CS 189 Introduction to Machine Learning
Fall 2019 Jennifer Listgarten & Stella Yu

HW 04

Due: October 10th, 2019

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 assignment on Gradescope, “HW4 Write-Up”. 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, “HW4 Code”.
3. If there is a test set, submit your test set evaluation results, “HW4 Test Set”.

2 Gaussian Features

In this question we will be exploring the difference between using the Gaussian function to augment our data directly with new features and using it as a kernel function. We might want to do something like this so that we can model larger classes of functions by considering linear combinations of basis functions $f(x) = \sum_{k=1}^{K} w_k \phi_k(x)$..., and we may want "handcraft" our features with functions $\phi(x)$ when we have intuition and insight about our data and its source.

(a) First we will attempt to augment our data using polynomial features to see if this is appropriate in the setting that our data comes from Gaussian distributions. Implement the function **polynomial_augmentation** which takes as input a data matrix with only one feature and return a data matrix augmented with polynomial features ($\phi_k(x) = x^k$) of degree up to **k**. Then, use the function **linear_regression** which performs linear regression on a data matrix and plots the resulting curve along side the data points. What do you observe about the fitted curve? What happens as the degree of the polynomial features increases?

(b) Now, we will use the Gaussian function (https://en.wikipedia.org/wiki/Gaussian_function) to augment our data matrix. The idea is that our data comes from a linear combination of a set number of Gaussian functions, each with a center. Analyze the plot of data to find 3 centers that seem like they would be good centers (hint: some of the Gaussians may be negatively weighted and be upside down). Then, implement the function **rbf_augmentation** which takes as input a data matrix with only one feature and outputs a data matrix augmented with the values of the Gaussian functions centered at the chosen centers with the data as input ($\phi_k(x) = e^{-\frac{1}{2}||x-c_k||}$). This function will take as input

$$\mathbf{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

and output

$$\begin{pmatrix} x_1 & e^{-\frac{1}{2}||x_1-c_1||} & e^{-\frac{1}{2}||x_1-c_2||} & e^{-\frac{1}{2}||x_1-c_3||} \\ \vdots & \vdots & \vdots & \vdots \\ x_n & e^{-\frac{1}{2}||x_n-c_1||} & e^{-\frac{1}{2}||x_n-c_2||} & e^{-\frac{1}{2}||x_n-c_3||} \end{pmatrix}$$

where c_1, c_2, c_3 are your centers. One problem is that with this augmentation there is an implicit assumption about the variance of the Gaussians that our data comes from. To resolve this assumption try augmenting with 5, 10, and 15 as the values of the variance and plot the best one fitted curve.

3 Probabilistic Principal Components Analysis (PPCA)

In this question, we explore a more general form of PCA, called PPCA. We start with a latent variable model

$$y = Wx + \mu + z$$

for $x \sim \mathcal{N}(0, I)$, noise $z \sim \mathcal{N}(0, \Sigma)$, and constant μ, W. We will constrain ourselves to isotropic noise i.e., $\Sigma = \sigma^2 I$.

(a) Characterize the distribution of y conditioned on x.

(b) Now consider i.i.d. $x \sim \mathcal{N}(0, I)$. Now characterize the distribution of y.

4 Canonical Correlation Analysis

Assume that you have a database of images of the words typed in two different fonts. $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n \times d}$ corresponds to the dataset of font 1 and font 2 respectively. Think of the database $\mathbf{X}$ as being composed on n independent draws (samples) from a random variable $\mathbf{x} \in \mathbb{R}^d$, and similarly $\mathbf{Y}$ as n draws from a random variable $\mathbf{y}$. Your goal is to use machine learning to build a text recognition of word images.

(a) Explain why you would want to consider using CCA in this problem.

(b) Assume that $\mathbf{X}$ and $\mathbf{Y}$ include zero-mean features of the word images. Given two vectors $u, v \in \mathbb{R}^d$, what is the correlation coefficient of the projected variables? Correlation coefficient between two scalar random variables P and Q is computed by:

$$\rho(P, Q) = \frac{cov(P, Q)}{\sigma_P \sigma_Q}$$

(c) Assume that the features of matrix $\mathbf{X}$ are rescaled to have values between -1 and 1. How does this change the correlation coefficient?

5 Total Least Squares

In most of the models we have looked at so far, we've accounted for noise in the observed $\mathbf{y}$ measurement and adjusted accordingly. However, in the real world it could easily be that our feature matrix $\mathbf{X}$ of data is also corrupted or noisy. Total least squares is a way to account for this. Whereas previously we were minimizing the $\mathbf{y}$ distance from the data point to our predicted line because we had assumed the features were definitively accurate, now we are minimizing the entire distance from the data point to our predicted line. In this problem we will explore the mathematical intuition for the TLS formula. We will then apply the formula to adjusting the lighting of an image which contains noise in its feature matrix due to inaccurate assumptions we make about the image, such as the image being a perfect sphere.

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{y} \in \mathbb{R}^n$ be the measurements. Recall that in the least squares problem, we want to solve for $\mathbf{w}$ in $\min_{\mathbf{w}} ||\mathbf{X}\mathbf{w} - \mathbf{y}||$. We measure the error as the difference between $\mathbf{X}\mathbf{w}$ and $\mathbf{y}$, which can be viewed as adding an error term $\boldsymbol{\epsilon}_{\mathbf{y}}$ such that the equation $\mathbf{X}\mathbf{w} = \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}}$ has a solution:

$$\min_{\boldsymbol{\epsilon}_{\mathbf{y}},\mathbf{w}} \|\boldsymbol{\epsilon}_{\mathbf{y}}\|_2, \text{ subject to } \mathbf{X}\mathbf{w} = \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}} \tag{1}$$

Although this optimization formulation allows for errors in the measurements of $\mathbf{y}$, it does not allow for errors in the feature matrix $\mathbf{X}$ that is measured from the data. In this problem, we will explore a method called *total least squares* that allows for both error in the matrix $\mathbf{X}$ and the vector $\mathbf{y}$, represented by $\boldsymbol{\epsilon}_{\mathbf{X}}$ and $\boldsymbol{\epsilon}_{\mathbf{y}}$, respectively. For convenience, we absorb the negative sign into $\boldsymbol{\epsilon}_{\mathbf{y}}$ and $\boldsymbol{\epsilon}_{\mathbf{X}}$ and define true measurements $\mathbf{y}$ and $\mathbf{X}$ like so:

$$\mathbf{y}^{true} = \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}} \tag{2}$$

$$\mathbf{X}^{true} = \mathbf{X} + \boldsymbol{\epsilon}_{\mathbf{X}} \tag{3}$$

Specifically, the total least squares problem is to find the solution for $\mathbf{w}$ in the following minimization problem:

$$\min_{\boldsymbol{\epsilon}_{\mathbf{y}},\boldsymbol{\epsilon}_{\mathbf{X}},\mathbf{w}} \left\|[\boldsymbol{\epsilon}_{\mathbf{X}}, \boldsymbol{\epsilon}_{\mathbf{y}}]\right\|_F^2, \text{ subject to } (\mathbf{X} + \boldsymbol{\epsilon}_{\mathbf{X}})\mathbf{w} = \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}} \tag{4}$$

where the matrix $[\boldsymbol{\epsilon}_{\mathbf{X}}, \boldsymbol{\epsilon}_{\mathbf{y}}]$ is the concatenation of the columns of $\boldsymbol{\epsilon}_{\mathbf{X}}$ with the column vector $\mathbf{y}$. Recall the subscript F from discussion denotes the Frobenius norm of a matrix. Notice that the minimization is over $\mathbf{w}$ because it's a free parameter, and it does not necessarily have to be in the objective function. Intuitively, this equation is finding the smallest perturbation to the matrix of data points $\mathbf{X}$ and the outputs $\mathbf{y}$ such that the linear model can be solved exactly. The constraint in the minimization problem can be rewritten as

$$[\mathbf{X} + \boldsymbol{\epsilon}_{\mathbf{X}}, \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}}]\begin{bmatrix}\mathbf{w} \\ -1\end{bmatrix} = \mathbf{0} \tag{5}$$

(a) Let the matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $\mathbf{y} \in \mathbb{R}^n$ and note that $\epsilon_X \in \mathbb{R}^{n\times d}$ and $\boldsymbol{\epsilon}_{\mathbf{y}} \in \mathbb{R}^n$. Assuming that $n > d$ and $\text{rank}(\mathbf{X} + \boldsymbol{\epsilon}_{\mathbf{X}}) = d$, **explain why rank**$([\mathbf{X} + \boldsymbol{\epsilon}_{\mathbf{X}}, \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}}]) = d$**.**

(b) For the solution $\mathbf{w}$ to be unique, the matrix $[\mathbf{X} + \boldsymbol{\epsilon}_{\mathbf{X}}, \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}}]$ must have exactly d linearly independent columns. Since this matrix has d+1 columns in total, it must be rank-deficient by 1. Recall that we can rewrite $[\mathbf{X}, \mathbf{y}]$ as its SVD decomposition $U\Sigma V^\top$ for orthonormal U, V and diagonal Σ. Eckart-Young-Mirsky Theorem tells us that the closest lower-rank matrix in the Frobenius norm is obtained by discarding the smallest singular values. Therefore, the matrix $[\mathbf{X} + \boldsymbol{\epsilon}_{\mathbf{X}}, \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}}]$ that minimizes

$$\left\|[\boldsymbol{\epsilon}_{\mathbf{X}}, \boldsymbol{\epsilon}_{\mathbf{y}}]\right\|_F^2 = \left\|[\mathbf{X}^{true}, \mathbf{y}^{true}] - [\mathbf{X}, \mathbf{y}]\right\|_F^2$$

is given by

$$[\mathbf{X} + \boldsymbol{\epsilon}_{\mathbf{X}}, \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}}] = U \begin{bmatrix} \Sigma_d & & & \\ & 0 & & \\ & & \dots & \\ & & & 0 \end{bmatrix} V^\top$$

Suppose we express the SVD of $[\mathbf{X}, \mathbf{y}]$ in terms of submatrices and vectors:

$$[\mathbf{X}, \mathbf{y}] = \begin{bmatrix} \mathbf{U}_{xx} & \mathbf{u}_{xy} \\ \mathbf{u}_{yx}^\top & u_{yy} \end{bmatrix} \begin{bmatrix} \Sigma_d & \\ & \sigma_{d+1} \end{bmatrix} \begin{bmatrix} \mathbf{V}_{xx} & \mathbf{v}_{xy} \\ \mathbf{v}_{yx}^\top & v_{yy} \end{bmatrix}^\top$$

$\mathbf{u}_{xy} \in \mathbb{R}^{n-1}$ is the first $(n-1)$ elements of the $(d+1)$-th column of $\mathbf{U}$, $\mathbf{u}_{yx}^\top \in \mathbb{R}^d$ is the first d elements of the n-th row of $\mathbf{U}$, u_{yy} is the n-th element of the $(d+1)$-th column of $\mathbf{U}$, $\mathbf{U}_{xx} \in \mathbb{R}^{(n-1)\times d}$ is the $(n-1)\times d$ top left submatrix of $\mathbf{U}$.

Similarly, $\mathbf{v}_{xy} \in \mathbb{R}^d$ is the first d elements of the $(d+1)$-th column of $\mathbf{V}$, $\mathbf{v}_{yx}^\top \in \mathbb{R}^d$ is the first d elements of the $(d+1)$-th row of $\mathbf{V}$, v_{yy} is the $(d+1)$-th element of the $(d+1)$-th column of $\mathbf{V}$, $\mathbf{V}_{xx} \in \mathbb{R}^{d\times d}$ is the $d \times d$ top left submatrix of $\mathbf{V}$. σ_{d+1} is the $(d+1)$-th eigenvalue of $[\mathbf{X}, \mathbf{y}]$. Σ_d is the diagonal matrix of the d largest singular values of $[\mathbf{X}, \mathbf{y}]$

Using this information show that

$$[\boldsymbol{\epsilon}_{\mathbf{X}}, \boldsymbol{\epsilon}_{\mathbf{y}}] = -\begin{bmatrix} \mathbf{u}_{xy} \\ u_{yy} \end{bmatrix} \sigma_{d+1} \begin{bmatrix} \mathbf{v}_{xy} \\ v_{yy} \end{bmatrix}^\top$$

(c) **Using the result from the previous part and the fact that v_{yy} is not 0 (see notes on Total Least Squares), find a nonzero solution to $[\mathbf{X} + \boldsymbol{\epsilon}_{\mathbf{X}}, \mathbf{y} + \boldsymbol{\epsilon}_{\mathbf{y}}] \begin{bmatrix} \mathbf{w} \\ -1 \end{bmatrix} = \mathbf{0}$ and thus solve for w in Equation** (5)**.**

HINT: Looking at the last column of the product $[\mathbf{X}, \mathbf{y}]\mathbf{V}$ *may or may not be useful for this problem, depending on how you solve it.*

(d) From the previous part, you can see that $\begin{bmatrix} \mathbf{w} \\ -1 \end{bmatrix}$ is a right-singular vector of $[\mathbf{X}, \mathbf{y}]$. **Show that**

$$(\mathbf{X}^\top\mathbf{X} - \sigma_{d+1}^2 I)\mathbf{w} = \mathbf{X}^\top\mathbf{y} \tag{14}$$

(e) In this problem, we will use total least squares to approximately learn the lighting in a photograph, which we can then use to paste new objects into the image while still maintaining the realism of the image. You will be estimating the lighting coefficients for the interior of St. Peter's Basilica, and you will then use these coefficients to change the lighting of an image of a

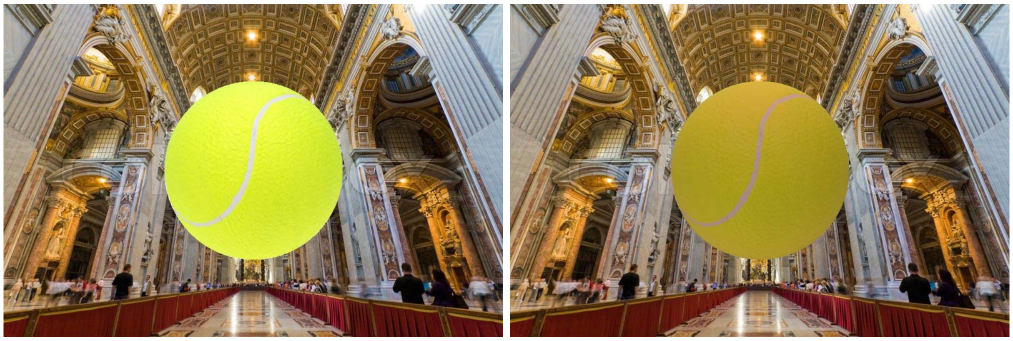

Figure 1: Tennis ball pasted on top of image of St. Peter's Basilica without lighting adjustment (left) and with lighting adjustment (right)

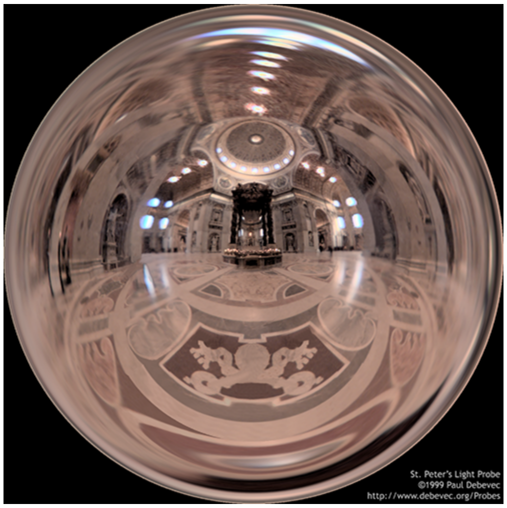

Figure 2: Image of a spherical mirror inside of St. Peter's Basilica

tennis ball so that it can be pasted into the image. In Figure 1, we show the result of pasting the tennis ball in the image without adjusting the lighting on the ball. The ball looks too bright for the scene and does not look like it would fit in with other objects in the image.

To convincingly add a tennis ball to an image, we need to need to apply the appropriate lighting from the environment onto the added ball. To start, we will represent environment lighting as a spherical function $\mathbf{f}(\mathbf{n})$ where $\mathbf{n}$ is a 3 dimensional unit vector ($||\mathbf{n}||_2 = 1$), and $\mathbf{f}$ outputs a 3 dimensional color vector, one component for red, green, and blue light intensities. Because $\mathbf{f}(\mathbf{n})$ is a spherical function, the input $\mathbf{n}$ must correspond to a point on a sphere. The function $\mathbf{f}(\mathbf{n})$ represents the total incoming light from the direction $\mathbf{n}$ in the scene. The lighting function of a spherical object $\mathbf{f}(\mathbf{n})$ can be approximated by the first 9 spherical harmonic basis functions.

The first 9 un-normalized spherical harmonic basis functions are given by:

$$L_1 = 1$$
$$L_2 = y$$
$$L_3 = x$$

$$\begin{aligned}
L_4 &= z \\
L_5 &= xy \\
L_6 &= yz \\
L_7 &= 3z^2 - 1 \\
L_8 &= xz \\
L_9 &= x^2 - y^2
\end{aligned}$$

where $\mathbf{n} = [x, y, z]^\top$. The lighting function can then be approximated as

$$\mathbf{f}(\mathbf{n}) \approx \sum_{i=1}^{9} \boldsymbol{\gamma}_i L_i(\mathbf{n})$$

$$\begin{bmatrix} — & \mathbf{f}(\mathbf{n_1}) & — \\ — & \mathbf{f}(\mathbf{n_2}) & — \\ & \vdots & \\ — & \mathbf{f}(\mathbf{n_n}) & — \end{bmatrix}_{n\times 3} = \begin{bmatrix} L_1(\mathbf{n_1}) & L_2(\mathbf{n_1}) & ... & L_9(\mathbf{n_1}) \\ L_1(\mathbf{n_2}) & L_2(\mathbf{n_2}) & ... & L_9(\mathbf{n_2}) \\ & \vdots & & \\ L_1(\mathbf{n_n}) & L_2(\mathbf{n_n}) & ... & L_9(\mathbf{n_n}) \end{bmatrix}_{n\times 9} \begin{bmatrix} — & \boldsymbol{\gamma}_1 & — \\ — & \boldsymbol{\gamma}_2 & — \\ & \vdots & \\ — & \boldsymbol{\gamma}_9 & — \end{bmatrix}_{9\times 3}$$

where $L_i(\mathbf{n})$ is the ith basis function from the list above.

The function of incoming light $\mathbf{f}(\mathbf{n})$ can be measured by photographing a spherical mirror placed in the scene of interest. In this case, we provide you with an image of the sphere as seen in Figure 2. In the code provided, there is a function extractNormals(img) that will extract the training pairs $(\mathbf{n}_i, \mathbf{f}(\mathbf{n}_i))$ from the image. An example using this function is in the code.

Use the spherical harmonic basis functions to create a 9 dimensional feature vector for each sample. Use this to formulate an ordinary least squares problem and solve for the unknown coefficients $\boldsymbol{\gamma}_i$. Report the estimated values for $\boldsymbol{\gamma}_i$ and include a visualization of the approximation using the provided code. The code provided will load the images, extracts the training data, relights the tennis ball with incorrect coefficients, and saves the results. Your task is to compute the basis functions and solve the least squares problems to provide the code with the correct coefficients. To run the starter code, you will need to use Python with `numpy` and `scipy`. Because the resulting data set is large, we reduce it in the code by taking every 50th entry in the data. This is done for you in the starter code, but you can try using the entire data set or reduce it by a different amount.

(f) When we extract from the data the direction $\mathbf{n}$ to compute $(\mathbf{n}_i, \mathbf{f}(\mathbf{n}_i))$, we make some approximations about how the light is captured on the image. We also assume that the spherical mirror is a perfect sphere, but in reality, there will always be small imperfections. Thus, our measurement for $\mathbf{n}$ contains some error, which makes this an ideal problem to apply total least squares. **Solve this problem with total least squares by allowing perturbations in the matrix of basis functions. Report the estimated values for $\boldsymbol{\gamma}_i$ and include a visualization of the approximation.** The output image will be visibly wrong, and we'll explore how to fix this problem in the next part. Your implementation may only use the SVD and the matrix inverse functions from the linear algebra library in `numpy` as well as `np.linalg.solve`.

(g) In the previous part, you should have noticed that the visualization is drastically different than the one generated using least squares. Recall that in total least squares we are minimizing $||[\boldsymbol{\epsilon}_{\mathbf{X}}, \boldsymbol{\epsilon}_{\mathbf{y}}]||_F^2$. Intuitively, to minimize the Frobenius norm of components of both the inputs and outputs, the inputs and outputs should be on the same scale. However, this is not the case here. Color values in an image will typically be in $[0, 255]$, but the original image had a much larger range. We compressed the range to a smaller scale using tone mapping, but the effect of the compression is that relatively bright areas of the image become less bright. As a compromise, we scaled the image colors down to a maximum color value of 384 instead of 255. Thus, the inputs here are all unit vectors, and the outputs are 3 dimensional vectors where each value is in $[0, 384]$. **Propose a value by which to scale the outputs $\mathbf{f}(\mathbf{n}_i)$ such that the values of the inputs and outputs are roughly on the same scale. Solve this scaled total least squares problem, report the computed spherical harmonic coefficients and provide a rendering of the relit sphere.**