Dimensionality Reduction

Part 1: Principal Component Analysis

Moses Marsh

galvanize

Objectives: answer the following

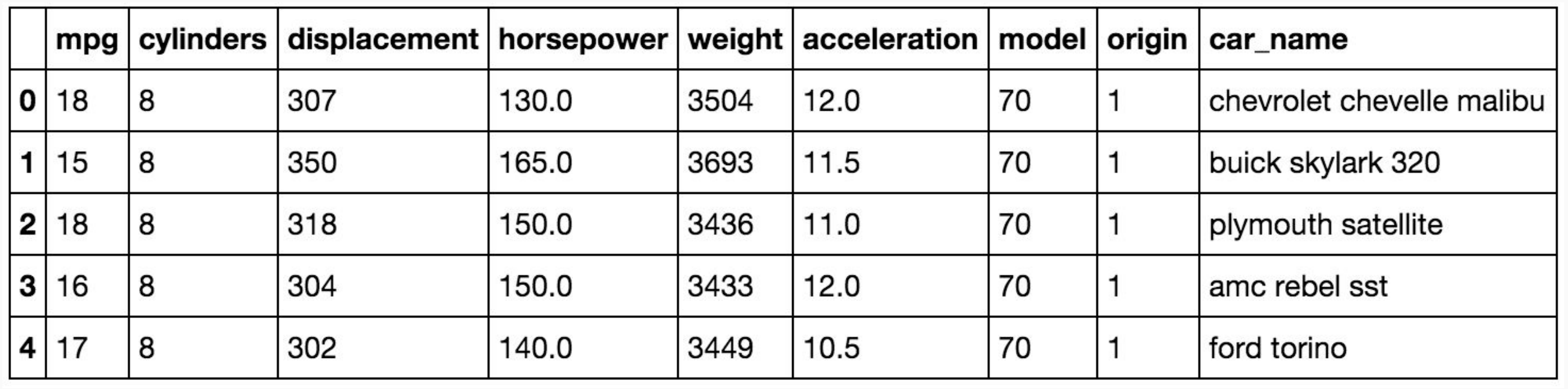
- Why reduce dimensionality?
- What are "principal components"?
- How is PCA used for dimensionality reduction?

Before we go on:

What is the *dimensionality* of our data?

Here are some different words for the same thing…

- “dimension” = “feature” = “predictor” = “covariates”
- “dimensionality” = “number of features” = “number of predictors”



	mpg	cylinders	displacement	horsepower	weight	acceleration	model	origin	car_name
0	18	8	307	130.0	3504	12.0	70	1	chevrolet chevelle malibu
1	15	8	350	165.0	3693	11.5	70	1	buick skylark 320
2	18	8	318	150.0	3436	11.0	70	1	plymouth satellite
3	16	8	304	150.0	3433	12.0	70	1	amc rebel sst
4	17	8	302	140.0	3449	10.5	70	1	ford torino

Before we go on:
What is the *dimensionality* of our data?

- MNIST handwritten digits dataset: each grayscale image is 28x28 pixels

Before we go on:

What is the *dimensionality* of our data?

- MNIST handwritten digits dataset: each grayscale image is 28x28 pixels

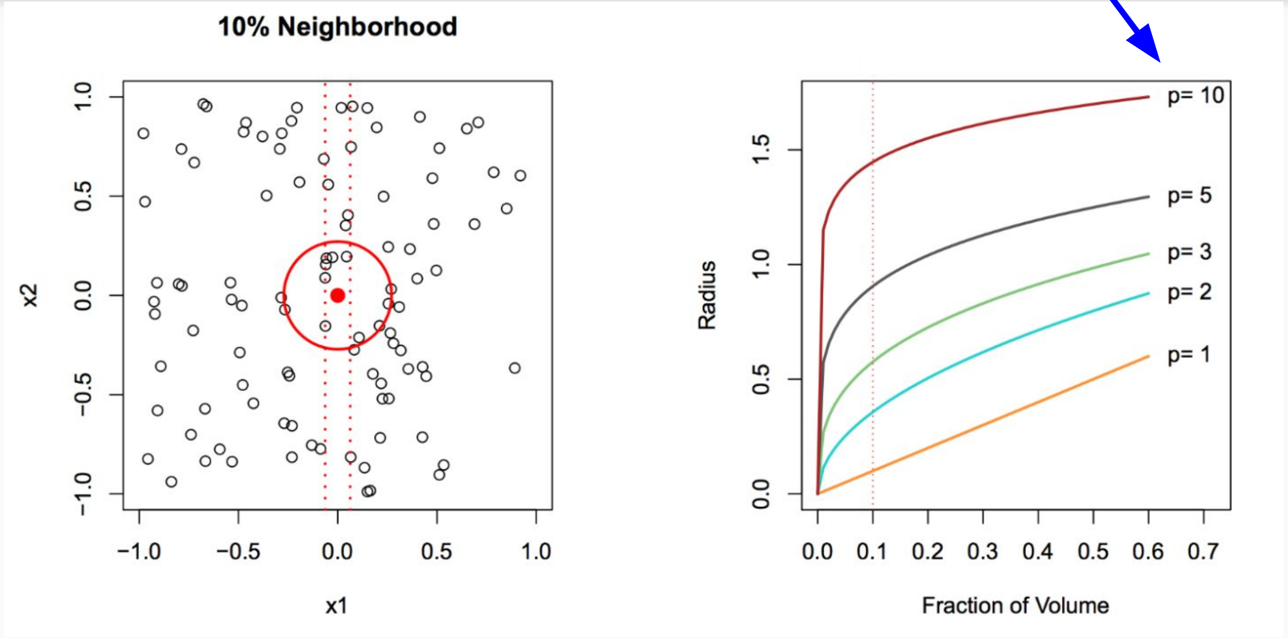
Each pixel is a feature, so each image is a 784-dimensional data point

Why reduce dimensionality?

- Combat the **Curse Of Dimensionality**
  - Remember how that ruined kNN and clustering?
- Facilitate **Visualization**
  - Most people only want to see 2 or 3 dimensions at a time
- Combine many raw features into a few meaningful **latent features**
  - More about this in the afternoon! And tomorrow!
- Remove **Correlated Features**
  - If you have **p** features, then you have **p(p-1)/2** pairs of features. If p=10,000, that's a lot. Many of those pairs will be correlated.
- Compress your data

Review: The Curse of Dimensionality

p = dimensionality



When p=1, we are only considering x1. When p=2, we are considering x1 and x2.

Notice the required radius in 2D is much larger than the required radius in 1D.

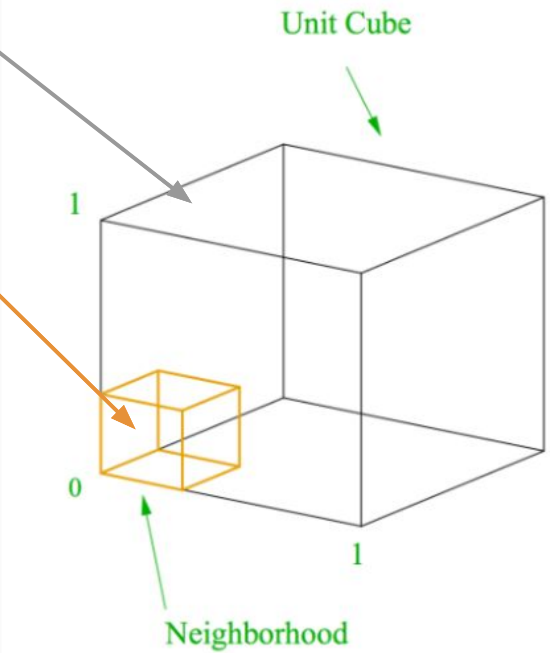
As we increase the dimensionality, we lose the concept of locality.

Review: The Curse of Dimensionality

Say we have a unit (hyper)cube.

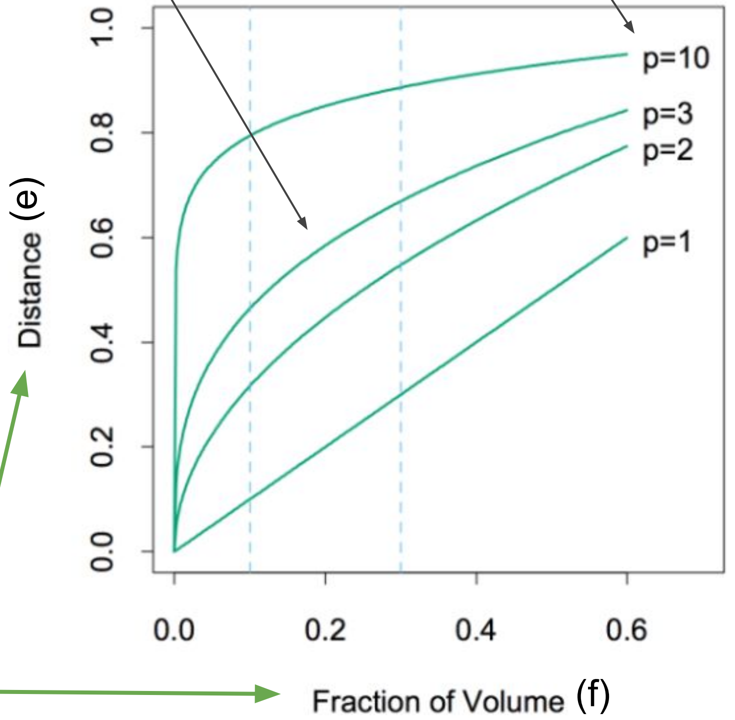
We want to create another (hyper)cube inside the outer cube so that we fill X% of the outer cube.

How long must the edges of the inner cube be?



$$e = f^{1/p}$$

p = dimensionality



Methods for reducing dimensionality

There are a number of ways to reduce the dimensionality of your dataset:

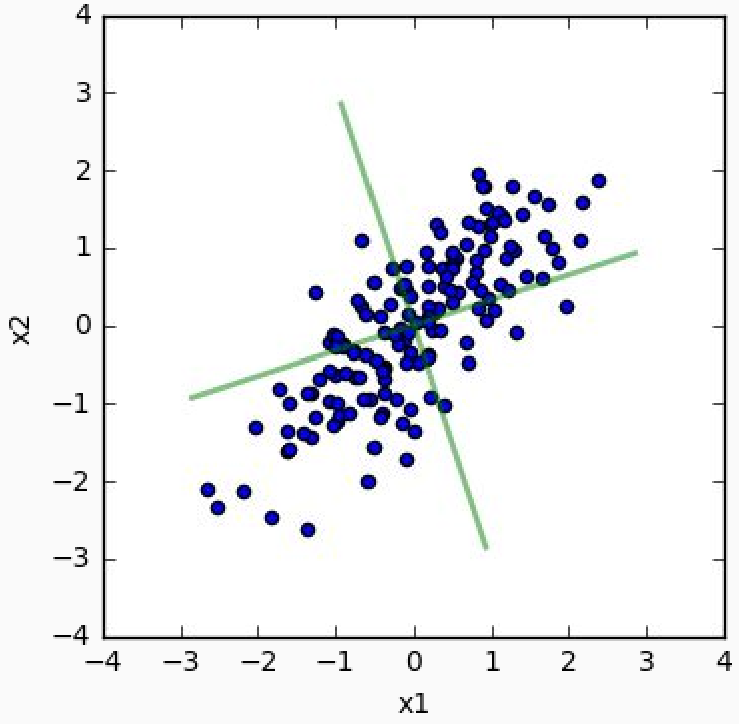
- Subset selection of features; e.g. forward stepwise selection
- LASSO regression
- Relaxed LASSO:
  - (1) do LASSO regression, (2) throw away unused features, (3) do OLS regression
- Neural networks:
  - (black box)
- **Principal Components Analysis (PCA)**

Review: Transformation Matrices

- Let X be a data set as a feature matrix, each row a data point
- Let A be a matrix of the form

$$A = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

Then **X' = XA** represents the data as coordinates along a ***rotated*** set of axes. The new axes, x1' & x2' are shown in green.



Individual

1. On your board, work out the following using:

$$A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

   a. Given points $x_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}; x_2 = \begin{bmatrix} 1 & 0 \end{bmatrix}; x_3 = \begin{bmatrix} 1 & 1 \end{bmatrix}$

      Plot x_1, x_2, x_3 on your whiteboard.

   b. Calculate $x_1' = x_1 A; x_2' = x_2 A; x_3' = x_3 A$

      Plot x_1', x_2', x_3' on a new plot.

   c. What has A done to the data points?

   d. What are the axes of your new plot?

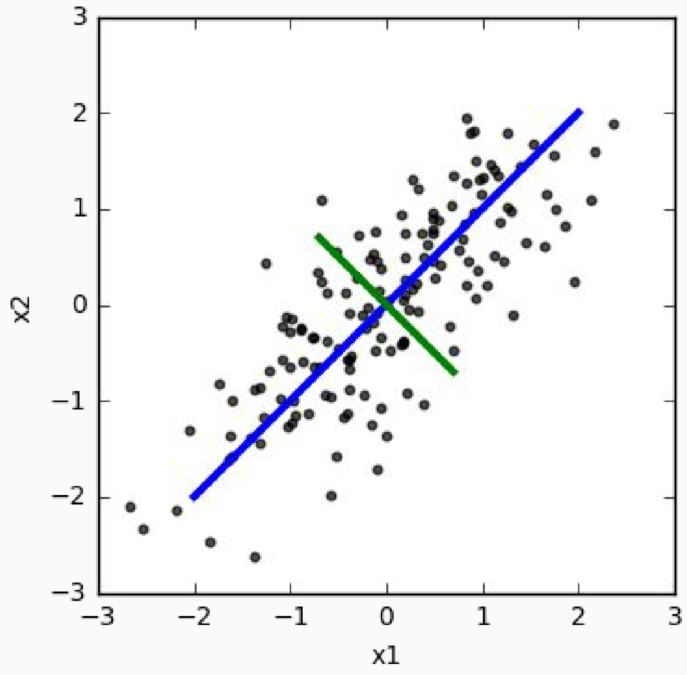
Pair Discussion

# hours at sea	# nets deployed	# fish caught
2	3	5000
1	1	1000
4	10	7000

1. If we want to predict number of fish caught, how do we split this data into X and y?
2. If we apply a rotation of 45 degrees to the X matrix, what is the resulting first row?
3. Does this rotation affect the number of fish caught?
4. Can we still use the X matrix after rotation for predicting number of fish caught?
5. Why might we want to apply this transformation?

Principal Component Analysis (PCA)

- PCA is simply a **rotation in feature space**: its goal is to construct a set of **transformed features** that have **no covariance**
- The **first principal component** points along the direction of **highest variance**, the second PC points along the next highest-variance direction orthogonal to the first, and so on.
- The **dimensionality reduction** comes in when we drop components that capture very little variance



Review: Covariance

galvanize

Raw data
(**n** rows ,
p columns)

$$X = \begin{bmatrix} 10 & 3 \\ 10 & 4 \\ 40 & 7 \\ 60 & 6 \\ 70 & 9 \\ 100 & 7 \\ 100 & 8 \end{bmatrix}$$

Standardized data
(each feature has
mean=0, std=1)

$$M = \begin{bmatrix} -1.306 & -1.660 \\ -1.306 & -1.155 \\ -0.449 & 0.361 \\ 0.122 & -0.144 \\ 0.408 & 1.371 \\ 1.266 & 0.361 \\ 1.266 & 0.866 \end{bmatrix}$$

Covariance Matrix

$$C = \frac{1}{n} M^T M =$$

$$\begin{bmatrix} 1.0 & 0.801 \\ 0.801 & 1.0 \end{bmatrix}$$

Review: Covariance

$$Cov(X_1, X_2) = E[(X_1 - \overline{X_1})(X_2 - \overline{X_2})]$$

$$\overline{X_1} = \overline{X_2} = 0$$

$$Cov(X_1, X_2) = E[X_1 X_2] = \frac{1}{n}\sum_{i=1}^{n} x_{i1}x_{i2} = (X^T X)_{1,2}$$

$$= \sigma_{12}$$

$$C = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$$

PCA: Derivation

- The goal is to find a transformation **V** (a matrix of orthonormal basis vectors) such that the covariance matrix of the transformed features is **diagonal**

$$M' = MV$$

$$C' = \frac{1}{n} M'^T M' = \frac{1}{n} (MV)^T (MV)$$

$$(MV)^T(MV) = V^T M^T M V = \begin{bmatrix} \lambda_1 & 0 & 0 & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots & 0 \\ 0 & 0 & \lambda_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \lambda_p \end{bmatrix}$$

PCA: Derivation

- The punchline: **V** is the matrix of eigenvectors of the covariance matrix

$$V = \begin{bmatrix} | & | & | & \cdots & | \\ u_1 & u_2 & u_3 & \cdots & u_p \\ | & | & | & \cdots & | \end{bmatrix}$$

$$V^T V = \begin{bmatrix} - & u_1 & - \\ - & u_2 & - \\ - & u_3 & - \\ - & \vdots & - \\ - & u_p & - \end{bmatrix} \begin{bmatrix} | & | & | & \cdots & | \\ u_1 & u_2 & u_3 & \cdots & u_p \\ | & | & | & \cdots & | \end{bmatrix}$$

$$V^T V = \begin{bmatrix} u_1 \cdot u_1 & u_1 \cdot u_2 & \cdots & u_1 \cdot u_p \\ u_2 \cdot u_1 & u_2 \cdot u_2 & \cdots & u_2 \cdot u_p \\ \vdots & \vdots & \ddots & \vdots \\ u_p \cdot u_1 & u_p \cdot u_2 & \cdots & u_p \cdot u_p \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

PCA: Derivation

- The punchline: **V** is the matrix of eigenvectors of the covariance matrix

$$V^T M^T M V = \begin{bmatrix} \lambda_1 & 0 & 0 & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots & 0 \\ 0 & 0 & \lambda_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \lambda_p \end{bmatrix}$$

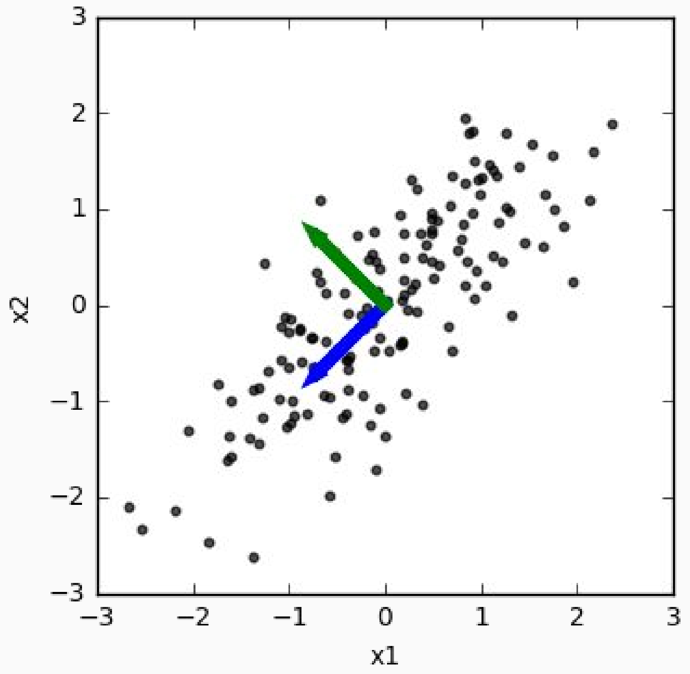
$$M^T M V = V \begin{bmatrix} \lambda_1 & 0 & 0 & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots & 0 \\ 0 & 0 & \lambda_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \lambda_p \end{bmatrix}$$

$$= \begin{bmatrix} | & | & | & \dots & | \\ u_1 & u_2 & u_3 & \dots & u_p \\ | & | & | & \dots & | \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots & 0 \\ 0 & 0 & \lambda_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \lambda_p \end{bmatrix}$$

$$M^T M u_i = \lambda_i u_i$$

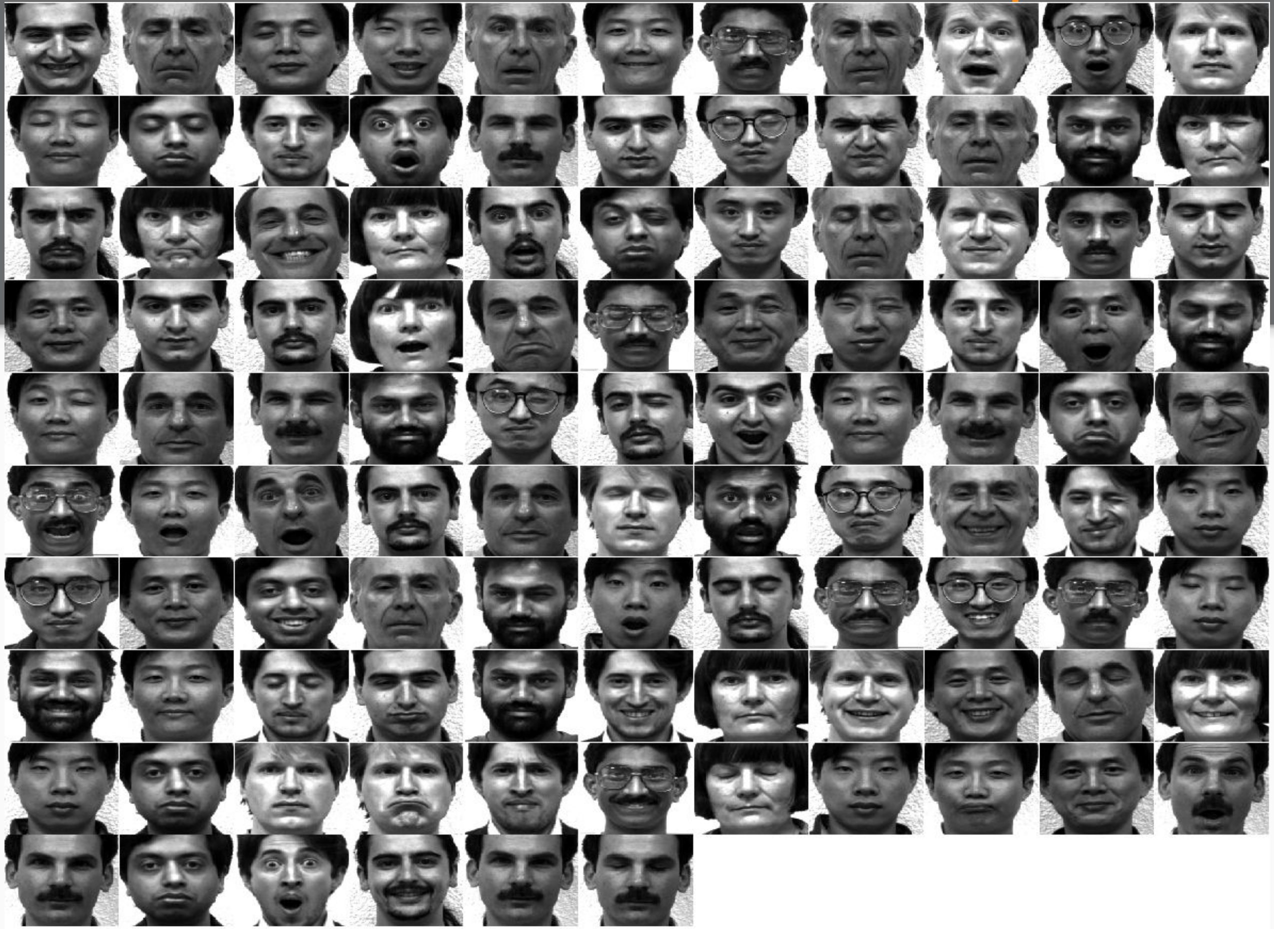
PCA: Derivation

- The eigenvectors are **directions in feature space** (i.e., linear combinations of the original features), and the eigenvalues are the **amount of variance** captured along that direction



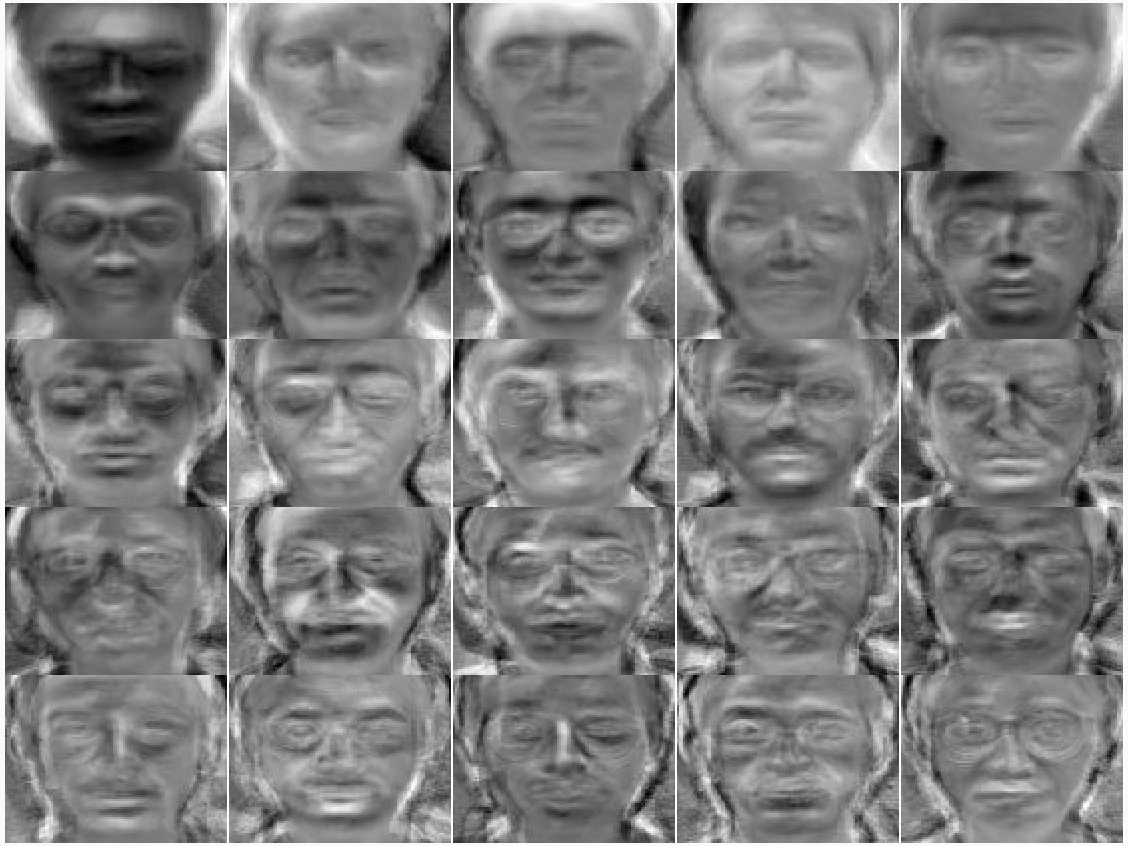
Example: Faces

- From the *Yale Face Database* (using subset of 105 images)
- 320x243 pixels each, grayscale
- Centered and cropped identically



PCA your face

- These are the top 25 **eigenfaces** representing directions in face-space



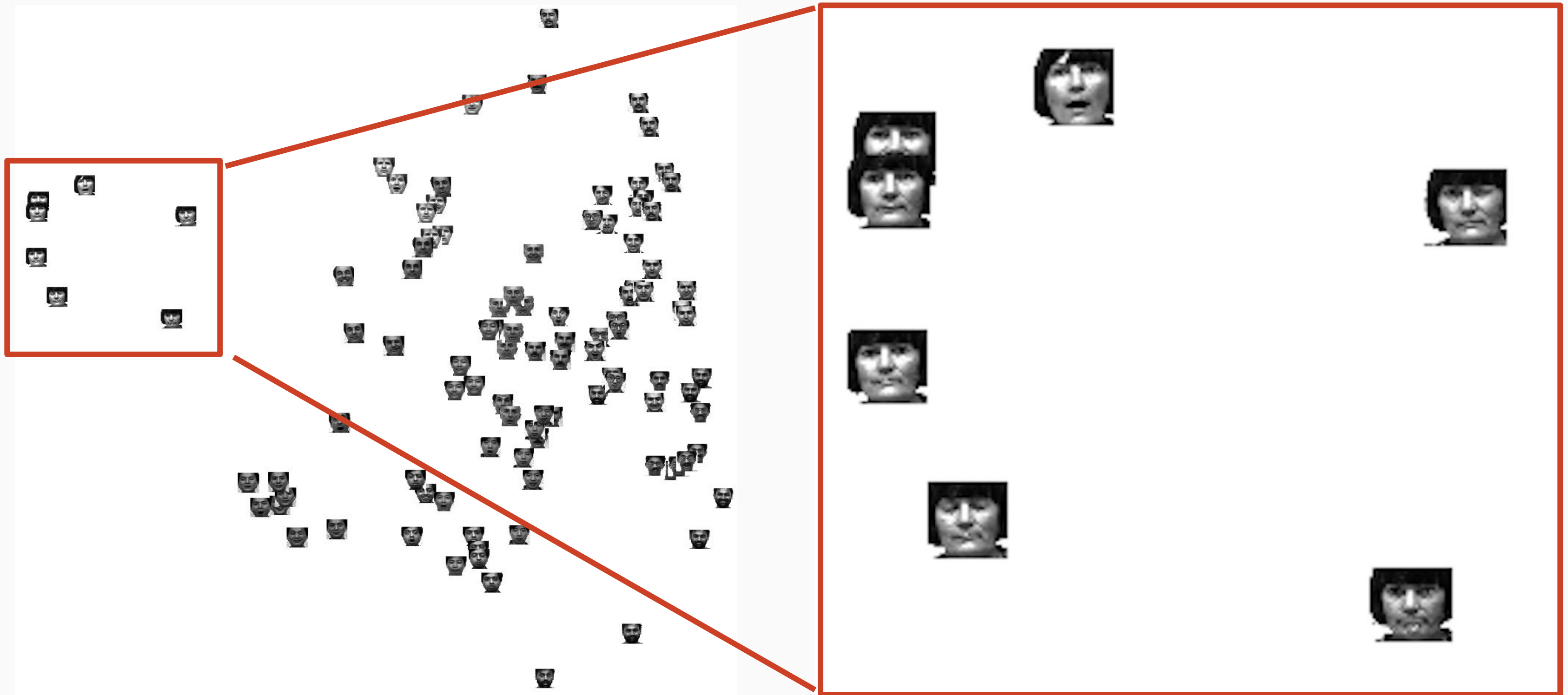
Drawbacks of Eigenfaces

- Faces must be aligned eyes-to-eyes, mouth-to-mouth -- differences in translation and scale are captured by PCA (which isn't what we want)
- Faces must be lit the same -- differences in lighting are captured by PCA (which isn't what we want).
- Old method: 1987, 1991

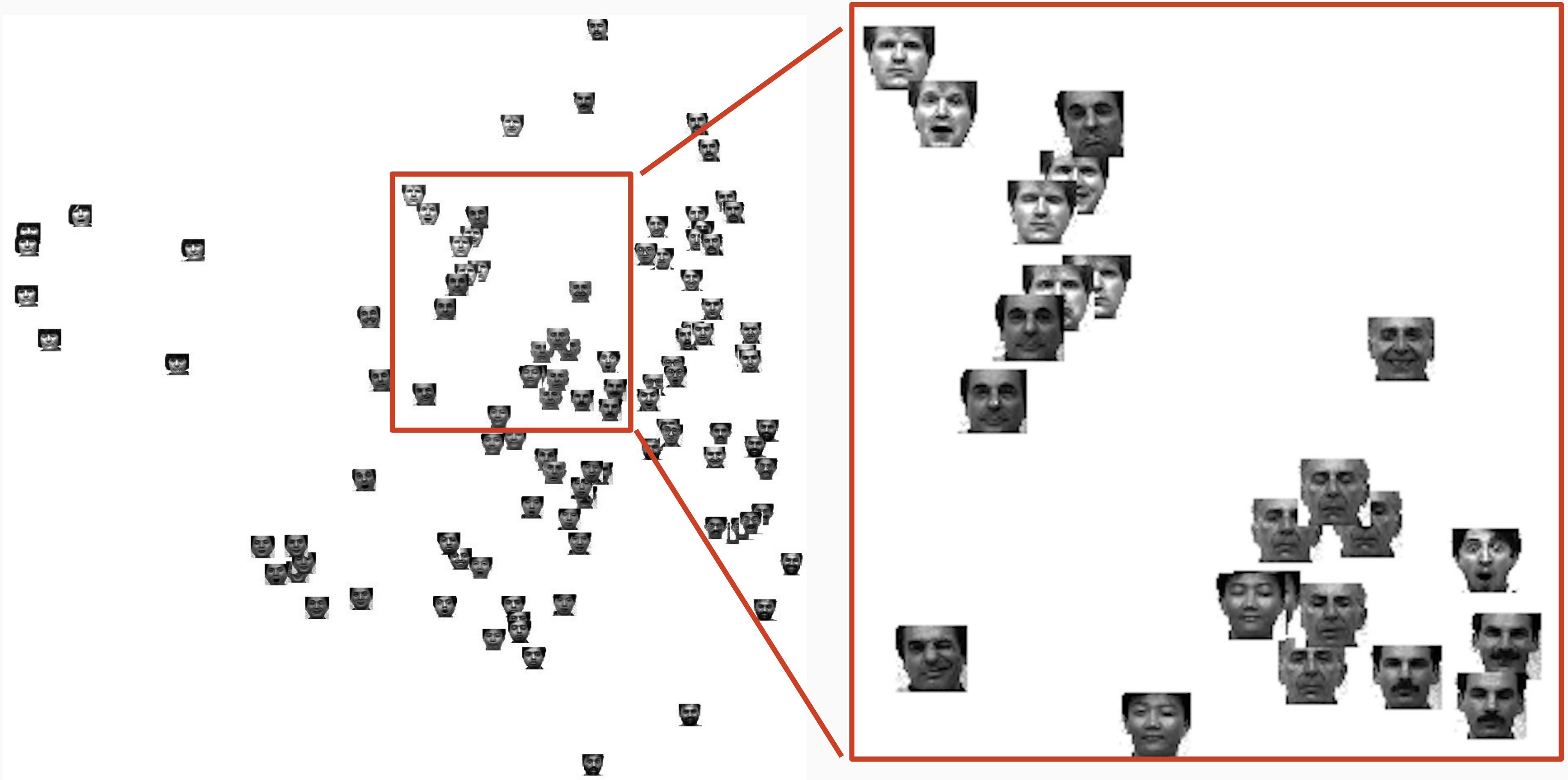
- Improvements:
  - "Fisherfaces": uses LDA and labels to help remove lighting effects
  - Use PCA on shapes detected in the image (search for Active Shape Model)

Crude facial recognition

Using PCA on cropped face images (i.e. eigenfaces), combined with kNN, we can do very crude facial recognition! To show this, let's look at the embedding onto 2d.



Crude facial recognition

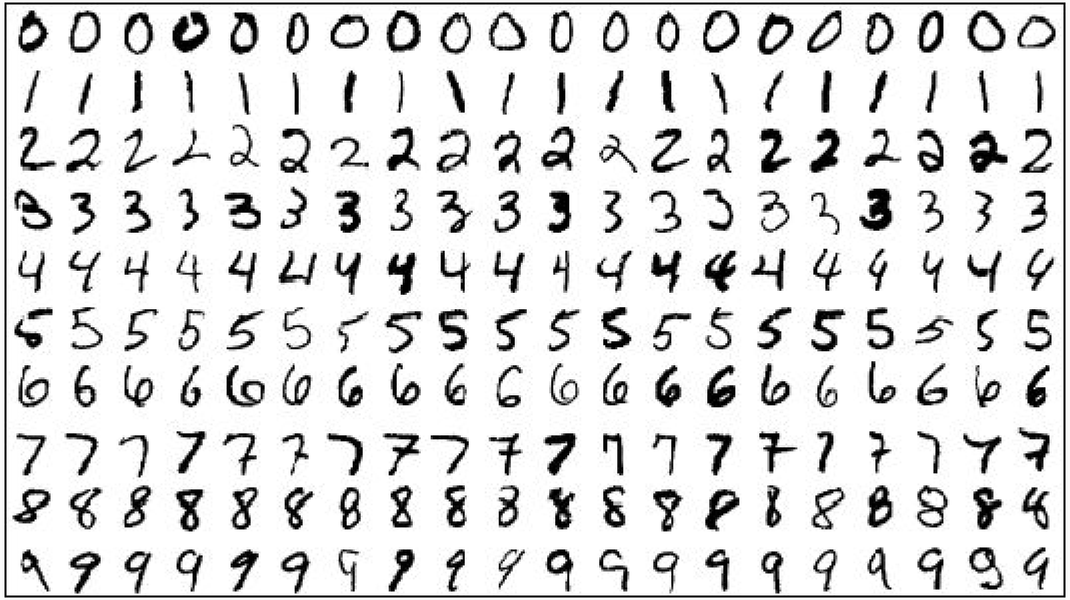


MNIST Dataset

- Dataset of handwritten digits
- 10 classes (0-9)

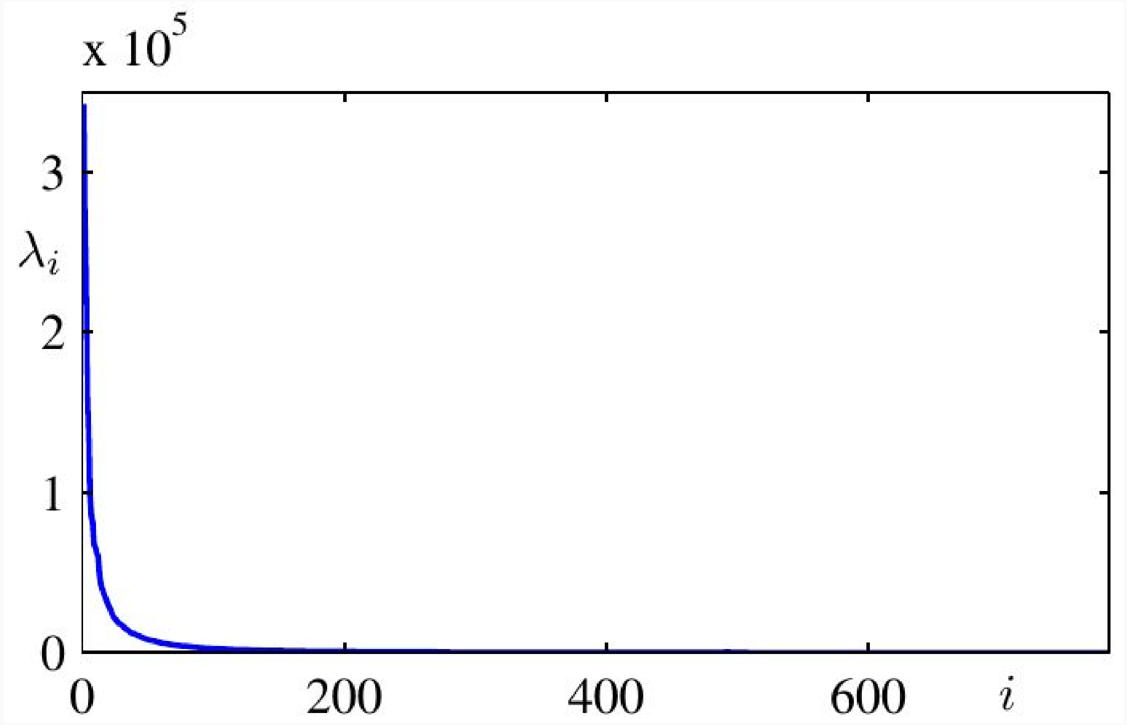
- 28x28 pixels, grayscale
  = 784 dimensions

- 60,000 training images
- 10,000 test images



PCA on MNIST: Scree Plot

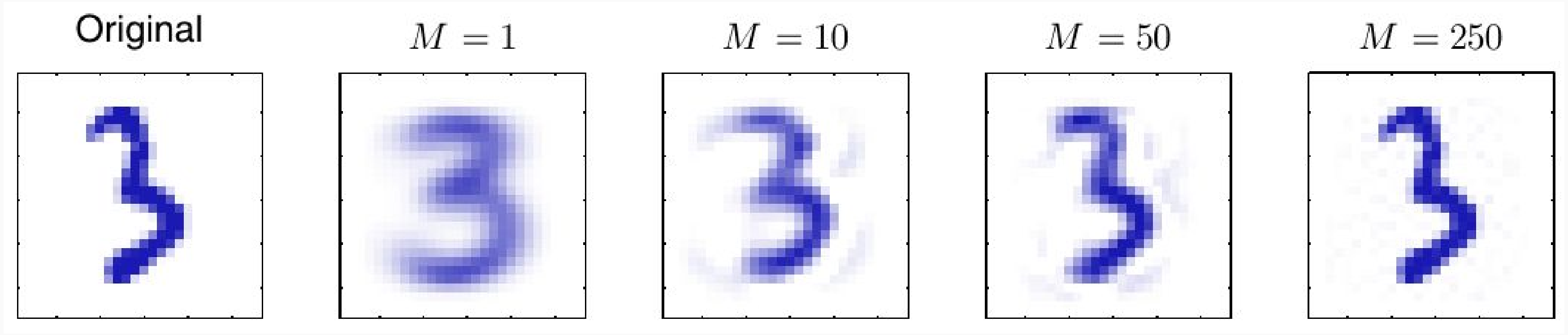
Size of eigenvalues (variance) for all components



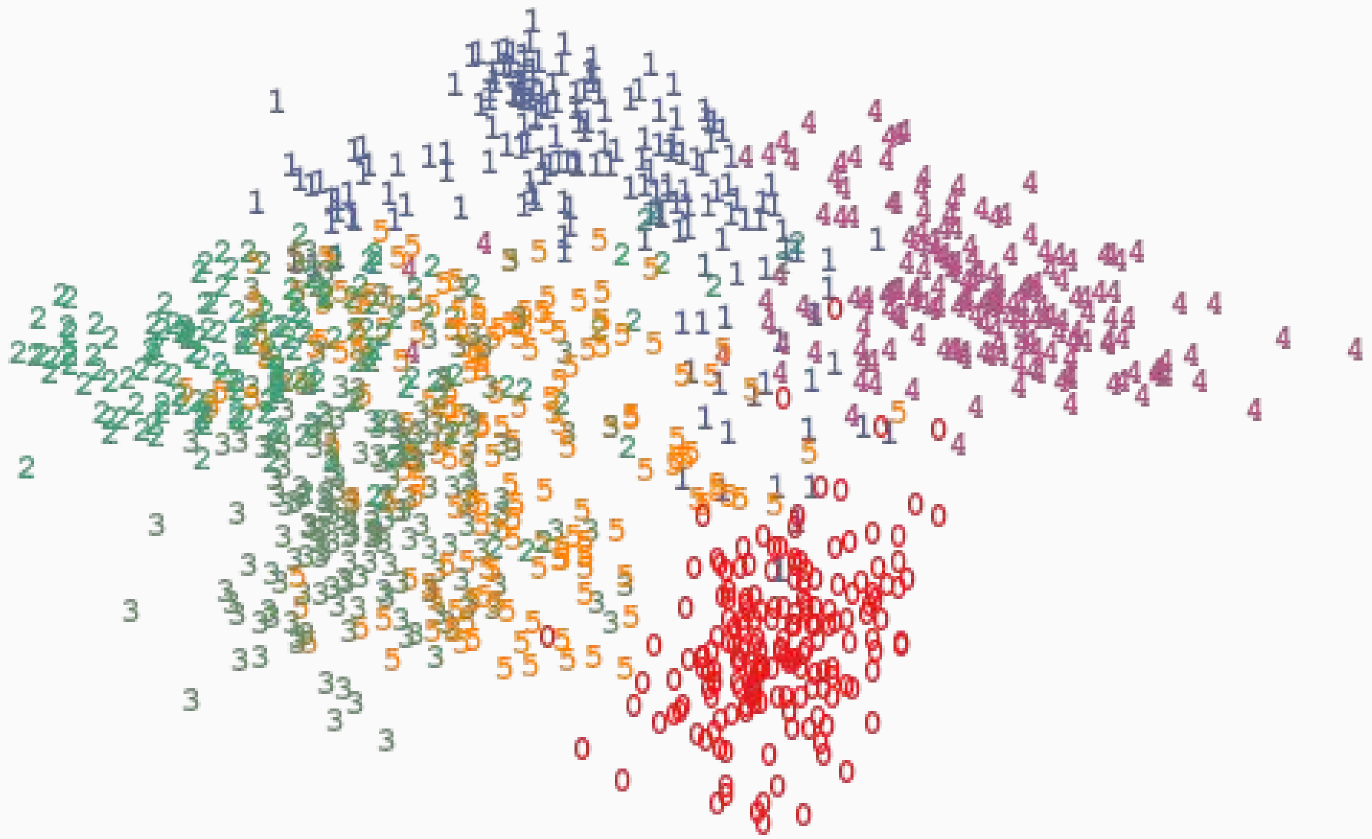
PCA on MNIST

Reconstructing the input by a linear combination of eigenvectors

$$x' = \phi_1 u_1 + \phi_2 u_2 + \phi_3 u_3 + \cdots$$



PCA on MNIST: Embedding in 2d



Pair Discussion

1. Why do we reduce dimensions when doing supervised learning? Are there particular models where it’s especially important?
2. Why do we reduce dimensions when doing unsupervised learning? Are there particular models where it’s especially important?
3. How do you use dimensionality reduction for data visualization?

Individual Discussion

1. How is the first principal component computed?
2. Draw a picture which demonstrates what the first principal component is.

Pair Discussion

Our new covariance matrix after doing a PCA transformation looks like this:

$$\begin{matrix} 8 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{matrix}$$

1. What does each value in the diagonal mean?
2. Why are the non-diagonal values all 0? Why is this desirable?

When to use PCA
(general advice only!)

Use when:

- kNN on high dimensional data
- Clustering high dimensional data
- Visualization
  (e.g. embeddings)
- Working with images
  (e.g. would it work well to feed an image into a decision tree?)

Don't use when:

- You need to retain interpretability of your feature space
- Your model doesn't *need* reduced dimensional data
  (e.g. OLS on relatively small data)

Dimensionality Reduction

Part 2: Singular Value Decomposition

Moses Marsh

galvanize

Objectives: answer the following

- What is SVD? How is it related to PCA?
- How do you interpret the U, Σ, and V matrices?
- What are “latent features”?

SVD: a general matrix factoring method

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

where

U is column orthogonal: $U^T U = I$

V is column orthogonal: $V^T V = I$

Σ is a diagonal matrix of positive values, where the diagonal is ordered in decreasing order

SVD: That looked a lot like PCA

Yep. In PCA, we derived $M^T M V = V \Lambda$ where Λ is the diagonal matrix of eigenvalues

And SVD is a magic math way to factor M: $M = U \Sigma V^T$

So we can show
$$\begin{aligned} M^T M &= (U \Sigma V^T)^T U \Sigma V^T \\ &= V \Sigma^T U^T U \Sigma V^T \\ &= V \Sigma^2 V^T \end{aligned}$$

Thus, $\Lambda = \Sigma^2$

The bonus here is that $M = U \Sigma V^T$ is much less computationally intensive to calculate than directly computing the eigenvalues of $M^T M$

SVD: What do those matrices mean?

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

n x p n x n n x p p x p

U is the **weights** matrix: each row is a projection of a data point onto the principal axes

Σ is the **singular values** matrix: its only non-zero elements are on the diagonal, and they are the square root of the variance across the corresponding principal component.

V^T is the **features** matrix: its columns are the eigenvectors, mapping the raw features to the principal axes

SVD: What do those matrices mean?

We'll see in a moment that this reduces to:

$$\underset{n \times p}{M} = \underset{n \times k}{U}\,\underset{k \times k}{\Sigma}\,\underset{k \times p}{V^T}$$

U is the **weights** matrix: each row is a projection of a data point onto the principal axes

Σ is the **singular values** matrix: its only non-zero elements are on the diagonal, and they are the square root of the variance across the corresponding principal component.

V^T is the **features** matrix: its columns are the eigenvectors, mapping the raw features to the principal axes

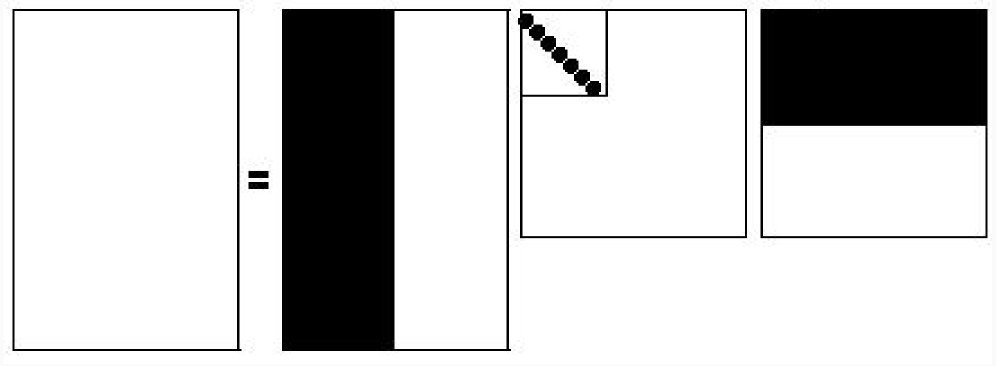
k is the **rank** of M

SVD: How to reduce dimensions from here

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

n x p   n x k   k x k   k x p

Dimensionality reduction here is simply lopping off these matrices after the first ***q*** singular values of **Σ**, the first ***q*** columns of **U**, and the first ***q*** rows of **V^T**



SVD: Latent Features

Principal components are a special case of ***latent features*** or ***topics***: linear combinations of the raw features that emerge from an algorithm and contain information in the way they group raw features together.

Example: See notebook!

Pair Discussion

	Seoul Garden	Izakaya Sozai	Yamo	Beretta	Delfina	Perbacco	Dosa	Udupi Palace
Kate	5	4	5	0	0	0	0	0
Lindsay	4	4	4	0	0	0	0	0
John	5	5	5	0	0	0	2	2
Cathy	0	0	0	0	0	0	5	5
Mary	1	1	1	4	5	5	0	0
Norma	0	0	0	4	4	4	0	0
Paul	0	0	0	5	5	5	2	2

Given the above matrix of restauraunt preferences of your friends:

1. What latent features do you expect to come up when you run SVD?

Pair Discussion

$$\begin{bmatrix} 5 & 4 & 5 & 0 & 0 & 0 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 & 0 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 5 \\ 1 & 1 & 1 & 4 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 5 & 5 & 5 & 2 & 2 \end{bmatrix} = \begin{bmatrix} -0.4 & 0.4 & -0.2 & 0.6 & -0.5 & -0.0 & 0.0 \\ -0.3 & 0.4 & -0.1 & -0.2 & 0.3 & -0.6 & 0.4 \\ -0.5 & 0.4 & 0.2 & -0.3 & 0.3 & 0.5 & -0.3 \\ -0.1 & -0.0 & 0.9 & -0.0 & -0.2 & -0.3 & -0.1 \\ -0.5 & -0.4 & -0.2 & -0.5 & -0.6 & 0.0 & -0.0 \\ -0.3 & -0.4 & -0.1 & 0.3 & 0.3 & -0.4 & -0.7 \\ -0.4 & -0.5 & 0.2 & 0.3 & 0.3 & 0.3 & 0.5 \end{bmatrix}$$

$$\times \begin{bmatrix} \boxed{14.6} & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & \boxed{13.3} & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & \boxed{7.6} & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & \boxed{0.7} & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & \boxed{0.6} & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \boxed{0.0} & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \boxed{0.0} \end{bmatrix}$$

$$\times \begin{array}{c} 1 \\ 2 \\ 3 \\ \\ \\ \\ \\ \\ \end{array} \begin{bmatrix} -0.4 & -0.4 & -0.4 & -0.4 & -0.4 & -0.4 & -0.2 & -0.2 \\ 0.4 & 0.4 & 0.4 & -0.4 & -0.4 & -0.4 & -0.0 & -0.0 \\ -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & -0.1 & 0.7 & 0.7 \\ 0.3 & -0.6 & 0.3 & 0.6 & -0.3 & -0.3 & -0.0 & -0.0 \\ -0.3 & 0.6 & -0.3 & 0.6 & -0.3 & -0.3 & -0.0 & -0.0 \\ -0.7 & -0.0 & 0.7 & -0.0 & 0.0 & 0.0 & -0.0 & -0.0 \\ -0.0 & 0.0 & 0.0 & 0.0 & 0.7 & -0.7 & 0.0 & 0.0 \\ 0.0 & 0.0 & -0.0 & 0.0 & 0.0 & -0.0 & -0.7 & 0.7 \end{bmatrix}$$

Seoul Garden	Izakaya Sozai	Yamo	Beretta	Delfina	Perbacco	Dosa	Udupi Palace

Given the above SVD decomposition:

1. What do the values in the diagonal tell you? (circled in blue)
2. Each of the rows circled in purple refers to a latent feature. What are the 3 latent features? Are they the 3 latent features you expected to come up?

Pair Discussion

Below you can see what happens when you plug in the SVD decomposition into the covariance matrix calculation.

$$M^T M = (U\Sigma V^T)^T U\Sigma V^T \tag{1}$$
$$= V\Sigma^T U^T U\Sigma V^T \tag{2}$$
$$= V\Sigma^2 V^T \tag{3}$$

1. What mathematical rule is used for each step?
2. How does this relate SVD to PCA?