EECS 189 Introduction to Machine Learning
Fall 2020

HW3

This homework is due **Wednesday, September 23 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 Probabilistic Model of Linear Regression

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

(a) Assume that X and Y are both one-dimensional random variables, i.e. $X, Y \in \mathbb{R}$. Assume an affine model between X and Y: $Y = Xw_1 + w_0 + Z$, where $w_1, w_0 \in \mathbb{R}$, and $Z \sim N(0, 1)$ is a

HW3, ©UCB EECS 189, Fall 2020. All Rights Reserved. This may not be publicly shared without explicit permission.

standard normal (Gaussian) random variable. Assume w_1, w_0 are fixed parameters (i.e., they are not random). **What is the conditional distribution of Y given X?**

(b) Given n points of training data $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ generated in an iid fashion by the probabilistic setting in the previous part, **derive the maximum likelihood estimator for w_1, w_0 from this training data.**

(c) Now, consider a different generative model. Let $Y = Xw + Z$, where $Z \sim U[-0.5, 0.5]$ is a continuous random variable uniformly distributed between -0.5 and 0.5. Again assume that w is a fixed parameter. **What is the conditional distribution of Y given X?**

(d) Given n points of training data $\{(X_1, Y_1), (X_2, Y_2), \cdots, (X_n, Y_n)\}$ generated in an i.i.d. fashion in the setting of the part (c) **derive a maximum likelihood estimator of w.** Assume that $X_i > 0$ for all $i = 1, \ldots, n$. (Note that MLE for this case need not be unique; but you are required to report only one particular estimate.)

(e) Take the model $Y = Xw + Z$, where $Z \sim U[-0.5, 0.5]$. **In Jupyter Notebook, simulate n training samples $\{(X_1, Y_1), (X_2, Y_2), \cdots, (X_n, Y_n)\}$ and illustrate what the likelihood of the data looks like as a function of w after $n = 5, 25, 125, 625$ training samples. Qualitatively describe what is happening as n gets large.**

(You may use the starter code provided in Jupyter Notebook. Note that you have considerable design freedom in this problem part. You get to choose how you draw the X_i as well as what true value w you want to illustrate. You have total freedom in using additional python libraries for this problem part. No restrictions.)

(f) Consider n training data points $\{(\mathbf{x}_1, Y_1), (\mathbf{x}_2, Y_2), \cdots, (\mathbf{x}_n, Y_n)\}$ generated according to $Y_i = \mathbf{w}^\top \mathbf{x}_i + Z_i$ where $Y_i \in \mathbb{R}, \mathbf{w}, \mathbf{x}_i \in \mathbb{R}^d$ with $\mathbf{w}$ fixed, and $Z_i \sim N(0, 1)$ iid standard normal random variables. **Argue why the maximum likelihood estimator for $\mathbf{w}$ is the solution to a least squares problem.**

(g) (Multi-dimensional ridge regression) Consider the setup of the previous part: $Y_i = \mathbf{w}^\top \mathbf{x}_i + Z_i$, where $Y_i \in \mathbb{R}, \mathbf{w}, \mathbf{x}_i \in \mathbb{R}^d$, and $Z_i \sim N(0, 1)$ iid standard normal random variables. Now we treat $\mathbf{w}$ as a random vector and assume a prior knowledge about its distribution. In particular, we use the prior information that the random variables W_j are i.i.d. $\sim N(0, \sigma^2)$ for $j = 1, 2, \ldots, d$. **Derive the posterior distribution of $\mathbf{w}$ given all the $\mathbf{x}_i, Y_i$ pairs. What is the mean of the posterior distribution of the random vector $\mathbf{w}$?**

Hint: Compute the posterior up-to proportionality, i.e. you may discard terms that do not depend on $\mathbf{w}$ to simplify the algebra. After a few steps, you should be able to identify the family of the distribution of the posterior. Then, you can determine the mean and the variance by completing the square. We find that it is simpler to work in matrix and vector format: $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}$ and $\mathbf{Y} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix}$

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

(Again, you may use the starter code. And like problem (e), there are no restrictions for using additional python libraries for this part as well.)

3 Estimation and approximation in linear regression

In typical applications, we are dealing with data generated by an *unknown* function (with some noise), and our goal is to estimate this function. So far we used linear and polynomial regressions. In this problem we will build towards being able ot explore the quality of polynomial regressions when the true function is not actually polynomial. Notice that this is reflective of the typical case in machine learning applications — our feature models are rarely exactly correct for the actual underlying patterns that we are trying to learn.

Suppose we are given a full column rank feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and an observation vector $\mathbf{y} \in \mathbb{R}^n$. Let the vector $\mathbf{y}$ represent the noisy measurement of a true signal $\mathbf{y}^*$:

$$\mathbf{y} = \mathbf{y}^* + \mathbf{z}, \tag{1}$$

with $\mathbf{z} \in \mathbb{R}^n$ representing the random noise in the observation $\mathbf{y}$, where $z_j \sim \mathcal{N}(0, \sigma^2)$ are i.i.d. We define the vectors $\mathbf{w}^*$ and $\hat{\mathbf{w}}$ as follows:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\mathbf{y}^* - \mathbf{X}\mathbf{w}\|_2^2 \qquad \text{and} \qquad \hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2.$$

Observe that for a given true signal $\mathbf{y}^*$ the vector $\mathbf{w}^*$ is fixed, but the vector $\hat{\mathbf{w}}$ is a random variable since it is a function of the random noise $\mathbf{z}$ that we are seeing in the observations.

Note that the vector $\mathbf{X}\mathbf{w}^*$ is the best linear fit of the true signal $\mathbf{y}^*$ in the column space of $\mathbf{X}$. We will use this as the proxy for the best approximation we can hope for given our features. Similarly, the vector $\mathbf{X}\hat{\mathbf{w}}$ is the best linear fit of the observed noisy signal $\mathbf{y}$ in the column space of $\mathbf{X}$.

Instead of using either the "parameter recovery" framework to see how well we are recovering the underlying parameters, or the "prediction error" perspective based on how well that we would do on a test point drawn according to a test distribution, we will take another more training data oriented point of view in this problem. After obtaining $\hat{\mathbf{w}}$, we would like to bound the error $\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}^*\|_2^2$, which is the *denoising error* incurred based on the specific n training samples. This treats the entire universe as being these n samples and our goal is to remove the noise from them as best we can using our model. If you want, you can think of this as a problem in which the test distribution is a uniform distribution over the observed training inputs with the test distribution for the outputs being the true noise-free outputs. In a sense, the denoising problem is as easy as it gets in machine learning since we don't have to worry about any potential difference between the training samples and where we will be tested.

In this problem we will see how to get a good estimate of this denoising error. When using polynomial features, we will also learn how to decide the degree of the polynomial when trying to fit a noisy set of observations from a smooth function.

Remark:

You can use the closed form solution for OLS for all parts of this problem. For parts (a)-(c), assume that the feature matrix $\mathbf{X}$ and the true signal vector $\mathbf{y}^*$ are fixed (and not random). Furthermore, in all parts the expectation is taken over the randomness in the noise vector $\mathbf{z}$.

(a) **Show that** $\mathbb{E}[\hat{\mathbf{w}}] = \mathbf{w}^*$ and use this fact to **show that**

$$\mathbb{E}\left[\|\mathbf{y}^* - \mathbf{X}\hat{\mathbf{w}}\|_2^2\right] = \|\mathbf{y}^* - \mathbb{E}[\mathbf{X}\hat{\mathbf{w}}]\|_2^2 + \mathbb{E}\left[\|\mathbf{X}\hat{\mathbf{w}} - \mathbb{E}[\mathbf{X}\hat{\mathbf{w}}]\|_2^2\right].$$

Note that the above decomposition of the squared error corresponds to the sum of bias-squared and the variance of our estimator $\mathbf{X}\hat{\mathbf{w}}$.

Hint 1: For the first part, you can start by writing out $\hat{\mathbf{w}}$ *and* $\mathbf{w}^*$ *in a closed form.*
Hint 2: To show the second part, you might want to use the Pythagorean theorem, and you can use this fact without proof: $\mathbf{y}^* - \mathbf{X}\mathbf{w}^*$ *is orthogonal to* $\mathbf{X}\mathbf{w}^* - \mathbf{X}\hat{\mathbf{w}}$*. You can also verify this yourself.*

(b) Recall that if $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, where $\Sigma \in \mathbb{R}^{(d\times d)}$ is the covariance matrix, then for any matrix $A \in \mathbb{R}^{k\times d}$, we have $\mathbf{A}\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{A}\Sigma\mathbf{A}^T)$. Use this fact to **show that** the distribution of the vector $\hat{\mathbf{w}}$ is given by

$$\hat{\mathbf{w}} \sim \mathcal{N}(\mathbf{w}^*, \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}).$$

(c) **Show that**

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

(Hint: The trace trick: $\text{trace}(AB) = \text{trace}(BA)$*, might be useful in doing this proof if you choose one natural route. Another useful way to understand this might involve thinking about what coordinate system makes life easier.)*

Notice that the above has a very intuitive interpretation: there are n samples, but only d degrees of freedom in what we are fitting. This allows only $\frac{d}{n}$ of the noise energy to come through after we "filter" the noisy $\mathbf{y}$ in this way.

(d) Assume the underlying model is a noisy linear model with scalar samples $\{\alpha_i, y_i\}_{i=1}^n$, i.e. $y_i = w_1\alpha_i + w_0 + z_i$. We construct matrix $\mathbf{X}$ by using $D+1$ polynomial features $\mathbf{P}_D(\alpha_i) = [1, \alpha_i, \ldots, \alpha_i^D]^\top$ of the *distinct* sampling points $\{\alpha_i\}_{i=1}^n$. For any $D \geq 1$, compare with model (1) and **compute** $\mathbf{w}^*$ **for this case. Also compute the bias** ($\|\mathbf{y}^* - \mathbf{X}\mathbf{w}^*\|_2$) **for this case.** Using the previous parts of this problem, **compute the number of samples** n **required to ensure that the average expected denoising squared error is bounded by** ϵ**?** Your answer should be expressed as a function of D, σ^2, and ϵ.

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

(e) Simulate the problem from part (d) for yourself. Set $w_1 = 1$, $w_0 = 1$, and sample n points $\{\alpha_i\}_{i=1}^n$ uniformly from the interval $[-1, 1]$. Generate $y_i = w_1\alpha_i + w_0 + z_i$ with z_i representing standard Gaussian noise.

Fit a D degree polynomial to this data and show how the average error $\frac{1}{n}\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}^*\|_2^2$ scales as a function of both D and n.

You may show separate plots for the two scalings. It may also be helpful to average over multiple realizations of the noise (or to plot point clouds) so that you obtain smooth curves.

(For this part, the libraries numpy.random and numpy.polyfit might be useful. You are free to use any and all python libraries. You may use the starter code provided in Jupyter Notebook.)

(f) Now, let us be more realistic. Assume that the true pattern we are trying to learn is the noisy exponential function with scalar samples $\{\alpha_i, y_i\}_{i=1}^n$ where $y_i = e^{\alpha_i} + z_i$ with *distinct* sampling points $\{\alpha_i\}_{i=1}^n$, in the interval $[-4, 3]$ and i.i.d. Gaussian noise $z_i \sim \mathcal{N}(0, 1)$. We again construct matrix $\mathbf{X}$ by using $D + 1$ polynomial features $[1, \alpha_i, \ldots, \alpha_i^D]^\top$ and use linear regression to fit the observations. Recall, the definitions of the bias and variance of the OLS estimator from part (a) and notice that in this case, for a fixed n, as the degree D of the polynomial increases: (1) the bias decreases and (2) the variance increases. **Use bounds or approximations as needed and argue that to get a good denoising error, a reasonable choice of D is given by $O(\log n / \log\log n)$.**

You can directly use previous parts of this problem, as well as the problem on the earlier homework (Approximating a 1D function). You may assume that n and D are large for approximation purposes. Feel free to use Stirling's approximation and whatever else you need to make your life easier.

One useful heuristic for balancing a tradeoff where you have the sum of two terms — one increasing and one decreasing — is to pick the point at which the two terms are equal.

(g) Simulate the problem in the previous part yourself. Sample n points $\{\alpha_i\}_{i=1}^n$ uniformly from the interval $[-4, 3]$. Generate $y_i = e^{\alpha_i} + z_i$ with z_i representing standard Gaussian noise. **Fit a D degree polynomial to this data and show how the average error $\frac{1}{n}\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}^*\|_2^2$ scales as a function of both D and n.** You may show separate plots for the two scalings. For scaling with D, choose $n = 120$. It may also be helpful to average over multiple realizations of the noise (or to plot point clouds) so that you obtain smooth curves. (For this part, the libraries numpy.random and numpy.polyfit might be useful and you are free to use any and all python libraries.)

(h) Comment on the differences in plots obtained in part (e) and part (g).

4 Orthonormalization of features

Note: This problem has a companion Jupyter notebook. Most of the parts have a corresponding demo/visualization in the Jupyter notebook but for parts (c) and (g) you are required to fill part of the code in the notebook.

Consider the problem setting of learning a 1-dimensional function $f(x)$ from noisy samples $(x_i, y_i)_{i=1,\ldots,n}$ like in Problem 8 of HW 1. We perform linear regression in feature space with the featurization $\phi : \mathbb{R} \to \mathbb{R}^d$ to learn coefficients $\mathbf{w}$. On a given test point x, we predict

$$\hat{f}(x) = \phi(x)^\top \mathbf{w}.$$

To evaluate our learned predictor we care about its "test performance", the average performance on data coming from our test distribution. Next we will formalize this.

Let X, Y denote the random variables corresponding to the test data. Further suppose that $Y = f(X)$, i.e each randomly sampled test point comes along with its true function value (there is no additional randomness in the Y at test time due to noise). Let X have the probability density function p, i.e $X \sim p$. Thus the test data is $(X, f(X))$ and its distribution is determined by p. On the random test point X, we make the prediction $Y_{\text{pred}} = \hat{f}(X) = \phi(X)^\top \mathbf{w}$.

Note: The test data distribution can be different than the train data distribution. One such case is if the training points x_i were not sampled from the same distribution p as the test point. Even if the training points were sampled from p, the training labels y_i may not be the true function value at points x_i due to the presence of noise and thus the test and train data distributions typically vary in practice.

For this question we assume that the training points and consequently the learned coefficients $\mathbf{w}$ are fixed and not random. We define the prediction error/test error as,

$$\mathcal{E}_{\text{pred}} = \mathbb{E}[(Y_{\text{pred}} - Y)^2] = \mathbb{E}[(\phi(X)^\top \mathbf{w} - f(X)^2] = \int_x (\phi(x)^\top \mathbf{w} - f(x)^2) p(x) dx$$

Note: The expectation is over the randomness due to the test data point X. Nothing else is random at test time here. We are using squared loss here because it is convenient and has nice geometry. By changing the loss function we can have other measures of prediction performance.

(a) **Prediction error vs parameter estimation error.**

Suppose the true function can be exactly represented in the feature space, $f(X) = \phi(X)^\top \mathbf{w}^*$ for some $\mathbf{w}^* \in \mathbb{R}^d$. In this case we can define another metric of test performance, the parameter estimation error given by

$$\mathcal{E}_{\text{est}} = \mathbb{E}\left\|\mathbf{w} - \mathbf{w}^*\right\|_2^2 = \left\|\mathbf{w} - \mathbf{w}^*\right\|_2^2,$$

since neither $\mathbf{w}$ nor $\mathbf{w}^*$ vary based on the test point X. Estimation error measures how well we are estimating the paramters of the function in feature space while test error measures how close the values predicted by the learned function are to the true function values.

Show that if $\mathbb{C} = \mathbb{E}[\phi(X)\phi(X)^\top] = \mathbf{I}_d$, then $\mathcal{E}_{\text{pred}} = \mathcal{E}_{\text{est}}$. In this case, the features are orthonormal with respect to the test distribution.

(b) The entries of the "covariance" matrix are given by,

$$C_{ij} = \mathbb{E}[\phi_i(X)^\top \phi_j(X)] = \int_x \phi_i(x)^\top \phi_j(x) p(x) dx,$$

where $\phi(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_d(x)]^\top$.
Consider a 3-dimensional polynomial featurization $\phi(x) = [1, x, x^2]^\top$ and let $p = \text{Uniform}[-1, 1]$.
Calculate C. Are the features orthonormal?

(c) *Orthonormalization of features: Polynomial example* Consider the same polynomial featurization from the previous part $\phi(x) = [1, x, x^2]^\top$. **Using the Gram-Schmidt procedure, orthonormalize these features with respect to the test distribution p = Uniform$[-1, 1]$. Then complete the corresponding part in the Jupyter notebook.**

(HINT: Denote the relevant inner product between features as $\langle \phi_1, \phi_2 \rangle = \frac{1}{2}\int_{-1}^{1} \phi_1(x)\phi_2(x)dx$.)

(d) *Orthonormalization of features using the covariance.* Now suppose X has density p and we have features $\phi(x)$ such that $C = \mathbb{E}[\phi(X)\phi(X)^\top] = K$ for a positive-definite matrix K. **Show that $\widetilde{\phi}(x) = K^{-\frac{1}{2}}\phi(x)$ is a set of features orthonormal with respect to the test distribution p. If we follow this technique for the previous part do we end up with set of orthonormal features? Explain.** It may be useful to look ahead in the Jupyter notebook, to see this technique applied numerically to the previous example.

Here, you can think about $K^{\frac{1}{2}}$ as the natural symmetric matrix so that $K^{\frac{1}{2}}K^{\frac{1}{2}} = K$ — basically the matrix you get by applying the spectral theorem for real symmetric matrices to K and then just taking the square-roots of the all eigenvalues but keeping the same orthogonal eigenvectors.

Next we will see why having an orthonormal set of features with respect to the test distribution is useful.

(e) *Transforming bases.* Consider an orthonormal featurization, $\phi(x)$ with respect to test distribution p and let the true function be $f(x) = \phi(x)^\top \mathbf{w}$ with $\|w\|_2 = \alpha \neq 0$. **Show we can find a new featurization $\widetilde{\phi}(x)$ whose features are also orthonormal with respect to p such that $f(x) = \alpha\widetilde{\phi}(x)^\top \mathbf{e}_1$ where $\mathbf{e}_1$ is the unit vector with one in the first entry and zero everywhere else.**

(f) *Decomposition of prediction error.* Assume that we have a featurization $\phi(x)$ and test distribution p such that $\mathbb{E}[\phi(X)] = \mathbf{0}$. i.e. The features have zero mean. (If we wanted, we could also let one of the features be the constant 1 to pull the constant part out and everything would still be fine.) Let the features be orthonormal with resepect to the test distribution. Let the true function be $f(x) = \phi(x)^\top \mathbf{w}^*$ for $\mathbf{w}^* = w_1^* \mathbf{e}_1$. We learn the predictor $\hat{f}(X) = \phi(X)^\top \mathbf{w}$ where $\mathbf{w} = [w_1, w_2, \ldots, w_d]^\top$. Thus,

$$\hat{f}(X) = w_1\phi_1(X) + \sum_{j=1^d} w_j\phi_j(X).$$

Let S and C be random variables defined as

$$S = (w_1 - w_1^*)\phi_1(X)$$
$$C = \sum_{j=2}^{d} w_j\phi_j(X).$$

Show that S and C are uncorrrelated, i.e $\mathbb{E}[SC] = \mathbb{E}[S]\mathbb{E}[C]$.

Further show that the prediction error can be decomposed as

$$\mathcal{E}_{\text{pred}} = \mathbb{E}(f(X) - \hat{f}(X))^2 = (w_1^*)^2(1 - \mathsf{SU})^2 + \mathsf{CN}^2,$$

where,

$$\mathsf{SU} = \frac{w_1}{w_1^*}$$
$$\mathsf{CN}^2 = \sum_{j=2}^{d} w_j^2.$$

Implement this decomposition in the Jupyter notebook and verify that they match with numerical observations.

(g) *Approximation error.* Finally consider the case where the true function cannot be expressed by the featurization that we train with. Concretely suppose that the true function is given by, $f(X) = \sum_{j=1}^{D} \phi(X) w_j^*$ and we learn a predictor $\hat{f}(X) = \sum_{j=1}^{d} \phi(X) w_j$ where $d < D$. Assume that the D-dimensional featurization is orthonormal with respect to the test distribution p and let $\mathbb{E}[\phi(X)] = \mathbf{0}$.

The residual between the true function and the predicted function can be decomposed as,

$$R = f(X) - \hat{f}(X) = \sum_{j=1}^{d} (w_j^* - w_j)\phi_j(X) + \sum_{j=d+1}^{D} w_j^* \phi_j(X).$$

Using a similar approach to the previous part **show that if,**

$$R = R_1 + R_2$$
$$R_1 = \sum_{j=1}^{d} (w_j^* - w_j)\phi_j(X)$$
$$R_2 = \sum_{j=d+1}^{D} w_j^* \phi_j(X),$$

R_1 **and** R_2 **are uncorrelated and Var**$[R_2] = \sum_{j=d+1}^{D} (w_j^*)^2$**.** Conclude that the prediction error in this case can be decomposed as,

$$\mathcal{E}_{\text{pred}} = \mathbb{E}(f(X) - \hat{f}(X))^2 = \text{Var}[R_2] + \sum_{j=1}^{d} (w_j^* - w_j)^2$$

Here $\text{Var}[R_2]$ is the component corresponding to the approximation error.

We have implemented this decomposition for you in the Jupyter notebook. **Interact with the demo and see how the two components of the error vary with** d **and the noise level in the system.**

5 Isotropic gaussians

In this problem we are going to observe some peculiar behaviour of high-dimensional vectors. We are going to talk about the vectors from gaussian distribution, whose covariance is equal to

identity matrix (they are also called isotropic). To obtain quantitative results, we will need some probabilistic bounds. First of all, we will need the following inequality for a standard normal random variable: for $\xi \sim \mathcal{N}(0,1)$ and any $t > 0$ it holds

$$\mathbb{P}(\xi > t) \leq \exp\left(-t^2/2\right).$$

Secondly, we will require analogous results for a $\chi^2(k)$ distribution (recall that this is the distribution of the sum of squares of k i.i.d. standard normals; in other words this is the distribution of the squared norm of an isotropic gaussian vector in $\mathbb{R}^k$). The result that we will use is: if $\xi \sim \chi^2(k)$, then for any $t > 0$

$$\mathbb{P}(\xi - k > 2\sqrt{kt} + 2t) \leq \exp(-t),$$
$$\mathbb{P}(\xi - k < -2\sqrt{kt}) \leq \exp(-t)$$

We will use these so often, that it will be convenient to use the following notation: denote

$$l_{\chi^2}(k,t) := k - 2\sqrt{kt}, \quad u_{\chi^2}(k,t) := k + 2\sqrt{kt} + 2t.$$

(a) Consider two independent random vectors: $\mathbf{u}, \mathbf{v} \sim \mathcal{N}(0, \mathbf{I}_d)$ in $\mathbb{R}^d$. **Show that for any** $t > 0$

$$\mathbb{P}\left(\|\mathbf{u}\|_2^2 \geq l_{\chi^2}(d,t)\right) \geq 1 - \exp(-t),$$
$$\mathbb{P}\left(|\mathbf{u}^\top \mathbf{v}|/\|\mathbf{v}\|_2 \leq \sqrt{2t}\right) \geq 1 - 2\exp(-t)$$

HINT: what is the distribution of the random variable $\frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{v}\|_2}$*?*

(b) **Use the result of the previous part to lower bound the angle between vectors v and u with probability at least** $1 - 3\exp(-t)$ **for any** $t \in (0, d/4)$**. What result do you get for** $t = 4$ **and** $d = 400$**?**

(c) Let's check how sharp our theoretical guarantees are. **Do the task from part A of the Jupyter notebook.**

(d) We've just seen that two random gaussian vectors in high dimensions are almost orthogonal to each other with high probability. What if we sample more than 2? Intuitively, if the dimension is high enough, and if we don't sample too many vectors, then the matrix comprised of those vectors should be close to orthogonal. For example, it seems reasonable to assume that the singular values of that matrix should be all close to each other. In this and the following parts we will give quantitative bounds that show exactly that.

Suppose $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a matrix with i.i.d. standard normal entries. In the next part we will use Gershgorin circle theorem (you can find its statement on this wikipedia page) to bound the singular values of $\mathbf{X}$ for $n \ll d$. To set up for that, **show that for any** $t > 0$ **for each Gershgorin disc of the matrix** $\mathbf{X}\mathbf{X}^\top$ **its center lies in the interval** $\left(l_{\chi^2}(d,t), u_{\chi^2}(d,t)\right)$ **with probability at least** $1 - 2\exp(-t)$**, and its radius is not larger than** $\sqrt{n u_{\chi^2}(d,t) u_{\chi^2}(n,t)}$ **with probability at least** $1 - 2\exp(-t)$**.**

HINT: when you compute the radius of the i-th Gershgorin disc of matrix $\mathbf{X}\mathbf{X}^\top$*, pull the norm of the i-th row of matrix* $\mathbf{X}$ *out of brackets. Then use Cauchy-Schwartz to obtain a* $\chi^2(n-1)$ *random variable.*

(e) **Use the previous part to conclude that with probability at least** $1-4n\exp(-t)$ **all the eigenvalues of** $\mathbf{X}\mathbf{X}^\top$ **lie between** $l_{\chi^2}(d,t) - \sqrt{n u_{\chi^2}(d,t) u_{\chi^2}(n,t)}$ **and** $u_{\chi^2}(d,t) + \sqrt{n u_{\chi^2}(d,t) u_{\chi^2}(n,t)}$**.**

What does this bound become for $d = 1000$**,** $n = 10$ **and** $t = 4 + \log n$**?**

HINT: you need the bound from the previous part to hold for every Gershgorin disc at the same time, so you should adjust the probability correspondingly.

(f) Once again, let's check our bounds numerically. **Do the task from part B of the Jupyter notebook.**

(g) The bounds in the previous parts showed that for $n \ll d$ the $n \times d$ matrix with i.i.d. standard normal entries is well-conditioned. In this part we will show that the opposite happens if $n = d$. Let's denote the rows of $\mathbf{X}$ as $\mathbf{X}_1, \ldots, \mathbf{X}_n$. Those are n vectors in n-dimensional space. Consider also a vector $\mathbf{u}$ of unit length, which is orthogonal to $\mathbf{X}_1, \ldots, \mathbf{X}_{n-1}$.

Show that $\sigma_{\max}(\mathbf{X}) \geq \|\mathbf{X}_n\|_2$ **and** $\sigma_{\min}(\mathbf{X}) \leq \|\mathbf{X}\mathbf{u}\|_2$**. Which upper bound on** $\sigma_{\min}(\mathbf{X})$ **and lower bound on** $\sigma_{\max}(\mathbf{X})$ **does that imply?**

HINT: What is the distribution of the vector $\mathbf{X}\mathbf{u}$*?*

(h) In the previous part we showed that the square of the minimum singular value $\sigma^2_{\min}$ of $\mathbf{X}$ can be upper bounded by a $\chi^2(1)$ random variable. However, one may notice that separating the last row to construct $\mathbf{u}$ in the previous part was somewhat arbitrary — any other row is just as good. So in fact, we can construct n different $\chi^2(1)$ random variables (one for each row), each of which gives an upper bound on $\sigma^2_{\min}$, so $\sigma_{\min}$ should perhaps be even smaller than the bound that we derived before. However, those $\chi^2(1)$ random variables are not independent, so it is rather tricky to develop a rigorous treatment.

In this part we will answer the question "what would our bound be if those $\chi^2(1)$ random variables were actually independent". Then in the last part we will check numerically if our result is correct.

Suppose $\xi_1, \ldots, \xi_n$ are i.i.d. non-negative random variables with continuous density f. Suppose also that $\infty > f(0) > 0$. **Show that** $n \min_{i \leq n} \xi_i$ **converges in distribution to some non-zero random variable. What intuition does this computation provide about** $\sigma_{\min}$**? How do you think, is this intuition correct or our assumption of independence is far from being true?**

HINT: a random variable with $\chi^2(1)$ *distribution has infinite density at zero, but its square root has finite density.*

(i) Finally, let's check our intuition by numerical simulation. **Do the task from part C of the Jupyter notebook.**

6 Your Own Question

Write your own question, and provide a thorough solution.

Writing your own problems is a very important way to really learn 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
- Anant Sahai
- Chawin Sitawarin
- Katia Patkin
- Raaz Dwivedi
- Rahul Arya
- Vignesh Subramanian