CS 189 Introduction to Machine Learning
Fall 2017 HW3

This homework is due **Friday, September 15 at 12pm noon.**

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 all code needed to reproduce your results, "HW[n] Code".
3. 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.

Throughout the homework, note that
$Z \in \mathbb{R}$ implies that Z is a scalar;
$Z \in \mathbb{R}^d$ is equivalent to saying that we have dimensions $\underbrace{Z}_{d \times 1}$;
$Z \in \mathbb{R}^{n \times d}$ is equivalent to saying that Z is a matrix with dimensions $\underbrace{Z}_{n \times d}$.
When we say that $Z, Z' \in \mathbb{R}$, we mean that $Z \in \mathbb{R}$ and $Z' \in \mathbb{R}$.

2 Probabilistic Model of Linear Regression

Both ordinary least squares and ridge regression have interpretations from a probabilistic standpoint. In particular, assuming a generative model for our data and a particular noise distribution, we will derive least squares and ridge regression as the maximum likelihood and maximum a-posteriori parameter estimates, respectively. This problem will walk you through a few steps to do that. (Along with some side digressions to make sure you get a better intuition for ML and MAP estimation.)

(a) Assume that X and Y are both one-dimensional random variables, i.e. $X, Y \in \mathbb{R}$. Assume an affine model between X and Y: $Y = Xw + b + Z$, where $w, b \in \mathbb{R}$ and $Z \sim N(0,1)$ is a standard normal (Gaussian) random variable. Assume w, b are not random. **What is the conditional distribution of Y given X?**

(b) Given n points of training data $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ generated in an iid fashion by the probabilistic setting in the previous part, **derive the maximum likelihood estimator for w, b from this training data.**

(c) Now, we are going to change the probabilistic model slightly in two ways. There is no more b and Z has a different distribution. So assume the linear model between X and Y is given by $Y = Xw + Z$, where $Z \sim U[-0.5, 0.5]$ is a continuous random variable uniformly distributed between -0.5 and 0.5. Assume w is not random. **What is the conditional distribution of Y given X?**

(d) Given n points of training data $\{(X_1, Y_1), (X_2, Y_2), \cdots, (X_n, Y_n)\}$ generated in an iid fashion in the setting of the previous part, **derive the maximum likelihood estimator of w.** (Note that this estimate may not be unique; you may report a set of values, or feel free to break ties in whatever fashion and report one value within this set.)

(e) Take the model $Y = Xw^* + b^* + Z$, where $Z \sim U[-0.5, 0.5]$ is a continuous random variable uniformly distributed between -0.5 and 0.5. **Use a computer to simulate n training samples $\{(X_1, Y_1), (X_2, Y_2), \cdots, (X_n, Y_n)\}$ and illustrate what the likelihood of the data looks like in the (w, b) model parameter space after $n = 5, 25, 125, 625$ training samples. Qualitatively describe what is happening as n gets large?**

(Note that you have considerable design freedom in this problem part. You get to choose how you draw the X_i as well as what true value (w^*, b^*) you want to illustrate. You have total freedom in using python libraries for this problem part. No restrictions.)

(f) (One-dimensional Ridge Regression) Now, let us return to the case of Gaussian noise. Given n points of training data $\{(X_1,Y_1),(X_2,Y_2),\cdots,(X_n,Y_n)\}$ generated according to $Y_i = X_i W + Z_i$, where $Z_i \sim N(0,1)$ are iid standard normal random variables. Assume $W \sim N(0,\sigma^2)$ is also a standard normal and is independent of both the Z_i and the X_i. **Use Bayes' Theorem to derive the posterior distribution of W given the training data. What is the mean of the posterior distribution of W given the data?**

(g) Consider n training data points $\{(X_1,Y_1),(X_2,Y_2),\cdots,(X_n,Y_n)\}$ generated according to $Y_i = w^T X_i + Z_i$ where $Y_i \in \mathbb{R}, w, X_i \in \mathbb{R}^d$ with w fixed, and $Z_i \sim N(0,1)$ iid standard normal random variables. **Derive why the maximum likelihood estimator for w is the solution to a least squares problem.**

(h) (Multi-dimensional ridge regression) Consider the setup of the previous part: $Y_i = W^T X_i + Z_i$, where $Y_i \in \mathbb{R}, W, X_i \in \mathbb{R}^d$, and $Z_i \sim N(0,1)$ iid standard normal random variables. Now, however, we have a prior for the vector W (now a random variable): $W_j \sim N(0,\sigma^2)$ are iid for $j = 1,2,\ldots,d$. **Derive the posterior distribution of W given all the X_i, Y_i pairs. What is the mean of the posterior distribution of the vector W?**

(i) Consider $d = 2$ and the setting of the previous part. **Use a computer to simulate and illustrate what the a-posteriori probability looks like for the W model parameter space after $n = 5, 25, 125$ training samples for different values of σ^2.**

(Note that you have considerable design freedom in this problem part. You have total freedom in using python libraries for this problem part. No restrictions. Either contour plots or 3d plots of the a-posteriori probability are fine. You don't have to label the contours with values if the story is clear from the plots visually.)

3 Simple Bias-Variance Tradeoff

Consider a random variable X, which has unknown mean μ and unknown variance σ^2. Given n iid realizations of training samples $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$ from the random variable, we wish to estimate the mean of X. We will call our estimate the random variable $\hat{X}$ with mean $\hat{\mu}$. There are a few ways we can estimate μ given the realizations of the n samples:

1. Average the n samples: $\frac{x_1+x_2+\ldots+x_n}{n}$.
2. Average the n samples and one sample of 0: $\frac{x_1+x_2+\ldots+x_n}{n+1}$.
3. Average the n samples and n_0 samples of 0: $\frac{x_1+x_2+\ldots+x_n}{n+n_0}$.
4. Ignore the samples: just return 0.

In the parts of this question, we will measure the *bias* and *variance* of each of our estimators. The *bias* is defined as

$$E[\hat{X} - \mu]$$

and the *variance* is defined as

$$\mathrm{Var}[\hat{X}].$$

(a) **What is the bias of each of the four estimators above?**

(b) **What is the variance of each of the four estimators above?**

(c) Hopefully you see that there is a trade-off. As we add more copies of 0 to our estimator, the bias increases and the variance decreases. It is a common mistake to assume that an unbiased estimator is always "best." Let's explore this a bit further. Define the *expected total error* as the sum of the variance and the square of the bias. **Compute the expected total error for each of the estimators above.**

(d) **Argue that all four estimators are really just special cases of the third estimator (with the hyperparameter n_0).**

(e) Say that $n_0 = \alpha n$. **Find the setting for α that would minimize the expected total error, assuming you secretly knew μ and σ.** Your answer will depend on σ, μ, and n.

(f) For this part, let's assume that we had some reason to believe that μ *should be small* (close to 0) and σ *should be large*. In this case, **what happens to the expression in the previous part?**

(g) In the previous part, we assumed there was reason to believe that μ *should be small*. Now let's assume that we have reason to believe that μ is not necessarily small, but *should be close to some fixed value* μ_0. **In terms of X and μ_0, how can we define a new random variable X' such that X' is expected to have a small mean?**

(h) Draw a connection between α in this problem and the regularization parameter λ in the ridge-regression version of least-squares. **What does this problem suggest about choosing a regularization coefficient and handling our data-sets so that regularization is most effective**? This is an open-ended question, so do not get too hung up on it.

4 Estimation in linear regression

In linear regression, we estimate a vector $y \in \mathbb{R}^n$ by using the columns of a feature matrix $A \in \mathbb{R}^{n \times d}$. Assume that the number of training samples $n \geq d$ and that A has full column rank. Let us now show how well we can estimate the underlying model as the number of training samples grows.

Assume that the true underlying model for our noisy training observations is given by $Y = \underbrace{Ax^*}_{y^*} + W$, with $W \in \mathbb{R}^n$ having iid $W_j \sim \mathcal{N}(0, \sigma^2)$ representing the random noise in the observation Y. Here, the $x^* \in \mathbb{R}^d$ is something arbitrary and not random. After obtaining $\hat{X} = \arg\min_x \|Y - Ax\|_2^2$, we would like to bound the error $\|A\hat{X} - y^*\|_2^2$, which is the prediction error incurred based on the specific n training samples we obtained.

Initially, it might seem that getting a good estimate of this prediction error is hopeless since the training data's A matrix is involved in some complicated way in the estimate. The point of this problem is to show you that this is not the case and we can actually understand this.

(a) Using the standard closed form solution to the ordinary least squares problem, **show that**

$$\|A\hat{X} - y^*\|_2^2 = \|A(A^\top A)^{-1}A^\top W\|_2^2.$$

(b) Given the *reduced* singular value decomposition $A = U\Sigma V^\top$ (with $U \in \mathbb{R}^{n\times d}$, $\Sigma \in \mathbb{R}^{d\times d}$, and $V \in \mathbb{R}^{d\times d}$. The reduced SVD just ignores all the singular vectors that correspond to directions that are in the nullspace of $A^\top$ since A is a tall skinny matrix.), **show that we have**

$$\|A\hat{X} - y^*\|_2^2 = \|U^\top W\|_2^2.$$

(Hint: Recall that the standard Euclidean ℓ_2 norm is unitarily invariant.)

(c) If W_0 is a vector with i.i.d standard Gaussians, recall that $a^\top W_0 \sim \mathcal{N}(0, \|a\|_2^2)$. **Use this fact to show that**

$$\frac{1}{n}\mathbb{E}\left[\|A\hat{X} - y^*\|_2^2\right] = \sigma^2\frac{d}{n}.$$

Notice that this kind of scaling is precisely how the average squared error for simple averaging scales when we have multiple iid samples of a random variable and we want to estimate its mean. This is why we often think about ordinary least squares as just being a kind of generalized averaging. However, the dimension of the parameters being estimated also matters.

(d) Let us now try to answer another basic question. Let us say we have a true linear model, but we try to estimate it using a polynomial of degree D. Clearly, this is overkill, but what do we lose?

Given scalar samples $\{x_i, Y_i\}_{i=1}^n$, let us say that the underlying model is a noisy linear model, i.e. $Y_i = mx_i + c + W_i$. Let us, however, perform polynomial regression: we form the matrix A by using $D+1$ polynomial features (including the constant) of the *distinct* sampling points $\{x_i\}_{i=1}^n$. **How many samples n do we need to ensure that our average squared error is bounded by ε?** Your answer should be expressed as a function of D, σ^2, and ε.

Conclude that as we increase model complexity, we require a proportionally larger number of samples for accurate prediction.

At this point, you are encouraged to take a fresh look at the discussion worksheet which is, in effect, a continuation of this problem to understand the ramifications of the bias/variance tradeoff in the context of ordinary least-squares regression.

(e) Try the above problem out by yourself, by setting $m = 1$, $c = 1$, and sample n points $\{x_i\}_{i=1}^n$ uniformly from the interval $[-1, 1]$. Generate $Y_i = mx_i + c + W_i$ with W_i representing standard Gaussian noise. **Fit a D degree polynomial to this data and show how the average error $\frac{1}{n}\|A\hat{X} - y^*\|_2^2$ scales as a function of both D and n.** You may show separate plots for the two scalings. It may also be helpful to average over multiple realizations of the noise (or to plot point clouds) so that you obtain smooth curves.

(For this part, any and all python libraries are allowed.)

5 Robotic Learning of Controls from Demonstrations and Images

Huey, a home robot, is learning to retrieve objects from a cupboard, as shown in Fig. 1. The goal is to push obstacle objects out of the way to expose a goal object. Huey's robot trainer, Anne, provides demonstrations via tele-operation.

During a demonstration, Huey records the RGB images of the scene for each timestep, $x_0, x_1, ..., x_T$, where $x_i \in \mathbb{R}^{30x30x3}$ and the controls for his mobile base, $u_0, u_1, \ldots, u_T$, where $u_i \in \mathbb{R}^3$. The controls correspond to making small changes in the 3D pose (i.e. translation and rotation) of his body. Examples of the data are shown in the figure.

Under an assumption (sometimes called the Markovian assumption) that all that matters for the current control is the current image, Huey can try to learn a linear *policy* π (where $\pi \in \mathbb{R}^{2700x3}$) which linearly maps image states to controls (i.e. $\pi^\top x = u$). We will now explore how Huey can recover this policy using linear regression. Note please use **numpy** and **numpy.linalg** to complete this assignment.

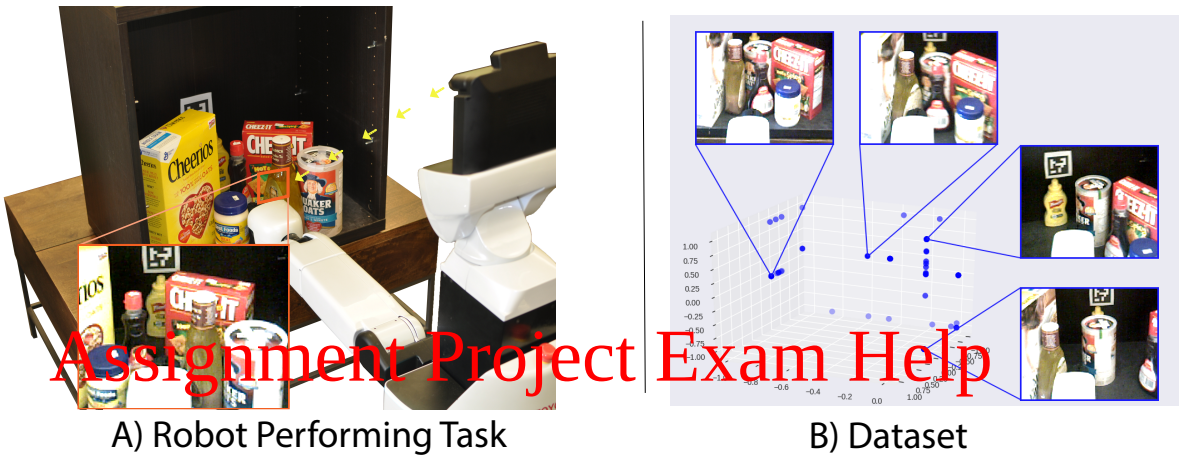

Figure 1: A) Huey trying to retrieve a mustard bottle. An example RGB image of the workspace taken from his head mounted camera is shown in the orange box. The angle of the view gives Huey and eye-in-hand perspective of the cupboard he is reaching into. B) A scatter plot of the 3D control vectors, or u labels. Notice that each coordinate of the labels lies within the range of $[-1, 1]$ for the change in position. Example images, or states x, are shown for some of the corresponding control points. The correspondence is indicated by the blue lines.

(a) Load the n training examples from x_train.p and compose the matrix X, where $X \in \mathbb{R}^{nx2700}$. Note, you will need to flatten the images to reduce them to a single vector. The flattened image vector will be denoted by $\bar{x}$ (where $\bar{x} \in \mathbb{R}^{2700x1}$). Next, load the n examples from y_train.p and compose the matrix U, where $U \in \mathbb{R}^{nx3}$. Try to perform ordinary least squares to solve:

$$\min_{\pi} ||X\pi - U||_2^F$$

to learn the *policy* $\pi \in \mathbb{R}^{2700\times 3}$. **Report what happens as you attempt to do this and explain why.**

(b) Now try to perform ridge regression:

$$\min_{\pi} ||X\pi - U||_2^2 + \lambda||\pi||_2^2$$

on the dataset for regularization values $\lambda = \{0.1, 1.0, 10, 100, 1000\}$. Measure the average squared Euclidean distance for the accuracy of the policy on the training data:

$$\frac{1}{n}\sum_{i=0}^{n-1} ||\bar{x}_i^T \pi - u_i||_2^2$$

Report the training error results for each value of λ.

(c) Next, we are going to try and improve the performance by standardizing the states. For each pixel value in each data point, x, perform the following operation:

$$x = \frac{x}{255} * 2 - 1.$$

Since we know the maximum pixel value is 255, this rescales the data to be between $[-1, 1]$. **Repeat the previous part and report the average squared training error for each value of λ.**

(d) To better understand how standardizing improved the loss function, we are going to evaluate the *condition number* k of the optimization, which is defined as

$$k = \frac{\sigma^2_{\max}(X^T X + \lambda I)}{\sigma^2_{\min}(X^T X + \lambda I)}$$

or the ratio of the maximum eigenvalue to the minimum eigenvalue of the relevant matrix. For the regularization value of $\lambda = 100$, **report the condition number with the standardization technique applied and without**.

(e) Finally, evaluate your learned *policy* on the new validation data x_test.p and y_test.p for the different values of λ. **Report the average squared Euclidean loss** and **qualitatively explain how changing the values of λ affects the performance in terms of bias and variance**.

6 Your Own Question

Write your own question, and provide a thorough solution.

Writing your own problems is a very important way to really learn material. The famous "Bloom's Taxonomy" that lists the levels of learning is: Remember, Understand, Apply, Analyze, Evaluate, and Create. Using what you know to create is the top-level. We rarely ask you any HW questions about the lowest level of straight-up remembering, expecting you to be able to do that yourself. (e.g. make yourself flashcards) But we don't want the same to be true about the highest level.

As a practical matter, having some practice at trying to create problems helps you study for exams much better than simply counting on solving existing practice problems. This is because thinking about how to create an interesting problem forces you to really look at the material from the perspective of those who are going to create the exams.

Besides, this is fun. If you want to make a boring problem, go ahead. That is your prerogative. But it is more fun to really engage with the material, discover something interesting, and then come up with a problem that walks others down a journey that lets them share your discovery. You don't have to achieve this every week. But unless you try every week, it probably won't happen ever.