MATH 123 Lecture Notes

Lecture 1: Linear Algebra Review, Bases/Encodings

To begin with Lecture 1, we'll do some linear algebra review:

Encoding and Decoding

If I take a vector

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

this is an object that is represented *uniquely* as a linear combination of two vectors e_1, e_2:

$$\begin{bmatrix} x \\ y \end{bmatrix} = x \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{e_1} + y \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{e_2}$$

We can draw this as something like

and this will allow us to put together all of $\mathbb{R}^2$ in an easy, straightforward way. This is the concept of a **basis** in a finite dimensional vector space.

Let's see a brief example:

Basis example

Let $\vec{x} = \begin{pmatrix} 6 \\ 4 \end{pmatrix}$, and let $B = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$ be a basis for $\mathbb{R}^2$. Decompose $\vec{x}$ as a linear combination relative to the basis:

Notationally, we can represent $\vec{x}$ in this basis as $[\vec{x}]_B = \begin{pmatrix} 6 \\ 4 \end{pmatrix}$

The coordinates in the basis vector represent the way that this vector is decomposed and how these elements are *expressed* in the given basis.

Encodings/decodings

We can think about this through describing basis representation as an **encoding** as well. That is, the vector $x \in \mathbb{R}^2$ exists as $\begin{pmatrix} 6 \\ 4 \end{pmatrix}$, and it is encoded in B as $\begin{pmatrix} 6 \\ 4 \end{pmatrix}$. We can then **decode** this back and figure out what the coefficients are as well. Obviously, this is a pretty trivial example.

Now, the standard basis B is only one of infinitely many that we can pick to represent our data! We'll now show how to find expressions of vectors in $\mathbb{R}^2$ in different coordinate systems.

Exercise

Let $x = \begin{pmatrix} 1 \\ \end{pmatrix}$ and consider the basis $B = \left\{ \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix} \right\}$ *side note: how do we know that this is a basis for* $\mathbb{R}^2$*?*

Describe the encoding and decoding steps of x in this basis

Solution

First, describing the encoding is as follows:

$$\begin{pmatrix} 1 \\ \end{pmatrix} = {}_1 \begin{pmatrix} 1 \\ 2 \end{pmatrix} + {}_2 \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

This can be represented as

$$\left(\begin{array}{cc|c} 1 & 2 & 1 \\ 2 & 1 & \end{array} \right) \left(\begin{array}{cc|c} 1 & 2 & 1 \\ 0 & & \end{array} \right)$$

This lets us solve for ${}_2 = 1$, and ${}_1 \ 2_2 = 1 \ \ {}_1 = .$

Therefore the encoding in B maps $\begin{pmatrix} 1 \\ \end{pmatrix} \ \ \begin{pmatrix} \\ 1 \end{pmatrix}$.

Now, using the obtained *encoded* coefficients, we can now decode to reconstruct the original vector as

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} + 1 \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ \end{pmatrix}$$

Pictographically,

$$x \underbrace{\qquad}_{\text{oi}} \underbrace{\qquad}_{\text{eoi}} x$$

Definition of a Basis

Formally, we can define a basis for $\mathbb{R}^n$ as follows. Let $B = \{v_1, v_2, \ldots, v_n\} = \{v_i\}_{ni=1}$ where each $v_i \in \mathbb{R}^n$. B forms a basis in $\mathbb{R}^n$ if and only if

1. The vectors $v_1, v_2, \ldots, v_n$ are linearly independent
2. Any vector in $\mathbb{R}^n$ can be expressed as a linear combination of the vectors in B.

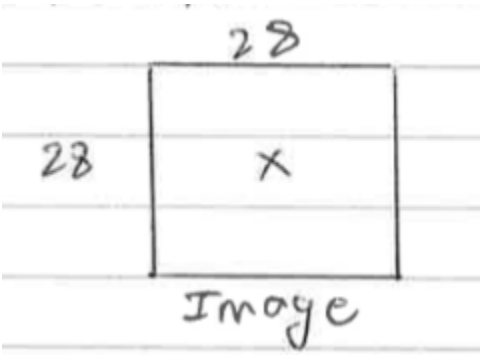
Brief Exercise:

What does property 1. give us? What does property 2. give us?

Extension to higher dimensions

From what we've seen to this point, there doesn't seem to be a point in choosing one basis over another, outside of simplicity. This is certainly the case when considering something like $\mathbb{R}^2$, which is small and has a relatively simple structure.

Consider however even taking 2×2 pixel images



and vectorizing them. These small images alone now live in $\mathbb{R}^4$.

In this high dimensional space, questions of

- Computation
- Visualization
- Analysis/Inference

become substantially harder and require more thought. The question of *good* bases and representations now becomes a lot more salient, and the problem of encodings matters more as well.

Linear Transformations

We now extend our review of linear algebra into the concept of linear transformations. We say that a map $: \mathbb{R}^n \to \mathbb{R}$ is linear if

- $(x + y) = (x) + (y)$ $x, y \in \mathbb{R}^n$
- $(x) = (x)$ $x \in \mathbb{R}^n \in \mathbb{R}$.

Now, how can we represent these transformations?

For this first example, we will take the standard coordinate system in $\mathbb{R}^n$ and $\mathbb{R}$. Let

$$B = \{e_1, e_2, \dots, e_n\}$$
$$B = \{e_1, e_2, \dots, e\}$$

Now, for any $x \in \mathbb{R}^n$, there exist scalars $\{_i\}_{ni=1}$ such that

$$x = {}_1e_1 + \ + {}_ne_n$$

Therefore,

$$\begin{aligned}(x) &= ({}_1e_1 + \ + {}_ne_n) \\ &= {}_1(e_1) + \ + {}_n(e_n)\end{aligned}$$

where the second line follows from linearity.

If we recall that matrix multiplication is defined for $A \in \mathbb{R}^{\times n}$ and $x \in \mathbb{R}^n$ as

$$Ax = \begin{matrix} | & | & & | \\ a_1 & a_2 & \dots & a_n \\ | & | & & | \end{matrix} \begin{matrix} x_1 \\ \\ x_n \end{matrix} = x_1a_1 + \ + x_na_n$$

We can apply the definition to our matrix multiplication above and see that

$$(x) = \underbrace{\begin{matrix} | & & | \\ (e_1) & \dots & (e_n) \\ | & & | \end{matrix}}_{n} \begin{matrix} 1 \\ \\ \underbrace{\ \ \ }_{[x]_B} \end{matrix}$$

We can seek to generalize this idea now!

Let $B = \{v_i\}_{ni=1}$ be a basis for $\mathbb{R}^n$ and let $B = \{w_i\}_{i=1}$ be a basis for $\mathbb{R}$. If we consider $\ : \mathbb{R}^n \to \mathbb{R}$, then we can do the following expression in matrix multiplication:

$$\begin{aligned}(x) &= {}_1(v_1) + \ + {}_n(v_n) \\ [(x)]_B &= {}_1[(v_1)]_B + \ + {}_n[(v_n)]_B \\ &= \begin{matrix} [(v_1)]_B & \dots & [(v_n)]_B \end{matrix} \begin{matrix} 1 \\ \\ \underbrace{\ \ \ }_{n} \\ [x]_B \end{matrix}\end{aligned}$$

The matrix defined above is the *matrix representation* of the linear map $\ $ with respect to the bases B and B. Note that this is specific to the representation of the matrix in the selected bases of the linear spaces.

Therefore,

$$[(x)]_B = [x]_B$$

Goal next time

Can we find a basis B such that the representation of is diagonal, i.e. a dilation operator?

Lecture 2: Eigenvectors

What is an eigenvector?

First, we say that a vector $x \in \mathbb{R}^n$ is an eigenvector of a matrix $A \in \mathbb{R}^{n \times n}$ if

$$Ax = x$$

for some $\in \mathbb{R}$. We call the corresponding constant an eigenvalue of A.

For an eigenvalue of a matrix A, it follows that

$$\begin{aligned} Ax &= x \\ Ax \ x &= 0 \\ (A \ ) &= 0 \end{aligned}$$

which will be a true statement if and only if

$$\mathrm{et}(A \ ) = 0$$

side note: why? what's one way to define the determinant in terms of the eigenvalues of a given matrix?

We call the polynomial in that comes from $\mathrm{et}(A \ )$ the **characteristic polynomial** of A.

⁝☰ Computing the eigenvalues and eigenvectors of a matrix

Consider the following matrix $A = \begin{bmatrix} & 2 \\ 1 & 4 \end{bmatrix}$. Compute its eigenvalues and eigenvectors

Solution: First, notice that

$$\begin{aligned} A \ &= \begin{bmatrix} & 2 \\ 1 & 4 \end{bmatrix} \\ \mathrm{et}(A \ ) &= 0 \ (\ )(4 \ ) \ 2 = 0 \\ {}^2 \ &+ 10 = 0 \\ (\ 2)(\ ) &= 0 \\ {}_1 = \ , {}_2 &= 2 \end{aligned}$$

To compute the eigenvectors, we will proceed as follows:
${}_2 = 2 \ \ (A \ 2)x = 0$, so

$$\begin{aligned} \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ x_1 + 2x_2 &= 0 \\ x_1 &= 2x_2 \\ v_2 &= \begin{pmatrix} 2 \\ 1 \ \end{pmatrix} \end{aligned}$$

${}_1 = \ \ (A \ )x = 0$, so

$$\begin{bmatrix} & 2 & 2 \\ 1 & & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$
$$x_1 \; x_2 = 0$$
$$x_1 = x_2$$
$$v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Next, we define the **spectrum** of a matrix A.

Spectrum

Let $A \in \mathbb{R}^{n \times n}$. We define the **spectrum** of A, denoted (A) as

$$(A) = \{ \in \mathbb{R} | \, \text{et}(A \;) = 0\}$$

In the previous example, the spectrum of $\begin{bmatrix} & 2 \\ 1 & 4 \end{bmatrix}$ is $\{2, \}$

Spectral Radius

The **spectral radius of a matrix** A, denoted (A), is given by

$$(A) = \text{soteei}(A)$$

In the previous example, $(A) =$.

Exercises

What's the spectral radius of the following matrices?

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}, B = \begin{bmatrix} 20 & 0 \\ 0 & 10 \end{bmatrix}$$

Answer: $(A) = 10$, $(B) = 20$.

Linear Independence of Eigenvectors

Let's return to the matrix given as $A = \begin{bmatrix} & 2 \\ 1 & 4 \end{bmatrix}$. Are the eigenvectors of A linearly independent?

Well, the eigenvectors were $v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $v_2 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$. To determine whether or not they're linearly independent, we can do the following:

$$
\begin{aligned}
{}_1\begin{pmatrix}1\\1\end{pmatrix} + {}_2\begin{pmatrix}2\\1\end{pmatrix} &= \begin{pmatrix}0\\0\end{pmatrix}\\
\begin{pmatrix}1 & 2\\1 & 1\end{pmatrix}\begin{pmatrix}{}_1\\{}_2\end{pmatrix} &= \begin{pmatrix}0\\0\end{pmatrix}\\
{}_1\ 2_2 &= 0\\
{}_1 + {}_2 &= 0\\
{}_1 &= 2_2\\
{}_1 &= {}_2\\
{}_1 = {}_2 &= 0
\end{aligned}
$$

Does this hold more generally?

Proposition

Eigenvectors corresponding to different eigenvalues must be linearly independent

Proof of Proposition

We will proceed by induction. First, we note that $n = 1$ is trivial. *Side note: why?*

Next, we assume the statement is true for 1 eigenvectors, and we seek to show that this holds for the -th eigenvector

First, if this holds for the first 1 eigenvectors v_1 through v_1, then the only solution to

$$
{}_1v_1 + \ + {}_1v_1 = 0
$$

is the trivial solution. Now, assume that we add the -th eigenvector v to the above equation and that there exists a non-trivial solution to the equation

$$
{}_1v_1 + \ + v = 0
$$

Now, we can apply $(A\ )$ to the above equation as follows:

$$
\begin{aligned}
(A\ )\,({}_1v_1 + \ + v) &= (A\ )0\\
(A\ ){}_1v_1 + \ + (A\ )v &= 0\\
{}_1({}_1\ )v_1 + \ + {}_1({}_1\ ) + (\ )v &= 0\\
{}_{1\ i=1}\ {}_i({}_i\ ) &= 0
\end{aligned}
$$

Now, by the induction assumption, $\{v_i\}_{1i=1}$ forms a linearly independent set. As such, as this is a sum over only the vectors $\{v_i\}_{1i=1}$, we know that for each ${}_i \neq$, that ${}_i({}_i\ ) = 0$ and therefore ${}_i = 0$ in each of these equations.

Now, returning to the original linear independence equation, it must be the case that $v = 0$ now, therefore implying that $= 0$, thus indicating that $\{v_i\}_{i=1}$ forms a linearly independent set.

Remark

An $n \times n$ matrix can only have at most n distinct eigenvalues. Why?

Diagonalizability

Diagonalizability is a really important property when considering matrices. We provide an example of a non-diagonalizable matrix below:

Example of a defective matrix

Consider the following matrix $A = \begin{pmatrix} & 2 \\ 0 & \end{pmatrix}$. Its eigenvalues are $_1 = _2 = $.

Now,

$$(A\ ) = 0 \begin{pmatrix} 0 & 2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$
$$x_2 = 0, x_1\text{ree}$$

So our only eigenvector is $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, so A is defective.

First, we say that a matrix is **non-defective** or **diagonalizable** if the following holds:

Diagonalizability

We say that a matrix $A \in \mathbb{R}^{n \times n}$ is diagonalizable, or non-defective, if its eigenvectors $\{v_i\}_{ni=1}$ span $\mathbb{R}^n$.

Now, consider the following. Let the matrix A have $\{u_i\}_{ni=1}$ linearly independent eigenvectors. As such,

$$Au_1 = _1u_1$$
$$Au_2 = _2u_2$$

$$Au_n = _nu_n$$

So, for the matrix $=$ ^{1} ^{2} , we can see that

$$A \quad \begin{matrix} | & | & & | \\ u_1 & u_2 & \dots & u_n \\ | & | & & | \end{matrix} = \begin{matrix} | & | & & | \\ u_1 & u_2 & \dots & u_n \\ | & | & & | \end{matrix} \begin{matrix} ^1 \\ \\ ^2 \\ \\ \\ \\ ^n \end{matrix}$$

This can be written as $A = $, implying $A = {}^1$, and therefore A and are similar matrices. Given that they are similar matrices, they share eigenvalues.

Proof of the above proposition (non-essential)

First, let A and B be similar matrices via , i.e. $A = B^{1}$. Then

$$\begin{aligned} \mathrm{et}(A\ ) &= \mathrm{et}(B^{1}\ ) \\ &= \mathrm{et}(B^{1}\ ^{1}) \\ &= \mathrm{et}(\ (B\ )^{1}) \\ &= \mathrm{et}()\,\mathrm{et}(B\ )\,\mathrm{et}(^{1}) \\ &= \mathrm{et}(B\ ) \end{aligned}$$

thus showing that these two matrices share a characteristic polynomial (implying their eigenvalues share an algebraic multiplicity.)

Next, we show that the eigenvalues share geometric multiplicity.
Now, if $Av = v$, then $B^{1}v = v\ \ B^{1} = {}^{1}v$. So, if v is an eigenvector of A with eigenvalue , the ${}^{1}v$ is an eigenvector of B with the same eigenvalue. As such, every eigenvalue of A is an eigenvalue of B, and interchanging these objects in the above computations implies that they share eigenvalues with the same geometric multiplicities.

Combining the two results gives the solution.

Proposition

All eigenvalues of a symmetric matrix are real. Eigenvectors corresponding to different eigenvalues are orthogonal

Spectral Theorem

A symmetric matrix has the following decomposition:

$$A =$$

where $\ =\ =$

Exercise

Let $A = \begin{pmatrix} 1 & \\ & 1 \end{pmatrix}$.

1. Is A diagonalizable? Answer without computing
2. Find the eigenvalues of A
3. Find the eigenvectors of A. What can you say about these eigenvectors?

4. Using the eigenvectors as a basis (why can we do this?), write the vector $\begin{pmatrix} 0 \\ 2 \end{pmatrix}$ as a linear combination of the two vectors.
5. Write the spectral decomposition of A.

Exercise

Let $: \mathbb{R}^2 \to \mathbb{R}^2$ be a linear map defined as

$$\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right) = \begin{pmatrix} x_1 + x_2 \\ x_1 + x_2 \end{pmatrix}$$

1. Find the matrix of with respect to $\left\{ \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\}$
2. Find the matrix of with respect to $\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}$

Lecture 3: Singular Value Decomposition (SVD)

In this lecture, we'll introduce the SVD as a generalization of the spectral theorem.

Spectral Theorem

We know from last lecture that. asymmetric matrix is diagonalized by orthogonal eigenvectors, which can be written as

$$A =$$

where the columns of are orthonormal eigenvectors of A. We say that

- is an **orthogonal** matrix, i.e. $= ^1$.
- is a diagonal matrix consisting of eigenvalues of A.

This is a nice property for matrices to have.

Question: Does this hold for wide classes of matrices?

Answer: No!

A reason this doesn't work is the non-diagonalizable matrices in $\mathbb{R}^{n \times n}$ as discussed previously.

Question: A reasonable question is can we extend this idea of a spectral theorem to non-diagonalizable matrices? What about non-square matrices?

Approach

We're going to use the diagonalization of AA as a potential guide to answer this question more thoroughly.

Proposition 1: $\mathrm{er}(A) = \mathrm{er}(AA)$

Proof:

- () Let $x \in \mathrm{er}(A)$. This means that $Ax = 0$. Trivially then $AAx = 0 \;\; x \in \mathrm{er}(AA)$.
- () Let $x \in \mathrm{er}(AA)$, so $AAx = 0$ by definition. Let's multiply the previous equation on both sides by x.

$$\begin{aligned} xAAx &= 0 \\ Ax_{22} &= 0 \\ Ax &= 0 \\ x &\in \mathrm{er}(A) \end{aligned}$$

This concludes the proof.

Side note

Before proceeding from where we left off last time, I want to return to the spectral decomposition to make the following interpretation.

As we've seen, for any $x \in \mathbb{R}^n$, we have that for a basis v_i that $x = {}_1v_1 + \; + {}_nv_n$. This means that $[x]_B = (_1 \quad \dots \quad {}_n \quad)$, and that $x = V[x]_B$

In essence, V is a map of x from one coordinate system to the next. How can one go back? Well, that's just V^1

So, how can we interpret the spectral decomposition through this lens? We previously denoted that V was an orthogonal matrix. Orthogonal matrices are special, as they correspond to **rotations** and **reflections**.

As such, what's happening with $A = VV$? As $V^1 = V$, we can say explicitly that V **rotates** a vector x into the basis $B = \{v_i\}_{ni=1}$ defined by the eigenvectors. This takes in the standard representation, and *represents it in the eigenbasis! It makes the vector* $[x]_B$
Then what happens? In this basis, we just need to **scale**, or dilate the vectors via .
Lastly, V rotates this representation back from $[y]_B$ back into the original basis, so that now I'm left with a standard representation for x (or whatever basis you were originally in. There's nothing special about the standard basis!)

Exercise: $\mathrm{er}(A) = \mathrm{er}(AA)$.

Now, let's take a non-square matrix $A \in \mathbb{R}^{\times n}$ where $= \mathrm{r}(A)$. We can recall the dimension theorem (sometimes known as rank nullity)

Dimension Theorem $n = \mathrm{ier}() + \mathrm{r}(A)$.

Exercise: $\mathrm{r}(A) = \mathrm{r}(A) = \mathrm{r}(AA) = \mathrm{r}(AA)$.

Building the SVD

Let's consider the diagonalization of AA. This matrix has

- Eigenvalues ${}_1, {}_2, \dots, {}_n$ where ${}_1 \quad {}_n$

- Eigenvectors $v_1, v_2, \dots v_n$ where each $v_i \in \mathbb{R}^n$

Now, per our notation convention, $AAv_j = {}_j v_j$.

Since AA is symmetric, we have that the eigenvectors are orthogonal.

Observation:

Since $\mathrm{r}(AA) = \mathrm{r}(A) =$, we have that exactly eigenvalues are positive as AA is positive semidefinite (PSD).

Exercise: Why is AA PSD?

Claim: Positive eigenvalues of AA are equal to the positive eigenvalues of AA.

Proof:

First, let's define the following objects:

- ${}_j = {}_j 1 \;\; j \;\; n$
- $u_j = {}_{1_j} A v_j 1 \;\; j$ (Note: Only up to since this isn't defined for $j \;\; + 1$ as division by zero.)

Now, notice that

$$\begin{aligned} AAu_j &= 1_j AAAv_j \\ &= 1_j A_j v_j \\ &= 1_{jj} A v_j \\ \\ &= 1_{jjj} u_j \\ &= {}_j u_j \end{aligned}$$

indicating that $AAu_j = {}_j u_j$ for all $1 \;\; j \;\;$.

Now, are these eigenvectors orthogonal? Notice that

$$\begin{aligned} u_j u &= 1_j v_j AAv \\ &= 1_j v_j v \\ &= \begin{cases} 1 & j = \\ 0 & \text{ese} \end{cases} \end{aligned}$$

This implies that the positive eigenvalues coincide, as claimed previously, and that $u_1, \dots, u$ are the eigenvectors corresponding to ${}_1, \dots,$ of AA.

Now, we can do the following: I can build a orthogonal matrix such that

$$= \begin{array}{cccccccc} | & | & & | & | & & | \\ u_1 & u_2 & \dots & u & u_{+1} & \dots & u \\ | & | & & | & | & & | \end{array}$$

where $u_{+1}, \dots, u$ complete the set to be an orthonormal basis of $\mathbb{R}$.

I can play the same game with a matrix that we call

$$V = \begin{matrix} | & | & & | & | & & | \\ v_1 & v_2 & \dots & v & v_{+1} & \dots & v_n \\ | & | & & | & | & & | \end{matrix}$$

where now the v_i's span $\mathbb{R}^n$.

Previously, we stated that $Av_j = {}_j u_j 1 \ \ j \ \ $. Now, for $j \ \ +1$, we know that $Av_j = 0$, corresponding to the zero eigenvalues of AA, indicating that we can say that

$$Av_1 = {}_1 u_1$$
$$Av_2 = {}_2 u_2$$

$$Av_{+1} = 0$$

$$Av_n = 0$$

So, from here we can build a matrix $\in \mathbb{R}^{\times n}$ where

$$= \begin{matrix} {}_1 & 0 & \dots & 0 \\ 0 & {}_2 & & \\ & & & \\ 0 & 0 & \dots & 0 \end{matrix}$$

Now, we can see that $AV =$ where $A \in \mathbb{R}^{\times n}$, $V \in \mathbb{R}^{n \times n}$, $\in \mathbb{R}^{\times}$, and $\in \mathbb{R}^{\times n}$ where and V are orthogonal, and only has entries along the diagonal.

Now,, as $VV =$, we have that $A = V$! Notice how this looks exactly like the spectral theorem!

Exercise: Is the SVD of a matrix A unique?

> **Definition of Singular Value and SVD**
>
> We call each of the objects ${}_j$ the **singular values of** A, and
> $A = V$ is the **singular value decomposition** of A

How can I interpret this like with the spectral decomposition? It's the same idea, except now you're allowing for more flexibility with your rotations and your dimensions.

Now what you might end up with is something that rotates in a lower dimension, then embeds in a higher dimension, or vice versa. It's still the exact same intuition, however.

Reduced SVD

Sometimes, we seek to discard the zero singular values since they aren't particularly relevant to the object.

For the matrix product AB, we can write this in the form of the outer product of its columns/rows.

That is,

$$\begin{matrix} | & | & & | \\ a_1 & a_2 & \dots & a_n \\ | & | & & | \end{matrix} \begin{matrix} {}^{1} \\ {}^{2} \\ \\ \end{matrix} = a_1 {}_1 + a_2 {}_2 + \ + a_n$$

Applying this to the SVD, we see that

$$\begin{matrix} | & & | \\ u_1 & \dots & u \\ | & & | \end{matrix} \begin{matrix} {}_1 v_1 \\ \\ v \end{matrix} = {}_1 u_1 v_1 + {}_2 u_2 v_2 + \ + uv$$

So we can say that $A = {}_{i=1} {}_i u_i v_i$.

Now, we can say that a reduced SVD is given by

$$\underbrace{\begin{matrix} | & & | \\ = u_1 & \dots & u \\ | & & | \end{matrix}}_{\in \mathbb{R}^{\times}} , \underbrace{V = \begin{matrix} | & & | \\ v_1 & \dots & v \\ | & & | \end{matrix}}_{\in \mathbb{R}^{n \times}} , \underbrace{\begin{matrix} {}^{1} \\ \\ \\ \end{matrix}}_{\in \mathbb{R}^{\times}}$$

So the reduced SVD can be given as $A = \quad V$.

Truncated SVD

It turns out, sometimes the most important information is only held in a few of the singular values of a matrix.

For example, consider a matrix where ${}_1 = 10$, and ${}_2 = 10^2$, $\quad$ 1. We might only need a couple of singular components to approximate this matrix effectively.

We can now consider $A \ {}_1 u_1 v_1$, or $A \ {}_1 u_1 v_1 + {}_2 u_2 v_2$, given by

$$A = \begin{matrix} | & | \\ u_1 & u_2 \\ | & | \end{matrix} \begin{pmatrix} {}^{1} & \\ & {}^{2} \end{pmatrix} \begin{pmatrix} & v_1 & \\ & v_2 & \end{pmatrix}$$

This might not always be a good approximation, but in some real datasets sometimes all you need is just a handful of singular values!

Lecture 4: More on the SVD and Matrix Norms

Starting off this lecture with a quiz. Once this is done...

Now, recall that the SVD of $A \in \mathbb{R}^{\times n}$ is given by $A = V$ for $\in \mathbb{R}^{\times}$, $V \in \mathbb{R}^{n \times n}$, $\in \mathbb{R}^{\times n}$, and $=$, $VV = {}_n$.

We also wrote the SVD of $A = {}_{i=1} {}_i u_i v_i$ where each of the summands is a rank-one outer product.

From here, we also discussed the reduced and truncated SVDs.

Exercise:

Compute the SVD of $A = xy$ for arbitrary $x \in \mathbb{R}$, $y \in \mathbb{R}^n$.

Step 1: Compute the right singular vectors

First, we need to compute the right singular vectors through finding the eigenvectors of AA to get V.

For us, $AA = yxxy = x_{22}yy$. Looking at this, it's clear the unit length eigenvector $v = _{yy_2}$.

Step 2: Compute the singular values

Using a Rayleigh quotient argument, we can compute the eigenvalue of AA by doing
$vAAv = _{x_{22}y_{22}}yyyy = x_{22}y_{22}$.

Step 3: Compute the left singular vectors

This is almost exactly identical to Step 1, so we can just see automatically that $u = _{xx_2}$.

Step 4: Full SVD

If you only care about the reduced SVD, we're done now. If you want to compute the full SVD, you just need to find $\ 1$ and $n\ 1$ vectors orthogonal to x and y, respectively, to pad out the matrices and V.

Exercise:

Let $A =$ be a diagonalizable matrix. How is the eigenvalue decomposition related to the singular value decomposition of A? If I restrict this to $A\ \ 0$ (i.e. A is positive semidefinite), how does this change?

Geometric Interpretation of the SVD

Here I walk through the matlab script SVD_interpretation.m.

Matrix Norms

In the same way that we have notions for how "big" a vector is, we can play the exact same game with matrices.

For example, I know that

$$A = \begin{pmatrix} 10 & 0 \\ 0 & 1000 \end{pmatrix}$$

is in some sense bigger than

$$B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

We'll now quantify this statement a little bit more exactly.

Frobenius norm

The first norm that we want to define is called the Frobenius norm.

> **Definition**
>
> For a matrix A, we say that
>
> $$A_2 = \quad_{ij} a_{2ij}$$
>
> is the **Frobenius norm of A**

Exercise: How can you relate the Frobenius norms to more standard vector norms?

Example:

Let's compute the Frobenius norm of $A = \begin{pmatrix} 2 & 1 \\ 0 & \end{pmatrix}$.

Well, $A_2 = \quad_{ij} a_{2ij} = 4 + 1 + \ = 14$, so $A = 14$.

Exercise: Prove that $A_2 = \quad_{i=1}\ 2i$.

Spectral/$_2$ norm

The Frobenius norm is useful in some contexts, but maybe not all. Another useful matrix norm is the spectral norm, which is a measure of how much a matrix A maximally stretches a vector.

$$A_2 = \quad_{1j}{}_{j\ =\ 1}$$

Side Note:

We define what's known as the **condition number** of a matrix as $\ = {}_1$. This appears a lot in numerical linear algebra, and may show up later in this course.

Eckart-Young(-Mirsky) Theorem

Consider a matrix $A \in \mathbb{R}^{\times n}$ where $\mathrm{r}(A) =$. Let $B \in \mathbb{R}^{\times n}$ be another matrix of $\mathrm{r}(B) =$. For any , if we define

$$A = \quad_{i=1}{}_i u_i v_i$$

for $A = V$, we have that

$$A() = \mathrm{ri}_{\mathrm{r}(B)=} A\ \ B$$

and furthermore that

$$A() = \mathrm{ri}_{\mathrm{r}(B)=} A\ \ B_2$$

Proof Sketch:

The Frobenius norm is what's known as "unitarily invariant", meaning that you can rotate things as freely as you want and the norm is preserved (like vectors). This allows us, effectively, to only consider i for some matrix .

Writing this out, as $\mathrm{ri}_{\mathrm{r}(B)=}A\ B = \mathrm{ri}_{\mathrm{r}(B)=}A\ B_2$, we're left with

$$_2 = \ _i(_i\ _{2ii}) + \ _{i\neq j}\ _{2ij}$$

The rightmost term is positive, so we only consider diagonal matrices . To minimize this such that $\mathrm{r}() = $, we should set $_{ii} = _i$.

Rotating back, we get the proof statement that we set out.

What's the remaining error in approximation?

Well, from above, we have that

$$A\ A() = _{+1} + \ +$$

and that

$$A\ A()_2 = _{+1}$$

This is super useful in the field of *low-rank approximation*, where we can compress and approximate matrices like A using truncated decompositions. In many cases, we still preserve much of the existing structure with only a few singular values!

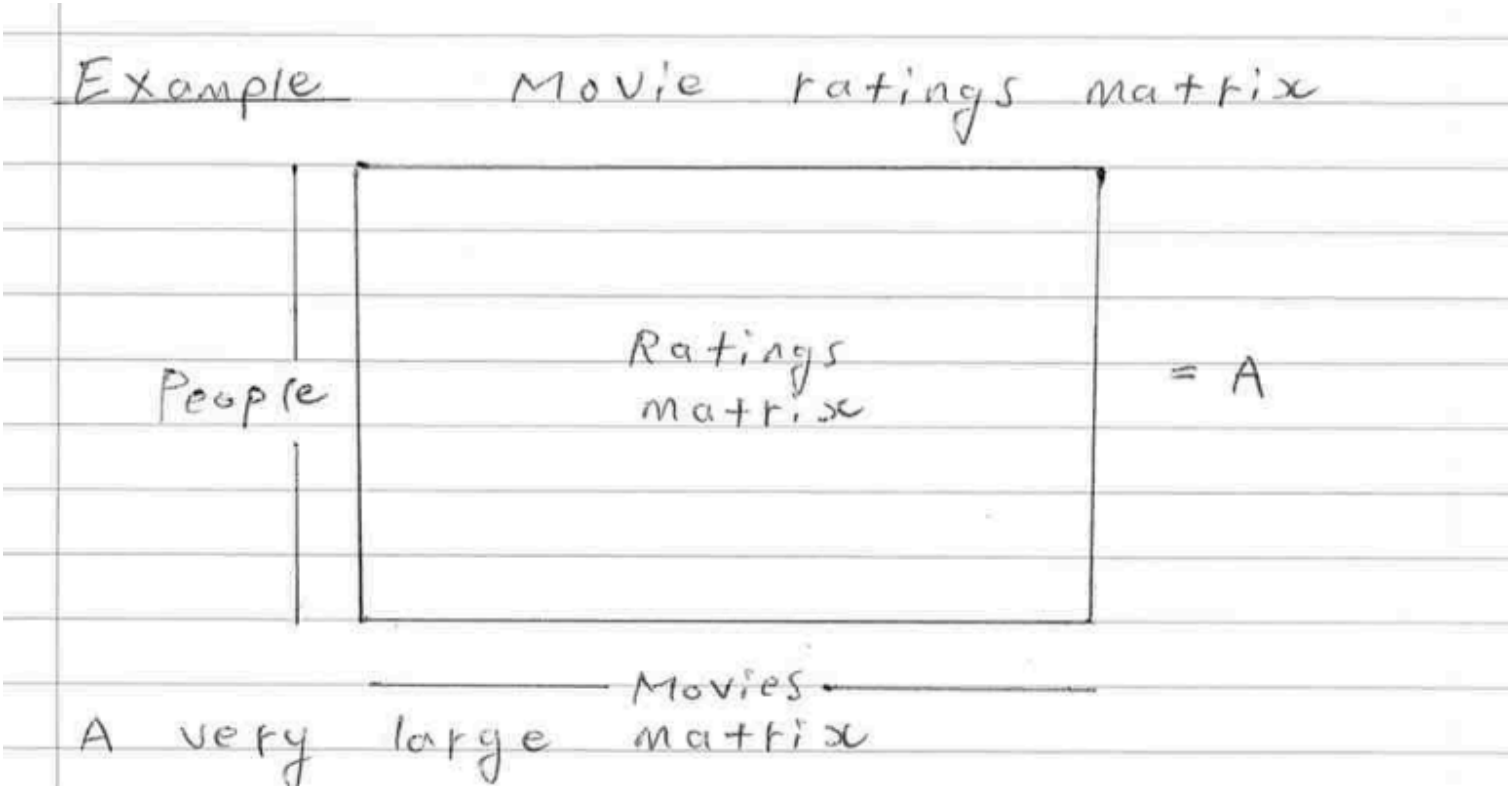
Now, we can understand this

Now, we can use the Low_Rank_Approximation.m file for the demonstration.

Lecture 5: Principal Component Analysis

Consider data represented as $\{x_i\}_{ni=1} \subset \mathbb{R}$ for some large .

A classic example is the Netflix Problem

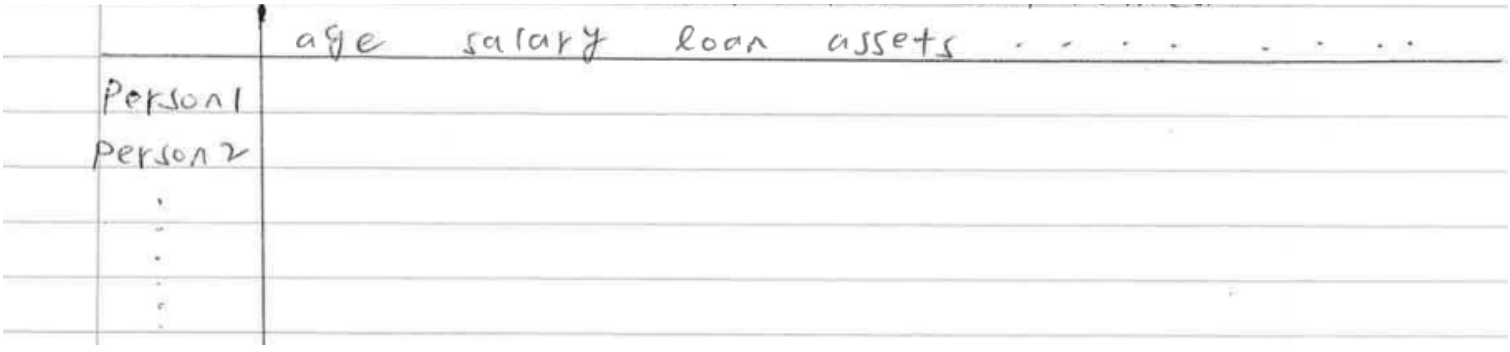


Problem: Can we reduce the dimension of the data in A so that the essential information is still captured?

We can think about our data as observations versus features. More specifically, if

$$= \begin{matrix} x_1 \\ x_2 \\ \\ x_n \end{matrix}$$

Each column is a *feature*, and each row is an **observation**.



We can think of each person as a vector in $\mathbb{R}$ where each entry informs a particular attribute.

For each person, there are lots of attributes. Some of these, however, might be redundant and not capture that much information. Is there a way that we can reduce the dimension of this data in order to make the analysis easier?

Basis vectors in $\mathbb{R}$

Let $\{u_i\}_{i=1}$ be an orthonormal basis for $\mathbb{R}$, and build a matrix such that

$$= \begin{vmatrix} & & \\ u_1 & \dots & u \\ & & \end{vmatrix}$$

As the columns are orthonormal, we know that $=$ $=$

We can represent a vector x in this basis as $x = {}_i {}_i u_i$

Problem: Compute the coefficients ${}_i$

To do this, we can see that for any u_j that

$$u_j x = {}_i {}_i u_j u_i$$

$$= {}_i {}^{iij}$$
$$= {}_i$$

As this holds for any j, we can say that $x = {}_i x, u_i u_i$ where $x, y = xy$

So, if I have that

$$x = \begin{vmatrix} & & \\ u_1 & \dots & u \\ & & \end{vmatrix} \underbrace{\begin{matrix} \\ \\ \end{matrix}}_{1}$$

it follows that $= x$.

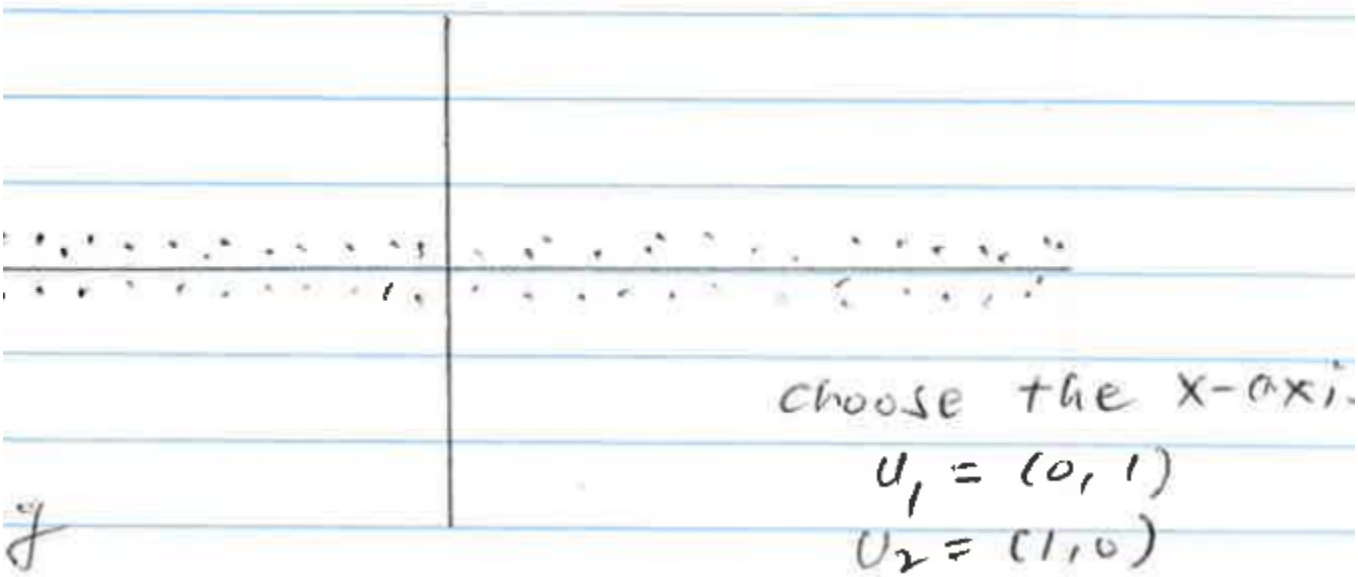
Which orthonormal basis is "best"?

This question will motivate the concept known as principal component analysis, or PCA.

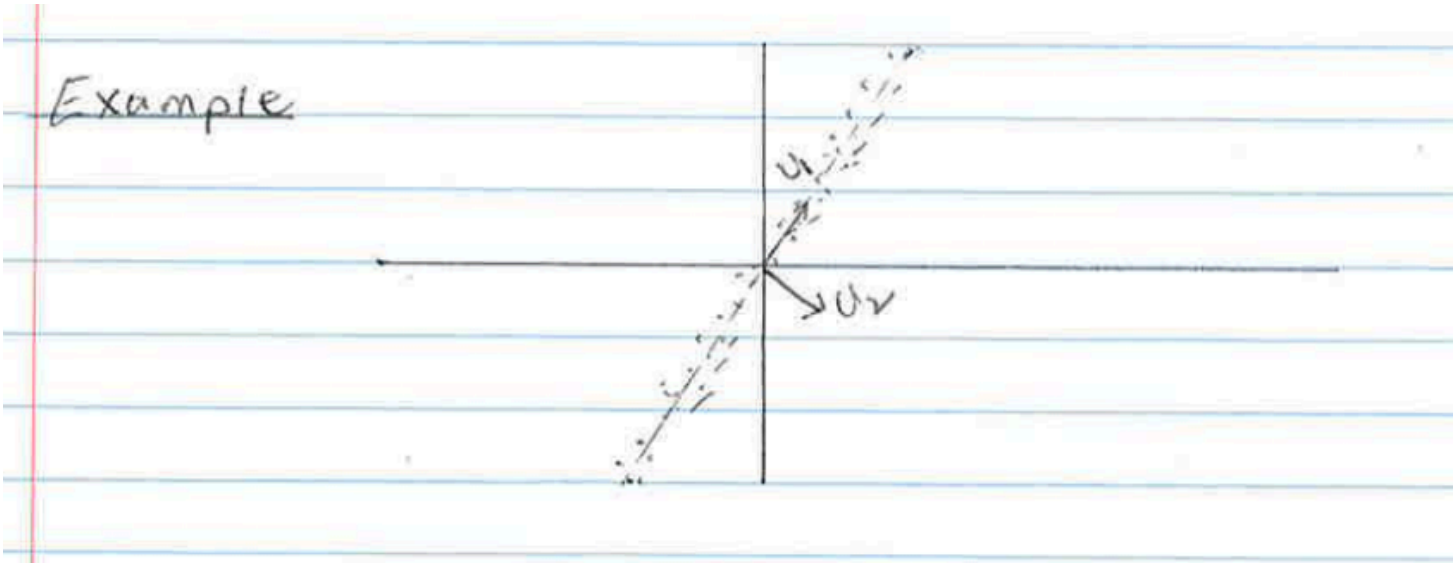
PCA Motivations

The idea is that

1. compute u_1 such that, over all projections onto 1 subspaces, it is **variance maximizing**



2. The projection of onto $\mathrm{s}\{u_1, u_2\}$ is variance maximizing on projections over 2 subspaces



Recall the ellipse view of the SVD...

Definition

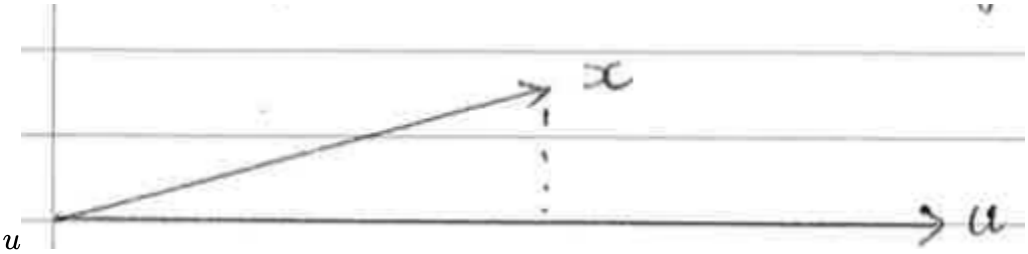
The variance of $= \{x_i\}_{ni=1}$ is defined as

$$\mathrm{r}() = 1n \ {}_{n}{}_{i=1} (x_i \ )^2$$

where

$$= 1n \ {}_{n}{}_{i=1} x_i$$

If the goal is to maximize the variance of a projection of , then we note that the projection of x onto u is given by $(ux)u$



Example:

Project $(\begin{array}{ccc} 1 & 4 & \end{array})$ onto $(\begin{array}{ccc} 0 & 0 & 1 \end{array})$.

Answer:

$\mathrm{ro}_x = ((\begin{array}{ccc} 1 & 4 & \end{array}) \ (\begin{array}{ccc} 0 & 0 & 1 \end{array})) (\begin{array}{ccc} 0 & 0 & 1 \end{array}) = (\begin{array}{ccc} 0 & 0 & \end{array})$.

PCA derivation

Without loss of generality, we assume that $= 0$ for simplicity.

Thus, variance maximization is given by

$$\mathrm{r}_u 1n \ _{n}{}_{i=1} \left(ux_i\right)^2$$

Expanding this out, we can see that

$$1n \ _{n}{}_{i=1} \left(ux_i\right)^2 = 1n \ _{n}{}_{i=1} ux_i x_i u$$

$$= u \underbrace{1n \ _{n}{}_{i=1} x_i x_i}_{\text{orietri}} u$$

So, the problem is given by

$$\mathrm{r}_u uu$$

Sidenote

We can rewrite $= 1n$ where

$$= \begin{matrix} x_1 \\ \\ x_n \end{matrix} \in \mathbb{R}^{n\times}$$

Now, how can we find u? Calculus and Lagrange Multipliers!

We can define a function $: \mathbb{R} \to \mathbb{R}$ given by $(u) = uu$.

Using the method of Lagrange multipliers, we can find the optimum of given the constraint that $uu = 1$.

This can be given by

$$\mathrm{r}_u \underbrace{uu}_{\text{oetie}} \underbrace{(uu\ 1}_{\text{ostrit}})$$

If we differentiate this with respect to u and set it equal to zero, we get that

$$\begin{aligned} 2u\ 2u &= 0 \\ u &= u \end{aligned}$$

This means that is an eigenvalue and u is an eigenvector!

Now, if the goal was to *maximize* the value uu, which value will give this?
()!!

If we want to compute the second direction of most variance, that is orthogonal to u_1, we can consider $\mathrm{r}_{uu_1} uu = u_2$, and inductively get all of the principle components.

PCA summary

So, given $\{x_i\}_{ni=1} \subset \mathbb{R}$, presuming that the data is centered, we can do the following:

1. Form the covariance matrix $=$ where the rows of are x_i
2. Compute the eigendecomposition of
    1. u_1 is the direction associated with maximum variance
    2. u_2 points in the direction of the second largest variance

Lastly, the larger the eigenvalue means the more **significant** the direction is in terms of the total variance of an object!

Lecture 6: More on PCA

PCA review

First, recall from the previous lectures that for a data matrix

$$= \begin{matrix} x_1 \\ \\ x_n \end{matrix}$$

Here, each vector $x_i \in \mathbb{R}$. We assume that the data is centered, or mean zero.

We defined the max principal component as the direction of *maximum variance:* here, we used the method of Lagrange multipliers and found that the direction that maximized the variance was the eigenvector corresponding to the largest eigenvalue of .

We saw a similar thing with respect to the second principal component, and so on and so forth.

Rayleigh Quotient

For symmetric matrices, are there easier ways to compute the eigenvalues of our matrices?

Quadratic form:

> **Definition**
>
> We say that a function $: x \ \ xAx$ is a **quadratic form**

Example

Let $A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$. Then for $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$,

$$\begin{aligned} xAx &= (x_1 \quad x_2) \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ &= (x_1 \quad x_2) \begin{pmatrix} x_1 + 2x_2 \\ 2x_1 + x_2 \end{pmatrix} \\ &= x_{21} + x_{22} + 4x_1x_2 \end{aligned}$$

Hence the name *quadratic form*

Proposition: $xAx = _{ij} a_{ij}x_ix_j$

Proof:

Notice that

$$\begin{aligned} xAx &= \ _i x_i(Ax)_i \\ &= \ _i x_i \ \ _j a_{ij}x_j \\ &= \ _{ij} a_{ij}x_ix_j \end{aligned}$$

Now, for the concept of the Rayleigh quotient, the goal is to either maximize or minimize the quadratic form.

Exercise: With no constraints on x, what is $_x \, xAx$? $\mathrm{i}_x \, xAx$?

Now, this is clearly unbounded. Let's consider instead the objective $_{x_2=1} \, xAx$:

Now, as $A = A$, we have that $A = VV$ where is diagonal and V is orthogonal. Now, we can reexamine the quadratic form as

$$xAx = xVVx = \big(Vx\big)\big(Vx\big) = yy$$

where $y = Vx$. Now, what is yy?

Well, as $= \mathrm{i}(_1, \ldots, _n) =$ ^{1} $_n$, we have that

$$yy = \quad _i{}^i y_{2i}$$

which clearly implies that

$$_n \quad _i y_{2i} \;\; yy \;\; _1 \quad _i y_{2i}$$

and furthermore, $y_{22} = _i y_{2i}$ so

$$_n y_{i22} \;\; yy \;\; _1 y_{22}$$

Exercise: Prove $x_{22} = y_{22}$.

Using the above exercise, dividing both sides by x_{22}, we have that

$$_{\mathrm{i}}(A) \qquad \underbrace{xAxxx}_{\text{eihotiet}} \;\; (A)$$

When is equality achieved? Let $x = v_1$. Then the numerator equals $v_1 A v_1 = v_{11} v_1 = _1 v_1 v_1$, and the denominator equals $v_1 v_1$.

Similarly, the minimum is achieved by v_n.

Exercise: How is the first principal component connected to the Rayleigh quotient?

Courant-Fischer Formula

Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix with $_1 \quad _n$ and corresponding eigenvectors $v_1, \ldots, v_n$. Then

$$\begin{aligned}
_n &= \mathrm{i} \underset{x \neq 0}{} xAxxx \\
_{n1} &= \mathrm{i} \underset{xv_n, x \neq 0}{} xAxxx \qquad\qquad = \underset{xv_1, \ldots, v_{n2}, x \neq 0}{} xAxxx \\
_2 &= \mathrm{i} \underset{xv_n, \ldots, v, x \neq 0}{} xAxxx \qquad\qquad = \underset{xv_1, x \neq 0}{} xAxxx \\
_1 &= \underset{x \neq 0}{} xAxxx
\end{aligned}$$

Exercise: How are Courant Fischer and PCA connected?

Applications to PCA

Now, returning to a data matrix as before.

We are interested in choosing principal components $v_1, \ldots, v$ to project the data onto (i.e. capturing the orthogonal directions that contain the *most variance* in the data).

Exercise:

Let V be the matrix of principal components.
1. Find the coefficients, (i.e.) the projections of the data onto v.
2. What is , the decoding of the data onto the principal components?

Solution

First considering x_1, we have that

$$\begin{aligned} x_1 &= (xv_1)v_1 + \ + (x_1v)v \\ &= v_1\left(v_1x_1\right) + \ + v\left(vx_1\right) \\ &= \left(v_1v_1 + \ + vv\right)x_1 \\ &= VVx_1 \end{aligned}$$

From this, we can read that the coefficient matrix is V.

If we want to reconstruct the output, we can approximate
$= VV$.

SVD and PCA

If we let $= V$, we can take the SVD of the data matrix and see that

$$\begin{aligned} &= \\ &= [V]V \\ &= VV \\ &= V^2V \end{aligned}$$

so the SVD might be a cheaper thing to consider instead of forming and then doing the full eigenvalue decomposition! "Better" is more of an NLA style question, however, so we can worry about this later.

Lecture 7: PCA Projection

First, want to recall what it means for a vector to be projected onto another vector. We write this as

$$\mathrm{ro}_u x = uxu_2u$$

Now, given that we have this in mind, we can do the following. Let u be a unit length vector. Then $\mathrm{ro}_u x = uxu = u(ux) = (uu)x$. What is the object in parenthesis? A matrix! This is a projection onto the 1 dimensional subspace $\mathrm{s}\{u\}$.

Now, if I have a collection of orthonormal vectors, how might I build a projection onto $\mathrm{s}\{v_1, \ldots, v\}$? Well, do it one by one.

$$\mathrm{ro}_{\mathrm{s}\{v_1,\ldots,v\}} x = v_1v_1x + \ + vvx$$

We call the matrix $= {}_{i=1} v_iv_i$ a **projection matrix**.

Now, what does this particular term look like? Tie this to the spectral decomposition. Mention how it's related to the SVD in essence. Ask questions about its rank. Draw a picture about how it relates to the eigenspaces of a matrix, and show geometrically what this looks like a projection onto. How would it maybe be different in a few different examples? Draw some pictures.

So, recall from previously what we've learned about PCA. Let's now think about projections onto the principal components.

If we choose principal components, i.e.

$$V = \begin{matrix} | & & | \\ v_1 & \dots & v \\ | & & | \end{matrix}$$

with , and ideally

We can project x_i onto the subspace spanned by the first principal components using the decomposition $x_i = VVx_i$.

So, our data matrix $=$ $\begin{matrix} x_1 \\ \\ x_n \end{matrix}$

Now, as $x_i = VVx_i$, we can use matrix multiplication to see the following:

$$\begin{matrix} | & & | & & | & & | \\ x_1 & \dots & x_n & = VV & x_1 & \dots & x_n \\ | & & | & & | & & | \end{matrix}$$
$$= VV$$
$$= \Big(VV\Big) = VV$$

This is a good formula for how to compute a projection given a data matrix.

Relationship to the SVD of

Let's see what the relationship to the SVD is now.

First, let $= V$. Now, let's compute .

$$= VVV$$

Now, notice that

$$\begin{matrix} v_1 \\ \\ v_n \end{matrix} \quad \begin{matrix} | & & | \\ v_1 & \dots & v \\ | & & | \end{matrix} = \begin{pmatrix} \\ 0 \end{pmatrix} \in \mathbb{R}^{n\times}$$

So,

$$\underbrace{}_{\in\mathbb{R}^{\times n}} \underbrace{\begin{pmatrix} \\ 0 \end{pmatrix}}_{\in\mathbb{R}^{n\times}} = \begin{matrix} 1 \\ \\ \underbrace{0 \qquad 0}_{\in\mathbb{R}^{\times}} \end{matrix}$$

Write this term out when doing the whole multiplication in explicit detail for them

What we're left with now is $= V$ where

$$= \begin{matrix} | & & | \\ u_1 & \dots & u \\ | & & | \end{matrix}, \text{ with } V = \begin{matrix} | & & | \\ v_1 & \dots & v \\ | & & | \end{matrix}, \text{ and } \text{ defined as above.}$$

Exercise: What is the term defined above?

It's the best r- approximation of !

Steps in PCA and Computational Complexity: Omitting

PCA is great, but we should understand how expensive it is for larger and larger matrices. In this section we'll review *how* one undertakes PCA and how much it costs computationally.

Step 1: Centering the data

From previously, we saw that we needed to center the data in . This involves computing the mean.

Question: What's the computational complexity of centering?

Step 1: Compute the mean $= \frac{1}{n} \sum_i x_i$. This involves n additions and 1 division.
Step 2: Centering each row takes subtractions
Step 3: Repeat for n rows

The final complexity is (n)!

Step 2: Build the covariance matrix

As $= \frac{1}{n}$, we can decompose this as $= \frac{1}{n} \sum_{ni=1} x_i x_i$, as we saw before. The cost to compute $x_i x_i$ is $(^2)$, and there are n terms in the sum. Thus, the computational complexity is (n^2).

Step 3: Eigendecomposition

From NLA theory, the cost of the eigendecomposition for an $\times$ matrix is $()$. As $\in \mathbb{R}^{\times}$, the cost is $()$.

Total cost = $(n) + (n^2) + ()$.

As such, for bigger and bigger this can be expensive, but this is relatively cheap in n. This indicates that it will be very helpful for investigating lots of points in relatively few dimensions.

Now, from before what we've seen is that we can do PCA with just the SVD of $= V$.

From this, we've seen that $= V^2 V$ from plugging in the definition of through the SVD.

So, do we need to explicitly form through forming ? **NO!**

Why would we prefer one or the other? Well, this is an NLA question. It turns out that forming and doing the eigendecomposition here as opposed to just the SVD is more expensive and introduces floating point errors. In practice, we just form the SVD alone.

Application: MNIST digits

This is going to be the subject of HW 2. Basically all implementation focused on PCA. I can show an example of this.

Last notes on PCA: Error quantification

Now, we can get a little bit more of interpretation on the reconstruction associated with PCA, specifically how *good* the reconstruction of the data through PCA is. Let's assume we have $\{x_i\}_{ni=1} \subset \mathbb{R}$. Let's say its centered.

Fix an orthogonal basis $\{u_i\}_{i=1}$ so that there exist constants $_{ij}$ such that

$$x_i = \quad _{j=1} {}_{ij} u_j$$

Now, if we want to find, for example, a $x_i = {}_{j=1} {}_{ij} u_j$ that lives in an dimensional subspace that "best" approximates x_i, where , we can do this by seeking to minimize

$$E = x_i \;\; x_{i2}$$

Question: Error when ?

Now, we have seen that the representation of x_i on $\{u_i\}_{i=1}$, we know that this is $x_i = x_i$. So, if we truncate and localize to the first principal components associated with , we might consider the projection

$$x_i = x_i$$

where $= \begin{vmatrix} & & \\ x_1 & \dots & x \\ & & \end{vmatrix}$

$$E = 1n \quad {}_{n}{}_{i=1} \quad {}_{j=1} u_j u_j x_i \quad {}_{j=1} u_j u_j x_i$$
$$22$$

$$= 1n \quad {}_{n}{}_{i=1} \quad {}_{j=+1} u_j u_j x_i$$
$$= 1n \quad {}_{n}{}_{i=1} x_i \left({}_{j=+1} u_j u_j \right) \left({}_{=+1} uu \right) x_i$$
$$= 1n \quad {}_{i=1} \quad {}_{j=+1} x_i u_j u_j x_i$$
$$= 1n \quad {}_{i=1} \quad {}_{j=+1} u_j \left(x_i x_i \right) u_j$$
$$= \quad {}_{j=+1} u_j \left(1n \quad {}_{n}{}_{i=1} x_i x_i \right) u_j$$
$$= \quad {}_{j=+1} u_j u_j$$

So! As the variance of the first principal components is ${}_{j=1} u_j u_j$, we ultimately want to find an such that

$$\underbrace{{}_{j=1} u_j u_j}_{\text{iiethis}} + \underbrace{{}_{j=+1} u_j u_j}_{\text{iiiethis}} = \underbrace{{}_{j=1} u_j u_j}_{\text{r()}}$$

Leave the trace expression as an exercise for the problem set.

This ties back into the lagrangian formulation that we investigated earlier. Divide by the trace term and we can use this to show that the percentage of error left is given by the eigenvalues of the covariance matrix.

Lecture 8: Kernel PCA

PCA is a wonderful technique, but it's not ideal for data that isn't well represented by a linear subspace.

Show a demonstration from Nonlinear_PCA_Demo.m

We can see form this example that we need new techniques to represent our data for nonlinearities.

Setup: points $x_1, \ldots, x$ in $\mathbb{R}$.

Let (x) be a non-linear map : $\mathbb{R} \to \mathbb{R}$ for some .

What if we do PCA on $(x_1), (x_2), \ldots, (x)$? Well, this could be expensive for large . Why?

Recall that the total computational complexity of PCA is given by

$$\underbrace{()}_{\text{eteri}} + \underbrace{(^2)}_{\text{iiorietri}} + \underbrace{()}_{\text{eieeositio}}$$

Assumption: ${}_{i=1} (x_i) = 0$

Now, in the same game as before, we can define the covariance matrix

$$= 1 \quad {}_{i=1} (x_i)(x_i)$$

Now, as $\in \mathbb{R}^{\times}$, we have eigenvalues and eigenvectors. These eigenvectors can be computed as $v = v$, and we can use the structural form of to see that

$$1 \quad _{i=1}(x_i)\left[(x_i)v\right] = v$$

We say that v are eigenvectors in the **feature space**

> **Feature Space**
>
> We say that the high dimensional space $\mathbb{R}$ is the **feature space** for kernel PCA

As such, we can see that each v is a linear combination of (x_i) just from the equation above. This can be written in the form, for some constants a_i,

$$v = \quad _{i=1} a_i(x_i)$$

Now, we can plug this term into the above expression and see that

$$1 \quad _{i=1}(x_i)(x_i) \quad _{j=1} a_j(x_j) = \quad _{i=1} a_i(x_i)()$$

Now, we make the following definition:

Kernel Function

We define a function $(,) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ as

$$(x_i, x_j) = (x_i)(x_j)$$

Now, if we return to $()$ and multiply both sides by (x),

$$1 \quad _{i=1}\underbrace{(x)(x_i)}_{(x,x_i)}(x_i) \quad _{j=1} a_j(x_j) = \quad _{i=1}{}_{i=1}a_i\underbrace{(x)(x_i)}_{(x,x_i)}$$

$$1 \quad _{i=1}(x, x_i) \quad _{j=1} a_j(x_i, x_j) = \quad _{i=1} a_i(x, x_i)$$

Now, let's define the kernel matrix $\in \mathbb{R}^{\times}$, where $_{ij} = (x_i, x_j)$. Now, we can write the above expression as

$$1 \quad _{i=1}{}^{i} \quad _{j=1} a_{ji}{}_{ij} = \quad _{i=1} a_{ii}$$

What do each of these terms represent? Well, the RHS can be written as the -th entry of the matrix vector product a for some vector $a \in \mathbb{R}$.

The lefthand side is similar. The first term, $_{j=1} a_{ji}{}_{ij}$ is the i-th entry of a. Then, this is summed over i, so the whole lefthand side is the -th row of multiplied against a. Therefore, for every $\in \{1, \ldots, \}$, we can collect these terms to see that

$$1^2 a = a$$

Now, for the non-zero eigenvectors a of , this directly implies that

$$a = a$$

Notice that the eigenvectors of the kernel matrix are not the same as the eigenvectors of the Covariance matrix . a live in a lower dimensional ($\mathbb{R}$) space than the vectors v ($\mathbb{R}$)

Now, we want normalized eigenvectors, and we want to be able to project our data onto these normalized eigenvectors for PCA.

I can represent $v = \sum_{i=1} a_i(x_i)$ from before. If I want to project a new datapoint (x) onto the direction defined by v, we can see that this is given by

$$(x)v = \sum_{i=1} a_i(x, x_i)$$

which never requires explicit evaluation of $()$, only of $(, x_i)$! This will allow us to do the same PCA-style computations *without ever knowing explicitly!*

The question of centering

One thing that we brushed under the rug is the assumption that $\sum_{i=1} (x_i) = 0$, which might not be the case necessarily. Let us now define

$$(x) = (x) \ 1 \ \frac{1}{} \sum_{i=1} (x_i)$$

The corresponding kernel function is given by evaluating

$$\begin{aligned} (x_i) \ (x_j) &= \left((x_i) \ 1 \ \frac{1}{} \sum_{=1} (x) \right) \left((x_j) \ 1 \ \frac{1}{} \sum_{=1} (x) \right) \\ &= (x_i)(x_j) \ 1 \ \frac{1}{} \sum_{=1} (x_i)(x) \ 1 \ \frac{1}{} \sum_{=1} (x)(x_j) + 1^2 \ \frac{1}{^2} \sum_{,=1} (x, x) \end{aligned}$$

This indexwise expression can be neatly re-represented in matrix language as

$$= \ 111 \ 111 + 1^2 1111$$

where $1 = [1, \ldots, 1] \in \mathbb{R}$.

Lecture 9: Kernel PCA Part 2

Recall some of the details of kernel PCA. First, we start off with points in $\mathbb{R}$ denoted $x_1, \ldots, x$. We showed previously that PCA is bad for *non-linear* data. How can we get around this? We know that we can use the kernel trick to embed data in higher dimensions **and never have to compute things in this high dimensional space**.

We define this embedding map $: \mathbb{R} \to \mathbb{R}$, and assume that the mapped data (x_i) is centered.

From here, we build up the covariance matrix $= \frac{1}{} \sum_{i=1} (x_i)(x_i)$

C has eigenvector/value pairs $v = v$, which is what we need for PCA. We really don't want to explicitly compute this. However, we acknowledge that from the definition of , we have that

$$\frac{1}{} \sum_{i=1} (x_i) \left[(x_i)v \right] = v$$

Once we realize we can write $v = \sum_{i=1} {}_i(x_i)$, we can get around defining the projection onto v through finding what each is.

This is where we introduced the kernel function idea, i.e. $(x_i, x_j) = (x_i)(x_j)$. We saw that from linear algebra

$$=$$

and we solved for and through just an eigendecomposition on .

Then we talked about how to consider this if we need to center our data, and saw that this was easily approachable by taking

$$= \ 111 \ 111 + 1^2 1111$$

Now, how do we project (x) onto v? Well, we just get that

$$(x)v = \ _{i=1}()_i(x, x_i)$$

All we need is the computation of the kernel!

Kernel examples

The most obvious kernel is the linear kernel $(x, y) = xy$. Doesn't buy us much.

One well known kernel is the Gaussian kernel, i.e. $(x, y) = \mathrm{e}\left(_{22}2^2\right)$

Question: When is the kernel method computationally cheaper? When n or n ?

Example: Polynomial kernel

Let's consider the map $\left(\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}\right) = \begin{matrix} u_{21} \\ u_1 u_2 \\ u_2 u_1 \\ u_{22} \end{matrix}$

If we want to compute the kernel of this function, we can do so by computing

$$(u)(v) = u_{21}v_{21} + 2u_1v_1u_2v_2 + u_{22}v_{22} = (u_1v_1 + u_2v_2)^2 = \left(uv\right)^2$$

We never need to visit the high dimensional space to compute the kernel! All we need is right here.

The generalized polynomial kernel can be given by $(x, y) = \left(xy + \right)$. If we take $= 2$ and $= 2$, we can show that for $= 1$ that for

$$(x) = \begin{pmatrix} 1 & 2x_1 & 2x_2 & 2x_1x_2 & x_{21} & x_{22} \end{pmatrix}$$

then

$$(x)(y) = 1 + 2x_1y_1 + 2x_2y_2 + 2x_1x_2y_1y_2 + x_{21}y_{21} + x_{22}y_{22} = (1 + x_1y_1 + x_2y_2)^2$$

Again, **we never had to visit the 6 dimensional space to get this result.**

Example: Gaussian Kernel

We consider now the Gaussian kernel $(x, x) = \mathrm{e}\left(x \;\; x_{22}\right)$. For simplicity, let's just consider the 1D case with $= 1$. What can we make of this kernel, and can we find the original feature map ?

$$\begin{aligned}(x, y) &= \mathrm{e}\left(\left(x^2 \;\; 2xy + y^2\right)\right) \\ &= \mathrm{e}(x^2)\,\mathrm{e}(y^2) \begin{pmatrix} \\ _{=0} 2xy \end{pmatrix}\end{aligned}$$

Notice that this is generated by a feature map that looks like

$$(x) = \begin{pmatrix} \qquad 1 \quad {}_{21}x \quad {}_{2^2 2}x^2 \quad {}_{2}x \quad \ldots \qquad \end{pmatrix}$$

This is a vector in an infinite dimensional space, so the function $() : \mathbb{R} \to$ is an infinite dimensional transform to a Hilbert space (irrelevant detail).

Very nice that instead of computing , we get to use this kernel and only compute the kernelized representations.

Properties of the Kernel matrix

Let's assume that is a finite dimensional map.

Proposition:

Let be the kernel matrix generated by $(x_i, x_j) = (x_i)(x_j)$ for data $\{x_i\}_{ni=1}$

1. is symmetric
2. is positive semidefinite

Proof

To show 1, notice that ${}_{ij} = (x_i)(x_j) = (x_j)(x_i) = {}_{ji}$.

To show 2, we need to see if $yy \;\; 0$ for any y. Let $= \begin{matrix} | & & | \\ (x_1) & \ldots & (x_n) \\ | & & | \end{matrix}$. Then we have that

$$\begin{aligned} yy &= \;\; {}_{ij} y_{ii j} y_j \\ &= \;\; {}_{ij} (x_i) y_i (x_j) y_j \\ &= \;\; {}_{i} (x_i) y_i \;\; {}_{j} (x_j) y_j \\ &= \left(y\right) \left(y\right) \\ &= y_{22}\, {}_{0} \end{aligned}$$

thus completing the proof.

Now, we usually aren't given the kernel, so how can we consider instead constructing conditions on $(,)$?

Mercer's condition

$(,)$ is a valid kernel function iff the kernel matrix is always PSD for data $\{x_i\}$.

Proof sketch:

We can explicitly construct the forward direction, but backwards is less obvious and we won't really touch on this.

If we consider a valid kernel function, then we have

$$= \begin{matrix} (x_1) \\ \\ (x_n) \end{matrix} \begin{matrix} | & & | \\ (x_1) & \dots & (x_n) \\ | & & | \end{matrix}$$

which as $= BB$ for some matrix B, it is PSD.

The other direction, where we have that is PSD, we can find a way to construct from . This is the basis of Reproducing Kernel Hilbert Space theory, which is important in the ML literature.

Limitations of Kernel PCA

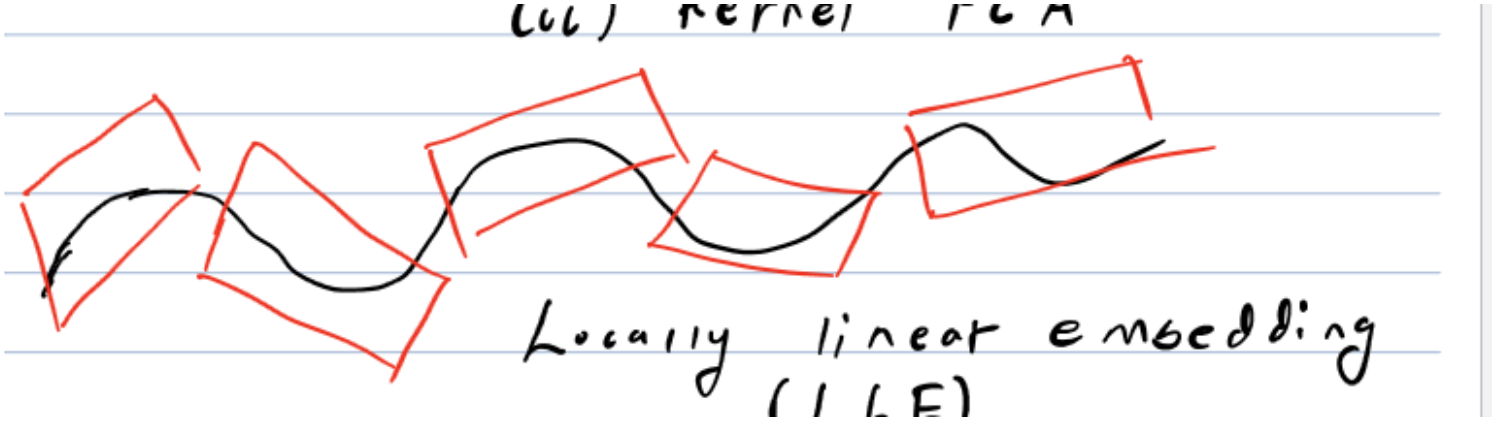
1. Choice of kernel function isn't obvious, and is relatively heuristic.
2. No decoding the pre-image problem in the same way that we were able to do before.

Other topics in dimensionality reduction

We covered

1. PCA
2. Kernel PCA

Another one that we can do is local linear embeddings (LLE)



The issues here is in determining the right neighborhood.

Other techniques include

- Isomap
- tSNE (stochastic neighbor embedding)
- Dictionary learning/sparse coding
- Nonlinear manifold learning

Lecture 10: Random Projections

The goal of this lecture is to find a way to project high dimensional data into a lower dimensional space.

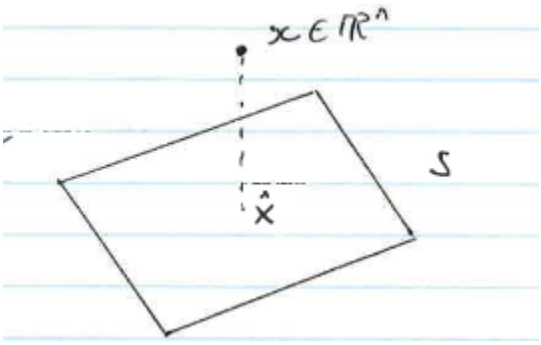
Goal: Retain as much "information" about the data as possible.

Why care about dimensionality reduction?

- Computational complexity
- Simple model/analysis
- Visualization

One idea that we can do is linear projections onto a subspace. How does one project? Well, given a subspace , we can minimize

$$\mathrm{i}_{y\in} x \;\; y_2$$



Projection Formula

1. Find an orthonormal basis $\{u_i\}_{ni=1}$
2. Project

$$\begin{aligned} x &= (u_1 x)u_1 + (u_2 x)u_2 + \; + (u_n x)u_n \\ &= u_1 u_1 x + \; + u_n u_n x \\ &= x \end{aligned}$$

where $= [u_1 \dots u_n]$. We've seen this before.

How do we choose ? With PCA/KPCA, we did this in a data driven/data based way. Sometimes, we might want an agnostic framework, however.

Question: How do we obtain the low dimensional representation in PCA?

Answer: The subspace was spanned by some number of the principal components

Random Projections

First, to define a random projection we define a random vector

Random vector

A vector $x \in \mathbb{R}^n$ is called a random vector of each entry is sampled from some underlying probability distribution.

Ex: $x = (x_1 \quad \dots \quad x_n)$ where $x_i \ (0, 1)$

We say that random variables are identically distributed and identical (i.i.d.) if they're pulled from the same distribution, e.g. you flip a coin 10 times or draw 10 samples from a gaussian distribution.

Take $\{x_i\}_{i=1} \subset \mathbb{R}^n$.
The idea behind a random projection is to take a random matrix $A \in \mathbb{R}^{\times n}$, where n such that we're left with $\{Ax_i\}_{i=1} \subset \mathbb{R}$.

How do these points relate to the original data? Not obvious.

Distance preservation

One thing that we might be able to say is that neighbors in the original space are neighbors in the new, low-dimensional space. This projection should respect neighbors.

Why is this important?

If we choose to cluster in the lower dimensional space, this should be meaningful and correspond to the geometry/topology of the original point cloud.

Johnson Lindenstrauss Lemma

Let $= \{x_1, x_2, \dots, x\} \subset \mathbb{R}^n$. Then there exists a random linear map $: \mathbb{R}^n \to \mathbb{R}$ such that for any points x_i, x_j

$$(1 \ )x_i \ x_{j22} \ (x_i) \ (x_j)_{22} \ (1 +)x_i \ x_{j22}$$

as long as $\ {}^{2}()^{2}$. Notice that this does not depend on n, which can be arbitrarily high dimensional.

We can reinterpret this bound to tell us that the relative error in the measurement is bounded by , as

$$|(x_i) \ (x_j)_2 \ x_i \ x_{j2}|x_i \ x_{j2}$$

Example

One such that works is the map $(x) = {}_1A$, where the entries of A are iid Gaussians (zero mean, unit variance).

Intuition: Random projections behave like orthogonal projections in high dimensions!

We also get norm preservation, i.e.

$$(1 \ )x_{22} \ (x)_{22} \ (1 +)x_{22}$$

and this property is satisfied with high probability, i.e. close to 1. This can be formulated more explicitly as

$$[(1\ )x_{22}\ \ (x)_{22}\ \ (1+)x_{22}]\ \ 1\ \ 2\,\mathrm{e}(^2)$$

Computational complexity of random projections?

Answer: As $A \in \mathbb{R}^{\times n}$ and $\ = [x_1 \dots x] \in \mathbb{R}^{n\times}$, we have that the projection occurs in (n) operations. This works even better if we have *sparse* random projections. This is faster than PCA!

Connection between JL to concentration of measure in high dimensions

High dimensional information gets really weird. We'll see this applied to a high dimensional gaussian random vector. We can define a gaussian vector $x\ \ (0,)$ where $\ $ is the dimension, and $\ $ is the covariance matrix of x (if you haven't seen this before, don't worry: this just means that you have $\ $ i.i.d. entries x_i.)

Now, without proving this statement (and using some properties of Gaussian random variables), we have that

- $E(x_{22}) =$
- $\mathrm{r}(_{22}) = 2.$

We want to compare the fluctuation scale more properly, which is given by $\mathrm{r}(_{22})$. To do this, we can see that by considering

$$\mathrm{r}(_{22})E(_{22}) = 2 \to 0$$

which indicates that there is tight peak around the mean in high dimensions.

This can be encoded in the following theorem

Theorem: Gaussian Annulus

For $\ \ (0,)$ and any $\ \in [0,1]$, we have that

$$\left[|_2\ |\ \right]\ 2\,\mathrm{e}\left(^2\right)$$

In high dimensions, we get **strong concentration effects!**

So, why does this happen and how does this relate?

When considering a random projection $A \in \mathbb{R}^{\times}$, we are, in essence, doing the following:

$Ax_{22} = _{i=1}\, a_i, x^2$, where a_i is a random Gaussian vector with mean 0. So, each a_i, x^2 is mean zero and variance x_{22}. So, by dividing by $_1$, we are *averaging $\ $ independent copies* of a random variable with mean x_{22}.

Since we have an average of many random variables, like we saw in the Gaussian annulus, **we don't get much fluctuation in high dimensions**.

Intuitively, projecting with JL is like taking many noisy measurements of a signal and averaging.

- Each measurement is too noisy
- Taking independent random projections smooths them out
- The resulting average length is almost exactly the true length

Lecture 11: Multidimensional Scaling

First begin lecture with a motivating question: You have a sensor in each corner of the room. We don't know where the sensors are, but they can communicate with each other. When they do this, they can get their pairwise distances. Now, can we get a configuration of these points in space? Let's try to see if we can find out.

We can phrase this mathematically as follows. Let's take points $\{_i\}_{ni=1} \subset \mathbb{R}$, and we can build a matrix

$$= \begin{matrix} 1 \\ \\ n \end{matrix}$$

. Goal is that we want to reconstruct this.

However, what information do we have? We don't know any sort of absolute positions of points here. What we know in this problem is a collection of scalars $\{_i \ _{j2}\}_{n(i,j)=1}$. What can we do with this?

We can appeal to our friend linear algebra. First thing i'm going to do is the following. I know the scalars in the set above, so there's nothing wrong with me squaring these. $\{_i \ _{j22}\}_{n(i,j)=1}$ is now the set that I have. Let $_{ij} = _i \ _{j2}$. Given this information, I want to pack this into a matrix as follows

$$= \begin{matrix} _{211} & _{212} & \cdots \\ _{221} & & \\ & & _{2nn} \end{matrix} = \begin{matrix} 0 & _{212} & \cdots \\ _{221} & & \\ & & 0 \end{matrix}$$

So far, it doesn't really look like i've done all that much. However, we can make a connection between an object intimately related to the points and the matrix .

Let's first define the following object. We say a Gram matrix $\in \mathbb{R}^{n \times n}$ is a symmetric matrix that's of the form $=$. Entrywise, we can see that (do out the matrix multiplication)

$$_{ij} = _{ij}$$

First off, what does this get us? This gets us that is a PSD matrix (sidenote: all PSD matrices are Gram matrices. That is to say, we can get a point decomposition that will give us the desired structure no matter what).

Next question: what's the rank of ? Hint: think about the SVD of and how this might relate to the geometric embedding of the points.

Now, we know that and are related. Can we relate to , in hopes that we can relate this to ?

Let's look entrywise. Notice that

$$
\begin{aligned}
{}_{2ij} &= {}_{i\ j22} \\
&= ({}_{i\ j})({}_{i\ j}) \\
&= {}_{ii\ ij\ ji} + {}_{jj} \\
&= {}_{i22} + {}_{j22\ 2ij} \\
&= {}_{ii} + {}_{jj\ 2ij}
\end{aligned}
$$

Interesting! Let's see what we can maybe do with this, now that we have this relationship. I ultimately want to isolate each entry ${}_{ij}$, as I have access to all the ${}_{2ij}$ but none of the ${}_{ij}$'s directly.

Now is where I'm going to add the simplifying assumption that $1 = 0$, where 1 in this case is a column vector of 1's. What does this mean? Show them on the board that this geometrically means that my point cloud will be centered at the origin. This will matter when we do the next derivation. (Also, what does this mean for $1 =$ Means that it's zero and that summing over the rows/columns is also zero.)

So, we have this assumption now. Let's try to disentangle all these terms and see if I can't isolate something. We're going to do this by considering the means on the distance matrix itself.

First, let's sum across the rows of the pairwise distance matrices:

$$
\begin{aligned}
{}_{i,} &= 1n\ {}_{n}{}_{=1\ 2i} \\
&= 1n\ {}_{n}{}_{=1\ ii} + \ {}_{2i} \\
&= 1n\ {}_{ii} + 1n\ \ {}_{2n\ i} \\
&= {}_{ii} + 1n \\
&= {}_{ii} + 1n\mathrm{r}()
\end{aligned}
$$

By symmetry, we can see that

$$
{}_{2,j} = {}_{jj} + 1n\ \ = {}_{jj} + 1n\mathrm{r}()
$$

Lastly, we can see that (do some of these steps in more detail than in the notes)

$$
\begin{aligned}
{}_{2,} &= 1n^2\ \ {}_{2} \\
&= 1n^2\ \ \ + \ 2 \\
&= 1n\ \ + 1n \\
&= 2n\mathrm{r}()
\end{aligned}
$$

Ok! Why did we do all this? We'll write the equations all up together on the board side by side, and then show that

$$
\begin{aligned}
{}_{2ij}\ \ {}_{2i,}\ \ {}_{2,j} + \ {}_{2,} &= {}_{ii} + {}_{jj\ 2ij\ ii}\ 1n\mathrm{r}\ {}_{jj}\ 1n\mathrm{r}() + 2n\mathrm{r}() \\
&= 2_{ij}
\end{aligned}
$$

We have a linear transformation! We can see from this now that (add some steps in the factorization process, then define the matrix at the end)

$$
\begin{aligned}
_{ij} &= 12\left(_{2ij} \quad _{2i,} \quad _{2,j} + \ _{2,}\right) \\
&= 12\left(_{ij} \ \left[1n11\right]_{ij} \ \left[1n11\right]_{ij} + \left[1n1111\right]_{ij}\right) \\
&= \left(\ 1n11\right) \left(\ 1n11\right) \\
&= 12
\end{aligned}
$$

We note that is a projection matrix onto the orthogonal complement to the subspace spanned by the one's vector.

Great! We've now established a matrix exists and can be computed from . What can we do now?

Well, we know that $=$, right? So why don't we just try to find a matrix that satisfies this?

First thing that we know about is that it has an eigenvalue decomposition. That is, $=$ where is a diagonal matrix. Make the point about how we're going to take the reduced eigenvalue decomposition, and throw away the nonzero stuff just like in the reduced SVD. Side comment that bc it's PSD, the eigenvalue decomposition is identical to the SVD.

Here I can define what it means to take the square root of a matrix. For diagonal matrices, it's just a matter of taking a square root of the diagonal entries (assuming that they're non-negative, important point.)

So now that we have the matrix $^{12} =$ ^{121} , we can see that if I form the matrix $^{12} =$, we get that

(show out the multiplication of the matrices in more detail)

$$= \ ^{12}(^{12}) = \ ^{1212} = \ =$$

Great! We've established that this is a recovered point matrix . Is it unique?

No! I can take any orthogonal matrix $\in \mathbb{R}^{\times}$ and append this to and get $=$!

This makes sense, as I can rotate points and show that this is something that doesn't affect our data at all.

Applications

Protein folding, sensor localization, dimensionality reduction, etc.

Here I should pause and use a script that I build in order to show them that this technique does actually work.

Procrustes Alignment

Let's say that I've recovered through MDS a point cloud , and I want to compare it to the actual ground truth point cloud . We'll assume that the data is centered.

So, we can think about this is equivalent to finding a matrix such that

$$
:= \mathrm{i}
$$

is minimized.

I think deriving this result would be somewhat of a disaster since it relies on defining a trace inner product between matrices. Instead, we just provide the result and then talk about how you can set some number of anchors in alignment to get global alignment of your points.

Now, we define the following matrix $= \in \mathbb{R}^{\times}$ where $= V$ is a full SVD. The minimizer to the problem is then given by the matrix V.

That is to say,

$$\mathrm{ri}_{:=} = V$$

So if we want to align with , all we need to do is compute the SVD of the matrix !

This is more involved if you want to scale points, or shift their centroid. Not impossible however.

So now, let's consider this problem in practice! Let's say that you only have points where the absolute distance is known (draw an anchor configuration on the board). Call this ground truth $\in \mathbb{R}^{\times}$. WLOG, let's say that the whole points are given by $= \begin{bmatrix} & \end{bmatrix}$ where is unknown.

If we know all of the pairwise distances, we can do MDS to compute a point cloud that would've generated those points.

What we can do now is take $= \begin{bmatrix} & \end{bmatrix}$. We can procrustes align the and to get the proper orientation (this will be really good to show in a demo).

Now that we have this, we can take those few points, get , compute an SVD to get the matrix $V \in \mathbb{R}^{\times}$, and now we can align the whole set of points by computing

$$V =$$

where the equality holds if things are computed exactly

Non-$_2$ distances

Sometimes we consider distances that aren't the Euclidean distance. You can still get embeddings of Euclidean points, but we have the following theorem

Theorem:

Let be a squared distance matrix where $_{ij} = (_i, _j)$ for some function . Let $= _{12}$. is a positive semidefinite matrix iff $(_i, _j) = _i \ _{j22}$.

Lecture 12: Compressive Sensing

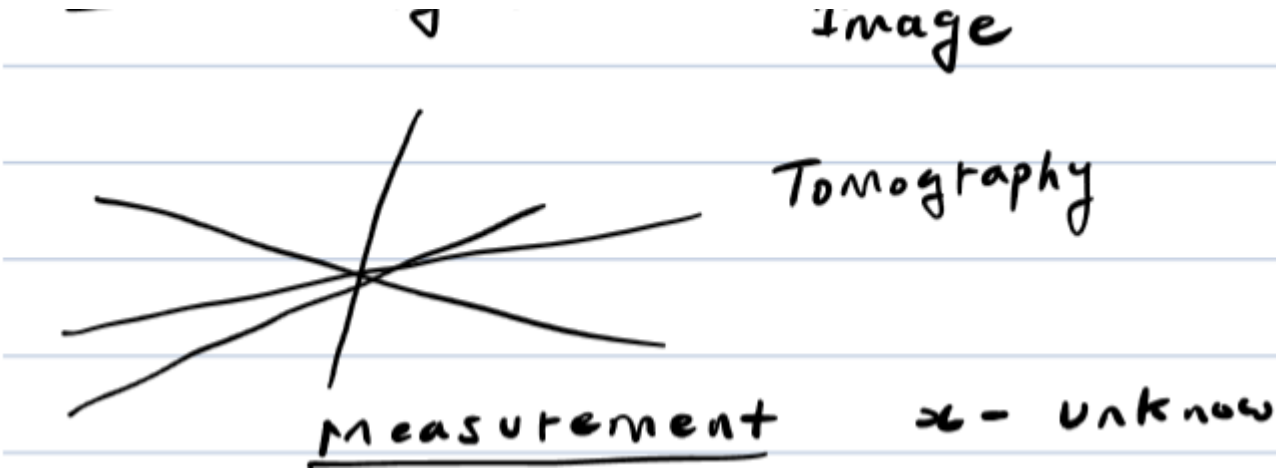
One of the more important signal processing techniques for signal processing has been the technique of compressed/compressive sensing. The main motivation can be summarized as follows:

Let's say that I have a measurement that takes the form

$$y = Ax$$

where $y \in \mathbb{R}$, $A \in \mathbb{R}^{\times n}$, and $x \in \mathbb{R}^n$. This, of course, is just a linear system. The problem is now that we consider the case where n . This system is now *underdetermined*, and cannot be uniquely constructed. At first assumption, we're totally out of luck.

Where's an example of this? One of the examples they we can have this in is called **tomography**, of CT scan fame. This works as follows



We can take a mass, like an organ in the body, and say that it can be modeled by some mathematical density. For simplicity, let's just say it's (x, y). Now, we can shoot x-rays through this. The x rays go through the cell, and can measure the thickness of (x, y) along the ray as the signal gets attenuated.

So, our measurements look something like $y_i = {}_i(x, y)_i$. What I mean by this is the line integral along the ray ${}_i$.

Now, this isn't exactly an easy thing for us to model. So, how can we approach this differently? I'm going to discretize my object (x, y) to create a vector x, where each entry corresponds to a grid element (draw this). Now I can represent my measurements as something that looks more like the length of each ray segment at each grid point (draw out how the integral can just be sort of written as multiplying by some of the coefficients)

So, now my measurement can be represented in the form of $y_i = {}_{nj=1} a_{ij} x_j = a_i x$ for some vector a_i. This puts us in the situation where we can now consider our measurement as a linear system $y = Ax$.

Again, our system is underdetermined if we don't take a bunch of measurements. What we really want to do is only take a few measurements, since this is faster.

This means that I'm looking for a representation of x that is **sparse** with reference to my measurements. I want just a few measurements to capture most of the information that I need.

What does this mean for a given basis $B = \{u_1, \ldots, u_n\}$? Well, as its a basis we have that $x = xu_1u_1 + + xu_nu_n$, and that

$$[x]_B = \begin{matrix} xu_1 \\ \\ xu_n \end{matrix}$$

I want most of these coefficients to be zero. We say that a vector x is -sparse with respect to a given basis B if at most of the above coefficients are non-zero.

Can also be captured as the $_0$ norm

$$x_0 = \text{erooeroeeetso}$$

From here on, we say that x is -sparse, meaning that x_0 .

Some other examples of where this comes about: Fourier bases - periodic signal processing. Wavelet bases for images - (JPEG Compression). Discontinuous data can be well modeled by Haar bases (draw).

Let's return to the original problem at hand. Let's say that I know y, and I know A. I want to solve for x in $y = Ax$. Now, A isn't invertible, so I can't just flip it over. Moreover, this system is underdetermined, so I can't get a unique solution.

What I can do, however, is I can look to solve this as the sparsest possible solution. The idea behind this is that, if I as a domain specific practitioner have done a good job taking my measurements, then it should be the case that x is sparse.

Here's an example of an MRI image using a wavelet basis. Only a few coefficents matter!

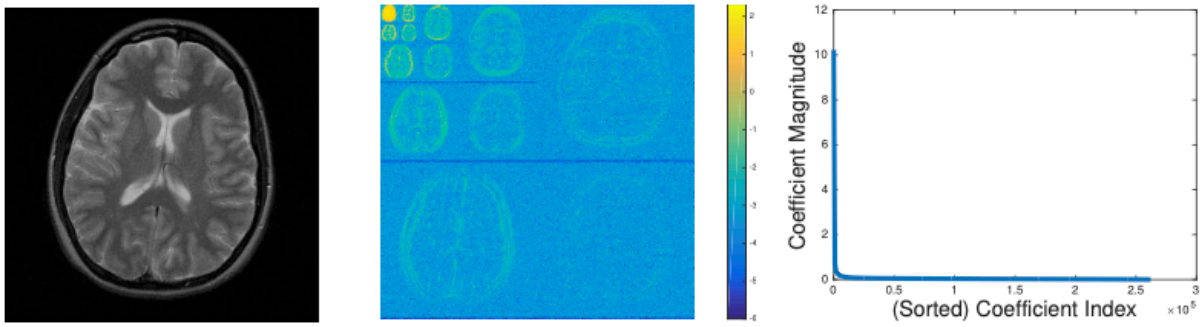


Figure 2.2 A Magnetic Resonance Image. Left: target image of a human brain. **Right:** coefficients in the wavelet decomposition $I = \sum_i \psi_i x_i$, and their magnitudes, sorted in descending order. The large wavelet coefficients concentrate around sharp edges in the image; wavelet coefficients corresponding to smooth regions are much smaller. The wavelet coefficients are highly compressible: their magnitude decays rapidly. Image reprinted with permission from Michael Lustig [Lus13].

Want to make it clear there that x being sparse is the exact same thing as saying y only needs a few pieces of information to be accurately represented! It means that the basis that we have for x built by the rows of A is a **good** (i.e. sparsifying) basis

Let's say that x is the true underlying solution we want to recover from our measurement vector y. So, under the assumption that x is sparse, if we want to solve the problem we can do it by doing something

like

$$\mathrm{i}_{x\in\mathbb{R}^n} x_0 \mathrm{settoy} = Ax$$

If we want to do this, what kinds of problems might we run into?

Well, we can think about solving it like this. Let's say a priori that we know that x is -sparse, for some reason. If I wanted to optimize this and find a **global** optimum, I need to go through all $\binom{}{n}$ collections of coefficients to solve for this! Not good! (Side note: greedy techniques are good at finding local optima). Doing this would take on the order of n operations, and with $=0$, $n=200$, and $=10$ (all considered not that big of a problem), an algorithm that solves this would take over 100 *centuries* to solve on a standard laptop. (Not discussing in detail, but I'll mention here that this is an hard problem)

Here's another issue with this approach. Let's say that $x \in (A)$. Then the optimization program will recover the vector $x=0$! That's no good either!

We can re-frame this problem by considering the fact that "Although the worst sparse recovery problem may be impossible to solve efficiently, perhaps my particular instance (or a subclass of instances) of interest is not so hard."

The beauty of compressive sensing is that this ultimately ends up being quite true.

Now, introduce a geometric notion of the convex function as something with a bowl-like shape

2.3 Relaxing the Sparse Recovery Problem 55

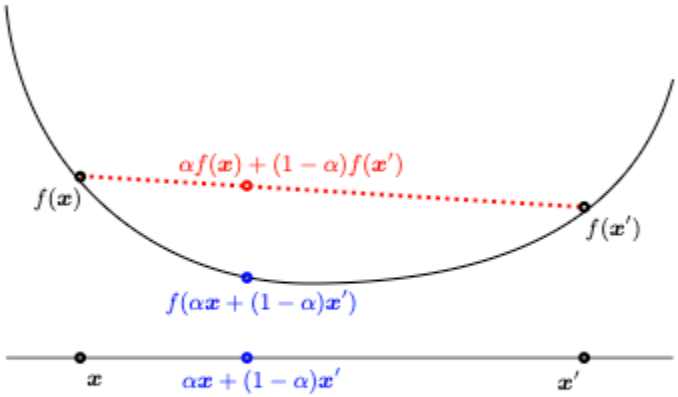


Figure 2.10 Definition of Convexity. A convex function $f : \mathbb{R}^n \to \mathbb{R}$ is one which satisfies the inequality $f(\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{x}') \leq \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{x}')$ for all $\alpha \in [0,1]$ and $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^n$. Geometrically, this means that if we take the points $(\boldsymbol{x}, f(\boldsymbol{x}))$ and $(\boldsymbol{x}', f(\boldsymbol{x}'))$ on the graph of f, and then draw a **line** joining them, the graph of the function falls below this line segment.

The main nice property is that for convex functions , the local minima of (where the derivatives vanish) are also **global** minima.

Next we define the $_1$ norm.

Here I'm going to try and draw a picture of what the unit balls $_0$ and $_1$ norms look like. The $_0$ one is the four dots on the axes $\mathbb{R}^2$, and the $_1$ is the inscribed diamond in it. Make a point about how now $_1$ is a **convex envelope** of the previous function.

So, what happens if instead I try considering

$$\mathrm{i}\underset{x\in\mathbb{R}^n}{\quad} x_1 \mathrm{set to} y = Ax$$

It turns out that this particular approximation and assumption is one of the most important mathematical contributions of the 21st century.

This isn't a foolproof method, but it does work for many problems. It is strongly determined by the size of the support of the sparse vector. Far from foolproof, and recovery is probabilistic (but with a sharp phase transition!)

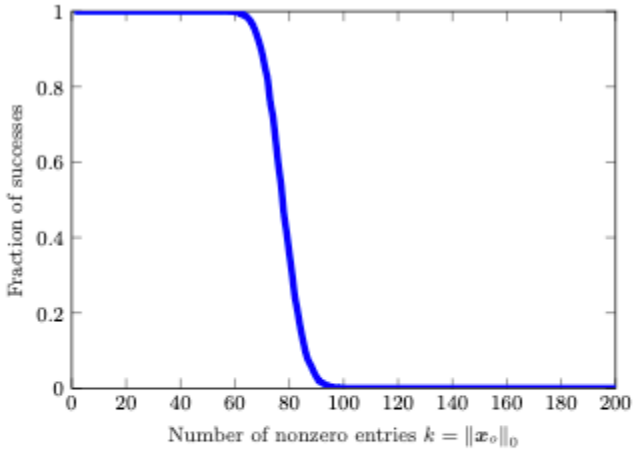


Figure 2.15 Phase Transition in ℓ^1 Minimization. We consider the problem of recovering a sparse vector $\boldsymbol{x}_o$ from measurements $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_o$, where $\boldsymbol{A} \in \mathbb{R}^{100\times 200}$ is a Gaussian matrix. We vary the number of nonzero entries $k = \|\boldsymbol{x}_o\|_0$ across $k = 0, 1, \ldots, 200$, and plot the fraction of instances where ℓ^1 minimization successfully recovers $\boldsymbol{x}_o$, over 50 independent experiments for each value of k. Notice that this probability of success exhibits a (rather sharp) transition from 1 (guaranteed success) to 0 (guaranteed failure) as k increases. Notice moreover, that *for sufficiently well-structured problems (k small), ℓ^1 minimization always succeeds.*

What's the intuition on why this works at all?

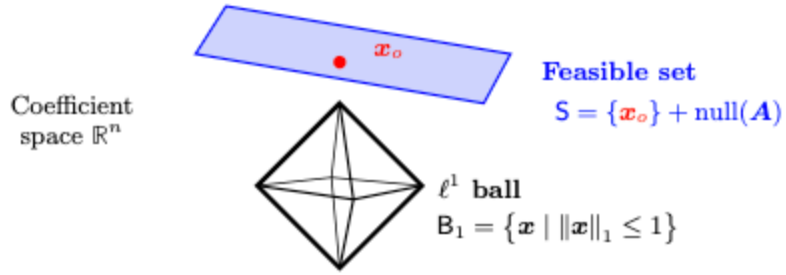


Figure 3.1 Coefficient-Space Picture. The set of all solutions x to the equation $Ax = y$ is an affine subspace S of the coefficient space $\mathbb{R}^n$. The ℓ^1 ball B_1 consists of all coefficient vectors x whose objective function is at most one.

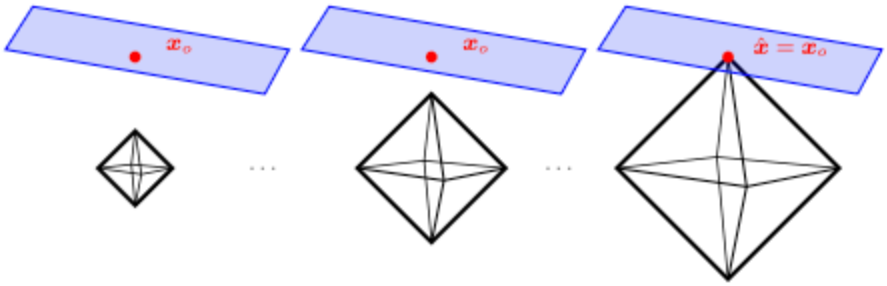


Figure 3.2 ℓ^1 Minimization in the Coefficient-Space Picture. ℓ^1 minimization can be visualized geometrically as follows: we squeeze the ℓ^1 ball down to zero, and then slowly expand it until it first touches the feasible set S. The point (or points) at which it first touches S is the ℓ^1 minimizer $\hat{x}$.

Notice that this won't work for every x, or for every matrix A! The structure of the problem plays a huge role in the effectiveness of this approach!

Lecture 13: Matrix Completion

First, we can consider a brief review of what compressive sensing accomplishes. We can recall that if we sought to solve an underdetermined linear system

$$y = Ax$$

under the right conditions we found that considering the approach

$$\mathrm{i}_{x \in \mathbb{R}^n} x_1 \mathrm{set to} y = Ax$$

that we got good numerical results for a wide class of sparse vectors x. That is to say that many vectors in the *right* basis are **compressible**. They contain less information than its embedding in $\mathbb{R}^n$ might lead you to think from first principles!

Let's consider now a new setting where we have a large degree of redundant information, and we might have a hope of reconstructing an object based on limited information.

We lead by returning to the netflix problem. Suppose we have a matrix of movie ratings, where users correspond to the rows and the columns correspond to movies (draw this out)

Now, of course one hasn't likely seen every movie that's available on netflix. There's missing information in this approach. But it's reasonable to infer (as we did before when I introduced PCA) that there's a good chunk of info that's not necessary. If you liked john wick, you'll probably like john wick 2, but maybe you're less interested in sense and sensibility and vice versa.

Now, let's return to a definition of rank. The rank of a matrix is the amount of linearly independent rows/columns. Let's take the row perspective.

Lets consider the i-th row of , denoted x_i. Let $\in \mathbb{R}^{\times n}$, so $x_i \in \mathbb{R}^n$. Let's now consider a different row x_j, but say that we don't know the 1st entry of x_j.

Now, if we go through entry by entry, and we have that for each entry of x_i and x_j that $(x_i) = 2(x_j)$. Again, we don't know that first entry of x_j.

If I know that $\mathrm{r}() = 1$, however, then I know that every row vector has to be parallel to x_i. In this case, then I know that the first entry of $(x_j)_1 = 12(x_i)_1$, and I've completed my vector.

If I had information that $\mathrm{r}() = 2$, I'd need a little bit more information then! I'd probably need to measure another row in order to determine what the first entry is. To see that you'd need to do this, consider the case that x_i is as before, but now you have $x = (1 \quad 0 \quad \dots \quad 0)$. Then if I'm missing the first entry of x_j, it could be that it could be anything since all I know is that x_j is a linear combination of x_i and x.

However, we're more or less on the right track now. We can now pose our problem.

Let's consider some low-rank matrix $\in \mathbb{R}^{\times n}$, where $\mathrm{r}() = \ \mathrm{i}\{, n\}$. Let $= \{(i, j) | i \in \{1, \dots, \}, j \in \{1, \dots, n\}\}$ (draw out a few entries of what this looks like).
I now want to sample index pairs *uniformly at random*, meaning I'm just going to pull a subset of these out, and say that this random subset is .

Now that I have random pairs of indices, I can pose the matrix completion problem as follows:
from $\{_{ij}\}_{(i,j)\in}$*, can I reconstruct ?*

We're going to take a hint from the compressive sensing lecture that we had before. We **know** that the rank of is small. So, given the observation before, why don't we consider the following constrained optimization problem?

$$\underset{\in \mathbb{R}^{\times n}}{\mathrm{i}} \ \mathrm{r}() \mathrm{setto}_{ij} = _{ij}(i, j) \in$$

Ok, great. How might this work in practice? Well, we run into issues again like we did with the $_0$ norm.

Notice the following: The matrix $= \begin{pmatrix} 1 & 0 \\ 2 & 0 \end{pmatrix}$ is rank 1. Let 0 be any real number. Then the matrix $= \begin{pmatrix} 1 & \\ 2 & 0 \end{pmatrix}$ is rank 2! But we also know that $=$ ... So matrices that are arbitrarily close to each other can have different rank! This means that rank is a poorly behaved, *non-smooth* object. To do this rank minimization problem here would be akin to doing the $_0$ problem with the compressive sensing approach, and we saw that was computationally infeasible.

So, what do we do? Well, recall a different definition of rank. For $= _{\mathrm{i}\{,n\}i=1}{}_i u_i v_i$, we have that $\mathrm{r}() = \mathrm{ooerosires}$. This kind of looks like the ${}_0$ norm, doesn't it? Maybe we'll find some success if we consider a *convex relaxation* of the rank.

So by direct analogy to before, we can define the following:

$$= \mathrm{i}\{,n\} \quad {}_{i=1}{}^i$$

This is a real matrix norm, and is called the **nuclear norm** of . Now, notice that I didn't take the absolute values of ${}_i$. That's because ${}_i$ is all positive. How is this related to other norms? Recall that the frobenius norm was $= {}_{i=1}\,{}_{2i}$. This is exactly the ${}_2$ norm on the singular values, and so this nuclear norm is *exactly* the ${}_1$ norm on the singular values! We are on the right track!

Here we can define the new, actual objective function

$$\underset{\in\mathbb{R}^{\times n}}{\mathrm{i}} \mathrm{setto}_{j} = {}_{ij}(i,j) \in$$

Solving this is something that requires a good amount of care algorithmically. However, we again just have a convex function, so it has a unique local minimum! How do we know when we have enough information, however. That is, how do we know when $||$ is large enough to have the algorithm reconstruct the right solution?

Theorem:

Let $\in \mathbb{R}^{\times n}$, and without loss of generality let n . Let $= \mathrm{r}()$, and assume n. Let $\subset \{(i,j)|i \in \{1,\ldots,\},j \in \{1,\ldots,n\}\}$ be sampled uniformly with replacement. If

$$|| \; n\,\mathrm{o}^2(n)$$

for some matrix specific constant , then

$$\underset{\in\mathbb{R}^{\times n}}{\mathrm{i}} \mathrm{setto}_{j} = {}_{ij}(i,j) \in$$

recovers the ground truth matrix with high probability (very very close to 1).

I mentioned that is a matrix specific constant however, and a lot of information is encoded in this. In particular, there is a constant that measures how "diffuse" the entrywise information in a matrix is.

Here's an example of when matrices aren't all that recoverable. Let

$$= \begin{matrix} 1 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ & & & \\ 0 & \ldots & & 0 \end{matrix}$$

Notice that this is a rank 1 matrix. But if I look to sample every entry from this matrix, i'm probably going to need to sample everything in order to pick up the only entry that matters! Most entries contain zero information, and one entry contains **all** the information.

Contrast this with the matrix

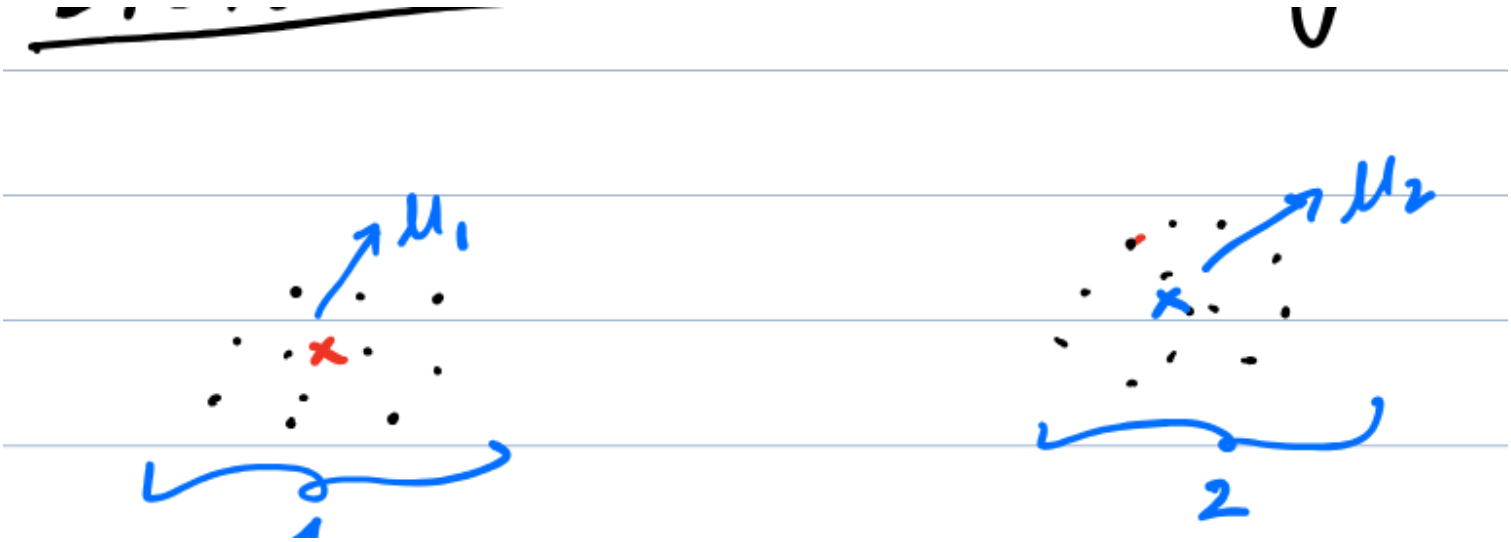
$$= \begin{matrix} 1 & \dots & 1 \\ 2 & \dots & 2 \\ & & \\ n & \dots & n \end{matrix}$$

Here we have a rank 1 matrix, and we could fully determine this matrix by sampling one row and then one entry from each other row!

Let's show an example of this actually working in practice now.

Lecture 14: K-Means Clustering

We're going to be considering data $x_1, \dots, x_n \in \mathbb{R}^n$. Now, if I have a bunch of data points, I might want to classify the data points in space by their location and cluster points that are close to one another.



An example of this is when we saw in PCA, easily separable numbers neatly formed classes. If we picked a data point from here, we'd want to figure out which class it belonged to by clustering our datapoints.

So, how do we go about computing this clustering? Let's assume that we have clusters denoted $_1, \dots, $. We want to assign points x_i to clusters $_1, \dots, $. We can define the function

$$(_1, \dots,) = \quad _{i=1\ x_i \in} x_i \ \ _{i22}$$

Question: What's the minimum value of if $= n$?

Each point would be its own cluster! So $= 0$.

Lets think about how to actually use that in practice... One thing we can consider in clustering is assuming that we know the centers $_i$ and these are fixed. How can we get an optimal partition?
Answer: We just assign each point x_j to the closest centroid $_i$. If there's a tie, break it arbitrarily.

Now, right now we're presuming that we know the clusters a priori. What happens if we know the clusters? We can compute the centroids $_i = \frac{1}{|i|} _{x_j \in _i} x_j$

First, we should determine whether or not the exact centroids are the best solutions to the problem of centroid assignments.

Lemma: Let $\{a_1, \ldots, a_n\}$ be a set of n points. Let x be an arbitrary point. Then $_{ni=1}a_i \ x_{22} = _{ni=1}a_i \ _{22} + n \ x_{22}$ where $= 1n \ _i \ a_i$.

Proof:

We start off by recognizing that

$$\begin{aligned} _i a_i \ x_{22} = & \ _i a_i \ + \ x_{22} \\ = & \ _i a_i \ _{22} + 2(\ x) \ \ _i(a_i \ ) + \ _i \ x_{22} \\ = & \ _i a_i \ _{22} + 2(\ x) \ \ _i(a_i \ ) + n \ x_{22} \end{aligned}$$

Now as is the centroid of $a_1, \ldots, a_n$, I have that

$$\begin{aligned} _i a_i \ = & \ _i \left(a_i \ 1n \ _j a_j \right) \\ = & \ _i a_i \ nn \ _j a_j \\ = & n \ n \\ = & 0 \end{aligned}$$

From here the lemma statement follows.

A corollary is that the point x that minimizes the distance to each a_i is the centroid! Matches intuition, but also important for our problem.

So, we know how to handle this problem of assignment now if we either know the clusters, or if we know the centroids.

How can we maybe deal with this in the case that I don't know the clusters or the centroids exactly? We have something known as Lloyd's algorithm for assignments.

Lloyd's algorithm

Initialize: pick random centers $_1, \ldots,$

Assignment: Set $y_i = \mathrm{ri}_j x_i \ _{j2}$

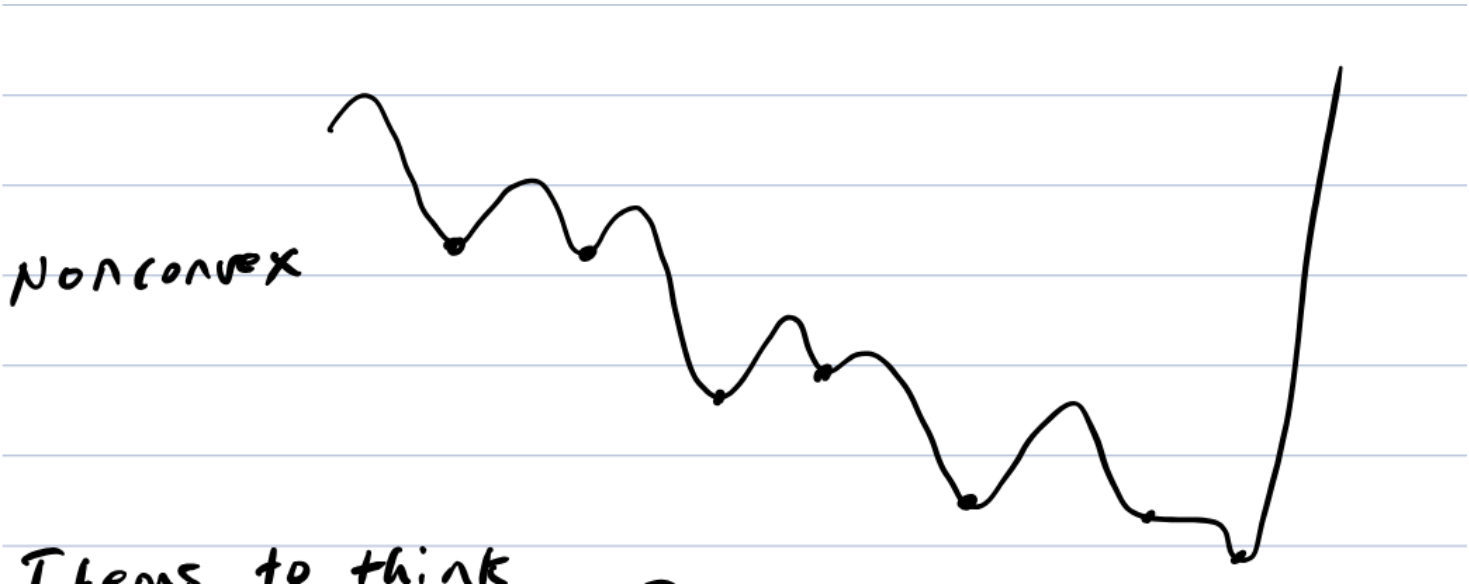
This places a datapoint x_i into a cluster $_j$ by doing this minimization

Update centers

Set $_j = 1|_j| \ _{x_i \in _j} x_i$

This is great! How well does this work in practice?

Well, it's a very non-convex algorithm. What does that mean?

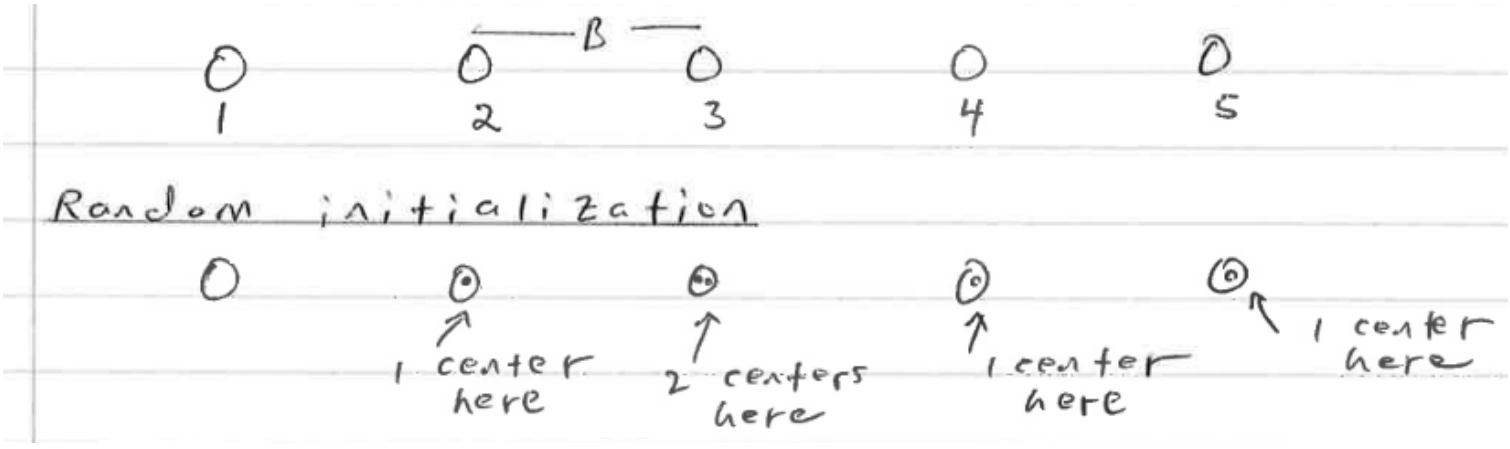


Any sort of optimization on this object is going to be challenging (think of a ball rolling down a hill gets stuck in the local minima)

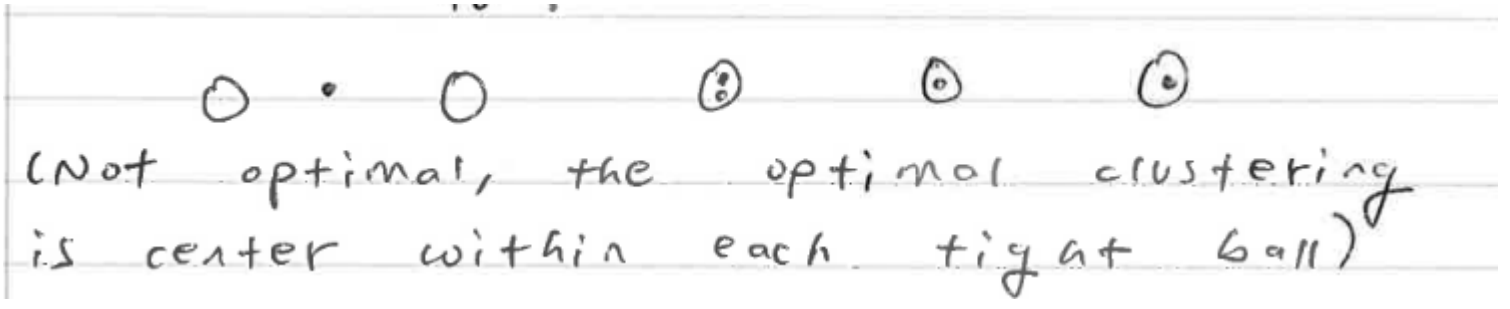
So...

1. How do we initialize? If we initialize close to the solution, we might find the optimal solution
2. Do we always find an optimal clustering? (No)
3. Convergence of the algorithm
4. What kind of data is this well suited for?

Let's consider an example where we have n data points in 5 tight clusters with radius each separated by a distance B. Picture looks like



Where do these points converge to? Well, we can show that Lloyd's k-means algorithm converges to something that looks more like



Which is not the optimal solution!

Let's think about some other algorithmic ways to do -means.

K means ++

We're staying in the same setting of considering $= \{x_i\}_{ni=1}$ as your datapoints. Now, we choose the centers as follows:

1. Choose $_1$ uniformly at random from .
2. For each point $x \in$ not yet chosen, compute $(x) = \mathrm{i}_{\in} x\ ^2$ where is the set of centers
3. Choose a new point as a center $_2$ with probability that a point $x \in$ is chosen equal to $(x\mathrm{seete}) = {}_{(x)_{y\in}}(y)$. This enforces that the next cluster is very dissimilar to the first cluster
4. Repeat until all centers have been chosen
5. Use standard means clustering such as Lloyd's algorithm

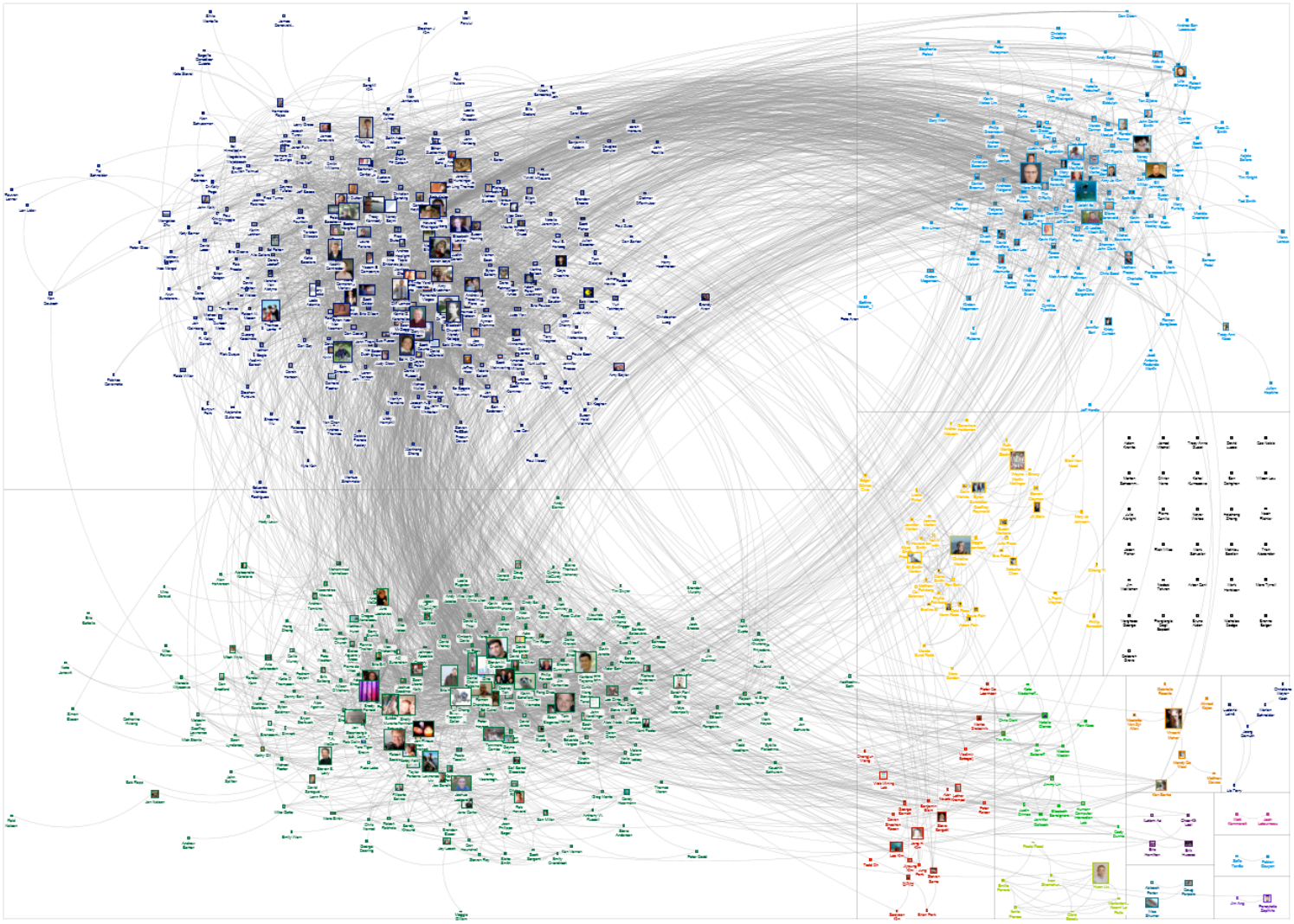
Practically, this allows for faster convergence of Lloyds, you also usually get better cluster quality than just pure random seeding. Very frequently used technique!

Draw on the board about how this would work...

Mention elbow method

Lecture 15: Intro to Graph Theory and Spectral Clustering

Draw a picture of the following image



Motivate it by considering mutual follows on instagram or twitter. This is in the context of communities. Strengthen this analogy by considering this connections to other universities in the area and the people who follow each other there.

Now, how might we represent this mathematically? What can we do to build out a representation of the local communities that might exist inside data that looks like this? We will use a tool called graphs. Graphs are mathematical structures that model pairwise relationships between objects.

Graph definition

A graph is an ordered pair $\mathcal{G} = (V, E)$ where

- V is a nonempty finite set whose elements are called **vertices** or **nodes**
- E is a set of **edges** where each edge is an unordered (or ordered) pair of vertices depending on whether the graph is **undirected** or **directed** (known as a digraph), respectively

In an undirected graph,

$$E \subset \{\{u, v\} : u, v \in V, u \neq v\}$$

In a directed graph,

$$E \subset V \times V$$

so the edges are ordered pairs. The point here is that there's a sense of directionality.

Comment that a graph without self loops is a **simple** graph. We only consider these from here on out.

Draw a couple of simple examples of undirected graphs or directed graphs on the board. Ask how this relates to the problem that I summarized at the start of class.

Weighted Graph

Now, sometimes we want to consider a bit more information than this in this scenario. One way that we can do this is by adding a **weight** to the graph. Draw an example here. These could appear in a context of airlines, where the vertices are airports, edges are flights, and the weight of the edges could be the costs.

Now we can consider a weight function $w : E \to \mathbb{R}$ where now we have a graph $\mathcal{G} = (V, E, w)$.

Graphs as matrices

Now, in the case of any finite graph, we can consider a matrix representation of this graph. We define the adjacency matrix of a graph $\mathcal{G} = (V, E)$ where $|V| = n$ with vertices $V = \{v_1, \ldots, v_n\}$. Then the adjacency matrix of $\mathcal{G}$ is the matrix

$$A = [a_{ij}]$$

where $a_{ij} = \begin{cases} 1 & \{v_i, v_j\} \in E \\ 0 & \text{otherwise} \end{cases}$ for a simple undirected graph. This can be changed to a (v_i, v_j) in the case of a digraph.

If $\mathcal{G}$ is weighted, its defined similarly but now $a_{ij} = \begin{cases} w(v_i, v_j) & \{v_i, v_j\} \in E \\ 0 & \text{otherwise} \end{cases}$

Give an example on the board for a 3 vertex graph.

Next thing I want to define is a function on a graph. A function $: V \to \mathbb{R}$ takes in vertices and outputs a scalar value. For a finite graph, there are only finitely many vertices, so this can actually be represented as a vector $u = \begin{matrix} (v_1) \\ \\ (v_n) \end{matrix}$. So functions on graphs are basically just vectors and vice versa.

One more matrix to define: we define a diagonal matrix $$ as the **degree matrix**, defined as $_{ii} = \mathrm{e}(v_i) = \sum_{nj=1} a_{ij}$ and $_{ij} = 0$ for $i \neq j$.

Using this, we can define the following object known as the graph Laplacian $= \quad A$.

We will use this graph laplacian for some useful things, but lets first actually go over what this means (and why it has such a suggestive name)

A Laplacian, for those who don't know, usually denoted $= {}^2x_{21} + \ + {}^2x_{2n}$ in n dimensions, where each of the x_i are the coordinates. In , it's just x y and .

Let's consider the 1 case where $= {}^2x^2$. Now, on a 1 uniform grid with spacing , we can set the edges between neighbors with weights $w_{i,i1} = 1^2$. Draw this out.

Now, if we then define our Graph laplacian out, it looks like

$$= \begin{matrix} 1 & 1 & 0 & \dots & & 0 \\ 1 & 2 & 1 & 0 & \dots & \\ & & & & & \\ 0 & \dots & & 0 & 1 & 1 \end{matrix}$$

Make the point that what we really want to consider is stuff away from the boundary just to not deal with the weird stuff. Just focusing on the interior.

Now, if functions are just vectors, what happens when I consider u for some vector $u \in \mathbb{R}^n$?

Now, picking i away from the boundary (draw this out in more detail)

$$(u)_i = 2u_i \;\; u_{i+1} \;\; {u_{i1}}^2$$

In the limit $\to 0$, this is just exactly the centered difference definition for the second derivative, i.e.

$${}^2x^2 = \mathrm{i}_{\to 0} (x +) \;\; 2(x) + (x \;\;)^2$$

Which is literally just the negative of what we have here!

Ok, so now we can interpret this graph laplacian structure a little bit. We can think of this object as being a "second derivative" on our graph. What can we do with it?

Let's now interpret this a little bit more. Let $= w$ for some vectors and w. Now, notice that

$$w(i) = (_i) = \mathrm{e}(v_i)_i \quad {}_{\substack{j \\ (i,j)\in E}} (j) = {}_{j:(i,j)\in E} (i) \;\; (j)$$

With this in mind, let's compute .

$$\begin{aligned} &= \quad {}_i (i) \; {}_{j:\{i,j\}\in E} (i) \;\; (j) \\ &= {}_{\{i,j\}\in E} (i) \, ((i) \;\; (j)) \\ &= {}_{ij:\{i,j\}\in E} (i) \, [(i) \;\; (j)] + (j)[(j) \;\; (i)] \\ &= {}_{ij:\{i,j\}\in E} [(i) \;\; (j)]^2 \end{aligned}$$

where the third line follows from the fact that the graph is simple and the original sum was double counting since we were going over all i and j.

I can draw a simple graph and give an example of how to evaluate something that looks like

Eigenvalues and eigenvectors of the graph Laplacian

If $\mathcal{G}$ is an undirected graph, we know automatically that is a symmetric matrix. This tells us that

1. All its eigenvalues are real
2. All its eigenvectors form an orthogonal basis to $\mathbb{R}^n$

Now, what else can we say? We know that 0. Why?

This also gives us that all our eigenvalues are nonnegative.

Lemma: the graph Laplacian always has at least one zero eigenvalue

Proof: Let $v = (1 \quad \dots \quad 1)$, and consider the formula for vv.

Theorem: The multiplicity (read: number) of zero eigenvalues of the graph Laplacian equals the number of connected components of $\mathcal{G}$

Proof: Assume that $\mathcal{G}$ has connected components. Partition V as follows into $V_1, \dots, V$. We can define vectors as follows

$$_i(j) = \begin{cases} 1 & j \in V_i \\ 0 & \text{ese} \end{cases}$$

Without loss of generality, we can consider this in a block structure as follows:

$$_i = \begin{matrix} _1 & 0 & \dots & \\ 0 & _2 & & \\ & & & \end{matrix} \begin{matrix} 0 \\ 1 \\ \\ 1 \\ \\ 0 \end{matrix} = 0$$

Repeat this for all other i and we've shown that there are at least as many zero eigenvalues as there are connected components.

Furthermore, one can show directly that $_{ij} = |V_i|_{ij}$, so we're getting orthogonal vectors and not double counting.

Next, let's assume that there exists a $_{+1}$ $\text{s}\{_1, \dots, \}$ that also has a zero eigenvalue.

This means that $_{+1+1} = _{ij:\{i,j\}\in E} \left(_{+1}(i) \ _{+1}(j)\right)^2$

This requires $_{+1}$ to be a constant value in its entries.

Now as $_{+1}$ is nonzero on some index i since it cant be the zero vector, we can pick the component where $i \in V_j$ for whichever one of those vertices $_{+1}(i) \neq 0$. Now as $_{+1}$ is constant on a subset of V_j, it follows directly that $_{+1j} \neq 0$, so it's not orthogonal to $_j$ and is thus not another eigenvector.

Ok, so what can the other eigenvectors and eigenvalues tell us? First thing is we can recall from courant fischer that we can represent

$$_i = _{xv_1,\ldots,v_{i1}} xAxxx$$

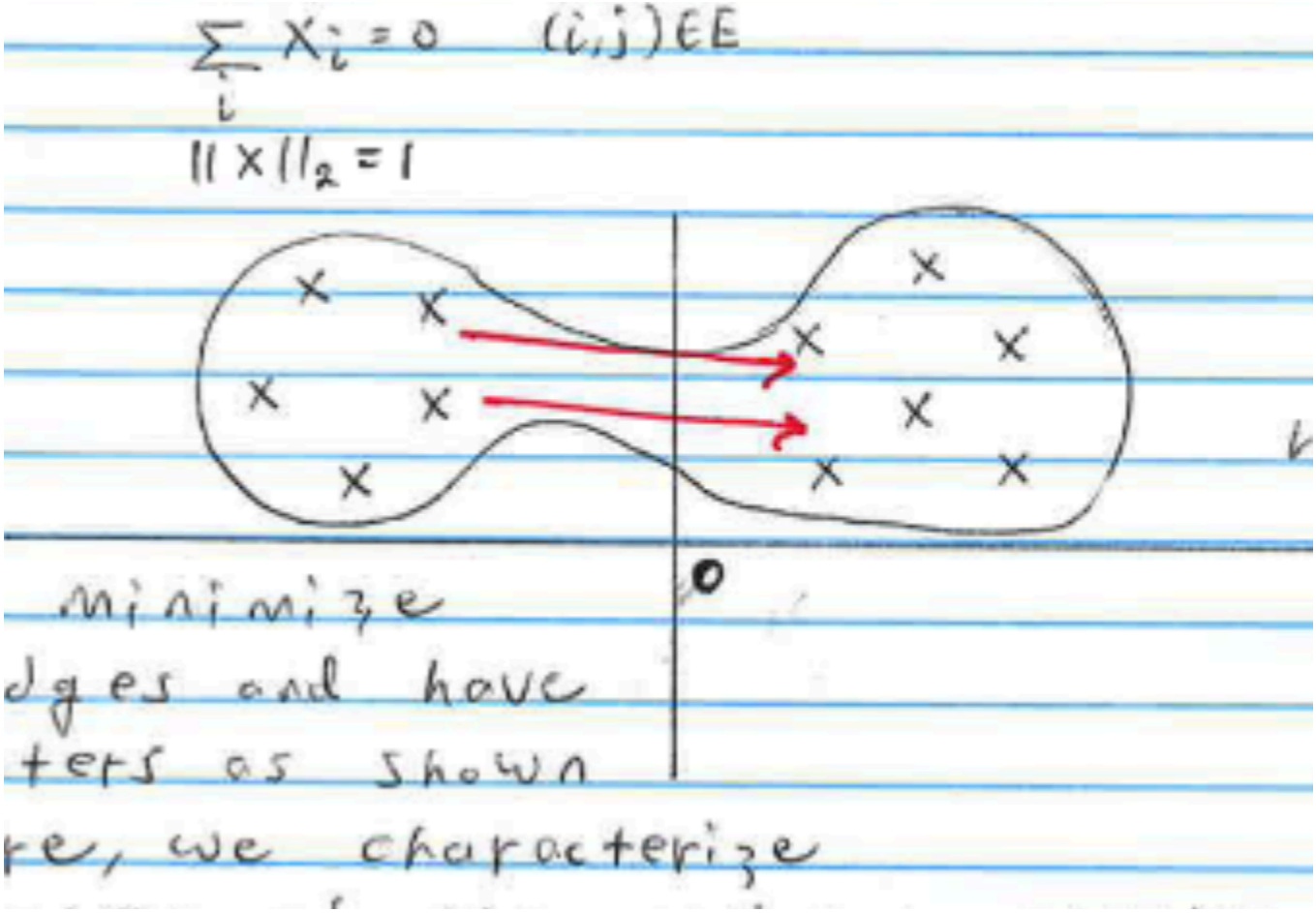
For the time being, let's assume that we have one connected component. This means that we only have one zero eigenvalue.

Let's compute the second eigenvector of the Laplacian. This is given by

$$_2 = \mathrm{i}_{x\neq 0, x1\ ij:\{i,j\}\in E} \left(x_i\ \ x_j\right)^2{}_{i\in V}\, x_{2i}$$

This eigenvalue measures the "distance" between adjacent points with a goal of this being small, i.e. small variability for things that are "close" in graph space (i.e. adjacent).

Let's think of graphs that have two densely connected regions that are connected by a sparse region.



Heuristically, if we decrease the edges connecting the two clusters, we're going to get closer and closer to having two connected components (i.e. we'll have two zero eigenvalues). In this sense, the smallest non-zero eigenvalue and its proximity to zero is a measure of how **close** one is to being separated. Captures this notion of separated clustering.

We can get a stronger result than this heuristic, however, and we can talk about approximating an "optimal cut"

Let $A \subset V$ in a graph $\mathcal{G}$. Consider the indicator vector

$$_i = \begin{cases} +1 & i \in A \\ 1 & i \ A \end{cases}$$

We say that the vector is the optimal cut if

$$\mathrm{ri}_{\in\{1,1\}^n} \sum_{ij:\{i,j\}\in E} (_i \ _j)^2$$

We note that this measures the smallest amount of connection between two clusters (of course, we can get smaller values for this functional if we relax our restriction on , but this captures our desired intuition)

This problem is hard to solve! NP hard in fact. We're going to add some assumptions to make this solvable. Let's assume that we want an equal partition in our optimal cut, i.e. # of $+1$ nodes is equal to the number of 1 nodes. Now, this condition implies that $\sum_{ni=1} {}_i = 0$.

If we relax this, however, to any function we can see that we're really looking at something like

$$\min_{\in\mathbb{R}^n} \sum_{ij:\{i,j\}\in E} (_i \ _j)^2 =$$

Now, this could pretty clearly be zero, because that would be a constant vector. Now, using the fact that we're considering $_i = 0$, this is equivalent to 1. Let's also force that $= n$, just to avoid the case where we're dealing with $= 0$.

Putting this all together, we can see that what we're actually just dealing with is a problem that looks like

$$\min_{\in\mathbb{R}^n, 1, =n} = n_2$$

from courant fischer! So by relaxing this problem, we can see that the second smallest eigenvalue corresponds to a problem where we can best approximate this "optimal cut"

Now, if we want to connect this to the original clustering problem, we can just take categorizing each cluster as $y = \mathrm{si}()$.

Show a computational example here. Build a stochastic block model, and include a figure where we consider just the sign of the points and one where we include the values. We can connect this to the eigenfunctions of $^2x^2$

There's one last connection that we can make. We know that the eigenfunctions of $^2x^2$ are $\mathrm{si}(x)$ and $\mathrm{os}(x)$.

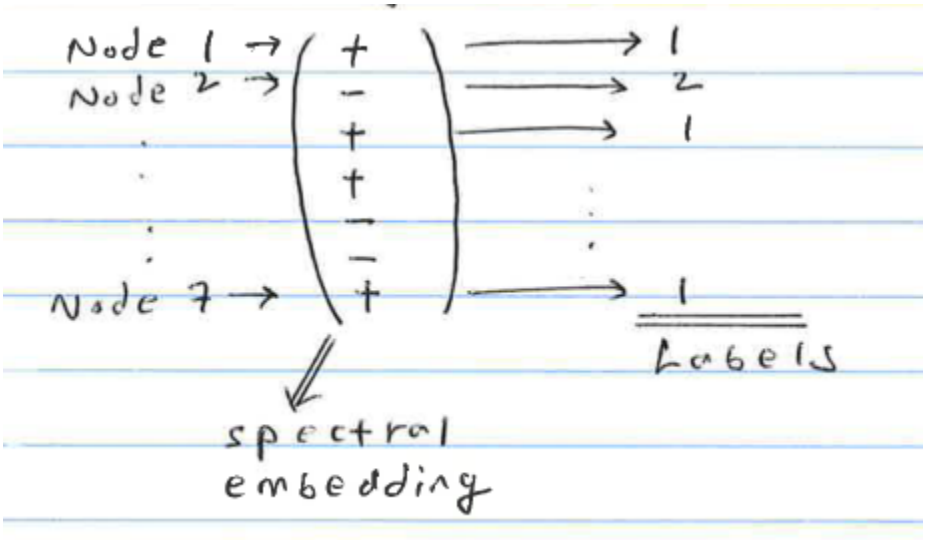
If we consider a bounded domain (line segment), we also know that we can show that this is something that just directly corresponds to increasing frequencies (draw a picture)

Now, what happens in the graph laplacian picture? It looks like we have periodic functions... so we can directly connect these two ideas and see that the eigenfunctions of the graph laplacian actually act as harmonics on a graph.

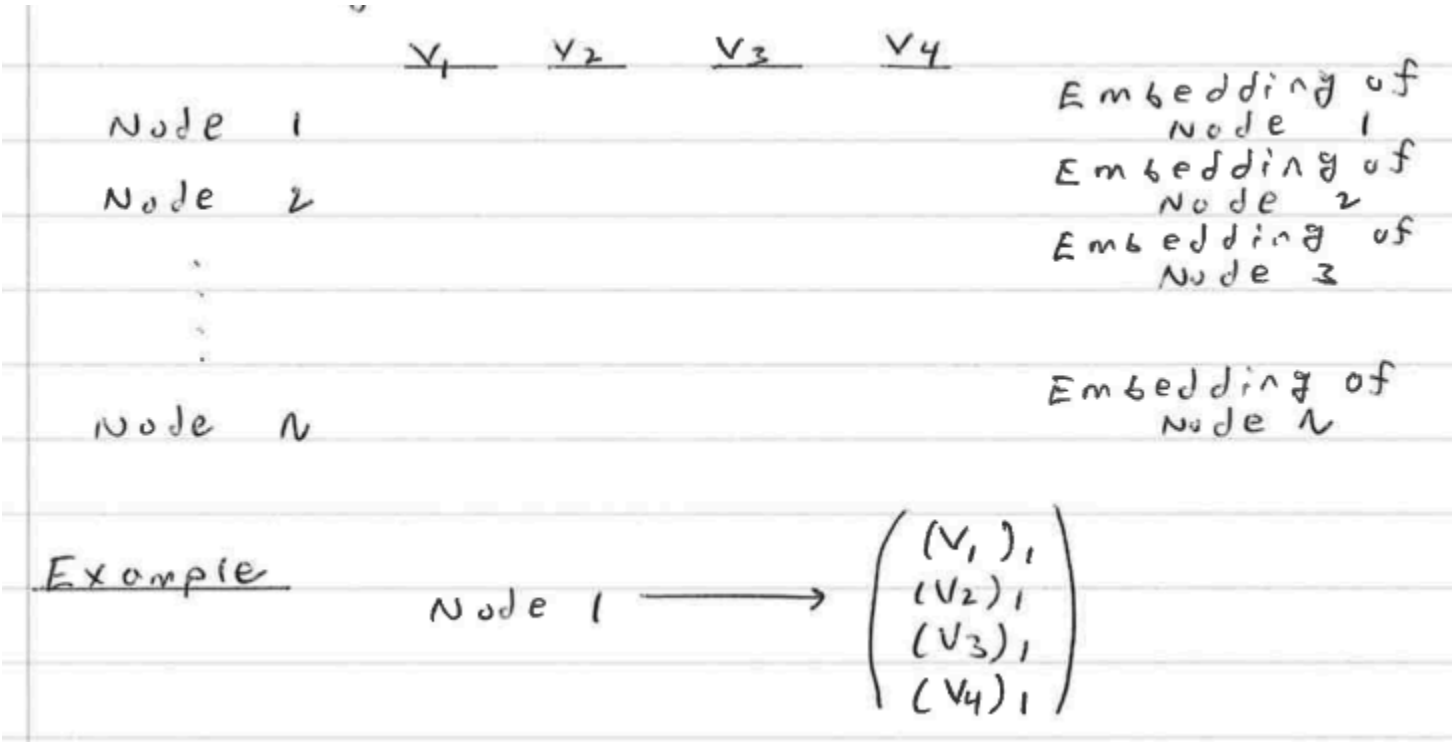
Lecture 16: Spectral Clustering Extended

Ok, so we've now seen that we can approximate the optimal graph partitioning through the Fiedler vector.

Let's try doing some clustering of data based on this. Let's assume that $\mathcal{G}$ has one connected component. We can cluster by assigning a label based on the sign of



We don't just have to use this first vector as well. We can do other vectors and see how well this works!



If we take the function value of the vectors, now we have a new representation of our data, where each point in our graph is assigned a vector. At this point, we can run means clustering on the embedding, giving us our **spectral clustering**.

Run the demo, vary the connectivity between the clusters, show some things about the label classification.

Ok, now clearly this in an example where I know the geometry/topology of my graph... All of my examples were specifically the stochastic block model. What happens now if I instead just consider a set of data $x_1, \ldots, x_n$? Now how can I go about approaching this?

Well, first thing that I need to consider is building a graph structure of some sort. First idea:

1. Build an neighborhood graph where I connect pairwise distances that are smaller than .
   Where this might run into issues: scaling! Sometimes things might be close, and sometimes things might be far. Draw a picture of heterogeneous data with isolated clusters, and really tight clusters.

Alright, let's think about the second approach: a nearest neighbor graph. Connect x_i to the closest points x_j, break ties arbitrarily.

Issue: This adjacency matrix might not end up being symmetric! Or, we violate a hard and fast parameter. Either way, this is trouble.

How can we solve this? We instead consider a fully connected *weighted* graph where we modulate the weight by some kernel.

Example: $w_{ij} = \mathrm{e}\left(x_i x_{j_{22}} 2^2\right)$

Now, we redefine our laplacian to be $=$! Now the degree matrix depends on the degree, which is now $_{ji,\{i,j\}\in E} w_{ij} = \mathrm{e}(v_i)$

We get all the same observations as before, it's just that now my laplacian has the following quadratic form:

$$xx = _{ij,\{i,j\}\in E} w_{ij}(x_i \ x_j)^2$$

Spectral Clustering Algorithm

Input: $\{x_i\}$.

1. Compute
2. Compute $=$
3. Compute the first eigenvectors of , $v_1, \ldots, v$ and set $V = [v_1 \ldots v] \in \mathbb{R}^{n\times}$
4. Let y_i be the ith row of V
5. Cluster $\{y_i\}$ using means, and the label of y_i is the label of x_i.

Lecture 17: Least Squares Regression and LASSO, Logistic Regression?

Lecture 18: Linear Support Vector Machines

Lecture 19: Kernel SVM