15.01.2025

Computer Vision (CS7.403)

Machine Learning Recap

Ravi Kiran

Makarand Tapaswi

Center for Visual Information Technology (CVIT)

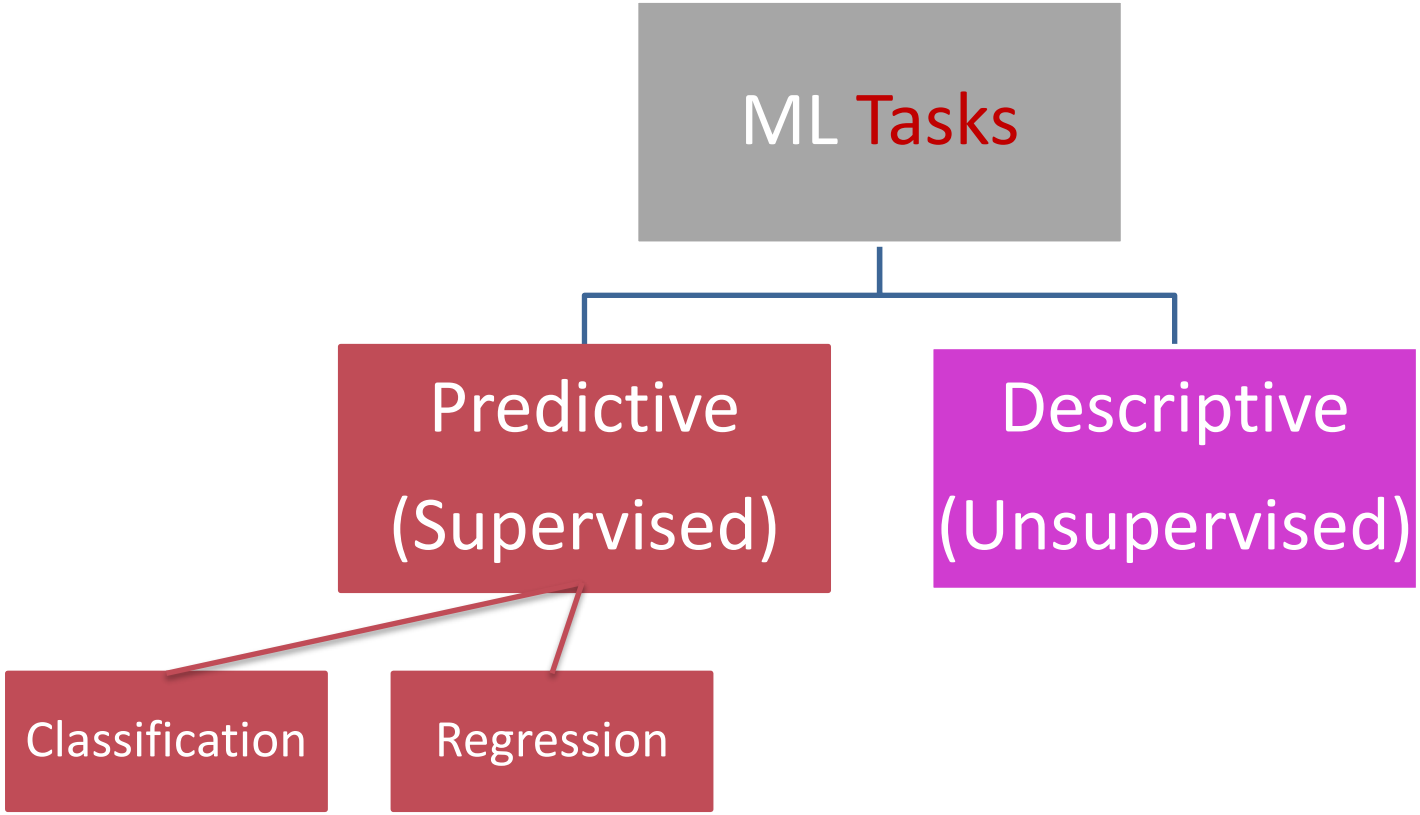
IIIT Hyderabad

Assignment Evals

Several students modified their code and reports prior to the evaluations and presented these altered versions during the assessment process.

You will get a straight 0 if you do any of the above

Some students whose code implementations were not functioning correctly included execution times, images, and data plots sourced from others in their reports.


ML Tasks
Predictive
(Supervised)
Descriptive
(Unsupervised)
Classification
Regression

ML::Tasks → Predictive → Classification

https://spectrum.ieee.org/ai-proves-trash-sorter-extraordinaire

Supervised Learning in Computer Vision

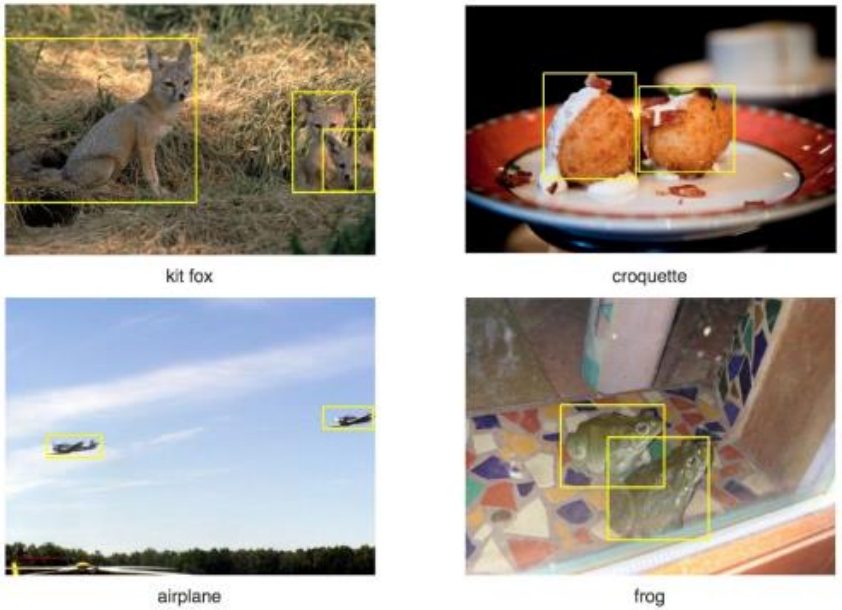
- Image Classification
  - x = raw pixels of the image, y = the main object

- There is too much info in raw data
- Relevant info is hidden
- Feature Extraction: Extract useful info (X) from raw data

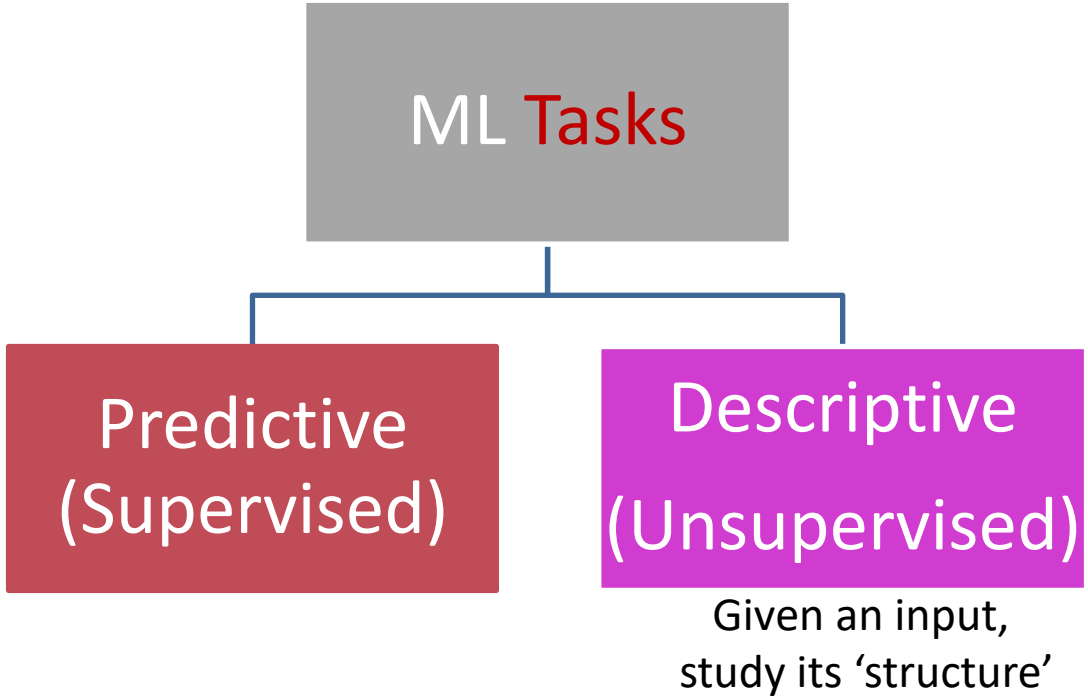
ImageNet Large Scale Visual Recognition Challenge. Russakovsky et al.'2015

Supervised Learning in Computer Vision

- Object localization and detection (Regression + Classification)
  - x = raw pixels of the image, y = the bounding boxes



ImageNet Large Scale Visual Recognition Challenge. Russakovsky et al.'2015

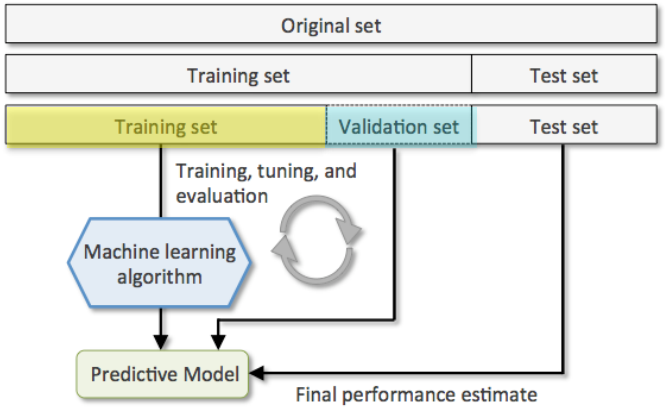

ML Tasks
Predictive (Supervised)
Descriptive (Unsupervised)
Given an input, study its ‘structure’

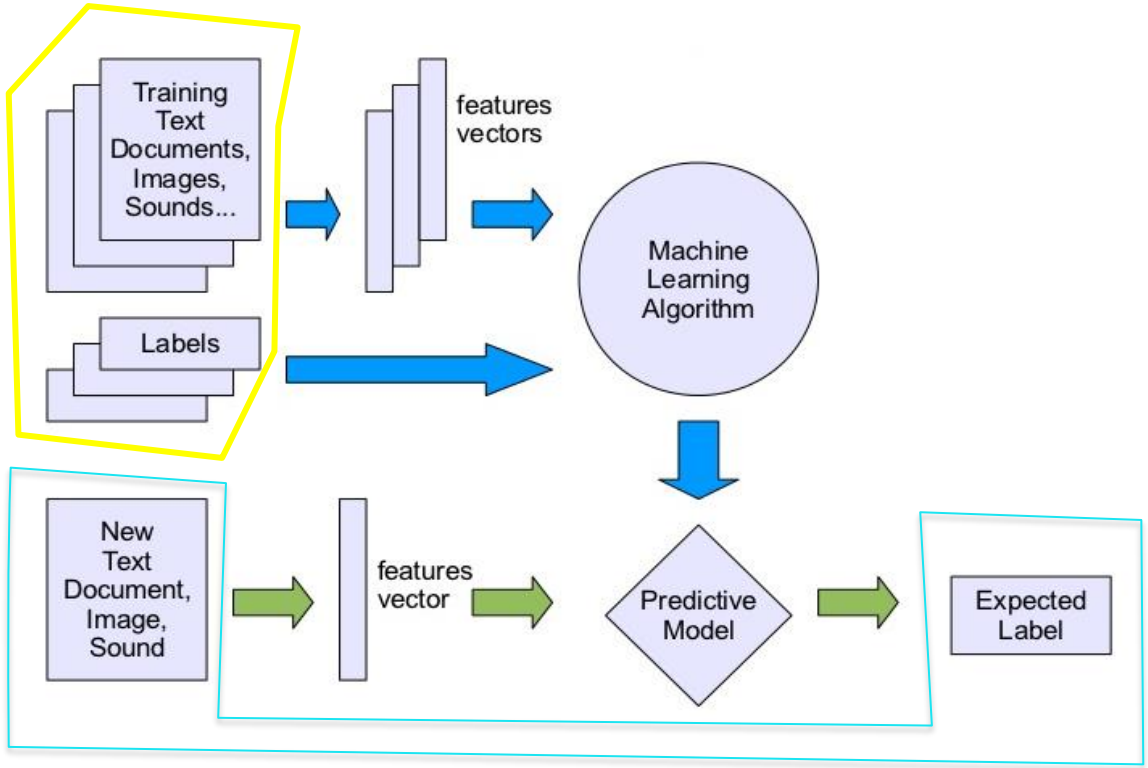
An interview analogy

1. Collect worked out problems (Q, S are both known)
2. Prepare on ALL the available problems.
3. Go for interview.

1. Collect worked out problems (Q,S are both known)
2. Randomly set aside a small number of problems.
3. Prepare on rest of the problems.
4. Take a mock interview containing all the 'set aside' problems.
5. Score answers and compare with solution.
6. Use mistakes to decide which topics to prepare better on.
7. Go for interview.

The Train-Validation-Test paradigm





Classification

Binary

Multi-class

Multi-label

Structure

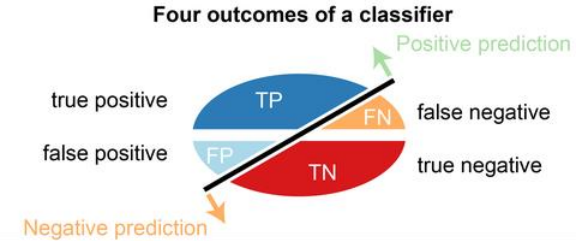
{0,1}

1-of-K

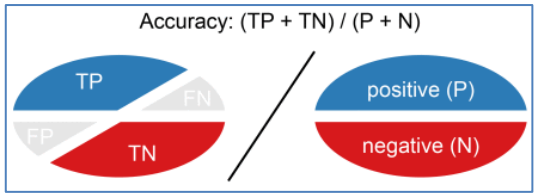
n-of-K

E.g. graph/sequence

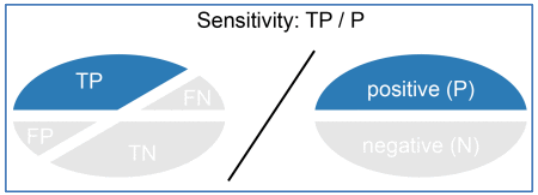
Summary of Measures [Two-class]



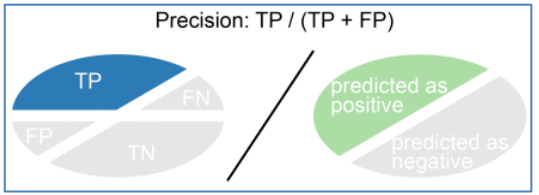
n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	



% of correct predictions



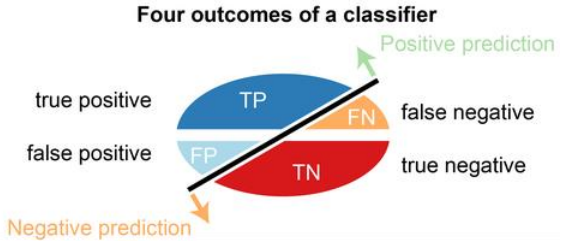
% of + class correctly predicted
[aka Recall / TPR]

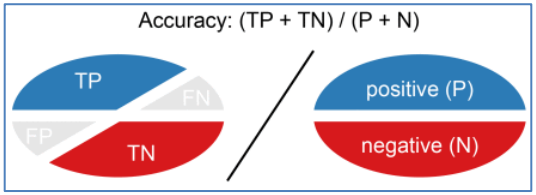


correct prediction of + class
[aka Precision]

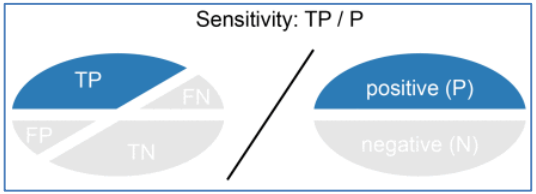
Accuracy vs Precision vs Recall

- Monitor **Precision** if a false positive carries higher cost.
- Monitor **Recall** if a false negative carries higher cost.

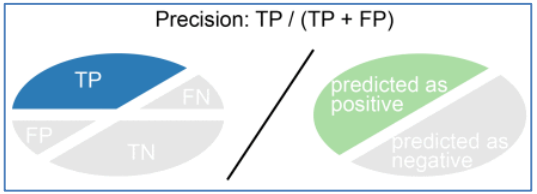




% of correct predictions



% of + class correctly predicted
[aka Recall / TPR]



correct prediction of + class
[aka Precision]

F1-Score

- What to do when one classifier has better Precision but worse Recall, while other classifier behaves exactly opposite?
  - F-measure (Information Retrieval)
    - $F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$

→ **F1 measure punishes extreme values more !**

→ **Definition of Recall and Precision have same numerator, different denominators. A sensible way to combine them is harmonic mean.**

Classification

Binary

Multi-class

Multi-label

Structure

{0,1}

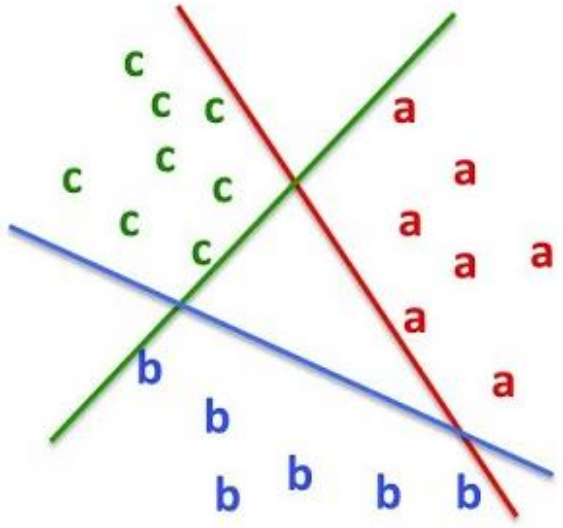
1-of-K

n-of-K

E.g. graph/sequence

How to use 2-class measures for multi-class ?

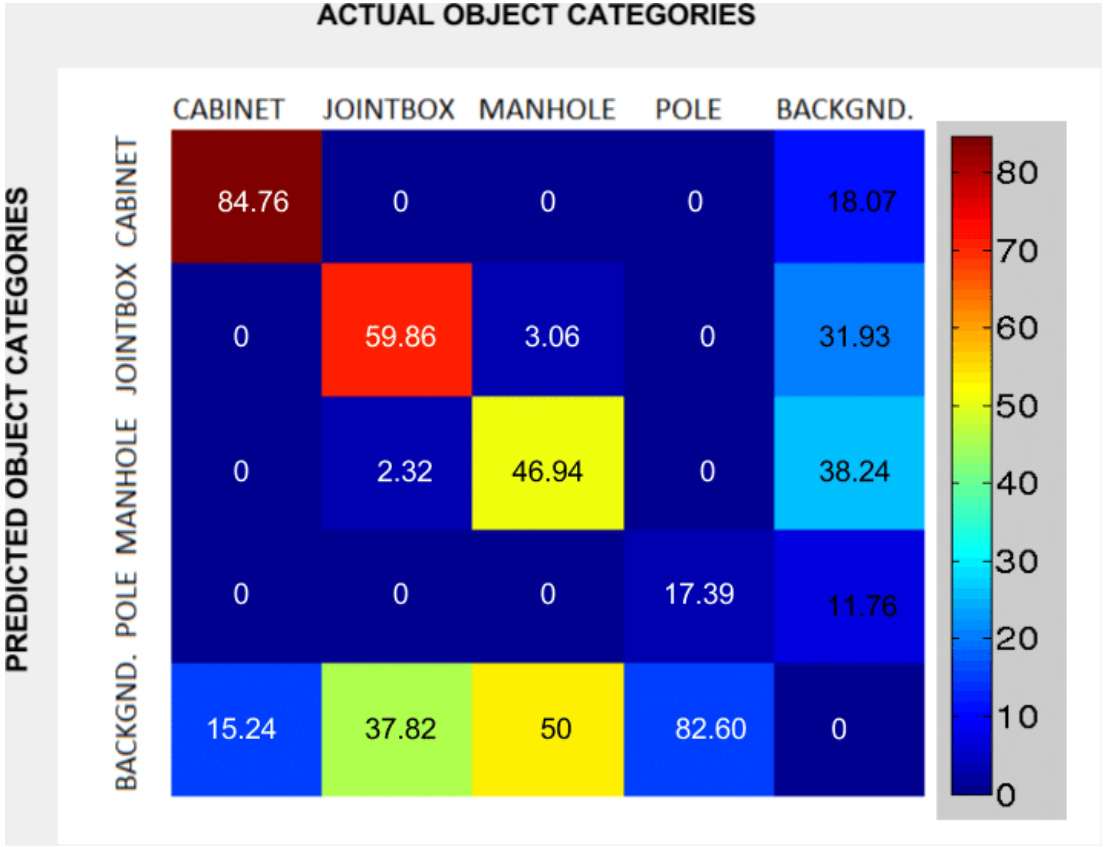
- Convert into 2-class problems !



- Average Precision, Recall etc.
- micro-F1, macro-F1

Avg. accuracy may not be very meaningful with imbalanced class label distribution

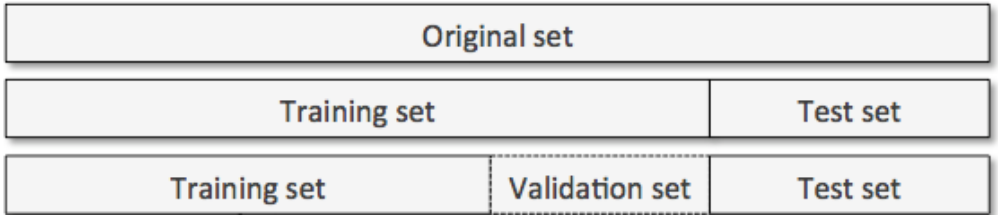
Multi-class Classification - Confusion matrix



- Reveals several performance aspects of the classifier:
  - Most confusing pairs
  - Least confusing pairs

https://www.researchgate.net/figure/Confusion-Matrix-Heat-Map-This-figure-depicts-that-Cabinets-jointboxes-and-manholes-have_fig5_315619273

Exam analogy: Did you prepare at least a little ?



- SANITY CHECK
- Compute <Performance Measure> (e.g. Accuracy) for TRAINING SET
- Verify it is “decent”

Classification

Binary

Multi-class

Multi-label

Structure

{0,1}

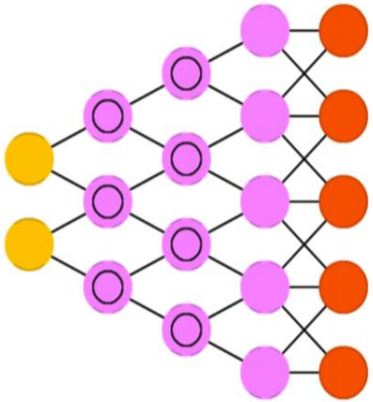
1-of-K

n-of-K

E.g. graph/sequence

Multi-Label Classification

X



	L	$\hat{y}$	y
black	•	0	0
white	•	0.95	1
orange	•	0.05	0
⋮			
small	•	0.8	1
medium	•	0.19	0
large	•	0.01	0

Multi Label Classifier

Multi-Label Classifier

https://pyimagesearch.com/2018/05/07/multi-label-classification-with-keras/

Metrics for Multi-Label Classification

- n is the number of examples.
- Y_i is the ground truth label assignment of the i^{th} example..
- x_i is the i^{th} example.
- $h(x_i)$ is the predicted labels for the i^{th} example.

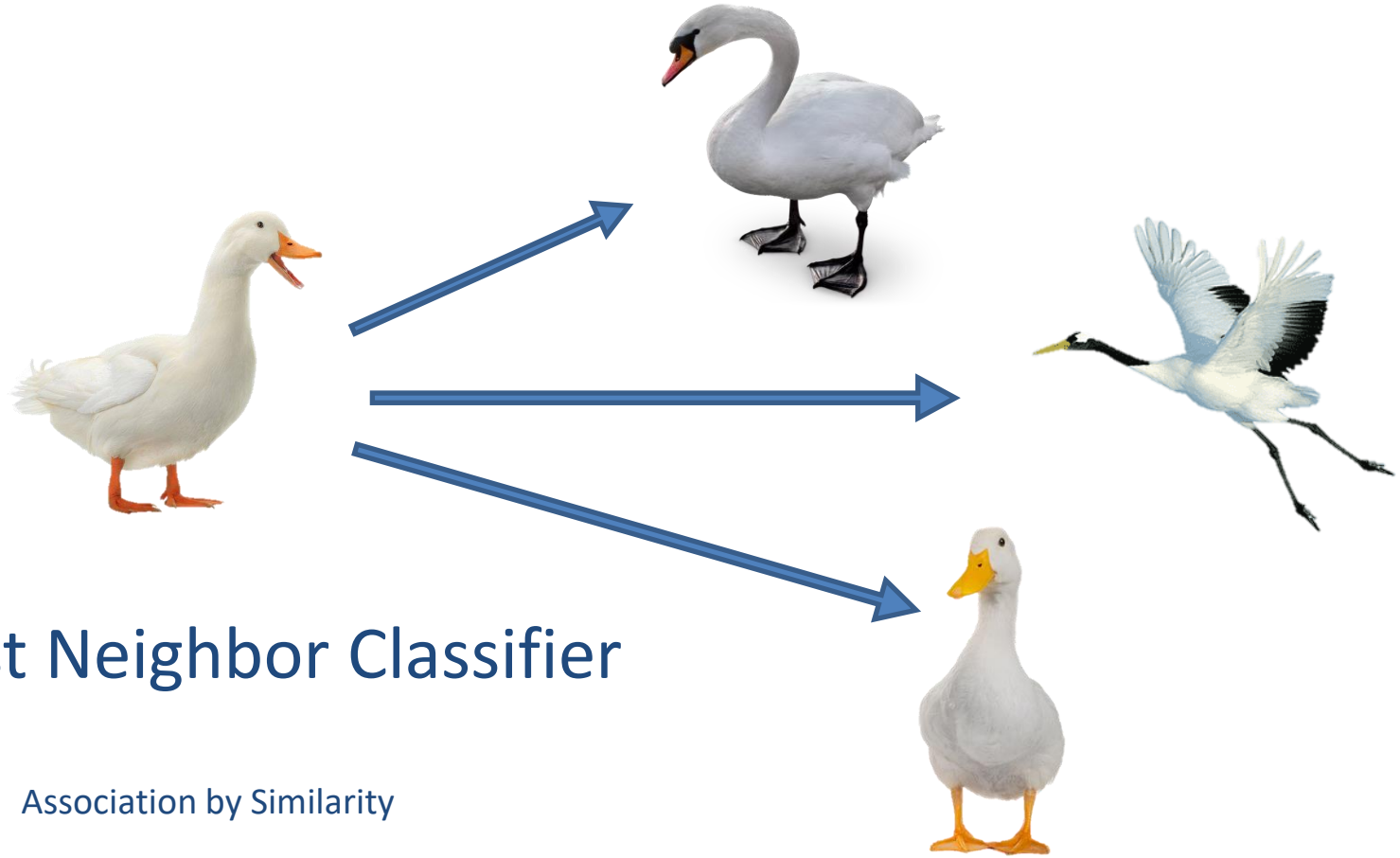
Precision = $\frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap h(x_i)|}{|h(x_i)|}$

What % of labels are predicted correctly ?

Recall = $\frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap h(x_i)|}{|Y_i|}$

What % of correct labels were predicted ?

Accuracy = Fraction of samples predicted correctly

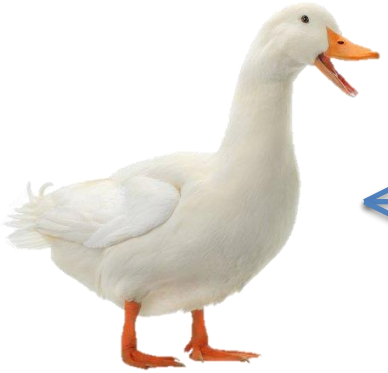


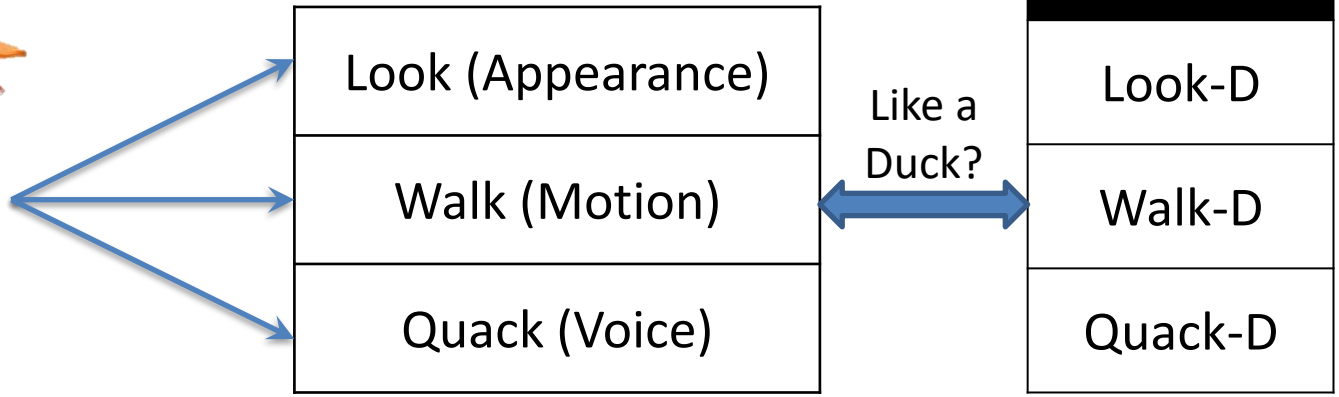
Nearest Neighbor Classifier

Association by Similarity

How do we compare?

If it looks like a duck, walks like a duck and quacks like a duck,





- Find distance to feature vectors of known classes

Nearest Neighbor Classifier

- Assign label of that sample which is nearest to the test sample

X_{test} :

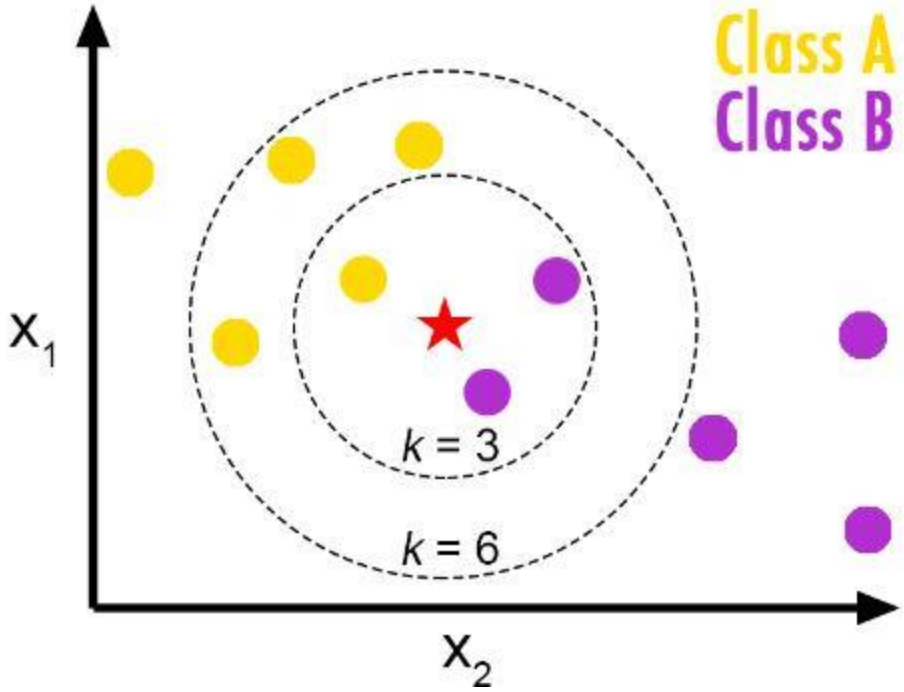
[30.9, 15.1, 1.32]

The test sample is a Duck

Training Samples

Distance	Feature Vector	Label
1.30	X_1 [32.1, 14.6, 1.42]	Duck
5.78	X_2 [25.3, 16.3, 2.11]	Swan
13.54	X_3 [42.2, 7.7, 0.38]	Crane
4.71	X_4 [26.7, 17.1, 2.04]	Swan
15.21	X_5 [44.1, 7.6, 0.32]	Crane
3.04	X_6 [31.4, 12.1, 1.29]	Duck
13.58	X_7 [41.9, 7.2, 0.35]	Crane

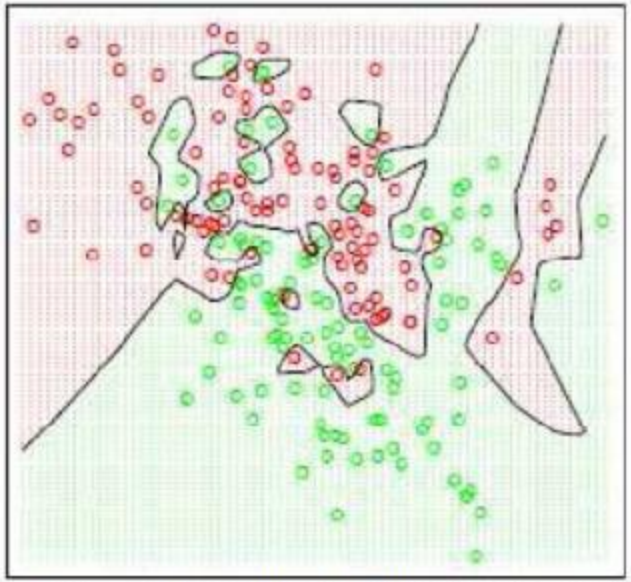
k-nearest neighbor classifier



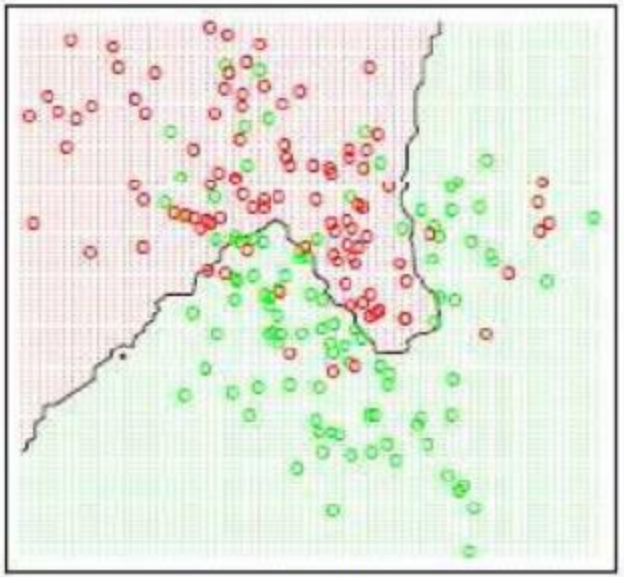
K is usually an odd number

Effect of K

K=1



K=15

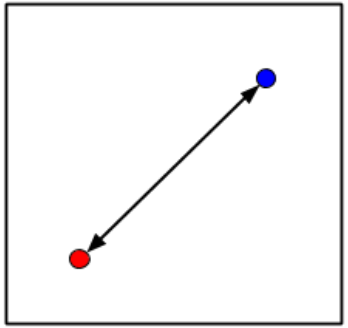


Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

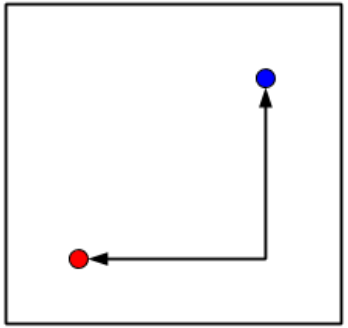
Larger k produces smoother boundary effect and can reduce the impact of class label noise.

Distance measures

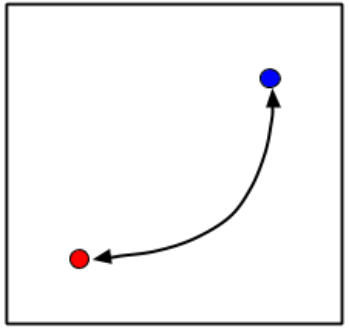
Euclidean



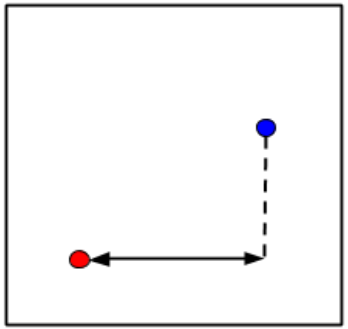
Manhattan



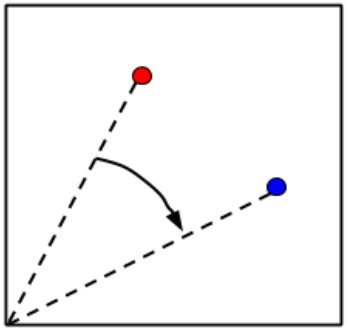
Minkowski



Chebychev



Cosine Similarity



Hamming

Some use cases for k-NN

- Decent performance when lots of data

- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

Some use cases for k-NN

- Problem: Where (e.g., which country or GPS location) was this picture taken?

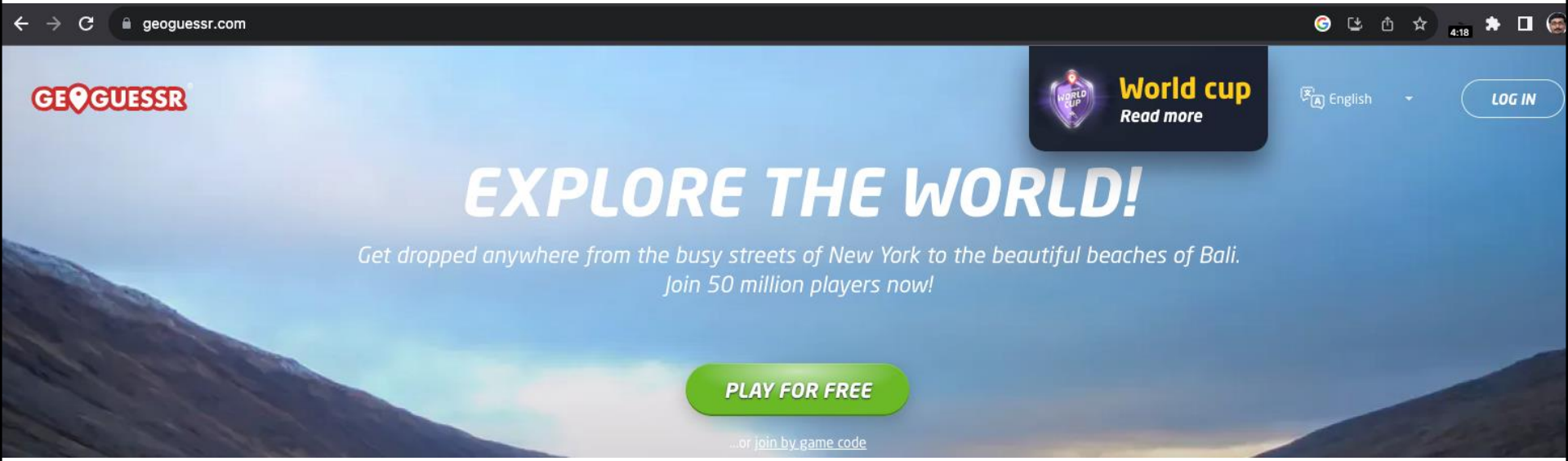
[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

Some use cases for k-NN

- Problem: Where (eg, which country or GPS location) was this picture taken?
  - Get 6M images from Flickr with gps info (dense sampling across world)
  - Represent each image with meaningful features
  - Do kNN (large k better, they use $k = 120$)!

[Paper: James Hays, Alexei A. Efros. im2gps: estimating geographic information from a single image. CVPR'08. Project page: http://graphics.cs.cmu.edu/projects/im2gps/]

GeoGuessr


geoguessr.com
GEOGUESSR
World cup
Read more
English
LOG IN
EXPLORE THE WORLD!
Get dropped anywhere from the busy streets of New York to the beautiful beaches of Bali.
Join 50 million players now!
PLAY FOR FREE
...or join by game code

WHO WOULD WIN?

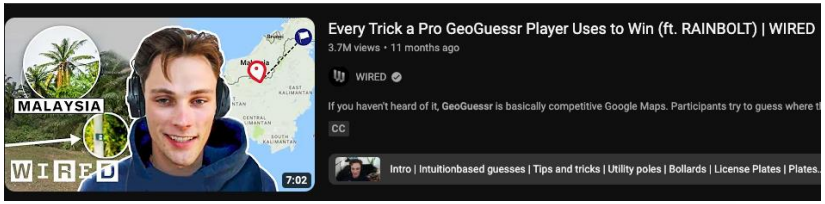
graphics.cs.cmu.edu/projects/im2gps/

IM2GPS: estimating geographic information from a single image

People

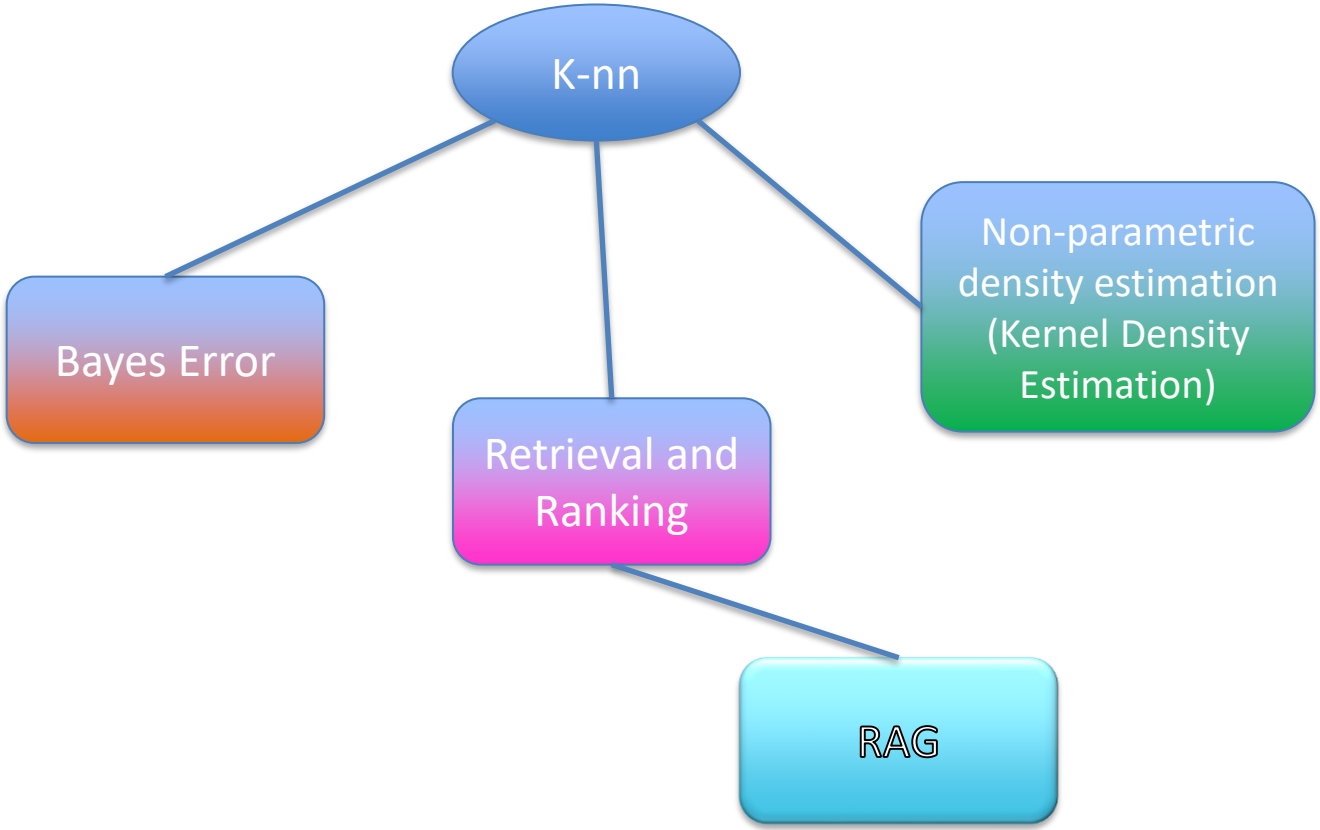
James Hays

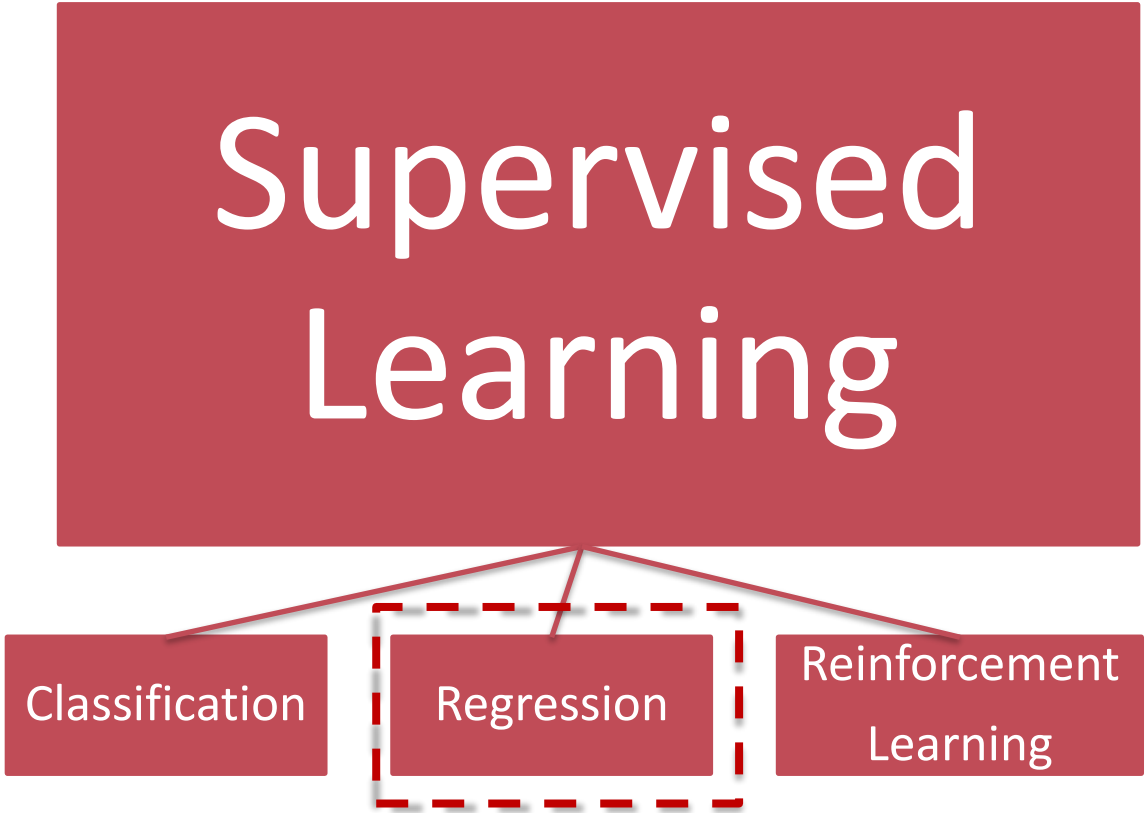
Alexei Efros



https://www.youtube.com/watch?v=0p5Eb4OSZCs

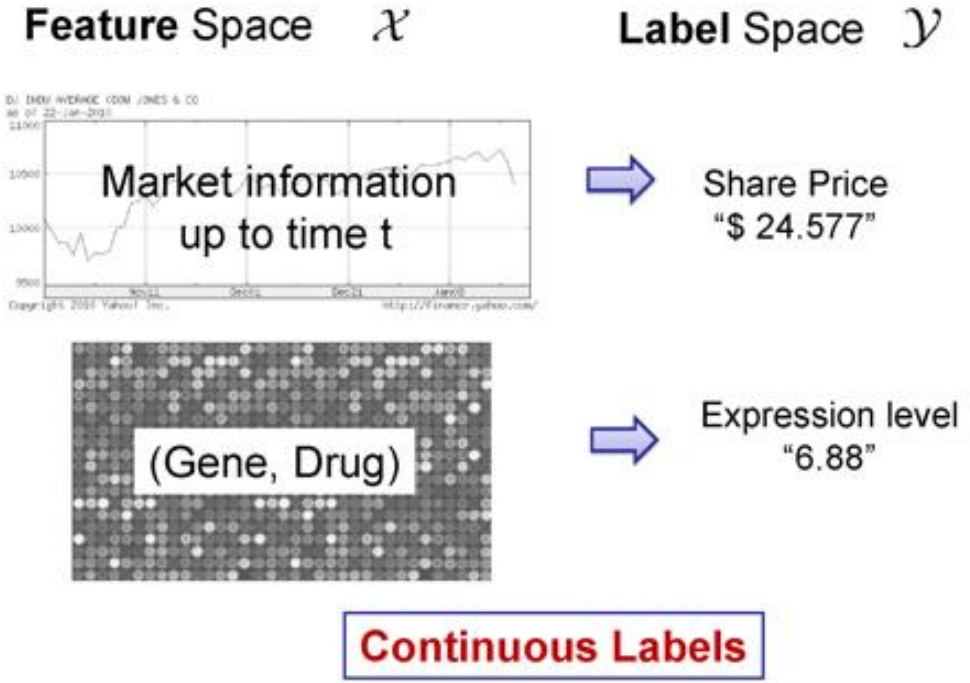
Related topics




Supervised Learning
Classification
Regression
Reinforcement Learning

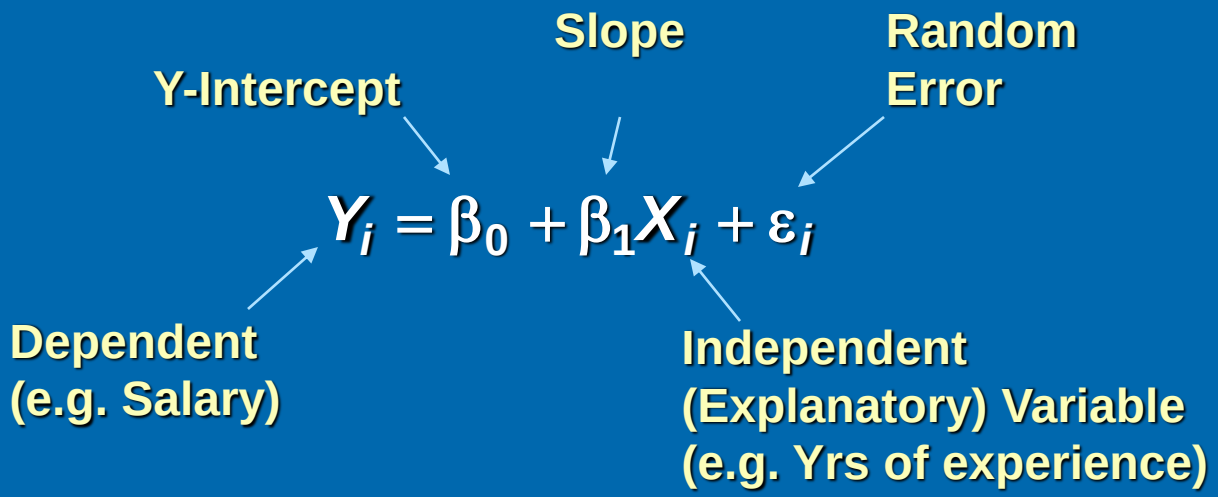
Regression model

- Regression model
  - Output is a real number



Linear Regression Model

- 1. Relationship Between Variables Is a Linear Function



Least Squares

- 1. ‘Best Fit’ Means Difference Between Actual Y Values & Predicted Y Values Are a Minimum. *But* Positive Differences Off-Set Negative. So square errors!

$$\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 = \sum_{i=1}^{n}\hat{\varepsilon}_i^2$$

- 2. LS Minimizes the Sum of the Squared Differences (errors) (SSE)

Coefficient Equations

- Prediction equation

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

- Sample slope

$$\hat{\beta}_1 = \frac{SS_{xy}}{SS_{xx}} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

- Sample Y - intercept

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Regression – Error measures

$$MAE = \frac{1}{n}\sum \left| y - \hat{y} \right|$$

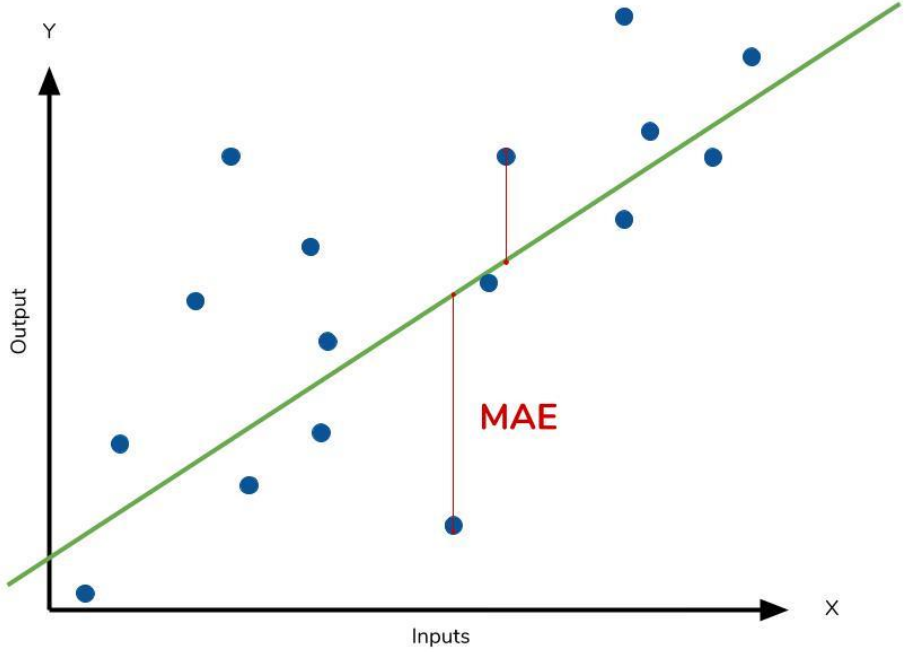
Divide by the total number of data points

Predicted output value

Actual output value

Sum of

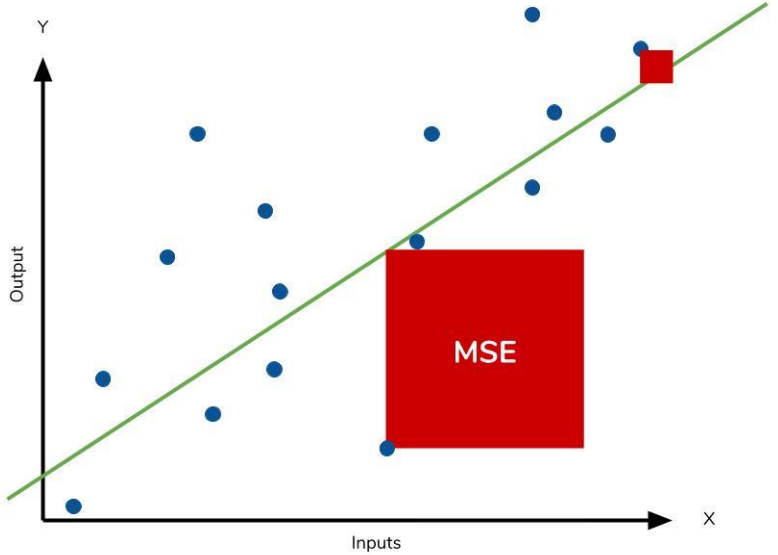
The absolute value of the residual



Regression – Error measures

$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \widehat{y} \right)}^{2}$$

The square of the difference between actual and predicted



Linear Regression – Matrix Form

Consider the model

$$Y = X\beta + \epsilon$$

where $Y = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix}$ $X = \begin{pmatrix} 1 & X_{11} & X_{12} & \dots & X_{1p} \\ 1 & X_{21} & X_{22} & \dots & X_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_{n1} & X_{n2} & \dots & X_{np} \end{pmatrix}$ $\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$ $\epsilon = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$

Then using matrix calculus we find that the least squares estimate for β is given by

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

Hence, the least squares regression line is $\hat{Y} = X\hat{\beta}$.

Gradient Descent

1. Initialize the parameters β to some random values.
2. Update the parameters using gradient descent rule

   $$\beta\ (t+1) = \beta(t) - \eta\ \nabla_{\beta} L\big(\beta(t)\big)$$

3. Repeat 2 until $|\nabla_w L\big(w(t)\big)|$ is close to 0

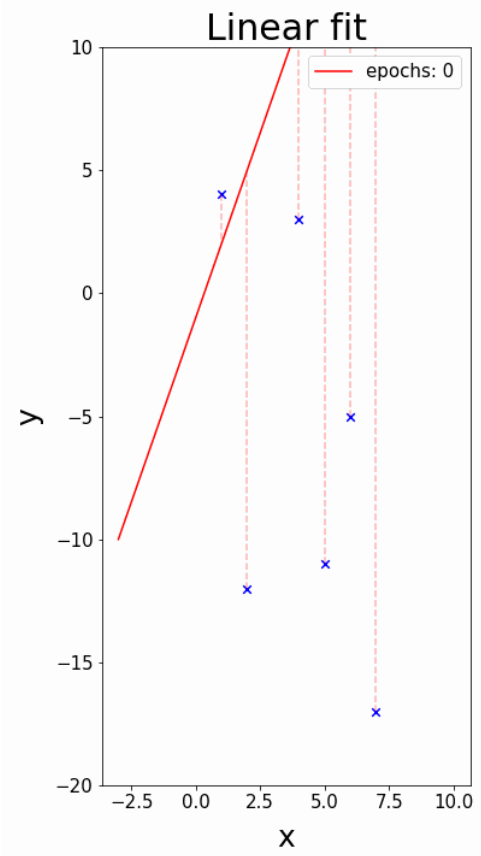
$$\mathbf{w} = [\beta_0 \ \ \beta_1]$$

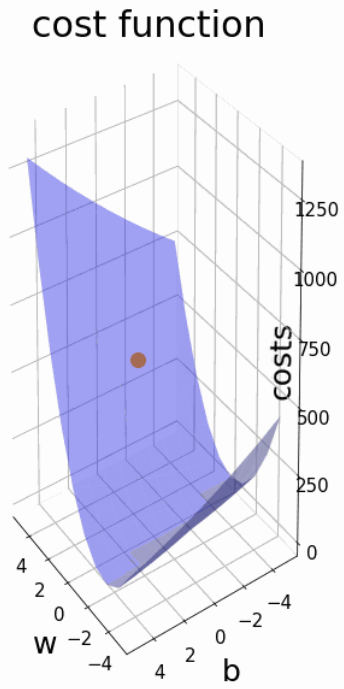
$$\hat{y} = \beta_0 + \beta_1 \mathbf{x}$$

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \mathcal{L}(\beta_0, \beta_1) \\ &= \frac{1}{n}\sum_{i=1}^{n}(\hat{y} - y)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(\beta_0 + \beta_1 \mathbf{x} - y)^2\end{aligned}$$

$$\implies \frac{\partial \mathcal{L}}{\partial \beta_0} = \frac{1}{n}\sum_{i=1}^{n} 2(\beta_0 + \beta_1 \mathbf{x} - y)$$

$$\implies \frac{\partial \mathcal{L}}{\partial \beta_1} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}(\beta_0 + \beta_1 \mathbf{x} - y)$$





Gradient Descent

1. Initialize the parameters w to some random values.
2. Update the parameters using gradient descent rule
   $w(t+1) = w(t) - \eta \nabla_w L\big(w(t)\big)$
3. Repeat 2 until $|\nabla_w L\big(w(t)\big)|$ is close to 0

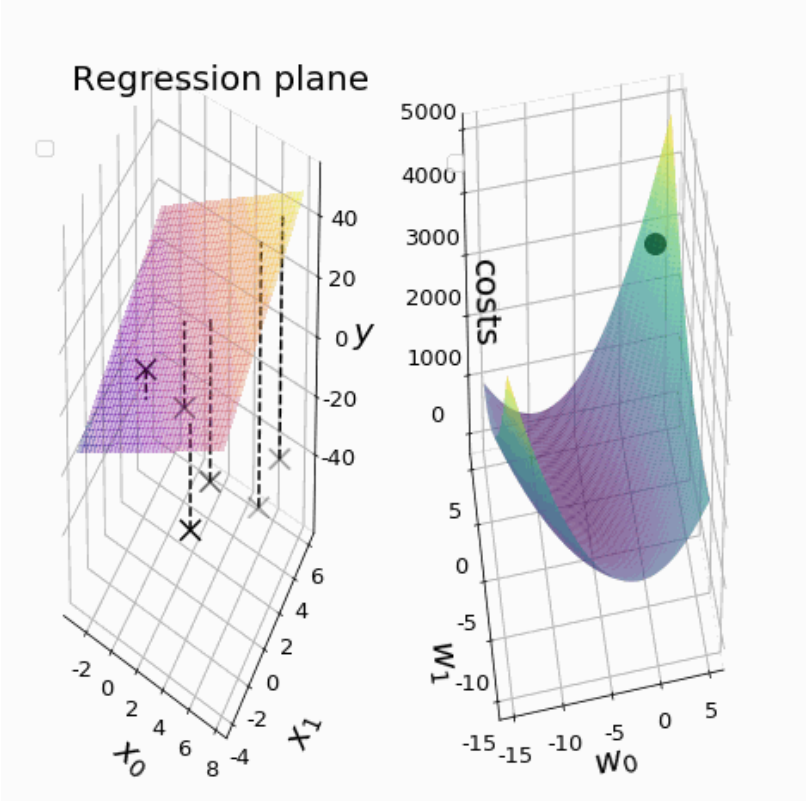
$$\mathbf{w} = [\beta_0 \;\; \beta_1]$$

$$\hat{y} = \beta_0 + \beta_1 \mathbf{x}$$

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \mathcal{L}(\beta_0, \beta_1) \\ &= \frac{1}{n}\sum_{i=1}^{n}(\hat{y} - y)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(\beta_0 + \beta_1\mathbf{x} - y)^2\end{aligned}$$

$$\implies \frac{\partial \mathcal{L}}{\partial \beta_0} = \frac{1}{n}\sum_{i=1}^{n} 2(\beta_0 + \beta_1\mathbf{x} - y)$$

$$\implies \frac{\partial \mathcal{L}}{\partial \beta_1} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}(\beta_0 + \beta_1\mathbf{x} - y)$$



Linear Regression

- Linear Regression ➔ Linear in coefficients and NOT variables

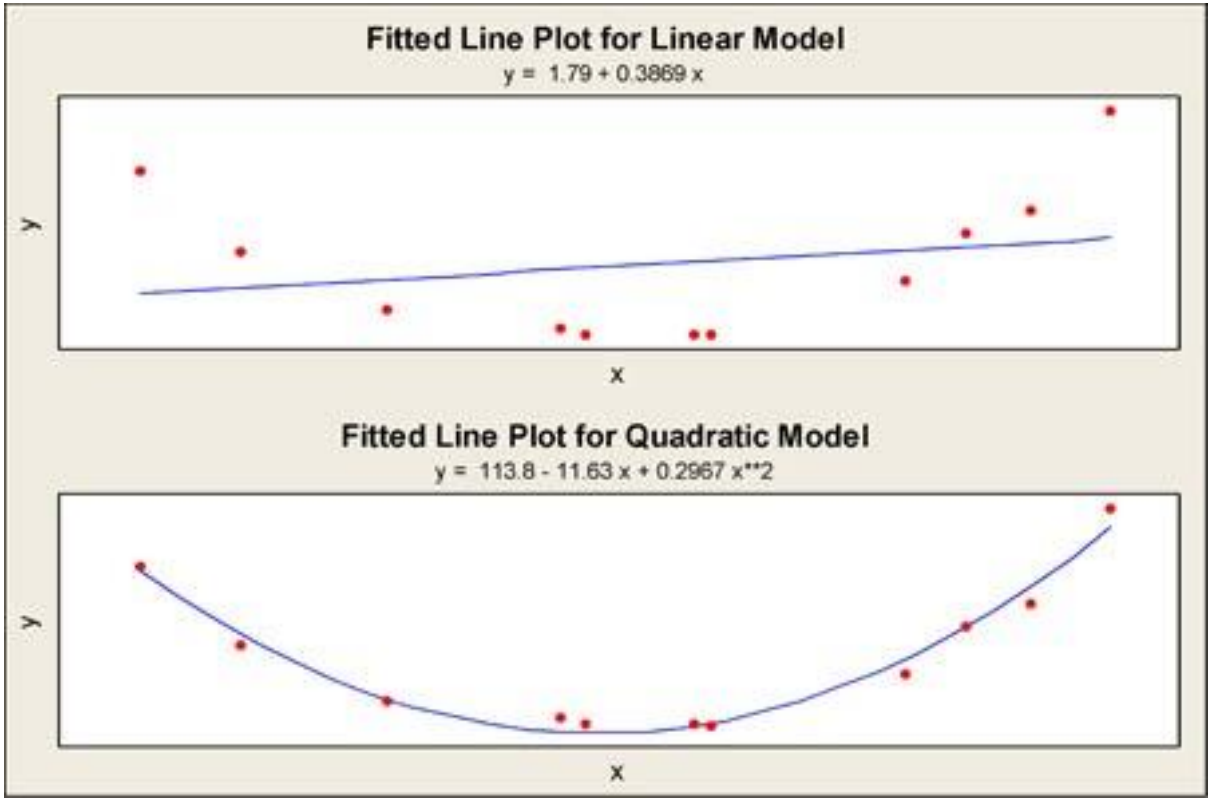
  - A second-order model (quadratic model):

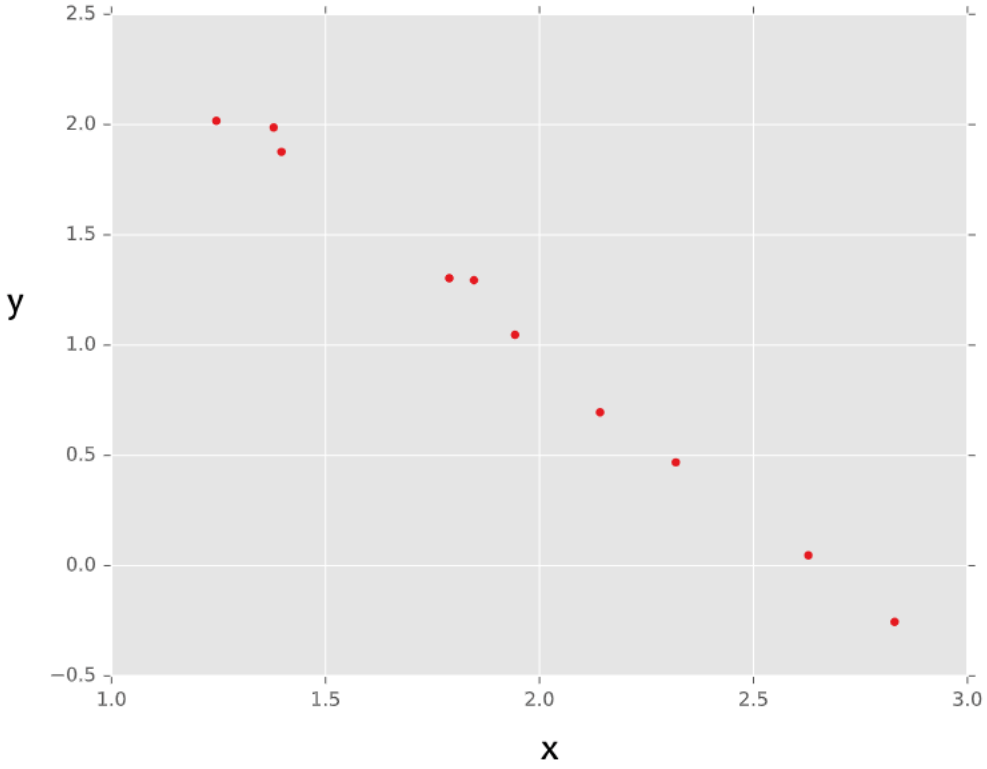
$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$$

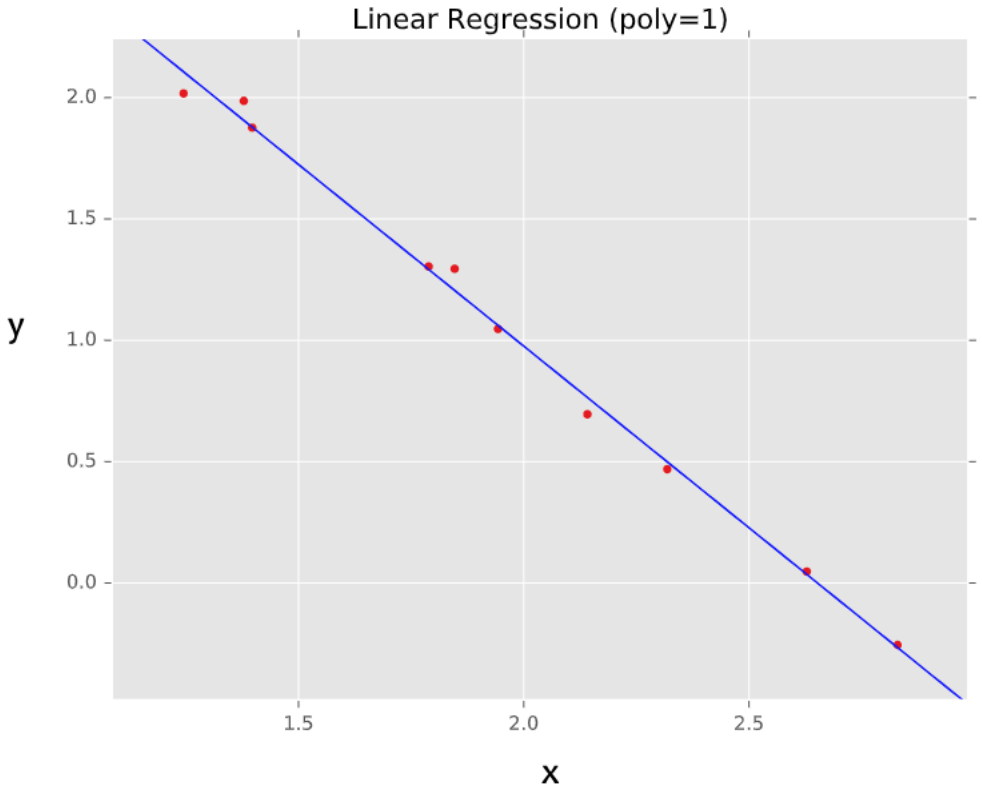
    - β_1: Linear effect parameter.
    - β_2: Quadratic effect parameter.

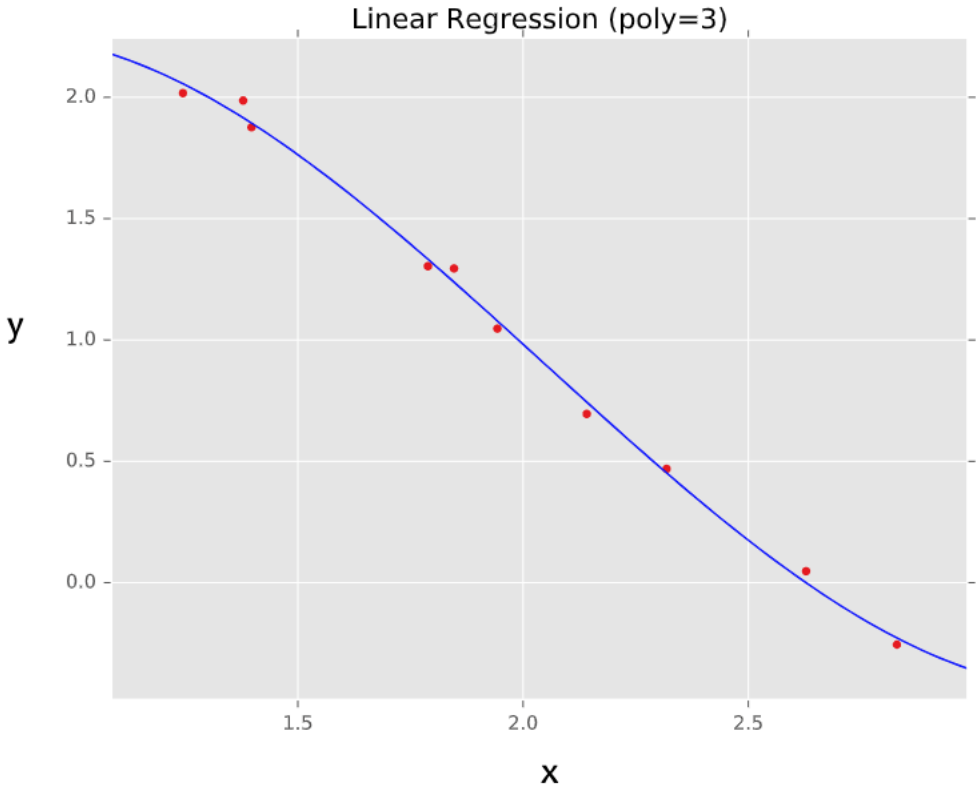
  kth order polynomial model in one variable

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_k x^k + \epsilon$$

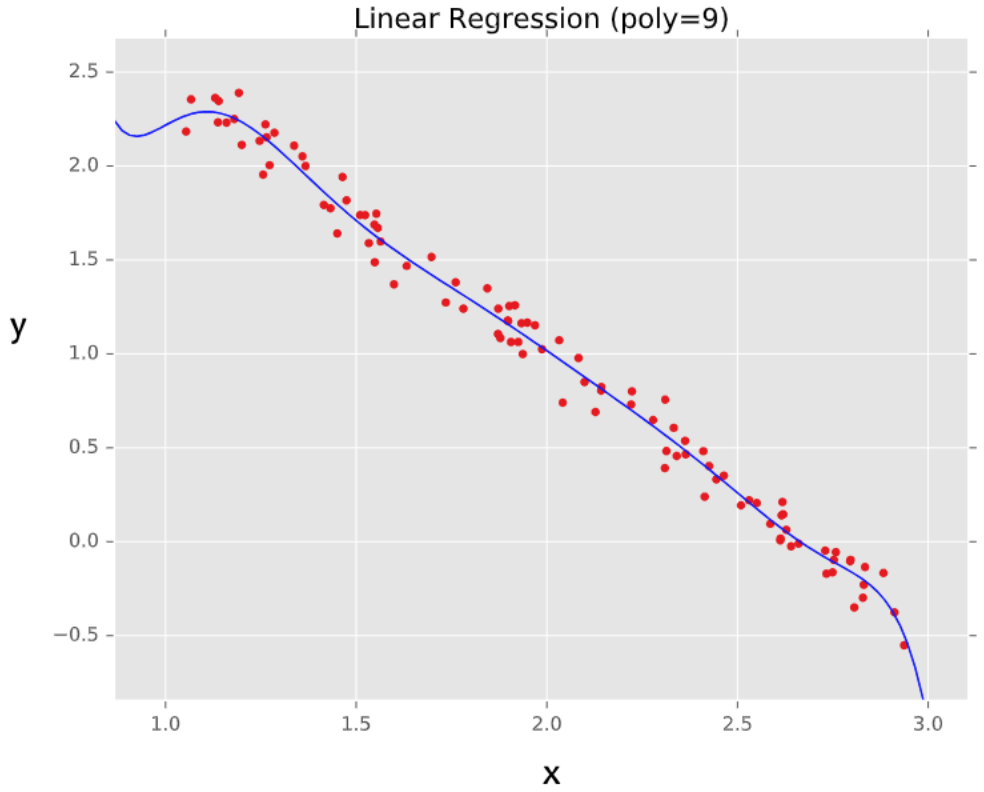

Fitted Line Plot for Linear Model
y = 1.79 + 0.3869 x
y
x
Fitted Line Plot for Quadratic Model
y = 113.8 - 11.63 x + 0.2967 x**2
y
x


2.5
2.0
1.5
1.0
0.5
0.0
−0.5
y
1.0
1.5
2.0
2.5
3.0
x


Linear Regression (poly=1)
y
x
2.0
1.5
1.0
0.5
0.0
1.5
2.0
2.5


Linear Regression (poly=3)
y
2.0
1.5
1.0
0.5
0.0
1.5
2.0
2.5
x

Test Time



Overfitting

The example shows:

- perfect fit on **in sample (training)** data

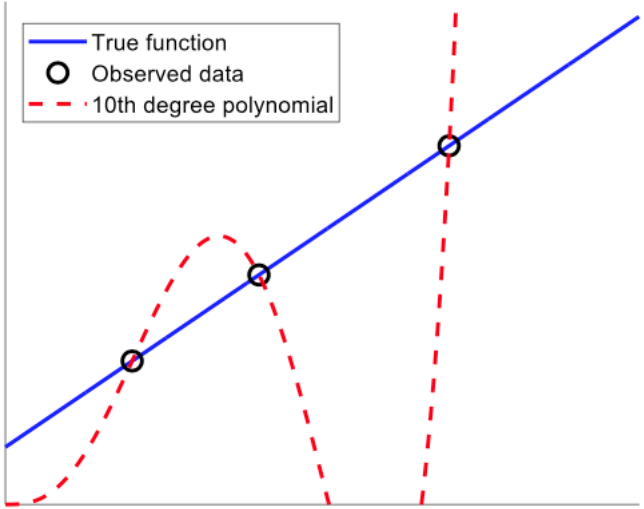
↓

$E_{in} = 0$

- low fit on **out of sample (test)** data

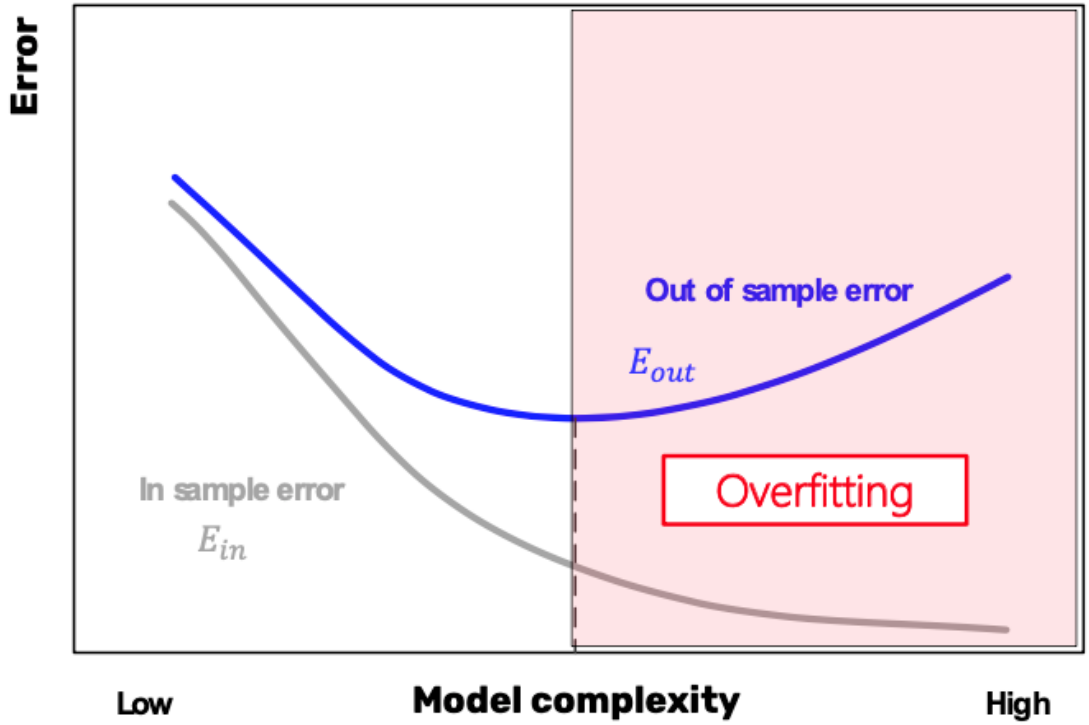
↓

E_{out} **huge**



Overfitting vs. model complexity

We talk of **overfitting** when decreasing E_{in} leads to increasing E_{out}

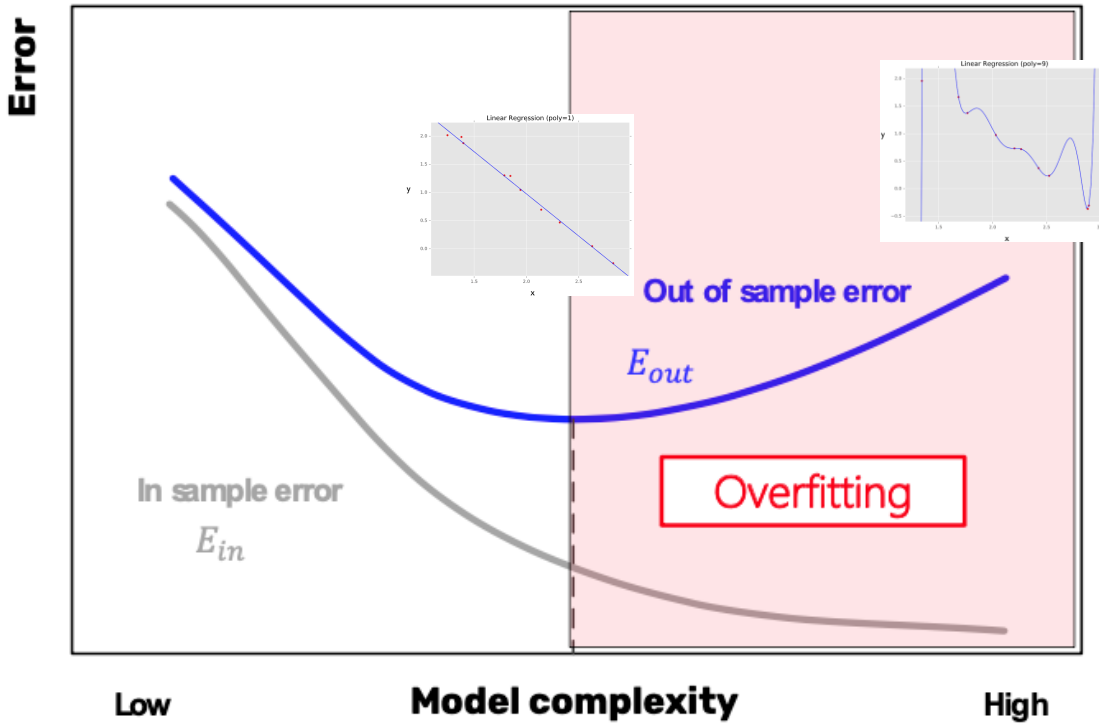


Overfitting vs. model complexity

We talk of **overfitting** when decreasing E_{in} leads to increasing E_{out}

Major source of failure for machine learning systems

Overfitting leads to bad generalization



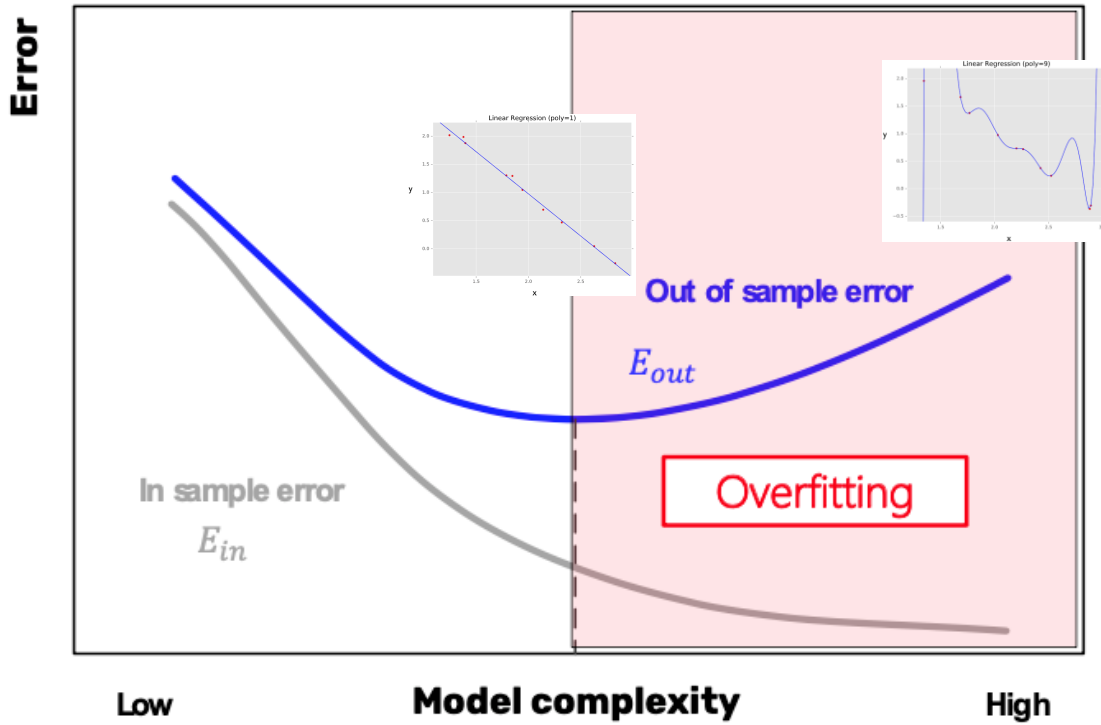
Overfitting vs. model complexity

We talk of **overfitting** when decreasing E_{in} leads to increasing E_{out}

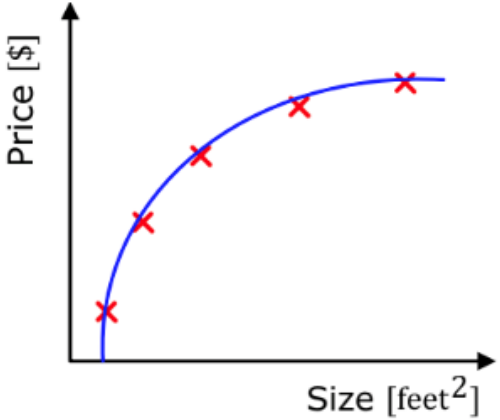
Major source of failure for machine learning systems

Overfitting leads to bad generalization

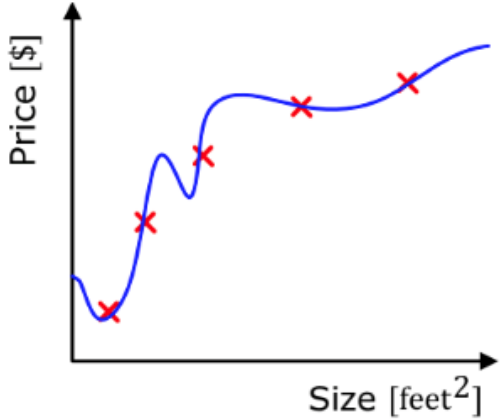
A model can exhibit bad generalization even if it does not overfit



A cure for overfitting

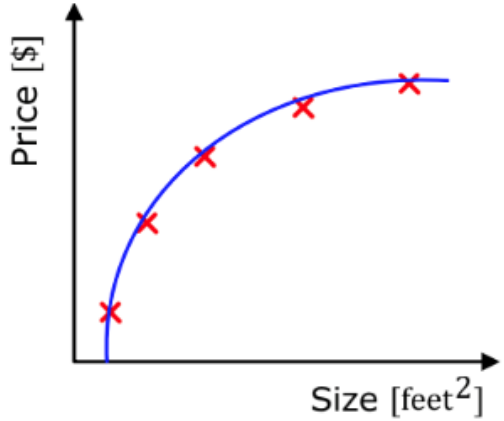


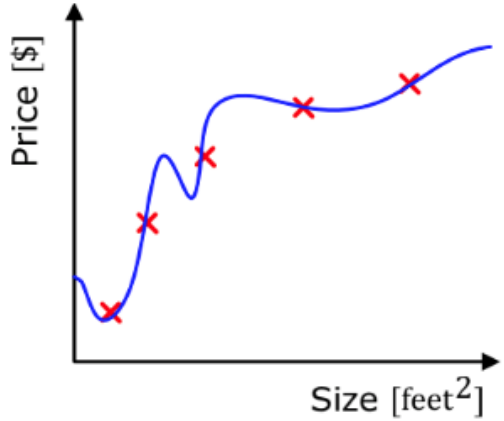
$\mathcal{H}_1$: $y = \theta_0 + \theta_1\varphi + \theta_2\varphi^2$



$\mathcal{H}_2$: $y = \theta_0 + \theta_1\varphi + \theta_2\varphi^2 + \theta_3\varphi^3 + \theta_4\varphi^4$

A cure for overfitting





$\mathcal{H}_1$: $y = \theta_0 + \theta_1\varphi + \theta_2\varphi^2$

$\mathcal{H}_2$: $y = \theta_0 + \theta_1\varphi + \theta_2\varphi^2 + \theta_3\varphi^3 + \theta_4\varphi^4$

We can «recover» the model $\mathcal{H}_1$ from the model $\mathcal{H}_2$ by **imposing** $\theta_3 = \theta_4 = 0$

This can be done by minimizing, along with $J(\boldsymbol{\theta})$, also the value of the parameters θ_3, θ_4

Regularization

$h(\cdot)$ is some function that represents our model

More generally, instead of minimizing the in-sample error E_{in} (i.e. the cost function $J(\boldsymbol{\theta})$), minimize the **augmented error:**

For simplicity, suppose $J(\boldsymbol{\theta})$ as squared error function

$$E_{aug}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(y(i) - h(\boldsymbol{\varphi}(i);\boldsymbol{\theta})\right)^2 + \lambda \cdot \sum_{j=0}^{d-1}\left(\theta_j\right)^2$$

- The term $\Omega = \sum_{j=0}^{d-1}\left(\theta_j\right)^2$ is called **regularizer**

- The term λ (**regularization hyper-parameter**) weights the importance of minimizing $J(\boldsymbol{\theta})$, with respect to minimizing Ω

Regularization

$h(\cdot)$ is some function that represents our model

More generally, instead of minimizing the in-sample error E_{in} (i.e. the cost function $J(\boldsymbol{\theta})$), minimize the **augmented error:**

For simplicity, suppose $J(\boldsymbol{\theta})$ as squared error function

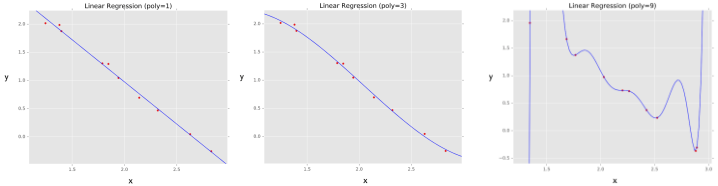
$$E_{aug}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(y(i) - h(\boldsymbol{\varphi}(i);\boldsymbol{\theta})\right)^2 + \lambda \cdot \sum_{j=0}^{d-1}\left(\theta_j\right)^2$$

- The term $\Omega = \sum_{j=0}^{d-1}\left(\theta_j\right)^2$ is called **regularizer**

- The term λ (**regularization hyper-parameter**) weights the importance of minimizing $J(\boldsymbol{\theta})$, with respect to minimizing Ω

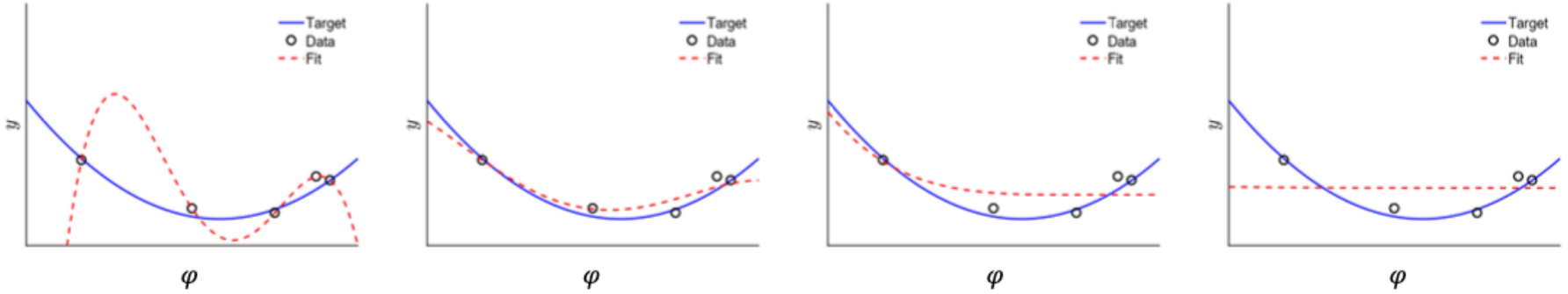
Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43



Effect of λ

λ_1 | $\lambda_2 > \lambda_1$ | $\lambda_3 > \lambda_2$ | $\lambda_4 > \lambda_3$



Overfit → → **Underfit**

$$E_{aug}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(y(i) - h(\boldsymbol{\varphi}(i);\boldsymbol{\theta})\right)^2 + \lambda \cdot \sum_{j=0}^{d-1}\left(\theta_j\right)^2$$

Choice of the regularizer

There are many choices of possible regularizers. The most used ones are:

- L_2 **regularizer:** also called **Ridge** regression $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1}(\theta_j)^2 = \boldsymbol{\theta}^\top \boldsymbol{\theta} = \|\boldsymbol{\theta}\|_2^2$
- L_1 **regularizer:** also called **Lasso** regression $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1}|\theta_j| = \|\boldsymbol{\theta}\|_1$
- **Elastic-net regularizer:** $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1}\beta(\theta_j)^2 + (1-\beta)\sum_{j=0}^{d-1}|\theta_j|$

The different regularizers behaves differently:

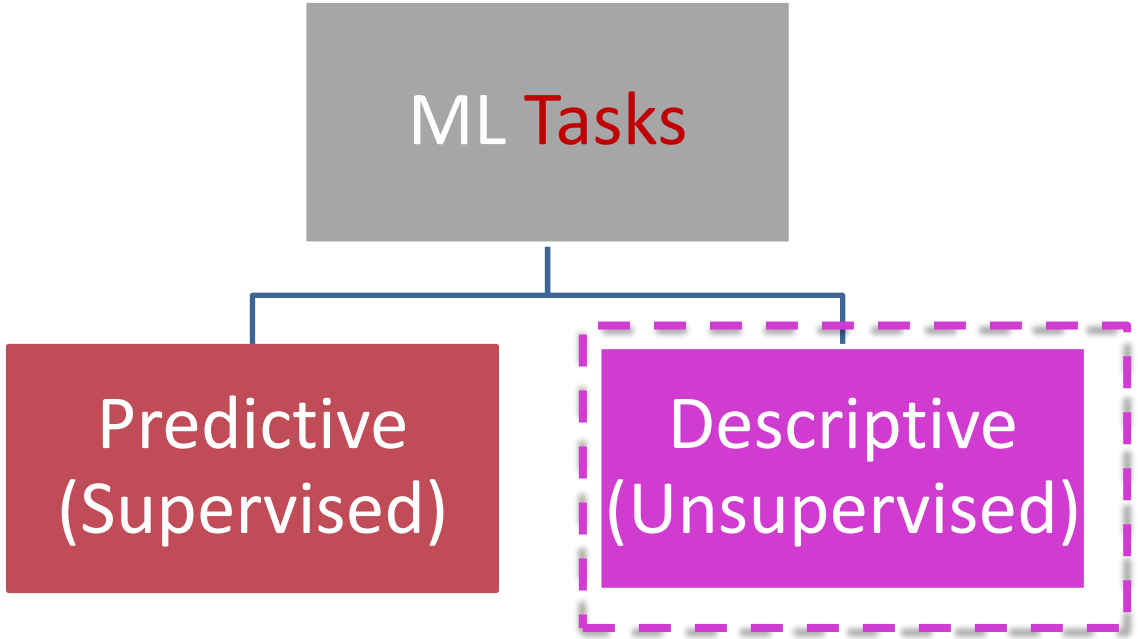
- The ridge penalty tends to shrink all coefficients to a **lower value**
- The lasso penalty tends to set more coefficients **exactly to zero**
- The elastic-net penalty is a compromise between ridge and lasso, with the β value controlling the two contributions

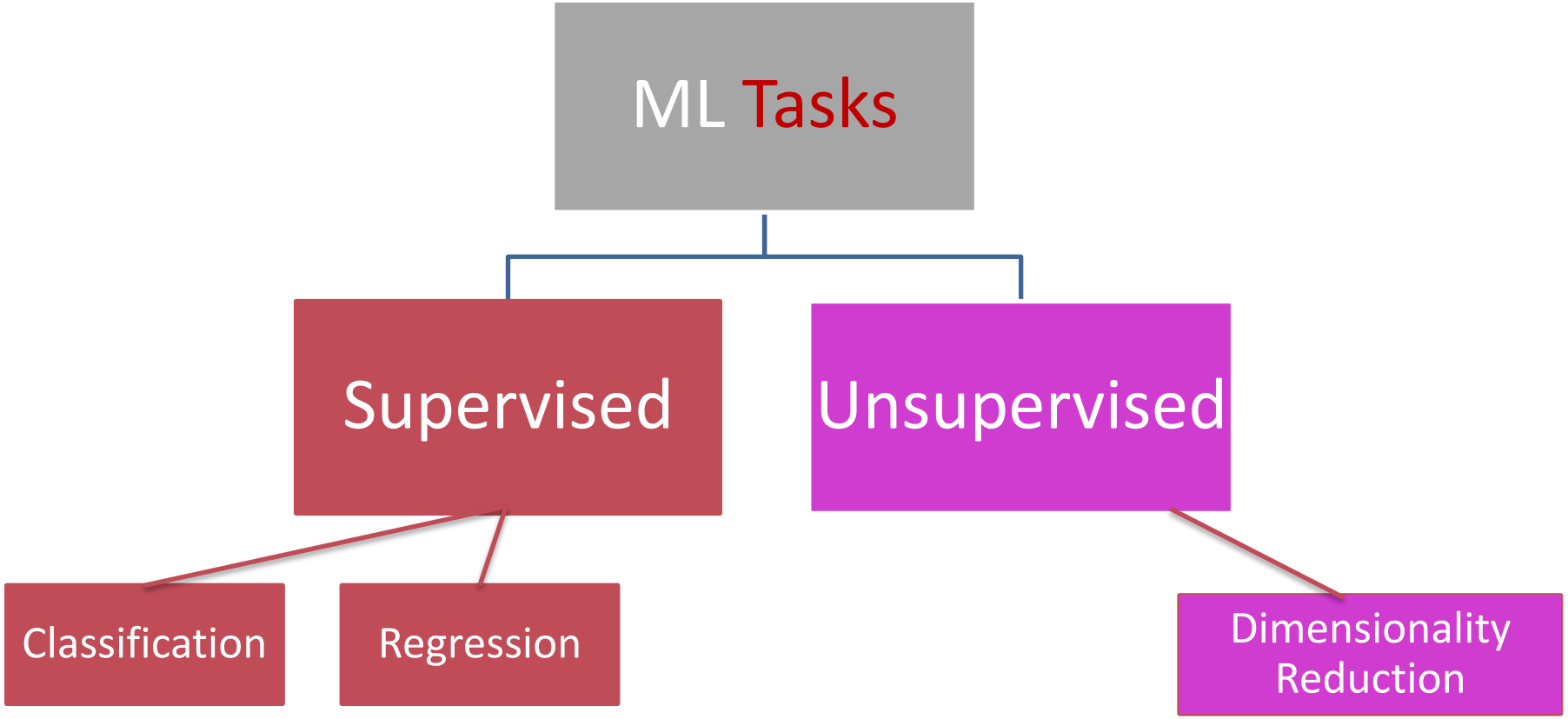
Regularization

- A way to reduce model complexity
- Not just for regression …

$$E_{aug}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\big(y(i) - h(\boldsymbol{\varphi}(i);\boldsymbol{\theta})\big)^2 + \lambda \cdot \sum_{j=0}^{d-1}(\theta_j)^2$$

$$L(y(i), h(\phi(i);\theta))$$


ML Tasks
Predictive
(Supervised)
Descriptive
(Unsupervised)

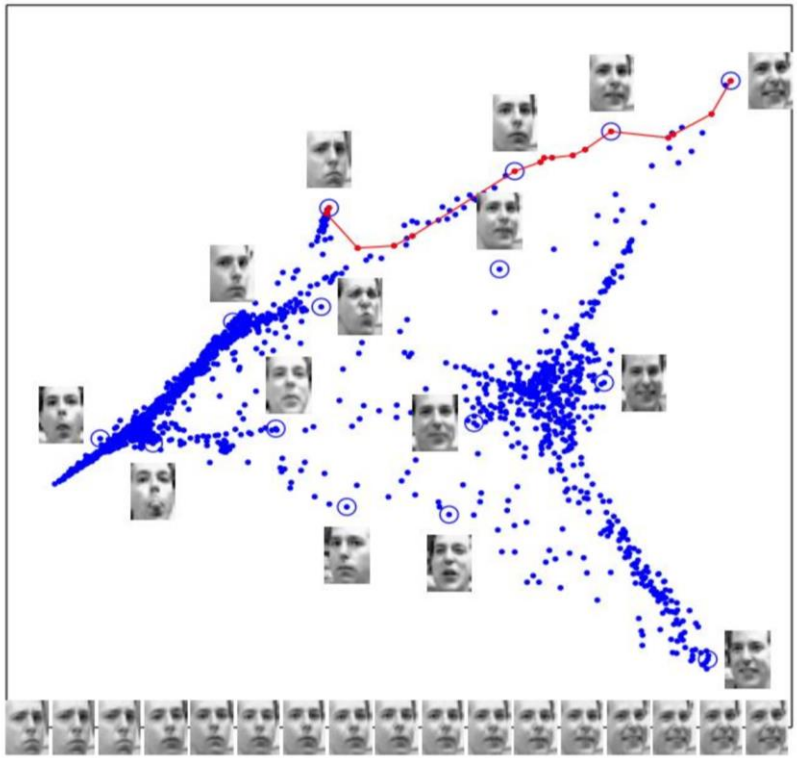

ML Tasks
Supervised
Unsupervised
Classification
Regression
Dimensionality Reduction

Unsupervised Learning → Dimensionality Reduction + Visualization

Task: Given $X \in \mathcal{X}$, learn $f(x)$.

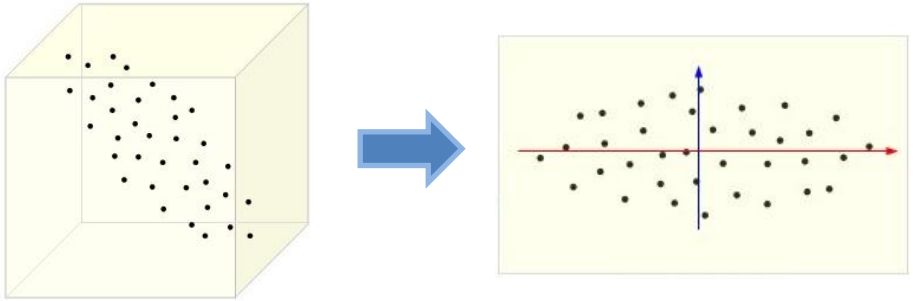
Images have thousands or millions of pixels.

Can we give each image a coordinate, such that similar images are near each other?



Selecting and Extracting Features

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.0 & 0.4 & 0.2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

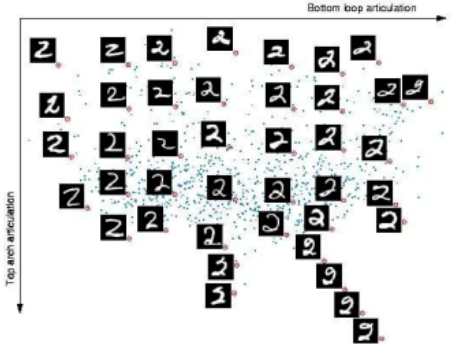


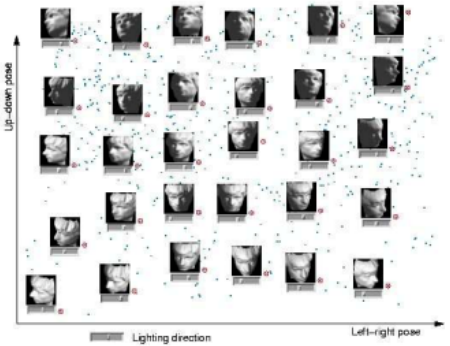
New Features as linear combination of old Features

$$X' = AX$$

Applications for Dimensionality Reduction

- To compress data by reducing dimensionality. E.g., representing each image in a large collection as a linear combination of a small set of "template" images
  - Also sometimes called dictionary learning (can also be used for other types of data, e.g., speech signals, text-documents, etc.)
- Visualization (e.g., by projecting high-dim data to 2D or 3D)





- To make learning algorithms run faster
- To reduce overfitting problem caused by high-dimensional data

Intro to Principal Components Analysis (PCA)

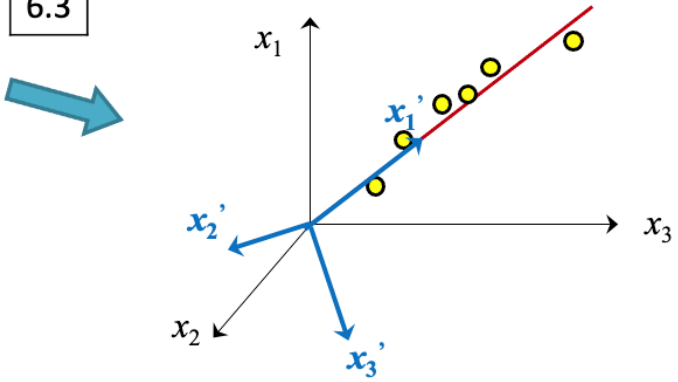
Finding informative feature axes

PCA strategy

- Construct new features that are **good** alternative representation of the original features

1	4	3	5.7	5.1	2.2
2	7.9	5.8	12	9.9	4.1
3.1	12	9	18	15	6.3

3.61	7.4	11.1	15.0	18.4	22.4

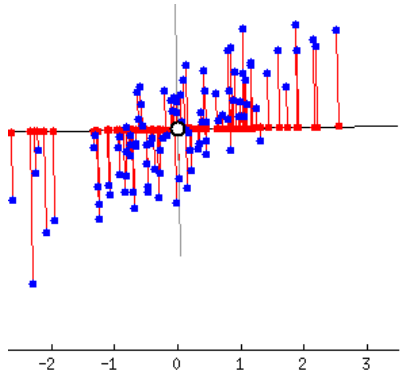


PCA strategy

- Construct new features that are **good** alternative representation of the original features
  - Good => Capture as much of original variation as possible

PCA strategy

- Construct new features that are **good** alternative representation of the original features
  - Good => Capture as much of original variation as possible



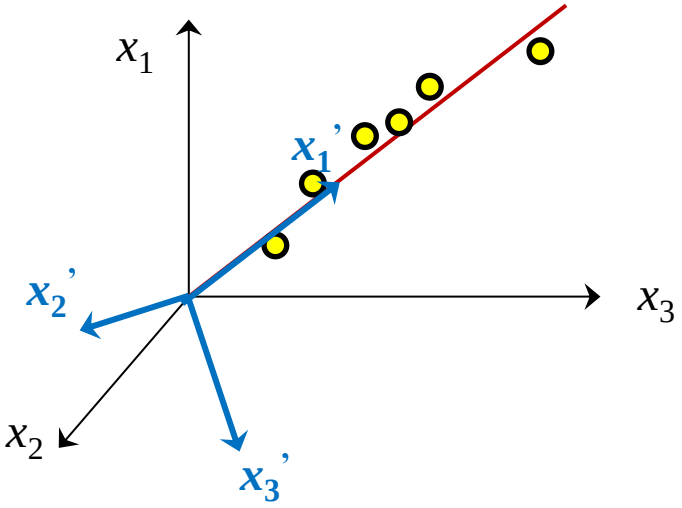
http://ethen8181.github.io/machine-learning/dim_reduct/PCA.html

PCA: How to find the PC ?

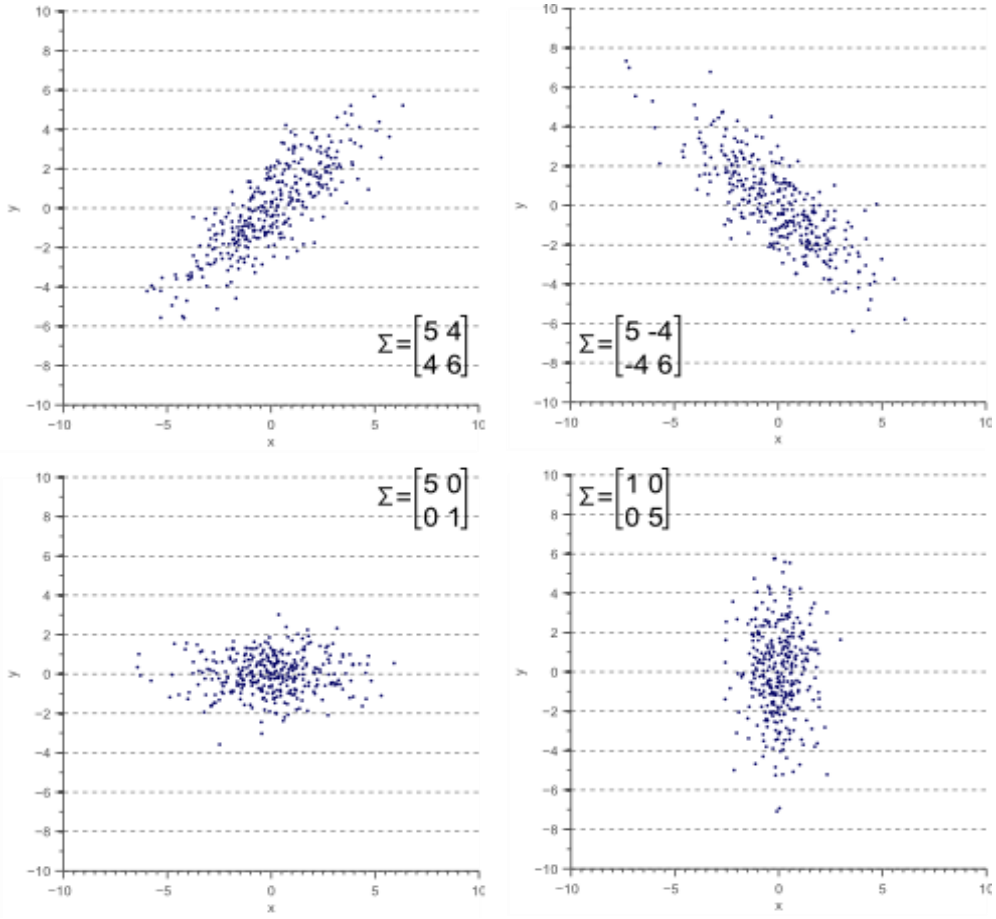
1	4	3	5.7	5.1	2.2
2	7.9	5.8	12	9.9	4.1
3.1	12	9	18	15	6.3

3.61	7.4	11.1	15.0	18.4	22.4
0.2	0.4	0.9	0.7	0.8	0.3
0.1	0.1	0.1	0.1	0.1	0.1

NOTE: These values are made up. Not exact.

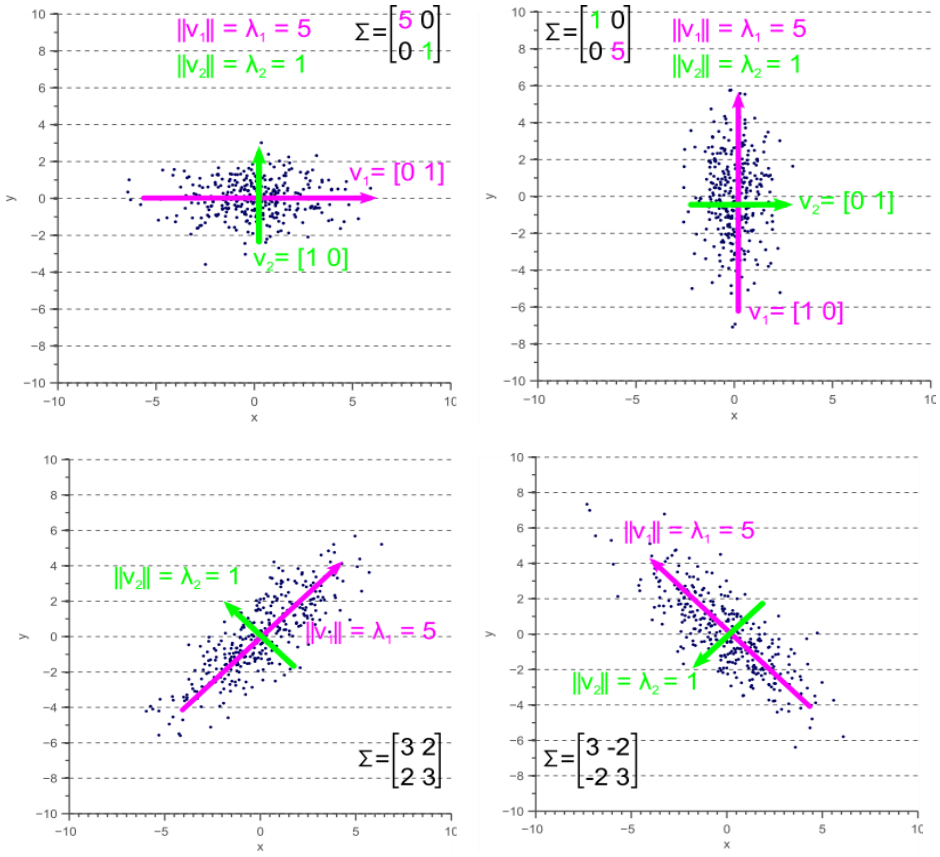


Covariance Matrix encodes spread and orientation of data



Eigen-analysis of Covariance Matrix

v_1, v_2: *Principal Components*



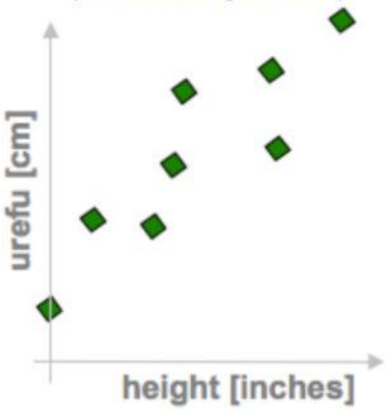
$$\Sigma\vec{v} = \lambda\vec{v}$$

Value of λ *indicates `variance' (spread) in direction of eigenvector* v *associated with* λ

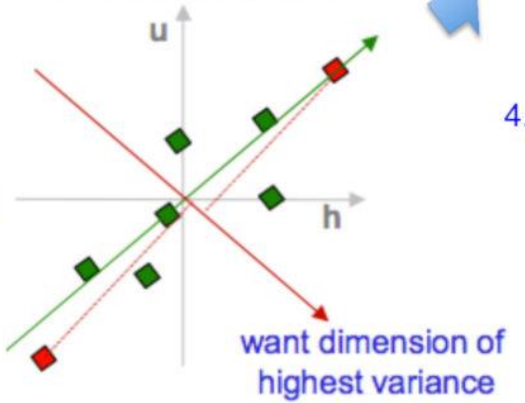
PCA in a nutshell

1. correlated hi-d data

("urefu" means "height" in Swahili)



2. center the points



3. compute covariance matrix

$$\begin{array}{c} \\ h \\ u \end{array}\begin{array}{c} \begin{array}{cc} h & u \end{array} \\ \begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \end{array} \rightarrow \mathrm{cov}(h,u) = \frac{1}{n}\sum_{i=1}^{n} h_i u_i$$

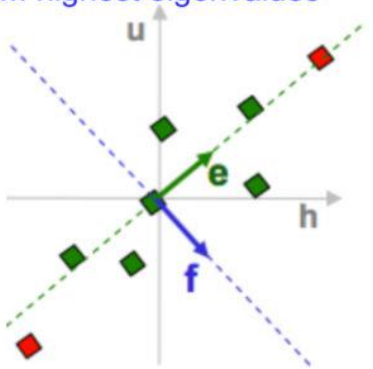
4. eigenvectors + eigenvalues

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} e_h \\ e_u \end{pmatrix} = \lambda_e \begin{pmatrix} e_h \\ e_u \end{pmatrix}$$

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} f_h \\ f_u \end{pmatrix} = \lambda_f \begin{pmatrix} f_h \\ f_u \end{pmatrix}$$

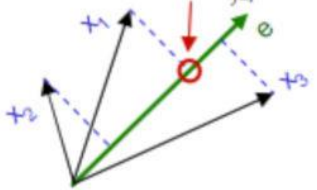
```
eig(cov(data))
```

5. pick m<d eigenvectors w. highest eigenvalues

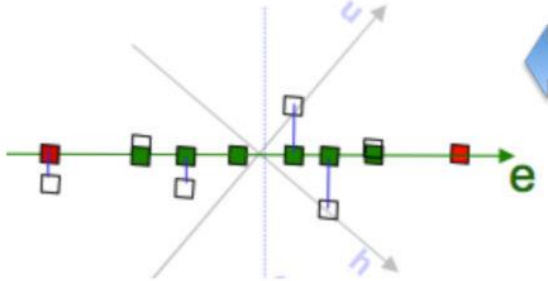


6. project data points to those eigenvectors

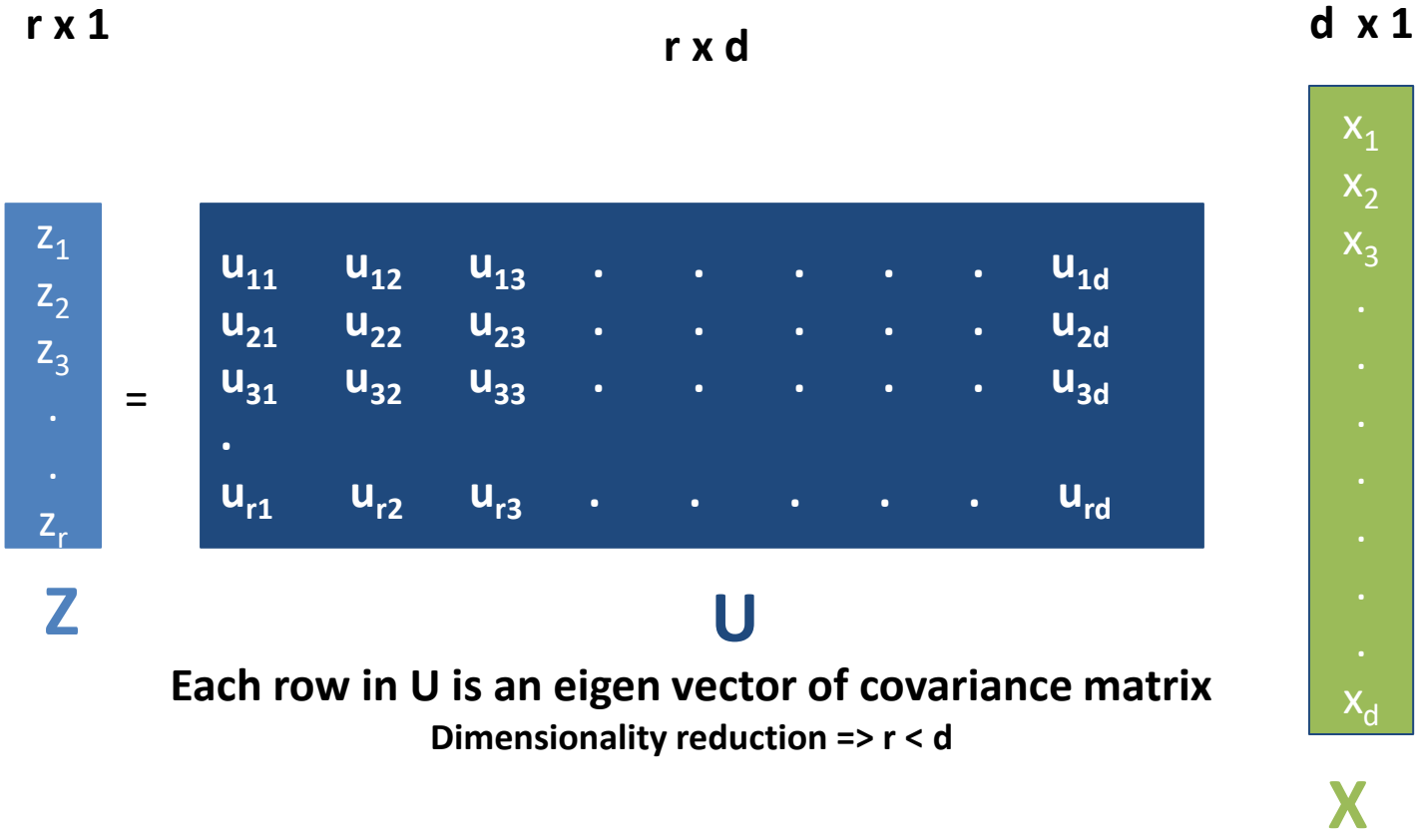
$$x'_e = x^T e = \sum_{j=1}^{d} x_{ij} e_j$$



7. uncorrelated low-d data



PCA: Dimensionality Reduction



Each row in U is an eigen vector of covariance matrix

Dimensionality reduction => r < d

Selecting and Extracting Features

$$\begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Selecting first and third feature

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.0 & 0.4 & 0.2 & 1.7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

New Features as linear combination of old Features

$$X' = AX$$

For PCA: Rows are Eigen vectors of the covariance matrix.

$$\begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

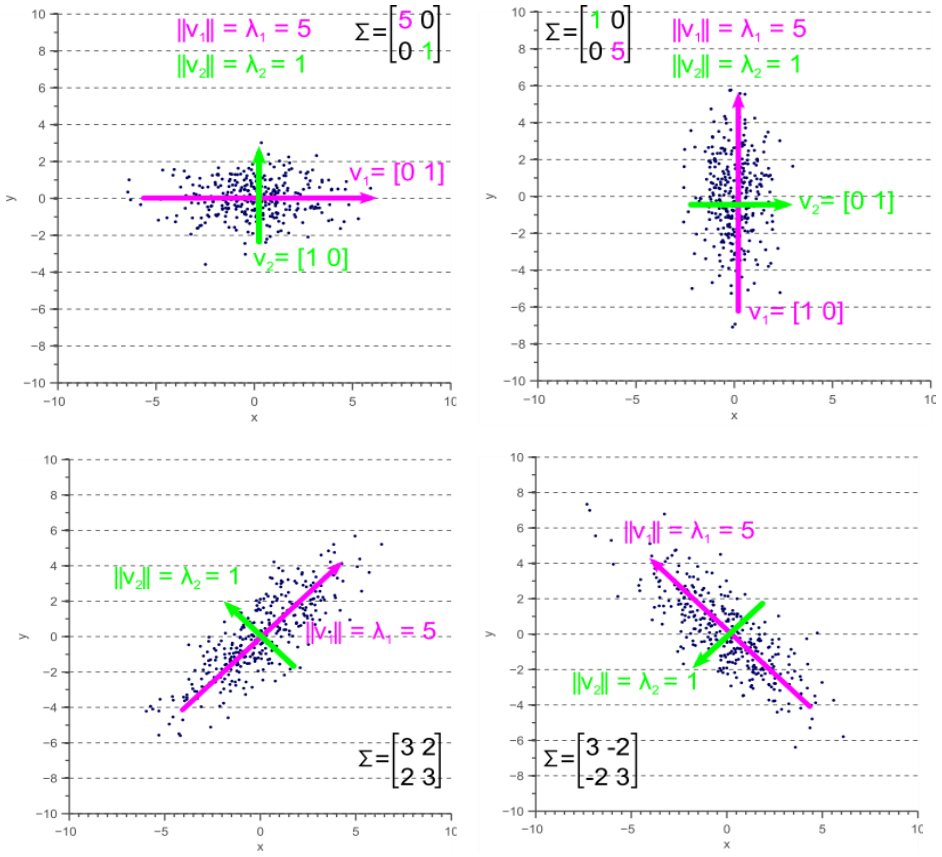
Selecting first and fourth feature

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \cdots & u_1^T & \cdots \\ \cdots & u_2^T & \cdots \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

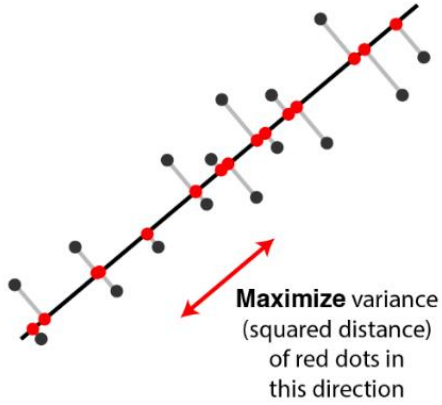
Eigen-analysis of Covariance Matrix

v_1, v_2: *Principal Components*

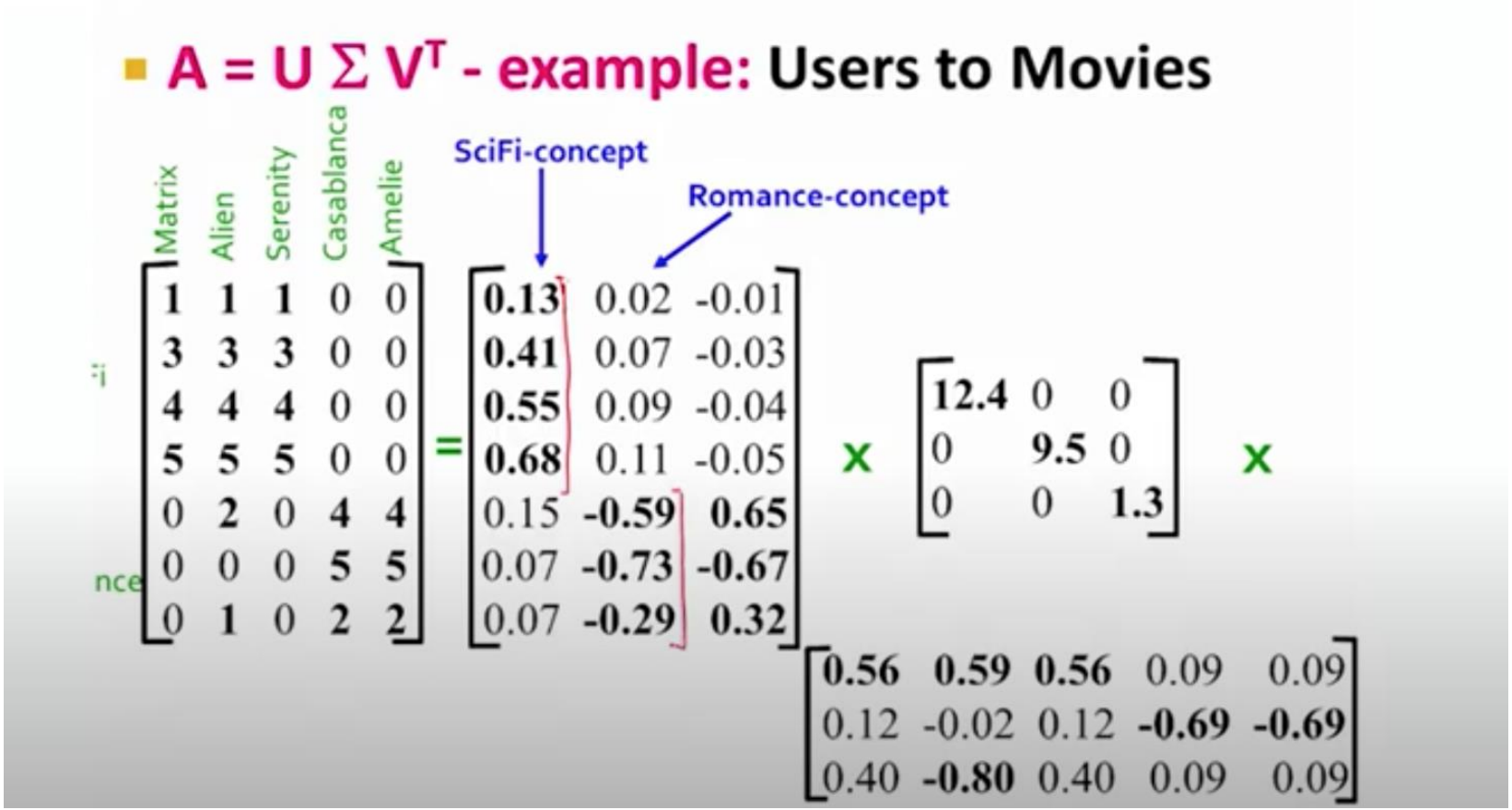
$$\Sigma\vec{v} = \lambda\vec{v}$$



Value of λ indicates `variance' (spread) in direction of eigenvector v associated with λ



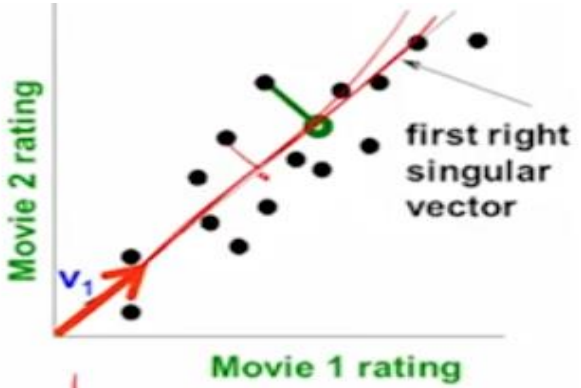
Singular Value Decomposition (SVD)
aka “billion-dollar algorithm”



https://www.youtube.com/watch?v=P5mlg91as1c

$A = U \Sigma V^T$ - example:

- $U \cdot \Sigma$: Gives the coordinates of the points in the projection axis



$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix}$$

Projection of users on the "Sci-Fi" axis ($(U\Sigma)^T$):

$$\begin{bmatrix} 1.61 & 0.19 & -0.01 \\ 5.08 & 0.66 & -0.03 \\ 6.82 & 0.85 & -0.05 \\ 8.43 & 1.04 & -0.06 \\ 1.86 & -5.60 & 0.84 \\ 0.86 & -6.93 & -0.87 \\ 0.86 & -2.75 & 0.41 \end{bmatrix}$$

https://www.youtube.com/watch?v=UyAfmAZU_WI

LoRA

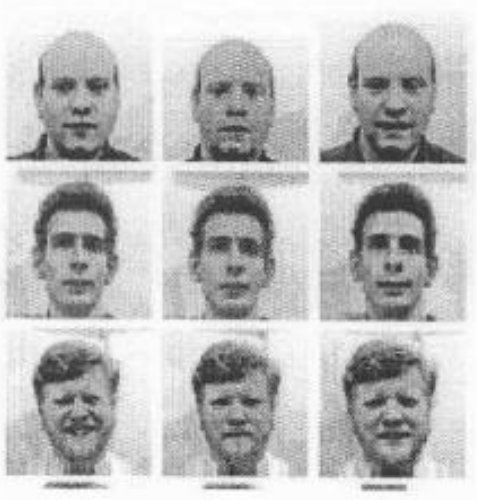
- Rank of a matrix ?
- PCA and SVD both involve finding a low-rank approximation of data X
- LoRA in modern LLMs involves finding a low-rank approximation for neural network weights
  - W = AB

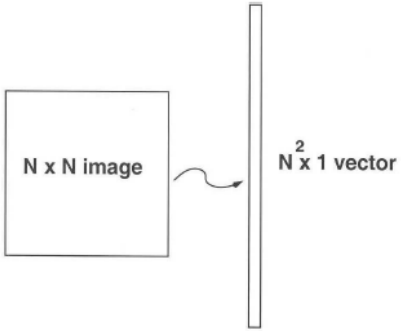
Application to Faces

- Computation of low-dimensional basis (i.e.,eigenfaces):

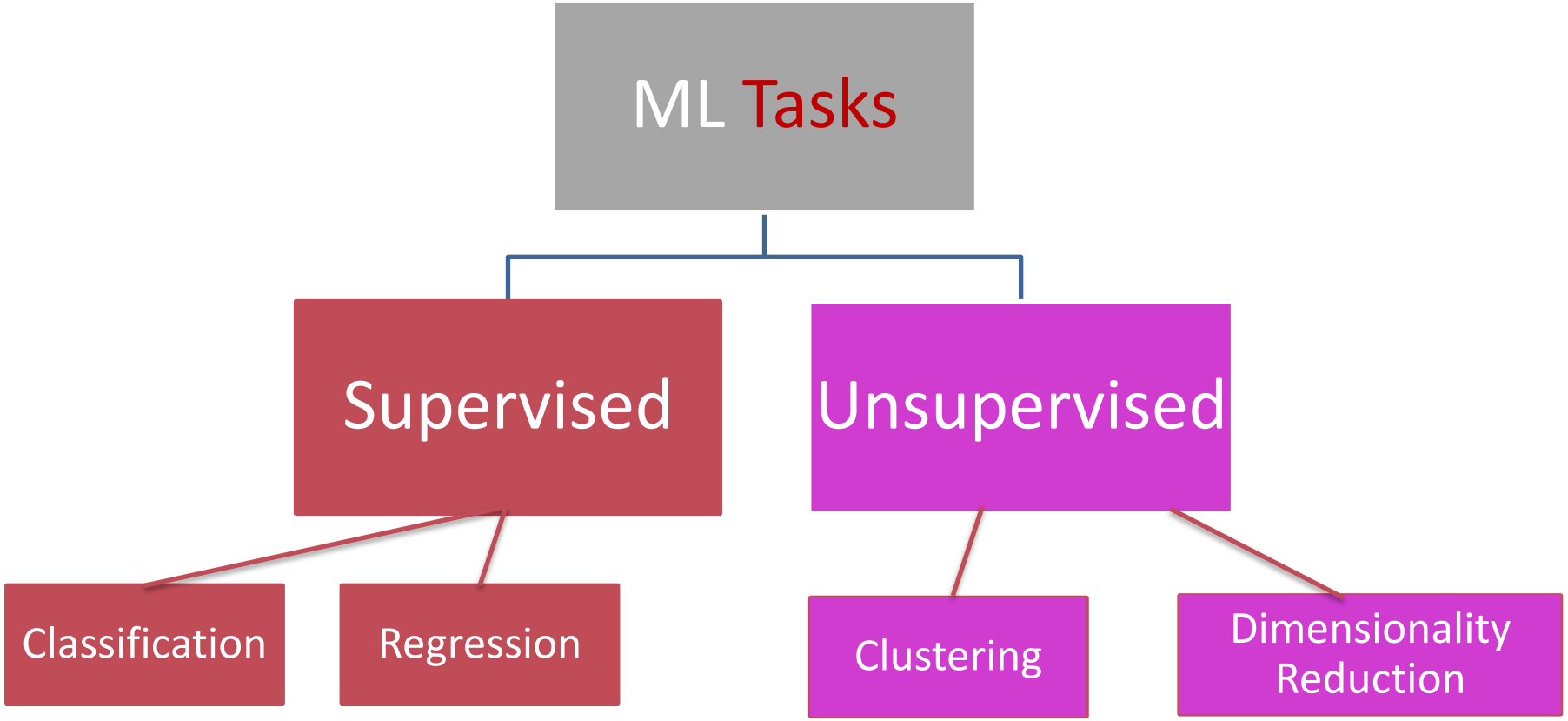
Step 1: obtain face images $I_1, I_2, ..., I_M$ (training faces)

(**very important:** the face images must be *centered* and of the same *size*)





Step 2: represent every image I_i as a vector Γ_i


ML Tasks
Supervised
Unsupervised
Classification
Regression
Clustering
Dimensionality Reduction

Unsupervised Learning → Clustering

Group similar things e.g. images

[Goldberger et al.]

- Determine groups of people in image above
  - based on clothing styles
  - gender, age, etc

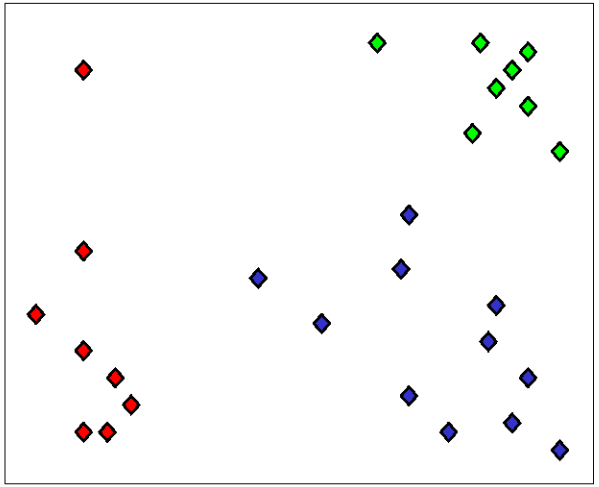
What is Clustering?

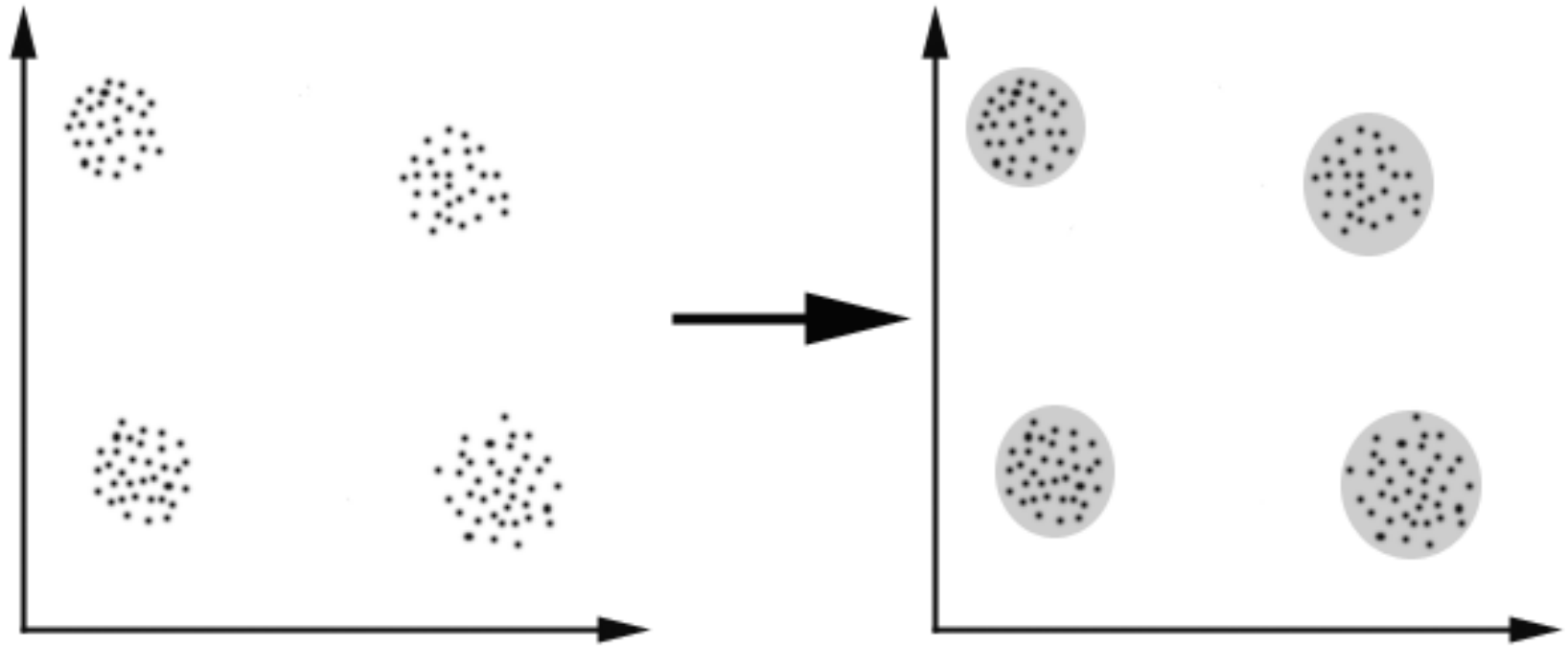
• Organizing data into *clusters* such that there is

  • high intra-cluster similarity

  • low inter-cluster similarity

•Informally, finding natural groupings among objects.





$\{x^{(1)}, \ldots, x^{(m)}\}$ $x^{(i)} \in \mathbb{R}^n$

The k-means clustering algorithm is as follows:

1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

   For every i, set

$$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

   For each j, set

$$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

   }

Assignment step: Assign each data point to the

Refitting step: Move each cluster center to the center of the data assigned to it

$\{x^{(1)}, \ldots, x^{(m)}\}$ $x^{(i)} \in \mathbb{R}^n$

The k-means clustering algorithm is as follows:

1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

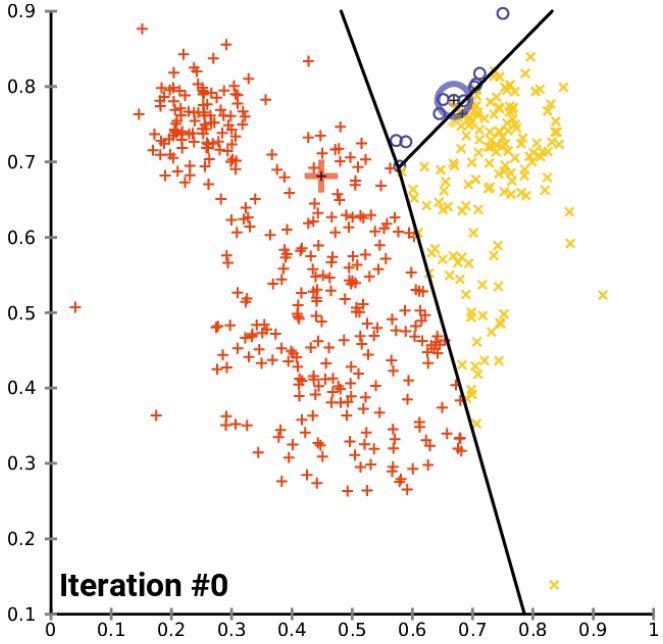
   For every i, set

   $$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

   For each j, set

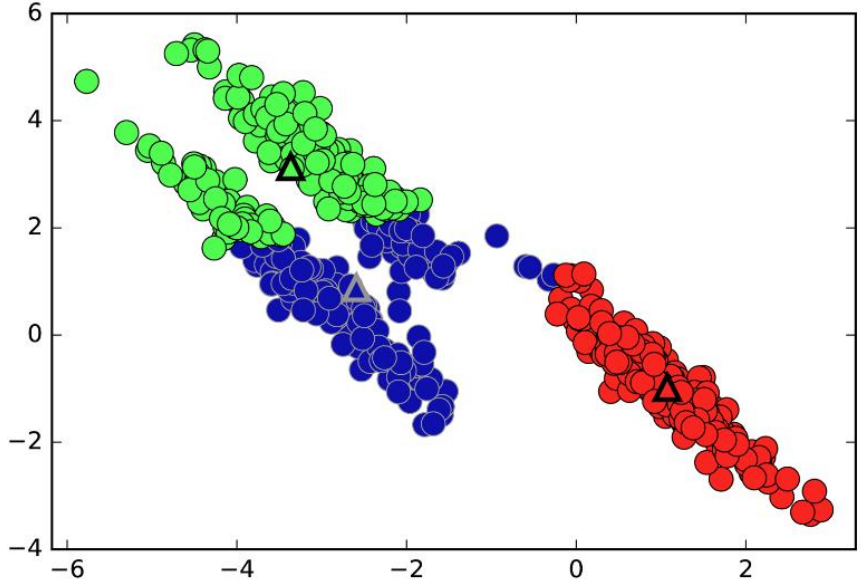
   $$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

   }



K-means: Additional issues

- ‘Hard’ assignments
- Euclidean ➔ Favours ‘spherical’ clusters of equal ‘contribution’
- Sensitive to initialization
- Sensitive to outliers



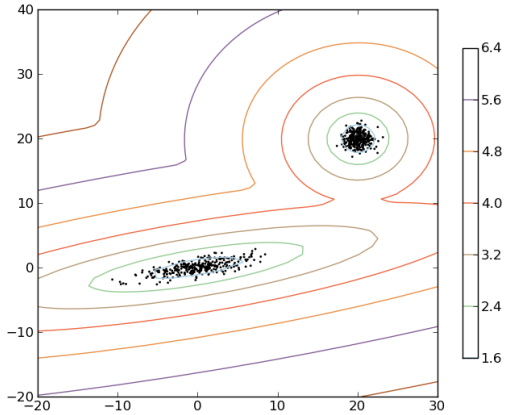
Mixture Models

Most common mixture model: Gaussian mixture model (GMM)

- A GMM represents a **distribution** as

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

with π_k the mixing coefficients, where:



$$\sum_{k=1}^{K} \pi_k = 1 \quad \text{and} \quad \pi_k \geq 0 \quad \forall k$$

- Can think of the data $\{x_1, x_n, \ldots, x_N\}$ using a "generative story"
  - For each example x_n, first choose its cluster assignment $z_n \in \{1, 2, \ldots, K\}$ as

$$z_n \sim \text{Multinoulli}(\pi_1, \pi_2, \ldots, \pi_K)$$

  - Now generate x from the Gaussian with id z_n

$$x_n | z_n \sim \mathcal{N}(\mu_{z_n}, \Sigma_{z_n})$$

Parameter Estimation in GMMs using EM

- Initialize the means μ_k, covariances Σ_k and mixing coefficients π_k
- Iterate until convergence:
  - E-step: Evaluate the responsibilities given current parameters

$$\gamma_k^{(n)} = p(z^{(n)}|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}^{(n)}|\mu_j, \Sigma_j)}$$

  - M-step: Re-estimate the parameters given current responsibilities

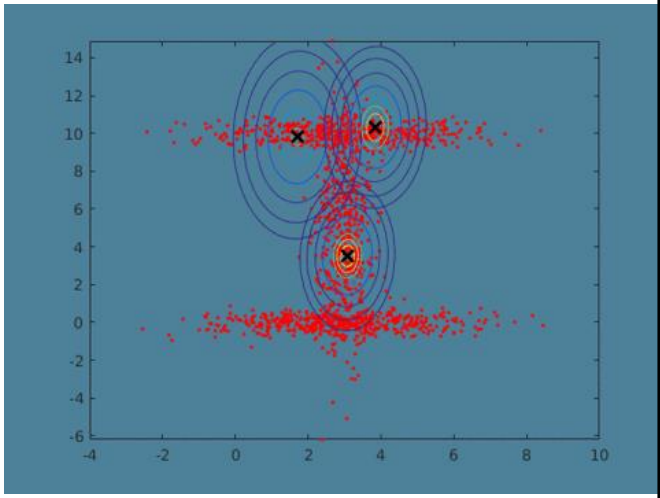
$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} \mathbf{x}^{(n)}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T$$

$$\pi_k = \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^{N} \gamma_k^{(n)}$$

  - Evaluate log likelihood and check for convergence

$$\ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, \Sigma_k) \right)$$

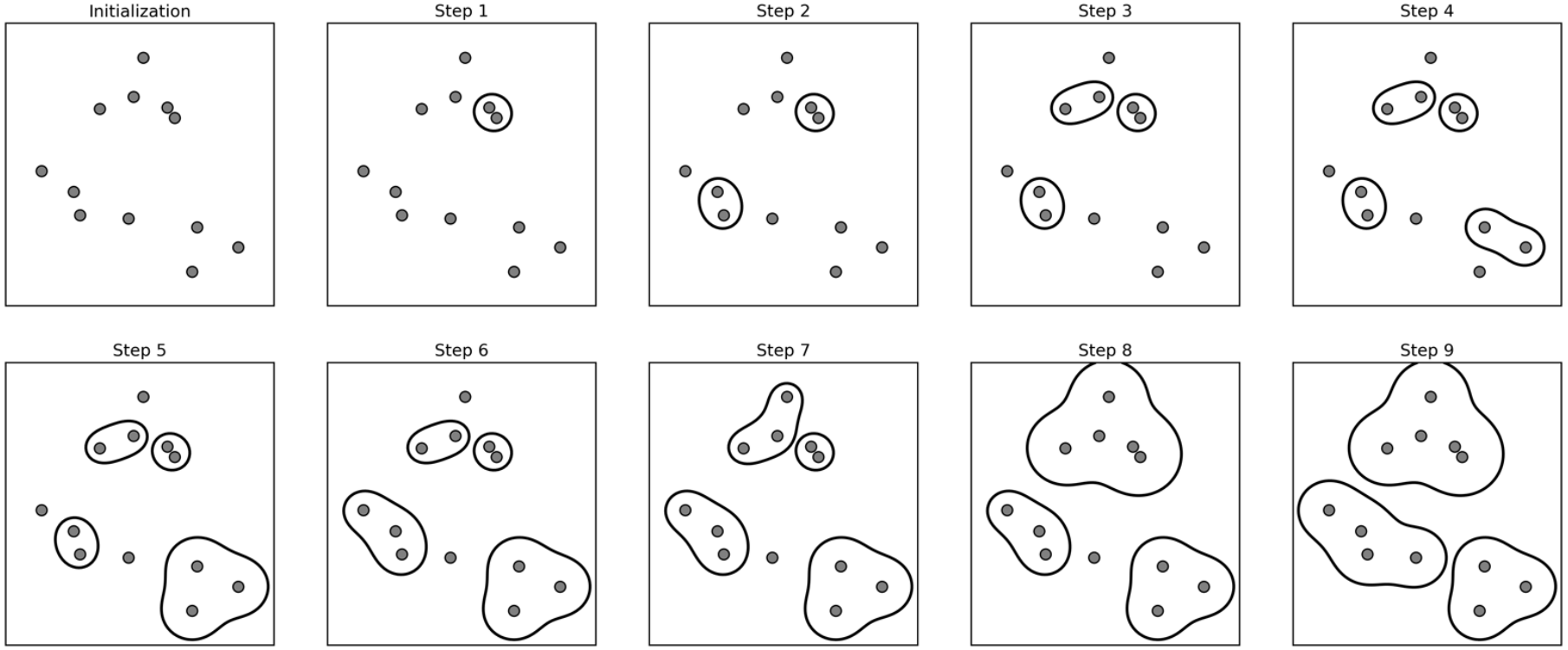


Hierarchical Clustering

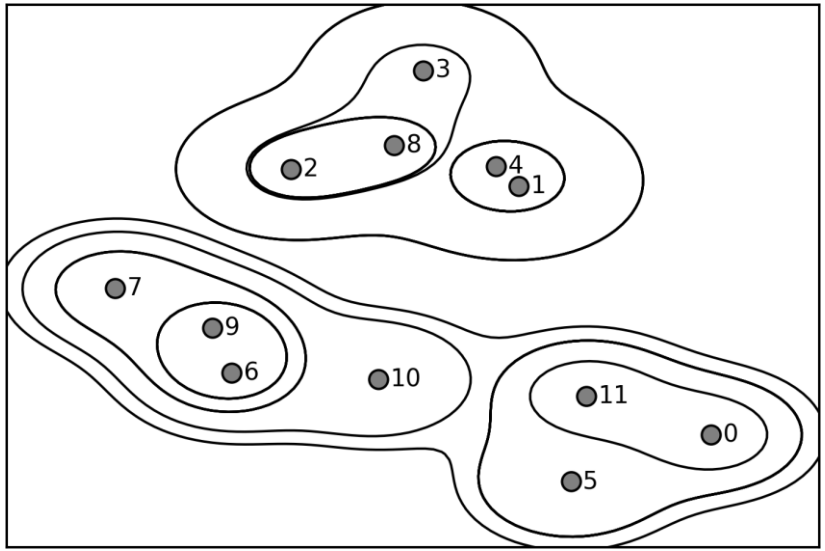
- Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of samples.

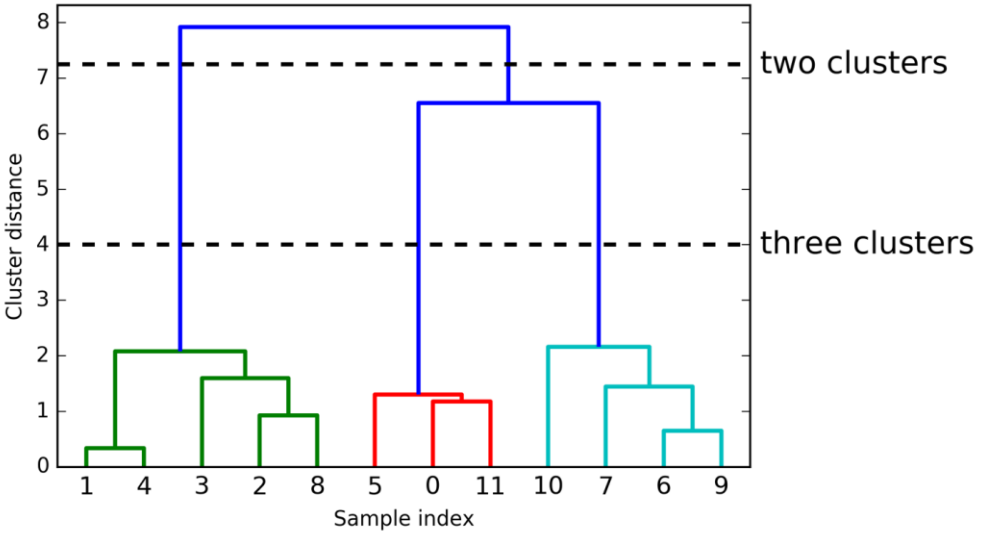
- *One approach*: recursive application of a partitional clustering algorithm.

Bottom-up (Agglomerative) Clustering



Dendrogram: A cluster visualization

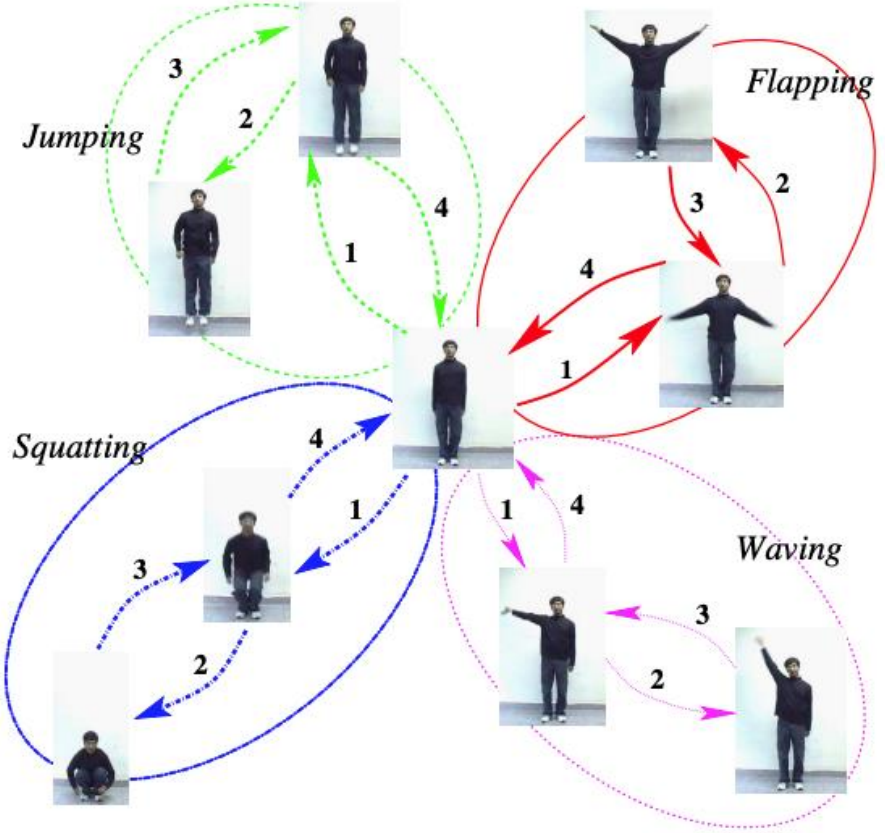




Evaluation of clustering

- Perhaps the most substantive issue in ML:
  - how do you measure goodness?
- Most measures focus on computational efficiency
  - Time and space
- For application of clustering to search:
  - Measure retrieval effectiveness

Recognizing Human Activities from Constituent Actions



https://cvit.iiit.ac.in/images/ConferencePapers/2005/ravi05humanact.pdf

Unsupervised Learning → Density Estimation

Task: Given $X \in \mathcal{X}$, learn $f(x)$.