Introduction to Data Compression
A Linear Algebra Approach

Layan Ansari, Adam Pearl, Iñaki Arango

December 12, 2022

Cover Letter

(We will write something here for the final product). Reflect on challenges and successes in the cover letter; what you planned to accomplish but couldn't; show that you put in effort and that you took a lot away from this project; reflect on what you learned while doing this project and how it may have helped you discover a new passion or delve into a passion you already had; thank whoever did your peer review.

Contents

Chapter 1

Introduction

Data flows everywhere, it is in every product that we interact with daily. More than 2.5 quintillion bytes are created every day (that is 2.5 followed by 18 zeros) [Ale15]. For the next 5 years data is expected to grow by 40% every year [Mar22].

We need, and will continue to need, more powerful and faster computers capable of processing this increasing amount of information, and better storage systems to safekeep it.

There has been a lot of improvement hardware-wise in the last couple of decades, which increased the density of our storage systems. However, what if we could improve our information storage density through other, non-physical, means? What if we could store the same amount of "information" but with fewer "bits"? This would allow us to work in parallel with scientists researching physical systems.

This is where data compression comes in. It allows us to store the same amount of "information" in less bits, digits, characters, or whatever the most basic unit of storage is in our medium of choice. An early example of compression work is the Morse Code, which was invented in 1838 for use in the telegraph industry. Telegraph bandwidth was scarce, so shorter code words were assigned for common letters in the English alphabet, such as "e" and "t", that would be sent more often [Wol02].

Compression can also be useful for analysis and recognition of data patterns. Suppose that given a telgraph system, we had decided to make a picture for every single letter in the English alphabet, and instead of sending Morse code over a telegraph, we had sent the all the pixel information for the letter images every time we wanted to transmit a sentence across the wire. Not only would that have taken so much longer and wasted bandwidth (related to compression), but it would also have been way harder to recognize and would have been more error prone (people's caligraphy is different from one another). One would have to record all the pixel information for every letter, reconstruct it, and then decide on which letter it is: a computationally hard and intensive task for both, a computer and a human.

Morse code, an example of compression, shows how crucial it is to find reliable ways of storing and sharing our information in a way that minimizes space used and maximizes reliability.

Chapter 2

Mathematical Background

There are various tools at our disposal that we can use to study and perform data compression and analysis. Many of the common techniques that are used today are the outcome of collaboration between scientists and engineers focused on different branches of Math, such as Probability, Statistics, Information Theory, and Linear Algebra.

While we have focused on Linear Algebra this semester, it is necessary to know some introductory concepts from these other branches to understand basic data compression. In this section we will introduce these branches and explain some of their basic concepts.

2.1 Probability & Statistics

Probability is the study of the the likelihood of events, independently and given the occurrence of other events (e.g., given that it is cloudy, what is the probability that it will rain today?). Statistics is the discipline that concerns the collection, organization, analysis, interpretation, and presentation of data [Wik22].

2.1.1 Sample Space

The most basic concept in probability is that of the **sample space** and **events**.

> **Definition 2.1.1: Sample Space**
>
> The **sample space** is the set of all possible outcomes that an experiment can have.

Example 2.1.1 (Flipping a coin)
If we flip a coin then the sample space is $S = \{\text{Heads}, \text{Tails}\}$.

Example 2.1.2 (Rolling a die)
If we roll a single 6-sided die the sample space is $S = \{1, 2, 3, 4, 5, 6\}$.

2.1.2 Events

> **Definition 2.1.2: Event**
>
> **Events** are then mathematically defined as a subset of the sample space.

Example 2.1.3 (Rolling a die)

For example, in the 6-sided die example let M be the event we roll a 4 or greater. Then $M = \{4, 5, 6\} \subseteq S$.

In this experiment we state that we are rolling the die once, so we cannot make any statements about what happens if you first roll something, and the something else. In that case, once would have to redefine the experiment to say that each die roll has possibilities $R = \{1, 2, 3, 4, 5, 6\}$ and the total experiment with n roles has sample space $S = R^n$.

2.1.3 Probability

> **Definition 2.1.3: Probability**
>
> The **probability** of an event (denote $P(M)$ for an event M) is the measure of how likely the event is to occur as the outcome of a random experiment. It is the sum of the probabilities of each element of the set being the outcome of the performed experiment.

Probability is measured out of 1, which means that events certain to occur have a probability of 1 and events that will never occur have a probability of 0.

Since the sample space contains all the possible outcomes for the experiment, it is guaranteed that one the sample space's elements will be the outcome of the experiment, so

$$P(S) = 1$$

.

2.1.4 Random Variables

> **Definition 2.1.4: Random Variables**
>
> **Random variables** (also r.v.'s) are functions that map the outcome of a random experiment (in the sample space) to a measurable quantity (e.g., real number) or a mathematical object (e.g., matrix).

When the experiment is performed, and the outcome is mapped to a value, we say the random variable "crystallizes" to that value. The set of all values a random variable crystallizes to is called the support, denoted R_X for a r.v. X. A specified random variable crystallizing to a certain value is an event.

We normally denote random variables with capital letters and their crystallized values with lower case letters.

Example 2.1.4 (Flipping a coin)

Let us flip a fair coin with equal probabilities of landing heads or tails. The sample space is

then $S = \{H, T\}$ with $P(\{H\}) = P(\{T\}) = 0.5$.

Let X be a random variable that crystallizes to -1 if the coin lands Heads and to 1 if the coin lands Tails. Then $(X = -1)$ and $(X = 1)$ are events and we write $P(X = -1) = P(\{H\})$ and $P(X = 1) = P(\{T\})$.

If we where to write the probability function for X, where x is the crystallized value that X takes, we would get

$$P(X = x) = \begin{cases} 0.5 & \text{if } x = -1 \\ 0.5 & \text{if } x = 1 \end{cases}$$

2.1.5 Expectation

> **Definition 2.1.5: Expectation**
>
> The **expectation** (also called the mean) of a random variable X, denoted $\mathbb{E}(X)$, is the weighted average of the values that X can take. The weight for a value x is determined by the probability of X crystallizing to that value.
>
> We also write the variable with a bar on top to represent its mean. Its formula is
>
> $$\bar{X} = \mathbb{E}(X) = \sum_{x \in R_X} x \cdot P(X = x)$$

Example 2.1.5 (Rolling a die)

Expanding on the previous example of rolling a 6-sided die once. Let X be a random variable that crystallizes to the number rolled. Then $R_X = \{1, 2, 3, 4, 5, 6\}$ and $P(X = x) = \frac{1}{6}$ for all $x \in R_X$, since each number has the same probability of being rolled.

The expected value of X is then

$$\mathbb{E}(X) = \sum_{x \in R_X} x \cdot P(X = x) = \frac{1}{6} \cdot 1 + \frac{1}{6} \cdot 2 + \frac{1}{6} \cdot 3 + \frac{1}{6} \cdot 4 + \frac{1}{6} \cdot 5 + \frac{1}{6} \cdot 6 = \frac{21}{6} = 3.5$$
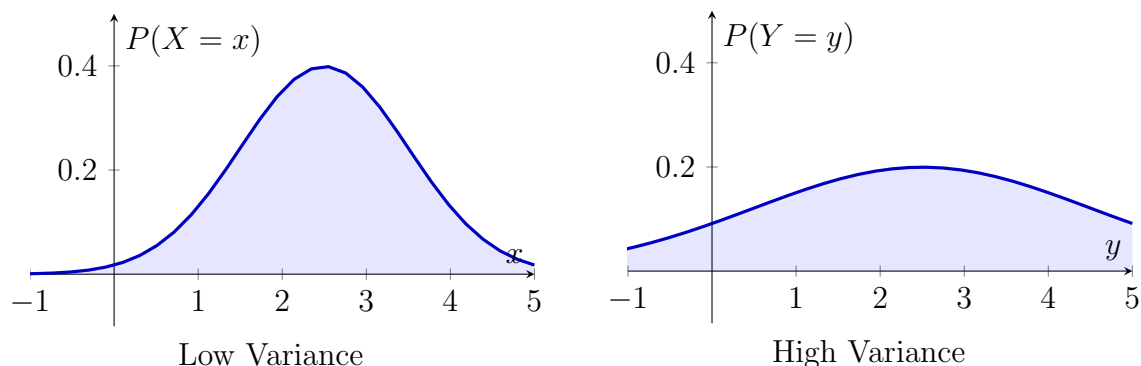
2.1.6 Variance

> **Definition 2.1.6: Variance**
>
> **Variance** is a measure of spread. It measures how far the values in the support of an r.v. are to the mean. For a r.v. X we denote its variance as $\text{Var}(X)$. The formula for variance is
>
> $$\text{Var}(X) = \mathbb{E}[(X - \bar{X})^2] = \mathbb{E}(X^2) - [\mathbb{E}(X)]^2$$

Below is a graphical comparison of between the probability function of a low variance r.v. X and the probability function of a high variance r.v. Y. The horizontal axis indicates a value in the support of the random variable and the vertical axis indicates the probability of the random variable taking that value.

Low Variance   High Variance
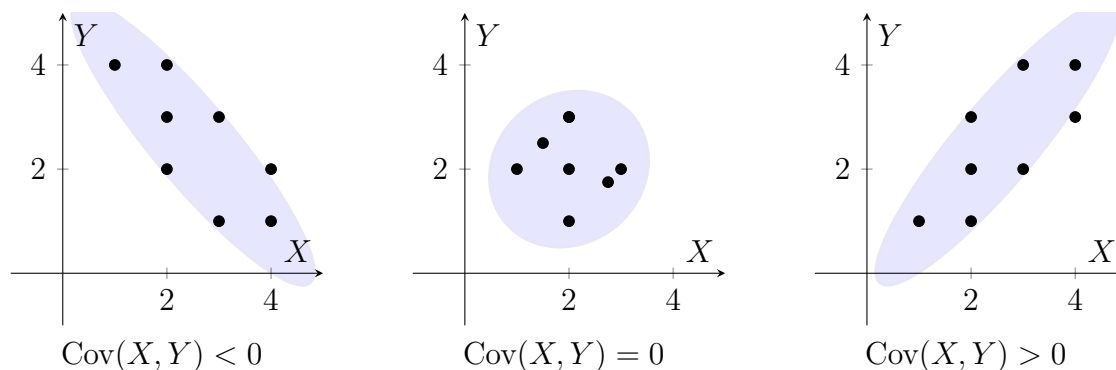
2.1.7 Covariance

> **Definition 2.1.7: Covariance**
>
> While the variance is the measure of spread for a single variables, the **covariance** examines the directional relationship between two variables [Ric07]. Its formula is
>
> $$\text{Cov}(X,Y) = \mathbb{E}[(X - \bar{X})(Y - \bar{Y})] = \mathbb{E}(X \cdot Y) - \mathbb{E}(X)\,\mathbb{E}(Y)$$

If greater values of one variable correspond to greater values of the other variable covariance is positive. If greater values of one variable correspond to lesser values of the other variable, covariance is negative.

Below is a graphical comparison of what negative, zero and positive covariance looks like between two random variables. Each point encodes one instance of the two variables crystallized together.

$\text{Cov}(X,Y) < 0$ $\text{Cov}(X,Y) = 0$ $\text{Cov}(X,Y) > 0$

If covariance is 0, we say that variables are "uncorrelated", and we interpret it as there being no linear relationship between the two.

Chapter 3

Start of the Story: What do we want to compress?

3.1 Handwriting Recognition

Communication is an essential part of relating to people, and one of the oldest and most accessible methods of communication within a given language is writing by hand. In spite of the major effort that has been expended to bring about a paper-free society, a very large number of paper-based documents are processed daily by computers all over the world in order to handle, retrieve, and store information. The problem is that the manual process used to enter the data from these documents into computers demands a great deal of time and money (Bortolozzi, de Souza Britto Jr., Oliveira and Morita, n.d.). These documents may need to be processed for a number of reasons, including historical record keeping, such as documenting significant social and political events. , among them historical documentation (e.g. digitally documenting culturally and historically significant documents and scripts, which until recently were more often than not handwritten or on print paper), recognition for medical prescriptions, or for tablet soft-wares to convert user's handwriting into digital text.

Thus, the task of handwriting recognition is the transcription of handwritten data into a digital format, and this task obviously benefits from data compression. The goal is to process handwritten data electronically with the same or nearly the same accuracy as humans (Gunter, n.d).

Basically, handwriting can be divided into two categories, cursive script and printed handwriting. Accuracy is the main problem in handwriting recognition for both categories because of the similar strokes and shapes some letters may possess. The software may have an inaccurate recognition of the letter, considering the possibility of the handwriting being illegible or some other factors (). One notable problem that makes this task difficult especially in cursive handwriting recognition is the fact that there may be no obvious character boundaries (the start and end of a character); compared to printed handwriting, it does not have gaps or spaces between each letter to know the start and stop of recognition per character (). This issue is compounded for languages like Arabic, where cursive is the only form of script and there exist "shortcuts" to further simplify the cursive script and make writing more fluid (e.g. removing the "dots"/accent marks that exist above/below certain letters).

This is where the data compression/dimension reduction and eigenface technique comes into play! In essence, if we have a large data set that consists of thousands or even millions of images of words (probably limited to one language), we want to find a way to recognize patterns in these images. From these patterns, we can then determine which ones have the most "importance" and attempt to express these images as a weighted combination of these most important patterns (and thus in a lower dimension).

Chapter 4

Principal Component Analysis

4.1 Extracting "high information" regions in the data space

In this chapter we will see how we can use the knowledge from Chapter 2 to find efficient representations for arbitrary data, and apply that to the challenge introduced in Chapter 3.

Data in its raw or most simple representation is likely to the potential to store many more values than we actually care about. For example, in the case of character recognition, lets say that the dataset that we have for each handwritten character consists a 28x28 pixel grayscale images.
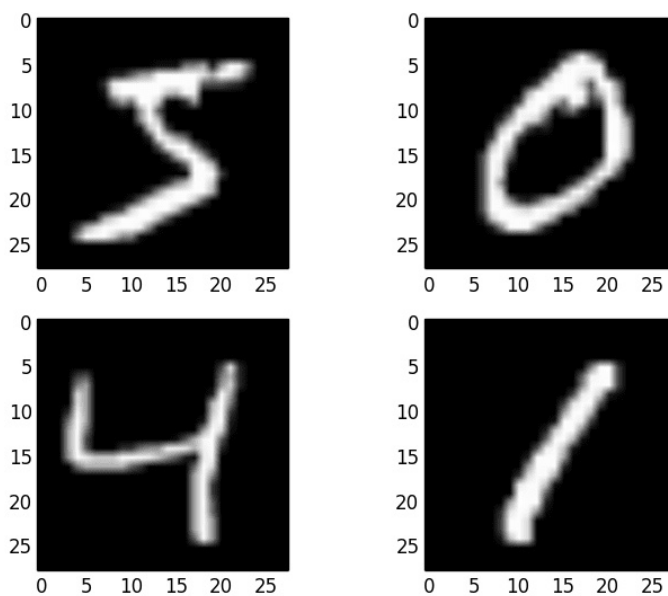

Figure 4.1: Samples of the numbers 5, 0, 4, and 1 from the MNIST handwriting dataset.

Theoretically, with this data space, we can encode signifcantly more than the letters and digits in the English alphabet. An all black or all white image exists in this data space but does not represent any of the glyphs that we care about.

The aim of data compression is to find a way of reducing the size of our data representation while preserving the ability to reconstruct the information in the original uncompressed data space at will.

In the case of the alphabet and humans, it is quite straight forward. A person can read a character, store it in the computer as a numerical value (0-9 for '0'-'9' and 10-35 for 'a'-'z'), and then when we want an image again, a human can write the digit out. But this techinque is very restricted. It can only be applied to the digit the characters the human is able to remember and that we had previously encoded as a numeric value. If the person sees a new digit they do not recognize, there is no way of storing this information using this compression format.

We need to establish a system that is able to solve this problem, that can be performed by computers without human intervention, and that can encode more than a predefined set of discrete values (ideally a continuous spectrum of variables).

A common techinque that has these capabilities is calles Principal Component Analysis (PCA). At a high level, it consists of finding patterns in the data (called components) and creating a system that allows us to express a datapoint in a our original data space in terms of how much of each component appears in that point.

Example 4.1.1 (Compressing image of plaid patterns)

In this example, we will look at a set of images that consists of plaid patterns and see how PCA can be useful in reducing the amount of data it takes to store these.

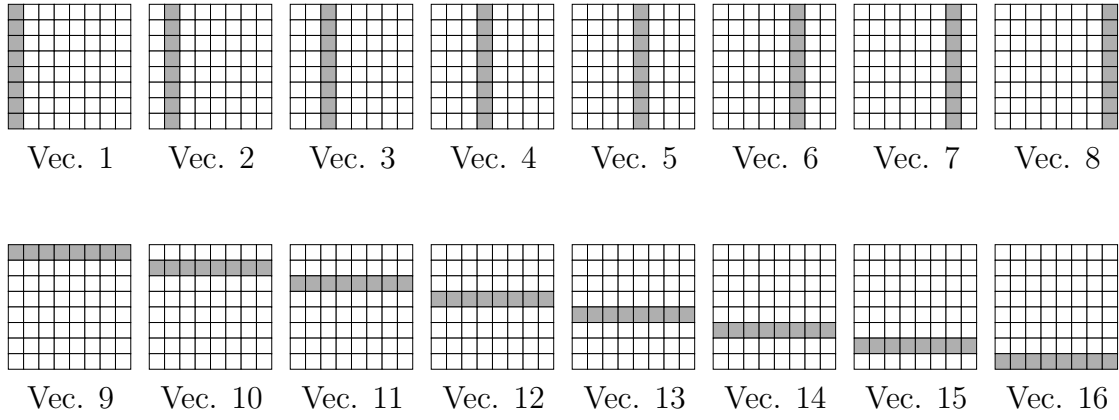Image 1   Image 2   Image 3   Image 4
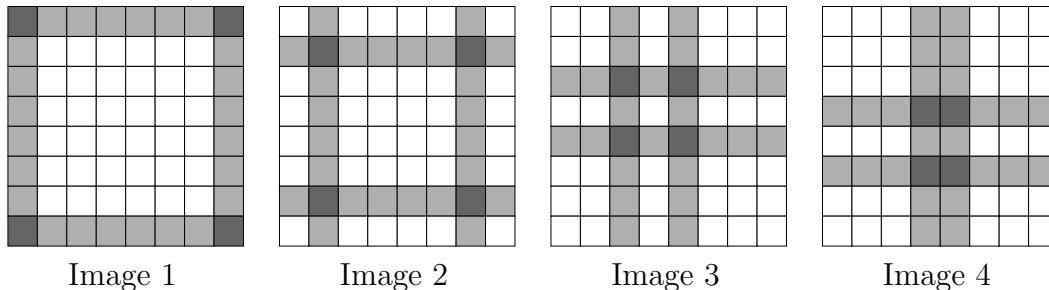
The standard representation of this data would be a vector of $8 \times 8 = 64$ dimensions, where each dimension can take one of three values: 0 (black), 1 (gray) or 2 (white). This data space would have $3^{64} \approx 3.43 \cdot 10^{30}$ elements but most of them would not represent plaid patterns, which is what we know our data will always represent.
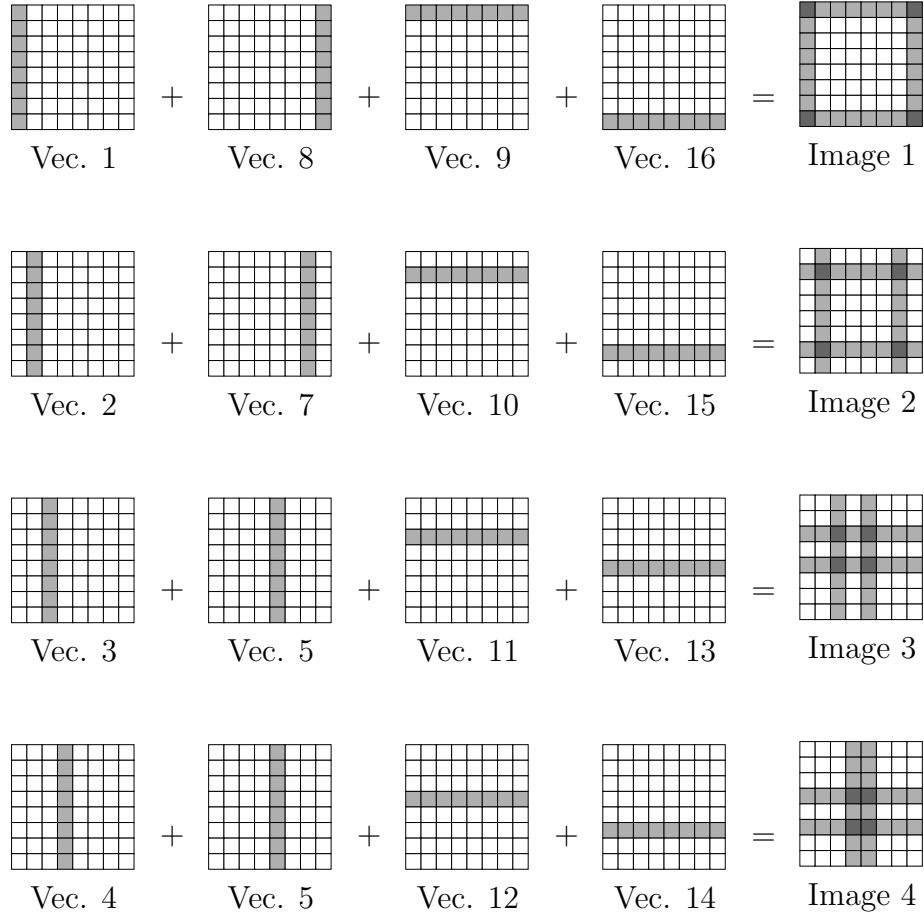
> **Note:**
> To store one of this datapoints would require storing the pixel value for each dimension, resulting in a storage size of 128 storage units.

What if we expressed these datapoints in a more favorable basis? Let the following be our new basis for expressing datapoints.

Vec. 1 Vec. 2 Vec. 3 Vec. 4 Vec. 5 Vec. 6 Vec. 7 Vec. 8

Vec. 9 Vec. 10 Vec. 11 Vec. 12 Vec. 13 Vec. 14 Vec. 15 Vec. 16

If we express our datapoints as a linear combination of these vectors, then we can recreate any of the images presented above, and any other plaid pattern that we want. Importantly, we represent any pattern by just 16 values, the coefficients for each basis vector.

With this basis, we can only represent $2^{16} = 65536$ datapoints, but those will be all the plaid pattern combinations that exists within the original dataspace. For example we can obtain the four images above by taking the following linear combinations of the basis vectors:

Vec. 1 + Vec. 8 + Vec. 9 + Vec. 16 = Image 1

Vec. 2 + Vec. 7 + Vec. 10 + Vec. 15 = Image 2

Vec. 3 + Vec. 5 + Vec. 11 + Vec. 13 = Image 3

Vec. 4 + Vec. 5 + Vec. 12 + Vec. 14 = Image 4

4.2   A linear algebra analogy

We can draw the analogy from the abstract high-level concepts that we just looked at the change of basis procedures that we studied in class. As mentioned, above, each datapoint is a vector, and an "expression system" is simply a set of basis vectors that can be used to write out datapoints in an expressed form.

By the theorem seen in class, if one has the same number of linearly independent vectors as dimensions in a space, that set of vectors forms a basis. But it we find a set of n vectors, to represent datapoints in an n dimensional space, we have achieved no compression, since we can use the standard basis and obtain the same storage efficiency.

Ideally the dataset that we want to represent only contains vectors in a lower-dimensional subspace of the data space. That way, we can express all the data points we are interested with a smaller number of vectors.

Unfortunately, real world data rarely fits a proper mathematical subspace, so this technique would not be reliable for all types of data. For example, assume all our data is on a plane within a 3-dimensional data space. With just one point that is not within the plane, we have increased the dimension of our subspace to 3, yielding not compression when expressing datapoints with new eigenvectors.

Our method must take into account how "important" these areas of space are to decide whether it is worth reducing our compression efficiency to be able to represent only a few infrequent datapoints.

4.3   Formalizing the calculation of "high information" eigenvectors

The motivation for using principal component analysis is to find what Turk and Pentland referred to as "face space," and which in our case we can conceptualize as the data space of letters. The face images, as previously explained, live as vectors in large dimensional spaces ($65,536$-dimensional space for a typical image). The goal at this stage is to reduce our large input vector space to a lower dimensional subspace that can be described be an eigenbasis.

> **Definition 4.3.1: Covariance**
>
> In order to construct... we need to define the **covariance**. Covariance measures the overall correlation between certain variables. The covariance σ of two variables, X and Y, is defined
> $$\sigma(X,Y) = \frac{1}{M}\sum_{i=1}^{M}(X_i - \bar{X})(Y_i - \bar{Y})$$

> **Definition 4.3.2: Covariance Matrix**
>
> **Covariance matrix**, then, is a matrix that measures the covariance of various variables. For variables $x_1, \ldots, x_p$, the covariance matrix is defined as follows
>
> $$C = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \ldots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & & \\ \vdots & & \ddots & \\ \sigma_{p1} & & & \sigma_{pp} \end{bmatrix}$$
>
> where σ_{ij} represents the covariance between the ith and j^{th} variable. In the case of eigenfaces, the p independent variables correspond to our N^2 bits, and thus p corresponds to the dimension of our images.

Having thus defined our covariance matrix, we must now discuss why we are taking the eigenvectors of this particular matrix to be our eigenfaces. In order for our eigenfaces to be as efficient as possible, we want each eigenface to encode as much variance as possible.

We take eigenvectors to simplify a relation defined by matrix multiplication to a relation defined by scalar multiplication, so that we may write any face as a linear combination of eigenvectors. Taking an eigenvector e, $Ce = \lambda e$. If λ is large, then the covariance between e and all many other faces is high. If λ is small, then the covariance is less. Therefore, we will see later that we want to prioritize large eigenvalues when constructing our eigenbasis.

Before we discuss the eigenvalues, however, we must derive some more definitions of C.

Firstly, given the large size of N^2 , it is convenient for us to find a more condensed description of the covariance matrix

> **Kai:** Is this N previously defined

> **Theorem 4.3.1**
>
> Let C be a $p \times p$ covariance matrix where each element $c_{ij} = \sigma_{ij}$. Let $\Phi_a = P_a - \bar{P}$, where
>
> $$P_a = \begin{bmatrix} x_{1a} \\ x_{2a} \\ \vdots \\ x_{pa} \end{bmatrix}$$
>
> with a from 1 to M.
> Then $C = \frac{1}{M} \sum_{a=1}^{M} \Phi_a \Phi_a^T$.

Proof: We can prove this theorem starting with

$$C = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & & \\ \vdots & & \ddots & \\ \sigma_{p1} & & & \sigma_{pp} \end{bmatrix}$$

From the definition of covariance, each entry $c_{ij} = \frac{1}{M} \sum_{i=1}^{M} (x_{ia} - \bar{x}_i)(x_{ia} - \bar{x}_i)$. Since

matrices add linearly, we can factor out the summation $\frac{1}{M}\sum_{i=1}^{M}$. Thus we can write

$$C = \frac{1}{M}\sum_{i=1}^{M} \begin{bmatrix} (x_{1a} - \bar{x}_1)(x_{1a} - \bar{x}_1) & \cdots & (x_{1a} - \bar{x}_1)(x_{pa} - \bar{x}_p) \\ & \vdots & \ddots \\ (x_{pa} - \bar{x}_p)(x_{1a} - \bar{x}_1) & & (x_{pa} - \bar{x}_p)(x_{pa} - \bar{x}_p) \end{bmatrix}$$

Now consider the term $\Phi_a \Phi_a^T = (P_i - \bar{P})(P_i - \bar{P})^T$, as defined in the theorem. Since the transpose operator is distributive,

$$(P_i - \bar{P})(P_i - \bar{P})^T = \left(\begin{bmatrix} x_{1a} \\ x_{2a} \\ \vdots \\ x_{pa} \end{bmatrix} - \bar{P} \right) \left(\begin{bmatrix} x_{1i} & x_{2i} & \cdots & x_{pi} \end{bmatrix} - \bar{P}^T \right)$$

$$= \left(\begin{bmatrix} x_{1a} \\ x_{2a} \\ \vdots \\ x_{pa} \end{bmatrix} - \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_p \end{bmatrix} \right) \left(\begin{bmatrix} x_{1a} & x_{2a} & \cdots & x_{pa} \end{bmatrix} - \begin{bmatrix} \bar{x}_1 & \bar{x}_2 & \cdots & \bar{x}_p \end{bmatrix} \right)$$

$$= \begin{bmatrix} x_{1a} - \bar{x}_1 \\ x_{2a} - \bar{x}_2 \\ \vdots \\ x_{pa} - \bar{x}_p \end{bmatrix} \begin{bmatrix} x_{1a} - \bar{x}_1 & x_{2a} - \bar{x}_2 & \cdots & x_{pa} - \bar{x}_p \end{bmatrix}$$

Therefore,

$$\Phi_a \Phi_a^T = \begin{bmatrix} (x_{1a} - \bar{x}_1)(x_{1a} - \bar{x}_1) & \cdots & (x_{1a} - \bar{x}_1)(x_{pa} - \bar{x}_p) \\ & \vdots & \ddots \\ (x_{pa} - \bar{x}_p)(x_{1a} - \bar{x}_1) & & (x_{pa} - \bar{x}_p)(x_{pa} - \bar{x}_p) \end{bmatrix}$$

$$C = \frac{1}{M}\sum_{a=1}^{M} \Phi_a \Phi_a^T$$

☺

By constructing

REFERENCE PRINCIPAL COMPONENT ANALYSIS

In the eigenface case, we wish to measure the difference between our face vectors Γ_i differ from the average face. We can define the average face $\bar{\Gamma}$ as follows,

$$\bar{\Gamma} = \frac{1}{M}\sum_{n=1}^{M} \Gamma_n$$

We can write the difference Φ_i between the average face and the ith face,

$$\Phi_i = \Gamma_i - \bar{\Gamma}$$

Hello

15

Iñaki: Clarify notation because $\bar{x}_p$ is not defined anywhere

Let C be an $N^2 \times N^2$ covariance matrix, defined as $C = \frac{1}{M}\sum_{n=1}^{M} \Phi_n \Phi_n^T$, where M is the dimension of the domain C. Let A be an $N^2 \times N^2$ matrix with columns $[\Phi_1\ \Phi_2\ \ldots \Phi_M]$. Then $C = AA^T$.

Proof: ☺

Going back to our matrix C,

$$C = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & & \\ \vdots & & \ddots & \\ \sigma_{p1} & & & \sigma_{pp} \end{bmatrix}$$

We can expand C according to the definition of σ,

$$C = \frac{1}{M}\sum_{a=1}^{M} \begin{bmatrix} \Phi_1 \Phi_1^T & \Phi_1 \Phi_2^T & \cdots & \Phi_1 \Phi_p^T \\ \Phi_2 \Phi_1^T & \Phi_2 \Phi_2^T & & \\ \vdots & & \ddots & \\ \Phi_p \Phi_1^T & & & \Phi_p \Phi_p^T \end{bmatrix}$$

Adam:
...

Now we can express our eigenvectors v_i and eigenvalues λ_i as

$$\lambda_i v_i = AA^T v_i$$

However, AA^T is an N^2 by N^2 matrix (where $N \times N$ is the resolution of the image), which for images of any discernable resolution will be much larger than desired in order to compute its eigenvectors.

Let

$$S = AA^T = \lambda_i v_i$$

Then,

$$S^T = \left(AA^T\right)^T = A^T A$$

We can show that S and S^T have the same eigenvalues.

Since the indetity matrix is symmetrical, since the transpose operation is distributive, and since $\det(A) = \det\left(A^T\right)$,

$$\det\left(S^T - \lambda_i I\right) = \det(S - \lambda_i I)^T = \det(S - \lambda_i I)$$

and therefore S and S^T have the same eigenvalues.

Applying this to our eigenfaces,

$$S^T = A^T A = \lambda_i u_i$$

16

The advantage of effectively reordering the A and A^T is that our new matrix S^T is an $M \times M$ matrix. M corresponds to the size of the training set, which in most cases is much smaller than the dimension of the resolution of the image. Therefore, it will be much easier to compute the eigenfaces. We can then write each face as a linear combination of the eigenfaces

MEGAREFERENCE

$$\Gamma_i = \sum_{n=1}^{M} c_{in} u_n$$

We can interpret the eigenvectors of the covariance matrix as vectors that encode the greatest variance in data, and as such are the most suited for reconstructing faces from (ghosts). The corresponding eigenvalues tell how much variance each eigenvector encodes. For this reason, the greater the eigenvalue the more adaptable and better suited the eigen vector. When choosing eigenvectors to form a basis for the eigenspace used for facial recognition, one therefore prioritizes them by eigenvalue from highest to lowest.

AWDAWD

Chapter 5

Uses Beyond Compression

We mainly discussed how PCA allows us to reduce the dimensions of our data space and store data more efficiently, but there are more uses for this beyond compression. Having a simpler representation for our data allows us to perform more complex operations on it.

5.1 Classification

We can use PCA to perform classification on our data. After we have chosen our eigenvectors based on our initial dataset, and have manually classified the original "training" data into categories, we can use Linear Classification [Mil19] to classify new data into the preexisting categories.

This process consists of projecting the new datapoint to our lower-dimension subspace and then making a category decision based on how much weight each basis vector has in the new datapoint.

5.2 Recognition

In cases were there are no discrete categories defined during training time, we can use dimensionality reduction to more easily (and faster) compare a new datapoint to a list of known datapoints that continually gets updates.

In the case of facial recognition, we can use PCA to reduce the dimensionality of our data and then use a distance metric to compare the new datapoint to the known datapoints. The datapoint with the smallest distance to the new datapoint is the most similar and is therefore the most likely to be the same person.

Bibliography

[Ale15]   Alexaqz. Data is expected to double every two years for the next decade, Aug 2015.

[Mar22]   Bernard Marr. How much data do we create every day? the mind-blowing stats everyone should read, Oct 2022.

[Mil19]   Imdadul Haque Milon. Linear classifiers: An introduction to classification, Aug 2019.

[Ric07]   John Rice. *Mathematical Statistics and Data Analysis*. Brooks/Cole Cengage Learning, 2007.

[Wik22]   Wikipedia. Statistics — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Statistics&oldid=1121639148, 2022. [Online; accessed 29-November-2022].

[Wol02]   Stephen Wolfram. *A New Kind of Science*. Wolfram Media, 2002.