CS 189 Introduction to Machine Learning
Fall 2017 HW0

This homework is due **Friday, August 25 at 10 p.m.**. Note that every homework after will be due at **noon** instead.

1 Getting Started

You may typeset your homework in latex or submit neatly handwritten and scanned solutions. Please make sure to start each question on a new page, as grading (with Gradescope) is much easier that way! Deliverables:

1. Submit a PDF of your writeup to assignment on Gradescope, "HW[n] Write-Up"
2. Submit your test set evaluation results, "HW[n] Test Set".

After you've submitted your homework, be sure to watch out for the self-grade form.

(a) Before you start your homework, write down your team. Who else did you work with on this homework? List names and email addresses. In case of course events, just describe the group. How did you work on this homework? Any comments about the homework?

(b) Please copy the following statement and sign next to it:

I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up.

2 Sample Submission

Please submit a plain text file to the Gradescope programming assignment "Homework 0 Test Set":

1. Containing 5 rows, where each row has only one value "1".
2. No spaces or miscellaneous characters.
3. Name it "submission.txt".

3 Eigendecomposition Review

Compute eigenvectors and eigenvalues for the following matrix. Show your work.

$$\begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix}$$

Solution: Recall that we consider the matrix $M = \begin{bmatrix} 1-\lambda & 3 \\ 3 & 1-\lambda \end{bmatrix}$

Compute the characteristic polynomial for M and find its roots.

$$\begin{aligned} (1-\lambda)^2 - 9 &= 0 \\ 1 - 2\lambda + \lambda^2 - 9 &= 0 \\ -8 - 2\lambda + \lambda^2 &= 0 \\ (\lambda - 4)(\lambda + 2) &= 0 \\ \lambda &= 4, -2 \end{aligned}$$

Now, compute eigenvectors for each eigenvalue, by plugging in λ for M.

$$\lambda = 4 : \begin{bmatrix} -3 & 3 \\ 3 & -3 \end{bmatrix} \sim \begin{bmatrix} -3 & 3 \\ 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\lambda = -2 : \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix} \sim \begin{bmatrix} 3 & 3 \\ 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} -1 & 1 \end{bmatrix}$$

4 Linear Regression and Adversarial Noise

Suppose we have training data consisting of n points (x_i, y_i), which we have modeled as coming from $y_i = ax_i + b$. All x_i are distinct. We will do standard linear ordinary least-squares regression on the data to recover estimates for a and b. Say that y_i are actually coming from $y_i = ax_i + b + \varepsilon_i$, for some unknown disturbance process ε_i. The ε_i are initially 0, before the ε_i is modified by the adversary.

(a) Can an adversary force the linear regression to recover any desired a, b (different from the original line) by setting exactly 1 of the ε_i to be a selected non-zero value?

Solution: No. Intuitively, the adversary is only in control of one degree of freedom. For a more concrete explanation: take a counterexample, with points $p_1 : (0,0), p_2 : (1,1)$. Given control of either point, the adversary could never attain $a = 1, b = 2$. Changing ε_2 is insufficient, since the y-intercept is still $b = 0$. So, the adversary must change ε_1. However, setting $\varepsilon_1 = 2$ so that $b = 0$ implies $a = -1$. Thus, $a = 1, b = 2$ is not attainable in this setup.

(b) What if the adversary sets 2 of the ε_i to be non-zero values?

Solution: Yes. Intuitively, the adversary now has control over two degrees of freedom. See part 3 for a more concrete explanation.

(c) How many does the adversary need to change and how would it do it?

Solution:

Note: An intuitive explanation citing two degrees of freedom or somehow referencing a fully determined system of equations is sufficient for full credit.

The adversary needs control of two points that have different x coordinates. Consider n points $\{(x_i, y_i)\}_{i=1}^n$, where linear regression recovers $\hat{a}, \hat{b}$. We can start by examining the closed form solution for least-squares, which the user will use. Without loss of generality, let's assume that the first two points are the ones that the adversary is going to want to corrupt (i.e. they have different x coordinates.). Let us construct the standard $X = \begin{bmatrix} x_1, 1 \\ x_2, 1 \\ \vdots, \vdots \\ x_n, 1 \end{bmatrix}$ so that we are approximately solving $X\vec{w} = \vec{y}$. At this point, it is immediately obvious that since the first two rows of X are linearly independent, that X has full rank 2. The solution of linear least-squares is:

$$\vec{w} = \begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix} = (X^T X)^{-1} X^T \hat{\vec{y}} = (X^T X)^{-1} X^T (\vec{y} + \vec{\varepsilon}) = (X^T X)^{-1} X^T \left(\begin{bmatrix} y_1 \\ y_2 \\ \vec{y}_{rest} \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vec{0} \end{bmatrix} \right)$$

Note that the adversary chooses $(\varepsilon_1, \varepsilon_2)$ and wants least squares to yield exactly the line the adversary desires. Call the adversary's desired line parameters $\vec{w}_* = \begin{bmatrix} a_* \\ b_* \end{bmatrix}$. Then, we just need to see that the adversary needs to solve the equation:

$$\begin{bmatrix} a_* \\ b_* \end{bmatrix} - (X^T X)^{-1} X^T \vec{y} = (X^T X)^{-1} X^T \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vec{0} \end{bmatrix} = (X^T X)^{-1} \begin{bmatrix} x_1, x_2 \\ 1, 1 \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix}.$$

But we know that since $x_1 \neq x_2$, the matrix $\begin{bmatrix} x_1, x_2 \\ 1, 1 \end{bmatrix}$ is invertible and so this has the solution:

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix} = \left(\begin{bmatrix} x_1, x_2 \\ 1, 1 \end{bmatrix} \right)^{-1} (X^T X) \left(\begin{bmatrix} a_* \\ b_* \end{bmatrix} - (X^T X)^{-1} X^T \vec{y} \right).$$

This explicit formula lets the adversary move the least-square solution to wherever it wants assuming it has control over the two "outliers" in the first two positions.

(a) Adversary solves for ε_i in terms of a_*, b_*.

(b) Adversary applies ε_i and gives data to you.

(c) You run least squares using the ε_i and retrieve $\hat{a}, \hat{b}$, which we've determined to be equal to a_*, b_*.

If there aren't two distinct points with different x coordinates, that means all the training points have the same x coordinate. At this point, it is clear that ordinary least squares will simply fail since it will not have enough rank in X itself. The adversary can't remedy this failure since it can only perturb the y measurements, not the x ones.

Why is this important? We wanted to show you that, when subject to unbounded noise or outlier data points, ordinary linear least-squares regression can yield a line *very* far from the true model. In fact, least squares is in general not robust. Other methods can be made robust to outliers. In far, for those of you who have seen OMP (Orthogonal Matching Pursuit) in 16A, it's actually possible to modify OMP to help hunt for outliers and remove them from the data in many cases. We will be learning other robust techniques later in the course. Interestingly, like OMP, they are practically often built using least-squares as an ingredient.

5 Your Own Question

Write your own question, and provide a thorough solution.