CSCC11 Introduction to Machine Learning, Winter 2024

Assignment 3

This assignment makes use of material up to week 12 of the course. Like the first and second assignments, it comprises two parts, a written part and a coding part. The written work asks for you to derive some solutions to problems and for you to demonstrate an understanding of certain concepts. The written and programming parts will have different deadlines (see below). Submission instructions can be found at the end of the handout.

We remind you that the work you hand in for this assignment must be your own. Students should not make use of large language models for programming assistants like Copilot.

Theory Questions (Due April 8, 11:59pm)

Question 1. Bias-Variance Decomposition and trade off (10 marks)

In the course for Bayesian model selection, we discussed how model selection depends on model complexity (described in detail in the course notes in Section 12.3). We discussed how increasing model complexity results in reducing the error as measured by the model likelihood during training, while increasing the divergence between the posterior and prior. This phenomenon is also characterized by the so-called bias-variance trade-off in machine learning. The bias of a model represents the average difference between the predicted values and the true values. The variance of a model, on the other hand, measures the variability of the predicted values around their mean. As the complexity of the model increases, the variance will tend to increase and the bias will tend to decrease.
With this context, in terms of the mean-squared-error (MSE) for the model $y = f(x) + \eta$, for which $E[\eta] = 0$ and $var(\eta) = \sigma^2$, answer the questions below:

a Proof that the expected MSE of the prediction of a LS regression model will be the sum of bias and variance. Hint: You should start by expanding $E[(y-f(x))^2]$. [You will find it helpful to familiarize yourself with bias-variance decomposition in Section 3-2 of Bishop's book: `https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf`]

b Briefly, discuss the concepts of under-fitting and over-fitting in terms of bias and variance. E.g., how do you expect the relative magnitudes of bias and variance to coincide with underfitting and/or overfitting.

c Intuitively, discuss the relation between bias and variance with respect to data size. For example, as the dataset size grows very large, how do you expect bias and variance to behave?

Question2. Principal Component Analysis (10 marks)

Beyond PCA's role in dimensionality reduction and visulization, the PCA algorithm finds extensive utility across various domains, such as image processing. For example, PCA has been employed for image rotation, where it assists in aligning images and correcting for orientation variations. This application is particularly valuable in fields of medical imaging where precise alignment of images is crucial for analysis and visualization purposes.

1. Figure 1 displays a magnetic resonance image (MRI) alongside its corresponding masked version. In this context, a mask is a binary image that highlights areas of interest with white pixels, while the rest of the image remains black. Creating such masks can be accomplished through various methods, including manual annotation and automated segmentation techniques such as intensity thresholds, texture analysis, machine learning algorithms, or a combination thereof. The masked image isolates regions of interest from the background, enabling us to concentrate specifically on the relevant areas for our analysis. In this question intuitevly discuss on employing PCA with the masked image to determine the rotation matrix required to align the MRI image with the Y-axis.

2. Missing data refers to the absence of values in a dataset that can occur due to various reasons, such as equipment malfunction, human error, or simply because certain information was not collected. In such cases, PCA can help in data completion or filling the missing values.

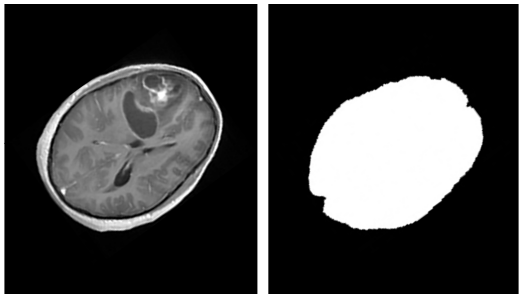

Figure 1: MRI image and its corresponding masked image.

To formulate this, let's assume we have images with d pixels represented as 2D arrays. Now suppose that we rearrange the pixels of each image image into a single column vector (e.g., by stacking the columns of the array on top of each other). Let's denote N such images as column vectors $\{\boldsymbol{y}_i\}_{i=1}^N$, where $\boldsymbol{y}_i \in \mathbb{R}^d$. And suppose we use PCA to find a low-dimensional representation of the images, given by $\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{y}$, where the matrix $\boldsymbol{W} \in \mathbb{R}^{d\times K}$ maps an image vector $\boldsymbol{y}$ into the K-dimensional subspace, the coefficients of which are $\boldsymbol{x}$. As discussed in the notes, the columns of W are the K principal eigenvectors of the covariance matrix of the image data.

Given a point $\boldsymbol{x} \in \mathbb{R}^K$ in the subspace, we can reconstruct the corresponding image using $\boldsymbol{y} = \boldsymbol{W}x$. PCA learns the mapping by trying to minimize the approximation error between the original data and the reconstructed images. But suppose we are given just a subset of the pixels in an image. Perhaps we only see the right half of the image, or we only see every other row of the image. It turns out that you can still find a subspace representation for this partially observed image, and in doing so, then reconstruct the image to fill in the missing pixels.

Your job will be to formulate this task, i.e., to formulate the task of filling in the unobserved (masked) part of an image. To begin, let's model the partial observation with a matrix that projects out the pixels we do not observe. Let $\tilde{\boldsymbol{y}} = \boldsymbol{P}\boldsymbol{y}$, where $\tilde{\boldsymbol{y}}$ only includes the observed pixels of $\boldsymbol{y}$. Here $\boldsymbol{P} \in \mathbb{R}^{m\times d}$ is a projection operator to a subspace of m observed pixels. You can think of $\boldsymbol{P}$ as comprising just M rows of the $d \times d$ identity matrix, so it selects only a subset of dimensions.

Please answer the following:

(a) Assume that images are vectorized (flattened into columns) in row-major order. Then, when the bottom half of the image is missing, show that $\boldsymbol{P} = \big[\boldsymbol{I}_m|\boldsymbol{0}\big]$, where $\boldsymbol{I}_m$ is an identity matrix.

(b) Now, given a partially observed image $\tilde{\boldsymbol{y}}$, derive the optimal subspace projection of the image that minimizes the squared reconstruction error, given by $||\boldsymbol{P}\boldsymbol{W}\boldsymbol{x} - \tilde{\boldsymbol{y}}||^2$.

Question 3. Clustering (10 marks)

In lectures we discussed the use of the K-Means algorithm to cluster data into K number of groups (see Section 16.1), for which we used Euclidean distance to measure the proximity of each data point to the centroid of its respective cluster.

1. When dealing with a dataset featuring varying scales among its features, normalization (or feature reweighting) is critical to prevent any single feature from becoming dominant. Normalization seeks to equalize the importance of each feature by transforming them to exhibit a distribution with a mean of zero and a standard deviation of one. Derive a mathematical expression to incorporate feature normalization into the K-Means objective function.

2. In addition to the Euclidean distance, alternative distance functions such as L1 and cosine distance can be utilized within the K-Means objective function. L1 distance, also known as Manhattan distance, is a measure of the distance calculated by summing the absolute differences between the coordinates of corresponding points. In other words, it's the sum of the absolute differences of the Cartesian coordinates. For instance, in two dimensions, the L1 distance between points (x_1, y_1) and (x_2, y_2) is given by $|x_1 - x_2| + |y_1 - y_2|$. Cosine distance measures the angular distance between two vectors in a multidimensional

space. It's calculated as the cosine of the angle between the vectors. The formula for cosine similarity between two vectors $\mathbf{u}$ and $\mathbf{v}$ is given by:

$$\text{cosine\_similarity}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}$$

where $\mathbf{u} \cdot \mathbf{v}$ denotes the dot product of $\mathbf{u}$ and $\mathbf{v}$, and $\|\mathbf{u}\|$ and $\|\mathbf{v}\|$ are the Euclidean norms (magnitudes) of vectors $\mathbf{u}$ and $\mathbf{v}$ respectively.

Derive the cosine distance into the K-Means objective function.

3. Let's suppose that numerous features within the K-Means algorithm are redundant. For instance, there exists a significant correlation among features, potentially leading the K-Means algorithm to favor highly correlated features. What recommendations do you propose to enhance the model's performance?

Programming Questions (Due April 8, 11:59pm)

Question 1. Dimensionality reduction and visualization (10 marks)

For this programming section, we will use the MNIST[1]. As we discussed before, the dataset comprising images of handwritten digits, centered and resized to 28×28 pixels. Each image has been saved as a vector with the dimension of $784 = 28 \times 28$, and is associated with a label, namely, digits 0 through 9. We also used previously introduced machine learning library for python programming language, *scikit-learn*, often referred to as *sklearn*. Please complete the code based on the text provided for each section in the Python file. The blank sections that need completion are indicated by `XXX`. The completed python files do not need to be submitted.

1. As we discussed, each digit in the dataset can be represented as a 784-dimensional vector, but visualizing data in 784 dimensions is not practically feasible. Here, we aim to reduce this to 2 dimensions for visualization purposes. We will utilize the first section of the *PCA.py* script to accomplish this task. After filling in the necessary sections of the script, you should be able to generate a 2D plot of the data. Upon examining the plot, discuss whether we can identify distinct clusters within the data. Do you expect differen runs of applying PCA results in the same output? Why?

2. In the second section of the *PCA.py* script, we will plot the cumulative variance explained by adding more principal components. Determine the number of components needed to capture 90% of the variance in the data.

3. In this section, we will use the third section of the *PCA.py* script to reconstruct input images using the first 50,100, and 500 principal components. Visualize the output and discuss the properties of added components?

4. As discussed in theoretical question 2, PCA can be employed to address missing data. In this section, we introduce missing data by removing the bottom half of an image shown in Figure 2 Using the mathematics derived from 2.b, complete the fourth section of the PCA.py script to reconstruct the input image. Visualize the output and discuss how increasing the number of principal components changes the reconstructed image

Question 2. Segmentation with Clustering (10 marks)

Image segmentation stands as a fundamental task in thecomputer vision, crucial for pinpointing objects and delineating boundaries within images. Essentially, image segmentation is the assignment of a label to each pixel in an image, grouping pixels with similar attributes together. This process finds extensive application in everyday scenarios like autonomous vehicles, facial recognition and detection, video surveillance, and satellite image analysis.
A multitude of algorithms can be enlisted to accomplish the segmentation of regions of interest within input data. Previously, a simple approach involved leveraging clustering algorithms atop RGB images. This method entailed grouping pixels based on their color similarity, allowing for the identification and isolation of distinct

[1]short for Modified National Institute of Standards and Technology

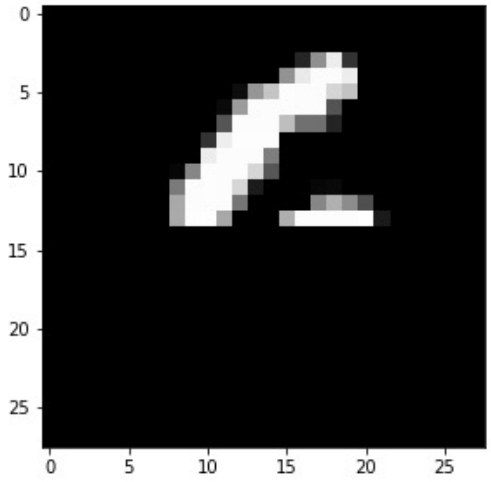

Figure 2: Partially observed data

regions within the image. In this question, we are looking to explore how to use KMeans and GMM clustering algorithm to segment an input image. We will use *sklearn* to perform clustering. Having this in mind, please answer below questions.

1. We will use the first section in *Clustering.py* to explore the argument for the KMeans algorithm. We are looking to segment an input image into 3 segments. Execute the algorithm and display the segmented image. Report the cluster center. If we run the algorithm three times, will we obtain consistent results? (Note: you should compare the center of the cluster). Will we consistently receive the same labels each time we run the model? Discuss your observation.

2. We will use the second section in *Clusteirng.py* to explore the argument for the GMM algorithm. We are interested to explore how full covariance shape to segment the input image into 3 segments. Show the results of the segmentation. Report the mean and covariance of each component.

Submission Instructions

Theory Questions This part may be done with pen and paper or a text editor (eg latex). Formatted answers are easier to read but likely require more time. In any case, your answers should be handed in electronically as a single pdf file (for all 3 questions). The file should be called `solution.pdf`, and it should be submitted to **Markus**. If you do work with pen and paper you could scan or take photos of your work then convert to pdf.

Programming Questions For this part, you need to answer the questions for each section and provide the requested plot for each part. Your answers should be submitted electronically as a single PDF file (for all sections). The file should be named `solutioncoding.pdf` and submitted to **Markus**

Note that in Markus you find **separate sections** dedicated to theory and programming parts of this assignment.