Machine Learning Foundations
2018/02/05-06 @ Taiwan AI Academy

Albert Y. C. Chen, Ph.D.
Vice President, R&D
Viscovery

Albert Y. C. Chen, Ph.D.
陳彥呈博士

albert@viscovery.com
http://www.linkedin.com/in/aycchen
http://slideshare.net/albertycchen

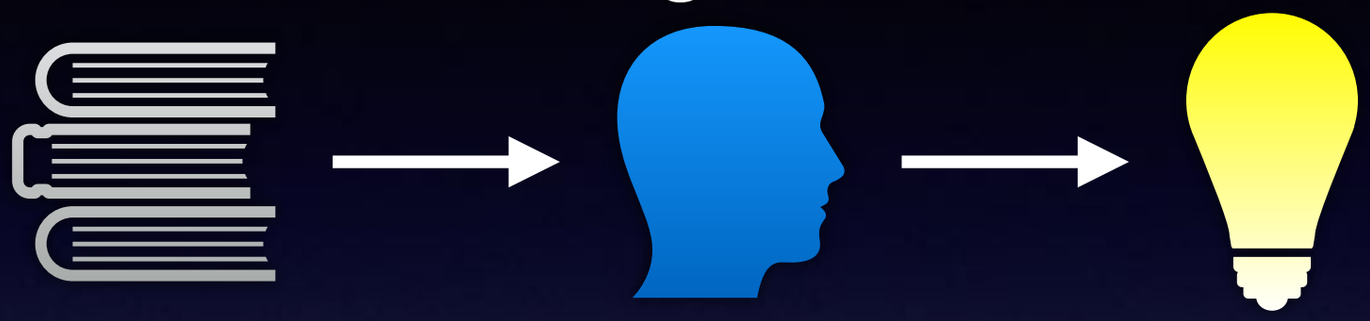
- *Experience*
  2017-present: Vice President of R&D @ Viscovery
  2015-2017: Chief Scientist @ Viscovery
  2015-2015: Principal Scientist @ Nervve Technologies
  2013-2014 Senior Scientist @ Tandent Vision Science
  2011-2012 @ GE Global Research, Computer Vision Lab
- *Education*
  Ph.D. in Computer Science, SUNY-Buffalo
  M.S. in Computer Science, NTNU
  B.S. in Computer Science, NTHU

When something is important enough,
you do it even if the odds are not in your favor.

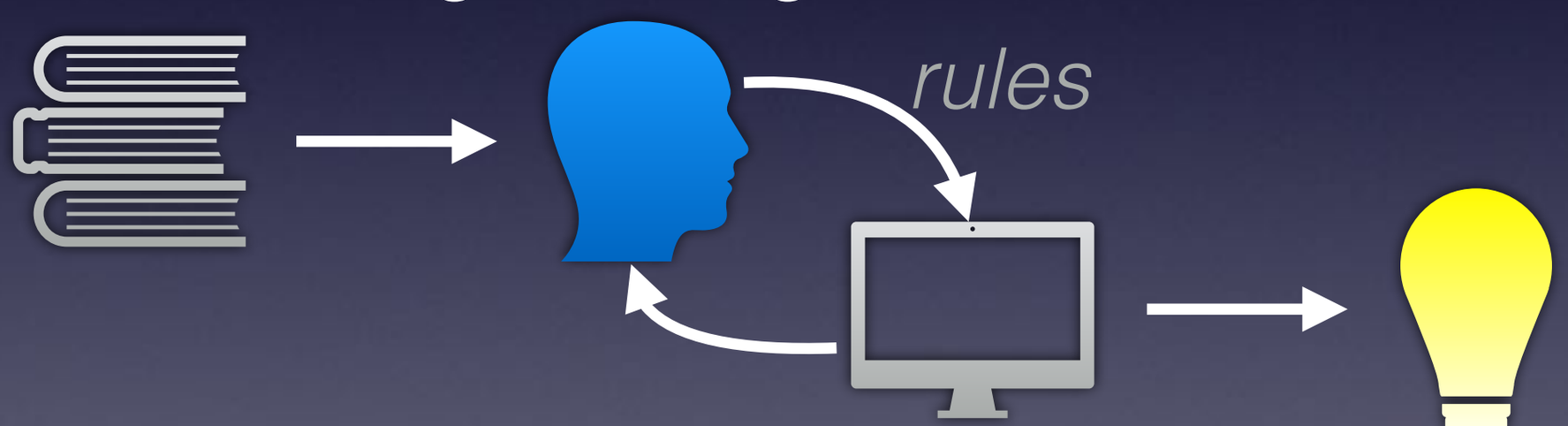
Elon Musk

What is “Machine Learning”?

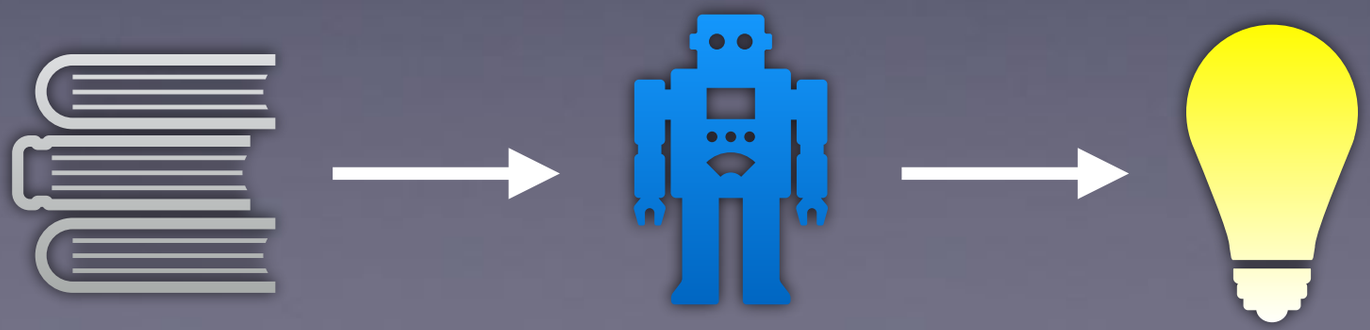
- Human Learning:



- Manual Programming:

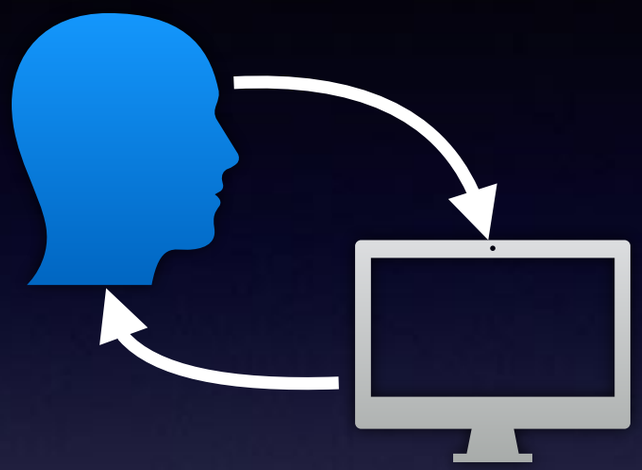


- Machine Learning (ML):



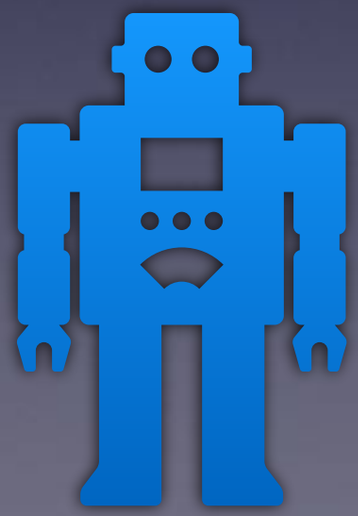
Manual Programming vs Machine Learning

When to manual program?



- Deterministic problems: repeat 1B times, still get the same answer,
- problems lacking data,
- problems with easily separable data.

When to use machine learning?



- Data with noise,
- data of high dimension,
- data of large volume,
- data that changes over time.

Deep Learning, directly?

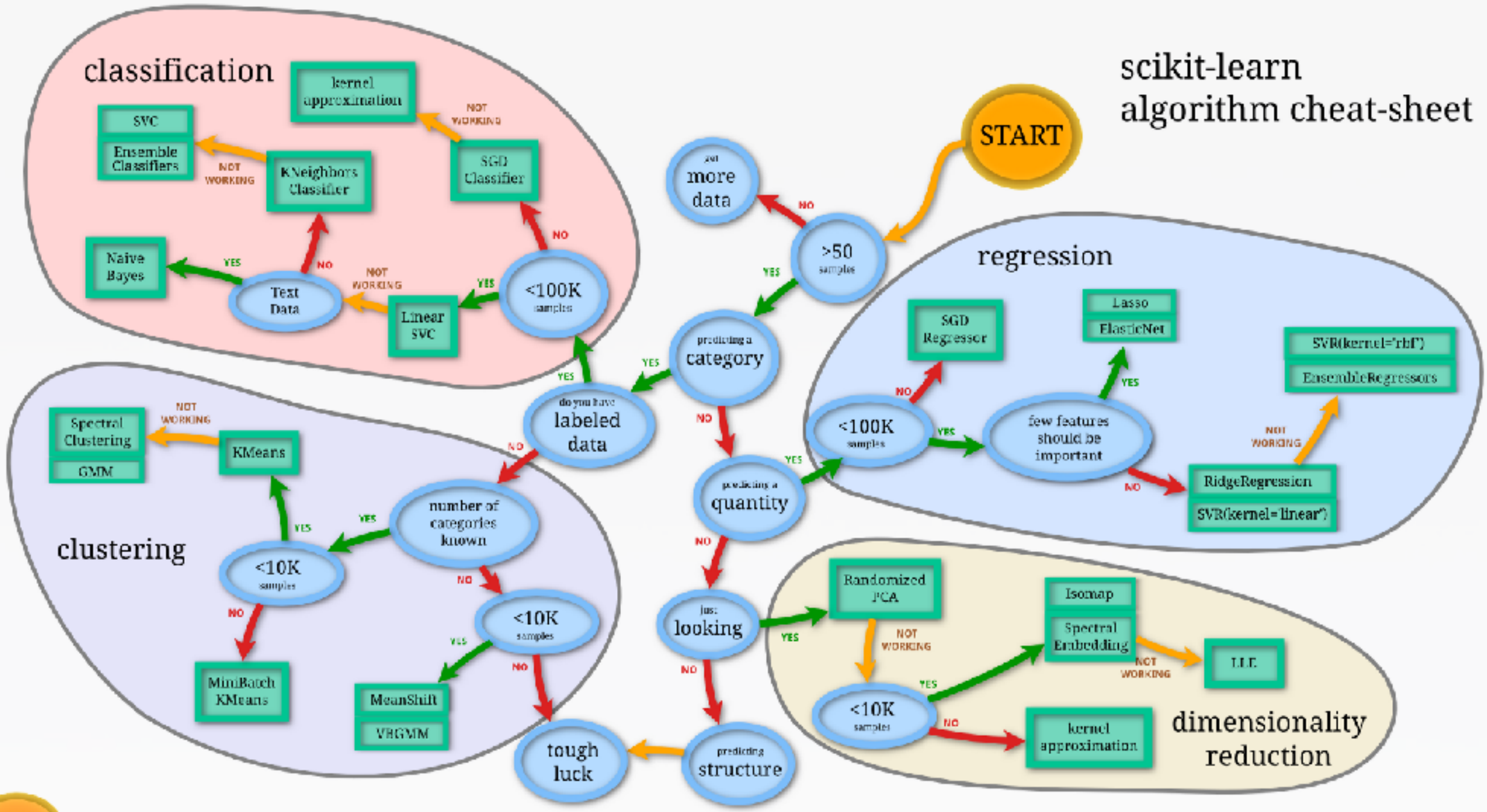
- Important concepts (lessons learned) from classical machine learning are still very important, from dimensionality, sampling, distance measures, error metrics, and generalization issues.
- Understand how things work, why things worked in the past, and why previously unattainable problems are solved by Deep Learning.

Where should we start?

We present you, a simple & usable map for ML!

unsupervised / supervised	continuous (predicting a quantity)	discrete (predicting a category)
unsupervised	**Dimension Reduction**	**Clustering**
supervised	**Regression**	**Classification**

ML Roadmap, in more detail



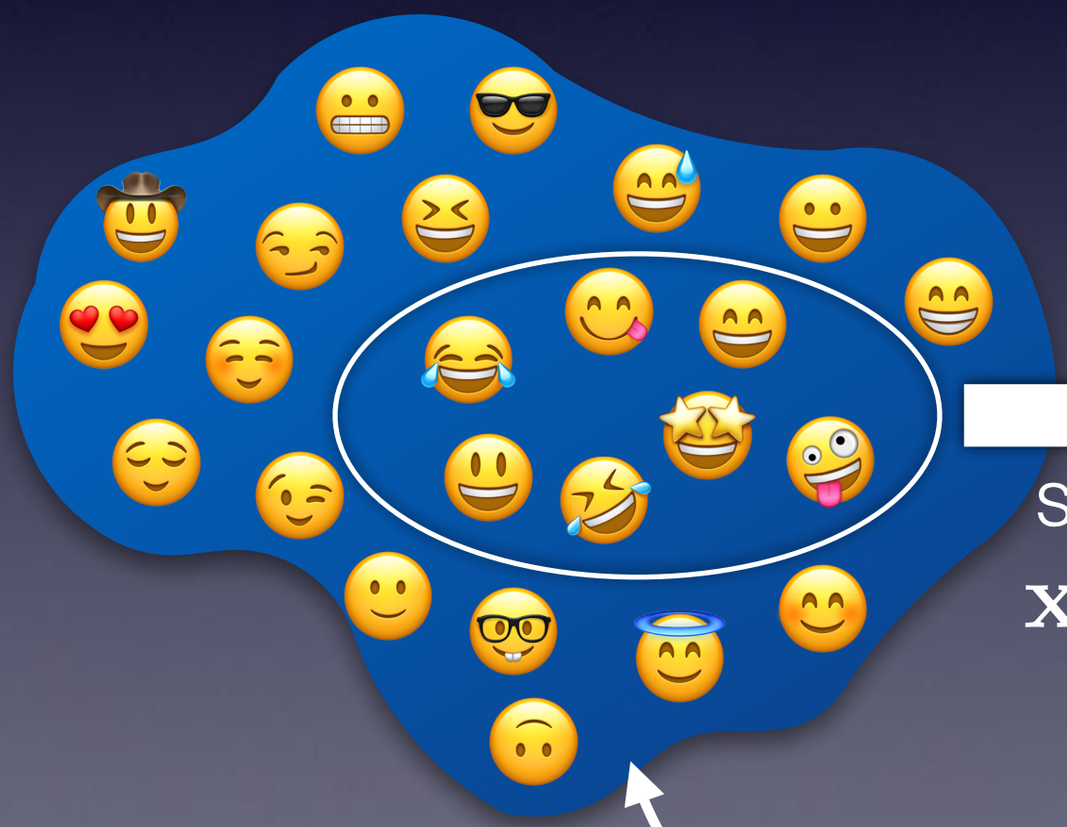
Steps for Supervised Learning

- Before we start, we need to estimate data distribution and develop sampling strategies,
- figure out how to measure/quantify data, or, in other words, represent them as features,
- figure out how to *split data* to training and validation set.
- After we learn a model, we need to measure the *fit*, or the *error* on validation set.
- Finally, how do we evaluate how well our trained model *generalize*.

Sampling & Distributions

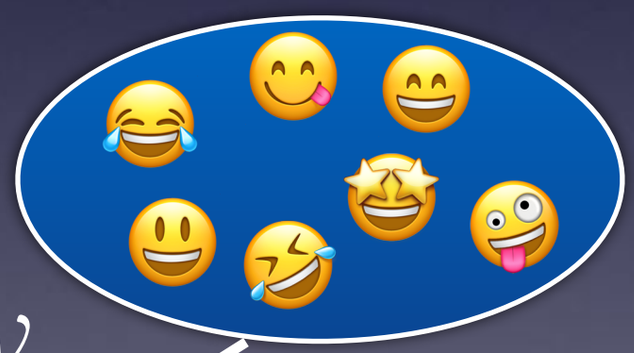
The importance of good sampling & distribution estimation.

Population $\mathcal{X}$ with attribute $\mathcal{Y}$
modeled by function $f : \mathcal{X} \rightarrow \mathcal{Y}$



Learn f' from $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$

sample
$\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}$



f' incorrectly predicts that everyone else "smiles crazily"

Sampling & Distributions

- The chances of getting a "perfect" sample of the population at first try is very very small. When the population is huge, this problem worsens.

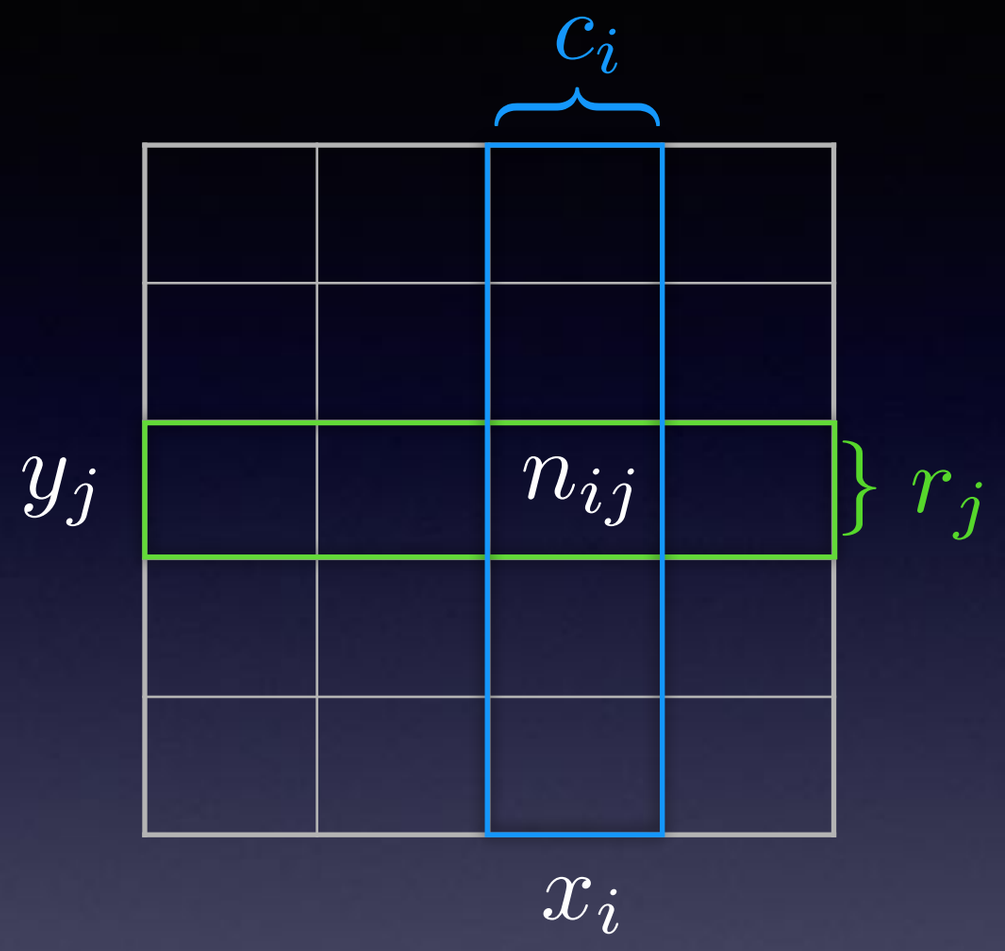
- Noise during the measurement process adds additional uncertainties.

- As a result, it is natural to try multiple times, and formulate the problem in a *probabilistic* way.

Probability Theory

- *Joint probability* of *X* taking the value x_i and *Y* taking the value y_i:

$$p(X = x_i, Y = y_i) = \frac{n_{ij}}{N}$$



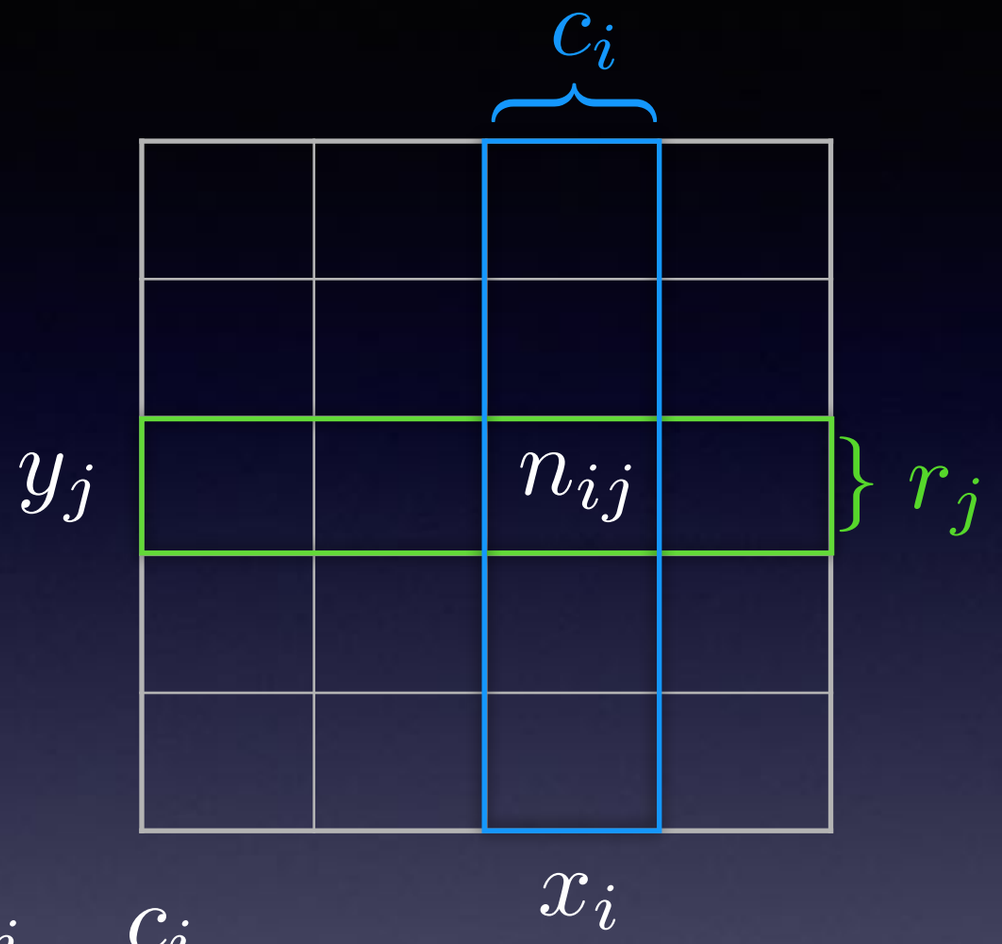
- *Marginalizing*: probability that *X* takes the value x_i irrespective of *Y*:

$$p(X = x_i) = \frac{c_i}{N}, \text{ where } c_i = \sum_j n_{ij}$$

Probability Theory

- *Conditional Probability: the fraction of instances where Y = y_j given that X = x_i.*

$$p(Y = y_j | X = x_i) = \frac{n_{ij}}{c_i}$$



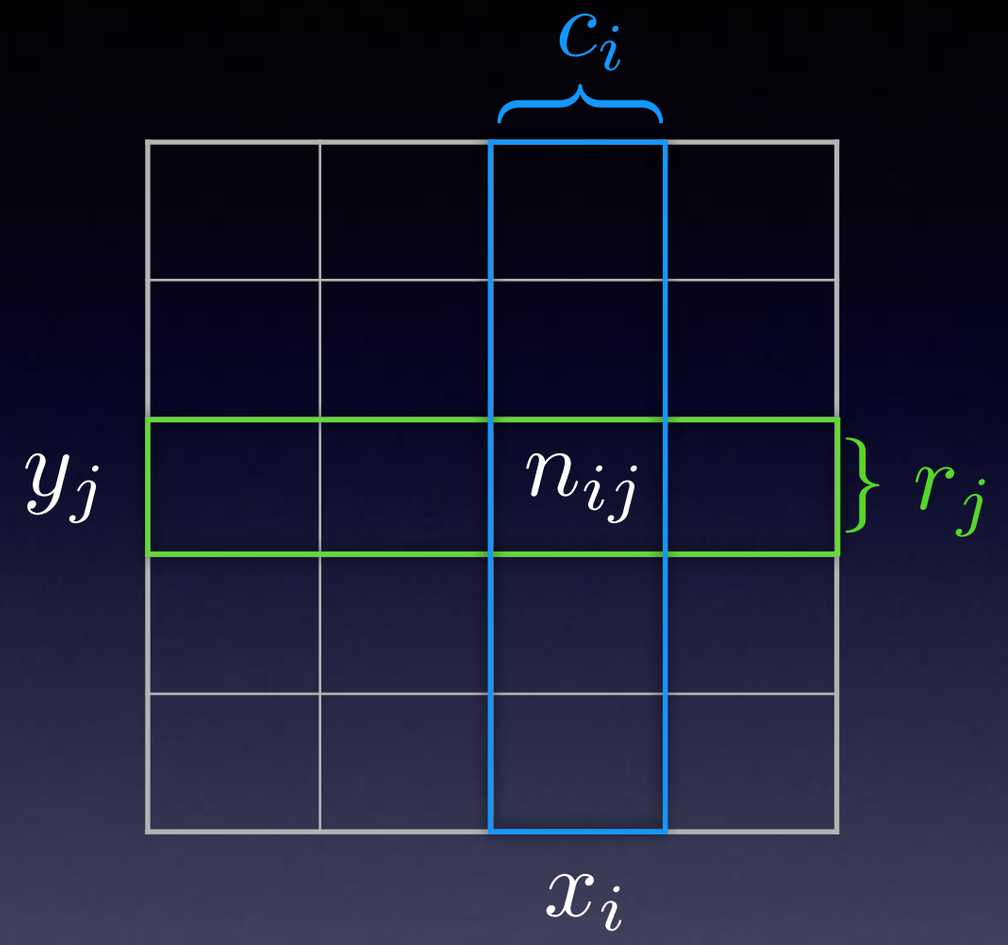
- *Product Rule*:

$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N} = \frac{n_{ij}}{c_i} \cdot \frac{c_i}{N}$$

$$= p(Y = y_j | X = x_i) p(X = x_i)$$

Bayes' Rule

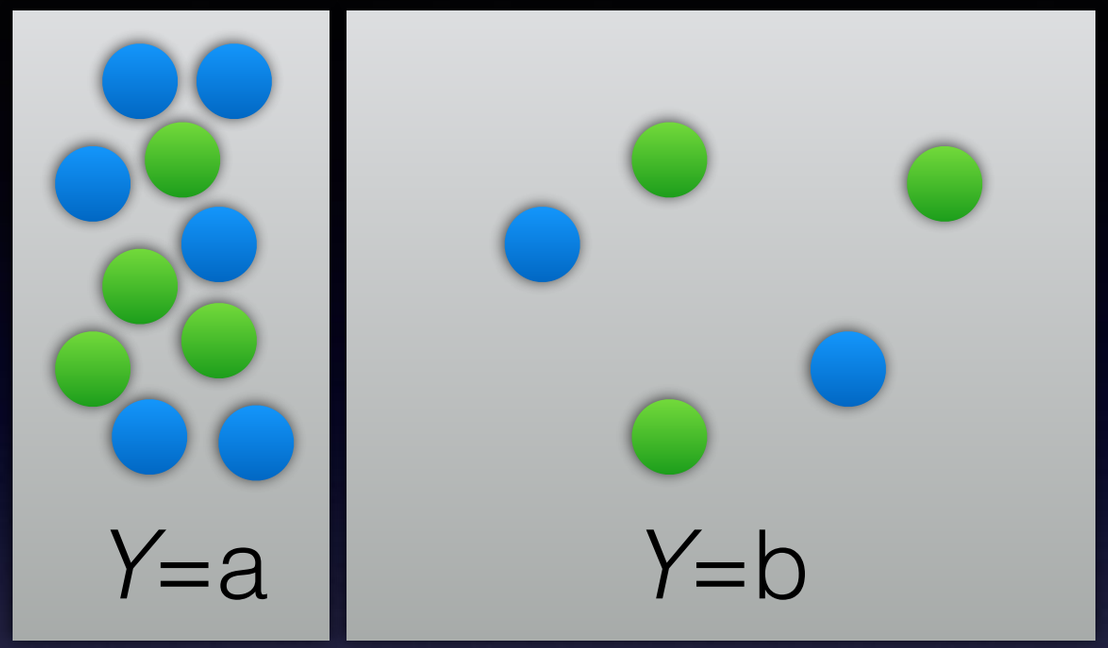
- Bayes' Rule plays a central role in pattern recognition and machine learning.
- From the product rule, together with the symmetric property $p(X,Y) = p(Y,X)$ we get:



$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}, \text{ where } p(X) = \sum_Y p(X|Y)p(Y)$$

Bayes' Rule Example 1

- p(Y = a) = 1/4, p(Y = b) = 3/4
- p(X = blue | Y = a) = 3/5
- p(X = green | Y = a) = 2/5



When we randomly draw a ball that is blue, the probability that it comes from Y=a is?

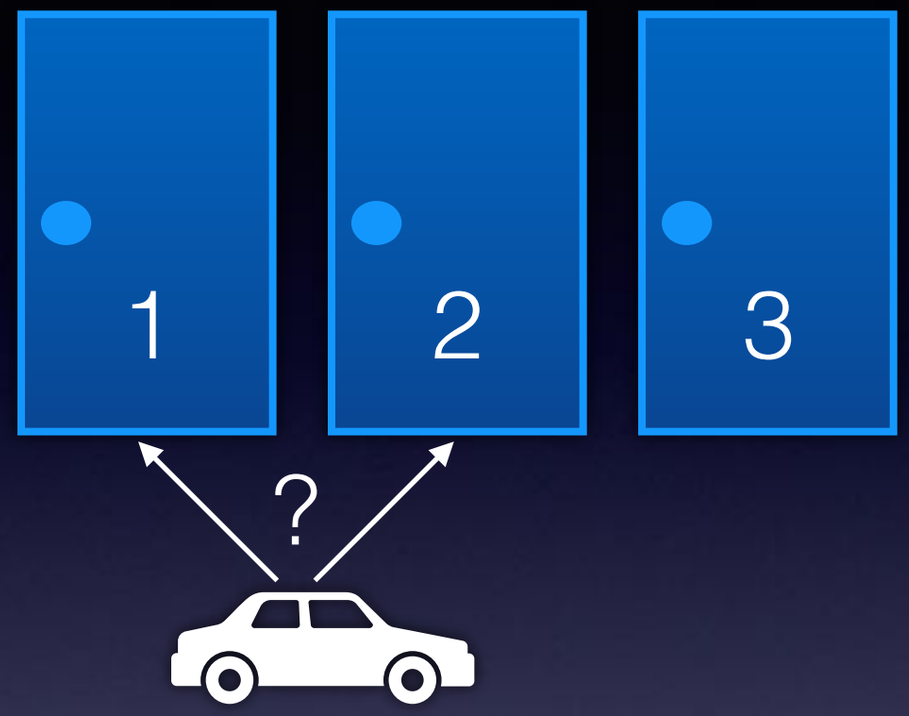
$$p(Y = a|X = \text{blue}) = \frac{p(X = \text{blue}|Y = a)p(Y = a)}{p(X = \text{blue})}$$

$$= \frac{p(X = \text{blue}|Y = a)p(Y = a)}{(p(X = blue|Y = a)p(Y = a) + (p(X = \text{blue}|Y = b)p(Y = b)}$$

$$= \frac{\frac{3}{5} \cdot \frac{1}{4}}{\frac{3}{5} \cdot \frac{1}{4} + \frac{2}{5} \cdot \frac{3}{4}} = \frac{\frac{3}{20}}{\frac{3}{20} + \frac{6}{20}} = \frac{\frac{3}{20}}{\frac{9}{20}} = \frac{1}{3}$$

Bayes' Rule Example 2

- Monty Hall problem
  - Prize behind one of the three doors. After choosing door 1, the host opens empty door 3 and asks if you want to switch your choice. Should you switch?



Behind door 1	Behind door 2	Behind door 3	Result if staying at door 1	Result if switching to the door offered
Car	Goat	Goat	Wins car	Wins goat
Goat	Car	Goat	Wins goat	Wins car
Goat	Goat	Car	Wins goat	Wins car

Features

vs

baseball tennis ball

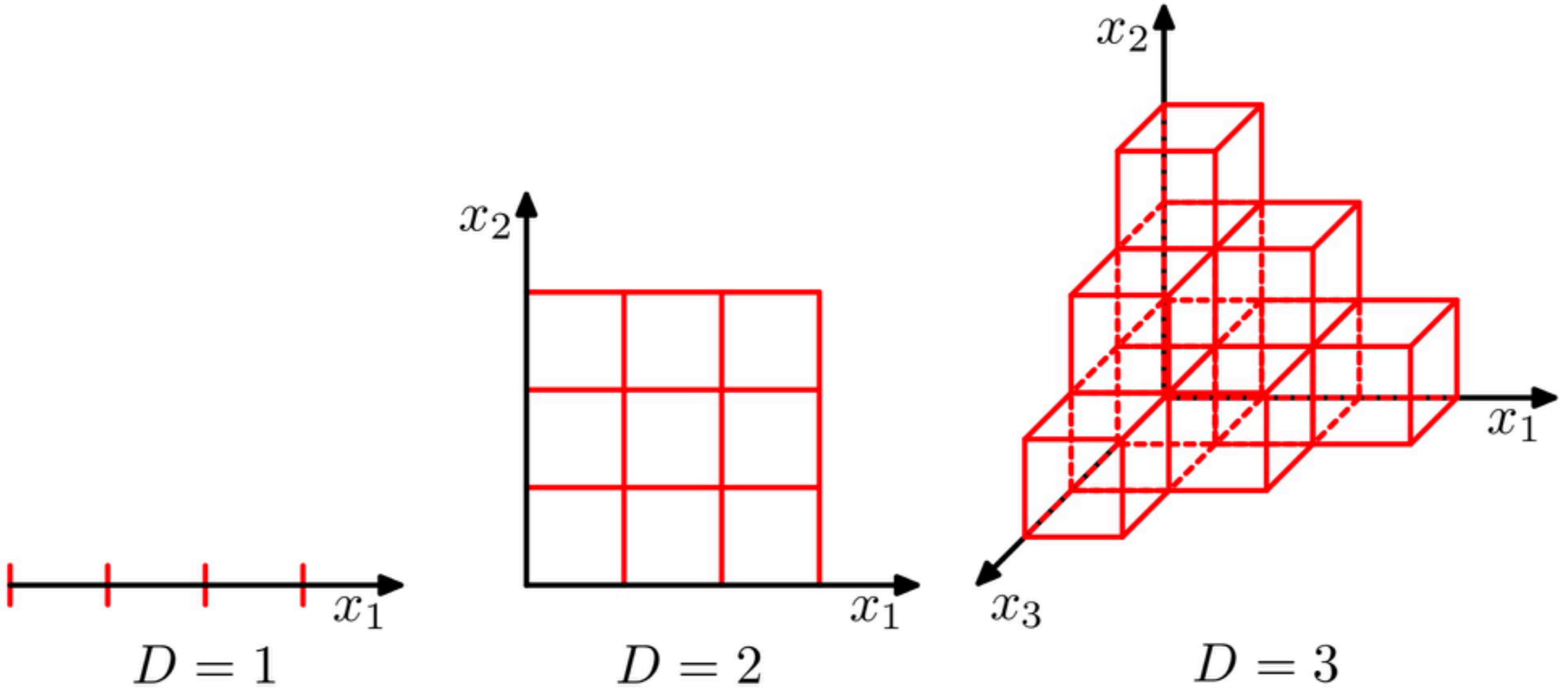
When we measure the wrong features, we'll need very complicated classifiers, and the results are still not ideal.

There's always "exceptions" that would ruin our perfect assumptions

yellow baseball?

we learn the best features from data with deep learning.

The curse of dimensionality

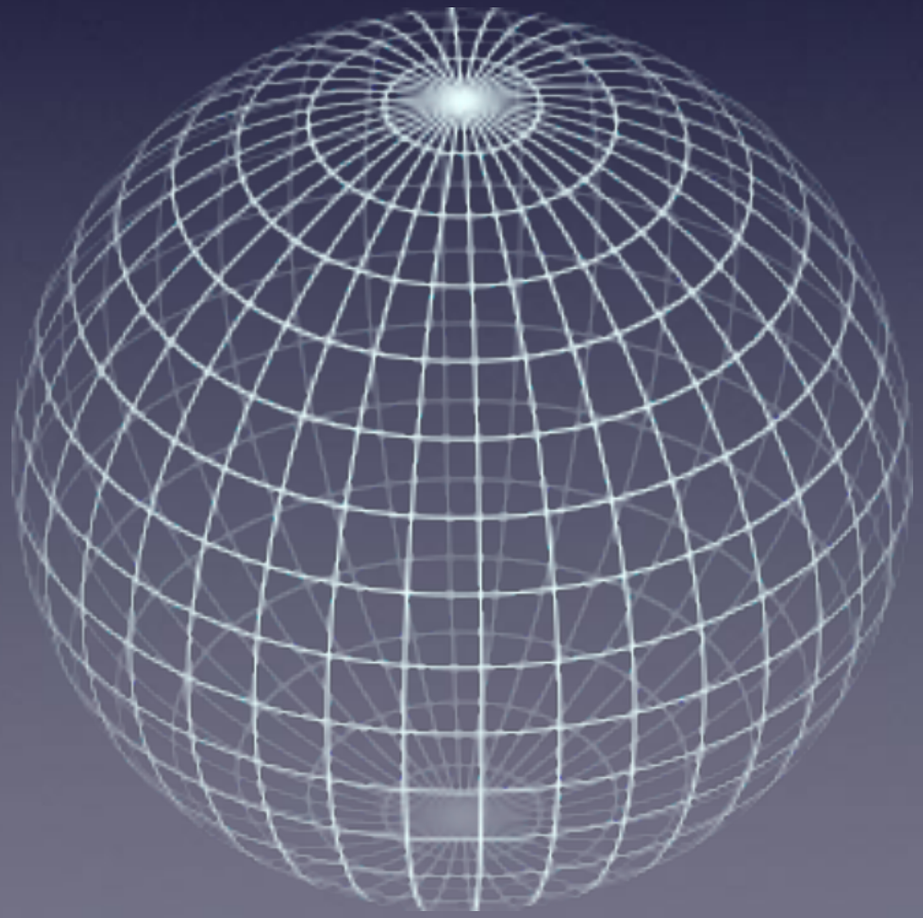


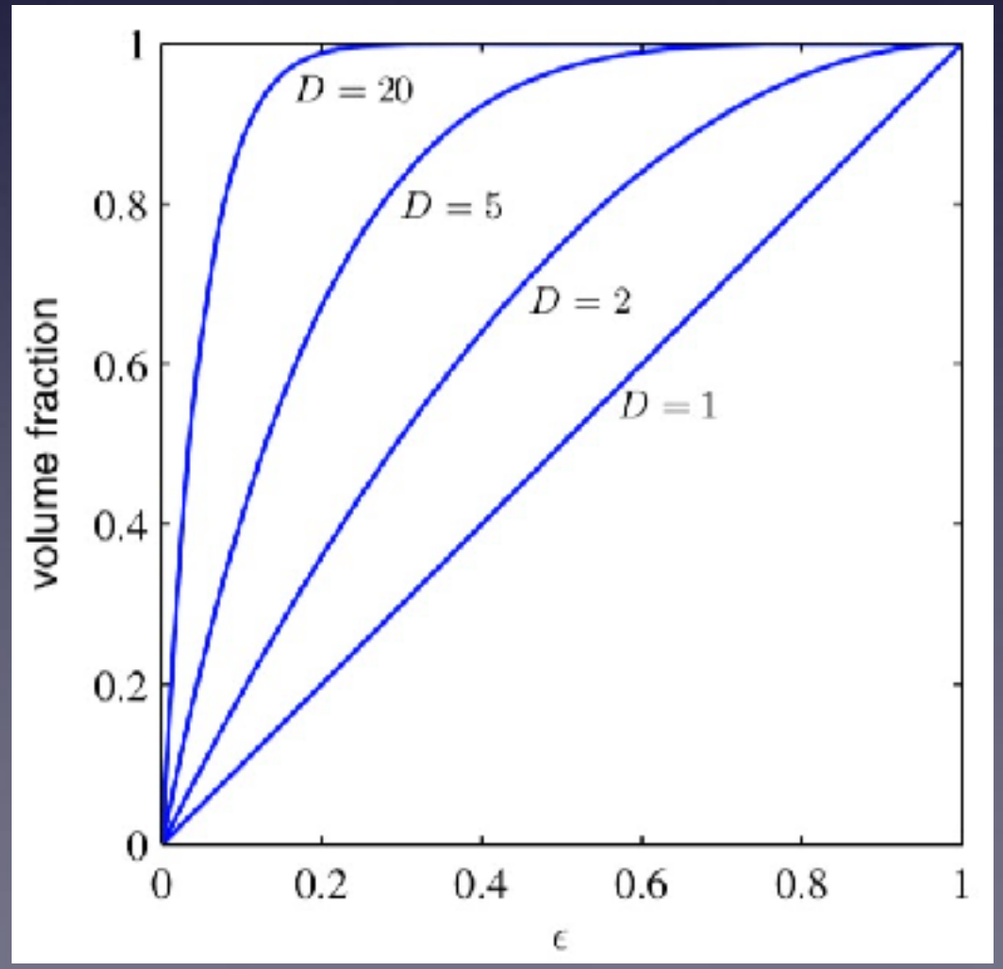
- **More features ≠ better**: number of features*N, feature space grows by ^N, the number of samples needed for ML grows proportionally as well.

The curse of dimensionality

- Most of the volume of an n-D sphere is concentrated in a thin shell near the surface!!!

- nD sphere of $r = 1$, the volume of sphere between $r = 1 - \epsilon$ and $r = 1$ is: $1 - (1 - \epsilon)^D$





High-dim. issue is prevalent

- The curse of dimensionality not just effects the feature space, but also input, output, and others.
  - Much more challenging to train a good n-class classifier, e.g., face recognition, 1-to-1 verification vs 1-to-n identification.
  - Much more issues arise from using a general purpose 1M-class classifier vs problem specific 1k-class classifier.

Prevalent high-dim issue, eg.1

- 1-to-N face identification, in the wild!

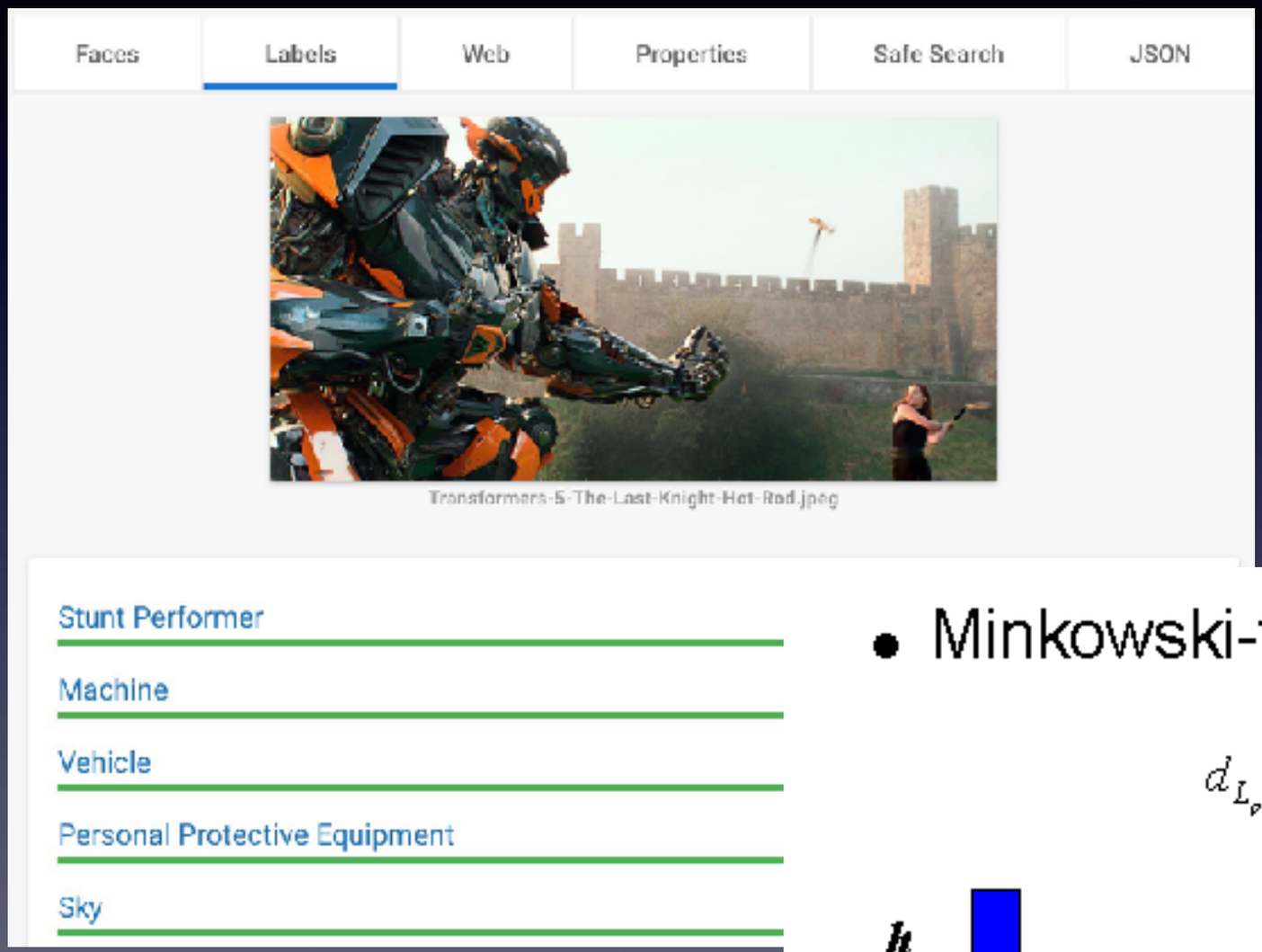
LFW dataset, common FN↑, FP↓

Recognition Accuracy:

- 1 to 1: 99%+
- 1 to 100: 90%
- 1 to 10,000: 50%-70%.
- 1 to 1M: 30%.

Prevalent high-dim issue, eg.2

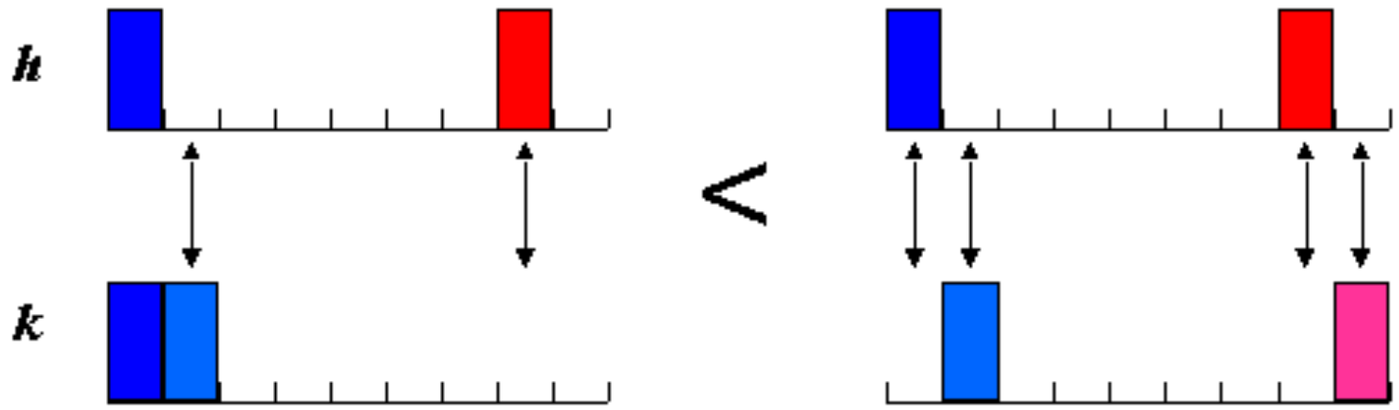
- Smart photo album, with Google Cloud Vision



Distance between histograms of 1M bins is very close to 0 for most of the time.

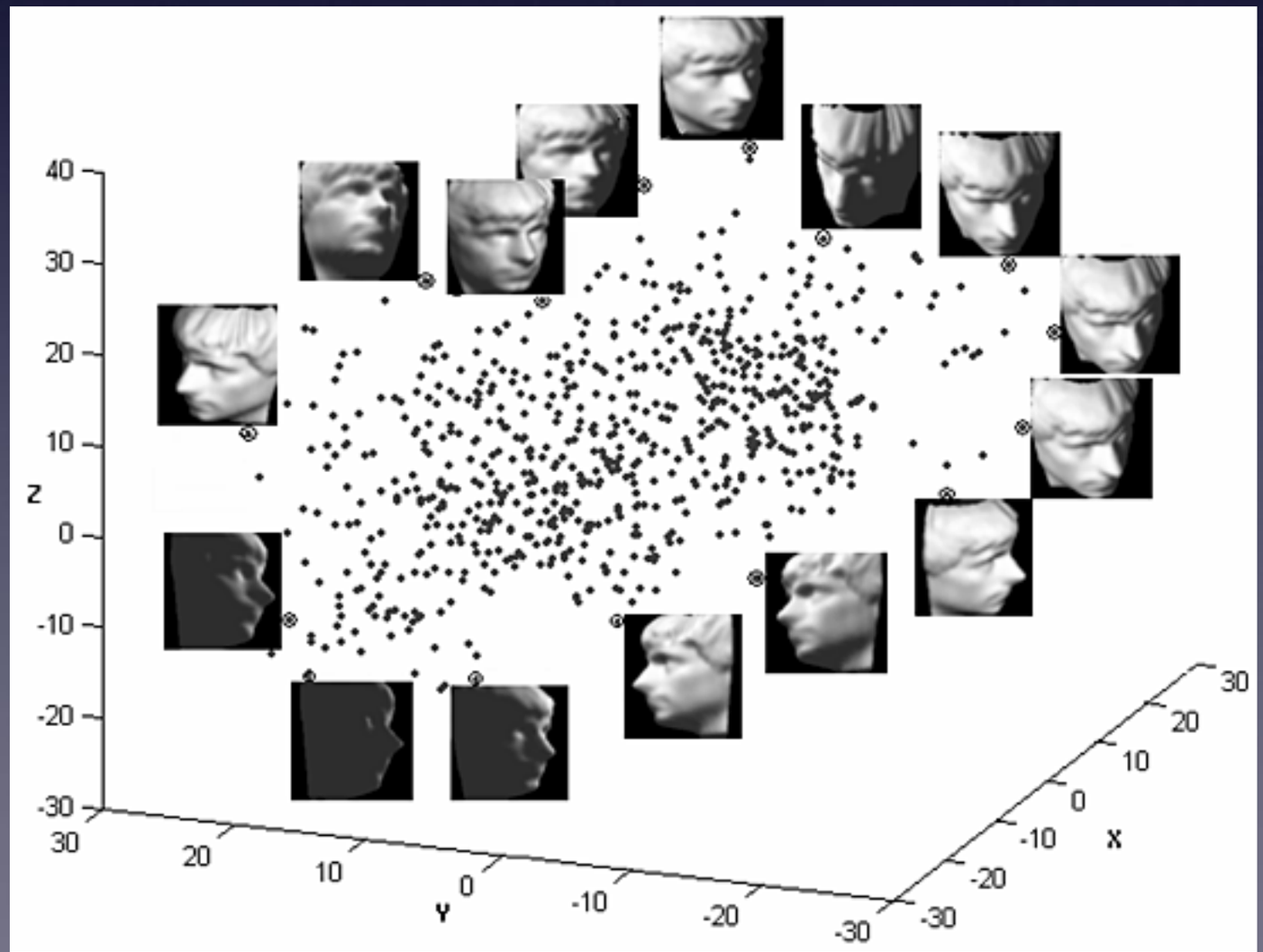
- Minkowski-type distance [e.g. Swain & Ballard 1991]

$$d_{L_p}(H,K) = \left[\sum_{\mathbf{i}} |h_{\mathbf{i}} - k_{\mathbf{i}}|^p\right]^{1/p}$$



Living with high dimensions

- Real data will often be confined to a region of the space having lower effective dimensionality.
- Data will typically exhibit some smoothness properties (at least locally).



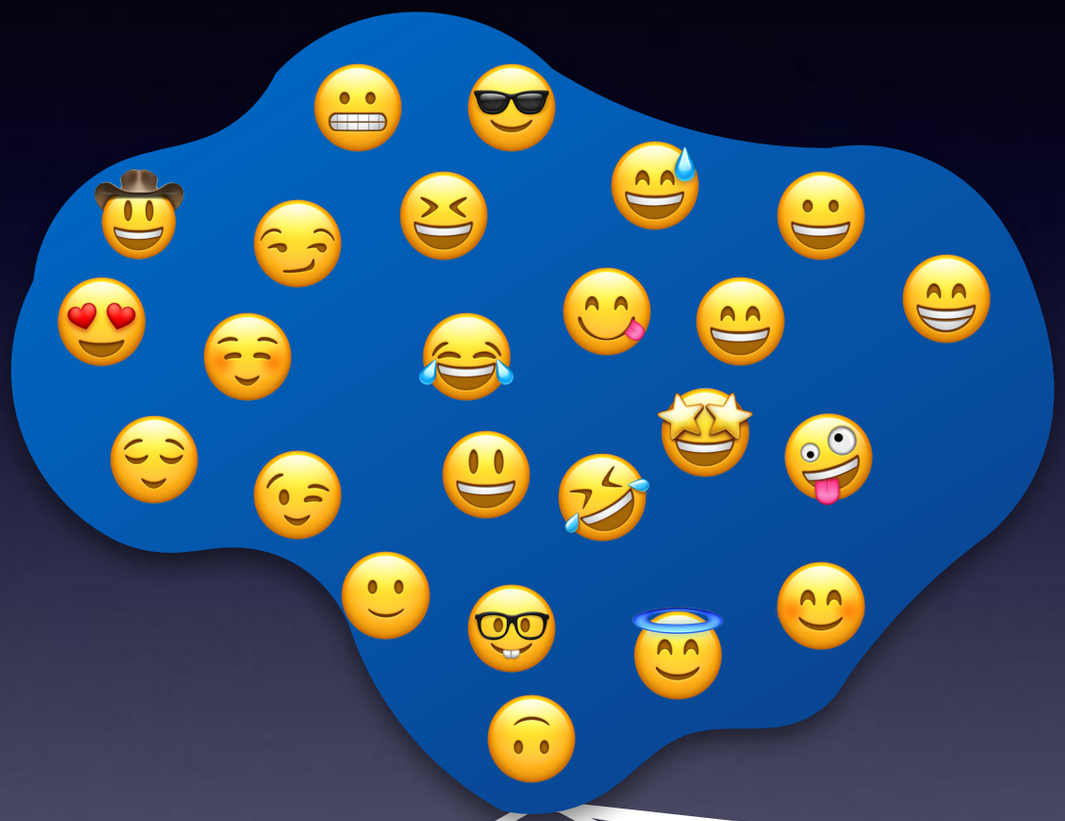
E.g., Low-dimensional "manifold" of faces, embedded within a high-dim space.

Keywords:

- dimension reduction,
- learned features,
- manifold learning.

Splitting data

- k-fold cross validation



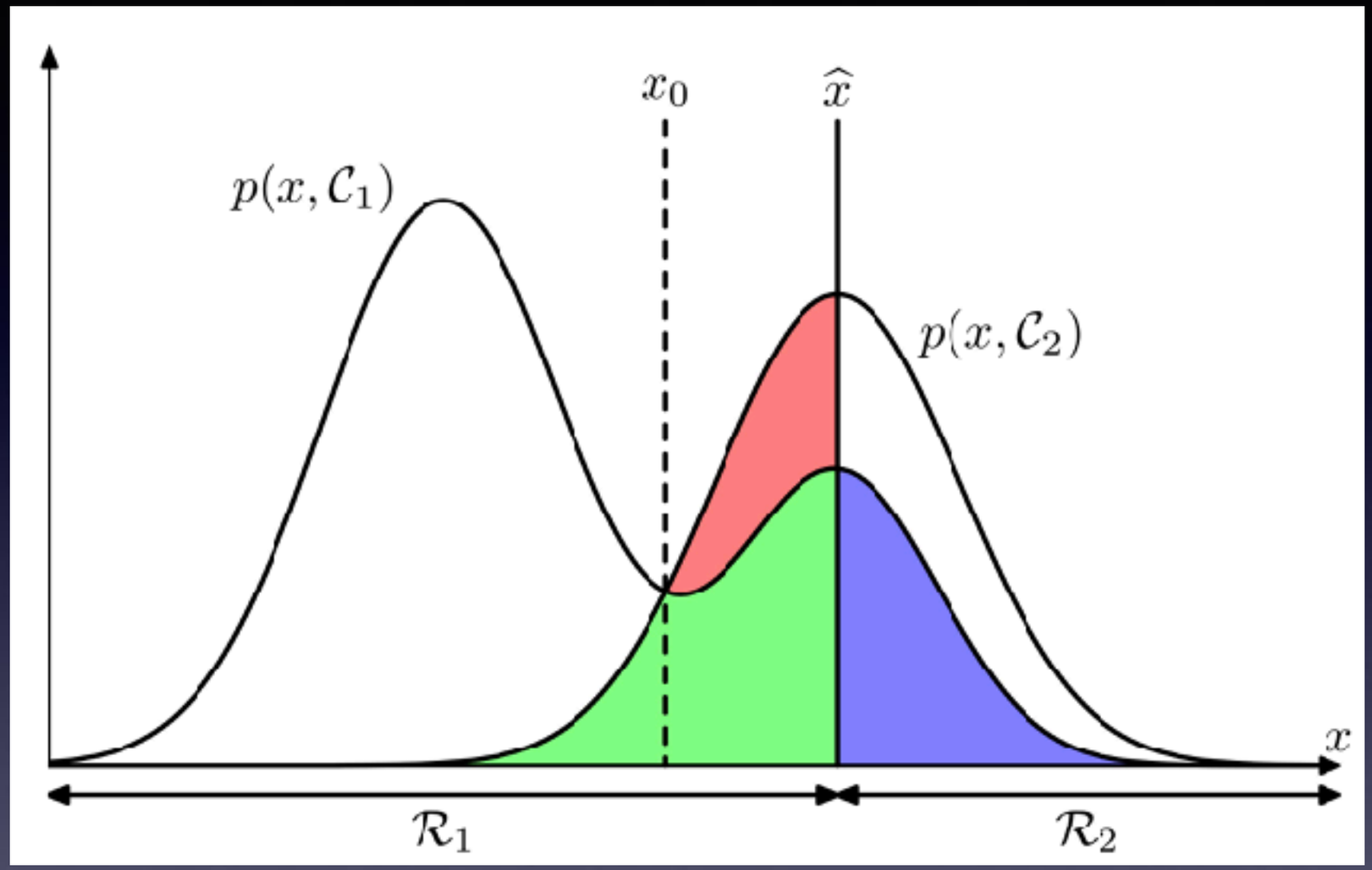
Repurposing the smily faces figures to represent the set of annotated data.

Randomly split into k groups

Decision Theory

- Minimizing the misclassification rate
- Minimizing the expected loss
- The reject option

Decision Boundary



- Decision boundary, or simply, in 1D, a threshold, s.t. anything larger than the threshold are classified as a class, and smaller than the threshold as another class.

True/False, Positive/Negative

- Different metrics & names used in different fields for measuring ML performance; however, the common cornerstones are:
  - **True positive (TP)**: sample is an apple, classified as an apple.
  - **False positive (FP)**: sample is not an apple, but classified as an apple.
  - **True negative (TN)**: sample is not an apple, classified as not an apple.
  - **False negative (FN)**: sample is an apple, but misclassified as "not an apple.

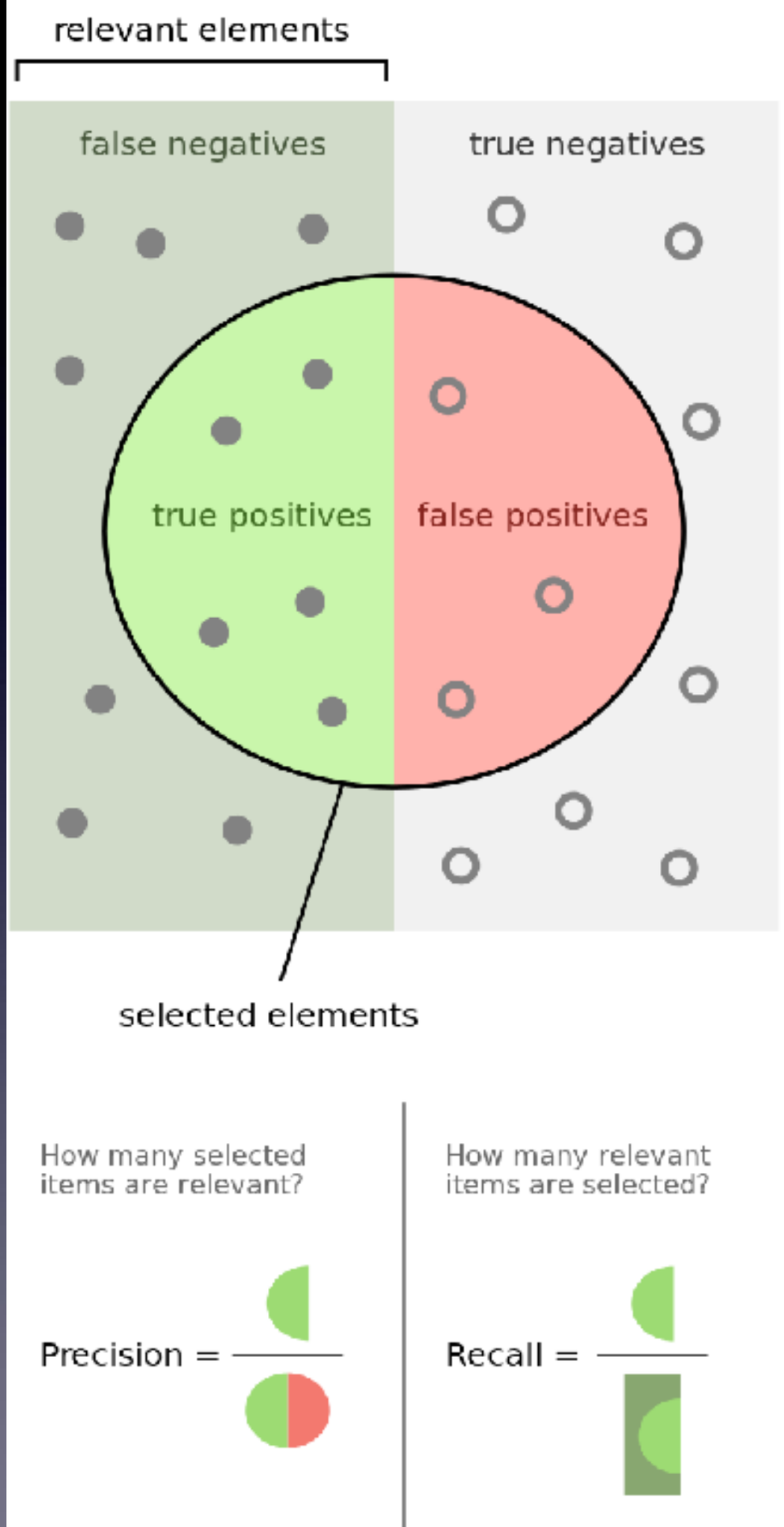
Precision vs Recall

- **Precision**: $\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$

  Classifier identified (TP+FP) apples, only TP are apples. (aka positive predictive value.)

- **Recall**: $\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$

  Total (TP+FN) apples, classifier identified TP. (aka, hit rate, sensitivity, true positive rate)



A single balanced metric?

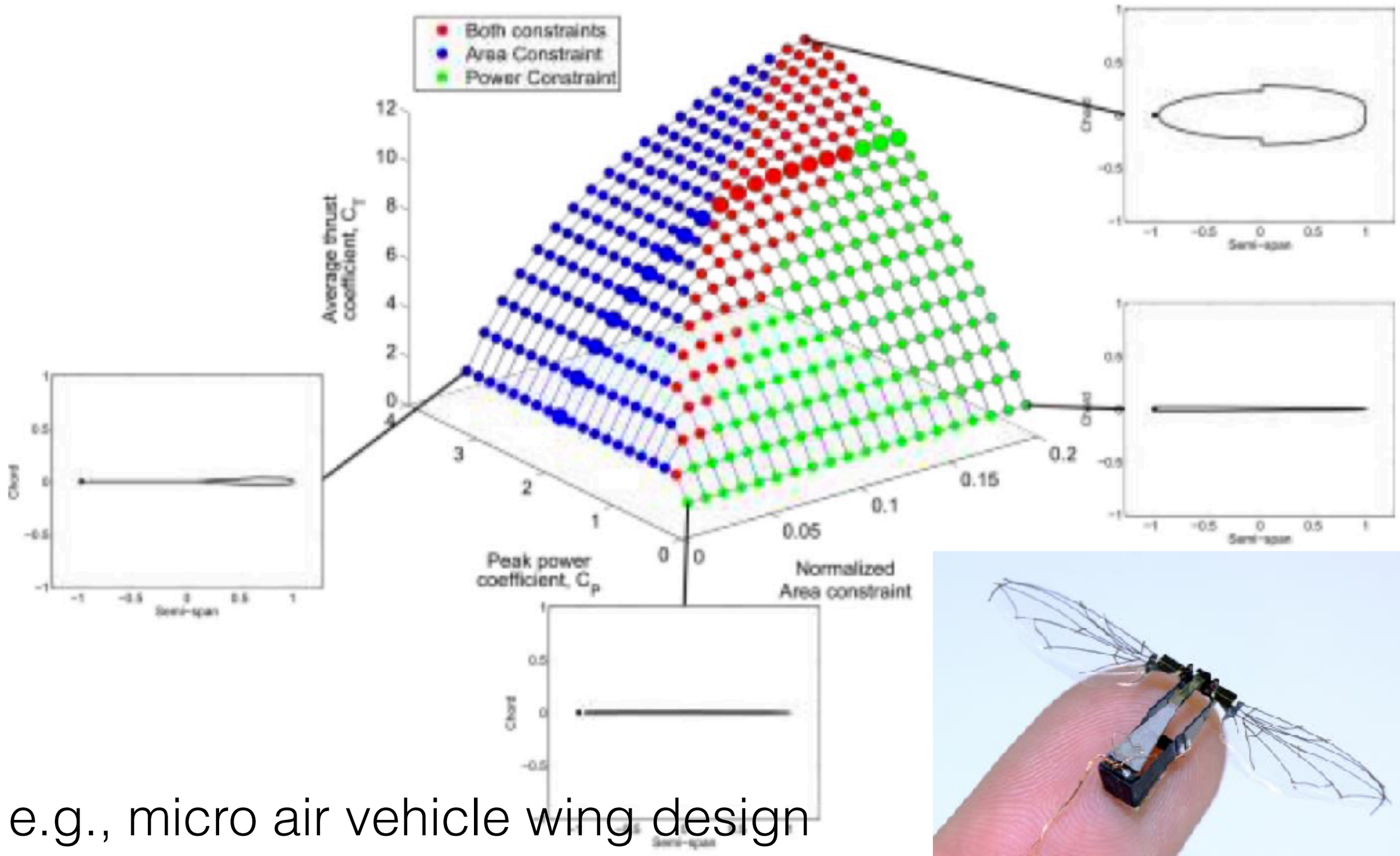
- **F-measure**: $2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$

  harmonic mean of precision and recall. F-measure is criticized outside Information Retrieval field for neglecting the true negative.

- **Accuracy** (ACC): $\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$

  a weighted arithmetic mean of precision and inverse precision, as well as the weighted arithmetic mean of recall and inverse recall.

Multi-objective Optimization



e.g., micro air vehicle wing design

Minimizing the expected loss

- Different types of errors are weighted differently; e.g., medical examinations, minimize false negative but can tolerate false positive.

- Reformulate objectives from maximizing probability to minimizing weighted loss functions.

- The reject option: refrain from making decisions on difficult cases (e.g., for samples within a certain region inside the decision boundary.)

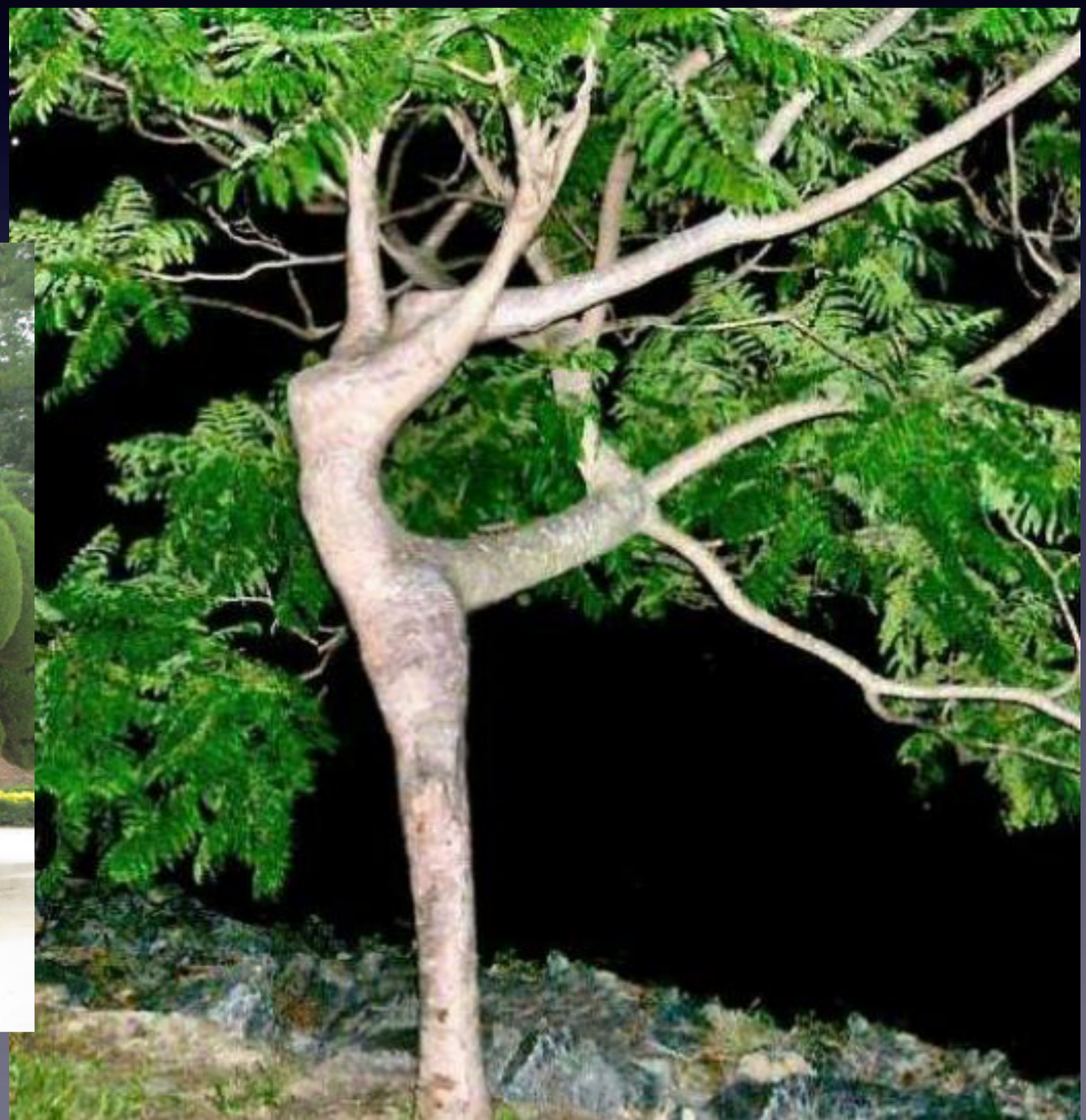
Generalization

- Minimizing Training and Validation Error, v.s. minimizing Testing Error.
- Memorizing every “practice exam” question ≠ doing well on new questions. Avoid overfitting.

E.g., training a classifier that recognizes trees

Odd trees of the world

Odd trees of the world

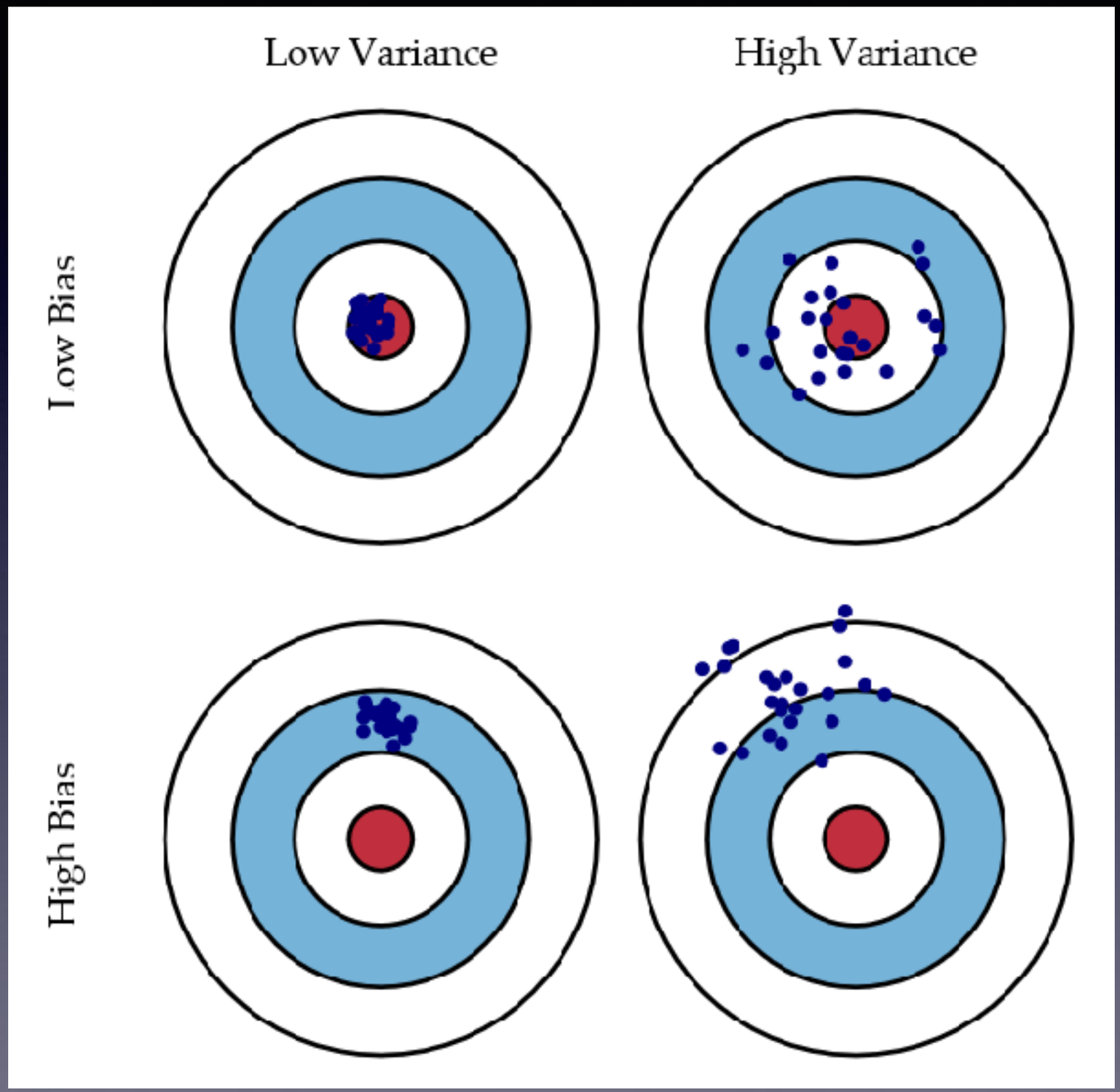


Odd trees of the world

Generalization Error

- Bias:
  - Difference between the expected (or averaged) prediction of our model and the correct value.
  - Error due to inaccurate assumptions/ simplifications.
- Variance:
  - Amount that the estimate of the target function will change if different training data was used.

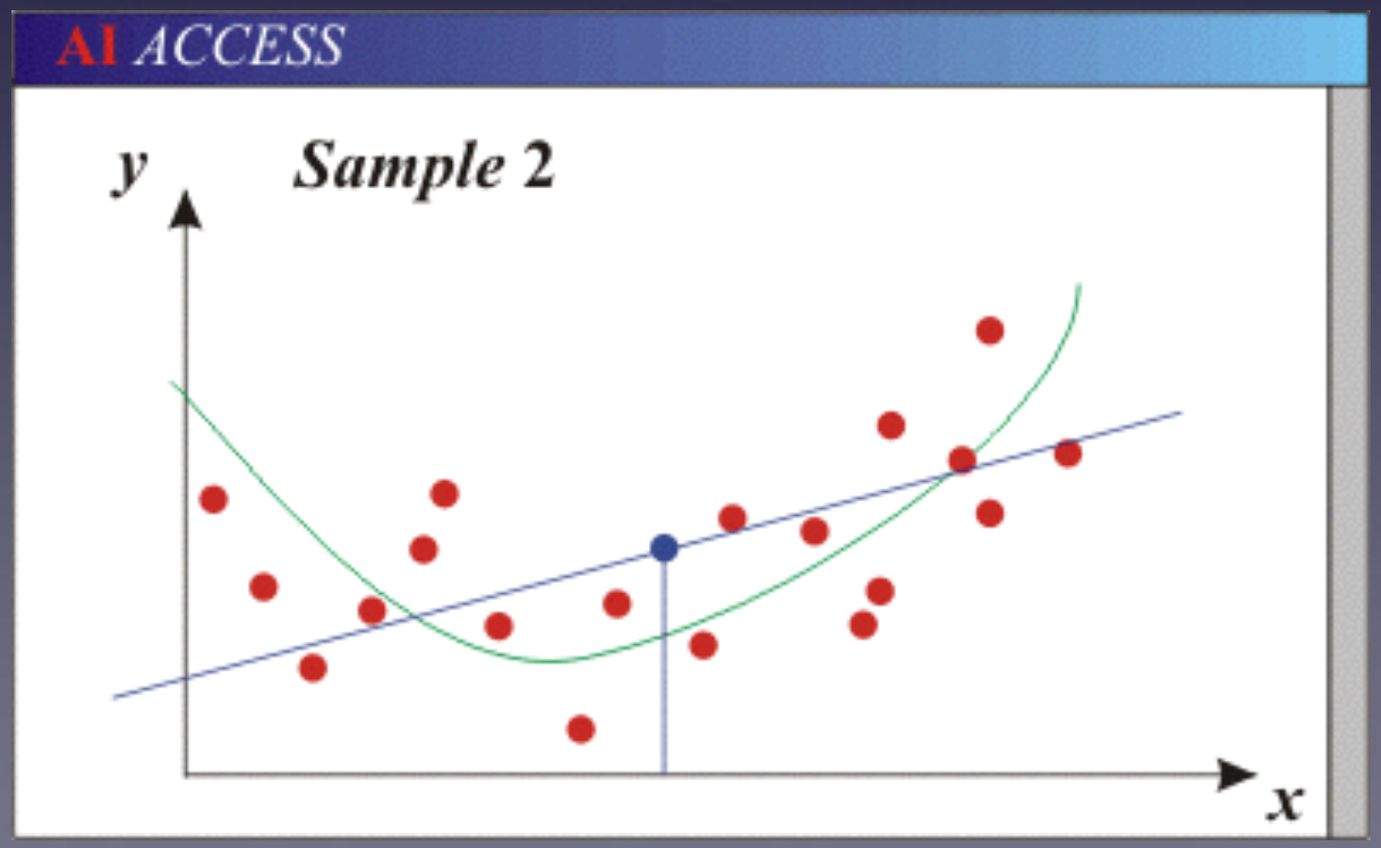
Bias/variance trade-off



Scott Fortmann-Roe

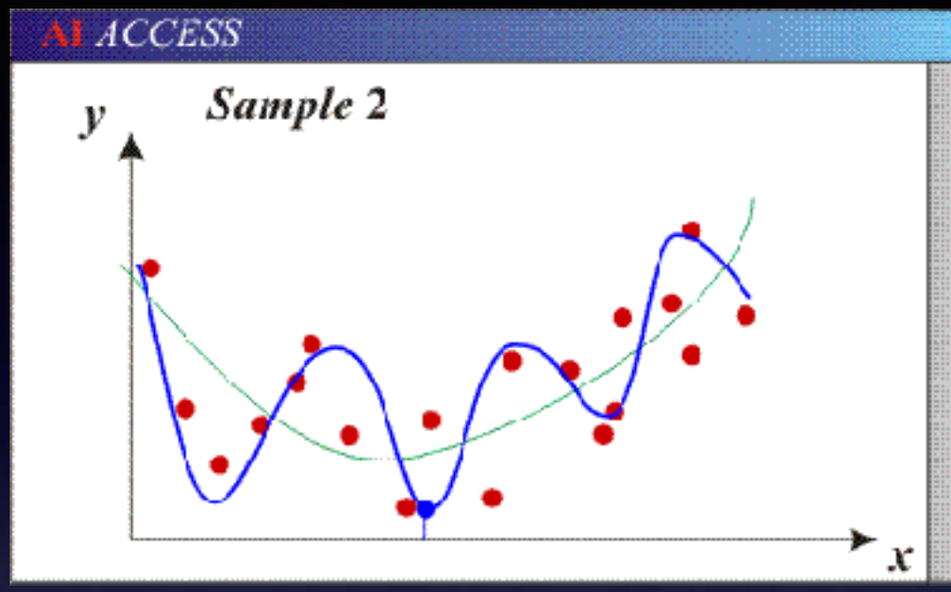
Underfitting

- Model is too simple to represent all the relevant class characteristics.
  - High bias (few degrees of freedom, DoF) and low variance.
  - High training error and high test error.

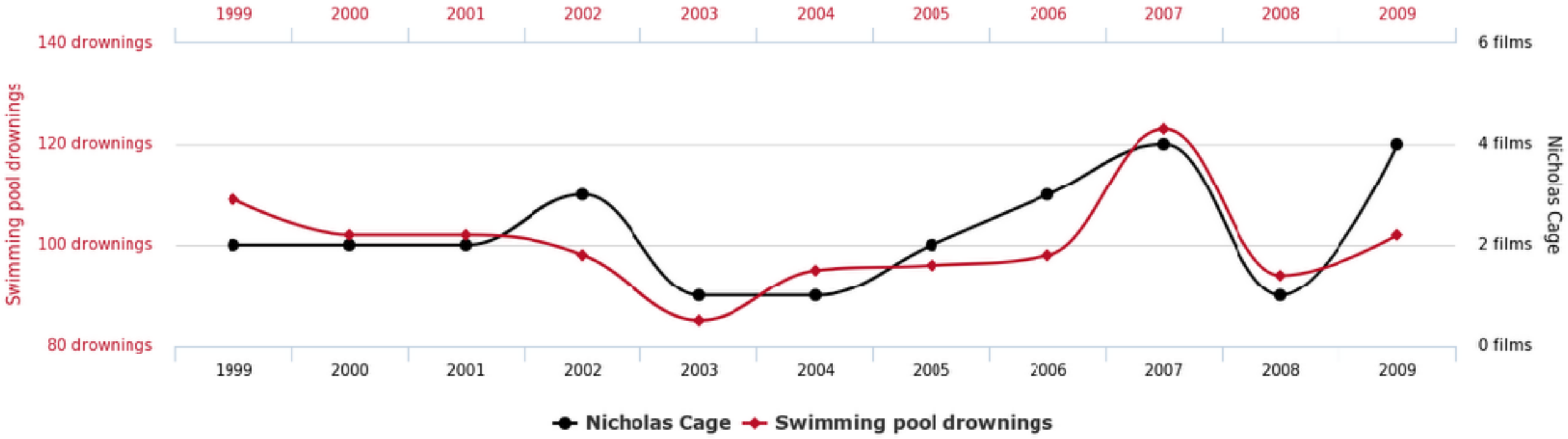


Overfitting

- Model is too complex and fits irrelevant noise in the data
  - Low bias, high variance
  - Low training error, high test error



Number of people who drowned by falling into a pool
correlates with
Films Nicolas Cage appeared in



Bias-Variance Trade-off

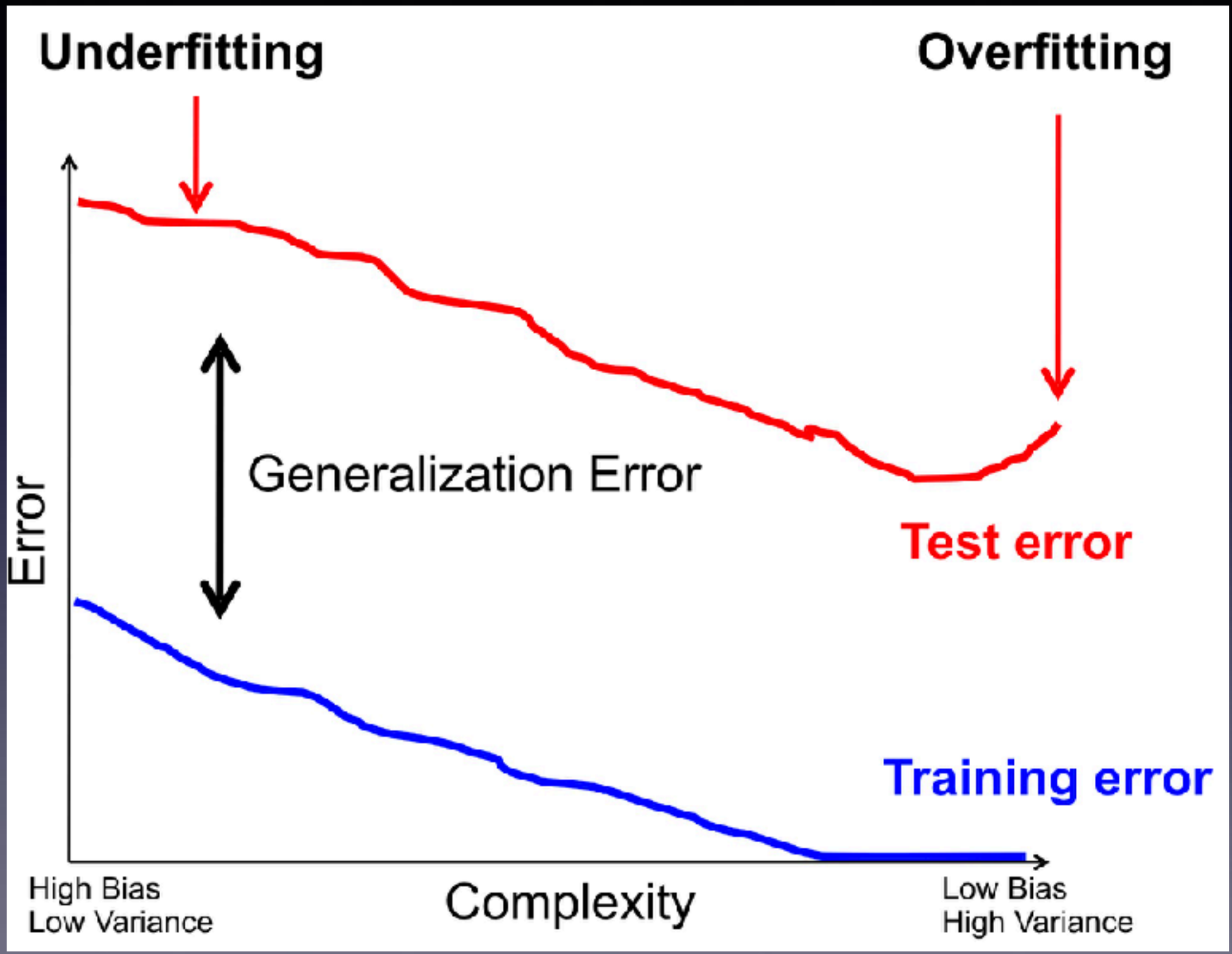
Error (mean square error, MSE)
= $\text{noise}^2 + \text{bias}^2 + \text{variance}$

unavoidable error

error due to incorrect assumptions made about the data

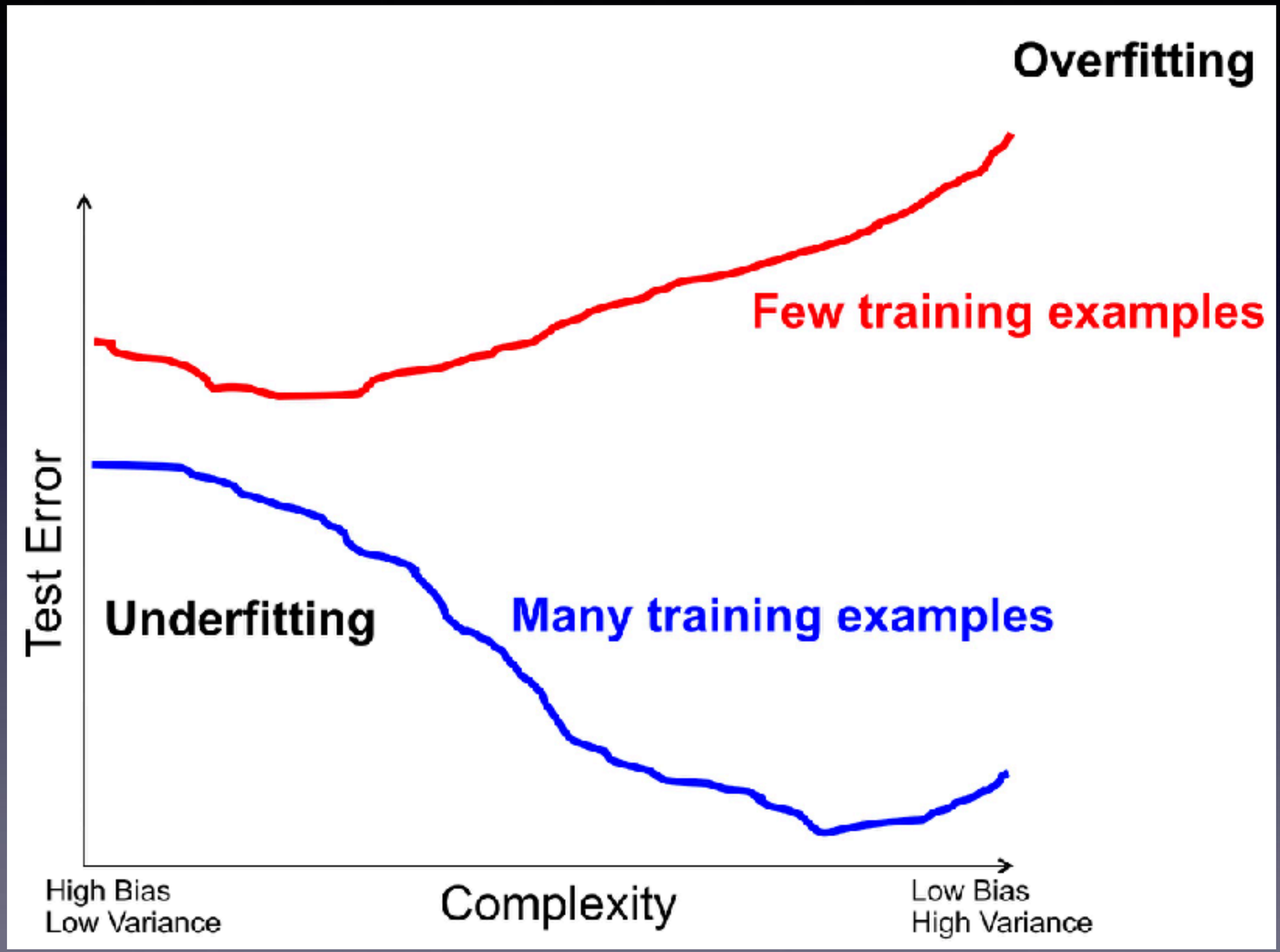
error due to variance of training samples

Model Complexity



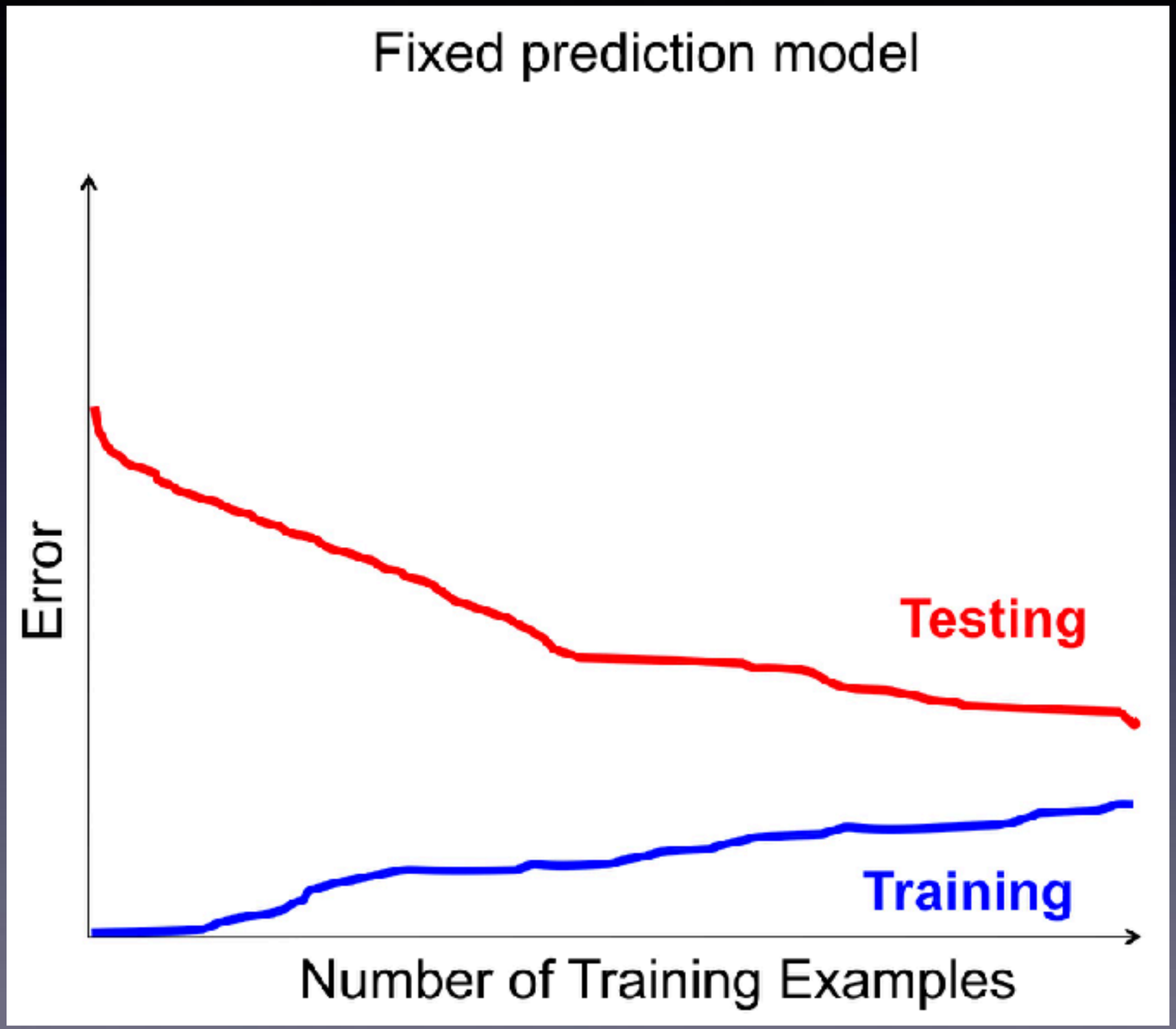
Slide credit: D. Hoiem

Training Sample vs Model Complexity



Slide credit: D. Hoiem

Effect of Training Sample Size



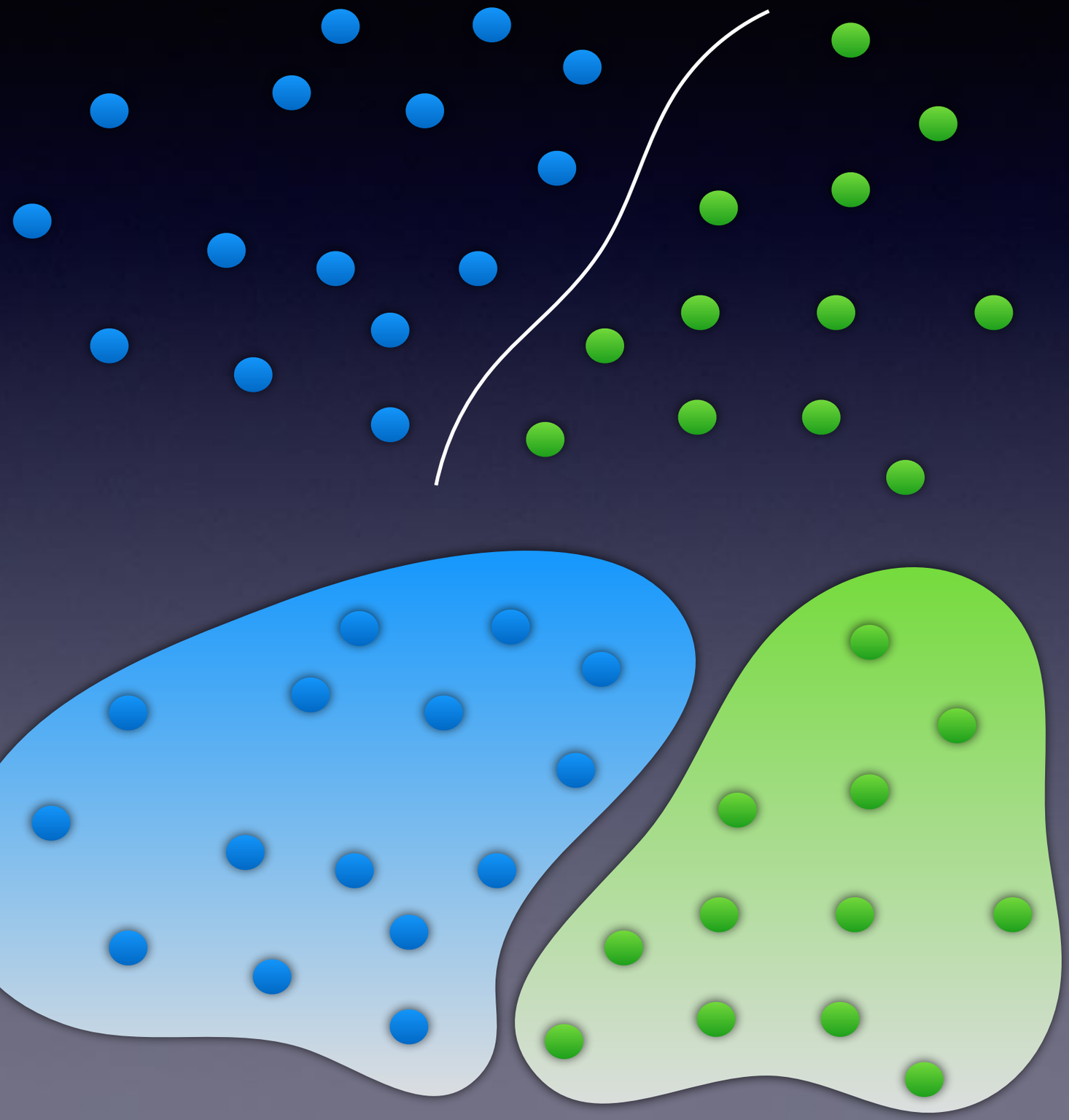
Slide credit: D. Hoiem

How do we learn models?

- Models: describe relationship between variables
  - **Deterministic models**: hypothesize exact relationships, OK when noise is negligible
  - **Probabilistic models**: deterministic part + random error. For example:
    - **Regression models**: one dependent variable + one or more *numerical or categorical* independent (explanatory) variable.
    - **Correlation models**: multiple independent variables.

Generative vs Discriminative Models

Discriminative Model: directly learn the data boundary



Generative Model: represent the data and boundary

Discriminative Models

- Learn to directly predict labels from the data
- Often uses simpler boundaries (e.g., linear) for hopes of better generalization.
- Often easier to predict a label from the data than to model the data.

- E.g.,
  - Logistic Regression
  - Support Vector Machines
  - Max Entropy Markov Model
  - Conditional Random Fields

Generative Models

- Represent both the data and the boundary.
- Often use conditional independence and priors.
- Modeling data is challenging; need to make and verify assumptions about data distribution
- Modeling data aids prediction & generalization.

- E.g.,
  - Naive Bayes
  - Gaussian Mixture Model (GMM)
  - Hidden Markov Model
  - Generative Adversarial Networks (GAN)

Distributions

- Bernoulli Distribution
- Uniform Distribution
- Binomial Distribution
- Normal Distribution
- Poisson Distribution
- Exponential Distribution

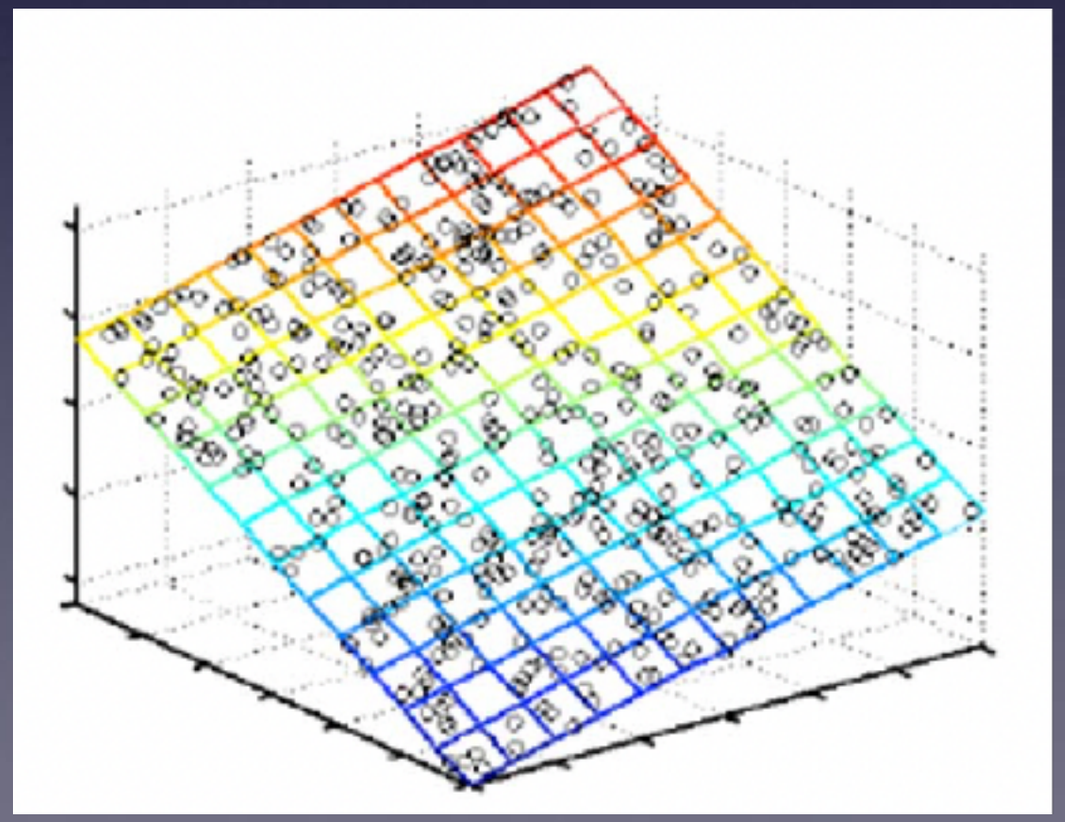
Dimension Reduction

Machine Learning Roadmap

supervised / unsupervised	continuous (predicting a quantity)	discrete (predicting a category)
unsupervised	**Dimension Reduction**	**Clustering**
supervised	**Regression**	**Classification**

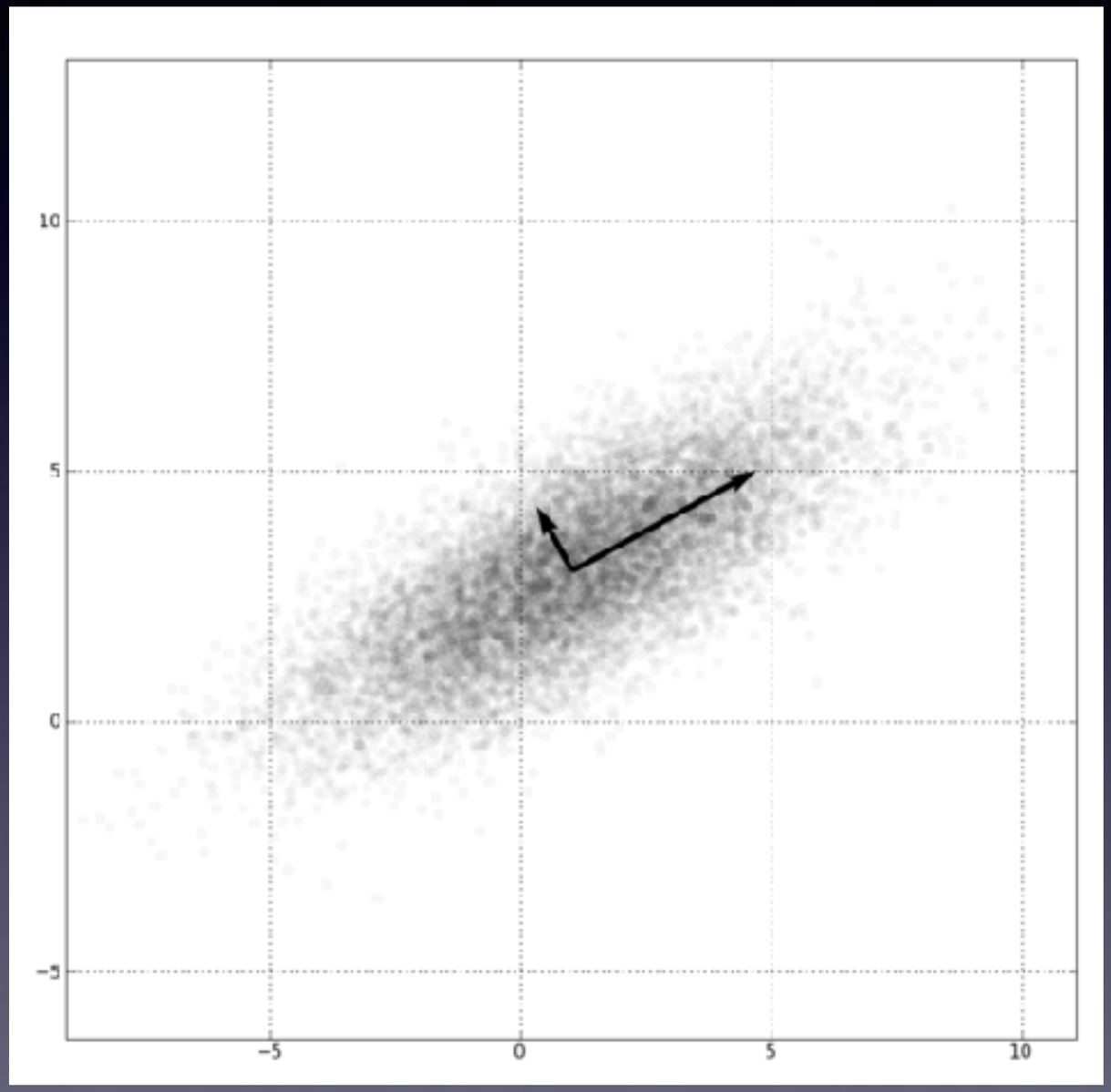
Unsupervised Dimension Reduction

- Goal: try to find a more compact representation of the data
  - Assume that the high dimensional data actually reside in an inherent low-dimensional space.
  - Additional dimensions are just random noise
  - Goal is to recover these inherent dimensions and discard noise.

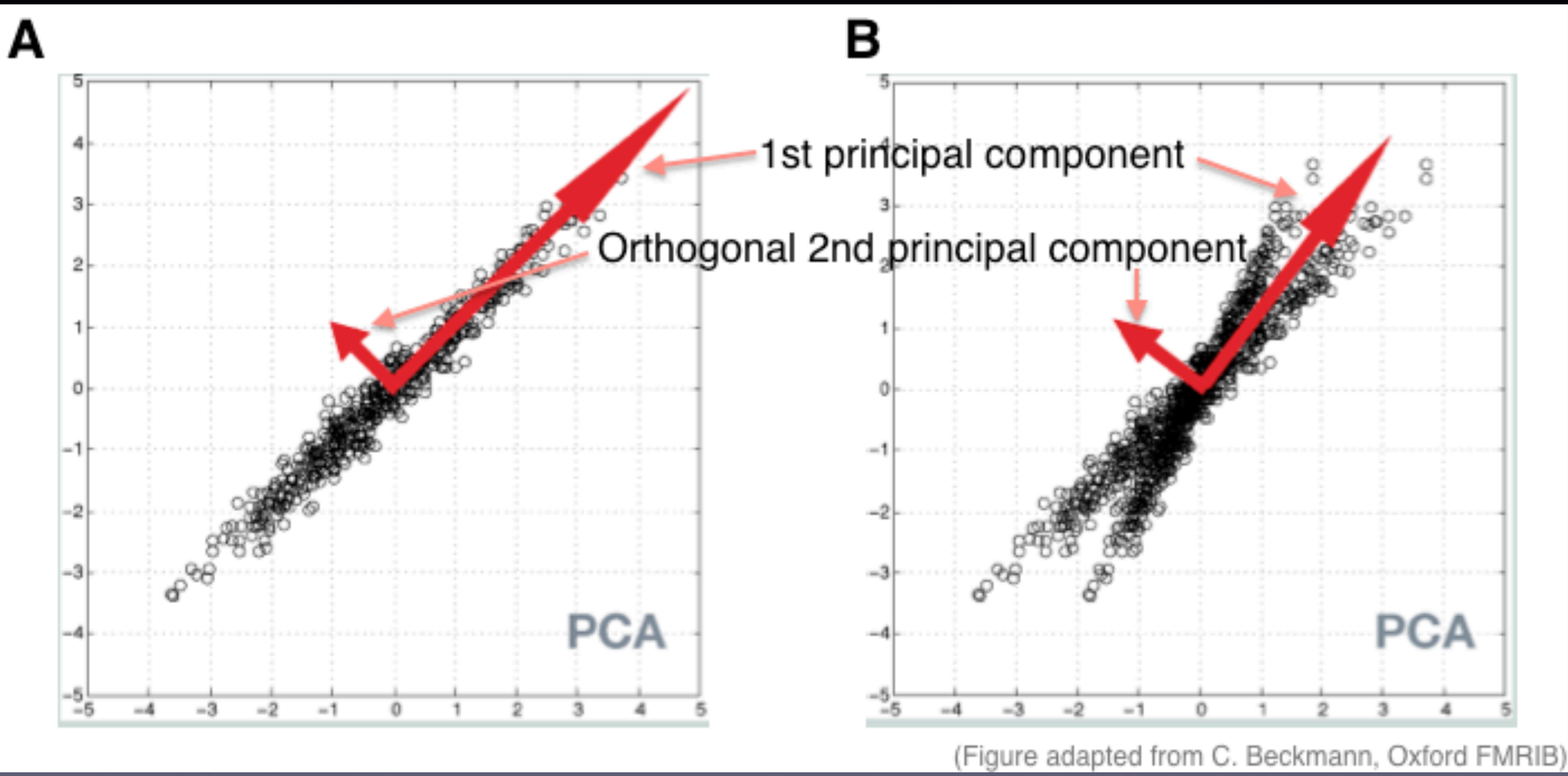


Principal Component Analysis (PCA)

- Create a basis where the axes represent the dimensions of variance, from high to low.
- Finds correlations in data dimensions to product best possible lower-dimensional representation based on linear projections.



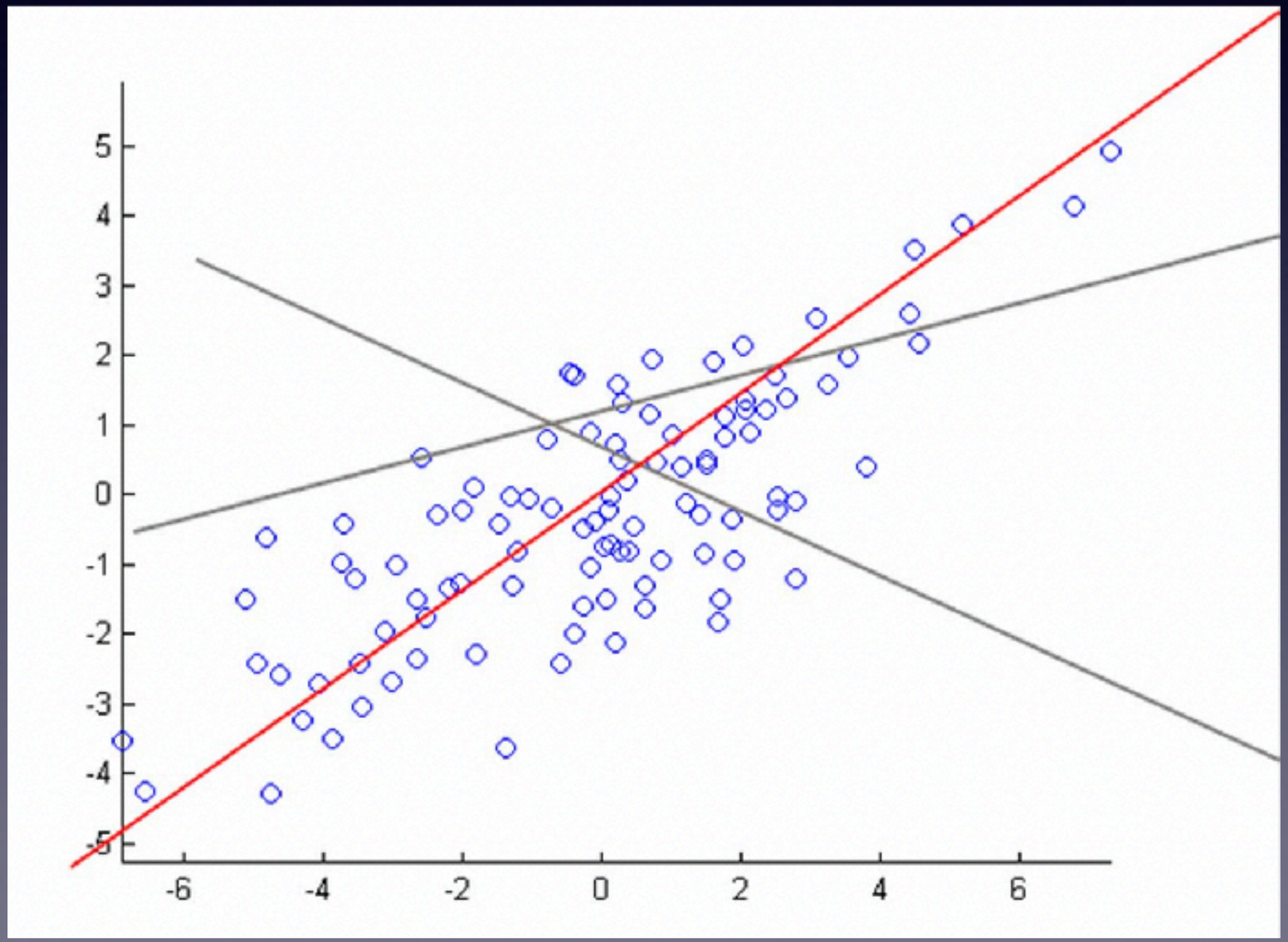
PCA



(Figure adapted from C. Beckmann, Oxford FMRIB)

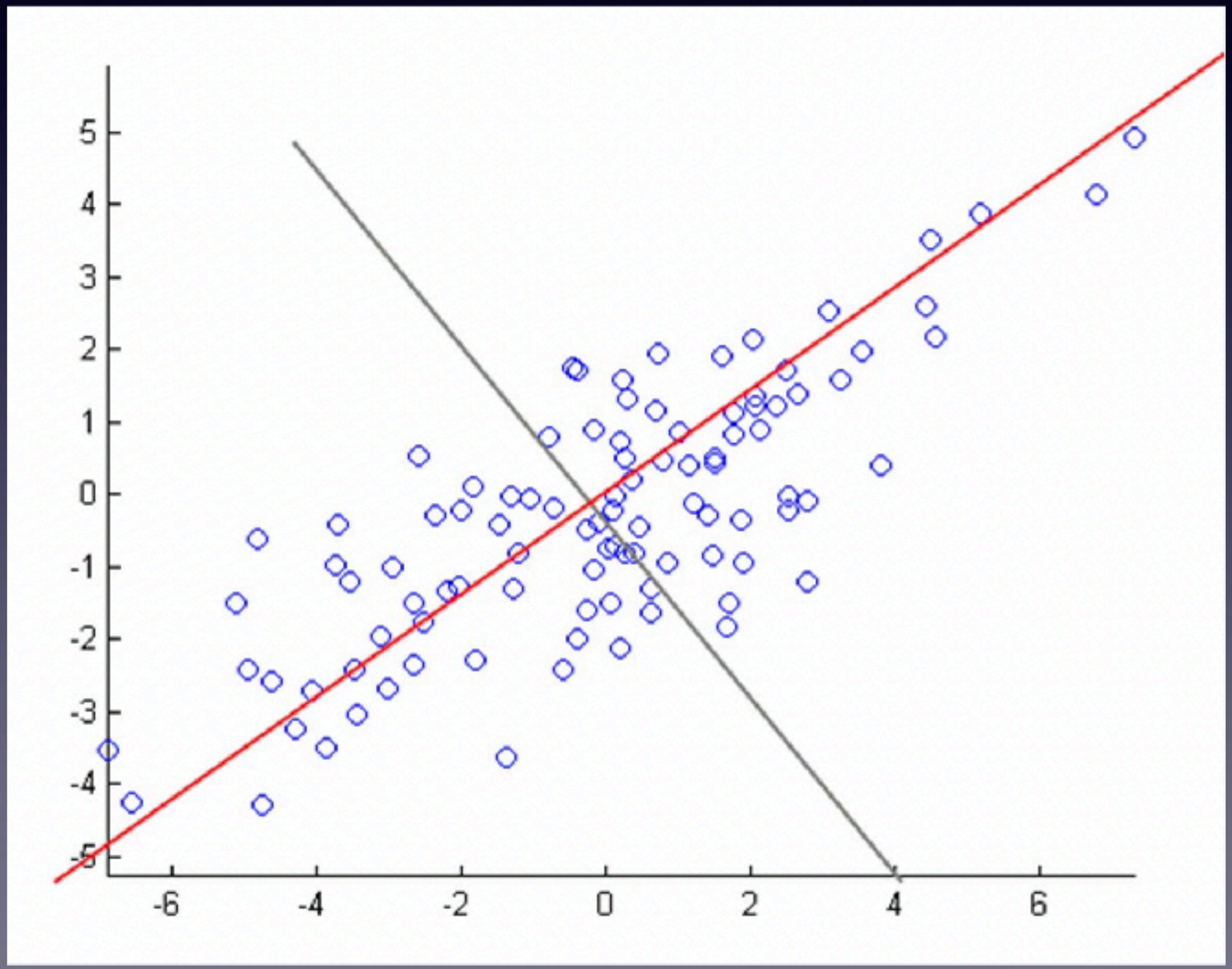
PCA algorithm, conceptual steps

- Find a line s.t. when data is projected onto the line, it has the maximum variance.



PCA algorithm, conceptual steps

- Find new line orthogonal to the first that has the maximum projected variance.



PCA algorithm, conceptual steps

- Repeated until d lines. The projected position of a point on these lines gives the coordinates in the m-dimensional reduced space.

- Computing these set of lines is achieved by eigen-decomposition of the covariance matrix.

PCA, maximizing variance

- Given n data points: x_1, ..., x_n
- Consider a linear projection specified by v
- The projection of x onto v is $z = v^{\mathrm{T}} x$
- The variance of the projected data is

$$\mathrm{var}(z) = \mathrm{var}(v^{\mathrm{T}} x v) = v^{\mathrm{T}} \mathrm{var}(x) v = v^{\mathrm{T}} S v$$

- The 1st Principal Component maximizes the variance subject to the constraint

PCA, maximizing variance

- Maximize $v^{\mathrm{T}}Sv$, subject to $v^{\mathrm{T}}v = 1$
- Lagrange: $v^{\mathrm{T}}Sv - \lambda(v^{\mathrm{T}}v - 1)$

$$\frac{d}{dv} = 0 \rightarrow Sv = \lambda v$$

- v is the eigen-vector of *S* with eigen-value λ

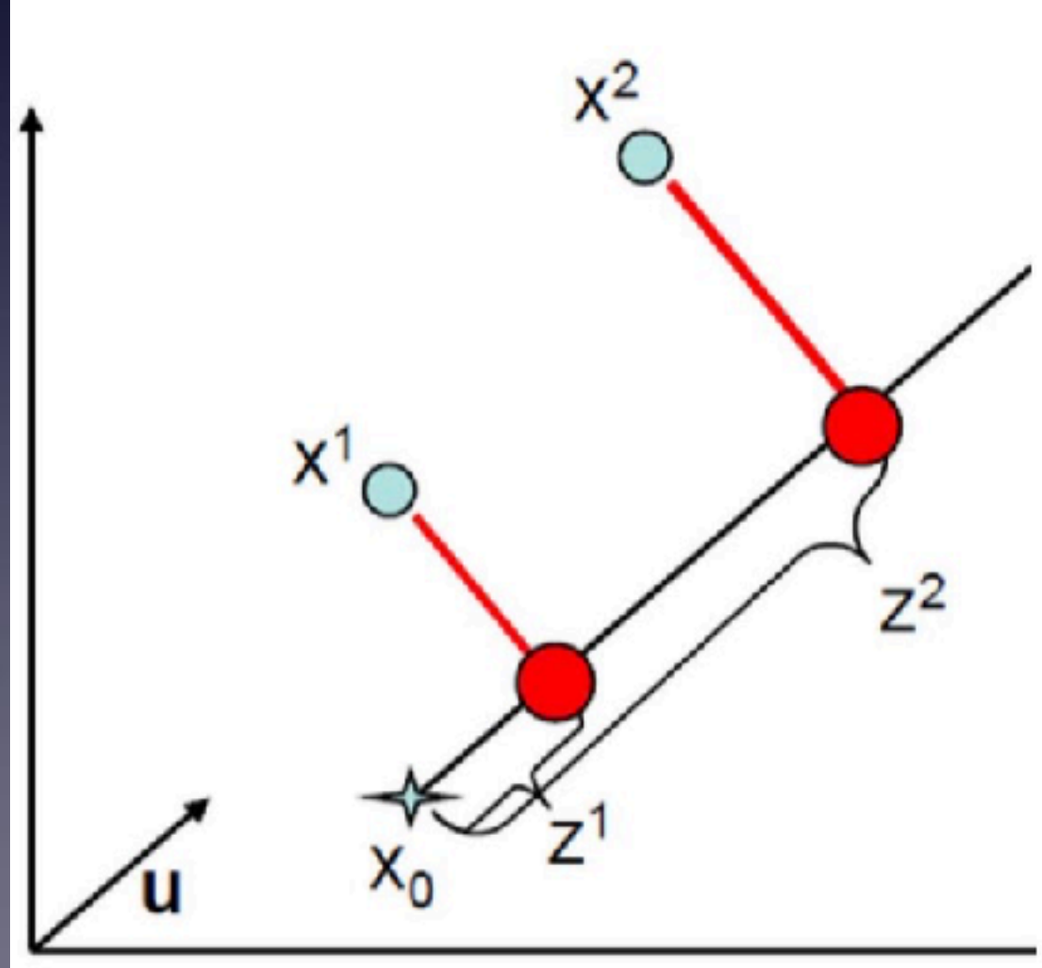
- Sample variance of the projected data

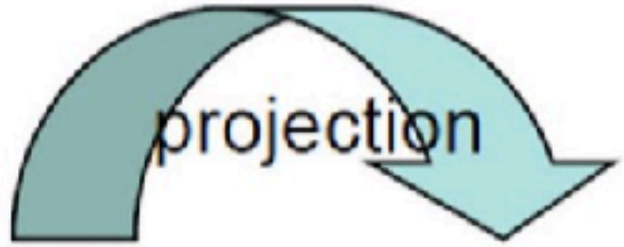
$$v^{\mathrm{T}}Sv = \lambda v^{\mathrm{T}}v = \lambda$$

- The eigen-values equals the amount of variance captured by each eigen-vector

Alternative view of PCA

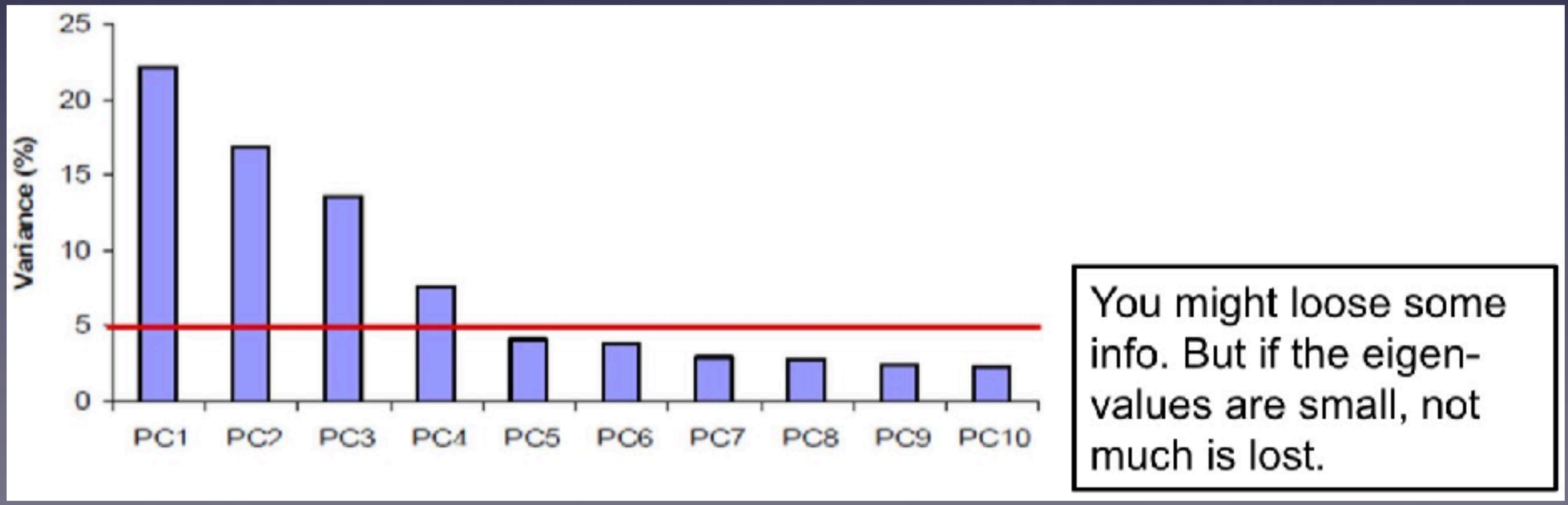
- View PCA as minimizing the reconstruction error of using a low-dimensional approximation of the original data: $x^1 \approx x_0 + z^1 u \quad x^2 \approx x_0 + z^2 u$





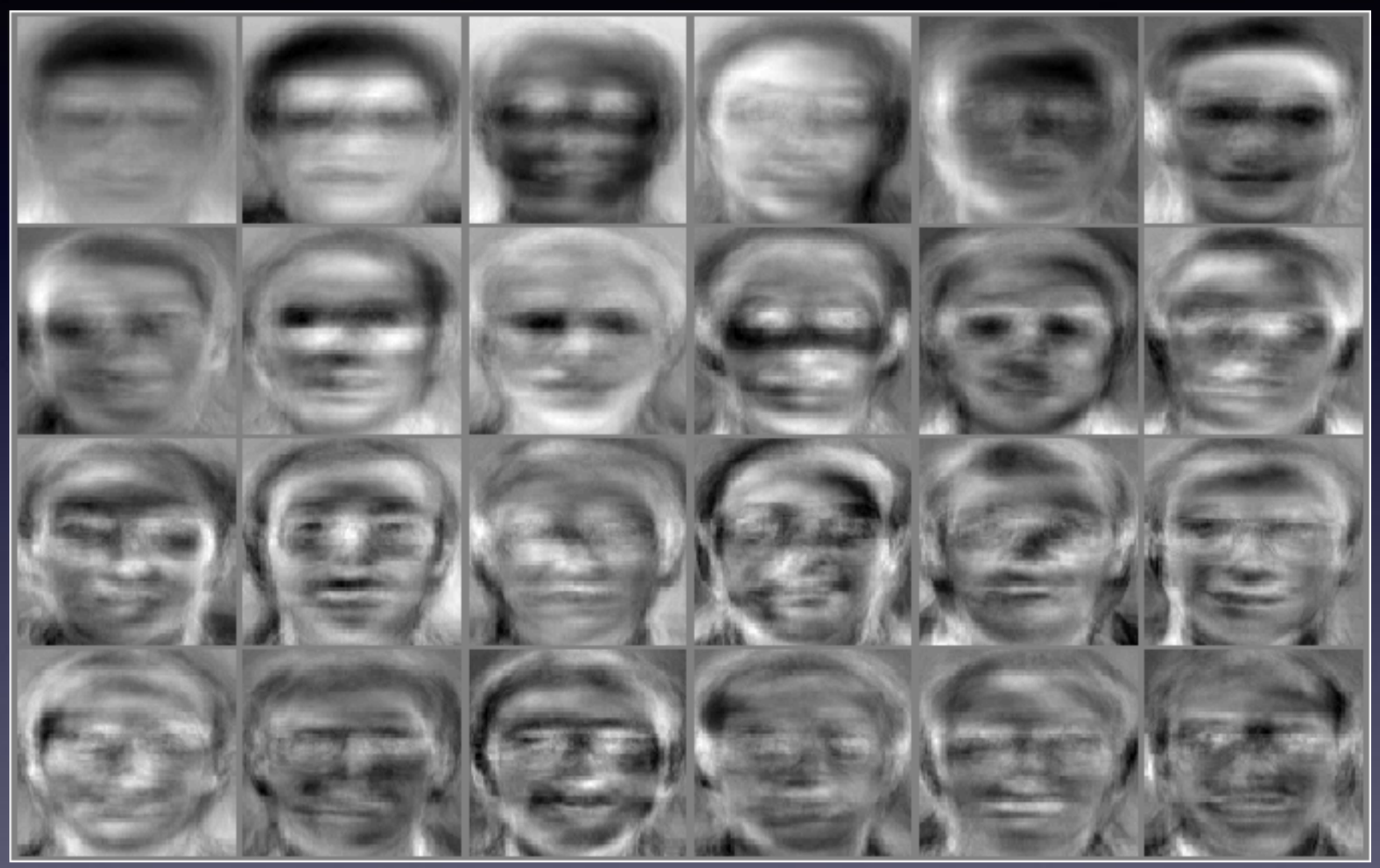
Dimension Reduction using PCA

- Calculate the covariance matrix of the data S
- Calculate the eigen-vectors/eigen-values of S
- Rank the eigen-values in decreasing order
- Select eigen-vectors that retain a fixed % of the variance, e.g., 80%, s.t., $\frac{\sum_{i=1}^{d} \lambda_i}{\sum_i \lambda_i} \geq 80\%$



PCA example: Eigenfaces

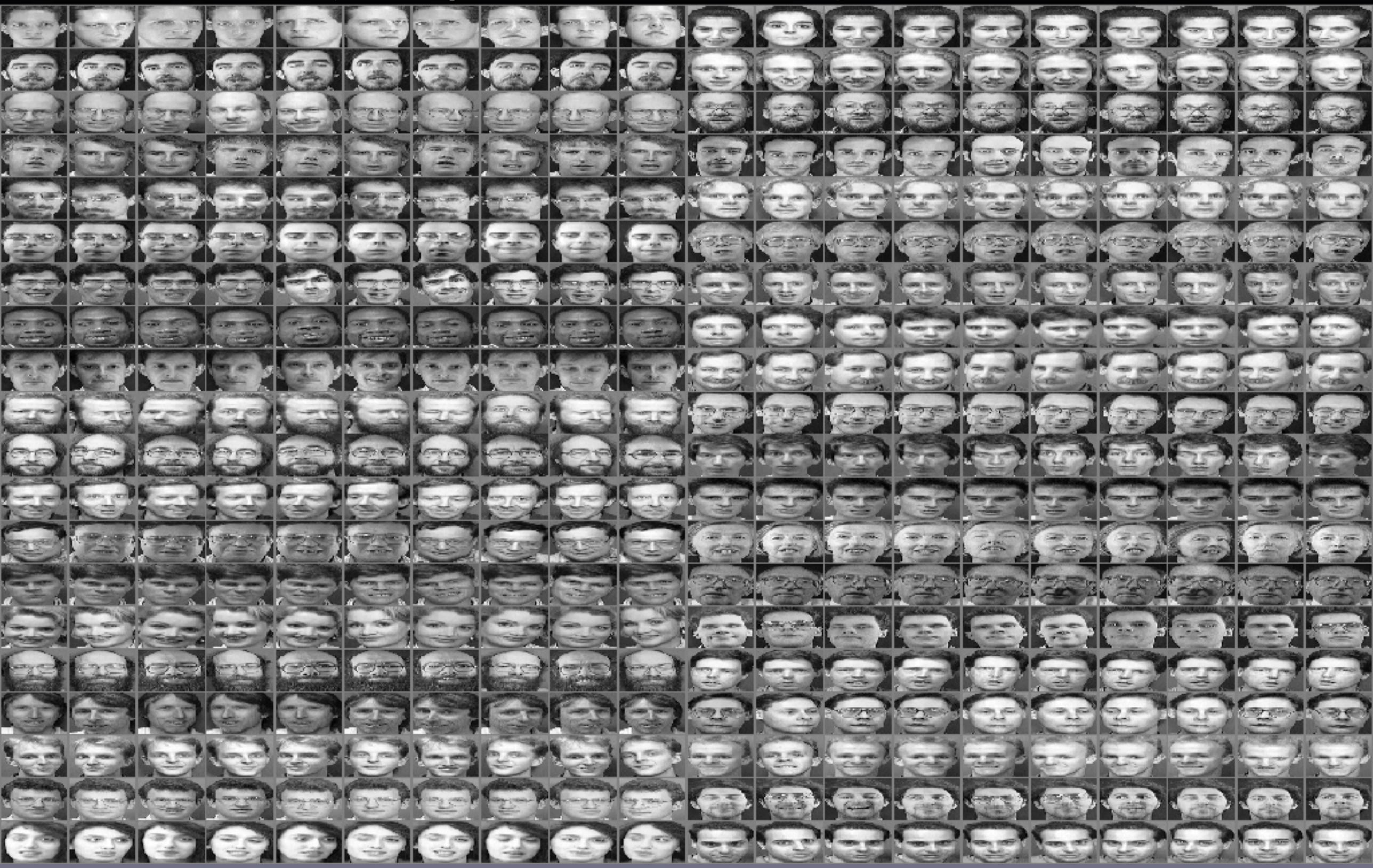
Mean face



Basis of variance (eigenvectors)

M. Turk; A. Pentland (1991). "Face recognition using eigenfaces".
Proc. IEEE Conference on Computer Vision and Pattern Recognition. pp. 586–591.

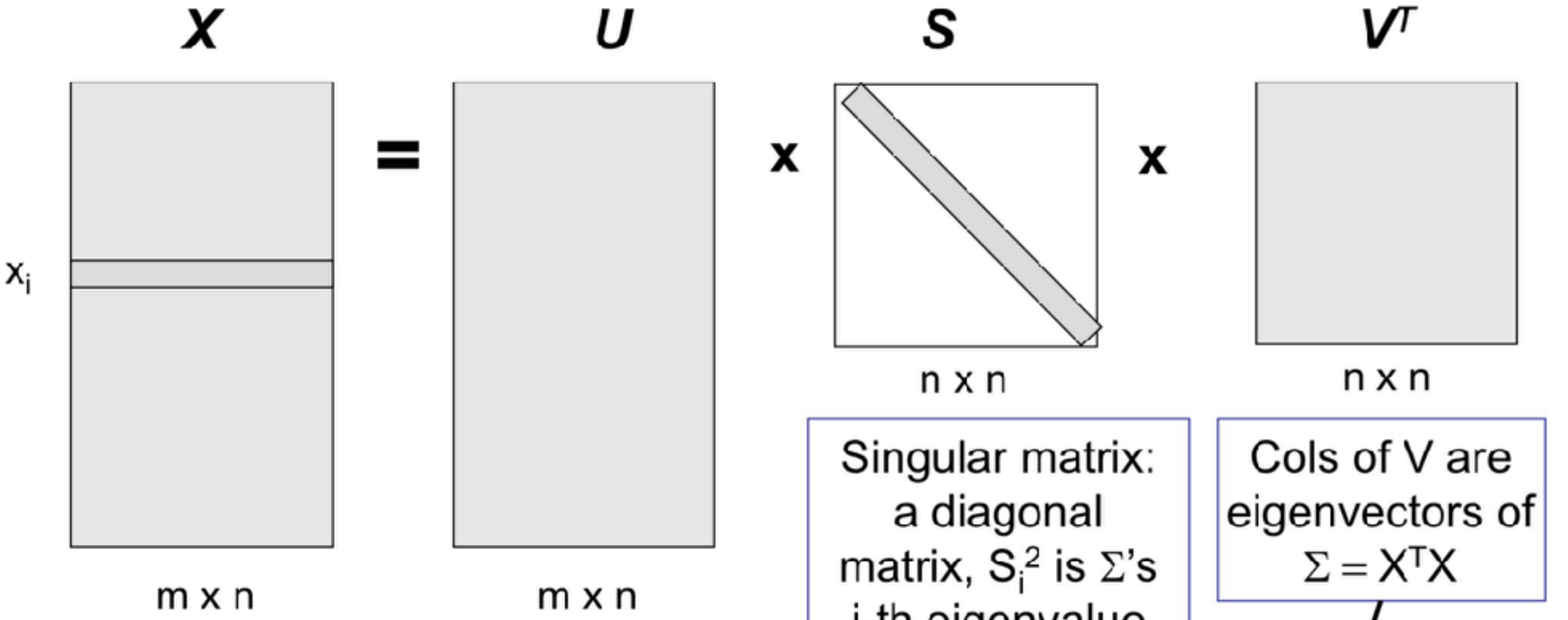
The ATT face database (formerly the ORL database), 10 pictures of 40 subjects each



PCA, scaling up

- Covariance of the image data is big. Finding eigenvector of large matrices is slow.
- Singular Value Decomposition (SVD) can be used to compute principal components.
- SVD steps:
  - Create centered data matrix X
  - Solve: $X = USV^T$
  - Columns of V are the eigenvectors of Σ sorted from largest to smallest eigenvalues.

Singular Value Decomposition



Singular matrix: a diagonal matrix, S_i^2 is Σ's i-th eigenvalue

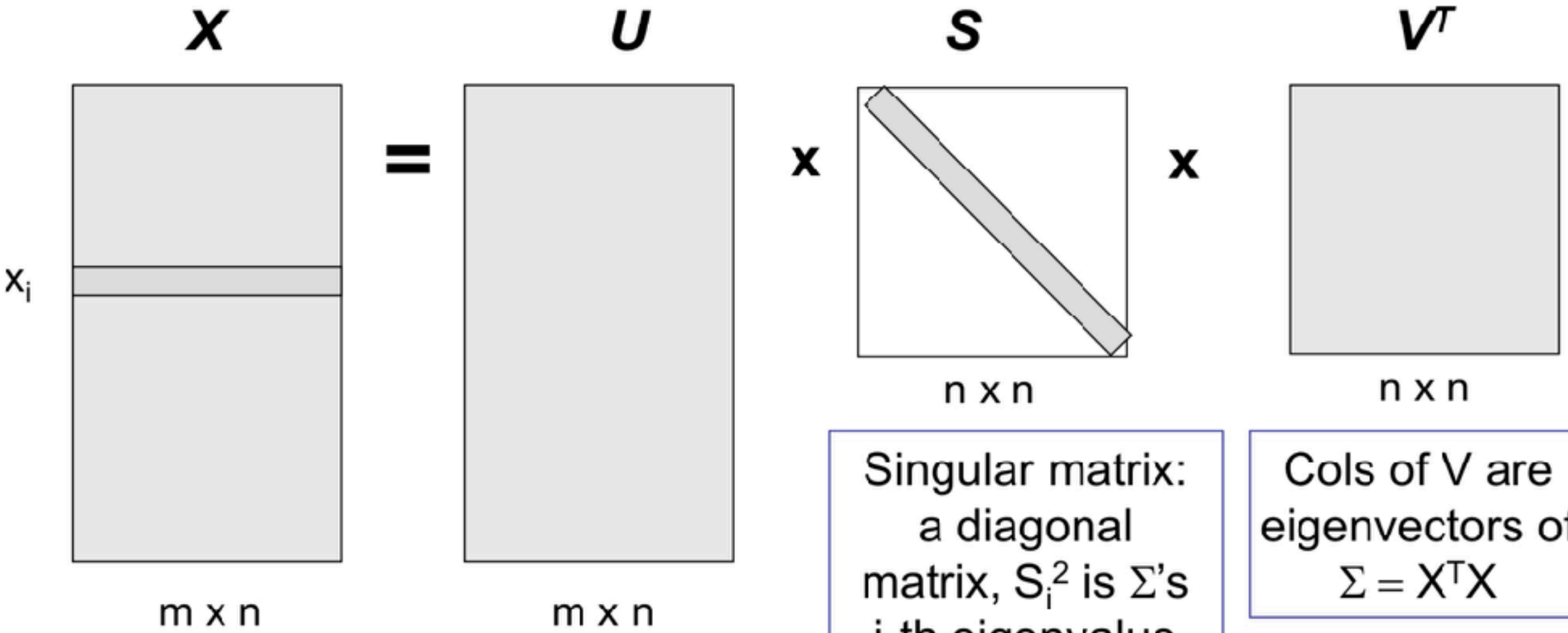
Cols of V are eigenvectors of $\Sigma = X^T X$

$$X^T X v_1 = VSU^T USV^T v_1 = s_1^2 v_1$$

X: our m x n data matrix, one row per data point

Each row of US gives coordinates of a data point in the projected space

Singular Value Decomposition



X: our m x n data matrix, one row per data point

Each row of US gives coordinates of a data point in the projected space

Singular matrix: a diagonal matrix, S_i^2 is Σ's i-th eigenvalue

Cols of V are eigenvectors of $\Sigma = X^T X$

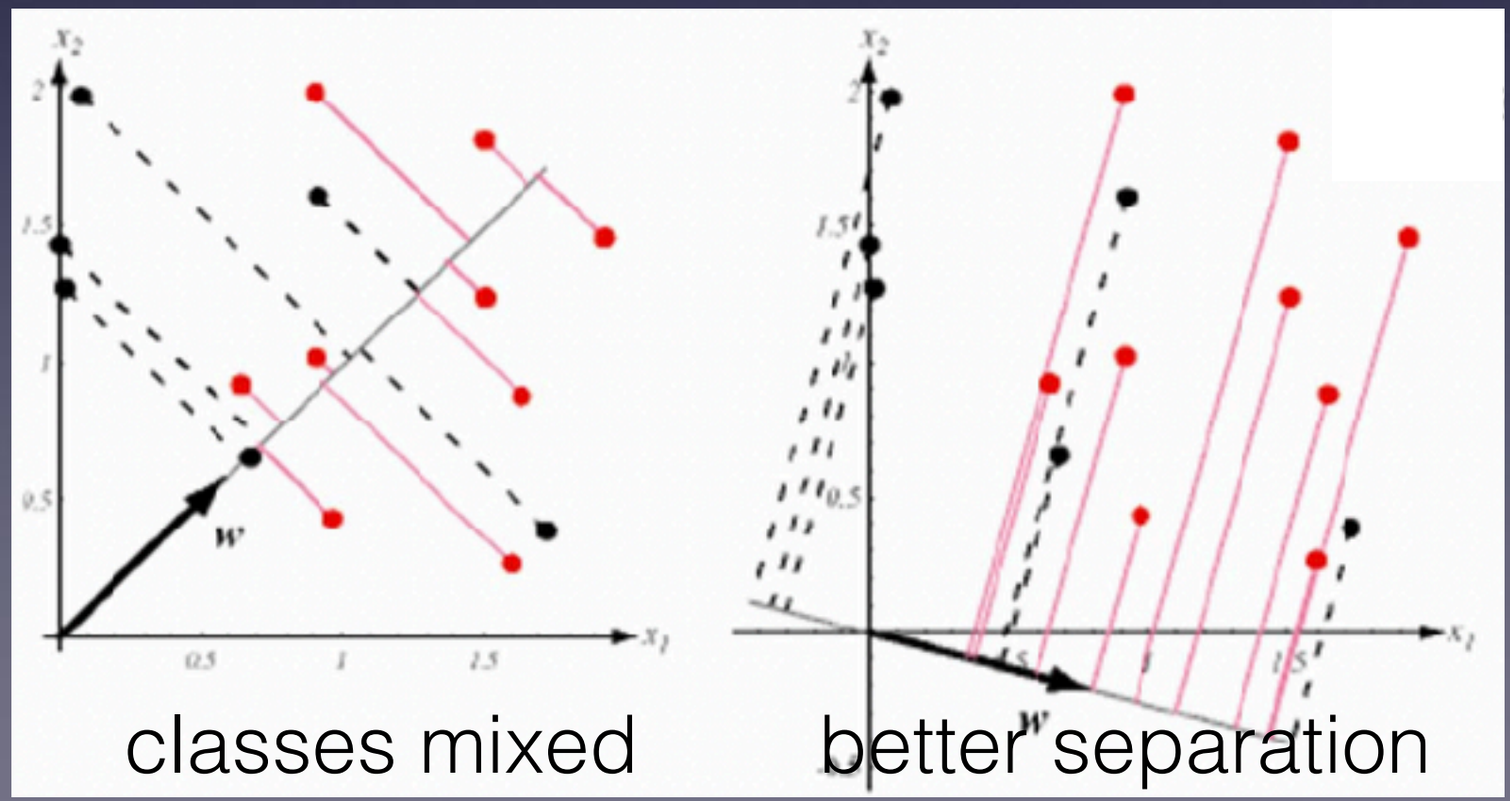
If X is centered, then cols of V are the principal components

PCA discussion

- Useful preprocessing for easing the "curse of dimensionality" problem.
  - Reduced dimension: simpler hypothesis space
  - Smaller VC dimension: less overfitting
- PCA can also be seen as noise reduction
- Fails when data consists of multiple separate clusters

Linear Discriminant Analysis (LDA)

- Also named Fisher Discriminant Analysis
- It can be viewed as
  - a dimension reduction method,
  - a generative classifier p(x|y), Gaussian with distinct μ for each class but shared Σ.



classes mixed better separation

LDA Objectives

- Find a project direction so that the separation between classes is maximized.
- Objective 1: maximize the distance between the projected means of different classes

original means:

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{\mathbf{x} \in C_1} \mathbf{x} \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{\mathbf{x} \in C_2} \mathbf{x}$$

projected means:

$$\mathbf{m}_1' = \frac{1}{N_1} \sum_{\mathbf{x} \in C_1} \mathbf{w}^{\mathrm{T}} \mathbf{x} \quad \mathbf{m}_2' = \frac{1}{N_2} \sum_{\mathbf{x} \in C_2} \mathbf{w}^{\mathrm{T}} \mathbf{x}$$

LDA Objectives

- Objective 2: minimize *scatter* (variance within class)

Total within class scatter for projected class i:

$$s_i^2 = \sum_{\mathbf{x} \in C_i} (\mathbf{w}^{\mathrm{T}} \mathbf{x} - \mathbf{m}_i')^2$$

Total within class scatter: $s_1^2 + s_2^2$

LDA Objective

- There are a number of different ways to combine the two objectives.
- LDA seeks to optimize the following objective:

$$\arg\max_{w} \frac{|m'_1 - m'_2|^2}{s_1^2 + s_2^2}$$

$$|m'_1 - m'_2|^2 = (w^T m_1 - w^T m_2)^2$$

$$= w^T (m_1 - m_2)(m_1 - m_2)^T w$$

$$= w^T S_B w$$

$$s_1^2 = \sum_{x \in C_1} (w^T x - w^T m_1)^2 = \sum_x w^T (x - m_1)(x - m_1)^T w$$

$$= w^T \left(\sum_x (x - m_1)(x - m_1)^T \right) w = w^T S_1 w$$

$$s_1^2 + s_2^2 = w^T (S_1 + S_2) w$$

$$= w^T S_w w$$

The LDA Objective

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_w \mathbf{w}}$$

$$S_i = \sum_{x \in \mathbf{C}_i} (x - m_i)(x - m_i)^T$$

$S_B = (m_1 - m_2)(m_1 - m_2)^T$
the between class scatter matrix

$S_w = S_1 + S_2$
the total within class scatter matrix, where

$$S_i = \sum_{x \in C_i} (x - m_i)(x - m_i)^T$$

- The above objective is known as generalized Reyleigh quotient, and it's easy to show a w that maximizes $J(w)$ must satisfy $S_B w = \lambda S_w w$
- Noticing that $S_B w = (m_1 - m_2)(m_1 - m_2)^T w$ always take the direction of $m_1 - m_2$

Scalar

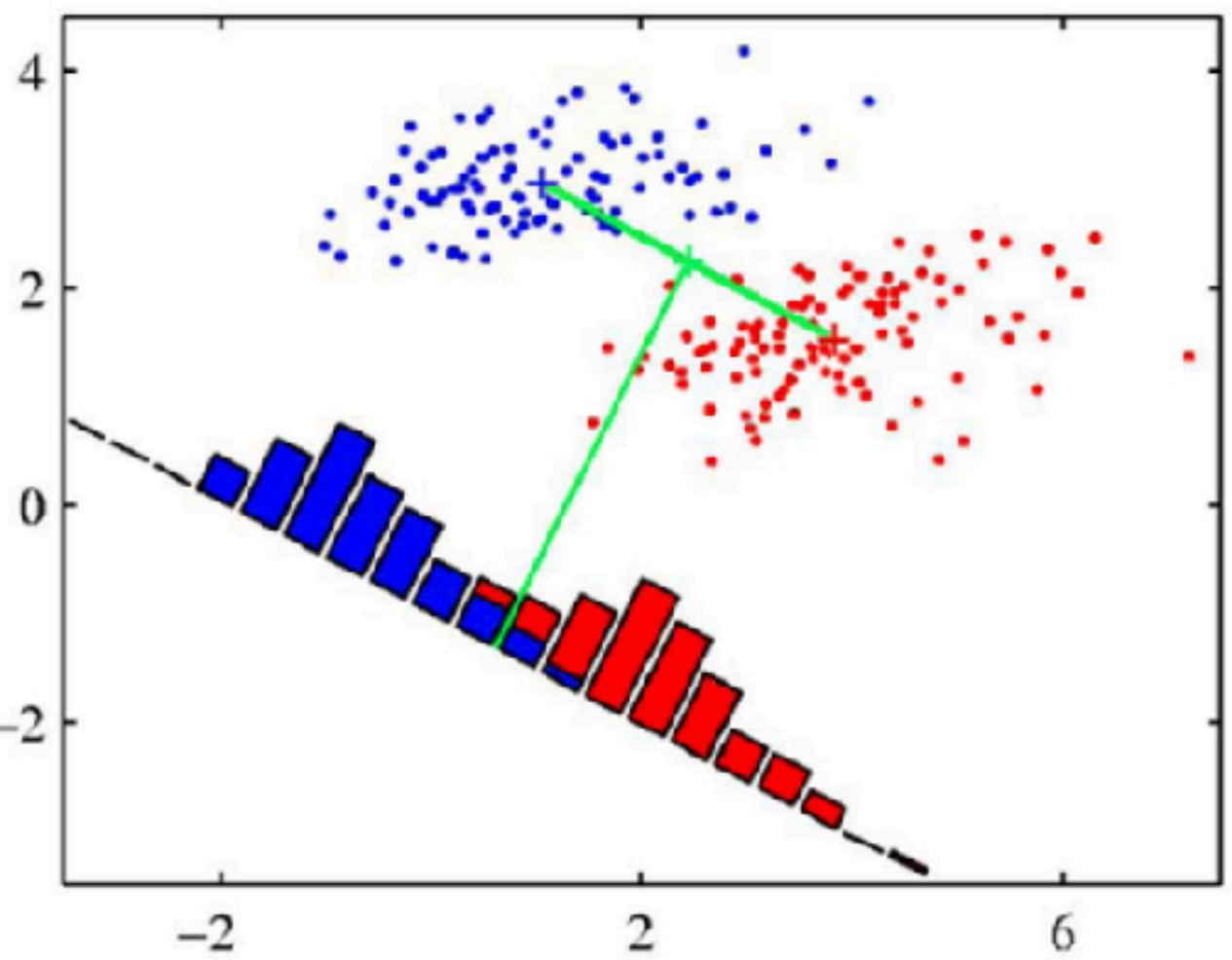
- Ignoring the scalars, this leads to:

$$(m_1 - m_2) = S_w w$$

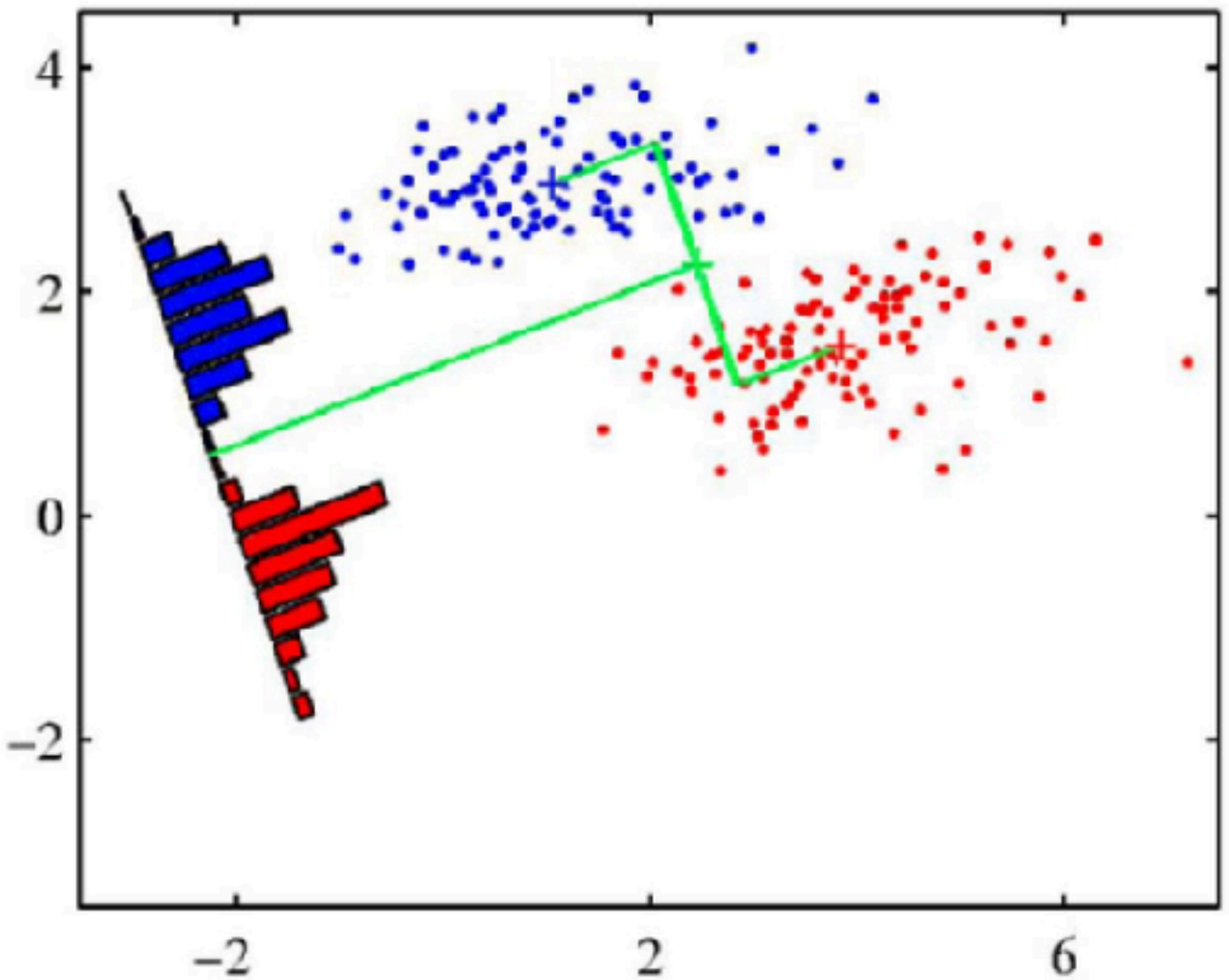
$$w = S_w^{-1}(m_1 - m_2)$$

LDA for two classes

$$\mathbf{w} = S_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$



$w = m_1 - m_2$



$w = S_w^{-1}(m_1 - m_2)$

LDA for Multi-Classes

$$J(\mathbf{w}) = \frac{\mathbf{w}^{\mathrm{T}} S_B \mathbf{w}}{\mathbf{w}^{\mathrm{T}} S_{\mathbf{w}} \mathbf{w}}$$

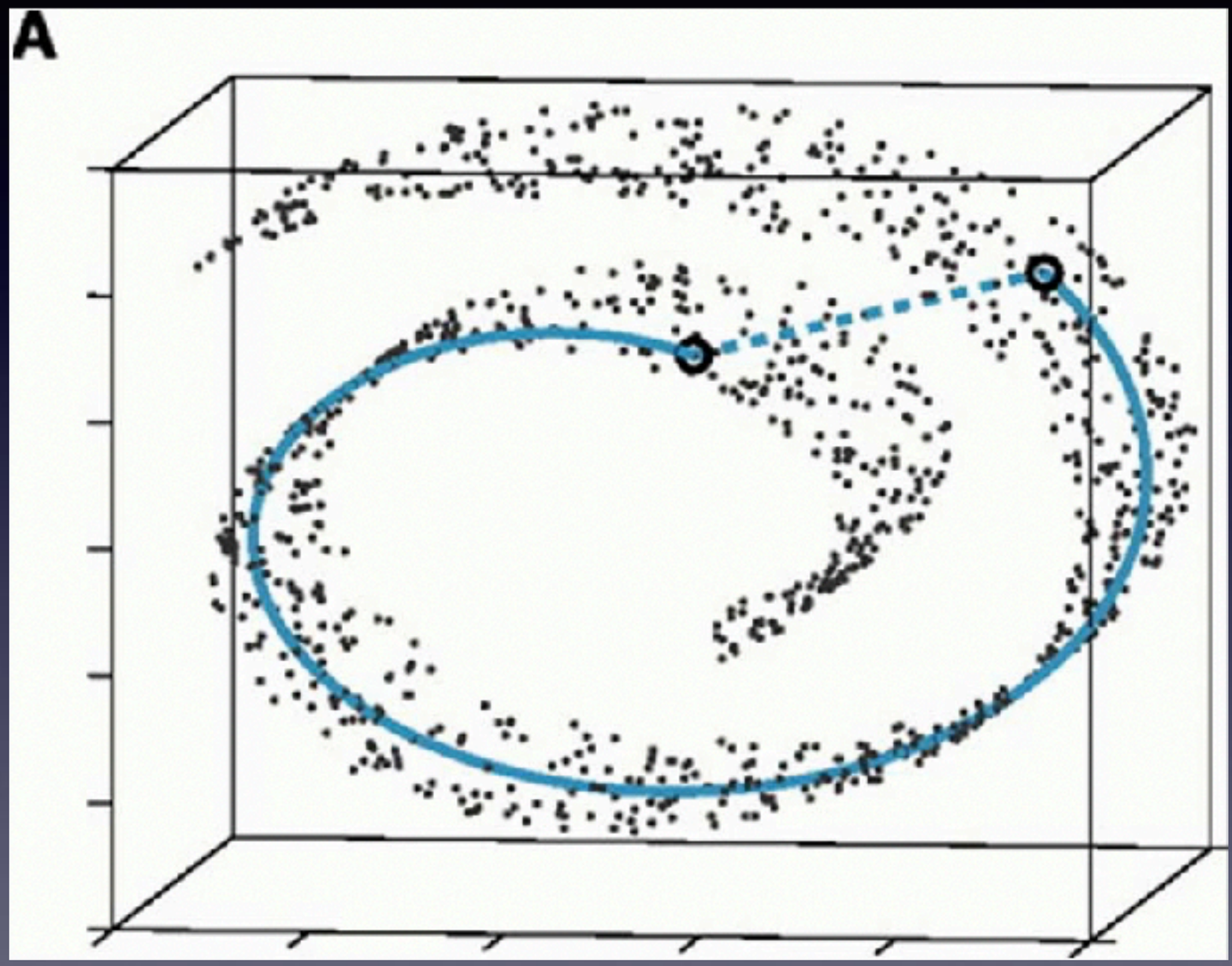
- Objective remains the same, with slightly different definition for between-class scatter:

$$S_B = \frac{1}{k} \sum_{i=1}^{k} (m_i - m)(m_i - m)^{\mathrm{T}}$$

- Solution: k-1 eigenvectors of $S_{\mathbf{w}}^{-1} S_B$

Nonlinear Dimension Reduction

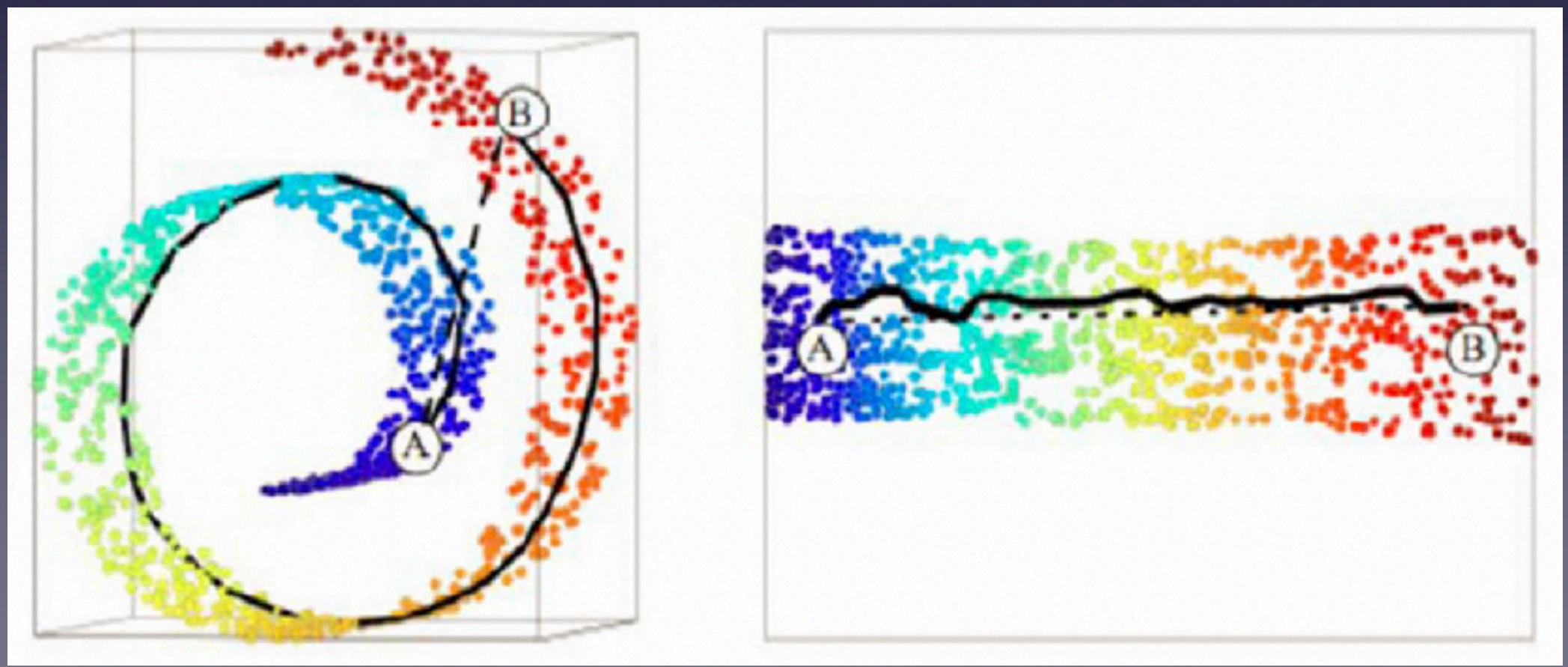
- Data often lies on or near a nonlinear low-dimensional curve.
- We call such a low-d structure *manifolds*
- Algorithms include: ICA, LLE, Isomap.



swiss roll data

ISOMAP: Isometric Feature Mapping

- A non-linear method for dimensionality reduction
- Preserves the global, nonlinear geometry of the data by preserving the geodesic distances.
- Geodesic: shortest route between two points on the surface of a manifold.



ISOMAP algorithm

1. Approximate the geodesic distance between every pair of points in the data.
    - The manifold is locally linear
    - Euclidean distance works well for points that are close enough.
    - For points that are far apart, their geodesic distance can be approximated by summing up local Euclidean distances.
2. Find a Euclidean mapping of the data that preserves the geodesic distance.

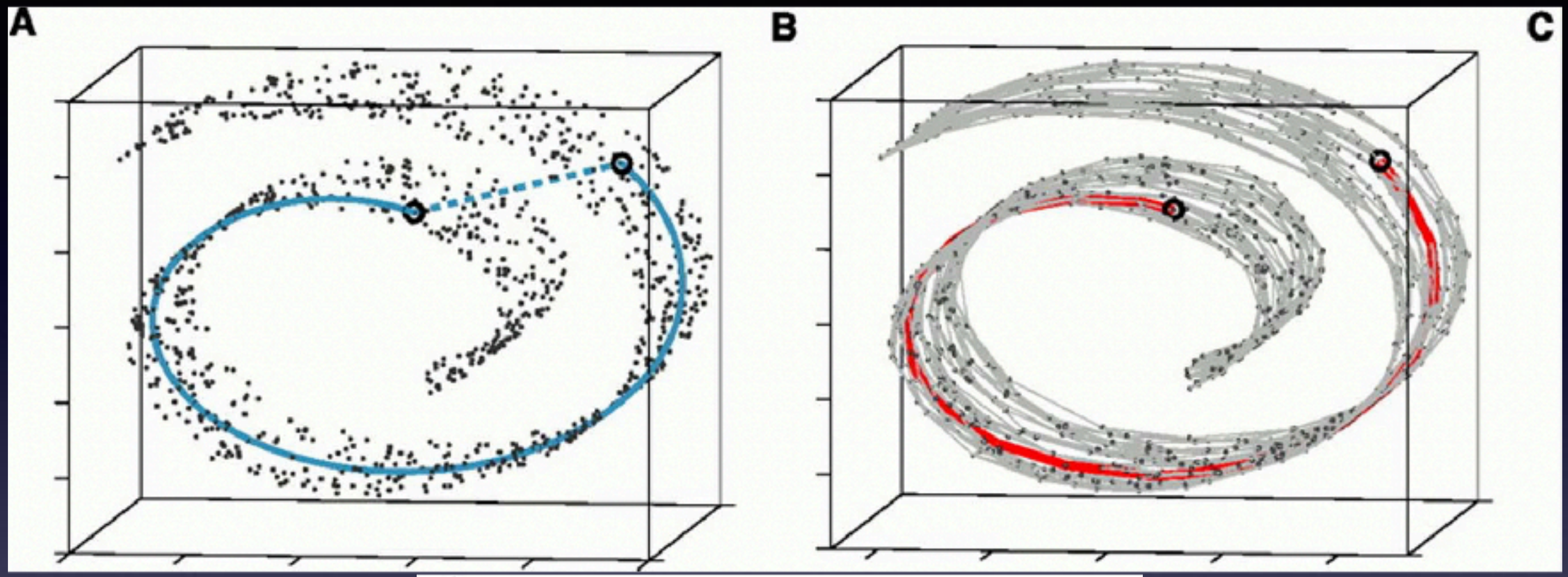
Geodesic Distance

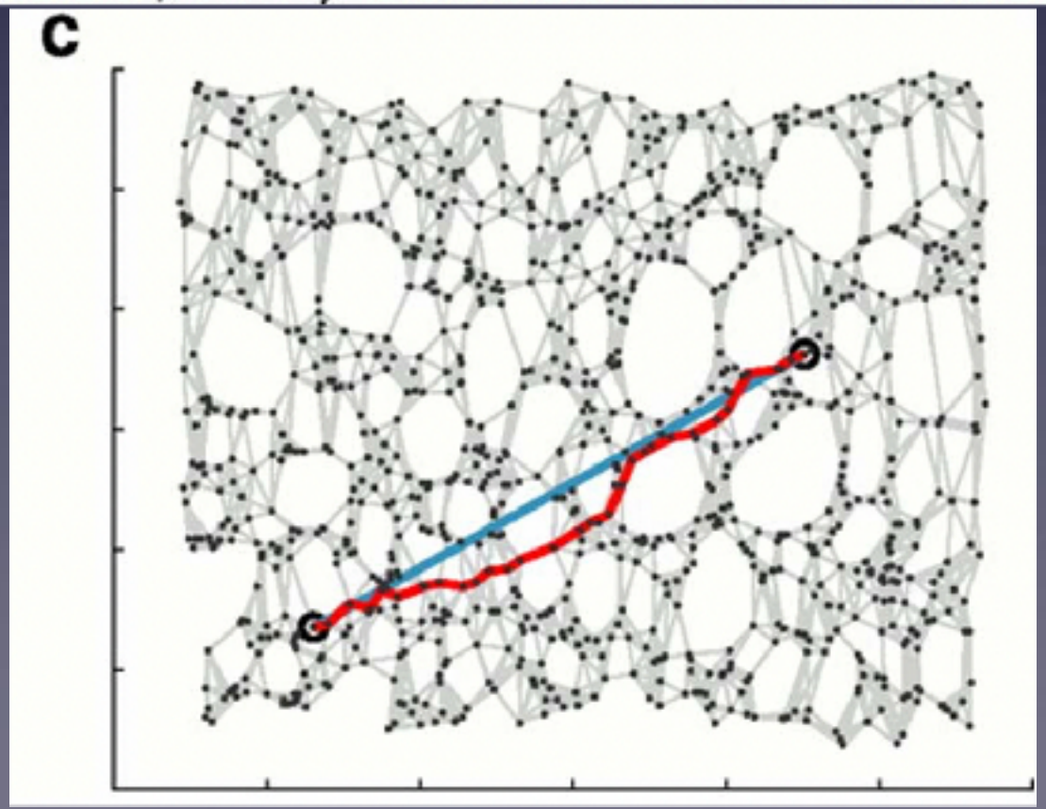
- Construct a graph by:
  - Connecting *i* and *j* if:
    - d(*i*,*j*) < ε (if computing ε-isomap), or
    - i is one of j's k nearest neighbors (k-isomap)
      - Set the edge weight equal d(i,j) - Euclidean distance
- Compute the Geodesic distance between any two points as the shortest path distance.

Compute low-dimensional mapping

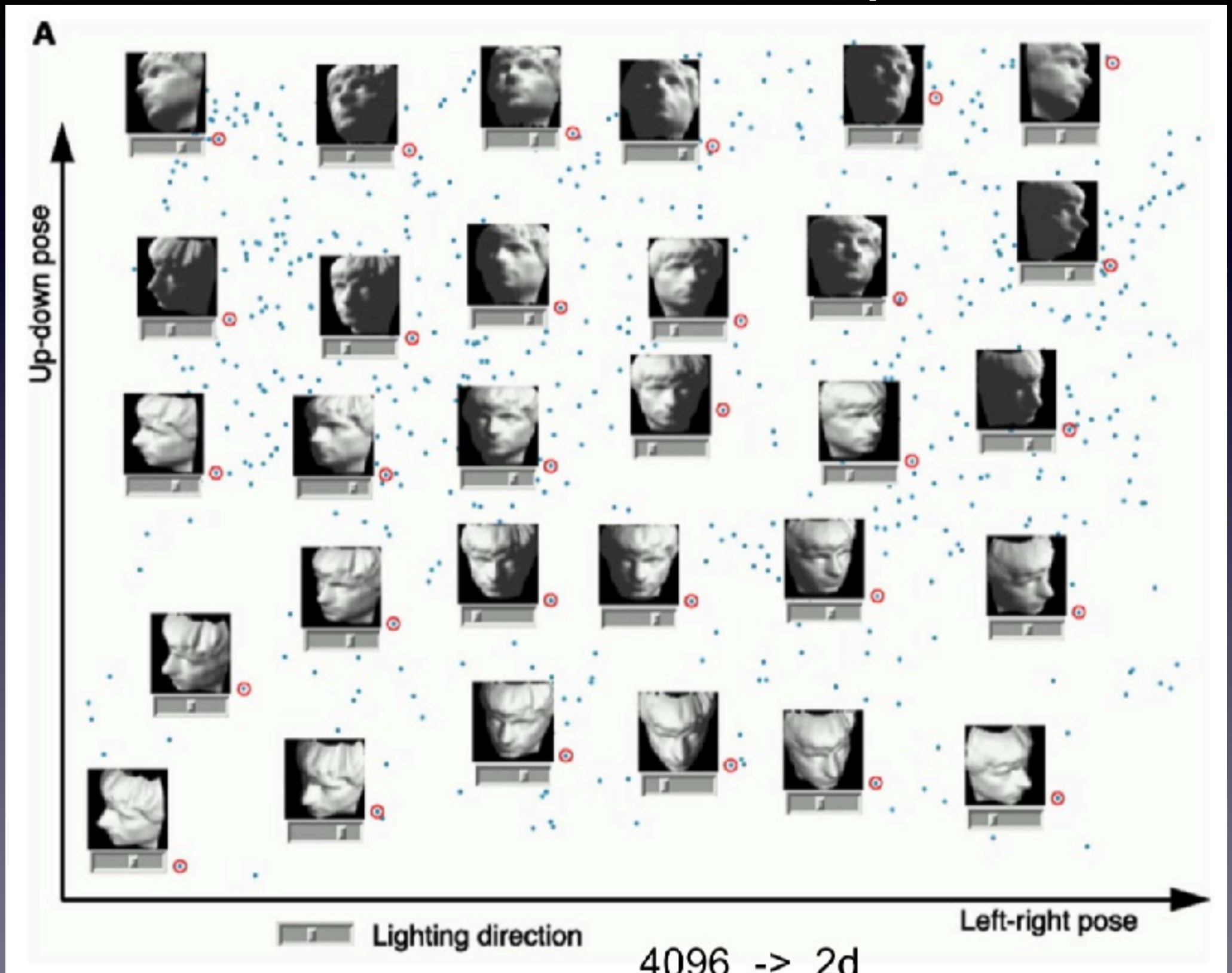
- We can use Multi-Dimensional Scaling (MDS), a class of statistical techniques that:
- Given:
  - n x n matrix of dissimilarities between n objects
- Outputs:
  - a coordinate configuration of the data in low-d space R^d whose Euclidean distances closely match given dissimilarities.

ISOMAP on Swiss Roll Data



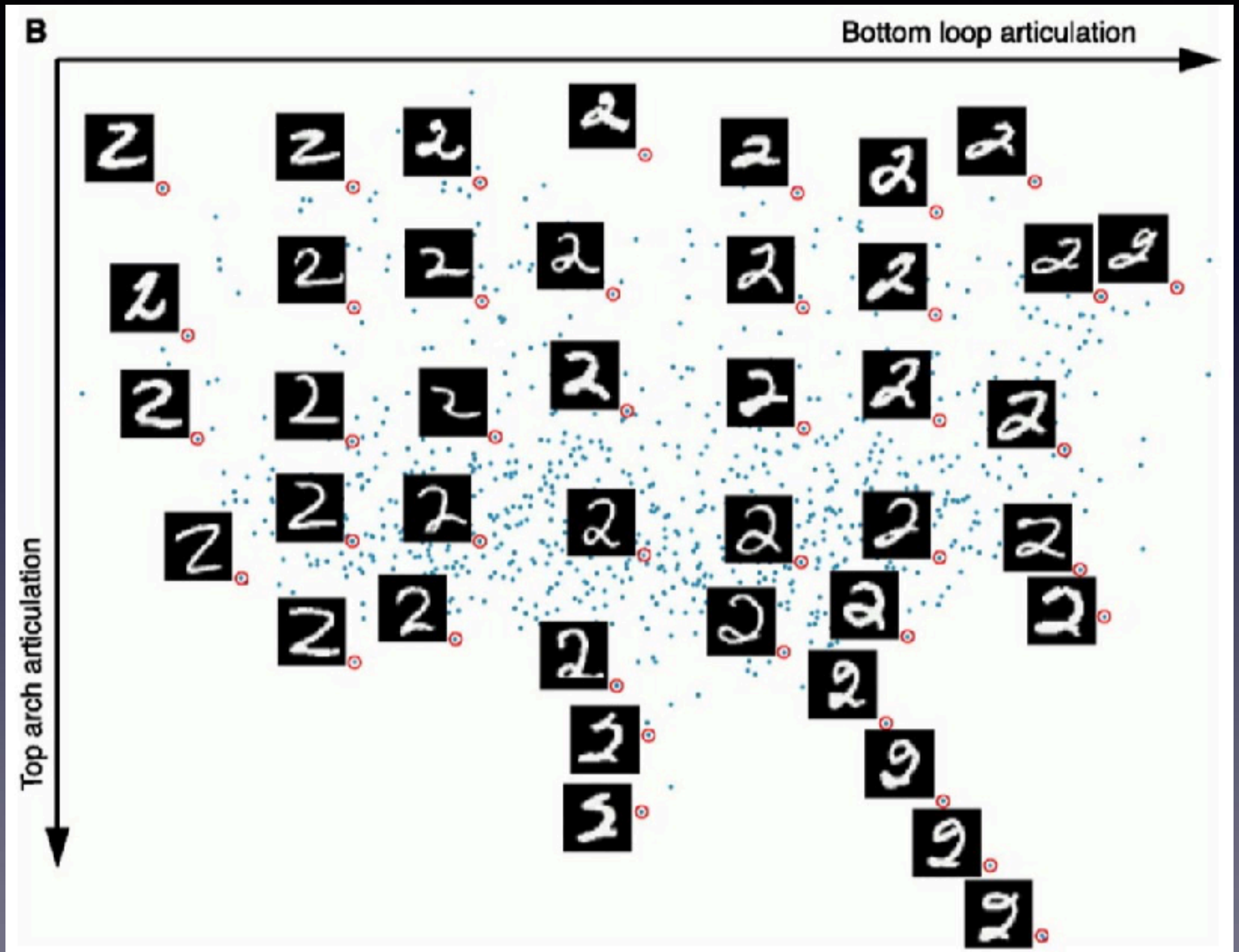


ISOMAP Examples



4096 -> 2d

ISOMAP Examples



Regression

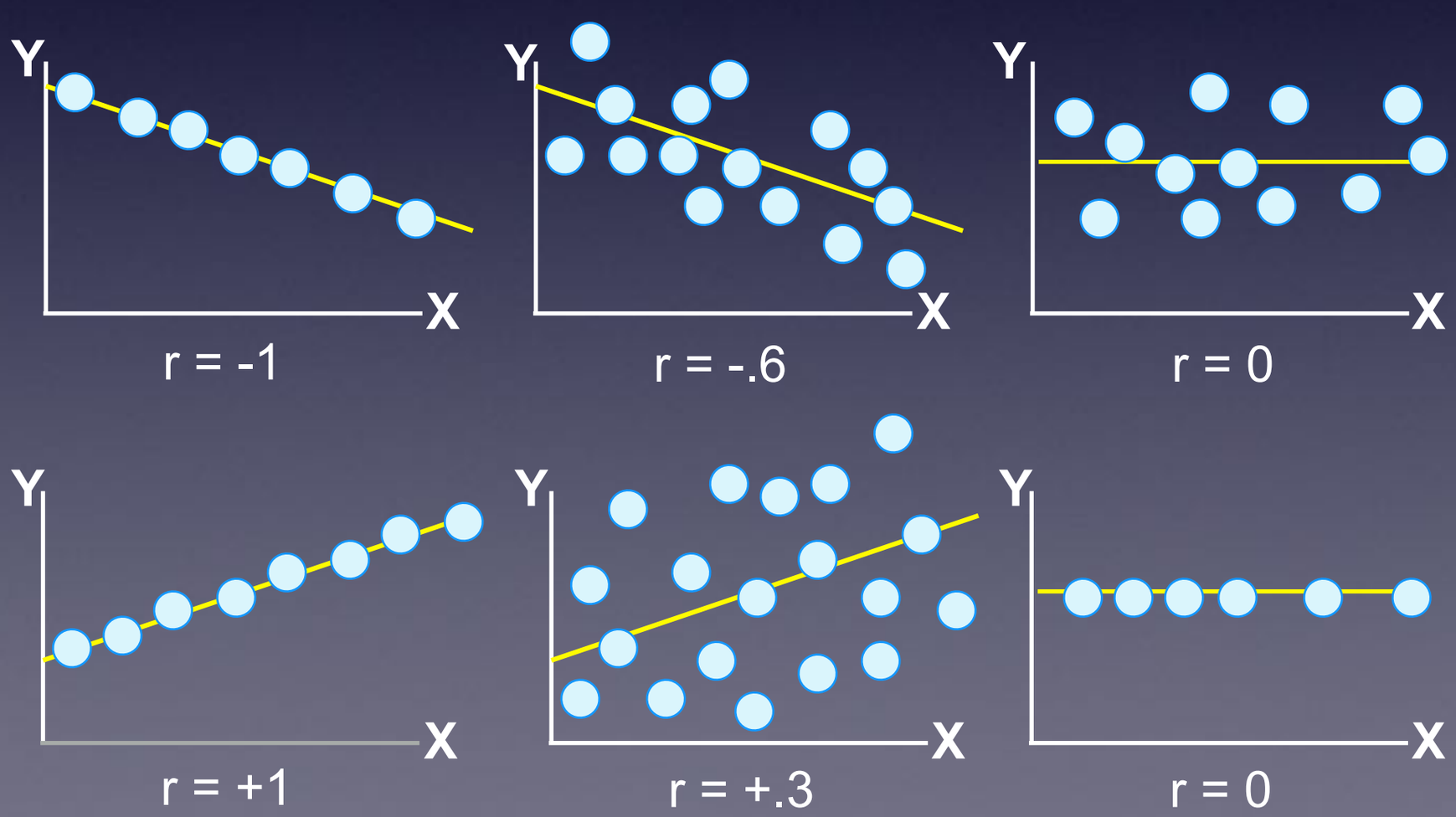
Machine Learning Roadmap

	continuous (predicting a quantity)	discrete (predicting a category)
unsupervised	**Dimension Reduction**	**Clustering**
supervised	**Regression**	**Classification**

Pearson's Correlation Coefficient

$$r = \frac{\mathrm{cov}(x, y)}{\sqrt{\mathrm{var}(x)}\sqrt{\mathrm{var}(y)}}$$

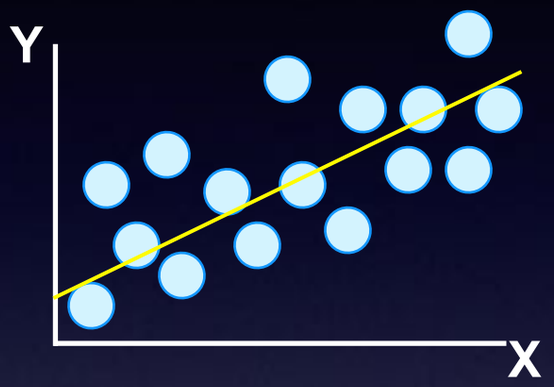
- Unit-less, normalized between [-1, 1]

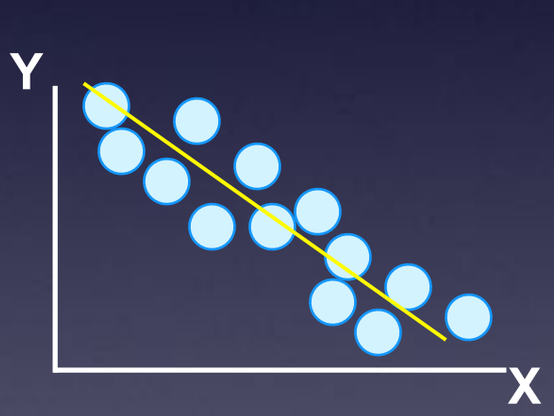


Figures modified from: Statistics for Managers Using Microsoft® Excel 4th Edition, 2004 Prentice-Hall

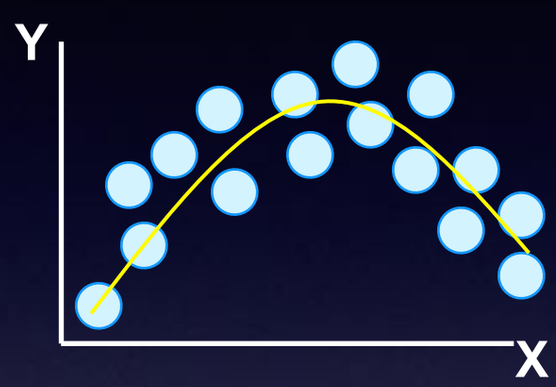
Linear Correlations

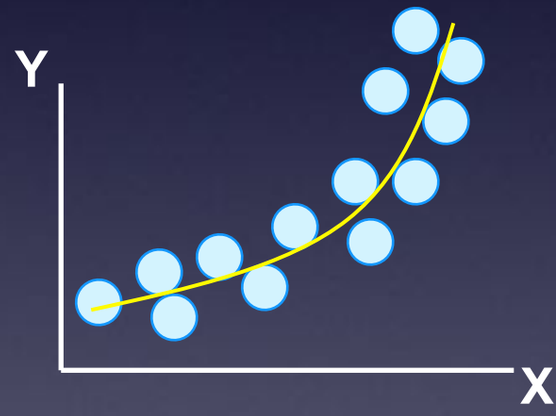
Linear relationships



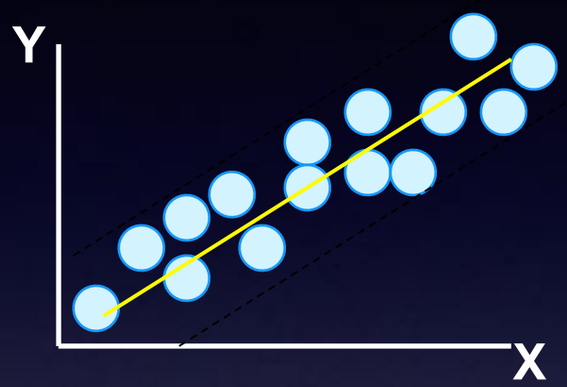


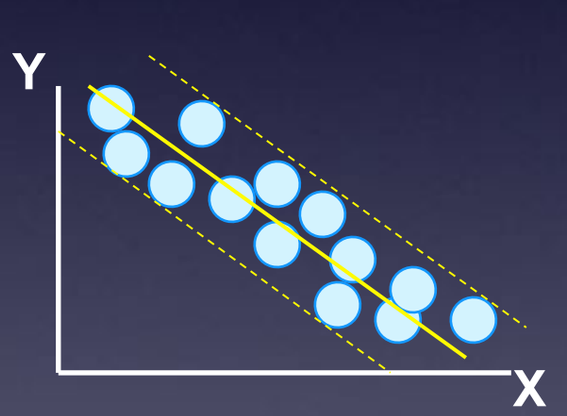
Curvilinear relationships



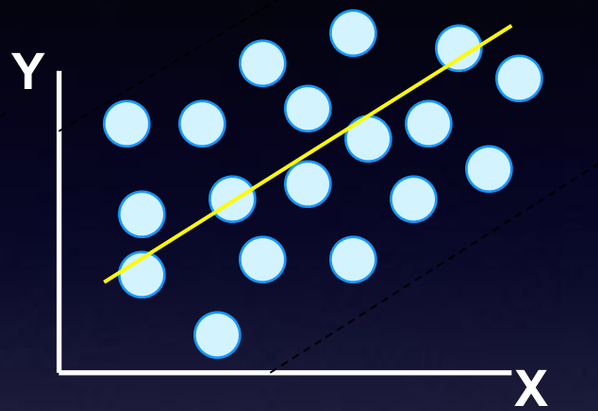


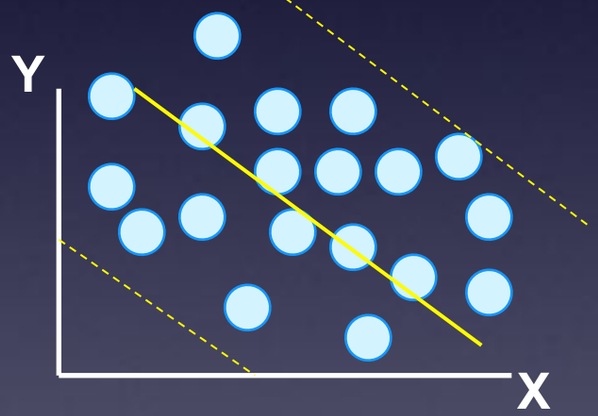
Strong relationships



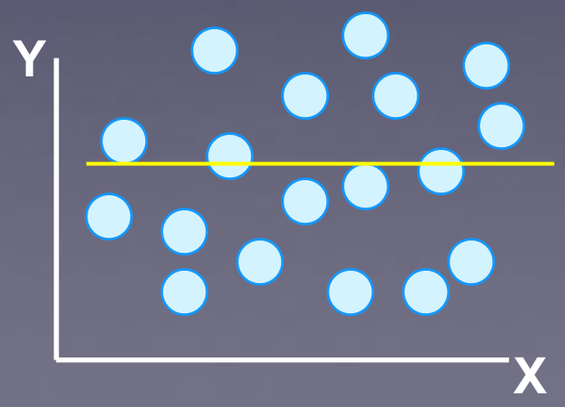


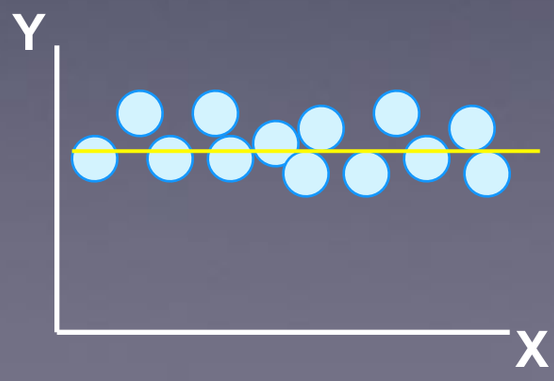
Weak relationships





No relationship





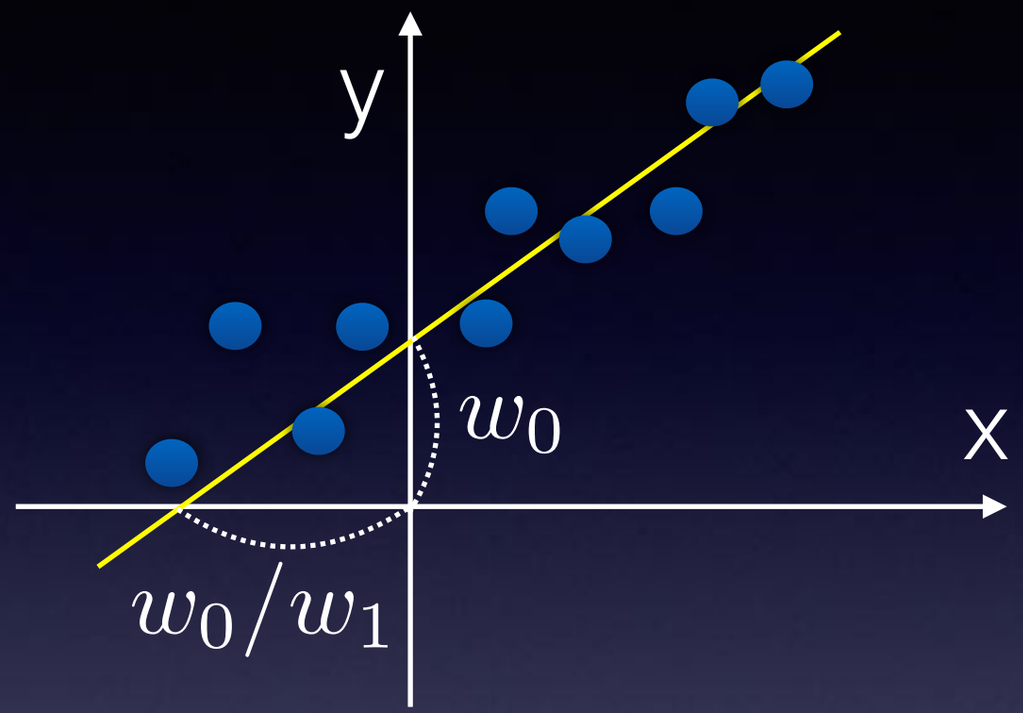
Regression

- In correlation, two variables are treated as independent.
- In regression, one variable (x) is independent, while the other (y) is dependent.
- Goal: if you know something about x, this would help you predict something about y.

Simple Linear Regression

- Expected value at a given level of x:

$$y = w_0 + w_1 x$$



- Predicted value for a new x:

$$y' = \underbrace{w_0 + w_1 x} + \varepsilon$$

fixed exactly on the line

random error ε that follows a normal distribution with 0 mean and variance σ^2

Multiple Linear Regression

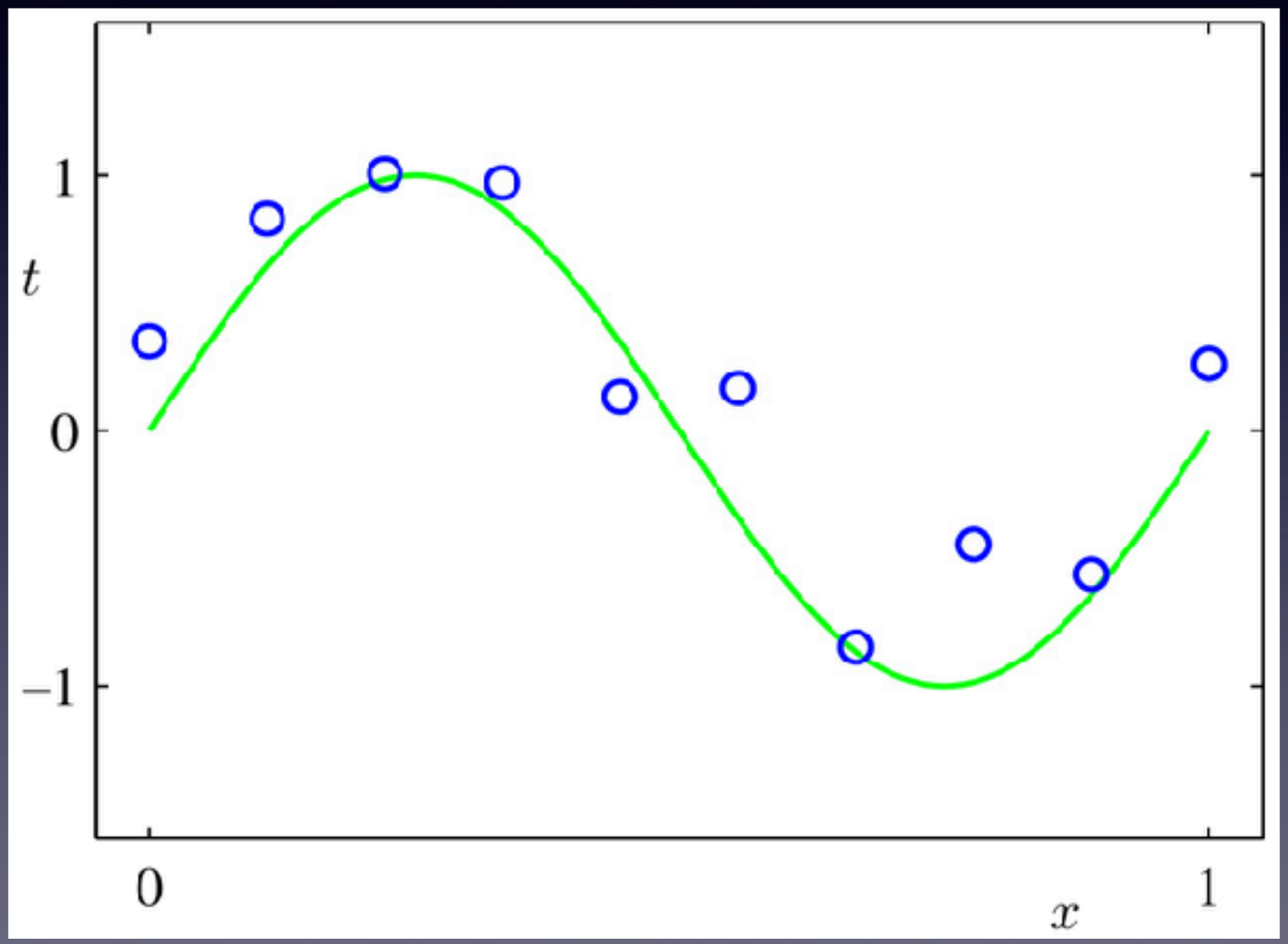
$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D$$

- Linear function of parameters $w_0, ..., w_D$, also a linear function of the input variables x_i , has very restricted modeling power (can't even fit curves).

- Assumes that:
  - The relationship between X and Y is linear.
  - Y is distributed normally at each value of X.
  - The variance of Y at each value of X is the same.
  - The observations are independent.

Linear Regression

- Before going further, let's take a look at polynomial line fitting (polynomial regression.)



Given N=10 blue dots, try to find the function $\sin(2\pi x)$ that is used for generating the data points.

Polynomial Regression

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M + \varepsilon$$

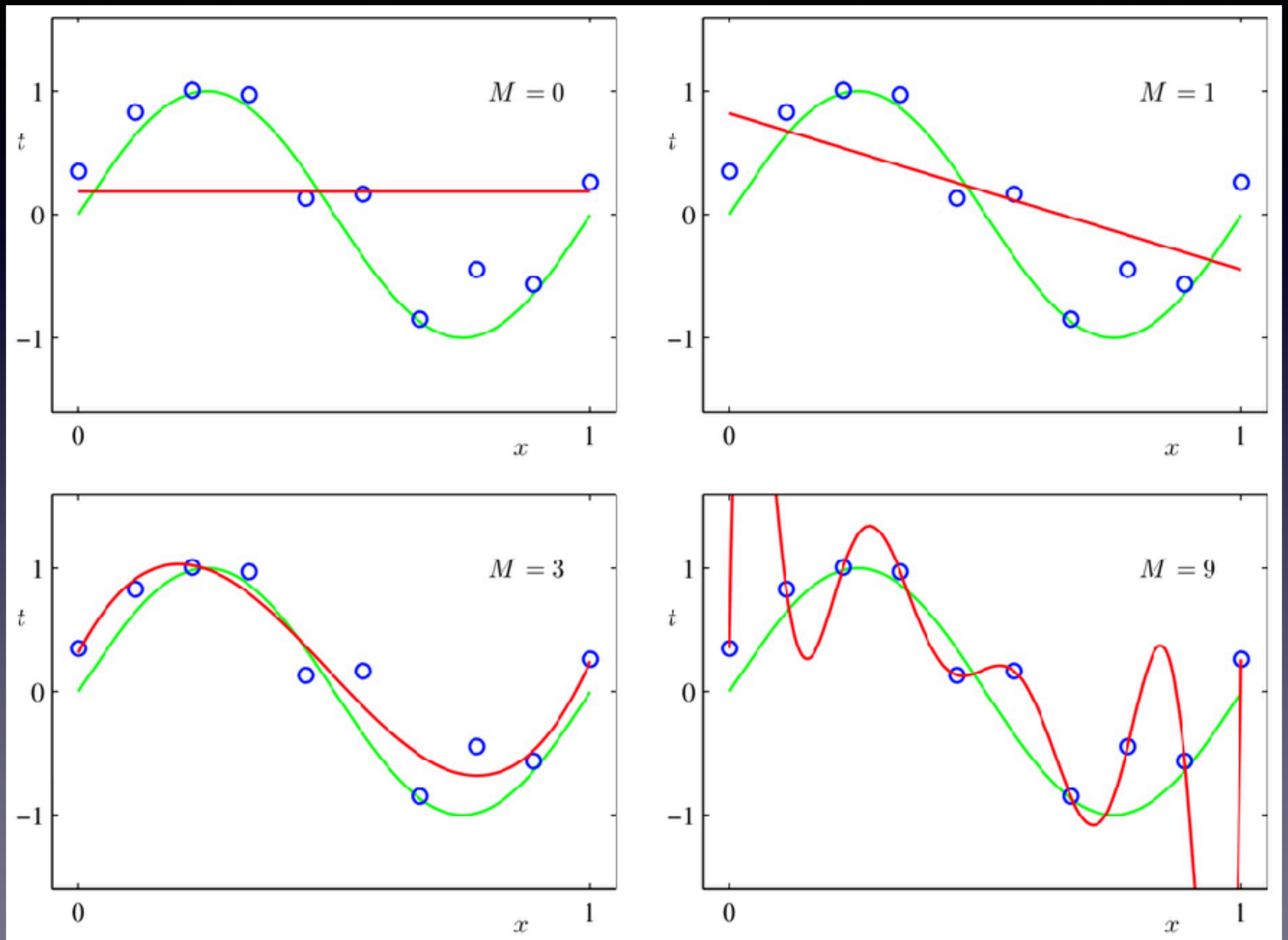
- Polynomial line fitting:
  - M is the order of the polynomial
  - linear function of the coefficients $\mathbf{w}$
  - nonlinear function of x

- Objective: minimize the error between the predictions $y(x_n, \mathbf{w})$ and the target value t_n of x_n

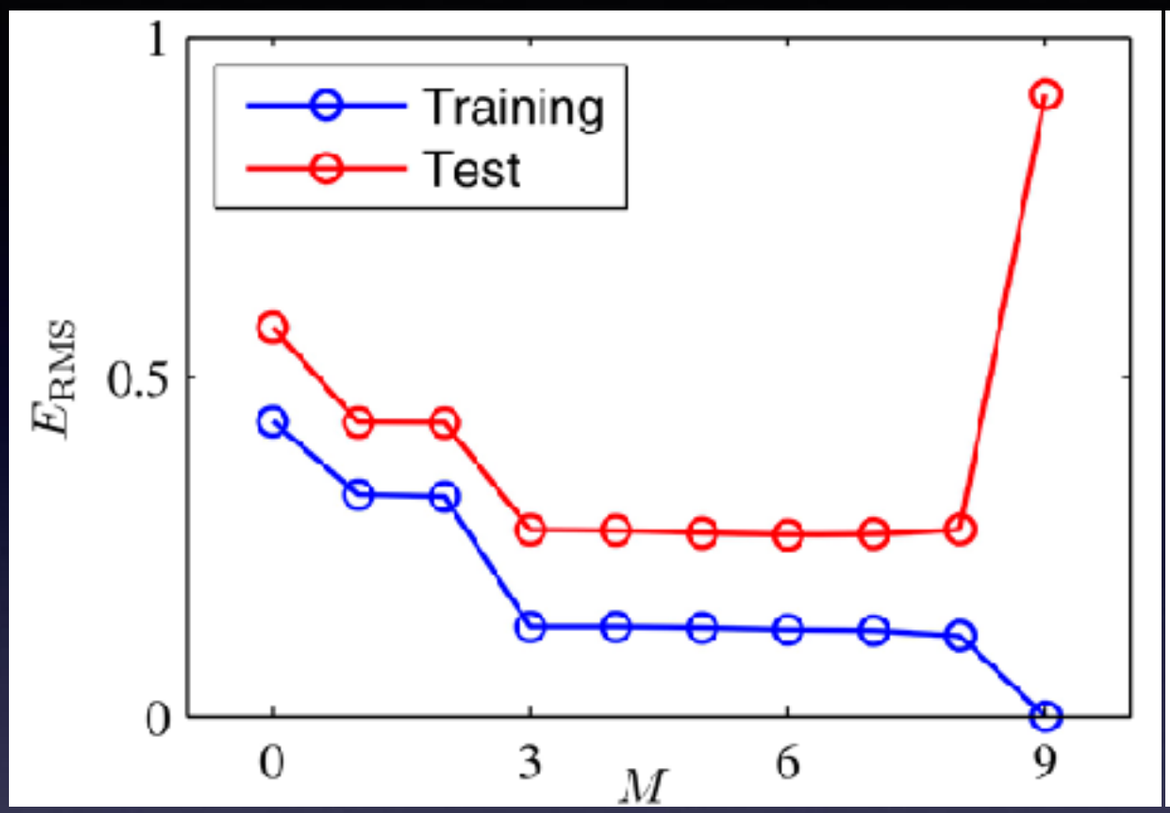
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

or, the root-mean-square error $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^*)/N}$

Polynomial regression w. var. M



Polynomial regression w. var. M

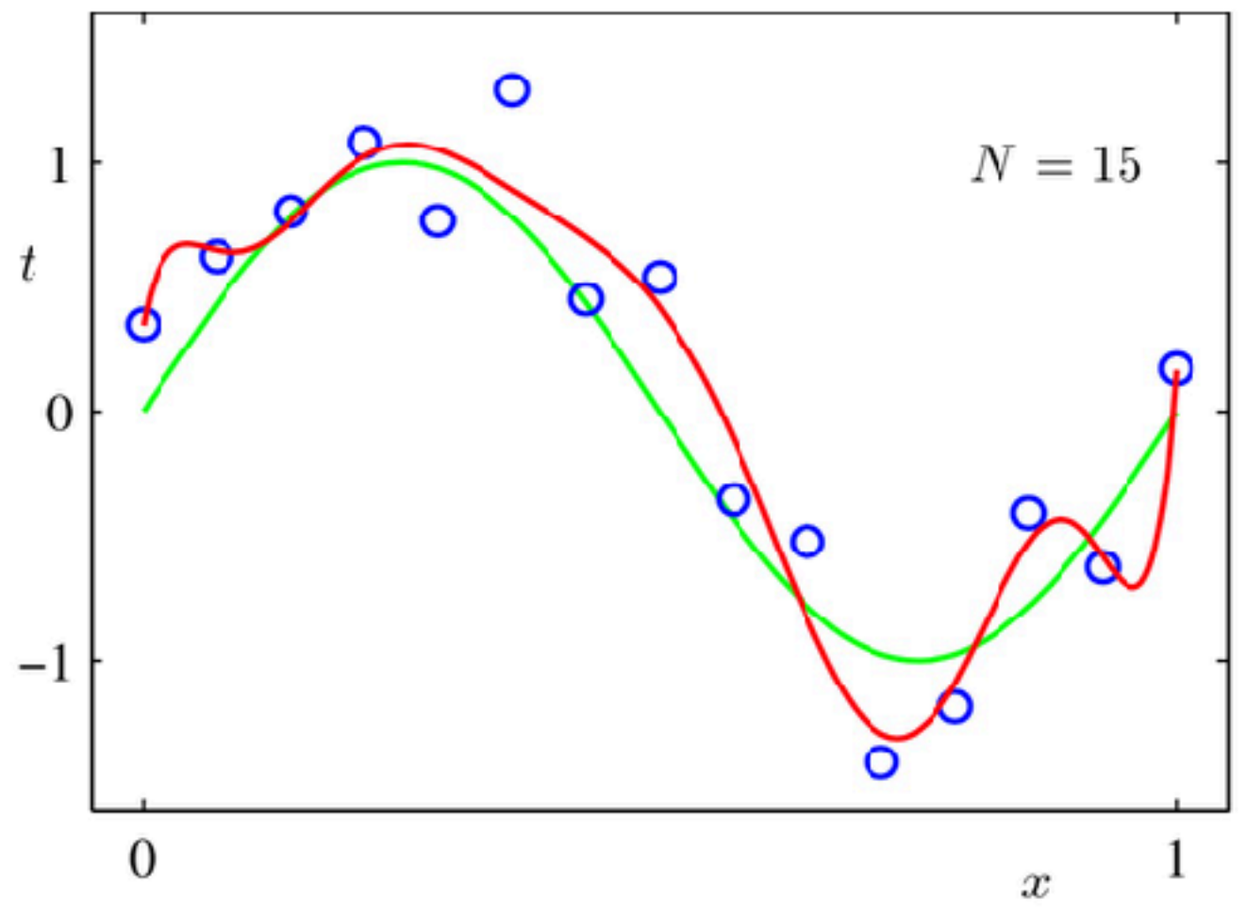


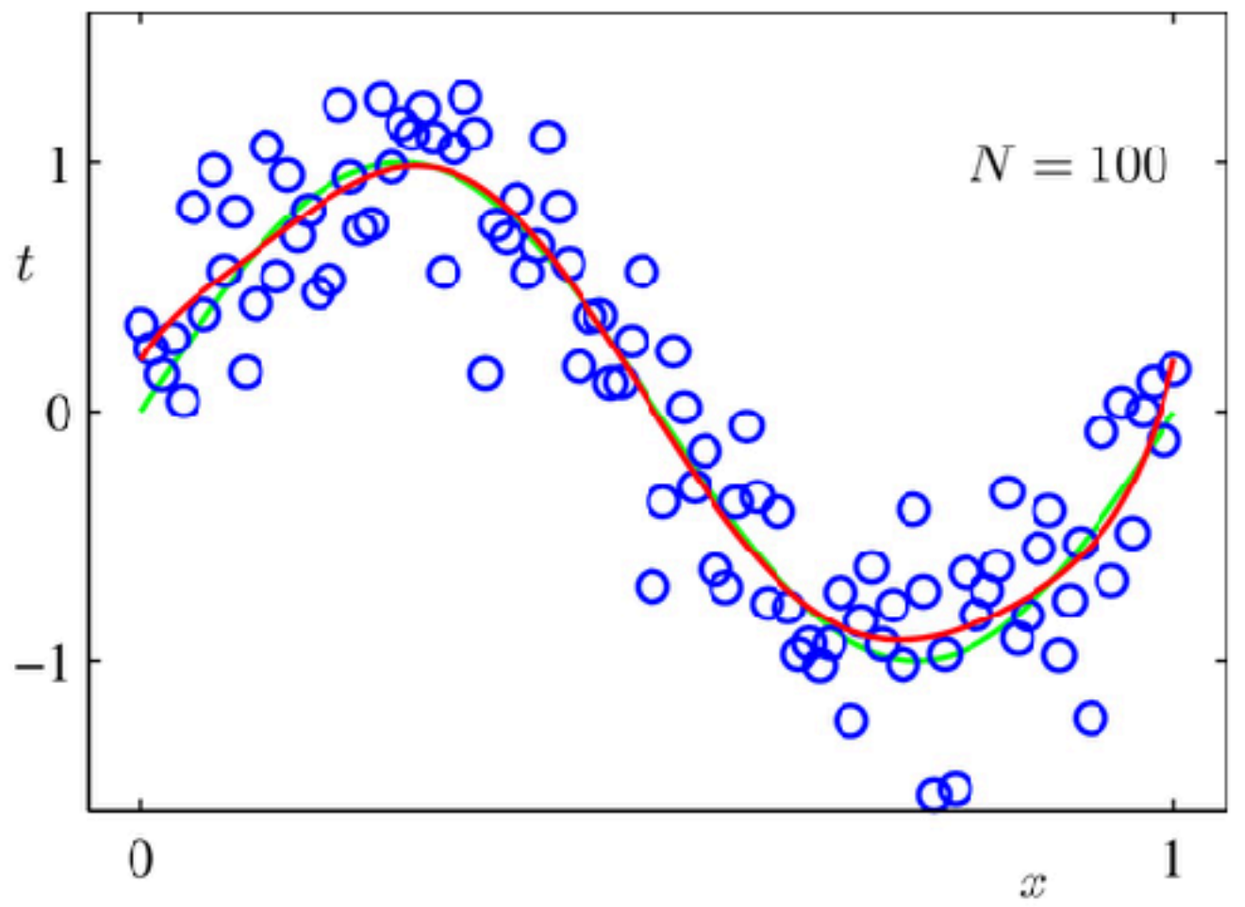
	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

- There's only 10 data points, i.e., 9 degrees of freedom; we can get 0 training error when M=9.
- Food for thought: make sure your deep neural network's is not just "memorizing the training data when its M >> data's DoF.

What happens with more data?

- With M=9, but N=15 (left) and N=100, the over-fitting problem is greatly reduced.
- ML is all about balancing M and N. One rough heuristic is that N should be 5x-10x of M (model complexity, not necessarily the number of param.)



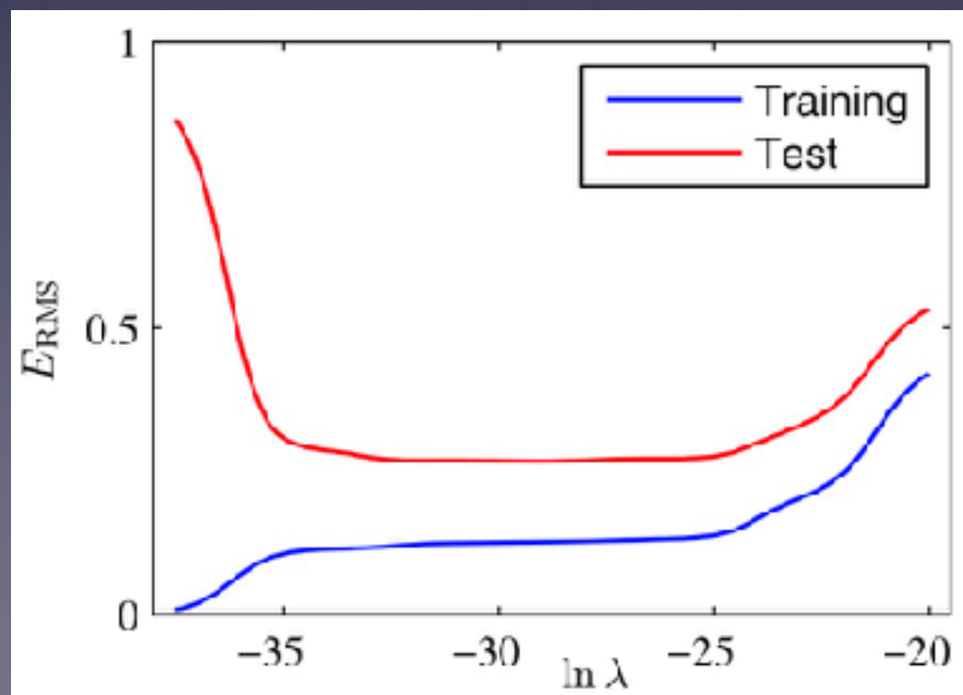


Regularization

- Regularization: used for controlling over-fitting.
  - E.g., discourage coefficients from reaching large values:

$$\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

where $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w} = w_0^2 + w_1^2 + \cdots + w_M^2$



	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Linear Models for Regression

- Extending linear regression to linear combinations of fixed nonlinear functions:

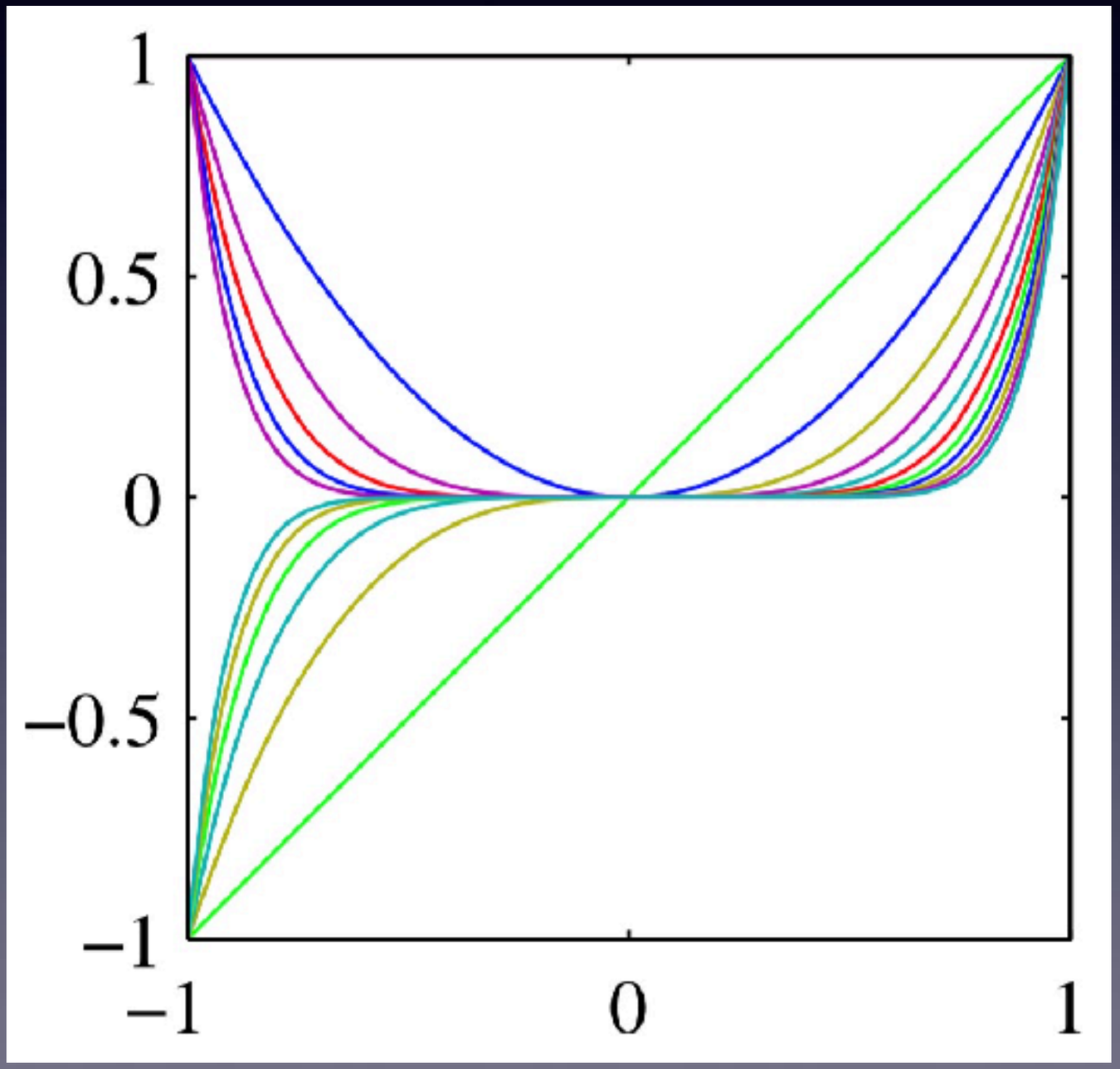
$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \boldsymbol{\phi}(\mathbf{x})$$

where $\mathbf{w} = (w_0, \ldots, w_{M-1})^T$, $\boldsymbol{\phi} = (\phi_0, \ldots, \phi_{M-1})^T$

- *Basis functions*: $\{\phi_j(\mathbf{x})\}$ act as "features" in ML.
  - Linear basis function: $\phi_j(x) = x_j$
  - Polynomial basis function: $\phi_j(\mathbf{x}) = x^j$
  - Gaussian basis function
  - Sigmoid basis function

Polynomial Basis Functions

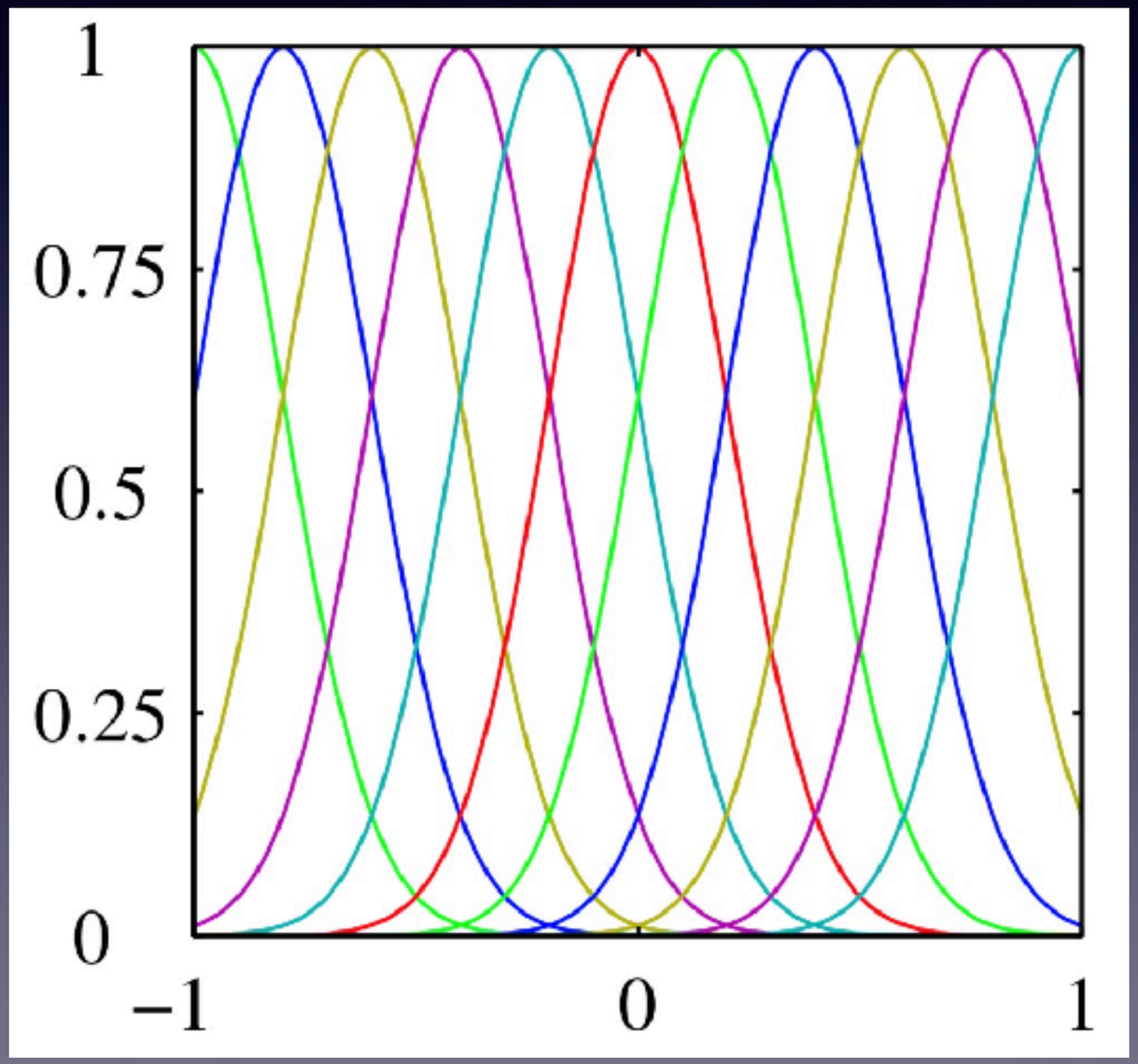
$$\phi_j(\mathbf{x}) = x^j$$



- Global functions of the input variable, s.t. changes in one region of input space affect all other regions.

Gaussian Basis Functions

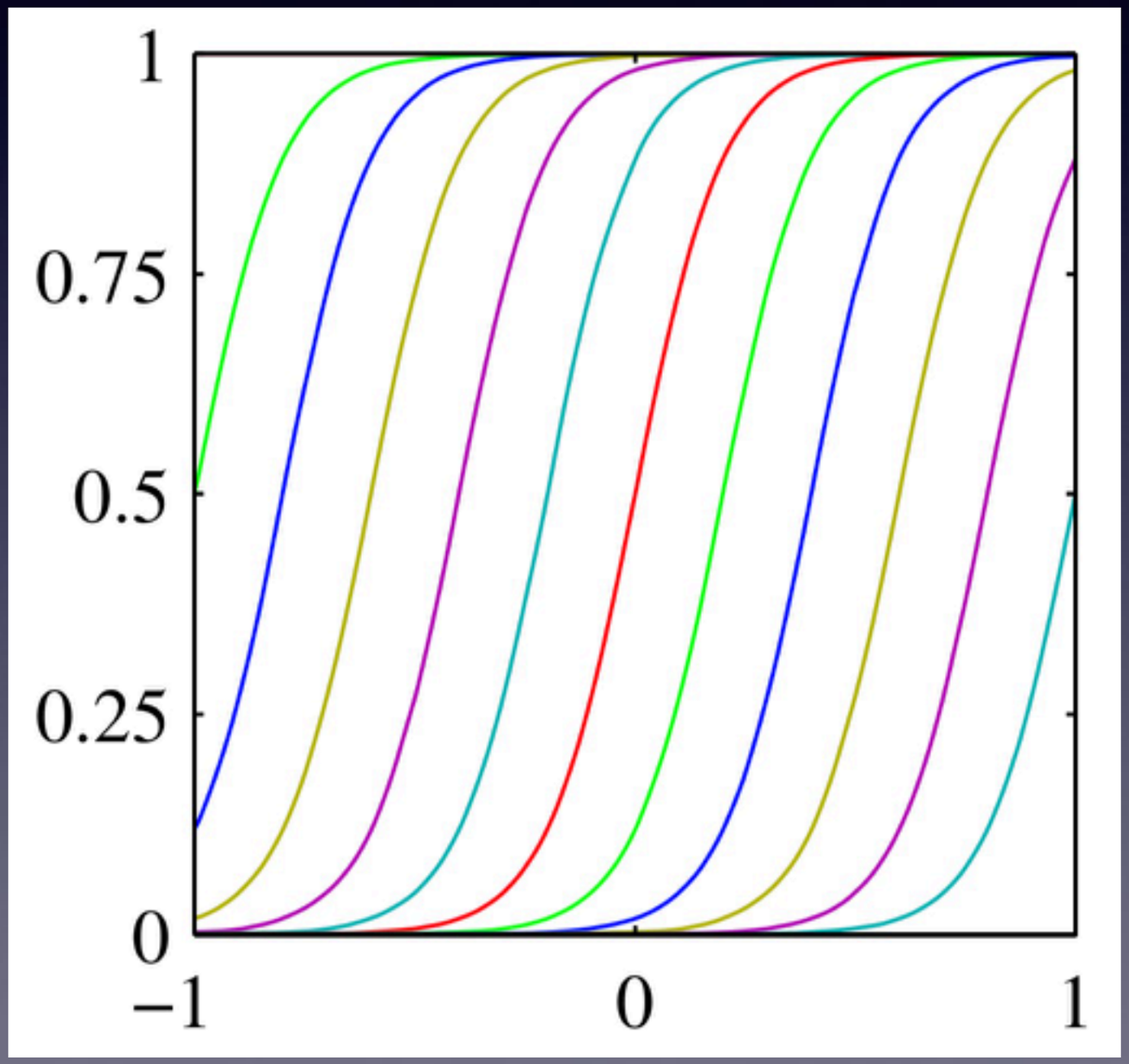
$$\phi_j(\mathbf{x}) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$



- Local functions, a small change in x only affect nearby basis functions.
- μ_j and s control the location and scale (width).

Sigmoidal Basis Functions

$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right) \text{ where } \sigma(a) = \frac{1}{1 + \exp(-a)}$$



- Local functions, a small change in x only affect nearby basis functions.
- μ_j and s control the location and scale (slope).

Regularized Least Squares

- Adding a regularization term to an error function:
  $E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$

- One of simplest forms of regularizer is sum-of-squares of the weight vector elements:
  $E_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$

- This type of *weight decay* regularizer (in ML), a.k.a., parameter shrinkage (in statistics) encourages weight values to decay towards zero, unless supported by the data.

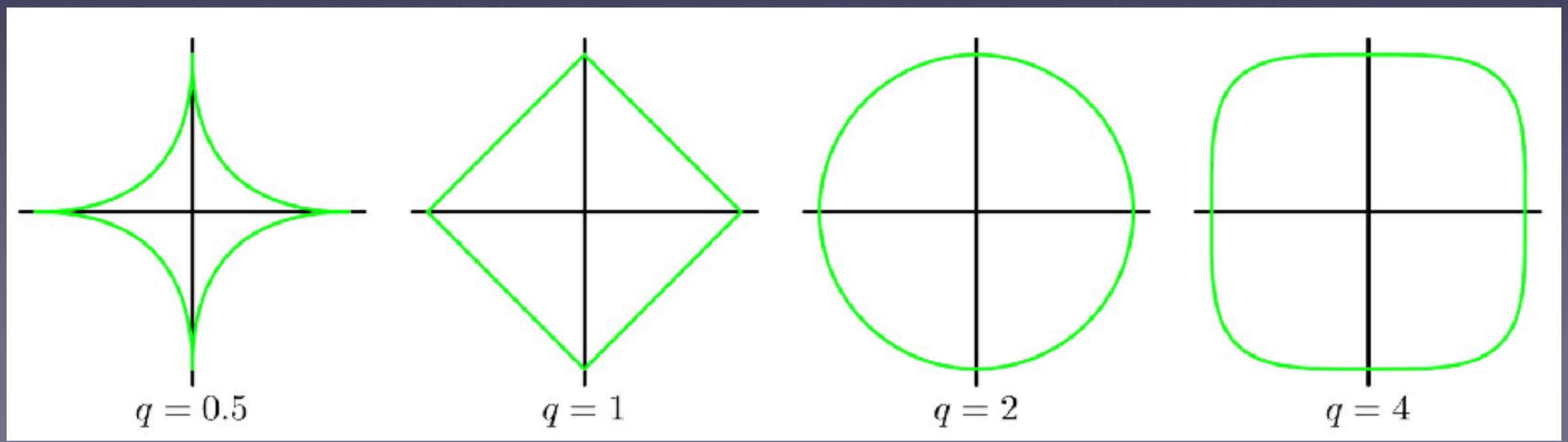
Regularized Least Squares

- A more general regularizer in the form of:

$$\frac{1}{2}\sum_{n=1}^{N}\left\{t_n - \mathbf{w}^{\mathrm{T}}\phi(x_n)\right\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

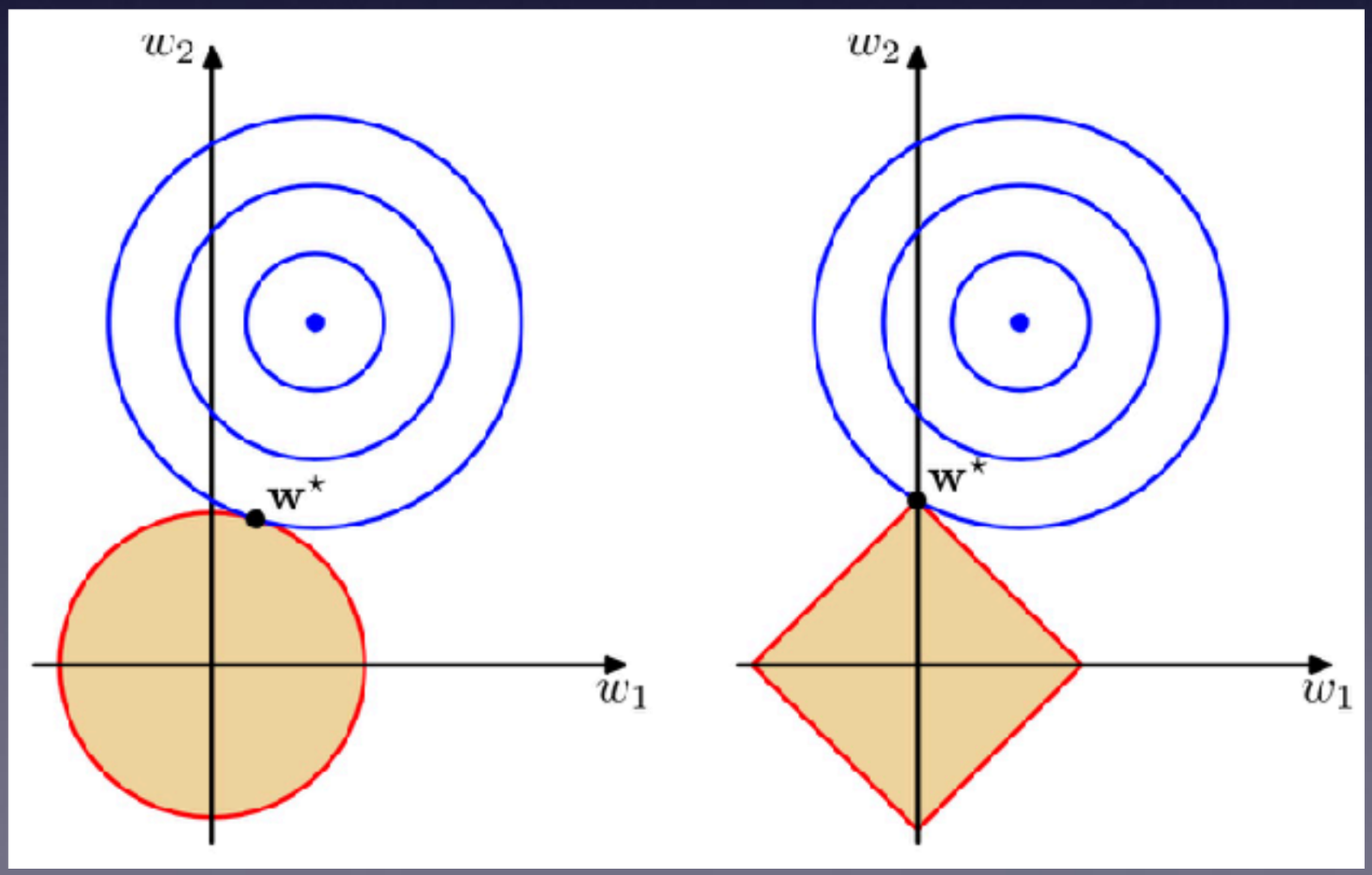
sum of squared error generalized regularizer,

- q=2 is the quadratic regularizer (last page).
- q=1 is known as *lasso* in statistics.

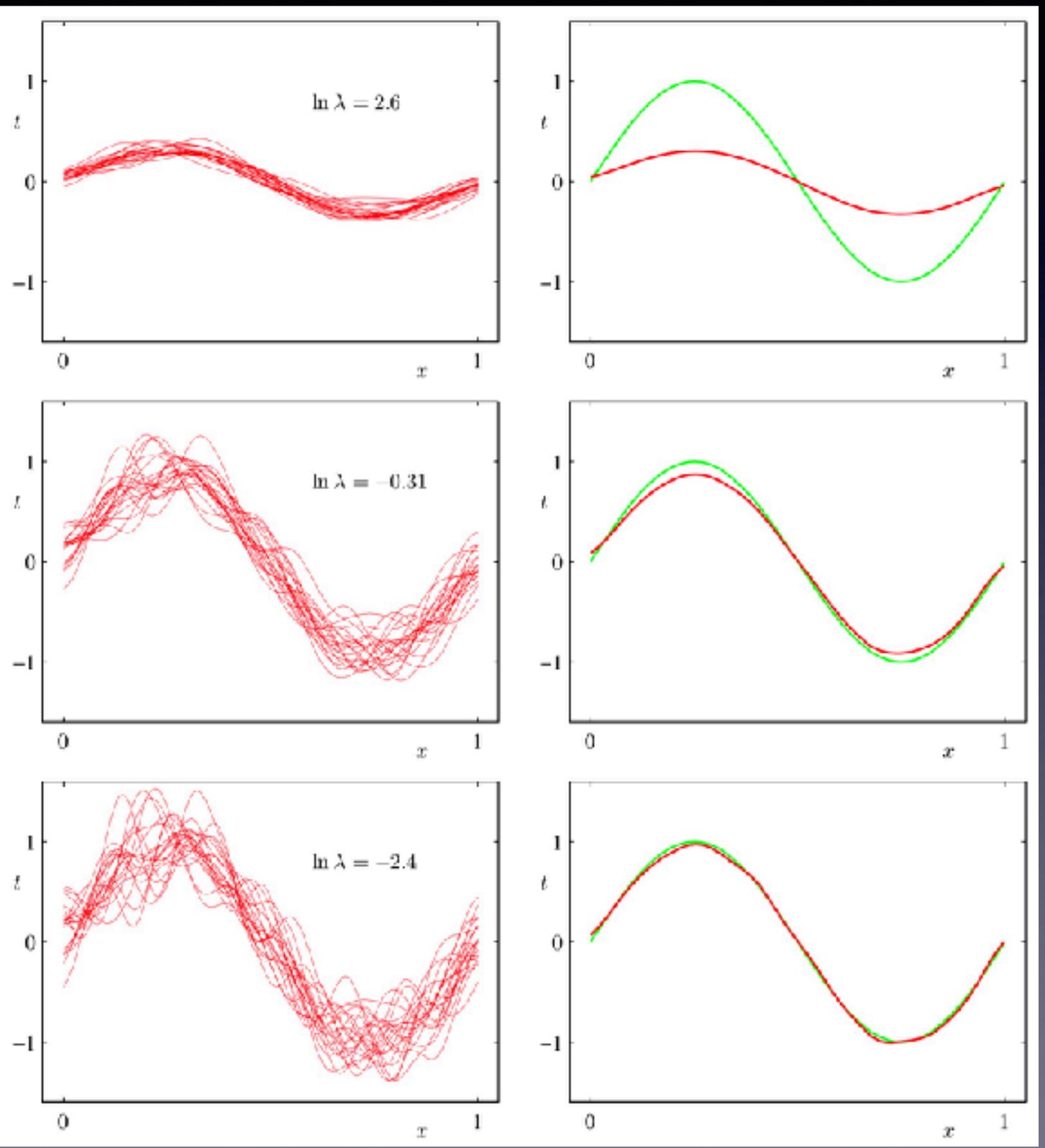


LASSO

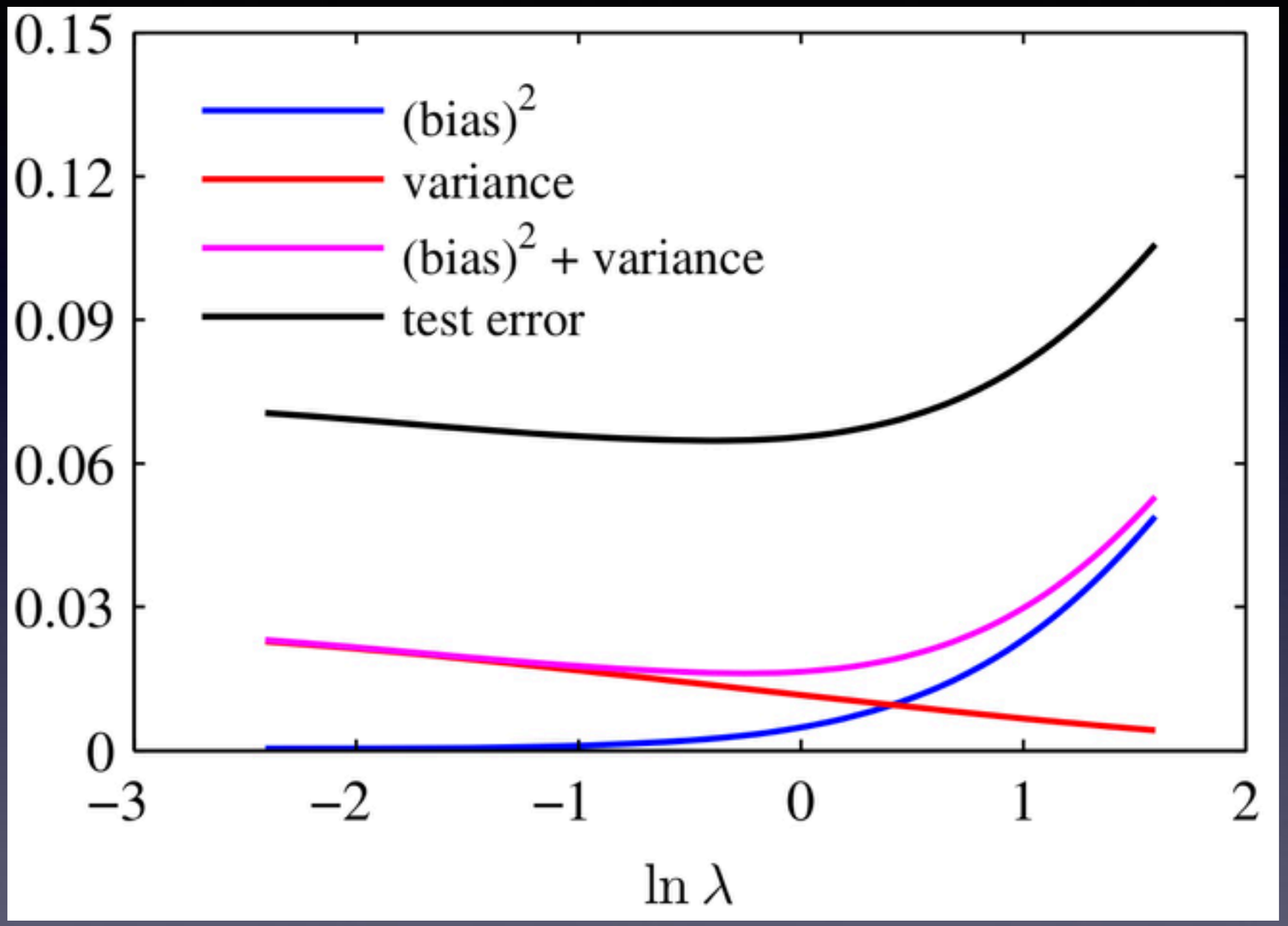
- LASSO: least absolute shrinkage and selection operator
- When λ is sufficiently large, some of the coefficients w_j are driven to zero, leading to a *sparse* model



The Bias-Variance Trade-off



The Bias-Variance Tradeoff



- Large values of λ: small variance but large bias
- Small values of λ: large variance, small bias