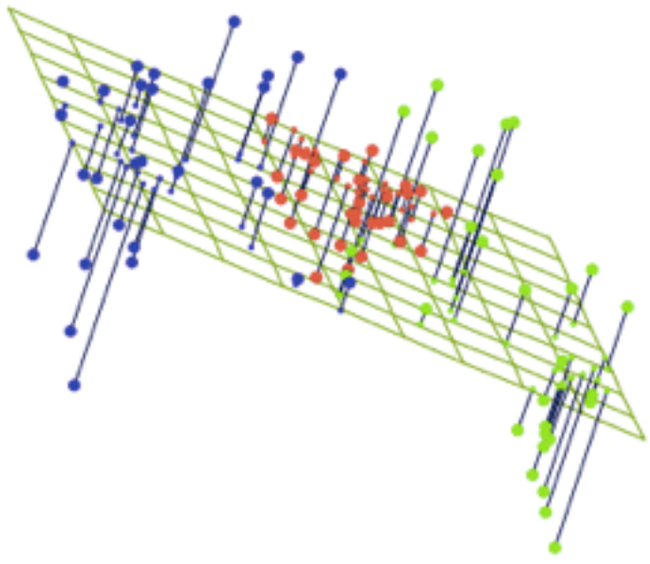
CS109 Data Science
Classification & PCA

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

Spam classification of e-mail
Given text, identify which language it is in

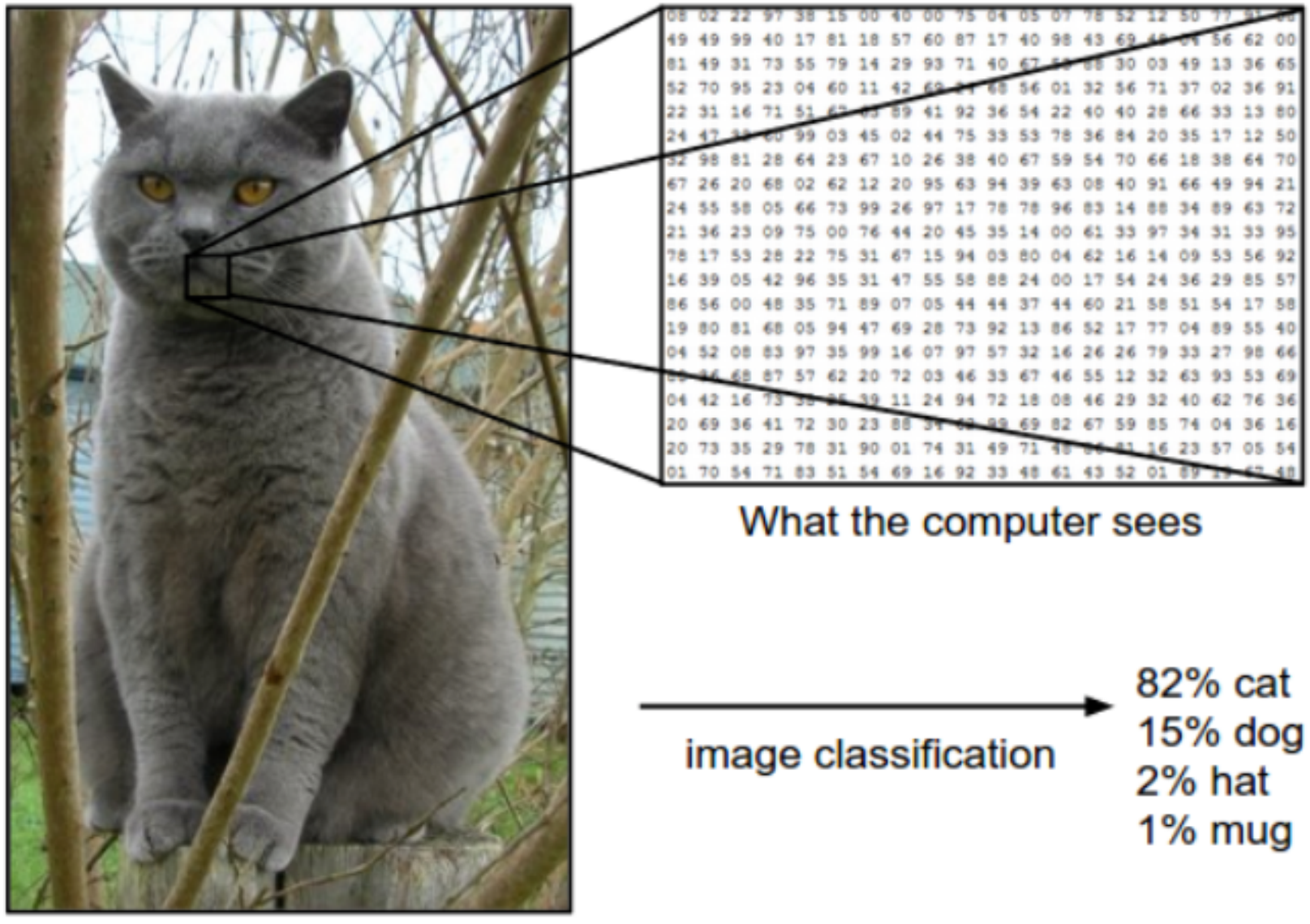
We take data, then assign labels to the data
Image classification: can we tell what is in the image.

Classification has gotten good thanks to machine learning

Classification

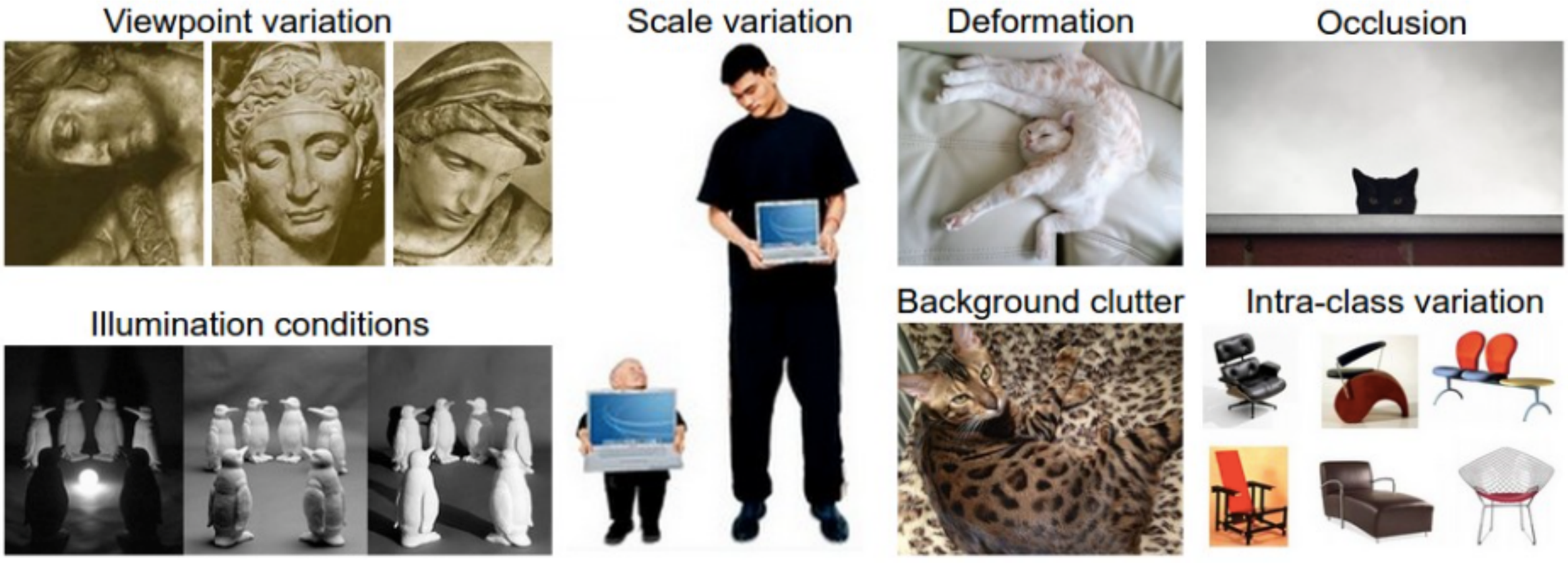
can’t hardcode: lots of possible cat pictures
angle needs to be considered
2D projection of 3D image,
needs to be invariant
to translation, rotation, etc

Classification



Alex Krizhevsky, Stanford

Problems



too many variations to account for in hard-coding
how did they make gud: they used training data categorized by people and then machine learning to learn from the training data.
works best by having more images (ie, google has 10^9 images)

Alex Krizhevsky, Stanford

Training Data

Alex Krizhevsky, Stanford

supervised learning: given dataset with labels

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.

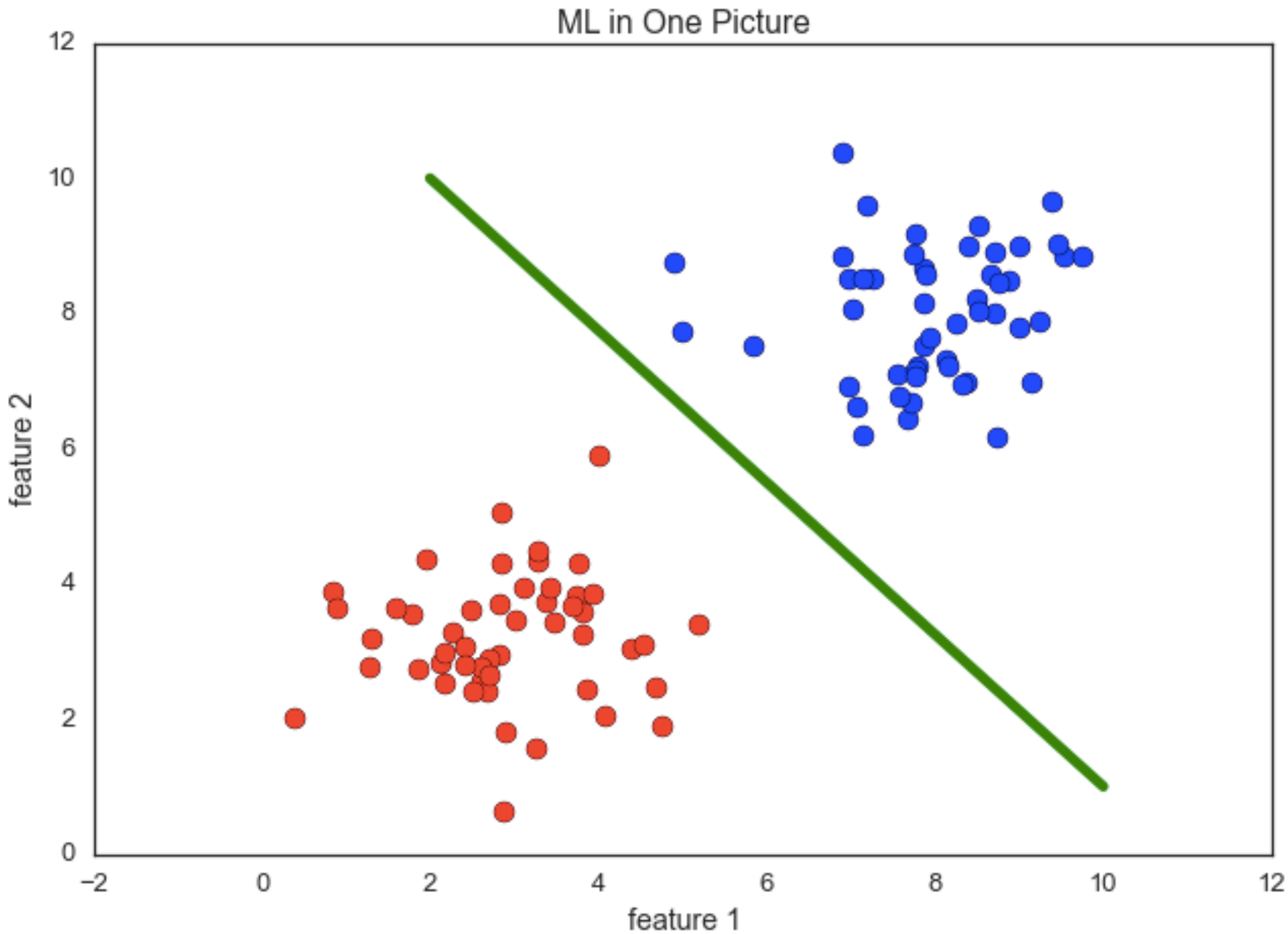
Data has labels beforehand
(labeled applied by humans)

- Find patterns in the data
  - data has no labels
  - unsupervised learning: PCA, MDS, Clustering

Can't label data: too big, don't know vategories
So, instead, lets take data and have machine find the patterns

High-dimensional space, spanned by features
have a decision boundary.

Classification



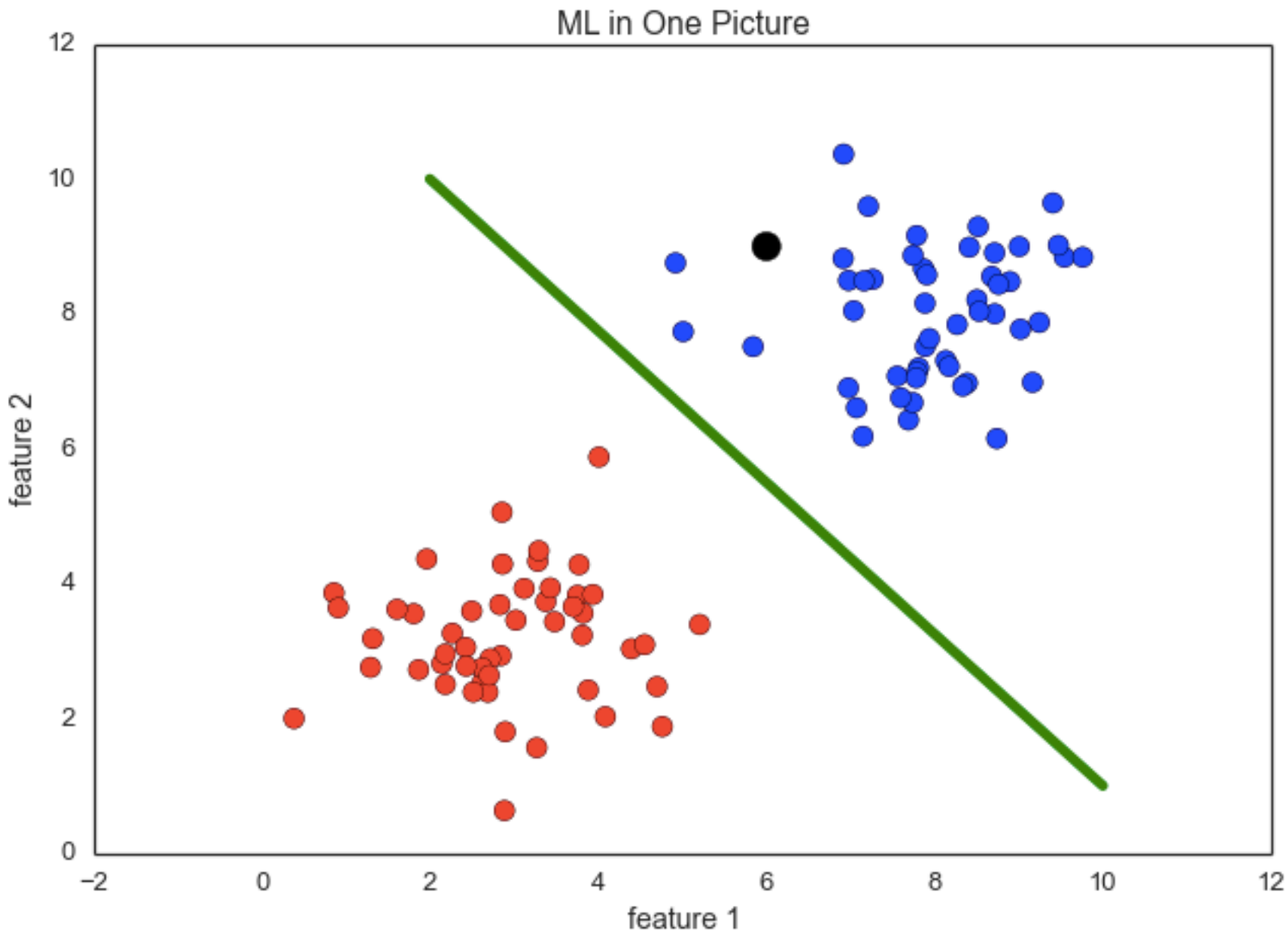
x: data points

y: labels

features

decision boundary

Classification



x: data points

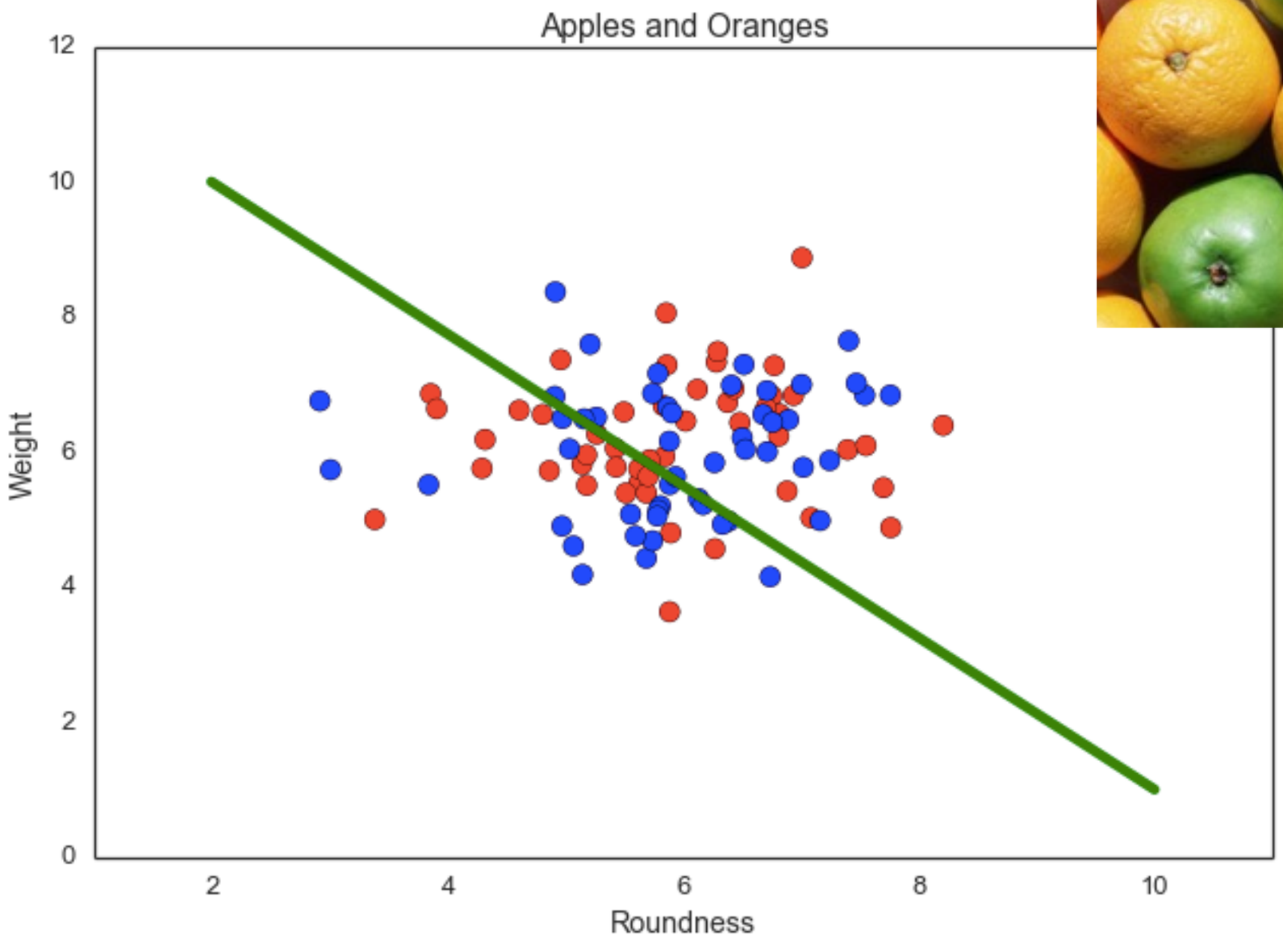
y: labels

features

decision boundary

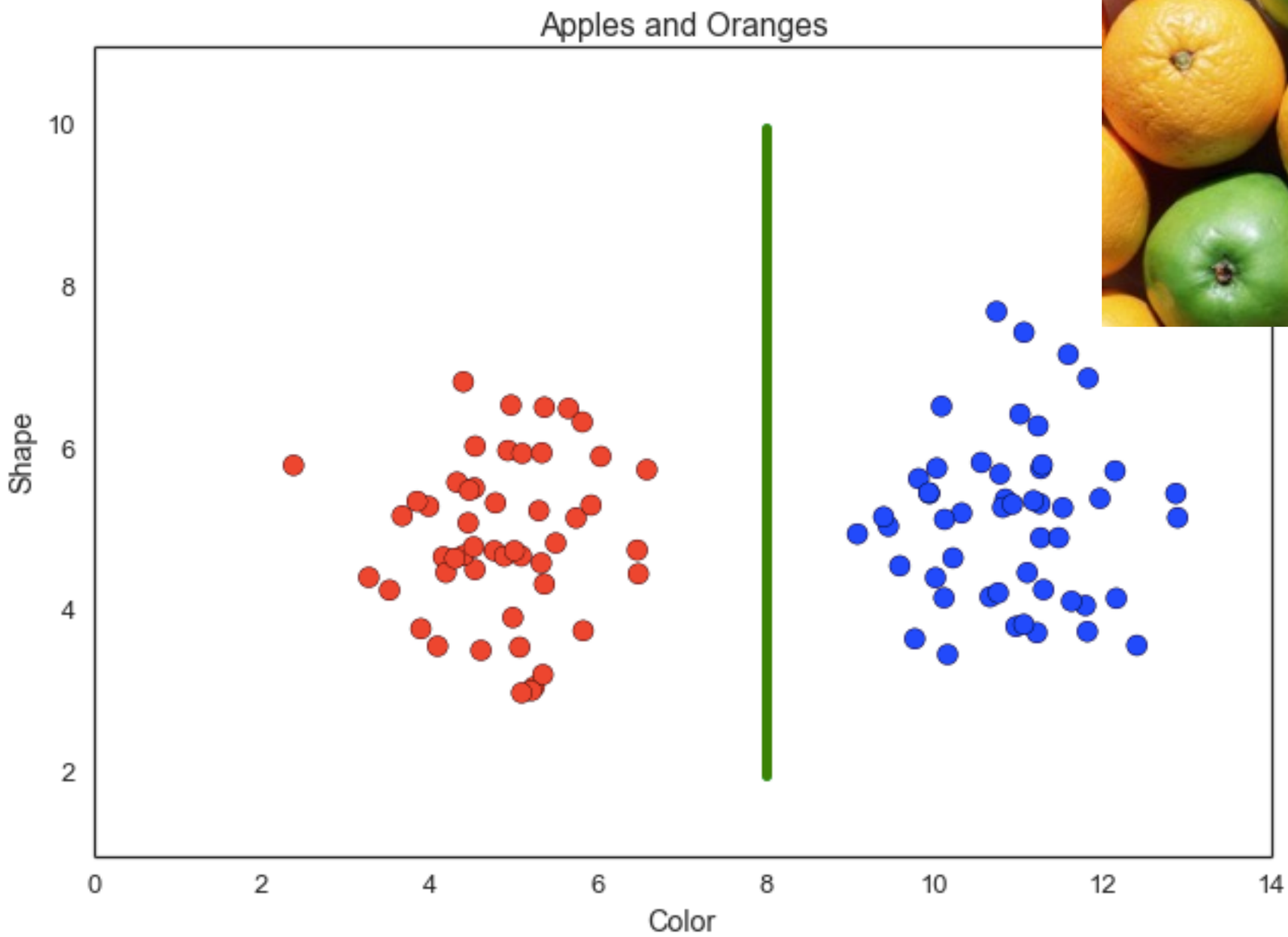
need to pick features that segregate data and allow a decision boundary.

Feature Selection



Can select features with cross validation

Feature Selection



Apples not orange.

Nearest Neighbor Classifier

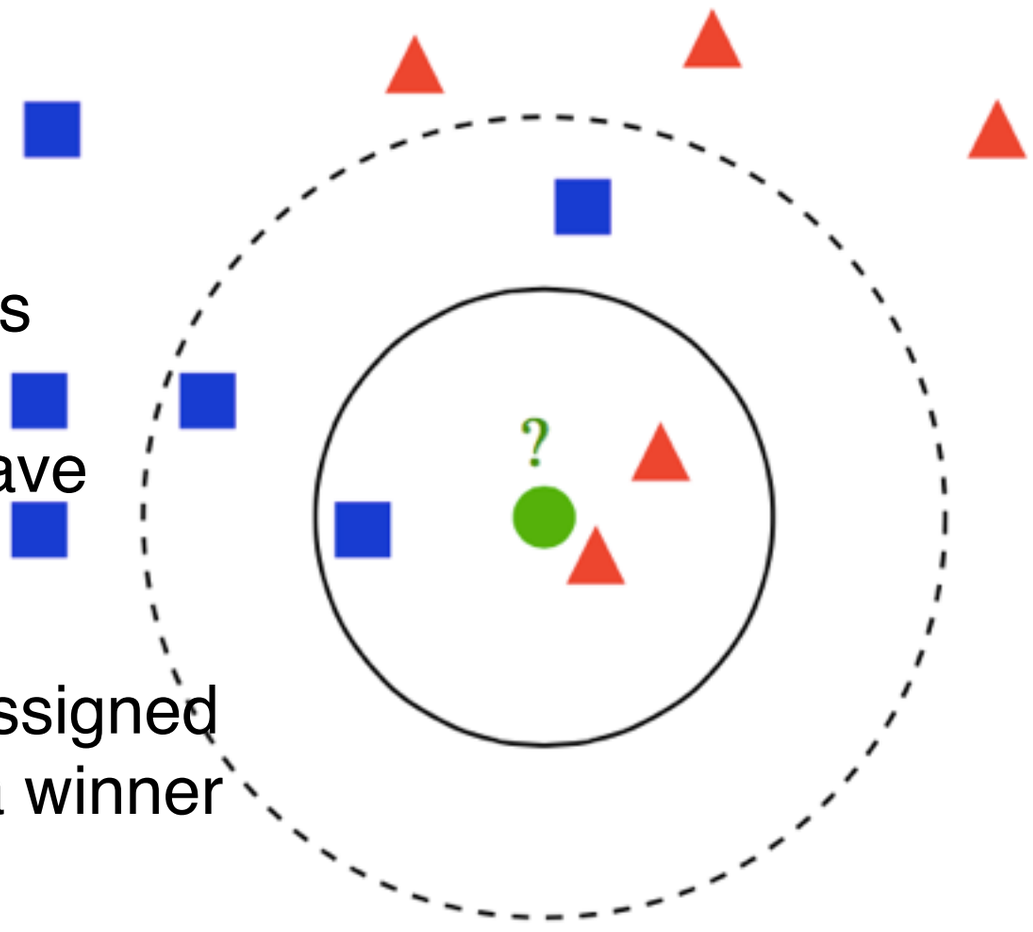
Nearest Neighbor

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

but, how do you decide how big neighborhood is

ie, could expand and have 3 squares vs 2 triangle

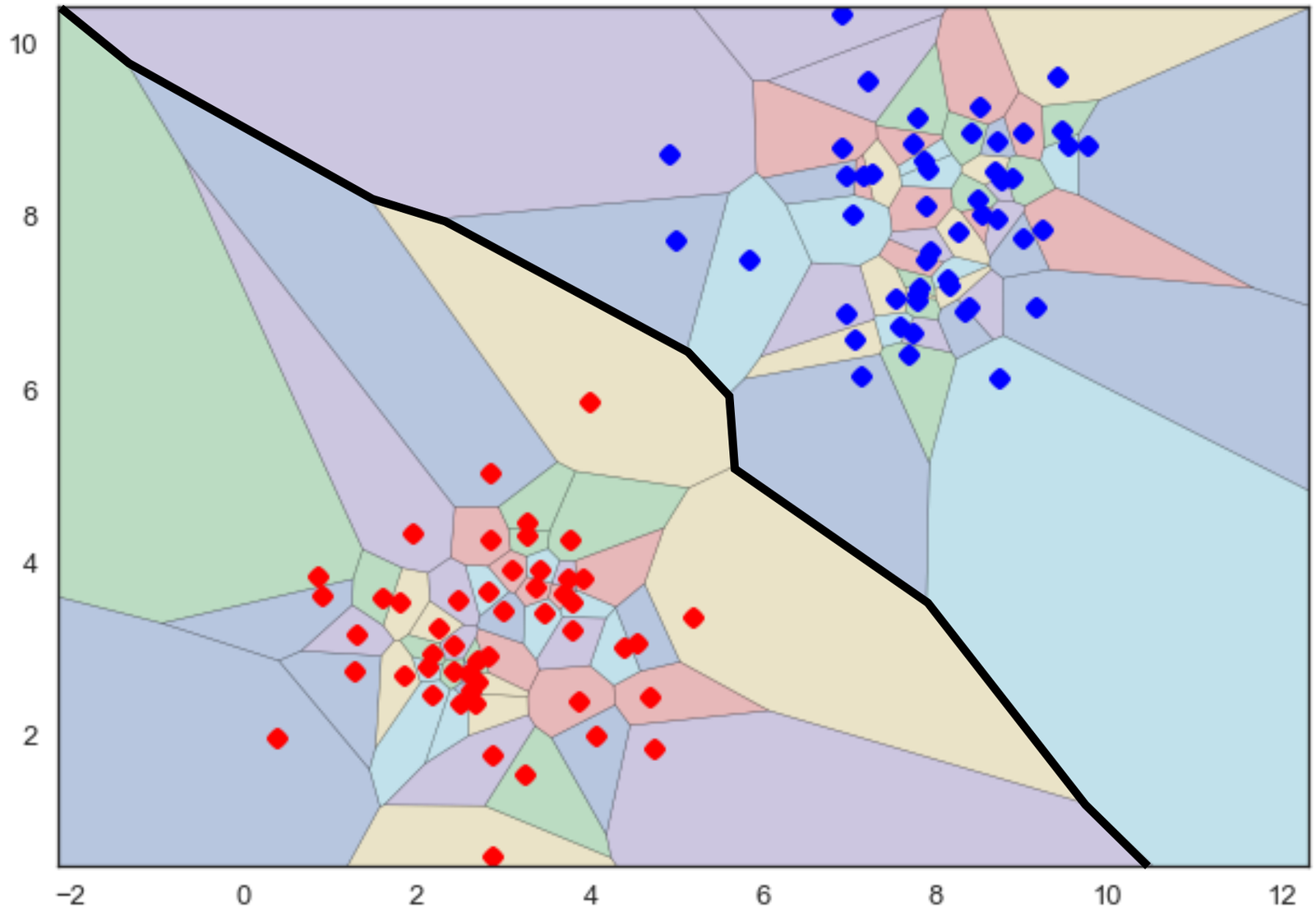
simplest way: K is an assigned odd number so it gets a winner



2 triangles, 1 square in first circle
—> green circle is a triangle

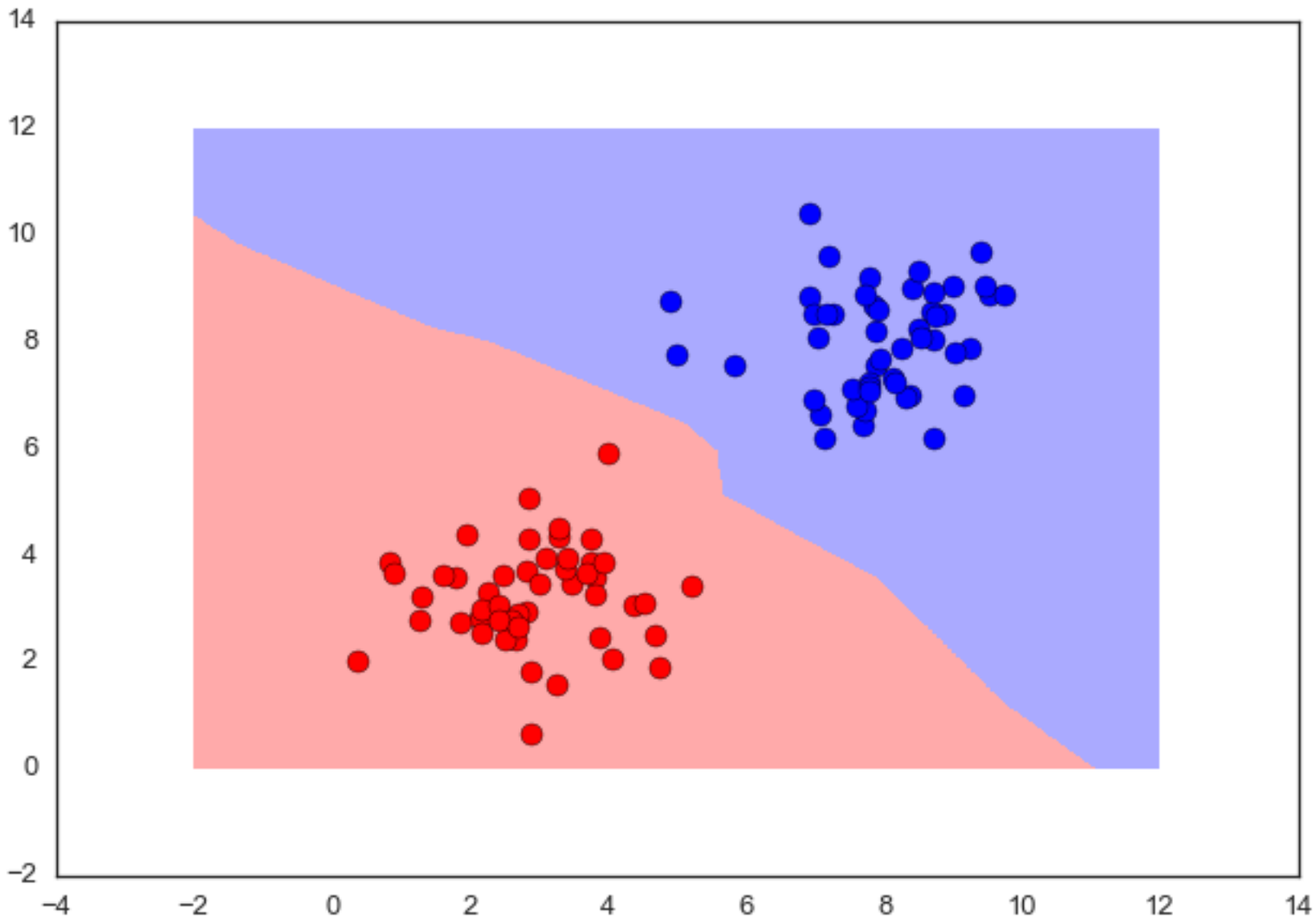
1-Nearest Neighbor

This is a representation of how near neighbors are



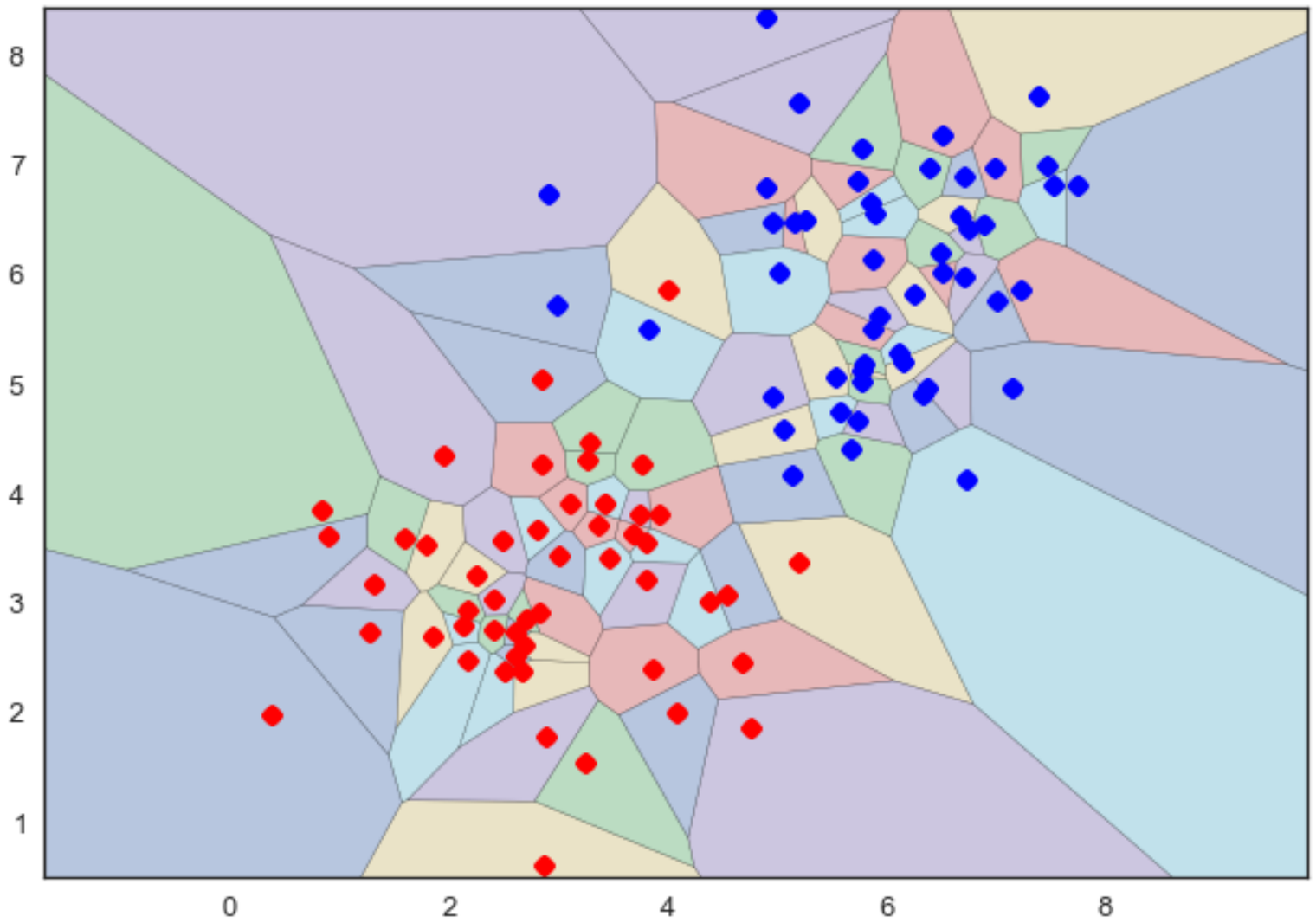
Bold is the decision boundary.
rough boundary: typical of nearest neighbor.

1-Nearest Neighbor



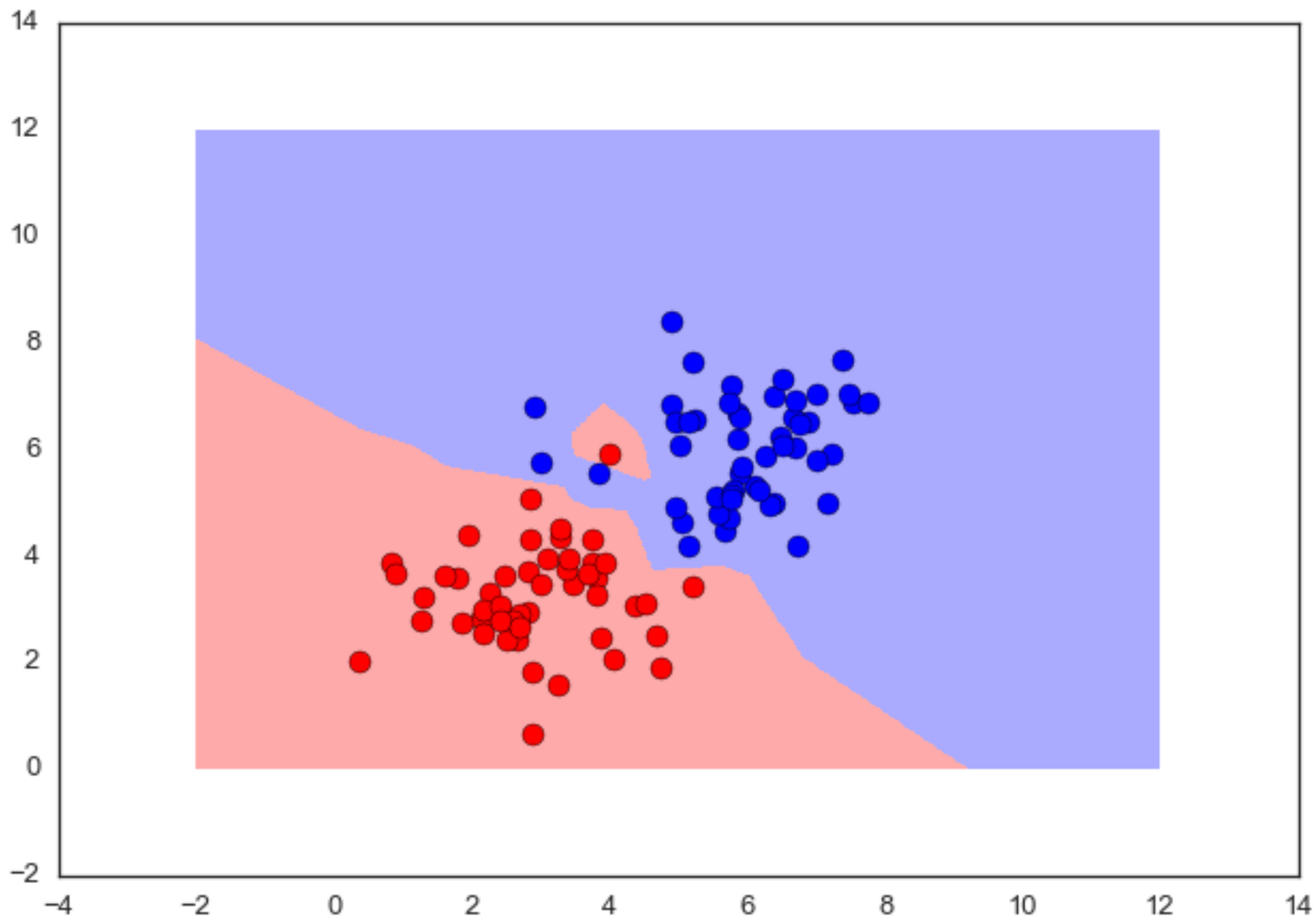
new dataset: new decision boundary

1-Nearest Neighbor



The island?
It doesn't ignore the island as an outlier.

1-Nearest Neighbor



Is this a good decision? -> no, a human would consider this an outlier
The nearest neighbor classifier isn't smart enough.

simple to code: store all data, find nearest neighbor —> runs surprisingly well.

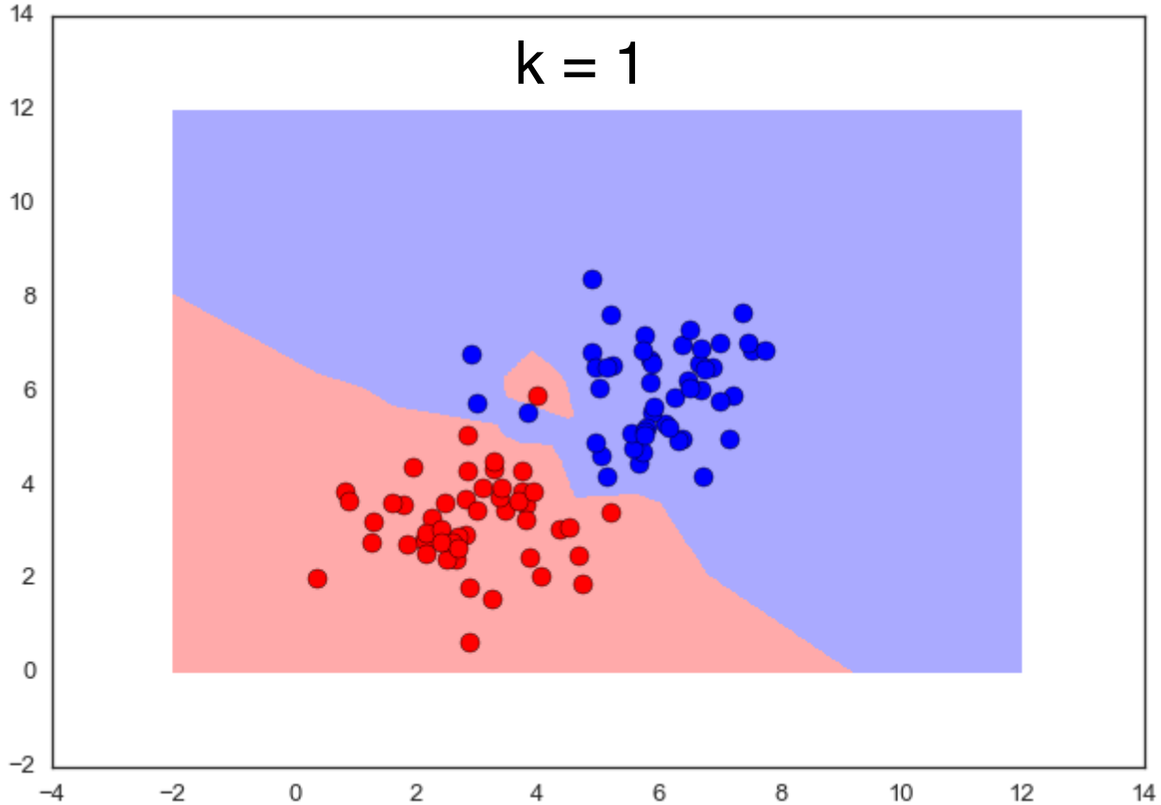
1-NN Properties

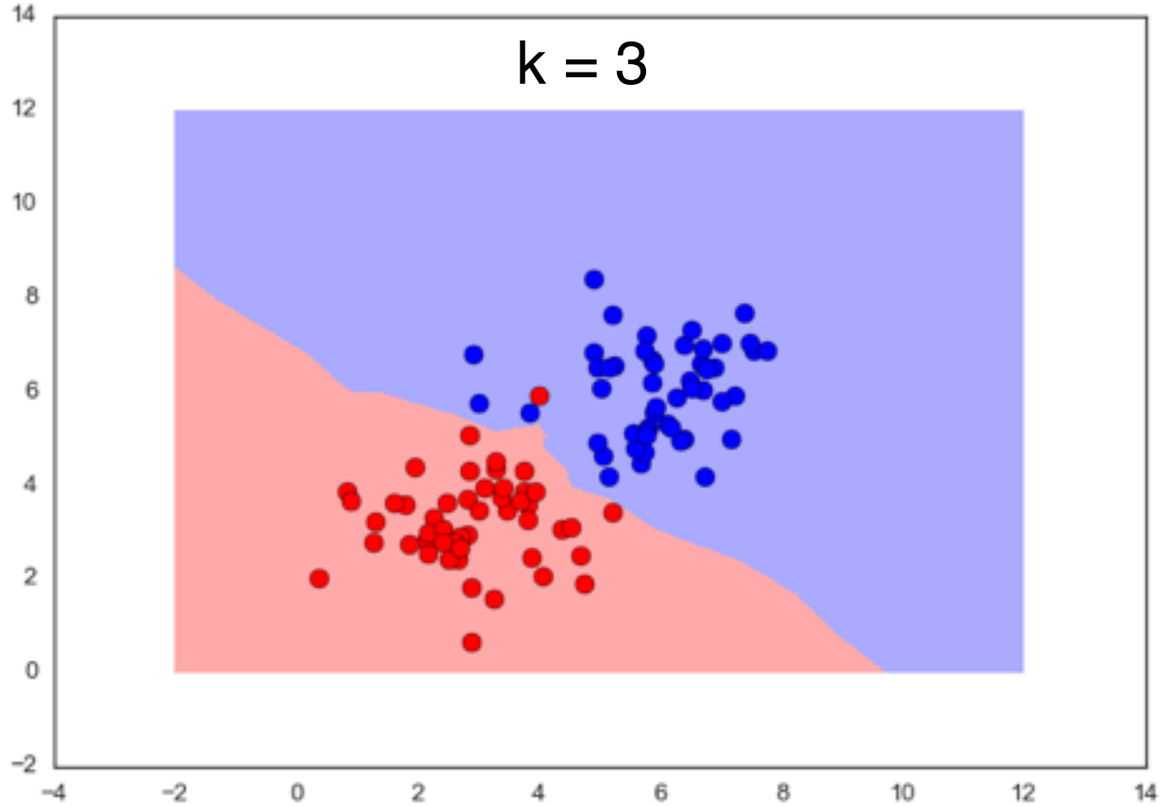
Better to make training time long and testing time short
ie, google image result returns based on testing time

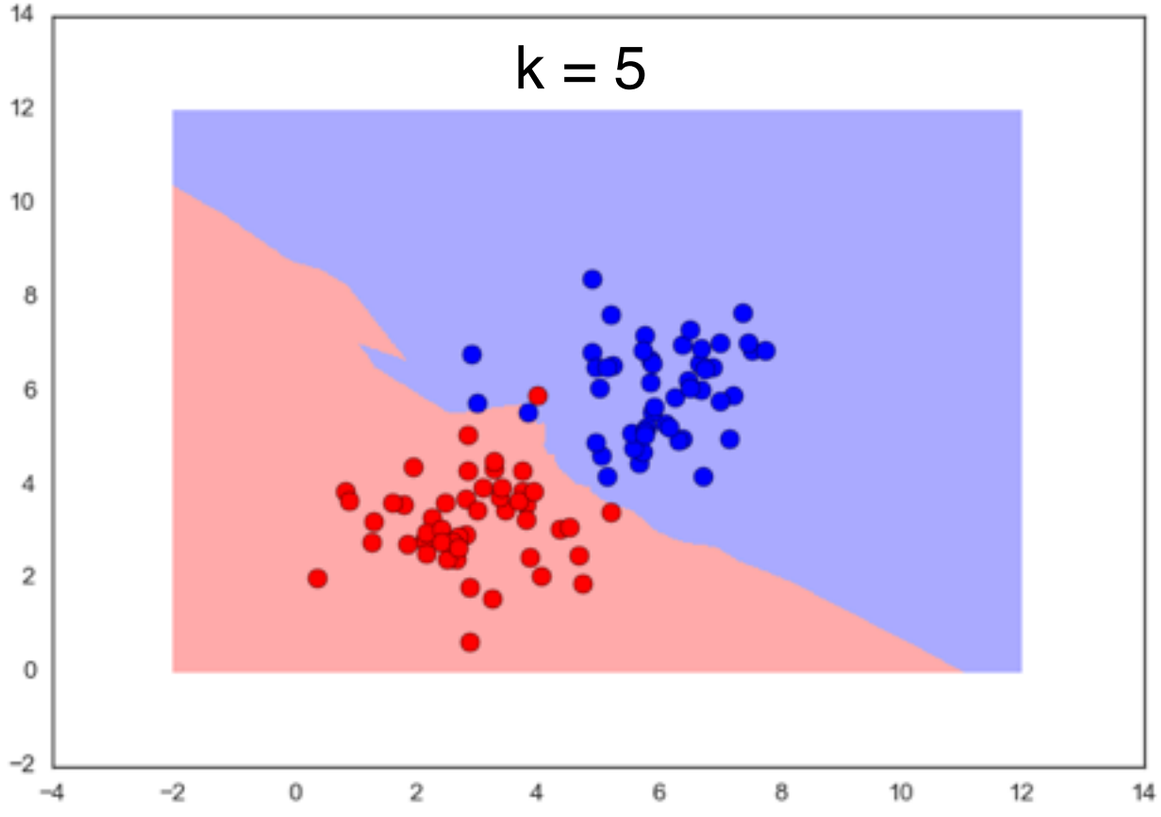
- Simple and quite good for low dimensional data
- "Rough" decision boundary, may have "islands"
- Training complexity for N data points? O(1), just appending
- Test complexity for M data points? O(n*m)
  n training samples
  m training samples
  must run through each pair
  —> All work done in testing, not training
- Error on training set? Nope
- Variance? Bias?

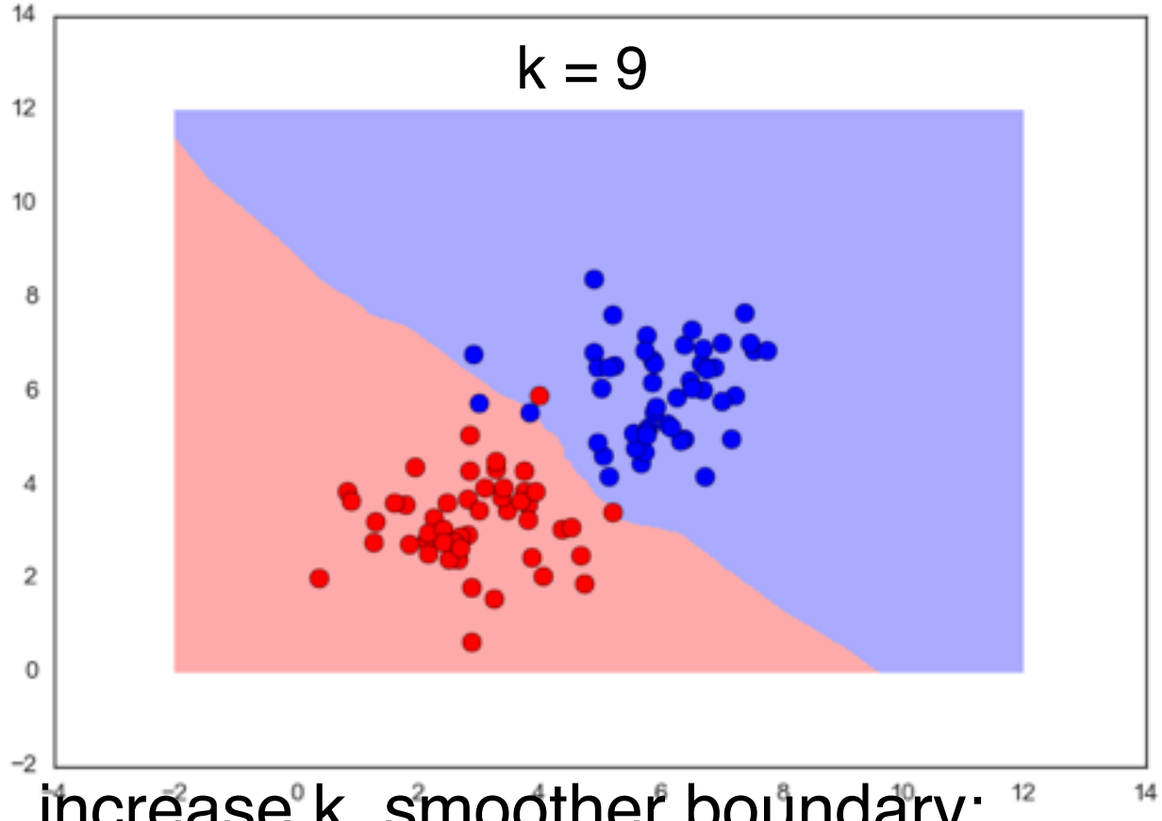
Variance: change in decision boundary by adding new data
this case: boundary changes w/ each new set of data
—> high variance
Bias: Quite low -> on avg each one performs well.
How can we reduce roughness/variance -> majority voting instead of nearest neighbo

comparing 3 neighbors









increase k, smoother boundary:
variance decreases, increases bias.

k-NN Properties

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

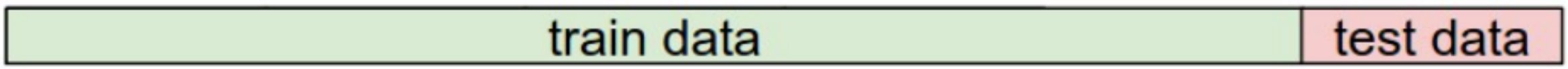
  Use cross validation.
  —> split data into training and test set: build model with training set and test the model on the test set
  —> proportion to split?

How do we measure distance? not necessarily Euclidean.
How do you decide majority voting (than probabilistically in these examples
—> hyperparameters to be determined by corss validation

Usually we don't have a lot of training data: Can't label so much.
Don't use entire training set: then you optimize once:
—> split training data into smaller sets

Validation

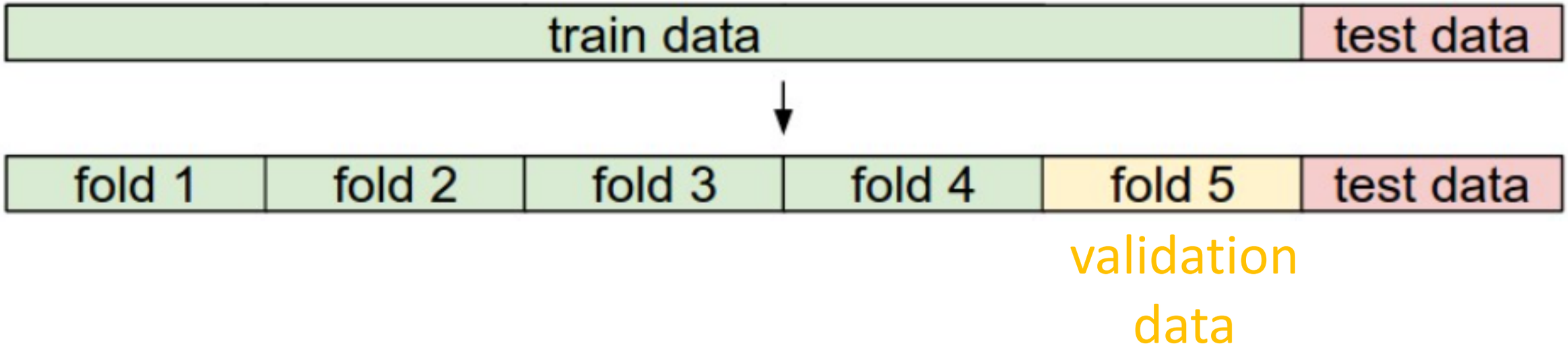
- Train on training data, test on test data
- Pick the k with the lowest test error



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

Split into fold: use one fold for hyper parameters -> validation data.
Pick optimal parameter -> then test on the test data (test only once at the end).

Cross-Validation



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

Pick test data randomly: hopefully unbiased

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 (hyper) parameters!

Test data valuable: use them for the end.
—> results not generalizable: contaminated the estimation, test data must be new

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

automobile

bird

cat

deer

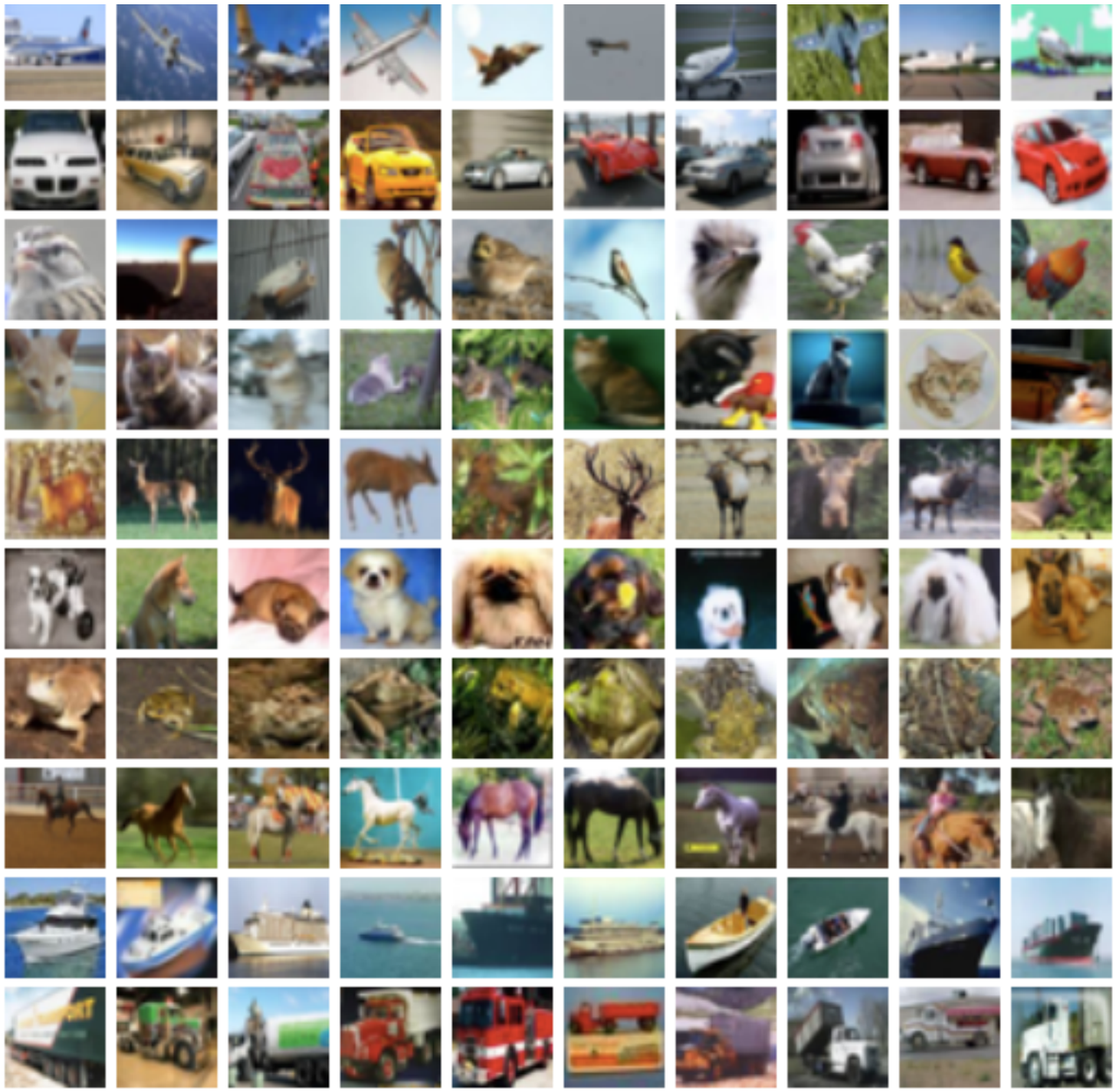
dog

frog

horse

ship

truck



60,000 images
32x32 pixels

10 classes

Training set:
50,000 images

Test set:
10,000 images

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton

High-Dimensional Data

Each image is a vector in 32 x 32 x 3 = 3,072 dimensional space

3 color values: RGB

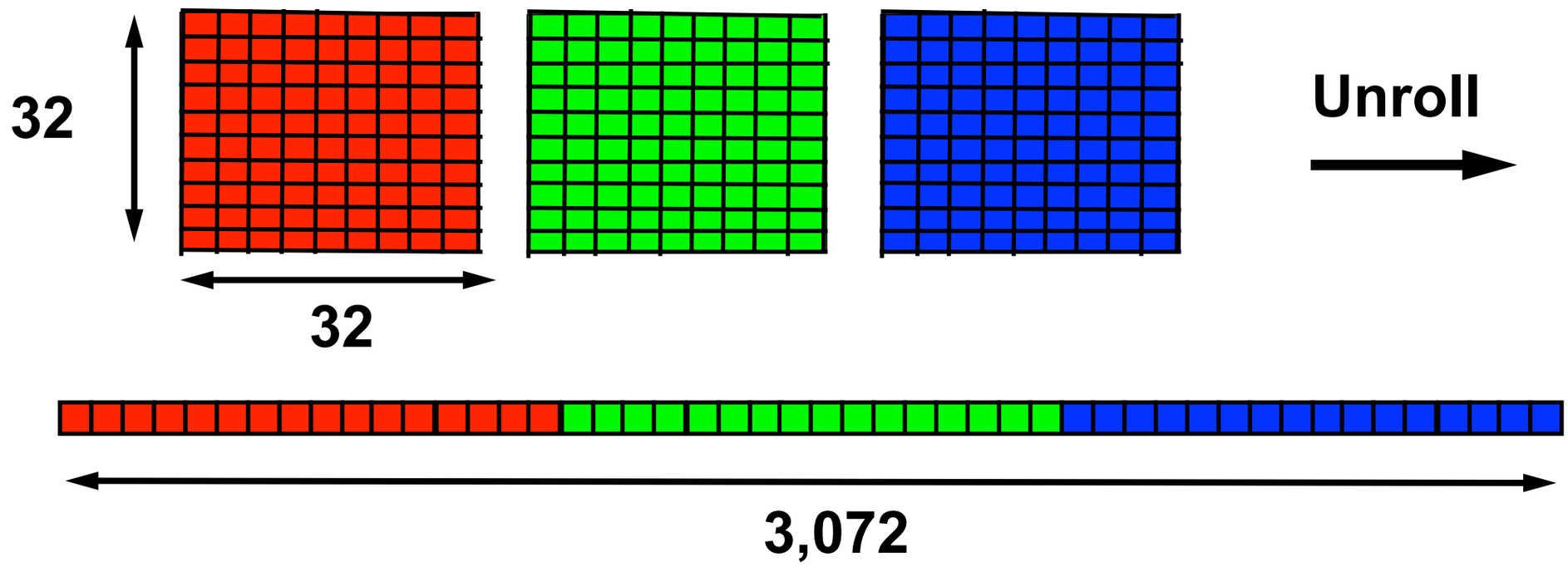


Image Comparison

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

→ 456

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

p: pixel

I_1: image 1

Manhattan distance:absolute distance between vectors

Distance Metrics

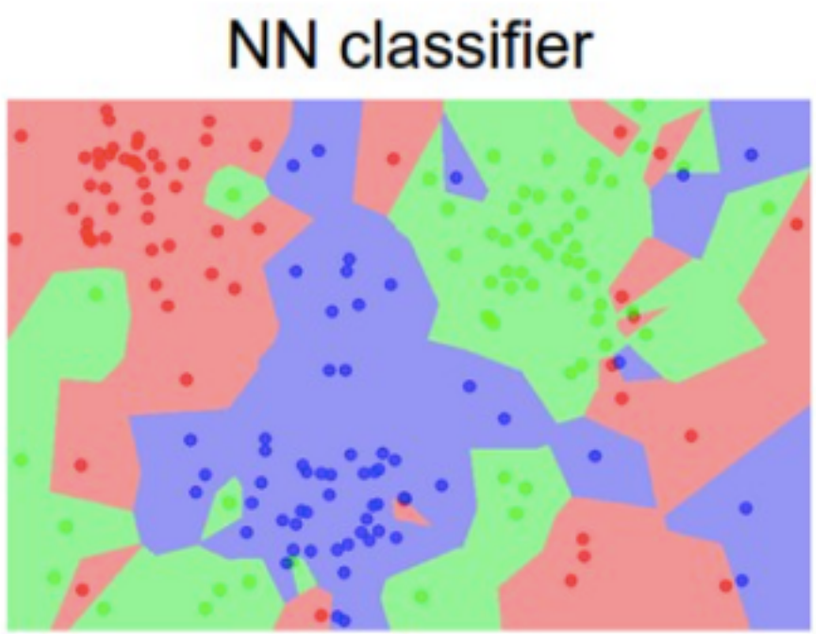
L1 (Manhattan) distance: $$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

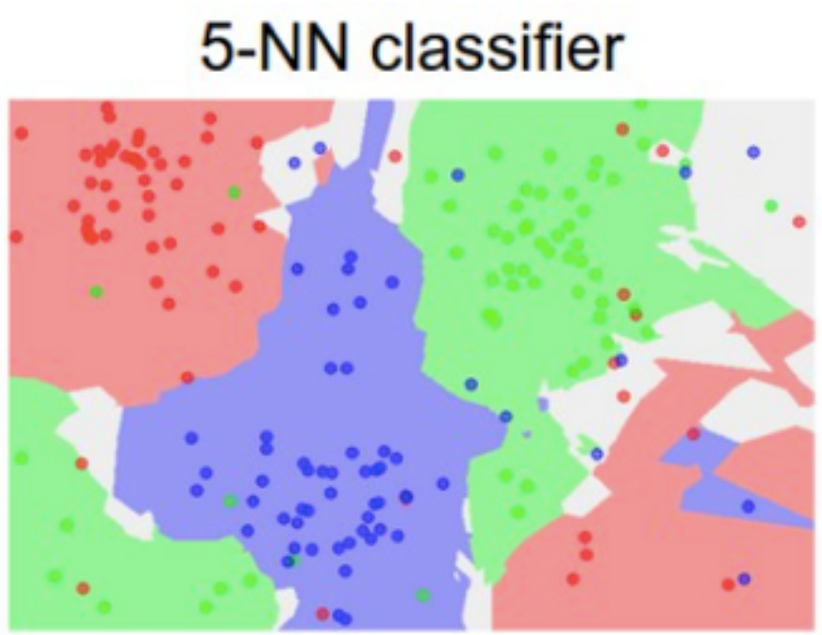
L2 (Euclidean) distance: $$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$

More general Lp norms:

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

k-NN Classifiers



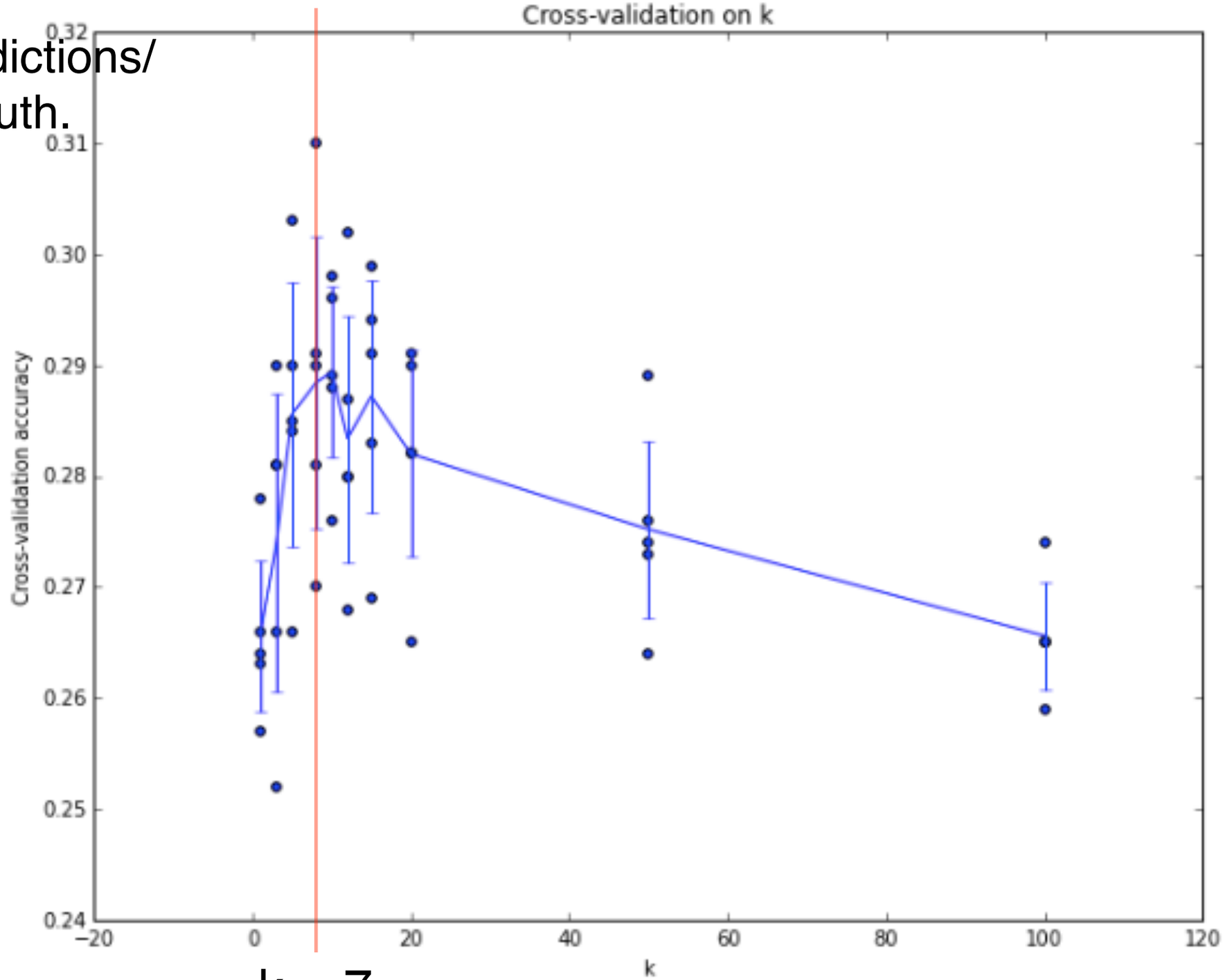


better than NN

https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

5-fold Cross Validation

of right predictions/
# of right in truth.



k = 7

Alex Krizhevsky, Stanford

10 Nearest Neighbors

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

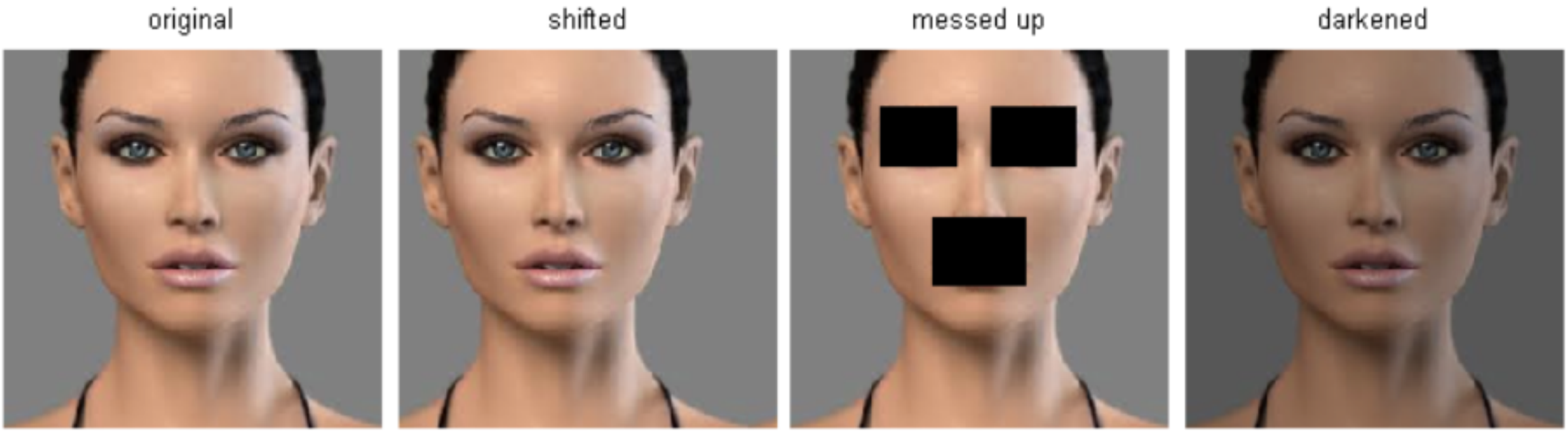
Looks mostly @ background most pixels in background —> bad metric

must pick diff feature than pixels and pixel distance

Alex Krizhevsky, Stanford

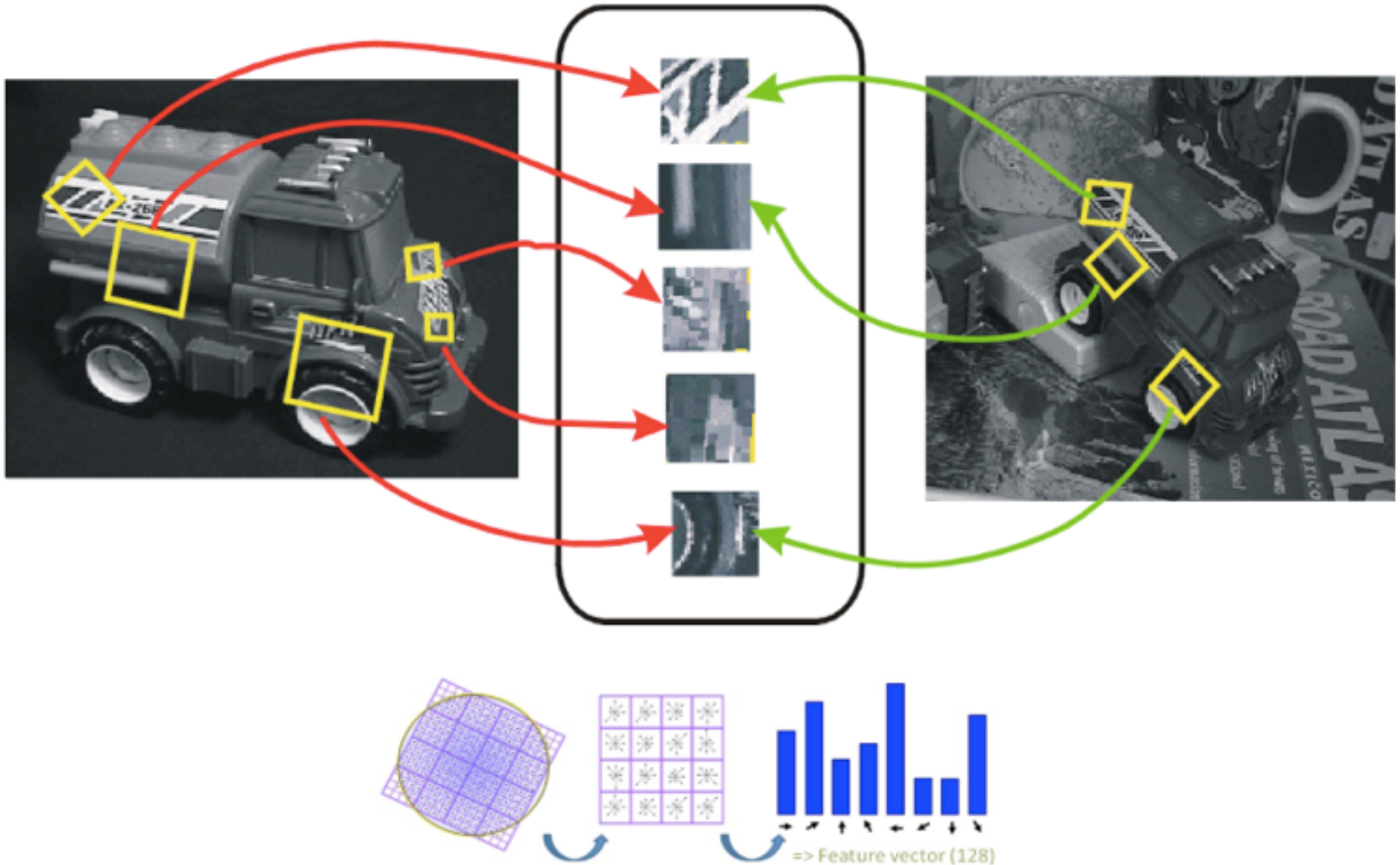
Pixel-based L2 Distances

These images have the same L2 pixel distance



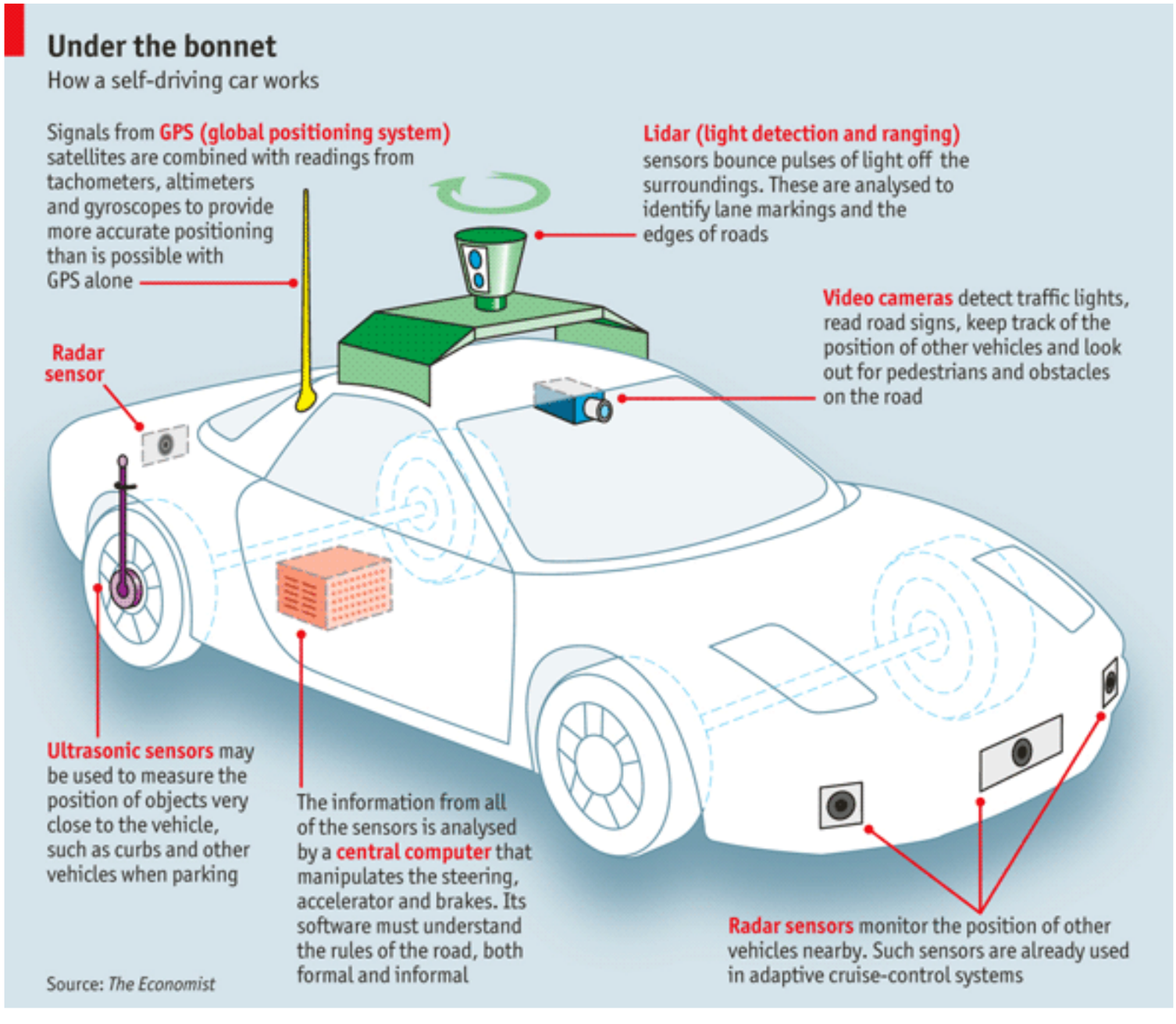
sift features - rotation invariant.
picking feature most important.

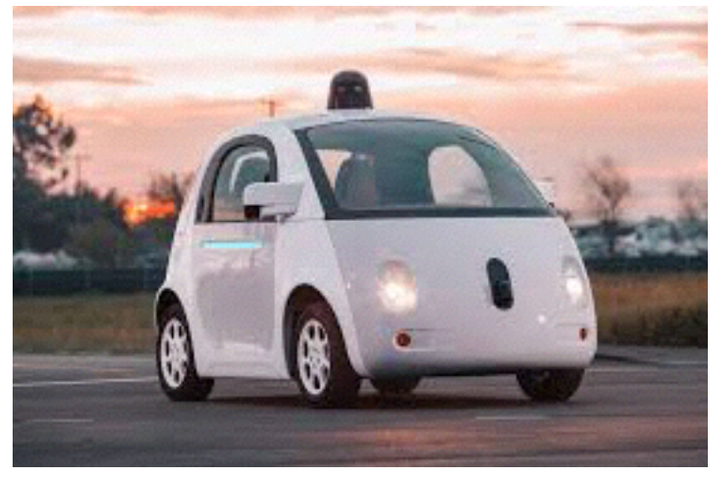
Image Features



"SIFT" & Object Recognition, David Lowe, 1999

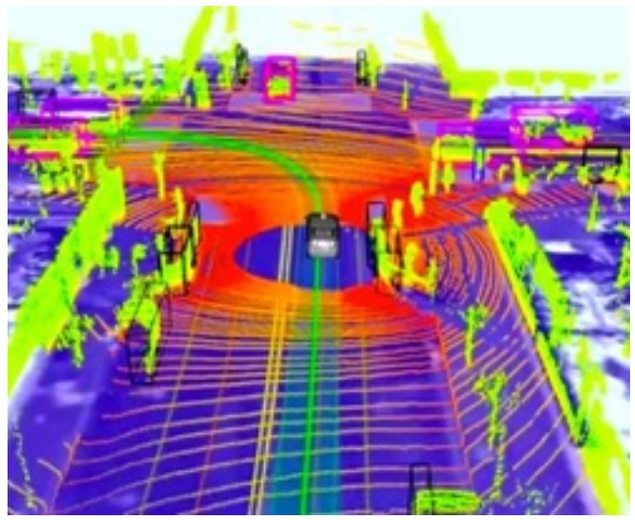
Self-Driving Cars





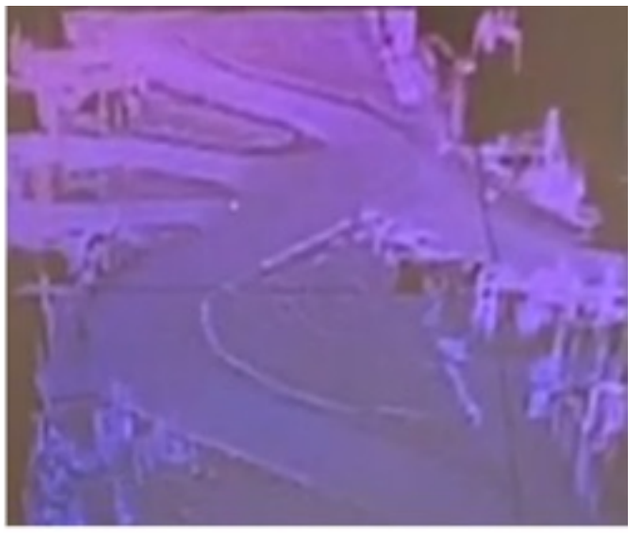
Car Features

Laser scan



Intensity model

Elevation model



Camera vision

2D stationary map

Lane model

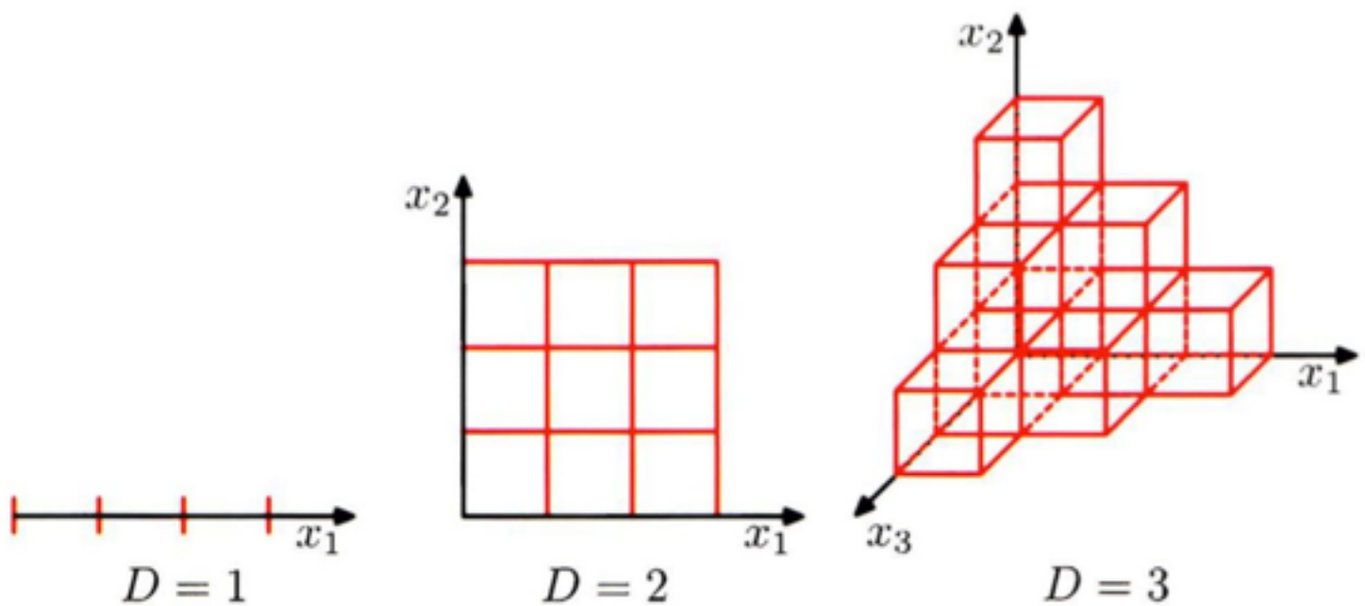
Takes more time
Data gets to sparse - curse of dimensionality

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

trade-off to classification being easier with too many features
—> let's break down into lower dimensions

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

want to preserve distance across high dimensional to low dimensional.

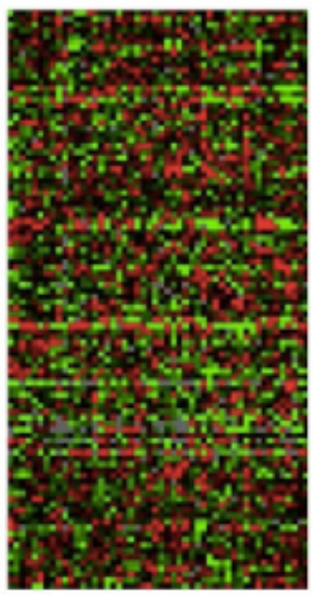
High-dimensional Data

face images

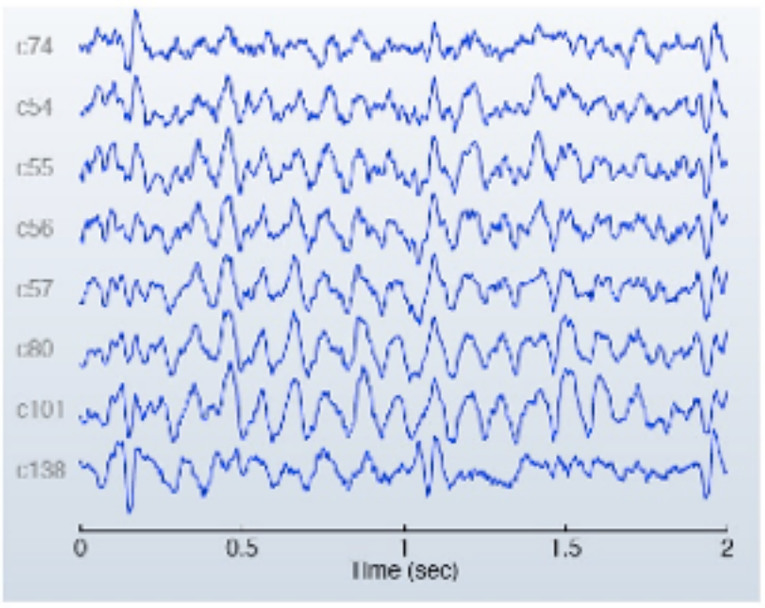
Zambian President Levy Mwanawasa has won a second term in office in an election his challenger Michael Sata accused him of rigging, official results showed on Monday.

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



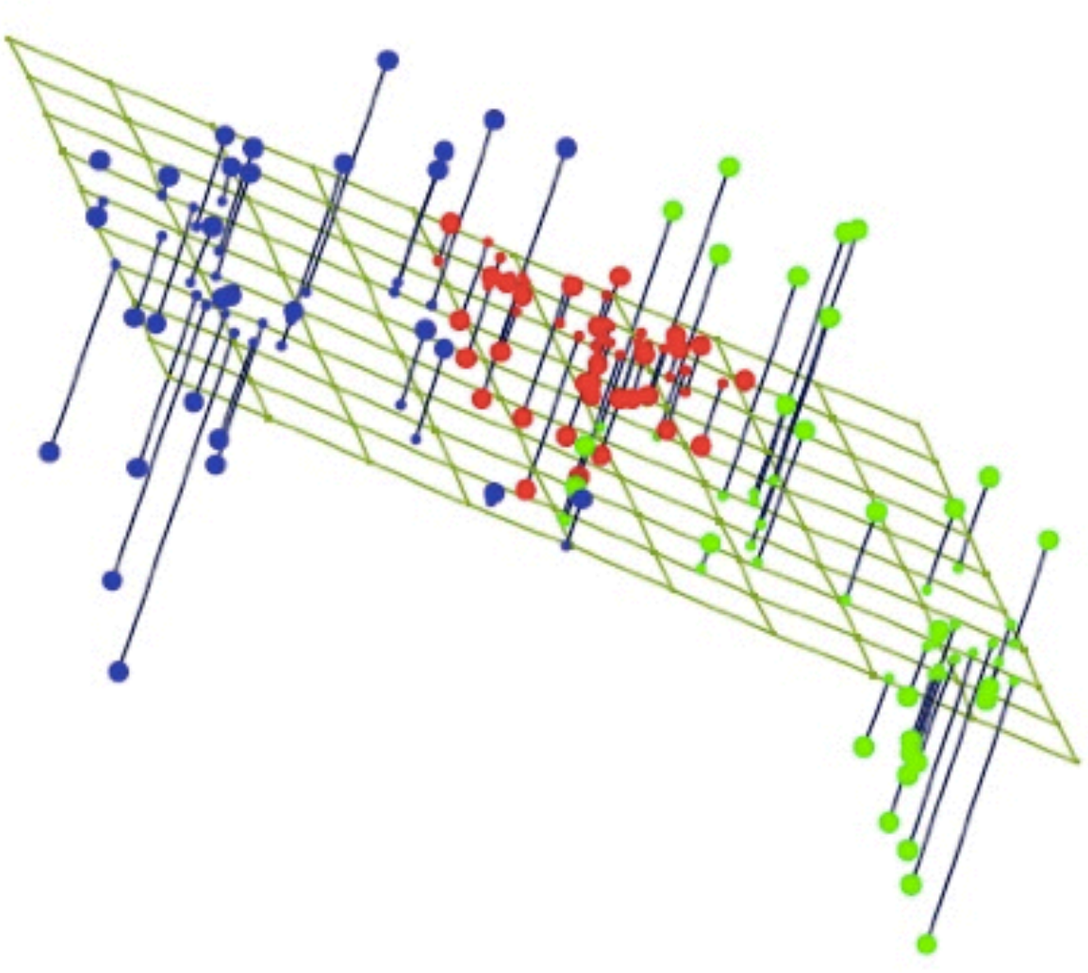
MEG readings

Based on slide from P. Liang

Lot of reduction techniques are linear (hyper planes)

Basic Idea

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



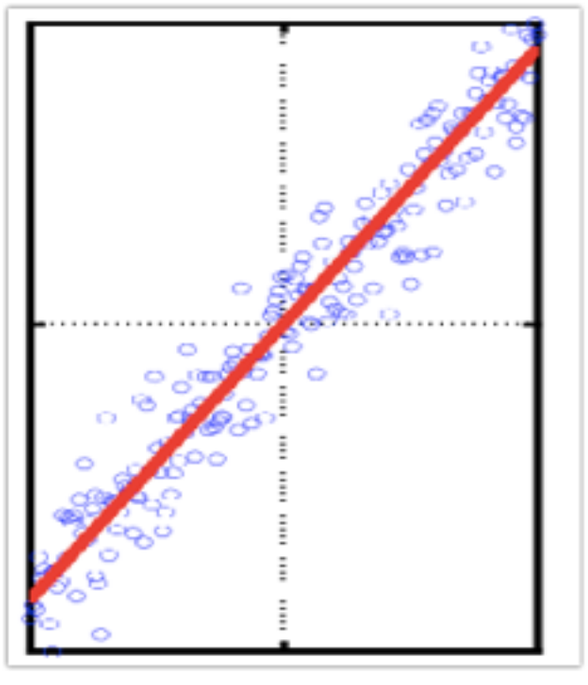
Based on slide from P. Liang

Linear Methods

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

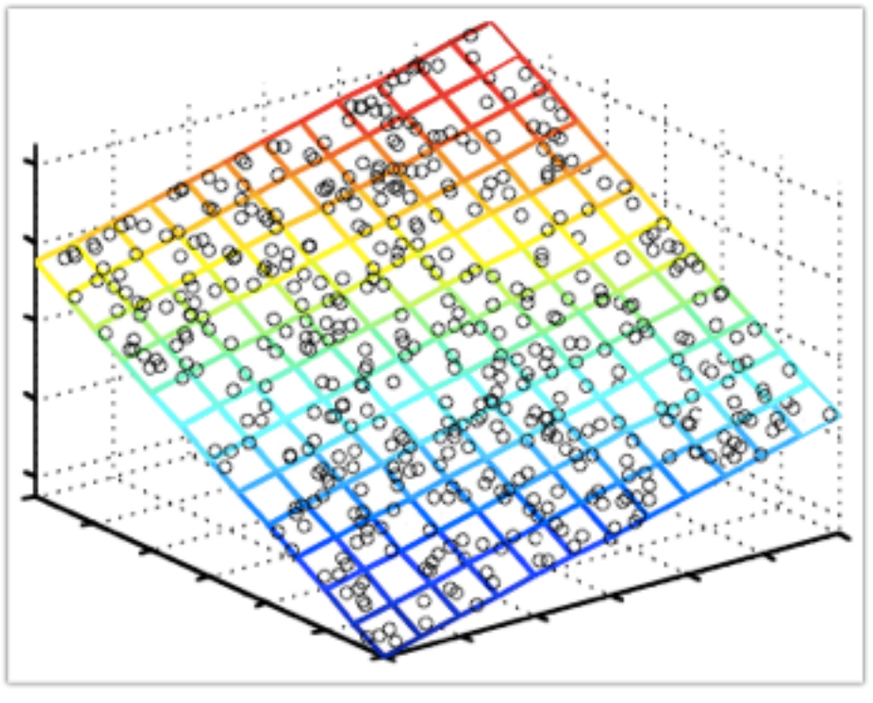
D = 2

d = 1



D = 3

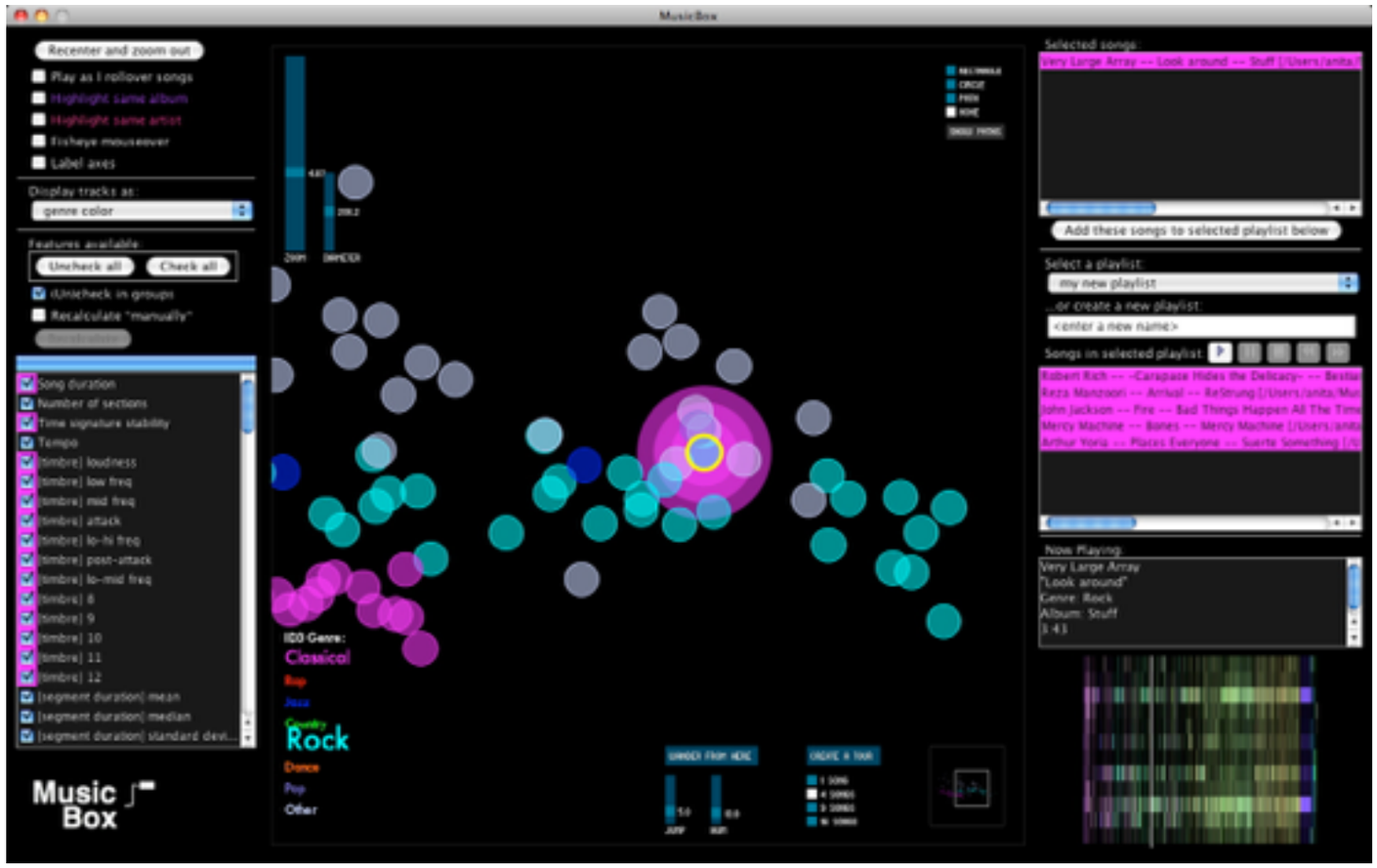
d = 2



Based on slide from F. Sha

Uses

- Dimensionality reduction for supervised learning
- Compression
- Visualization



MusicBox, Anita Lillie, MIT

MusicBox

Recenter and zoom out

- [ ] Play as I rollover songs
- [ ] Highlight same album
- [ ] Highlight same artist
- [ ] Fisheye mouseover
- [ ] Label axes

Display tracks as:

genre color

Features available:

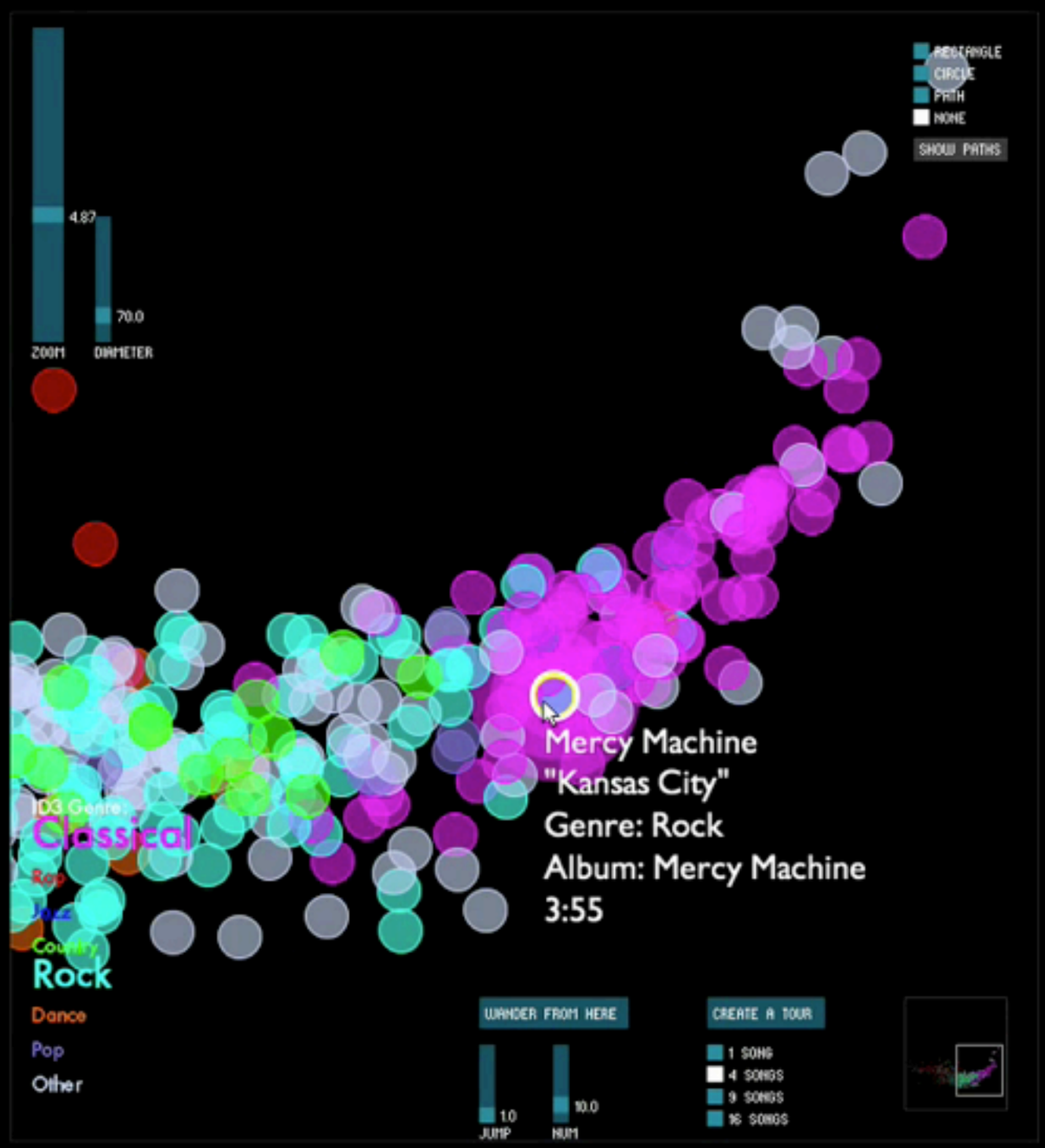
Uncheck all | Check all

- [x] (Un)check in groups
- [ ] Recalculate "manually"

Recalculate

- [x] Song duration
- [x] Number of sections
- [x] Time signature stability
- [x] Tempo
- [x] [timbre] loudness
- [x] [timbre] low freq
- [x] [timbre] mid freq
- [x] [timbre] attack
- [x] [timbre] lo-hi freq
- [x] [timbre] post-attack
- [x] [timbre] lo-mid freq
- [x] [timbre] 8
- [x] [timbre] 9
- [x] [timbre] 10

Music Box



Selected songs:

Mercy Machine -- Kansas City -- Mercy Machine [/Users

Add these songs to selected playlist below

Select a playlist:

empty

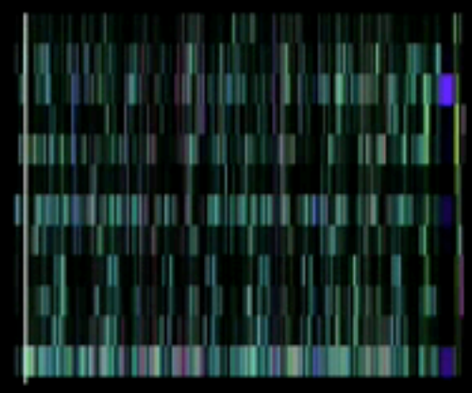
...or create a new playlist:

<enter a new name>

Songs in selected playlist:

Now Playing:

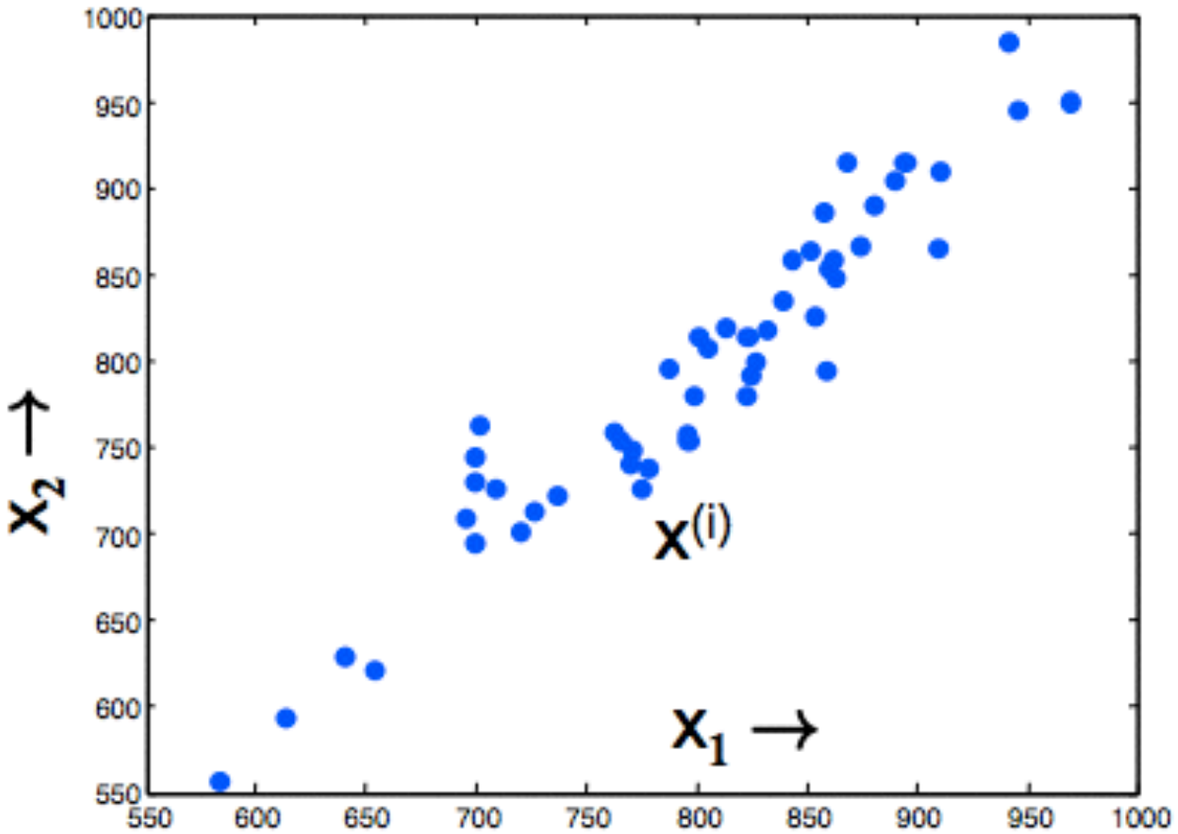
Mercy Machine
"Kansas City"
Genre: Rock
Album: Mercy Machine
3:55

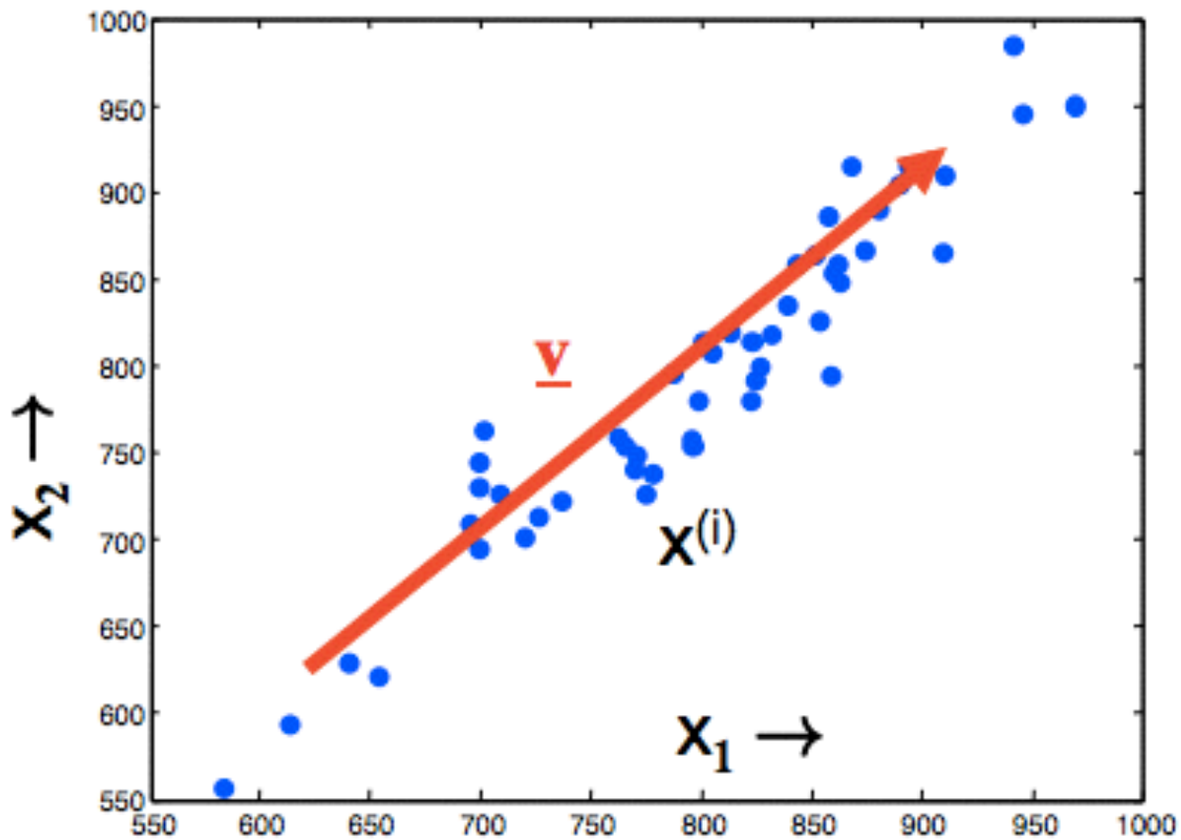


projecting into lower dimensions.

Principal Components Analysis (PCA)

Example



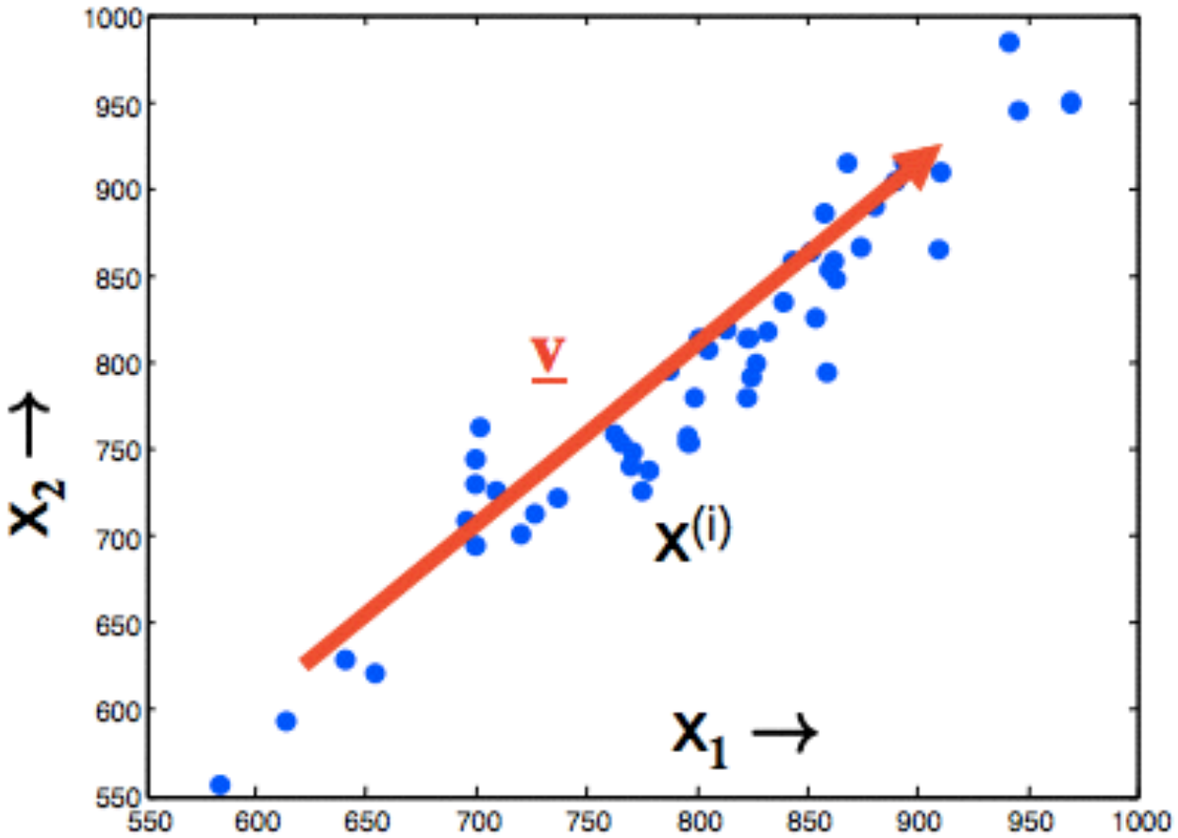


intrinsically 1d -> project to principal component
projection also like a rotation in this case

v takes direction of maximizing variance (same as minimizing orthogonal projection)

Based on slide from A. Ihler

Example



v: chosen to minimize orthogonal distances

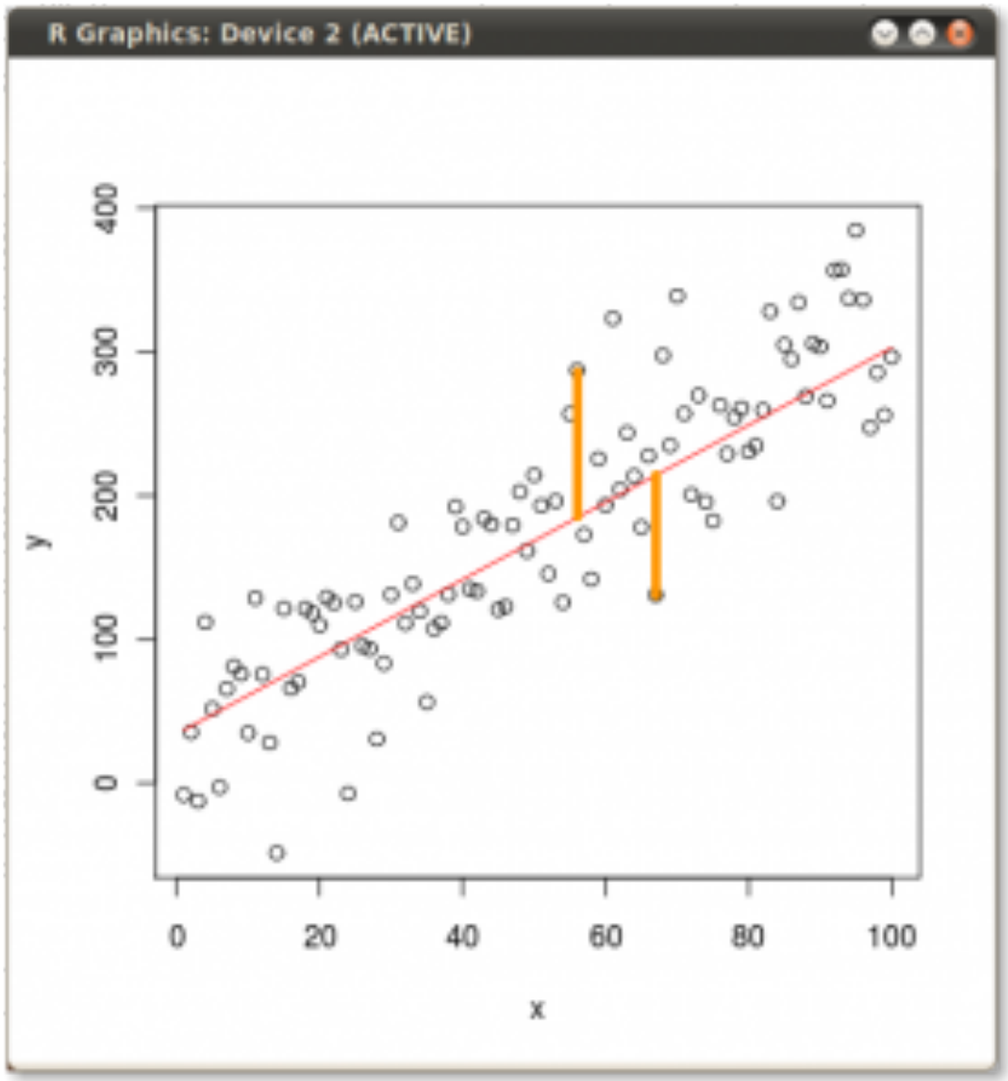
Equivalent: v is the direction of maximum variance (max. the spread along v)

Based on slide from A. Ihler

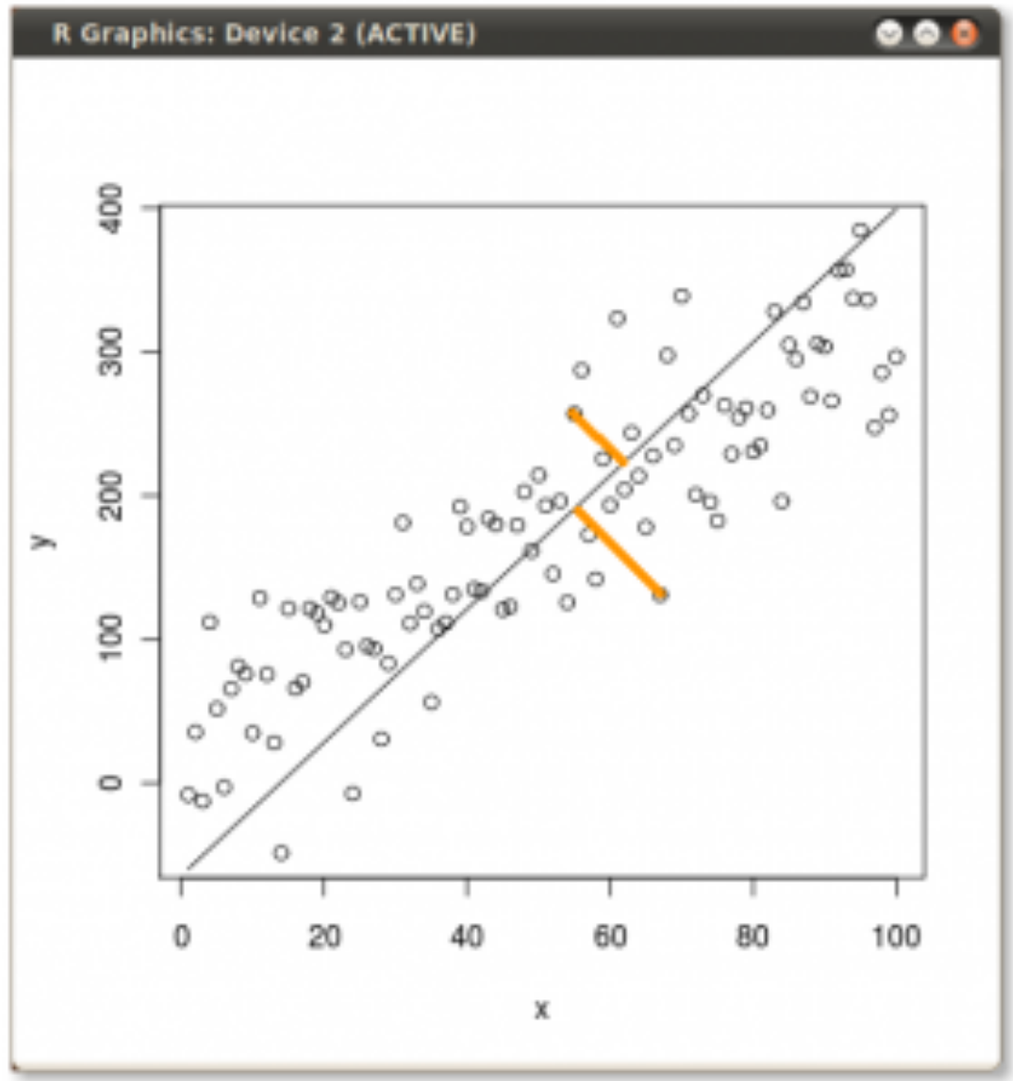
Linear Regression vs. PCA

same idea of projection

Linear Regression



PCA



http://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

PCA Algorithm

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

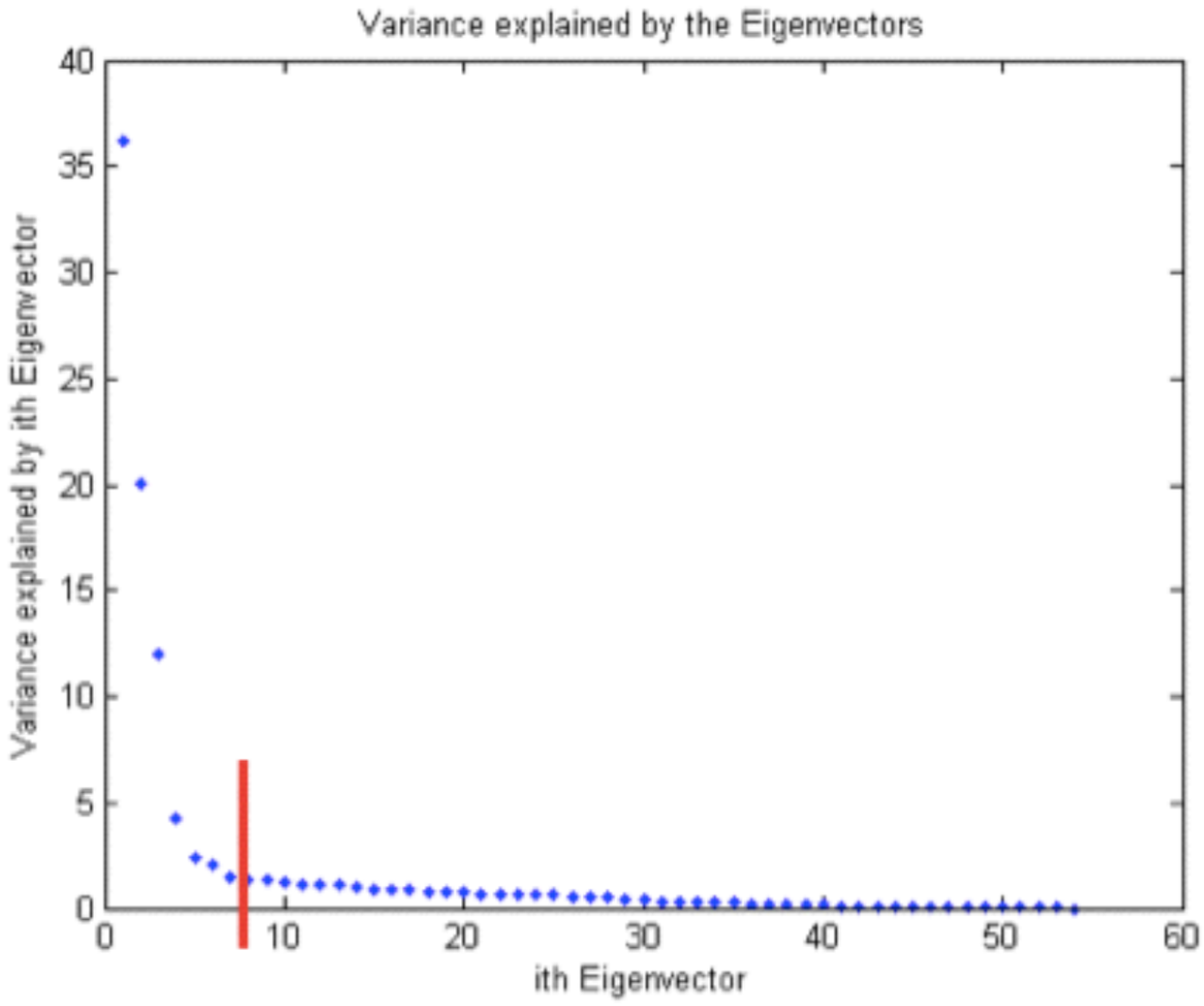
Scale: often can derive by variance to bring into same range.
eigenvectors are principal components: often use singular value decomposition

Based on slide from A. Ihler

simple formula that shows how much variance eigenvector explains (proprotional to the eigenvalue)

How many PC vectors?

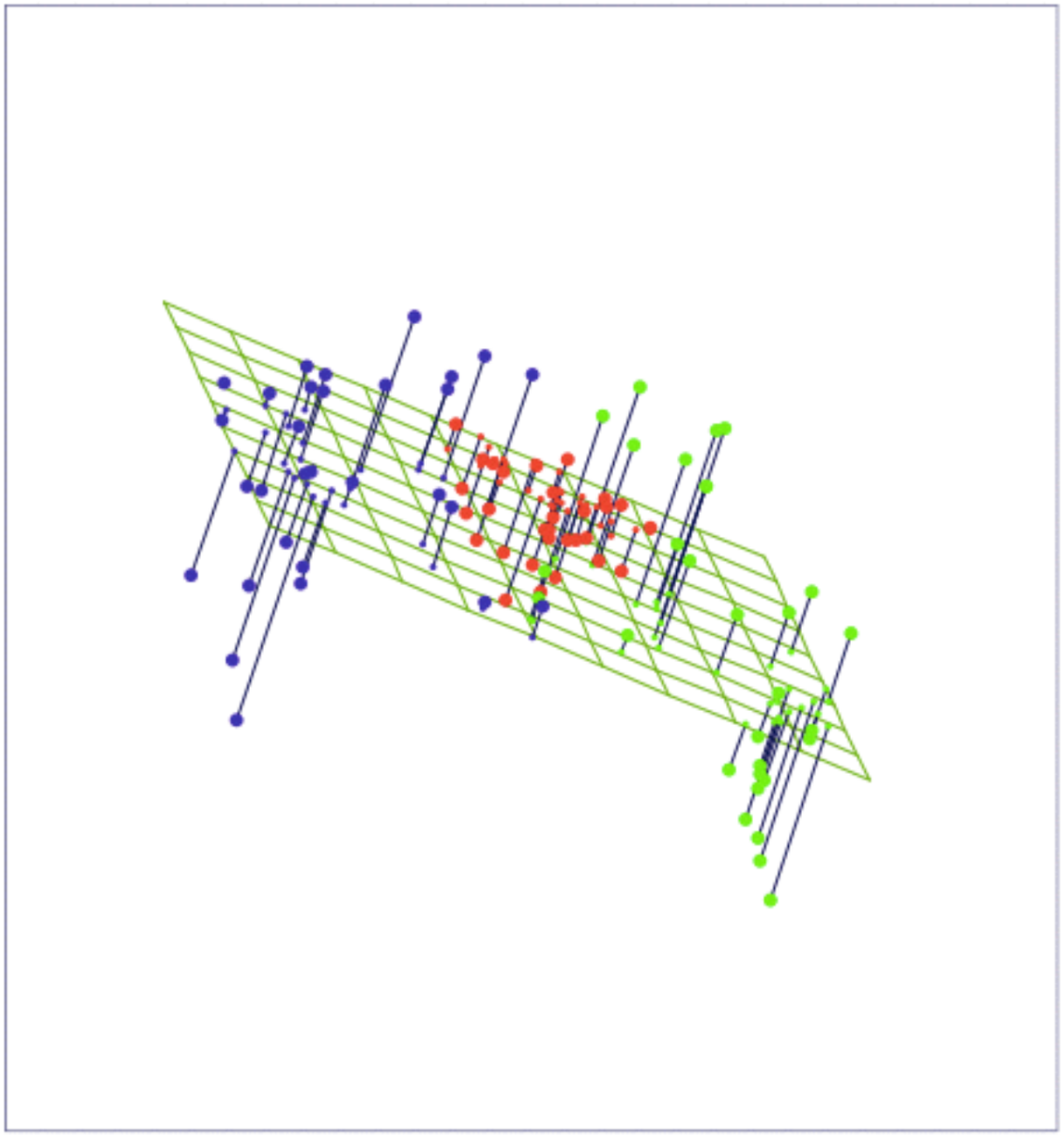
Enough PC vectors to cover 80-90% of the variance



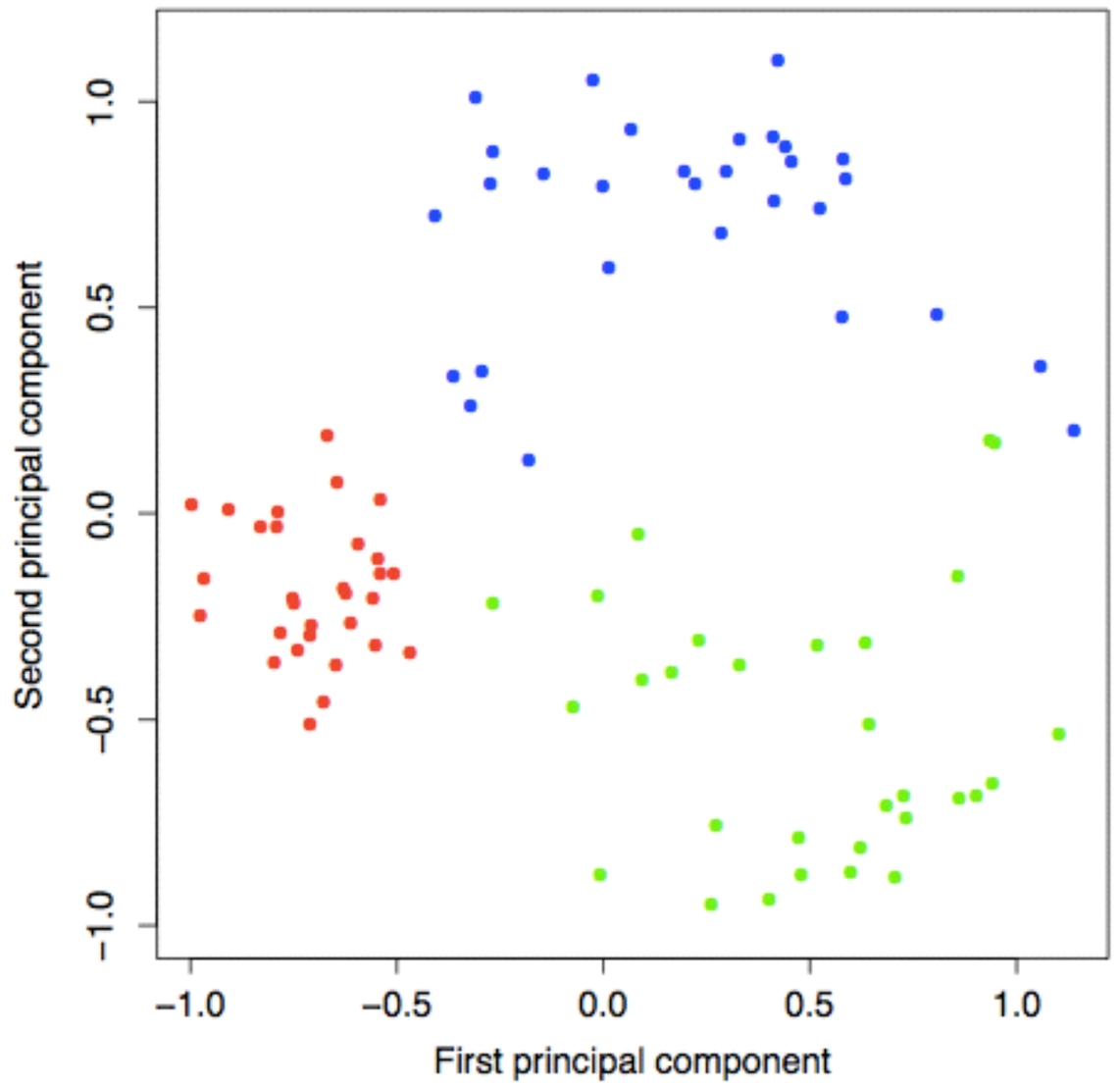
Screeplot

Based on slide from J. Leskovec

Dimensionality Reduction



Visualization: pick 1st two principal components to display.



Hastie et al.,"The Elements of Statistical Learning: Data Mining, Inference, and Prediction", Springer (2009)

post office cares: problem is considered solved now.

PCA for Handwritten Digits

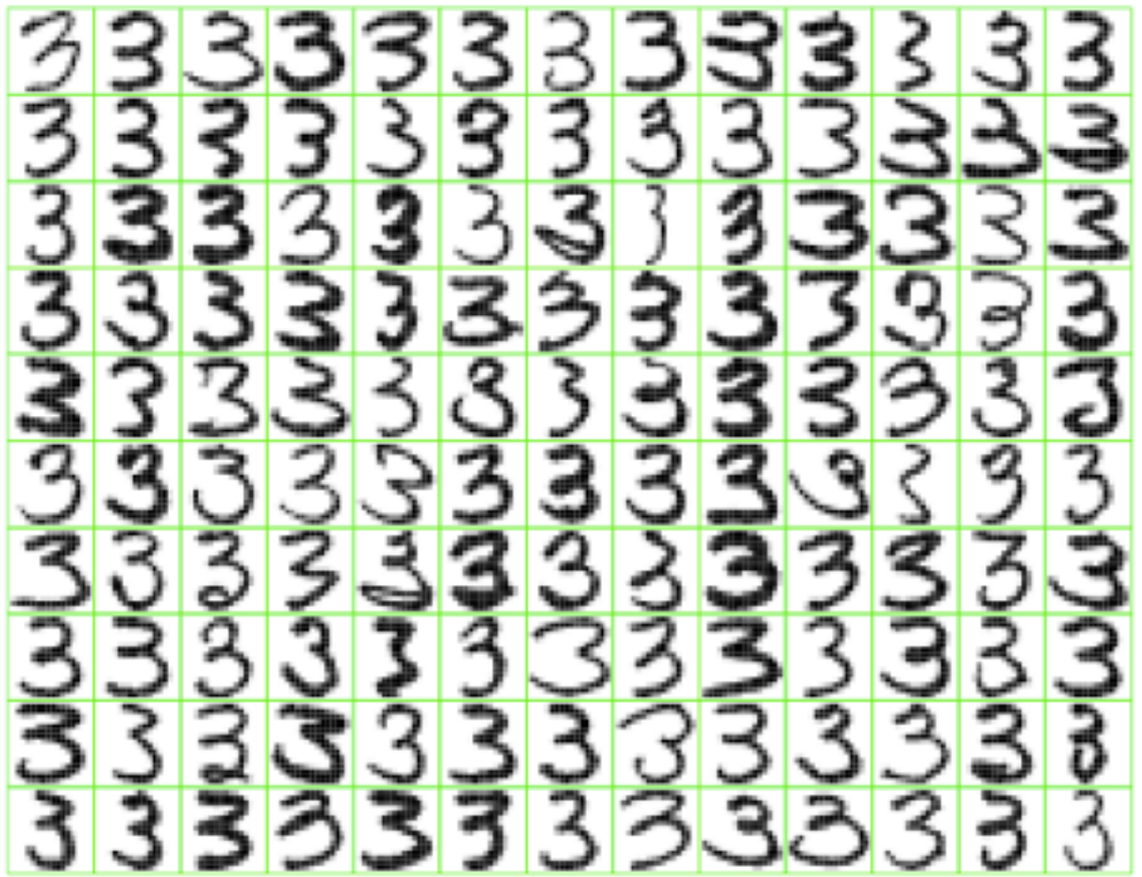


FIGURE 14.22. *A sample of* 130 *handwritten 3's shows a variety of writing styles.*

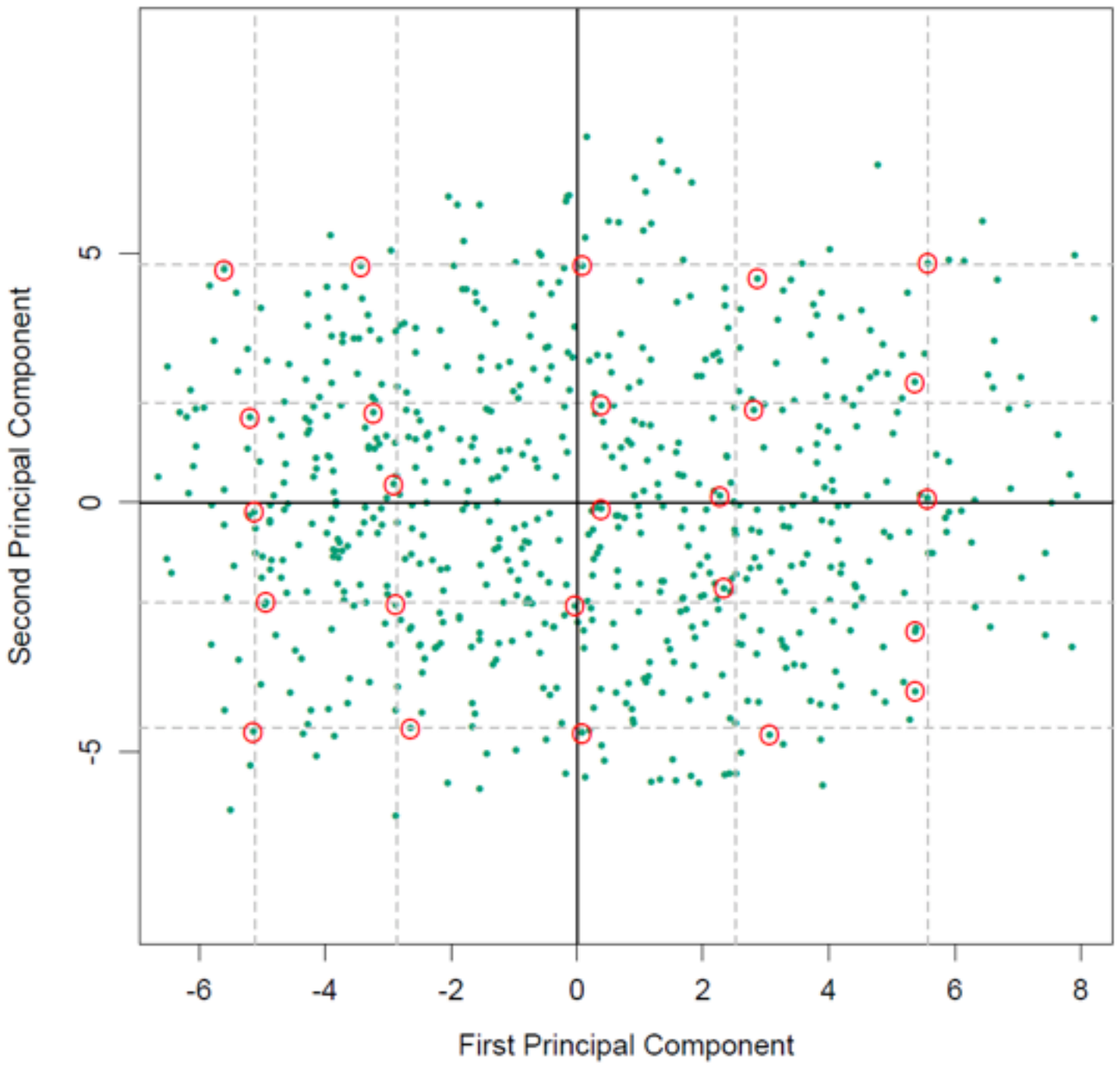
explain any image by the linear combination of mean and two principal components —> what is reconstruction error?

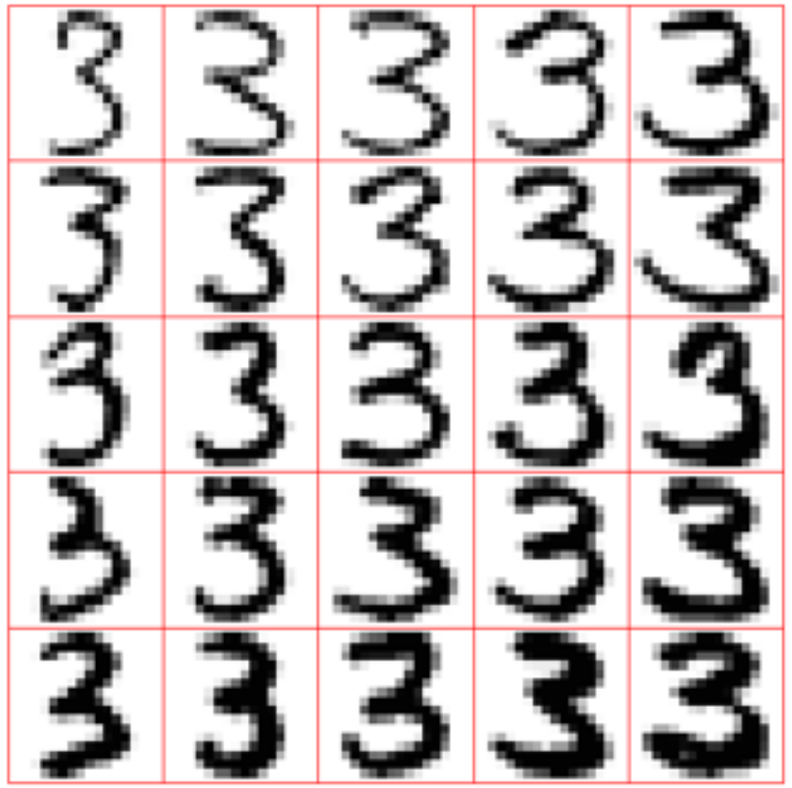
$$\hat{f}(\lambda) = \bar{x} + \lambda_1 v_1 + \lambda_2 v_2$$

$$= \bar{x} + \lambda_1 \cdot v_1 + \lambda_2 \cdot v_2 .$$

Hastie et al.,"The Elements of Statistical Learning: Data Mining, Inference, and Prediction", Springer (2009)

PCA for Handwritten Digits

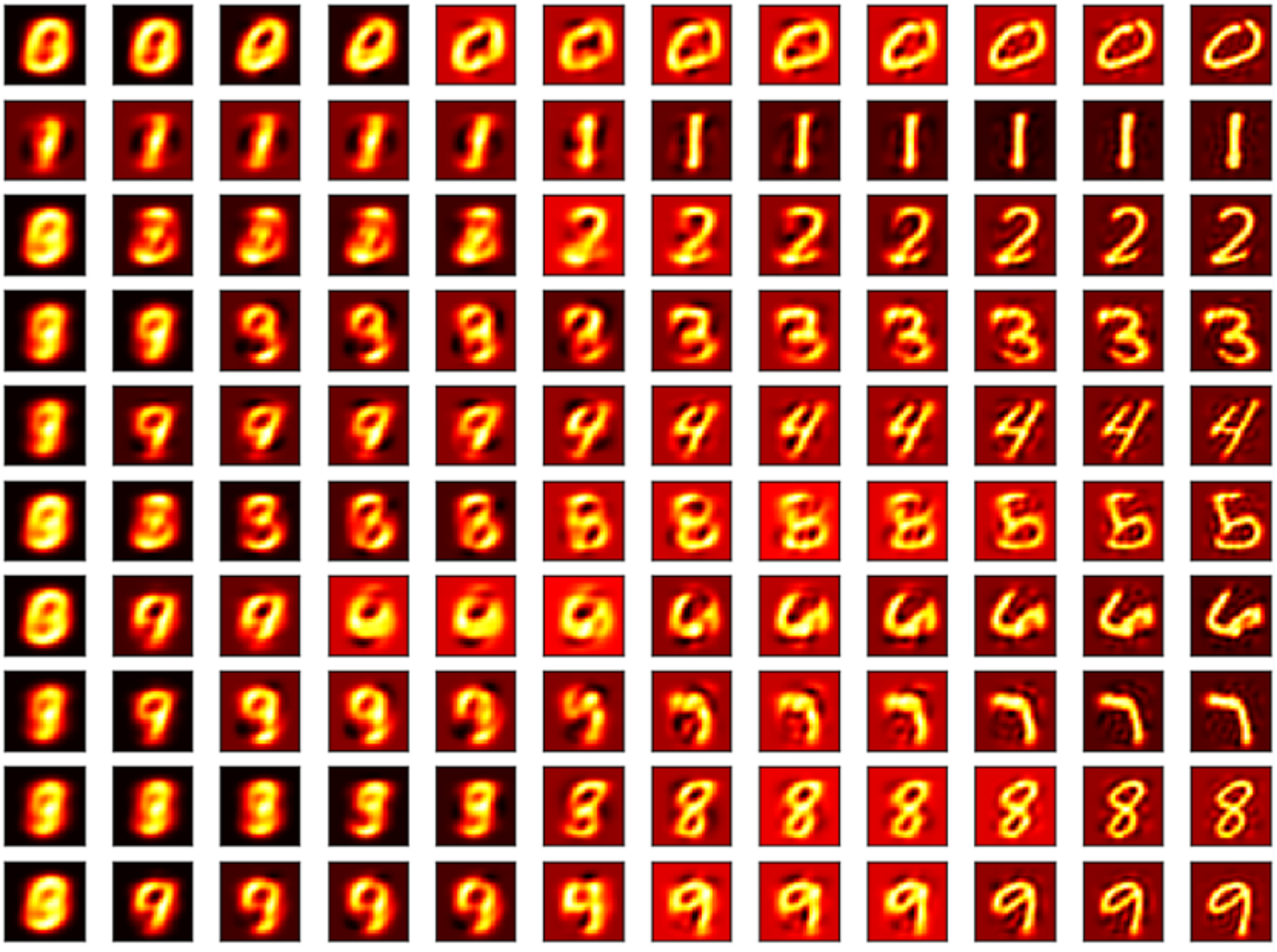




Hastie et al.,"The Elements of Statistical Learning: Data Mining, Inference, and Prediction", Springer (2009)

Image reconstruction after reduction with PCA

More components: better recponstruction.



PCA for Face Images

Based on slide from T. Yang

PCA for Face Images

- 64x64 images of faces = 4096 dimensional data

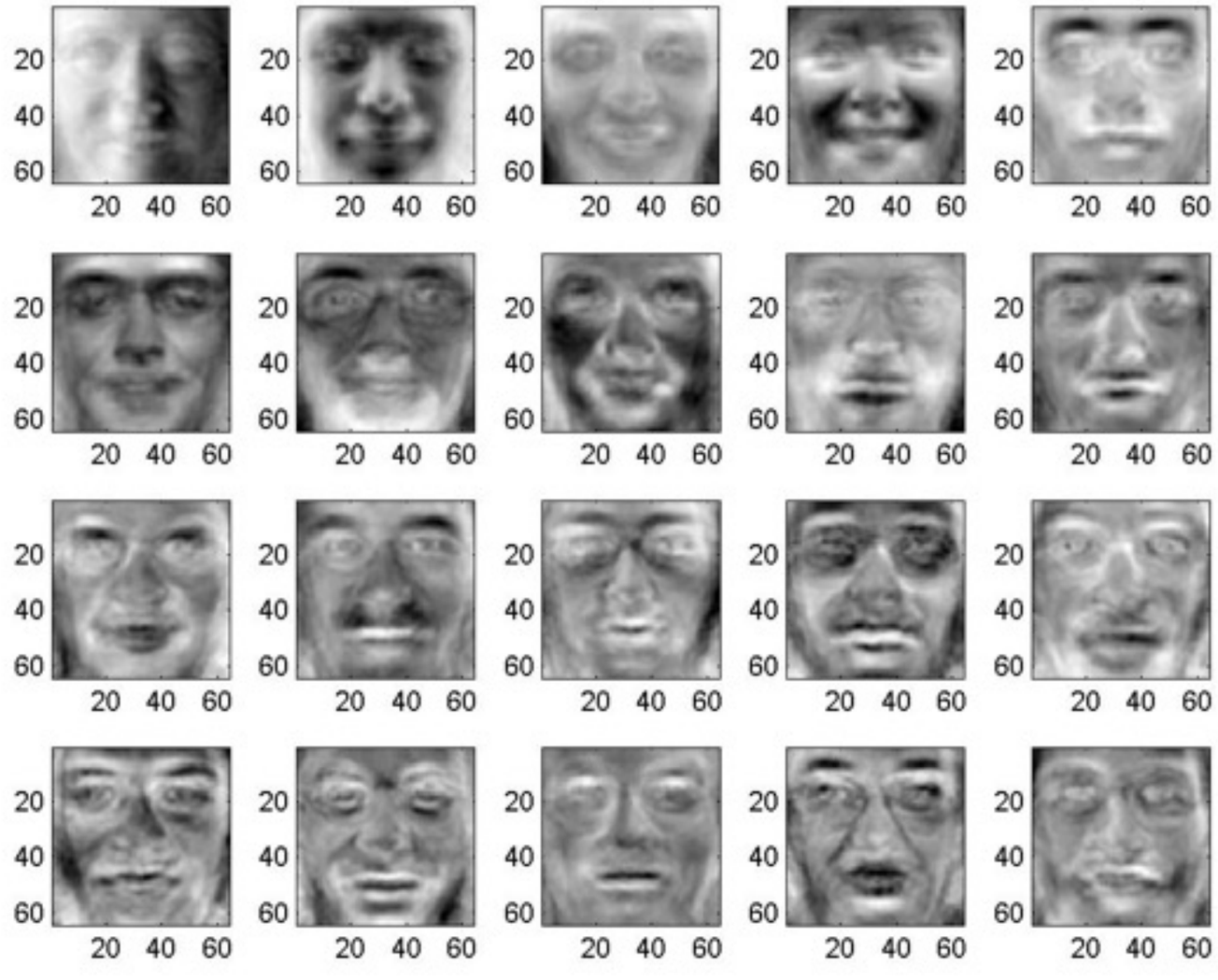
Based on slide from A. Ihler

“Eigenfaces”

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

Average Face

+



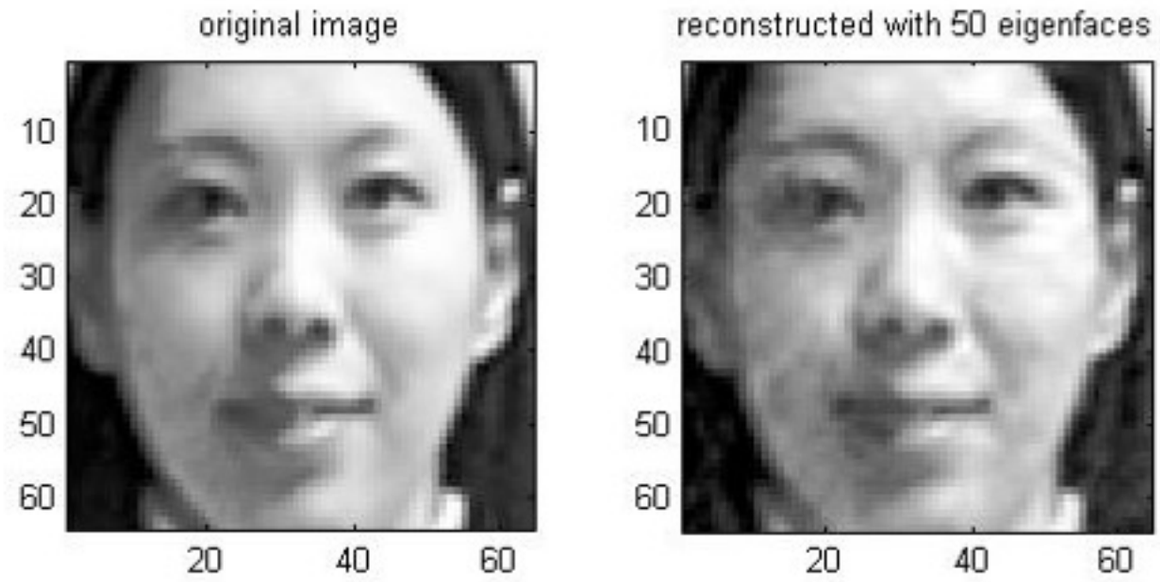
Eigenfaces

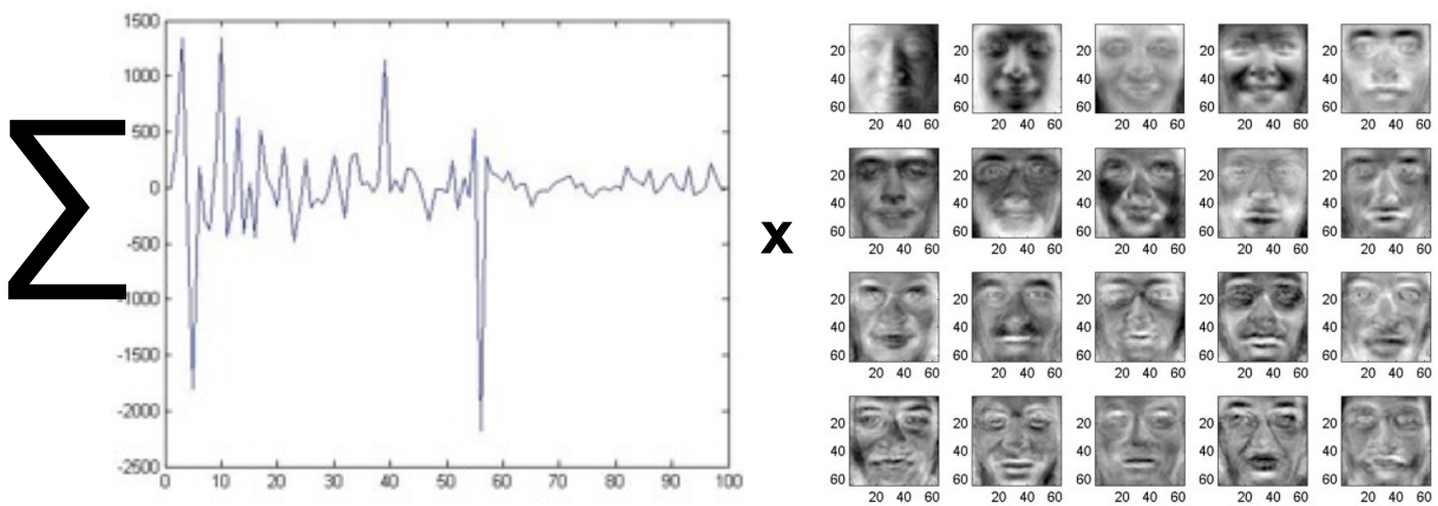
Only first few faces seem to matter: then only incremental changes.

Based on slide from T. Yang

Reconstruction

90% variance is captured by the first 50 eigenvectors





Based on slide from T. Yang

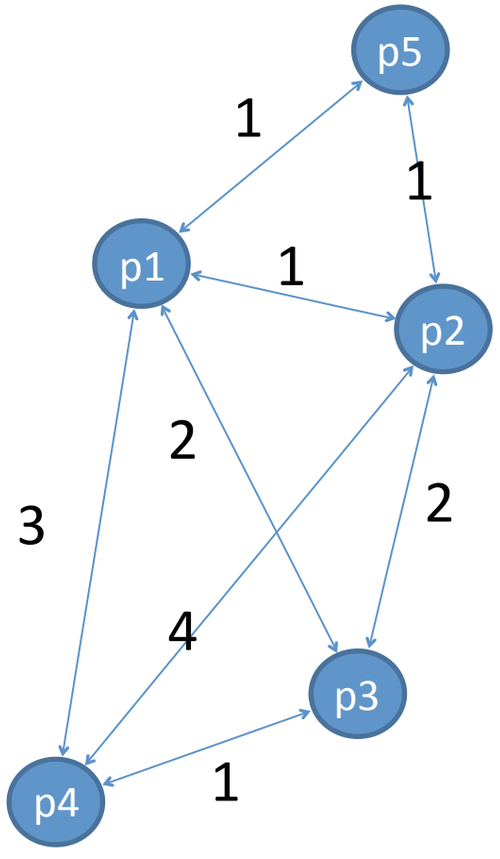
Multidimensional Scaling (MDS)

Instead of giving raw data: give matrix of distance vectors (pair-wise)

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

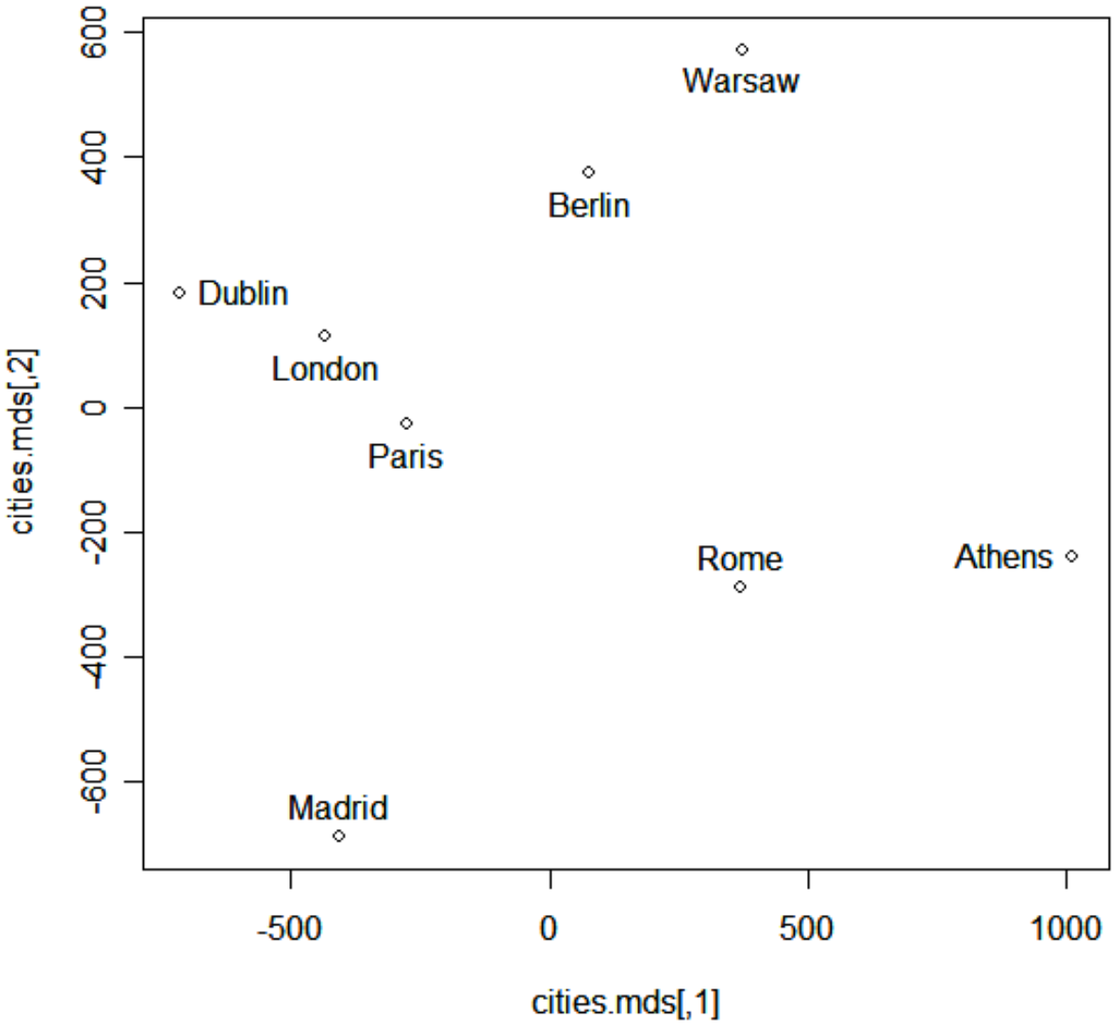
do projection and preserves distances as much as possible
(not minimizing orthogonal distances)

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



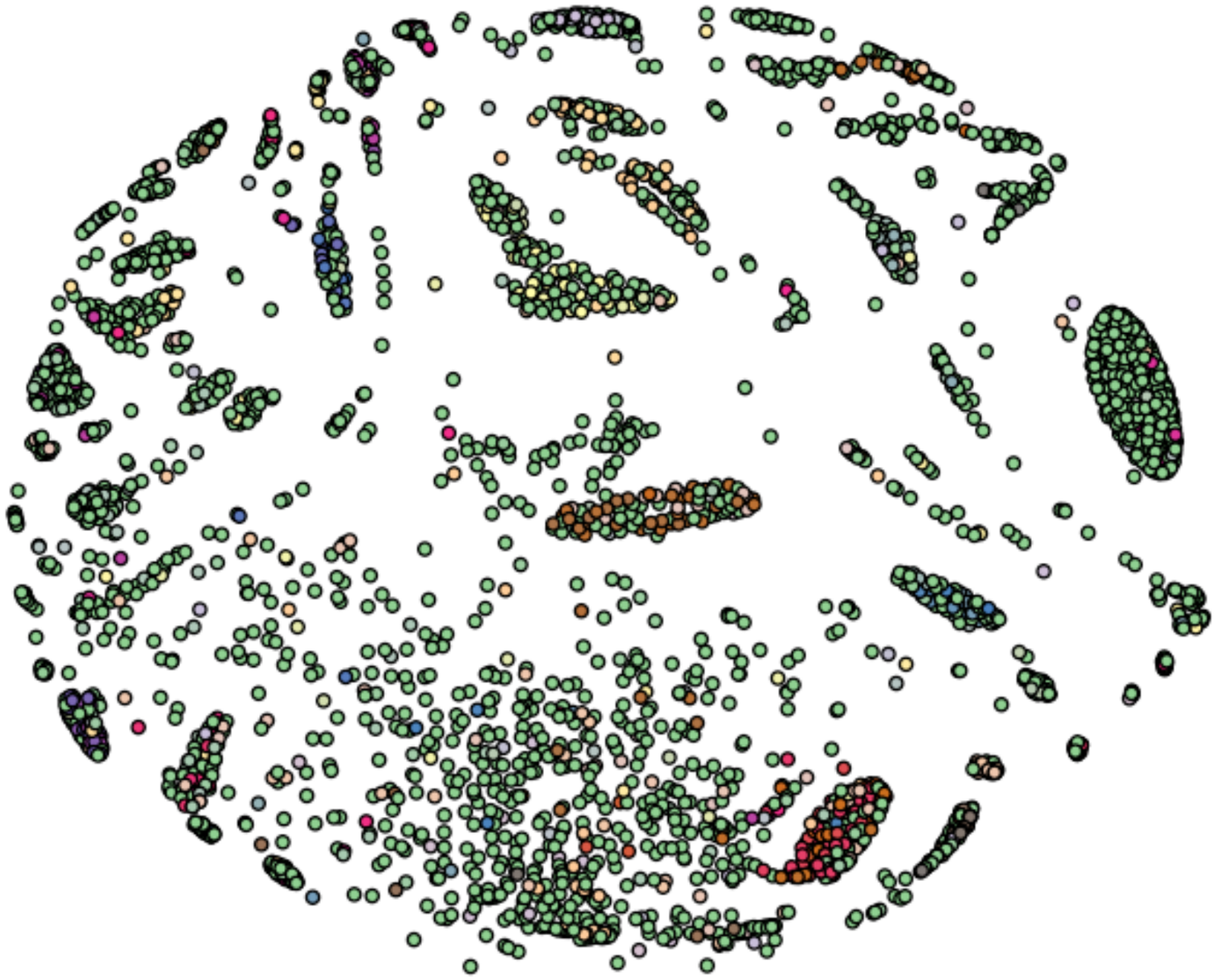
projection preserved original distances.

Based on slide from T. Yang

Color Images

N. Bonneel

Facebook Friends



– Distance = 1 for friends

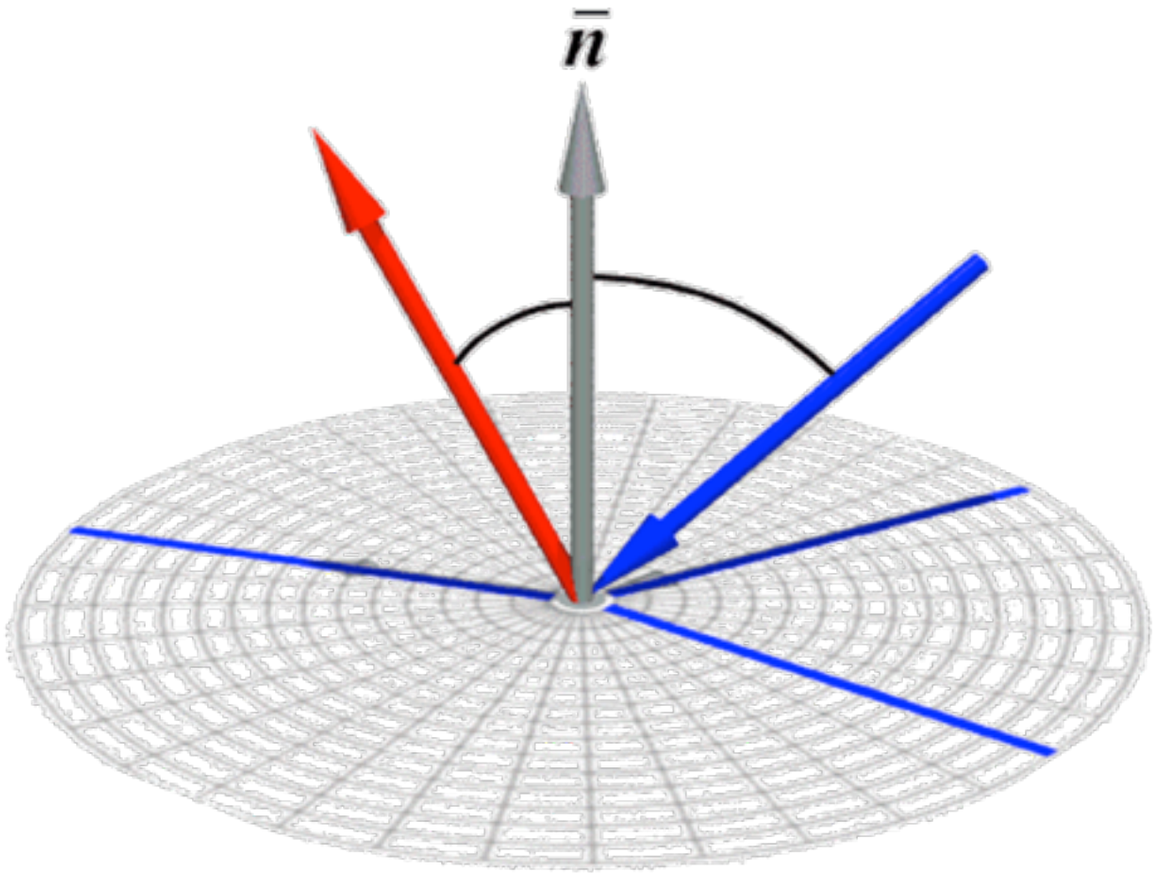
– Distance = 2 for friends of friends ; etc.

N. Bonneel

Nonlinear Methods

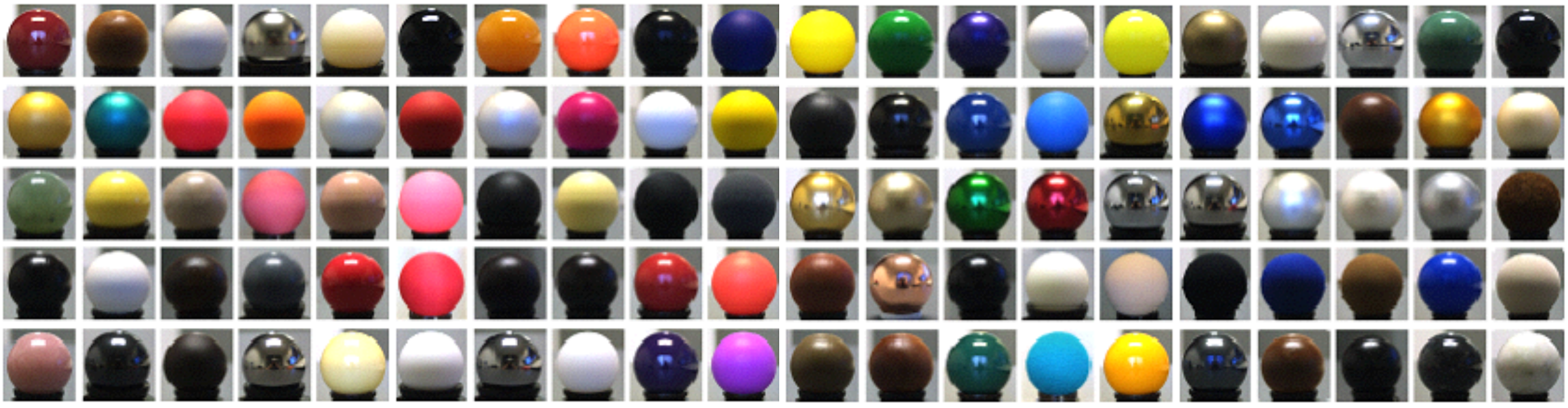
Data-Driven BRDFs

- Bi-Directional Reflectance Distribution Functions



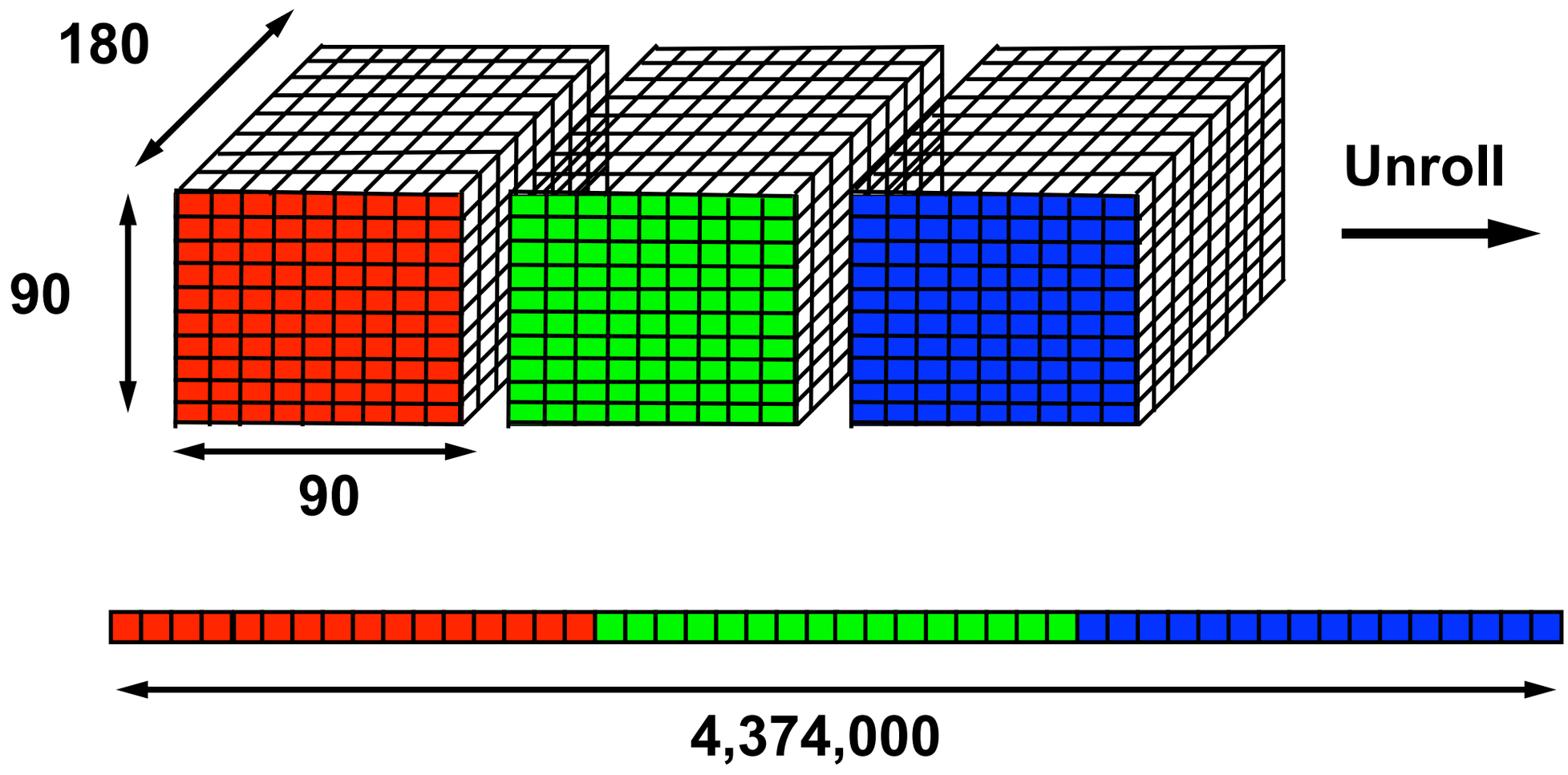
Data-Driven BRDFs

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

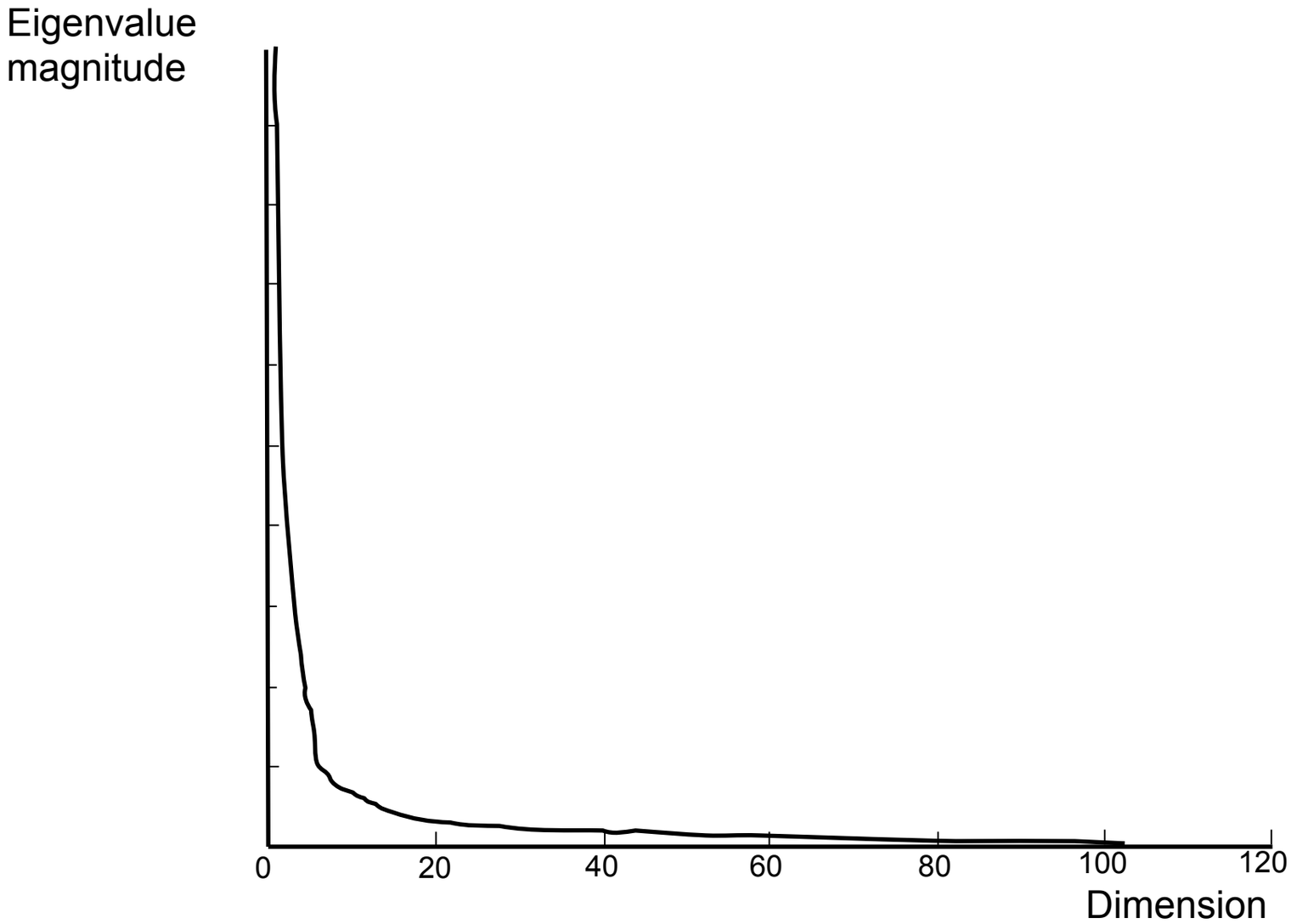


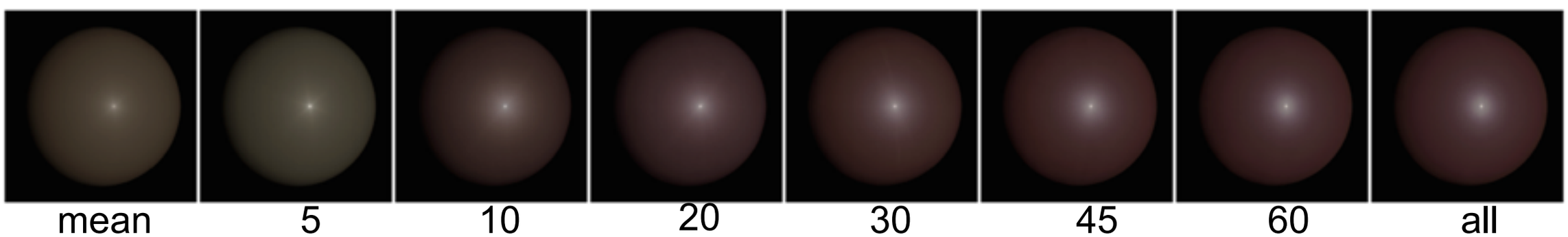
Data-Driven BRDFs

- Each tabulated BRDF is a vector in 90x90x180x3 =4,374,000 dimensional space



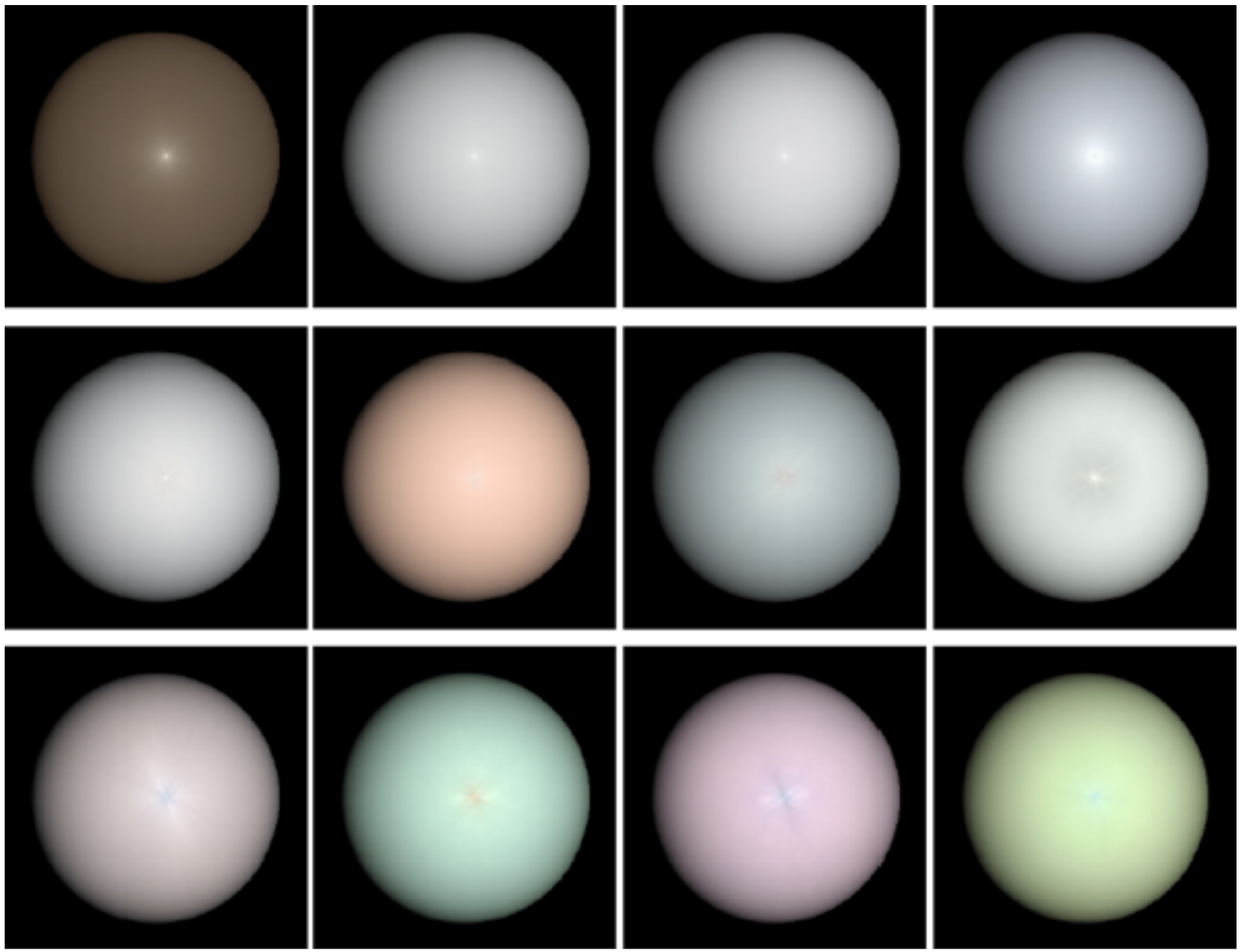
PCA


Eigenvalue
magnitude
0
20
40
60
80
100
120
Dimension


mean
5
10
20
30
45
60
all

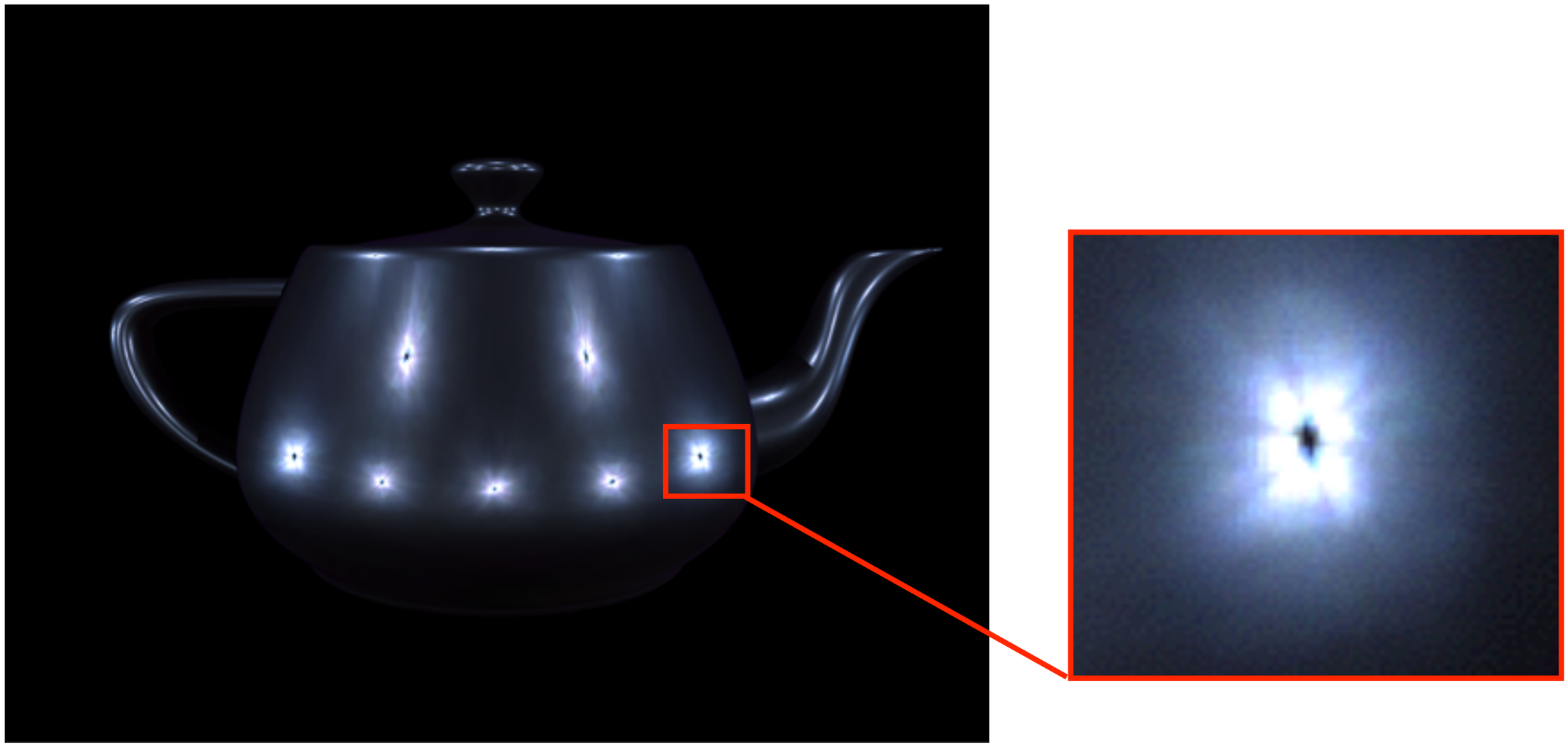
PCA

- First 11 PCA components

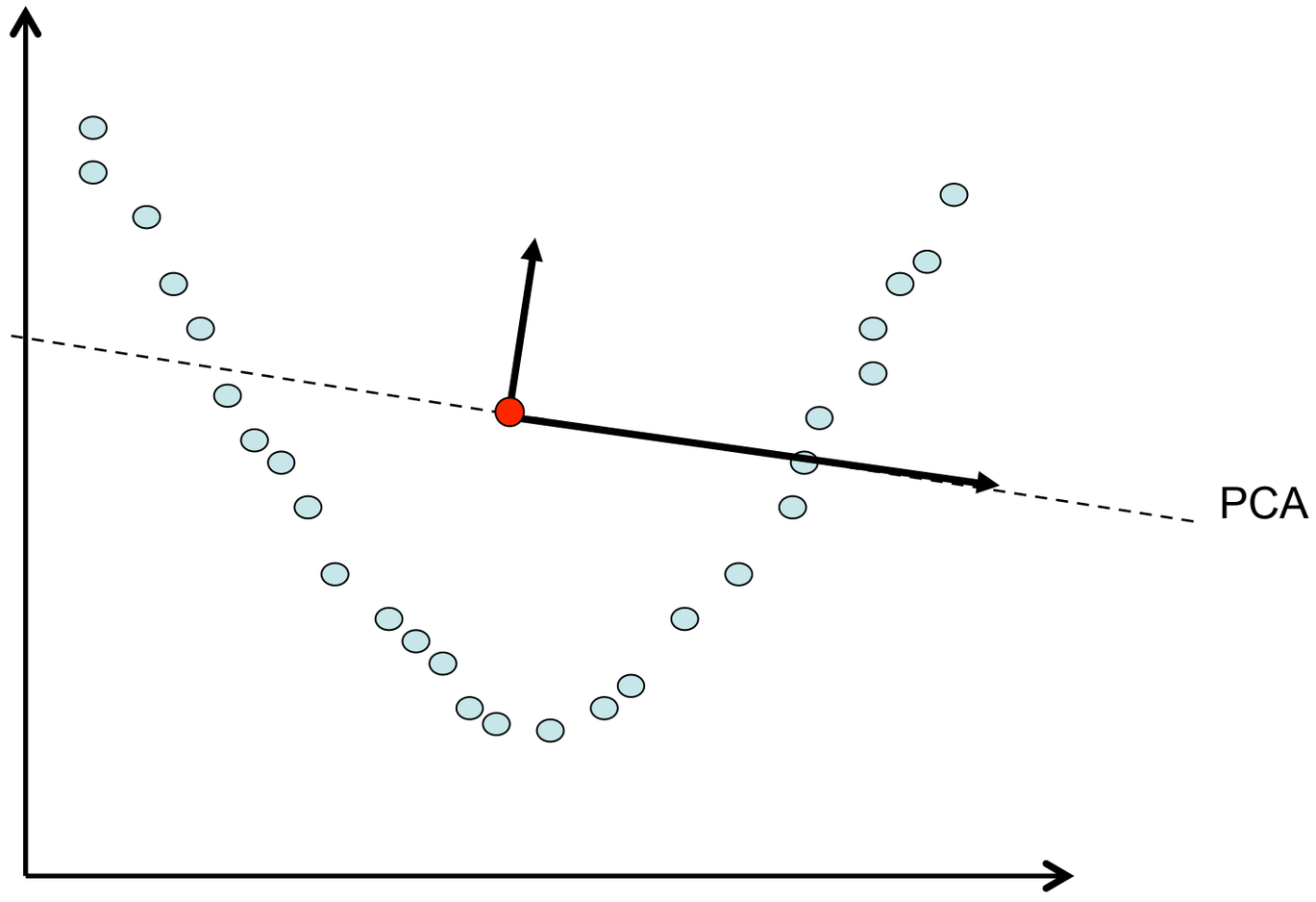


PCA Interpolation

Then, one day...

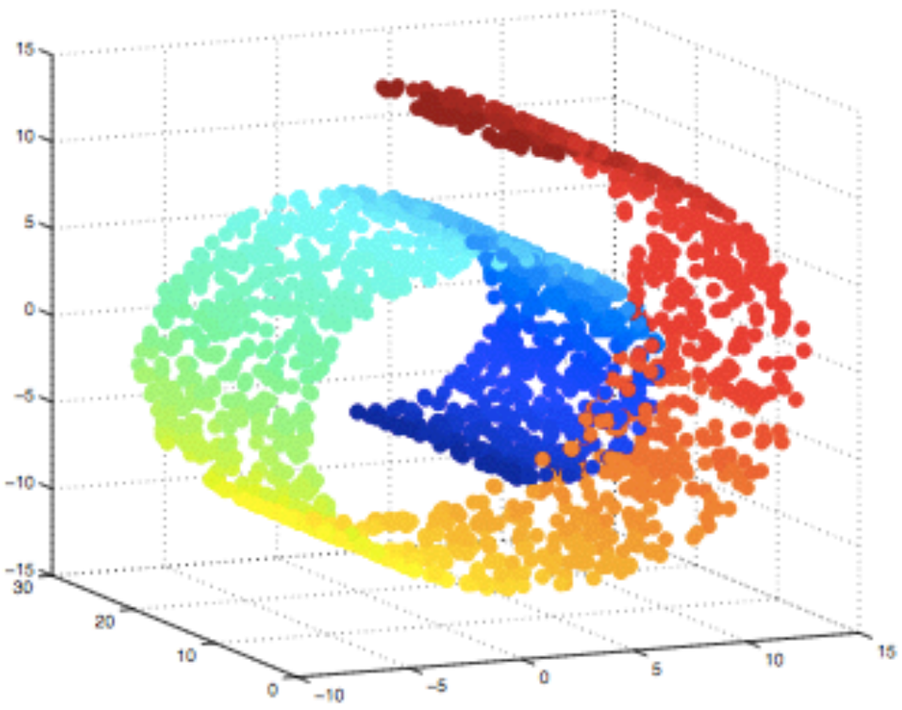


Why do linear models fail?

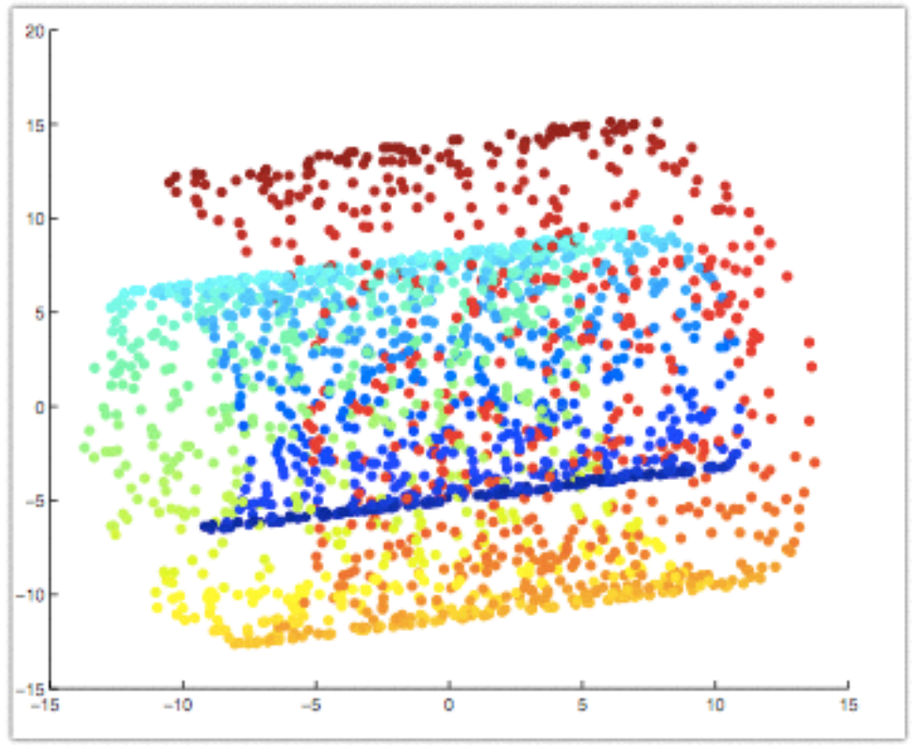


Why do linear models fail?

- Classic "Swiss Roll" example



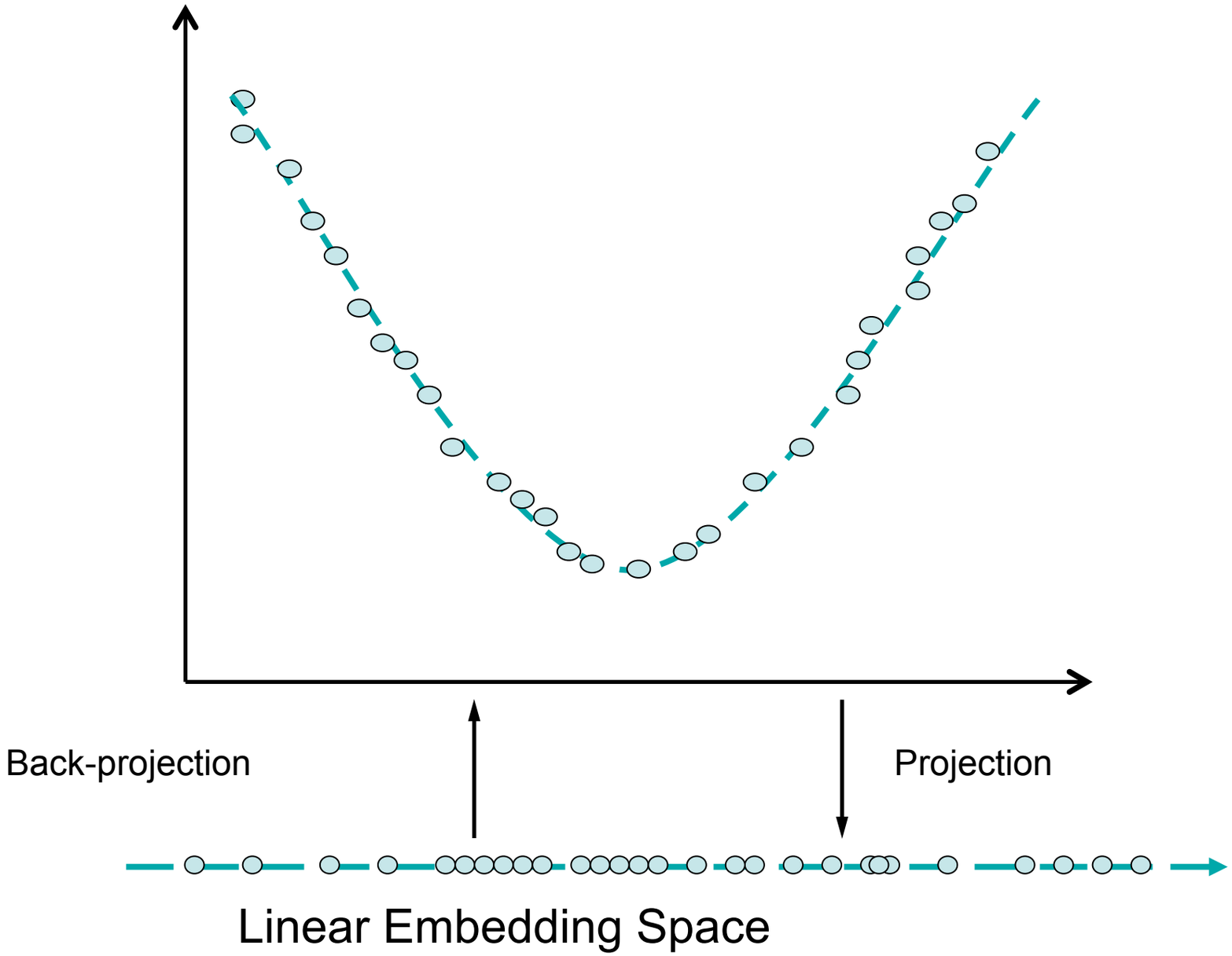
x_i



PCA

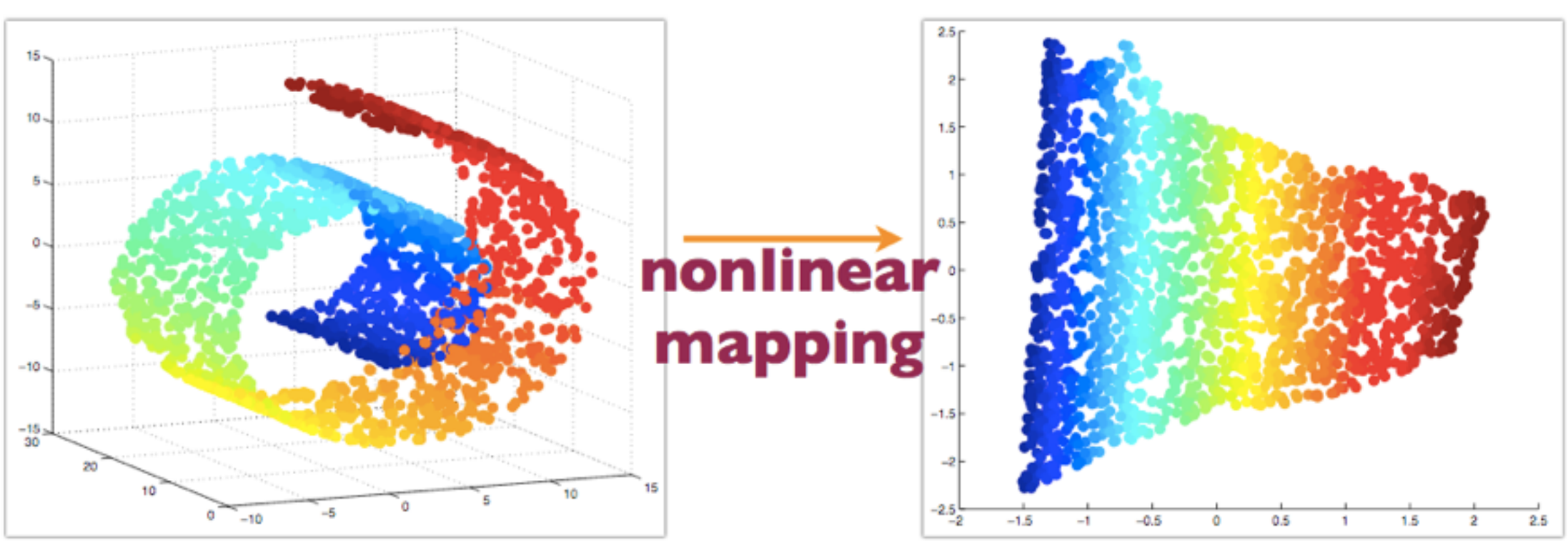
Based on slide from F. Sha

Non-Linear Manifold Methods



Non-Linear Manifold Methods

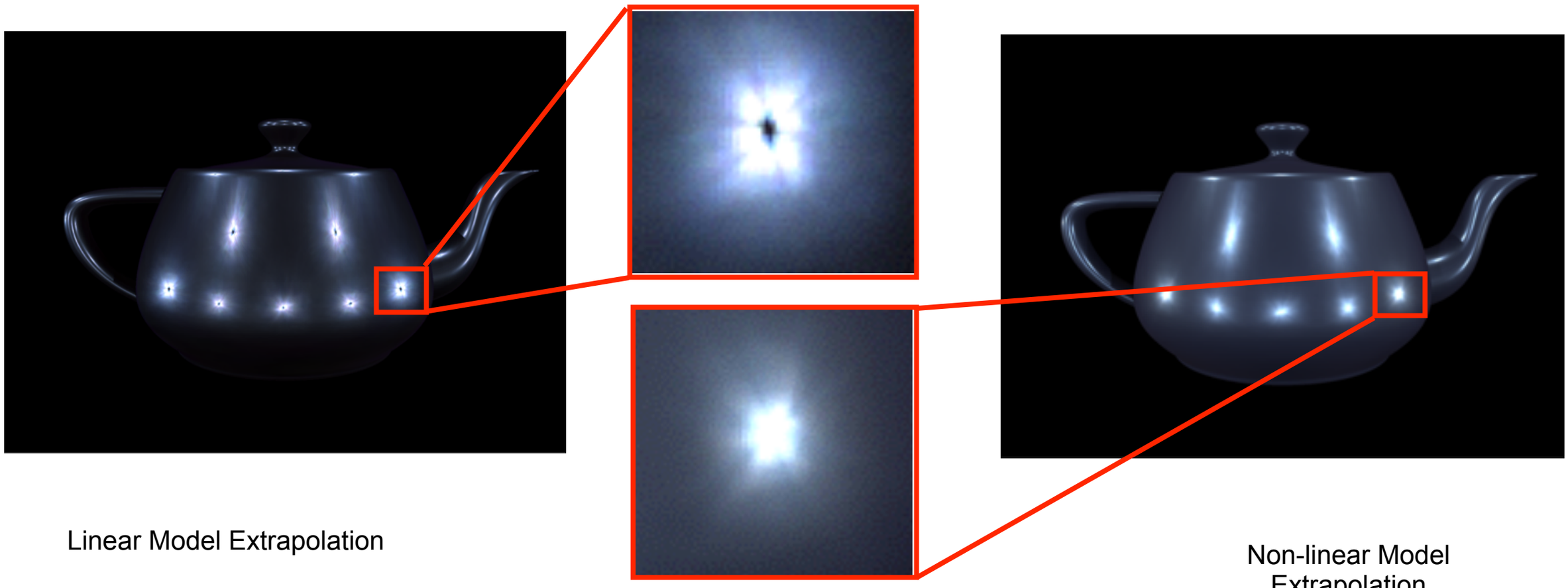
- Intuition: Distortion in local areas, but faithful in the global structure



Based on slide from F. Sha

Non-Linear BRDF Model

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



Linear Model Extrapolation

Non-linear Model Extrapolation

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 annote 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