CS 189:289A Introduction to Machine Learning
Fall 2021 Jennifer Listgarten and Jitendra Malik

HW1

Due 9/20/21 at 11:59pm

- Homework 1 consists of both written and coding questions.
- We prefer that you typeset your answers using LaTeX or other word processing software. If you haven't yet learned LaTeX, one of the crown jewels of computer science, now is a good time! Neatly handwritten and scanned solutions will also be accepted for the written questions.
- In all of the questions, **show your work**, not just the final answer.

Deliverables:

1. Submit a PDF of your homework to the Gradescope assignment entitled "HW1 Write-Up". **Please start each question on a new page.** If there are graphs, include those graphs in the correct sections. **Do not** put them in an appendix. We need each solution to be self-contained on pages of its own.
    - In your write-up, please state with whom you worked on the homework. This should be on its own page and should be the first page that you submit.
    - In your write-up, please copy the following statement and sign your signature next to it. (Mac Preview and FoxIt PDF Reader, among others, have tools to let you sign a PDF file.) We want to make it *extra* clear so that no one inadvertently cheats. *"I certify that all solutions are entirely in my own words and that I have not looked at another student's solutions. I have given credit to all external sources I consulted."*
    - **Replicate all your code in an appendix**. Begin code for each coding question in a fresh page. Do not put code from multiple questions in the same page. When you upload this PDF on Gradescope, *make sure* that you assign the relevant pages of your code from appendix to correct questions.

1 Multivariate Gaussians: A review

(a) Consider a two dimensional random variable $Z \in \mathbb{R}^2$. In order for the random variable to be jointly Gaussian, a necessary and sufficient condition is that

- Z_1 and Z_2 are each marginally Gaussian, and
- $Z_1|Z_2 = z$ is Gaussian, and $Z_2|Z_1 = z$ is Gaussian.

A second characterization of a jointly Gaussian RV $Z \in \mathbb{R}^2$ is that it can be written as $Z = AX$, where $X \in \mathbb{R}^2$ is a collection of i.i.d. standard normal RVs and $A \in \mathbb{R}^{2\times 2}$ is a matrix.

Note that the probability density function of a multivariate Gaussian RV with mean vector, μ, and covariance matrix, Σ, is:

$$f(\mathbf{z}) = \exp\left(-\frac{1}{2}(\mathbf{z}-\mu)^T\Sigma^{-1}(\mathbf{z}-\mu)\right) / \sqrt{(2\pi)^k|\Sigma|}$$

.

Let X_1 and X_2 be i.i.d. standard normal RVs. Let U denote a binary random variable uniformly distributed on $\{-1, 1\}$, independent of everything else. Use one of the two characterizations given above to determine whether the following RVs are jointly Gaussian, and calculate the covariance matrix (regardless of whether the RVs are jointly Gaussian).

- $Z_1 = X_1$ and $Z_2 = X_2$.
- $Z_1 = X_1$ and $Z_2 = X_1 + X_2$.
- $Z_1 = X_1$ and $Z_2 = -X_1$.
- $Z_1 = X_1$ and $Z_2 = UX_1$.

Solution: Before diving into the solution, recall that the covariance matrix of a vector random variable X with mean (vector) μ is given by $\Sigma = \mathbb{E}[(X-\mu)(X-\mu)^\top]$. In other words, entry i, j of the covariance matrix denotes the covariance between the random variables X_i and X_j, i.e., $\Sigma_{ij} = \mathsf{cov}(X_i, X_j) = \mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)]$.

Additionally, two random variables U and V are said to be uncorrelated if $\mathsf{cov}(U, V) = 0$

- **First Characterization:** Z_1 and Z_2 are i.i.d. standard Gaussian, and so $(Z_1|Z_2 = z) \sim N(0, 1)$. Also, $Z_2|Z_1 = z \sim N(0, 1)$. Hence, the RVs are jointly Gaussian.

  Second Characterization: Z is jointly Gaussian, with $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

  Covariance Matrix: $\Sigma_Z = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.
- **First Characterization:** $Z_1 \sim N(0, 1)$, and $Z_2 \sim N(0, 2)$, but these RVs are not independent. Also, we have $(Z_2|Z_1 = z) \sim N(z, 1)$. In order to calculate the distribution of $(Z_1|Z_2 = z)$, see part (e).

  Second Characterization: Z is jointly Gaussian, with $A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$.

  Covariance Matrix: $\Sigma_Z = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$.

- **First Characterization:** We have $Z_1 \sim N(0,1)$ and $Z_2 \sim N(0,1)$ marginally. However, we have $(Z_1|Z_2 = z) \sim N(-z, 0)$, which is a degenerate Gaussian. The other conditional distribution is identical. Hence, the RVs are jointly Gaussian.

  Second Characterization: Z is jointly Gaussian, with $A = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}$.

  Covariance Matrix: $\Sigma_Z = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$.

- **First Characterization:** As before, we have $Z_1 \sim N(0,1)$ and $Z_2 \sim N(0,1)$ marginally. In order to see this, write

$$\begin{aligned} p(Z_2 = z_2) &= p(Z_2 = z_2|U = 1)p(U = 1) + p(Z_2 = z_2|U = -1)p(U = -1) \\ &= \frac{1}{2}p(X_1 = z_2|U = 1) + \frac{1}{2}p(X_1 = -z_2|U = -1) \\ &= \frac{1}{2}p(X_1 = z_2) + \frac{1}{2}p(X_1 = -z_2) \\ &= \frac{1}{2}p(X_1 = z_2) + \frac{1}{2}p(X_1 = z_2) \\ &= p(X_1 = z_2) \end{aligned}$$

  The random variable $(Z_2|Z_1 = z)$ is uniformly distributed on $\{-z, z\}$, and is therefore not Gaussian. The RVs are therefore not jointly Gaussian.

  Covariance Matrix: $\Sigma_Z = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

(b) Use the above example to show that two Gaussian random variables can be uncorrelated, but not independent. On the other hand, show that two uncorrelated, jointly Gaussian RVs are independent.

Solution: By definition, two random variables X and Y are independent iff $P(X = x, Y = y) = P(X = x)P(Y = y), \forall x, y$.

The last example in the previous part shows uncorrelated Gaussians that are not independent. In order to show that jointly Gaussian RVs (with individual variances σ_1^2 and σ_2^2) that are uncorrelated are also independent, assume without loss of generality that the RVs have zero mean, and notice that one can write the joint pdf as

$$\begin{aligned} f_Z(z_1, z_2) &= \frac{1}{(2\pi)\det\left(\Sigma_Z^{1/2}\right)} \exp\left(-\frac{1}{2}\begin{bmatrix} z_1 & z_2 \end{bmatrix}(\Sigma_Z)^{-1}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}\right) \\ &= \frac{1}{\sqrt{2\pi\sigma_1^2}}\frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{1}{2\sigma_1^2}z_1^2\right)\exp\left(-\frac{1}{2\sigma_1^2}z_2^2\right) \\ &= f_{Z_1}(z_1)f_{Z_2}(z_2). \end{aligned}$$

The decomposition follows since Σ_Z is a diagonal matrix when the RVs are uncorrelated. Since we have expressed the joint PDF as a product of the individual PDFs, the RVs are independent.

(c) With the setup above, let $Z = VX$, where $V \in \mathbb{R}^{2\times 2}$, and $Z, X \in \mathbb{R}^2$. What is the covariance matrix Σ_Z? Is this also true for a RV other than Gaussian?

Solution: The covariance matrix of a random vector Z (by definition) is given by $\mathbb{E}(Z - \mathbb{E}[Z])(Z-\mathbb{E}[Z])^\top$. Since the mean $\mathbb{E}[Z]$ is 0, we may write $\Sigma_Z = \mathbb{E}[VXX^\top V^\top] = V\mathbb{E}[XX^\top]V^\top = VV^\top$. This follows by linearity of expectation applied to vector random variables (write it out to convince yourself!)

Yes, this relation is also true for other distributions, since we didn't use the Gaussian assumption in this proof.

(d) Use the above setup to show that $X_1 + X_2$ and $X_1 - X_2$ are independent. Give another example pair of linear combinations that are independent.

Solution: By our previous arguments, it is sufficient to show that these are uncorrelated. Calculating the covariance matrix, we have $\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, which is diagonal. Any linear combination $Z = VX$ with $VV^\top = D$ for a diagonal matrix D results in uncorrelated random variables.

(e) Given a jointly Gaussian RV $Z \in \mathbb{R}^2$ with covariance matrix $\Sigma_Z = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{bmatrix}$, how would you derive the distribution of $Z_1|Z_2 = z$?

Hint: The following identity may be useful

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{b}{c} & 1 \end{bmatrix} \begin{bmatrix} \left(a - \frac{b^2}{c}\right)^{-1} & 0 \\ 0 & \frac{1}{c} \end{bmatrix} \begin{bmatrix} 1 & -\frac{b}{c} \\ 0 & 1 \end{bmatrix}.$$

Solution:

One can do this from first principles, by manipulating the densities themselves. However, we will show a linear algebraic method to derive the density. Using the hint, we begin by writing

$$\Sigma^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{\Sigma_{12}}{\Sigma_{22}} & 1 \end{bmatrix} \begin{bmatrix} \left(\Sigma_{11} - \frac{\Sigma_{12}^2}{\Sigma_{22}}\right)^{-1} & 0 \\ 0 & \frac{1}{\Sigma_{22}} \end{bmatrix} \begin{bmatrix} 1 & -\frac{\Sigma_{12}}{\Sigma_{22}} \\ 0 & 1 \end{bmatrix}.$$

We can now plug this into the density function. Recall that

$$\begin{aligned} f_{Z_1,Z_2}(z_1, z_2) &\propto \exp\left(-\frac{1}{2}\begin{bmatrix} z_1 & z_2 \end{bmatrix} \Sigma^{-1} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}\right) \\ &= \exp\left(-\frac{1}{2}\begin{bmatrix} z_1 & z_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{\Sigma_{12}}{\Sigma_{22}} & 1 \end{bmatrix} \begin{bmatrix} \left(\Sigma_{11} - \frac{\Sigma_{12}^2}{\Sigma_{22}}\right)^{-1} & 0 \\ 0 & \frac{1}{\Sigma_{22}} \end{bmatrix} \begin{bmatrix} 1 & -\frac{\Sigma_{12}}{\Sigma_{22}} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}\right) \\ &= \exp\left(-\frac{1}{2}\begin{bmatrix} z_1 - \frac{\Sigma_{12}}{\Sigma_{22}} z_2 & z_2 \end{bmatrix} \begin{bmatrix} \left(\Sigma_{11} - \frac{\Sigma_{12}^2}{\Sigma_{22}}\right)^{-1} & 0 \\ 0 & \frac{1}{\Sigma_{22}} \end{bmatrix} \begin{bmatrix} z_1 - \frac{\Sigma_{12}}{\Sigma_{22}} z_2 \\ z_2 \end{bmatrix}\right). \end{aligned}$$

Now see that since the square matrix is diagonal, our density decomposes to yield

$$f_{Z_1,Z_2}(z_1,z_2) \propto \exp\left(-\frac{1}{2}(z_1 - \frac{\Sigma_{12}}{\Sigma_{22}}z_2)^2\left(\Sigma_{11} - \frac{\Sigma_{12}^2}{\Sigma_{22}}\right)^{-1}\right)\exp\left(-\frac{1}{2\Sigma_{22}}z_2^2\right).$$

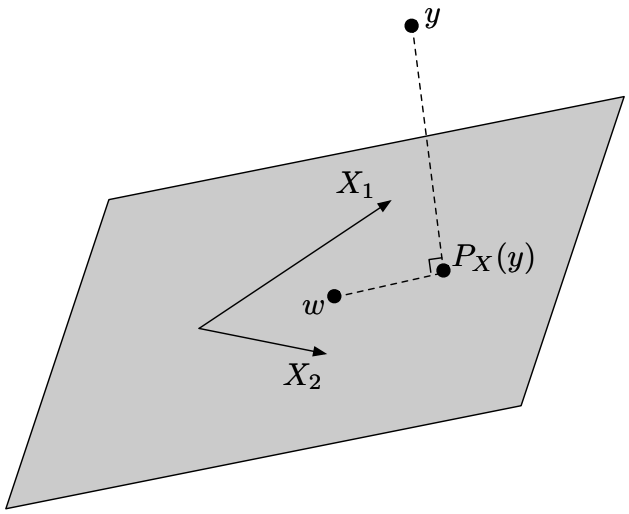
Conditional on $Z_2 = z_2$, we see that $Z_1|Z_2 = z_2 \sim N(\frac{\Sigma_{12}}{\Sigma_{22}}z_2, \Sigma_{11} - \frac{\Sigma_{12}^2}{\Sigma_{22}})$.

2 Linear Regression, Projections and Pseudoinverses

We are given $X \in \mathbb{R}^{n \times d}$ where $n > d$ and $\text{rank}(X) = d$. We are also given a vector $y \in \mathbb{R}^n$. Define the orthogonal projection of y onto $\text{range}(X)$ as $P_X(y)$.

(a) Prove that $P_X(y) = \underset{w \in \text{range}(X)}{\arg\min} |y - w|^2$.

Solution:



Note that $|y-w|^2 = |y-P_X(y)+P_X(y)-w|^2 = |y-P_X(y)|^2+|P_X(y)-w|^2+2(y-P_X(y))^\top(P_X(y)-w)$. Now we can easily see from the figure above that $(y - P_X(y))$ is orthogonal to any vector in the columnspace of X. Hence $|y - w|^2 = |y - P_X(y)|^2 + |P_X(y) - w|^2 \geq |y - P_X(y)|^2$. This shows that $w = P_X(y)$.

Side Note: In lecture, we learned how to find $\hat{\theta}$ that minimizes the least squares loss $L(\theta) = |y - X\theta|^2$. In other words, we tried to find θ such that $X\theta$ is the vector in the columnspace of X that is closest to our response vector y. Hence, $P_X(y) = X\theta$.

(b) An orthogonal projection is a linear transformation. Hence, we can define $P_X(y) = Py$ for some projection matrix P. Specifically, given $1 \leq d \leq n$, a matrix $P \in \mathbb{R}^{n \times n}$ is said to be a rank-d orthogonal projection matrix if $\text{rank}(P) = d$, $P = P^\top$ and $P^2 = P$. Prove that P is a rank-d projection matrix if and only if there exists a $U \in \mathbb{R}^{n \times d}$ such that $P = UU^\top$ and $U^\top U = I$

Hint Use the eigendecomposition of P.

Solution:

Since P is symmetric, $P = V\Sigma V^\top$ for some orthogonal $V \in \mathbb{R}^{n \times n}$ and real, diagonal $\Sigma \in \mathbb{R}^{n \times n}$. Let v be an eigenvector of P with eigenvalue λ. Then, $\lambda^2 v = P^2 v = Pv = \lambda v$, so that $\lambda \in \{0, 1\}$. Hence, the diagonals of Σ are binary-valued. Letting U denote the matrix whose columns correspond to the indices i for which $\Sigma_{ii} = 1$, we have $P = V\Sigma V^\top = UU^\top$. Since P has rank d, there that there are d such 1-valued indices.

Since the columns of U are a subset of those of V, they are orthonormal, whence $U^\top U = I$.

Conversely, if $P = UU^\top$, then $P = P^\top$ trivially, and $P^2 = UU^\top UU^\top = UU^\top$. Moreover, P has rank at most d since $P = UU^\top$, and rank at least $\text{rank}(PU) = \text{rank}(UU^\top U) = \text{rank}(U) = d$.

(c) Prove that if P is a rank d projection matrix, then $\text{tr}(P) = d$.

Solution:

Using the trace trick $\text{tr}(AB) = \text{tr}(BA)$, $\text{tr}(P) = \text{tr}(UU^\top) = \text{tr}(U^\top U) = \text{tr}(I_d) = d$.

(d) The Singular Value Decomposition theorem states that we can write any matrix X as

$$X = \sum_{i=1}^{\min\{n,d\}} \sigma_i u_i v_i^\top = \sum_{i:\sigma_i>0} \sigma_i u_i v_i^\top$$

where $\sigma_i \geq 0$, and $\{u_i\}$ and $\{v_i\}$ are an orthonormal. Show that

(a) $\{v_i : \sigma_i > 0\}$ are an orthonormal basis for the row space of of X

(b) Similarly, $\{u_i : \sigma_i > 0\}$ are an orthonormal basis for the columnspace of X
Hint: consider $X^\top$.

Solution: Since $\{v_i : \sigma_i > 0\}$ are orthonormal, it suffices to show that their span is the row space of X. Since the row space is the orthogonal complement of the nullspace of X, it suffices to show that $v \in \text{span}(\{v_i : \sigma_i > 0\})^\perp$ if and only if $Xv = 0$. We have that

$$Xv = \sum_{i:\sigma_i>0} \sigma_i u_i (v_i^\top v).$$

Since $\sigma_i u_i$ are all linearly independent, $Xv = 0$ if and only if $(v_i^\top v) = 0$ for all i, as needed.

The the second part,

$$X^\top = \sum_{i:\sigma_i>0} \sigma_i v_i u_i^\top,$$

which means that u_i are a basis for the row space of $X^\top$ by the above. Hence, u_i are a basis for the columnspace of X.

(e) Prove that if $X \in \mathbb{R}^{n\times d}$ and $\text{rank}(X) = d$, then $X(X^\top X)^{-1}X^\top$ is a rank-d orthogonal projection matrix. What is the corresponding matrix U?

Solution:

Let $X = U\Sigma V^\top$ denote the SVD of X, with $U \in \mathbb{R}^{n\times d}$, and $\Sigma \in \mathbb{R}^{d\times d}$, and $V \in \mathbb{R}^{d\times d}$. Then, $X^\top X = V\Sigma^2 V^\top$, and since $\text{rank}(X) = d$, Σ^2 is invertible, with $(X^\top X)^{-1} = V^\top \Sigma^{-2} V$. Hence,

$$X(X^\top X)^{-1}X^\top = U\Sigma V^\top (V\Sigma^{-2}V^\top)V\Sigma U^\top = U\Sigma\Sigma^{-2}\Sigma U = UU^\top,$$

which shows that $X(X^\top X)^{-1}X^\top$ is the projection matrix $UU^\top$, where U is the left singular-vectors matrix of X.

(f) Define the Moore-Penrose pseudoinverse to be the matrix:

$$X^\dagger = \sum_{i:\sigma_i>0} \sigma_i^{-1} v_i u_i^\top,$$

To what operator does the matrix $X^\dagger X$ correspond? What is $X^\dagger X$ if $\text{rank}(X) = d$? If $\text{rank}(X) = d$ and $n = d$?

Solution:

$$\begin{aligned} X^\dagger X &= \sum_{i:\sigma_i>0} \sigma_i^{-1} v_i u_i^\top \sum_{j:\sigma_j>0} \sigma_j u_j v_j^\top \\ &= \sum_{i:\sigma_i>0} \sum_{j:\sigma_j>0} \sigma_j \sigma_i^{-1} u_i^\top u_j \cdot v_i v_j^\top \\ &= \sum_{i:\sigma_i>0} \sum_{j:\sigma_j>0} \sigma_j \sigma_i^{-1} \mathbf{I}(i=j) \cdot v_i v_j^\top \\ &= \sum_{i:\sigma_i>0} v_i v_i^\top. \end{aligned}$$

We see that $X^\dagger X$ is an orthogonal projection onto the span of v_i, which is precisely the row space of X. If $\text{rank}(X) = d$, then $X^\dagger X = I$, and thus if $d = n$, $X^\dagger = X^{-1}$.

3 Convergence for Logistic Regression

In this problem, we will take steps toward proving that gradient descent converges to a unique minimizer of the logistic regression cost function, binary cross-entropy, when combined with L_2 regularization. In particular, we will consider the simplified case where we are minimizing this cost function for a single data point. For weights $w \in \mathbb{R}^d$, data $x \in \mathbb{R}^d$, and a label $y \in \{0, 1\}$, the L_2-regularized logistic regression cost function is given by

$$J(w) = -y \log s(x \cdot w) - (1 - y) \log(1 - s(x \cdot w)) + \frac{1}{2}\lambda \|w\|_2^2$$

Where $s(\gamma) = 1/(1 + \exp(-\gamma))$ is the logistic function (also called the sigmoid) and $\lambda > 0$. You may assume that $x \neq 0$.

(a) To start, write the gradient descent update function $G(w)$, which maps w to the result of a single gradient descent update with learning rate $\epsilon > 0$.

Solution: From lecture, we know $s'(\gamma) = s(\gamma)(1 - s(\gamma))$. Letting $z = s(w \cdot x)$, we get

$$\nabla_w J = -(y - z)x + \lambda w$$

Hence, the the gradient descent update is

$$G(w) = w + \epsilon[(y - z)x - \lambda w]$$

(b) Show that the cost function J has a unique minimizer w^* by proving that J is strictly convex.
Hint: A sufficient condition for a function to by strictly convex is that its Hessian is positive definite.

Solution: From the last part, the gradient is $-(y - s(w \cdot x))x + \lambda w$. Taking the Jacobian of this, we have

$$\nabla_w^2 J = z(1 - z)xx^T + \lambda I.$$

This is positive definite, since for any vector $v \in \mathbb{R}^d$, $v^T(z(1 - z)xx^T + \lambda I)v = (z(1 - z))(v^T x)^2 + \lambda v^T v > 0$. This holds because $0 < z < 1$, $\lambda > 0$ and $x \neq 0$.

(c) Next, show that for a sufficiently chosen ϵ, G is a *contraction*, which means that there is a constant $0 < \rho < 1$ such that, for any $w, w' \in \mathbb{R}^d$, $\left\|G(w) - G(w')\right\|_2 < \rho \|w - w'\|_2$.
Hint: this is equivalent to showing that the Jacobian of G has bounded spectral norm: $\left\|\nabla_w G(w)\right\|_2 < \rho$.
Hint 2: for a PSD matrix, the eigenvalues and singular values are equivalent.
Hint 3: consider the eigenvalues of $A + \alpha I$ for a square matrix, A and $\alpha \in \mathbb{R}$.

Solution: First let's consider Hint 3. Let v be an eigenvector of A with eigenvalue, β. Then we have,

$$\begin{aligned}(A + \alpha I)v &= Av + \alpha Iv \\ &= \beta v + \alpha v\end{aligned}$$

$$= (\beta + \alpha)v.$$

We see that v is an eigenvector of $A + \alpha I$ with eigenvalue $\beta + \alpha$. Thus, adding αI to A shifts the corresponding eigenvalues by α.

Now, we compute the Jacobian of G:

$$\nabla_w G(w) = \nabla_w (w + \epsilon[(y - z)x - \lambda w]) = I - \epsilon[z(1 - z)xx^T + \lambda I]$$

Recognize that $z(1 - z)xx^T$ is a rank-one PSD matrix with one non-zero eigenvalue (equivalently, one non-zero singular value) equal to $z(1 - z)\|x\|_2^2$. Since $0 < z(1 - z) < 1$ we can bound the maximum singular value by $\|x\|_2^2$. Similarly, λI is PD with eigenvalues (equivalently, singular values), all equal to λ. Applying Hint 3, we see that maximum eigenvalue of $z(1-z)xx^T + \lambda I$ is $\left(z(1 - z)\|x\|_2^2 + \lambda\right) < \|x\|_2^2 + \lambda$.

Thus, if we take ϵ such that $0 < \epsilon \leq \frac{1}{\|x\|_2^2 + \lambda}$, we can bound all of the eigenvalues / singular values of $\epsilon[z(1 - z)xx^T + \lambda I]$ between 0 and 1.

Let $A = \epsilon[z(1 - z)xx^T + \lambda I]$ so that $\nabla_w G = I - A$. Since A has eigenvalues between 0 and 1, $-A$ has eigenvalues between -1 and 0. Applying the Hint 3, the eigenvalues $I - A$ are between 0 and 1. This guarantees that $\nabla_w G$ is PSD and by Hint 2, that the singular values are bounded above by 1.

So, $\|\nabla_w G\|_2 = \|I - \epsilon[z(1 - z)xx^T + \lambda I]\|_2 = \rho$ for a constant $0 < \rho < 1$.

(d) Finally, calling $w^{(t)}$ the t-th iterate of gradient descent, show that $\left\|w^* - w^{(t)}\right\|_2 < \rho^t \left\|w^* - w^{(0)}\right\|_2$, so that $\lim_{t\to\infty} \left\|w^* - w^{(t)}\right\|_2 = 0$.

Solution: For a single step:

$$\left\|w^* - G(w)\right\|_2 = \left\|G(w^*) - G(w)\right\|_2 < \rho \left\|w^* - w\right\|_2$$

So, for n steps, we get

$$\left\|w^* - w^{(t)}\right\|_2 = \left\|w^* - G(w^{(t-1)})\right\|_2 < \rho \left\|w^* - G(w^{(t-2)})\right\|_2 < \rho^2 \left\|w^* - G(w^{(t-3)})\right\|_2 < \ldots < \rho^t \left\|w^* - w^{(0)}\right\|_2$$

4 Geometry of Ridge Regression

You recently learned ridge regression and how it differs from ordinary least squares. In this question we will explore how ridge regression is related to solving a constrained least squares problem in terms of their parameters and solutions.

Solution: *Note:* the "formal derivation" sections are strictly for your interest and are not necessary in order to receive full credit for this problem.

(a) Given a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and a vector $\mathbf{y} \in \mathbb{R}^n$, define the optimization problem

$$\begin{aligned} &\text{minimize } \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2. \\ &\text{subject to } \|\mathbf{w}\|_2^2 \leq \beta^2. \end{aligned} \tag{1}$$

We can utilize Lagrange multipliers to incorporate the constraint into the objective function by adding a term which acts to "penalize" the thing we are constraining. Rewrite the constrained optimization problem into an unconstrained optimization problem.

Solution: Let us introduce the Lagrange multiplier $\lambda \geq 0$ into the constrained problem, thus turning the constraint $\|\mathbf{w}\|_2^2 \leq \beta^2$ into a "penalty" $\lambda(\|\mathbf{w}\|_2^2 - \beta^2)$. The unconstrained objective therefore takes the form

$$\min_w \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda(\|\mathbf{w}\|_2^2 - \beta^2).$$

Formal derivation: Given a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and a vector $\mathbf{y} \in \mathbb{R}^n$, define the optimization problem

$$\begin{aligned} &\text{minimize } \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \\ &\text{subject to } \|\mathbf{w}\|_2^2 \leq \beta^2. \end{aligned} \tag{2}$$

In the first step we show that the above problem is equivalent (i.e. it has the same solution) as the following one:

$$\min_{\mathbf{w}} \max_{\lambda \geq 0} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \cdot \underbrace{(\|\mathbf{w}\|_2^2 - \beta^2)}_{\Delta(\mathbf{w})} \tag{3}$$

We see this as follows:

$$\max_{\lambda \geq 0} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \cdot \Delta(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \begin{cases} \infty & \text{if } \Delta(\mathbf{w}) > 0 \\ 0 & \text{if } \Delta(\mathbf{w}) \leq 0. \end{cases}$$

In the last step we used that if $\Delta(\mathbf{w}) > 0$, the term $\lambda\Delta(\mathbf{w})$ can be made arbitrarily large by making λ large. On the other hand if $\Delta(\mathbf{w}) \leq 0$, the maximization problem is solved by $\lambda = 0$.

It turns out that something called strong duality[1] holds here, which means we can exchange the minimization and maximization in (3) without changing the resulting optimal objective

[1] An understanding of duality is one of the key concepts provided by studying optimization. In general without strong duality, there can be a gap between what are called the primal and dual problems. Strong duality is when there is no gap. A full treatment of duality is beyond the scope of 189/289A, but is referenced here so that students can begin to understand why such understanding can be relevant. The example being studied here is almost paradigmatic in its strong duality.

function. We get

$$\max_{\lambda \geq 0} \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \cdot (\|\mathbf{w}\|_2^2 - \beta^2). \tag{4}$$

For every β, the maximum will be attained at some $\lambda^*(\beta)$ and this establishes the equivalence of the constrained problem (2) with the unconstrained problem

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda^*(\beta) \cdot (\|\mathbf{w}\|_2^2 - \beta^2). \tag{5}$$

i.e. There is a $\lambda^*(\beta)$ so that the problem has the same resulting $\mathbf{w}$ solution as if we had solved the original constrained optimization problem.

(b) Recall that ridge regression is given by the unconstrained optimization problem

$$\min_{w} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \nu \|\mathbf{w}\|_2^2. \tag{6}$$

One way to interpret "ridge regression" is as the Lagrangian form of a constrained problem. Qualitatively, how would increasing β in our previous problem be reflected in the desired penalty ν of ridge regression (i.e. if our threshold β increases, what should we do to ν)?

Solution: Intuitively, β is the radius of the ball in which the solution $\mathbf{w}^*$ of the constrained optimization problem (1) is sought and ν is the weight that is put on making $\|\mathbf{w}\|^2$ small. Therefore, if we increase ν, this corresponds to a stronger regularization of $\mathbf{w}$ to zero, which corresponds to a smaller β. In the limit $\nu \to 0$, no regularization takes place which corresponds to $\beta \to \infty$. Similarly, in the case $\nu \to \infty$, we force $\mathbf{w} = 0$, which corresponds to $\beta = 0$ in the constrained setting.

Formal derivation: Again to solve this formally, we need some tools from convex optimization like Lagrangians and KKT conditions that you are not required to know for this course, but that we recommend you get familiar with if you want to deepen your understanding of machine learning. Remember, machine learning is the application of optimization to problems formulated using the language of linear algebra and probability. There really is no avoiding optimization if you are serious.

The Lagrangian corresponding to the constrained optimization problem is

$$\mathcal{L}(\mathbf{w}, \lambda) = \left\|\mathbf{y} - \mathbf{X}\mathbf{w}\right\|_2^2 + \lambda(\|\mathbf{w}\|_2^2 - \beta).$$

Writing down the KKT conditions on optimality, we get

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^*, \lambda^*) = 0$$
$$\lambda^*(\left\|\mathbf{w}^*\right\|_2^2 - \beta) = 0.$$

If $\mathbf{w}^*(\nu)$ is a solution of (6), it satisfies the KKT conditions with $\beta = \left\|\mathbf{w}^*(\nu)\right\|_2^2$ and $\lambda^* = \nu$. Therefore, the constrained and the unconstrained problem are equivalent if and only if $\beta = \left\|\mathbf{w}^*(\nu)\right\|_2^2$ where $\left\|\mathbf{w}^*(\nu)\right\|_2^2$ is the solution of $\left\|\mathbf{w}^*(\nu)\right\|_2^2$.

Now how does β behave if ν increases? From $\mathbf{w}^*(\nu) = (\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$ and the singular value decomposition $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top$ we get

$$\left\|\mathbf{w}^*(\nu)\right\|_2^2 = \left\|(\Sigma^2 + \nu\mathbf{I})^{-1}\Sigma\mathbf{U}^\top\mathbf{y}\right\|_2^2$$

and therefore β is monotonically decreasing for ν — in agreement with our intuition above.

(c) 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 $\mathbf{x}$?

Solution: The change in prediction is given by

$$|\mathbf{w}^\top(\mathbf{x} + \boldsymbol{\epsilon}) - \mathbf{w}^\top\mathbf{x}| = |\mathbf{w}^\top\boldsymbol{\epsilon}| \leq \|\mathbf{w}\|_2\|\boldsymbol{\epsilon}\|_2 \tag{7}$$

where the second step is a result of the Cauchy-Schwarz inequality. To understand intuition as to why this is true, you can divide both sides by $\|\mathbf{w}\|_2\|\boldsymbol{\epsilon}\|_2$ which gives you

$$\left|\left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|_2}, \frac{\boldsymbol{\epsilon}}{\|\boldsymbol{\epsilon}\|_2} \right\rangle\right| \leq 1 \tag{8}$$

In words, this inequality is saying that if you project a unit vector $\frac{\boldsymbol{\epsilon}}{\|\boldsymbol{\epsilon}\|_2}$ onto some direction $\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$, the resulting length must be less than or equal to 1. The prediction will thus vary from the original prediction within a distance of at most $\|\mathbf{w}\|_2\|\boldsymbol{\epsilon}\|_2$.

(d) Derive that the solution to ridge regression (6) is given by $\hat{\mathbf{w}}_\mathbf{r} = (\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$. What happens when $\nu \to \infty$? It is for this reason that sometimes regularization is referred to as "shrinkage."

Solution: Since this was derived in lecture, we keep the discussion brief. The objective is

$$\begin{aligned} f(\mathbf{w}) &= \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \nu\|\mathbf{w}\|_2^2 \\ &= \mathbf{w}^\top(\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})\mathbf{w} - 2\mathbf{y}^\top\mathbf{X}\mathbf{w} + \mathbf{y}^\top\mathbf{y}. \end{aligned}$$

Taking the derivative with respect to w and setting it to zero, we obtain

$$0 = 2\mathbf{w}^\top(\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I}) - 2\mathbf{y}^\top\mathbf{X}.$$

Thus, the solution is given by

$$\mathbf{w} = (\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}.$$

As $\nu \to \infty$ the matrix $(\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})^{-1}$ converges to the zero matrix, and so we have $\mathbf{w} = \mathbf{0}$.

(e) Note that in computing $\hat{\mathbf{w}}_\mathbf{r}$, we are trying to invert the matrix $\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I}$ instead of the matrix $\mathbf{X}^\top\mathbf{X}$. If $\mathbf{X}^\top\mathbf{X}$ has eigenvalues $\sigma_1^2, \ldots, \sigma_d^2$, what are the eigenvalues of $\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I}$? Comment on why adding the regularizer term $\nu\mathbf{I}$ can improve the inversion operation numerically.

Solution: The eigenvalues of $\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I}$ are given by $\sigma_1^2 + \nu, \sigma_2^2 + \nu, \cdots, \sigma_d^2 + \nu$. In order to see this, note that if v_i is an eigenvector of $\mathbf{X}^\top\mathbf{X}$ with eigenvalue σ_i^2, then we have

$$(\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})v_i = \sigma_i^2 v_i + \nu v_i = (\sigma_i^2 + \nu)v_i.$$

Notice that $\sigma_i^2 \geq 0$ since the matrix $\mathbf{X}^\top\mathbf{X}$ is positive semi-definite, but some eigenvalues may be very close to 0. The eigenvalues of the inverse of a matrix are the inverses of the eigenvalues. Consequently, small eigenvalues lead to numerical instability of calculating the inverse of a matrix, since the inverse blows these up. By adding a regularizer, we increase the values of the eigenvalues of the original matrix above a threshold ν, and thus improve the inversion operation.

A careful student would notice that there is an issue with the simple explanation here. "Small" relative to what? The answer to this (take a numerical linear algebra course to understand this more deeply) is that we consider small eigenvalues relative to the largest eigenvalue — it is the dynamic range of the scale of eigenvalues that matter. This is what is effectively captured in something called the condition number of the matrix. The operation of adding ν changes greatly the relative scale of eigenvalues that are smaller than ν and does almost nothing (in a relative sense) to eigenvalues that are much bigger than ν. So regularization improves the numerical conditioning of the problem.

(f) Let the number of parameters $d = 3$ and the number of datapoints $n = 5$, and let the eigenvalues of $\mathbf{X}^\top\mathbf{X}$ be given by 1000, 1 and 0.001. We must now choose between two regularization parameters $\nu_1 = 100$ and $\nu_2 = 0.5$. Which do you think is a better choice for this problem and why?

Solution: $\nu = 0.5$ is a better option. We are looking for a regularization constant for better numerical conditioning without changing the problem substantially.

Notice that the eigenvalue 0.001 could cause us numerical issues as noted above, which we would like to eliminate. We would therefore like to preserve the relative size of the original eigenvalues. By choosing $\nu = 100$, we also eliminate the effect of the eigenvalue 1, thereby altering our problem unnecessarily.

(g) 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. $\mathbf{X} \in \mathbb{R}^{4\times 5}$. Now this is clearly an underdetermined system, since $n < d$. Show that ridge regression with $\nu > 0$ results in a unique solution, whereas ordinary least squares has 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.

Solution: First, we show that ridge regression always leads to a unique solution. We know that the minimizer is given by

$$\mathbf{w} = (\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}.$$

We also know that the eigenvalues of the matrix $\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I}$ are all at least ν, and so the matrix is invertible, thus leading to a unique solution.

For ordinary least squares, let us assume that $\mathbf{w}'$ minimizes $\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2$, i.e., $\|\mathbf{X}\mathbf{w}' - \mathbf{y}\|_2 \leq \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2$ for all $\mathbf{w} \in \mathbb{R}^d$.

Since $d > n$, the matrix $\mathbf{X}$ has a non-trivial nullspace. We can take any vector $\mathbf{w_0}$ in the nullspace of $\mathbf{X}$ and consider the new vector $\mathbf{w}''(\alpha) = \mathbf{w}' + \alpha\mathbf{w_0}$.

Notice that $\|\mathbf{X}\mathbf{w}''(\alpha) - \mathbf{y}\|_2 = \|\mathbf{X}\mathbf{w}' - \mathbf{y}\|_2 \leq \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2$ for all $\mathbf{w} \in \mathbb{R}^d$ because $\mathbf{w_0}$ is in the null space of $\mathbf{X}$. Thus, the vector $\mathbf{w}''(\alpha)$ is a minimizer for any choice of α. We have thus shown that the least squares problem has an infinite number of solutions.

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

Hint: Think about the SVD of $\mathbf{X}$.

Solution:

Plugging the singular value decomposition $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top$ into the solution of the ridge regression, we get

$$\begin{aligned}\mathbf{w}^*(\nu) &= (\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y} \\ &= \mathbf{V}(\Sigma^\top\Sigma + \nu\mathbf{I})^{-1}\Sigma^\top\mathbf{U}^\top\mathbf{y}\end{aligned}$$

and for $\nu \to 0$ this converges to the unique solution $\mathbf{w}^* = \mathbf{V}\Sigma^{-1}\mathbf{U}^\top\mathbf{y}$, also called the *minimum norm solution*. (Notice here that Σ^{-1} has the obvious definition of just involving the relevant square matrix of nonzero singular values.)

This is also closely connected to the idea of the Moore-Penrose inverse. This problem continues the trend of illustrating the utility of the SVD while reasoning about such problems. It really is a workhorse of machine learning reasoning.

(i) 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

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

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

Solution: The objective is

$$\begin{aligned}f(w) &= \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \nu\|\Gamma\mathbf{w}\|_2^2 \\ &= \mathbf{w}^\top(\frac{1}{2}\mathbf{X}^\top\mathbf{X} + \nu\Gamma^\top\Gamma)\mathbf{w} - \mathbf{y}^\top\mathbf{X}w + \frac{1}{2}\mathbf{y}^\top\mathbf{y}.\end{aligned}$$

Taking derivatives with respect to w and setting the result to zero, we obtain

$$0 = \mathbf{w}^\top(\mathbf{X}^\top\mathbf{X} + 2\nu\Gamma^\top\Gamma) - \mathbf{y}^\top\mathbf{X}.$$

Thus, the solution is now given by $w = (\mathbf{X}^\top\mathbf{X} + 2\nu\Gamma^\top\Gamma)^{-1}\mathbf{X}^\top y$, and one can again verify that this is a minimizer by showing that the Hessian is positive definite since the matrix Γ has full rank, implying that $\Gamma^\top\Gamma$ is itself positive definite.

5 Blair and their giant peaches

Make sure to include the code you write for this problem in an appendix

Blair is a mage testing how long they can fly a collection of giant peaches. They have n training peaches – with masses given by $x_1, x_2, \ldots x_n$ – and flies these peaches once to collect training data. The experimental flight time of peach i is given by y_i. They believe that the flight time is well approximated by a polynomial function of the mass

$$y_i \approx w_0 + w_1 x_i + w_2 x_i^2 \cdots + w_D x_i^D$$

where their goal is to fit a polynomial of degree D to this data. Include all text responses and plots in your write-up. You can use `Numpy`, `Matplotlib`, and the function `scipy.io.loadmat`.

(a) Show how Blair's problem can be formulated as a linear regression problem.

Solution: The problem is to find the coefficients w_d such that the squared error is minimized:

$$\min_{w_0,\ldots,w_D} \sum_i (w_0 + w_1 x_i + w_2 x_i^2 \cdots + w_D x_i^D - y_i)^2$$

Assume that we construct the following matrix:

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^D \\ 1 & x_2 & x_2^2 & \ldots & x_2^D \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^D \end{bmatrix}$$

Then the problem can be written as:

$$\min_{\mathbf{w}} \|X\mathbf{w} - \mathbf{y}\|^2$$

(b) You are given data of the masses $\{x_i\}_{i=1}^n$ and flying times $\{y_i\}_{i=1}^n$ in the "x_train" and "y_train" keys of the file `1D_poly.mat` with the masses centered and normalized to lie in the range $[-1, 1]$. Write code to perform the least squares fit. You may not use `numpy.linalg.lstsq`, but you can use other linear algebra functions, such as `numpy.linalg.solve`[2]. Letting f_D denote the fitted polynomial, plot the average training error $R(D) = \frac{1}{n}\sum_{i=1}^n (y_i - f_D(x_i))^2$ against D in the range $D \in \{0, 1, 2, 3, \ldots, n-1\}$.

Solution:

See Figure 1 for the plot. See code below:

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.io as spio
```

[2]Outside of this class you should use `numpy.linalg.lstsq`.

```python
def lstsq(A, b):
    return np.linalg.solve(A.T @ A, A.T @ b)

def main():
    # Loading data
    data = spio.loadmat('1D_poly.mat', squeeze_me=True)
    x_train = np.array(data['x_train'])
    y_train = np.array(data['y_train']).T

    n = 20  # max degree
    err = np.zeros(n-1 )

    for d in range(n -1):
        D = d + 1
        for i in range(D + 1):
            if i == 0:
                Xf = np.array([1] * x_train.size)
            else:
                Xf = np.vstack([np.power(x_train, i), Xf])
        Xf = Xf.T

        w = lstsq(Xf, y_train)
        y_predicted = Xf @ w
        err[d] = (np.linalg.norm(y_train - y_predicted)**2) / n

    plt.plot(range(1,n),err)
    plt.xlabel('Degree of Polynomial')
    plt.ylabel('Training Error')
    plt.show()


if __name__ == "__main__":
    main()
```

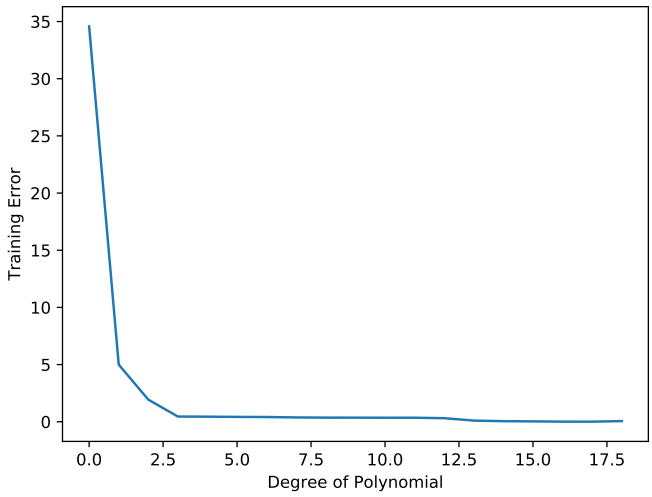


Figure 1: Training Error: **Result for 1D_poly.mat**

(c) How does the average training error behave as a function of D, and why? What happens if you try to fit a polynomial of degree n with a standard matrix inversion method?

Solution: The training error decreases since we have more degrees of freedom to fit the dataset. If we try to fit a polynomial of degree $n - 1$, we will have enough parameters to fit the data

exactly, so the training error will be zero. Using a polynomial of degree n results in a non-invertible data matrix, and so we cannot use a standard matrix inversion method to find our solution.

(d) Blair has taken CS189 so they decide that they need to run another experiment before deciding that their prediction is true. They run another fresh experiment of flight times using the same peaches, to obtain the data with key "y_fresh" in 1D_POLY.MAT. Denoting the fresh flight time of peach i by $\tilde{y}_i$, plot the average error $\tilde{R}(D) = \frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i - f_D(x_i))^2$ for the same values of D as in part (b) using the polynomial approximations f_D also from the previous part. How does this plot differ from the plot in (b) and why?

Solution: The plots are shown in Figure 2. The plots are different from the training errors since the data was generated afresh. While increasing the polynomial degree served to fit the noise in the training data, we cannot hope to fit noise by using the same model for fresh data. Hence, our performance degrades as we increase polynomial degree, with a minimum seen at the true model order.

```
def main():
    data = spio.loadmat('1D_poly.mat', squeeze_me=True)
    x_train = np.array(data['x_train'])
    y_train = np.array(data['y_train']).T
    y_fresh = np.array(data['y_fresh']).T

    n = 20  # max degree
    err_train = np.zeros(n - 1)
    err_fresh = np.zeros(n - 1)

    for d in range(n - 1):
        D = d + 1
        for i in range(D + 1):
            if i == 0:
                Xf = np.array([1] * n)
            else:
                Xf = np.vstack([np.power(x_train, i), Xf])
        Xf = Xf.T

        w = lstsq(Xf, y_train)
        err_train[d] = (np.linalg.norm(y_train - Xf @ w)**2) / n
        err_fresh[d] = (np.linalg.norm(y_fresh - Xf @ w)**2) / n

    plt.figure()
    plt.ylim([0, 6])
    plt.plot(err_train, label='train')
    plt.plot(err_fresh, label='fresh')
    plt.legend()
    plt.show()


if __name__ == "__main__":
    main()
```

(e) How do you propose using the two plots from parts (b) and (d) to "select" the right polynomial model for Blair?

Solution: The right model is the one that minimizes the error in the fresh dataset. The minimizer is at $D = 4$.

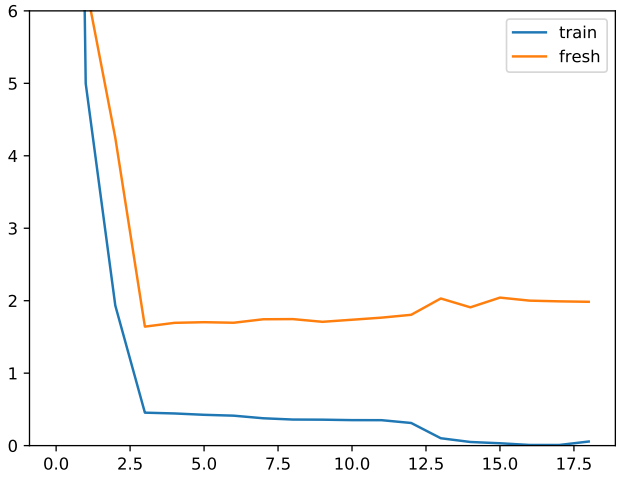


Figure 2: Fresh Error yfresh: **Result for 1D_poly.mat**

(f) Blair has a new hypothesis – the flying time is actually a function of the mass, smoothness, size, and sweetness of the peach, and some multivariate polynomial function of all of these parameters. A D-multivariate polynomial function looks like

$$f_D(\mathbf{x}) = \sum_j \alpha_j \prod_i x_i^{p_{ji}},$$

where $\forall j : \sum_i p_{ji} \leq D$. Here α_j is the scale constant for jth term and p_{ji} is the exponent of x_i in jth term. The data in `polynomial_regression_samples.mat` (100000×5) with columns corresponding to the 5 attributes of the peach. Use 4-fold cross-validation to decide which of $D \in \{0, 1, 2, 3, 4, 5\}$ is the best fit for the data provided. Specifically, the best fit is defined as the choice of parameters that has the lowest mean squared error averaged across the four validation folds. For this part, compute the polynomial coefficients via ridge regression with penalty $\lambda = 0.1$, instead of ordinary least squares. You may not use `numpy.linalg.lstsq`, but you can use other linear algebra functions, such as `numpy.linalg.solve`. You should implement the 4-fold cross-validation using only `numpy` (i.e. do not use `sci-kit learn` or another machine learning library). The `assemble_feature` function provided in `hw1_prob5.py` can be used to construct the data matrix representing the D-multivariate polynomial basis functions from the 5 raw features.

Hint: To implement ridge regression using `numpy.linalg.solve`, consider the matrix equation you get by setting the gradient of the ridge regression loss function to zero.

Solution: Please refer to the next part for the general implementation.

(g) Now redo the previous part, but use 4-fold cross-validation on all combinations of $D \in \{1, 2, 3, 4, 5\}$ and $\lambda \in \{0.05, 0.1, 0.15, 0.2\}$ - this is referred to as a grid search. Find the best D and λ that best explains the data using ridge regression. Print the average training/validation error per sample for all D and λ.

Solution: Minimum of cross validation error happens at $D = 4$ and $\lambda = 0.15$.

```
Average train error:
[[0.05856762066 0.05856762225 0.05856762491 0.05856762862]
 [0.05848948229 0.0584902034  0.05849122    0.05849243261]
 [0.05848702853 0.05848893458 0.05849036454 0.05849178744]
 [0.05848700898 0.05848892445 0.05849035741 0.05849178171]
 [0.05848700861 0.05848892426 0.05849035728 0.05849178161]]
Average valid error:
[[0.05857468536 0.0585746856  0.05857468691 0.05857468927]
 [0.05852112813 0.05852038752 0.05852010745 0.05852016181]
 [0.0585210268  0.05852032221 0.05852006042 0.0585201252 ]
 [0.05852102354 0.05852032035 0.05852005896 0.05852012386]
 [0.05852102357 0.05852032036 0.05852005896 0.05852012386]]
```

```python
data = spio.loadmat('polynomial_regression_samples.mat', squeeze_me=True)
data_x = data['x']
data_y = data['y']
Kc = 4  # 4-fold cross validation
KD = 5  # max D = 5
LAMBDA = [0.05, 0.1, 0.15, 0.2]

feat_x = 0


def lstsq(A, b, lambda_=0):
    return np.linalg.solve(A.T @ A + lambda_ * np.eye(A.shape[1]), A.T @ b)


def assemble_feature(x, D):
    n_feature = x.shape[1]
    Q = [(np.ones(x.shape[0]), 0, 0)]
    i = 0
    while Q[i][1] < D:
        cx, degree, last_index = Q[i]
        for j in range(last_index, n_feature):
            Q.append((cx * x[:, j], degree + 1, j))
        i += 1
    return np.column_stack([q[0] for q in Q])


def fit(D, lambda_):
    Ns = int(data_x.shape[0] * (Kc - 1) / Kc)  # training
    Nv = int(Ns / (Kc - 1))  # validation

    Etrain = np.zeros(4)
    Evalid = np.zeros(4)
    for c in range(4):
        valid_x = feat_x[c * Nv:(c + 1) * Nv]
        valid_y = data_y[c * Nv:(c + 1) * Nv]
        train_x = np.delete(feat_x, list(range(c * Nv, (c + 1) * Nv)), axis=0)
        train_y = np.delete(data_y, list(range(c * Nv, (c + 1) * Nv)))

        w = lstsq(train_x, train_y, lambda_=lambda_)
        Etrain[c] = np.mean((train_y - train_x @ w)**2)
        Evalid[c] = np.mean((valid_y - valid_x @ w)**2)

    return np.mean(Etrain), np.mean(Evalid)


def main():
xo    np.set_printoptions(precision=11)
    Etrain = np.zeros((KD, len(LAMBDA)))
    Evalid = np.zeros((KD, len(LAMBDA)))
    for D in range(KD):
        global feat_x
```

```
        feat_x = assemble_feature(data_x, D + 1)
        print(feat_x.shape)
        for i in range(len(LAMBDA)):
            Etrain[D, i], Evalid[D, i] = fit(D + 1, LAMBDA[i])

    print('Average train error:', Etrain, sep='\n')
    print('Average valid error:', Evalid, sep='\n')

    D, i = np.unravel_index(Evalid.argmin(), Evalid.shape)
    print("D =", D + 1)
    print("lambda =", LAMBDA[i])

if __name__ == "__main__":
    main()
```