October 20, 2020

QUESTION 4

The Ridge Regression problem has already been discussed in class. The Loss Function $L(\mathbf{w})$ is given by: $L(\mathbf{w}) = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \right\}$. It is convex, twice differentiable, $\nabla L(\mathbf{w}) = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_D\right)\mathbf{w} - \mathbf{X}^T\mathbf{y}$ and $\nabla^2 L(\mathbf{w}) = \mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_D$. Moreover, the optimal weight $\hat{\mathbf{w}}$ is given by $\hat{\mathbf{w}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_D\right)^{-1}\mathbf{X}^T\mathbf{y}$.

Newton's Method iteratively minimizes the second-order (quadratic) approximation (L_2) of the original Loss function.

$$L_2(\mathbf{w}, \mathbf{w}^{(t)}) = L(\mathbf{w}^{(t)}) + \nabla L(\mathbf{w}^{(t)})^T \left(\mathbf{w} - \mathbf{w}^{(t)}\right) + \frac{1}{2}\left(\mathbf{w} - \mathbf{w}^{(t)}\right) \nabla^2 L(\mathbf{w}^{(t)})^T \left(\mathbf{w} - \mathbf{w}^{(t)}\right)$$

The next weight vector can be found using $\mathbf{w}^{(t+1)} = \arg\min_{\mathbf{w}} L_2(\mathbf{w}, \mathbf{w}^{(t)})$. Now we have, $\nabla L_2(\mathbf{w}) = \nabla L(\mathbf{w}^{(t)}) + \nabla^2 L(\mathbf{w}^{(t)})\left(\mathbf{w} - \mathbf{w}^{(t)}\right)$ and $\nabla^2 L_2(\mathbf{w}) = \nabla^2 L(\mathbf{w}^{(t)})$. For $\lambda > 0$, $\nabla^2 L_2(\mathbf{w})$ is a Positive Semi-Definite matrix $\implies L_2(\mathbf{w}, \mathbf{w}^{(t)})$ is convex.

Setting $\nabla_{\mathbf{w}} L_2(\mathbf{w}) = 0$, the vector minimizing $L_2(\mathbf{w}, \mathbf{w}^{(t)})$ is given by $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \left(\nabla^2 L(\mathbf{w}^{(t)})\right)^{-1} \nabla L(\mathbf{w}^{(t)})$. This is also the required general update-step equation. Substituting the derivatives for the given $L(\mathbf{w})$ we get:

$$\boxed{\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_D\right)^{-1} \left\{ \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_D\right)\mathbf{w}^{(t)} - \mathbf{X}^T\mathbf{y} \right\}}$$

Upon simplification, we get $\mathbf{w}^{(t+1)} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_D\right)^{-1}\mathbf{X}^T\mathbf{y}$. This is the expected closed form solution and independent of $\mathbf{w}^{(t)}$. Thus, Newton's Method converges to the optimal solution in a single iteration in the given problem. This happens because the loss function is smooth and quadratic.

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

Student Name: Utkarsh Gupta
Roll Number: 180836
Date: October 20, 2020

QUESTION 5

Let $K = 6$ for this question. All observations are assumed to be i.i.d. The parameter we need to estimate is $\boldsymbol{\pi} = [\pi_1, \pi_2, \dots, \pi_K]$. Let us first get the Maximum Likelihood Estimate or $\boldsymbol{\pi}_{MLE}$. The likelihood we are going to use is Multinoulli.

$$p(y|\boldsymbol{\pi}) = \prod_{k=1}^{K} \pi_k^{N_k}$$

N_k corresponds to the number of times k showed up on the dice. To get the MLE estimate, we can minimize the negative log-likelihood subjected to the condition $\sum_{k=1}^{K} \pi_k = 1$. This is a constrained optimisation problem, so we use Lagrangian Optimisation.

$$\mathcal{L}(\boldsymbol{\pi}, \lambda) = -\sum_{k=1}^{K} N_k log\left(\pi_k\right) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1\right)$$

We know that $-log(x)$ is a convex function, and non-negatively weighted sums of convex function is also convex. Therefore, the above Lagrangian is surely convex for $\lambda > 0$. Now, the optimal solution to $\arg\min_{\boldsymbol{\pi}} \mathcal{L}(\boldsymbol{\pi}, \lambda)$ can be found by equating $\nabla \mathcal{L}(\boldsymbol{\pi}, \lambda)$ to zero.

$$\boldsymbol{\pi}_{MLE} = \left[\frac{N_1}{N}, \frac{N_2}{N}, \dots, \frac{N_K}{N}\right]$$

where $N = \sum_{k=1}^{K} N_k$. For MAP Estimation and finding Posterior Distribution, the appropriate conjugate prior ($p(\boldsymbol{\pi})$) would be **Dirchlet** distribution (similar to how we took Beta prior in case of Bernoulli likelihood). The Posterior distribution is given by $p(\boldsymbol{\pi}|y) = \frac{p(y|\boldsymbol{\pi})p(\boldsymbol{\pi})}{p(y)}$.

$$\boldsymbol{\pi}_{MAP} = \arg\max_{\pi_k} p(\boldsymbol{\pi}|y)$$

subjected to $\sum_{k=1}^{K} \pi_k = 1$. Dirchlet distribution is given by $p(\boldsymbol{\pi}|\boldsymbol{\alpha})$. $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_K]$ is a constant hyper-parameter vector.

$$p(\boldsymbol{\pi}|\boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\prod_{k=1}^{K} \Gamma\alpha_k} \prod_{k=1}^{K} \pi_k^{\alpha_k - 1}$$

The MAP objective can be simplified by taking negative log of the posterior. Minimizing the objective will yield the required MAP solution as, the Lagrangian will be convex and the global optima obtained will therefore, be a minima.

$$-log(p(\boldsymbol{\pi}|y)) = log(p(y)) - log(\frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\prod_{k=1}^{K} \Gamma\alpha_k}) - \sum_{k=1}^{K}(\alpha_k - 1)log(\pi_k) - \sum_{k=1}^{K}(N_k)log(\pi_k)$$

ignoring the terms that don't depend on π_k and writing the Lagrangian for the constrained optimisation problem at hand:

$$\mathcal{L}(\boldsymbol{\pi}, \lambda) = -\sum_{k=1}^{K}(N_k + \alpha_k - 1)log\,(\pi_k) + \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right)$$

Using arguments similar to the ones made for MLE problem, we know that the Lagrangian is convex for $\lambda > 0$. Let $\alpha = \sum_{k=1}^{K}\alpha_k$,

$$\frac{\partial}{\partial \pi_k}\mathcal{L}(\boldsymbol{\pi}, \lambda) = -\frac{N_k + \alpha_k - 1}{\pi_k} + \lambda = 0$$

$$\implies \lambda = \frac{N_k + \alpha_k - 1}{\pi_k} \implies \pi_k = \frac{N_k + \alpha_k - 1}{\lambda}$$

Substituting π_k into the constraint $\sum_{k=1}^{K}\pi_k = 1$ we get $\lambda = N + \alpha - K$. Hence, we get the required MAP parameter (global minima of the constrained MAP objective),

$$\boldsymbol{\pi}_{MAP} = \left[\frac{N_1 + \alpha_1 - 1}{N + \alpha - K}, \frac{N_2 + \alpha_2 - 1}{N + \alpha - K}, \ldots, \frac{N_K + \alpha_K - 1}{N + \alpha - K}\right]$$

The MAP estimate would be better than the MLE estimate in situations where the observations maximized upon aren't very likely themselves, which could happen when N is too small or N_ks are disproportionate (for example, in the cases where we have little to no examples from some classes).

In such cases, MAP estimate makes better predictions as it has access to the information contained in $\boldsymbol{\alpha}$. Here, it $(\alpha_k - 1)$ can be interpreted as the number of times the k^{th} number showed up on the dice before the experiment began.

Now, to get the full Posterior Distribution, we require the marginal likelihood $p(y)$.

$$p(y) = \int p(\boldsymbol{\pi})p(y|\boldsymbol{\pi})d\boldsymbol{\pi}$$

$$\implies p(y) = \int\limits_{\sum_{k=1}^{K}\pi_k = 1} \prod_{k=1}^{K}\pi_k^{N_k}\frac{\Gamma\left(\sum_{k=1}^{K}\alpha_k\right)}{\prod_{k=1}^{K}\Gamma\alpha_k}\prod_{k=1}^{K}\pi_k^{\alpha_k - 1}d\boldsymbol{\pi}$$

$$\implies p(y) = \frac{\Gamma\left(\sum\limits_{k=1}^{K}\alpha_k\right)}{\prod_{k=1}^{K}\Gamma\alpha_k}\int\limits_{\sum_{k=1}^{K}\pi_k = 1}\prod_{k=1}^{K}\pi_k^{N_k + \alpha_k - 1}d\boldsymbol{\pi} \tag{1}$$

We know that Dirchlet and Multinoulli are Conjugate Distributions, we can obtain $p(y)$ without actually solving the integral. Let $\beta_k = N_k + \alpha_k$, we can get a Dirchlet distribution with hyper-parameter $\boldsymbol{\beta}$:

$$p(\boldsymbol{\pi}|\boldsymbol{\beta}) = \frac{\Gamma\left(\sum_{k=1}^{K}\beta_k\right)}{\prod_{k=1}^{K}\Gamma\beta_k}\prod_{k=1}^{K}\pi_k^{\beta_k - 1}$$

Since, this is a probability distribution, integrating it over all values of $\boldsymbol{\pi}$ will give 1 (subjected to the condition $\sum_{k=1}^{K} \pi_k = 1$).

$$\int p(\boldsymbol{\pi}|\boldsymbol{\beta})d\boldsymbol{\pi} = \int \frac{\Gamma\left(\sum_{k=1}^{K} \beta_k\right)}{\prod_{k=1}^{K} \Gamma\beta_k} \prod_{k=1}^{K} \pi_k^{\beta_k - 1} d\boldsymbol{\pi} = 1$$

$$\implies \int \prod_{k=1}^{K} \pi_k^{\beta_k - 1} d\boldsymbol{\pi} = \frac{\prod_{k=1}^{K} \Gamma\beta_k}{\Gamma\left(\sum_{k=1}^{K} \beta_k\right)} \tag{2}$$

Substituting 2 in 1 we get:

$$p(y) = \frac{\left\{\Gamma\left(\sum\limits_{k=1}^{K} \alpha_k\right)\right\}\left\{\prod_{k=1}^{K} \Gamma\left(N_k + \alpha_k\right)\right\}}{\left\{\prod_{k=1}^{K} \Gamma\alpha_k\right\}\left\{\Gamma\left(\sum_{k=1}^{K}\left(N_k + \alpha_k\right)\right)\right\}} \tag{3}$$

Thus, the full posterior distribution can be written as:

$$p(y) = \frac{\left\{\prod_{k=1}^{K} \Gamma\alpha_k\right\}\left\{\Gamma\left(\sum_{k=1}^{K}\left(N_k + \alpha_k\right)\right)\right\}}{\left\{\Gamma\left(\sum\limits_{k=1}^{K} \alpha_k\right)\right\}\left\{\prod_{k=1}^{K} \Gamma\left(N_k + \alpha_k\right)\right\}} \prod_{k=1}^{K} \pi_k^{N_k} \frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\prod_{k=1}^{K} \Gamma\alpha_k} \prod_{k=1}^{K} \pi_k^{\alpha_k - 1}$$

Simplifying we get:

$$\boxed{p(\boldsymbol{\pi}|y) = \frac{\Gamma\left(\sum_{k=1}^{K}\left(N_k + \alpha_k\right)\right)}{\prod_{k=1}^{K} \Gamma\left(N_k + \alpha_k\right)} \prod_{k=1}^{K} \pi_k^{(N_k + \alpha_k) - 1}}$$

with the constraint that $\sum_{k=1}^{K} \pi_k = 1$.

Both MAP and MLE estimates can be found from the Posterior directly. MAP distribution is nothing but the mode of this distribution. MLE can be found by taking a uniform prior (setting all $\alpha_k = 1$).

In order to calculate the mode, we replace $\pi_K = 1 - \sum_{k=1}^{K-1} \pi_k$ in order to account for the constraint $\sum_{k=1}^{K} \pi_k = 1$. Now, take negative log and let $\beta_k = N_k + \alpha_k - 1$.

$$-log(p(\boldsymbol{\pi}|y)) = -\beta_k log(\pi_k) - \beta_K log(1 - \sum_{k=1}^{K-1} \pi_k)$$

Differentiating for k=1, . . . , K-1, and setting the derivatives to 0 we get $\frac{\beta_k}{\pi_k} = \frac{\beta_K}{1 - \sum_{k=1}^{K-1} \pi_k} \implies \pi_k = \frac{\beta_k}{\beta_K} \pi_K$. Using $\sum_{k=1}^{K} \pi_k = 1$, we get $\pi_k = \frac{\beta_k}{\beta}$ where $\beta = N + \alpha - K$. This is the required MAP as obtained earlier.

Setting $\alpha_k = 1$, we get $\beta_k = N_k \implies \pi_k = \frac{N_k}{N}$, the MLE estimate as derived earlier.