Lecture 2: Principal Components and Eigenfaces

Mark Hasegawa-Johnson

ECE 417: Multimedia Signal Processing, Fall 2020

Outline

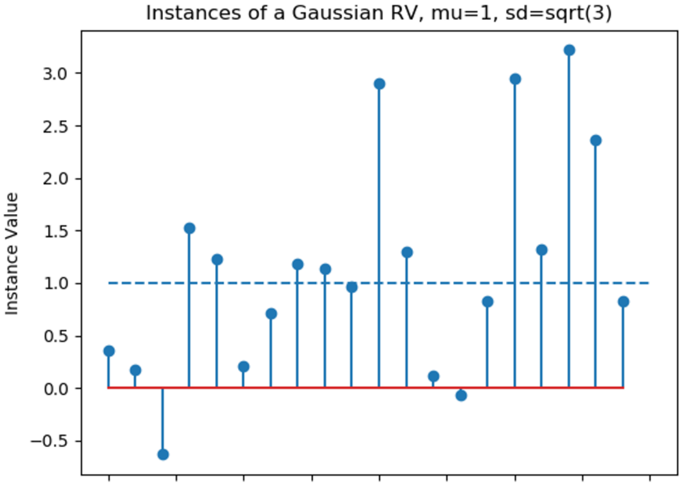
Outline of today's lecture

1. **MP 1**
2. Review: Gaussians and Eigenvectors
3. Eigenvectors of a symmetric matrix
4. Images as signals
5. Principal components = eigenfaces
6. How to make it work: Gram matrix and SVD

Outline

Scalar Gaussian random variables

$$\mu = E[X], \quad \sigma^2 = E[(X-\mu)^2]$$

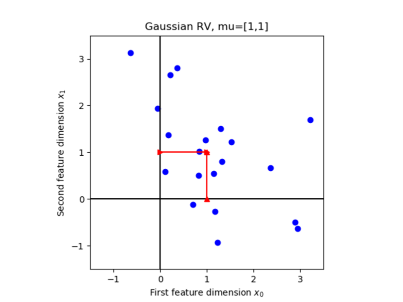


Gaussian random vector

$$\vec{x} = \begin{bmatrix} x_0 \\ \cdots \\ x_{D-1} \end{bmatrix}$$

$$\vec{\mu} = E[\vec{x}] = \begin{bmatrix} \mu_0 \\ \cdots \\ \mu_{D-1} \end{bmatrix}$$

Example: Instances of a Gaussian random vector



Gaussian random vector

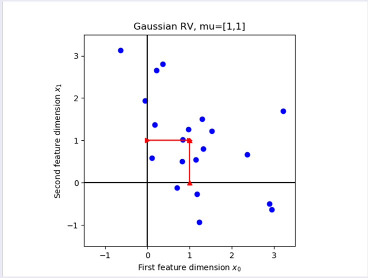
$$\Sigma = \begin{bmatrix} \sigma_0^2 & \rho_{01} & \ddots \\ \rho_{10} & \ddots & \rho_{D-2,D-1} \\ \ddots & \rho_{D-1,D-2} & \sigma_{D-1}^2 \end{bmatrix}$$

where

$$\rho_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)]$$

$$\sigma_i^2 = E[(x_i - \mu_i)^2]$$

Example: Instances of a Gaussian random vector



Sample Mean, Sample Covariance

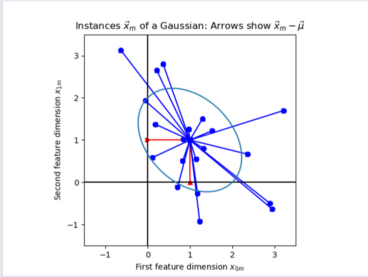
In the real world, we don't know $\vec{\mu}$ and Σ! If we have M instances $\vec{x}_m$ of the Gaussian, we can estimate

$$\vec{\mu} = \frac{1}{M}\sum_{m=0}^{M-1}\vec{x}_m$$

$$\Sigma = \frac{1}{M-1}\sum_{m=0}^{M-1}(\vec{x}_m - \vec{\mu})(\vec{x}_m - \vec{\mu})^T$$

Sample mean and sample covariance are not the same as real mean and real covariance, but we'll use the same letters ($\vec{\mu}$ and Σ) unless the problem requires us to distinguish.

Examples of $\vec{x}_m - \vec{\mu}$



Review: Eigenvalues and eigenvectors

The eigenvectors of a $D \times D$ square matrix, A, are the vectors $\vec{v}$ such that

$$A\vec{v} = \lambda\vec{v} \tag{1}$$

The scalar, λ, is called the eigenvalue. It's only possible for Eq. (1) to have a solution if

$$|A - \lambda I| = 0 \tag{2}$$

Left and right eigenvectors

Weve been working with right eigenvectors and right eigenvalues:

$$A\vec{v}_d = \lambda_d \vec{v}_d$$

There may also be left eigenvectors, which are row vectors $\vec{u}_d$ and corresponding left eigenvalues κ_d:

$$\vec{u}_d^T A = \kappa_d \vec{u}_d^T$$

Eigenvectors on both sides of the matrix

You can do an interesting thing if you multiply the matrix by its eigenvectors both before and after:

$$\vec{u}_i^T(A\vec{v}_j) = \vec{u}_i^T(\lambda_j\vec{v}_j) = \lambda_j\vec{u}_i^T\vec{v}_j$$

...but...

$$(\vec{u}_i^T A)\vec{v}_j = (\kappa_i\vec{u}_i^T)\vec{v}_j = \kappa_i\vec{u}_i^T\vec{v}_j$$

There are only two ways that both of these things can be true. Either

$$\kappa_i = \lambda_j \quad \text{or} \quad \vec{u}_i^T\vec{v}_j = 0$$

Left and right eigenvectors must be paired!!

There are only two ways that both of these things can be true. Either

$$\kappa_i = \lambda_j \quad \text{or} \quad \vec{u}_i^T \vec{v}_j = 0$$

Remember that eigenvalues solve $|A - \lambda_d I| = 0$. In almost all cases, the solutions are all distinct (A has distinct eigenvalues), i.e., $\lambda_i \neq \lambda_j$ for $i \neq j$. That means there is **at most one** λ_i that can equal each κ_i:

$$\begin{cases} i \neq j & \vec{u}_i^T \vec{v}_j = 0 \\ i = j & \kappa_i = \lambda_i \end{cases}$$

Outline

Properties of symmetric matrices

If A is symmetric with D eigenvectors, and D distinct eigenvalues, then

$$VV^T = V^T V = I$$

$$V^T A V = \Lambda$$

$$A = V \Lambda V^T$$

Symmetric matrices: left=right

If A is symmetric ($A = A^T$), then the left and right eigenvectors and eigenvalues are the same, because

$$\lambda_i \vec{u}_i^T = \vec{u}_i^T A = (A^T \vec{u}_i)^T = (A\vec{u}_i)^T$$

... and that last term is equal to $\lambda_i \vec{u}_i^T$ if and only if $\vec{u}_i = \vec{v}_i$.

Symmetric matrices: eigenvectors are orthonormal

Let's combine the following facts:

- $\vec{u}_i^T \vec{v}_j = 0$ for $i \neq j$ — any square matrix with distinct eigenvalues
- $\vec{u}_i = \vec{v}_i$ — symmetric matrix
- $\vec{v}_i^T \vec{v}_i = 1$ — standard normalization of eigenvectors for any matrix (this is what $\|\vec{v}_i\| = 1$ means).

Putting it all together, we get that

$$\vec{v}_i^T \vec{v}_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

The eigenvector matrix

So if A is symmetric with distinct eigenvalues, then its eigenvectors are orthonormal:

$$\vec{v}_i^T \vec{v}_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

We can write this as

$$V^T V = I$$

where

$$V = [\vec{v}_0, \ldots, \vec{v}_{D-1}]$$

The eigenvector matrix is orthonormal

$$V^TV = I$$

...and it also turns out that

$$VV^T = I$$

Proof: $VV^T = VIV^T = V(V^TV)V^T = (VV^T)^2$, but the only matrix that satisfies $VV^T = (VV^T)^2$ is $VV^T = I$.

Eigenvectors orthogonalize a symmetric matrix

So now, suppose A is symmetric:

$$\vec{v}_i^T A \vec{v}_j = \vec{v}_i^T (\lambda_j \vec{v}_j) = \lambda_j \vec{v}_i^T \vec{v}_j = \begin{cases} \lambda_j, & i = j \\ 0, & i \neq j \end{cases}$$

In other words, if a symmetric matrix has D eigenvectors with distinct eigenvalues, then its eigenvectors orthogonalize A:

$$V^T A V = \Lambda$$

$$\Lambda = \begin{bmatrix} \lambda_0 & 0 & 0 \\ 0 & \ldots & 0 \\ 0 & 0 & \lambda_{D-1} \end{bmatrix}$$

A symmetric matrix is the weighted sum of its eigenvectors:

One more thing. Notice that

$$A = VV^TAVV^T = V\Lambda V^T$$

The last term is

$$[\vec{v}_0, \ldots, \vec{v}_{D-1}] \begin{bmatrix} \lambda_0 & 0 & 0 \\ 0 & \ldots & 0 \\ 0 & 0 & \lambda_{D-1} \end{bmatrix} \begin{bmatrix} \vec{v}_0^T \\ \vdots \\ \vec{v}_{D-1}^T \end{bmatrix} = \sum_{d=0}^{D-1} \lambda_d \vec{v}_d \vec{v}_d^T$$

Summary: properties of symmetric matrices

If A is symmetric with D eigenvectors, and D distinct eigenvalues, then

$$A = V\Lambda V^T$$

$$\Lambda = V^T A V$$

$$VV^T = V^T V = I$$

Outline

How do you treat an image as a signal?

- An RGB image is a signal in three dimensions: $f[i,j,k] =$ intensity of the signal in the i^{th} row, j^{th} column, and k^{th} color.
- $f[i,j,k]$, for each (i,j,k), is either stored as an integer or a floating point number:
  - Floating point: usually $x \in [0,1]$, so $x = 0$ means dark, $x = 1$ means bright.
  - Integer: usually $x \in \{0,\ldots,255\}$, so $x = 0$ means dark, $x = 255$ means bright.
- The three color planes are usually:
  - $k = 0$: Red
  - $k = 1$: Blue
  - $k = 2$: Green

How do you treat an image as a vectors?

A vectorized RGB image is created by just concatenating all of the colors, for all of the columns, for all of the rows. So if the m^{th} image, $f_m[i,j,k]$, is $R \approx 200$ rows, $C \approx 400$ columns, and $K = 3$ colors, then we set

$$\vec{x}_m = \left[x_{m0}, \ldots, x_{m,D-1}\right]^T$$

where

$$x_{m,(iC+j)K+k} = f_m[i,j,k]$$

which has a total dimension of

$$D = RCK \approx 200 \times 400 \times 3 = 240,000$$

How do you classify an image?

Suppose we have a test image, $\vec{x}_{\text{test}}$. We want to figure out: who is this person?

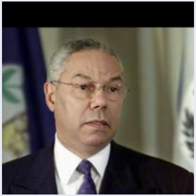
Test Datum $\vec{x}_{\text{test}}$:

Training Data?

In order to classify the test image, we need some training data. For example, suppose we have the following four images in our training data. Each image, $\vec{x}_m$, comes with a label, y_m, which is just a string giving the name of the individual.

Training Datum: $y_0 =$**Colin Powell:**

$\vec{x}_0 =$



Training Datum $y_1 =$**Gloria Arroyo:**

$\vec{x}_1 =$

Training Datum $y_2 =$**Megawati Sukarnoputri:**

$\vec{x}_2 =$

Training Datum $y_3 =$**Tony Blair:**

$\vec{x}_3 =$

Nearest Neighbors Classifier

A "nearest neighbors classifier" makes the following guess: the test vector is an image of the same person as the closest training vector:

$$\hat{y}_{\text{test}} = y_{m^*}, \quad m^* = \underset{m=0}{\overset{M-1}{\operatorname{argmin}}} \|\vec{x}_m - \vec{x}_{\text{test}}\|$$

where "closest," here, means Euclidean distance:

$$\|\vec{x}_m - \vec{x}_{\text{test}}\| = \sqrt{\sum_{d=0}^{D-1} (x_{md} - x_{\text{test},d})^2}$$

Improved Nearest Neighbors: Eigenface

- The problem with nearest-neighbors is that subtracting one image from another, pixel-by-pixel, results in a measurement that is dominated by noise.
- We need a better measurement.
- The solution is to find a signal representation, $\vec{y}_m$, such that $\vec{y}_m$ summarizes the way in which $\vec{x}_m$ differs from other faces.
- If we find $\vec{y}_m$ using principal components analysis, then $\vec{y}_m$ is called an "eigenface" representation.

Outline

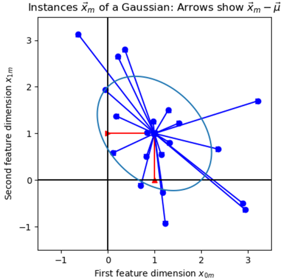
Sample covariance

$$\Sigma = \frac{1}{M-1}\sum_{m=0}^{M-1}(\vec{x}_m-\vec{\mu})(\vec{x}_m-\vec{\mu})^T$$
$$= \frac{1}{M-1}X^TX$$

...where X is the centered data matrix,

$$X = \begin{bmatrix} (\vec{x}_0-\vec{\mu})^T \\ \vdots \\ (\vec{x}_{M-1}-\vec{\mu})^T \end{bmatrix}$$

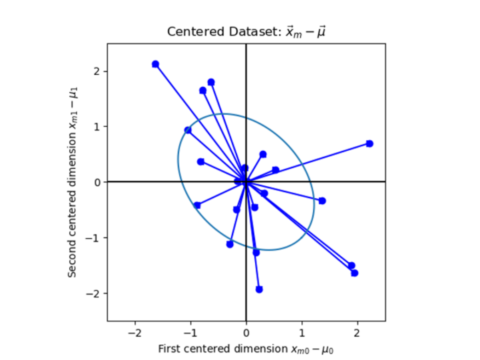
Examples of $\vec{x}_m - \vec{\mu}$



Centered data matrix

$$X = \begin{bmatrix} (\vec{x}_0 - \vec{\mu})^T \\ \vdots \\ (\vec{x}_{M-1} - \vec{\mu})^T \end{bmatrix}$$

Examples of $\vec{x}_m - \vec{\mu}$



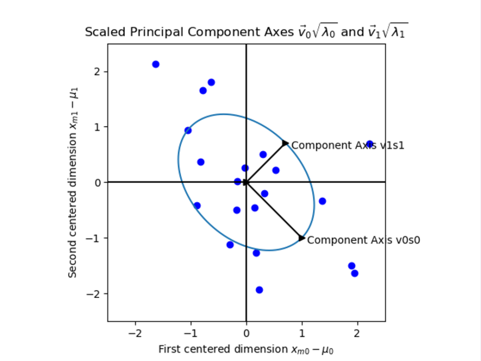
Principal component axes

X^TX is symmetric!
Therefore,

$$X^TX = V\Lambda V^T$$

$V = [\vec{v}_0, \ldots, \vec{v}_{D-1}]$, the eigenvectors of X^TX, are called the principal component axes, or principal component directions.

Principal component axes



Principal components

Remember that the eigenvectors of a matrix diagonalize it. So if V are the eigenvectors of X^TX, then

$$V^TX^TXV = \Lambda$$

Let's write $Y = XV$, and $Y^T = V^TX^T$. In other words,

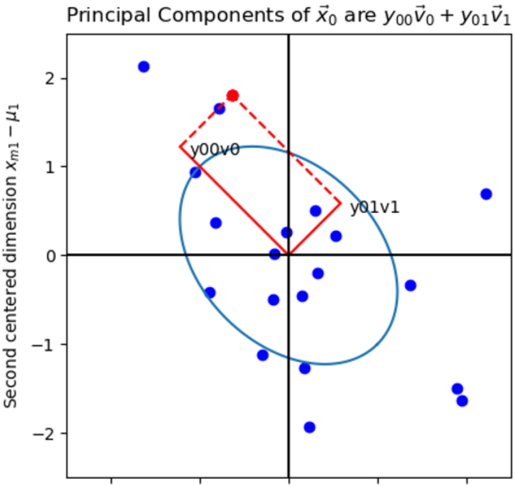
$$\vec{y}_m = V^T(\vec{x}_m - \vec{\mu})$$

$\vec{y}_m = [y_{m0}, \ldots, y_{m,D-1}]^T$ is the vector of principal components of $\vec{x}_m$. Expanding the formula $Y^TY = \Lambda$, we discover that PCA orthogonalizes the dataset:

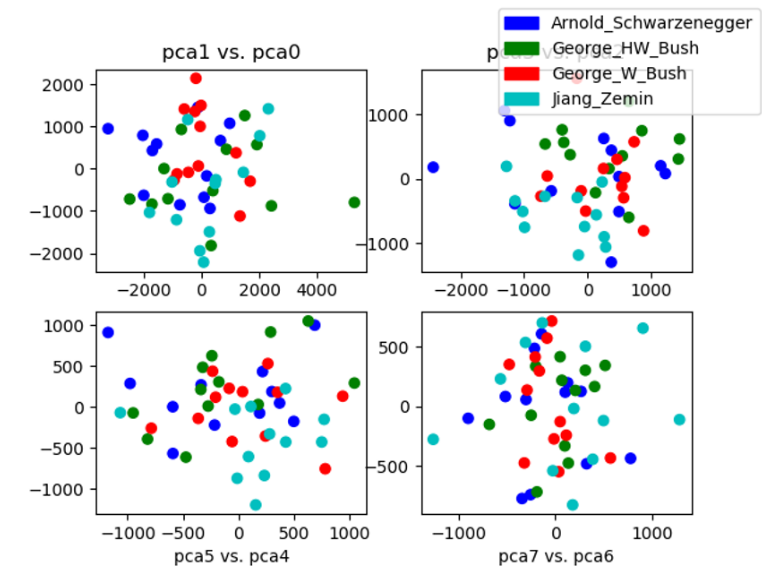
$$\sum_{m=0}^{M-1} y_{im}y_{jm} = \begin{cases} \lambda_i & i = j \\ 0 & i \neq j \end{cases}$$

Principal components

$$\vec{y}_m = V^T(\vec{x}_m - \vec{\mu})$$



Principal components with larger eigenvalues have more energy



Eigenvalue=Energy of the Principal Component

The total dataset energy is

$$\sum_{m=0}^{M-1} y_{mi}^2 = \lambda_i$$

But remember that $V^T V = I$. Therefore, the total dataset energy is the same, whether you calculate it in the original image domain, or in the PCA domain:

$$\sum_{m=0}^{M-1} \sum_{d=0}^{D-1} (x_{md} - \mu_d)^2 = \sum_{m=0}^{M-1} \sum_{i=0}^{D-1} y_{mi}^2 = \sum_{i=0}^{D-1} \lambda_i$$

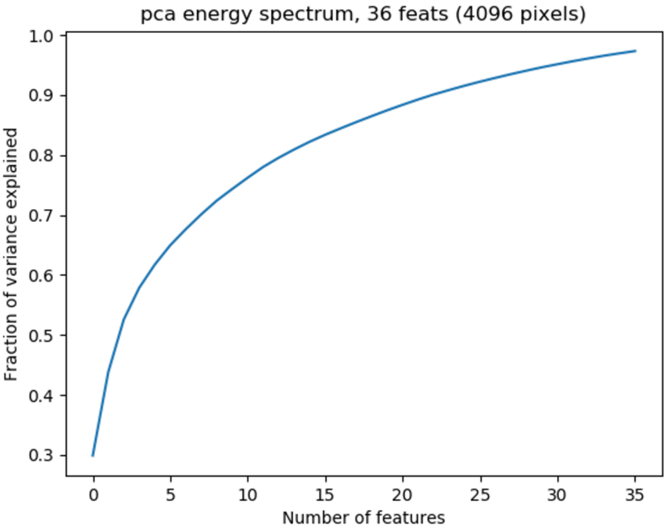
Energy spectrum=Fraction of energy explained

The "energy spectrum" is energy as a function of basis vector index. There are a few ways we could define it, but one useful definition is:

$$E[k] = \frac{\sum_{m=0}^{M-1} \sum_{i=0}^{k-1} y_{mi}^2}{\sum_{m=0}^{M-1} \sum_{i=0}^{D-1} y_{mi}^2}$$

$$= \frac{\sum_{i=0}^{k-1} \lambda_i}{\sum_{i=0}^{D-1} \lambda_i}$$

Energy spectrum=Fraction of energy explained



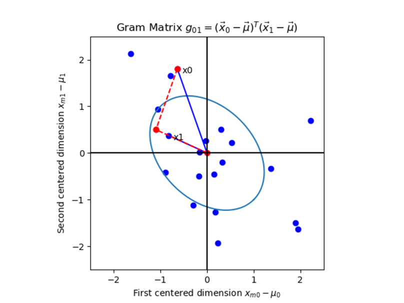
Outline

Gram matrix

- $X^T X$ is usually called the sum-of-squares matrix. $\frac{1}{M-1}X^T X$ is the sample covariance.
- $G = XX^T$ is called the gram matrix. Its $(i,j)^{\text{th}}$ element is the dot product between the i^{th} and j^{th} data samples:

$$g_{ij} = (\vec{x}_i - \vec{\mu})^T(\vec{x}_j - \vec{\mu})$$

Gram matrix $g_{01} = (\vec{x}_0 - \vec{\mu})^T(\vec{x}_1 - \vec{\mu})$



Eigenvectors of the Gram matrix

XX^T is also symmetric! So it has orthonormal eigenvectors:

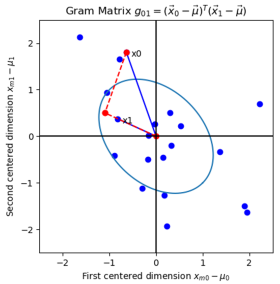
$$XX^T = U\Lambda U^T$$

$$UU^T = U^T U = I$$

$X^T X$ and XX^T have the same eigenvalues (Λ), but different eigenvectors (V vs. U).

Gram matrix

$g_{01} = (\vec{x}_0 - \vec{\mu})^T(\vec{x}_1 - \vec{\mu})$



Why the Gram matrix is useful:

Suppose (as in MP1) that $D \sim 240000$ pixels per image, but $M \sim 240$ different images. Then, in order to perform this eigenvalue analysis:

$$X^T X = V \Lambda V^T$$

...requires factoring a 240000^{th}-order polynomial ($|X^T X - \lambda I| = 0$), then solving 240000 simultaneous linear equations in 240000 unknowns to find each eigenvector ($X^T X \vec{v}_d = \lambda_d \vec{v}_d$). If you try doing that using `np.linalg.eig`, your PC will be running all day. On the other hand,

$$X X^T = U \Lambda U^T$$

requires only 240 equations in 240 unknowns. Educated experts agree: $240^2 \ll 240000^2$.

Singular Values

- Both $X^T X$ and XX^T are positive semi-definite, meaning that their eigenvalues are non-negative, $\lambda_d \geq 0$.
- The **singular values** of X are defined to be the square roots of the eigenvalues of $X^T X$ and XX^T:

$$S = \begin{bmatrix} s_0 & 0 & 0 \\ 0 & \ldots & 0 \\ 0 & 0 & s_{D-1} \end{bmatrix}, \quad \Lambda = S^2 = \begin{bmatrix} s_0^2 & 0 & 0 \\ 0 & \ldots & 0 \\ 0 & 0 & s_{D-1}^2 \end{bmatrix}$$

Singular Value Decomposition

$$X^T X = V \Lambda V^T = VSSV^T$$
$$XX^T = U \Lambda U^T = USSU^T$$

Singular Value Decomposition

$$X^TX = VSSV^T = VSISV^T = VSU^TUSV^T = (USV^T)^T(USV^T)$$

$$XX^T = USSU^T = USISU^T = USV^TVSU^T = (USV^T)(USV^T)^T$$

Singular Value Decomposition

Any matrix, X, can be written as $X = USV^T$.

- $U = [\vec{u}_0, \ldots, \vec{u}_{M-1}]$ are the eigenvectors of XX^T.
- $V = [\vec{v}_0, \ldots, \vec{v}_{D-1}]$ are the eigenvectors of X^TX.
- $S = \begin{bmatrix} s_0 & 0 & 0 & 0 & 0 \\ 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & s_{\min(D,M)-1} & 0 & 0 \end{bmatrix}$ are the singular values.

S has some all-zero columns if $M > D$, or all-zero rows if $M < D$.

What np.linalg.svd does

First, `np.linalg.svd` decides whether it wants to find the eigenvectors of X^TX or XX^T: it just checks to see whether $M > D$ or vice versa. If it discovers that $M < D$, then:

1. Compute $XX^T = U\Lambda U^T$, and $S = \sqrt{\Lambda}$. Now we have U and S, we just need to find V.
2. Since $X^T = VSU^T$, we can get V by just multiplying:

$$\tilde{V} = X^T U$$

…where $\tilde{V} = VS$ is exactly equal to V, but with each column scaled by a different singular value. So we just need to normalize:

$$\|\vec{v}_i\| = 1, \quad v_{i0} > 0$$

Methods that solve MP1

- Direct eigenvector analysis of $X^T X$ gives the right answer, but takes a very long time. When I tried this, it timed out the autograder.
- Applying `np.linalg.svd` to X should give the right answer, very fast. I haven't tried it this year, but it worked on last year's dataset.
- What I tried, this year, is the gram matrix method: Apply `np.linalg.eig` to get U from XX^T. Multiply $\tilde{V} = X^T U$, then normalize the columns to get V.

Outline

Summary

- Symmetric matrices:

$$A = V\Lambda V^T, \quad V^T A V = \Lambda, \quad V^T V = VV^T = I$$

- Centered dataset:

$$X = \begin{bmatrix} (\vec{x}_0 - \vec{\mu})^T \\ \vdots \\ (\vec{x}_{M-1} - \vec{\mu})^T \end{bmatrix}$$

- Singular value decomposition:

$$X = USV^T$$

where V are eigenvectors of the sum-of-squares matrix, U are eigenvectors of the gram matrix, and $\Lambda = S^2$ are their shared eigenvalues.