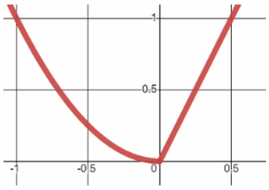
Nick 322315318

1. Let $f(x) = \begin{cases} x^2, & x < 0 \\ 2x, & x \geq 0 \end{cases}$.



1.1. Is f convex? No need to explain.

1.2. Propose a sub-derivative function g for f. That is, $g \in \partial f$. Use the above definition to prove that $g(u) \in \partial f(u), \forall u \in \mathbb{R}$.

1.3. Set a learning rate of $\eta = 0.25$ and a starting point $x_0 = -1.5$. Running subgradient descent, will the algorithm converge to a minimum? Prove your answer by filling the following table like we did in Tutorial 07 using as many rows as needed.

i	x_i	$f(x_i)$	$\frac{\partial}{\partial x} f(x_i) = g(x_i)$
0	-1	1	
1			
⋮			

1.4. Repeat 1.3 with $\eta = 1,\ x_0 = -1.5$.

1.1. yes

1.2. $g(u) = \begin{cases} 2u, & u < 0 \\ 2, & u \geq 0 \end{cases}$

$\underline{u<0:}$ $f(v) \geq f(u) + 2u(v-u)$

- $\underline{v<0:}$ $v^2 \geq u^2 + 2uv - 2u^2$
  $v^2 - 2uv + u^2 \geq 0$
  $(v-u)^2 \geq 0$ which holds always
- $\underline{v \geq 0:}$ $2v \geq u^2 + 2u(v-u)$
  $\underbrace{u^2 + 2v - 2uv}_{-uv \geq 0} \geq \underbrace{u^2 + 2v \geq 0}_{\text{from } u,v \text{ definition}}$ which always holds

$\underline{u \geq 0:}$ $f(v) \geq f(u) + 2(v-u)$

- $\underline{v<0:}$ $v^2 \geq 2u + 2v - 2u$
  $v^2 - 2v \geq 0$ which always holds since $v^2 > 0,\ -2v > 0$
- $\underline{v \geq 0:}$ $2v \geq 2u + 2v - 2u$
  $0 \geq 0$ which is always true

1.3. $f(x) = \begin{cases} x^2, & x<0 \\ 2x, & x \geq 0 \end{cases}$ $\nabla f(x) = \begin{cases} 2x, & x<0 \\ 2, & x \geq 0 \end{cases}$

$x_{i+1} = x_i - \eta \cdot f'(x_i)$

$x_0 = -1.5$; $f(x_0) = \frac{9}{4}$; $\nabla f(x_0) = -3$

$x_1 = -1.5 - \frac{1}{4} \cdot (-3) = -\frac{3}{4}$; $f(x_1) = \frac{9}{16}$; $\nabla f(x_1) = -\frac{3}{2}$

$x_2 = -\frac{3}{4} - \frac{1}{4}\ -\frac{3}{2}) = -\frac{3}{8}$; $f(x_2) = \frac{9}{64}$; $\nabla f(x_2) = -\frac{3}{4}$

$x_3 = -\frac{3}{8} - \frac{1}{4}\left(-\frac{3}{4}\right) = -\frac{3}{16}$; $f(x_3) = \frac{9}{256}$; $\nabla f(x_3) = -\frac{3}{8}$

we can see that $x_i \xrightarrow{i \to \infty} 0$; $f(x_i) \xrightarrow[i \to \infty]{} 0$ and $\nabla f(x_i) \xrightarrow[i \to \infty]{} 0$

1.4 $x_0 = -1.5$; $f(x_0) = \frac{9}{4}$; $\nabla f(x_0) = -3$

$x_1 = -1.5 - 1 \cdot (-3) = 1.5$; $f(x_1) = 3$; $\nabla f(x_1) = 2$

$x_2 = 1.5 - 1 \cdot 2 = -\frac{1}{2}$; $f(x_2) = \frac{1}{4}$; $\nabla f(x_2) = -1$

$x_3 = -\frac{1}{2} - 1 \cdot (-1) = \frac{1}{2}$; $f(x_3) = 1$; $\nabla f(x_3) = 2$

$x_4 = \frac{1}{2} - 1 \cdot 2 = -1.5$; $f(x_4) = \frac{9}{4}$; $\nabla f(x_4) = -3$

$x_0 = x_4 \Rightarrow$ we'll be in an infinite non-converging loop

2. This exercise will investigate the regularization coefficient λ as it was presented in the ridge linear regression section of this course. Suppose we are trying to fit a polynomial to the following data:

X	Y
0	0
1	3
2	12

Our hypothesis class for this problem will be

$$\mathcal{H} = \{w_0 + w_1x + w_2x^2 + w_3x^3 : (w_0, w_1, w_2, w_3) \in \mathbb{R}^4\}.$$

2.1. Show that we can fit the data with $w_0 = 0, w_1 = 2, w_2 = 0, w_3 = 1$.

2.2. Show that our hypothesis class is too expressive for the problem we're dealing with. In other words, find a simple quadratic polynomial that fits the data perfectly.

2.1 $\quad 0 + 2x + 0 + x^3 = x^3 + 2x$

$x = 0 \rightarrow y = 0$
$x = 1 \rightarrow y = 3$
$x = 2 \rightarrow y = 12$

2.2 $\quad 3x^2$

$x = 0 \rightarrow y = 0$
$x = 1 \rightarrow y = 3$
$x = 2 \rightarrow y = 12$

2.3. Denote the mean squared error (MSE)

$$\mathcal{L}(w) = \frac{1}{m}\|Xw - y\|_2^2,$$

Where X is the appropriate Vandermonde matrix.

Calculate $\mathcal{L}(w)$ for the quadratic model in (2.2) and the cubic model in (2.1).

$$X = \begin{bmatrix} 0^0 & 0^1 & 0^2 & 0^3 \\ 1^0 & 1^1 & 1^2 & 1^3 \\ 2^0 & 2^1 & 2^2 & 2^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \end{bmatrix}$$

$$W_{2.1} = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 1 \end{bmatrix} \quad W_{2.2} = \begin{bmatrix} 0 \\ 0 \\ 3 \\ 0 \end{bmatrix} \quad y = [0 \ 3 \ 12]$$

$$\mathcal{L}(w_{2.1}) = \|[0 \ 3 \ 12] - [0 \ 3 \ 12]\|_2^2 = 0$$

$$\mathcal{L}(w_{2.2}) = \|[0 \ 3 \ 12] - [0 \ 3 \ 12]\|_2^2 = 0$$

2.4. The best <u>line</u> for fitting the data is $y = 6x - 1$. Calculate $\mathcal{L}(w)$ for this line.

$$W_{2.4} = \begin{bmatrix} -1 \\ 6 \\ 0 \\ 0 \end{bmatrix} \quad \mathcal{L}(w_{2.4}) = \frac{1}{3}\|[-1 \ 5 \ 11] - [0 \ 3 \ 12]\|_2^2 = \frac{1}{3}\|[-1 \ 2 \ -1]\|_2^2 = \frac{1}{3} \cdot 6 = 2$$

2.5. Now denote the MSE with regularization as show in class

$$\mathcal{L}_\lambda(w) = \frac{1}{m}\|Xw - y\|_2^2 + \lambda\|w\|_2^2.$$

Here $\lambda > 0$ is a hyperparameter, which is not given. As we learned in class, the regularization imposes a "cost" on models with large coefficients. Calculate $\mathcal{L}_\lambda(w)$ for each of the three models in (2.1), (2.2) and (2.4).

2.5.1 $\mathcal{L}_\lambda(w_{2.1}) = 5\lambda$

2.5.2 $\mathcal{L}_\lambda(w_{2.2}) = 9\lambda$

2.5.4 $\mathcal{L}_\lambda(w_{2.4}) = 2 + 37\lambda$

2.6. As it turns out, $\mathcal{L}_\lambda(w)$ would never prefer the simple quadratic polynomial over the cubic polynomial we found, no matter the value of $\lambda > 0$. Can you explain why?

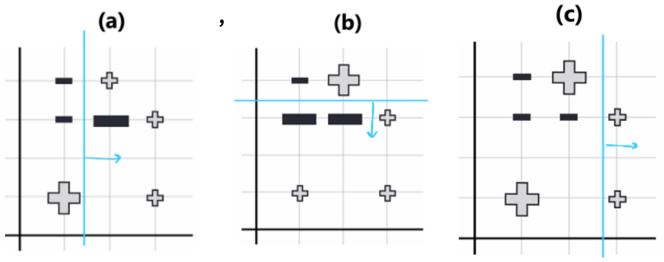
as explained in 2.5, the regularization imposes a "cost" on models with large coefficients therefore, in our example we'd always prefer the cubic polynomial since the $\|\ \|_2^2$ of its coefficients is smaller.

2.7. Suggest a way to fix the regularization method to prefer the model we consider to be simpler.

choose λ to be a vector that punishes x^3 and rewards x^2, so we can choose something like: $\lambda = [0,0,0,1]$ and then $\mathcal{L}_\lambda(w) = \frac{1}{m}\|Xw - y\|_2^2 + \sum_{i=0}^{3} \lambda_i w_i^2$

3. Only some of the following figures depict possible distributions that can be obtained after <u>one</u> iteration of AdaBoost. **Which ones?** For each such distribution, propose a weak classifier that can lead to its figure (use a <u>clear</u> drawing or a short description of that classifier).

Only figures (a), (b) and (c) can be obtained after one iteration of AdaBoosed



(the arrow points towards the side that is supposed to be labeled as positive (+))

4. Show that the error of h_t w.r.t the distribution D^{t+1} is exactly 1/2. That is, show that $\forall t \in [T]$

$$\sum_{i=1}^{m} D_i^{t+1}\,\mathbb{1}_{[y_i \neq h_t(x_i)]} = 1/2.$$

<u>First way:</u>

prove using induction:

Base: <u>t=1</u>: without limitation of generality, let $0 \le k \le m$ be the amount of wrongly classified points by the chosen model in $t=0$

$$\varepsilon_1 = \sum_{i=1}^{m} D^1_{(i)} = \frac{1}{m}\cdot k = \frac{k}{m} \;;\; \alpha_1 = \frac{1}{2}\log\left(\frac{1}{\varepsilon_0}-1\right) = \log\left(\sqrt{\frac{m-k}{k}}\right)$$

if wrongly classified: $D_i^{t+1} \propto \frac{1}{m} e^{\log\left(\sqrt{\frac{m-k}{k}}\right)} = \frac{1}{m}\sqrt{\frac{m-k}{k}} \Longrightarrow \sum_{i=1}^{m} D_i^{t+1}\mathbb{I}_{[y_i \neq h_t(x_i)]} = k\cdot\frac{1}{m}\sqrt{\frac{m-k}{k}} = \frac{\sqrt{k(m-k)}}{m}$

if rightly classified: $D_i^{t+1} \propto \frac{1}{m} e^{\log\left(\sqrt{\frac{k}{m-k}}\right)} = \frac{1}{m}\sqrt{\frac{k}{m-k}} \Longrightarrow \sum_{i=1}^{m} D_i^{t+1}\mathbb{I}_{[y_i = h_t(x_i)]} = (m-k)\cdot\frac{1}{m}\sqrt{\frac{k}{m-k}} = \frac{\sqrt{k(m-k)}}{m}$

sum of both is equal to 1 (as the sum of probabilities), therefore each one is 1/2

Assume correctness for 't';

Show correctness for 't+1':

$$\sum_{i=1}^{m} D_i^{t+1}\mathbb{I}_{[y_i \neq h_t(x_i)]} = \sum_{i=1}^{m} \frac{D_i^t e^{-\frac{1}{2}\log\left(\frac{1}{\underbrace{\sum D_i^t \mathbb{I}_{[y_i \neq h_t(x_i)]}}_{1/2}}-1\right)}}{\sum_{j=1}^{m} D_j^t e^{-\frac{1}{2}\log\left(\frac{1}{\underbrace{\sum D_r^t \mathbb{I}_{[y_r \neq h_t(x_r)]}}_{1/2}}-1\right)}}\mathbb{I}_{[y_i \neq h_t(x_i)]} = \sum_{i=1}^{m} \frac{D_i^t e^0}{\underbrace{\sum_{j=1}^{m} D_j^t e^0}_{1}}\mathbb{I}_{[y_i \neq h_t(x_i)]} =$$

$$= \sum_{i=1}^{m} \underbrace{D_i^t}_{1}\,\underbrace{\mathbb{I}_{[y_i \neq h_t(x_i)]}}_{1/2} = \frac{1}{2}$$

<u>Second way:</u> $\sum_{i=1}^{m} D_i^{t+1}\mathbb{I}_{[N]} = \sum_{i=1}^{m} \frac{D_i^t e^{-\alpha_t y_i h_t(x_i)}}{Z_t}\mathbb{I}_{[N]} = \sum_{i=1}^{m} \frac{D_i^t e^{\alpha_t}}{Z_t}\mathbb{I}_{[N]} = \frac{1}{Z_t}\sum_{i=1}^{m} D_i^t e^{\alpha_t}\mathbb{I}_{[N]} \overset{(*)}{=} \frac{1}{2\sqrt{\varepsilon_t(1-\varepsilon_t)}}\overset{(***)}{\sqrt{\varepsilon_t(1-\varepsilon_t)}} = \frac{1}{2}$ as needed

$$Z_t \overset{(*)}{=} \sum_{i=1}^{m} D_i^t\,\mathbb{I}_{[N]}\,e^{\alpha_t} + \sum_{i=1}^{m} D_i^t\,\mathbb{I}_{[\bar N]}\,e^{-\alpha_t} = e^{\alpha_t}\sum_{i=1}^{m} D_i^t\,\mathbb{I}_{[N]} + e^{-\alpha_t}\sum_{i=1}^{m} D_i^t\,\mathbb{I}_{[\bar N]} = e^{\alpha_t}\varepsilon_t + e^{-\alpha_t}(1-\varepsilon_t) = e^{\frac{1}{2}\log\left(\frac{1}{\varepsilon_t}-1\right)}\varepsilon_t + e^{-\frac{1}{2}\log\left(\frac{1}{\varepsilon_t}-1\right)}(1-\varepsilon_t) = \sqrt{\frac{1}{\varepsilon_t}-1}\,\varepsilon_t + \frac{1}{\sqrt{\frac{1}{\varepsilon_t}-1}}(1-\varepsilon_t) =$$

$$= \varepsilon_t\sqrt{\frac{1}{\varepsilon_t}-1} + \frac{1-\varepsilon_t}{\sqrt{\frac{1}{\varepsilon_t}-1}} = \sqrt{\varepsilon_t(1-\varepsilon_t)} + \sqrt{\varepsilon_t(1-\varepsilon_t)} = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$$

(***) $\sum_{i=1}^{m} D_i^t e^{\alpha_t}\mathbb{I}_{[N]}$

(**) $[N] = [y_i \neq h_t(x_i)]$

$[\bar N] = [y_i = h_t(x_i)]$

5. Prove that $\forall \eta > 0$ the perceptron algorithm will perform the same number of iterations, and will converge to a vector that points to the same direction.

first, we'll show that the perceptron converges to a vector that points in the same direction

let $\eta_1, \eta_2 > 0$. if the data is linearly sepperable, the algorithm won't converge for either of the η's otherwise it'll converge after $n \in \mathbb{N}$ steps; $w_n = w_{n-1} + \eta y_{i_n} x_{i_n} = \dots = w_0 + \eta y_{i_1} x_{i_1} + \dots + \eta y_{i_n} x_{i_n} = \eta\sum_{k=1}^{n} y_{i_k} x_{i_k}$

where i_k are the indexes on which the preceptron made a mistake in the labeling

we see that for each η we'll get the same answer except the coefficient η

i.e: $\eta_1\sum_{k=1}^{n} y_{i_k} x_{i_k}$, $\eta_2\sum_{k=1}^{n} y_{i_k} x_{i_k}$ we can see that η only scales the sum (it's a scalar), therefore it converges to a vector that points in the same direction

secondly, we'll show that the preceptron algorithm performs the same number of iterations

$$\hat{y}_j = \text{Sign}(\langle w, x_j\rangle) \overset{\forall i \in n}{=} \text{Sign}\left(\left\langle \eta\sum^{j} y_{i_k} x_{i_k}, x_j\right\rangle\right) \overset{\eta>0}{=} \text{Sing}\left(\left\langle \sum_{k=1}^{j} y_{i_k} x_{i_k}, x_j\right\rangle\right)$$

therefore, we get the same missclassifications, no matter the η => we'll have the same number of iterations