Practical

exercises

October 16, 2019

1. (CS 189 Introduction to Machine Learning Spring 2018 HW5) TLS practical

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

(a)

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

PRACTICAL EXERCISE

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 Basillica, and you will then use these coefficients to change the lighting of an image of a tennis ball so that it can be pasted into the image. In a figure, 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 $f(n)$ where n is a 3 dimensional unit vector ($\|u\|_2 = 1$), and f outputs a 3 dimensional color vector, one component for red, green, and blue light intensities. Because $f(n)$ is a spherical function, the input n must correspond to a point on a sphere. The function $f(n)$ represents the total incoming light from the direction n in the scene. The lighting function of a spherical object $f(n)$ can be approximated by the first 9 spherical harmonic basis functions.

The first 9 unnormalized sperical harmonic basis functions are given by:

$$L_1 = 1$$
$$L_2 = y$$
$$L_3 = x$$
$$L_4 = z$$
$$L_5 = xy$$
$$L_6 = yz$$
$$L_7 = 3z^2 - 1$$
$$L_8 = xz$$
$$L_9 = x^2 - y^2$$

where $n = [x, y, z]^\top$. The lightning function can be approximated as

$$f(n) \approx \sum_{i=1}^{9} \gamma_i L_i(n)$$

$$\begin{bmatrix} f(n_1) \\ f(n_2) \\ \vdots \\ f(n_n) \end{bmatrix}_{n\times 3} = \begin{bmatrix} L_1(n_1) & L2(n_1) & \dots & L_9(n_1) \\ L_1(n_2) & L2(n_2) & \dots & L_9(n_2) \\ & \vdots & & \\ L_1(n_n) & L2(n_n) & \dots & L_9(n_n) \end{bmatrix}_{n\times 9} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_9 \end{bmatrix}_{9\times 3}$$

where $L_i(n)$ is the i-th basis function from the list above.

The function of incoming light $f(n)$ 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 a figure. In the code provided, there is a function extractNormals(img) that will extract the training pairs $(n_i; f(n_i))$ 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 γ_i. Report the estimated values for γ_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.

(b) PRACTICAL EXERCISE When we extract from the data the direction n to compute $(ni; f(ni))$, 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 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 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.

(c) PRACTICAL EXERCISE 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 $\|\epsilon_X, \epsilon_y\|_{\text{Fro}}^2$. 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 $f(n_i)$ 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.

2. (CS189 Summer 2018 - Introduction to Machine Learning HW3) OR (CS189 Spring 2018 - Introduction to Machine Learning HW5 - also has solutions) **(PCA and Random Projections)**

   (a) PRACTICAL EXERCISE In the next few parts, we visualize the effect of PCA projections and random projections on a classification task. You are given 3 datasets in the data folder and a starter code. Use the starter code to load the three datasets one by one. Note that there are two unique values in y. Visualize the features of X these datasets using (1) Top-2 PCA components, and (2) 2-dimensional random projections. Use the code to project the features to 2 dimensions and then scatter plot the 2 features with

a different color for each class. Note that you will obtain 2 plots for each dataset (total 6 plots for this part). Do you observe a difference in PCA vs random projections? Do you see a trend in the three datasets?

(b) PRACTICAL EXERCISE For each dataset, we will now fit a linear model on different projections of features to perform classification. The code for fitting a linear model with projected features and predicting a label for a given feature, is given to you. Use these functions and write a code that does prediction in the following way: (1) Use top k-PCA features to obtain one set of results, and (2) Use k-dimensional random projection to obtain the second set of results (take average accuracy over 10 random projections for smooth curves). Use the projection functions given in the starter code to select these features. You should vary k from 1 to d where d is the dimension of each feature xi. Plot the accuracy for PCA and Random projection as a function of k. Comment on the observations on these accuracies as a function of k and across different datasets.

(c) PRACTICAL EXERCISE Now plot the singular values for the feature matrices of the three datasets. Do you observe a pattern across the three datasets? Does it help to explain the performance of PCA observed in the previous parts?