36-462: Data Mining **Spring 2019**

Lecture 13: February 28

Lecturer: Siva Balakrishnan

Some announcements:

1. Riccardo's office hours – Thursday 3-4pm, PH A20. Today will be done by Pratik.
2. Gradescope and selecting pages – You **must** select pages for each question when you upload your assignment. Make sure that you are able to do this (if not, let us know early).

   If you do not select pages you will be penalized for each question where you fail to select pages properly.

Our first goal today will be to quickly review some ideas in linear algebra. Before we get there lets try to understand some of the motivation. Linear algebra roughly is the study of matrices (really linear operators) so what does this have to do with statistical learning or data analysis?

The answer is most prominent when we leave the realm of supervised learning and enter the realm of *unsupervised learning*. In supervised learning (mainly classification and regression) we tried to predict some response (class label) y from input features x. In unsupervised learning, we are just given the data (with no response that we are interested in).

Our input in unsupervised learning is typically just $X \in \mathbb{R}^{n \times d}$, i.e. the d-dimensional feature vectors for n individuals.

Of course, now there is not really a clearly pre-defined goal (predict something). Instead we might care about:

1. **Visualization/dimensionality reduction:** Visualizing a high-dimensional dataset can be quite challenging (scatter-plots are most likely not so useful anymore). Roughly, we want to extract an informative set of 2 or 3 directions in which we can project our data and visualize this lower-dimensional projection. This is sometimes called *dimensionality reduction*.
2. **Clustering:** If we are just given a set of observations, a canonical task is to group them into *clusters*, i.e. groups of similar observations.
3. **Density estimation:** Another classical unsupervised task is smoothing or density estimation (some versions of this are sometimes called *graphical models*) where given samples we would like to understand or visualize the underlying density in some way.

We will mainly focus on the first two tasks. One thing to generally note is that unlike in supervised learning – where our goal was relatively clear and in most cases we just wrote down a loss function that made sense and then tried to make the loss small – in unsupervised learning there are different goals and the methods are much more ad hoc.

Before we get there however we'll need to step back. We are given as input just a matrix X and we'd like to extract structure from it so we should really understand matrices which brings us back to linear algebra. Our review will focus mostly on eigenvectors and eigenvalues (and the information they give us). The review will be brief (really one can spend an entire course on linear algebra and some of you have) – you can read for instance Gilbert Strang's book to get more detail on things we cover.

13.1 A Bit More Motivation

Representing a matrix by its eigenvectors and eigenvalues is sometimes called computing its *spectral decomposition*. Many popular unsupervised learning techniques are just *spectral decompositions* of different matrices. These terms don't need to make sense for now (they will over the next few lectures) but just notice the theme. Roughly:

1. **Principal Components Analysis (PCA):** Spectral decomposition of the covariance matrix.

2. **Multi-dimensional Scaling:** Spectral decomposition of the matrix of distances between data points.

3. **Kernel PCA:** Spectral decomposition of the matrix of similarities (measured with a kernel) between data points.

4. **Spectral Clustering:** Spectral decomposition of the *graph Laplacian* between the data points.

5. **Canonical Correlations Analysis (CCA):** Spectral decomposition of the cross-covariance matrix between two different sets of features.

The list goes on and really suggests that we should understand spectral decompositions better.

13.2 Vectors and Orthonormal Bases

Before we get to matrices, just quickly vectors in $\mathbb{R}^d$ are just a collection of d (real) numbers:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{bmatrix}.$$

We will typically use the convention of representing vectors as column vectors (as far as possible). Given a vector v one basic thing we can compute is its length:

$$\|v\|_2 := \sqrt{\sum_{i=1}^{d} v_i^2} = \sqrt{v^T v}.$$

Another is the projection of one vector onto another. If we are given a vector b and want to project it onto a (i.e. find the closest vector to b that lines up with a) then we can use the formula:

$$\text{proj}_a(b) = \frac{a^T b}{\|a\|_2} \frac{a}{\|a\|_2}.$$

(Draw a picture here)

This gives us some indication why we like projecting onto unit vectors. In particular, if a had length 1, then:

$$\text{proj}_a(b) = (a^T b)a.$$

Another obvious but nice fact is that we can write a vector by decomposing it in terms of its value along each axis, i.e.:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{bmatrix} = \sum_{i=1}^{d} v_i e_i,$$

where e_i is called the i-th canonical basis vector – just a complicated way of saying $e_i = (0, 0, \ldots, 1, 0, \ldots, 0)^T$ is a d-dimensional column vector with a 1 in its i-th position (and 0 everywhere else).

We can collect these basis vectors $\{e_1, \ldots, e_d\}$ and put them into a matrix (say each column of our matrix is one of these vectors):

$$M = \begin{bmatrix} \uparrow & \uparrow & \cdots & \uparrow \\ e_1 & e_2 & \cdots & e_d \\ \downarrow & \downarrow & \cdots & \downarrow \end{bmatrix}.$$

This matrix is more familiarly called the identity matrix.

This collection of vectors satisfies another important property: it is a collection of d mutually *orthonormal* vectors, i.e.

$$e_i^T e_j = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{otherwise.} \end{cases}$$

In words we have d vectors that are all perpendicular to each other and all have length 1. Such a collection of vectors is called an *orthonormal basis*. You can imagine rotating the canonical basis vectors and obtaining other collections of vectors that are all perpendicular to each other and all have length 1.

13.2.1 Some Important Properties of Orthonormal Bases

Writing a collection of orthonormal vectors:

$$Q = \begin{bmatrix} \uparrow & \uparrow & \cdots & \uparrow \\ q_1 & q_2 & \cdots & q_d \\ \downarrow & \downarrow & \cdots & \downarrow \end{bmatrix},$$

we can see immediately that:

$$Q^T Q = I.$$

Alternatively, this is telling us that

$$Q^{-1} = Q^T.$$

So orthonormal matrices are very easy to invert (we just take their transpose). This incidentally also tells us that:

$$QQ^T = I,$$

which is actually a bit harder to prove directly (but follows just because $Q^T = Q^{-1}$).

The other nice fact about orthonormal bases is that we can write any vector v in the basis in a simple way:

$$v = \sum_{i=1}^{d} (v^T q_i) q_i.$$

This is just saying that we can project v on to each of these orthonormal directions and put them together to get back v (this is exactly what we did before when we decomposed v in the canonical basis). A slightly more mathematical proof is:

$$\sum_{i=1}^{d} (v^T q_i) q_i = QQ^T v = v.$$

Another way of saying this is that for some (easy to determine) coefficients α_i we can write:

$$v = \sum_{i=1}^{d} \alpha_i q_i,$$

where the $\alpha_i = (v^T q_i)$. Another important property is that we can measure the length of a vector in any orthonormal basis, i.e.:

$$\|v\|_2 = \sqrt{\sum_{i=1}^{d} v_i^2} = \sqrt{\sum_{i=1}^{d} (v^T q_i)^2} = \sqrt{\sum_{i=1}^{d} \alpha_i^2}.$$

Again a quick proof:

$$v^T v = v^T Q Q^T v = \sum_{i=1}^{d} \alpha_i^2.$$

13.3 Eigenvectors and Eigenvalues

Throughout this section we will assume that we are starting with a square, symmetric, real-valued matrix $M \in \mathbb{R}^{d \times d}$. Eigenvectors can be defined more generally but they are not as useful/pretty in those cases.

For a symmetric, real-valued matrix M, it turns out that we can always *diagonalize* it, i.e. we can re-write it as a product of three matrices which are very special. We can write every such M as:

$$M = UDU^T,$$

where D is a diagonal matrix:

$$D = \begin{bmatrix} d_{11} & 0 & 0 & \cdots & 0 \\ 0 & d_{22} & 0 & \cdots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & d_{dd} \end{bmatrix},$$

and U is an orthonormal basis, i.e. $U \in \mathbb{R}^{d \times d}$ and the columns of U are orthonormal (they have length 1, and are perpendicular to each other).

The columns of U are called the *eigenvectors* of M and each eigenvector has a corresponding diagonal entry in D which are the *eigenvalues*.

This is a very, very, very special decomposition of the matrix. It is difficult to emphasize how useful, and magical this decomposition is.

Let us note some facts:

1. **The eigenvectors form an orthonormal basis.** They are a very special basis that capture nicely some properties of M.

2. You should always think about eigenvectors and their corresponding eigenvalues as a pair i.e. (u_i, d_{ii}). That said you can convince yourself that flipping the sign of every entry in u_i does not change anything (the product remains the same and the matrix U is still an orthonormal basis). So we say that eigenvectors are only determined up to sign (i.e. u_i and $-u_i$ are both eigenvectors with the same eigenvalue).

3. You can expand out the eigendecomposition (just multiply the three matrices back) as:

$$M = \sum_{i=1}^{d} d_{ii} u_i u_i^T.$$

4. Perhaps, a more classical way of introducing eigenvectors is by noticing that:

$$M \times U = U \times D,$$

   (just by the fact that U is orthonormal) or in a more familiar form:

$$Mu_i = d_{ii} u_i \quad \text{for each } i \in \{1, \ldots, d\}.$$

   This is just telling us that eigenvectors of M are special. When we map them through the matrix M (i.e. multiply them by M) they don't rotate (they are just scaled by the eigenvalue).

   Roughly, matrices take vectors and change them to other vectors – matrices take *eigenvectors* and only change their length.

5. Another fact (that is obvious in some sense but still worth noting) is that the eigenvectors and eigenvalues completely determine the matrix, i.e. they are just a different more convenient way of representing the information in a matrix.

6. Finally, it won't be very important for us but the eigendecomposition is not always unique. For instance, if we take the identity matrix then it is already diagonal, but we can also diagonalize it with *any* orthonormal matrix, i.e.:

$$I = Q \times I \times Q^T.$$

   Eigendecompositions are unique when the eigenvalues are all different (and not otherwise).

7. The rank of a matrix is just the number of non-zero diagonal entries of the matrix D.

All of this is fine but does not quite explain why these decompositions are interesting in data analysis (and we really will dig deeper into this in future lectures).

Some rough intuition is suppose we arranged things so that $|d_{11}| \geq |d_{22}| \geq \ldots \geq |d_{dd}|$. It will be convenient to always do this re-arrangement. Then the directions corresponding to the top eigenvalues are in essence most important to the matrix. These directions are where the matrix is really having an effect. We will return to this intuition and try to formally say what we mean by the top few eigenvectors (and corresponding eigenvalues) are "most important" for a matrix.

Lets keep at it.

13.3.1 Eigendecomposition for Matrix Operations

Suppose we wanted to compute powers of the matrix M.

For instance, we want to know what M^{1000} is. The naïve way would be to do this via matrix multiplication. The smarter way would be to use the eigendecomposition. Let us start with M^2:

$$M^2 = (UDU^T)(UDU^T) = UD^2U^T,$$

because $U^TU = I$. Similarly, you can easily check that for any integer $k \geq 0$

$$M^k = UD^kU^T.$$

The same thing works for the matrix inverse. Suppose that the diagonal entries of D are all non-zero (otherwise the matrix is not invertible) then:

$$M^{-1} = UD^{-1}U^T,$$

since we can check that:

$$M^{-1}M = UD^{-1}U^TUDU^T = UD^{-1}DU^T = UU^T = I.$$

Of course, we still need to invert D, but this is very easy because D is diagonal. So we can check:

$$D^{-1} = \begin{bmatrix} \frac{1}{d_{11}} & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{d_{22}} & 0 & \cdots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & \frac{1}{d_{dd}} \end{bmatrix}.$$

The same this works if we want to compute the exponential of a matrix.

$$\exp(M) = U\exp(D)U^T,$$

and we get $\exp(D)$ just by exponentiating the diagonal entries of D.

Another matrix operation is to multiply a matrix by a vector v. Suppose we for some reason knew how to represent v in the eigenbasis of M, i.e. we know:

$$v = \sum_{i=1}^{d} \alpha_i u_i,$$

where α_i are some coefficients (really just $v^T u_i$) and u_i are the eigenvectors of M. Then we can see that:

$$Mv = \sum_{i=1}^{d} \alpha_i d_{ii} u_i.$$

So the matrix M when acting on the vector v produces a new vector by scaling all the coefficients α_i.

13.3.2 Positive Semi-Definite Matrices

A subset of symmetric matrices, are the symmetric matrices which have all positive (≥ 0) eigenvalues. These matrices are called *positive semi-definite (PSD) matrices*.

In some sense, these matrices take their eigenvectors and scale them but do not flip them. More interestingly, we will often in the future look at *quadratic forms*, which are just quantities of the form $v^T M v$ for vectors v. For positive semi-definite matrices:

$$v^T M v > 0,$$

so the quadratic form is always positive. To see this suppose we write v in the basis of M, i.e.:

$$v = \sum_{i=1}^{d} \alpha_i u_i.$$

Then we have that,

$$\begin{aligned} v^T M v &= \left(\sum_{i=1}^{d} \alpha_i u_i \right)^T \left(\sum_{i=1}^{d} d_{ii} \alpha_i u_i \right) \\ &= \sum_{i=1}^{d} \alpha_i^2 d_{ii} \geq 0, \end{aligned}$$

since $d_{ii} \geq 0$. Where again we used the fact that the eigenvectors are orthonormal. Roughly, PSD matrices are like "squares of matrices" and behave like squares of numbers. As we will see later – the covariance matrix of a dataset is an important example of a PSD matrix. Just like the variance is a positive number always the covariance matrix will be a PSD matrix always.

13.4 Singular Value Decomposition

Given our emphasis on symmetric matrices so far a basic question to ask is– what do we do with matrices that are not symmetric (or not even square for that matter)? The story is a bit messier now – but roughly a useful decomposition that plays a similar role to the eigendecomposition is called the *singular value decomposition*. The SVD says that any (real) matrix $M \in \mathbb{R}^{n \times d}$ with rank r (no longer has to be symmetric or square) can be written as the product of three nice matrices:

$$M = U \times \Sigma \times V^T,$$

where $U \in \mathbb{R}^{n \times r}$, $\Sigma \in \mathbb{R}^{r \times r}$ and $V \in \mathbb{R}^{d \times r}$. Additionally, U and V are matrices with columns that are orthonormal (they don't necessarily form a full basis – since r can be less than n and d the matrices are not square), and Σ is diagonal (and additionally all its entries are positive).

1. The matrices U and V both satisfy: $U^TU = V^TV = I$ (here I is an $r \times r$ identity matrix).

2. The columns of U are called the *left singular vectors* the columns of V are the *right singular vectors* and the diagonal entries of Σ are called the *singular values*.

3. As we did before we can multiply things out to see that:

$$M = \sum_{i=1}^{d} \sigma_{ii} u_i v_i^T.$$

4. One basic question is how do we compute the SVD (we did not quite answer this for the eigendecomposition either) and how does the SVD related to eigenvector decompositions?

   It turns out that the SVD just comes from two eigen-decompositions, i.e. we can see that:

$$M^TM = V \times \Sigma \times U^T \times U\Sigma V^T = V\Sigma^2 V^T,$$

   and similarly,

$$MM^T = U \times \Sigma \times V^T \times V \times \Sigma \times U^T = U\Sigma^2 U^T.$$

   This tells us a bit more about what the components of the SVD are actually. The U matrix is just the eigenvectors of MM^T and the V matrix is just the eigenvectors of M^TM.

   One nice fact is that both the matrices MM^T and M^TM are PSD (why? Remember our intuition, they look like squares don't they?) – so the entries of Σ^2 are all positive (as we would hope they are since we need to take their square-root for the SVD).

5. Again just rough intuition and we'll return to this next lecture – you should think about a rectangular matrix as have a row space and a column space (i.e. vectors you can obtain by combining rows and by combining columns respectively). Symmetric matrices are nice because they have the same row and column space but for general, rectangular matrices these are different. The matrix U is a basis for the column-space of the matrix M and similarly V is a basis for the row-space.

13.5 Some Special Matrices

We have already studied one special type of matrix (orthonormal matrices). In this section, we will dig a bit deeper into a few others that will be useful for us. We will discuss covariance matrices and projection matrices (which you have seen before).

13.5.1 The Covariance Matrix

In data analysis we are typically given a data matrix $X \in \mathbb{R}^{n \times p}$. We will assume that the features are standardized (so we subtract the mean of each column).

One thing that we can compute is the covariance matrix:

$$\widehat{\Sigma} = \frac{1}{n} X^T X.$$

The covariance matrix can also be written as:

$$\widehat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T,$$

where $x_i \in \mathbb{R}^d$ is the i-th data sample (the i-th row of X represented as a column vector).

The entries of the covariance matrix tell us the variance and co-variance between the different features (see next section).

Furthermore, we note some properties of the covariance matrix:

1. The covariance matrix is symmetric (and so has an eigendecomposition). Again, to remind you PCA (which we'll talk about soon is just computing this eigendecomposition).

2. The covariance matrix is PSD (how do we know? It looks like a "square" of the data matrix).

3. Finally, let us note a simple fact. Suppose we pick a unit vector (a vector with length 1) v and compute the quadratic form $v^T\widehat{\Sigma}v$ we see that:

$$\begin{aligned} v^T\widehat{\Sigma}v &= v^T\left[\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T\right]v \\ &= \frac{1}{n}\sum_{i=1}^{n}(v^T x_i)^2. \end{aligned}$$

The collection $\{v^T x_1, v^T x_2, \ldots, v^T x_n\}$ is just the projection of our data along the direction v. This collection has mean 0 (because we centered our data to begin with). So the quantity $v^T\widehat{\Sigma}v$ is just the variance of the data along the direction v. We will use this fact in the next lecture.

13.5.1.1 Some simple graphical modeling ideas

Graphical modeling is roughly trying to draw meaningful pictures that represent the correlations (really dependencies) between variables.

(draw a picture here)

One simple "graphical model" (we may return to more interesting ones if we have time) is to just draw all the variables and connect them if they have a strong (or non-zero) correlation. This plot will give us some basic insights into how the different variables/features are related to each other.

13.5.2 Projection Matrices and Least Squares

We have already seen that if we want to project a vector b onto a vector a then we use:

$$\text{proj}_a(b) = \frac{a^T b}{\|a\|_2}\frac{a}{\|a\|_2}.$$

We can also represent this as a matrix:

$$\text{proj}_a(b) = \frac{aa^T}{\|a\|_2^2}b = P_a b.$$

This matrix has some special properties: it is symmetric, i.e. $P_a^T = P_a$ and we can also see that $P_a^2 = P_a$. Matrices that satisfy these conditions are called *projection matrices*.

You have of course seen linear regression and least squares many times before (and probably also in this form). We think of least squares, given some training data, as trying to find us

solutions $\widehat{\beta}$ such that:

$$X\widehat{\beta} \approx y.$$

If there was an exact solution then we could just find it by solving the linear system. Most often, there is not an exact solution and instead we find the closest solution in a least-squares sense.

One thing that is clear is that $X\widehat{\beta}$ is a vector that lives in the column space of X (i.e. it is a linear combination of columns of X) so we could alternatively say we want to find the closest vector $\widehat{y}$ to y where $\widehat{y}$ lives in the column space of X. We will then find $\widehat{\beta}$ such that:

$$X\widehat{\beta} = \widehat{y}.$$

In linear algebra language, we want to project y onto the column space of X. One way to accomplish this is to note that if $\widehat{y}$ is the projection of y on to the span of the columns of X then the residual $y - \widehat{y}$ must be orthogonal to columns of X (draw a picture), i.e.

$$X^T(y - \widehat{y}) = 0,$$

and so we see that:

$$X^T y = X^T X\widehat{\beta},$$

which gives us our familiar least-squares solution and and further that

$$\widehat{y} = X(X^T X)^{-1}X^T y = Py,$$

where the matrix P is a projection matrix that projects y on to the span of the columns of X. You can check that it satisfies our two conditions $P^T = P$ and $P^2 = P$. Notice that its form is very similar to the matrix that projected onto a single vector a.

As a thing to ponder, what does the eigendecomposition of a projection matrix look like? (It is clearly square and symmetric) In particular, what can we say about the eigenvalues of P?