Introduction to Machine Learning Course

Short HW4 – Optimization, Regression, and Boosting

Submitted <u>individually</u> by Wednesday, 21.08.24, at 23:59.

You may answer in Hebrew or English and write on a computer or by hand (but be clear).

Please submit a PDF file named like your ID number, e.g., 123456789.pdf.

Bonus (maximal grade is 100): Writing on a computer (using LyX/LaTeX, Word + Equation tool, etc.) = 2 pts.

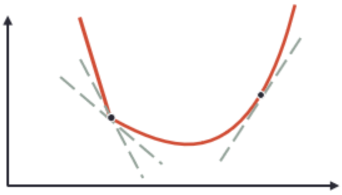
חיילים בשירות מילואים ממושך המעוניינים לקבל פטור מאחת השאלות במטלה (לבחירתנו) מוזמנים לפנות למייל הקורס.

Part A – Optimization

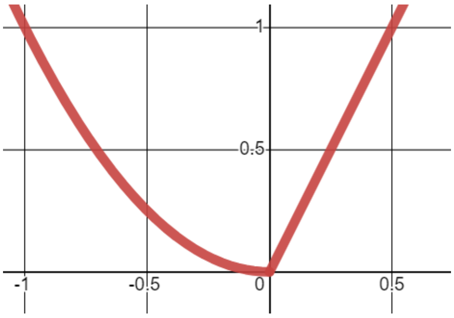
As we saw in Tutorial 08, subgradients generalize gradients to convex functions which are not necessarily differentiable. Notice: you can solve this exercise even before watching Tutorial 08.

Definition: the set of subgradients of $f: V \to \mathbb{R}$ at point $\boldsymbol{u} \in V$ is:

$$\partial f(\boldsymbol{u}) \triangleq \{\boldsymbol{q} \in V \mid \forall \boldsymbol{v} \in V: f(\boldsymbol{v}) \geq f(\boldsymbol{u}) + \boldsymbol{q}^\top(\boldsymbol{v} - \boldsymbol{u})\}.$$



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.

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

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

Part B – Regression

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

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

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.

2.3. 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).

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

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).

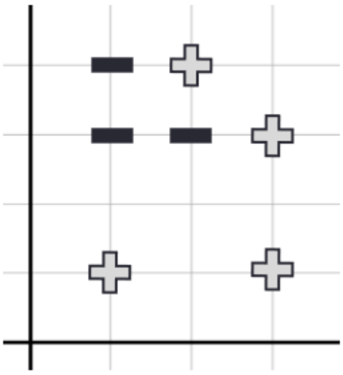
2.6. As it turns out, $\mathcal{L}_\lambda(w)$ would never prefer the simple quadratic polynomial over the cubic polynomial we found, no matter the value of $\lambda > 0$. Can you explain why?

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

Part C – Boosting

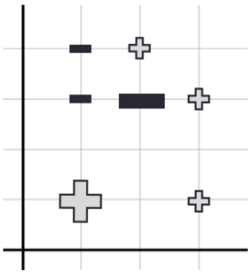
3. Given the following data with binary labels (“+”, “-“).

   We run AdaBoost with Decision stumps as weak classifiers.
   The sizes of the shapes in the figures indicate the probabilities that the algorithm assigns to each sample (high probability = large shape). Initially, the algorithm starts from a uniform distribution.

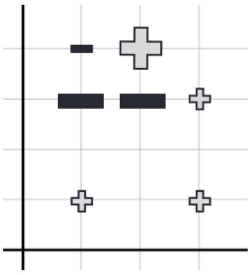


   Only some of the following figures depict possible distributions that can be obtained after <u>one</u> iteration of AdaBoost. **Which ones?** For each such distribution, propose a weak classifier that can lead to its figure (use a <u>clear</u> drawing or a short description of that classifier).

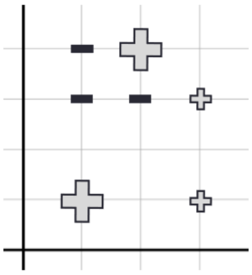
(a)



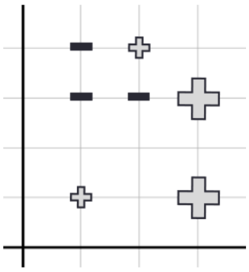
(b)



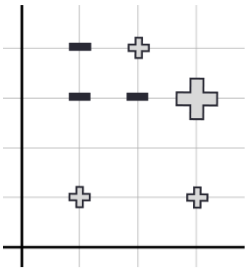
(c)



(d)



(e)



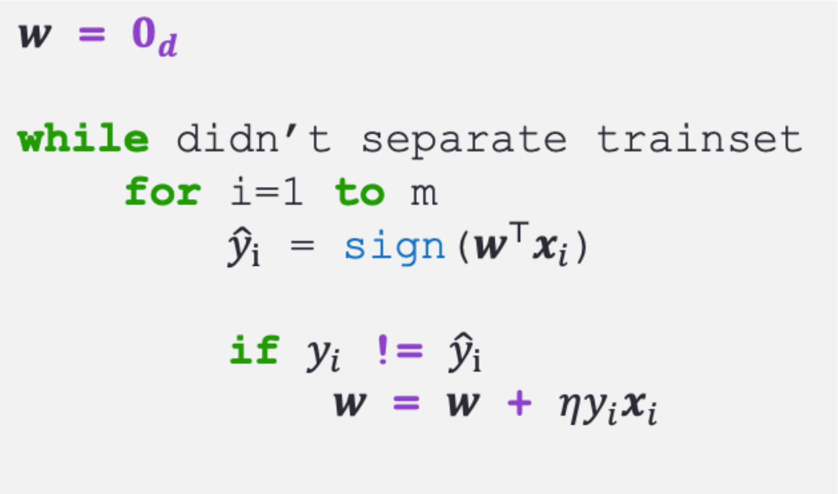
4. We have informally argued that the AdaBoost algorithm uses the weighting mechanism to “force” the weak learner to focus on the problematic examples in the next iteration. In this question we will find some rigorous justification for this argument.

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

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

Part D – Perceptron

5. Recall the vanilla perceptron algorithm: For an input trainset $(x_1, y_1), \dots, (x_m, y_m)$



```
$\boldsymbol{w} = 0_d$

while didn't separate trainset
    for i=1 to m
        $\hat{y}_\mathrm{i} = \mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x}_i)$

        if $y_i \;!=\; \hat{y}_\mathrm{i}$
            $\boldsymbol{w} = \boldsymbol{w} + \eta y_i \boldsymbol{x}_i$
```

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.