CS 446 / ECE 449 — Homework 5

acard6

Version 1.0

Instructions.

- Homework is due **Tuesday, April 18, at noon CST**; no late homework accepted.
- Everyone must submit individually on Gradescope under `hw5` and `hw5code`.
- The "written" submission at `hw5` **must be typed**, and submitted in any format Gradescope accepts (to be safe, submit a PDF). You may use LaTeX, Markdown, Google Docs, MS Word, whatever you like; but it must be typed!
- When submitting at `hw5`, Gradescope will ask you to select pages for each problem; please do this precisely!
- Please make sure your NetID is clear and large on the first page of the homework.
- Your solution **must** be written in your own words. Please see the course webpage for full academic integrity information. Briefly, you may have high-level discussions with at most 3 classmates, whose NetIDs you should place on the first page of your solutions, and you should cite any external reference you use; despite all this, your solution must be written in your own words.
- We reserve the right to reduce the auto-graded score for `hw5code` if we detect funny business (e.g., your solution lacks any algorithm and hard-codes answers you obtained from someone else, or simply via trial-and-error with the autograder).
- Coding problems come with suggested "library routines"; we include these to reduce your time fishing around APIs, but you are free to use other APIs.
- When submitting to `hw5code`, only upload the two python files `hw5.py` and `hw5_utils.py`. Don't upload a zip file or additional files.

Version history.

1.0. Initial version.

1. SVM with Biases.

This problem is about SVMs over $\mathbb{R}^d$ with linearly separable data (i.e., the hard margin SVM).

Our formulation of SVM required separators to pass through the origin, which does not provide a geometrically pleasing notion of maximum margin direction.

A first fix is provided by lecture 18: by appending a 1 to the inputs, we obtain the convex program

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \frac{1}{2}\|\boldsymbol{u}\|^2 \\ \text{subject to} \quad & \boldsymbol{u} \in \mathbb{R}^{d+1} \\ & y_i \left[\begin{smallmatrix} \boldsymbol{x}_i \\ 1 \end{smallmatrix}\right]^\top \boldsymbol{u} \geq 1 \qquad \forall i, \end{aligned}$$

and let $\bar{\boldsymbol{u}}$ denote the optimal solution to this program.

A second standard fix is to incorporate the bias directly into the optimization problem:

$$\begin{aligned} \min_{\boldsymbol{v},b} \quad & \frac{1}{2}\|\boldsymbol{v}\|^2 \\ \text{subject to} \quad & \boldsymbol{v} \in \mathbb{R}^d, b \in \mathbb{R} \\ & y_i(\boldsymbol{v}^\top \boldsymbol{x}_i + b) \geq 1 \qquad \forall i, \end{aligned}$$

and let $(\bar{\boldsymbol{v}}, \bar{b}) \in \mathbb{R}^d \times \mathbb{R}$ denote an optimal solution to this program. This second version is standard, but we do not use it in lecture for various reasons.

(a) In lecture, we stated that the first formulation is a *convex program* (formally defined in lecture 19). Show that the second formulation is also a convex program.

(b) Suppose there is only one datapoint: $\boldsymbol{x}_1 = \boldsymbol{e}_1$, the first standard basis vector, with label $y_1 = +1$. The first formulation will have a unique solution $\bar{\boldsymbol{u}}$, as discussed in lecture. Show that the second formulation does not have a unique solution.

(c) Let's add another datapoint: $\boldsymbol{x}_2 = -a\boldsymbol{e}_1$ for some $a \geq 3$, with label $y_2 = -1$. Now that we have two data points, both of the convex programs now have two constraints. Write out the explicit constraints to the first convex program.

(d) The optimal solution to the first convex program is $\bar{\boldsymbol{u}} = \frac{1}{2}e_1 + \frac{1}{2}e_{d+1}$, and the optimal solution to the second convex program is $(\bar{\boldsymbol{v}}, \bar{b}) = (\frac{2}{a+1}e_1, 1 - \frac{2}{a+1})$ (you do **not** need to prove this). Using this, determine and formally prove the limiting behavior of $\lim_{a\to\infty} \frac{1}{2}\|\bar{\boldsymbol{u}}_a\|^2$ and $\lim_{a\to\infty} \frac{1}{2}\|\bar{\boldsymbol{v}}_a\|^2$.

(e) Between the two versions of SVM with bias, which do you prefer? Any answer which contains at least one complete sentence will receive full credit.

Remark: Initially it may have seemed that both optimization problems have the same solutions; the purpose of this problem was to highlight that small differences in machine learning methods can lead to observably different performance.

Solution.

(a) we can prove that the second formulation is convex by 2nd order definition. Second order defn: $\nabla^2 f(x) \geq 0,\ \forall x, x' \to \frac{d^2}{dv^2}\frac{1}{2}\|\boldsymbol{v}\|^2 = 1 \geq 0$
As for the constraints, since it is a linear inequation and all linear equations are considered convex then with the function and its constraints being convex thus proves that this formulation is aconvex program

(b) Given that x_i and $\boldsymbol{v}$ are $dx1$ dimensions, where $x_1 = e_1$ and $y_1 = 1$ means that $\boldsymbol{v}^\top + b \geq 1$. taking the $min\frac{1}{2}\|\boldsymbol{v}\|^2$ for $\boldsymbol{v}$ we get that $\boldsymbol{v} = 1$, now we optimize $min\frac{1}{2}$ for b and see that the only constraint we have is that b ≥ 0 as an optimal solution

(c) pluggin in the values for i=1,2 we see that the explicit constraints of the first program are $\left[\begin{smallmatrix} e_1 \\ 1 \end{smallmatrix}\right]^\top \geq 1$ and $-\left[\begin{smallmatrix} ae_1 \\ 1 \end{smallmatrix}\right]^\top \geq 1$. We see that the constraint can be written as $\boldsymbol{u} \in R^{d+1}$, $u_1 + u_{d+1} \geq 1, \& -au_1 + u_{d+1} \geq 1$

(d)

(e) The first formulation would be more preferable since it would work better and more efficient with larger input datasets

2. SVM Implementation.

Recall that the dual problem of an SVM is

$$\max_{\boldsymbol{\alpha}\in\mathcal{C}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j),$$

where the domain $\mathcal{C} = [0,\infty)^n = \{\boldsymbol{\alpha} : \alpha_i \geq 0\}$ for a hard-margin SVM, and $\mathcal{C} = [0,C]^n = \{\boldsymbol{\alpha} : 0 \leq \alpha_i \leq C\}$ for a soft-margin SVM. Equivalently, we can frame this as the minimization problem

$$\min_{\boldsymbol{\alpha}\in\mathcal{C}} f(\boldsymbol{\alpha}) := \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i.$$

This can be solved by projected gradient descent, which starts from some $\boldsymbol{\alpha}_0 \in \mathcal{C}$ (e.g., $\mathbf{0}$) and updates via

$$\boldsymbol{\alpha}_{t+1} = \Pi_{\mathcal{C}}\left[\boldsymbol{\alpha}_t - \eta\nabla f(\boldsymbol{\alpha}_t)\right],$$

where $\Pi_{\mathcal{C}}[\boldsymbol{\alpha}]$ is the *projection* of $\boldsymbol{\alpha}$ onto $\mathcal{C}$, defined as the closest point to $\boldsymbol{\alpha}$ in $\mathcal{C}$:

$$\Pi_{\mathcal{C}}[\boldsymbol{\alpha}] := \arg\min_{\boldsymbol{\alpha}'\in\mathcal{C}} \|\boldsymbol{\alpha}' - \boldsymbol{\alpha}\|_2.$$

If $\mathcal{C}$ is convex, the projection is uniquely defined.

(a) Prove that

$$\left(\Pi_{[0,\infty)^n}[\boldsymbol{\alpha}]\right)_i = \max\{\alpha_i, 0\},$$

and

$$\left(\Pi_{[0,C]^n}[\boldsymbol{\alpha}]\right)_i = \min\{\max\{0, \alpha_i\}, C\}.$$

Hint: Show that the ith component of any other $\boldsymbol{\alpha}' \in \mathcal{C}$ is further from the ith component of $\boldsymbol{\alpha}$ than the ith component of the projection is. Specifically, show that $\left|\alpha'_i - \alpha_i\right| \geq \left|\max\{0, \alpha_i\} - \alpha_i\right|$ for $\boldsymbol{\alpha}' \in [0,\infty)^n$ and that $\left|\alpha'_i - \alpha_i\right| \geq \left|\min\left\{\max\{0,\alpha_i\}, C\right\} - \alpha_i\right|$ for $\boldsymbol{\alpha}' \in [0,C]^n$.

(b) Implement an `svm_solver()`, using projected gradient descent formulated as above. Initialize your $\boldsymbol{\alpha}$ to zeros. See the docstrings in `hw5.py` for details.

Remark: Consider using the `.backward()` function in pytorch. However, then you may have to use in-place operations like `clamp_()`, otherwise the gradient information is destroyed.

Library routines: `torch.outer, torch.clamp, torch.autograd.backward, torch.tensor(..., requires_grad=True), with torch.no_grad():, torch.tensor.grad.zero_, torch.tensor.detach.`

(c) Implement an `svm_predictor()`, using an optimal dual solution, the training set, and an input. See the docstrings in `hw5.py` for details.

Library routines: `torch.empty.`

(d) On the area $[-5,5]\times[-5,5]$, plot the contour lines of the following kernel SVMs, trained on the XOR data. Different kernels and the XOR data are provided in `hw5_utils.py`. Learning rate 0.1 and 10000 steps should be enough. To draw the contour lines, you can use `hw5_utils.svm_contour()`.

- The polynomial kernel with degree 2.
- The RBF kernel with $\sigma = 1$.
- The RBF kernel with $\sigma = 2$.
- The RBF kernel with $\sigma = 4$.

Include these four plots in your written submission.

Solution.

(a) Let α' be the projection of α onto $[0, \infty]^n$, and let β be any other point in $[0, \infty]^n$.
Since the projection onto a convex set is unique, we can assume without loss of generality that α' and β have the same sign pattern as α. Then, we have: $||\alpha - \alpha'||^2 = ||\alpha||^2 + ||\alpha'||^2 - 2(\alpha * \alpha')$
$= ||\alpha||^2 + ||\alpha - \Pi_{[0,\infty]^n}[\alpha]||^2$ (by definition of projection)
$= \sum_{i=1}^n \alpha_i^2 + || \max(-\alpha, 0)||^2$, (since $\Pi_{[0,\infty]^n}[\alpha]$ has non-negative values)
$= \sum_{i=1}^n \max(\alpha_i, 0)^2 <= \sum_{i=1}^n \max(\alpha_i, 0)^2 + \sum_{i=1}^n \max(-\alpha_i, 0)^2$
$= \sum_{i=1}^n \alpha_i^2 + \sum_{i=1}^n (-\alpha_i)^2 = ||\alpha||^2 + || - \alpha||^2 = ||\alpha - \beta||^2$
Therefore, we have shown that $||\alpha - \alpha'|| <= ||\alpha - \beta||,$ and so $(\Pi_{[0,\infty]^n}[\alpha])_i = \max(\alpha_i, 0)$.
The same process can be followed for $\Pi_{[0,C]^n}[\alpha]$, after the definition of projection we change the plane to be $[0, C]^n$ and get as follows
$\sum_{i=1}^n \alpha_i^2 + || \min(\max(-\alpha, 0), C)||^2$(, since $\Pi_{[0,C]^n}[\alpha]$ has non-negative values no greater than C)
$= \sum_{i=1}^n \min(\max(\alpha_i, 0), C)^2$

(d)

3. Logistic Regression.

Recall the empirical risk $\widehat{\mathcal{R}}$ for logistic regression (as presented in lecture 17):

$$\widehat{\mathcal{R}}_{\log}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \ln(1+\exp(-y_i \boldsymbol{w}^\top \boldsymbol{x}_i)).$$

Here you will minimize this risk using gradient descent.

(a) In your written submission, derive the gradient descent update rule for this empirical risk by taking the gradient. Write your answer in terms of the learning rate η, previous parameters $\boldsymbol{w}$, new parameters $\boldsymbol{w}'$, number of examples n, and training examples $(\boldsymbol{x}_i, y_i)$. Show all of your steps.

(b) Implement the `logistic()` function in `hw5.py`. You are given as input a training set `X`, training labels `Y`, a learning rate `lrate`, and number of gradient updates `num_iter`. Implement gradient descent to find parameters $\boldsymbol{w}$ that minimize the empirical risk $\widehat{\mathcal{R}}_{\log}(\boldsymbol{w})$. Perform gradient descent for `num_iter` updates with a learning rate of `lrate`, initializing $\boldsymbol{w} = 0$ and returning $\boldsymbol{w}$ as output. Don't forget to prepend `X` with a column of ones.

Library routines: `torch.matmul (@), torch.tensor.t, torch.exp.`

(c) Implement the `logistic_vs_ols()` function in `hw5.py`. Use `utils.load_logistic_data()` to generate a training set `X` and training labels `Y`. Run `logistic(X,Y)` from part (b) taking `X` and `Y` as input to obtain parameters $\boldsymbol{w}$. Also run `linear_normal(X,Y)` from Homework 4 Problem 2 to obtain parameters $\boldsymbol{w}$. Plot the decision boundaries for your logistic regression and least squares models along with the data `X`. Which model appears to classify the data better? Explain why you believe your choice is the better classifier for this problem.

Library routines: `torch.linspace, plt.scatter, plt.plot, plt.show, plt.gcf.`

Hints:

- The positive and negative points are guaranteed to be linearly separable (though an algorithm may or may not find the optimal line to separate them).
- The "decision boundary" in the problem description refers to the set of points $\boldsymbol{x}$ such that $\boldsymbol{w}^\top \boldsymbol{x} = 0$ for the chosen predictor. In this case, it suffices to plot the corresponding line.
- In order to make the two models significantly different, we recommend that you train the logistic regression with a large `num_iter` (e.g., 1,000,000 or even larger).

Solution.

(a) $\nabla\widehat{\mathcal{R}}_{log}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \frac{-y_i x_i * exp(-y_i w^\top x_i)}{1+exp(-y_i w_i)}$

$w' = w - \eta * \frac{1}{N}\sum_{i=1}^{N} \frac{-y_i x_i * exp(-y_i w^\top x_i)}{1+exp(-y_i w_i)}$

(c) I believe that linear seperation classifies the data better according to my visualization below as it seems the one outlier in the data heavily skewed the decision boundary when using the weights predicted by the logistic optimization whereas this was not as present using weights from linear grad.

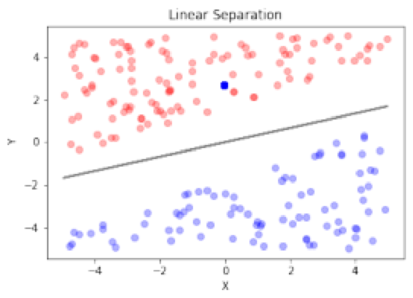

figure 1: linear seperation

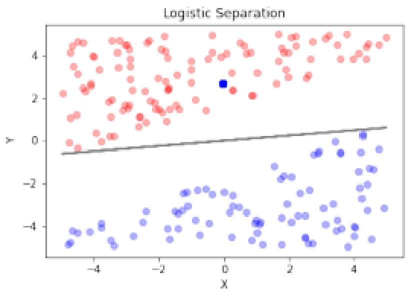

figure 2: logistic seperation

4. Robustness of the Majority Vote Classifier.

The purpose of this problem is to further investigate the behavior of the majority vote classifier using Hoeffding's inequality (*this will be covered in the lecture 22, the Ensemble Methods lecture*). Simplified versions of Hoeffding's inequality are as follows.

Theorem 1. *Given independent random variables* $(Z_1, \ldots, Z_k)$ *with* $Z_i \in [0, 1]$,

$$\Pr\left[\sum_{i=1}^{k} Z_i \geq \sum_{i=1}^{k} \mathbb{E}[Z_i] + k\epsilon\right] \leq \exp\left(-2k\epsilon^2\right), \tag{1}$$

and

$$\Pr\left[\sum_{i=1}^{k} Z_i \leq \sum_{i=1}^{k} \mathbb{E}[Z_i] - k\epsilon\right] \leq \exp\left(-2k\epsilon^2\right). \tag{2}$$

In this problem we have an odd number n of classifiers $\{f_1, \ldots, f_n\}$ and only consider their behavior on a fixed data example $(\boldsymbol{x}, y)$; by classifier we mean $f_i(\boldsymbol{x}) \in \{\pm 1\}$. Define the majority vote classifer $\textsc{Maj}$ as

$$\textsc{Maj}(\boldsymbol{x}; f_1, \ldots, f_n) := 2 \cdot \mathbb{1}\left[\sum_{i=1}^{n} f_i(\boldsymbol{x}) \geq 0\right] - 1 = \begin{cases} +1 & \sum_{i=1}^{n} f_i(\boldsymbol{x}) > 0, \\ -1 & \sum_{i=1}^{n} f_i(\boldsymbol{x}) < 0, \end{cases}$$

where we will not need to worry about ties since n is odd.

To demonstrate the utility of Theorem 1 in analyzing $\textsc{Maj}$, suppose that $\Pr[f_i(\boldsymbol{x}) = y] = p > 1/2$ independently for each i. Then, by defining a random variable $Z_i := \mathbb{1}[f_i(\boldsymbol{x}) \neq y]$ and noting $\mathbb{E}[Z_i] = 1 - p$,

$$\begin{aligned}
\Pr[\textsc{Maj}(\boldsymbol{x}; f_1, \ldots, f_n) \neq y] &= \Pr\left[\sum_{i=1}^{n} \mathbb{1}[f_i(\boldsymbol{x}) \neq y] \geq \frac{n}{2}\right] \\
&= \Pr\left[\sum_{i=1}^{n} Z_i \geq n(1-p) - \frac{n}{2} + np\right] \\
&= \Pr\left[\sum_{i=1}^{n} Z_i \geq n\mathbb{E}[Z_1] + n(p - 1/2)\right] \\
&\leq \exp\left(-2n(p - 1/2)^2\right).
\end{aligned}$$

The purpose of this problem is to study the behavior of $\textsc{Maj}(\boldsymbol{x})$ when not all of the classifiers $\{f_1, \ldots, f_n\}$ are independent.

(a) Assume n is divisible by 7 and $5n/7$ is odd, and that of the n classifiers $\{f_1, \ldots, f_n\}$, now only the first $5n/7$ of them (i.e., $\{f_1, \ldots, f_{5n/7}\}$) have independent errors on $\boldsymbol{x}$. Specifically, $\Pr[f_i(\boldsymbol{x}) = y] = p := 4/5$ for classifiers $\{f_1, \ldots, f_{5n/7}\}$. By contrast, we make no assumption on the other $2n/7$ classifiers (i.e., $\{f_{5n/7+1}, \ldots, f_n\}$) and their errors. Now use Hoeffding's inequality to show that

$$\Pr\left[\sum_{i=1}^{5n/7} \mathbb{1}[f_i(\boldsymbol{x}) \neq y] \geq \frac{3n}{14}\right] \leq \exp\left(-\frac{n}{70}\right).$$

(b) Continuing from (a), further show that the majority vote classifier over all n classifiers is still good, specifically showing

$$\Pr\left[\textsc{Maj}(\boldsymbol{x}; f_1, \ldots, f_n) \neq y\right] \leq \exp\left(-\frac{n}{70}\right).$$

For full points: You need to derive the inequality $\Pr\left[\textsc{Maj}(\boldsymbol{x}; f_1, \ldots, f_n) \neq y\right] \leq \exp(-n/70)$ rigorously for ANY possible behavior of the $\frac{2n}{7}$ arbitrary classifiers.

(c) Is the probability of correctly classifying $\boldsymbol{x}$ reasonably good in part (b) for large n? Do you have any interesting observations? Any answer which contains at least one complete sentence will receive full credit.

Solution.

(a) let $Z_i := \mathbb{1}[f_i(x) \neq y]$, then we see that $Pr[\sum_{i=1}^{5n/7} \mathbb{1}[f_i(x) \neq y] \geq \frac{3n}{14}]$
$= Pr[\sum_{i=1}^{5n/7} Z_i \geq 5n/7(1-p) - 3n/14 + 5np/7] = Pr[\sum_{i=1}^{5n/5} Z_i \geq 5/7n\mathbb{E}[Z_1] + n(5p/7 - 3/14)]$
$\leq \exp(-2 * 5/7n(5/7p - 3/14)^2) = \exp(-\frac{n}{70})$

(b) $MAJ(x; f_1, ..., f_n) = (\sum_{i=1}^{n} fi(x))$. Let E_i denote the event that $f_i(x) \neq y$, and let E_i^c denote the complement of E_i. Thus we have $Pr[M(x; f_1, ..., f_n) \neq y] \leq Pr[\sum_{i=1}^{n} \mathbb{1}[E_i] > n/2] + Pr[\sum_{i=1}^{n} 1[E_i] = n/2]$ By the union bound, we have $Pr[\sum_{i=1}^{n} \mathbb{1}[E_i] > n/2] \leq \sum_{i=1}^{n} Pr[E_i] = n(1-p)^{2n/7}$ Similarly, we have $Pr[\sum_{i=1}^{n} 1[E_i] = n/2] \leq \sum_{i=1}^{n} Pr[E_i, E_i^c] \leq \sum_{i=1}^{n} Pr[E_i^c] = n/7$ Combining the above inequalities, we have $Pr[M(x; f1, ..., fn) \neq y] \leq n(1-p)^{2n/7} + n/7$ Substituting $p = 4/5$ and simplifying the above expression, we get $Pr[M(x; f1, ..., fn) \neq y] \leq \exp(-n/70)$

(c) The probability of correctly classifying x using the majority vote classifier is very high for large n because the probability of making an error decreases exponentially with n due to the negative exponential