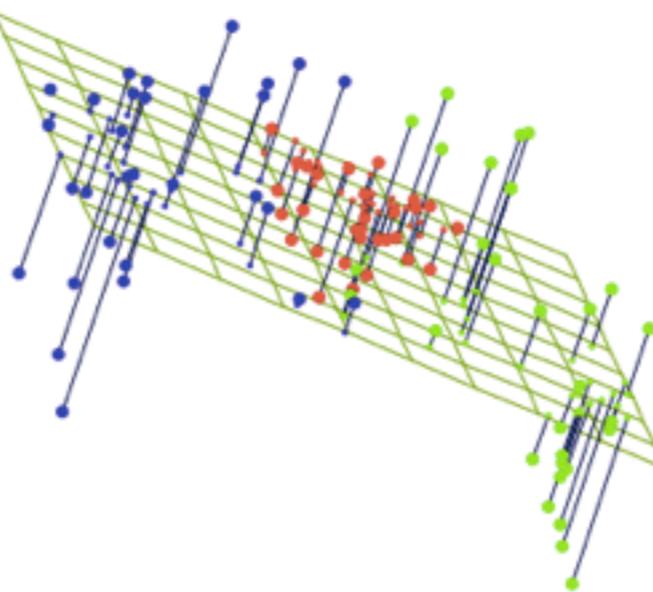


# CS109 Data Science Classification & PCA

Hanspeter Pfister, Joe Blitzstein, and Verena Kaynig



# This Week

- HW2 due on Thursday
- HW1 grades will be out Sunday night
- HW1 solution is available next Tuesday

# Classification

# Classification



05	02	22	97	38	15	00	40	00	75	04	05	07	78	52	12	50	77	31	00
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	40	01	56	62	00
81	49	31	73	55	79	14	29	93	71	40	67	50	08	30	03	49	13	36	65
52	70	95	23	04	60	11	42	69	21	48	56	01	32	56	71	37	02	36	91
22	31	16	71	51	67	03	59	41	92	36	54	22	40	40	28	66	33	13	80
24	47	38	60	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
32	98	81	28	64	23	67	10	26	38	40	67	59	54	70	66	18	38	64	70
67	26	20	68	02	62	12	20	95	63	94	39	63	08	40	91	66	49	94	21
24	55	58	05	66	73	99	26	97	17	78	78	96	83	14	88	34	89	63	72
21	36	23	09	75	00	76	44	20	45	35	14	00	61	33	97	34	31	33	95
78	17	53	28	22	75	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	96	35	31	47	55	58	88	24	00	17	54	24	36	29	85	57
86	56	00	48	35	71	89	07	05	44	44	37	44	60	21	58	51	54	17	58
19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04	89	55	40
04	52	08	83	97	35	99	16	07	97	57	32	16	26	26	79	33	27	98	66
00	94	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69
04	42	16	73	35	35	39	11	24	94	72	18	08	46	29	32	40	62	76	36
20	69	36	41	72	30	23	88	31	42	89	69	82	67	59	85	74	04	36	16
20	73	35	29	78	31	90	01	74	31	49	71	10	00	81	16	23	57	05	54
01	70	54	71	83	51	54	49	16	92	33	48	61	43	52	01	89	23	47	40

What the computer sees

image classification

82% cat  
15% dog  
2% hat  
1% mug

# Problems

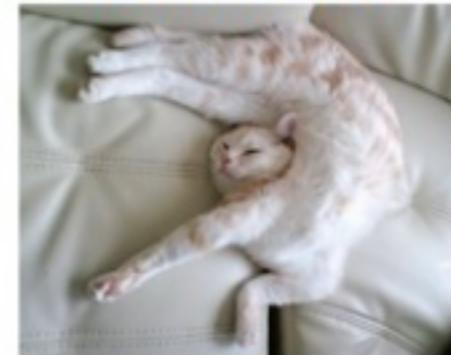
Viewpoint variation



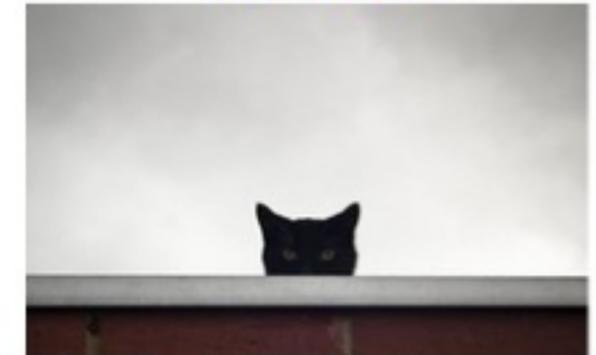
Scale variation



Deformation



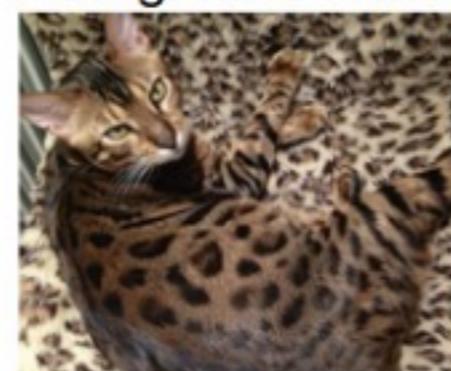
Occlusion



Illumination conditions



Background clutter



Intra-class variation



# Training Data

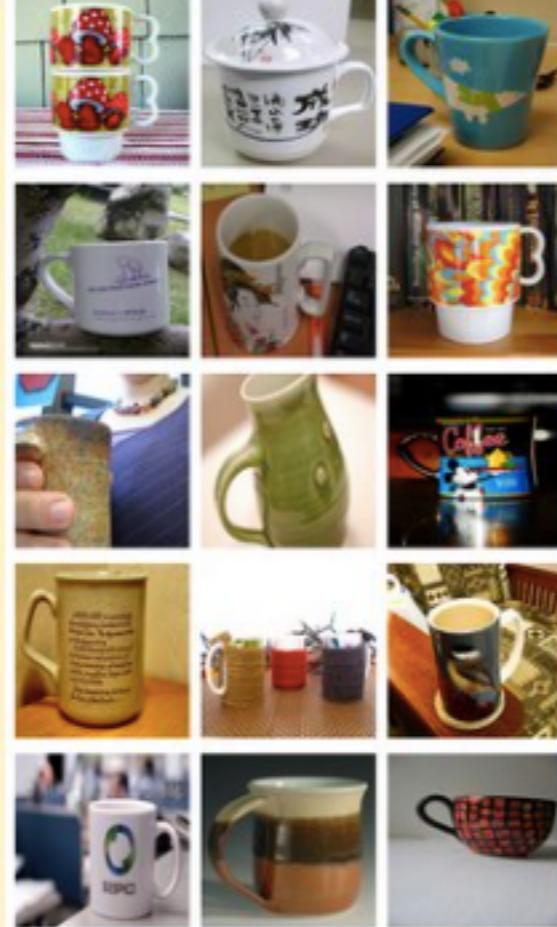
cat



dog



mug



hat



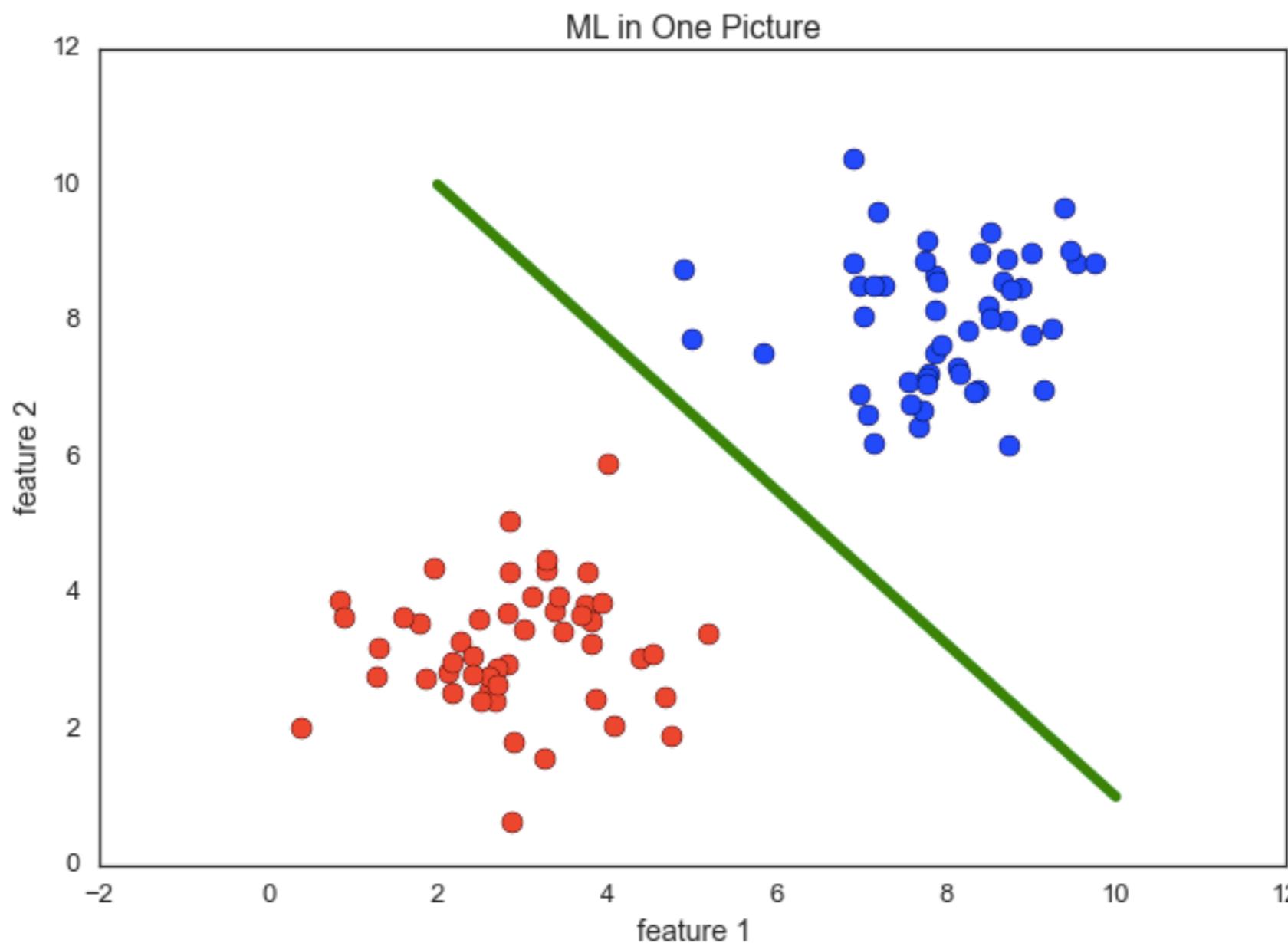
# Machine Learning

- Input: A **training set** of  $N$  data points, each labeled with one of  $K$  different classes.
- Learning: Use the training set to learn what every one of the classes looks like.
- Evaluation: Predict labels for a **test set** of data and compare the true labels (**ground truth**) to the ones predicted by the classifier.

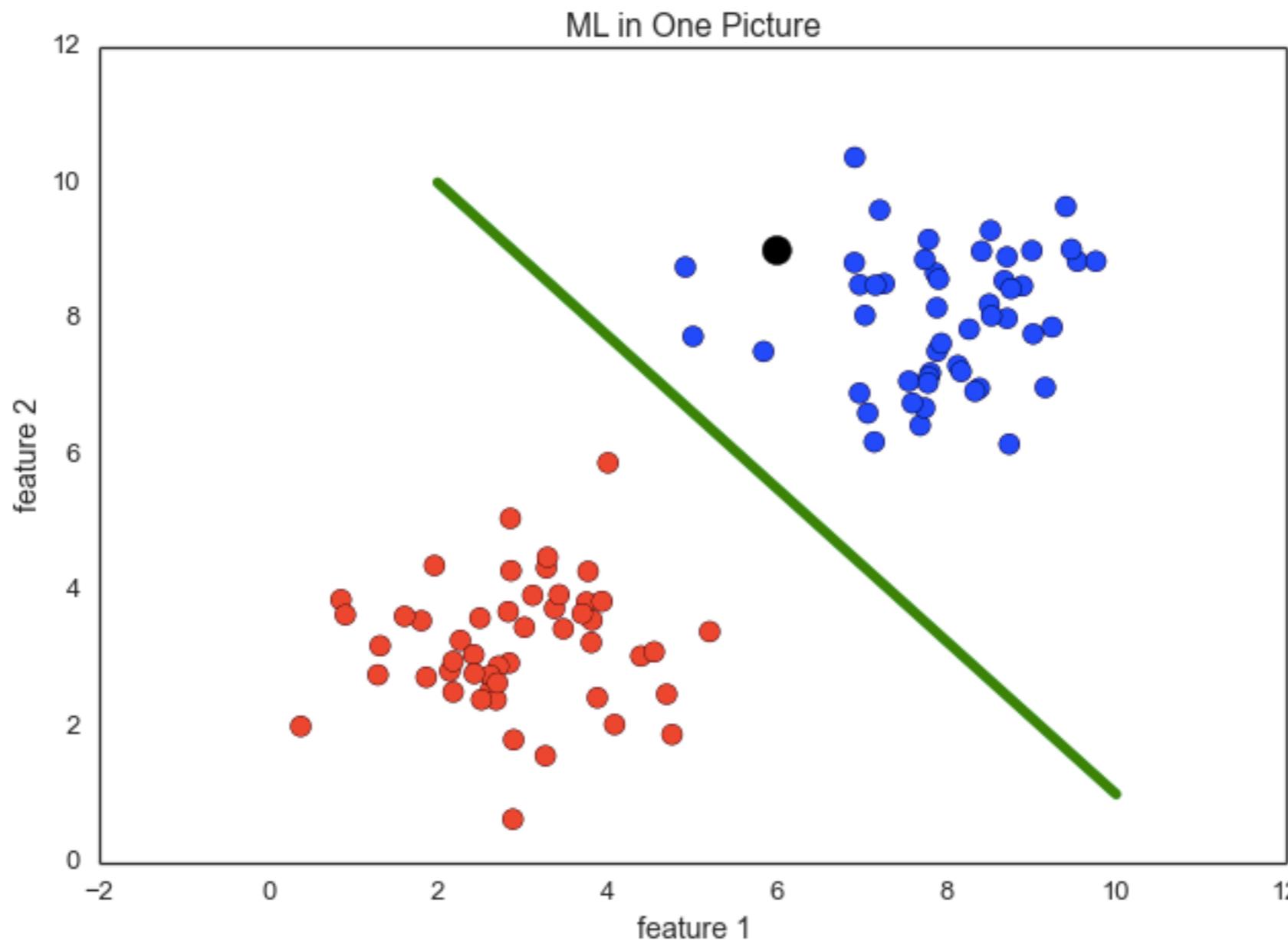
# Machine Learning

- Make predictions for new data points
  - data has labels
  - supervised learning: kNN, SVM, Decision Trees, Random Forests, Bagging, Boosting, etc.
- Find patterns in the data
  - data has no labels
  - unsupervised learning: PCA, MDS, Clustering

# Classification

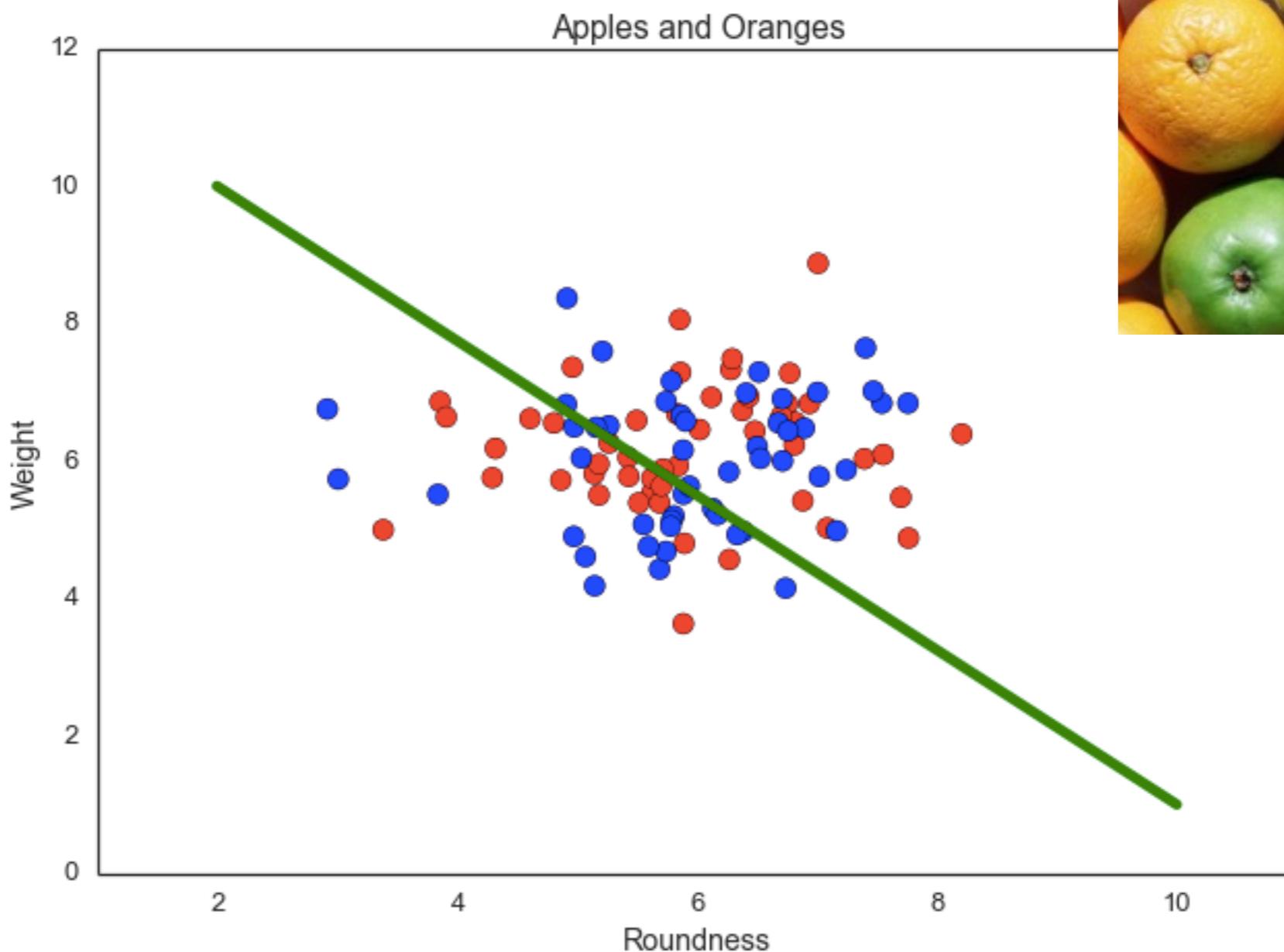


# Classification

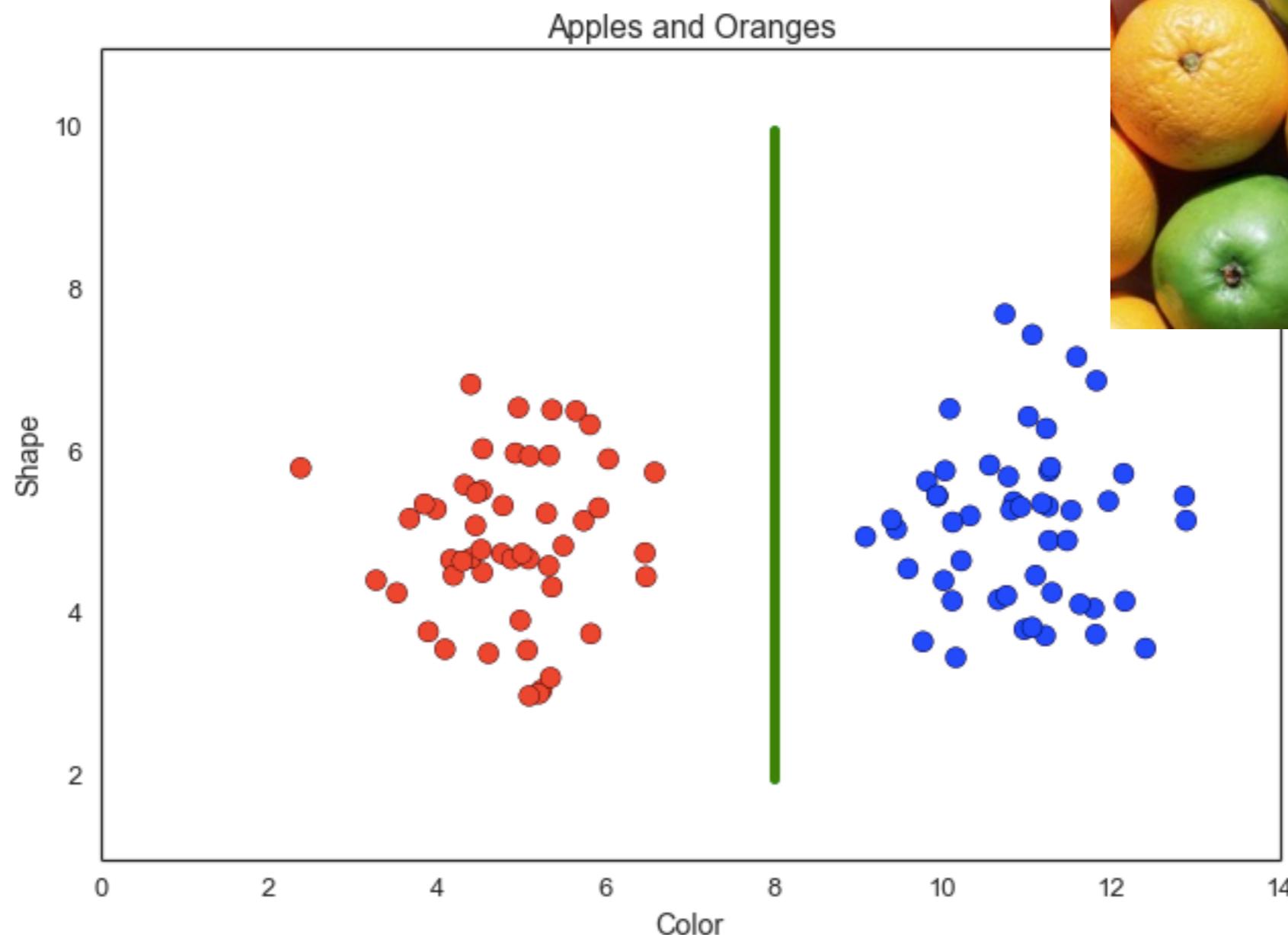


x: data points  
y: labels  
features  
decision  
boundary

# Feature Selection



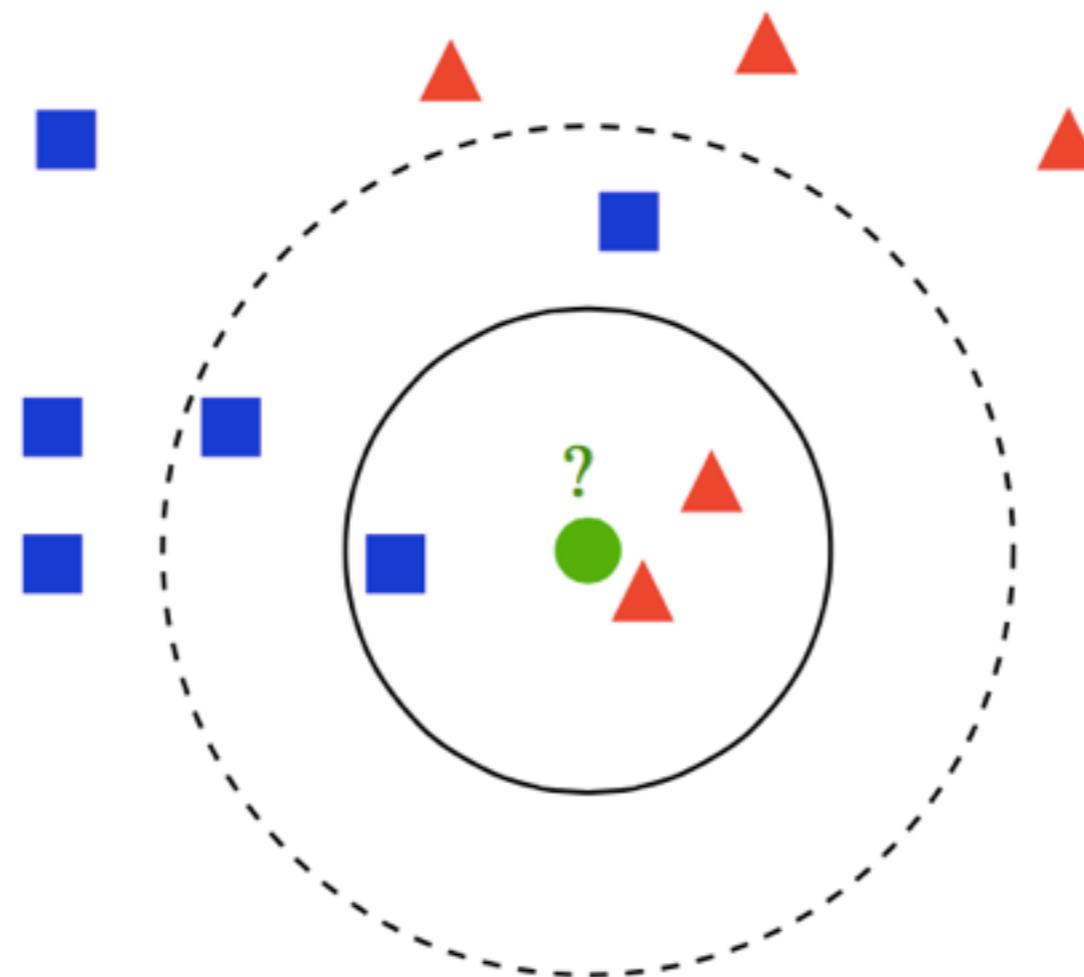
# Feature Selection



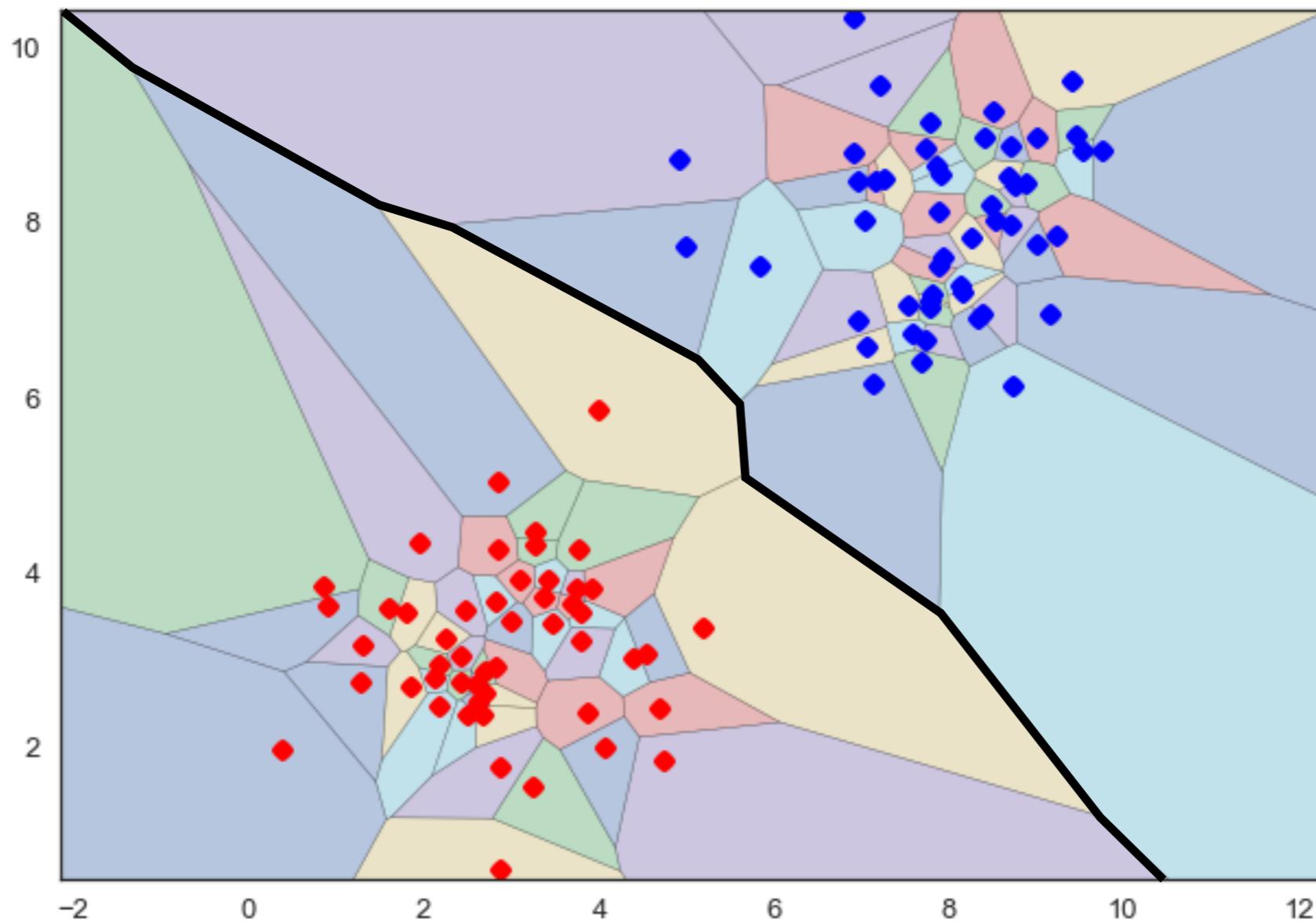
# Nearest Neighbor Classifier

# Nearest Neighbor

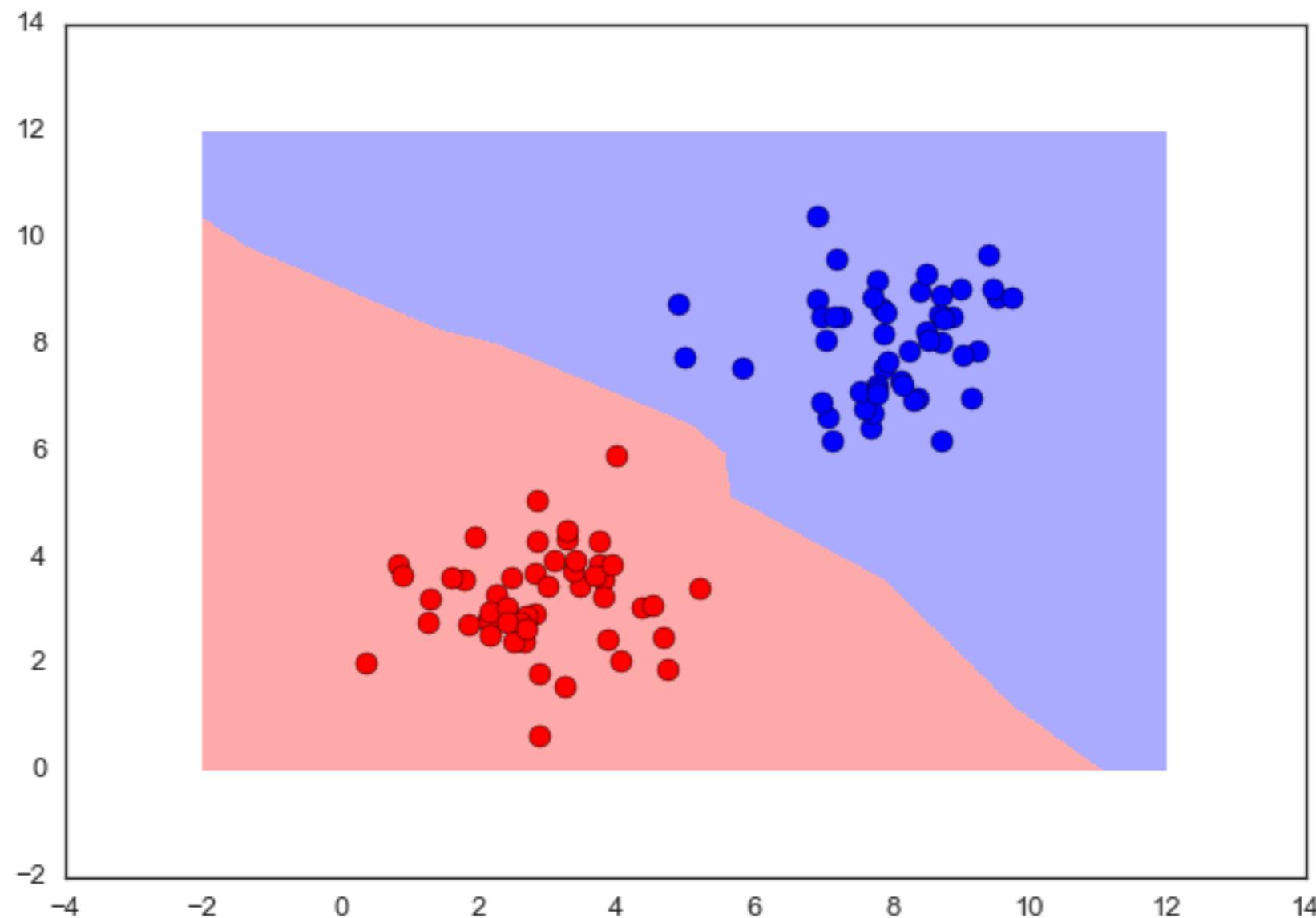
Predict class of new data point by majority vote  
of K nearest neighbors



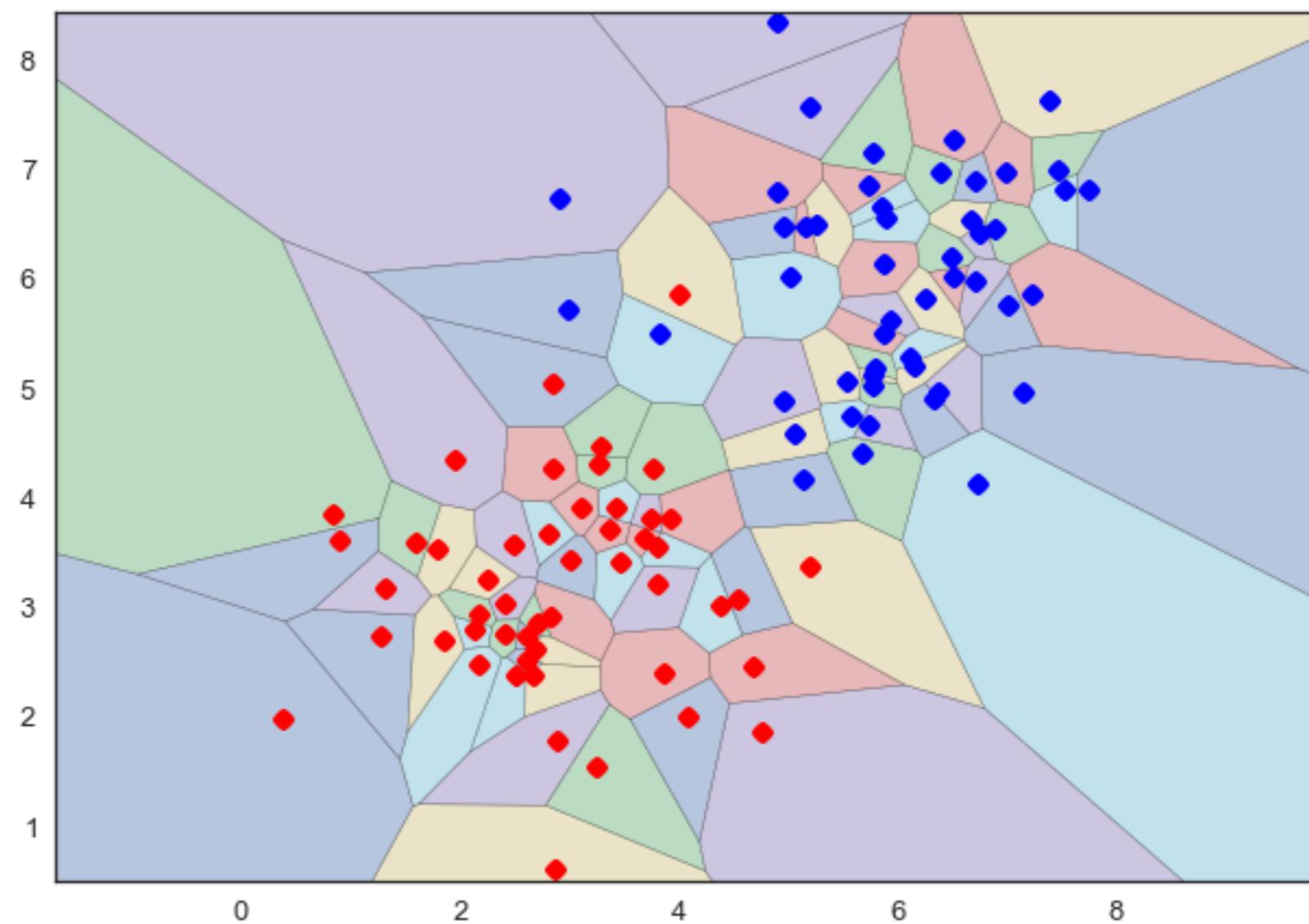
# I-Nearrest Neighbor



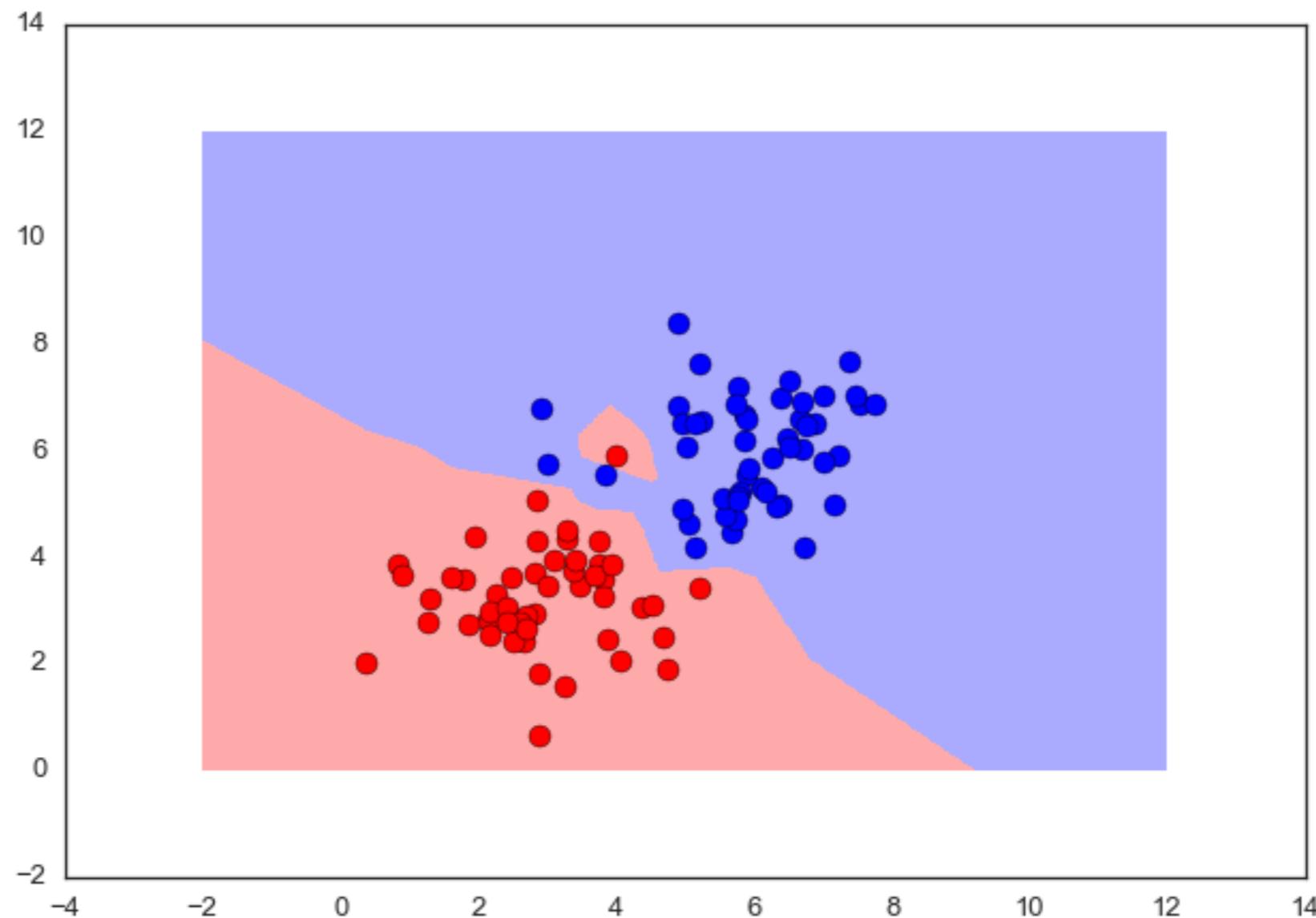
# I-Nearrest Neighbor



# I-Nearrest Neighbor

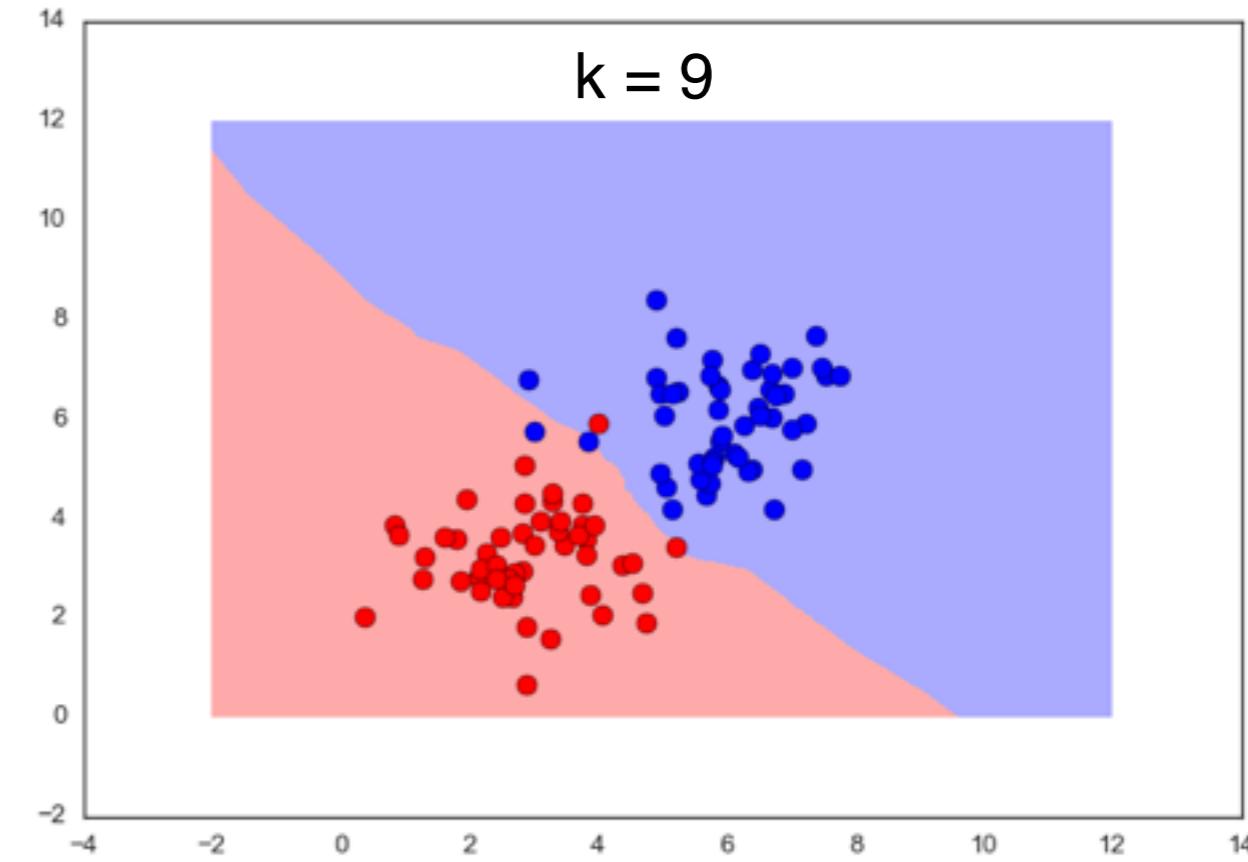
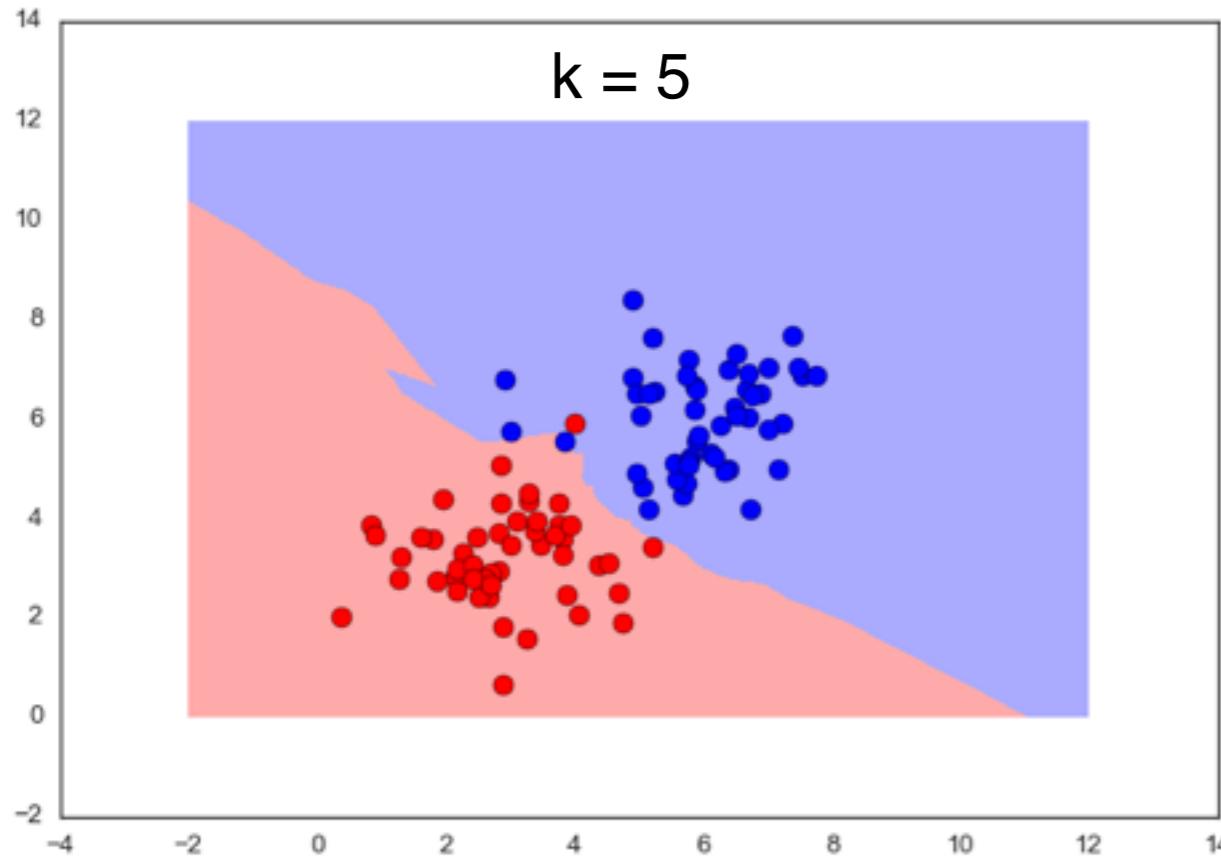
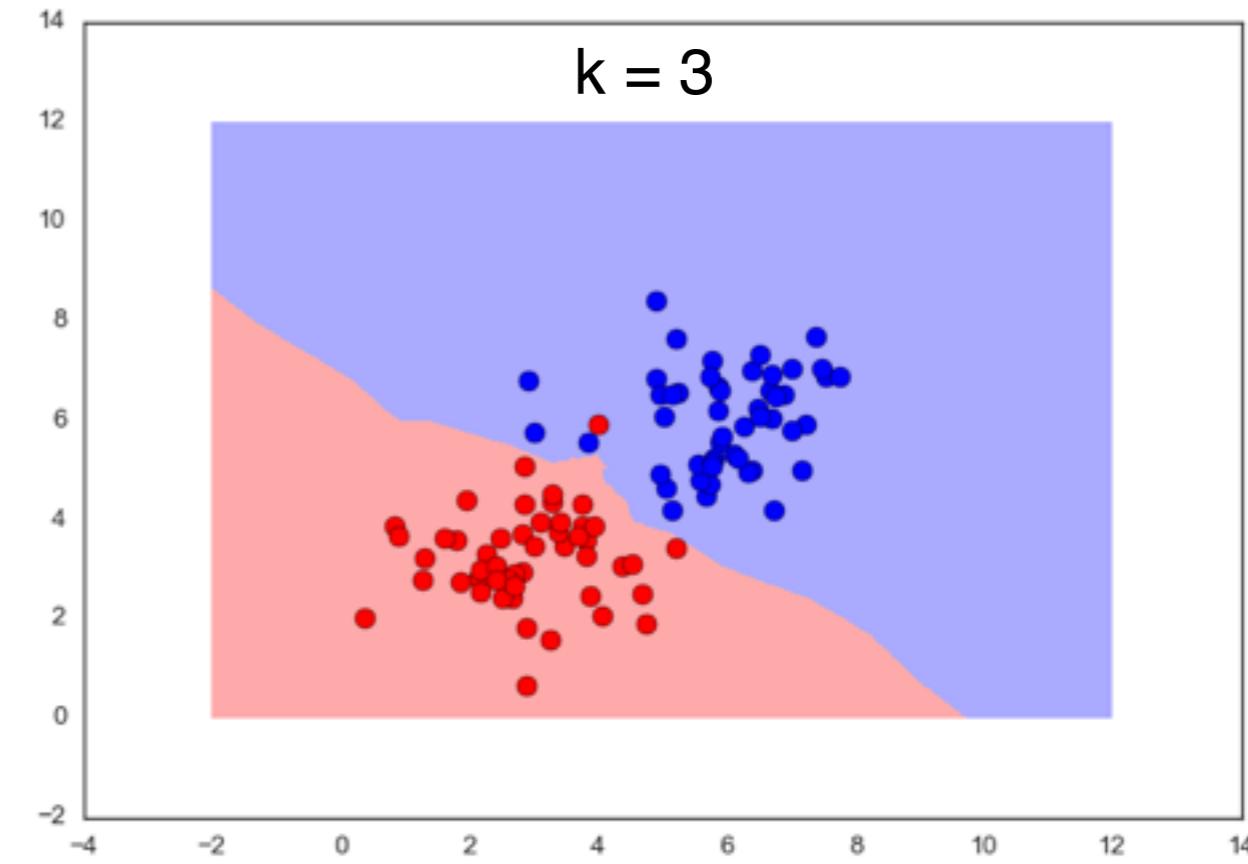
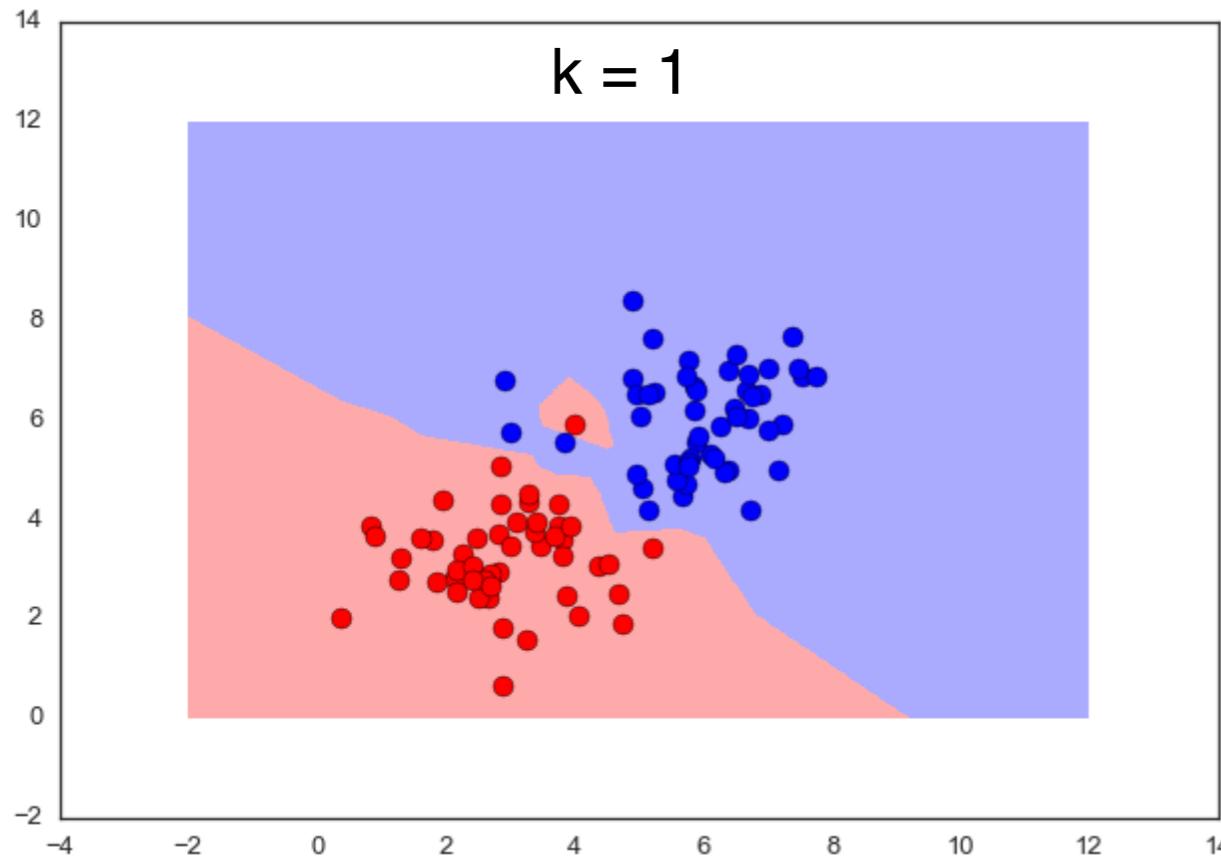


# I-Nearrest Neighbor



# I-NN Properties

- Simple and quite good for low dimensional data
- “Rough” decision boundary, may have “islands”
- Training complexity for  $N$  data points?
- Test complexity for  $M$  data points?
- Error on training set?
- Variance? Bias?

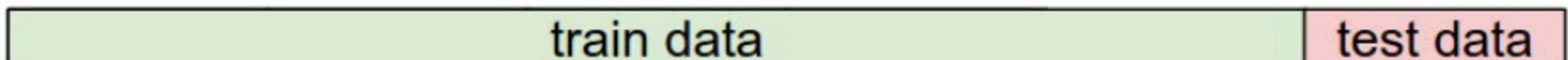


# k-NN Properties

- Gets rid of “islands”
- If  $k$  is too large, the boundary may become too smooth
- Lower variance, but increased bias
- How do we choose the ideal  $k$ ?

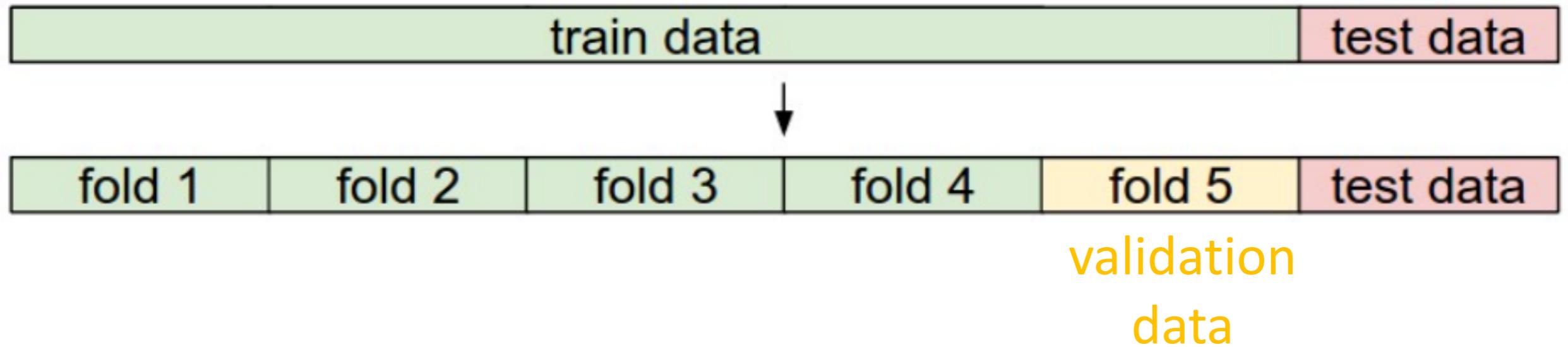
# Validation

- Train on training data, test on test data
- Pick the k with the lowest test error



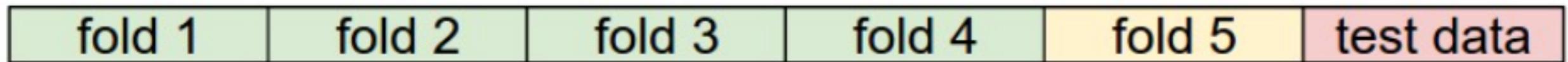
- **Training data:** train classifier
- **Test data:** measure performance

# Cross-Validation



- **Training data:** train classifier
- **Validation data:** estimate (hyper) parameters ( $k$ )
- **Test data:** measure performance

# Cross-Validation



1. Iterate over choice of validation fold
2. For all parameter values:
  - a. Train with training data
  - b. Validate with validation data
3. Average the parameters with best performance on validation data

The test data is NOT used to determine the parameters!

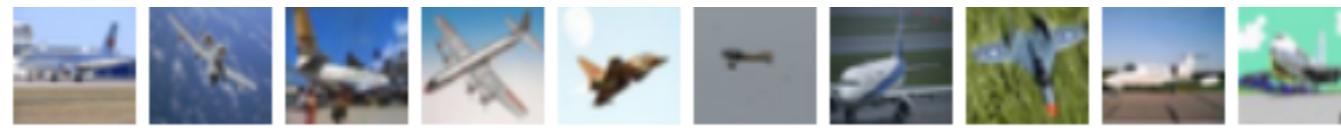
# Cross-Validation

- Take best parameters 
- Train on training data and validation data   together
- Test performance on test data 

Evaluate on the test set only a single time, at the very end!

# CIFAR-10 Data Set

**airplane**



60,000 images  
32x32 pixels

**automobile**



**bird**



**cat**



10 classes

**deer**



**dog**

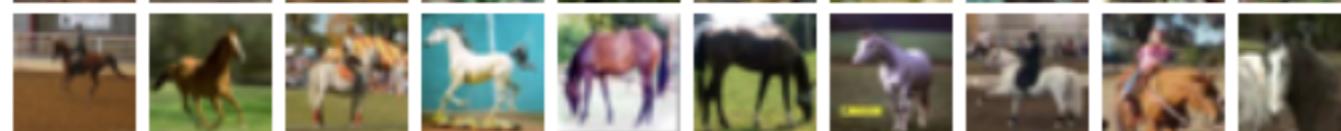


Training set:  
50,000 images

**frog**



**horse**



**ship**



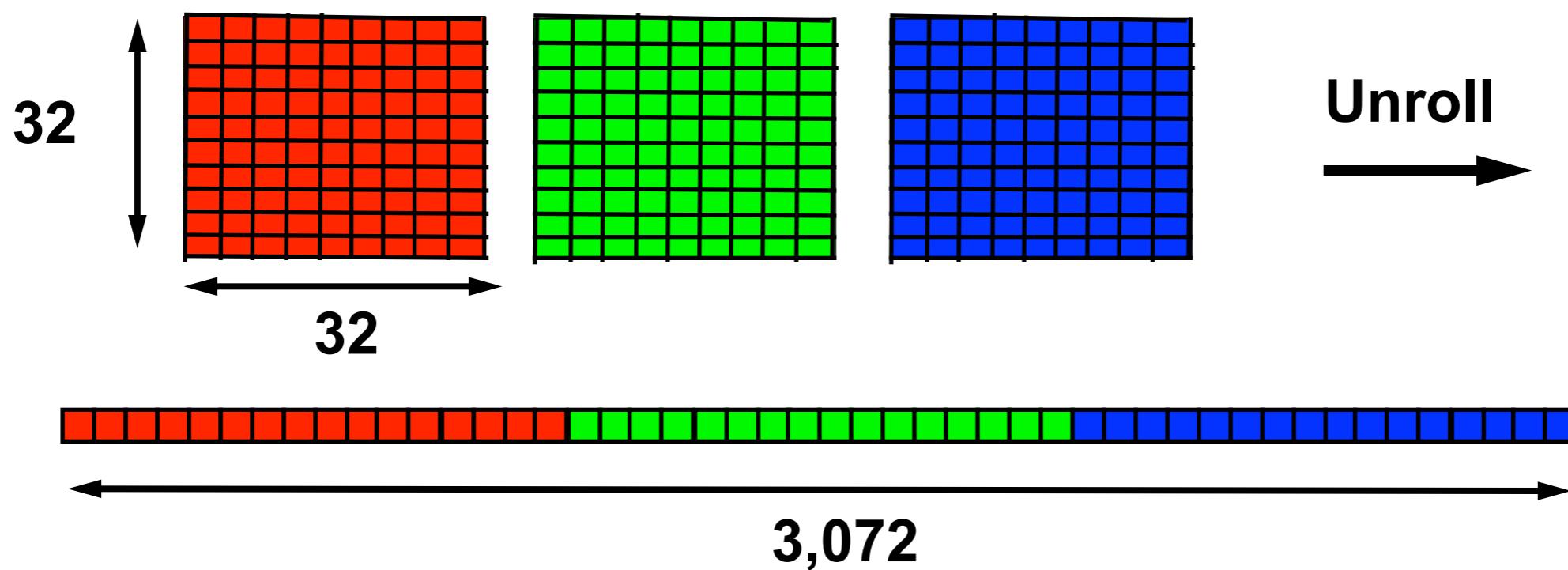
Test set:  
10,000 images

**truck**



# High-Dimensional Data

Each image is a vector in  $32 \times 32 \times 3 = 3,072$  dimensional space



# Image Comparison

test image				training image				pixel-wise absolute value differences				→ 456
56	32	10	18	10	20	24	17	46	12	14	1	
90	23	128	133	8	10	89	100	82	13	39	33	
24	26	178	200	12	16	178	170	12	10	0	30	
2	0	255	220	4	32	233	112	2	32	22	108	

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

p: pixel  
I<sub>1</sub>: image 1

# Distance Metrics

L1 (Manhattan) distance:  $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

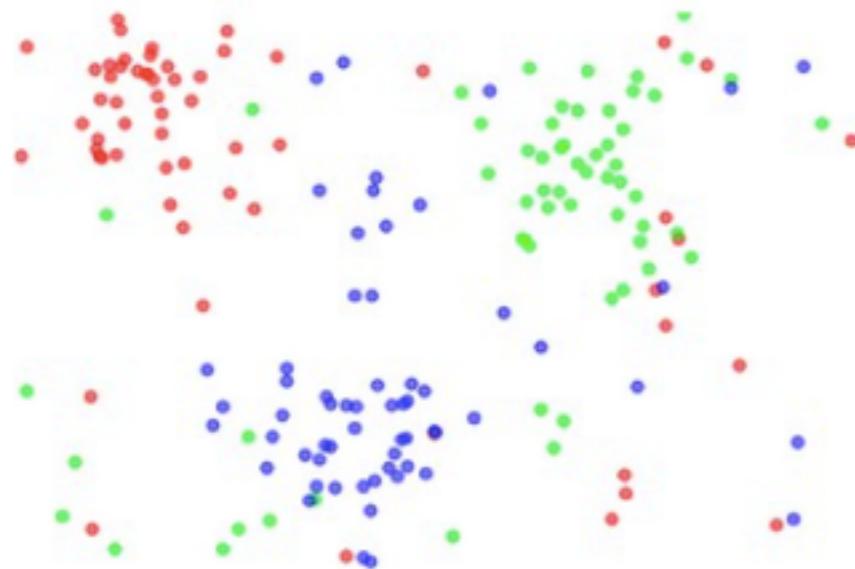
L2 (Euclidean) distance:  $d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$

More general L<sub>p</sub> norms:

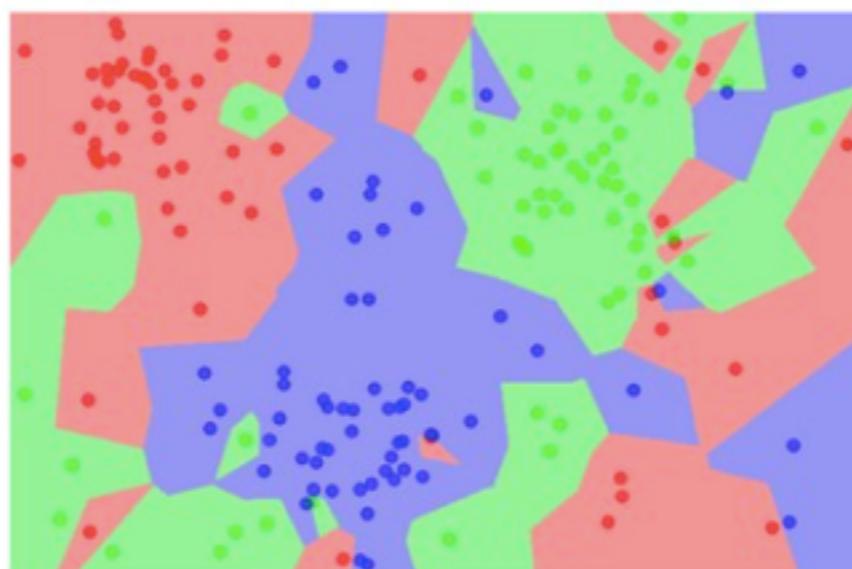
$$\|x\|_p = \left( |x_1|^p + \cdots + |x_n|^p \right)^{\frac{1}{p}} \quad p \geq 1, x \in \mathbb{R}^n$$

# k-NN Classifiers

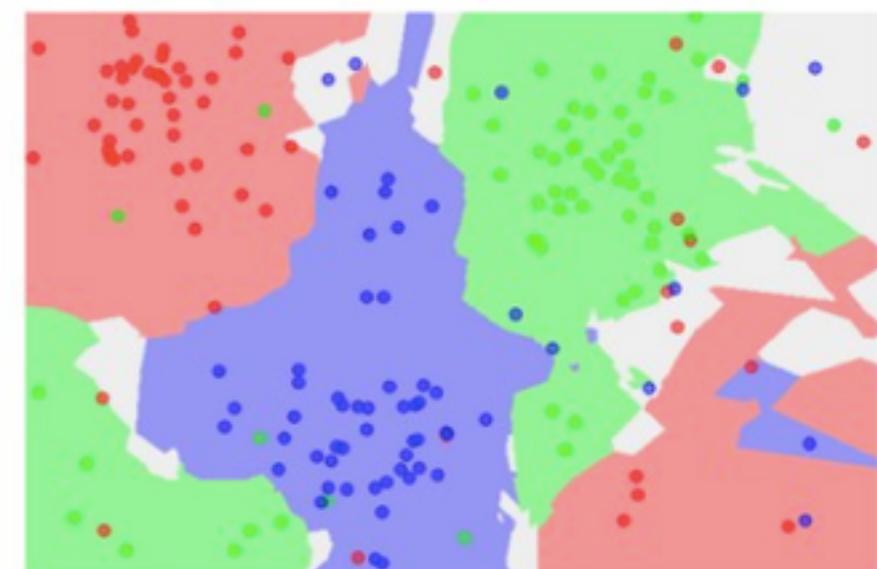
the data



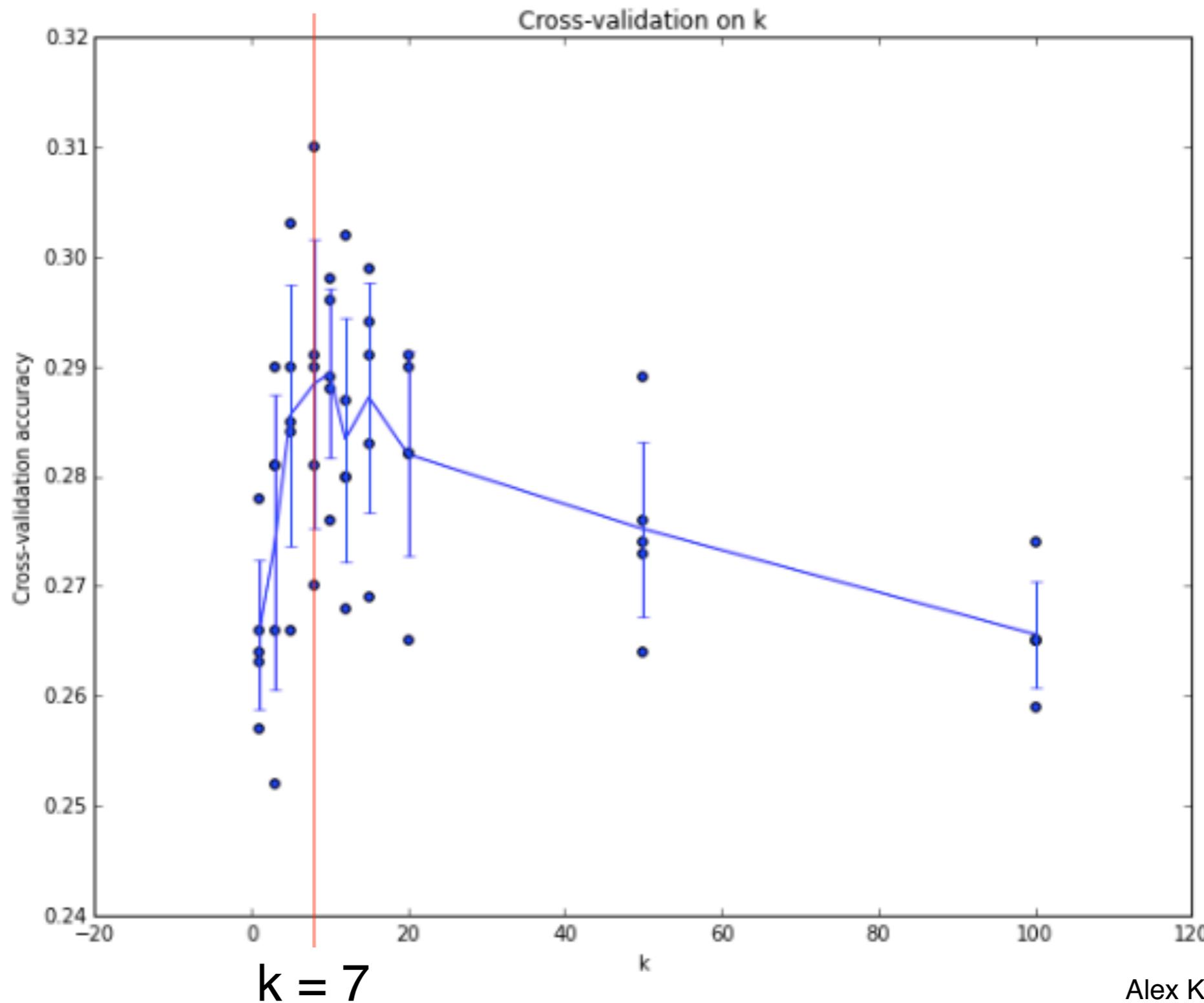
NN classifier



5-NN classifier

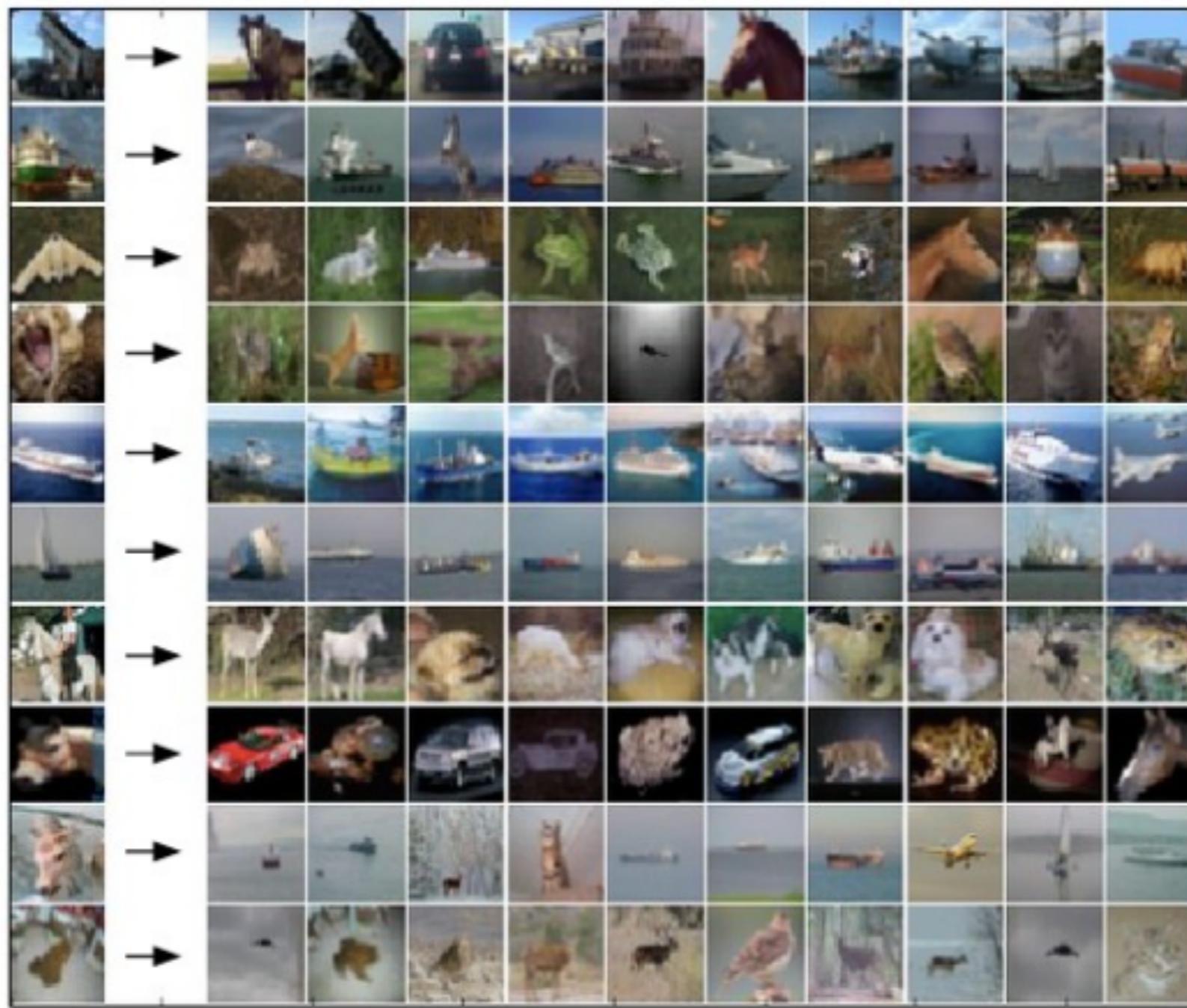


# 5-fold Cross Validation



# 10 Nearest Neighbors

Accuracy: L1 distance: 38% / L2 distance 35%



Alex Krizhevsky, Stanford

# Pixel-based L2 Distances

These images have the same L2 pixel distance

original



shifted



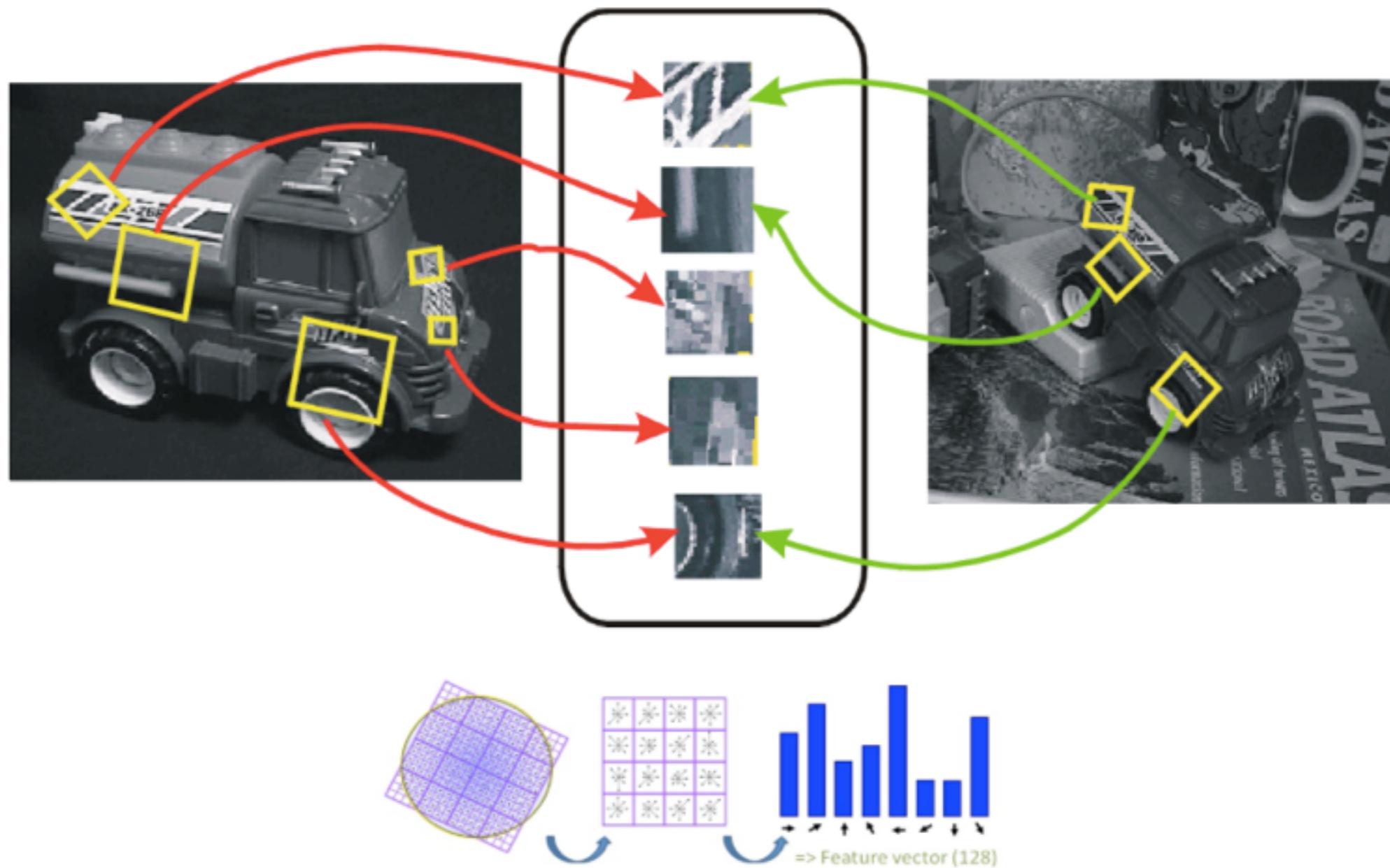
messed up



darkened

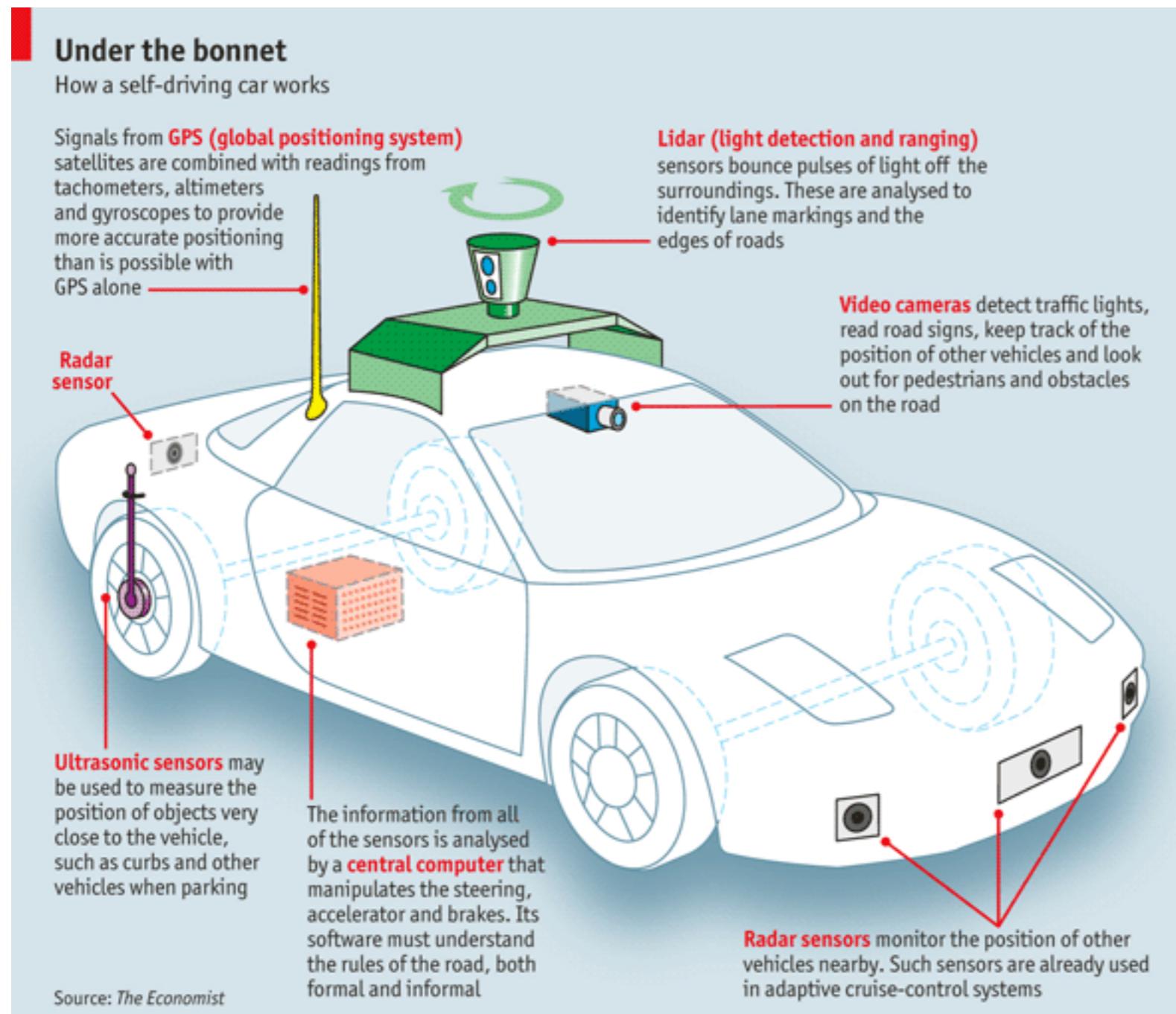


# Image Features



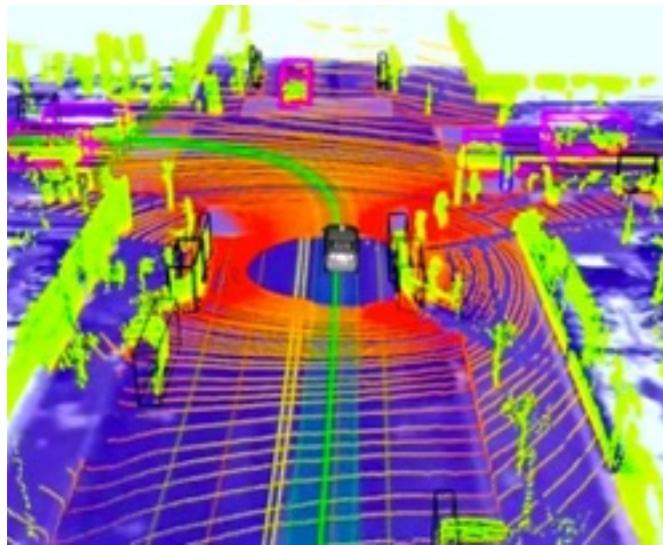
"SIFT" & Object Recognition, David Lowe, 1999

# Self-Driving Cars

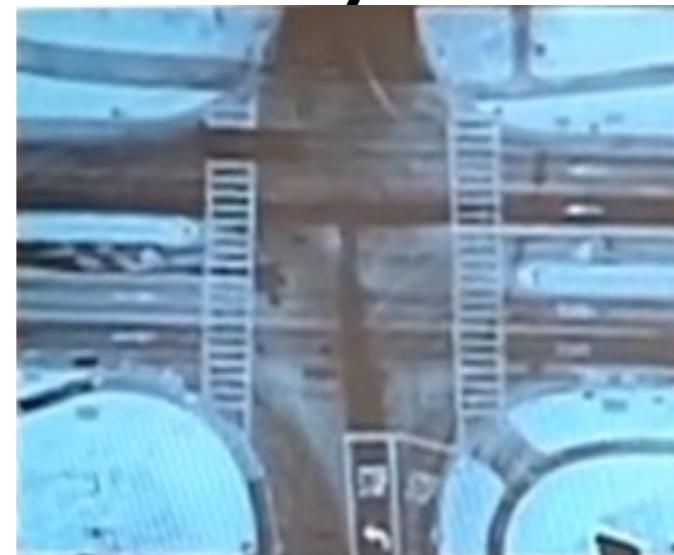


# Car Features

Laser scan



Intensity model

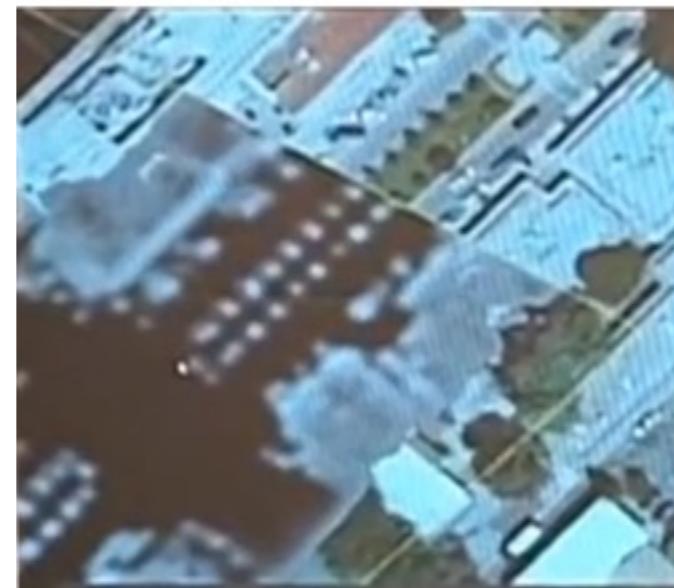


Elevation model



Camera vision

2D stationary map



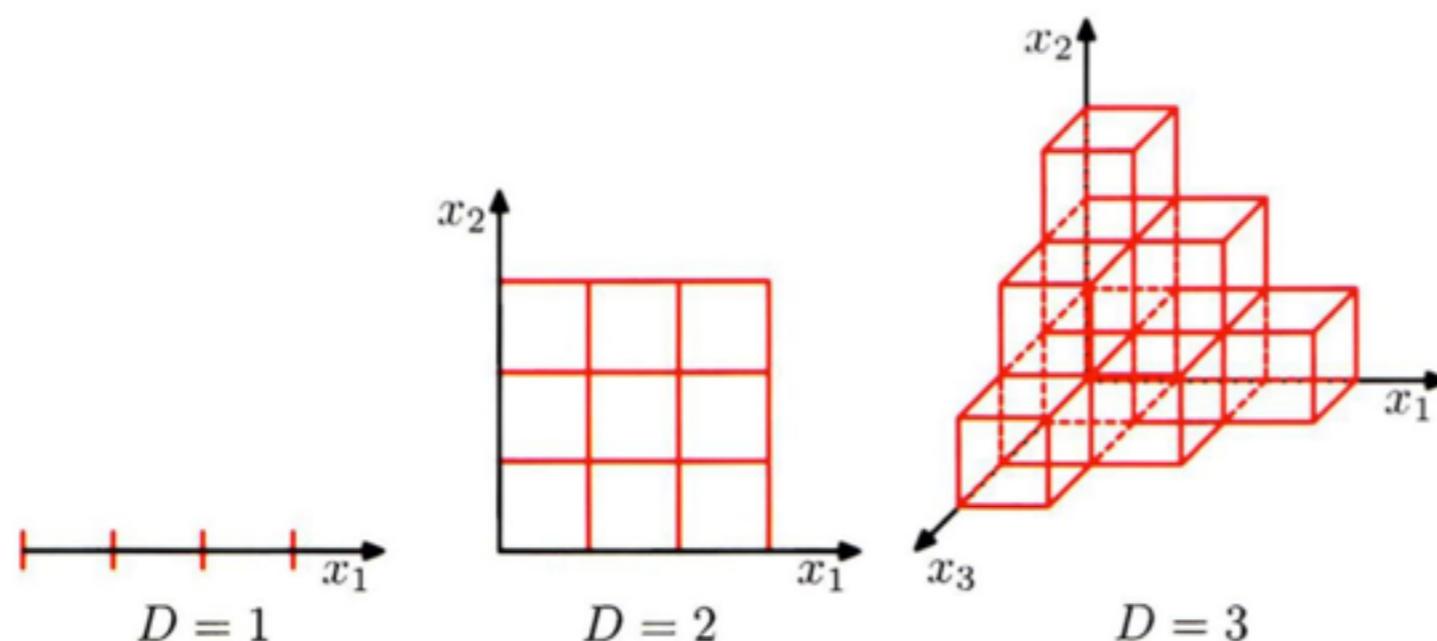
Lane model



Why don't we just use  
more and more features?

# Curse of Dimensionality

- When dimensionality increases, the volume of the space increases so fast that the available data becomes sparse
- Statistically sound result requires the sample size  $N$  to grow exponentially with  $d$



# Dimensionality Reduction

# High-dimensional Data

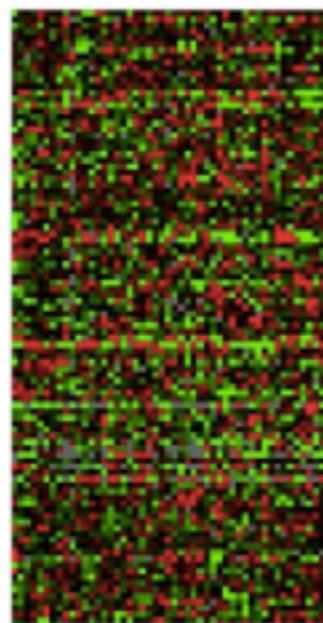


face images

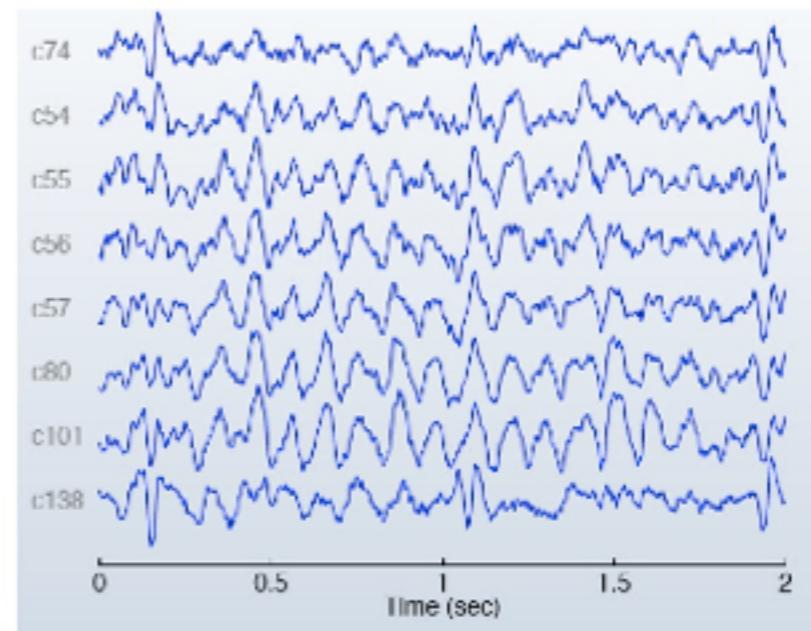
Zambian President Levy Mwanawasa has won a second term in office in an election his challenger Michael Sata accused him of rigging, official results showed on Monday.

According to media reports, a pair of hackers said on Saturday that the Firefox Web browser, commonly perceived as the safer and more customizable alternative to market leader Internet Explorer, is critically flawed. A presentation on the flaw was shown during the ToorCon hacker conference in San Diego.

documents



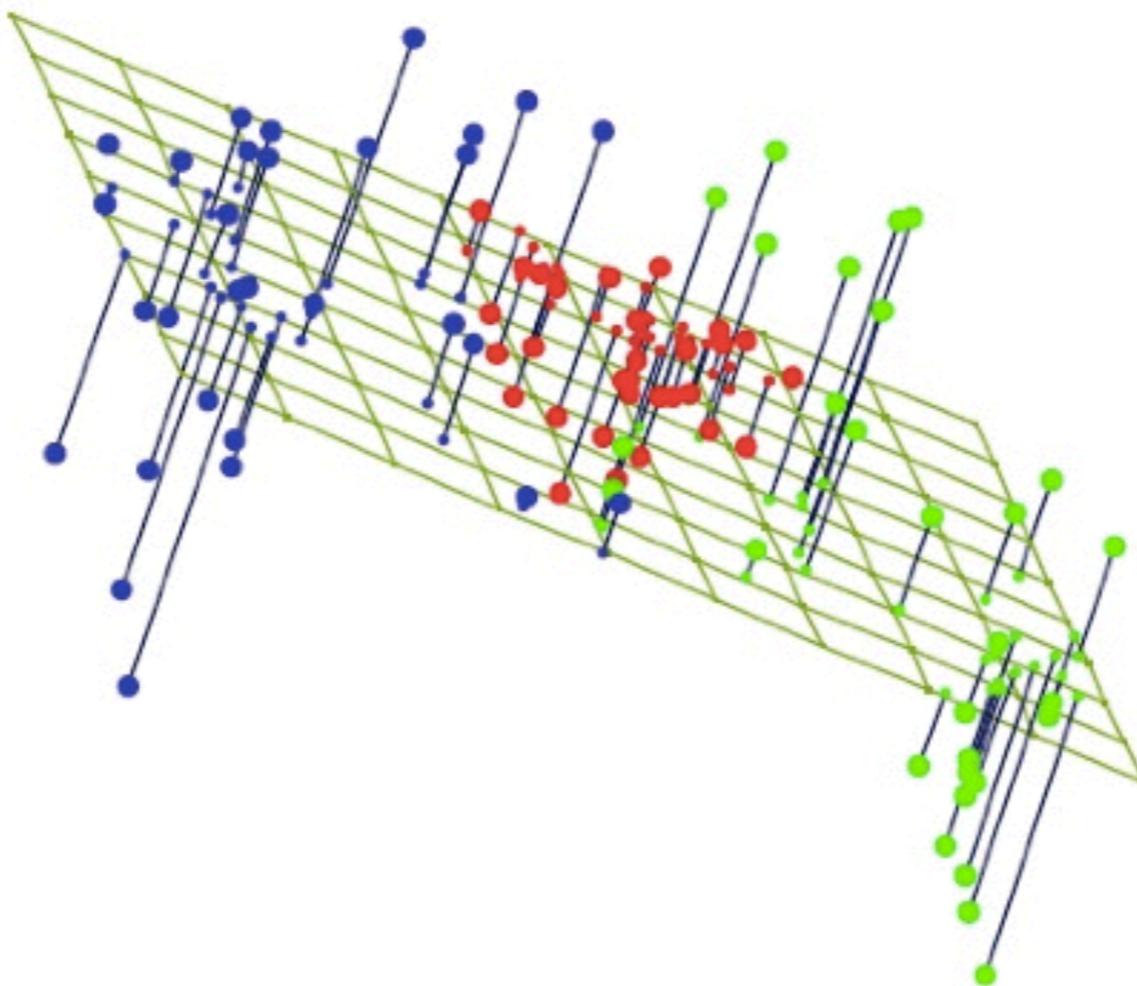
gene expression data



MEG readings

# Basic Idea

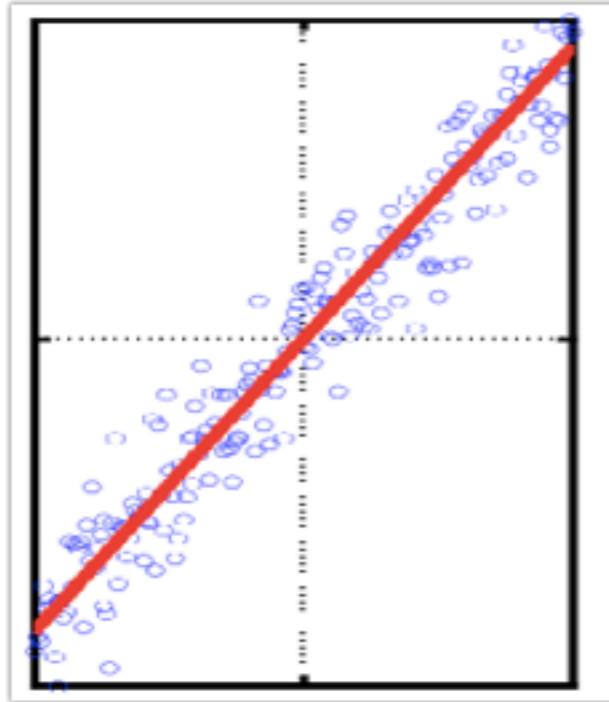
Project the high-dimensional data onto a lower-dimensional subspace that best “fits” the data



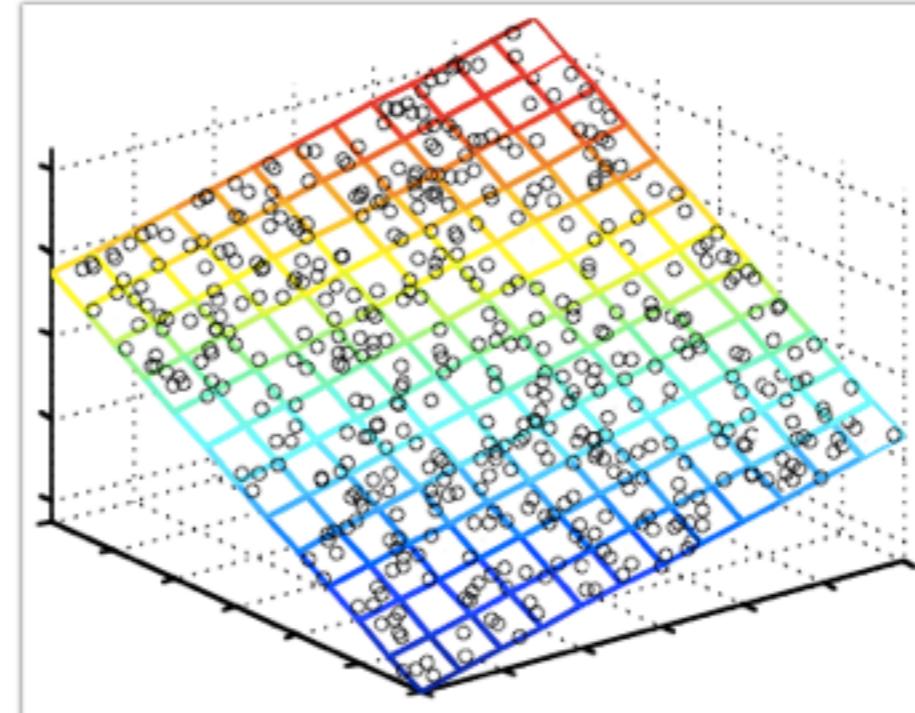
# Linear Methods

- Does the data lie mostly in a hyperplane?
- If so, what is its *intrinsic* dimensionality  $d$ ?

$$\begin{aligned} \mathbf{D} &= 2 \\ \mathbf{d} &= 1 \end{aligned}$$



$$\begin{aligned} \mathbf{D} &= 3 \\ \mathbf{d} &= 2 \end{aligned}$$

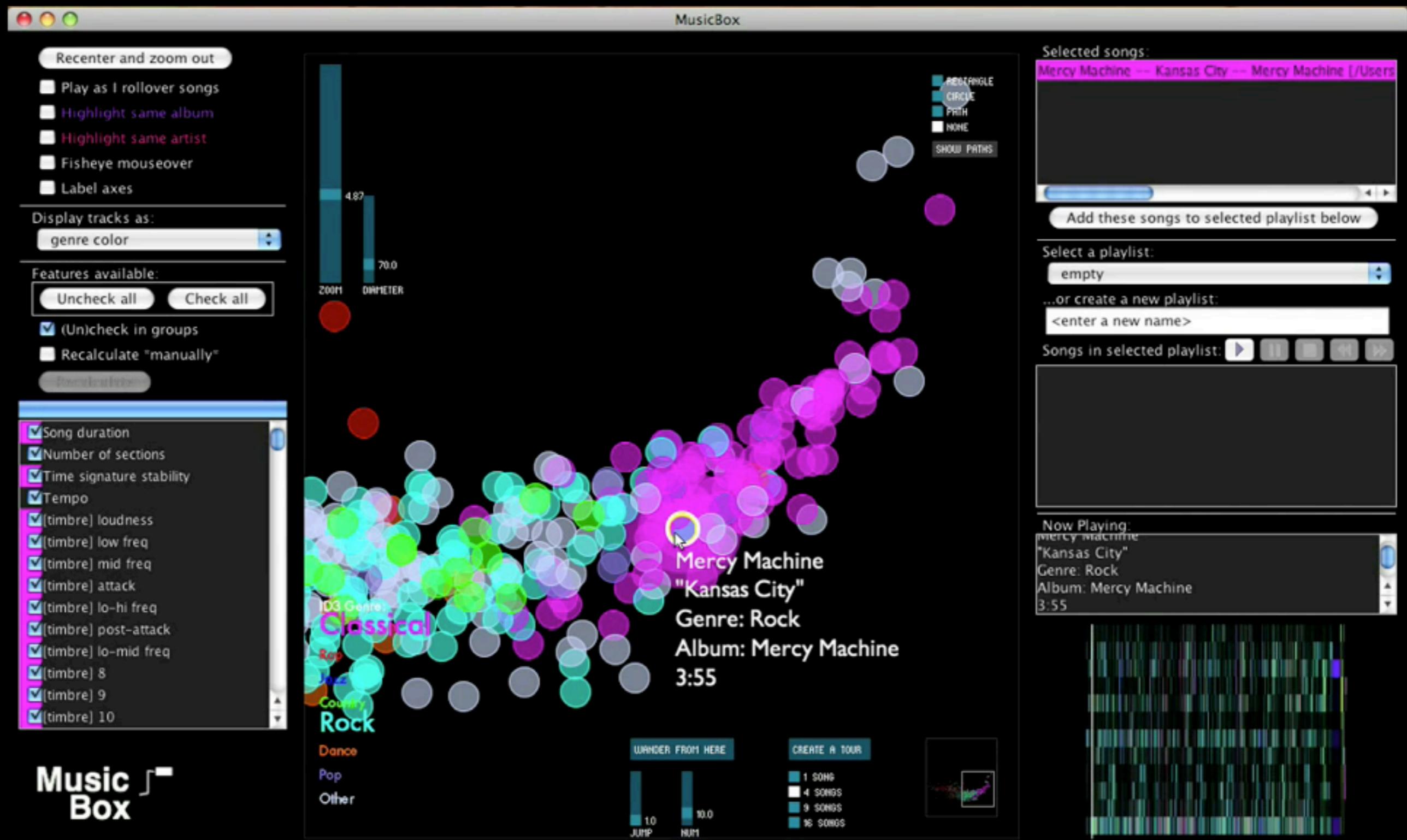


# Uses

- Dimensionality reduction for supervised learning
- Compression
- Visualization

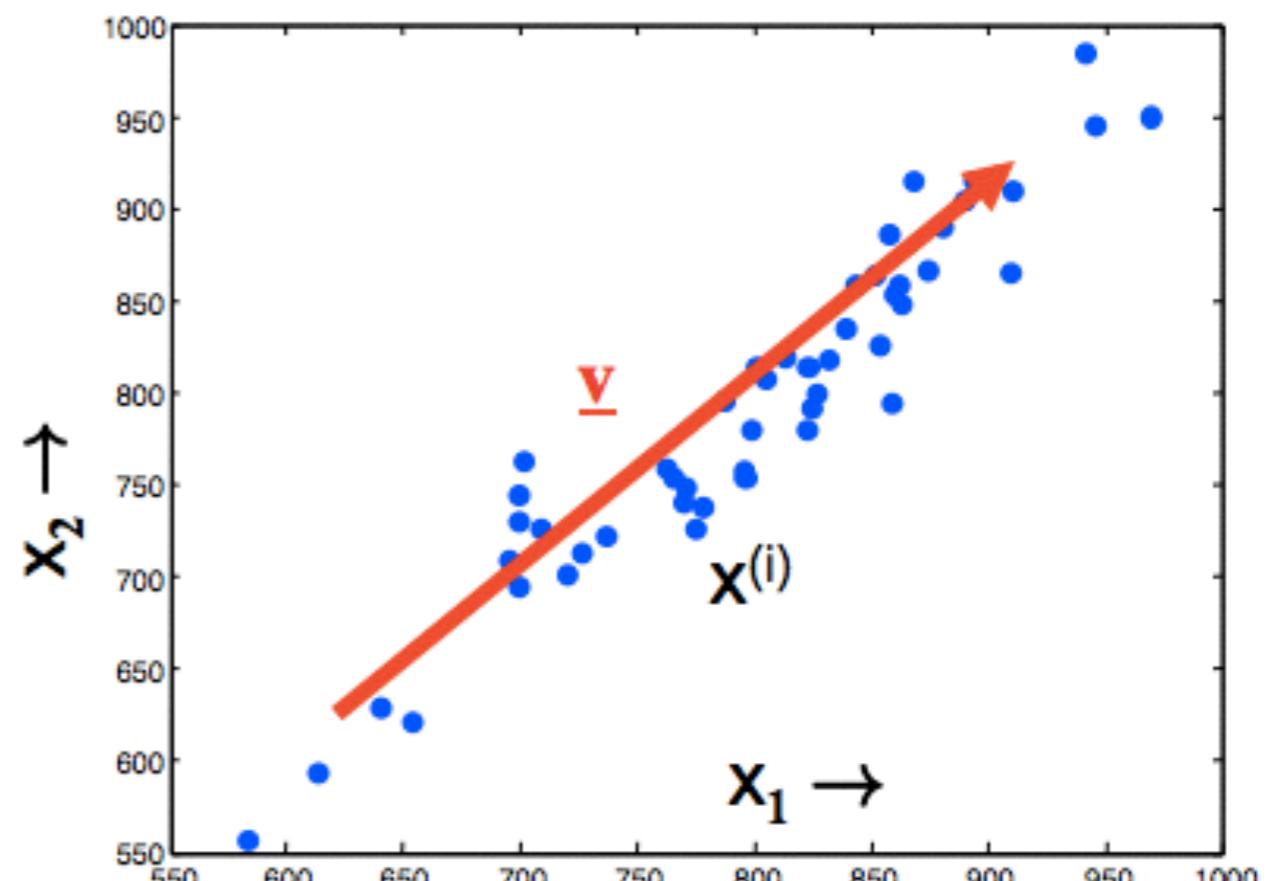
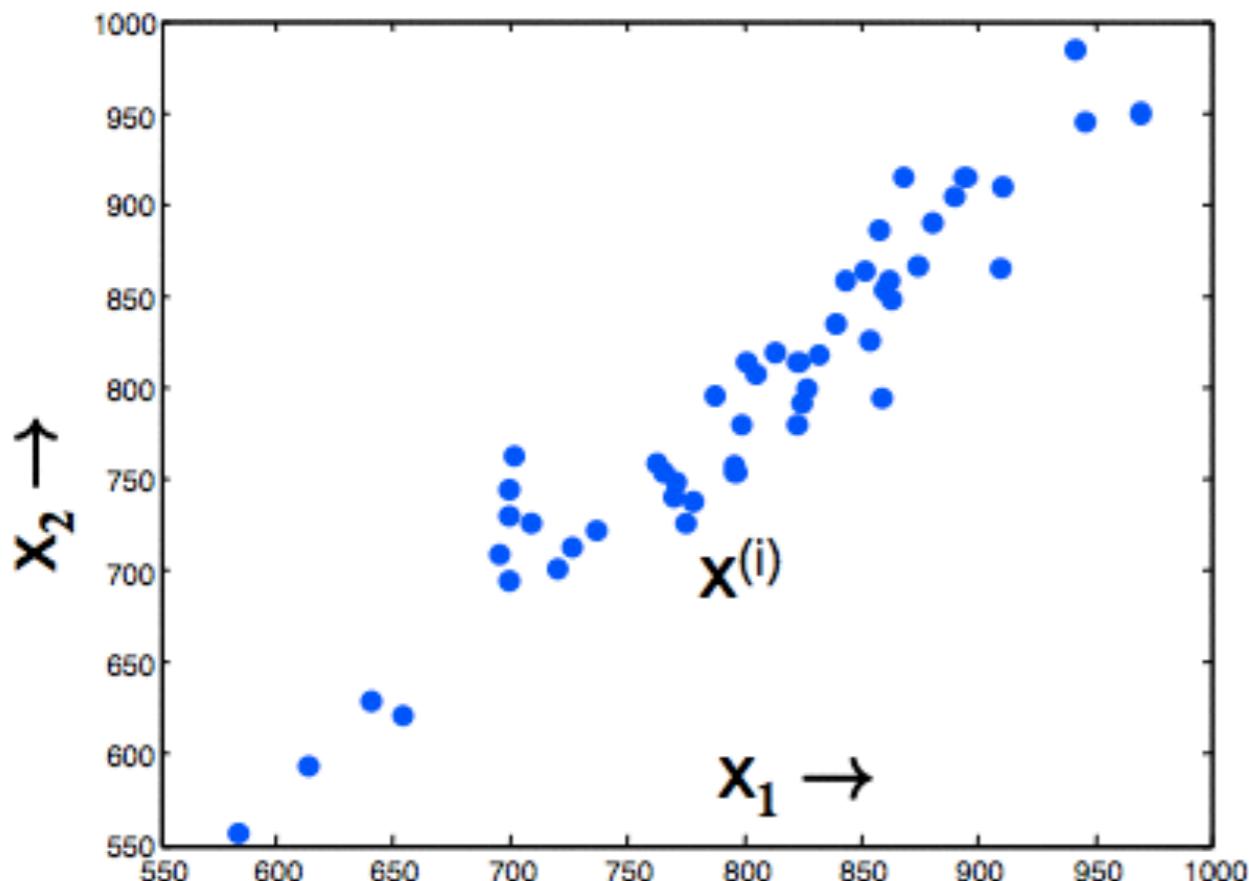


MusicBox, Anita Lillie, MIT

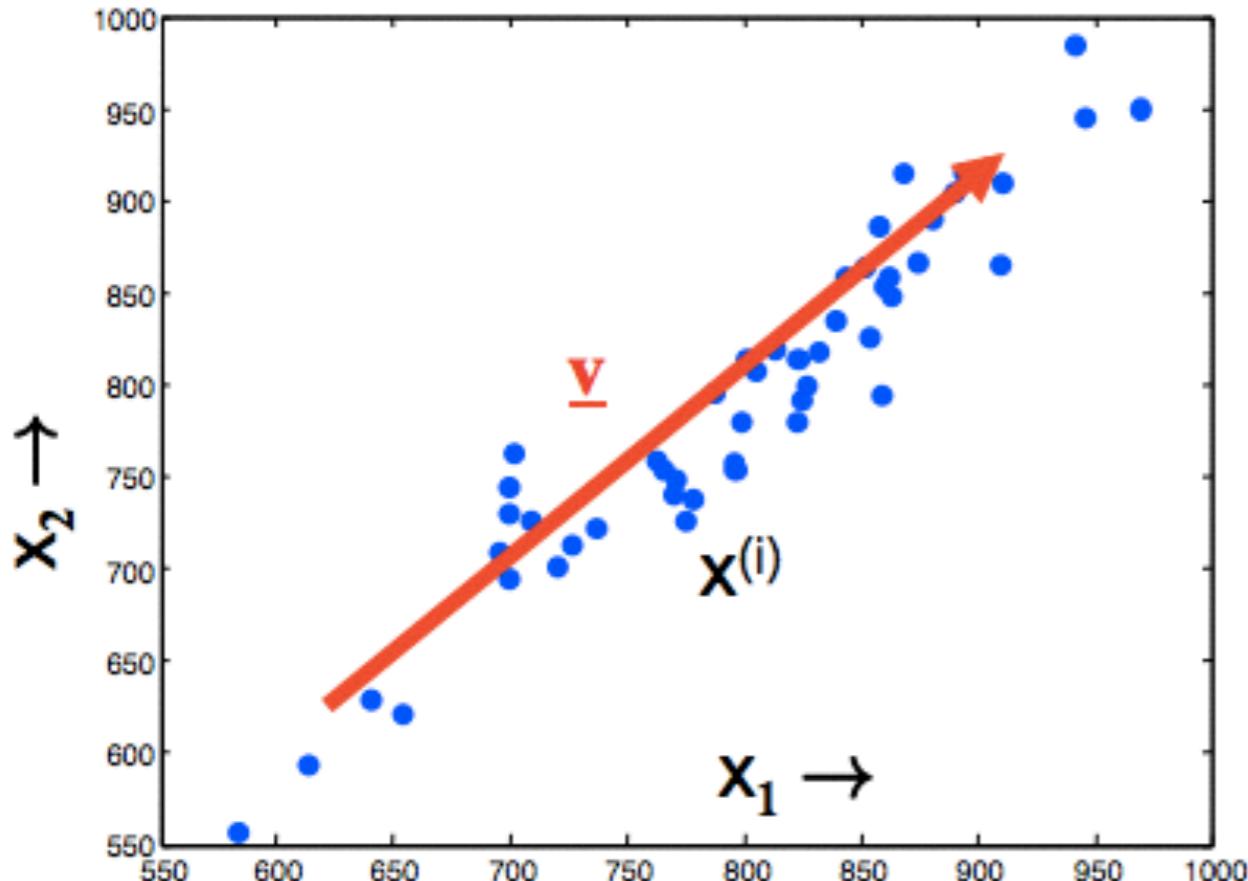


# **Principal Components Analysis (PCA)**

# Example



# Example

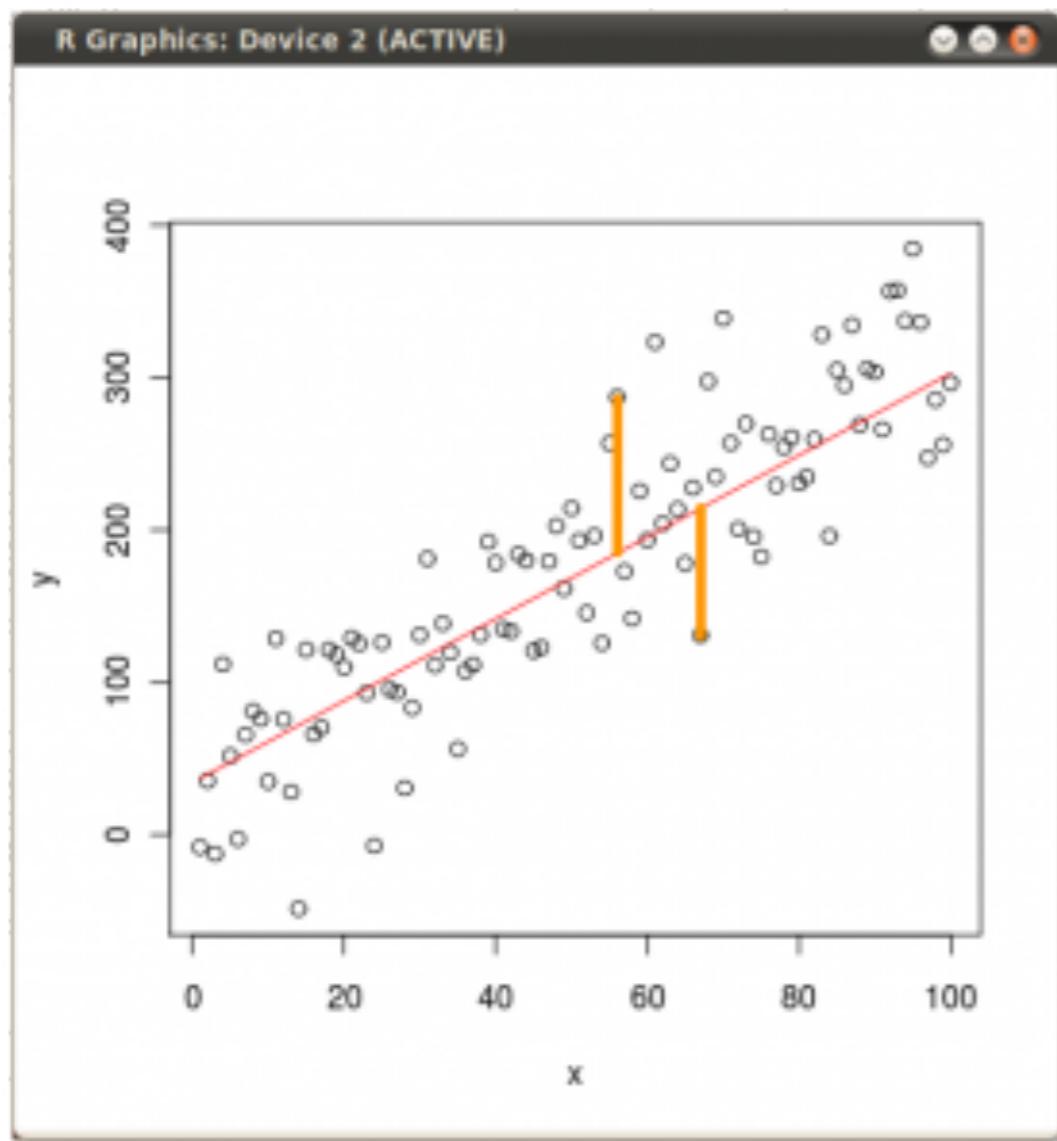


$v$ : chosen to minimize  
orthogonal distances

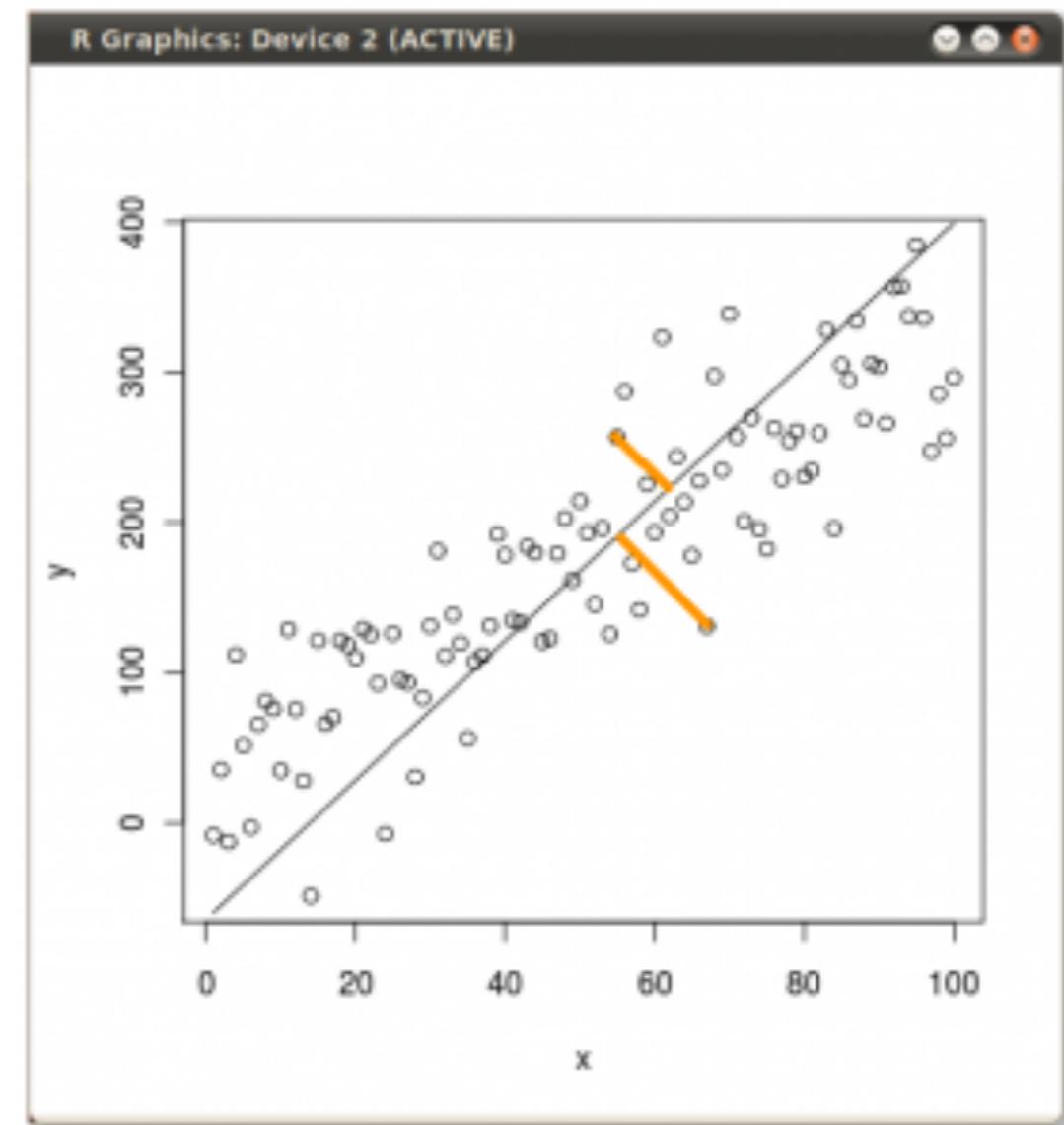
Equivalent:  $v$  is the direction of  
maximum variance  
(max. the spread along  $v$ )

# Linear Regression vs. PCA

Linear Regression



PCA

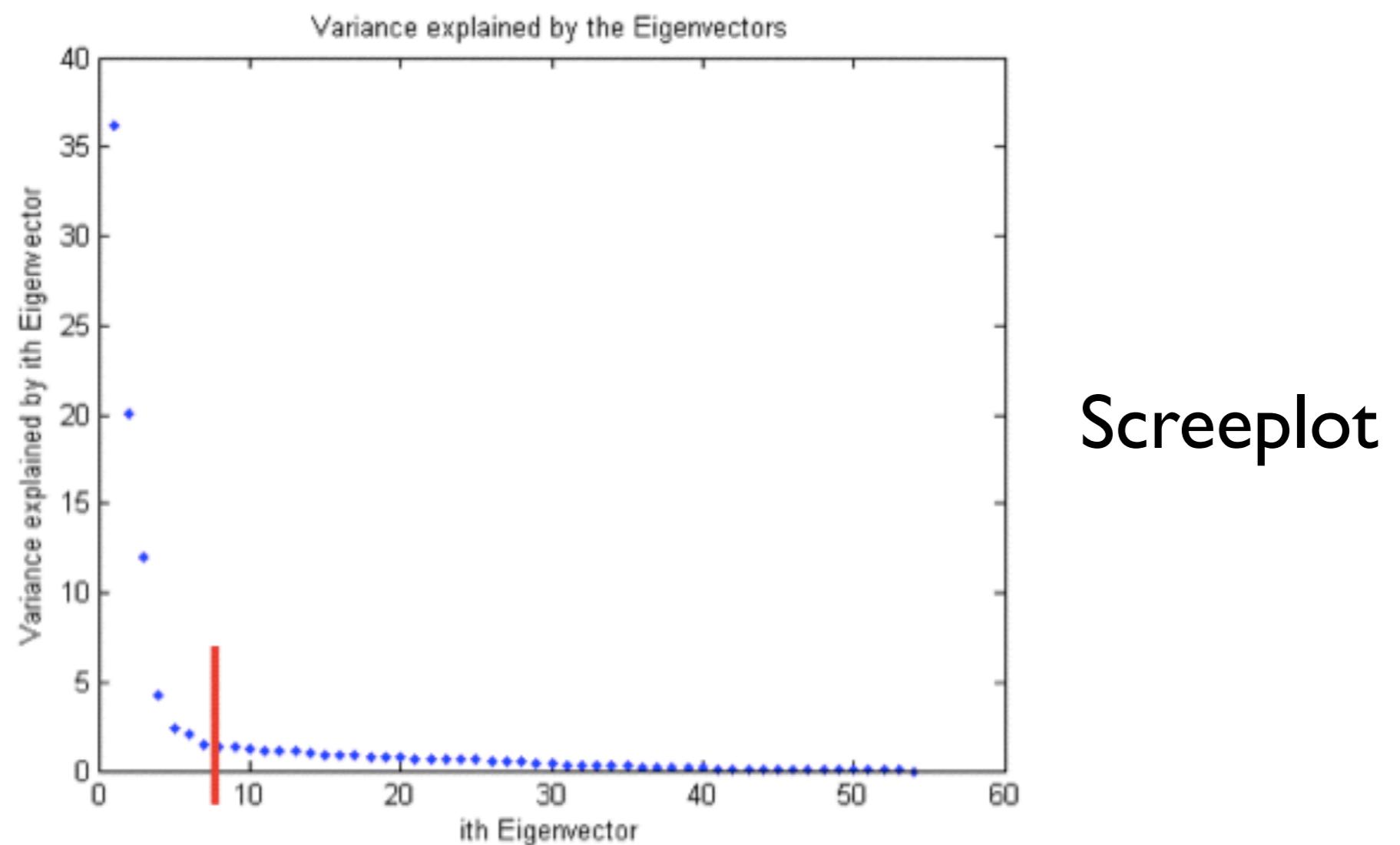


# PCA Algorithm

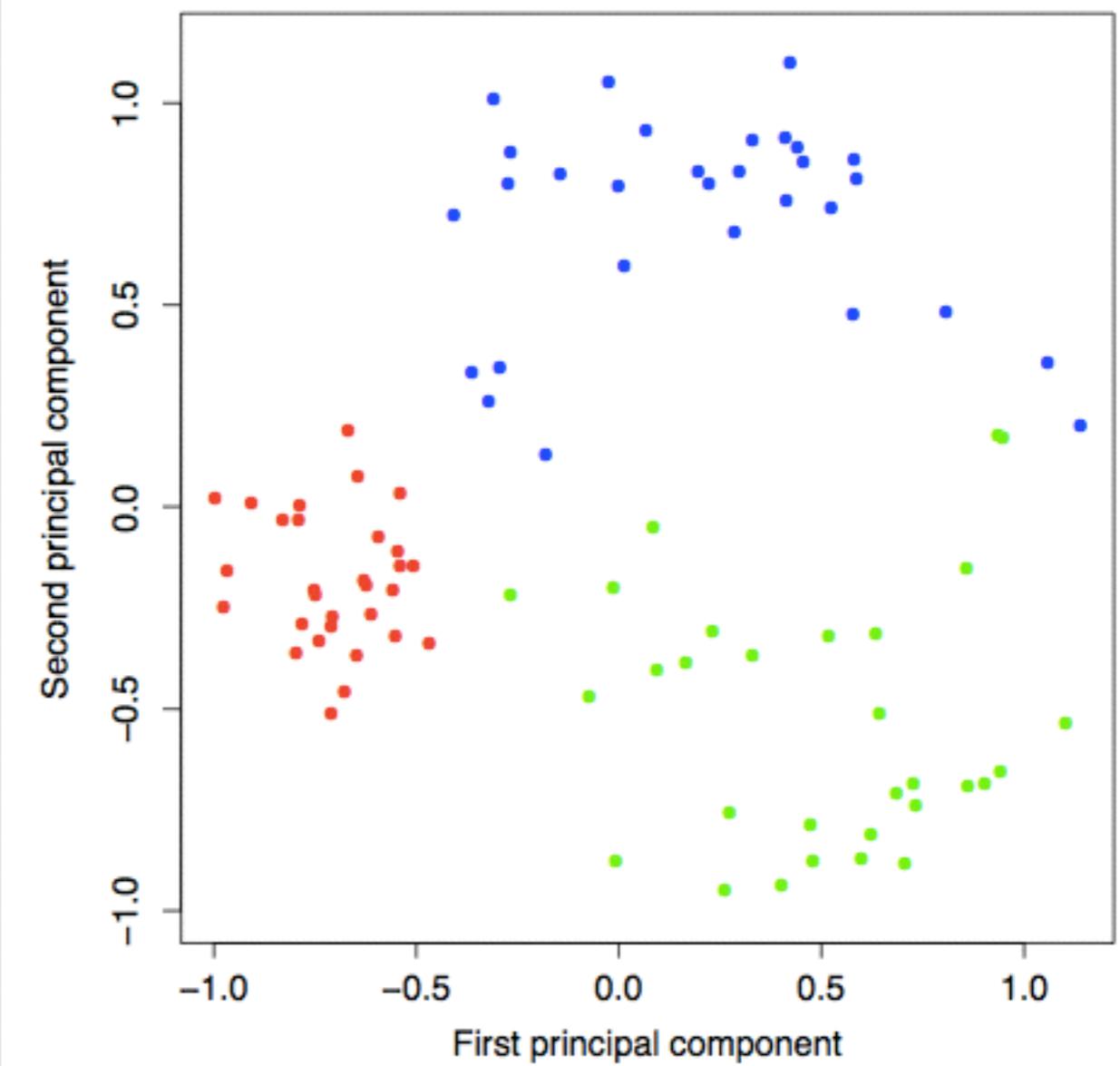
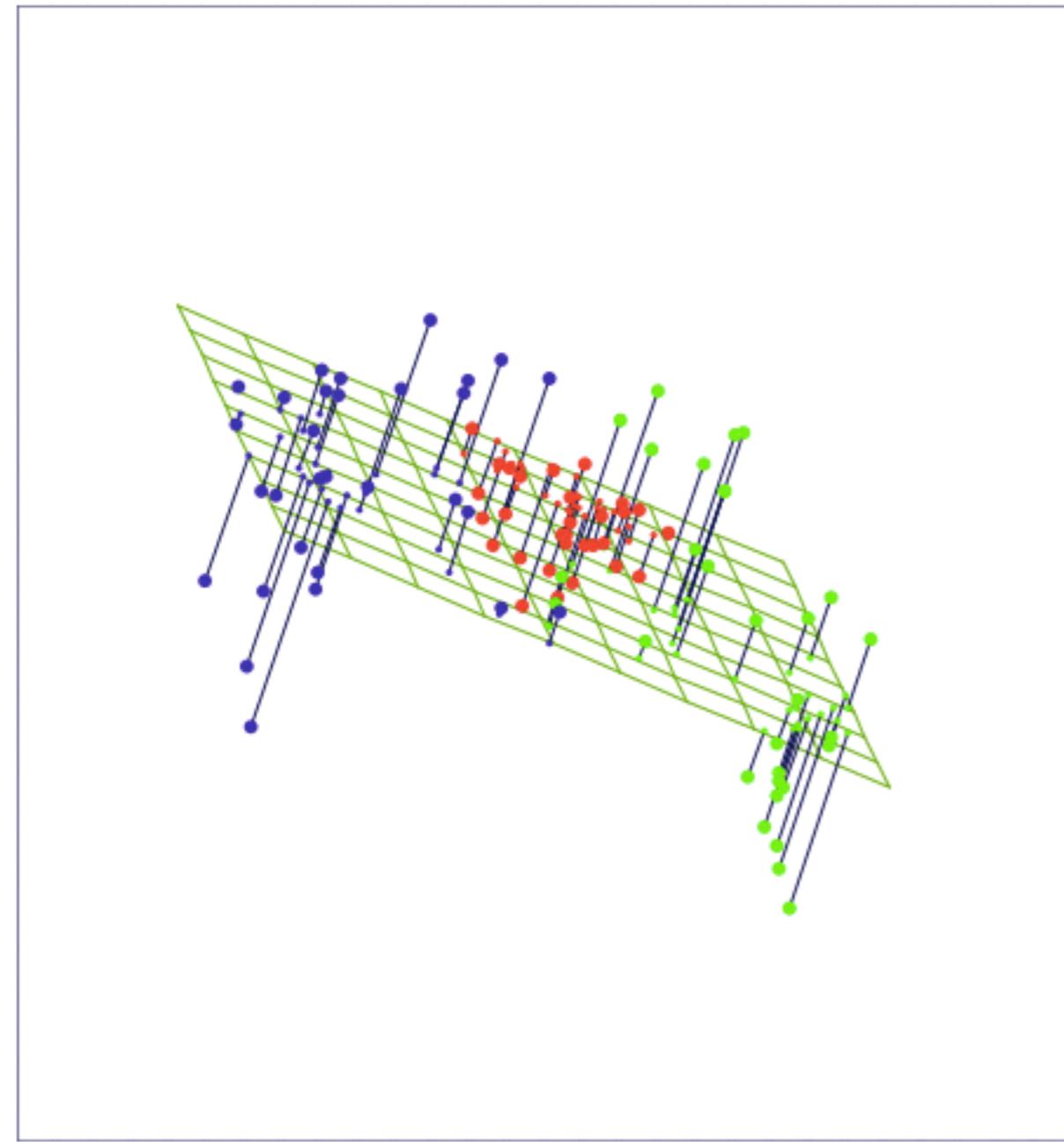
- Subtract mean from data (center  $\mathbf{X}$ )
  - (Typically) scale each dimension by its variance
    - Helps to pay less attention to magnitude of dimensions
  - Compute covariance matrix  $\mathbf{S}$
  - Compute  $k$  largest eigenvectors of  $\mathbf{S}$
  - These eigenvectors are the  $k$  principal components
- $$\mathbf{S} = \frac{1}{N} \mathbf{X}^\top \mathbf{X}$$

# How many PC vectors?

Enough PC vectors to cover 80-90% of the variance



# Dimensionality Reduction



# PCA for Handwritten Digits

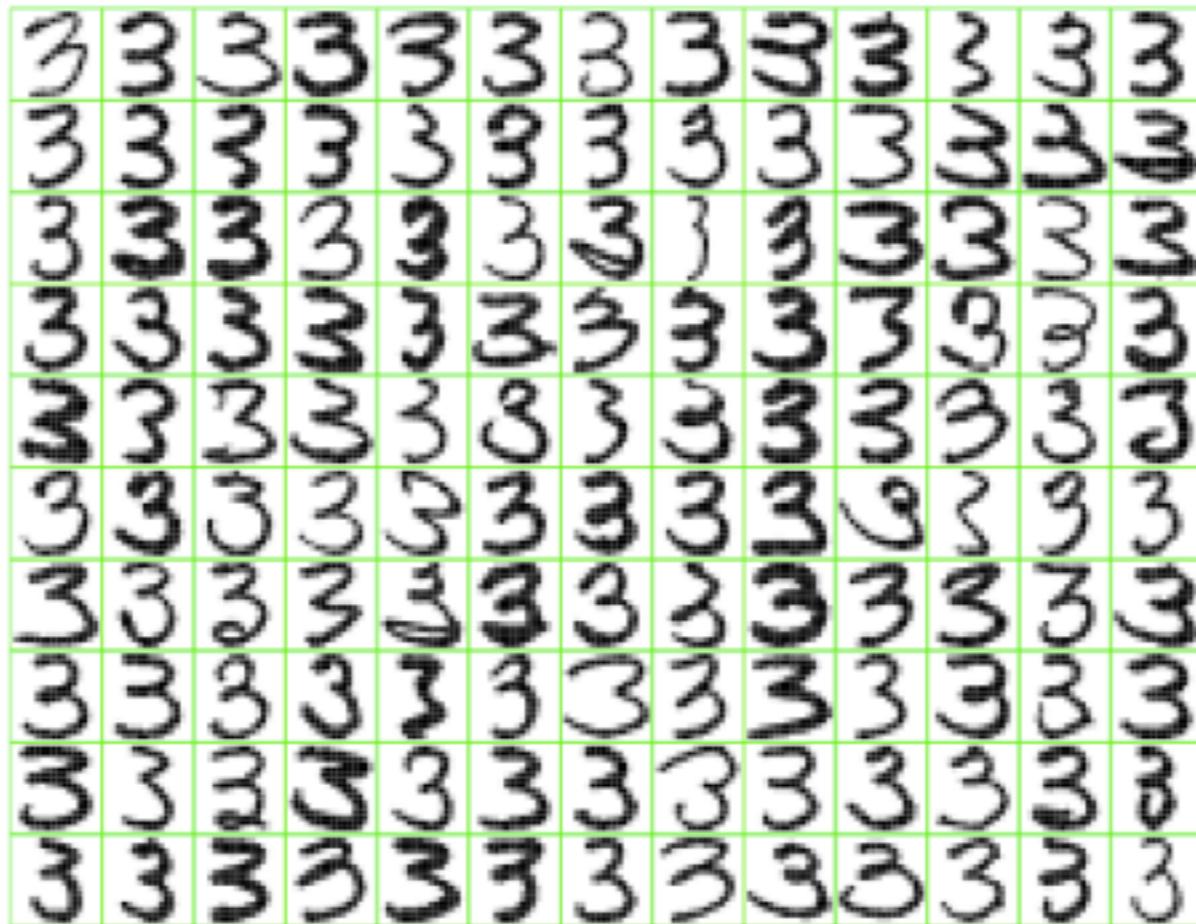


FIGURE 14.22. A sample of 130 handwritten 3's shows a variety of writing styles.

$$\begin{aligned}\hat{f}(\lambda) &= \bar{x} + \lambda_1 v_1 + \lambda_2 v_2 \\ &= \boxed{3} + \lambda_1 \cdot \boxed{3} + \lambda_2 \cdot \boxed{3}.\end{aligned}$$

# PCA for Handwritten Digits

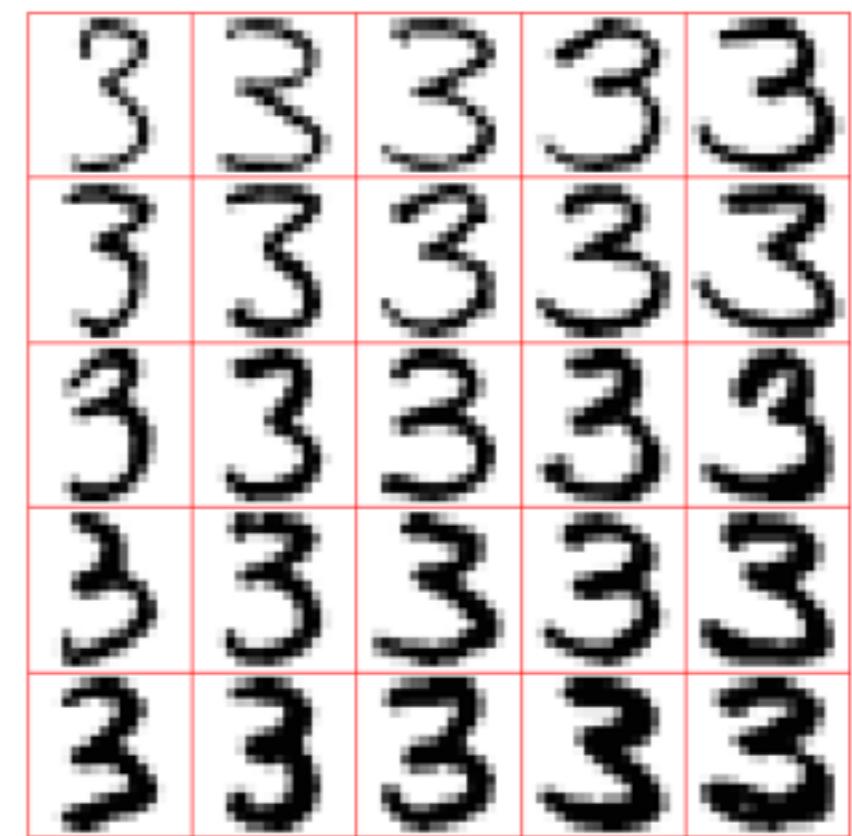
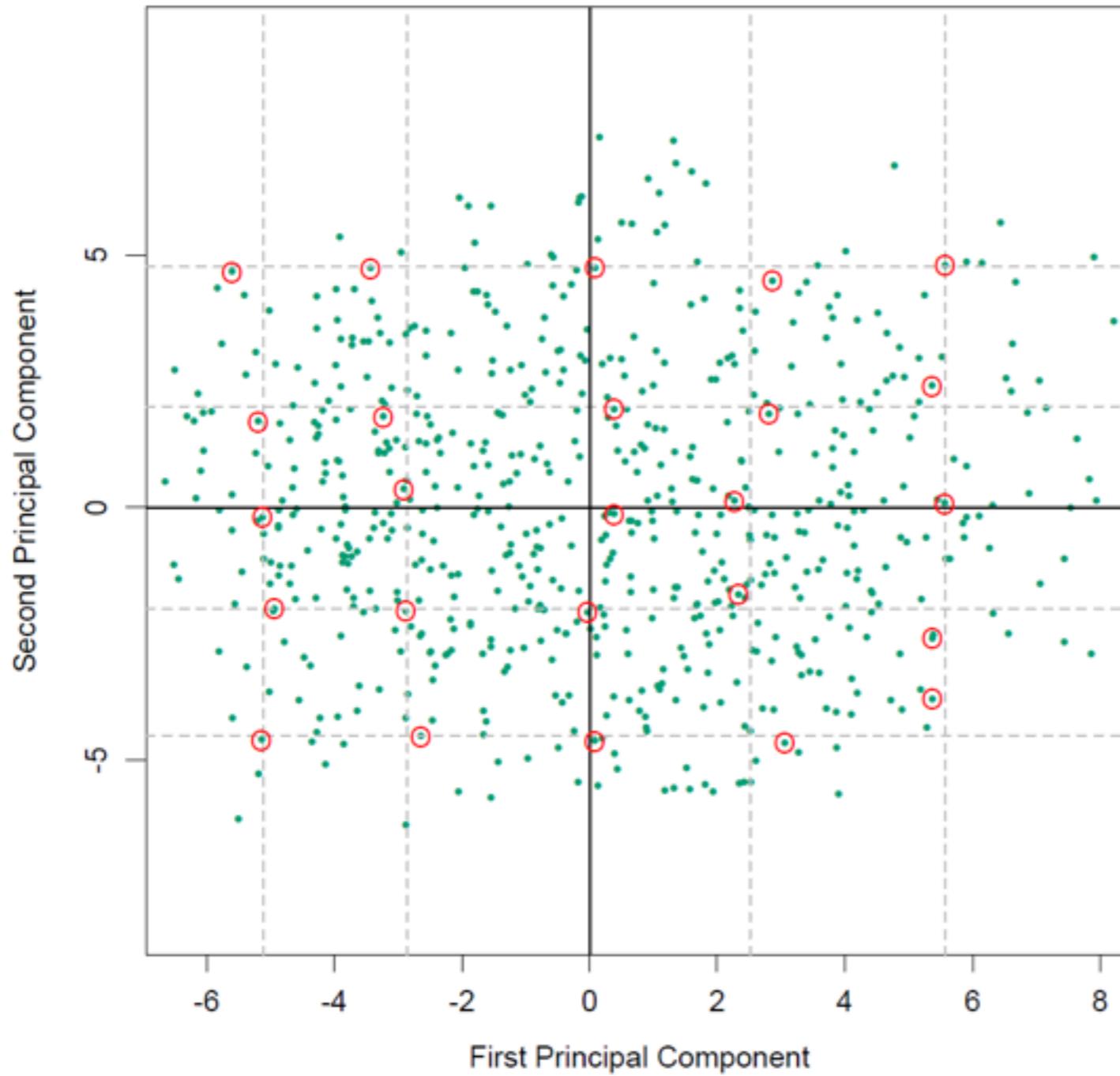


Image reconstruction after reduction with PCA



# PCA for Face Images



# PCA for Face Images

- 64x64 images of faces = 4096 dimensional data



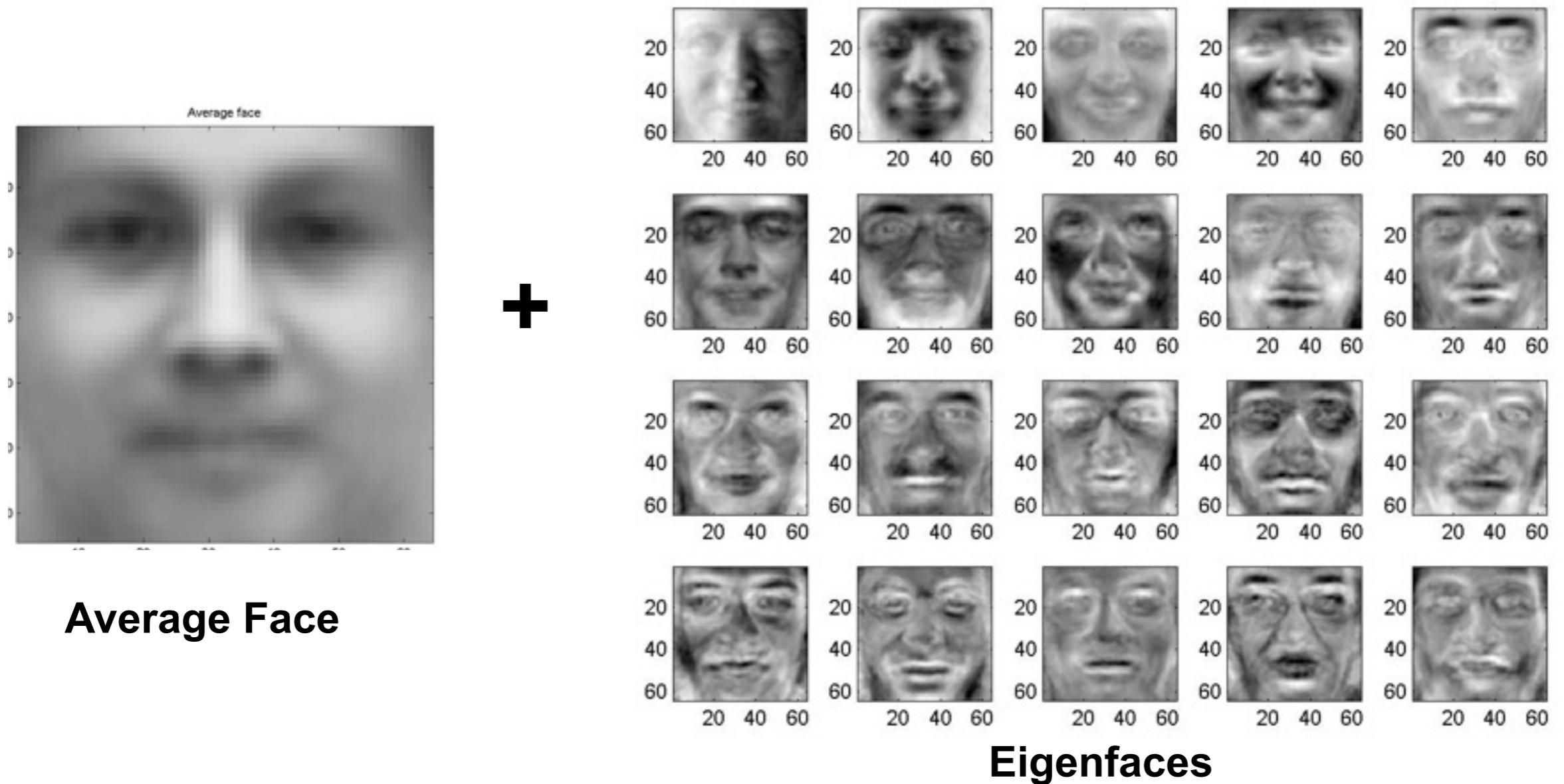
⋮

⋮

X  
N x D

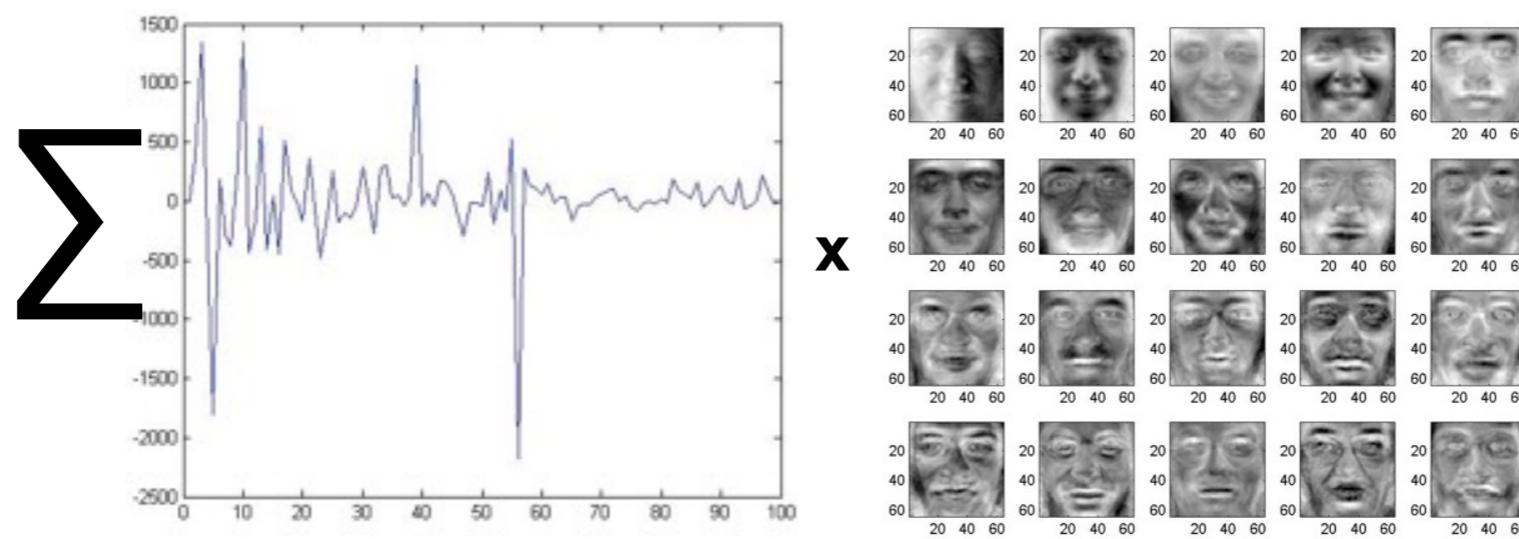
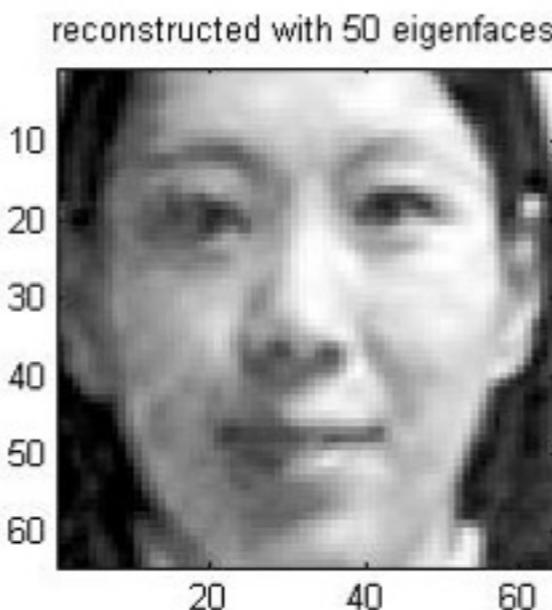
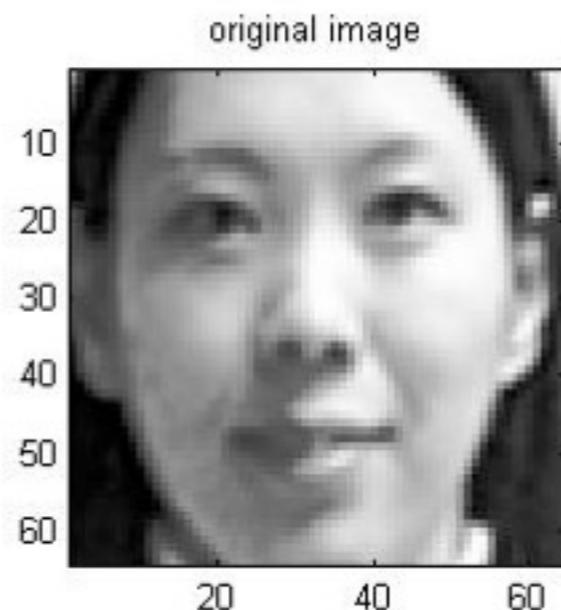
# “Eigenfaces”

We can reconstruct each face as a linear combination of “basis” PC vectors, or Eigenfaces [M.Turk and A. Pentland (1991)]



# Reconstruction

90% variance is captured by the first 50 eigenvectors



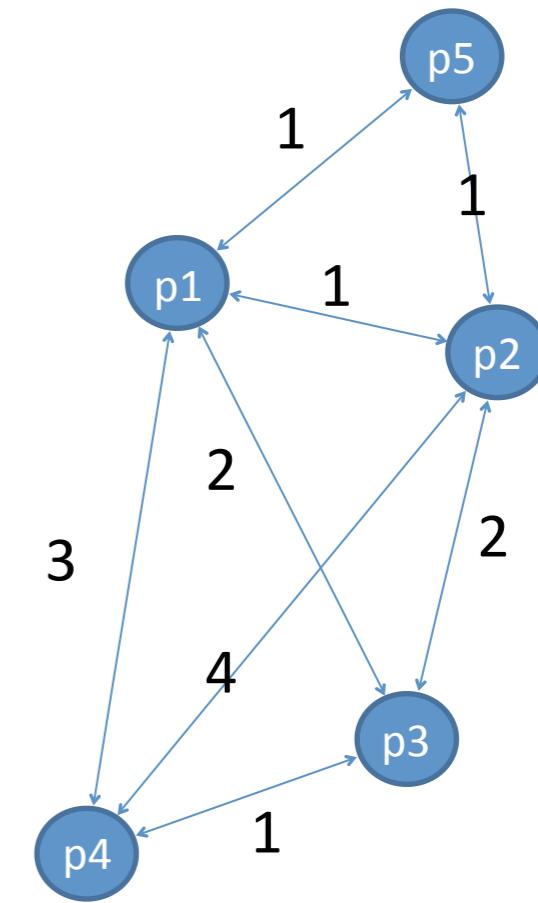
Based on slide from T. Yang

# **Multidimensional Scaling (MDS)**

# Multi-Dimensional Scaling

- A different goal :
  - Find a set of points whose pairwise distances match a given distance matrix

	p1	p2	p3	p4	p5
p1	0	1	2	3	1
p2	1	0	2	4	1
p3	2	2	0	1	3
p4	3	4	1	0	1
p5	1	1	3	1	0



# Classical MDS

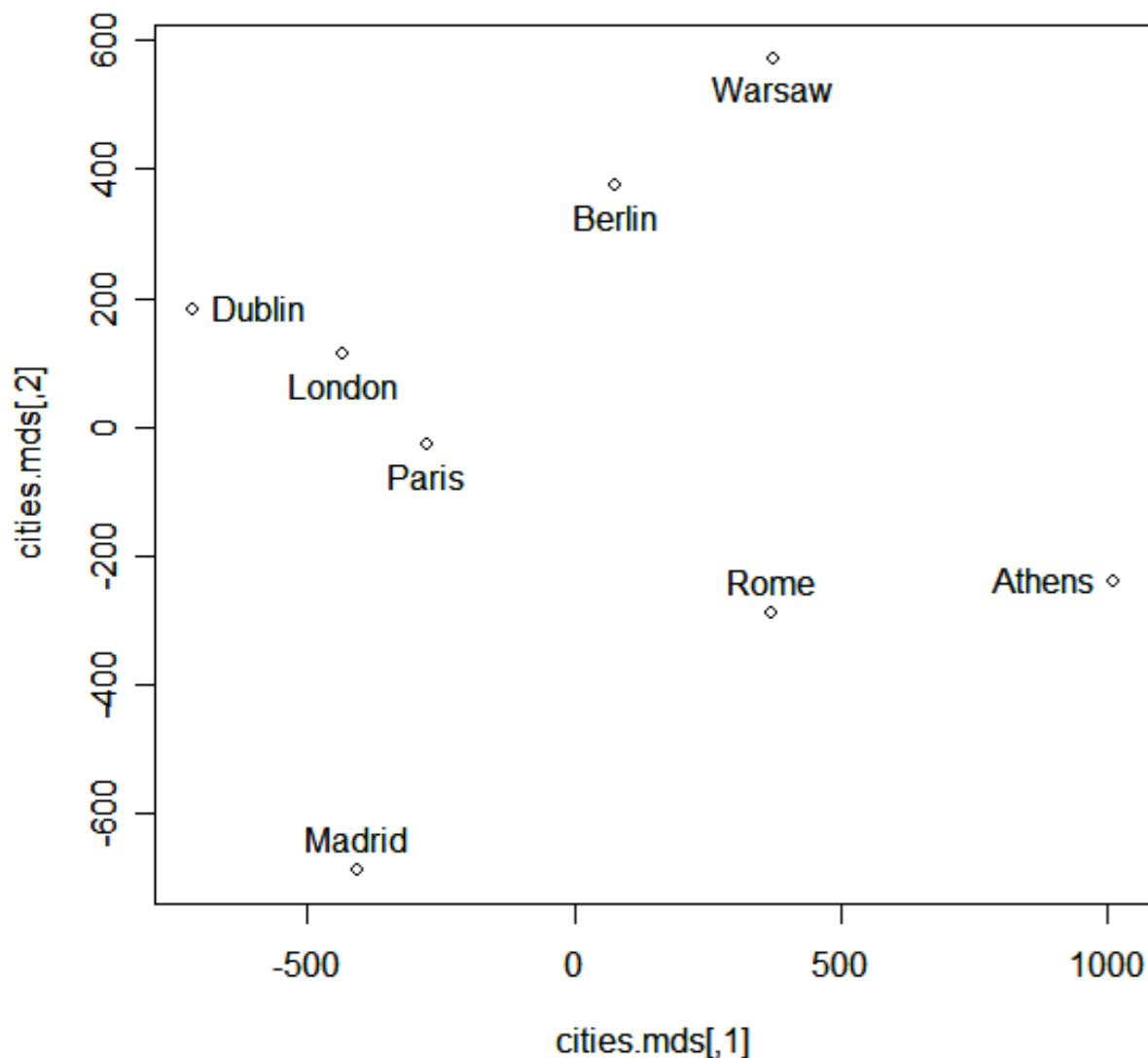
- Given  $n \times n$  matrix of pairwise distances between data points
- Compute  $n \times k$  matrix  $X$  with coordinates of distances with some linear algebra magic
- Perform PCA on this matrix  $X$

# European Cities Data

- Distances between European cities:

	Athens	Berlin	Dublin	London	Madrid	Paris	Rome	Warsaw
Athens	0	1119	1777	1486	1475	1303	646	1013
Berlin	1119	0	817	577	1159	545	736	327
Dublin	1777	817	0	291	906	489	1182	1135
London	1486	577	291	0	783	213	897	904
Madrid	1475	1159	906	783	0	652	856	1483
Paris	1303	545	489	213	652	0	694	859
Rome	646	736	1182	897	856	694	0	839
Warsaw	1013	327	1135	904	1483	859	839	0

# Result of MDS

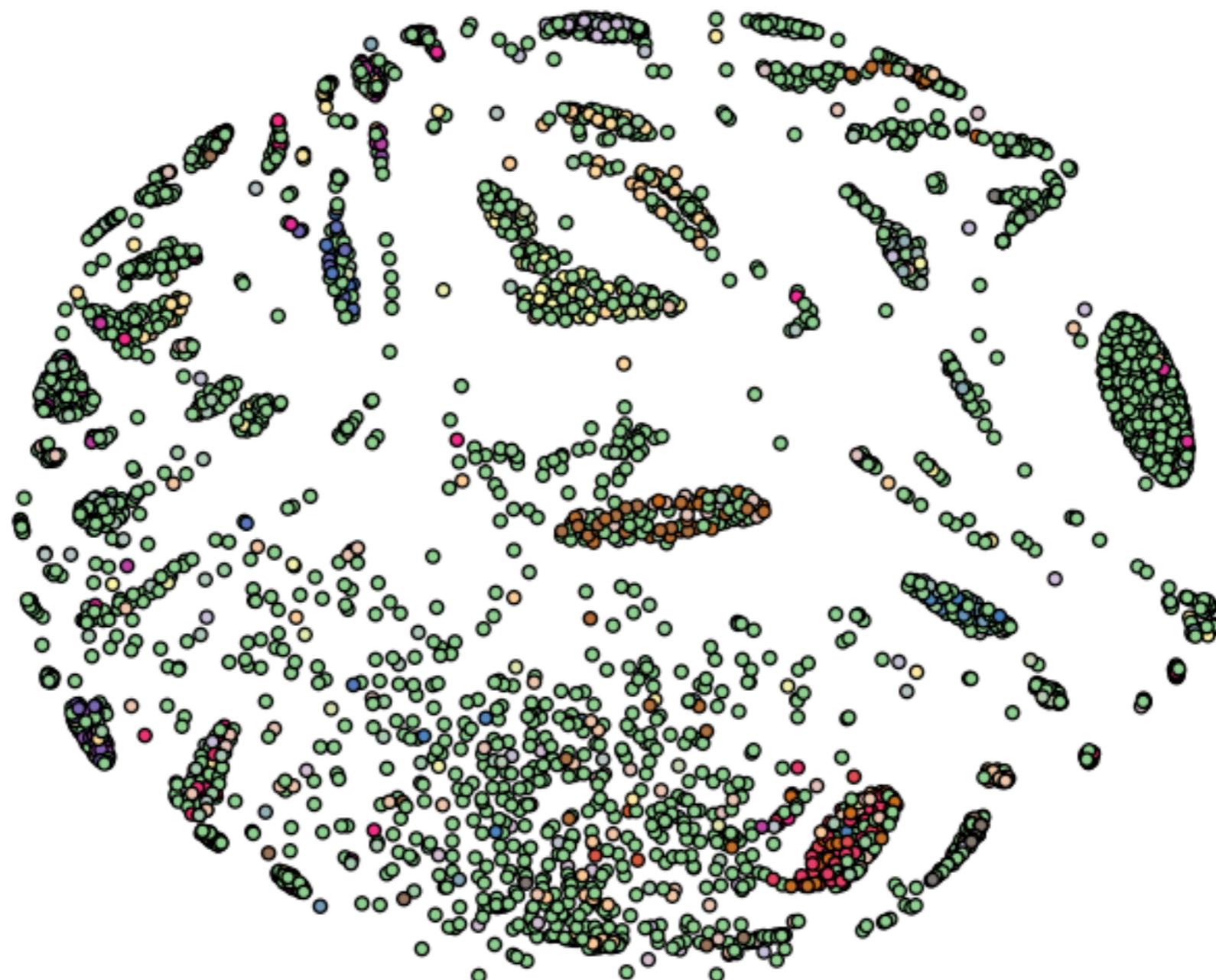


Based on slide from T. Yang

# Color Images



# Facebook Friends

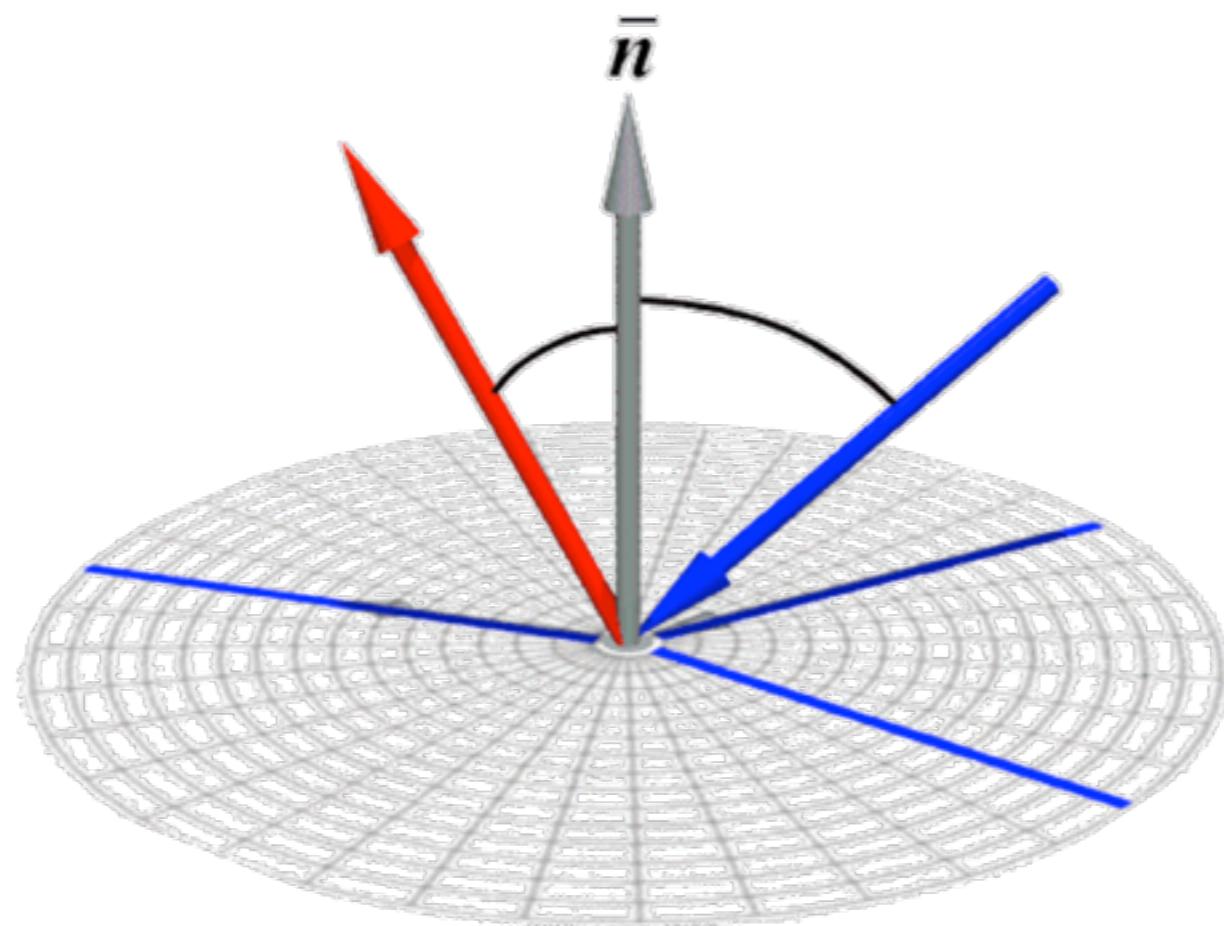


- Distance = 1 for friends
- Distance = 2 for friends of friends ; etc.

# Nonlinear Methods

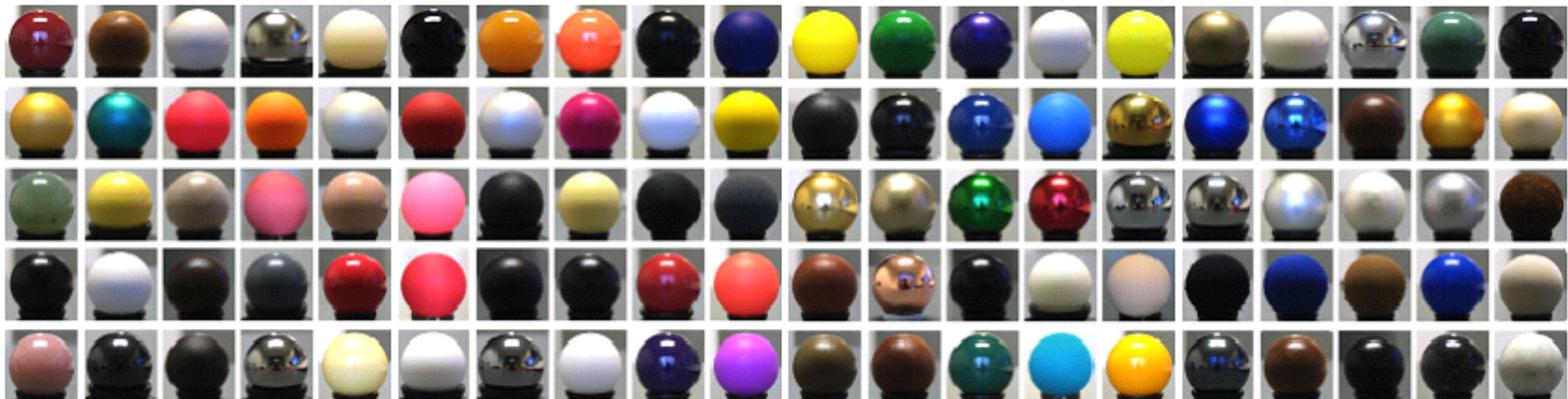
# Data-Driven BRDFs

- Bi-Directional Reflectance Distribution Functions



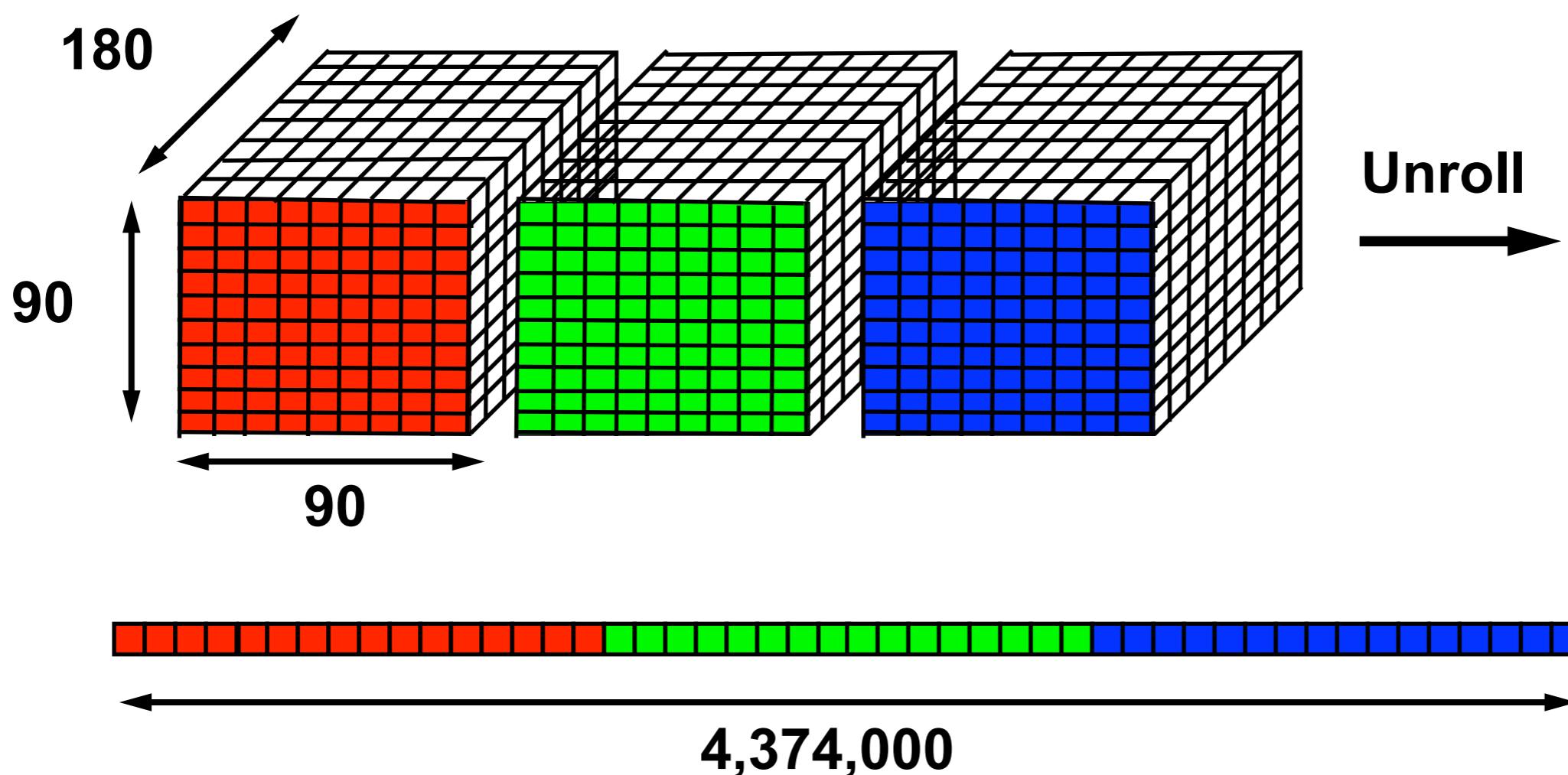
# Data-Driven BRDFs

- Measure light reflected off a sphere
- 20-80 million measurements (6000 images) per material



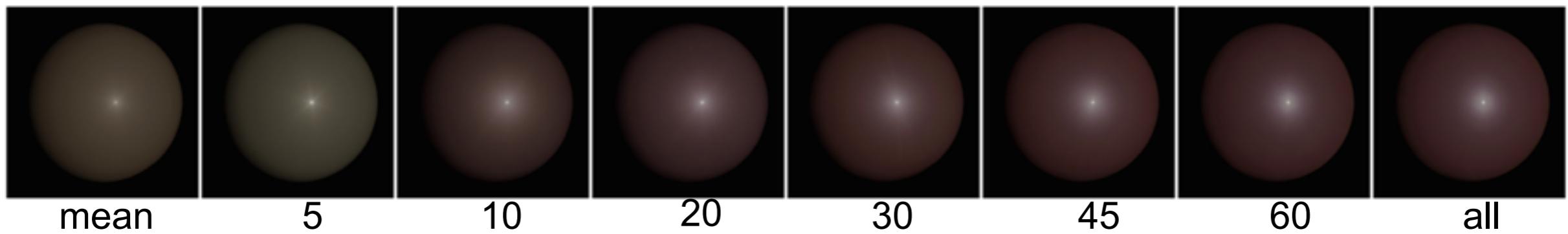
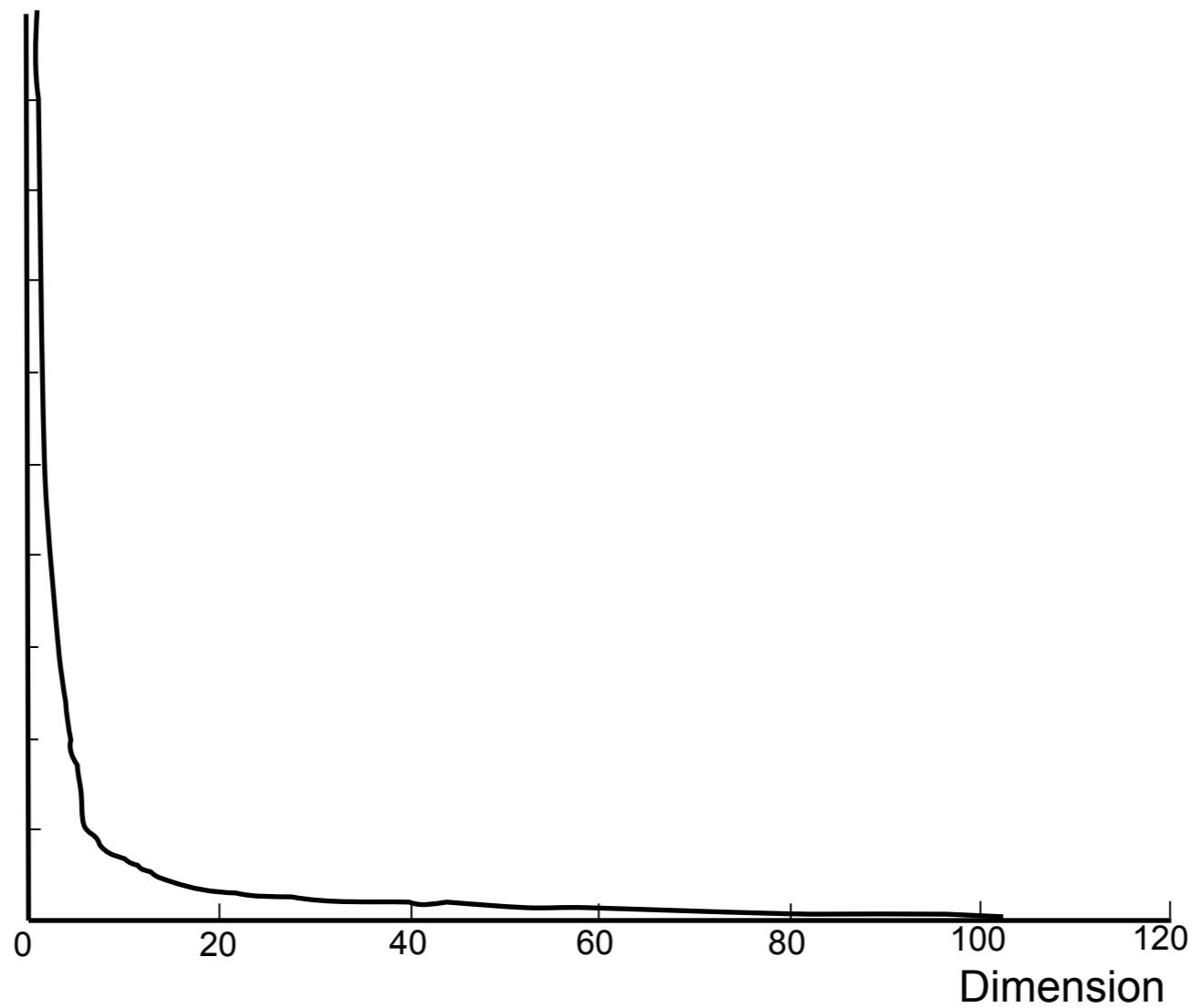
# Data-Driven BRDFs

- Each tabulated BRDF is a vector in  $90 \times 90 \times 180 \times 3 = 4,374,000$  dimensional space



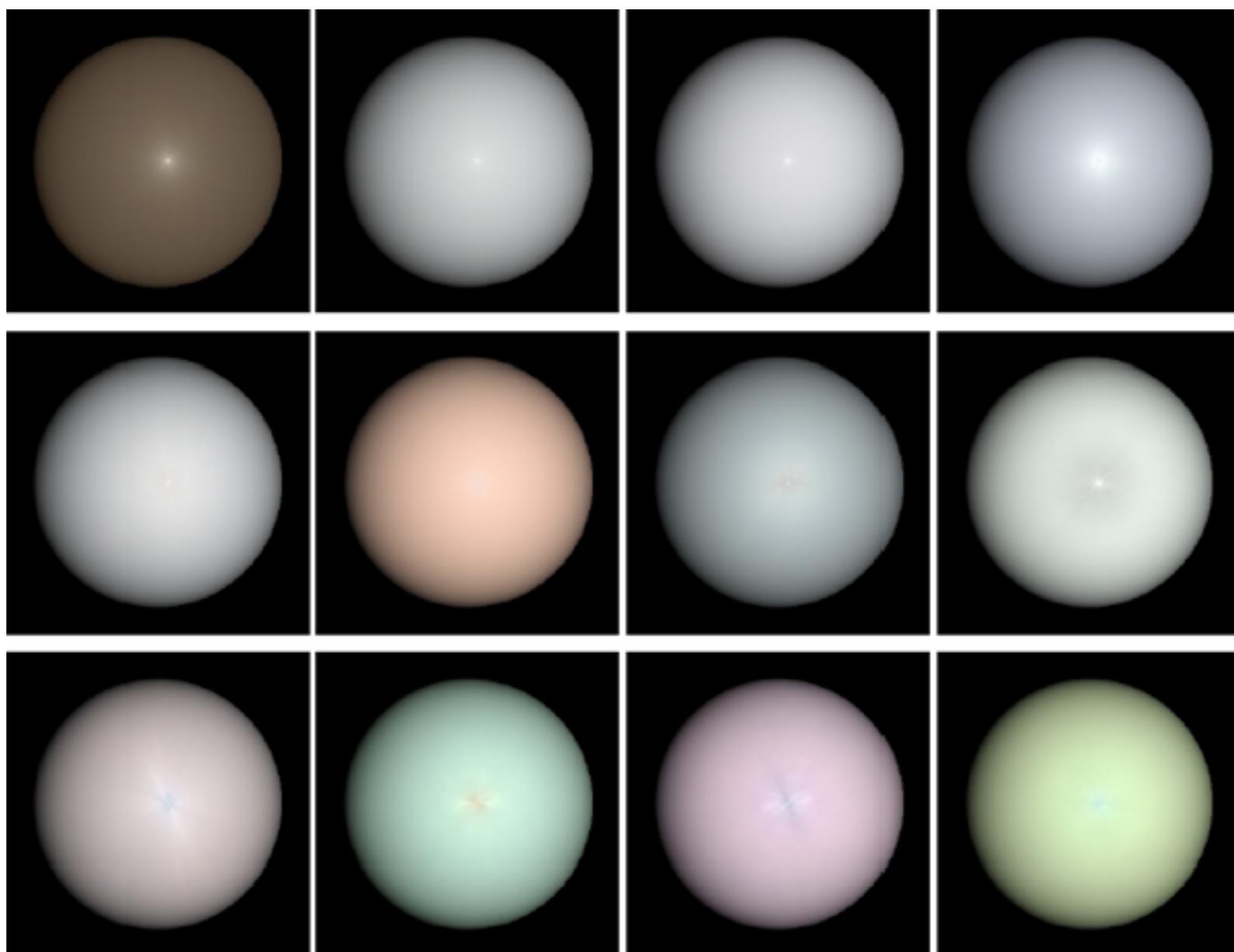
# PCA

Eigenvalue  
magnitude



# PCA

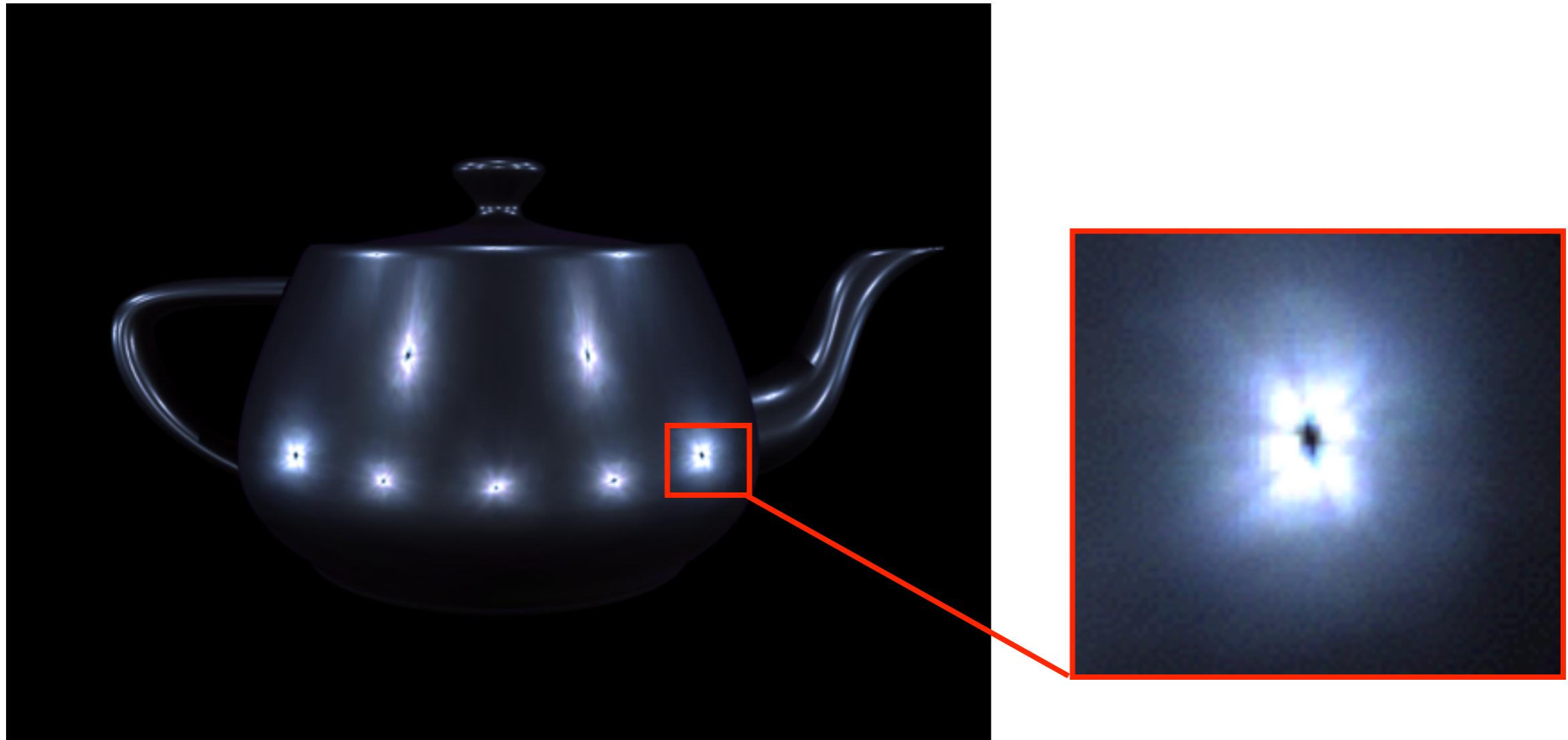
- First 11 PCA components



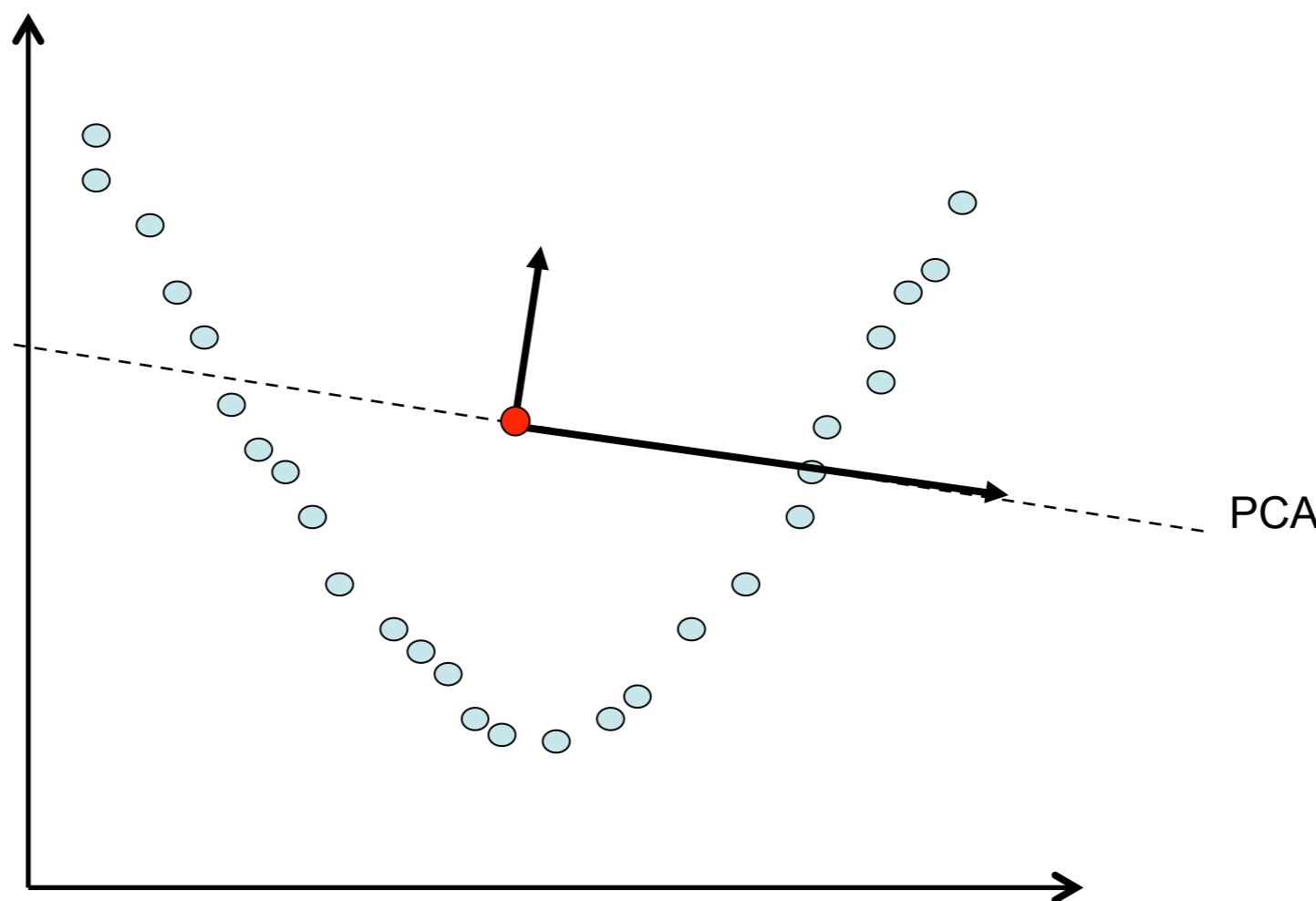
# PCA Interpolation



# Then, one day...

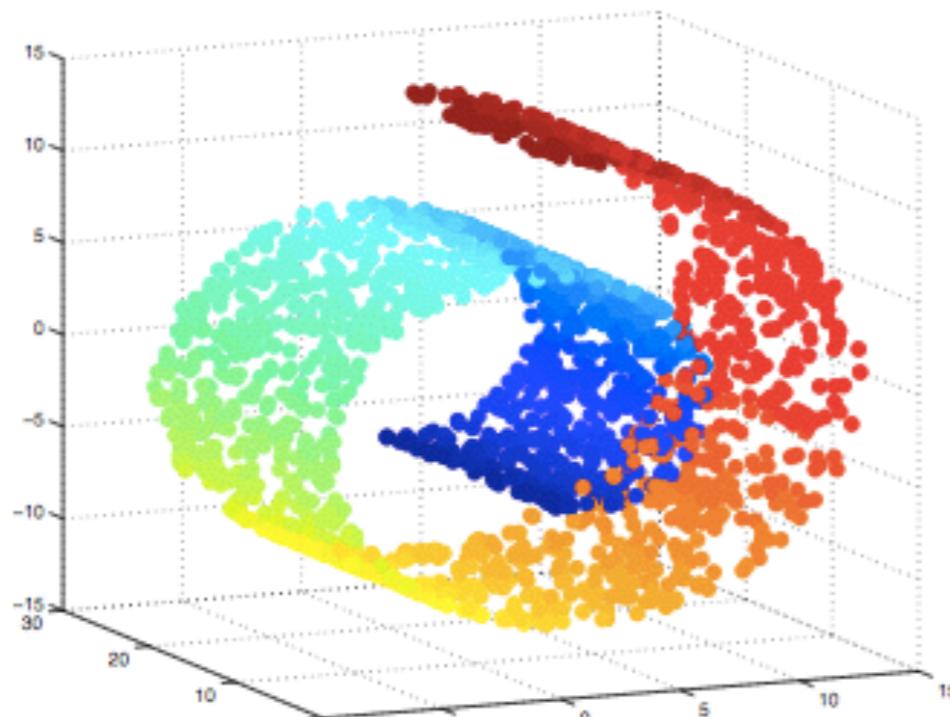


# Why do linear models fail?

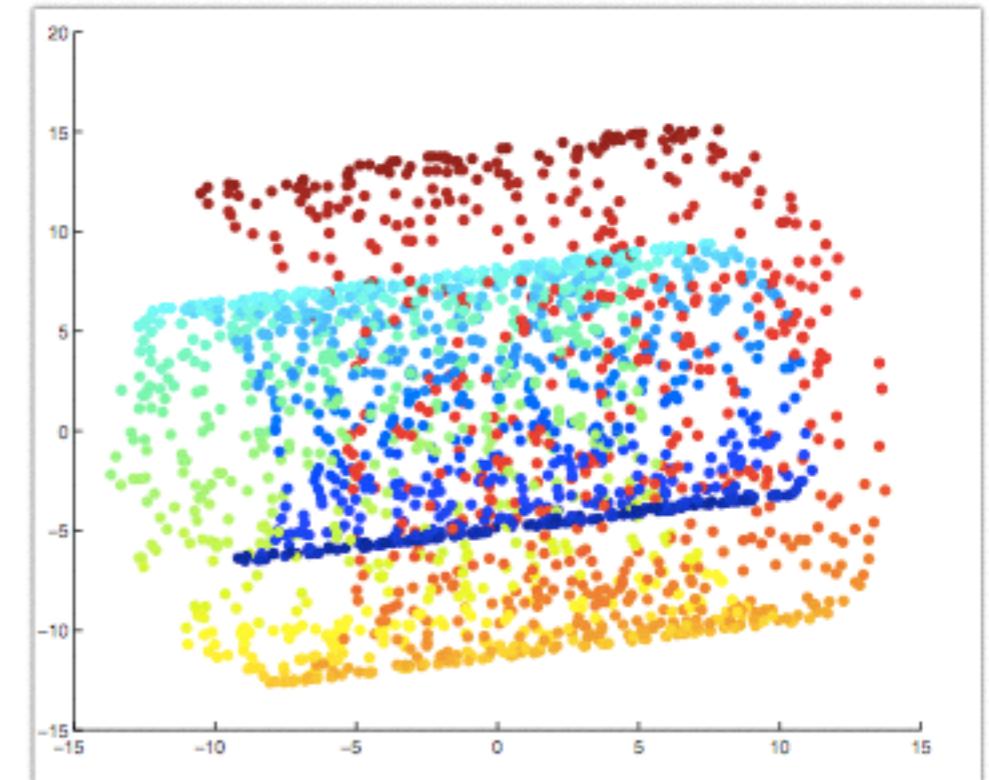


# Why do linear models fail?

- Classic “Swiss Roll” example

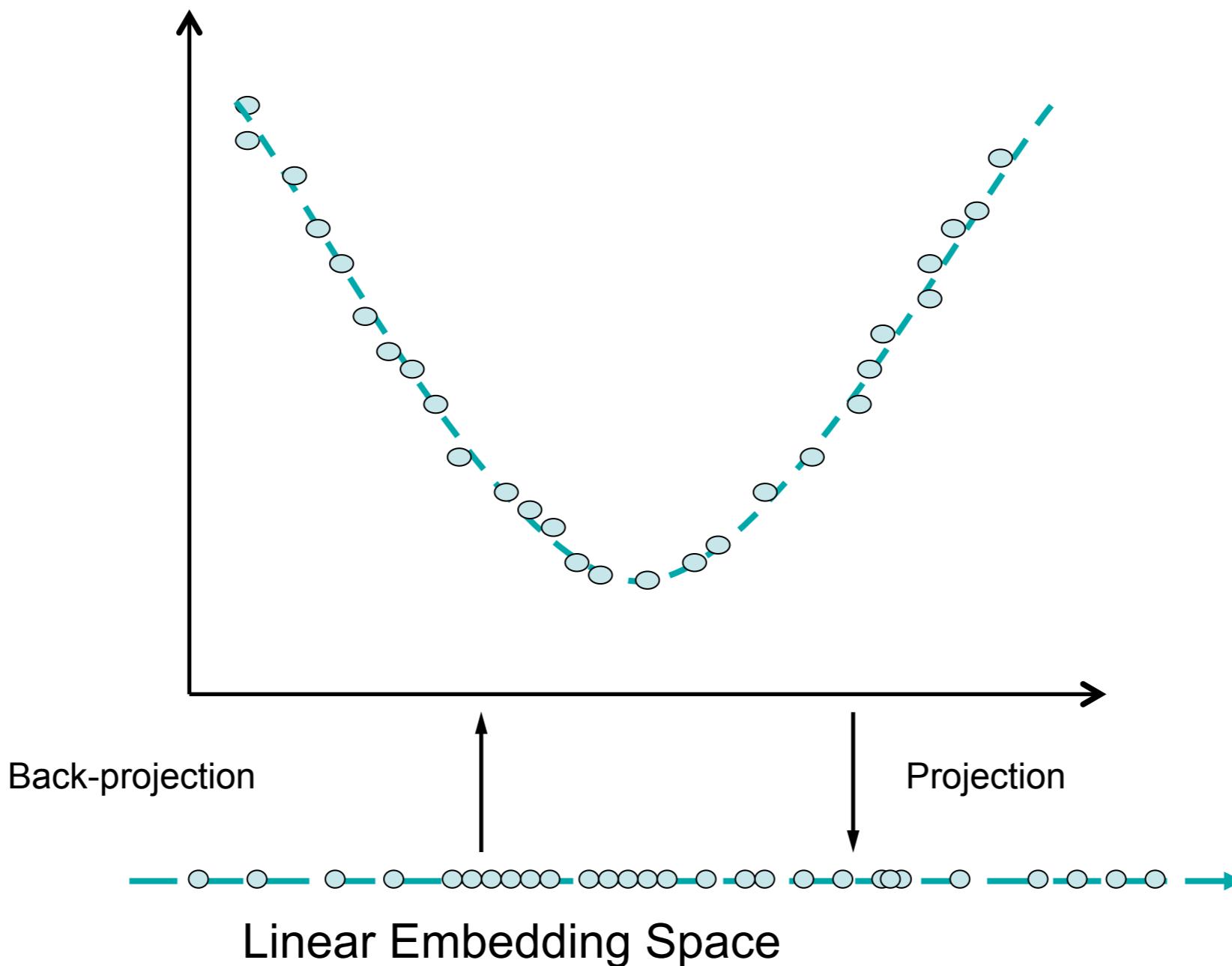


$x_i$



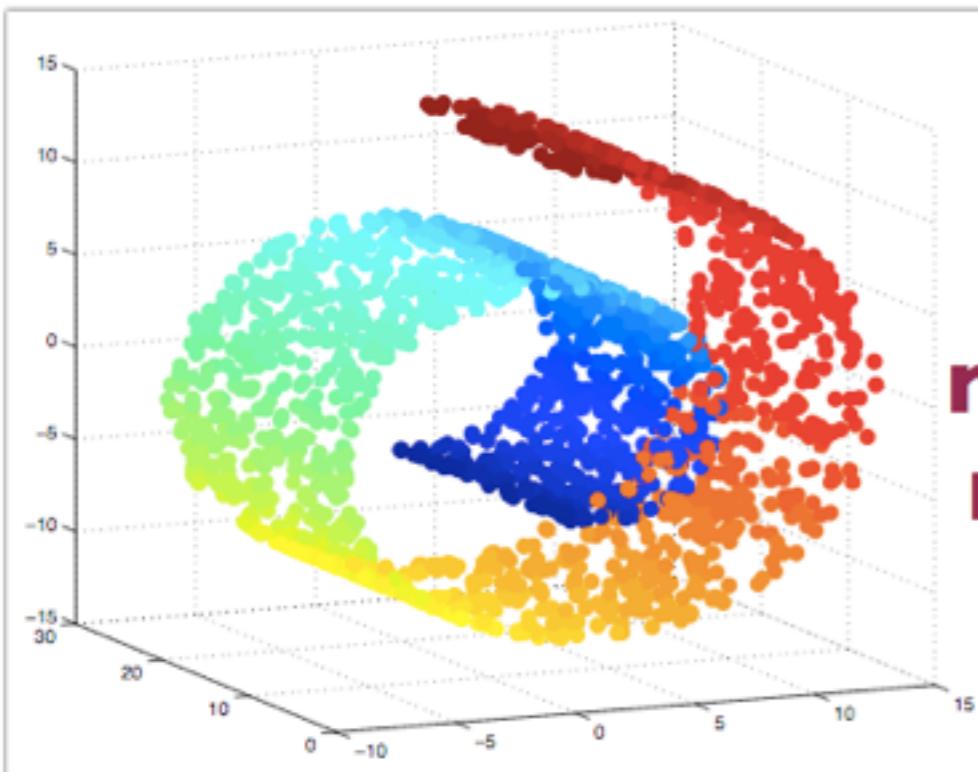
PCA

# Non-Linear Manifold Methods

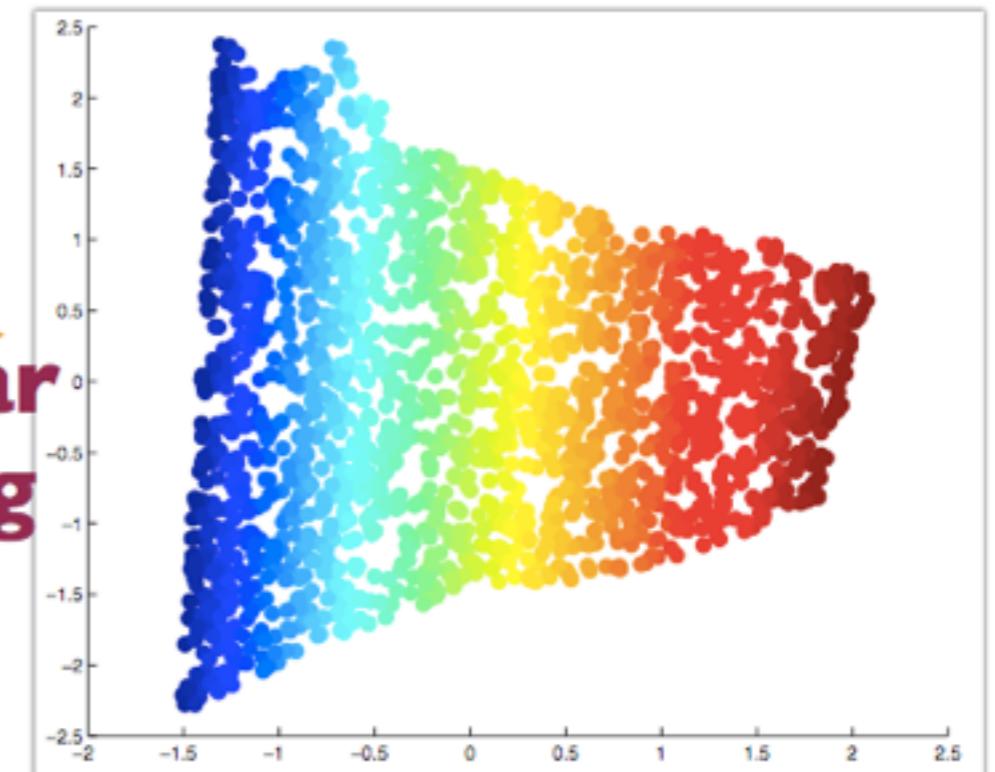


# Non-Linear Manifold Methods

- Intuition: Distortion in local areas, but faithful in the global structure

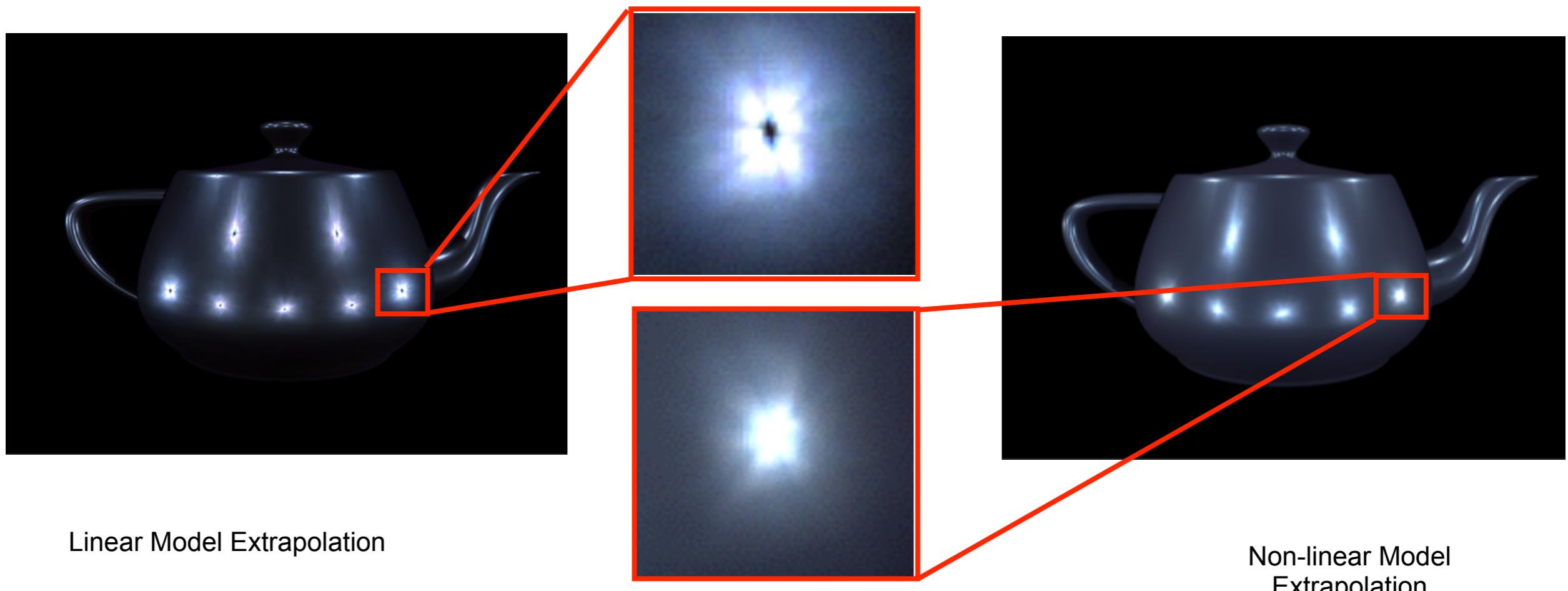


**nonlinear  
mapping**



# Non-Linear BRDF Model

- 15-dimensional space (instead of 45 PCs)
- More robust - allows extrapolations



# Dimensionality Reduction

- Linear methods:
  - Principal Component Analysis (PCA) – Hotelling[33]
  - Singular Value Decomposition (SVD) – Eckart/Young[36]
  - Multidimensional Scaling (MDS) – Young[38]
- Nonlinear methods:
  - IsoMap – Tenenbaum[00]
  - Locally Linear Embeddings (LLE) – Roweis[00]

# Further Reading

## CS231n Convolutional Neural Networks for Visual Recognition

These notes accompany the Stanford CS class [CS231n: Convolutional Neural Networks for Visual Recognition](#). Feel free to ping [@karpathy](#) if you spot any mistakes or issues, or submit a pull request to our [git repo](#). We encourage the use of the [hypothes.is](#) extension to annotate comments and discuss these notes inline.

### Assignments

[Assignment #1: Image Classification, kNN, SVM, Softmax](#)

[Assignment #2: Neural Networks, ConvNets I](#)

[Assignment #3: ConvNets II, Transfer Learning, Visualization](#)

### Module 0: Preparation

[Python / Numpy Tutorial](#)

[IPython Notebook Tutorial](#)

[Terminal.com Tutorial](#)

<http://cs231n.github.io>

### Module 1: Neural Networks

[Image Classification: Data-driven Approach, k-Nearest Neighbor, train/val/test splits](#)

[L1/L2 distances, hyperparameter search, cross-validation](#)

[Linear classification: Support Vector Machine, Softmax](#)