Statistical Machine Learning
Spring 2020, Homework 2
(due on Feb 19, 11.59pm EST)

Jean Honorio jhonorio@purdue.edu

The homework is based on a total of 10 points. Please read submission instructions at the end. **Failure to comply to submission instructions will cause your grade to be reduced.**

For questions 1, 2 and 3, you can use the function **createsepratingdata.m** to create some synthetic separable rating (ordinal regression) data:

```
% Input: number of samples n
%        number of features d
%        number of labels k
% Output: matrix X of features, with n rows (samples), d columns (features)
%             X(i,j) is the j-th feature of the i-th sample
%         vector y of labels, with n rows (samples), 1 column
%             y(i) is the label (1 or 2 ... or k) of the i-th sample
% Example on how to call the function: [X y] = createsepratingdata(10,2,3);
function [X y] = createsepratingdata(n,d,k)

if n < k
    error('n should be at least k');
end
X = rand(n,d);
y = zeros(n,1);
i = 0;
for l = 1:k
    j = ceil(l/k*n);
    X(i+1:j,1) = X(i+1:j,1) + 1.5*l;
    y(i+1:j) = l;
    i = j;
end
U = orth(rand(d));
X = X*U;
```

For questions 4, 5 and 6, you can use the function **createlinregdata.m** to create some synthetic linear regression data:

```
% Input: number of samples n
%        number of features d
% Output: matrix X of features, with n rows (samples), d columns (features)
%             X(i,j) is the j-th feature of the i-th sample
%         vector y of scalar values, with n rows (samples), 1 column
%             y(i) is the scalar value of the i-th sample
% Example on how to call the function: [X y] = createlinregdata(10,2);
function [X y] = createlinregdata(n,d)

w = 2*rand(d,1)-1;
w = w/norm(w);
X = randn(n,d);
y = X*w + 0.25*randn(n,1);
```

Additionally, questions 4, 5 and 6 require a way to solve the linear regression problem, with training data $x_t \in \mathbb{R}^d$, $y_t \in \mathbb{R}$ for $t = 1, \ldots, n$.

$$\widehat{\theta} \leftarrow \arg\min_{\beta \in \mathbb{R}^d} \frac{1}{2} \sum_{t=1}^{n} (y_t - \beta \cdot x_t)^2$$

If you assume that $n > d$, a solution to the above problem is given by the following function **linreg.m**:

```
% Input: matrix X of features, with n rows (samples), d columns (features)
%             X(i,j) is the j-th feature of the i-th sample
%         vector y of scalar values, with n rows (samples), 1 column
%             y(i) is the scalar value of the i-th sample
% Output: vector theta, with d rows, 1 column
function theta = linreg(X,y)

theta = pinv(X)*y;
```

Here are the questions:

1) [2 points] Implement the PRank algorithm for rating (ordinal regression), introduced in Lecture 6. Recall that:

$$s_{tl} = \begin{cases} -1 & \text{if } y_t \leq l \\ +1 & \text{if } y_t > l \end{cases}$$

The algorithm to be implemented is as follows.

Input: number of iterations L, number of labels k, training data $x_t \in \mathbb{R}^d$, $y_t \in \{1, \ldots, k\}$ for $t = 1, \ldots, n$

Output: $\theta \in \mathbb{R}^d$, $b \in \mathbb{R}^{k-1}$
$\theta \leftarrow 0$
for $l = 1, \ldots, k-1$ **do**
  $b_l \leftarrow l$
end for
for $\text{iter} = 1, \ldots, L$ **do**
  for $t = 1, \ldots, n$ **do**
    $E \leftarrow \{l \mid s_{tl}(\theta \cdot x_t - b_l) \leq 0\}$
    if E is not an empty set **then**
      $\theta \leftarrow \theta + \left(\sum_{l \in E} s_{tl}\right) x_t$
      for $l \in E$ **do**
        $b_l \leftarrow b_l - s_{tl}$
      end for
    end if
  end for
end for

The header of your **MATLAB function ratingprank.m** should be:

```
% Input: number of iterations L
%        number of labels k
%        matrix X of features, with n rows (samples), d columns (features)
%            X(i,j) is the j-th feature of the i-th sample
%        vector y of labels, with n rows (samples), 1 column
%            y(i) is the label (1 or 2 ... or k) of the i-th sample
% Output: vector theta of d rows, 1 column
%         vector b of k-1 rows, 1 column
function [theta b] = ratingprank(L,k,X,y)
```

2) [1 point] Implement the following rating (ordinal regression) predictor function, introduced in Lecture 6. Here we present k conditionals (if) for clarity. Your implementation should work for any number k.

Input: number of labels k, $\theta \in \mathbb{R}^d$, $b \in \mathbb{R}^{k-1}$, testing point $x \in \mathbb{R}^d$
Output: label $\in \{1, \ldots, k\}$
if $\theta \cdot x \leq b_1$ **then** label $\leftarrow 1$
if $b_1 < \theta \cdot x \leq b_2$ **then** label $\leftarrow 2$
if $b_2 < \theta \cdot x \leq b_3$ **then** label $\leftarrow 3$
$\vdots$
if $b_{k-2} < \theta \cdot x \leq b_{k-1}$ **then** label $\leftarrow k-1$
if $b_{k-1} < \theta \cdot x$ **then** label $\leftarrow k$

The header of your **MATLAB function ratingpred.m** should be:

```
% Input: number of labels k
%        vector theta of d rows, 1 column
```

```
%         vector b of k-1 rows, 1 column
%         vector x of d rows, 1 column
% Output: label (1 or 2 ... or k)
function label = ratingpred(k,theta,b,x)
```

3) [1.5 points] Now we ask you to implement the following support vector machines for rating (ordinal regression), introduced in Lecture 6. Recall that:

$$s_{il} = \begin{cases} -1 & \text{if } y_i \leq l \\ +1 & \text{if } y_i > l \end{cases}$$

The problem to be implemented is as follows.

$$\begin{aligned} &\text{minimize } \frac{1}{2}\theta \cdot \theta \\ &\text{subject to } s_{il}(x_i \cdot \theta - b_l) \geq 1 \text{ for } i = 1, \ldots, n, l = 1, \ldots, k-1 \\ &\qquad b_l \leq b_{l+1} \text{ for } l = 1, \ldots, k-2 \end{aligned}$$

First some general notation for clarity: for any integers p and q, let $I_{p\times p} \in \mathbb{R}^{p\times p}$ be the identity matrix with p rows and p columns. Let $0_{p\times q} \in \mathbb{R}^{p\times q}$ be a matrix of zeros, with p rows and q columns. Let $1_{p\times q} \in \mathbb{R}^{p\times q}$ be a matrix of ones, with p rows and q columns.

Now, recall that we have n training samples and that $x_i \in \mathbb{R}^d$ for $i = 1, \ldots, n$. Let $H = \begin{bmatrix} I_{d\times d} & 0_{d\times(k-1)} \\ 0_{(k-1)\times d} & 0_{(k-1)\times(k-1)} \end{bmatrix} \in \mathbb{R}^{(d+k-1)\times(d+k-1)}$. Let $f = 0_{(d+k-1)\times 1}$. Let $A \in \mathbb{R}^{(n(k-1)+k-2)\times(d+k-1)}$ defined as follows. (Recall that $x_{i,j}$ is the j-th feature of the i-th sample.)

$$A = \begin{bmatrix}
-s_{1,1}x_{1,1} & -s_{1,1}x_{1,2} & \ldots & -s_{1,1}x_{1,d} & s_{1,1} & 0 & 0 & \ldots & 0 & 0 \\
-s_{1,2}x_{1,1} & -s_{1,2}x_{1,2} & \ldots & -s_{1,2}x_{1,d} & 0 & s_{1,2} & 0 & \ldots & 0 & 0 \\
\vdots & & & & & & & & & \\
-s_{1,k-1}x_{1,1} & -s_{1,k-1}x_{1,2} & \ldots & -s_{1,k-1}x_{1,d} & 0 & 0 & 0 & \ldots & 0 & s_{1,k-1} \\
\vdots & & & & & & & & & \\
\vdots & & & & & & & & & \\
\vdots & & & & & & & & & \\
-s_{n,1}x_{n,1} & -s_{n,1}x_{n,2} & \ldots & -s_{n,1}x_{n,d} & s_{n,1} & 0 & 0 & \ldots & 0 & 0 \\
-s_{n,2}x_{n,1} & -s_{n,2}x_{n,2} & \ldots & -s_{n,2}x_{n,d} & 0 & s_{n,2} & 0 & \ldots & 0 & 0 \\
\vdots & & & & & & & & & \\
-s_{n,k-1}x_{n,1} & -s_{n,k-1}x_{n,2} & \ldots & -s_{n,k-1}x_{n,d} & 0 & 0 & 0 & \ldots & 0 & s_{n,k-1} \\
0 & 0 & \ldots & 0 & 1 & -1 & 0 & \ldots & 0 & 0 \\
0 & 0 & \ldots & 0 & 0 & 1 & -1 & \ldots & 0 & 0 \\
\vdots & & & & & & & & & \\
0 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 1 & -1
\end{bmatrix}$$

Let $c = \begin{bmatrix} -1_{(n(k-1))\times 1} \\ 0_{(k-2)\times 1} \end{bmatrix} \in \mathbb{R}^{n(k-1)+k-2}$. Since $\theta \in \mathbb{R}^d$ and $b \in \mathbb{R}^{k-1}$, lets define $z = \begin{bmatrix} \theta \\ b \end{bmatrix} \in \mathbb{R}^{d+k-1}$ and we can rewrite the rating SVM problem as:

$$\begin{aligned} \text{minimize } & \frac{1}{2} z^{\mathrm{T}} H z + f^{\mathrm{T}} z \\ \text{subject to } & Az \leq c \end{aligned}$$

Fortunately, the standard MATLAB function **quadprog.m** can solve exactly the above problem by doing: **z = quadprog(H,f,A,c);** The header of your **MATLAB function ratingsvm.m** should be:

```
% Input: number of labels k
%        matrix X of features, with n rows (samples), d columns (features)
%            X(i,j) is the j-th feature of the i-th sample
%        vector y of labels, with n rows (samples), 1 column
%            y(i) is the label (1 or 2 ... or k) of the i-th sample
% Output: vector theta of d rows, 1 column
%         vector b of k-1 rows, 1 column
function [theta b] = ratingsvm(k,X,y)
```

Notice that for prediction you can reuse the **ratingpred.m** function that you wrote for question 2.

4) [1.5 points] Let S be a set of features. We define $x_{t,S}$ as a vector corresponding to taking the value of features in S of the t-th sample. (For instance, if $S = \{1, 4, 6\}$ and $t = 20$, then $x_{t,S}$ is a 3-dimensional vector containing the value of the features 1, 4 and 6, of sample 20.) Implement the following *greedy subset selection* algorithm, introduced in Lecture 7.

Input: number of features F, training data $x_t \in \mathbb{R}^d$, $y_t \in \mathbb{R}$ for $t = 1, \ldots, n$
Output: feature set S, $\theta_S \in \mathbb{R}^F$
$S \leftarrow$ empty set
for $f = 1, \ldots, F$ **do**
  for each $j \notin S$ **do**
    $\widehat{\theta}_{S\cup j} \leftarrow \arg\min_{\beta \in \mathbb{R}^f} \frac{1}{2} \sum_{t=1}^{n} (y_t - \beta \cdot x_{t,S\cup j})^2$
    $J(S \cup j) \leftarrow \frac{1}{2} \sum_{t=1}^{n} (y_t - \widehat{\theta}_{S\cup j} \cdot x_{t,S\cup j})^2$
  end for
  $\widehat{j} \leftarrow \arg\min_{j \notin S} J(S \cup j)$
  $S \leftarrow S \cup \{\widehat{j}\}$
end for
$\theta_S \leftarrow \widehat{\theta}_S$

The header of your **MATLAB function greedysubset.m** should be:

```
% Input: number of features F
%        matrix X of features, with n rows (samples), d columns (features)
%            X(i,j) is the j-th feature of the i-th sample
%        vector y of scalar values, with n rows (samples), 1 column
%            y(i) is the scalar value of the i-th sample
% Output: vector of selected features S, with F rows, 1 column
%         vector thetaS, with F rows, 1 column
%             thetaS(1) corresponds to the weight of feature S(1)
%             thetaS(2) corresponds to the weight of feature S(2)
%             and so on and so forth
function [S thetaS] = greedysubset(F,X,y)
```

You can assume that $n > d > F$.

5) [2 points] Let S be a set of features. We define $x_{t,S}$ as a vector corresponding to taking the value of features in S of the t-th sample. (For instance, if $S = \{1, 4, 6\}$ and $t = 20$, then $x_{t,S}$ is a 3-dimensional vector containing the value of the features 1, 4 and 6, of sample 20.) Implement the following *forward fitting* algorithm, introduced in Lecture 7.

Input: number of features F, training data $x_t \in \mathbb{R}^d$, $y_t \in \mathbb{R}$ for $t = 1, \ldots, n$
Output: feature set S, $\theta_S \in \mathbb{R}^F$
$S \leftarrow$ empty set
$\widehat{\theta}_S \leftarrow$ empty array
for $f = 1, \ldots, F$ **do**
  for each $j \notin S$ **do**

$$\widehat{\theta}_j \leftarrow \arg\min_{\beta \in \mathbb{R}} \frac{1}{2} \sum_{t=1}^{n} (y_t - \widehat{\theta}_S \cdot x_{t,S} - \beta\, x_{t,j})^2$$

$$J(\widehat{\theta}_S, j) \leftarrow \frac{1}{2} \sum_{t=1}^{n} (y_t - \widehat{\theta}_S \cdot x_{t,S} - \widehat{\theta}_j\, x_{t,j})^2$$

  end for
  $\widehat{j} \leftarrow \arg\min_{j \notin S} J(\widehat{\theta}_S, j)$
  $\widehat{\theta}_{S \cup j} \leftarrow (\widehat{\theta}_S, \widehat{\theta}_{\widehat{j}})$
  $S \leftarrow S \cup \{\widehat{j}\}$
end for
$\theta_S \leftarrow \widehat{\theta}_S$

The header of your **MATLAB function forwardfitting.m** should be:

```
% Input: number of features F
%        matrix X of features, with n rows (samples), d columns (features)
%            X(i,j) is the j-th feature of the i-th sample
%        vector y of scalar values, with n rows (samples), 1 column
%            y(i) is the scalar value of the i-th sample
```

```
% Output: vector of selected features S, with F rows, 1 column
%         vector thetaS, with F rows, 1 column
%             thetaS(1) corresponds to the weight of feature S(1)
%             thetaS(2) corresponds to the weight of feature S(2)
%             and so on and so forth
function [S thetaS] = forwardfitting(F,X,y)
```

You can assume that $n > d > F$.

6) [2 points] Let S be a set of features. We define $x_{t,S}$ as a vector corresponding to taking the value of features in S of the t-th sample. (For instance, if $S = \{1, 4, 6\}$ and $t = 20$, then $x_{t,S}$ is a 3-dimensional vector containing the value of the features 1, 4 and 6, of sample 20.) Implement the following *myopic forward fitting* algorithm, introduced in Lecture 7.

Input: number of features F, training data $x_t \in \mathbb{R}^d$, $y_t \in \mathbb{R}$ for $t = 1, \ldots, n$
Output: feature set S, $\theta_S \in \mathbb{R}^F$
$S \leftarrow$ empty set
$\widehat{\theta}_S \leftarrow$ empty array
for $f = 1, \ldots, F$ **do**
  for each $j \notin S$ **do**
    $DJ(\widehat{\theta}_S, j) \leftarrow -\sum_{t=1}^{n} (y_t - \widehat{\theta}_S \cdot x_{t,S})\, x_{t,j}$
  end for
  $\widehat{j} \leftarrow \arg\max_{j \notin S} |DJ(\widehat{\theta}_S, j)|$
  $\widehat{\theta}_{\widehat{j}} \leftarrow \arg\min_{\beta \in \mathbb{R}} \frac{1}{2} \sum_{t=1}^{n} (y_t - \widehat{\theta}_S \cdot x_{t,S} - \beta\, x_{t,\widehat{j}})^2$
  $\widehat{\theta}_{S \cup j} \leftarrow (\widehat{\theta}_S, \widehat{\theta}_{\widehat{j}})$
  $S \leftarrow S \cup \{\widehat{j}\}$
end for
$\theta_S \leftarrow \widehat{\theta}_S$

The header of your **MATLAB function myopicfitting.m** should be:

```
% Input: number of features F
%        matrix X of features, with n rows (samples), d columns (features)
%            X(i,j) is the j-th feature of the i-th sample
%        vector y of scalar values, with n rows (samples), 1 column
%            y(i) is the scalar value of the i-th sample
% Output: vector of selected features S, with F rows, 1 column
%         vector thetaS, with F rows, 1 column
%             thetaS(1) corresponds to the weight of feature S(1)
%             thetaS(2) corresponds to the weight of feature S(2)
%             and so on and so forth
function [S thetaS] = myopicfitting(F,X,y)
```

You can assume that $n > d > F$.

Submission: Please, submit a single ZIP file **through Blackboard**. Your MATLAB code (**ratingprank.m**, **ratingpred.m**, etc.) should be directly inside the ZIP file. **There should not be any folder inside the ZIP file**, just MATLAB code. The ZIP file should be named by the first letter of your first name followed by your last name. For instance, for Jean Honorio, the ZIP file should be named **jhonorio.zip**