CS 6375 Spring 2020

Problem Set 1

Netid: SAS19003
Name: Shubham Sanghavi
Email: shubham.sanghavi@utdallas.edu

Warm-Up : Subgradients and More

1. Recall that a function $f : \mathbb{R}^n \to \mathbb{R}$ is convex for all $x, y \in \mathbb{R}^n$ and $\lambda \in [0, 1], if$ $\lambda f(x) + (1-\lambda)f(y) \geq f(\lambda x + (1-\lambda)y)$. Using this definition, show that

   (a) $f(x) = wf_1(x)$ is a convex function for $x \in \mathbb{R}^n$ whenever $f_1 : \mathbb{R}^n \to \mathbb{R}$ is a convex function and $w \geq 0$

   Ans:

$$
\begin{aligned}
& \lambda f(x) + (1-\lambda)f(y) \\
= \;& \lambda w f_1(x) + (1-\lambda)w f_1(y) \quad [\because f(x) = wf_1(x)\ ] \\
= \;& w(\lambda f_1(x) + (1-\lambda)f_1(y)) \\
\geq \;& wf_1(\lambda x + (1-\lambda)y) \\
& \qquad [\because w \geq 0 \qquad \text{and} \\
& \qquad\ \because f_1 \text{ is a convex function} \\
& \qquad\qquad \iff \lambda f_1(x) + (1-\lambda)f_1(y) \geq f_1(\lambda x + (1-\lambda)y)\ ] \\
= \;& f(\lambda x + (1-\lambda)y) \\
\therefore \;& \lambda f(x) + (1-\lambda)f(y) \geq f(\lambda x + (1-\lambda)y \\
\therefore \;& f \text{ is a convex function}
\end{aligned}
$$

   (b) $f(x) = f_1(x) + f_2(x)$ is a convex function for $x \in \mathbb{R}^n$ whenever $f_1 : \mathbb{R}^n \to \mathbb{R}$ and $f_2 : \mathbb{R}^n \to \mathbb{R}$ are convex functions

   Ans:

$$
\begin{aligned}
& \lambda f(x) + (1-\lambda)f(y) \\
= \;& \lambda(f_1(x) + f_2(x)) + (1-\lambda)(f_1(y) + f_2(y)) \\
= \;& \lambda f_1(x) + (1-\lambda)f_1(y) + \lambda f_2(x) + (1-\lambda)f_2(y) \\
\geq \;& f_1(\lambda x + (1-\lambda)y) + f_2(\lambda x + (1-\lambda)y) \\
& \qquad [\because f_1 \text{ and } f_2 \text{ are convex functions }] \\
= \;& f(\lambda x + (1-\lambda)y) \\
\therefore \;& \lambda f(x) + (1-\lambda)f(y) \geq f(\lambda x + (1-\lambda)y \\
\therefore \;& f \text{ is a convex function}
\end{aligned}
$$

(c) $f(x) = max\{f_1(x), f_2(x)\}$ is a convex function for $x \in \mathbb{R}^n$ whenever $f_1 : \mathbb{R}^n \to \mathbb{R}$ and $f_2 : \mathbb{R}^n \to \mathbb{R}$ are convex functions

Ans:

$$
\begin{array}{ll}
 & \lambda f(x) + (1-\lambda) f(y) \\
= & \lambda\ max\{f_1(x), f_2(x)\} + (1-\lambda)\ max\{f_1(y), f_2(y)\} \\
\geq & \lambda f_1(x) + (1-\lambda) f_1(y) \qquad \text{and} \qquad \geq \quad \lambda f_2(x) + (1-\lambda) f_2(y) \\
 & \quad [\ \because\ max\{f_1(x), f_2(x)\} \geq f_1(x)\ ] \qquad [\ \because\ max\{f_1(x), f_2(x)\} \geq f_2(x)\ ] \\
\geq & f_1(\lambda x + (1-\lambda) y) \qquad \text{and} \qquad \geq \quad f_2(\lambda x + (1-\lambda) y) \\
 & \quad [\ \because\ f_1 \text{ is a convex function }] \qquad [\ \because\ f_2 \text{ is a convex function }] \\
\Leftrightarrow & \lambda f(x) + (1-\lambda) f(y) \geq max\{f_1(\lambda x + (1-\lambda) y), f_2(\lambda x + (1-\lambda) y)\} \\
\Leftrightarrow & \lambda f(x) + (1-\lambda) f(y) \geq f(\lambda x + (1-\lambda) y) \\
\therefore & f \text{ is a convex function}
\end{array}
$$

Warm-up

Q2. Compute subgradients

(a) $f(x) = \max\{x^2-2x, |x|\}$

(i) $x=0$

$x^2-2x = |x| = 0$

gradient does not exist

case 1

let $x = -\epsilon$ where ϵ is a small +ve value.

$\therefore f(x) = \max\{\epsilon^2+2\epsilon, |-\epsilon|\}$
$= \max\{\epsilon^2+2\epsilon, \epsilon\}$

$f(\epsilon) = \epsilon^2+2\epsilon$

substituting x,

$f(x) = x^2-2x$

$\boxed{\frac{\partial f(x)}{\partial x} = 2x-2}$ for $x=-\epsilon$ small +ve value

(1)

case 2

let $x=\epsilon$ where ϵ is a small +ve value & $\epsilon < 1$

$\therefore f(x) = \max\{\epsilon^2-2\epsilon, \epsilon\}$

Here, $\epsilon^2-2\epsilon < \epsilon$
because $\epsilon < 1$

$\therefore$ $f(\epsilon) = \epsilon$ for $x=\epsilon$ small +ve value

$\therefore$ $\frac{\partial f(x)}{\partial x} = 1$ ——— (2)

$\therefore$ From (1) and (2),

Substituting $x=0$ gives us the range of subgradients.

$$2x-2 \leq \left(\frac{\partial f(x)}{\partial x}\right)_{x=0} \leq 1$$

$$-2 \leq \left(\frac{\partial f(x)}{\partial x}\right)_{x=0} \leq 1$$

All the lines who has slope between these slopes are subgradients at $x=0$.

$\therefore$ $\frac{\partial f(x)}{\partial x} = 0$, is one of the subgradients

$$\boxed{\frac{\partial f(x)}{\partial x} = 0}$$

(ii) $f(x) = \max\{x^2 - \cancel{x}\, x, |x|\}$

$x = -2$

$x^2 - 2x = +4 + 4 = 8$

$|x| = 2$

$\therefore f(x) = x^2 - 2x$ $(x^2 - 2x > |x|$ at $x = -2)$

$\dfrac{\partial f(x)}{\partial x} = 2x - 2$

for $x = -2$;

$\dfrac{\partial f(x)}{\partial x} = 2(-2) - 2 = -6$

//

b). $g(x) = \max\{(x-1)^2, (x-2)^2\}$

(i) $x = 1.5$

At $x = 1.5$,
$(x-1)^2 = (x-2)^2 = (0.5)^2 = 0.25$

The gradient does not exist.

Take $\epsilon \neq$ some small +ve value.

Case 1

If $x = 1.5 - \epsilon$

$(x-1)^2 = (0.5-\epsilon)^2$

$(x-2)^2 = (-0.5-\epsilon)^2$

$= (0.5+\epsilon)^2$

$\therefore (x-2)^2 > (x-1)^2$ for $x = 1.5-\epsilon$

$\therefore g(x) = (x-2)^2$

$\therefore \frac{\partial g(x)}{\partial x} = 2(x-2) \cdot 1$

$= 2(x-2)$ —— (1)

Case 2

$x = 1.5 + \epsilon$

$(x-1)^2 = (0.5+\epsilon)^2$

$(x-2)^2 = (-0.5+\epsilon)^2$

$= (0.5-\epsilon)^2$

$\therefore (x-1)^2 > (x-2)^2$ for $x = 1.5+\epsilon$

$\therefore g(x) = (x-1)^2$

$\therefore \frac{\partial g(x)}{\partial x} = 2(x-1)$

At $x = 1.5$,

Substituting $x = 1.5$ in ① and ②,

$$2(x-2) \leq \left(\frac{\partial g(x)}{\partial x}\right)_{x=1.5} \leq 2(x-1)$$

$$-1 \leq \frac{\partial g(x)}{\partial x} \leq 1$$

All the lines with this slope at $x = 1.5$ are subgradients.

$\frac{\partial g(x)}{\partial (x)} = 0$ is a subgradient at $x = 1.5$

(ii) For $g(x) = \max\{(x-1)^2, (x-2)^2\}$

at $x = 0$

$$g(x) = (x-2)^2 \qquad \left(\because (x-2)^2 > (x-1)^2 \text{ at } x=0\right)$$

$$\therefore \frac{\partial g(x)}{\partial x} = 2(x-2)$$

$$\therefore \boxed{\frac{\partial g(x)}{\partial x} = -4}$$

is subgradient at $x = 0$ //

Problem 1: Perceptron Learning (30 pts)

1. Standard subgradient descent with the step size Gamma= 1 for each iteration.

Code:

```
perceptron_data = importdata('perceptron.data',',');

X = perceptron_data(:,1:end-1);
Y = perceptron_data(:,end);

w = zeros(1,size(X,2));
b = 0;

w_first3 = zeros(3,size(X,2));
b_first3 = zeros(3,1);

grad_w = ones(size(w));
grad_b = ones(size(b));

p_loss = Inf;
iter = 0;
max_iter = 1000;

% step size
gamma = 1;
loss_history = [];

while iter < max_iter

    pred = (X * w.') + b ;

    loss_each = -1 * ( Y .* pred);
    incorrect = loss_each >= 0;

    p_loss = sum(loss_each .* incorrect) ;
    loss_history = [loss_history ,p_loss] ;

    grad_w = -1 * (sum((incorrect .* Y) .* X));
    grad_b = -1 *(sum(incorrect .* Y));

    % check if all our gradients are zero, stop if they are
    if (grad_b == 0) && (all(grad_w==0))
        break
    end

    iter = iter + 1;

    w = w - gamma * grad_w;
    b = b - gamma * grad_b;

    % to note the first 3 values
    if iter <= 3
```

```
        w_first3(iter,:) = w;
        b_first3(iter,:) = b;
    end

end

plot(loss_history(2:end));
w_first3
b_first3
iter
w
b
```

Results:

Standard Gradient Descent ; Step size = 1

First 3 iterations weight vectors =

1.0e+03 *

1.2790	0.4601	-0.1086	-1.6723
1.3073	0.4327	-0.0276	-1.5238
1.2552	0.4255	0.0188	-1.4347

First 3 iterations bias values =

-354
-493
-625

Total number of iterations = 46

Final weights
w = [685.7993 , 243.8995, 8.2420, -797.6251]

Final bias
b = -1485

2. Stochastic subgradient descent where exactly one component of the sum is chosen to approximate the gradient at each iteration. Instead of picking a random component at each iteration, you should iterate through the data set starting with the first element, then the second, and so on until the Mth element, at which point you should start back at the beginning again.

Again, use the step size t = 1.

Code:

```
perceptron_data = importdata('perceptron.data',',');

X = perceptron_data(:,1:end-1);
Y = perceptron_data(:,end);

w = zeros(1,size(X,2));
b = 0;

w_first3 = zeros(3,size(X,2));
b_first3 = zeros(3,1);

grad_w = ones(size(w));
grad_b = ones(size(b));

p_loss = Inf;
iter = 0;
max_iter = Inf;

gamma = 1;
loss_history = [];
idx = 1;

while iter < max_iter

    pred = (X * w.') + b ;

    loss_each = -1 * ( Y .* pred);
    incorrect = loss_each >= 0;

    if sum(incorrect) == 0
        break
    end

    p_loss = sum(loss_each .* incorrect) ;
    loss_history = [loss_history ,p_loss] ;

    idx = mod(idx,size(X,1)) + 1;

    % go to the next incorrect sample
    if incorrect(idx) == 0
        offset = find(incorrect(idx:end),1);
```

```
        if isempty(offset)
            idx = 1;
            offset = find(incorrect(idx:end),1);
        end
        idx = idx + offset - 1;
    end

    grad_w = -1 * (Y(idx) * incorrect(idx) * X(idx,:));
    grad_b = -1 * (Y(idx) * incorrect(idx));

    % check if all our gradients are zero, stop if they are
    % if (grad_b == 0) && (all(grad_w==0))
    % For the sake of uniformity, we use this instead in assignment

    w = w - gamma * grad_w;
    b = b - gamma * grad_b;

    iter = iter + 1;
    idx = idx + 1;

    if iter <= 3
        w_first3(iter,:) = w;
        b_first3(iter,:) = b;
    end

end
```

Results:

Stochastic Gradient Descent, Stepsize = 1

First 3 iterations weight vectors =

1.1110	2.5928	-1.1492	-0.5725
-2.6028	1.5825	1.6626	-4.6442
-1.6534	3.9397	3.8814	-2.4734

First 3 iterations bias values =

-1
-2
-3

Total number of iterations = 5371

Final weights
w = [114.8349, 41.2125, 1.7244, -133.2039]

Final Bias
b = -249

3. How does the rate of convergence change as you change the step size? Provide some example step sizes to back up your statements.

Answer:

For the standard as well as stochastic gradient descent, for the perceptron algorithm, The code takes the same number of iterations to converge with step sizes 0.001, 1 and 1000.

The rate of convergence doesn't change with the step size because our hypothesis is scale invariant in w and b.

i.e.

Our hypothesis consists of an hyperplane whose equation is

$$w^T x + b = 0$$

For some iteration t, let w_t, b_t be the w and b.

For that iteration, let $\Delta w_t, \Delta b_t$ be the gradients

$$\therefore \left.\begin{aligned} w_{t+1} &= w_t - \alpha \Delta w_t \\ b_{t+1} &= b_t - \alpha \Delta b_t \end{aligned}\right\} \alpha \text{ is the step size}$$

Therefore, Our hyperplane becomes:

$$w_{t+1}^T x + b_{t+1} = 0$$

$$(w_t - \alpha \Delta w_t)^T x + (b_t - \alpha \Delta b_t) = 0$$

$$w_t^T x - \alpha \Delta w_t^T x + b_t - \alpha \Delta b_t = 0$$

$$\underbrace{w_t^T x + b_t}_{=0} - \alpha \Delta w_t^T x - \alpha \Delta b_t = 0$$

this is our previous ~~of~~ hyperplane

$$\therefore \quad -\alpha \Delta w_t^T x - \alpha \Delta b_t = 0$$

$$\therefore \quad -\alpha (\Delta w_t^T x + \Delta b_t) = 0$$

$$\Leftrightarrow \Delta w_t^T x + \Delta b_t = 0 \qquad (\because \alpha \neq 0)$$

$\therefore$ No matter what α (step size) we select, we always iteratively predict the same hyperplane.

$\therefore$ The rate of convergence do not change with α.

4. What is the smallest, in terms of number of data points, two-dimensional data set containing both class labels on which the algorithm, with step size one, fails to converge? Use this example to explain why the method may fail to converge more generally.

Answer:

The smallest number of data points, in a 2-D data set containing both class labels on which the algorithm fails to converge is 2.

i.e.

if for some input x1,x2 we get both +ve and -ve label, the algorithm will fail to converge.
(1,1) -> 1
(1,1) -> -1

In general, If we contain an ambiguous dataset where in we get both the labels for same inputs, the algorithm fails to converge.

Also, if no linear separator is possible for the data in our feature space, the algorithm fails to converge.

Problem 2

1. Is the data linearly seperable

(a) $\phi(x_1, x_2) = \begin{bmatrix} x_1 + x_2 \\ x_1 - 2x_2 \end{bmatrix}$

x_1	x_2	$x_1' = x_1 + x_2$	$x_2' = x_1 - 2x_2$	label
-1	-1	-2	1	+
1	1	2	-1	+
-1	1	0	-3	−
1	-1	0	3	−

$\therefore$ We need to find a line
$w_1 x_1' + w_2 x_2' + b = 0$

s.t.

$$b + -2w_1 + w_2 > 0 \qquad b - 3w_2 < 0$$
$$b + 2w_1 - w_2 > 0 \qquad b + 3w_2 < 0$$

(+) $2b > 0$ — ① (+) $2b < 0$ — ②

From ① and ②;
$\boxed{b = 0.}$

now, $-2w_1 + w_2 > 0 \Rightarrow w_2 > 2w_1$
$2w_1 - w_2 > 0 \Rightarrow 2w_1 > w_2$

$\omega_2 > 2\omega_1$ & $\omega_1 > 2\omega_2$

cannot be satisfied with any values of ω_1, ω_2

$\therefore$ Linear seperator does not exist.

(b) $\phi(x_1, x_2) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1 x_2 \end{bmatrix}$

This set contains a linear seperator.

I have solved it using code and the seperator is given by

$\Rightarrow \omega_1 x_1^2 + \omega_2 x_2^2 + \omega_3 x_1 x_2 + b = 0$

where $\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} \ \end{bmatrix}$

and $b =$

(c) $\phi(x_1, x_2) = \begin{bmatrix} \exp(x_1) \\ \exp(x_2) \end{bmatrix}$

$\therefore$ We want $w_1 e^{x_1} + w_2 e^{x_2} + b = 0$

for

$$w_1 e^{-1} + w_2 e^{-1} + b > 0 \quad \text{--- } (1)$$
$$w_1 e + w_2 e + b > 0 \quad \text{--- } (2)$$

$$w_1 e^{-1} + w_2 e + b < 0 \quad \text{--- } (3)$$
$$w_1 e + w_2 e^{-1} + b < 0 \quad \text{--- } (4)$$

Add (1) and (2);

$$w_1 (e^{-1} + e) + w_2 (e + e^{-1}) + 2b > 0$$
$$\text{--- } (5)$$

Add (3) and (4),

$$w_1 (e^{-1} + e) + w_2 (e^{-1} + e) + 2b < 0$$
$$\text{--- } (6)$$

From (5) and (6),
We conclude that a linear seperator is not possible.

Problem 2: Q2

Polynomial regression for 2-D data points.

The data in 2-d is of the form (x, y).

Thus, for polynomial regression, we need a K-degree polynomial of x for which the output is y.

$\therefore$ Our hypothesis function:

$$\underbrace{w_K x^K + w_{K-1} x^{K-1} + \ldots\ldots + b x^0}_{} = \underbrace{y'}_{\text{Predictions.}}$$

In vector notation;

$$\underline{f(x) = w^T X + b}$$

where $X = \underbrace{\begin{bmatrix} x^K \\ x^{K-1} \\ \vdots \\ \vdots \\ x \end{bmatrix}}_{\text{Feature Vectors}}$ $\quad w = \begin{bmatrix} w^K \\ w^{K-1} \\ \vdots \\ \vdots \\ w_1 \end{bmatrix}$

Feature vector X is generated from our input x.

The loss function: mean square error

$$L(f) = \frac{1}{M} \sum_{m} \left(f(x^m) - y^m \right)^2$$

$$L(f) = \frac{1}{M} \sum_{m} \left((w^T x^m + b) - y^m \right)^2$$

And the objective function is given by

$$\min_{w,b} L(f) = \min_{w,b} \frac{1}{M} \sum_{m} \left((w^T x^m + b) - y^m \right)^2$$

Given the objective, we will use gradient descent to compute the w,b for minimum loss

∴ Gradient descent

$$\frac{\partial L}{\partial w} = \frac{2}{M} \sum_{m} \left((w^T x^m + b) - y^m \right) \cdot x^m$$

$$\frac{\partial L}{\partial b} = \frac{2}{M} \sum_{m} \left((w^T x^m + b) - y^m \right)$$

Iteratively compute;

$$w_{t+1} := w_t - \alpha \frac{\partial L}{\partial w}$$

$$b_{t+1} := b_t - \alpha \frac{\partial b}{\partial b}$$

Untill, $\frac{\partial L}{\partial w} \simeq 0$ and $\frac{\partial L}{\partial b} \simeq 0$.

Time Complexity (K degree polynomial, m data points)

For a K degree polynomial, the predictions will take $O(Km)$ time. And the gradient computing using the predictions also takes $O(Km)$ time.

∴ Time Complexity $= O(Km) + O(Km) = O(Km)$

Problem 3 Exponential Regression.

) Input $(x, y) \in \mathbb{R}^2$

M input data points.

Hypothesis

$$f(x) = \exp(ax+b)$$

loss

$$L(f) = \frac{1}{M} \sum_m \left(\exp(ax^m+b) - y^m\right)^2$$

objective

$$\min_{a,b} L(f) = \min_{a,b} \frac{1}{M} \sum_m \left(\exp(ax^m+b) - y^m\right)^2$$

We minimize this function using gradient descent.

Gradient Descent

$$\min_{a,b} L(f) = \min_{a,b} \frac{1}{M} \sum_{m} \left(\exp(a x^m + b) - y^m \right)^2$$

We have to compute a, b to minimize loss.

$$\therefore \frac{\partial L}{\partial a} = \frac{2}{M} \sum_{m} \left(\exp(a x^m + b) - y^m \right) \exp(a x^m + b) \cdot x^m$$

$$\frac{\partial L}{\partial b} = \frac{2}{M} \sum_{m} \left(\exp(a x^m + b) - y^m \right) \exp(a x^m + b)$$

Iteratively: (simultaneously)

$$a_{t+1} = a_t - \alpha \frac{\partial L}{\partial a}$$

$$b_{t+1} = b_t - \alpha \frac{\partial L}{\partial b}$$

Until, $\frac{\partial L}{\partial a} \approx 0$ and $\frac{\partial L}{\partial b} \approx 0$.