Informal Notes for Mathematics for Machine Learning
Imperial College London

Lecture 13: Principal Component Analysis

Lecturer: Matthew Wicker

1 Learning Objectives

Throughout the course we have studied how different branches of mathematics enable us to develop a framework for understanding and analyzing machine learning models. In this lecture, using many of the skills we have developed throughout the course, we will analyze one of the most popular unsupervised learning algorithms aimed at dimensionality reduction: principal component analysis.

2 Unsupervised Learning

Recall from the beginning of our course that unsupervised learning, sometimes known as knowledge discovery, is characterized by the setting in which we have feature vectors but no labels. That is a dataset $\{\boldsymbol{x}^{(i)}\}_{i=1}^{n}$ where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ As we discussed in lecture 2, there are many sub-areas of unsupervised learning; however, in this lecture we will focus specifically on dimensionality reduction.

2.1 Dimensionality reduction

Visualizing data when $d > 3$ is a challenging task, but data visualization is an important tool in understanding the best ways to attack a machine learning problem. As a motivating example, when studying the MNIST handwritten digit recognition dataset, one can roughly identify the number of classes (distinct digits) in the dataset by looking at point cloud that results from projecting each 28 by 28 pixel image down into just $\mathbb{R}^2$. In addition to simple data visualization, more advanced models of dimensionality reduction have enabled much more sophisticated understanding of underlying data set struture (e.g., VAEs). We will discuss some further practical applications after developing one of the most popular algorithms for dimensionality reduction: principal component analysis (PCA).

3 Principal Component Analysis

Throughout this course, we have always taken the perspective that our observed dataset is a collection of random variables:

$$\mathcal{D} = \{X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)}, \ldots, X^{(n)} = x^{(n)}\}$$

and often, we make an assumption that these variables are independently and identically distributed. One perspective on visualizing this is to consider each sampled point as a point in space and subsequently to consider the entire dataset as a point cloud. Doing so for different datasets in $\mathbb{R}$ or $\mathbb{R}^2$ motivated our study of density estimation and we were quickly able to look at the point clouds that described our dataset and think about the model's we should fit e.g., uni-modal distributions like Gaussian distributions. When $d > 4$, understanding what model is best to use is difficult. However, we understand the principals we used before, so without making *a priori* assumptions about the dataset, let us start by considering the problem in terms of standard probability-theory and statistics quantities.

3.1 Probability perspective

Given an dataset i.i.d. from some distribution that we do not have access to, we can use the tools and techniques from our lecture on concentration to think about the statistics of this dataset. Our prior concern was primarily estimation of the mean with statistical guarantees. We showed that the sample mean, $\hat{\boldsymbol{x}} = \frac{1}{n}\sum i = 1^n \boldsymbol{x}^{(i)}$ is an unbiased estimator of the mean of the distribution from which our dataset was drawn. As a starting step that will make much of our analysis simpler, we will subtract off the mean from our dataset, that is, we assume $\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i)} - \mu$. This is equivalent to enforcing that the distribution we are looking at has mean zero. Visualizing this transformation reveals that all we are doing is shifting the data we observed to be centered around the origin, but preserves all of the inter-feature structure of the data. So if manipulating the mean of the data does not have a significant affect of the structure of the data, we ought to look elsewhere to understand the structure of our data. Assuming a high-dimensional feature space there is only one other place that we can easily look: the covariance matrix. From our data we have that:

$$\begin{aligned}\hat{\Sigma}_n = S &= \frac{1}{n}\sum_{i=1}^{n}(\hat{\boldsymbol{x}} - \boldsymbol{x}^{(i)})(\hat{\boldsymbol{x}} - \boldsymbol{x}^{(i)})^\top \\ &= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)\top}\end{aligned}$$

as zero mean i.e $\hat{x} = 0$

covariance:

$\text{var}(x) = \text{covar}(x, x)$

$= E[(x - E(x))(x - E(x))]$

↓ empirical - sum & divide

$\frac{1}{n}\sum(x - \bar{x})(x - \bar{x})$

where the second equation just leverages the fact that we have normalized our data to have mean zero. The above matrix which we will denote S contains a great deal of structural information about our data that we would like to understand. In order to extract such information from a matrix, we must turn to linear algebra.

can do $x - \hat{x}$ or $\hat{x} - x$

as long as consistent

3.2 Linear algebra framing

Given our dataset of i.i.d. distributed features, $\boldsymbol{x}$, we have seen above how to compute the sample covariance matrix S. However, we now turn to linear algebra to get a better understanding of exactly what S is telling us. A natural first thing to do is to think about general properties we can compute of a matrix. For example, what does the determinant of S tell us? In essence, it tells us how spread apart our data is. Similarly, the rank of the matrix tells us the effective dimension that our dataset lies in. If S has rank less than d, then we know somehow two columns must be linearly dependent on one another. Another way to analyze matrices, especially symmetric matrices (which S is), that we have seen in previous lectures is the eigendecomposition of the matrix, which tells us how the matrix operates as a linear transformation. We can write the eigendecomposition of S as:

$$S = P\Lambda P^\top$$

as $P^{-1} = P^T$ in this instance

(covar matrix is PSD & symmetric

where, as before, P contains the matrix containing eigenvectors of S as columns and Λ is a diagonal matrix containing the eigenvalues corresponding to each eigenvector. In our previous section after computing the mean we decided to normalize our data based on that computation. A natural question is can we do the same here? To see that we can, consider a new dataset which is a transformation of the original such that $\boldsymbol{y}^{(i)} = P^\top \boldsymbol{x}^{(i)}$. We now might want to know about the covariance structure of $\boldsymbol{y}$ in terms of the prior covariance structure S. We can plug in the definition of our sample covariance to get:

$S' = \Sigma\, y y^T$

mean of y still 0

$$\begin{aligned} S' &= \frac{1}{n}\sum_{i=1}^{n} P^\top \boldsymbol{x}^{(i)} (P^\top \boldsymbol{x}^{(i)})^\top \\ &= \frac{1}{n}\sum_{i=1}^{n} P^{T} \boldsymbol{x}^{(i)} \boldsymbol{x}^{(i)\top} P \\ &= P^\top S P \end{aligned}$$

$$= P \underbrace{\left[\frac{1}{n}\Sigma\, x^i x^{iT}\right]}_{=S} P$$

where the last step is simply us oberving that the central term of the sum of our second equation is identical to the definition of S from our previous section. Of course, we can See that by plugging in our eigendecomposition for S we get:

$$S' = P^\top (P\Lambda P^\top) P = \Lambda$$

as $P^T = P^{-1}$

where the last equality is due to the fact that P is by definition an orthogonal basis. So what do we know about out newly normalized data $\boldsymbol{y}$? We have that its covariance structure is a diagonal matrix meaning that each covariance is zero. If we were to draw this out, we would essentially have that the vectors $P^\top \boldsymbol{x}$ have simply been rotated in order to ensure that they are axis aligned with the standard basis vectors. Mathematically this means that $\text{Cov}(\boldsymbol{y}_i, \boldsymbol{y}_j) = 0$ if $i \neq j$ and that $\text{Cov}(\boldsymbol{y}_i, \boldsymbol{y}_i) = \text{Var}(\boldsymbol{y}_i, \boldsymbol{y}_i) = \lambda_i$.

It seems now that we have our data in a nice, standardized form, we should begin thinking about how we ought to visualize our data. We know that we want to project our data from $\mathbb{R}^d$ down to $\mathbb{R}^m$, and we know linear algebra operators that do this: matrices that are $Q \in \mathbb{R}^{m\times d}$. Of course, assuming we have a matrix $\boldsymbol{x} \in \mathbb{R}^{d\times 1}$ then the product Qx returns a column vector in $\mathbb{R}^m$. But how should we pick such a matrix? This is where an optimization approach helps.

3.3 Optimization formulation

In prior sections we discussed estimating properties of our dataset, and then we used linear algebra to manipulate and normalize our dataset such that it is nice to analyze. However choosing the appropriate projection matrix Q requires us to think carefully about what aspects of our data we want to preserve. Of course, in a very general sense, what we want to preserve is the *information* and relationships that exist in our data. One approach, the PCA approach, is to preserve as much of the covariance structure as possible. To think about this in simple terms, let us first consider $m = 1$ where we want to project our data onto the reals but keeping as much of the covariance structure as possible. Additionally, we want our projection to consist of an orthonormal basis. By the definition, the variance of our projection is given by:

$\because$ want q to be orthonormal basis

$$\max_{\boldsymbol{q}} \mathbb{V}[\boldsymbol{q}^\top \boldsymbol{x}], \qquad \text{s.t.,} \quad ||\boldsymbol{q}||_2^2 = 1 \tag{1}$$

$$\mathbb{V}[\boldsymbol{q}^\top \boldsymbol{x}] = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{q}^\top \boldsymbol{x}^{(i)})^2 \quad = (q^T x^i)(q^T x^i)^T \tag{2}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{q}^\top \boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)\top}\boldsymbol{q} \tag{3}$$

$$= \boldsymbol{q}^\top \left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)\top}\right)\boldsymbol{q} \tag{4}$$

$$= \boldsymbol{q}^\top (S)\boldsymbol{q} \tag{5}$$

$$= \boldsymbol{q}^\top (P^\top \Lambda P)\boldsymbol{q} \quad \because \beta = Pq \tag{6}$$

$$= \sum_{i=1}^{d}(\lambda_i \beta_i)^2 \tag{7}$$

Where our last step is due to the substitution $\beta = P\boldsymbol{q}$. So now, we return to thinking about the initial optimization goal which is to select a vector $\boldsymbol{q}$ which maximizes this final equation. A minor hiccup is that our final equation is in terms of β not $\boldsymbol{q}$ so, we need to transform our constraint on $\boldsymbol{q}$ into a constraint on β:

need $||q||_2^2 = 1 \therefore ||\beta||_2^2 = 1$

$$||\boldsymbol{q}||_2^2 := \boldsymbol{q}^\top \boldsymbol{q} = \boldsymbol{q}^\top \underbrace{P^\top P}_{}\overset{=I}{}\boldsymbol{q} = \beta^\top \beta = ||\beta||_2^2$$

So we can see that β itself must be a unit vector, and the question remains how do we select beta to maximize the variance of our projection? Well given that it is a linear combination of our eigenvalues, it is straightforward to see that no unit vector has larger variance then simply setting $\beta_j = 1$ where j is the entry with the largest eigenvalue (by convention, the 1,1 entry of the eigenvalue matrix). Then, the value of $\boldsymbol{q}$ which realizes this selection of β is of course the first eigenvector. ie $\beta = [1,0,0,\cdots,0]^T$

Λ ordered s.t. $\lambda_i > \lambda_j$, $\forall i<j$

We have now seen a very elegant solution to the $m = 1$ case, but can we extend this to $m = 2$? Well we know that we will want to pick two vectors $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ that will serve as the columns of our orthonormal projection and so we want $\boldsymbol{q}_1 \perp \boldsymbol{q}_2$ as well as $||\boldsymbol{q}_1|| = 1$ and $||\boldsymbol{q}_2|| = 1$. Well, one vector that satisfies this is the eigenvector with the second largest eigenvalue. And indeed, if we follow the same logic from above, we see that this will maximize the variance (left as an exercise).

4 Algorithm and Discussion

Algorithm 1 PCA
Input: $\boldsymbol{X}$ - Feature/Design matrix, K - Number of components

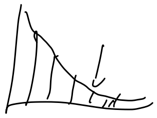
```
$\hat{S}_n \leftarrow \sum_{i=1}^{n} \boldsymbol{x}^{(i)} \boldsymbol{x}^{(i)\top}$    sample covar
Compute eigendecomposition $S = P \Lambda P^\top$
Ensure $\Lambda = \text{diag}(\lambda)$ with $\lambda_i \geq \lambda_j \forall i < j$
$B = P_{[:,:k]}$    first k cols of P (P is orthonormal basis of S)
return $\{B\boldsymbol{x}^{(i)}\}_{i=1}^{n}$
```

Above is the fully assembled PCA algorithm which takes in our points and a number of dimensions on which to project them and returns points in that space such that we have preserved the maximum amount of covariance information. A few key questions to consider (that will be discussed in lecture):

- How do we select k?
- Beyond visualization how can we use this in ML?
- What is the complexity of computing this solution?

Picking k:

- Can choose 2 or 3 for visualisation purposes
- look at variance for diff k vals, choose k at inflection point



PCA is a linear technique

TSNE is a non-linear one

A Lecture 13: PCA

Question 1 (Connections to linear auto-encoders)**.** Consider a linear auto-encoder defined as follows for an input $\boldsymbol{x} \in \mathbb{R}^{D\times 1}$:

Encoder: $\boldsymbol{z} = enc(\boldsymbol{x}) = \mathbf{B}\boldsymbol{x},\ \mathbf{B} \in \mathbb{R}^{M\times D}, M < D,$ Decoder: $\hat{\boldsymbol{x}} = dec(\boldsymbol{z}) = \mathbf{A}\boldsymbol{z},\ \mathbf{A} \in \mathbb{R}^{D\times M}.$

Let us assume $rank(\mathbf{A}) = rank(\mathbf{B}) = M$. Given a dataset $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ such that mean$(\boldsymbol{x}_n) = \mathbf{0}$ and the covariance matrix computed on $\mathcal{D}$ is invertible, we wish to train the model by minimising the ℓ_2 reconstruction error:

$$\min_{\mathbf{A},\mathbf{B}} L(\mathbf{A},\mathbf{B}), \quad L(\mathbf{A},\mathbf{B}) := \frac{1}{N}\sum_{n=1}^{N} ||\boldsymbol{x}_n - \mathbf{A}\mathbf{B}\boldsymbol{x}_n||_2^2.$$

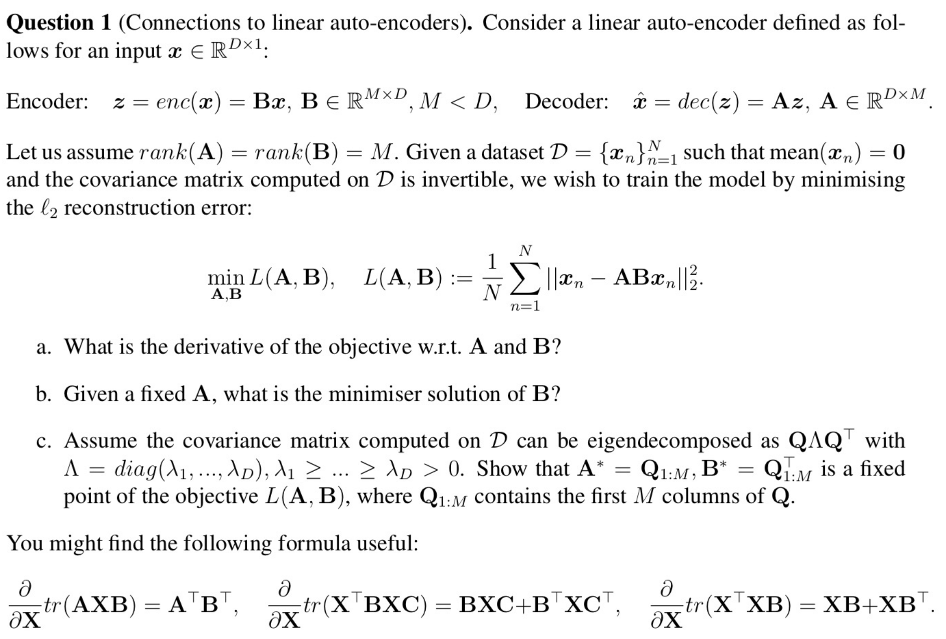
a. What is the derivative of the objective w.r.t. $\mathbf{A}$ and $\mathbf{B}$?

b. Given a fixed $\mathbf{A}$, what is the minimiser solution of $\mathbf{B}$?

c. Assume the covariance matrix computed on $\mathcal{D}$ can be eigendecomposed as $\mathbf{Q}\Lambda\mathbf{Q}^\top$ with $\Lambda = diag(\lambda_1, ..., \lambda_D), \lambda_1 \geq ... \geq \lambda_D > 0$. Show that $\mathbf{A}^* = \mathbf{Q}_{1:M}, \mathbf{B}^* = \mathbf{Q}_{1:M}^\top$ is a fixed point of the objective $L(\mathbf{A},\mathbf{B})$, where $\mathbf{Q}_{1:M}$ contains the first M columns of $\mathbf{Q}$.

You might find the following formula useful:

$$\frac{\partial}{\partial \mathbf{X}} tr(\mathbf{A}\mathbf{X}\mathbf{B}) = \mathbf{A}^\top\mathbf{B}^\top, \quad \frac{\partial}{\partial \mathbf{X}} tr(\mathbf{X}^\top\mathbf{B}\mathbf{X}\mathbf{C}) = \mathbf{B}\mathbf{X}\mathbf{C}+\mathbf{B}^\top\mathbf{X}\mathbf{C}^\top, \quad \frac{\partial}{\partial \mathbf{X}} tr(\mathbf{X}^\top\mathbf{X}\mathbf{B}) = \mathbf{X}\mathbf{B}+\mathbf{X}\mathbf{B}^\top.$$



Question 1 (Connections to linear auto-encoders)**.** Consider a linear auto-encoder defined as follows for an input $\boldsymbol{x} \in \mathbb{R}^{D\times 1}$:

$$\text{Encoder:} \quad \boldsymbol{z} = enc(\boldsymbol{x}) = \mathbf{B}\boldsymbol{x},\ \mathbf{B} \in \mathbb{R}^{M\times D}, M < D, \quad \text{Decoder:} \quad \hat{\boldsymbol{x}} = dec(\boldsymbol{z}) = \mathbf{A}\boldsymbol{z},\ \mathbf{A} \in \mathbb{R}^{D\times M}.$$

Let us assume $rank(\mathbf{A}) = rank(\mathbf{B}) = M$. Given a dataset $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ such that mean$(\boldsymbol{x}_n) = \mathbf{0}$ and the covariance matrix computed on $\mathcal{D}$ is invertible, we wish to train the model by minimising the ℓ_2 reconstruction error:

$$\min_{\mathbf{A},\mathbf{B}} L(\mathbf{A},\mathbf{B}), \quad L(\mathbf{A},\mathbf{B}) := \frac{1}{N}\sum_{n=1}^N ||\boldsymbol{x}_n - \mathbf{A}\mathbf{B}\boldsymbol{x}_n||_2^2.$$

a. What is the derivative of the objective w.r.t. $\mathbf{A}$ and $\mathbf{B}$?

b. Given a fixed $\mathbf{A}$, what is the minimiser solution of $\mathbf{B}$?

c. Assume the covariance matrix computed on $\mathcal{D}$ can be eigendecomposed as $\mathbf{Q}\Lambda\mathbf{Q}^\top$ with $\Lambda = diag(\lambda_1, ..., \lambda_D), \lambda_1 \geq ... \geq \lambda_D > 0$. Show that $\mathbf{A}^* = \mathbf{Q}_{1:M}, \mathbf{B}^* = \mathbf{Q}_{1:M}^\top$ is a fixed point of the objective $L(\mathbf{A},\mathbf{B})$, where $\mathbf{Q}_{1:M}$ contains the first M columns of $\mathbf{Q}$.

You might find the following formula useful:

$$\frac{\partial}{\partial \mathbf{X}} tr(\mathbf{AXB}) = \mathbf{A}^\top\mathbf{B}^\top, \quad \frac{\partial}{\partial \mathbf{X}} tr(\mathbf{X}^\top\mathbf{BXC}) = \mathbf{BXC}+\mathbf{B}^\top\mathbf{XC}^\top, \quad \frac{\partial}{\partial \mathbf{X}} tr(\mathbf{X}^\top\mathbf{XB}) = \mathbf{XB}+\mathbf{XB}^\top.$$

$AB: D\times m \cdot m\times D : d\times d$

$A^TA: m\times m$

$B^TA^TAB: d\times m\ m\times m\ m\times d: d\times d$

a. $L : \frac{1}{N}\sum_{n=1}^{N} ||x_n - ABx_n||_2^2$

$: \frac{1}{N}\sum (x_n - ABx_n)^T(x_n - ABx_n) : \frac{1}{N}\sum(x_n^Tx_n - x_n^TABx_n - x_n^TB^TA^Tx_n + x_n^TB^TA^TABx_n)$

$= \frac{1}{N}\sum(x_n^Tx_n - 2x_n^TABx_n + x_n^TB^TA^TABx_n)$ last 2 terms are scalars - tr(scalar): scalar

$= \frac{1}{N}\sum(x_n^Tx_n - 2\,tr(x_n^TABx_n) + tr(x_n^TB^TA^TABx_n))$

$= \frac{1}{N}\sum x_n^Tx_n - \frac{1}{N}\sum 2tr(AB \cdot x_nx_n^T) + \frac{1}{N}\sum tr(B^TA^TABx_nx_n^T)$

$=\quad '' \quad - 2\,tr(AB\frac{1}{N}\sum x_nx_n^T) + tr(B^TA^TAB\frac{1}{N}\sum x_nx_n^T)$

$= \frac{1}{N}\sum x_n^Tx_n - 2\underbrace{tr(ABS)}_{= tr(SAB)} + \underbrace{tr(B^TA^TABS)}_{tr(A^TABSB^T)}$

$\frac{\partial}{\partial A}: -2S^TB^T + ABSB^T + A(BSB)^T$

$= -2SB^T + 2ABSB^T$

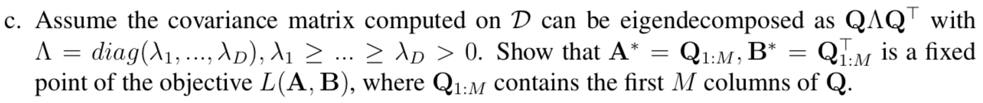
$(BSB^T)^T: BS^TB^T = BSB^T$

$\frac{\partial}{\partial B}: -2A^TS^T + A^TABS + A^TABS^T$

$= -2A^TS + 2A^TABS$

b. $A^TAB\cancel{S} : A^T\cancel{S}$

$B = (A^TA)^{-1}A^T$

C. 

c. Assume the covariance matrix computed on $\mathcal{D}$ can be eigendecomposed as $\mathbf{Q}\Lambda\mathbf{Q}^\top$ with $\Lambda = diag(\lambda_1, ..., \lambda_D), \lambda_1 \geq ... \geq \lambda_D > 0$. Show that $\mathbf{A}^* = \mathbf{Q}_{1:M}, \mathbf{B}^* = \mathbf{Q}_{1:M}^\top$ is a fixed point of the objective $L(\mathbf{A},\mathbf{B})$, where $\mathbf{Q}_{1:M}$ contains the first M columns of $\mathbf{Q}$.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \to \begin{pmatrix} a & b & 0 \\ d & e & e \\ g & h & i \end{pmatrix}$$

$ABSB^T : SB^T$

$A: SB^T(BSB)^{-1} \quad \to \quad Q\Lambda Q^TQ_{1:m}(Q_{1:m}^TQ\Lambda Q^TQ_{1:m})^{-1}$

$= Q\Lambda\begin{bmatrix} I_m & 0 \\ 0 & 0 \end{bmatrix}\left(\begin{bmatrix} I_m & 0 \\ 0 & 0 \end{bmatrix}\Lambda\begin{bmatrix} I_m & 0 \\ 0 & 0 \end{bmatrix}\right)^{-1}$

$= Q\Lambda\begin{bmatrix} \Lambda_m & 0 \\ 0 & 0 \end{bmatrix}^{-1} = Q\begin{bmatrix} I_m & 0 \\ 0 & 0 \end{bmatrix} = Q_{1:m} : A^*$

$B = (A^T A)^{-1} A^T \rightarrow (Q_{1:n}^T Q_{1:n})^{-1} Q_{1:n}^T = Q_{1:n}^T = B^*$