

Tools & Models for Data Science

Dimensionality Reduction

Chris Jermaine & Risa Myers

Rice University



Our Scenario

- We have high dimensional data and we want to make predictions on the data
- ... or we want to classify the data
- but, there's too much data
- ? What can we do?
- Note: This is a “What” topic, not a “How” topic

Too much Data Problem

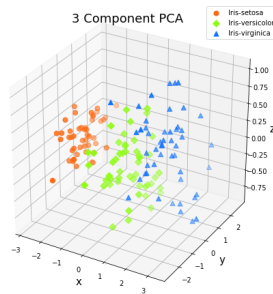
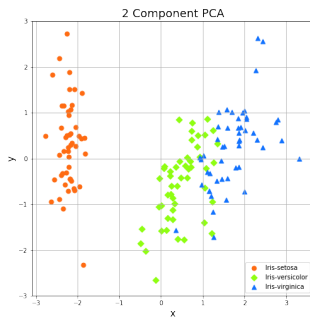
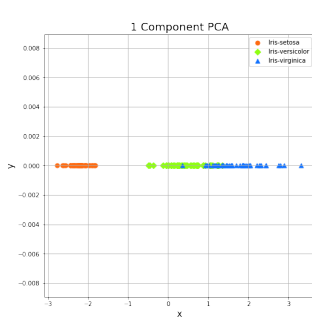
? What can we do?

- 1 We can use an approach that randomly samples from the data
... not today
- 2 We can throw away data points
... that's almost always a bad idea
- 3 We can throw away features
... let's explore this option

- What do we mean by “high dimension”?
 - Many, many features
 - Can be millions!
 - Lots of useless features obscure useful ones
 - Example: Genomic data
 - Comparing genome sequences
 - Classifying genomes

The Curse of Dimensionality

- "As the number of features or dimensions grows, the amount of data we need to generalize accurately grows exponentially."
 - Charles Isbell, Professor and Senior Associate Dean, School of Interactive Computing, Georgia Tech
- Data become sparse in high dimensions
- All points are basically the same distance from one another



Dimensionality Reduction: Basic Idea

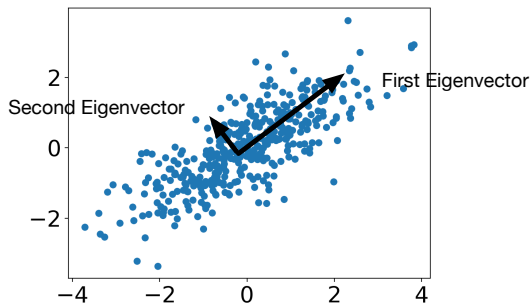
- Keep the dimensions that contain the most information
- Discard the least helpful dimensions
- Get the data down to a manageable size
- This is a type of feature extraction

Dimensionality Reduction vs. Feature Selection

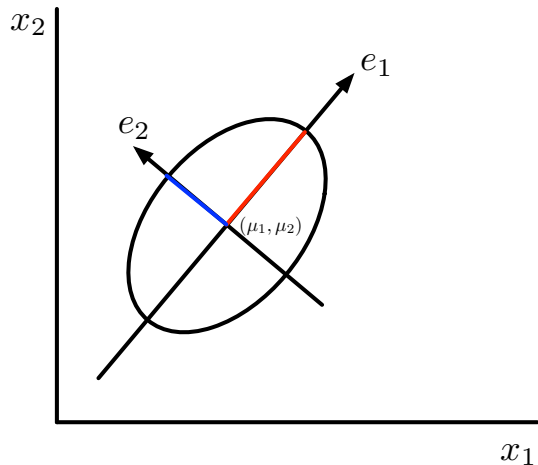
- Creates new, linear combinations of features vs. selecting subsets of features
- Built in attention to correlation
- New features may not be interpretable

How to do Dimensionality Reduction

- Goal: Find the line(s) on which to project the data to show most of the variability
- Approach: Compute a set of orthogonal (perpendicular) basis vectors
- We do this by computing the Eigenvalues and Eigenvectors of the data



Eigenvalues and Eigenvectors



- The ellipse describes the shape of the data
- The length of the red line is the value of the first Eigenvalue
- The length of the blue line is the value of the second Eigenvalue
- Each Eigenvector has direction based on the tilt of the data
- ...and the length is the Eigenvalue

What Does Our Data Look Like?

- \mathbf{X} is $n \times d$

- $n \ll d$

- Genomic data
- Many more dimensions than data points
- Each row is a patient
- Each item is a feature: nucleotide, age, fasting blood glucose level, ...
- Underconstrained / underspecified problem

- $n \gg d$

- Netflix
- Lots of users
- Not so many movies
- Not so many reviews
- Not addressed by this approach
- Overconstrained / overspecified problem

$$\mathbf{X} = \begin{matrix} & \overbrace{\hspace{10em}}^d \\ n \left\{ \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & \cdots & \cdots & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & \cdots & \cdots & \cdots & x_{2,d} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n,1} & x_{n,2} & \cdots & \cdots & \cdots & \cdots & x_{n,d} \end{bmatrix} \right. \end{matrix}$$

Dimensionality Reduction

- \mathbf{X} can be too big to operate on directly
- We want a linear transform to compute data matrix \mathbf{X}' with a reduced number of dimensions

$$\mathbf{X} = n \left\{ \begin{array}{c} \overbrace{\left[\begin{array}{cccccccccccccccc} x_{1,1} & x_{1,2} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{2,d} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n,1} & x_{n,2} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{n,d} \end{array} \right]}^d \end{array} \right.$$
$$\mathbf{X}' = n \left\{ \begin{array}{c} \overbrace{\left[\begin{array}{cccc} x'_{1,1} & x'_{1,2} & \cdots & x'_{1,m} \\ x'_{2,1} & x'_{2,2} & \cdots & x'_{2,m} \\ \cdots & \cdots & \cdots & \cdots \\ x'_{n,1} & x'_{n,2} & \cdots & x'_{n,m} \end{array} \right]}^m \end{array} \right.$$

Computing \mathbf{X}'

- Linear transform realized by a mapping matrix \mathbf{W} (d rows, m columns)
- So that $\mathbf{X}' = \mathbf{XW}$

$$\begin{bmatrix} \mathbf{X}' \end{bmatrix} = \begin{bmatrix} \mathbf{X} \end{bmatrix} \times \begin{bmatrix} \mathbf{W} \end{bmatrix}$$

How Big are these Matrices?

- \mathbf{X}' is $n \times m$
- $m \ll d$
- Recall $\mathbf{X}' = \mathbf{X}\mathbf{W}$
- Want to compute \mathbf{W} such that
- The columns of \mathbf{W} are the basis functions in the reduced space

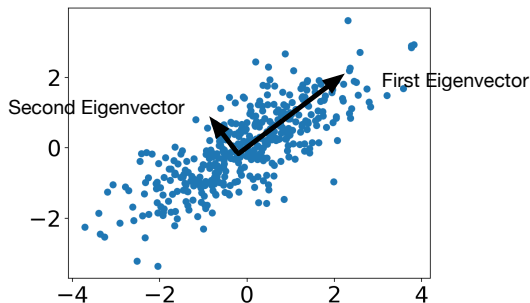
What can **W** do?

- **$X' = XW$**
- It can weight columns by zero
- It can create new columns that are weighted sums of other columns
- ...

Classic Dimensionality Reduction Method: PCA

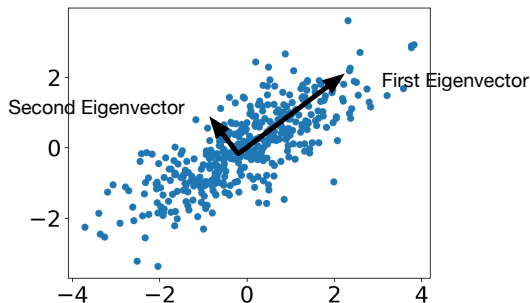
- Basic idea: Compute a set of orthogonal (perpendicular) basis vectors
- Then choose most important basis vectors (those with the most variance)
- Those become the rows of the mapping matrix \mathbf{W}
- Then \mathbf{XW} projects down onto that basis, giving us \mathbf{X}'

? If we want to draw a line that best describes the data, what line do we draw?



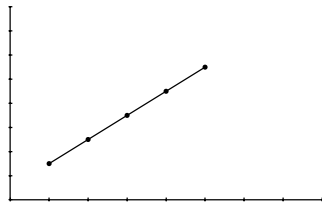
Classic Dimensionality Reduction Method: PCA

- Basic idea: Compute a set of orthogonal (perpendicular) basis vectors
 - Then choose most important basis vectors (those with the most variance)
 - Those become the rows of the mapping matrix \mathbf{W}
 - Then \mathbf{XW} projects down onto that basis, giving us \mathbf{X}'
-
- If we want to draw a line that best describes the data, what line do we draw?
 - We draw the first Eigenvector



An Extreme Data Set

- Here there is no variability along the second dimension
- In the extreme case, we could completely drop it
- Note: When $d \gg n$ there will be some dimensions with Eigenvalue = 0



Classic Dimensionality Reduction Method: PCA

- Principal Component Analysis
- We want to lose the least amount of information
- So we remove the dimensions with NO variability
- Note that the number of principle components = the number of dimensions in our data
- We want to drop out the low variance ones that aren't helpful
- For example:
 - If all of our data points are about people who are ages 18 - 19
 - ... or have a fasting blood glucose value in a very narrow, equivalent range
 - That piece of information is highly unlikely to be useful in discriminating between people

Consider the Case of the Multivariate Normal Distribution

- \mathbf{Z} is an equation variable
- From the data matrix, \mathbf{X} we get
 - μ is the vector of means, 1 for each dimension
 - Σ is the covariance matrix of \mathbf{X}
- Each entry, $\sigma_{i,j} = E[(\mathbf{X}_i - \mu_i)(\mathbf{X}_j - \mu_j)]$
- Σ is a symmetric matrix
- If Σ is diagonal
 - The dimensions of \mathbf{X} are independent
- We leverage the covariance matrix to figure out which dimensions are most important

$$\text{pdf}(\mathbf{Z}) = \frac{\exp(-\frac{1}{2}(\mathbf{Z}-\mu)^T \Sigma^{-1}(\mathbf{Z}-\mu))}{\sqrt{(2\pi)^k |\Sigma|}}$$

$$\Sigma = \begin{bmatrix} \sigma_{1,1} & \sigma_{1,2} & \cdots & \sigma_{1,d} \\ \sigma_{2,1} & \sigma_{2,2} & \cdots & \sigma_{2,d} \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_{d,1} & \sigma_{d,2} & \cdots & \sigma_{d,d} \end{bmatrix}$$

Consider the Case of the Multivariate Normal Distribution

- From Linear Algebra, we know
 - The Eigenvectors of Σ describe the direction of the spread in \mathbf{X}
 - The Eigenvalues of Σ describe the magnitude of the Eigenvectors
- Now we know why we need them
 - To project data onto them to form the mapping matrix \mathbf{W}
- We also know what they are
 - They determine the shape and spread of the data cloud
- ? How do we determine the Eigenvalues & Eigenvectors?

How do we determine the Eigenvalues & Eigenvectors?

- 1 Eigenvalue Decomposition
- 2 Singular Value Decomposition

Computing the Eigenvalues & Eigenvectors

- Method 1: EVD
- The covariance matrix Σ can be decomposed into $\Sigma = \mathbf{V}\mathbf{L}\mathbf{V}^T$
- This is the **Eigenvalue Decomposition** of Σ
- Where \mathbf{L} is a diagonal matrix of values, typically sorted in decreasing order, from left to right
- These values are called Eigenvalues
- The top left value is the largest / most significant
- The columns of \mathbf{V} are Eigenvectors
- ...and will be sorted by Eigenvalue magnitude
- Each Eigenvalue is how much variance exists across the Eigenvector

Why EVD Doesn't Work in High Dimensions

- It decomposes the covariance matrix Σ , which is $d \times d$, to get the Eigenvalues and Eigenvectors
- Recall that the $d \gg n$
- Computationally difficult
- The covariance matrix is $d \times d$, so taking the inverse is $O(d^3)$
- This can be done while $d < 50K$ or so
- Once the data gets “big” we can't use closed form approaches

Why use the Covariance Matrix?

- Because it's square
- Because it describes the relationships between the different dimensions
- It solves the Normal Equation from Linear Regression (closed form solution)
 - $\mathbf{X}r = b$
 - Where r is the vector of regression coefficients
 - and b is the vector of intercept terms
 - Note that we are not guaranteed to have a solution to this equation
 - Because the problem is underconstrained

Why use the Covariance Matrix?

1 $\mathbf{X}r = b$

2 $\mathbf{X}^T \mathbf{X}r = \mathbf{X}^T b$

3 $\frac{1}{n-1} \mathbf{X}^T \mathbf{X}r = \mathbf{X}^T b \frac{1}{n-1}$

1 Normal Equation

2 Multiply both sides by \mathbf{X}^T

3 Multiply both sides

■ Note that $\frac{1}{n-1} \mathbf{X}^T \mathbf{X}$ is the definition of a covariance matrix, Σ for data with mean 0

■ The $n - 1$ term gives us a bias free estimator

$$\frac{1}{n-1} \mathbf{X}^T \mathbf{X} r = \mathbf{X}^T b \frac{1}{n-1}$$

- To solve for r , compute $(\mathbf{X}^T \mathbf{X})^{-1}$
- But this is very expensive to compute
- Instead we can approximate the calculation
- EVD gives the best approximation
- So, we do an EVD of Σ to learn which dimensions are important

- Eigenvalue decomposition is $O(d^3)$
- Which leads us to Method 2: Singular Value Decomposition (SVD) of \mathbf{X}
- In SVD, we operate on \mathbf{X} directly, instead of Σ because it is smaller
- \mathbf{X} is $n \times d$ and Σ is $d \times d$
- Recall $n \ll d$

- We want to avoid computing and using covariance matrix Σ because it is $\mathbf{X}^T \mathbf{X}$ and is of size $d \times d$
- ? Can we get the Eigenvalues and Eigenvectors of Σ by using \mathbf{X} instead?

SVD Decomposition

- We want to avoid computing and using covariance matrix Σ because it is $\mathbf{X}^T \mathbf{X}$ and is of size $d \times d$
- Can we get the Eigenvalues and Eigenvectors of Σ by using \mathbf{X} instead?
- The SVD of $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$
 - \mathbf{U} is a unitary matrix ($\mathbf{U}\mathbf{U}^T = \mathbf{I}$)
 - \mathbf{V} is matrix of right singular vectors
 - \mathbf{S} is a diagonal matrix of singular values
- Idea is that we can factorize \mathbf{X} instead of Σ
- We can reduce the number of dimensions by projecting onto the space determined by the right singular vectors with the largest values
- In this case, the biggest singular values correspond to the Eigenvalues with the largest magnitude

Using the SVD of \mathbf{X}

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

$$\mathbf{X}^T\mathbf{X} = (\mathbf{V}\mathbf{S}^T\mathbf{U})(\mathbf{U}\mathbf{S}\mathbf{V}^T)$$

$$= \mathbf{V}\mathbf{S}^T\mathbf{S}\mathbf{V}^T$$

- Where $\mathbf{S}^T\mathbf{S}$ is \mathbf{L} from EVD
- Where \mathbf{L} is a diagonal matrix of values, typically sorted in decreasing order, from left to right

Dimensions & Complexity, Revisited

- \mathbf{X} is $n \times d$
- $\mathbf{X}^T \mathbf{X}$ is $d \times d$
- When $d \gg n$
- EVD is $O(d^3)$, operating on a $d \times d$ covariance matrix
- SVD is $O(\min(d^2 n, n^2 d))$, because we are operating on \mathbf{X} , instead of Σ

How Do We Find the Dimensions with the Most Variance?

■ Options

- 1 Traditional approach: Eigenvalue Decomposition (EVD)
- 2 More efficient approach: Singular Value Decomposition (SVD)
- 3 Highly efficient approach: Random Matrix

■ Taking into consideration

- \mathbf{X} is not square
- So, it can't have Eigenvalues / Eigenvectors
- So, we use the covariance matrix Σ instead

Which Dimensions Matter?

■ AKA How to Compute the Basis Vectors?

1 Center your data

- Subtract out the mean in each dimension
- So the data has mean 0

2 Compute Σ

3 Do the EVD of $\Sigma = \mathbf{V}\mathbf{L}\mathbf{V}^T$

- Then the basis vectors are the Eigenvectors of data covariance matrix, Σ

$$\Sigma = \frac{\mathbf{X}^T \mathbf{X}}{(n-1)}$$

- Where Σ is a $d \times d$ matrix

4 Get \mathbf{W} from the vectors in \mathbf{V} , using the largest values in \mathbf{L}

How to Compute the Basis Vectors?

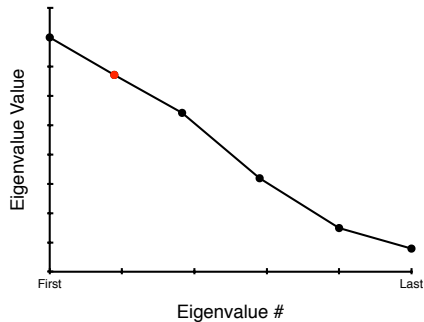
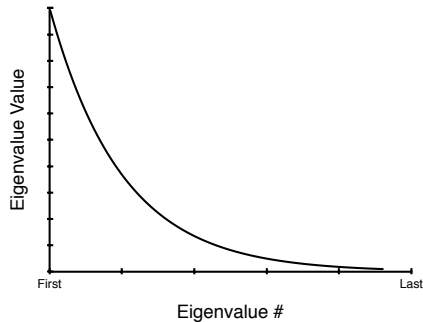
- From the EVD of $\Sigma = \mathbf{V}\mathbf{L}\mathbf{V}^T$
- Construct mapping matrix \mathbf{W} from the m most important Eigenvectors
- That is, the leftmost columns of \mathbf{V}

$$\begin{bmatrix} \vdots \\ \mathbf{x}' \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \mathbf{x} \\ \vdots \end{bmatrix} \times \begin{bmatrix} \vdots \\ \mathbf{w} \\ \vdots \end{bmatrix}$$

- \mathbf{X} is $n \times d$
- $\mathbf{X}^T \mathbf{X}$ is $d \times d$
- When $d \gg n$
- EVD is $O(d^3)$, operating on a $d \times d$ covariance matrix
- SVD is $O(\min(d^2 n, n^2 d))$, because we are operating on \mathbf{X} , instead of Σ

How Many Eigenvectors to Use?

- There's no magic number
- Look at the values
- Decide how many you can handle, computationally
- Red point is probably not a good choice to stop including Eigenvectors



Even SVD Can Be Too Expensive

- In practice, just fill \mathbf{W} with samples from $\text{Normal}(0, 1)$
 - Known as a random projection¹
 - Often does a very nice job in practice!
 - Johnson-Lindenstrauss Lemma
 - Basic premise is that the distances between points in a high dimensional space are “nearly preserved” when they are projected on to a large enough random subspace

¹Bingham E, Mannila H, editors. Random projection in dimensionality reduction: applications to image and text data. Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining; 2001: ACM.

- Why do it at all?
 - Counteract the Curse of Dimensionality
 - Reduce computational burden

Summary of Methods

- PCA via Eigenvalue decomposition
 - Classic approach
 - Can be difficult for even good software to do the math correctly (consider ill-conditioned matrices)
- PCA via SVD
 - Avoids the covariance matrix entirely, we operate directly on \mathbf{X}
 - Considered more stable numerically
 - Computationally faster
 - But still very expensive
 - Results in the same \mathbf{W}
- Random Projections
 - Not as precise
 - May not get the dimensions with the most variance
 - But is computationally feasible

Questions?

- What do we know now that we didn't know before?
 - We know what the curse of dimensionality is and why it is a problem
 - We know that it's possible to reduce the dimensions of our data
 - We know some approaches to perform dimensionality reduction
 - We know some trade-offs between the approaches
- How can we use what we learned today?
 - We can choose an appropriate method for dimensionality reduction