EECS 189 Introduction to Machine Learning
Fall 2020 HW1

This homework is due **Tuesday, September 8 at 11:59 p.m.**

# 1 Getting Started

**Read through this page carefully.** You may typeset your homework in latex or submit neatly handwritten/scanned solutions. Please start each question on a new page. Deliverables:

1. Submit a PDF of your writeup, **with an appendix for your code**, to the appropriate assignment on Gradescope. If there are graphs, include those graphs in the correct sections. Do not simply reference your appendix.
2. If there is code, submit all code needed to reproduce your results.
3. If there is a test set, submit your test set evaluation results.

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

(a) Who else did you work with on this homework? 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. We just want to make it *extra* clear so that no one inadvertently cheats.

*I certify that all solutions are entirely in my words and that I have not looked at another student's solutions nor have I looked at any online solutions to any of these problems. I have credited all external sources in this write up.*

# 2 Study Group Icebreaker

Once you have received your group assignment, meet at least once with everyone in the group to introduce yourselves and discuss expectations for the semester. As an icebreaker, everyone should talk about what they want to get out of this machine learning class and why they decided to take it. **Your written answer for this problem should be a summary of the group's answers.** This meeting is an excellent opportunity for your group to set up times to work on the homeworks together and discuss how the members prefer to study for this course.

 

# 3 Exam Policy and Practice

Please read through the entirety of the EECS189 exam proctoring policy (click here) carefully before proceeding. This question is designed to familiarize you with some of the things you will have to do during the exam.

(a) After reading through the Exam Policy carefully, please answer the following questions.

(i) Given you experience no disruptions during the exam, how many total minutes do you have for scanning and submission? Does it matter if you are using a tablet or whether you are using paper to write your answers?

(ii) Are you required to record locally during the exam? How much space should you have available on your computer for a local recording?

(iii) How should you contact the course staff for an emergency situation during the exam?

(b) Please configure your Zoom link.

(i) Fill out the following Google Form (click here) to submit the Zoom link you will be using. You must use this Zoom link for this assignment as well as for the exams. This can easily be done by submitting your Personal Meeting Room link and setting your Personal Meeting ID as your default on all devices you will be using for the final.

(ii) Ensure that anyone can join your Zoom link and that there is no waiting room for your Zoom meeting. You can try to do this by entering the meeting on one device that is logged in to your official Berkeley Zoom account and then entering the meeting on another device that is logged into some other Zoom account. If you are able to do that, then your Zoom link is joinable. If you are not put into a waiting room, then your Zoom meeting will not have a waiting room. (You might want to have a member of your study group try this out with you if you don't already have two Zoom accounts.)

(c) You will now conduct a Zoom recording. You should use this recording to work through a homework problem or other study material to simulate the actual circumstances of the final exam.

(i) Start the Zoom call for the link you provided above. Turn on your microphone and recording device (webcam, phone camera). Turn off your speaker. Share your entire desktop (not just a particular window).

(ii) Start recording via Zoom. You may record locally or on the cloud (see the exam policy document for details).

(iii) Hold your CalID next to your face and record yourself saying your name into the webcam. Both your face and your entire CalID should be visible in the video. We should be able to read your name and SID. This step should take **at least** 3 seconds. See figure 2. *If you do not have a CalID for some reason, please hold up some document which has an image of you and proves your identity, such as a driver's license.*

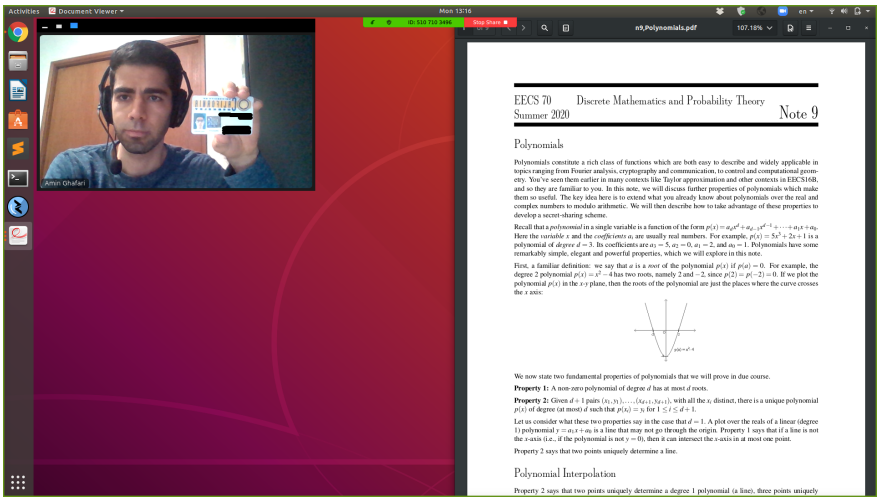


Figure 1: ID card demonstration. Do not actually black out your SID and name.

(iv) Turn your recording device (webcam, phone) around 360° **slowly** so that we can see your entire room clearly. There should be no uncovered screens anywhere in the room during your exam. Only admin TAs and instructors will be able to see your videos (both for this assignment and for the actual exams).

(v) Position your recording device in such a way that we can see your workspace and your hands. We don't care if we can see your face or not. If you are not using your phone to record your workspace, then it should be visible in the recording, face down. See figure 3. **Think during this test run. On the actual exam, you will want to use the desktop to see the exam itself. If you are using your laptop's built-in camera, the angle might be bad. In this case, think about how you might position the laptop or consider using your phone or an external webcam on a stand to record your workspace. We want you to iron out these details ahead of the actual exam so that the exam itself has no additional stress due to proctoring logistics.**

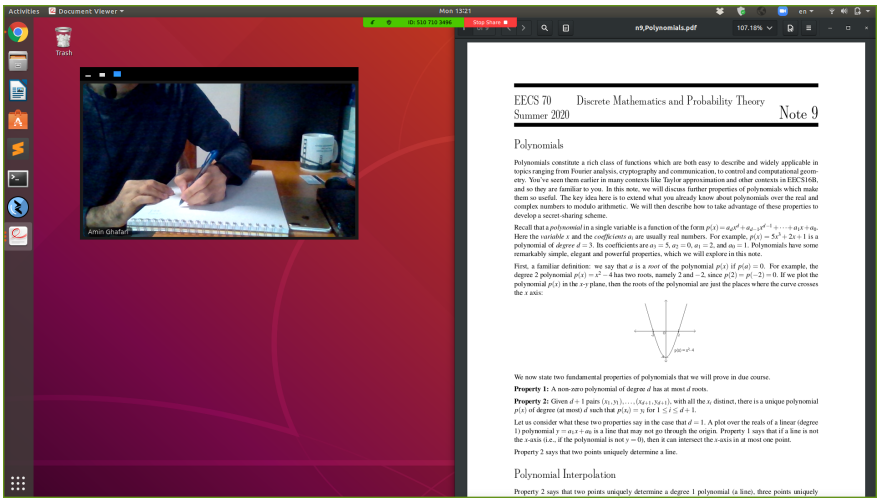


Figure 2: Demonstration of taking your exam. Your setup should look like this while you are taking the exam.

(vi) Your microphone should be on at all times. We should be able to see the time on your desktop at all times.

(vii) Record for a full two and a half hours. You should use this time to work through a homework problem or other study material for the course. The more realistic it is to actually taking an exam, the better practice it will be for you. (We also want to make sure that your computer can handle the video encoding task if your are doing a local recording.)

(viii) After two and a half hours, stop the recording. Check your recording to confirm that it contains your video as well as your desktop throughout its duration. Upload your video to Google drive and submit the link to the video using this Google Form (click here). You must make sure that the link sharing permissions are set so that we may view the video.

(d) A Midterm Google document should be shared with you with instructions for the midterm and the time which you will start taking the exam a few days before the midterm. More details will be made available closer to the exam date.

Link for policy:
https://docs.google.com/document/d/14qO3xRt724j8Fs12i9d4E-pMWuyqk4h8b5gBbnd3hVo/edit

Form to submit Zoom link:
https://forms.gle/P7CDYVpaqLBV4zez5

Form to submit video link:
https://forms.gle/F5V1J89NKf21Rj7Y6

# 4 Linear Regression and Adversarial Noise

In this question, we will investigate how the presence of noise in the data can adversely affect the model that we learn from it.

Suppose we obtain a training dataset consisting of $n$ points $(x_i, y_i)$ where $n \geq 2$. In case of no noise in the system, these set of points lie on a line given by $y = w[1]x + w[0]$, i.e, for each $i$, $y_i = w[1]x_i + w[0]$. The variable $x$ is commonly referred to as the covariate [1] and $y$'s are referred to as the observation. Suppose that all $x_i$'s are distinct and non-zero. Our task is to estimate the slope $w[1]$ and the $y$-intercept $w[0]$ from the training data. We call the pair $(w[1], w[0])$ as the true model.

Suppose that an adversary modifies our data by corrupting the observations and we now have the training data $(x_i, \tilde{y}_i)$ where $\tilde{y}_i = y_i + \epsilon_i$ and the noise $\epsilon_i$ is chosen by the adversary. Note that the adversary has access to the features $x_i$ but *can not* modify them. Its goal is to trick us into learning a wrong model $(\hat{w}[1], \hat{w}[0])$ from the dataset $\{(x_i, \tilde{y}_i), i = 1, \ldots, n\}$. We denote by $(\hat{w}[1], \hat{w}[0])$ the model that we learn from this dataset $\{(x_i, \tilde{y}_i), i = 1, \ldots, n\}$ using the standard ordinary least-squares regression.

[1] Besides covariate, some other names for $x$ include feature, regressor, predictor.

(a) Suppose that the adversary wants us to learn a particular wrong model $(w[1]^*, w[0]^*)$. If we use standard ordinary least-squares regression, **can the adversary *always* (for any choice of $w[1]^*$ and $w[0]^*$) fool us by setting a particular value for exactly one $\epsilon_i$ (and leaving other observations as it is, i.e., $\tilde{y}_j = y_j, j \neq i$), so that we obtain $\hat{w}[1] = w[1]^*$ and $\hat{w}[0] = w[0]^*$?** If yes, justify by providing a mathematical mechanism for the adversary to set the value of the noise term as a function of the dataset $\{(x_i, y_i), i = 1, \ldots, n\}$ and $(w[1]^*, w[0]^*)$? If no, provide a counter example.

(b) **Repeat part (a) for the case when the adversary can corrupt two observations, i.e., for the case when the adversary can set up at most two of the $\epsilon_i$'s to any non-zero values of its choice.**

(c) In the context of machine learning and applications, **what lessons do you take-away after working through this problem?**

# 5 Outlier Removal via OMP

*[This is another rerun problem from 16B that serves as a way to also force the review of a key approach from 16A, orthogonal matching pursuit. The ideas introduced in this problem are very important, as you will see later in this HW (and later HWs). ]*

The problem of "outliers" (bad data points) is ubiquitous in the real world of experimentally collected data. This problem is about how we can leverage known techniques (from 16A) to do something about them in a way that doesn't require a human to manually look at points and subjectively decide which ones are good or bad.

Suppose we have a system where we believe that vector-inputs $\mathbf{x}$ lead to scalar outputs in a linear way $\mathbf{p}^T\mathbf{x}$. However, the parameters $\mathbf{p}$ are unknown and must be learned from data. Our data collection process is imperfect and instead of directly seeing $\mathbf{p}^T\mathbf{x}$, we get observations $y = \mathbf{p}^T\mathbf{x} + \epsilon$ where the $\epsilon$ is some kind of disturbance or noise that nature introduces.

To allow us to learn the parameters, we have $n$ experimentally obtained data points: input-output pairs $(\mathbf{x}_i, y_i)$ where the index $i = 1, \ldots, n$. However, because we believe that our observed outputs might be a bit noisy, we only have an approximate system of equations. In particular, if we group the data points into a matrix and vector as follows: $X = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix}$ where clearly $X$ is a matrix that has $d$ columns and $n$ rows, and $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$ is an $n$-vector. Then we can express the approximate system of equations that we want to solve as $X\mathbf{p} \approx \mathbf{y}$.

The classic least-squares problem views the goal as minimizing

$$\|\mathbf{y} - X\mathbf{p}\|^2 = \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \mathbf{p})^2 \tag{1}$$

over the choice of $\mathbf{p}$ and thereby making the residual $\mathbf{y} - X\mathbf{p}$ have as small a Euclidean norm as possible. This is a reasonable thing to do when we believe that the individual residuals $y_i - \mathbf{x}_i^T \mathbf{p}$ are all small on average, meaning that we believe that the true disturbance vector $\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$ is small.

However, nature (or our observation process) isn't always this well behaved. When a small subset of observations don't line up well like the other data points, we call these *outliers*. For these, we don't always know what is wrong. It could be that there was a lot of disturbance or observation noise. It could also be that there was something wrong with how the experiment that we conducted and the $\mathbf{x}$ in reality was very different from what we think it was. Or it could simply be that the physical system was just behaving in a very unusual way that we cannot hope to predict or understand with our model.

An exaggerated example is when we have a set of observations that satisfy perfectly $y_i = \sum_{j=1}^{d} \mathbf{x}_i[j]$ with the $|y_i| < 1$ for all $i \neq c$, but there is one crazy $\mathbf{x}_c$ such that $y_c = \sum_{j=1}^{d} \mathbf{x}_c[j] + 10^{100}$. Then, as we can see, if we were to do a standard least-squares solution that attempts to minimize (1), this single crazy observation would be enough to shift our estimated $\hat{\mathbf{p}}$ from the "true" $\mathbf{p}^* = [1, \ldots, 1]^T$ by a large amount. Why? Because $\hat{\mathbf{p}} = (X^T X)^{-1} X^T \mathbf{y}$ is a linear function of $\mathbf{y}$ and hence the crazy huge deviation in the $c$-th component of $\mathbf{y}$ is going to cause a huge multiple of the $c$-th column of $(X^T X)^{-1} X^T$ to be added to the estimate for $\mathbf{p}$. The $c$-th column of $(X^T X)^{-1} X^T$ is just $(X^T X)^{-1} \mathbf{x}_c$ by our definition of the matrix $X$ above. And so, from the perspective of being an outlier that corrupts our estimate, it really doesn't much matter whether the observation fault was in the scalar $y_c$ or in the vector $\mathbf{x}_c$ — whichever side of the approximate equation representing the $c$-th data point is actually messed up, our estimate is going to be pretty badly messed up if we just blindly use least-squares.

Consequently, it is evident that the least-squares-estimated $\hat{\mathbf{p}}$ is not what we really always want. Is there a way that allows us to reliably remove outliers when we know only a small proportion of the data points are outliers?

In this problem, we will demonstrate one of the simplest outlier removal methods that leverages the orthogonal matching pursuit (OMP) approach you learned in 16A.

(a) As we saw in our example, we could improve our solution if we could remove outliers. One way to do this is by augmenting our matrices in the following way. Let $\mathbf{e}_i$ be the $i$-th standard

basis vector. That is, $\mathbf{e}_i = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$ where the solitary 1 is in the $i$-row of the vector $\mathbf{e}_i$. Consider the augmented data matrix and parameter vectors:

$$X_{new} = \begin{bmatrix} X & \mathbf{e}_i \end{bmatrix}, \quad \mathbf{y}_{new} = \mathbf{y}, \quad \mathbf{p}_{new} = \begin{bmatrix} \mathbf{p} \\ f_i \end{bmatrix}. \tag{2}$$

Here the variable $f_i$ is effectively a "fakeness" variable that we associate with the $i$-th data point. Apply the standard understanding of least squares to find the solution to

$$\min_{\mathbf{p}_{new}} \|\mathbf{y}_{new} - X_{new}\mathbf{p}_{new}\|^2. \tag{3}$$

and **write out the formula for the least-squares estimate** $\widehat{\mathbf{p}}_{new} = \begin{bmatrix} \widehat{\mathbf{p}} \\ \widehat{f_i} \end{bmatrix}$.

(b) In the previous part, the $(d+1)\times(d+1)$ matrix $X_{new}^T X_{new}$ played a role. Let's look at this matrix in "block" form:

$$X_{new}^T X_{new} = \begin{bmatrix} X & \mathbf{e}_i \end{bmatrix}^T \begin{bmatrix} X & \mathbf{e}_i \end{bmatrix} = \begin{bmatrix} X^T \\ \mathbf{e}_i^T \end{bmatrix} \begin{bmatrix} X & \mathbf{e}_i \end{bmatrix} = \begin{bmatrix} X^T X & X^T \mathbf{e}_i \\ \mathbf{e}_i^T X & \mathbf{e}_i^T \mathbf{e}_i \end{bmatrix}$$

**What are** $X^T \mathbf{e}_i, \quad \mathbf{e}_i^T X, \quad \mathbf{e}_i^T \mathbf{e}_i$**?**

Simplify these as much as possible in terms of the individual data points $\mathbf{x}_i$, etc.

(c) Based on what you know from the previous part, revisit the formula you got in the part before that and **prove that the value for** $y_i$ **in the** $i$**-th data point only impacts the estimate** $\widehat{f_i}$ **and does not effect the least-squares estimate for** $\widehat{\mathbf{p}}$ **at all.**

*(HINT:* $y_i$ *sits in the i-th row of the vector* $\mathbf{y}$*. What does the nature of matrix multiplication imply for its influence on the least-squares estimate? Read the full problem text above for a reminder. What does the inverse of a matrix do to a particular column of that matrix?)*

(d) Continuing the previous part, use what you know to **prove that** $y_i = \widehat{\mathbf{p}}^T \mathbf{x}_i + \widehat{f_i}$. That is, regardless of what $y_i$ is, there is no residual left in the $i$-th position.

(e) Given that $\widehat{\mathbf{p}}_{new} = \begin{bmatrix} \widehat{\mathbf{p}} \\ \widehat{f_i} \end{bmatrix}$ minimizes (3), **show that** $\widehat{\mathbf{p}}$ **minimizes** $\sum_{j\neq i}(y_j - \mathbf{x}_j^T \mathbf{p})^2$**.**

Note that here, $\sum_{j\neq i}$ is just a compact way to write sum up over all indices $j$ from 1 to $n$ but skip the $j = i$ term in the sum.

*(HINT: There are many ways to proceed. Let* $\mathbf{p}'$ *be the vector that minimizes* $r = \sum_{j\neq i}(y_j - \mathbf{x}_j^T \mathbf{p})^2$.

*Use this to construct an $f_i'$ so that $\mathbf{p}'_{new} = \begin{bmatrix} \mathbf{p}' \\ f_i' \end{bmatrix}$ achieves the same $r$ in (3). Could the cost in (3) possibly go any lower? Recall that a square of a real number is always $\geq 0$.)*

(f) **Argue why the above implies that the resulting solution is the same parameter estimate that we would have gotten had we simply removed the $i$-th data-point entirely from our data set.**

(g) From what you have seen above together with what you see from running the attached jupyter notebook, **argue in words why augmenting the data matrix $X$ with an identity and then running OMP is a reasonable approach to systematically eliminating outliers.**

In the next homework, you will see how we can find the right stopping condition for OMP.

# 6 Geometry of Ridge Regression

You recently learned ridge regression and how it differs from ordinary least squares. In this question we will explore useful interpretations and reconceptualizations of ridge regression.

(a) Recall that ridge regression can be understood as the unconstrained optimization problem

$$\min_{w} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2, \tag{4}$$

where $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a design matrix (data), and $\mathbf{y} \in \mathbb{R}^n$ is the target vector of measurement values.

One way to interpret "ridge regression" is as the ordinary least squares for an augmented data set — i.e. adding a bunch of fake data points to our data. Consider the following augmented measurement vector $\hat{\mathbf{y}}$ and data matrix $\hat{\mathbf{X}}$:

$$\hat{\mathbf{y}} = \begin{pmatrix} \mathbf{y} \\ \mathbf{0}_d \end{pmatrix} \quad \hat{\mathbf{X}} = \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{I}_d \end{pmatrix},$$

where $\mathbf{0}_d$ is the zero vector in $\mathbb{R}^d$ and $\mathbf{I}_d \in \mathbb{R}^{d \times d}$ is the identity matrix. **Show that the optimization problem $\min_w \|\hat{\mathbf{y}} - \hat{\mathbf{X}}\mathbf{w}\|_2^2$ has the same minimizer as (4).**

(b) Perhaps more surprisingly, one can achieve the same effect as in the previous part by adding fake features to each data point instead of adding fake data points. (Does this remind you of a previous problem?) Let's construct the augmented design matrix in the following way:

$$\hat{\mathbf{X}} = [\mathbf{X}\ \alpha\mathbf{I_n}]$$

i.e. we stack $\mathbf{X}$ with $\alpha\mathbf{I}_n$ horizontally. Here $\alpha$ is a scalar multiplier. Now our problem is underdetermined: the new dimension $d + n$ is larger than the number of points $n$. Therefore, there are infinitely many values $\boldsymbol{\eta} \in \mathbb{R}^{d+n}$ for which $\hat{\mathbf{X}}\boldsymbol{\eta} = \mathbf{y}$. Consider the following problem:

$$\min_{\boldsymbol{\eta}} \|\boldsymbol{\eta}\|_2^2 \text{ s.t. } \hat{\mathbf{X}}\boldsymbol{\eta} = \mathbf{y}. \tag{5}$$

**Find the $\alpha$ that that if $\eta^*$ is the minimizer of (5), then the first $d$ coordinates of $\eta^*$ form the minimizer of (4).**

**Can you interpret what the final $n$ coordinates of $\eta^*$ represent?**

(c) Suppose the SVD of $\mathbf{X}$ is $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. Let's make a change of coordinates in the feature space, such that $\mathbf{V}$ becomes identity: $\mathbf{X}_{\text{new}} = \mathbf{X}\mathbf{V}$ and $\mathbf{w}_{\text{new}} = \mathbf{V}^\top\mathbf{w}$. Denote the the solution to ridge regression (4) in these new coordinates as $\hat{\mathbf{w}}_{\text{new}}$. **Show that the $i$-th coordinate of $\mathbf{w}_{\text{new}}$ can be obtained from the corresponding coordinate of $\mathbf{U}^\top\mathbf{y}$ by multiplication by $\frac{\sigma_i}{\sigma_i^2+\lambda}$, where $\sigma_i$ is the $i$-th singular value of $\mathbf{X}$ (or zero if $i$ is greater than the rank of $X$.)**

(d) One reason why we might want to have small weights $\mathbf{w}$ has to do with the sensitivity of the predictor to its input. Let $\mathbf{x}$ be a $d$-dimensional list of features corresponding to a new test point. Our predictor is $\mathbf{w}^\top\mathbf{x}$. **What is an upper bound on how much our prediction could change if we added noise $\boldsymbol{\epsilon} \in \mathbb{R}^d$ to a test point's features $\mathbf{x}$?**

(e) We know that the solution to ridge regression (4) is given by $\hat{\mathbf{w}}_\mathbf{r} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$. **What happens when $\lambda \to \infty$?** It is for this reason that sometimes ridge regularization is referred to as "shrinkage."

(f) Another advantage of ridge regression can be seen for under-determined systems. Say we have the data drawn from a $d = 5$ parameter model, but only have $n = 4$ training samples of it, i.e. $\mathbf{X} \in \mathbb{R}^{4\times5}$. Now this is clearly an underdetermined system, since $n < d$. **Show that ridge regression with $\lambda > 0$ results in a unique solution, whereas ordinary least squares can have an infinite number of solutions.**

*[Hint: To make this point, it may be helpful to consider $\mathbf{w} = \mathbf{w}_0 + \mathbf{w}^*$ where $\mathbf{w}_0$ is in the null space of $\mathbf{X}$ and $\mathbf{w}^*$ is a solution.]*

*[Alternative Hint: You might want to consider (5) as the way to interpret ridge regression.]*

(g) For the previous part, **what will the answer be if you take the limit $\lambda \to 0$ for ridge regression?**

*[Hint: You might want to consider (5) as the way to interpret ridge regression.]*

(h) Tikhonov regularization is a general term for ridge regression, where the implicit constraint set takes the form of an ellipsoid instead of a ball. In other words, we solve the optimization problem

$$\mathbf{w} = \arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\Gamma\mathbf{w}||_2^2$$

for some full rank matrix $\Gamma \in \mathbb{R}^{d\times d}$. **Derive a closed form solution to this problem.**

# 7 Approximating a 1D function

It is really useful for students to be able to anchor your intuitions about machine learning in intuitions that you already have from calculus, etc. Universal function approximation can sometimes

come across as mysterious, whereas in reality, it is something that you are already very familiar with. This is a problem designed to remind you of things that you already essentially know.

(a) For a $p$-times differentiable function $f : \mathbb{R} \to \mathbb{R}$, the Taylor series expansion of order $m \leq p-1$ about the point $x_0$ is given by

$$f(x) = \sum_{i=0}^{m} \frac{1}{i!} f^{(i)}(x_0)(x - x_0)^i + \frac{1}{(m+1)!} f^{(m+1)}(a(x))(x - x_0)^{m+1}. \tag{6}$$

Here, $f^{(m)}$ denotes the $m$th derivative of the function $f$, and $a(x)$ is some value between $x_0$ and $x$. By definition, $f^{(0)} = f$.

The last term $r_m(x) := \frac{1}{(m+1)!} f^{(m+1)}(a(x))(x - x_0)^{m+1}$ of this expansion is typically referred to as the remainder term when approximating $f(x)$ by an $m$-th degree polynomial.

We denote by $\phi_m$ the $m$-th degree Taylor polynomial (also called the Taylor *approximation*), which consists of the Taylor series expansion of order $m$ without the remainder term and thus reads

$$f(x) \approx \phi_m(x) = \sum_{i=0}^{m} \frac{1}{i!} f^{(i)}(x_0)(x - x_0)^i$$

where the sign $\approx$ indicates approximation of the left hand side by the right hand side.

For functions $f$ whose derivatives are bounded in the neighborhood $I$ around $x_0$ of interest, if we have $|f^{(m)}(x)| \leq T$ for $x \in I$, we know that for $x \in I$ that the *approximation error* of the $m$-th order Taylor approximation $|f(x) - \phi_m(x)| = |r_m(x)|$ is upper bounded $|f(x) - \phi_m(x)| \leq \frac{T|x - x_0|^{m+1}}{(m+1)!}$.

**Compute the 1st, 2nd, 3rd, and 4th order Taylor approximation of the following functions around the point $x_0 = 0$.**

- $e^x$
- $\sin x$

(b) For $f(x) = e^x$, **plot the Taylor approximation from order 1 through 4 at $x_0 = 0$ for $x$ in the domain $I := [-4, 3]$ using the provided jupyter notebook.**

We denote the maximum approximation error on the domain $I$ by $\|f - \phi_m\|_\infty := \sup_{x \in I} |f(x) - \phi_m(x)|$, where $\|\cdot\|_\infty$ is also called the sup-norm with respect to $I$. **Compute $\|f - \phi_m\|_\infty$ for $m = 2$. Compute the limit of $\|f - \phi_m\|_\infty$ as $m \to \infty$.**

Hint: Use Stirling's approximation for integers $n$ which is: $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$.

**Now plot the Taylor approximation of $f$ up to order 4 on the interval $[-20, 8]$. How does the approximation error behave outside the bounded interval $I$?**

(c) Let's say we would like an accurate polynomial approximation of the functions in part (a) for all $x \in I$. Given the results of the previous parts, we can in fact find a Taylor polynomial of degree $D$ such that $|f(x) - \phi_D(x)| \leq \epsilon$ for all $x \in I$. **What is an upper bound of $\|f - \phi_m\|_\infty$ for arbitrary non-zero integers $m$? Using this upper bound, show that if $D$ is larger than $O(\log(1/\epsilon))$, we can guarantee that $\|f - \phi_D\|_\infty \leq \epsilon$ for both choices of $f(x)$ in part (a).** Note that constant factors are not relevant here, and assume sufficiently small positive $\epsilon \ll 1$.

(d) **Conclude that a univariate polynomial of high enough degree can approximate any function $f$ on a closed interval $I$, that is continuously differentiable infinitely many times and has bounded derivatives $|f^{(m)}(x)| \leq T$ for all $m \geq 1$ and $x \in I$.** Mathematically speaking, we need to show that for any $\epsilon > 0$, there exists a degree $D \geq 1$ such that $\|f - \phi_D\|_\infty < \epsilon$, where the sup-norm is taken with respect to the interval $I$.

This universal approximation property illustrates the power of polynomial features, even when we don't know the underlying function $f$ that is generating our data! Later, we will see that neural networks are also universal function approximators.)

(e) Now we will approximate a function using periodic functions instead of polynomials. Let $f\colon [0, 2\pi] \to \mathbb{R}$ be a piecewise differentiable function. Suppose it were possible to find coefficients $c_k \in \mathbb{C}$ such that

$$f(x) = \sum_{k=-\infty}^{\infty} c_k \exp(jkx).$$

**Give a formula to compute the coefficient $c_k$ given access to the function $f(x)$.**

*[Hint: $\int_0^{2\pi} e^{jnx} = 0$ for a nonzero integer $n$. Think about the set of functions $\exp(jkx)$ on the interval $[0, 2\pi]$ as a set of functions that are orthogonal to each other using the natural complex inner product on these functions. What is that natural inner product? Do the natural replacement of a sum with an integral.]*

(f) Of course, it is not guaranteed that a given function $f$ can be written in this form. We define the complex Fourier series of $f$ as

$$\hat{f}(x) = \sum_{k=-\infty}^{\infty} c_k \exp(jkx),$$

where $c_k$ is what you just computed. The truncated Fourier series is given by

$$\hat{f}_m(x) = \sum_{k=-m}^{m} c_k \exp(jkx).$$

**Give a formula for the Fourier coefficients (i.e., $c_k$) of $f(x) = \begin{cases} 1, & 0 \leq x < \pi \\ -1, & \text{otherwise} \end{cases}$ and use the jupyter notebook to plot the Fourier series for this function for $m = 1$, 3, and 5.**

(g) Now you'll show that the Fourier series nicely approximates an appropriately square-integrable function. Parseval's equality states that $\sum_{j=-\infty}^{\infty} |c_j|^2 = \frac{1}{2\pi} \int_0^{2\pi} f^2(x)dx$. In other words, $\lim_{m\to\infty} \sum_{j=-m}^{m} |c_j|^2 = \frac{1}{2\pi} \int_0^{2\pi} f^2(x)dx$. If you want to understand why this is true, you can take appropriate limits to prove it, but we don't ask you to do that here.

**Use this result to show that $\lim_{m\to\infty} \int_0^{2\pi} |f(x) - \hat{f}_m(x)|^2 dx = 0$.**

# 8 Learning 1-D functions

In this problem we explore the canonical example of learning one dimensional functions from samples. Read the questions in the associated notebook and fill in code in the required places. Most of the code functions as a playground for you to play around with different parameters and understand their effect.

(a) **Read the question mentioned in the jupyter notebook and write your solution here.**

(b) **Read the question mentioned in the jupyter notebook and write your solution here.**

(c) **Read the question mentioned in the jupyter notebook and write your solution here (excluding code).**

(d) **Read the question mentioned in the jupyter notebook and write your solution here (excluding code).**

(e) **Read the question mentioned in the jupyter notebook and write your solution here (excluding code).**

(f) **Read the question mentioned in the jupyter notebook and write your solution here (excluding code).**

(g) **Read the question mentioned in the jupyter notebook and write your solution here.**

(h) **Read the question mentioned in the jupyter notebook and write your solution here.**

(i) **Read the question mentioned in the jupyter notebook and write your solution here.**

(j) **Read the question mentioned in the jupyter notebook and write your solution here.**

# 9 Fourier features for regularly spaced samples

In this problem we want to show that Fourier features interact with measurements on a grid in a pleasing way. Suppose we measure our functions in points $\{x_j = 2\pi m/(2n+1)\}_{m=-n}^{n}$ — a grid on $(-\pi, \pi)$. If we introduce $2n+1$ features

$$\left\{\frac{1}{\sqrt{2n+1}}\right\}, \left\{\frac{\sqrt{2}\cos(kx)}{\sqrt{2n+1}}\right\}_{k=1}^{n} \text{ and } \left\{\frac{\sqrt{2}\sin(kx)}{\sqrt{2n+1}}\right\}_{k=1}^{n},$$

the resulting $(2n+1)\times(2n+1)$ matrix will be orthogonal. To prove that directly requires dealing with trigonometry, so we take another approach: look at the complex features

$$\left\{\frac{\exp(jkx)}{\sqrt{2n+1}}\right\}_{k=-n}^{n},$$

where $j$ stands for imaginary unit.

(a) **Show that for the complex features the matrix of data will be unitary.**

(b) **Use the expression $e^{j\phi} = \cos(\phi) + j\sin(\phi)$ and the previous part to show that the data matrix in real features is orthogonal.**

(*HINT: you can express* $2\cos(x) = e^{jx} + e^{-jx}$ *and* $2\sin(x) = -j(e^{jx} - e^{-jx})$*, and you already know that columns with complex exponents are orthonormal.*)

# 10 A Simple Classification Approach

**Make sure to submit the code you write in this problem to "HW1 Code" on Gradescope.**

Classification is an important problem in applied machine learning and is used in many different applications like image classification, object detection, speech recognition, machine translation and others.

In *classification*, we assign each datapoint a class from a finite set (for example the image of a digit could be assigned the value $0, 1, \ldots, 9$ of that digit). This is different from *regression*, where each datapoint is assigned a value from a continuous domain like $\mathbb{R}$ (for example features of a house like location, number of bedrooms, age of the house, etc. could be assigned the price of the house).

In this problem we consider the simplified setting of classification where we want to classify data points from $\mathbb{R}^d$ into *two* classes. For a linear classifier, the space $\mathbb{R}^d$ is split into two parts by a hyperplane: All points on one side of the hyperplane are classified as one class and all points on the other side of the hyperplane are classified as the other class.

The goal of this problem is to show that even a regression technique like linear regression can be used to solve a classification problem. This can be achieved by regressing the data points in the training set against $-1$ or 1 depending on their class and then using the level set of 0 of the regression function as the classification hyperplane (i.e. we use 0 as a threshold on the output to decide between the classes).

Later in lecture we will learn why linear regression is not the optimal approach for classification and we will study better approaches like logistic regression, SVMs and neural networks.

(a) The dataset used in this exercise is a subset of the MNIST dataset. The MNIST dataset assigns each image of a handwritten digit their value from 0 to 9 as a class. For this problem we only keep digits that are assigned a 0 or 1, so we simplify the problem to a two-class classification problem.

**Download and visualize the dataset (example code included). Include three images that are labeled as 0 and three images that are labeled as 1 in your submission.**

(b) We now want to use linear regression for the problem, treating class labels as real values $y = -1$ for class "zero" and $y = 1$ for class "one". In the dataset we provide, the images have already been flattened into one dimensional vectors (by concatenating all pixel values of the two dimensional image into a vector) and stacked as rows into a feature matrix $\mathbf{X}$. We want to set up the regression problem $\min_{\mathbf{w}} \left\|\mathbf{X}\mathbf{w} - \mathbf{y}\right\|_2^2$ where the entry $y_i$ is the value of the class

(−1 or 1) corresponding to the image in row $\mathbf{x}_i^\top$ of the feature matrix. **Solve this regression problem for the training set and report the value of $\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$ as well as the weights w.** For this problem you may only use pure Python and numpy (no machine learning libraries!).

(c) Given a new flattened image $\mathbf{x}$, one natural rule to classify it is the following one: It is a zero if $\mathbf{x}^\top\mathbf{w} \leq 0$ and a one if $\mathbf{x}^\top\mathbf{w} > 0$. **Report what percentage of the digits in the training set are correctly classified by this rule. Report what percentage of the digits in the test set are correctly classified by this rule.**

(d) **Why is the performance typically evaluated on a separate test set (instead of the training set) and why is the performance on the training and test set similar in our case?** We will cover these questions in more detail later in the class.

(e) Unsatisfied with the current performance of your classifier, you call your mother for advice, and she suggests to use random features instead of raw pixel features. Specifically, she suggests to use the feature map

$$\phi(\mathbf{x}) = \cos\left(\mathbf{G}^\top\mathbf{x} + \mathbf{b}\right),$$

where each entry of $G \in \mathbb{R}^{d\times p}$ is drawn i.i.d. as $\mathcal{N}(0, 0.01)$ and each entry in the vector $b \in \mathbb{R}^p$ is drawn i.i.d from the uniform distribution on $[0, 2\pi]$. Note that cos should be taken point wise, i.e. $\phi(x)$ should output a vector in $\mathbb{R}^p$. **With $p = 2500$, report what percentage of digits are correctly classified using this approach on the training set and test set. Make sure to adapt the same classification rule, i.e. the threshold set for the outputs.**

# 11 System Identification by ordinary least squares regression

Making autonomous vehicles involves machine learning for different purposes. One of which is learning how cars actually behave based on their data.

**Make sure to submit the code you write in this problem to "HW1 Code" on Gradescope.**

(a) Consider the time sequence of scalars $x_t \in \mathbb{R}$ and $u_t \in \mathbb{R}$ in which $x_{t+1} \approx Ax_t + Bu_t$. In control theory, $x_t$ usually represents the state, and $u_t$ usually represents the control input. **Find the numbers $A$ and $B$ so that $\sum_t (x_{t+1} - Ax_t - Bu_t)^2$ is minimized.** The values of $x_t$ and $u_t$ are stored in `a.mat`.

(b) Consider the time sequences of vectors $\mathbf{x}_t \in \mathbb{R}^3$ and $\mathbf{u}_t \in \mathbb{R}^3$ in which $\mathbf{x}_{t+1} \approx \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t$. **Find the matrix $\mathbf{A} \in \mathbb{R}^{3\times3}$ and $\mathbf{B} \in \mathbb{R}^{3\times3}$ so that the sum of the squared $\ell^2$-norms of the error, $\sum_t \|\mathbf{x}_{t+1} - \mathbf{A}\mathbf{x}_t - \mathbf{B}\mathbf{u}_t\|_2^2$, is minimized**. The values of $\mathbf{x}_t$ and $\mathbf{u}_t$ are stored in `b.mat`.

(c) Consider a *car following model* that models how cars line up on a straight 1D highway at a given time. The acceleration of a car can be approximated by a linear function of the positions and velocities of its own and the car in front of it. Mathematically, we can formulate this as

$$\ddot{x}_i \approx ax_i + b\dot{x}_i + cx_{i-1} + d\dot{x}_{i-1} + e,$$

where $x_i$, $\dot{x}_i$, and $\ddot{x}_i$ are the position, velocity, and acceleration of the $i$th car in the line.

**Find $a$, $b$, $c$, $d$, and $e$ that minimize**

$$\sum_i \|-\ddot{x}_i + ax_i + b\dot{x}_i + cx_{i-1} + d\dot{x}_{i-1} + e\|_2^2$$

**using data file `train.mat`**, which contains the status of 40 000 cars at a given point from the I-80 highway in California. The data were sampled from the Next Generation Simulation (NGSIM) dataset so that the $i$ may not be continuous. For your convenience, we give you the profiles of each sampled car and the car that is in front of it.

(d) **Try to justify why your result in (c) is physically reasonable.** Hint: You can reorganize your equation to be

$$\ddot{x}_i = h(x_{i-1} - x_i) + f(\dot{x}_{i-1} - \dot{x}_i) - g(\dot{x}_i - L) + w_i,$$

and try to explain the physical meaning for each term, with $L$ being the speed limit.

# 12 Your Own Question

**Write your own question, and provide a thorough solution.**

Writing your own problems is a very important way to really learn the 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.

Contributors:

- Alexander Tsigler
- Alvin Wan
- Anant Sahai
- Ashwin Pananjady
- Chawin Sitawarin
- Fanny Yang
- Inigo Incer
- Jane Yu
- Josh Sanz
- Kate Sanders
- Khalil Sarwari
- Kuan-Yun Lee
- Peter Wang
- Philipp Moritz
- Raaz Dwivedi
- Ross Bozcar
- Ruta Jawale
- Satish Rao
- Soroush Nasiriany
- Vignesh Subramanian
- Yichao Zhou