EECS 189 Introduction to Machine Learning
Fall 2020 HW6

This homework is due **Wednesday, October 14 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.

The most important problem on this homework is technically optional if you have already done this and are completely confident in your setup. However, if you are not fully confident, it is worth doing this again to reduce stress ahead of the actual midterm. Also, be aware that if you experience any kind of technical difficulties and do not have a successfully fully completed practice on file, you will be given zero accommodations of any kind even if the disruption was entirely out of your control. If a student can't be bothered to actually test out their exam setup, they are taking that risk entirely on their own head.

2 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.*

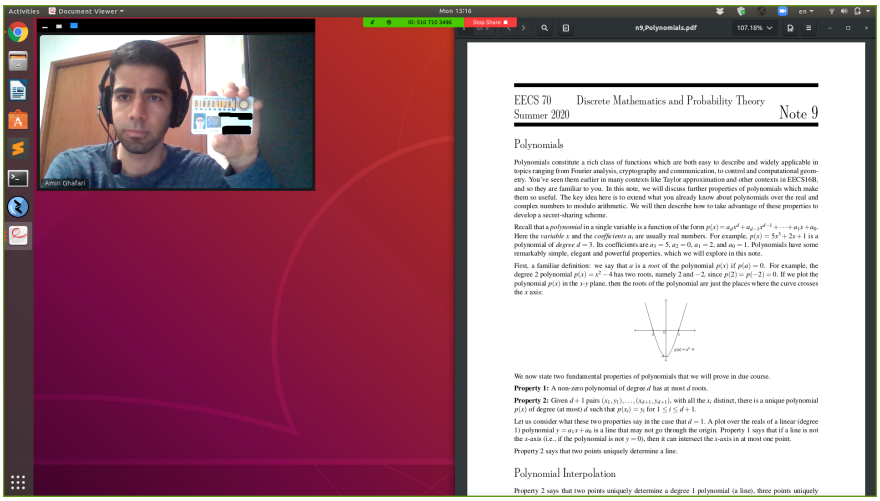


Figure 1: ID card demonstration. Do not actually black out your SID and name.

(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.**

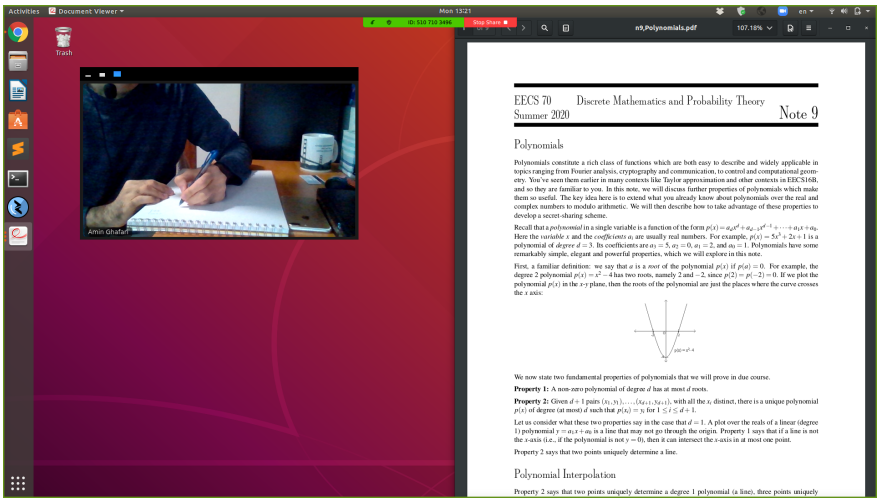


Figure 2: Demonstration of taking your exam. Your setup should look like this while you are taking the exam.

(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

The first real problem in this homework is designed to exercise your understanding of probability, Gaussians, and coordinate changes. It is not a previous exam problem, but you have not had quite enough practice on this and these skills will be useful on the midterm exam this semester. This isn't really a machine-learning question. Consider it a warm-up exercise.

3 Gaussian Quadrant Probability

Consider a two-dimensional Gaussian random variable $\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim \mathcal{N}(0, \Sigma)$ where Σ is a positive definite matrix,

$$\Sigma = \begin{bmatrix} a & c \\ c & b \end{bmatrix}.$$

In this problem we calculate the probability $P(X_1 > 0, X_2 > 0)$.

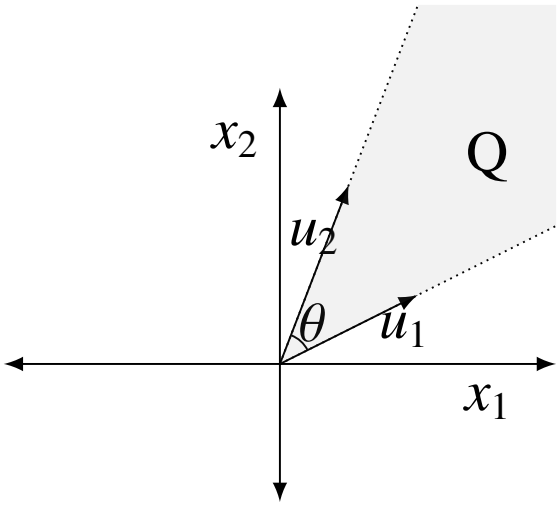
(a) **Define an appropriate $\Sigma^{-\frac{1}{2}}$ and show that $\mathbf{Y} = \Sigma^{-\frac{1}{2}}\mathbf{X} \sim \mathcal{N}(0, \mathbf{I})$.**

(b) Let Q denote the region in the 2D-plane given by,

$$Q = \{(x_1, x_2) \mid x_1 > 0, x_2 > 0\}.$$

Note that $P(X_1 > 0, X_2 > 0) = P(\mathbf{X} \in Q)$. **Show that $P(\mathbf{X} \in Q) = P(\mathbf{Y} \in S)$ where S is a sector of the 2D-plane centered around the origin with angle $\theta \in [0, \pi]$ as shown in the figure with,**

$$\theta = \cos^{-1}\left(\frac{c}{\sqrt{ab}}\right).$$



(Hint 1: Compute the mapping of the points $(1, 0), (0, 1)$ *under the mapping* $\Sigma^{-\frac{1}{2}}$ *to get* $\mathbf{u}_1, \mathbf{u}_2$*.)*
(Hint 2: Note that ,

$$\Sigma^{-1} = \frac{1}{|ab - c^2|}\begin{bmatrix} b & -c \\ -c & a \end{bmatrix}.$$

)

(c) **Conclude that $P(\mathbf{Y} \in S) = \frac{1}{2\pi}\cos^{-1}\left(\frac{c}{\sqrt{ab}}\right)$.**

(HINT: If you look at $\mathbf{Y}$ in polar coordinates, what is the probability distribution of the angle? Remember the distribution for $\mathbf{Y} \sim \mathcal{N}(0, \mathbf{I})$.)

Thus we have shown that

$$P(X_1 > 0, X_2 > 0) = \frac{1}{2\pi}\cos^{-1}\left(\frac{-c}{\sqrt{ab}}\right).$$

The second problem on this homework is designed to further exercise skills and concepts that you have seen in earlier homeworks this semester, but were not in scope in earlier semesters. This problem is designed to help make sure that you get more practice with these ideas ahead of the exam, since these ideas are definitely in scope.

4 Linear Regression in Overparameterized Regime: Part II

Recall from the previous homework how conducting linear regression in the overparameterized regime, where many features aliased each other on the training points, led to an effect similar to explicit ridge regularization.

We repeat the setup from that homework problem below, for your reference.

[Begin setup from previous homework]

We are trying to learn a one-dimensional function $f(x)$ for $x \in [0, 1]$. For mathematical ease, consider a standard complex Fourier featurization:

$$\phi_k^u(x) = e^{j2\pi kx}$$

for $k = \{0, 1, \ldots, d-1\}$. Notice that this featurization is orthonormal relative to a uniform test-distribution on $[0, 1]$.

Assume that the number of features $d = (M+1)n$ for a positive integer M.

We have access to observations $\mathbf{y} \in \mathbb{C}^n$ where

$$\mathbf{y} = f(\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix}) + \boldsymbol{\epsilon} \tag{1}$$

where the scalar complex function $f(x)$ is applied element-wise to the vector of training sampling locations $\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix}$ and $\boldsymbol{\epsilon}$ refers to the noise in the training data and we assume $\boldsymbol{\epsilon} \sim \mathcal{CN}(0, \sigma^2 \mathbf{I}_n)$. We further assumed that $f(x) = \phi_t^u(x)$.

Instead of performing linear regression using this featurization we first scale the features to obtain *scaled features*,

$$\phi_k(x) = c_k \phi_k^u(x)$$

where $c_k \in \mathbb{R}$ for $k = 0, 1, \ldots, d-1$. Let $\boldsymbol{\Phi}_t$ denote the column vector with entries $\phi_t(x_0), \phi_t(x_1), \ldots, \phi_t(x_{n-1})$, and similarly $\boldsymbol{\Phi}_t^u$ the unscaled version of the same thing. To do learning, we solve the minimum norm interpolation problem using the scaled features:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\|\boldsymbol{\beta}\right\|_2$$

$$\text{s.t. } \mathbf{y} = \sum_{k=0}^{d-1} \beta_k \mathbf{\Phi}_k,$$

Having found $\hat{\boldsymbol{\beta}}$ we convert it into an equivalent set of coefficients $\hat{\boldsymbol{\alpha}}$ to obtain,

$$\mathbf{y} = \sum_{k=0}^{d-1} \hat{\alpha}_k \mathbf{\Phi}^u_k,$$

where $\hat{\alpha}_k = c_k \hat{\beta}_k$.

The test data $x_{\text{test}}, y_{\text{test}}$ is drawn from the distribution p and is noiseless. Recall that the features $\{\phi^u_k\}_{k=0,\ldots,d-1}$ are orthornormal with respect to the test distribution.

We make a prediction at test time using the learned coefficients,

$$y_{\text{pred}} = \sum_{k=0}^{d-1} \hat{\beta}_k \phi_k(x_{\text{test}}) = \sum_{k=0}^{d-1} \hat{\alpha}_k \phi^u_k(x_{\text{test}}).$$

[End setup from previous homework]

Last time, we showed that the expected squared test error

$$\mathbb{E}_{pred} = \mathbb{E}[(f(x) - \Phi(x)\hat{\beta})^2] = \left(1 - \frac{c_t^2}{V(t)}\right)^2 + \frac{1}{V^2(t)} \sum_{k \in A(t)} c_k^4 + \frac{\sigma^2}{n} \left(\sum_{q=0}^{n-1} \frac{1}{V^2(q)} \sum_{k \in R(q)} c_k^4 \right),$$

where $f(x) = \phi^u_t(x)$ is the true function that is a pure unscaled feature with $t < n$ and $A(t) = \{n + t, 2n + t, \ldots\}$ is the aliasing set of all feature indices that alias ϕ^u_t, and $R(t) = A(t) \cup \{t\}$ is the set of all relevant feature indices that alias ϕ^u_t, as well as t itself. Here, the notation $V(q) = \sum_{k \in R(q)} c_k^2$ was essentially a measure of the total energy in all the scaled features in the group $R(q)$.

Now, we will consider particular choices of the scalings c_i intended to have a regularizing effect. Last time, we looked at the strict "bi-level" model, where the first s features were given identically large scalings, and the remaining $d - s$ features were given comparatively small scalings, also identical. We saw that the effects of the noise present in our training data were distributed over the remaining $d - s$ features in a manner that did not significantly affect our performance on the test data, without significantly affecting the learned weight on the tth (true) feature.

We will now explore what happens when we perturb the strict Bi-Level scalings to no longer be strictly identical. Recall that in the previous homework, that as long as the true function was some linear combination of the first s features, we were able to treat our true function as though it were a single feature $\phi^u_t(x)$, because we could always rotate our orthonormal featurization to make that the case, without affecting the feature scaling. This was because the first s features all had the same scaling, so we could rotate them to produce s new features with that same scaling — and then in spirit appliy the identical rotation to each group of aliases of these s features. Since minimum-norm solutions would essentially be unaffected by these kinds of unitary transformations, the entire story would still go through and the expressions for the expected squared-loss on test points would be unchanged.

However, now that we are going to be perturbing the scalings of each feature independently, we can no longer rely on this "trick" to simplify our setup. Instead, we need to consider the general case, when our true function is of the form

$$y = f(x) = \sum_{k=0}^{t-1} \alpha_k \phi_k^u(x), \tag{2}$$

where the α_k are bounded complex coefficients such that $|\alpha_k| \leq 1$. We also give ourselves the freedom to choose $t \leq s$, so that all the features in the true function are in the "favored" group of features, as per the original bi-level model.

(a) Suppose we had noise-free observations of the form

$$\mathbf{y} = \sum_{k=0}^{n-1} w_k \phi_k^u(x),$$

on the same regularly spaced sampling points as the previous homework problem.

Show that the min-norm interpolating solution yields

$$\hat{\alpha}_k = \frac{c_k^2}{V(q)} w_k.$$

(Hint: this is a simple consequence of the work that you already did in the previous homework. You don't have to redo that work. What is the property of the min-norm interpolating solution that we are using here?)

(b) Now, let us assume that the true noiseless function is given by (2) and we have noisy observations of it as in (1).

Find the expected test error $\mathbb{E}_{pred}$ as a function of the $\{\alpha_k\}$, the $\{c_k\}$, and σ, taking an expectation over both the test distribution for x and the training noise ϵ. This calculation will be fairly similar to what you did in the previous homework, and you can verify your result by checking that it matches your previous result when all the $\alpha_k = 0$, except for $\alpha_t = 1$.

(HINT: Remember to invoke the same "frequency-domain" change-of-coordinates on the noise as you did last time — transform the training noise into the i.i.d δ_i that were each $\mathcal{CN}(0, \frac{\sigma^2}{n})$. This will let you decompose the problem across different groups $R(q)$ and deal with each separately by leveraging the orthonormality of the features relative to the test distribution.)

(c) Now we will look at how these error terms respond to changes in the feature scalings. There are essentially four different windows of interest that one could think about. There's the favored true features from 0 to $t-1$, there are the favored features that happen not to be true from t to $s-1$, there are the unfavored features from s to $n-1$, and then there is the tail of aliases from s to d.

In the strict bi-level model, recall that all c_q for $q \geq n$ are set equal to some fixed quantity: denote it as c for this part. Let's leave that alone.

For some $q \in [s, n)$, imagine that we increase or decrease c_q. In each case, what happens to $\mathcal{E}_{\text{pred}}$? **Show that it monotonically increases as c_q increases to be bigger than c and monotonically increases as c_q decreases smaller than c with a unique global optimum at $c_q = c$.**

(HINT: Does this affect the survival of the true pattern? Does this affect any contamination that might be coming from the true pattern itself? Does this affect the contamination that is coming from noise in the training data? Once you have identified the relevant term that is impacted, take a derivative to see what is going on.)

(d) Now that we understand the qualitative effects of these perturbations, we will show that we do not need *exactly* the bi-level model to obtain the desired regularization effects. One key property of the bi-level scalings is that they "favor" the first s features, including the t that are involved in the true functions, so the learned weights associated with those features do not get too thinly distributed over the M aliases. Another key property is that the training noise, having independent and identically distributed components in all n aliasing groups, is "spread" and thus diluted over the many aliases in the trailing $d - s$ features, since their scalings are all low and roughly the same.

In this part, we will examine this second property. From the previous part, you should have seen that decreasing c_q increases the test error, due to noise contamination. One way to visualize this is to imagine that the noise is now spread less evenly over the unfavored features, since the qth feature is absorbing less of it.

Let's see what happens in the extreme case. Start with the bi-level model with the first s features scaled as

$$\sqrt{\frac{\gamma d}{s}}$$

and the remaining $d - s$ features scaled as

$$\sqrt{\frac{(1-\gamma)d}{d-s}}.$$

We calculated the test error for this model in the previous homework. **What does the test error become if all $c_t, \ldots, c_{n-1}$ are reduced all the way to zero?** As in the previous homework, assume that $d \gg n$, and let $\lambda = s(1-\gamma)/(n\gamma)$, for convenience.

(Hint: reuse your results from the previous homework)

The third problem continues to further exercise skills and concepts that are being introduced this semester. Since there is no other way that you can practice these ideas ahead of the exam.

5 Junk features can have regularizing effect

We have seen previously that increasing the number of parameters of a model (e.g. the number of features that we use for linear regression) can lead to increase in sensitivity to noise (recall the problem "Learning 1d function" from HW1), and one needs to introduce regularization to cope with it. On the other hand, in HW4 we observed that kernel methods can sometimes learn a noisy function with very small amount of regularization, despite the number of implicit features being infinite! The reason is that not only the number of features is important, but also how those features are weighted.

In the previous homework, you saw a simple 1d function model involving Fourier features and regularly-spaced training data. You saw how you extra features can have a regularizing effect in what we called the bi-level scaling model where the true pattern was a low-frequency complex exponential chosen from one of s favored features that had a high scaling and there were lots of high frequency complex exponentials around that had lower scalings but acted as perfect aliases for the their low-frequency counterparts.

The previous problem in this homework gave you a hint that the dependence on the details of the bi-level scaling is not so strict for the essential results to continue holding. However, you might believe that the complex exponentials are a very peculiar set of features and that perfectly regularly spaced samples are a very special case for the sampling matrix. This problem is designed to help you see how a similar story about the regularizing effect of extra features holds more generally as well as how we need to have sufficiently loud features where the actual pattern resides for us to be able to avoid over-regularization.

In machine-learning, the "signal-processing" intuition provided by Fourier features often carries over to Gaussian features at the expense of some technical difficulties. Different frequencies correspond to independent Gaussian features, and the scalings of the Fourier features correspond to the standard deviations on those Gaussians. Because all jointly Gaussian random variables are just a unitary transformation away from being independent scaled Gaussians, the story is thus completely generic.

Consequently, in this problem we are going to explore how weights of the features influence our prediction error in a simple spiked covariance model: imagine that our data comes from a multivariate gaussian distribution with covariance matrix

$$\Sigma = \mathrm{diag}(\underbrace{\lambda_L, \ldots, \lambda_L}_{k \text{ times}}, \underbrace{\lambda_S, \ldots, \lambda_S}_{p \text{ times}}),$$

where $k \ll n \ll p$, $\lambda_S \ll \lambda_L$ (L stands for "large" and S — for "small"), and n is the number of data points. In words, the dimension of our data is much larger than the number of samples that we have. However, most of the signal comes from the subspace of dimension k, which is much smaller than n. Notice that k here is playing the role of s in the bi-level weighting for Fourier features and p here is essentially playing the role of $d = Mn$ in the bi-level weighting for Fourier features.

Notation: due to the structure of the covariance matrix above, we will often separate the coordinates that correspond to the spiked part (the first k coordinates) from the coordinates that correspond to the non-spiked part. When we do that we use "programming notation" from Python to denote such things: if $\mathbf{A}$ is a matrix, $\mathbf{A}[:, 0:k]$ denotes the matrix comprised from the first k columns of $\mathbf{A}$, and $\mathbf{A}[:, k:p+k]$ denotes the matrix comprised from the columns of $\mathbf{A}$ starting from $k+1$-st and ending p-th. Analogously, if $\mathbf{a}$ is a vector, $\mathbf{a}[0:k]$ means a vector in $\mathbb{R}^k$ comprised of the first k components of $\mathbf{a}$, and $\mathbf{a}[k:p+k]$ — a vector in $\mathbb{R}^p$ comprised of components from $k+1$-st to p-th.

(a) Before even thinking about our linear regression, let's understand how we are going to measure it's performance. Suppose that the true value for any point $\mathbf{x}$ is given by $\mathbf{x}^\top \mathbf{w}^*$, where $\mathbf{w}^*$ is the vector of true parameters. **Show that for any $\hat{\mathbf{w}}$ and $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \Sigma)$ which is independent of $\hat{\mathbf{w}}$ we have**

$$\begin{aligned} R(\hat{\mathbf{w}}, \mathbf{w}^*) :=& \mathbb{E}[(\mathbf{x}^\top \hat{\mathbf{w}} - \mathbf{x}^\top \mathbf{w}^*)^2 | \hat{\mathbf{w}}] \\ =& \left\| \sqrt{\Sigma}(\hat{\mathbf{w}} - \mathbf{w}^*) \right\|_2^2 \\ =& \lambda_L \|(\hat{\mathbf{w}} - \mathbf{w}^*)[0:k]\|_2^2 + \lambda_S \|(\hat{\mathbf{w}} - \mathbf{w}^*)[k:p+k]\|_2^2, \end{aligned}$$

where we defined our notion of risk (expected test error) in the first line.

Comment on how this measure of error is qualitatively different from $\|\hat{\mathbf{w}} - \mathbf{w}^*\|_2^2$. (which components do you need to estimate more accurately? What is the connection between this and the α and β weights that you saw in the previous problem?)

(b) Recall that interpolating min norm solution is given by

$$\hat{\mathbf{w}} = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top)^{-1} \mathbf{y}. \tag{3}$$

To simplify the computation a bit more, we will assume that the true vector $\mathbf{w}^*$ has all zero coordinates starting from $k+1$-st (i.e. $\mathbf{w}^*[k:p+k] = \mathbf{0}$): that is $\mathbf{y} = \mathbf{X}[:, 0:k]\mathbf{w}^*[0:k] + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$ — independent of $\mathbf{X}$ (i.e. our true signal only lives in the spiked part, and the independent noise is i.i.d. centered gaussian with variance σ^2).

Since $\hat{\mathbf{w}}$ is linear in $\mathbf{y}$, we can split it into two components:

$$\hat{\mathbf{w}} = \hat{\mathbf{w}}_B + \hat{\mathbf{w}}_V,$$

where

$$\begin{aligned} \hat{\mathbf{w}}_B &= \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top)^{-1} \mathbf{X}[:, 0:k]\mathbf{w}^*[0:k], \\ \hat{\mathbf{w}}_V &= \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top)^{-1} \boldsymbol{\varepsilon}. \end{aligned}$$

Here B and V correspond to "Bias" and "Variance". Here, we follow traditional terminology in which any impact of the learning rule on the squared error that we face even when there is no training noise is called "bias" and the impact of training noise is called "variance."

Denote expectation in $\boldsymbol{\varepsilon}$ as $\mathbb{E}_{\boldsymbol{\varepsilon}}$.

Look at the following decomposition of the error:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\varepsilon}} R(\hat{\mathbf{w}}, \mathbf{w}^*) =& \| \sqrt{\Sigma}(\hat{\mathbf{w}}_B - \mathbf{w}^*)\|_2^2 + \mathbb{E}_{\boldsymbol{\varepsilon}} \| \sqrt{\Sigma}\hat{\mathbf{w}}_V\|_2^2 \\ =& \lambda_L \|(\mathbf{X}[:, 0:k]^\top (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}[:, 0:k] - \mathbf{I}_k)\mathbf{w}^*[0:k]\|_2^2 \\ &+ \lambda_S \|\mathbf{X}[:, k:p+k]^\top (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}[:, 0:k]\mathbf{w}^*[0:k]\|_2^2 \\ &+ \sigma^2 \lambda_L \|\mathbf{X}[:, 0:k]^\top (\mathbf{X}\mathbf{X}^\top)^{-1}\|_F^2 \\ &+ \sigma^2 \lambda_S \|\mathbf{X}[:, k:p+k]^\top (\mathbf{X}\mathbf{X}^\top)^{-1}\|_F^2.\end{aligned}$$

How would you interpret each component of the error? (Which comes from the noise and which from the bias? Which corresponds to the error in the estimation of the first k coordinates of $\mathbf{w}^*$ (i.e. the spiked part), and which to contamination that comes from learning non-zero coefficients in the non-spiked part?)

Show that this decomposition is indeed correct.

HINT: show that if $\mathbf{A}$ *is independent from* $\boldsymbol{\varepsilon}$*, then* $\mathbb{E}_{\boldsymbol{\varepsilon}}\|\mathbf{A}\boldsymbol{\varepsilon}\|_2^2 = \sigma^2 Trace\left(\mathbf{A}^\top\mathbf{A}\right) = \sigma^2\|\mathbf{A}\|_F^2$.

(c) Now that we know what we are looking for, let's start with evaluating how well linear regression predicts values on unobserved data. If our training data vectors were i.i.d. Gaussian from the test distribution itself, the data matrix would be $\mathbf{X} = \left[\sqrt{\lambda_L}\mathbf{Z}[:, 0:k],\ \sqrt{\lambda_S}\mathbf{Z}[:, k:p+k]\right]$, where $\mathbf{Z} \in \mathbb{R}^{n\times(k+p)}$ is a matrix with i.i.d. standard normal entries. However, considering such random matrix would add quite some technical work to the argument that we are going to make, so we'll want to make a simplifying assumption. But what should that assumption be?

Here, we recall what was going on in the regularly spaced training data for complex exponential features. Notice that there, we had M aliasing groups. For the favored features, the columns in the un-scaled training matrix were orthogonal and each had norm squared equal to n. For Gaussians, the counterpart statement would be to have $\mathbf{Z}[:, 0:k]^\top\mathbf{Z}[:, 0:k]$ equal to n times the $k \times k$ identity matrix. After all, orthogonality is like independence and the sum of the variances is like the norm squared.

But what was going on for the un-favored features? There were lots of exact aliases here. The key is to observe that if we ignore the special zone from s to n (which the previous problem tells us really wouldn't matter qualitatively even if we gave it zero weight), that although columns kept repeating in the tail, the *n rows* there were orthogonal to each other (they inherited that orthogonality from the standard DFT matrix) and each row basically had a norm squared of Mn. For Gaussians, the counterpart statement would be to have $\mathbf{Z}[:, k:p+k]\mathbf{Z}[:, k:p+k]^\top$ equal to p times the $n \times n$ identity matrix. This is because p is playing the role of Mn here.

So, we know what to wish for. But is this wish reasonable, and can we do anything with it?

Recall that in problem 5 ("Isotropic gaussians") of HW3 we saw that a $n \times d$ matrix with i.i.d. standard normal entries has all its non-zero singular values concentrate in the vicinity of $\sqrt{d}$ if $d \gg n$. For our case that means $\mathbf{Z}[:, 0:k]^\top\mathbf{Z}[:, 0:k] \approx n\mathbf{I}_k$ and $\mathbf{Z}[:, k:p+k]\mathbf{Z}[:, k:p+k]^\top \approx p\mathbf{I}_n$. Since for isotropic gaussians as far as singular values go, it doesn't matter if a matrix very

tall or very wide. Singular values don't care about transposing. This means that our wish is indeed reasonable.

To make life simpler and to avoid dealing with technicalities, in this problem we **assume that matrix $\mathbf{X} \in \mathbb{R}^{n\times(k+p)}$ is such that those approximate equalities are actually exact equalities, i.e.**

$$\mathbf{X}[:,0:k]^\top \mathbf{X}[:,0:k] = n\lambda_L \mathbf{I}_k \text{ and } \mathbf{X}[:,k:p+k]\mathbf{X}[:,k:p+k]^\top = p\lambda_S \mathbf{I}_n. \tag{4}$$

(Notice that these conditions are effectively satisfied for the Fourier training matrix from the previous homework. That can help you check your work.)

It may look tricky from the first glance to deal with the inverse matrix in (3). We got away with things in the Fourier case by really leveraging the detailed structure of the exact aliases. Here, we can't do that and have to stick to more linear-algebraic arguments. Luckily, under our assumptions, this matrix becomes easily tractable. Recall that $\mathbf{X}$ consists of two parts: the first k columns form a spiked part, and we may view their contribution as a low rank correction when we look at $\mathbf{X}\mathbf{X}^\top$. There is a tool just for this case: an inverse of a low-rank correction of a matrix can be computed via the Woodbury matrix identity (see this link to wikipedia). **Use the Woodbury matrix identity to show that**

$$(\mathbf{X}\mathbf{X}^\top)^{-1} = \frac{1}{p\lambda_S}\left(\mathbf{I}_n - \frac{1}{p\lambda_S + n\lambda_L}\mathbf{X}[:,0:k]\mathbf{X}[:,0:k]^\top\right). \tag{5}$$

HINT: $\mathbf{X}\mathbf{X}^\top = \mathbf{X}[:,0:k]\mathbf{X}[:,0:k]^\top + \mathbf{X}[:,k:p+k]\mathbf{X}[:,k:p+k]^\top$.

(d) In the next part we will put everything together to obtain the final theoretical result. This part introduces one small computation that will be useful. **Use (5) to show that**

$$(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}[:,0:k] = \frac{1}{p\lambda_S + n\lambda_L}\mathbf{X}[:,0:k]. \tag{6}$$

(e) **Plug (4) , (5) and (6) in the decomposition of $\mathbb{E}_\varepsilon R(\hat{\mathbf{w}}, \mathbf{w}^*)$ into 4 terms from part b to express each of those terms in terms of $\|\mathbf{w}^*\|_2$, λ_S, λ_L, n, k and p.**

HINT 1: this is a lengthy but simple computation. In the next part you will check your result numerically.

HINT 2: $\|\mathbf{a}\|_2^2 = \mathbf{a}^\top\mathbf{a}$ *and* $\|\mathbf{A}\|_F^2 = Trace\left(\mathbf{A}\mathbf{A}^\top\right) = Trace\left(\mathbf{A}^\top\mathbf{A}\right)$.

HINT 3: show that $\|\mathbf{X}[:,k:p+k]^\top\mathbf{A}\|_F^2 = p\lambda_S\|\mathbf{A}\|_F^2$ *for any* $\mathbf{A}$.

(f) In the bi-level model with Fourier features, we saw that there were also effectively four terms to the error. **Comment on the relationship between those terms and the ones here.**

(g) Finally, we can compare our theoretical findings with the results of numerical experiments. **Do the tasks from the associated jupyter notebook and report your results.**

The remaining problems are curated problems from past midterms and we have provided the very rough approximate times that we expect these to take to be solved in an exam setting, where you experience stress and random confusion. In an exam setting, you are however expected to have your full and complete attention on the problem in front of you and are expected to be committed to making progress. In homeworks, much time is sometimes lost to just sitting there and pondering whether the next step you want to take is the right one, etc... On an exam, you need to YOLO it and move forward. Because these times were computed expecting students to already have fully studied, your times might be slower at first try. Don't panic, but do make sure that you can redo the problems in less than the given times once you've figured out how to solve them.

6 Finding the Centroid

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$. We consider computing the centroid of this dataset. Consider the loss function

$$\mathcal{L}(\mathbf{w}) := \frac{1}{2n} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{w}\|_2^2.$$

(a) (Est.: 5 minutes) First, we compute the gradient of the loss function. **Show that**

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \mathbf{w} - \bar{\mathbf{x}},$$

where $\bar{\mathbf{x}} := \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$.

(b) (Est.: 5 minutes) **Show that the minimizer of the loss function is given by $\bar{\mathbf{x}}$, i.e.** $\arg\min_{\mathbf{w} \in \mathbb{R}^d} \mathcal{L}(\mathbf{w}) = \bar{\mathbf{x}}$. Make sure to justify your answer.

(c) (Est.: 10 minutes) Suppose $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are identically and independently distributed according to a normal distribution with mean $\mathbf{x}_*$ and diagonal covariance, i.e. $\mathbf{x}_i \sim \mathcal{N}(\mathbf{x}_*, \sigma^2 \mathbf{I}_d)$ for $i = 1, \ldots, n$.
Calculate $\mathbb{E}[\|\bar{\mathbf{x}} - \mathbf{x}_*\|_2^2]$.

7 Checking Kernels

Recall that for a function k to be a valid kernel, it must be symmetric in its arguments and its Gram matrices must be positive semi-definite. More precisely, for every sample $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, the Gram matrix

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & k(\mathbf{x}_i, \mathbf{x}_j) & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

must be positive semi-definite. Also, recall that a matrix is positive semi-definite if it is symmetric and all its eigenvalues are non-negative.

(a) (Est.: 5 minutes) **Give an example of two positive semi-definite matrices A_1 and A_2 in $\mathbb{R}^{2\times 2}$ such that $A_1 - A_2$ is not positive semi-definite.**

As a consequence, **show that the function k defined by $k(\mathbf{x}_i, \mathbf{x}_j) = k_1(\mathbf{x}_i, \mathbf{x}_j) - k_2(\mathbf{x}_i, \mathbf{x}_j)$ is not necessarily a kernel even when k_1 and k_2 are valid kernels.**

(b) (Est.: 5 minutes) **Show that the function k defined by $k(\mathbf{x}_i, \mathbf{x}_j) = \left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2^2$ is not a valid kernel.**

8 Bias-Variance for Ridge Regression (Est.: 24 minutes total)

Consider the scalar data-generation model:

$$Y = xw^* + Z$$

where x denotes the scalar input feature, Y denotes the scalar noisy measurement, $Z \sim \mathcal{N}(0, 1)$ is standard unit-variance zero-mean Gaussian noise, and w^* denotes the true generating parameter that we would like to estimate.

We are given a set of n training samples $\{x_i, y_i\}_{i=1}^n$ that are generated by the above model with i.i.d. Z_i and distinct x_i. Our goal is to fit a linear model and get an estimate $\widehat{w}$ for the true parameter w^*. For all parts, assume that x_i's are given and fixed (not random).

For a given training set $\{x_i, y_i\}_{i=1}^n$, the ridge-regression estimate for w^* is defined by

$$\widehat{w}_\lambda = \arg\min_{w \in \mathbb{R}} \lambda w^2 + \sum_{i=1}^{n} (y_i - x_i w)^2 \qquad \text{with } \lambda \geq 0.$$

For the rest of the problem, assume that this has been solved and written in the form:

$$\widehat{w}_\lambda = \frac{S_{xy}}{s_x^2 + \lambda} \tag{7}$$

where $S_{xy} = \sum_{i=1}^n x_i Y_i$ and $s_x^2 = \sum_{i=1}^n x_i^2$.

(This is given, no need to rederive it).

(a) (Est.: 8 minutes) **Compute the squared-bias of the ridge estimate $\widehat{w}_\lambda$ defined as follows**

$$\text{Bias}^2(\widehat{w}_\lambda) = (\mathbb{E}[\widehat{w}_\lambda] - w^*)^2. \tag{8}$$

It is fine if your answer depends on w^* or s_x, but it should not depend directly or indirectly on the realizations of the random Z noise. (So, no S_{xy} allowed.)

Hint: First compute the expectation of the estimate $\widehat{w}_\lambda$ over the noises Z in the observation.

(b) (Est.: 8 minutes) **Compute the variance of the estimate $\widehat{w}_\lambda$ which is defined as**

$$\text{Var}(\widehat{w}_\lambda) = \mathbb{E}[(\widehat{w}_\lambda - \mathbb{E}[\widehat{w}_\lambda])^2]. \tag{9}$$

Hint: It might be useful to write $\widehat{w}_\lambda = \mathbb{E}[\widehat{w}_\lambda] + R$ for some random variable R.

(c) (Est.: 8 minutes) **Describe how the squared-bias and variance of the estimate $\widehat{w}_\lambda$ change as we change the value of λ? What happens as $\lambda \to 0$? $\lambda \to \infty$? Is the bias increasing or decreasing? Is the variance increasing or decreasing? In what sense is there a bias/variance tradeoff?**

9 Hospital (25 min)

You work at hospital A. Your hospital has collected patient data to build a model to predict who is likely to get sepsis (a bad outcome). Specifically, the data set contains the feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, and associated real number labels $\mathbf{y} \in \mathbb{R}^n$, where n is the number of patients you are learning from and d is the number of features describing each patient. You plan to fit a linear regression model $\widehat{y} = \mathbf{w}^\top \mathbf{x}$ that will enable you to predict a label for future, unseen patients (using their feature vectors).

However, your hospital has only started collecting data a short time ago. Consequently the model you fit is not likely to be particularly accurate. Hospital B has exactly the same set up as your hospital (i.e., their patients are drawn from the same distribution as yours and they have the same measurement tools). For privacy reasons, Hospital B will not share their data. However, they tell you that they have trained a linear model on their own sepsis-relevant data: ($\mathbf{X}_B$ and $\mathbf{y}_B$) and are willing to share their learned model $\widehat{y} = \widehat{\mathbf{w}}_B^\top \mathbf{x}$ with you. In particular, Hospital B shares their entire Gaussian posterior distribution on $\mathbf{w}$ with you: $\mathcal{N}(\widehat{\mathbf{w}}_B, \mathbf{\Psi})$.

(a) (10 min) Assume that we use the posterior from Hospital B as our own prior distribution for $\mathbf{w} \sim \mathcal{N}(\widehat{\mathbf{w}}_B, \mathbf{\Psi})$. Suppose that our Hospital A model is given by $\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$, where the noise, $\boldsymbol{\epsilon}$, has an assumed distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. **Derive the MAP estimate $\widehat{\mathbf{w}}$ for w using Hospital A's data X, y and the prior information from Hospital B.**

HINT: Recall that traditional ridge regression could be derived from a MAP perspective, where the parameter $\mathbf{w}$ *has a zero mean Gaussian prior distribution with a scaled identity covariance. How could you use reparameterization (i.e. change of variables) for the problem here?*

(b) (15 min) Now, for simplicity, consider $d = 1$ so that the w is a scalar parameter. Suppose that instead of giving you their posterior distribution, Hospital B only gave you their mean $\widehat{w}_B$. How can you use this information to help fit your model? **Describe in detail how you should use your own hospital's patient data and combine it with the mean $\widehat{w}_B$ from Hospital B in a procedure to find your own $\widehat{w}$ for predicting sepsis in Hospital A.**

Hint 1: You might want to consider introducing an appropriate hyperparameter and doing what you usually do with hyperparameters.

Hint 2: What does the λ hyperparameter in ridge-regression correspond to from a probabilistic perspective?

10 Ridge regression vs. PCA (Est.: 24 minutes total)

Assume we are given n training data points $(\mathbf{x}_i, y_i)$. We collect the target values into $\mathbf{y} \in \mathbb{R}^n$, and the inputs into the matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ where the rows are the $d-$dimensional feature vectors $\mathbf{x}_i^\top$ corresponding to each training point. Furthermore, assume that $\frac{1}{n}\sum_{i=1}^n \mathbf{x_i} = \mathbf{0}$, $n > d$ and $\mathbf{X}$ has rank d.

In this problem we want to compare two procedures: The first is ridge regression with hyperparameter λ, while the second is applying ordinary least squares after using PCA to reduce the feature

dimension from d to k (we give this latter approach the short-hand name k-PCA-OLS where k is the hyperparameter).

Notation: The singular value decomposition of $\mathbf{X}$ reads $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ where $\mathbf{U} \in \mathbb{R}^{n\times n}$, $\mathbf{\Sigma} \in \mathbb{R}^{n\times d}$ and $\mathbf{V} \in \mathbb{R}^{d\times d}$. We denote by $\mathbf{u}_i$ the n-dimensional column vectors of $\mathbf{U}$ and by $\mathbf{v}_i$ the $d-$dimensional column vectors of $\mathbf{V}$. Furthermore the diagonal entries $\sigma_i = \Sigma_{i,i}$ of $\mathbf{\Sigma}$ satisfy $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d > 0$. For notational convenience, assume that $\sigma_i = 0$ for $i > d$.

(a) (Est.: 6 minutes) It turns out that the ridge regression optimizer (with $\lambda > 0$) in the $\mathbf{V}$-transformed coordinates

$$\widehat{\mathbf{w}}_{\text{ridge}} = \arg\min_{\mathbf{w}} \|\mathbf{X}\mathbf{V}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

has the following expression:

$$\widehat{\mathbf{w}}_{\text{ridge}} = \text{diag}(\frac{\sigma_i}{\lambda + \sigma_i^2})\mathbf{U}^\top\mathbf{y}. \tag{10}$$

Use $\widehat{y}_{test} = \mathbf{x}_{test}^\top\mathbf{V}\widehat{\mathbf{w}}_{\text{ridge}}$ to denote the resulting prediction for a hypothetical $\mathbf{x}_{test}$. Using (10) and the appropriate $\{\beta_i\}$, this can be written as:

$$\widehat{y}_{test} = \mathbf{x}_{test}^\top \sum_{i=1}^{d} \mathbf{v}_i\beta_i\mathbf{u}_i^\top\mathbf{y}. \tag{11}$$

What are the β_i for this to correspond to (10) from ridge regression?

(b) (Est.: 12 minutes) Suppose that we do k-PCA-OLS — i.e. ordinary least squares on the reduced k-dimensional feature space obtained by projecting the raw feature vectors onto the $k < d$ principal components of the covariance matrix $\mathbf{X}^\top\mathbf{X}$. Use $\widehat{y}_{test}$ to denote the resulting prediction for a hypothetical $\mathbf{x}_{test}$,

It turns out that the learned k-PCA-OLS predictor can be written as:

$$\widehat{y}_{test} = \mathbf{x}_{test}^\top \sum_{i=1}^{d} \mathbf{v}_i\beta_i\mathbf{u}_i^\top\mathbf{y}. \tag{12}$$

Give the β_i coefficients for k-PCA-OLS. Show work.

Hint 1: some of these β_i will be zero. Also, if you want to use the compact form of the SVD, feel free to do so if that speeds up your derivation.

Hint 2: some inspiration may be possible by looking at the next part for an implicit clue as to what the answer might be.

(c) (Est.: 6 minutes) For the following part, $d = 5$. The following $\boldsymbol{\beta} := (\beta_1, \ldots, \beta_5)$ (written out to two significant figures) are the results of OLS (i.e. what we would get from ridge regression in the limit $\lambda \to 0$), λ-ridge-regression, and k-PCA-OLS for some $\mathbf{X}, \mathbf{y}$ (identical for each method) and $\lambda = 1, k = 3$. **Write down which procedure was used for each of the three sub-parts below.**

We hope this helps you intuitively see the connection between these three methods.

Hint: It is not necessary to find the singular values of $\mathbf{X}$ *explicitly, or to do any numerical computations at all.*

(i) $\boldsymbol{\beta} = (0.01, 0.1, 0.5, 0.1, 0.01)$

(ii) $\boldsymbol{\beta} = (0.01, 0.1, 1, 0, 0)$

(iii) $\boldsymbol{\beta} = (0.01, 0.1, 1, 10, 100)$

11 Nearest Neighbors Generalize (Est.: 30 minutes)

In this question, we consider whether and why nearest-neighbor approaches generalize from the training set to test sets, assuming that both were drawn from the same distribution.

For simplicity, we set $k = 1$ and focus on 1-nearest neighbors. The algorithm operates on a training dataset $S = \{z_1, z_2, \ldots, z_n\}$ of pairs $z_i = (\mathbf{x}_i, y_i)$ consisting of features $\mathbf{x}_i \in \mathbb{R}^d$ and labels $y_i \in \{0, 1\}$. Each point z_i is drawn i.i.d. from a distribution $\mathcal{D}$. For the entire data set, this is written $S \sim \mathcal{D}^n$.

The 1-NN algorithm computes a prediction $f_S(\mathbf{x})$ for input $\mathbf{x}$ by finding the $\mathbf{x}_i \in S$ that is closest to $\mathbf{x}$, and using y_i as the prediction.

The overall goal of this question is to bound the generalization error of 1-NN:

$$L_{\mathcal{D}}(f_S) = \mathbb{E}_{z\sim\mathcal{D}}[\ell(f_S, z)]$$

where ℓ is the classification error for z. (i.e. $\ell(f, z) = 0$ if $f(\mathbf{x}) = y$ and $\ell(f, z) = 1$ if $f(\mathbf{x}) \neq y$).

We want to say that the generalization error is not that much bigger on average than the leave-one-out error (i.e. n-fold cross-validation error) given by

$$L_{\text{loo}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_{S^{(i)}}, z_i)$$

where $S^{(i)}$ is defined as the dataset with the i-th point dropped, i.e.

$$S^{(i)} = \{z_1, z_2, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n\}.$$

(a) (Est.: 3 minutes)**What is the value for the training error for 1-NN classification? (i.e you run the training data through the trained 1-NN classifier and compute the classification error)**

(b) (Est.: 11 minutes) **Show that** $\mathbb{E}_{z\sim\mathcal{D}}[|\ell(f_{S^{(i)}}, z) - \ell(f_S, z)|]$ **is less than or equal to the probability that a random point** $z = (\mathbf{x}, y)$ **drawn according to** $\mathcal{D}$ **has its** $\mathbf{x}$ **closer to** $\mathbf{x}_i$ **than to any other point** $\mathbf{x}_j \in S^{(i)}$**.**

Hint: focus your attention separately on these two kinds of potential points, $z = (\mathbf{x}, y)$*: (i) those that have their* $\mathbf{x}$ *closer to* $\mathbf{x}_i$ *than to any other point* $\mathbf{x}_j \in S^{(i)}$*, and (ii) those for which some other point in* $S^{(i)}$ *is their nearest neighbor in* S*. Does the addition or removal of the point* $\mathbf{x}_i$ *have any effect on the classification decision for the latter set of points?*

(c) (Est.: 11 minutes) Build on the previous part to **show**

$$\mathbb{E}_{S\sim\mathcal{D}^n, z\sim\mathcal{D}}[|\ell(f_{S^{(i)}}, z) - \ell(f_S, z)|] \leq \frac{1}{n}.$$

In other words, when averaged over both the possible training data and the possible test points, the average generalization error of 1-NN is on average less than $\frac{1}{n}$ away from the average leave-one-out cross-validation error.

Hint: Show that $\frac{1}{n}$ is the average probability of a point having the i-th random training point $\mathbf{x}_i$ as its nearest neighbor in the random training set S.

Hint for Hint: Is there anything special about the i-th training point or can you somehow leverage the symmetry of the i.i.d. drawing of the training set S?

(d) (Est.: 5 minutes) For this part, we give you that with probability $1 - \delta$ over the draw of the training set $S \sim \mathcal{D}^n$

$$L_{\mathcal{D}}(f_S) \leq L_{\text{loo}}(f_S) + \sqrt{\frac{7}{2\delta n}}$$

holds. (This can be shown with tedious algebra looking at the expected squared difference of the two errors and using Markov's inequality. For the exam, we are not making you show this.)

On a given training set of size $n = 350000$, we measure 3% average leave-one-out cross-validation error $L_{\text{loo}}(f)$ for the 1-nearest-neighbor classifier. **What can you say with** 90% **confidence about the probability of classification error on a test point drawn from the same distribution that the training points were drawn from?**

This last problem is an example of multiple-choice problems. However, since the time of this exam, our understanding of the material and what we have taught you has improved. Consequently, some of these questions are more confusing than we would hope. As a homework problem, this problem doesn't count. However, it is included so that you can be familiar with how multiple choice problems might work. You can rest assured that our grading philosophy is not a "gotcha" oriented one — we give partial credit fairly even for multiple choice problems when we notice that some answers can kind of be correct even though others are more correct.

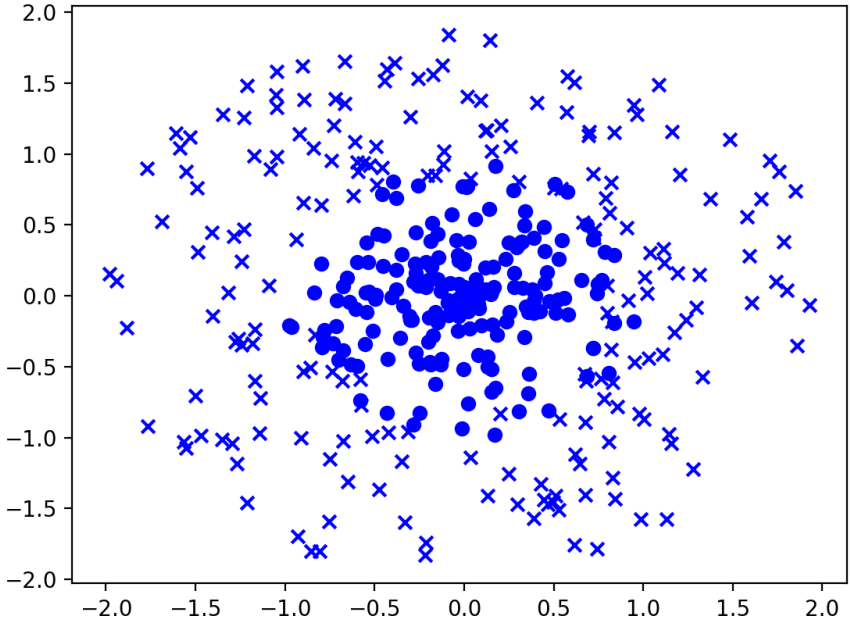
12 Multiple Choice Questions (Est.: 15 minutes)

For these questions, select **all** the answers which are correct. You will get full credit for selecting all the right answers.

(a) Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ with $n \geq d$. Suppose $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ is the singular value decomposition of $\mathbf{X}$ where $\sigma_i = \Sigma_{i,i}$ are the diagonal entries of $\mathbf{\Sigma}$ and satisfy $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d$ while $\mathbf{u}_i$ and $\mathbf{v}_i$ are the ith columns of $\mathbf{U}$ and $\mathbf{V}$ respectively. **Which of the following is the rank k approximation to $\mathbf{X}$ that is best in the Froebenius norm.** That is, which low rank approximation, $\mathbf{X}_k$, for $\mathbf{X}$ yields the lowest value for $\|\mathbf{X} - \mathbf{X}_k\|_F^2$?

○ $\sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_{n-i}^\top$ ○ $\sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$ ○ $\sum_{i=d-k+1}^{d} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$ ○ $\sum_{i=1}^{k} \sigma_i \mathbf{u}_{n-i} \mathbf{v}_i^\top$

(b) Consider a simple dataset of points $(x_i, y_i) \in \mathbb{R}^2$, each associated with a label b_i which is -1 or $+1$. The dataset was generated by sampling data points with label -1 from a disk of radius 1.0 (shown as filled circles in the figure) and data points with label $+1$ from a ring with inner radius 0.8 and outer radius 2.0 (shown as crosses in the figure). **Which set of polynomial features would be best for performing linear regression, assuming at least as much data as shown in the figure?**



- ◯ $1, x_i$
- ◯ $1, x_i, y_i$
- ◯ $1, x_i, y_i, x_i^2, x_i y_i, y_i^2$
- ◯ $1, x_i, y_i, x_i^2, x_i y_i, y_i^2, x_i^3, y_i^3, x_i^2 y_i, x_i y_i^2$

(c) **Which of the following is a valid kernel function** for vectors of the same length, $\mathbf{x}$ and $\mathbf{y}$?

- ◯ $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$
- ◯ $k(\mathbf{x}, \mathbf{y}) = e^{-\frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2}$
- ◯ $k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^\top \mathbf{y})^p$ for some degree p
- ◯ $k(\mathbf{x}, \mathbf{y}) = k_1(\mathbf{x}, \mathbf{y}) - k_2(\mathbf{x}, \mathbf{y})$ for valid kernels k_1 and k_2.

(d) During training of your model, both independent variables in the matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and dependent target variables $\mathbf{y} \in \mathbb{R}^n$ are corrupted by noise. At test time, the data points you are computing predictions for, $\mathbf{x}_{test}$, are noiseless. **Which method(s) should you use to estimate the value of $\hat{\mathbf{w}}$ from the training data in order to make the most accurate predictions $\mathbf{y}_{test}$ from the noiseless test input data, $\mathbf{X}_{test}$?** Assume that you make predictions using $\mathbf{y}_{test} = \mathbf{X}_{test}\hat{\mathbf{w}}$.

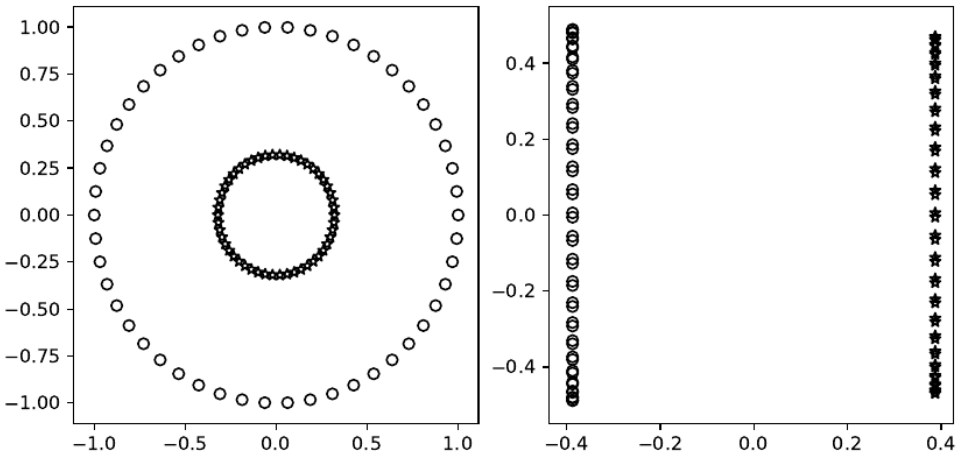
- ◯ OLS
- ◯ Ridge regression
- ◯ Weighted Least Squares
- ◯ TLS

(e) Assume you have n input data points, each with d high quality features ($\mathbf{X} \in \mathbb{R}^{n \times d}$) and associated labels ($\mathbf{y} \in \mathbb{R}^n$). Suppose that $d \gg n$ and that you want to learn a linear predictor. **Which**

of these approaches would help you to avoid overfitting?

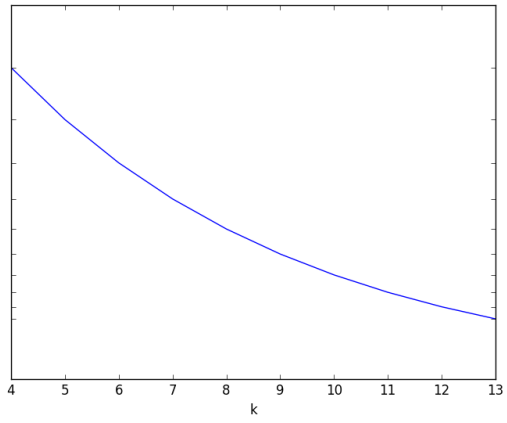
- ◯ Preprocess $\mathbf{X}$ using $k \ll n$ random projections
- ◯ Preprocess $\mathbf{X}$ using PCA with $k \ll n$ components.
- ◯ Preprocess $\mathbf{X}$ using PCA with n components.
- ◯ Add polynomial features
- ◯ Use a kernel approach
- ◯ Add a ridge penalty to OLS
- ◯ Do weighted least squares

(f) **Which methods could yield a transformation to go from the two-dimensional data on the left to the two-dimensional data on the right?**



- ◯ Random projections
- ◯ PCA
- ◯ Use of a kernel
- ◯ Adding polynomial features

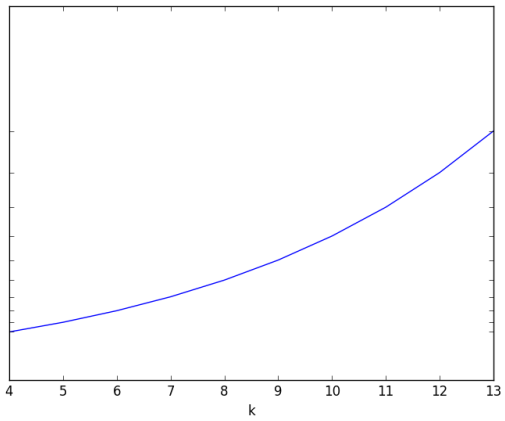
(g) Your friend is training a machine learning model to predict disease severity based on k different health indicators. She generates the following plot, where the value of k is on the x axis.



Which of these might the y axis represent?

- ◯ Training Error
- ◯ Validation Error
- ◯ Bias
- ◯ Variance

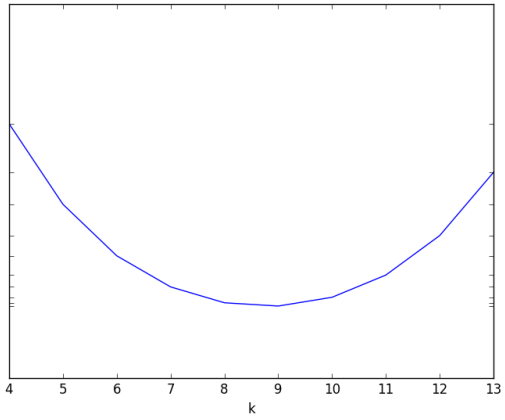
(h) Your friend is training a machine learning model to predict disease severity based on k different health indicators. She generates the following plot, where the value of k is on the x axis.



Which of these might the y axis represent?

- ◯ Training Error
- ◯ Validation Error
- ◯ Bias
- ◯ Variance

(i) Your friend is training a machine learning model to predict disease severity based on k different health indicators. She generates the following plot, where the value of k is on the x axis.



Which of these might the y axis represent?

- ◯ Training Error
- ◯ Validation Error
- ◯ Bias
- ◯ Variance

13 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
- Alvin Won
- Anant Sahai
- Chawin Sitawarin
- Esther Rolf
- Fanny Yang
- Jane Yu
- Jennifer Listgarten
- Kate Sanders
- Khalil Sarwari
- M. J. McDonald
- Max Simchowitz
- Peter Wang
- Philipp Moritz
- Raaz Dwivedi
- Rahul Arya
- Ross Boczar
- Sarah Dean
- Satish Rao
- Vignesh Subramanian
- Yang Gao