CS6375 HomeWork I

Hailiang Dong

September 9, 2018

Warm-Up: Subgradients and More

∗∗ Independent work declaration: all 'we' used in this report are just as same as 'I'. The reason I use 'we' instead of 'I' is to make reading much more naturally.
∗∗ Codes for problem 1 and 4 are submitted in separate files.

Q1

Proof: Since $f(x) = max\{f_1(x), f_2(x)\}$ and both $f_1(x)$ and $f_2(x)$ are convex functions, we have

$$\begin{aligned} f(\lambda x + (1-\lambda)y) &= max\{f_1(\lambda x + (1-\lambda)y), f_2(\lambda x + (1-\lambda)y)\} \\ &\leq max\{\lambda f_1(x) + (1-\lambda)f_1(y), \lambda f_2(x) + (1-\lambda)f_2(y)\} \end{aligned}$$

Denote $\lambda f_1(x) + (1-\lambda)f_1(y) = A$ and $\lambda f_2(x) + (1-\lambda)f_2(y) = B$, since A and B are less or equal than $C = \lambda max\{f_1(x), f_2(x)\} + (1-\lambda)max\{f_1(y), f_2(y)\}$, we have $max\{A, B\} \leq C$. Therefore, we have

$$\begin{aligned} f(\lambda x + (1-\lambda)y) &\leq max\{\lambda f_1(x) + (1-\lambda)f_1(y), \lambda f_2(x) + (1-\lambda)f_2(y)\} \\ &\leq \lambda max\{f_1(x), f_2(x)\} + (1-\lambda)max\{f_1(y), f_2(y) \\ &= \lambda f(x) + (1-\lambda)f(y) \end{aligned}$$

Hence, $f(x)$ is a convex function.

Q2

(a) By solving function $x^2 - 2x = |x|$, we have root $x = 0, 3$, therefore we can write $f(x)$ in another form,

$$f(x) = \begin{cases} x, 0 \leq x \leq 3 \\ x^2 - 2x, otherwise \end{cases}$$

It is obvious that function $f(x)$ is not differentiable at point 0 and 3. Thus, the subgradient at $x = 0$ is not unique, the minimum value is $(x^2 - 2x)' \mid_{x=0} = -2$, and the maximum value is 1. We can take any value in closed interval $[-2, 1]$ as the subgradient for $f(x)$ at $x = 0$. As for $x = -2$, the corresponding subgradient is $(x^2 - 2x)' \mid_{x=-2} = -6$.

(b) Similarly, we can write g(x) as

$$g(x) = \begin{cases} (x-2)^2, x \leq 1.5 \\ (x-1)^2, otherwise \end{cases}$$

When at point $x = 1.5$, $g(x)$ is not differentiable, the minimum subgradient is $((x-2)^2)' \mid_{x=1.5} = -1$ and the maximum subgradient is $((x-1)^2)' \mid_{x=1.5} = 1$. Hence, we can take any subgradient in interval $[-1, 1]$. For $x = 0$, the corresponding subgradient is $((x-2)^2)' \mid_{x=0} = -4$.

Problem 1: Perceptron Learning

Q1

(1) The number of iterations to find the perfect classifier: 46
(2) The values of w and b for the first three iteration:

$w^{(1)} = [1278.99646108, 460.06125801, -108.55851404, -1672.31572948] \quad b^{(1)} = -354.0$
$w^{(2)} = [1307.29472974, 432.74778799, -27.55191988, -1523.78895446] \quad b^{(2)} = -493.0$
$w^{(3)} = [1255.18981362, 425.50402882, 18.7965404, -1434.66754197] \quad b^{(3)} = -625.0$

(3) The final weights and bias:
$w = [685.79932892, 243.89947473, 8.24199193, -797.62505314] \quad b = -1485.0$

Q2

(1) The number of iterations to find the perfect classifier: 1091000

(2) The values of w and b for the first three iteration:
$w^{(1)} = [4.61754424, 2.46967938, 1.96766079, -1.81335551] \quad b^{(1)} = -1.0$
$w^{(2)} = [4.61754424, 2.46967938, 1.96766079, -1.81335551] \quad b^{(2)} = -1.0$
$w^{(3)} = [3.45322288, 0.16943482, 2.62801595, -4.64709851] \quad b^{(3)} = -2.0$

(3) The final weights and bias:
$w = [149.27714019, 52.53347317, 1.67167265, -172.89194014] \quad b = -322.0$

Q3

The observations can be divided into two categories. The first one is **fixed step size**(constant), and the other one is **varying step size**.

For any fixed step size such as $0.01, 0.1, 0.5, 1, 5, 10$, the total number of iterations for the algorithm to converge will not change. In other words, the rate of convergence is also a constant. This is because that (1) the gradient of w and b for the first iteration is a fixed value; (2) the initial value of w and b are 0. With these two facts, assume we have two fixed step size a and b, and we denote the w in the first iteration for step size a and b is $w_a^{(1)}$ and $w_b^{(1)}$, respectively. We will have

$$\frac{w_a^{(1)}}{w_b^{(1)}} = \frac{a}{b}$$

For example, when the step size is 0.5, we will obtain
$w^{(1)} = [639.49823054, 230.03062901, -54.27925702, -836.15786474]$
which is exactly half of the $w^{(1)}$ in Q1. This observation holds true for any $w_a^{(i)}$ and $w_b^{(i)}$. Therefore, the total number of iterations for the algorithm to converge will be a fix number.

When varying step size is used, consider general function $1/(a + bt^c)$, the detailed results are shown in Table 1.

Table 1: Number of iterations to converge for different varying step size

$\gamma_i(t)$	$\gamma_i(1)$	$\gamma_i(100)$	**#Iteration to converge**
1	1	1	46
$\frac{1}{\sqrt{t}}$	1	0.1	188
$\frac{1}{10+t}$	0.091	0.0091	96
$\frac{1}{5+t}$	0.167	0.009524	388
$\frac{1}{3+t}$	0.25	0.009709	2172
$\frac{1}{3.5+0.5t}$	0.25	0.0187	179

From the results above, there are two observations. (1) The slower the step size decreases, the faster the algorithm converges. For example, consider the function $\gamma_1(t)$ and $\gamma_2(t)$, they have the same value when $t = 1$, which means that the value of w in the first iteration for these two step sizes are same. However, for $\gamma_1(t)$, it outputs a constant value, and the step size never decreases. As for $\gamma_2(t)$, the step size decreases faster when compared to $\gamma_1(t)$, and the value of step size is 0.1 when $t = 100$. Therefore, it needs more iterations to converge. What is more, the comparison between $\gamma_5(t)$ and $\gamma_6(t)$ again proves the observation. (2) If the step size decreases at same rate (i.e., have same b and c), the bigger the a is, the faster the algorithm converges. Consider function $\gamma_3(t)$, $\gamma_4(t)$ and $\gamma_5(t)$, we can see that $\gamma_3(t)$ converges fastest since it has the largest a, which proves the observation.

Q4

Generally, when the data is not separable, the algorithm cannot converge. For a two-dimensional data set contains both class labels, the minimum number of points such that the data not separable is 3. Specifically, when three points lie on the same line and middle one has different label, the data set will not be separable. For example, consider the data set D that contains the following three points:

$$x^{(1)} = (1,1), y^{(1)} = 1 \qquad x^{(2)} = (2,1), y^{(2)} = -1 \qquad x^{(3)} = (3,1), y^{(3)} = 1$$

It is obvious that D is not separable, when apply the Perceptron to this dataset with step size $\gamma_t = 1$, the gradient can never be zero and w, b are updated in a periodic way. Specifically, for this example, the period is 2, which means $w^{(i)} = w^{(i\%2)}, b^{(i)} = b^{(i\%2)}$. Hence, the algorithm cannot converge.

Problem 2: Separability & Feature Vectors

Q1

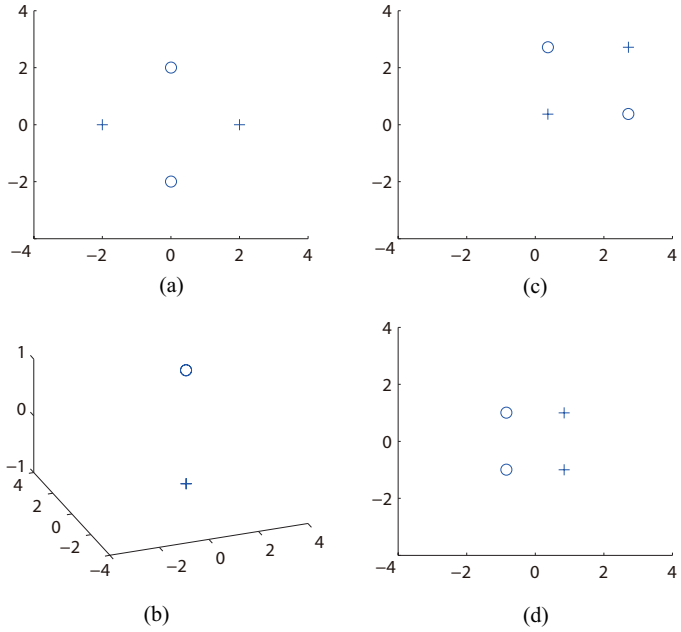

Figure 1: Representation of the data set using different feature vectors

We represent the data set under different feature vector in Figure 1 (The 'o' mark stands for the negative samples). Based on the figure, it is obvious that data is not separable under the feature vector (a) and (c). For feature vector (b), it maps all positive sample to point (1,1,1) and all negative sample to point (1,1,-1). Therefore, the data set is separable under vector (b) and we can simply use plane $z = 0$ to be the separator. As for feature vector (d), we can use line $x = 0$ to be the separator.

Q2

Assume we have n data points with form $(x^{(i)}, y^{(i)})$, to fit a polynomial of degree k on two-dimensional space, one possible loss function is $\sum_{i=1}^{n} \left(w_k (x^{(i)})^k + w_{k-1}(x^{(i)})^{k-1} + ... + w_1(x^{(i)}) + b - y^{(i)}\right)^2$. We can turn the polynomial regression problem into a linear regression problem in $k+1$ dimensional space by using the feature vector

$$\phi(x) = \left[x^k, x^{k-1}, ..., x^2, x\right]^T$$

. Then we can rewrite the loss function as $\sum_{i=1}^{n} \left(w^T \phi(x^{(i)}) + b - y^{(i)}\right)^2$. Denote $\phi(x^{(i)})$ as $p^{(i)}$, we can easily obtain the gradient for w and b, which is $\nabla w = 2\sum_i \left(w^T p^{(i)} + b - y^{(i)}\right) \cdot p^{(i)}$ and $\nabla b = 2\sum_i w^T p^{(i)} + b - y^{(i)}$. Therefore, for each iteration of gradient descent, the computational complexity is $O(n \cdot k)$.

Problem 3: Piecewise Linear Regression

Q1

Assume we have n data points with form $(x^{(i)}, y^{(i)})$, we can formulate the problem as

$$\min_{a_1,a_2,b_1,b_2 \in R} \quad \frac{1}{2}\sum_{i=1}^{n}\left(f(x^{(i)}) - y^{(i)}\right)^2$$

i.e.,

$$\min_{a_1,a_2,b_1,b_2 \in R} \quad \frac{1}{2}\sum_{i=1}^{n}\left(max\{a_1x^{(i)} + b_1, a_2x^{(i)} + b_2\} - y^{(i)}\right)^2$$

Q2

The corresponding subgradients for a_1, a_2, b_1, b_2 are

$$\nabla a_1 = \sum_i \left(max\{a_1x^{(i)} + b_1, a_2x^{(i)} + b_2\} - y^{(i)}\right) \cdot x^{(i)} \cdot 1_{a_1x^{(i)}+b_1 \geq a_2x^{(i)}+b_2}$$
$$\nabla b_1 = \sum_i \left(max\{a_1x^{(i)} + b_1, a_2x^{(i)} + b_2\} - y^{(i)}\right) \cdot 1_{a_1x^{(i)}+b_1 \geq a_2x^{(i)}+b_2}$$
$$\nabla a_2 = \sum_i \left(max\{a_1x^{(i)} + b_1, a_2x^{(i)} + b_2\} - y^{(i)}\right) \cdot x^{(i)} \cdot 1_{a_1x^{(i)}+b_1 \leq a_2x^{(i)}+b_2}$$
$$\nabla b_2 = \sum_i \left(max\{a_1x^{(i)} + b_1, a_2x^{(i)} + b_2\} - y^{(i)}\right) \cdot 1_{a_1x^{(i)}+b_1 \leq a_2x^{(i)}+b_2}$$

Q3

This optimization problem is not convex. By setting $b_1 = 0$, $a_2 = 0$, $b_2 = -1$, the loss function will be a function of variable a_1. Assume we have only one sample $(1, 1)$, we can write the loss function as $(max(a_1, -1) - 1)^2$. Part of the function graph is shown in Figure 2. It clearly indicates that the loss function is not convex. Therefore, this is not a convex optimization problem. ($**$ Discussed with Omeed Ashtiani)

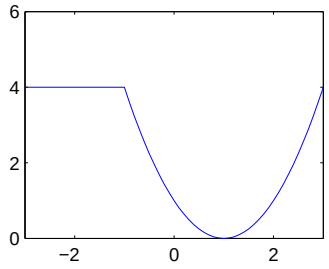

Figure 2: An example indicates that loss function is not convex

Problem 4: Support Vector Machines

Assume the function of hyperplane is $w^Tx + b = 0$. Our objective is to maximize the margin, which is $1/||w||$, such that all the data points lie outside the margin, i.e.,

$$\min_{w,b} \frac{1}{2}||w||^2$$
$$s.t.$$
$$y^{(i)}(w^Tx^{(i)} + b) \geq 1, \forall i$$

However, if we directly apply quadratic programming solver on this problem, the solver will tell us there is no solution, which means the data is not separable in the original 4-dimensional space.

So I begin to try non-linear separators. Just like P2:Q2, we can utilize feature vectors that map original samples to a higher dimensional space and then find linear separators in that space. This is equivalent to finding a non-linear separator in the original space. When considering non-linear separator with degree k, we assume the corresponding feature vector is $\phi_k(x)$. Then our problem would be

$$\begin{aligned}
&\min_{w,b} \frac{1}{2}||w||^2 \\
&s.t. \\
&\quad y^{(i)}(w^T\phi_k(x^{(i)}) + b) \geq 1, \forall i
\end{aligned}$$

First, we first consider curves of degree 2 and the corresponding $\phi_k(x)$ is

$$\phi_2(x) = [x_1^2, x_2^2, x_3^2, x_4^2, x_1x_2, x_1x_3, x_1x_4, x_2x_3, x_2x_4, x_3x_4, x_1, x_2, x_3, x_4]^T$$

In this case, the quadratic programming solver outputs the optimal solution, which means the data is separable under feature vector $\phi_2(x)$. The corresponding weight and bias obtained are

$$\begin{aligned}
w =&[413.6589, -76.1129, -326.2872, 14.4021, 779.7009, -456.2277, 31.8074, \\
&82.8830, -7.9145, -7.8671, 365.7695, 68.0797, 34.9701, -9.5999]^T \\
b =& -13.238951
\end{aligned}$$

After we have obtain w and b, the optimal margin is given as $1/||w|| = 0.000895$. Also, we can obtain the support vectors (shown in the original space, totally 14 support vectors):

$$\begin{aligned}
&(0.224468, 0.075993, 0.493421, 0.584895) \\
&(0.434927, 0.547633, 0.974760, 0.019790) \\
&(0.042116, 0.929565, 0.462324, 0.187416) \\
&(0.026371, 0.142476, 0.187497, 0.564988) \\
&(0.629116, 0.200368, 0.916962, 0.299793) \\
&(0.054959, 0.019799, 0.175049, 0.199957) \\
&(0.245372, 0.557390, 0.773293, 0.314391) \\
&(0.020697, 0.059949, 0.121393, 0.930666) \\
&(0.013393, 0.946371, 0.103407, 0.845245) \\
&(0.348068, 0.575657, 0.905885, 0.951837) \\
&(0.781259, 0.059405, 0.940766, 0.966893) \\
&(0.006876, 0.531593, 0.281975, 0.278394) \\
&(0.065007, 0.089742, 0.286557, 0.709984) \\
&(0.214893, 0.896882, 0.858453, 0.565993)
\end{aligned}$$