COMS 4771 Machine Learning (Spring 2018)
Problem Set #3

Jianfu Yang - jy2863 Wenda Xu - wx2195 Fan Yang - fy2232
- wx2195@columbia.edu

March 27, 2018

Problem 1

(a) $\boldsymbol{X}$ could be written as:

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{d1} & x_{d2} & \cdots & x_{dn} \end{bmatrix}$$

$\boldsymbol{y}$ could be written as:

$$\begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix}$$

Now transform $\boldsymbol{X}$ to $\boldsymbol{X'}$:

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} & \sqrt{\lambda\alpha} & 0 & \cdots & 0 \\ x_{21} & x_{22} & \cdots & x_{2n} & 0 & \sqrt{\lambda\alpha} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & x_{d2} & \cdots & x_{dn} & 0 & 0 & \cdots & \sqrt{\lambda\alpha} \end{bmatrix}$$

$\boldsymbol{y}$ to $\boldsymbol{y'}$:

$$\begin{bmatrix} y_1 & y_2 & \cdots & y_n & 0 & 0 & \cdots & 0 \end{bmatrix}$$

Then,

$$||\boldsymbol{w}\boldsymbol{X'} - \boldsymbol{y'}||_2^2 = ||\boldsymbol{w}\boldsymbol{X} - \boldsymbol{y}||_2^2 + \lambda\alpha||\boldsymbol{w}||_2^2$$

We have,

$$||\boldsymbol{w}\boldsymbol{X} - \boldsymbol{y}||_2^2 + \lambda[\alpha||\boldsymbol{w}||_2^2 + (1-\alpha)||\boldsymbol{w}||_1] = ||\boldsymbol{w}\boldsymbol{X'} - \boldsymbol{y'}||_2^2 + \lambda(1-\alpha)||\boldsymbol{w}||_1$$

Now, the new equivalent objective function is:

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{w}\boldsymbol{X'} - \boldsymbol{y'}||_2^2 + \lambda(1-\alpha)||\boldsymbol{w}||_1$$

It's a lasso regression with $\lambda' = \lambda(1-\alpha)$.

(b) $\boldsymbol{w}$ is a row vector, while $\boldsymbol{x}_i$ is a column vector.

$$y_i = \boldsymbol{w}\boldsymbol{x}_i + \epsilon_i \sim \mathcal{N}(\boldsymbol{w}\boldsymbol{x}_i, \sigma^2)$$
$$w_j \sim \mathcal{N}(0, \tau^2)$$

$P(w_1, \cdots, w_d|(x_1, y_1), \cdots, (x_n, y_n))$ should consist of the likelihood and the prior. That is,

$$\begin{aligned}
P &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y_i - \boldsymbol{w}\boldsymbol{x}_i)^2}{2\sigma^2}) \cdot \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi\tau^2}} \exp(-\frac{w_j^2}{2\tau^2}) \\
\ln P &= -\sum_{i=1}^{n} \frac{(y_i - \boldsymbol{w}\boldsymbol{x}_i)^2}{2\sigma^2} - \sum_{j=1}^{d} \frac{w_j^2}{2\tau^2} + constant \\
&= -\frac{1}{2\sigma^2}||\boldsymbol{w}\boldsymbol{X} - \boldsymbol{y}||_2^2 - \frac{1}{2\tau^2}||\boldsymbol{w}||_2^2 + constant \\
\arg\max_{\boldsymbol{w}} P &= \arg\max_{\boldsymbol{w}} \ln P = \arg\min_{\boldsymbol{w}}(-\ln P) \\
&= \arg\min_{\boldsymbol{w}} \frac{1}{2\sigma^2}||\boldsymbol{w}\boldsymbol{X} - \boldsymbol{y}||_2^2 + \frac{1}{2\tau^2}||\boldsymbol{w}||_2^2 \\
&= \arg\min_{\boldsymbol{w}} ||\boldsymbol{w}\boldsymbol{X} - \boldsymbol{y}||_2^2 + \frac{\sigma^2}{\tau^2}||\boldsymbol{w}||_2^2 \\
&= \arg\min_{\boldsymbol{w}} ||\boldsymbol{w}\boldsymbol{X} - \boldsymbol{y}||_2^2 + \lambda||\boldsymbol{w}||_2^2
\end{aligned}$$

Thus, maximizing P is s equivalent to minimizing the ridge optimization criterion. Proven.

Problem 2

(i)

$$\begin{aligned}
D_{T+1}(i) &= \frac{D_{T+1}(i)}{\sum_i D_{T+1}(i)} \\
&= \frac{1}{Z_T} D_T(i) e^{-\alpha_T y_i f_T(x_i)} \\
&= \frac{1}{Z_T} e^{-\alpha_T y_i f_T(x_i)} \frac{1}{Z_{T-1}} e^{-\alpha_{T-1} y_i f_{T-1}(x_i)} D_{T-1}(i) \\
&= \frac{1}{Z_T} e^{-\alpha_T y_i f_T(x_i)} \frac{1}{Z_{T-1}} e^{-\alpha_{T-1} y_i f_{T-1}(x_i)} \cdots \frac{1}{Z_1} e^{-\alpha_1 y_i f_1(x_i)} D_1(i) \\
&= \frac{1}{\prod_t Z_t} e^{-y_i \sum_t \alpha_t f_t(x_i)} D_1(i) \\
&= \frac{1}{m} \frac{1}{\prod_t Z_t} e^{-y_i g(x_i)}
\end{aligned}$$

Proven.

(ii) Firstly, since $D_{T+1}(i)$ is after normalizing,

$$\sum_i D_{T+1}(i) = 1$$

$$\sum_i \frac{1}{m} \frac{1}{\prod_t Z_t} e^{-y_i g(x_i)} = 1$$

$$\sum_i \frac{1}{m} e^{-y_i g(x_i)} = \prod_t Z_t$$

Then, using the fact that 0-1 loss is upper bounded by exponential loss:

$$\begin{aligned}
err(g) &= \frac{1}{m} \sum_i \mathbf{1}[y_i \neq sign(g(x_i))] \\
&\leqslant \frac{1}{m} \sum_i e^{-y_i g(x_i)} \\
&= \prod_t Z_t \\
err(g) &\leqslant \prod_t Z_t
\end{aligned}$$

Proven.

(iii)

$$\begin{aligned}
Z_t &= \sum_i D_t(i)e^{-\alpha_t y_i f_t(x_i)} \\
&= \sum_i D_t(i)\mathbf{1}[y_i = f_t(x_i)]e^{-\alpha_t} + \sum_i D_t(i)\mathbf{1}[y_i \neq f_t(x_i)]e^{\alpha_t} \qquad (\sum_i D_t(i) = 1) \\
&= (1 - \sum_i D_t(i)\mathbf{1}[y_i \neq f_t(x_i)])e^{-\alpha_t} + \sum_i D_t(i)\mathbf{1}[y_i \neq f_t(x_i)]e^{\alpha_t} \\
&= (1-\epsilon_t)e^{-\frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}} + \epsilon_t e^{\frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}} \\
&= (1-\epsilon)\sqrt{\frac{\epsilon_t}{1-\epsilon_t}} + \epsilon_t\sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \\
&= 2\sqrt{\epsilon_t(1-\epsilon_t)}
\end{aligned}$$

Proven.

(iv)

$$\begin{aligned}
err(g) &\leqslant \prod_t Z_t = \prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} \qquad (\epsilon_t = \frac{1}{2} - \gamma_t) \\
&= \prod_t 2\sqrt{(\frac{1}{2}-\gamma_t)(\frac{1}{2}+\gamma_t)} = \prod_t 2\sqrt{\frac{1}{4}-\gamma_t^2} \\
&= \prod_t \sqrt{1-4\gamma_t^2} \qquad (1+x \leqslant e^x \Rightarrow \sqrt{1+x} \leqslant e^{\frac{1}{2}x},\ for\ any\ x \in \mathbb{R}) \\
&\leqslant \prod_t e^{-2\gamma_t^2} = e^{-2\sum_t \gamma_t^2}
\end{aligned}$$

Proven.

- wx2195@columbia.edu

Problem 3

(i) Let $\boldsymbol{b}'$ denotes the vector mapped from $\boldsymbol{x}_i$. Then:

$$b'_k = \sum_{j=1}^{n} A_{kj}x_{ij} \mod 2 \qquad (1 \leqslant k \leqslant p)$$

Assume J contains the indexes of elements in x_i which are 1. Since x_i is none-zero, $0 < |J| \leqslant n$. Then,

$$\sum_{j=1}^{n} A_{kj}x_{ij} = \sum_{j \in J} A_{kj}$$

Denote X as $\sum_{j \in J} A_{kj}$. Since A_{kj} obeys the Bernoulli distribution with $p = 0.5$, we have $X \sim B(J, \frac{1}{2})$. Then,

$$P(X) = C_J^X (\frac{1}{2})^X (\frac{1}{2})^{J-X} = C_J^X (\frac{1}{2})^J$$

$b'_k = 0$ means X is even, $b'_k = 1$ means X is odd, that is,

$$P(b'_k = 0) = (\frac{1}{2})^J \sum_{X \ is \ even; X \leqslant |J|} C_J^X$$

$$P(b'_k = 1) = (\frac{1}{2})^J \sum_{X \ is \ odd; X \leqslant |J|} C_J^X$$

For binomial coefficient, we have

$$(1+x)^J = C_J^0 + C_J^1 x + \cdots + C_J^J x^J$$

$$(1+1)^J = C_J^0 + C_J^1 + \cdots + C_J^J = 2^J$$

$$(1-1)^J = C_J^0 - C_J^1 + C_J^2 - C_J^3 + \cdots = 0$$

Hence,

$$\sum_{X \ is \ even; X \leqslant |J|} C_J^X = \sum_{X \ is \ odd; X \leqslant |J|} C_J^X = 2^{J-1}$$

Now, we have,

$$P(b'_k = 0) = P(b'_k = 1) = \frac{1}{2}$$

Since $\boldsymbol{b}$ has p elements, the probability of $\boldsymbol{b} = \boldsymbol{b}'$ is:

$$P = (\frac{1}{2})^p$$

Proven.

- wx2195@columbia.edu

(ii) Denotes $\boldsymbol{b}$ as the vector mapped from $\boldsymbol{x}_i$, and $\boldsymbol{b}'$ as the vector mapped from $\boldsymbol{x}_j$. Then the probability of collision is the probability of $\boldsymbol{b} = \boldsymbol{b}'$:

$$P(collision) = \frac{1}{2^p}$$

(iii)

$$\begin{aligned}
P(no\ collisions) &= \prod_{1 \leqslant i < j \leqslant n} P(no\ collision\ between\ \boldsymbol{x}_i\ and\ \boldsymbol{x}_j) \\
&= (1 - \frac{1}{2^p})^{C_m^2} \\
&= (1 - \frac{1}{2^p})^{\frac{m^2-m}{2}} \qquad (p = 2\log_2 m) \\
&= (1 - \frac{1}{m^2})^{\frac{m^2-m}{2}} \qquad (m \geqslant 1) \\
&= f(m) \\
\frac{d\ f(m)}{dm} &= \frac{(1-\frac{1}{m^2})^{\frac{m^2-m}{2}}((2m^2+m-1)\log(1-\frac{1}{m^2})+2)}{2(m+1)} \\
&< 0 \qquad (for\ m \geqslant 1)
\end{aligned}$$

Hence, the minimum of $f(m)$ will be:

$$\min f(m) = \lim_{m\to\infty} f(m) = \lim_{m\to\infty} (1 - \frac{1}{m^2})^{\frac{m^2-m}{2}} = \frac{1}{\sqrt{e}} \approx 0.6065$$

Thus, there will be no collision among $\boldsymbol{x}_i$ with the probability of at least $\frac{1}{2}$. Proven.

Problem 4

We firstly using linear mode (`sklearn.linear_mode.LinearRegression`)to fit the data directly, and get an MAE around 102. Then we scale the data to have zero mean (`sklearn.preprocessing.StandardScaler`), and apply linear mode again, then get an MAE of 6.82.
To improve, we apply `sklearn.preprocessing.PolynomialFeatures` to the data, and again use linear mode (i.e., applying quadratic mode to the original data). This method decrease the MAE to 6.5.
We apply `sklearn.linear_model.Ridge`, no significant improvement observed.
Then we try to use `sklearn.svm.SVR` and `sklearn.kernel_ridge.KernelRidge` to fit the data. The former is too slow to get the result, while the latter uses too much memory.
We use `sklearn.linear_model.SGDRegressor`, setting the loss function as epsilon_insensitive, and get an MAE of 6.2.
We also try to use K-D tree and `sklearn.multioutput` to implement this regression problem as a classification problem, it doesn't make improvement.
Finally, we try to use MLP (`sklearn.neural_network.MLPRegressor`) to do the regression. We set the hidden layer size as [25,5], using 'sgd' as solver, 'adaptive' as step_size, and finally get an MAE of around 6.00.