Statistical Machine Learning
Spring 2020, Homework 3
(due on Mar 3, 11.59pm EST)

Jean Honorio jhonorio@purdue.edu

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

For questions 1 and 2, you can use the following function **createsepdata.m** to create some synthetic separable 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 labels, with n rows (samples), 1 column
%             y(i) is the label (+1 or -1) of the i-th sample
% Example on how to call the function: [X y] = createsepdata(10,3);
function [X y] = createsepdata(n,d)

y = ones(n,1);
y(ceil(n/2)+1:end) = -1;
X = rand(n,d);
X(y==1,1) = 0.1+X(y==1,1);
X(y==-1,1) = -0.1-X(y==-1,1);
U = orth(rand(d));
X = X*U;
```

For questions 3 and 4, 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, for questions 3 and 4, use the following way to solve the linear regression problem, with training data $x_t \in \mathbb{R}^d$, $y_t \in \mathbb{R}$ for $t = 1, \ldots, n$.

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

If $n > d$, a solution to the above 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) [3.5 points] Implement a simplified version of the boosting algorithm (Lecture 8). We will use the exponential loss, and a simple type of weak classifiers, which take the sign of one feature. Recall that y_t is the label of the t-th sample and $x_{t,j}$ is the j-th feature of the t-th sample. Note that $\mathrm{sgn}(z) = 1$ if $z > 0$, and $\mathrm{sgn}(z) = -1$ if $z \leq 0$. Here is the algorithm:

Input: number of iterations L, training data $x_t \in \mathbb{R}^d$, $y_t \in \{+1, -1\}$ for $t = 1, \ldots, n$
Output: $\alpha \in \mathbb{R}^L$, $\theta \in \{1, \ldots, d\}^L$
$\alpha \leftarrow (0, 0, \ldots, 0)$
$\theta \leftarrow (1, 1, \ldots, 1)$
for $t = 1, \ldots, n$ **do**
  $W_t \leftarrow 1/n$
end for
for $r = 1, \ldots, L$ **do**
  $\theta_r \leftarrow \arg\min_{j \in \{1,\ldots,d\}} \; -\sum_{t=1}^{n} W_t \, y_t \, \mathrm{sgn}(x_{t,j})$

$\epsilon \leftarrow \min_{j \in \{1,\ldots,d\}} -\sum_{t=1}^{n} W_t\, y_t\, \mathrm{sgn}(x_{t,j})$
if $\epsilon \geq 0$ **then**
stop
end if
$\epsilon \leftarrow \max(\epsilon, -0.99)$
$\alpha_r \leftarrow \frac{1}{2} \log\left(\frac{1-\epsilon}{1+\epsilon}\right)$
for $t = 1, \ldots, n$ **do**
$W_t \leftarrow W_t \exp(-\alpha_r\, y_t\, \mathrm{sgn}(x_{t,\theta_r}))$
end for
$Z = \sum_{t=1}^{n} W_t$
for $t = 1, \ldots, n$ **do**
$W_t \leftarrow W_t / Z$
end for
end for

Since θ_r is a feature index, x_{t,θ_r} denotes taking the feature θ_r from the t-th sample. The header of your **MATLAB function adaboost.m** should be:

```
% Input: number of iterations L
%        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 -1) of the i-th sample
% Output: vector alpha of weights, with L rows, 1 column
%         vector theta of feature indices, with L rows, 1 column
function [alpha theta] = adaboost(L,X,y)
```

2) [1.5 points] Implement the Adaboost predictor function. Note that $\mathrm{sgn}(z) = 1$ if $z > 0$, and $\mathrm{sgn}(z) = -1$ if $z \leq 0$.

Input: $\alpha \in \mathbb{R}^L$, $\theta \in \{1, \ldots, d\}^L$, testing point $x \in \mathbb{R}^d$
Output: label $\in \{+1, -1\}$

$$\text{label} \leftarrow \mathrm{sgn}\left(\sum_{r=1}^{L} \alpha_r\, \mathrm{sgn}(x_{\theta_r})\right)$$

Since θ_r is a feature index, x_{θ_r} denotes taking the feature θ_r from the testing point x. The header of your **MATLAB function adapred.m** should be:

```
% Input: vector alpha of weights, with L rows, 1 column
%        vector theta of feature indices, with L rows, 1 column
%        vector x of d rows, 1 column
% Output: label (+1 or -1)
function label = adapred(alpha,theta,x)
```

3) [2.5 points] Implement k-fold cross validation (Lecture 9) with linear regression. (The function $\lfloor w \rfloor$ denotes the largest integer less than or equal to $w \in \mathbb{R}$,

i.e., the "floor" function.)

Input: number of folds k, data $x_t \in \mathbb{R}^d$, $y_t \in \mathbb{R}$ for $t = 1, \ldots, n$
Output: mean square error $z \in \mathbb{R}^k$
for $i = 1, \ldots, k$ **do**
  $T \leftarrow \{\lfloor n(i-1)/k \rfloor + 1, \ldots, \lfloor n\, i/k \rfloor\}$
  $S \leftarrow \{1, \ldots, n\} - T$
  $\widehat{\theta} \leftarrow \arg\min_{\beta \in \mathbb{R}^d} \frac{1}{|S|} \sum_{t \in S} (y_t - \beta \cdot x_t)^2$
  $z_i \leftarrow \frac{1}{|T|} \sum_{t \in T} (y_t - \widehat{\theta} \cdot x_t)^2$
end for

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

```
% Input: number of folds k
%        matrix X of features, with n rows (samples), d columns (features)
%        vector y of scalar values, with n rows (samples), 1 column
% Output: vector z of k rows, 1 column
function z = kfoldcv(k,X,y)
```

4) [2.5 points] Implement bootstrapping (Lecture 9) with linear regression.

Input: number of bootstraps B, data $x_t \in \mathbb{R}^d$, $y_t \in \mathbb{R}$ for $t = 1, \ldots, n$
Output: mean square error $z \in \mathbb{R}^B$
for $i = 1, \ldots, B$ **do**
  $u \leftarrow (0, \ldots, 0)$ (an array of n zeros)
  $S \leftarrow$ emptyset
  for $j = 1, \ldots, n$ **do**
    choose k uniformly at random from $\{1, \ldots, n\}$
    $u_j \leftarrow k$ (repeated elements are allowed in the array u)
    $S \leftarrow S \cup \{k\}$ (repeated elements are not allowed in the set S)
  end for
  $T \leftarrow \{1, \ldots, n\} - S$ (repeated elements are not allowed in the set T)
  $\widehat{\theta} \leftarrow \arg\min_{\beta \in \mathbb{R}^d} \frac{1}{n} \sum_{j=1}^{n} (y_{u_j} - \beta \cdot x_{u_j})^2$
  $z_i \leftarrow \frac{1}{|T|} \sum_{t \in T} (y_t - \widehat{\theta} \cdot x_t)^2$
end for

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

```
% Input: number of bootstraps B
%        matrix X of features, with n rows (samples), d columns (features)
%        vector y of scalar values, with n rows (samples), 1 column
% Output: vector z of B rows, 1 column
function z = bootstrapping(B,X,y)
```

Submission: Please, submit a single ZIP file **through Blackboard**. Your MATLAB code (**kfoldcv.m**, **bootstrapping.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**