PCA

Reading list for this week:
Bishop Chap 12.1; optional add ons 12.2, 12.3 and Hastie Chap 14.5

Slides in this presentation courtesy of Matt Gormley and MIT 10-301

https://forms.gle/Dbhp6UoLhpw3ToaJ7

We need to fix this class!

High Dimension Data

Examples of high dimensional data:

– Brain Imaging Data (100s of MBs per scan)

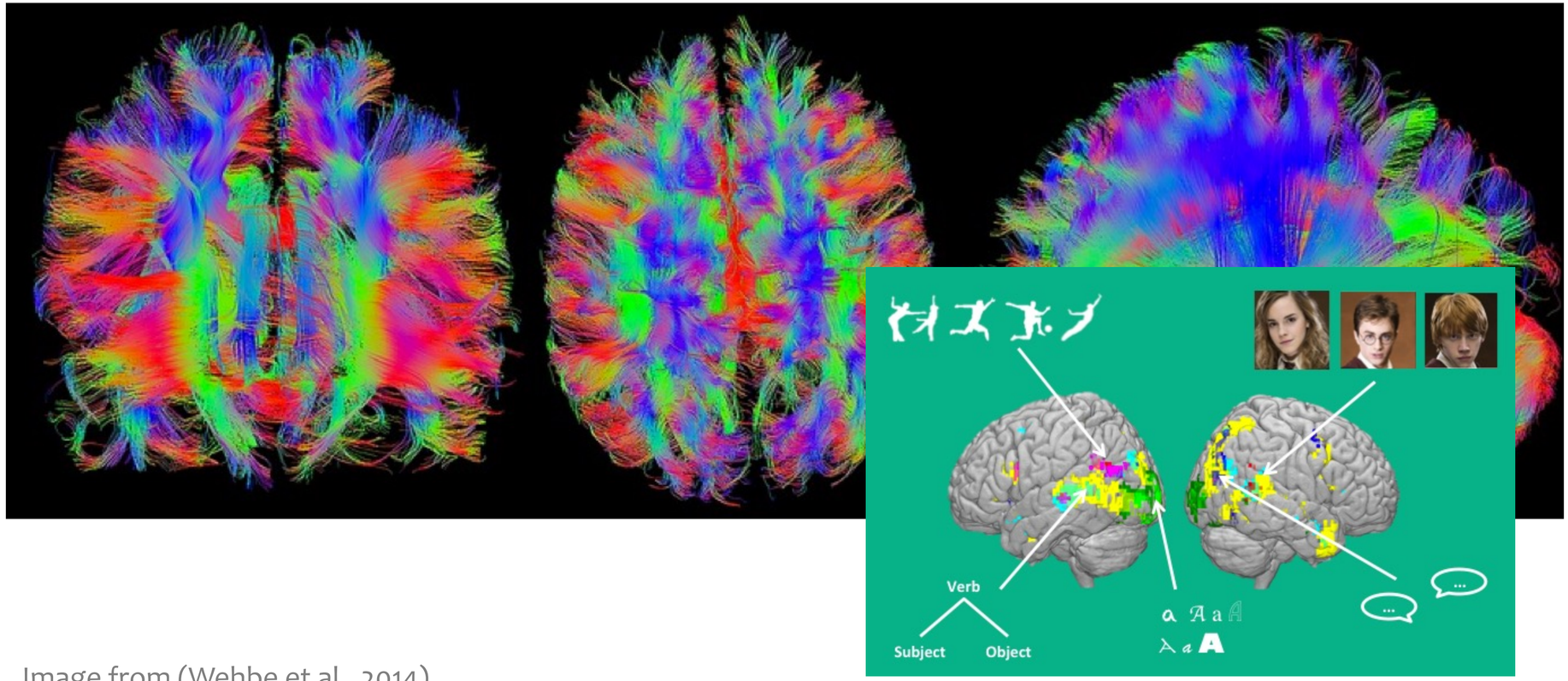


Image from (Wehbe et al., 2014)

Image from https://pixabay.com/en/brain-mrt-magnetic-resonance-imaging-1728449/

High Dimension Data

Examples of high dimensional data:

– High resolution images (millions of pixels)

High Dimension Data

Examples of high dimensional data:

– Multilingual News Stories

Vocabulary of millions of words!

Learning Representations

Dimensionality Reduction Algorithms:

Powerful (often unsupervised) learning techniques for extracting hidden (potentially lower dimensional) structure from high dimensional datasets.

Examples:

PCA, Kernel PCA, ICA, CCA, t-SNE, Autoencoders, Matrix Factorization

Useful for:

- Visualization
- More efficient use of resources (e.g., time, memory, communication)
- Statistical: fewer dimensions → better generalization
- Noise removal (improving data quality)

Slide adapted from Nina Balcan

Dimensionality reduction by random projection

Random Projection

Example: 2D to 1D

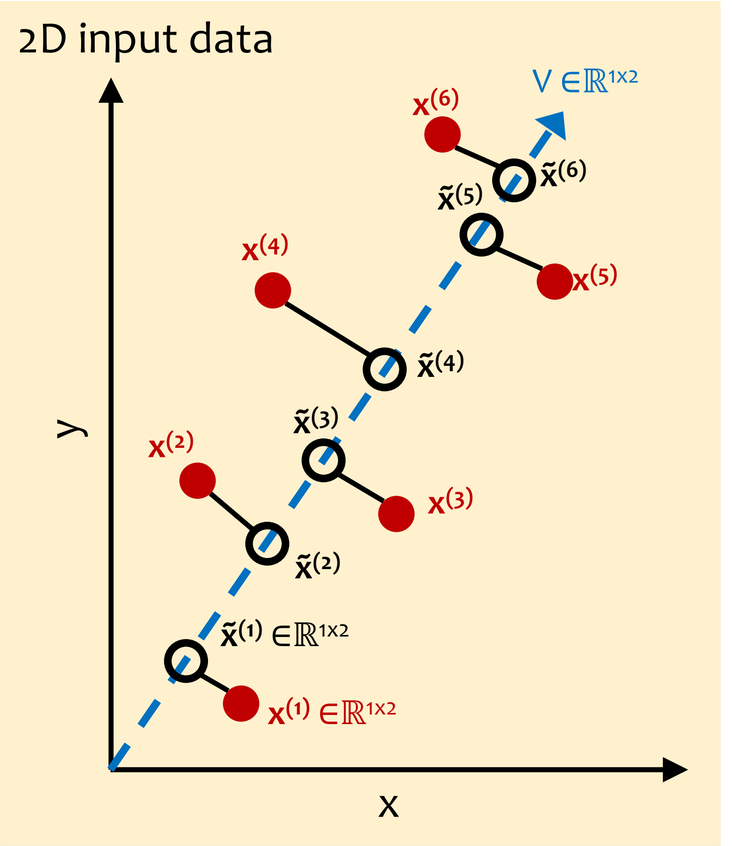
Goal: project from M-dimensions down to K-dimensions

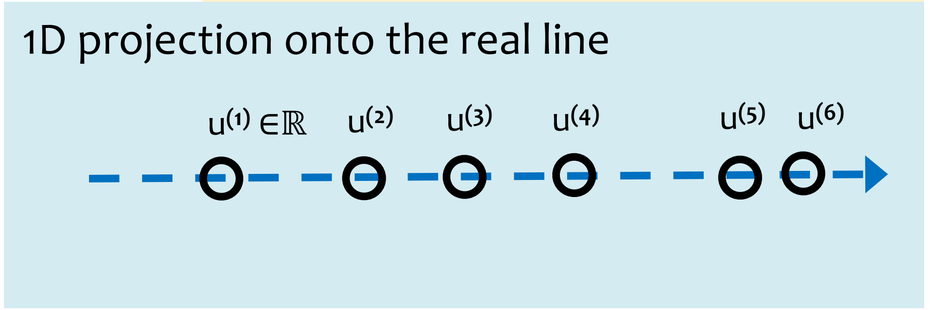
Data:

$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$

Algorithm:

1. Randomly sample matrix: $\mathbf{V} \in \mathbb{R}^{K \times M}$
   $V_{km} \sim \text{Gaussian}(0, 1)$
2. Project down: $\underbrace{\mathbf{u}^{(i)}}_{K \times 1} = \underbrace{\mathbf{V}}_{K \times M} \underbrace{\mathbf{x}^{(i)}}_{M \times 1}$
3. Project up: $\underbrace{\tilde{\mathbf{x}}^{(i)}}_{M \times 1} = \underbrace{\mathbf{V}^T}_{M \times K} \underbrace{\mathbf{u}^{(i)}}_{K \times 1} = \mathbf{V}^T(\mathbf{V}\mathbf{x}^{(i)})$





Random Projection

Goal: project from M-dimensions down to K-dimensions

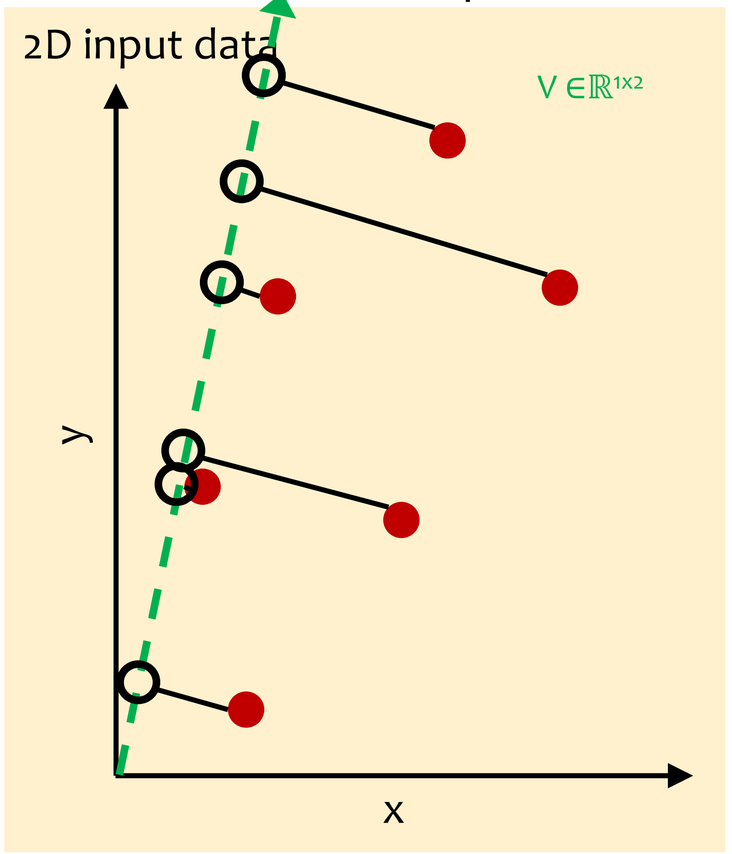
Data:

$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$

Algorithm:

1. Randomly sample matrix: $\mathbf{V} \in \mathbb{R}^{K \times M}$
   $V_{km} \sim \text{Gaussian}(0, 1)$
2. Project down: $\underbrace{\mathbf{u}^{(i)}}_{K \times 1} = \underbrace{\mathbf{V}}_{K \times M} \underbrace{\mathbf{x}^{(i)}}_{M \times 1}$
3. Project up: $\underbrace{\mathbf{x}^{(i)}}_{M \times 1} = \underbrace{\mathbf{V}^T}_{M \times K} \underbrace{\mathbf{u}^{(i)}}_{K \times 1} = \mathbf{V}^T(\mathbf{V}\mathbf{x}^{(i)})$

Example: 2D to 1D



Problem: a random projection might give us a poor low dimensional representation of the data

Johnson-Lindenstrauss Lemma

Q: But how could we ever hope to preserve any useful information by randomly projecting into a low-dimensional space?

A: Even random projection enjoys some surprisingly impressive properties. In fact, a standard of the J-L lemma starts by assuming we have a random linear projection obtained by sampling each matrix entry from a Gaussian(0,1).

An Elementary Proof of a Theorem of Johnson and Lindenstrauss

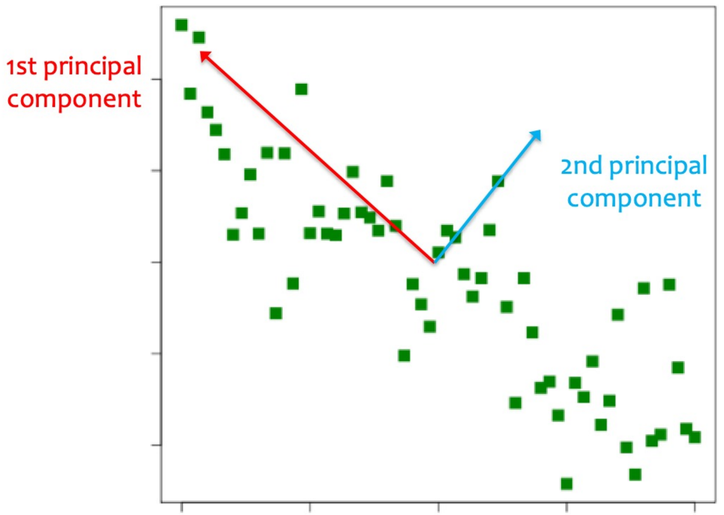
Sanjoy Dasgupta,[1] Anupam Gupta[2]

ABSTRACT: A result of Johnson and Lindenstrauss [13] shows that a set of n points in high dimensional Euclidean space can be mapped into an $O(\log n/\epsilon^2)$-dimensional Euclidean space such that the distance between any two points changes by only a factor of $(1 \pm \epsilon)$. In this note, we prove this theorem using elementary probabilistic techniques. © 2003 Wiley Periodicals, Inc. Random Struct. Alg., 22: 60–65, 2002

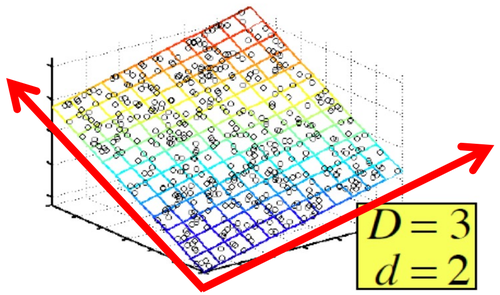
Principle Components Analysis

Principal Component Analysis (PCA)

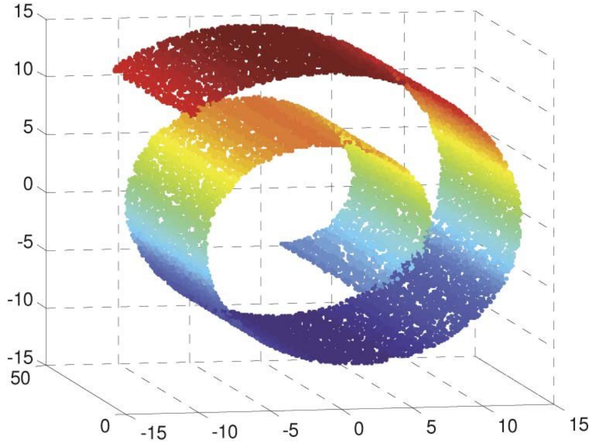
- **Assumption:** the data lies on a low K-dimensional linear subspace
- **Goal:** identify the axes of that subspace, and project each point onto hyperplane
- **Algorithm:** find the K eigenvectors with largest eigenvalue using classic matrix decomposition tools



PCA Example: 2D Gaussian Data



Non-linear!



https://commons.wikimedia.org/wiki/File:Scatter_diagram_for_quality_characteristic_XXX.svg

Data for PCA

$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$$

$$\mathbf{x}^{(i)} \in \mathbb{R}^M$$

$$\mathbf{X} = \begin{bmatrix} (\mathbf{x}^{(1)})^T \\ (\mathbf{x}^{(2)})^T \\ \vdots \\ (\mathbf{x}^{(N)})^T \end{bmatrix}$$

We assume the data is **centered**, i.e. the **sample mean** is zero

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^{(i)} = \mathbf{0}$$

Q: What if your data is **not** centered?

A: Subtract off the sample mean

$$\tilde{\mathbf{x}}^{(i)} = \mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}}, \forall i$$

Sample Covariance Matrix

The **sample covariance matrix** $\boldsymbol{\Sigma} \in \mathbb{R}^{M \times M}$ is given by:

$$\Sigma_{jk} = \frac{1}{N} \sum_{i=1}^{N} (x_j^{(i)} - \mu_j)(x_k^{(i)} - \mu_k)$$

Since the data matrix is centered, we rewrite as:

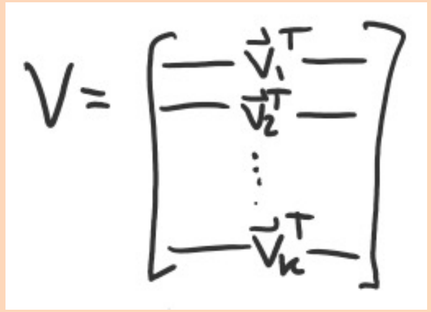
$$\boldsymbol{\Sigma} = \frac{1}{N} \mathbf{X}^T \mathbf{X}$$

$$\mathbf{X} = \begin{bmatrix} (\mathbf{x}^{(1)})^T \\ (\mathbf{x}^{(2)})^T \\ \vdots \\ (\mathbf{x}^{(N)})^T \end{bmatrix}$$

Principal Component Analysis (PCA)

Linear Projection:
Given KxM matrix **V**, and Mx1 vector $\mathbf{x}^{(i)}$ we obtain the Kx1 projection $\mathbf{u}^{(i)}$ by:

$$\mathbf{u}^{(i)} = \mathbf{V}\,\mathbf{x}^{(i)}$$



Definition of PCA:

PCA repeatedly chooses a next vector $\mathbf{v}_j$ that minimizes the reconstruction error s.t. $\mathbf{v}_j$ is orthogonal to $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{j-1}$.

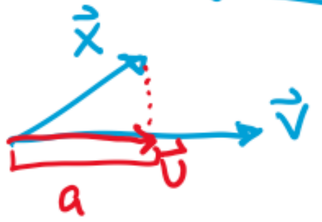
Vector $\mathbf{v}_j$ is called the **jth principal component**.

Notice: Two vectors **a** and **b** are **orthogonal** if $\mathbf{a}^T\mathbf{b} = 0$.
➔the K-dimensions in PCA are uncorrelated

Vector projection

Vector Projection

Recall: Projection



length of projection of $\vec{x}$ onto $\vec{v}$

$$a = \frac{\vec{v}^T\vec{x}}{\|v\|_2} \quad \text{if } \|v\|_2 = 1 \quad \text{otherwise}$$

projection of $\vec{x}$ onto $\vec{v}$

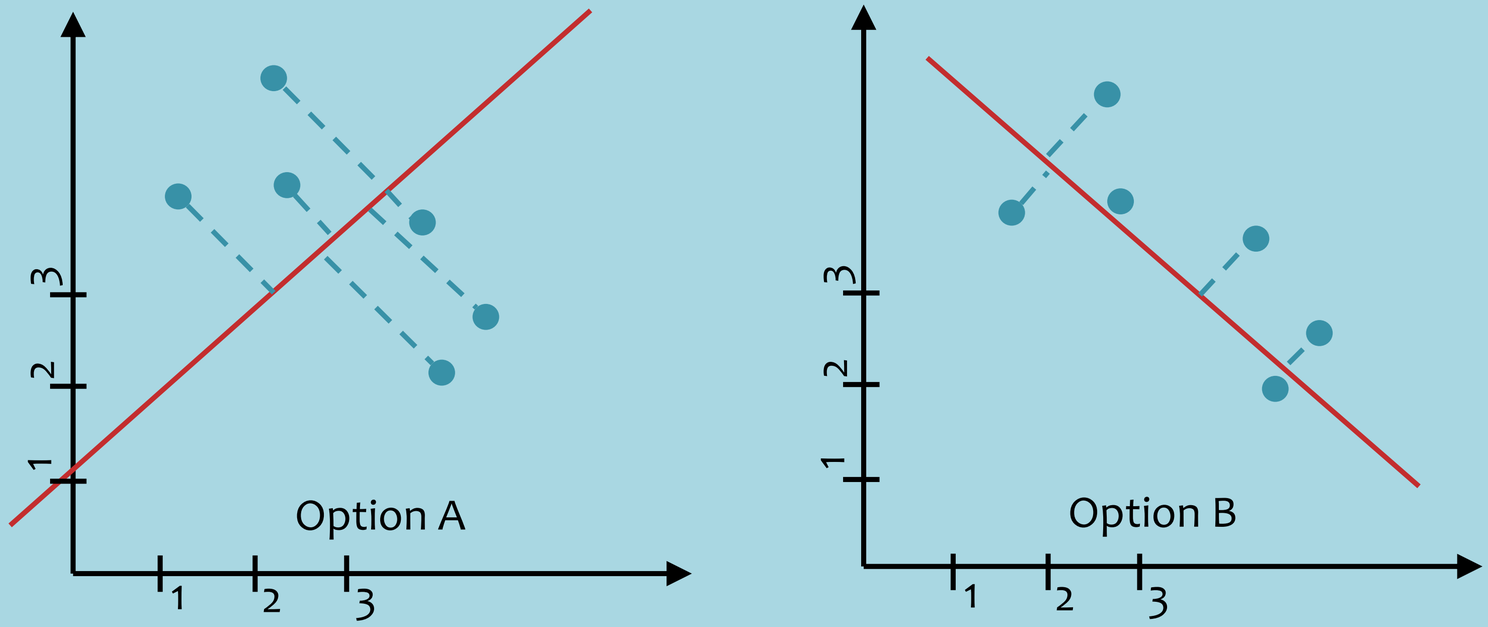
$$\vec{u} = a\vec{v} = \frac{(\vec{v}^T\vec{x})\vec{v}}{\|\vec{v}\|_2^2} \quad \text{if } \|v\|_2 = 1 \quad \text{otherwise}$$

Projection Example

Question:
Below are two plots of the same dataset D. Consider the two projections shown.

1. Which maximizes the variance?
2. Which minimizes the reconstruction error?

Answer:



PCA Objective Functions

What is the first principal component $\mathbf{v}_1$ chosen by PCA?

Option 1: The vector that *minimizes* the **reconstruction error**

$$\mathbf{v}_1 = \underset{\mathbf{v}:||\mathbf{v}||^2=1}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{x}^{(i)} - (\mathbf{v}^T \mathbf{x}^{(i)})\mathbf{v}||^2$$

Option 2: The vector that *maximizes* the **variance**

$$\mathbf{v}_1 = \underset{\mathbf{v}:||\mathbf{v}||^2=1}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} (\mathbf{v}^T \mathbf{x}^{(i)})^2$$

Subject to constraints on magnitude of $\mathbf{v}$ because otherwise a dumb solution exists!

Equivalence of Maximizing Variance and Minimizing Reconstruction Error

PCA

Claim: Minimizing the reconstruction error is equivalent to maximizing the variance.

Proof: First, note that:

$$||\mathbf{x}^{(i)} - (\mathbf{v}^T\mathbf{x}^{(i)})\mathbf{v}||^2 = ||\mathbf{x}^{(i)}||^2 - (\mathbf{v}^T\mathbf{x}^{(i)})^2 \tag{1}$$

since $\mathbf{v}^T\mathbf{v} = ||\mathbf{v}||^2 = 1$.

Substituting into the minimization problem, and removing the extraneous terms, we obtain the maximization problem.

$$\mathbf{v}^* = \underset{\mathbf{v}:||\mathbf{v}||^2=1}{\text{argmin}} \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{x}^{(i)} - (\mathbf{v}^T\mathbf{x}^{(i)})\mathbf{v}||^2 \tag{2}$$

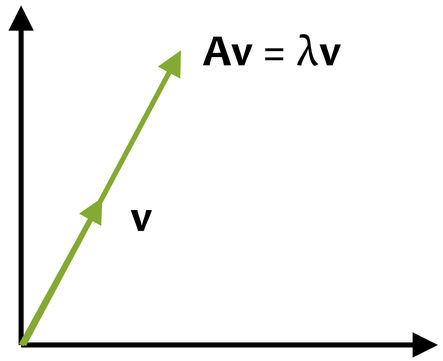
$$= \underset{\mathbf{v}:||\mathbf{v}||^2=1}{\text{argmin}} \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{x}^{(i)}||^2 - (\mathbf{v}^T\mathbf{x}^{(i)})^2 \tag{3}$$

$$= \underset{\mathbf{v}:||\mathbf{v}||^2=1}{\text{argmax}} \frac{1}{N}\sum_{i=1}^{N} (\mathbf{v}^T\mathbf{x}^{(i)})^2 \tag{4}$$

Background: Eigenvectors & Eigenvalues

For a square matrix **A** (n x n matrix), the vector **v** (n x 1 matrix) is an **eigenvector** iff there exists **eigenvalue** λ (scalar) such that:

$$\mathbf{Av} = \lambda\mathbf{v}$$



The linear transformation **A** is only stretching vector **v**.

That is, λ**v** is a *scalar multiple* of **v.**

Background: Eigenvectors & Eigenvalues

Fact #1: The eigenvectors of a **symmetric matrix** are **orthogonal** to each other.

Fact #2: The **covariance matrix Σ** is **symmetric**.

Single constraint [edit]

For the case of only one constraint and only two choice variables (as exemplified in Figure 1), consider the optimization problem

$$\begin{aligned} &\underset{x,y}{\text{maximize}} && f(x,y) \\ &\text{subject to} && g(x,y) = 0. \end{aligned}$$

(Sometimes an additive constant is shown separately rather than being included in g, in which case the constraint is written $g(x,y) = c$, as in Figure 1.) We assume that both f and g have continuous first partial derivatives. We introduce a new variable (λ) called a **Lagrange multiplier** (or **Lagrange undetermined multiplier**) and study the **Lagrange function** (or **Lagrangian** or **Lagrangian expression**) defined by

$$\mathcal{L}(x,y,\lambda) = f(x,y) + \lambda \cdot g(x,y),$$

where the λ term may be either added or subtracted. If $f(x_0,y_0)$ is a maximum of $f(x,y)$ for the original constrained problem and $\nabla g(x_0,y_0) \neq 0$, then there exists λ_0 such that (x_0,y_0,λ_0) is a *stationary point* for the Lagrange function (stationary points are those points where the first partial derivatives of $\mathcal{L}$ are zero). The assumption $\nabla g \neq 0$ is called constraint qualification. However, not all stationary points yield a solution of the original problem, as the method of Lagrange multipliers yields only a necessary condition for optimality in constrained problems.[9][10][11][12][13] Sufficient conditions for a minimum or maximum also exist, but if a particular candidate solution satisfies the sufficient conditions, it is only guaranteed that that solution is the best one *locally* – that is, it is better than any permissible nearby points. The *global* optimum can be found by comparing the values of the original objective function at the points satisfying the necessary and locally sufficient conditions.

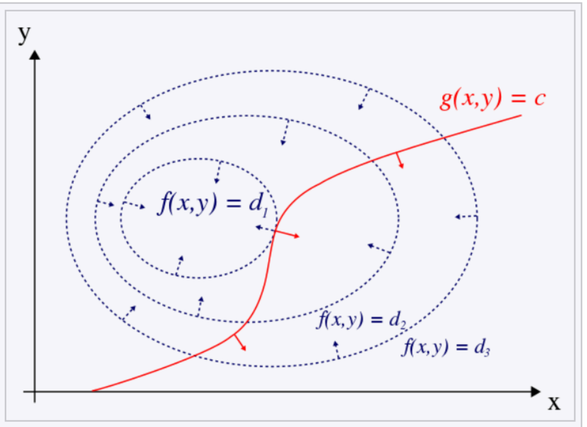


Figure 1: The red curve shows the constraint $g(x, y) = c$. The blue curves are contours of $f(x, y)$. The point where the red constraint tangentially touches a blue contour is the maximum of $f(x, y)$ along the constraint, since $d_1 > d_2$.

PCA

The First Principal Component

Claim: The vector that maximizes the variances is the eigenvector of $\mathbf{\Sigma}$ with largest eigenvalue.

Proof Sketch: To find the first principal component, we wish to solve the following constrained optimization problem (variance minimization).

$$\mathbf{v}_1 = \underset{\mathbf{v}:||\mathbf{v}||^2=1}{\operatorname{argmax}} \mathbf{v}^T \mathbf{\Sigma} \mathbf{v} \tag{1}$$

So we turn to the method of Lagrange multipliers. The Lagrangian is:

$$\mathcal{L}(\mathbf{v}, \lambda) = \mathbf{v}^T \mathbf{\Sigma} \mathbf{v} - \lambda(\mathbf{v}^T \mathbf{v} - 1) \tag{2}$$

Taking the derivative of the Lagrangian and setting to zero gives:

$$\frac{d}{d\mathbf{v}} \left(\mathbf{v}^T \mathbf{\Sigma} \mathbf{v} - \lambda(\mathbf{v}^T \mathbf{v} - 1) \right) = 0 \tag{3}$$

$$\mathbf{\Sigma} \mathbf{v} - \lambda \mathbf{v} = 0 \tag{4}$$

$$\mathbf{\Sigma} \mathbf{v} = \lambda \mathbf{v} \tag{5}$$

Recall: For a square matrix $\mathbf{A}$, the vector $\mathbf{v}$ is an **eigenvector** iff there exists **eigenvalue** λ such that:

$$\mathbf{A} \mathbf{v} = \lambda \mathbf{v} \tag{6}$$

Rewriting the objective of the maximization shows that not only will the optimal vector $\mathbf{v}_1$ be an eigenvector, it will be one with maximal eigenvalue.

$$\mathbf{v}^T \mathbf{\Sigma} \mathbf{v} = \mathbf{v}^T \lambda \mathbf{v} \tag{7}$$

$$= \lambda \mathbf{v}^T \mathbf{v} \tag{8}$$

$$= \lambda ||\mathbf{v}||^2 \tag{9}$$

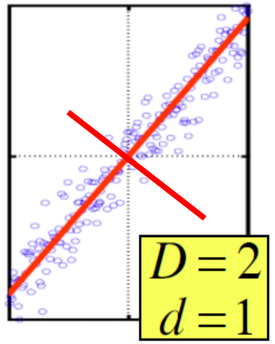
$$= \lambda \tag{10}$$

Principal Component Analysis (PCA)

$(\mathrm{X}\,\mathrm{X}^{\mathrm{T}})\mathrm{v} = \lambda\mathrm{v}$, so v (the first PC) is the eigenvector of sample correlation/covariance matrix $X\ X^T$

Sample variance of projection $\mathrm{v}^T X\ X^T\mathrm{v} = \lambda\mathrm{v}^T\mathrm{v} = \lambda$

Thus, the eigenvalue λ denotes the amount of variability captured along that dimension (aka amount of energy along that dimension).



Eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots$

- The 1st PC v_1 is the the eigenvector of the sample covariance matrix $X\ X^T$ associated with the largest eigenvalue
- The 2nd PC v_2 is the the eigenvector of the sample covariance matrix $X\ X^T$ associated with the second largest eigenvalue
- And so on …

Algorithms for PCA

Singular Value Decomposition (SVD)

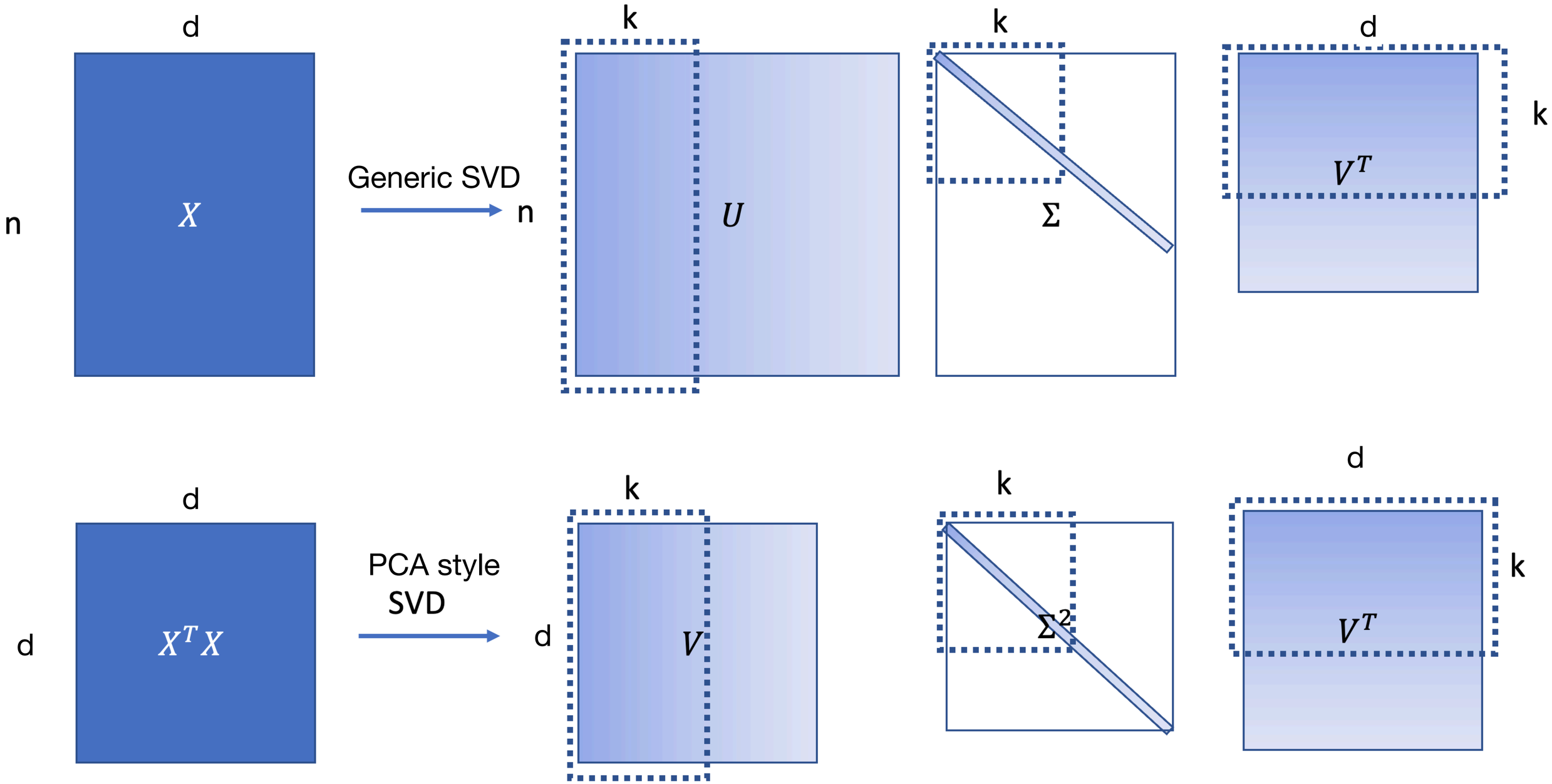
Singular value decomposition is the key part of principal components analysis.

The SVD of the $N \times p$ matrix $\mathbf{X}$ has the form $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$.

- $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N)$ is an *N × N orthogonal matrix*. $\mathbf{u}_j, j = 1, \cdots, N$, form an orthonormal basis for the space spanned by the column vectors of $\mathbf{X}$.
- $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_p)$ is an *p × p orthogonal matrix*. $\mathbf{v}_j, j = 1, \cdots, p$, form an orthonormal basis for the space spanned by the row vectors of $\mathbf{X}$.
- Σ is a *N x p rectangular matrix* with nonzero elements along the first *p x p* submatrix diagonal. $diag(s_1, s_2, \ldots s_p)$ $s_1 \geq s_2 \geq \ \ldots \ s_p \geq 0$ are the singular values of $\mathbf{X}$ with N > p.

The columns of $\mathbf{V}$ (i.e., $\mathbf{v}_j, j = 1, \cdots, p$) are the eigenvectors of $\mathbf{X}^T\mathbf{X}$. They are called *principal component direction* of $\mathbf{X}$.

The diagonal values in Σ (i.e., $s_j, j = 1, \cdots, p$) are the square roots of the eigenvalues of $\mathbf{X}^T\mathbf{X}$.


d
n
X
Generic SVD
n
k
U
k
Σ
d
k
V^T
d
d
$X^T X$
PCA style
SVD
d
k
V
k
Σ^2
d
k
V^T

Singular Value Decomposition

To generate principle components:

- Subtract mean $\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^n$ from each data point, to create zero-centered data
- Create matrix X with one row vector per (zero centered) data point
- Solve SVD: $X = USV^T$
- Output Principle components: columns of V (= rows of V^T)
  - Eigenvectors in V are sorted from largest to smallest eigenvalues
  - S is diagonal, with s_k^2 giving eigenvalue for kth eigenvector

Singular Value Decomposition

To project a point (column vector x) into PC coordinates:

$V^T x$

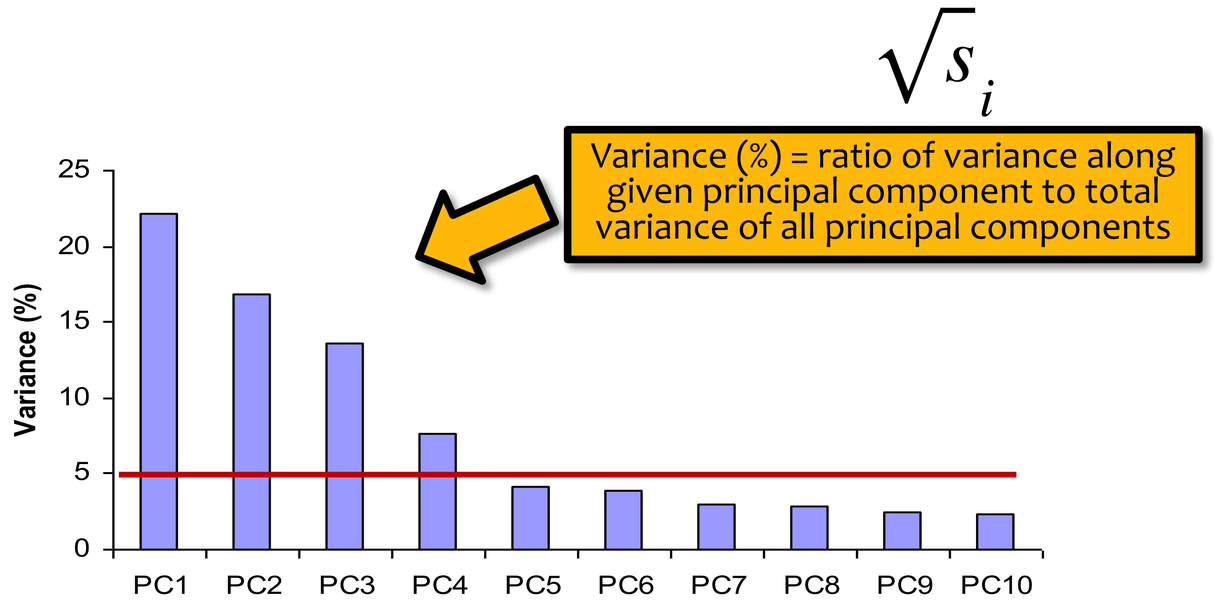
If x_i is i[th] row of data matrix X, then

- (i[th] row of US) $= V^T x_i^T$
- $(US)^T = V^T X^T$

To project a column vector x to M dim Principle Components subspace, take just the first M coordinates of $V^T x$

Dimensionality REDUCTION

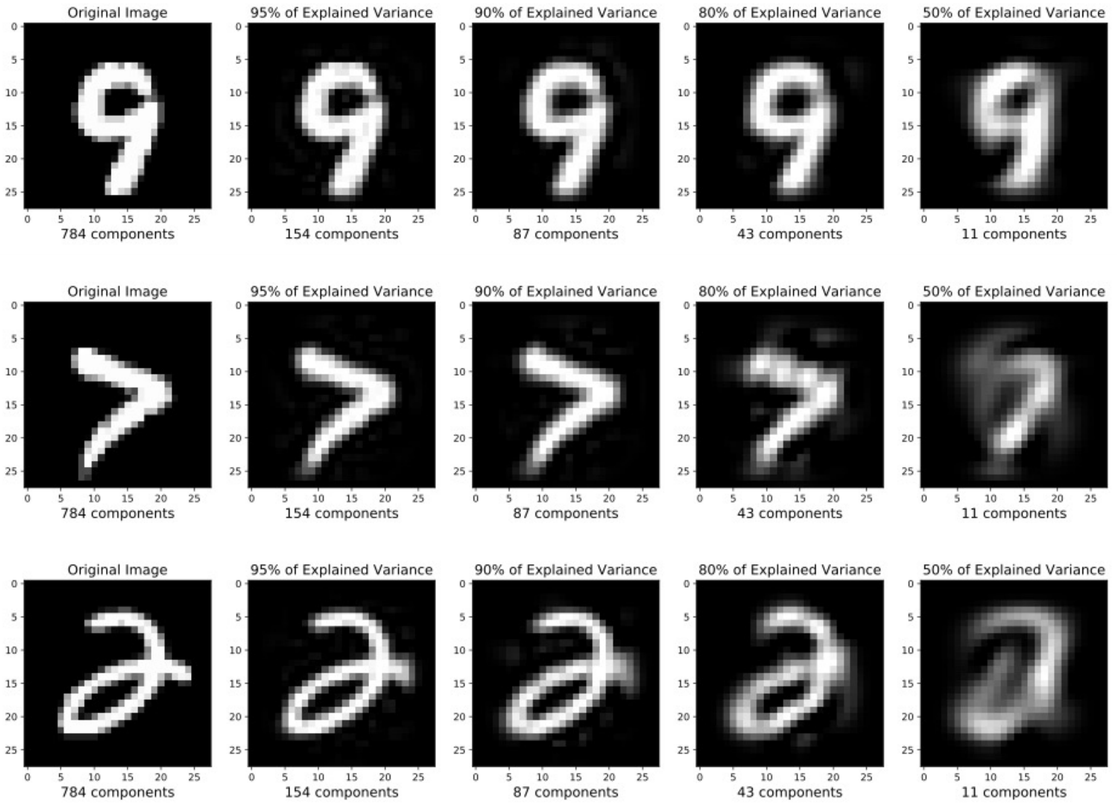
Choose k biggest eigenvectors



Projecting MNIST digits

Task Setting:

1. Take each 28x28 image of a digit (i.e. a vector $\mathbf{x}^{(i)}$ of length 784) and project it down to K components (i.e. a vector $\mathbf{u}^{(i)}$)
2. Report percent of variance explained for K components
3. Then project back up to 28x28 image (i.e. a vector $\tilde{\mathbf{x}}^{(i)}$ of length 784) to visualize how much information was preserved



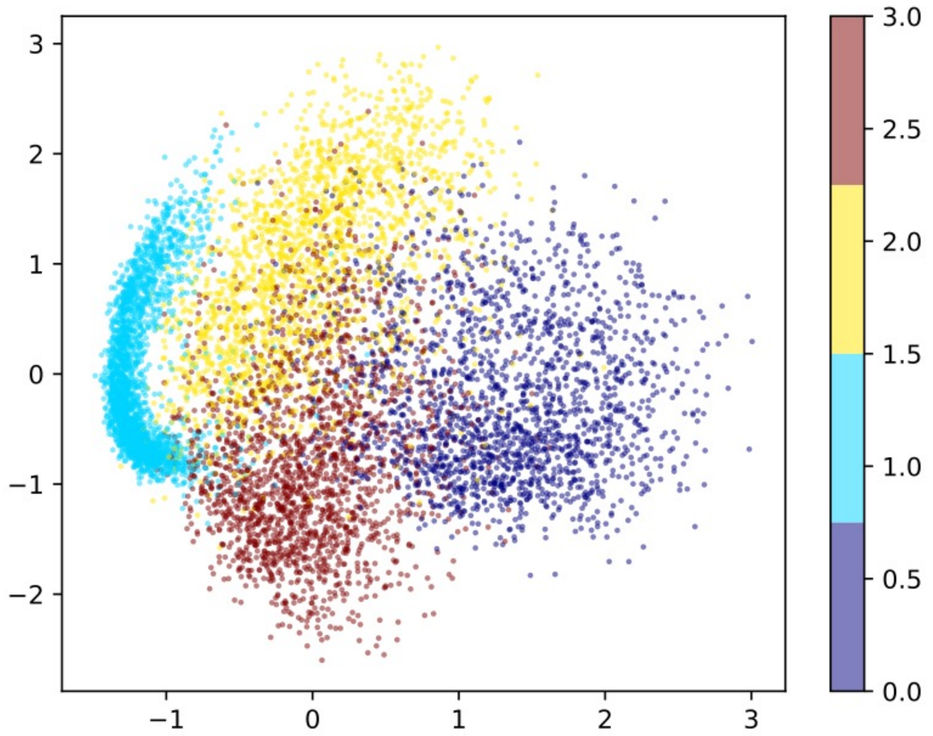
Takeaway:
Using fewer principal components K leads to higher reconstruction error.

But even a small number (say 43) still preserves a lot of information about the original image.

Projecting MNIST digits

Task Setting:

1. Take each 28x28 image of a digit (i.e. a vector $\mathbf{x}^{(i)}$ of length 784) and project it down to K=2 components (i.e. a vector $\mathbf{u}^{(i)}$)
2. Plot the 2 dimensional points $\mathbf{u}^{(i)}$ and label with the (unknown to PCA) label $y^{(i)}$ as the color
3. Here we look at just four digits 0, 1, 2, 3

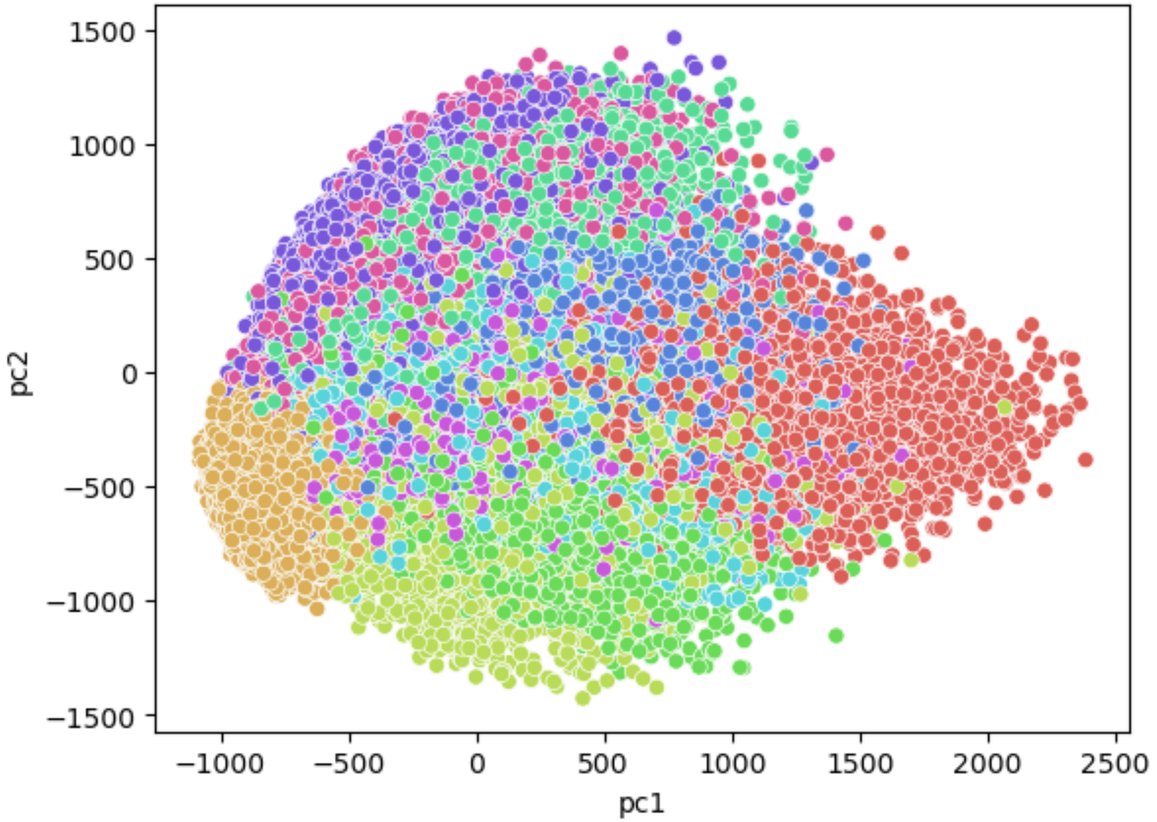


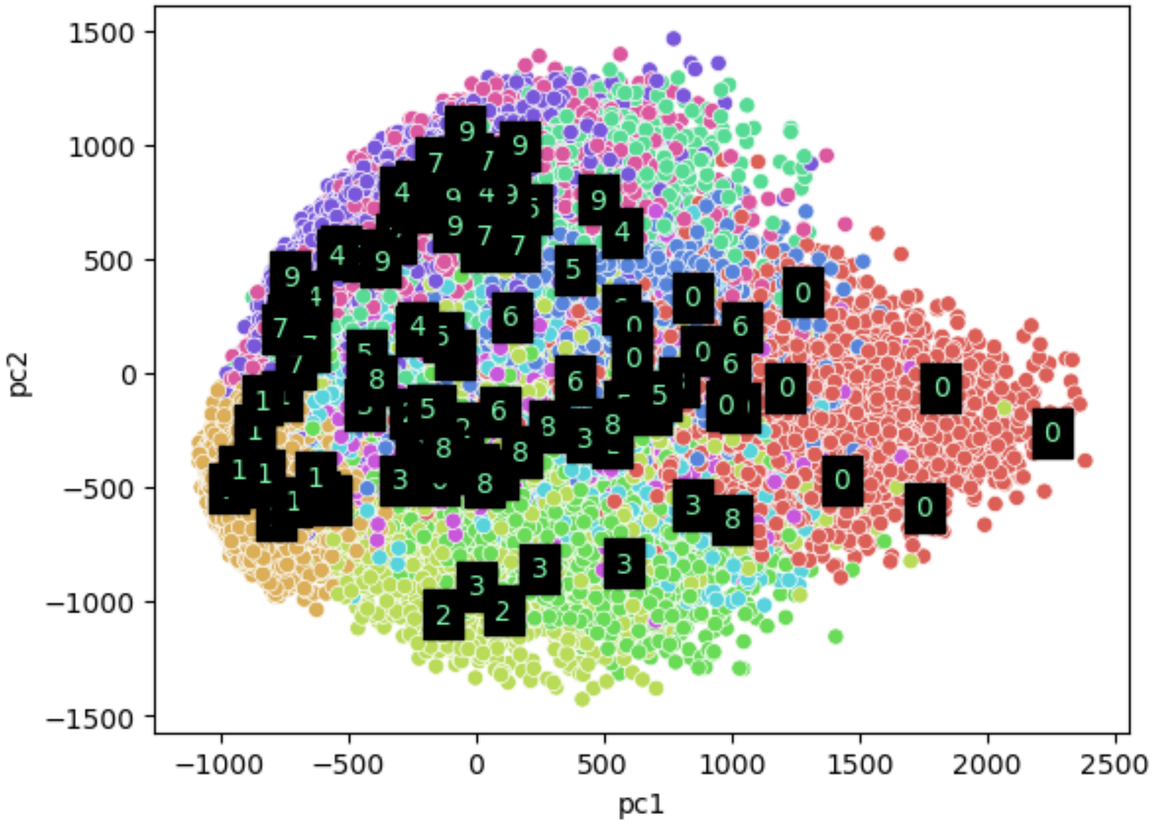
Takeaway:
Even with a tiny number of principal components K=2, PCA learns a representation that captures the *latent* information about the type of digit

Projecting MNIST digits

Task Setting:

1. Take each 28x28 image of a digit (i.e. a vector $\mathbf{x}^{(i)}$ of length 784) and project it down to K=2 components (i.e. a vector $\mathbf{u}^{(i)}$)
2. Plot the 2 dimensional points $\mathbf{u}^{(i)}$ and label with the (unknown to PCA) label $y^{(i)}$ as the color
3. Here we look at all ten digits 0 - 9

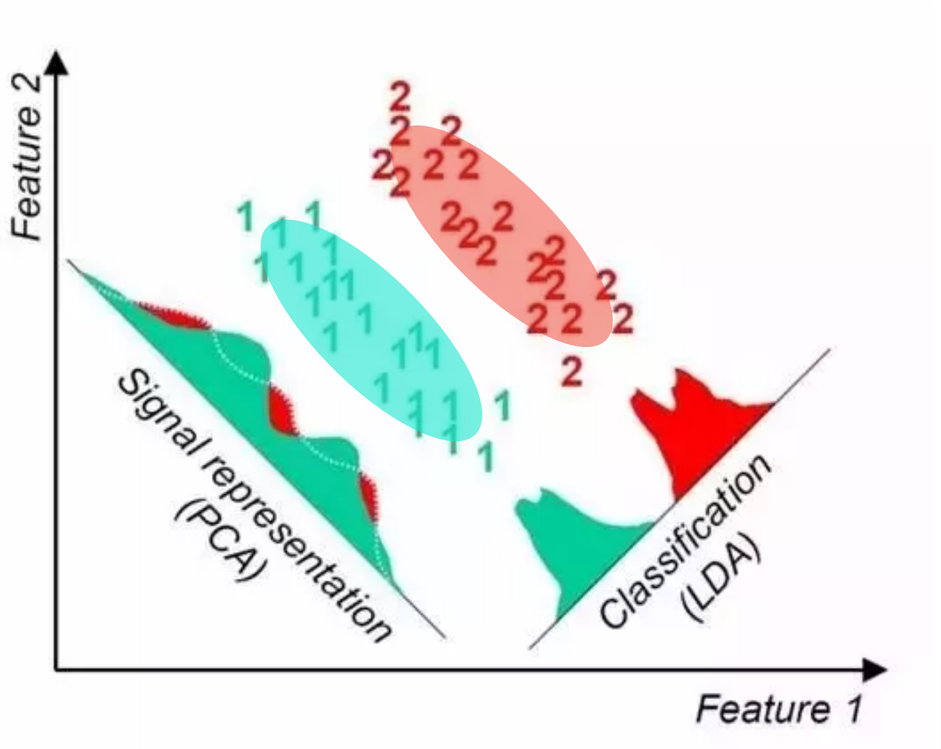




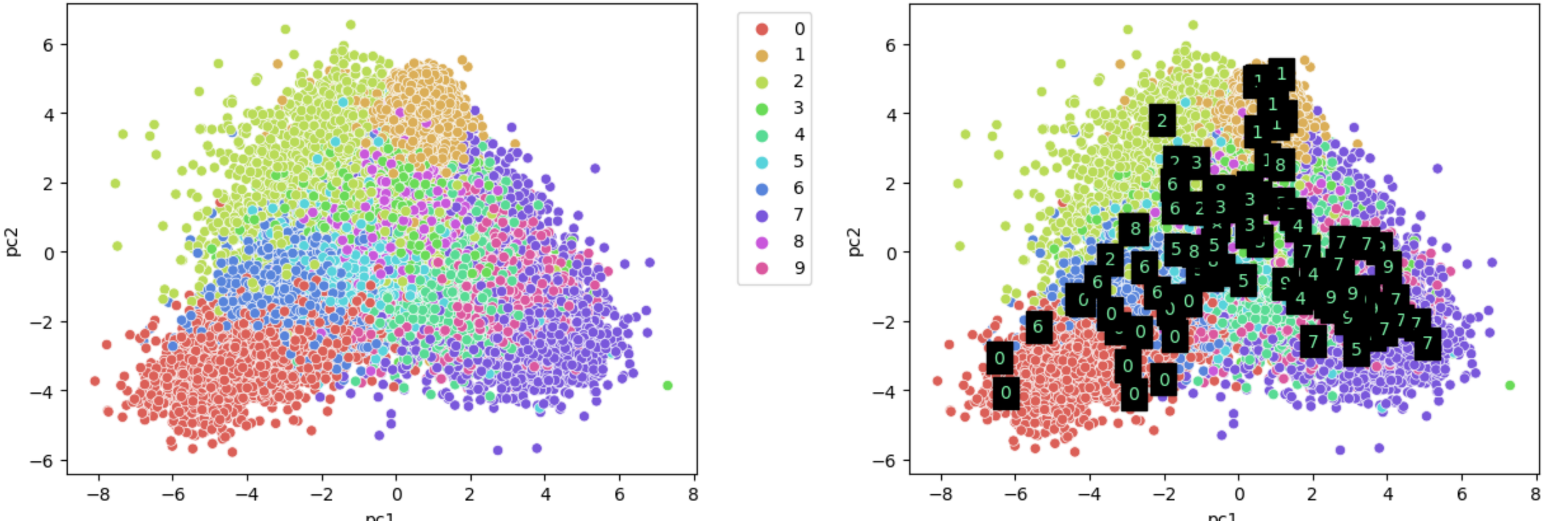
Linear Discriminant Analysis

An extension of the Fisher discriminant

- Fisher: Maximize the ratio of SSE_between over SSE_within
- Bayes: Maximize class posterior probability with pooled covariance across classes but different prior probabilities per class
- Very similar to a GMM with tied covariance
- Dimensionality reduction: the direction that best separates classes is the one where there's the most standard deviations between the class centers

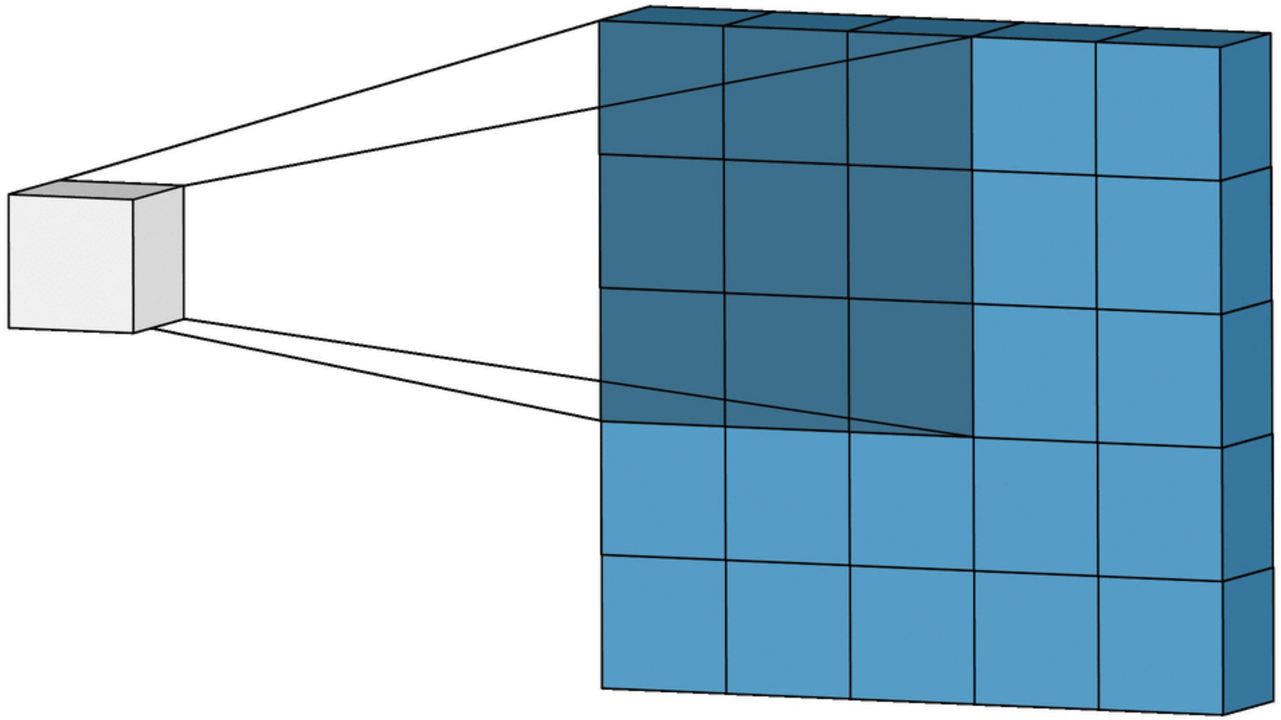


Projecting MNIST using LDA



Common kinds of kernels

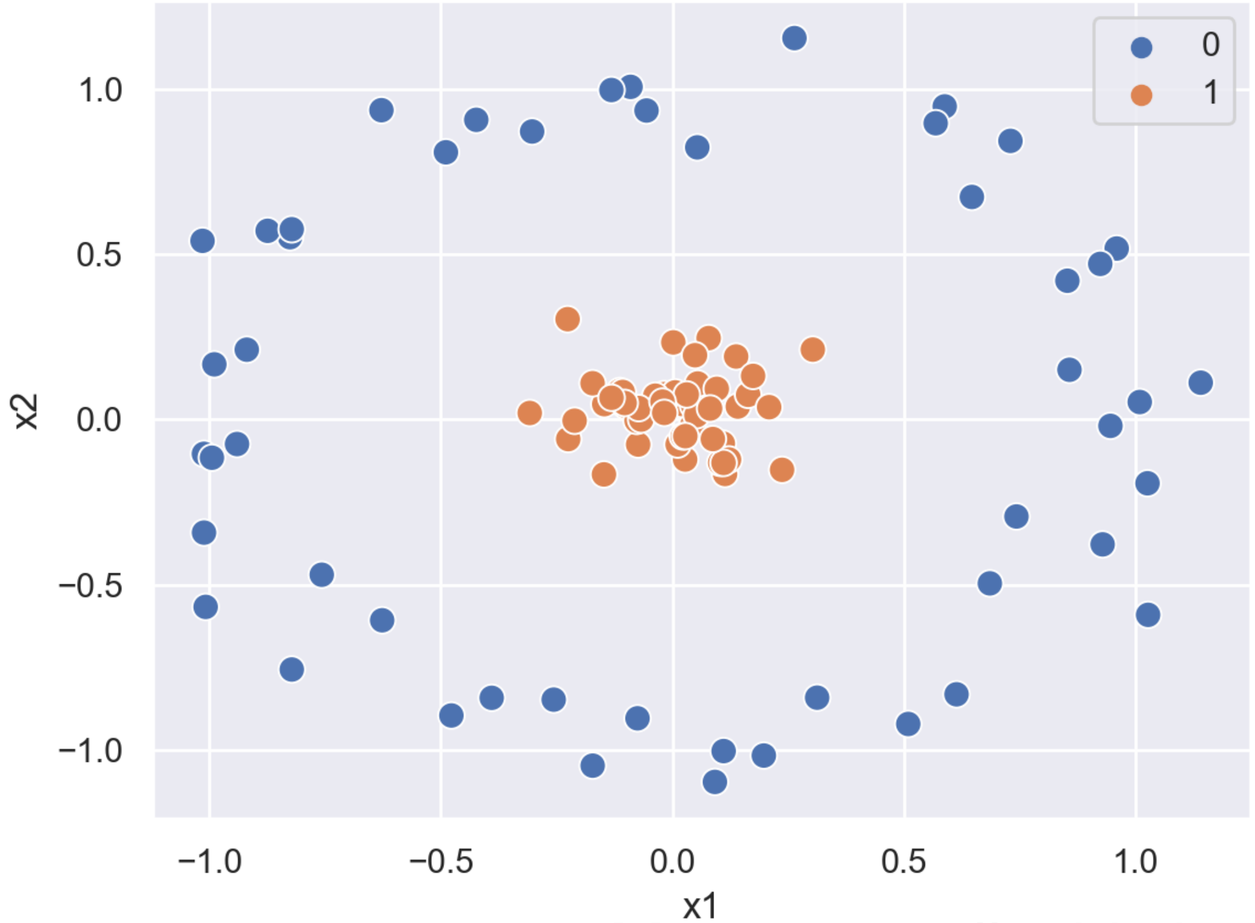
- Moving average window
- Polynomial of order d
- Radial basis / Gaussian
- Sigmoid / tanh

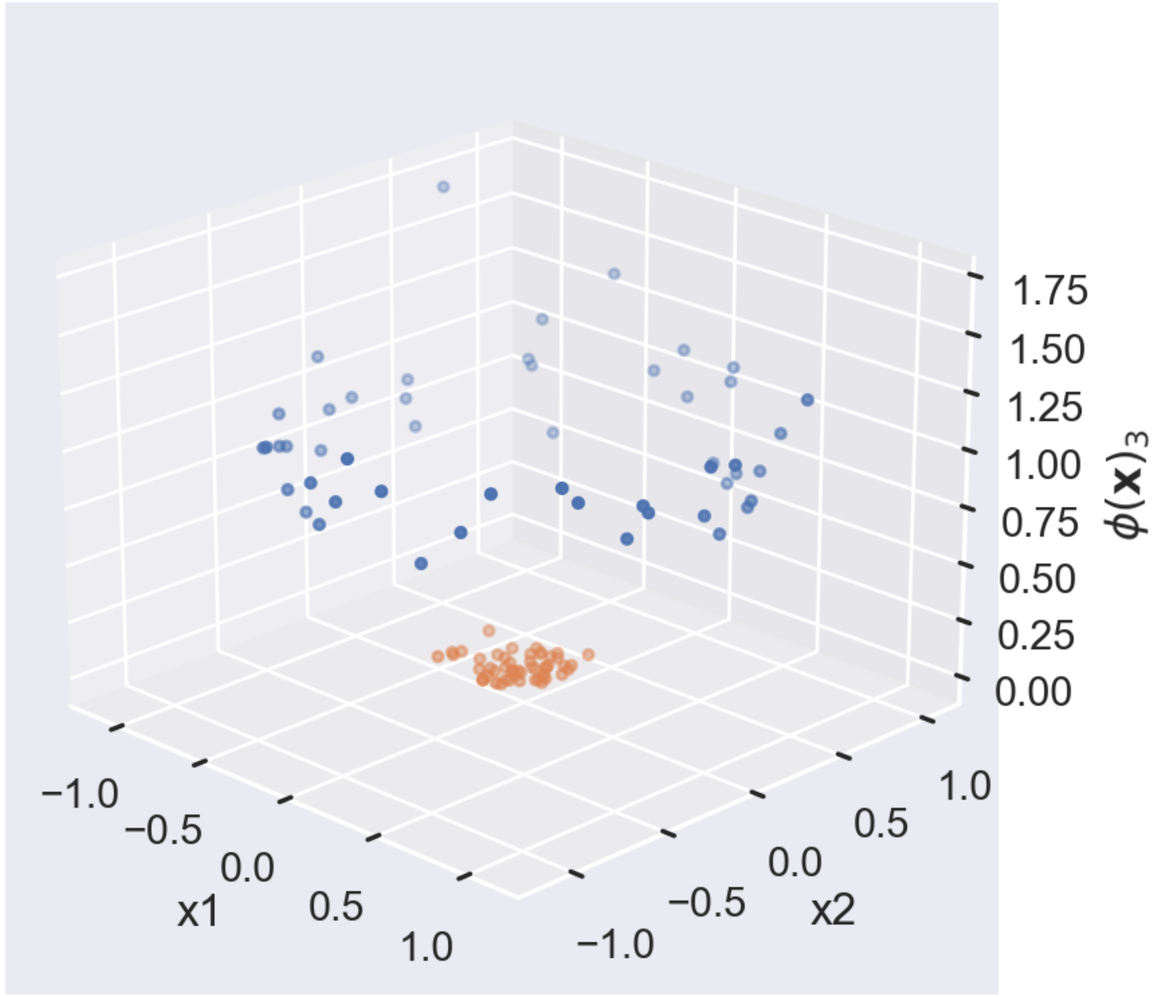


Kernel PCA

- Transform the data via a kernel, then PCA it, duh.
- Formally: what's a kernel?
  - A function $\Phi(x) : \mathbb{R}^d \mapsto \mathbb{R}^m$ where in general $m > d$
  - Can be expressed as a dot product $K(x, x') = \langle \Phi(x)^\top \Phi(x') \rangle_{\mathbb{R}^d}$

$$\phi(\mathbf{x}) = \left[x_1,\ x_2,\ x_1 \cdot x_2 + x_1^2 \cdot x_2^2\right]$$





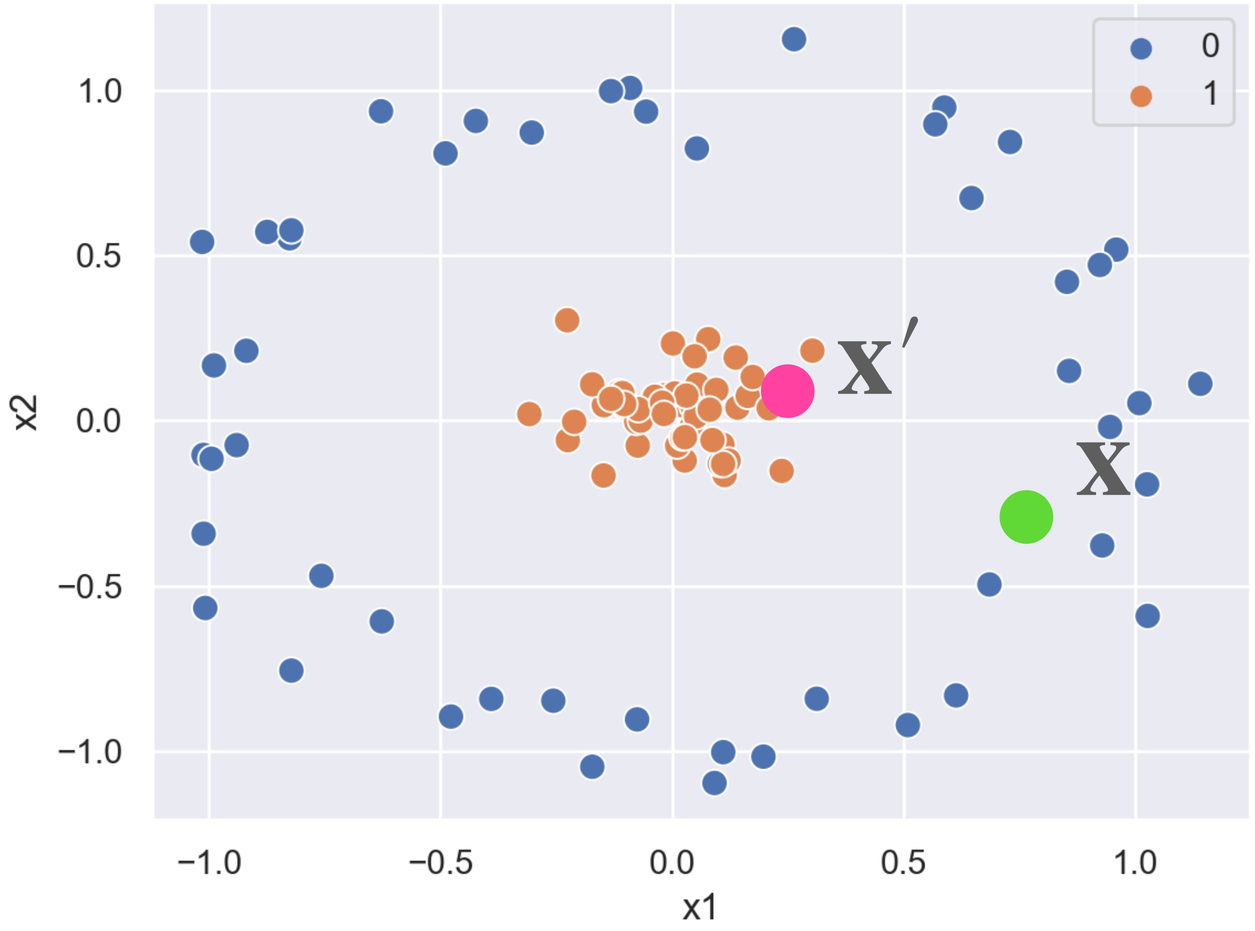
More generally this is the polynomial kernel

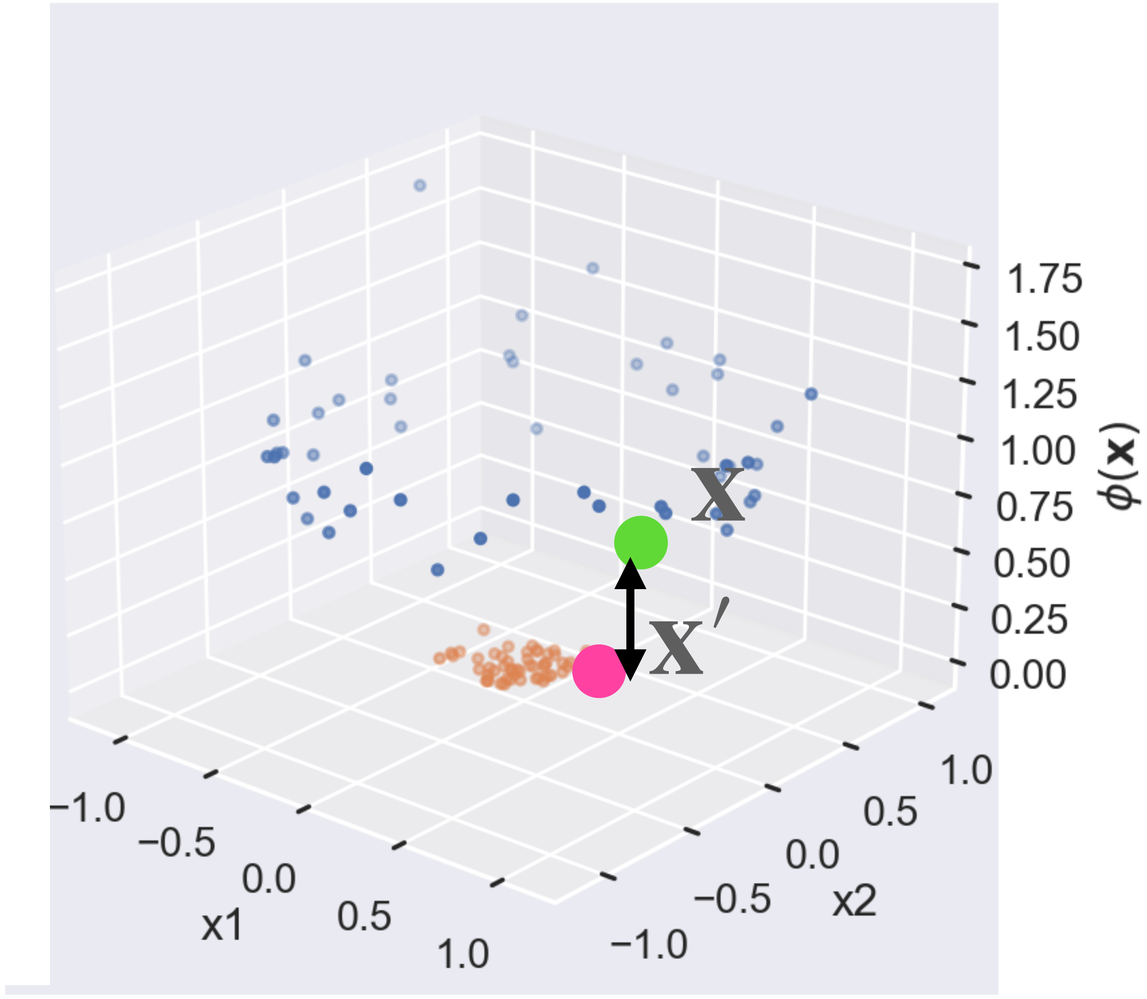
$$k(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + c\right)^d = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$$

$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ means you never have to calculate

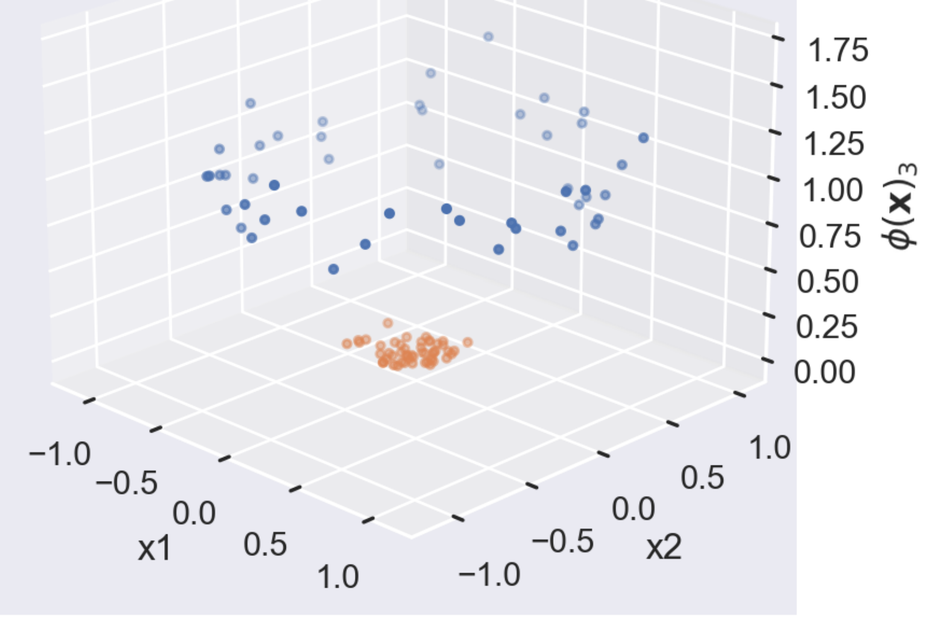
the decision boundary in either $\mathbb{R}^d$ or $\mathbb{R}^m$…

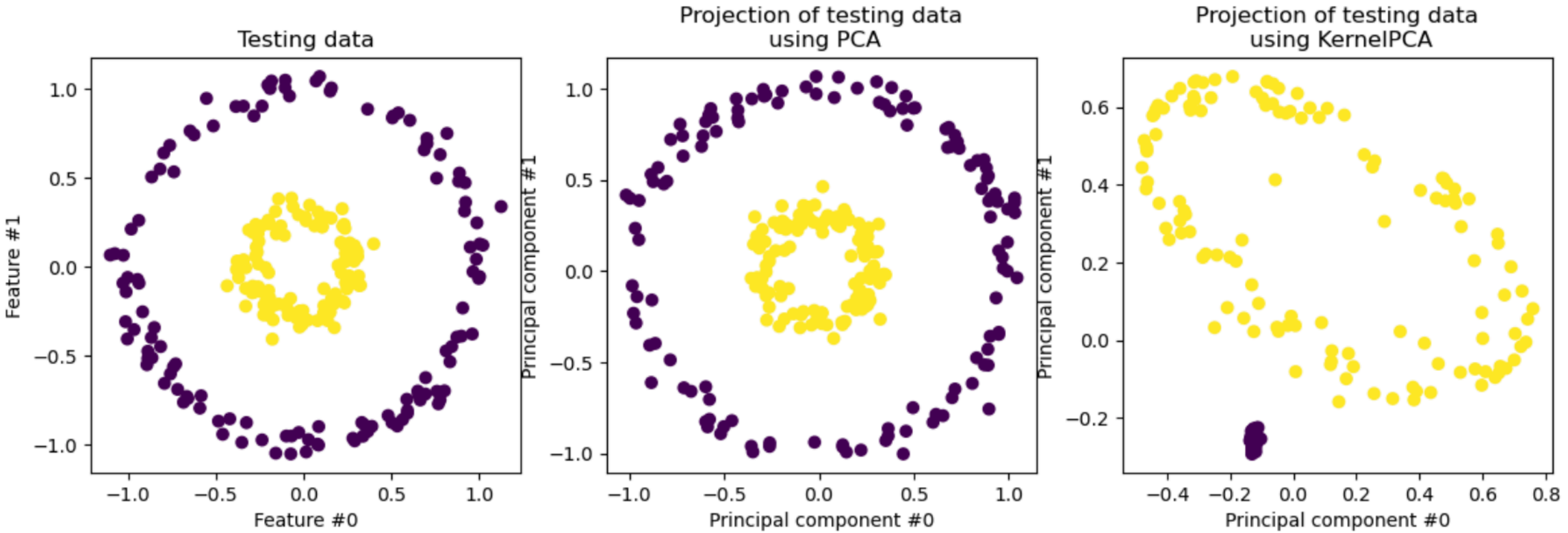
its in terms of a scalar dot product of vectors in $\mathbb{R}^d$!





- Transform the data via a kernel, then PCA it, duh.





Some ML from scratch notebooks

on dimensionality reduction

- https://sebastianraschka.com/notebooks/ml-notebooks/
  - https://nbviewer.org/github/rasbt/pattern_classification/blob/master/dimensionality_reduction/projection/principal_component_analysis.ipynb
  - https://nbviewer.org/github/rasbt/pattern_classification/blob/master/dimensionality_reduction/projection/linear_discriminant_analysis.ipynb
  - https://nbviewer.org/github/rasbt/pattern_classification/blob/master/dimensionality_reduction/projection/kernel_pca.ipynb