Instructor:
Prof. Nakul Verma
COMS W4771–Spring, 2022

Student:
Vishweshwar Tyagi
vt2353 @columbia.edu

Homework 3

Due: 11:59pm, April 10, 2022

Homework Problems

Solution-1:

(i) From the pseudo-code we have,

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i f_t(x_i))}{\sum_{i=1}^{m} D_t(i)\exp(-\alpha_t y_i f_t(x_i))} \quad \forall\, t = 1, 2, \ldots T$$

Using this recurrence repeatedly with $D_1(i) = \frac{1}{m}$, we get,

$$\begin{aligned} D_{T+1}(i) &= \frac{D_T(i)\exp(-\alpha_T y_i f_T(x_i))}{\sum_{i=1}^{m} D_T(i)\exp(-\alpha_T y_i f_T(x_i))} \\ &= \frac{D_{T-1}(i)\exp\big(-y_i \sum_{t=T-1}^{T} \alpha_t f_t(x_i)\big)}{\prod_{t=T-1}^{T}\sum_{i=1}^{m} D_t(i)\exp(-\alpha_t y_i f_t(x_i))} \\ &\cdots \\ &= \frac{D_1(i)\exp\big(-y_i \sum_{t=1}^{T} \alpha_t f_t(x_i)\big)}{\prod_{t=1}^{T}\sum_{i=1}^{m} D_t(i)\exp(-\alpha_t y_i f_t(x_i))} \\ &= \frac{1}{m}\frac{1}{\prod_{t=1}^{T} Z_t}\exp(-y_i g(x_i)) \end{aligned} \tag{1}$$

(ii) From (1), we have

$$\frac{1}{m}\sum_{i=1}^{m}\exp(-y_i g(x_i)) = \prod_{t=1}^{T} Z_t \sum_{i=1}^{m} D_{T+1}(i)$$
$$= \prod_{t=1}^{T} Z_t$$

where we used $\sum_{i=1}^{m} D_{T+1}(i) = 1$ because $D_{T+1}(i)$ are normalized for each i

Using this, and the fact $\sum_{i=1}^{m} \mathbf{1}[y_i \neq \text{sign}(g(x_i))] \leq \sum_{i=1}^{m} \exp(-y_i g(x_i))$ ($0-1$ loss is upper bounded by exponential loss), we get,

$$\begin{aligned}\text{err}(g) &= \frac{1}{m}\sum_{i=1}^{m}\mathbf{1}[y_i \neq \text{sign}(g(x_i))] \\ &\leq \frac{1}{m}\sum_{i=1}^{m}\exp(-y_i g(x_i)) \\ &\leq \prod_{t=1}^{T} Z_t \end{aligned} \tag{2}$$

(iii) Note that, $\forall\, t = 1, 2 \ldots T$

$$\begin{aligned}\sum_{i=1}^{m} D_t(i)\mathbf{1}[y_i = f_t(x_i)] &= \sum_{i=1}^{m} D_t(i) - \sum_{i=1}^{m} D_t(i)\mathbf{1}[y_i \neq f_t(x_i)] \\ &= \sum_{i=1}^{m} D_t(i) - \epsilon_t \\ &= 1 - \epsilon_t \end{aligned} \tag{3}$$

where we used $\sum_{i=1}^{m} D_t(i) = 1$ because at the end of each iteration (in the pseudo-code), $D_t(i)$ are normalized for each $i = 1, 2 \ldots m$ and initially we began with $D_1(i) = \frac{1}{m}\ \ \forall\, i = 1, 2 \ldots m$, which again sum to 1 over i

Further, note that $y_i, f_t(x_i) \in \{\pm 1\}$.

Hence, $y_i \neq f_t(x_i) \iff y_i f_t(x_i) = -1$ and $y_i = f_t(x_i) \iff y_i f_t(x_i) = 1$ (4)

Also, $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \rightarrow \exp(\alpha_t) = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$ and $\rightarrow \exp(-\alpha_t) = \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}$

Uisng this, together with (3) and (4), from the pseudo-code, we have $\forall\ t = 1, 2 \dots T$

$$\begin{aligned}
Z_t &= \sum_{i=1}^{m} D_{t+1}(i) \\
&= \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i f_t(x_i)) \\
&= \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i f_t(x_i)) \mathbf{1}[y_i \neq f_t(x_i)] + \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i f_t(x_i)) \mathbf{1}[y_i = f_t(x_i)] \\
&= \exp(\alpha_t) \sum_{i=1}^{m} D_t(i) \mathbf{1}[y_i \neq f_t(x_i)] + \exp(-\alpha_t) \sum_{i=1}^{m} D_t(i) \mathbf{1}[y_i = f_t(x_i)] \\
&= \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \cdot \epsilon_t + \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} (1 - \epsilon_t) \\
&= 2\sqrt{\epsilon_t (1 - \epsilon_t)} \qquad (5)
\end{aligned}$$

(iv) Using (2) and (5), we get,

$$\begin{aligned}
\text{err}(g) &\leq \prod_{t=1}^{T} Z_t \\
&\leq \prod_{t=1}^{T} 2\sqrt{\epsilon_t (1 - \epsilon_t)} \qquad (6)
\end{aligned}$$

Now, suppose for each $t = 1, 2 \dots T$, $\exists \gamma_t > 0$ such that $\epsilon_t = \frac{1}{2} - \gamma_t$, we have,

$$\begin{aligned}
\prod_{t=1}^{T} 2\sqrt{\epsilon_t (1 - \epsilon_t)} &= \prod_{t=1}^{T} \sqrt{4 \cdot (\frac{1}{2} - \gamma_t)(\frac{1}{2} + \gamma_t)} \\
&= \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \\
&\leq \prod_{t=1}^{T} \exp(-2\gamma_t^2) \qquad (7) \\
&\leq \exp(-2 \sum_{t=1}^{T} \gamma_t^2) \qquad (8)
\end{aligned}$$

where we used $1 + x \leq e^x\ \forall x \in \mathbb{R}$, and hence, $1 - 4\gamma_t^2 \leq \exp(-4\gamma_t^2) \rightarrow \sqrt{1 - 4\gamma_t^2} \leq \exp(-2\gamma_t^2)$ to obtain (7)

Thus, using (6) and (8)

$$\text{err}(g) \leq \exp(-2 \sum_{t=1}^{T} \gamma_t^2)$$

Solution-2:

(i) Let $x_i \in \mathbb{R}^d$ and let $x_i^j \in \mathbb{R}$ denote the j$^{\text{th}}$ component of x_i.

We're given that $x_i^j \sim^{\text{iid}} N(0,1) \;\; \forall j = 1, 2 \dots d$

Because x_i^j are iid, we have, $f(x_i) = \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2} {x_i^j}^2) \rightarrow f(x_i) = \frac{1}{(2\pi)^{d/2}} \exp(-\frac{1}{2} x_i^T x_i)$ is independent of β

Also, $f(y_i|x_i) = \frac{1}{\sqrt{2\pi||x_i||^2}} \exp(-\frac{(y_i - x_i^T\beta)^2}{2||x_i||^2})$

We have,

$$
\begin{aligned}
\arg\max_{\|\beta\| \le 1} Q &= \arg\max_{\|\beta\| \le 1} \frac{1}{n} \sum_{i=1}^{n} \ln f(x_i, y_i) \\
&= \arg\max_{\|\beta\| \le 1} \frac{1}{n} \sum_{i=1}^{n} \ln \big(f(y_i|x_i) f(x_i) \big) \\
&= \arg\max_{\|\beta\| \le 1} \frac{1}{n} \sum_{i=1}^{n} \ln f(y_i|x_i) + \frac{1}{n} \sum_{i=1}^{n} \ln f(x_i) \\
&= \arg\max_{\|\beta\| \le 1} \frac{1}{n} \sum_{i=1}^{n} \ln f(y_i|x_i) \qquad \{\text{because } f(x_i) \text{ is independent of } \beta\} \\
&= \arg\max_{\|\beta\| \le 1} \frac{1}{n} \sum_{i=1}^{n} \ln \big\{ \frac{1}{\sqrt{2\pi||x_i||^2}} \exp(-\frac{(y_i - x_i^T\beta)^2}{2||x_i||^2}) \big\} \\
&= \arg\max_{\|\beta\| \le 1} \frac{1}{n} \sum_{i=1}^{n} -\frac{(y_i - x_i^T\beta)^2}{2||x_i||^2} + \text{constant} \\
&= \arg\max_{\|\beta\| \le 1} \frac{1}{n} \sum_{i=1}^{n} -\frac{(y_i - x_i^T\beta)^2}{2||x_i||^2} \\
&= \arg\min_{\|\beta\| \le 1} \frac{1}{n} \sum_{i=1}^{n} \frac{(y_i - x_i^T\beta)^2}{2\|x_i\|^2} \qquad (1)
\end{aligned}
$$

It is left to show that RHS of (1) is convex optimization problem.

Let

$$
\begin{aligned}
f(\beta) &= \frac{1}{n} \sum_{i=1}^{n} \frac{(y_i - x_i^T\beta)^2}{2\|x_i\|^2} \\
\rightarrow \nabla_\beta f(\beta) &= -\frac{1}{n} \sum_{i=1}^{n} \frac{x_i(y_i - x_i^T\beta)}{\|x_i\|^2} \qquad (2) \\
\rightarrow \nabla^2_{\beta\beta^T} &= \frac{1}{n} \sum_{i=1}^{n} \frac{x_i x_i^T}{\|x_i\|^2} \qquad (3)
\end{aligned}
$$

Let $z \in \mathbb{R}^d$ be arbitrary vector, we have,

$$\begin{aligned} z^T \nabla^2_{\beta\beta^T} z &= \frac{1}{n} \sum_{i=1}^{n} \frac{z^T x_i x_i^T z}{\|x_i\|^2} \\ &= \frac{1}{n} \sum_{i=1}^{n} \frac{(x_i^T z)^T (x_i^T z)}{\|x_i\|^2} \\ &= \frac{1}{n} \sum_{i=1}^{n} \frac{\|x_i^T z\|^2}{\|x_i\|^2} \geq 0 \end{aligned}$$

Hence, the Hessian in (3) is positive semi-definite. Also, the constraint $\|\beta\| \leq 1$ represents a unit $d-$dimensional ball, which is also convex. Hence, RHS of (1) is convex optimization problem.

Thus, the required convex optimization problem that maximizes Q is

$$\arg\min_{\|\beta\| \leq 1} \frac{1}{n} \sum_{i=1}^{n} \frac{(y_i - x_i^T \beta)^2}{2\|x_i\|^2}$$

(ii) Losing the norm constraint, we have from part (i),

$$\arg\max_{\beta} Q = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \frac{(y_i - x_i^T \beta)^2}{2\|x_i\|^2} \tag{4}$$

Losing the norm constraint, the RHS of (4) becomes unconstrained convex optimization problem. To solve this, it is sufficient to take the gradient and set it to zero.

Let $f(\beta) = \frac{1}{n} \sum_{i=1}^{n} \frac{(y_i - x_i^T \beta)^2}{2\|x_i\|^2}$ and consider

$$\begin{aligned} \nabla_\beta f(\beta) &= 0 \\ \rightarrow -\frac{1}{n} \sum_{i=1}^{n} \frac{x_i (y_i - x_i^T \beta)}{\|x_i\|^2} &= 0 \\ \rightarrow \Big(\sum_{i=1}^{n} x_i x_i^T \Big) \beta &= \sum_{i=1}^{n} x_i y_i \\ \rightarrow A\beta &= b \end{aligned} \tag{5}$$

where $A = \sum_{i=1}^{n} x_i x_i^T \in M_{d\times d}(\mathbb{R})$ and $b = \sum_{i=1}^{n} x_i y_i \in M_{d\times 1}(\mathbb{R})$ and (5) represents a system of d linear equations

Solution-3:

(i) Let us say we begin with initial positive example x_0. x_0 will have some components set to 1 and others set to 0.

WLOG, we can assume $x_0 = (x_0^1, x_0^2, \dots x_0^i, x_0^{i+1}, \dots x_0^d)$ where $x_0^j = 1 \quad \forall\ 1 \leq j \leq i$ and $x_0^j = 0 \quad \forall\ (i+1) \leq j \leq d$ for some $i \in \{0, 1, \dots d\}$, where $i = 0$ means all components are 0 and $i = d$ means all components are 1.

Because x_0 is a positive conjunction, we know that neither $\neg x_0^j \quad \forall\ 1 \leq j \leq i$ nor $x_0^j \quad \forall\ (i+1) \leq j \leq d$ are present in f^*.

Now, we can make d queries to establish whether or not $x_0^j \quad \forall\ 1 \leq j \leq i$ and $\neg x_0^j \quad \forall\ (i+1) \leq j \leq d$ are present in f^* as follows:

For $1 \leq j \leq i$, query $y(j) = (y^1, y^2 \dots y^d)$ where $y(j)$ is a function of j and is same as x_0 in all components except the jth component, where we instead set $y^j = 0$ (in x_0, it was 1). It is easy to see that if $y(j)$ evaluates to negative $\rightarrow$ x_j is present in $f*$ and if $y(j)$ evaluates to positive $\rightarrow$ neither x_j nor $\neg x_j$ is present in f^*.

Hence, with these i queries, we are able to establish for sure if x_j or $\neg x_j$ or neither is present in $f^* \quad \forall 1 \leq j \leq i$

For $(i+1) \leq j \leq d$, query $y(j) = (y^1, y^2 \dots y^d)$ where $y(j)$ is a function of j and is same as x_0 in all components except the jth component, where we instead set $y^j = 1$ (in x_0, it was 0). It is easy to see that if $y(j)$ evaluates to negative $\rightarrow$ $\neg x_j$ is present in $f*$ and if $y(j)$ evaluates to positive $\rightarrow$ neither x_j nor $\neg x_j$ is present in f^*.

Hence, with these $(d-i)$ queries, we are able to establish for sure if x_j or $\neg x_j$ or neither is present in $f^* \quad \forall (i+1) \leq j \leq d$

Hence, with a total of exact d queries, we are able to learn f^*.

(ii) Let $(x_i, y_i)_{i=1}^m \sim D$ be the m drawn samples

Let $\mathcal{G} = \{g \in \mathcal{F} | \text{err}(g) \geq \epsilon\} \subseteq \mathcal{F}$. Now, $\forall g \in \mathcal{G}$, we have, $\text{err}(g) = \mathbb{P}_{(x,y)\sim D}[g(x) \neq y] \geq \epsilon \rightarrow \mathbb{P}[g \text{ satisfies the } m \text{ samples}] \leq (1-\epsilon)^m$

Note that $|\mathcal{F}| = 3^d$ because $\forall j = 1, 2, \dots d$, either x_j or $\neg x_j$ or neither exist in arbitrary $g \in \mathcal{F}$. Hence, $|\mathcal{G}| \leq 3^d$

Therefore, using union bound,

$\mathbb{P}[\cup_{g\in\mathcal{G}}(g \text{ satisfies the } m \text{ samples})] \leq \sum_{g\in\mathcal{G}} \mathbb{P}[g \text{ satisfies the } m \text{ samples}] \leq 3^d(1-\epsilon)^m \leq 3^d e^{-m\epsilon}$

where we used $1 + x \leq e^x \quad \forall x \in \mathbb{R}$ in the last inequality.

Now, suppose $3^d e^{-m\epsilon} \leq \delta$ for sufficiently large $m \rightarrow \mathbb{P}[\cup_{g \in \mathcal{G}}(g \text{ satisfies the } m \text{ samples})] \leq \delta$
Also, $3^d e^{-m\epsilon} \leq \delta \rightarrow e^{-m\epsilon} \leq 3^{-d}\delta$.

Taking $\ln(\cdot)$ on both sides gives $-m\epsilon \leq -d \ln 3 + \ln \delta \rightarrow m \geq \frac{1}{\epsilon}\left(d \ln 3 + \ln \frac{1}{\delta}\right)$

Therefore, $m = \frac{1}{\epsilon}\left(d \ln 3 + \ln \frac{1}{\delta}\right)$ will ensure that $\mathbb{P}[\cup_{g \in \mathcal{G}}(g \text{ satisfies the } m \text{ samples})] \leq \delta$

In other words, $m = \frac{1}{\epsilon}\left(d \ln 3 + \ln \frac{1}{\delta}\right)$ will ensure the probability that there exists some $g \in \mathcal{G}$ which satisfies all m samples is at most δ

Or, $m = \frac{1}{\epsilon}\left(d \ln 3 + \ln \frac{1}{\delta}\right)$ will ensure the probability that there exists no $g \in \mathcal{G}$ that satisfies all all m samples is alteast $(1 - \delta)$

Now, using the definition of $\mathcal{G}$, we conclude that $m = \frac{1}{\epsilon}\left(d \ln 3 + \ln \frac{1}{\delta}\right)$ will ensure the probability that arbitrary $f \in \mathcal{F}$ which satisfies all the m samples has $\text{err}(f) < \epsilon$ is alteast $(1 - \delta)$

(iii)

Solution-4: Profile - `https://www.kaggle.com/vishweshw4rtyagi` (vt2353 on the leaderboard)

Pre-processing: Intuitively, time and date would play a crucial role in influencing trip duration. Hence, day $(1-31)$, month $(1-12)$ and hour $(0-23)$ were extracted from feature_0.
$\text{feature}_2, \text{feature}_8, \text{feature}_9$ were treated as numerical features because their values were in $\mathbb{R}$.
$\text{feature}_1, \text{feature}_3, \text{feature}_4, \text{hour}, \text{month}, \text{day}$ were treated as nominal features because their values were in a finite set of natural numbers. Remaining features were dropped because including them resulted in low validation score. Data was then split into training (80%) and validation set (20%). Numeric features were standardized to their Z-scores using column mean and standard deviation (of the train data) and nominal features were one-hot-encoded, resulting in a total of 610 features.

Regressor: I used a deep neural network with drop-out (using PyTorch framework) as my regressor.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 784]         479,024
              ReLU-2                  [-1, 784]               0
           Dropout-3                  [-1, 784]               0
            Linear-4                  [-1, 624]         489,840
              ReLU-5                  [-1, 624]               0
           Dropout-6                  [-1, 624]               0
            Linear-7                  [-1, 312]         195,000
              ReLU-8                  [-1, 312]               0
           Dropout-9                  [-1, 312]               0
           Linear-10                  [-1, 156]          48,828
             ReLU-11                  [-1, 156]               0
          Dropout-12                  [-1, 156]               0
           Linear-13                   [-1, 78]          12,246
             ReLU-14                   [-1, 78]               0
          Dropout-15                   [-1, 78]               0
           Linear-16                    [-1, 1]              79
================================================================
Total params: 1,225,017
Trainable params: 1,225,017
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.04
Params size (MB): 4.67
Estimated Total Size (MB): 4.72
----------------------------------------------------------------
```

I used drop-out layers (with drop probability 0.2) to prevent overfitting and ReLU layers so that the neural net can learn non-linear functions.

Hyper-parameter config - Batch size: 764, Learning rate: 0.001, EPOCHS: 100

Model with the lowest validation MAE was used to make predictions, which gave public MAE of 316.49334 on the leaderboard. Later, an ensemble of such deep nets with some layers changed was used to obtain public MAE of xxx on the leaderboard.

On the same processed data, other models such as LinearRegression (least-squares), XGBoostRegressor, LightGBMRegressor were also used, but none gave public MAE < 350 even after hyper-parameter tuning.

No external sources apart from python libraries were used.