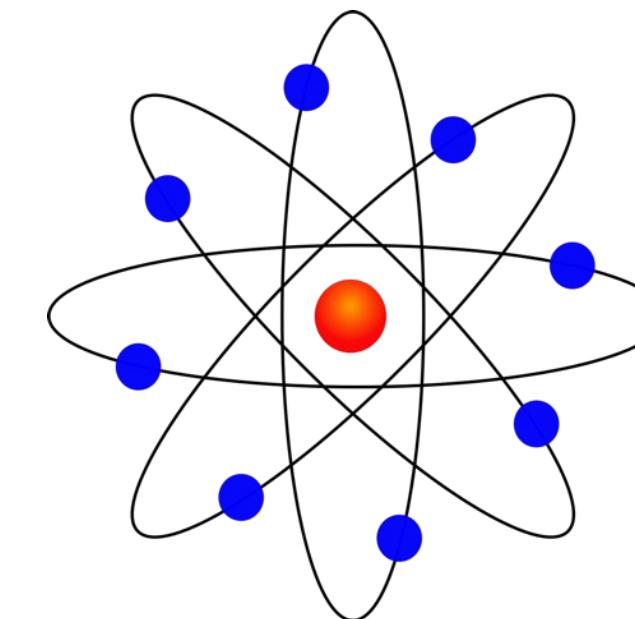


# Machine Learning Intro

# Rapid growth of massive datasets

E.g., Online activity, Science, Sensor networks

Data



facebook®



# Distributed Clusters are Pervasive

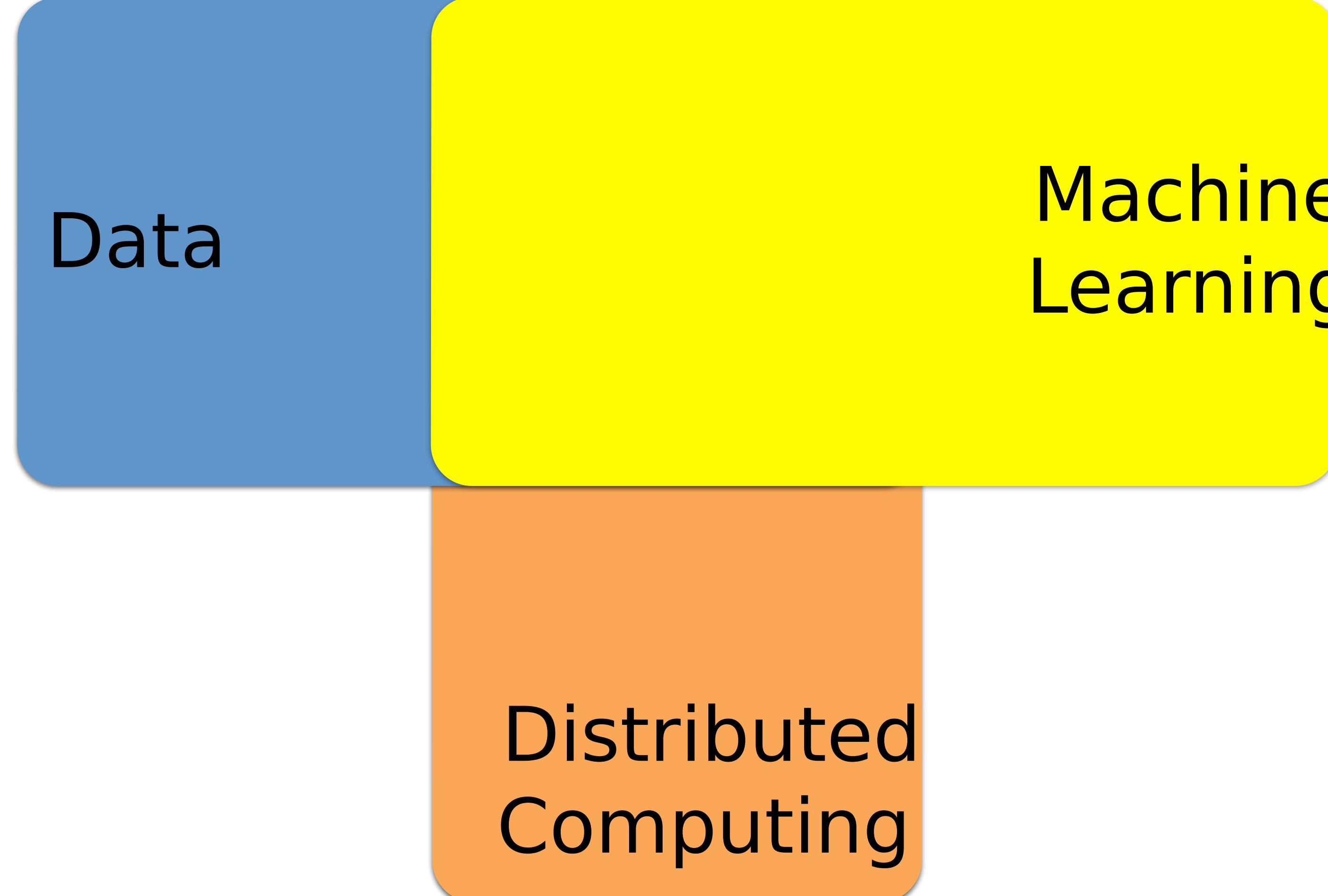
Data

Distributed  
Computing



# Mature Methods for Common Problems

e.g., classification, regression, collaborative filtering, clustering



Data

Machine  
Learning

Distributed  
Computing

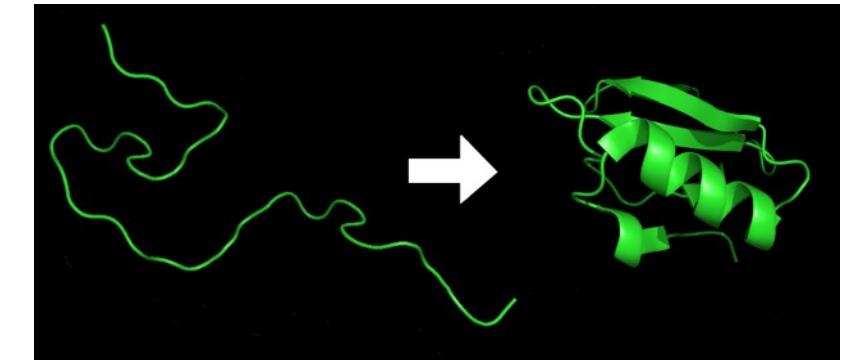
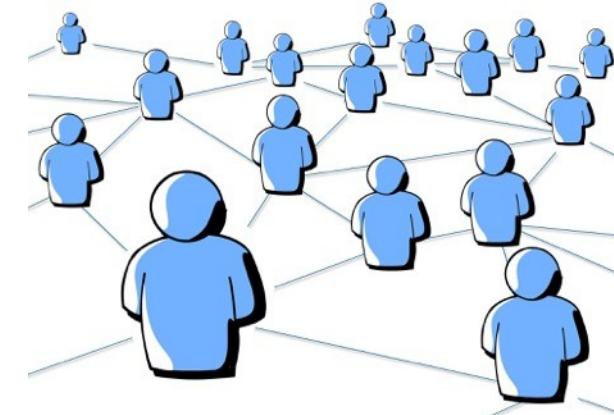
# ML is Applied Everywhere

E.g., Personalized recommendations, Speech recognition, Face detection, Protein structure, Fraud detection, Spam filtering, Playing chess or Jeopardy, Unassisted vehicle control, Medical diagnosis, ...

Data

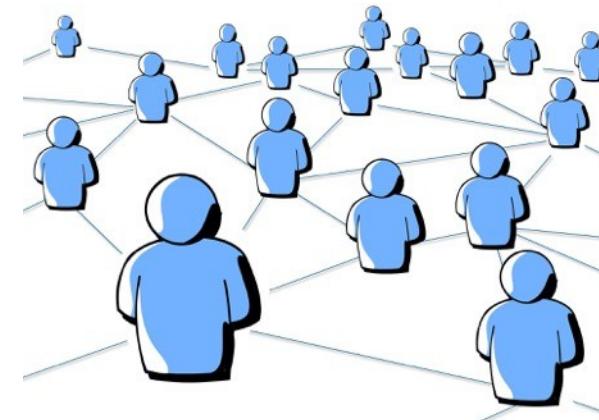
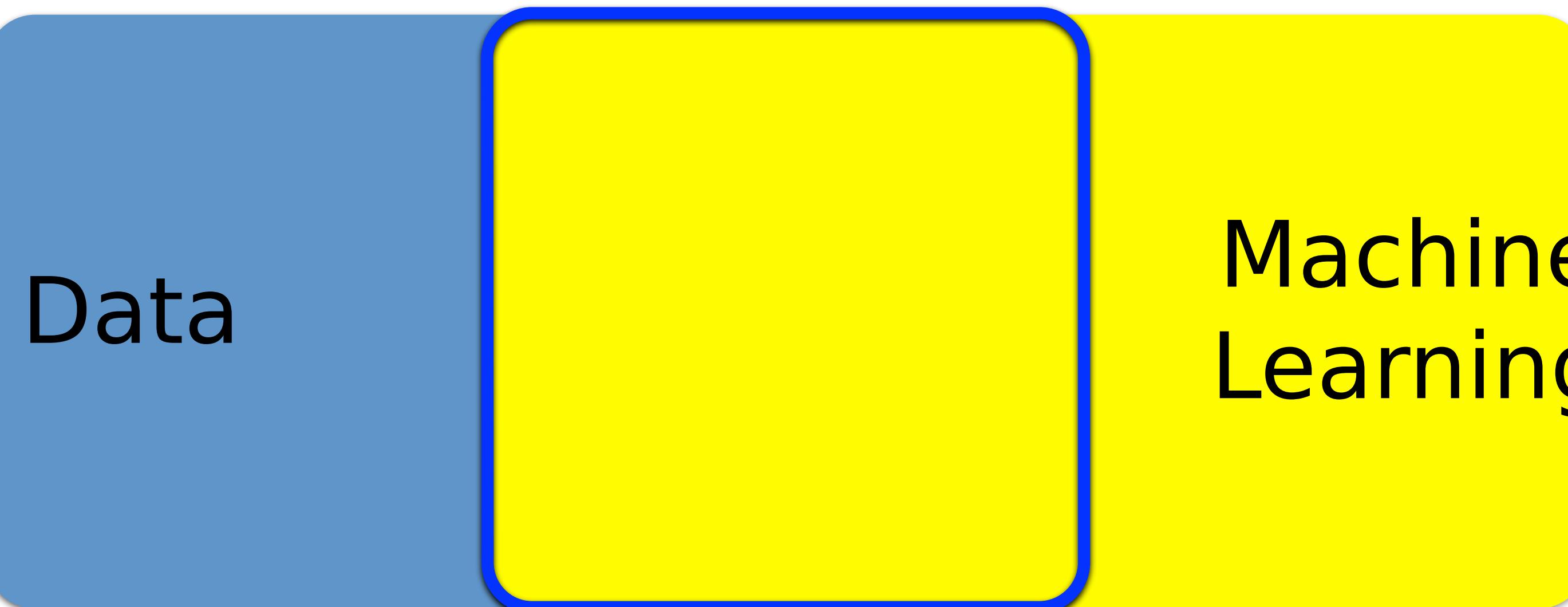
Machine  
Learning

Distributed  
Computing

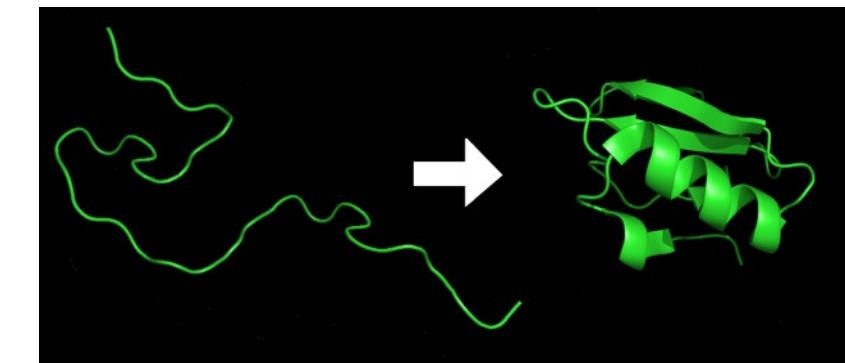


# ML is Applied Everywhere

E.g., Personalized recommendations, Speech recognition, Face detection, Protein structure, Fraud detection, Spam filtering, Playing chess or Jeopardy, Unassisted vehicle control, Medical diagnosis, ...

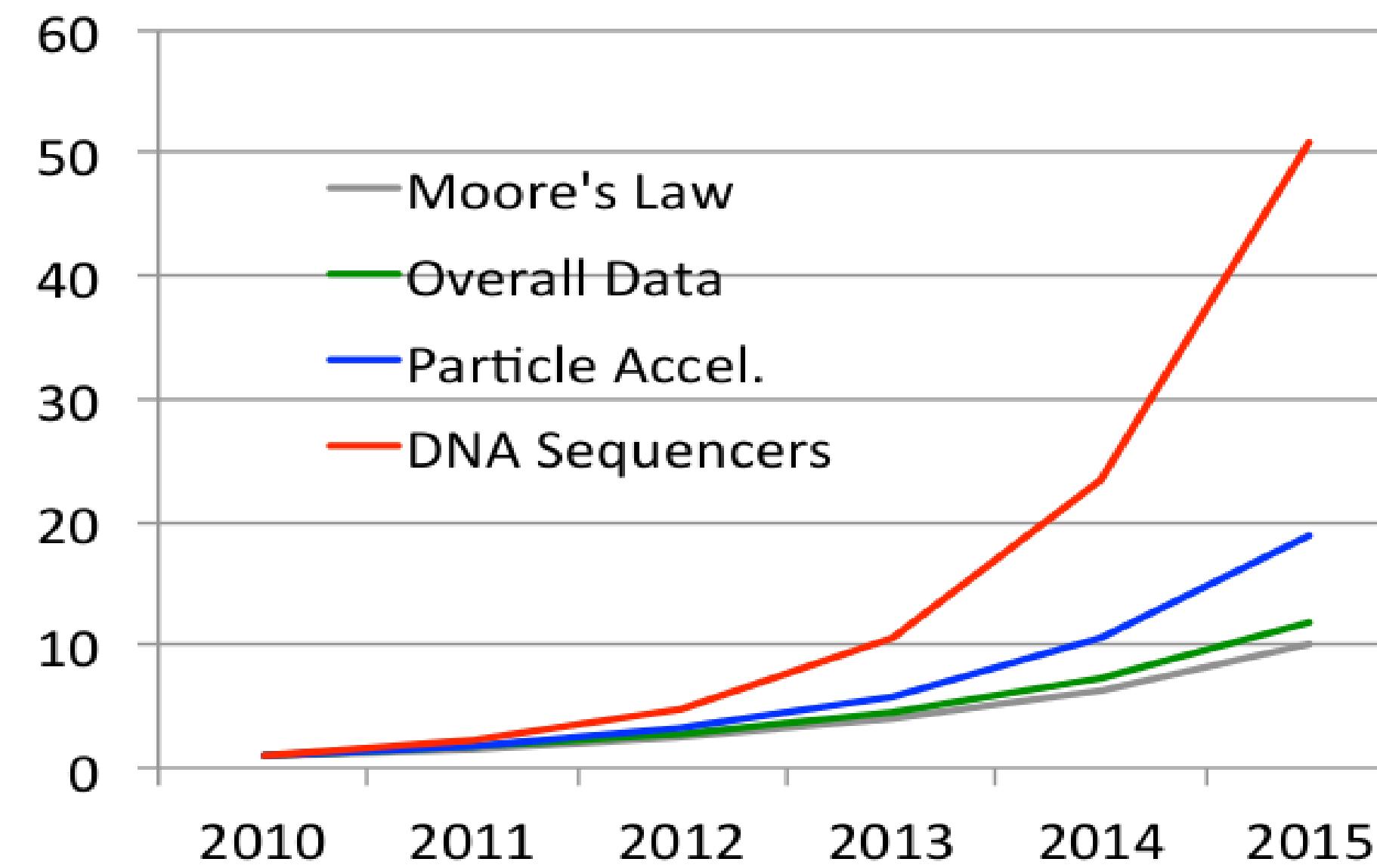


Distributed  
Computing

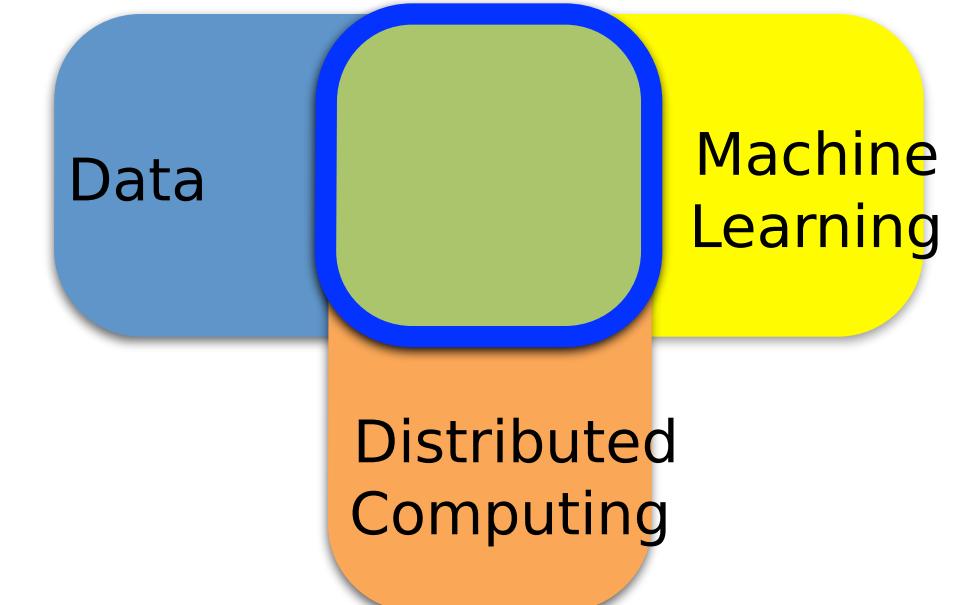


# Challenge: Scalability

Classic ML techniques are not always suitable for modern datasets



Data Grows Faster  
than Moore's Law  
[IDC report, Kathy Yelick, LBNL]



# Course Goals

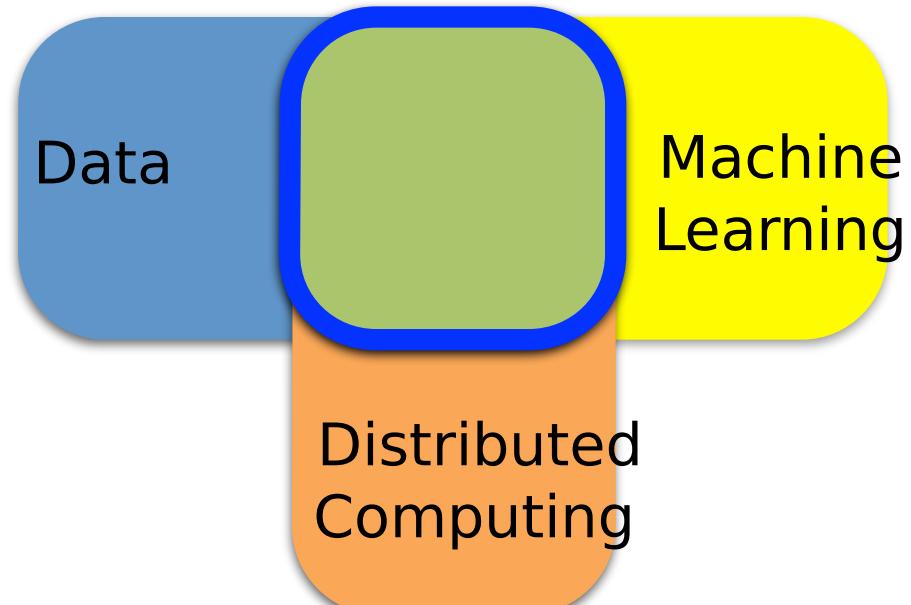
Focus on scalability challenges for common ML tasks

How can we use raw data to train statistical models?

- Study typical ML pipelines
- Classification, regression, exploratory analysis

How can we do so at scale?

- Study distributed machine learning algorithms
- Implement distributed pipelines in Apache Spark using real datasets
- Understand details of MLLib (Spark's ML library)



# Prerequisites

## Basic Python, ML, math background

- First week provides review of ML and useful math concepts

## A good Python Tutorial

- <http://codehero.co/series/python-desde-cero.html>

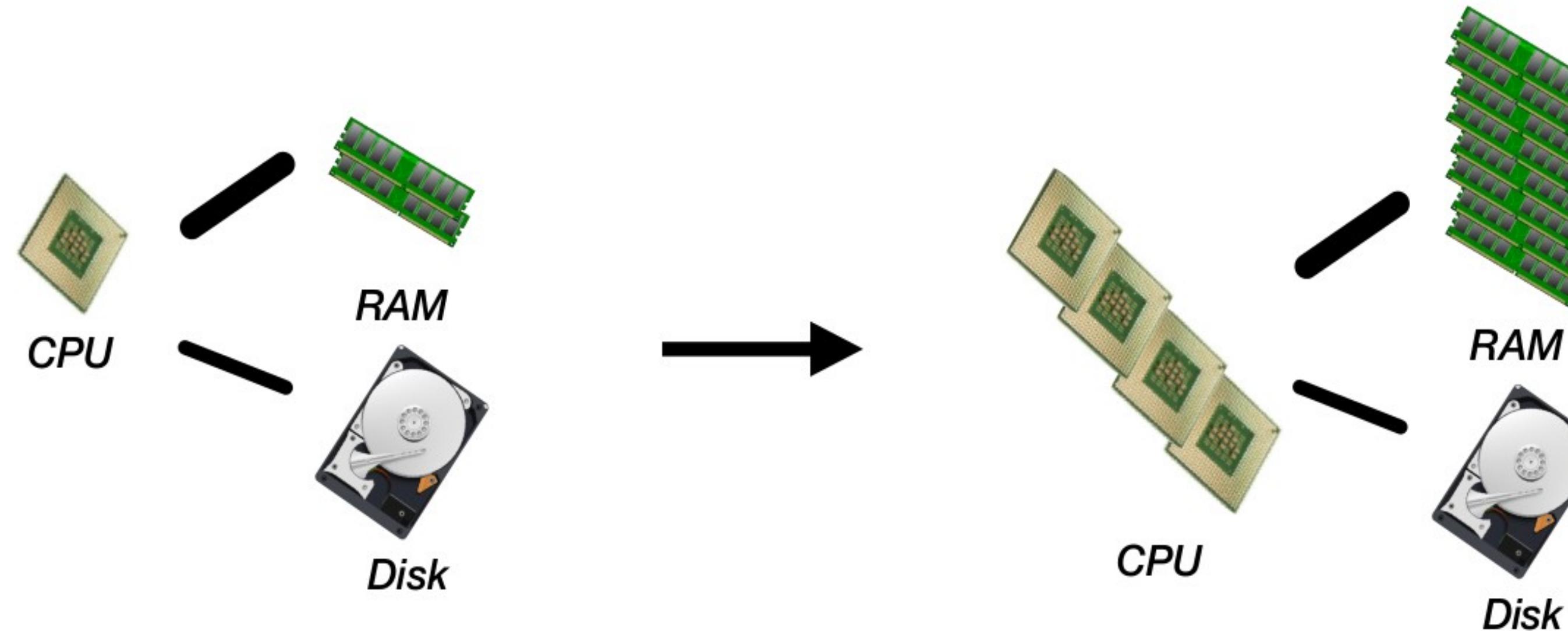
# **Distributed Computing and Apache Spark**

# How to Handle Massive Data?

Need more hardware to store / process modern data

## Scale-up(one big machine)

- Can be very fast for medium scale problems
- Expensive, specialized hardware
- Eventually hit a wall

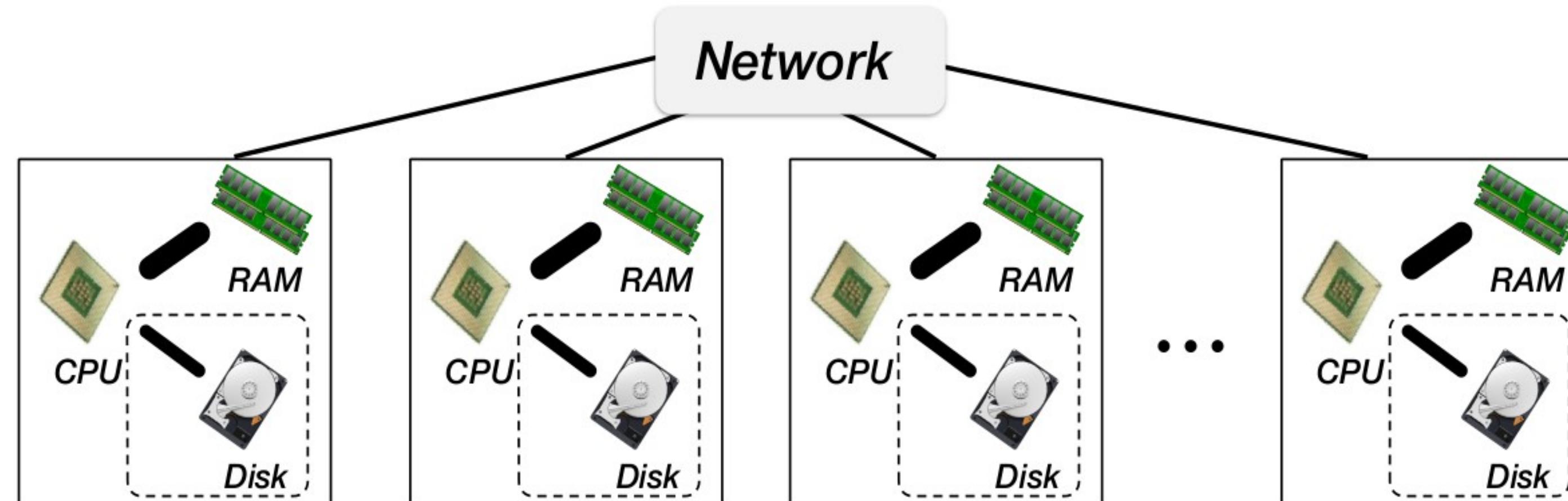


# How to Handle Massive Data?

Need more hardware to store / process modern data

**Scale-out** (distributed, e.g., cloud-based)

- Commodity hardware, scales to massive problems
- Need to deal with network communication
- Added software complexity



# What is Apache ?

General, open-source cluster computing engine

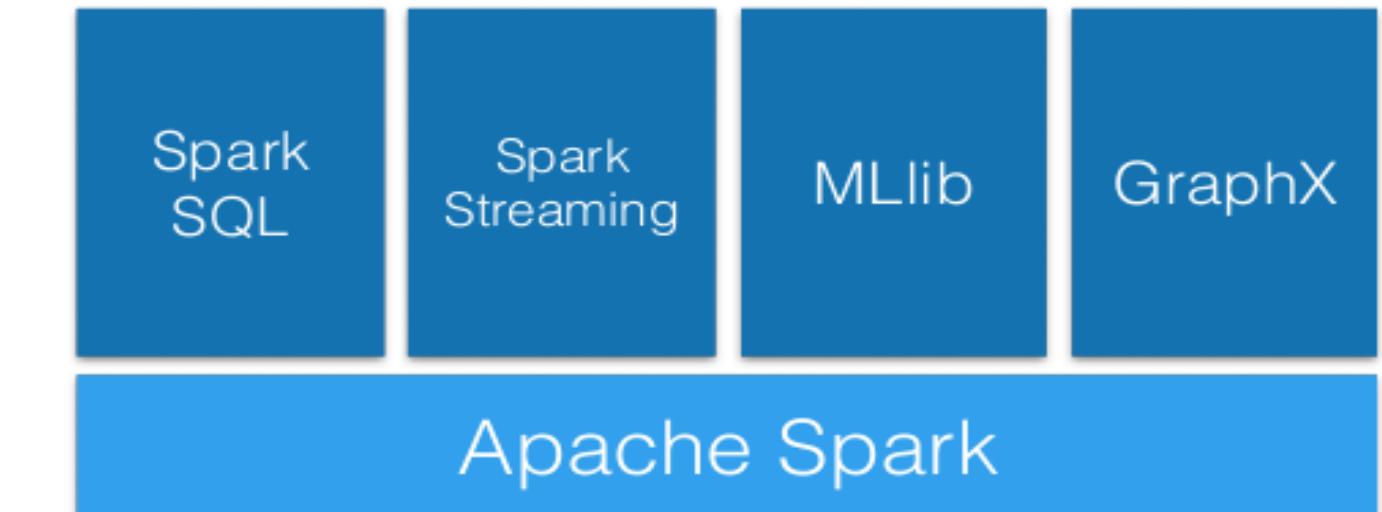
## Well-suited for machine learning

- Fast iterative procedures
- Efficient communication primitives

## Simple and Expressive

- APIs in Scala, Java, Python, R
- Interactive Shell

## Integrated Higher-Level Libraries



# **what is Machine Learning?**

# A Definition

Constructing and studying methods that learn from and make predictions on data

Broad area involving tools and ideas from various domains

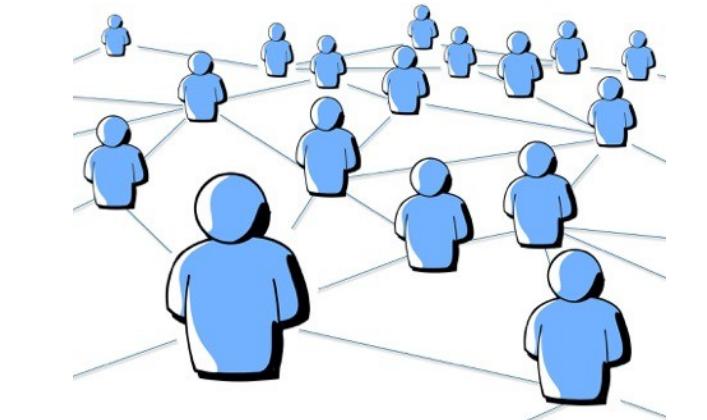
- Computer Science
- Probability and Statistics
- Optimization
- Linear Algebra

# Some Examples

Face recognition



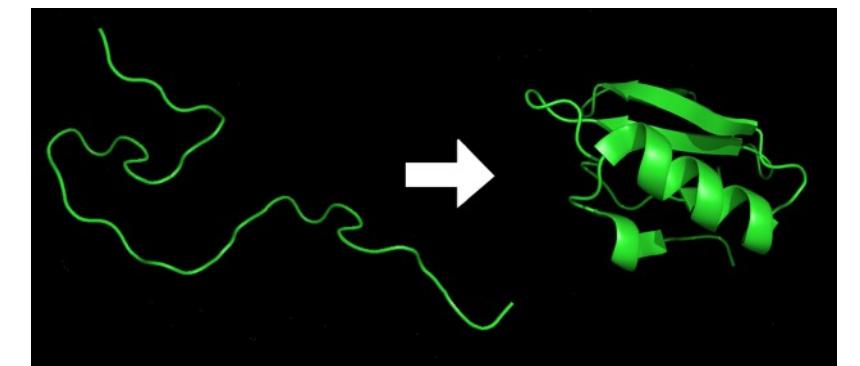
Link prediction



Text or document classification, e.g., spam detection



Protein structure prediction



Games, e.g., Backgammon or Jeopardy



# Terminology

**Observations** Items or entities used for learning or evaluation, e.g., emails

**Features** Attributes (typically numeric) used to represent an observation, e.g., length, date, presence of keywords

**Labels** Values / categories assigned to observations, e.g., *spam*, *not-spam*

**Training and Test Data.** Observations used to train and evaluate a learning algorithm, e.g., a set of emails along with their labels

- Training data is given to the algorithm for training
- Test data is withheld at train time

# Two Common Learning Settings

**Supervised learning.** Learning from labeled observations

- Labels ‘teach’ algorithm to learn mapping from observations to labels

**Unsupervised learning** Learning from unlabeled observations

- Learning algorithm must find latent structure from features alone
- Can be goal in itself (discover hidden patterns, exploratory data analysis)
- Can be means to an end (preprocessing for supervised task)

# Examples of Supervised Learning

**Classification** Assign a category to each item, e.g., spam detection

- Categories are discrete
- Generally no notion of ‘closeness’ in multi-class setting

**Regression** Predict a real value for each item, e.g., stock prices

- Labels are continuous
- Can define ‘closeness’ when comparing prediction with label

# Examples of Unsupervised Learning

**Clustering** Partition observations into homogeneous regions, e.g., to identify “communities” within large groups of people in social networks

**Dimensionality Reduction** Transform an initial feature representation into a more concise representation, e.g., representing digital images

# Typical Supervised Learning Pipeline

Obtain Raw Data

Raw data comes from many sources

# Data Types

## Web hypertext

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD
  XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/
  xhtml1-transitional.dtd">
3
4 <html xmlns="http://www.w3.org/1999/
  xhtml">
5   <head>
6     <meta http-equiv="Content-
      Type" content=
7       "text/html; charset=us-
      ascii" />
8     <script type="text/
      javascript">
9       function reDo() {top.
  location.reload();}
10      if (navigator.appName ==
    'Netscape') {top.onresize = reDo;}
11      dom=document.
    getElementById;
12     </script>
13   </head>
14   <body>
15   </body>
16 </html>
```

# Data Types

Warriors Inbox x 

to me 12/8/14  ◀ Reply ▼

Ameet,

We recently released our popular **Holiday Hoops** packs. The packs also include an exclusive Warriors Holiday Card! These packs provide our biggest games from January to March! A great gift for the holidays!!!

**Holiday Hoops West Pack** (*Club 200 Sideline-\$303, Club 200 Baseline- \$260*)

Mon 1/5 vs Oklahoma City Thunder @ 7:30pm  
Wed 1/21 vs Houston Rockets @ 7:30pm  
Sun 3/8 vs LA Clippers @ 12:30pm  
Mon 3/16 vs LA Lakers @ 7:30pm

**Holiday Hoops East Pack** (*Club 200 Sideline-\$328, Club 200 Baseline- \$283*)

Fri 1/9 vs Cleveland @ 7:30pm  
Wed 1/14 vs Miami @ 7:30pm  
Tues 1/27 vs Chicago @ 7:30pm  
Sat 3/14 vs New York @ 7:30pm

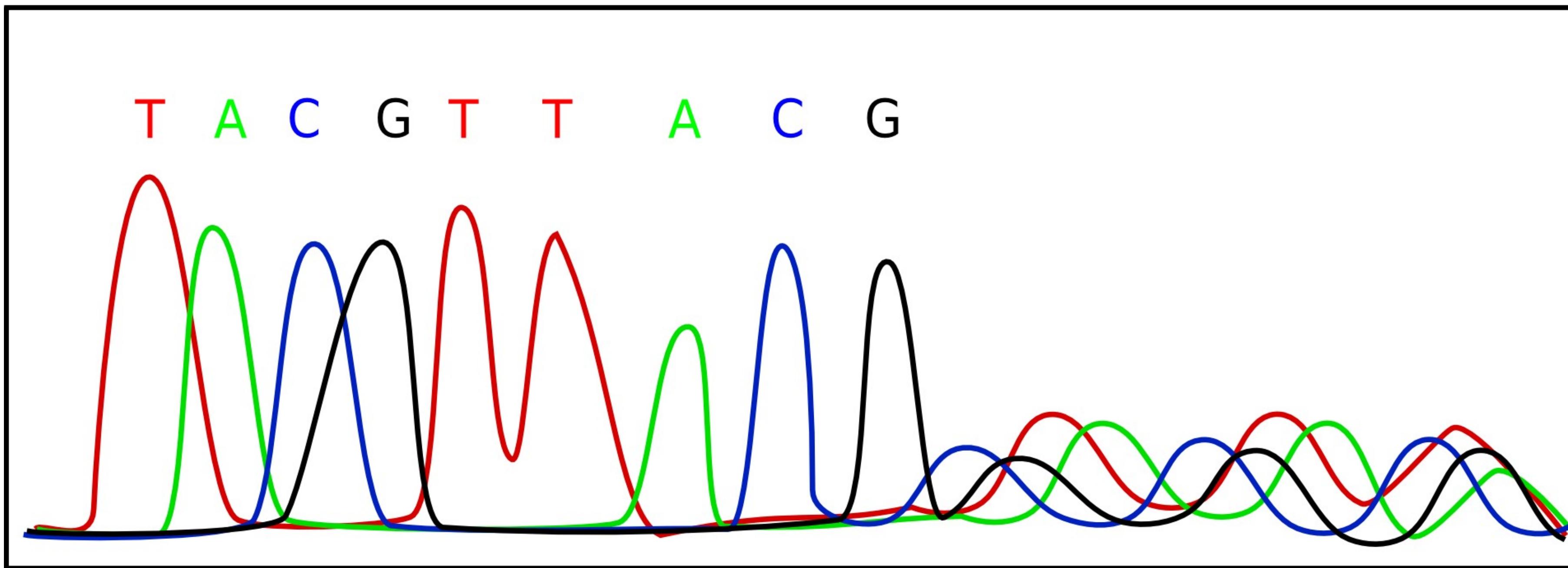
\*Ability to exchange one game for a different date if needed.

**Flex Plan 6+**  
If you are looking to attend 6 or more game games then you are able to pick any games from the remaing of the schedule.

If you would like to purchase one or have any questions/concerns give me a call.

Email

# Data Types



Genomic Data, e.g., SNPs

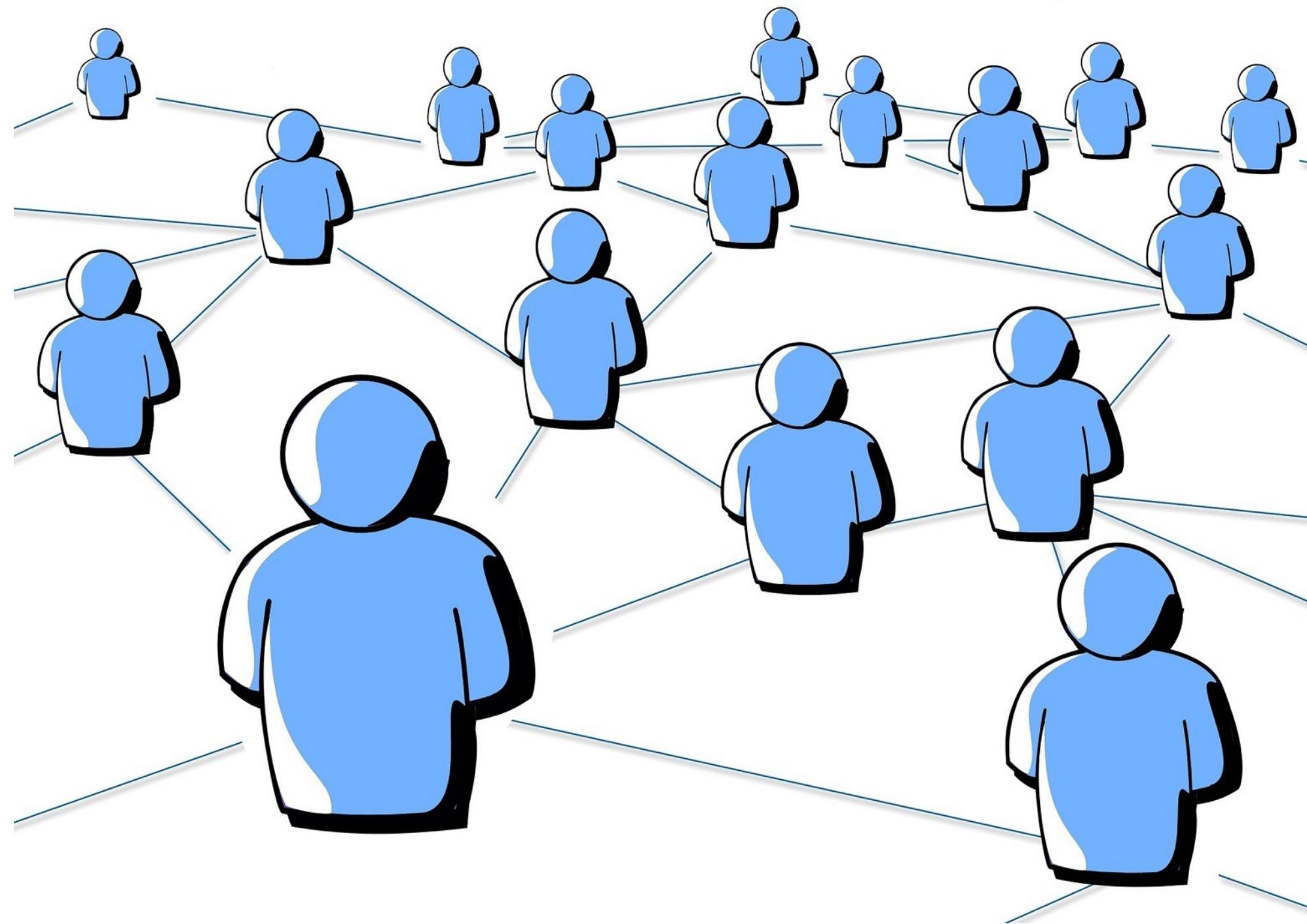
# Data Types



Images

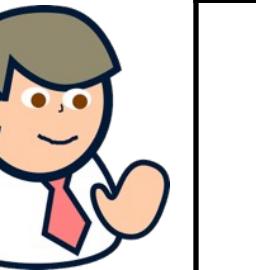
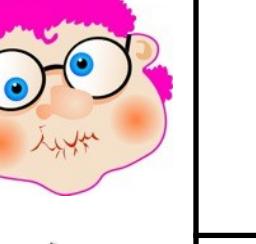
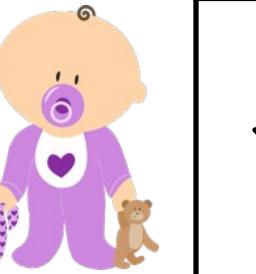
# Data Types

(Social) Networks / Graphs

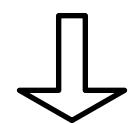


# Data Types

## User Ratings

			
	★	★★★★	
	★	★★★	★★
	★★★★		★
	★		★★
		★★★	★★
	★★★★	★★	

Obtain Raw Data



Feature Extraction

Initial observations can be in arbitrary format

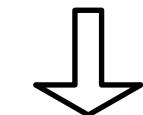
We extract *features* to represent observations

We can incorporate domain knowledge

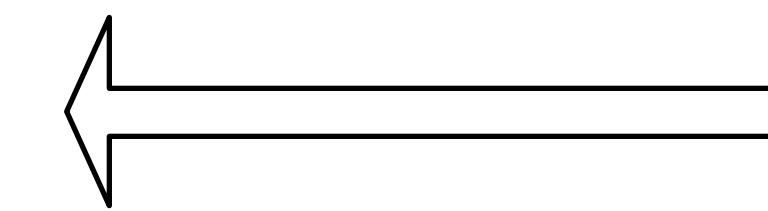
We typically want numeric features

Success of entire pipeline often depends on  
choosing good descriptions of observations!!

Obtain Raw Data



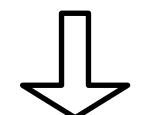
Feature Extraction



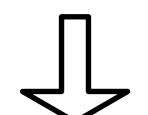
Unsupervised  
Learning

Preprocessing step for supervised learning

Obtain Raw Data



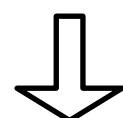
Feature Extraction



Supervised Learning

Train a supervised model using labeled data,  
e.g., Classification or Regression model

Obtain Raw Data



Feature Extraction

Q: How do we determine the quality of the model we've just trained?

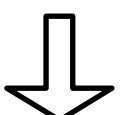
Supervised Learning

A: We can evaluate it on test / hold-out data, i.e., labeled data not used for training

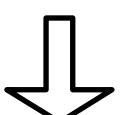
Evaluation

If we don't like the results, we iterate...

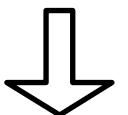
Obtain Raw Data



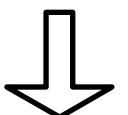
Feature Extraction



Supervised Learning



Evaluation

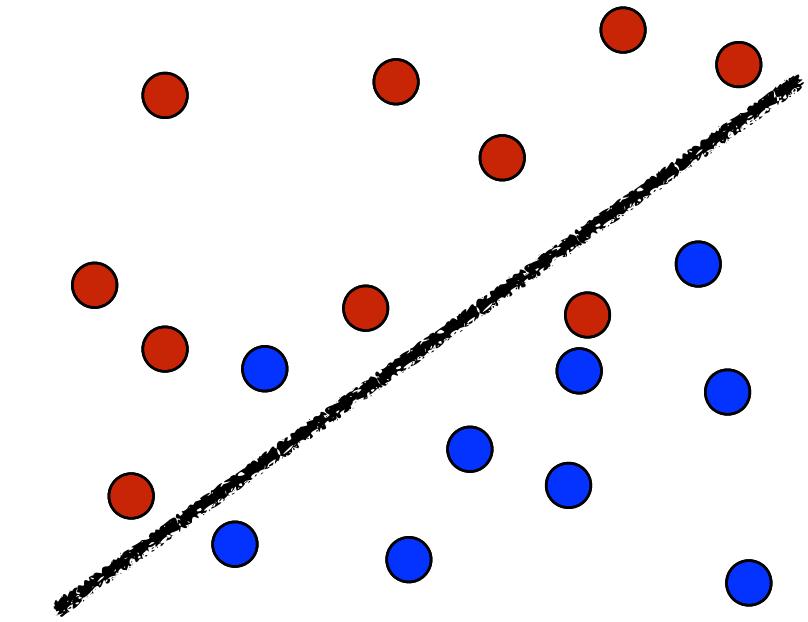


Predict

Once we're happy with our model, we can use it to make predictions on future observations, i.e., data without a known label

# Sample Classification Pipeline

# Classification

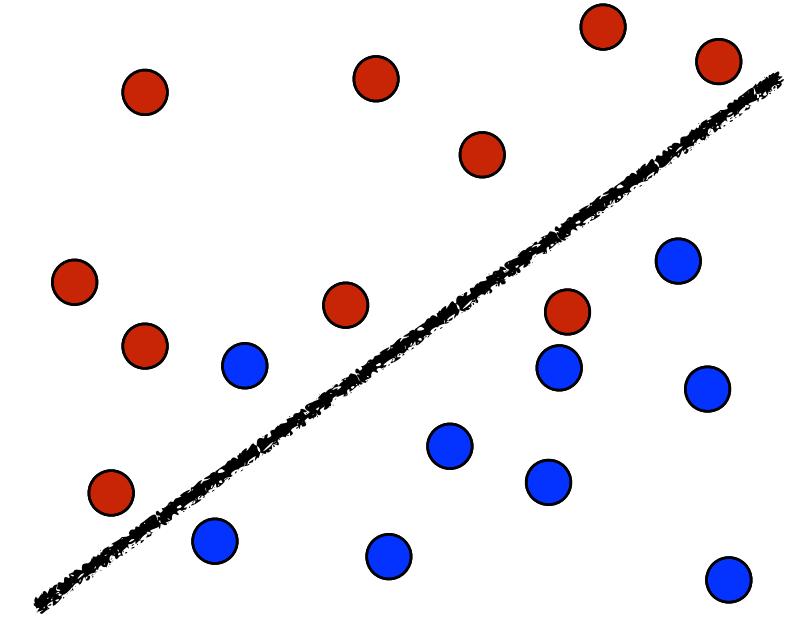


Goal: Learn a mapping from observations to discrete labels given a set of training examples (supervised learning)

## Example Spam Classification

- Observations are emails
- Labels are {spam, not-spam} (Binary Classification)
- Given a set of labeled emails, we want to predict whether a new email is spam or not-spam

# Other Examples



**Fraud detection:** User activity  $\rightarrow \{\text{fraud}, \text{not fraud}\}$

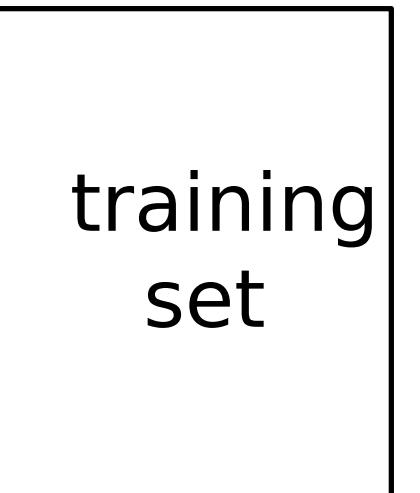
**Face detection :** Images  $\rightarrow$  set of people

**Link prediction:** Users  $\rightarrow \{\text{suggest link}, \text{don't suggest link}\}$

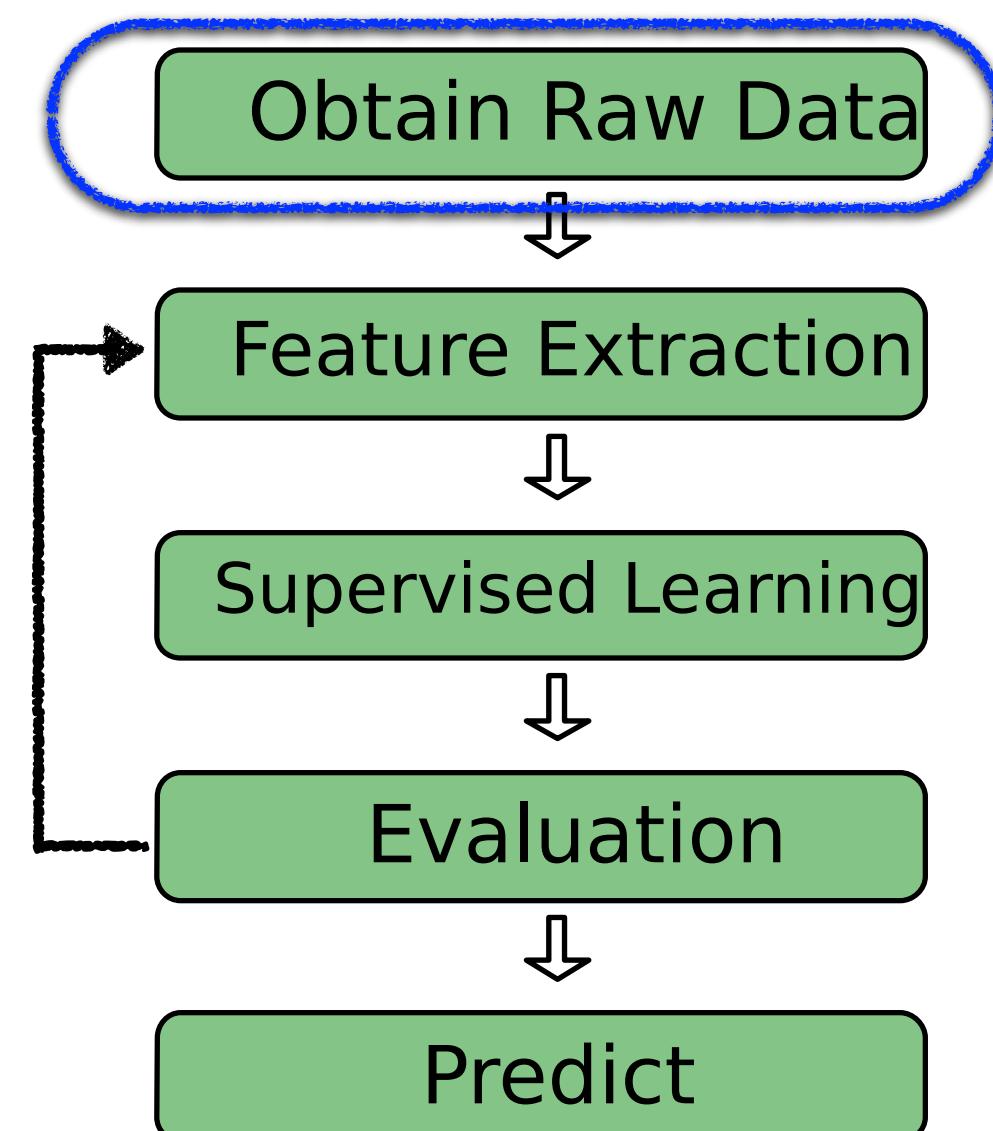
**Clickthrough rate prediction:** User and ads  $\rightarrow \{\text{click}, \text{no click}\}$

Many others...

# Classification Pipeline



**Raw data** consists of a set of labeled training observations



# E.g., Spam Classification

Observation

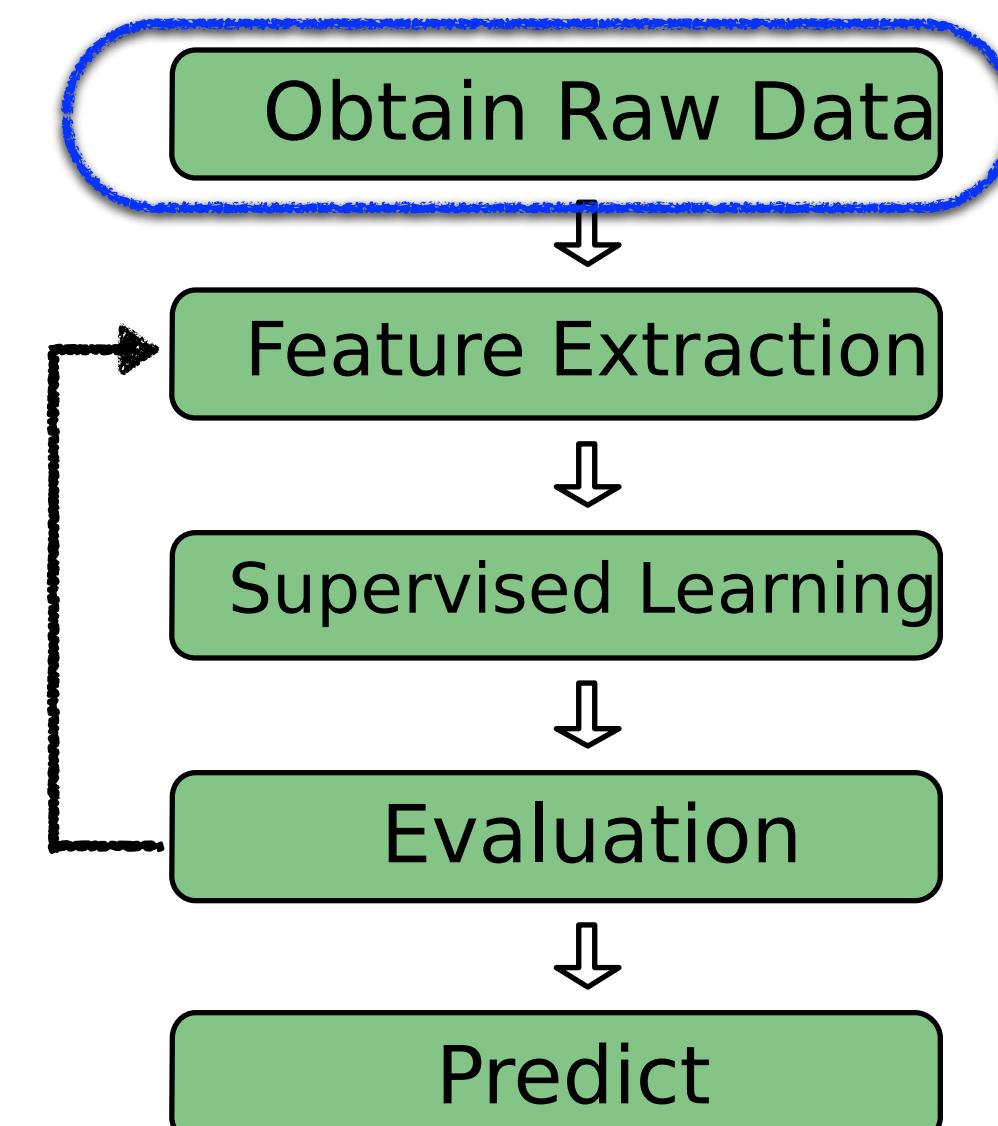
```
From: illegitimate@bad.com  
  
"Eliminate your debt by  
giving us your money..."
```

Label

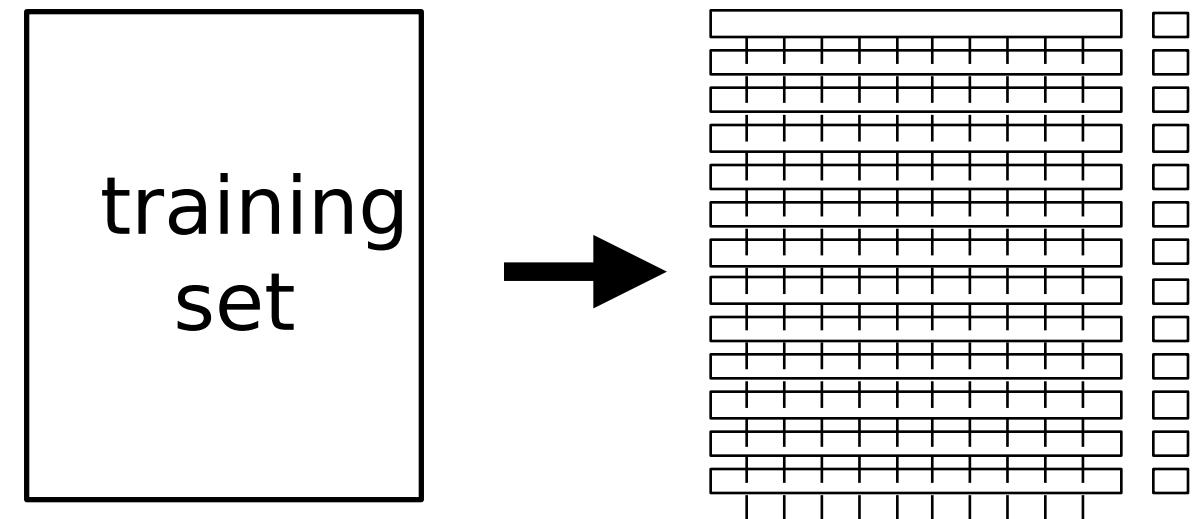
spam

```
From: bob@good.com  
  
"Hi, it's been a while!  
How are you? ..."
```

not-spam

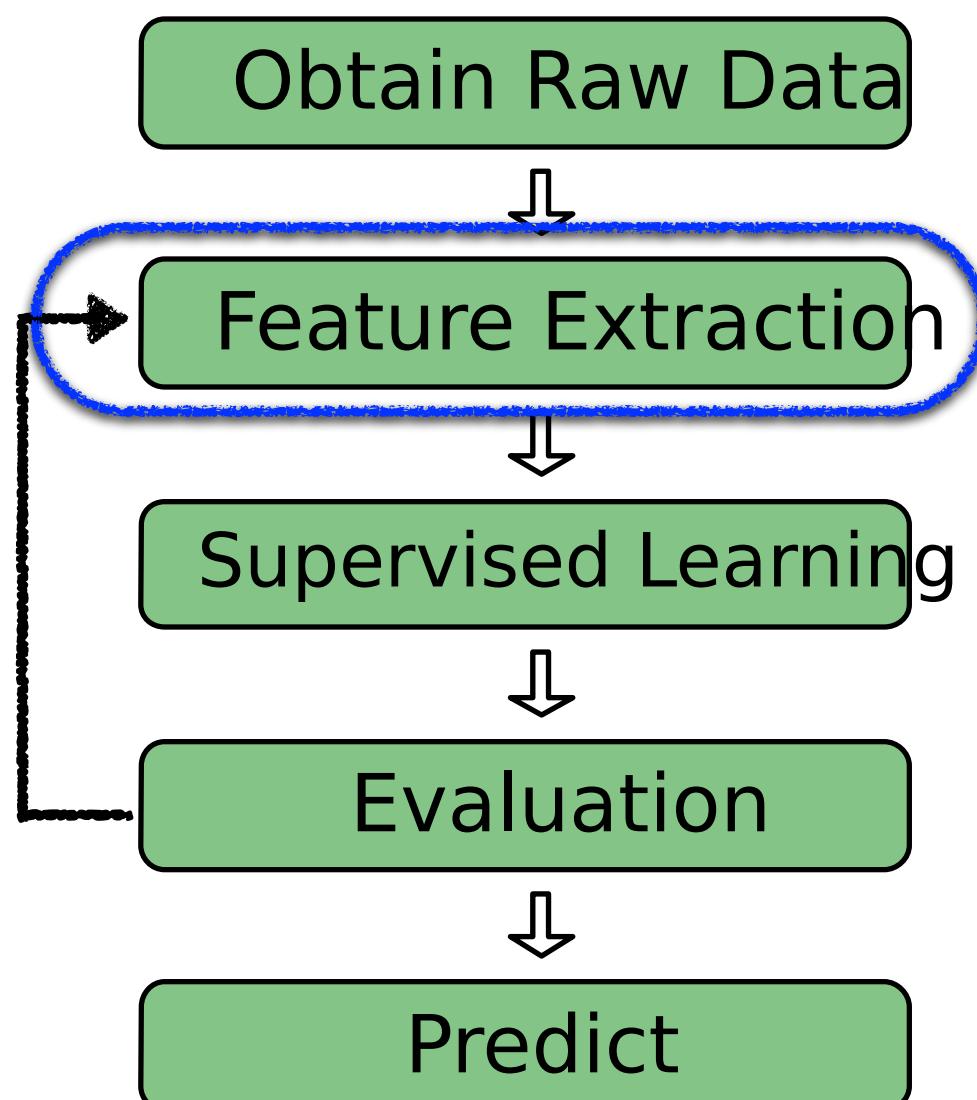


# Classification Pipeline

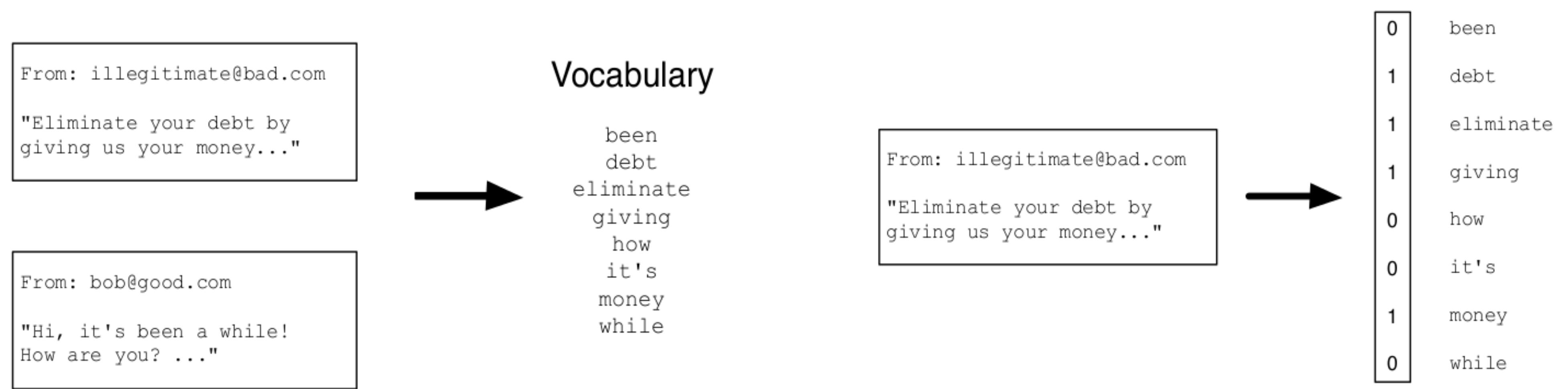


**Feature extraction** typically transforms each observations into a vector of real numbers (features)

Success or failure of a classifier often depends on choosing good descriptions of observations!!



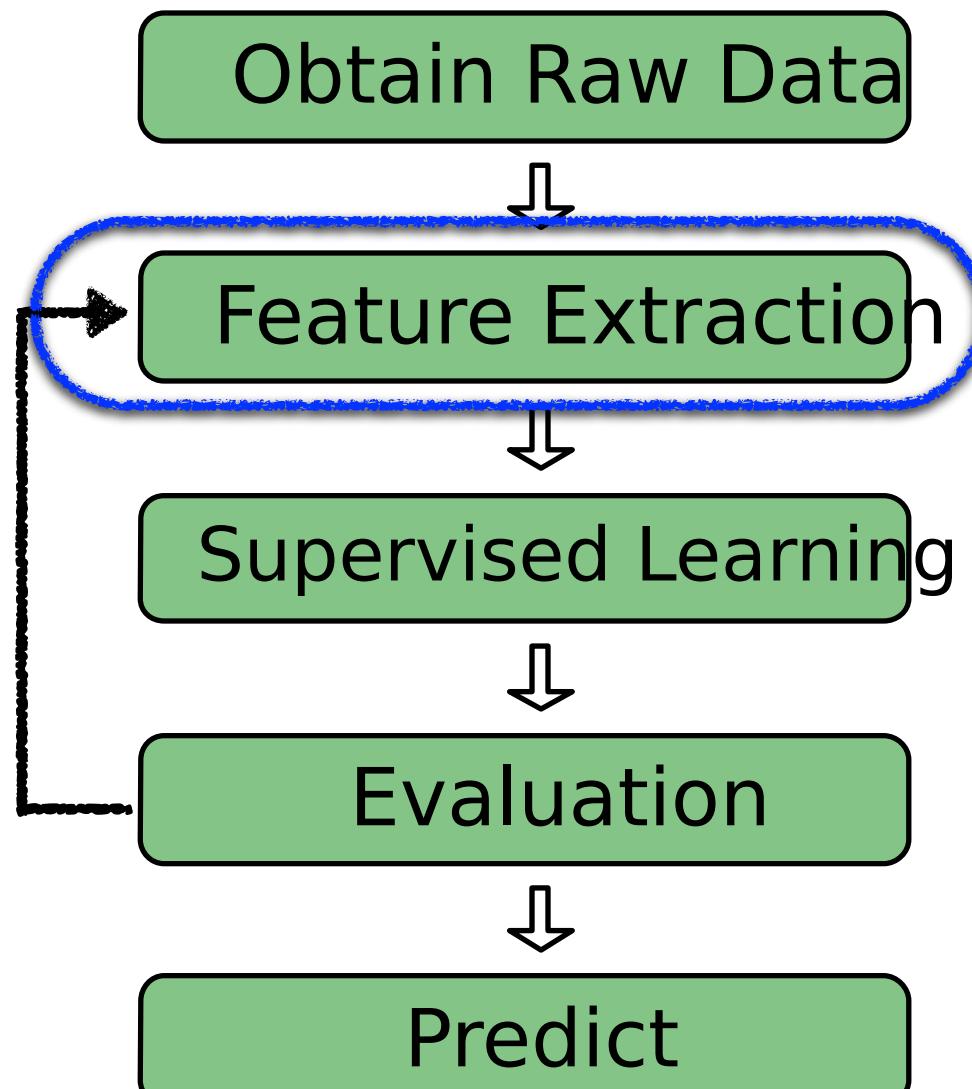
# E.g., “Bag of Words”



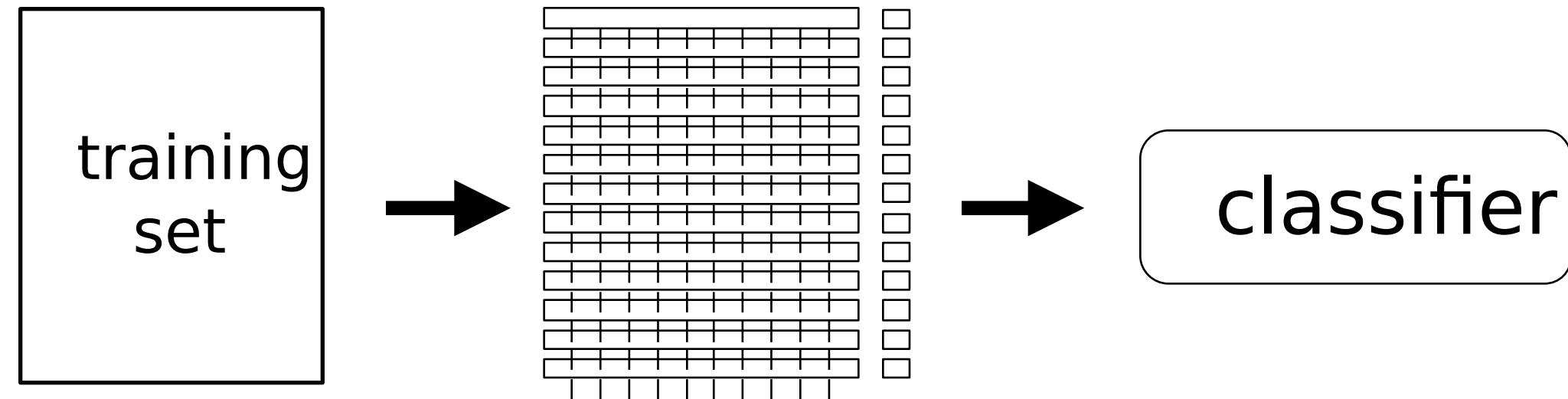
Observations are documents

Build Vocabulary

Derive feature vectors from Vocabulary



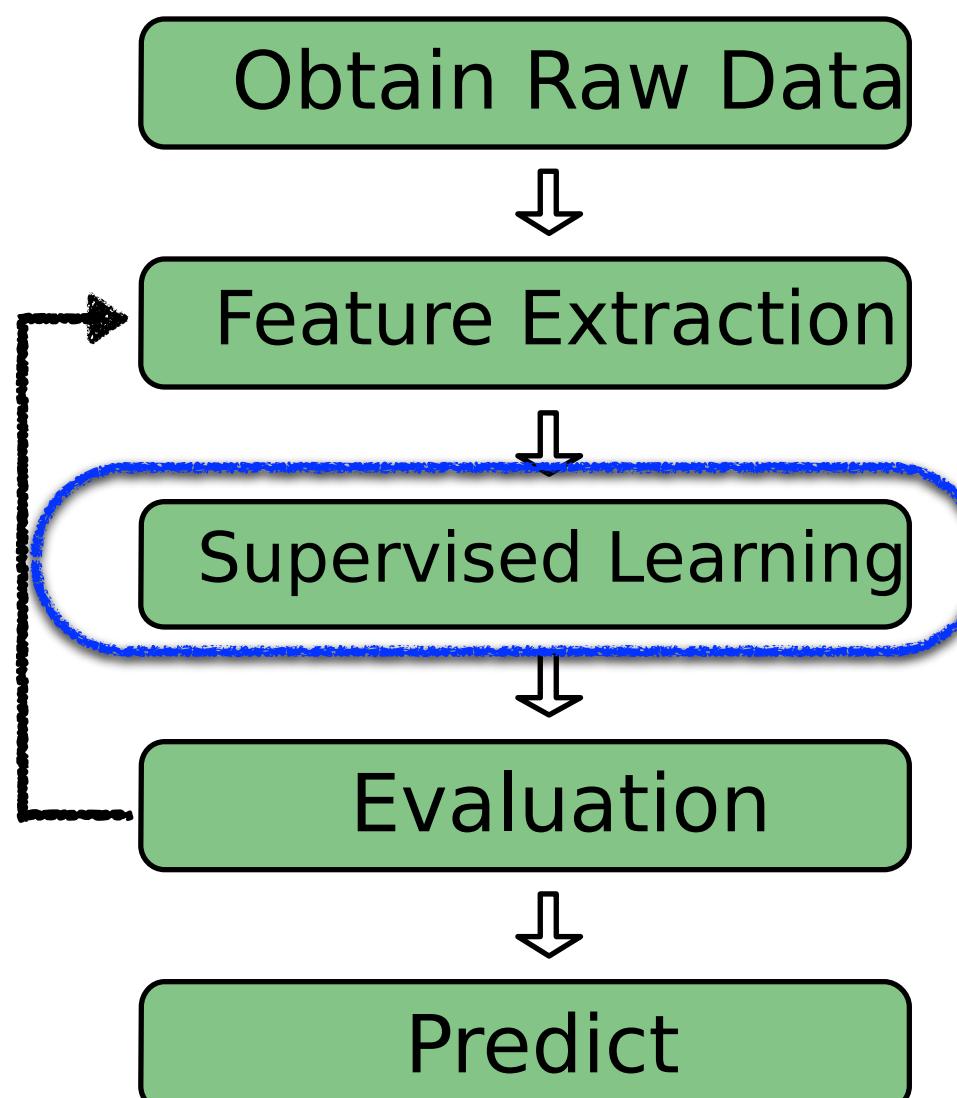
# Classification Pipeline



**Supervised Learning:** Train classifier using training data

- Common classifiers include Logistic Regression, SVMs, Decision Trees, Random Forests, etc.

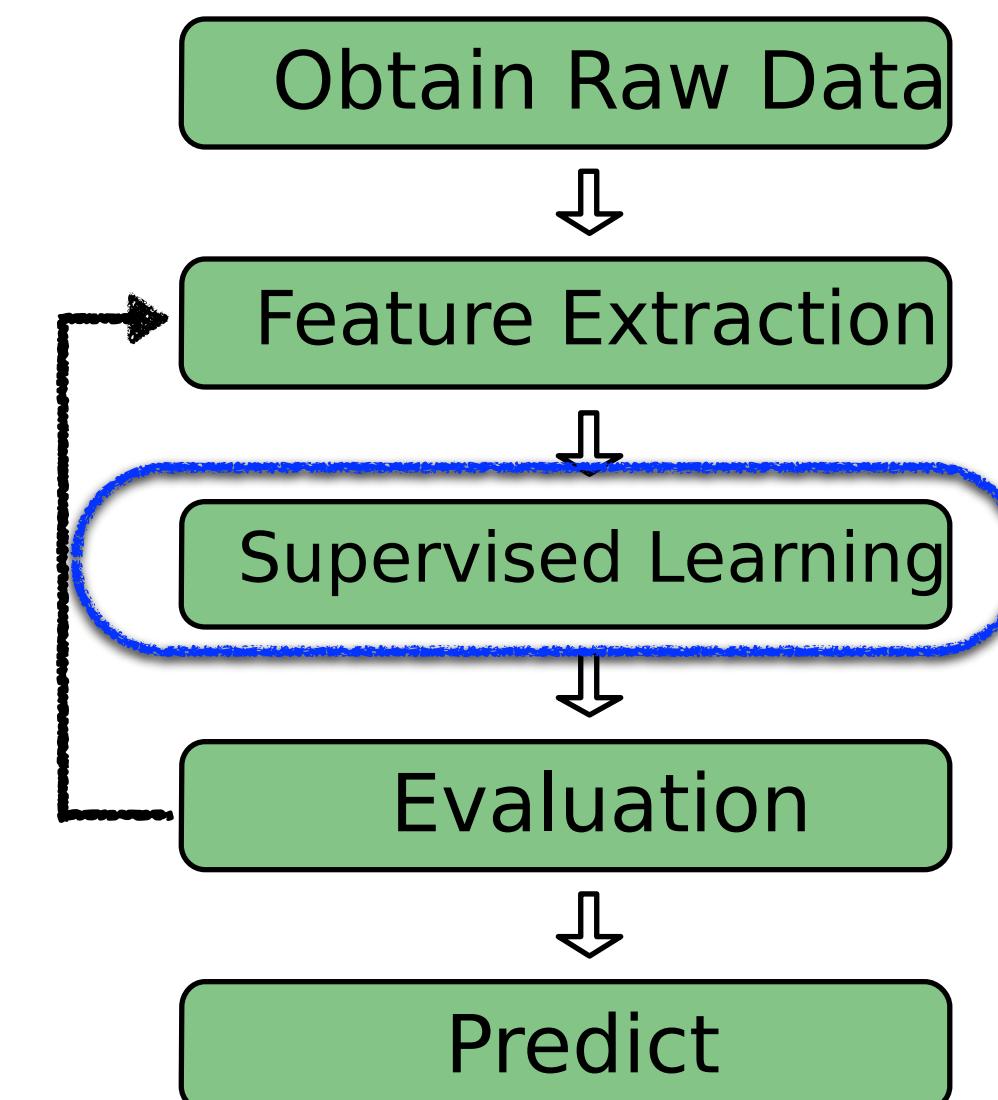
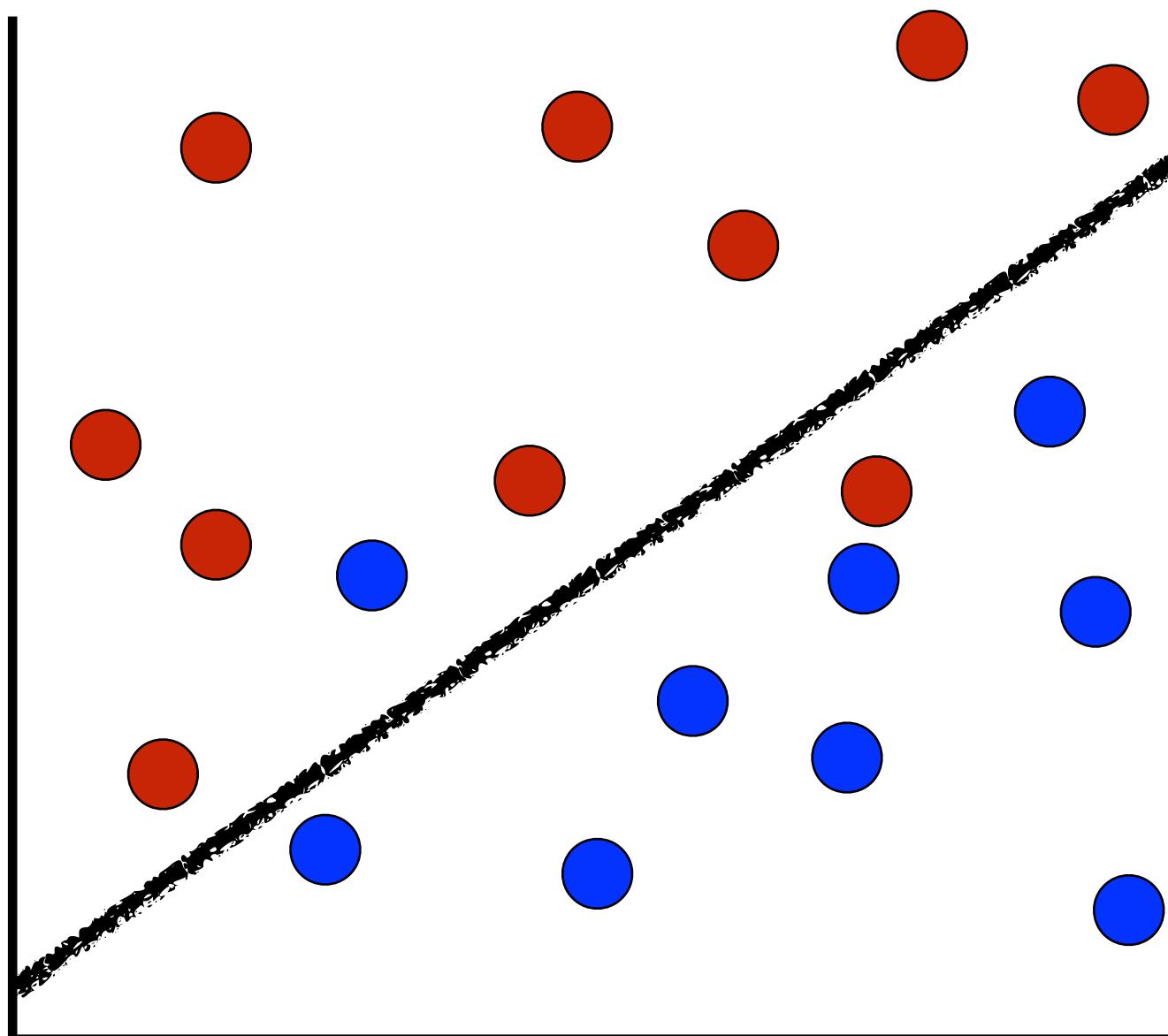
Training (especially at scale) often involves iterative computations, e.g., gradient descent



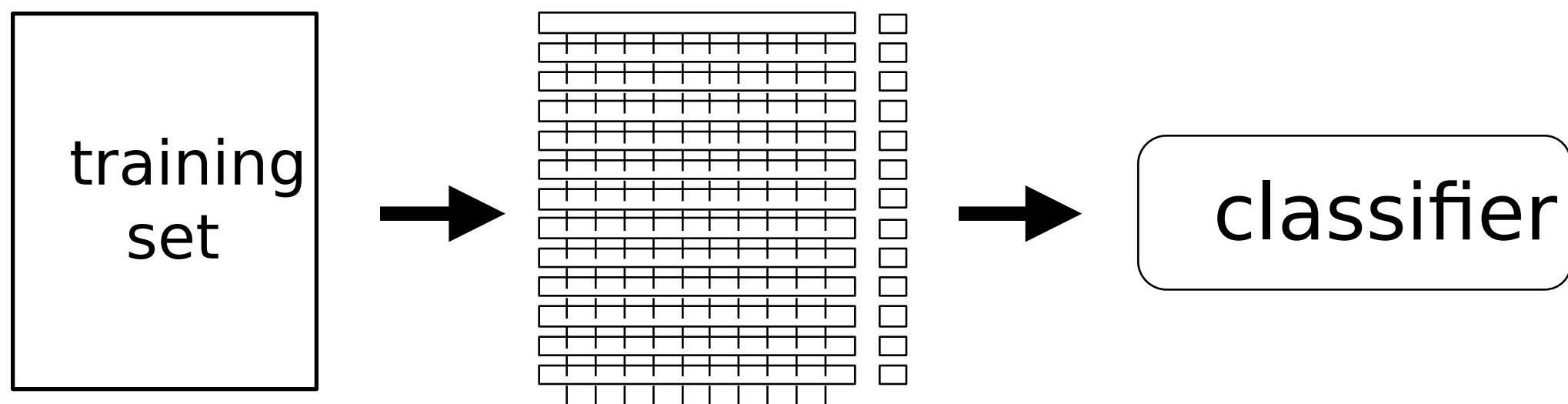
# E.g., Logistic Regression

Goal: Find linear decision boundary

- Parameters to learn are feature weights and offset
- Nice probabilistic interpretation
- Covered in more detail later in course



# Classification Pipeline



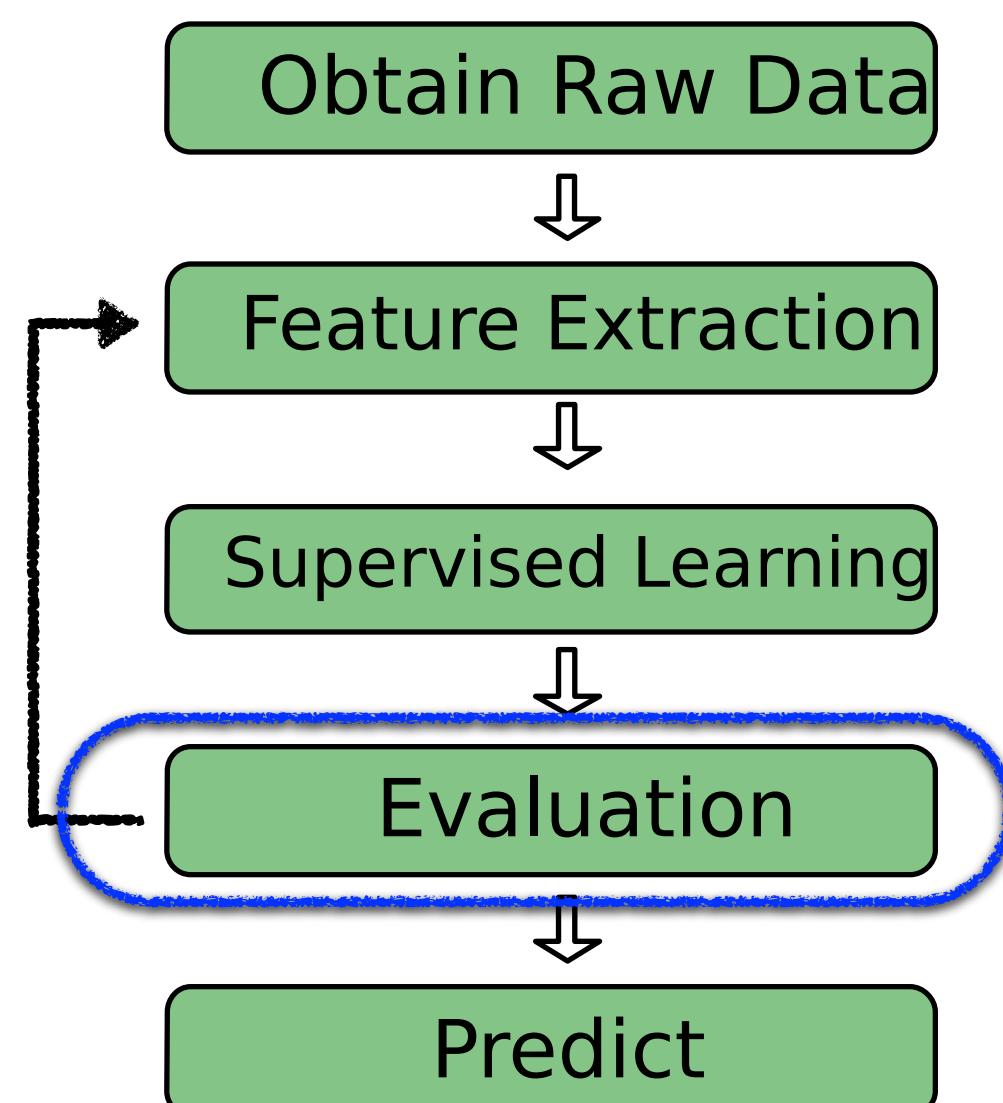
How can we evaluate the quality of our classifier?

We want good predictions on unobserved data

- 'Generalization' ability

Accuracy on training data is overly optimistic since classifier has already learned from it

- We might be 'overfitting'



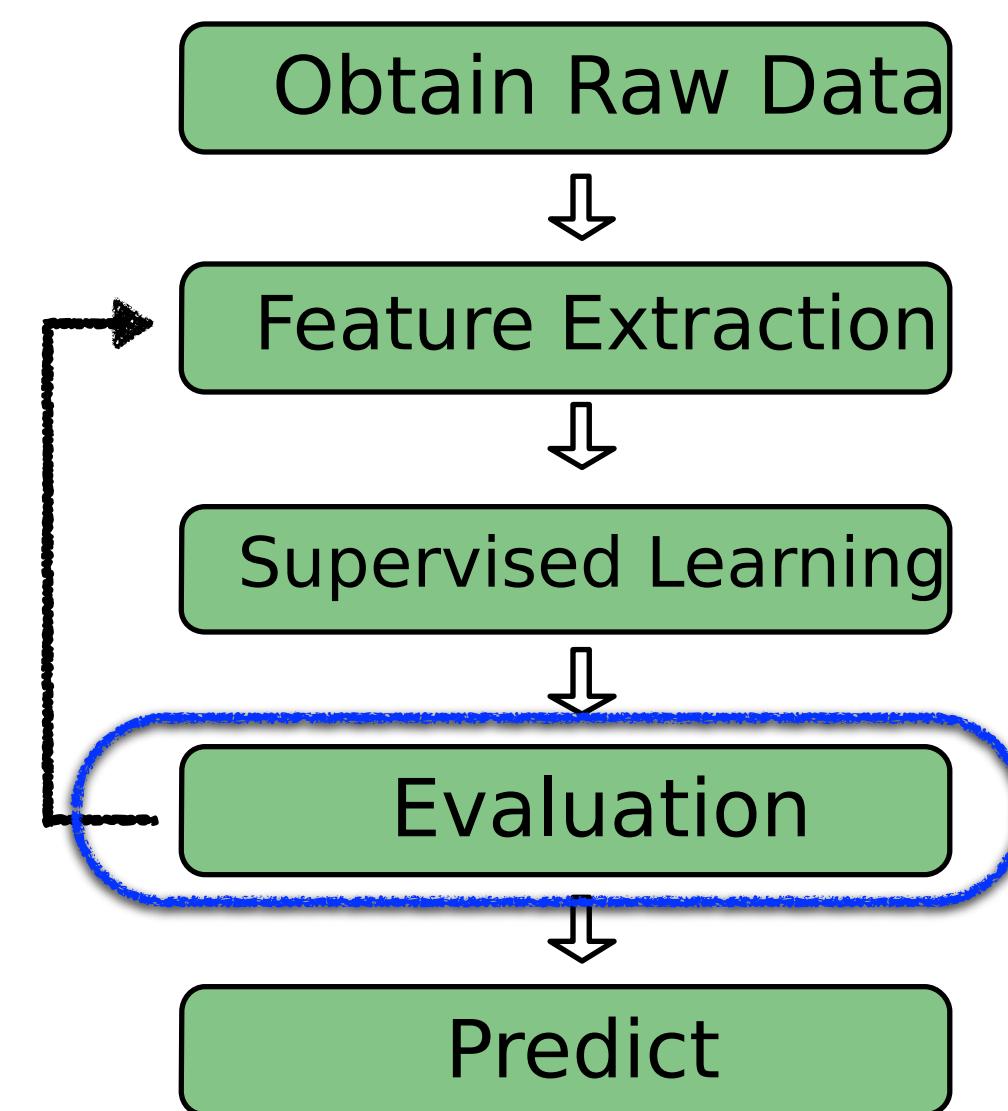
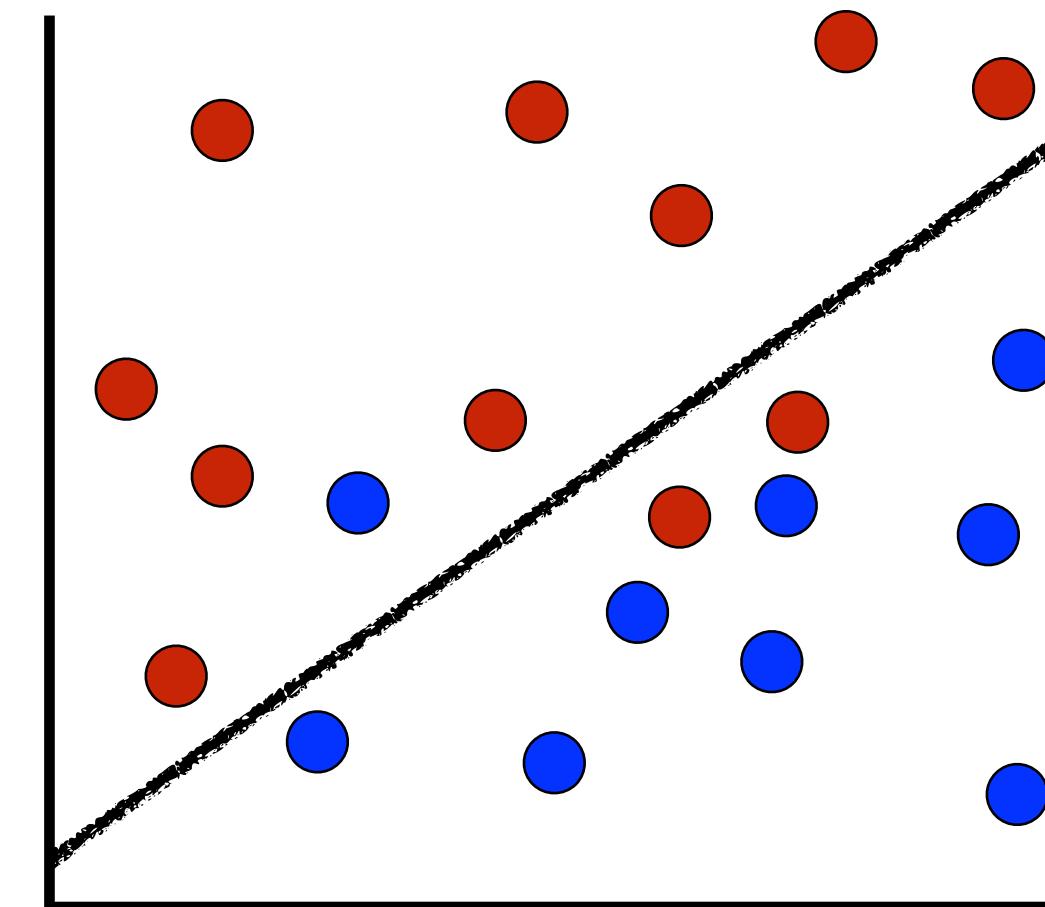
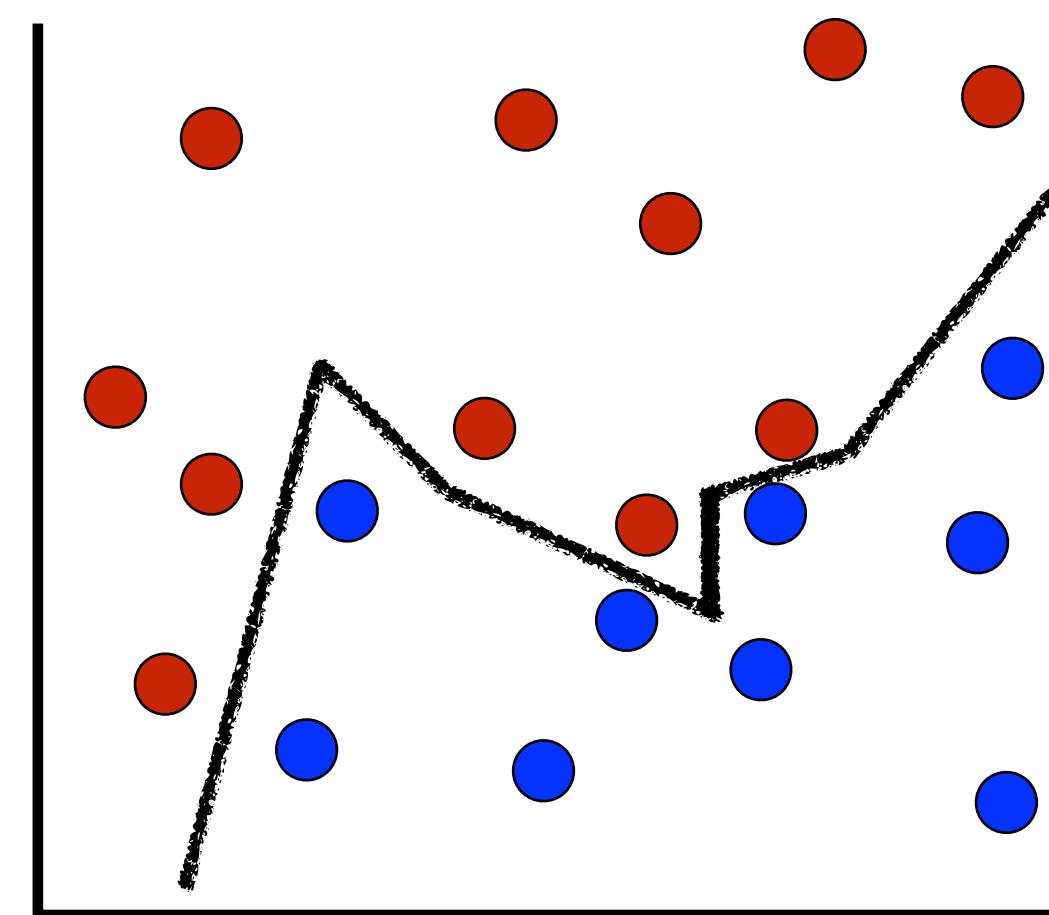
# Overfitting and Generalization

Fitting training data does not guarantee generalization, e.g., lookup table

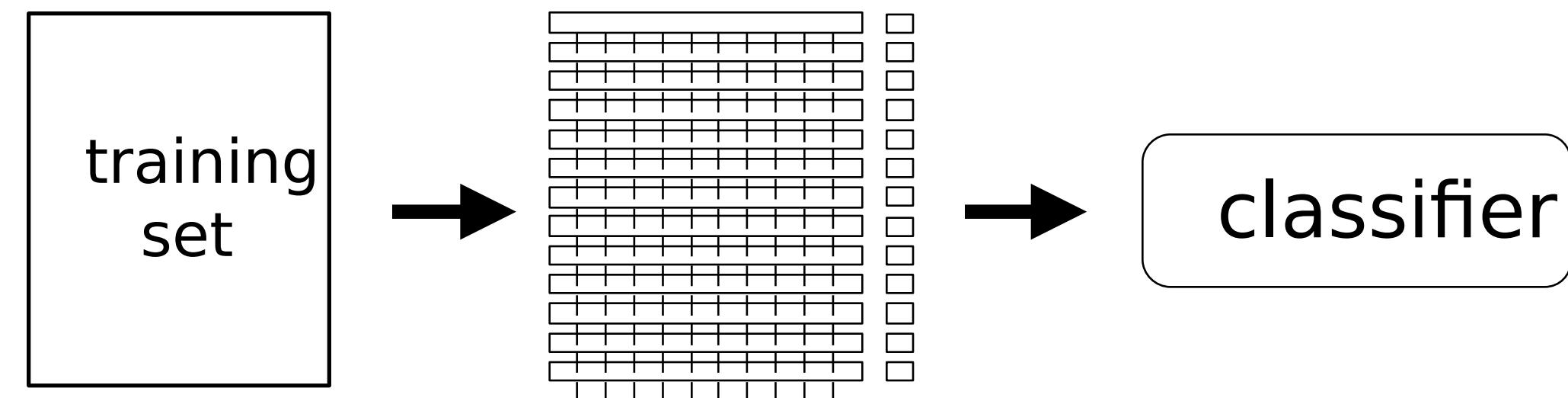
Left: perfectly fits training samples, but it is complex / overfitting

Right: misclassifies a few points, but simple / generalizes

Occam's razor

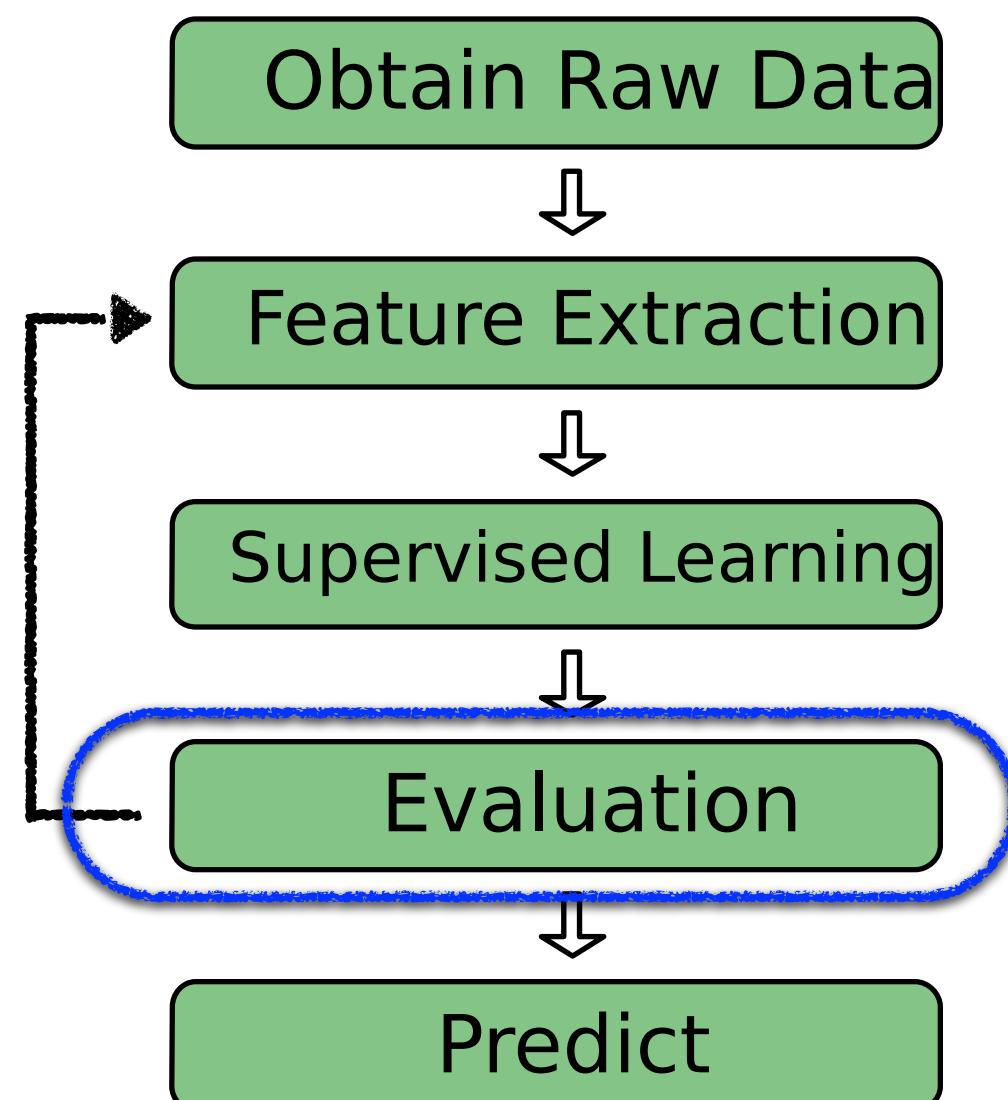


# Classification Pipeline

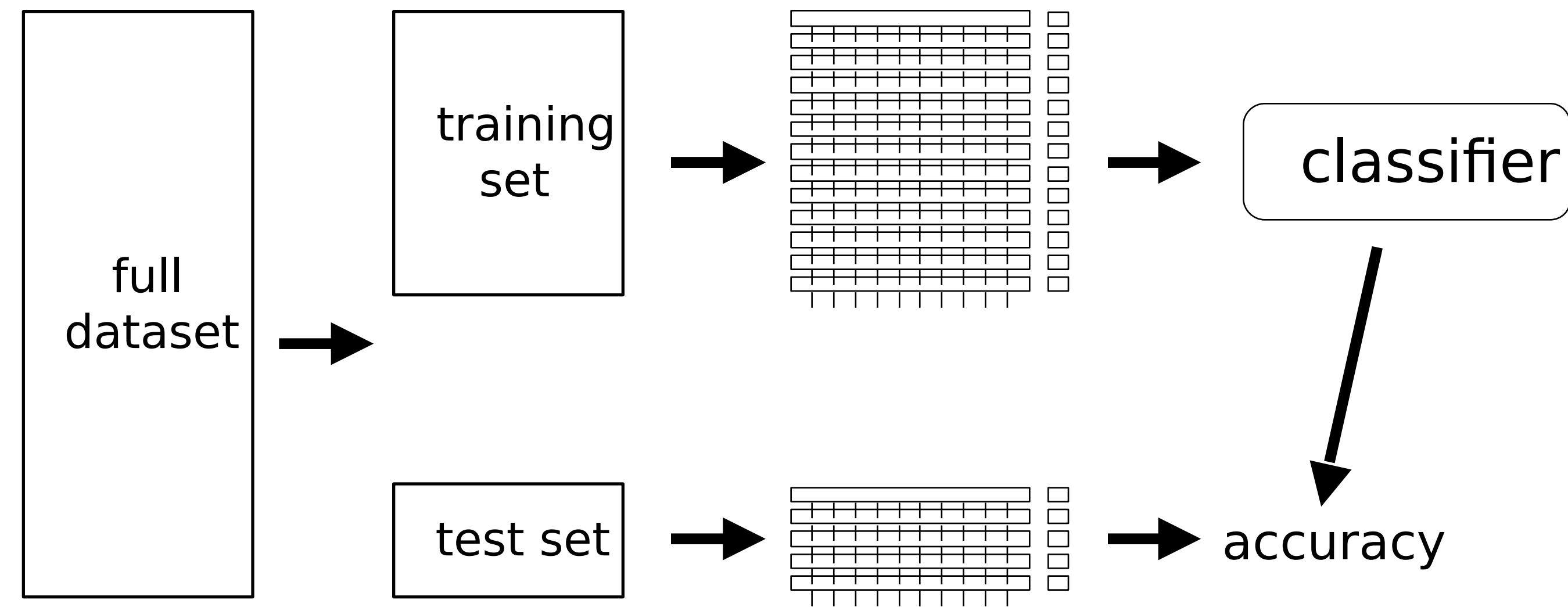


How can we evaluate the quality of our classifier?

Idea: Create test set to simulate unobserved data

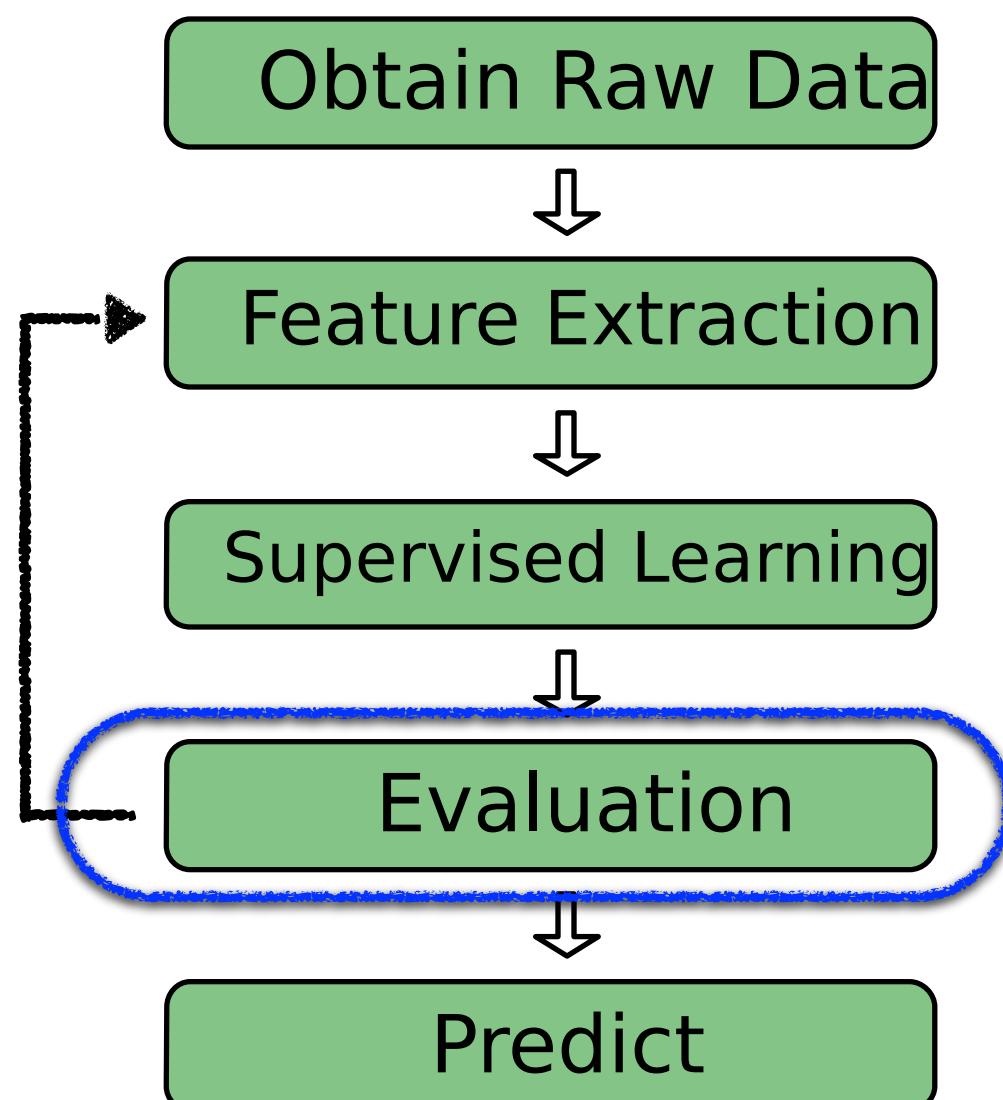


# Classification Pipeline

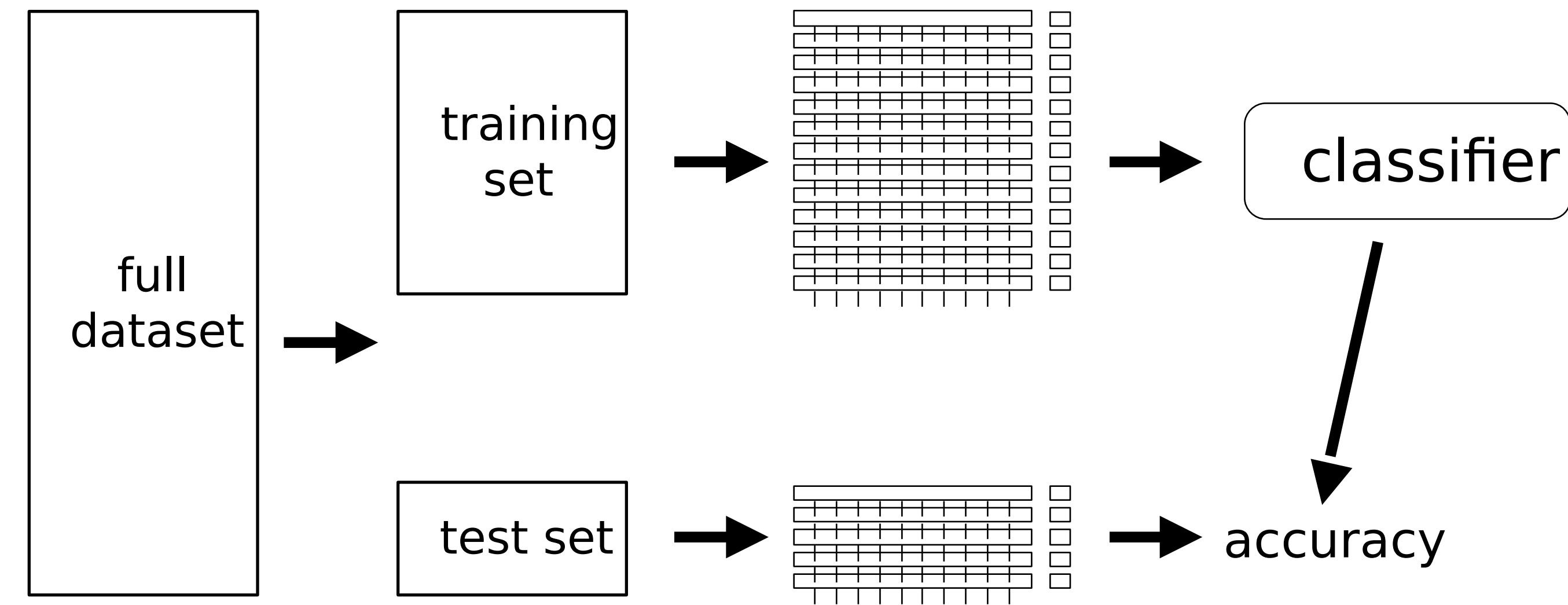


**Evaluation:** Split dataset into training / testing datasets

- Train on training set (don't expose test set to classifier)
- Make predictions on test set (ignoring test labels)
- Compare test predictions with underlying test labels

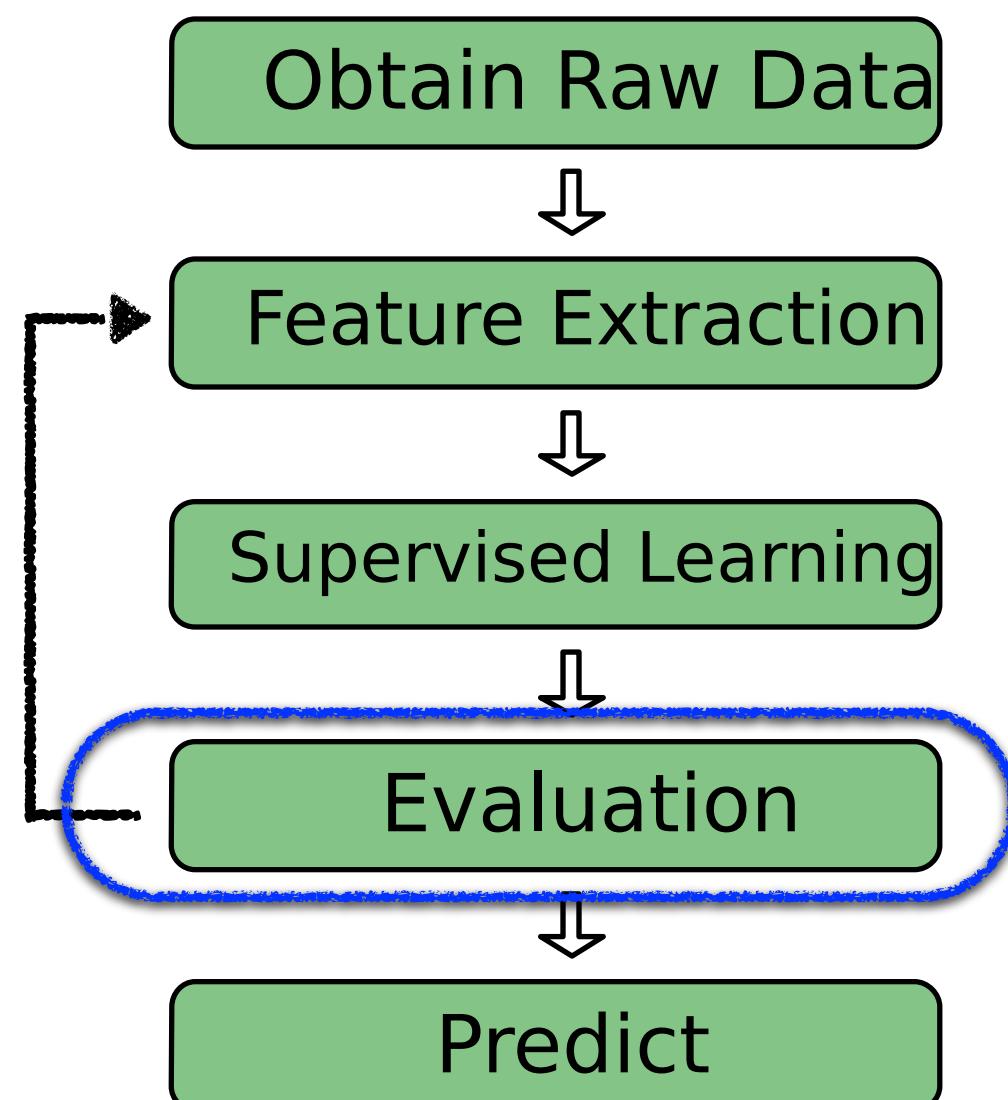


# Classification Pipeline

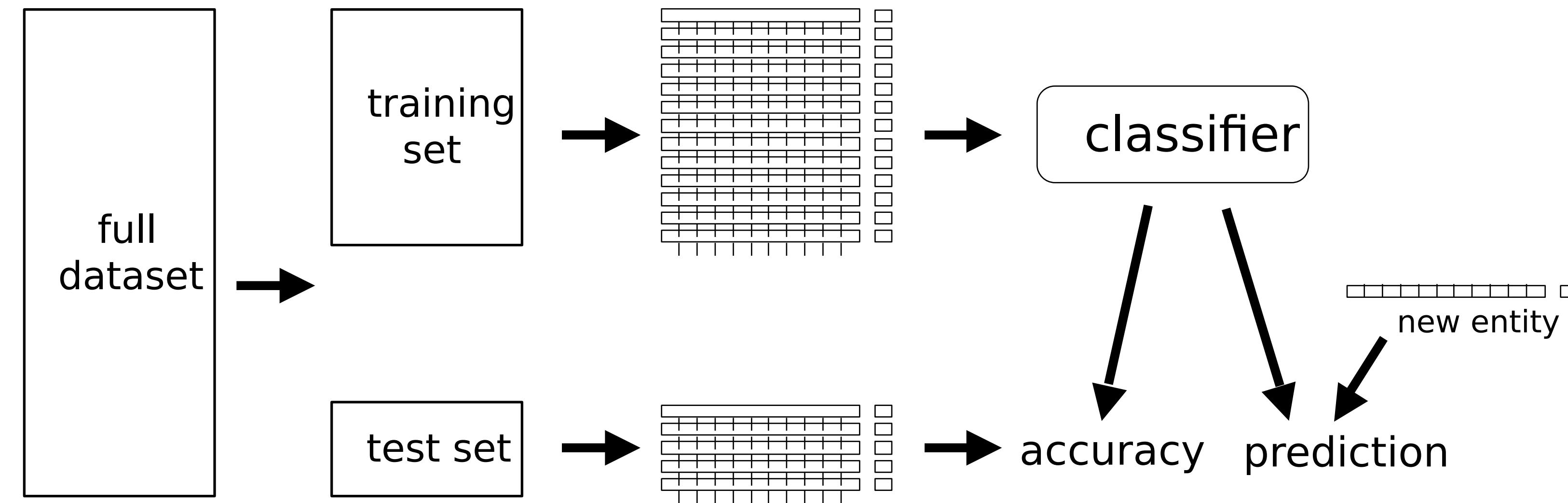


**Evaluation:** Split dataset into training / testing datasets

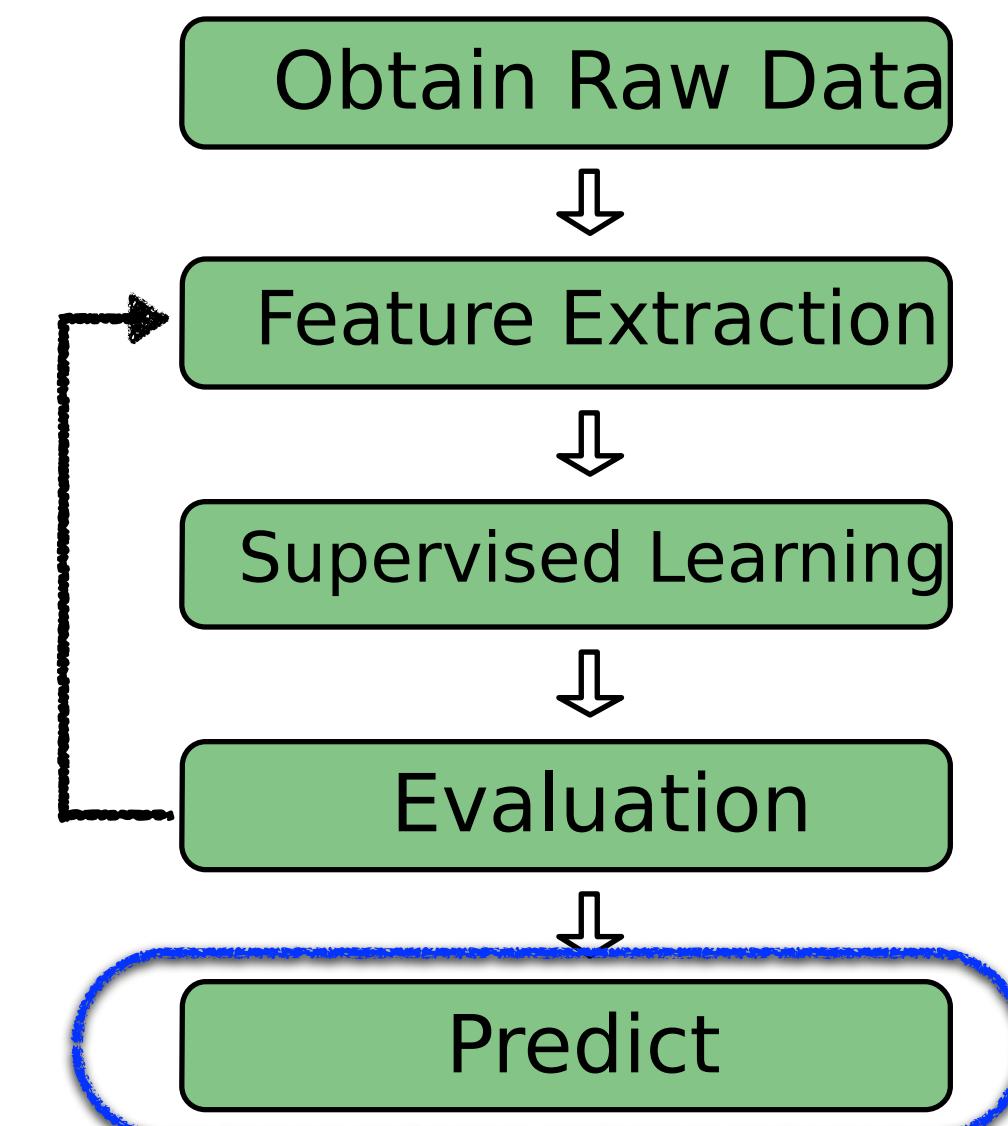
- Various ways to compare predicted and true labels
- Evaluation criterion is called a ‘loss’ function
- Accuracy (or 0-1 loss) is common for classification



# Classification Pipeline



**Predict:** Final classifier can then be used to make predictions on future observations, e.g., new emails we receive



# **Linear Algebra Review**

# Matrices

A matrix is a 2-dimensional array

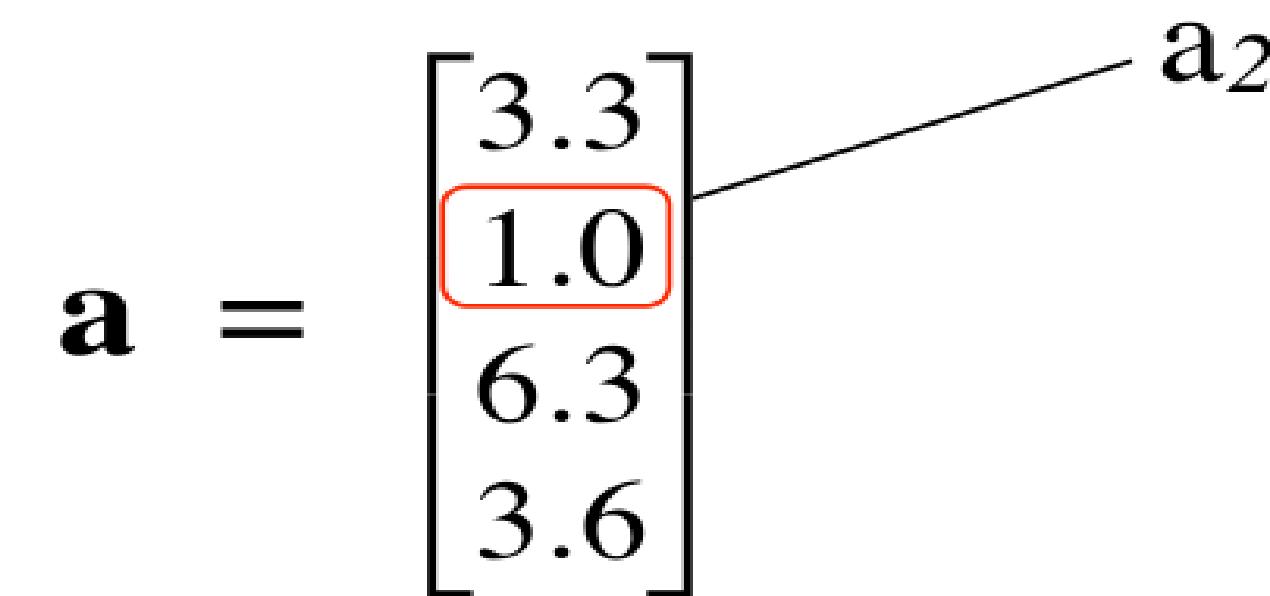
$$\mathbf{A} = \begin{matrix} & & \mathbf{A}_{11} \\ \begin{matrix} 3.3 & 5.3 & 4.5 \\ 1.0 & 4.5 & 3.4 \\ 6.3 & 1.0 & 2.2 \\ 3.6 & 4.7 & 8.9 \end{matrix} & & \mathbf{A}_{32} \\ 4 \times 3 & & \end{matrix}$$

Notation:

- Matrices are denoted by bold uppercase letters
- $\mathbf{A}_{ij}$  denotes the entry in  $i$ th row and  $j$ th column
- If  $\mathbf{A}$  is  $n \times m$ , it has  $n$  rows and  $m$  columns
- If  $\mathbf{A}$  is  $n \times m$ , then  $\mathbf{A} \in \mathbb{R}^{n \times m}$

# Vectors

A vector is a matrix with many rows and one column

$$\mathbf{a} = \begin{bmatrix} 3.3 \\ 1.0 \\ 6.3 \\ 3.6 \end{bmatrix}$$


Notation:

- Vectors are denoted by bold lowercase letters
- $\mathbf{a}_i$  denotes the  $i$ th entry
- If  $\mathbf{a}$  is  $m$  dimensional, then  $\mathbf{a} \in \mathbb{R}^m$

# Transpose

Swap the rows and columns of a matrix

$A_{12}$

$$\begin{bmatrix} 1 & 4 \\ 6 & 1 \\ 3 & 5 \end{bmatrix}$$

$3 \times 2$

$(A^\top)_{21}$

$$\begin{bmatrix} 1 & 6 & 3 \\ 4 & 1 & 5 \end{bmatrix}$$

$2 \times 3$

$$\begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix}$$

$3 \times 1$

$$\begin{bmatrix} 3 & 4 & 1 \end{bmatrix}$$

$1 \times 3$

Properties of matrix transposes:

- $A_{ij} = (A^\top)_{ji}$
- If  $\mathbf{A}$  is  $n \times m$ , then  $\mathbf{A}^\top$  is  $m \times n$

# Addition and Subtraction

These are element-wise operations

Addition:

$$\begin{bmatrix} 3 & 5 \\ 6 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 5 \\ 8 & 12 \end{bmatrix} = \begin{bmatrix} 3+4 & 5+5 \\ 6+8 & 1+12 \end{bmatrix}$$
$$= \begin{bmatrix} 7 & 10 \\ 14 & 13 \end{bmatrix}$$

Subtraction:

$$\begin{bmatrix} 5 \\ 1 \end{bmatrix} - \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 5-4 \\ 1-3 \end{bmatrix}$$
$$= \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

# Addition and Subtraction

The matrices must have the same dimensions

$$\begin{bmatrix} 5 \\ 1 \end{bmatrix} - \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 5 & 4 \\ 6 & 1 & 2 \end{bmatrix} + \begin{bmatrix} 4 & 5 & 1 \\ 8 & 12 & 9 \end{bmatrix} = \begin{bmatrix} 7 & 10 & 5 \\ 14 & 13 & 11 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 5 & 4 \\ 6 & 1 & 2 \end{bmatrix} + \begin{bmatrix} 4 & 5 \\ 8 & 12 \end{bmatrix} \neq$$

# Matrix Scalar Multiplication

We multiply each matrix element by the scalar value

$$3 \times \begin{bmatrix} 3 & 5 & 4 \\ 6 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 9 & 15 & 12 \\ 18 & 3 & 6 \end{bmatrix}$$

$$-0.5 \times \begin{bmatrix} 3 \\ 8 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -4 \end{bmatrix}$$

# Scalar Product

A function that maps two vectors to a scalar

$$\begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 2 \\ -7 \end{bmatrix} = -9$$

$1 \times 4$

Performs pairwise multiplication of vector elements

# Scalar Product

A function that maps two vectors to a scalar

$$\begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 2 \\ -7 \end{bmatrix} = -9$$

$$1 \times 4 + 4 \times 2$$

Performs pairwise multiplication of vector elements

# Scalar Product

A function that maps two vectors to a scalar

$$\begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 2 \\ -7 \end{bmatrix} = -9$$

$$1 \times 4 + 4 \times 2 + 3 \times (-7) = -9$$

Performs pairwise multiplication of vector elements

The two vectors must be the same dimension

Also known as *dot* product or *inner* product

# Matrix-Vector Multiplication

Involves repeated scalar products

$$\begin{bmatrix} 1 & 4 & 3 \\ 6 & 1 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ -7 \end{bmatrix} = \begin{bmatrix} -9 \end{bmatrix}$$

$$1 \times 4 + 4 \times 2 + 3 \times (-7) = -9$$

# Matrix-Vector Multiplication

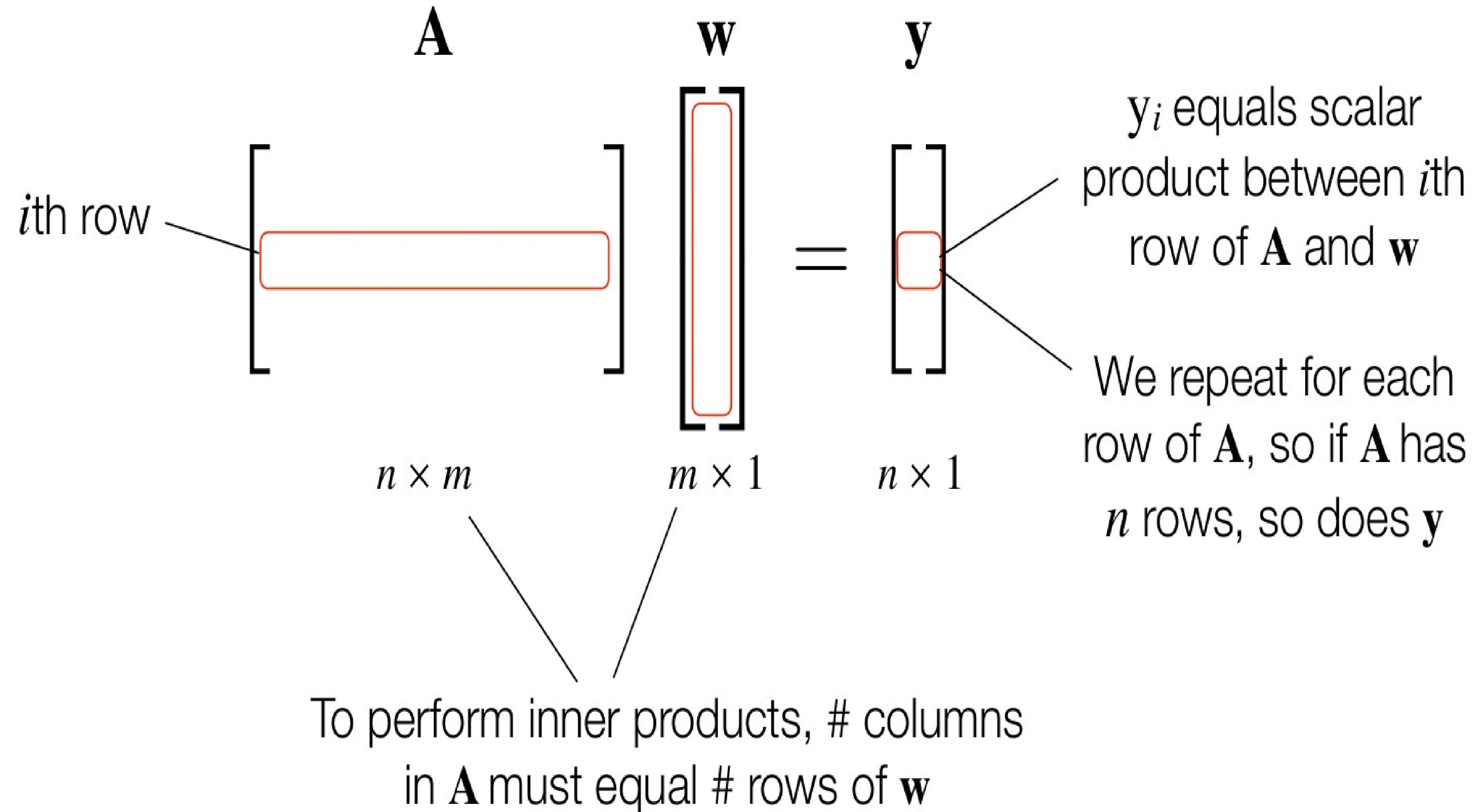
Involves repeated scalar products

$$\begin{bmatrix} 1 & 4 & 3 \\ 6 & 1 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ -7 \end{bmatrix} = \begin{bmatrix} -9 \\ 12 \end{bmatrix}$$

$$1 \times 4 + 4 \times 2 + 3 \times (-7) = -9$$

$$6 \times 4 + 1 \times 2 + 2 \times (-7) = 12$$

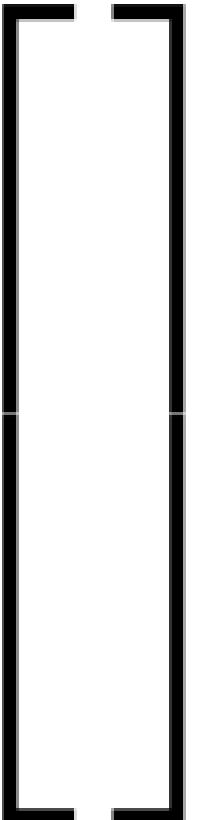
# Matrix-Vector Multiplication



# Scalar Product Revisited

Special case of Matrix-  
Vector Multiplication

$$\mathbf{x}^\top \quad \mathbf{w} \quad \mathbf{y}$$

  = scalar product

$1 \times m$        $m \times 1$       scalar ( $1 \times 1$ )

Vectors assumed to be in column form (many rows, one column)

Transposed vectors are row vectors

Common notation for scalar product:  $\mathbf{x}^\top \mathbf{w}$

# Matrix-Matrix Multiplication

Also involves several scalar products

$$\begin{bmatrix} 9 & 3 & 5 \\ 4 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & -5 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 28 \\ \end{bmatrix}$$

$$9 \times 1 + 3 \times 3 + 5 \times 2 = 28$$

# Matrix-Matrix Multiplication

Also involves several scalar products

$$\begin{bmatrix} 9 & 3 & 5 \\ 4 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & -5 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 28 & 18 \end{bmatrix}$$

$$9 \times 1 + 3 \times 3 + 5 \times 2 = 28$$

$$9 \times 2 + 3 \times (-5) + 5 \times 3 = 18$$

# Matrix-Matrix Multiplication

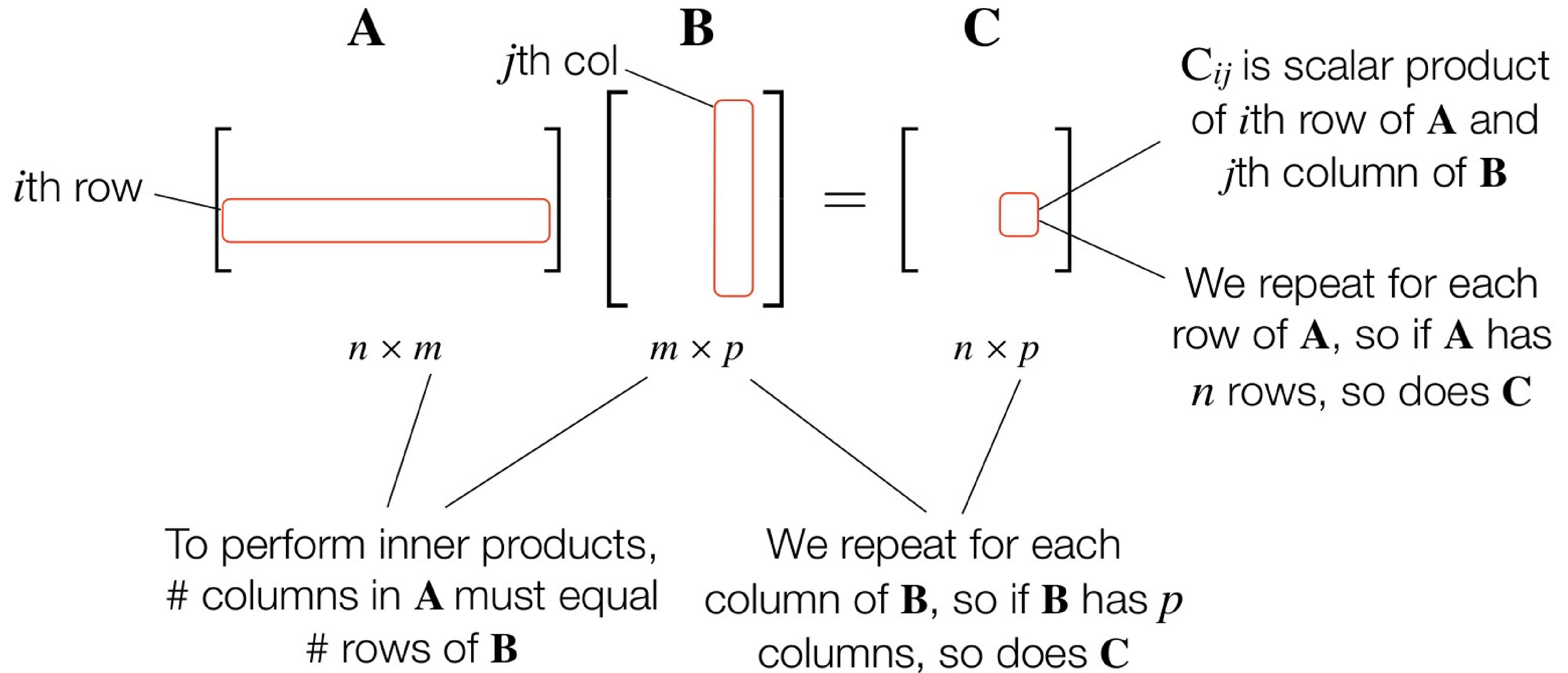
Also involves several scalar products

$$\begin{bmatrix} 9 & 3 & 5 \\ 4 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & -5 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 28 & 18 \\ 11 & 9 \end{bmatrix}$$

$$9 \times 1 + 3 \times 3 + 5 \times 2 = 28$$

$$9 \times 2 + 3 \times (-5) + 5 \times 3 = 18$$

# Matrix-Matrix Multiplication



# Matrix-Matrix Multiplication

$$\begin{matrix} \mathbf{A} & \mathbf{B} & \mathbf{C} \\ \left[ \begin{array}{c} \text{ } \\ \text{ } \\ \text{ } \\ \text{ } \\ \boxed{\text{ }} \end{array} \right] & \left[ \begin{array}{c} \text{ } \\ \text{ } \\ \boxed{\text{ }} \\ \text{ } \\ \text{ } \end{array} \right] & = \left[ \begin{array}{c} \text{ } \\ \text{ } \\ \text{ } \\ \boxed{\text{ }} \end{array} \right] \\ n \times m & m \times p & n \times p \end{matrix}$$

Associative, i.e.,  $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$

Not commutative, i.e.,  $\mathbf{AB} \neq \mathbf{BA}$

# Outer Product

Special case of Matrix-Matrix  
Multiplication involving two vectors

$$\mathbf{x} \quad \mathbf{w}^\top \quad \mathbf{C}$$
$$\begin{bmatrix} \end{bmatrix} \begin{bmatrix} \end{bmatrix} = \begin{bmatrix} \end{bmatrix}$$

$n \times 1$        $1 \times m$        $n \times m$

$C_{ij}$  is “inner product” of  $i$ th entry of  $\mathbf{x}$  and  $j$ th entry of  $\mathbf{w}$

# Identity Matrix

One is the scalar multiplication identity, i.e.,  $c \times I = c$

$\mathbf{I}_n$  is the  $n \times n$  identity matrix, i.e.,  $\mathbf{I}_n\mathbf{A} = \mathbf{A}$  and  $\mathbf{A}\mathbf{I}_m = \mathbf{A}$  for any  $n \times m$  matrix  $\mathbf{A}$

$$\begin{bmatrix} 9 & 3 & 5 \\ 4 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 9 & 3 & 5 \\ 4 & 1 & 2 \end{bmatrix}$$

Identity matrices are square, with ones on the diagonal entries

# Inverse Matrix

$I/c$  is the scalar inverse, i.e.,  $c \times I/c = I$

Multiplying a matrix by its inverse results in the identity matrix

- For an  $n \times n$  matrix  $\mathbf{A}$ ,  $\mathbf{A}^{-1}$  denotes its inverse (when it exists)
- $\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_n$

Only a square matrix (when  $n = m$ ) can have an inverse

# Euclidean Norm for Vectors

The magnitude / length of a scalar is its absolute value

Vector norms generalize this idea for vectors

The Euclidean norm for  $\mathbf{x} \in \mathbb{R}^m$  is denoted by  $\|\mathbf{x}\|_2$

- $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \dots + x_m^2}$
- Equals absolute value when  $m=1$
- Related to scalar product:  $\|\mathbf{x}\|_2^2 = \mathbf{x}^\top \mathbf{x}$

# **Big O Notation for Time and Space Complexity**

# Big O Notation

Describes how algorithms respond to changes in input size

- Both in terms of processing time and space requirements
- We refer to complexity and Big O notation synonymously

Required space proportional to units of storage

- Typically 8 bytes to store a floating point number

Required time proportional to number of ‘basic operations’

- Arithmetic operations (+, -, ×, /), logical tests (<, >, ==)

# Big O Notation

Notation:  $f(x) = O(g(x))$

- Can describe an algorithm's time or space complexity

Informal definition:  $f$  does not grow faster than  $g$

Formal definition:  $|f(x)| \leq C|g(x)| \quad \forall x > N$

Ignores constants and lower-order terms

- For large enough  $x$ , these terms won't matter
- E.g.,  $x^2 + 3x \leq Cx^2 \quad \forall x > N$

# E.g., O(1) Complexity

**Constant** time algorithms perform the same number of operations every time they're called

- E.g., performing a fixed number of arithmetic operations

Similarly, constant space algorithms require a fixed amount of storage every time they're called

- E.g., storing the results of a fixed number of arithmetic operations

# E.g., $O(n)$ Complexity

**Linear** time algorithms perform a number of operations proportional to the number of inputs

- E.g., adding two  $n$ -dimensional vectors requires  $O(n)$  arithmetic operations

Similarly, linear space algorithms require storage proportional to the size of the inputs

- E.g., adding two  $n$ -dimensional vectors results in a new  $n$ -dimensional vector which requires  $O(n)$  storage

# E.g., $O(n^2)$ Complexity

**Quadratic** time algorithms perform a number of operations proportional to the square of the number of inputs

- E.g., outer product of two  $n$ -dimensional vectors requires  $O(n^2)$  multiplication operations (one per each entry of the resulting matrix)

Similarly, quadratic space algorithms require storage proportional to the square of the size of the inputs

- E.g., outer product of two  $n$ -dimensional vectors requires  $O(n^2)$  storage (one per each entry of the resulting matrix)

# Time and Space Complexity Can Differ

Inner product of two  $n$ -dimensional vectors

- $O(n)$  time complexity to multiply  $n$  pairs of numbers
- $O(1)$  space complexity to store result (which is a scalar)

Matrix inversion of an  $n \times n$  matrix

- $O(n^3)$  time complexity to perform inversion
- $O(n^2)$  space complexity to store result

# E.g., Matrix-Vector Multiply

Goal: multiply an  $n \times m$  matrix with an  $m \times 1$  vector

Computing result takes  $O(nm)$  time

- There are  $n$  entries in the resulting vector
- Each entry computed via dot product between two  $m$ -dimensional vectors (a row of input matrix and input vector)

Storing result takes  $O(n)$  space

- The result is an  $n$ -dimensional vector

# E.g., Matrix-Matrix Multiply

Goal: multiply an  $n \times m$  matrix with an  $m \times p$  matrix

Computing result takes  $O(npm)$  time

- There are  $np$  entries in the resulting matrix
- Each entry computed via dot product between two  $m$ -dimensional vectors

Storing result takes  $O(np)$  space

- The result is an  $n \times p$  matrix