CS 6375
Introduction to Machine Learning

Rishabh Iyer

University of Texas at Dallas

Slides adapted from Nick Rouzzi and Sriraam Natarajan

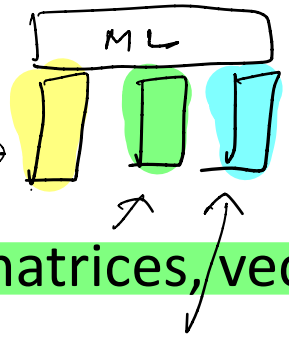
Grading

- 4 problem sets (50%)
  - eLearning
  - Mix of theory + programming
  - 1 assignment every 2/3 weeks
- Mid term (25%)
- Final Project (25%)

(subject to change)

Prerequisites

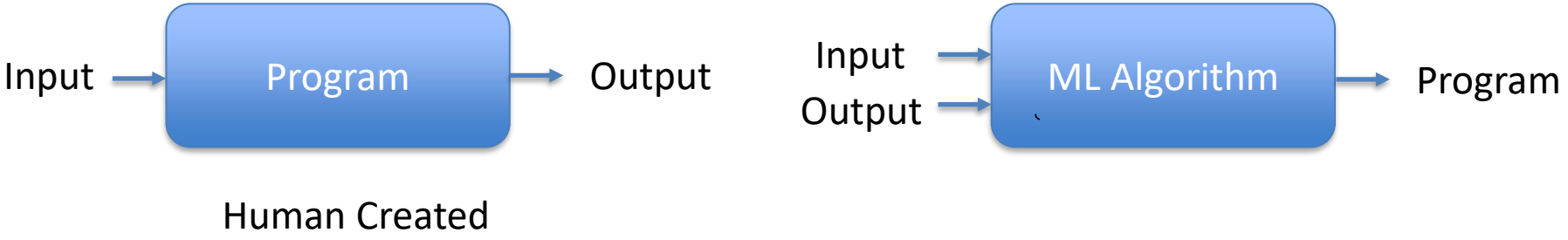
- CS3345, Data Structures and Algorithms
- CS3341, Probability and Statistics in Computer Science
- "Mathematical sophistication"
  - Basic probability
  - Linear algebra: eigenvalues/vectors, matrices, vectors, etc.
  - Multivariate calculus: derivatives, gradients, etc.



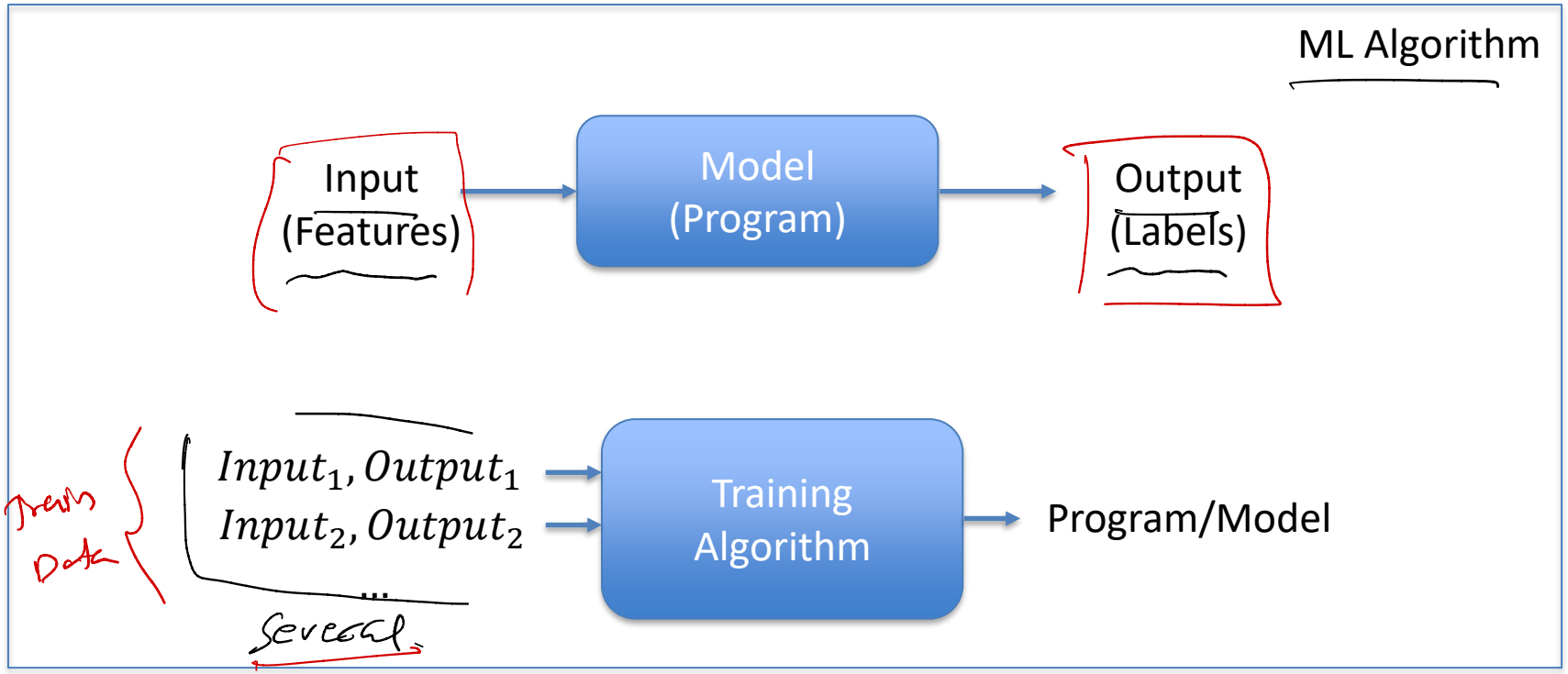
- I'll review some concepts as we come to them, but you should brush up on areas that you aren't as comfortable
- Take prerequisite "quiz" on eLearning

What is Machine Learning?

- Programming:
  - A human writes a program (set of rules/conditions/algorithm) to do a specific task
  - For a given input, the program generates an output
- Machine Learning Paradigm:
  - Generate training data consisting of ("input", "output") pairs
  - The "ML Model" automatically generates a program (set of rules/conditions) to generate an output for a new (unseen) input



Basic Machine Learning Paradigm



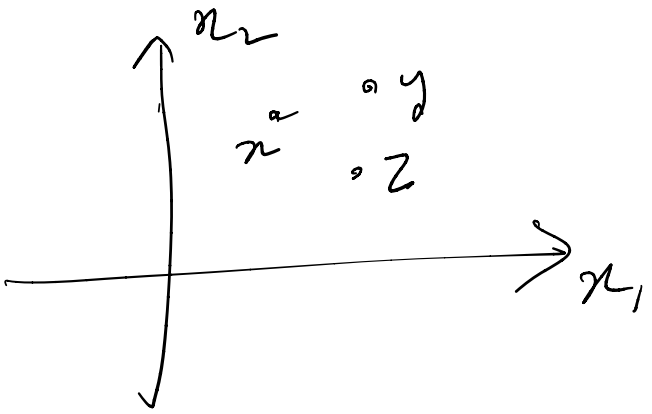
Basics: Vectors

$x \in \mathbb{R}^d$ (d - dimensional vector)

$x = \begin{bmatrix} \\ \\ \end{bmatrix}$ d-dim. $= \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$

e.g. $x \in \mathbb{R}^2$

$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ x_1, x_2



Operations

(1) vector addⁿ

$z = x + y$

if $z_i = x_i + y_i$, $\forall i = 1:d$

e.g.

$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $y = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$

$z = \begin{bmatrix} 4 \\ 6 \end{bmatrix}$

(2) vector sub

$z = x - y$

(3) Dot product (inner product)

$\langle x, y \rangle$

$z = x \cdot y$ [scalar]

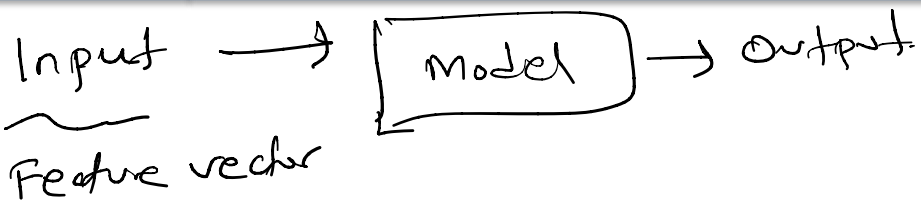
$= \sum_{i=1}^{d} x_i y_i$

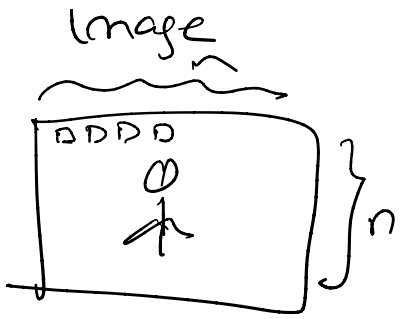
e.g.:

$z = 1 \cdot 3 + 2 \cdot 4$

$= 11$

Basics: Feature Vectors

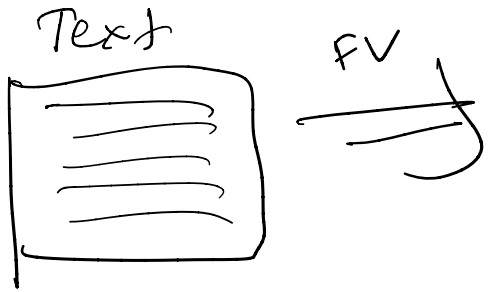


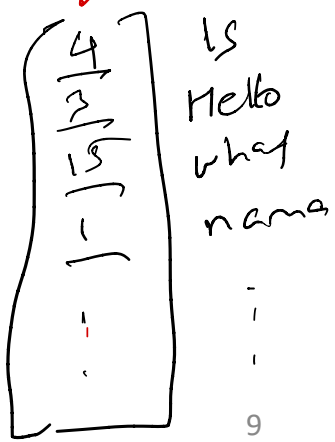


Model
(Is there a person or not)

person / no-person

Feature Vec: $n \times n \times 3$
RGB





(Bag of words Rep)

(1) Email: Spam / Not Spam

(2) webpage: Sports / News / ..

Vector Operations

1. Addition
2. Multiplication (Dot product)
3. subtraction.
4. Scalar multiplication

$$y = c \cdot x$$

$$= c \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} cx_1 \\ \vdots \\ cx_d \end{bmatrix}$$

Matrices and Matrix Vector Product

If $A \in \mathbb{R}^{m \times n}$ and $x \in \mathbb{R}^n$, we can define $y = Ax$ where $y \in \mathbb{R}^m$ is a m dimensional vector.

Matrix vector product is defined as below:

$$\mathbf{Ax} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}$$

Matrix Vector Product Example

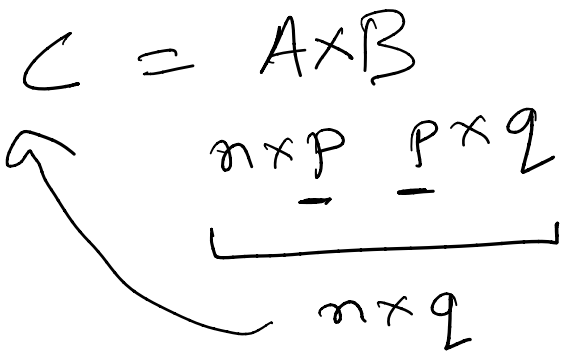
For example, if

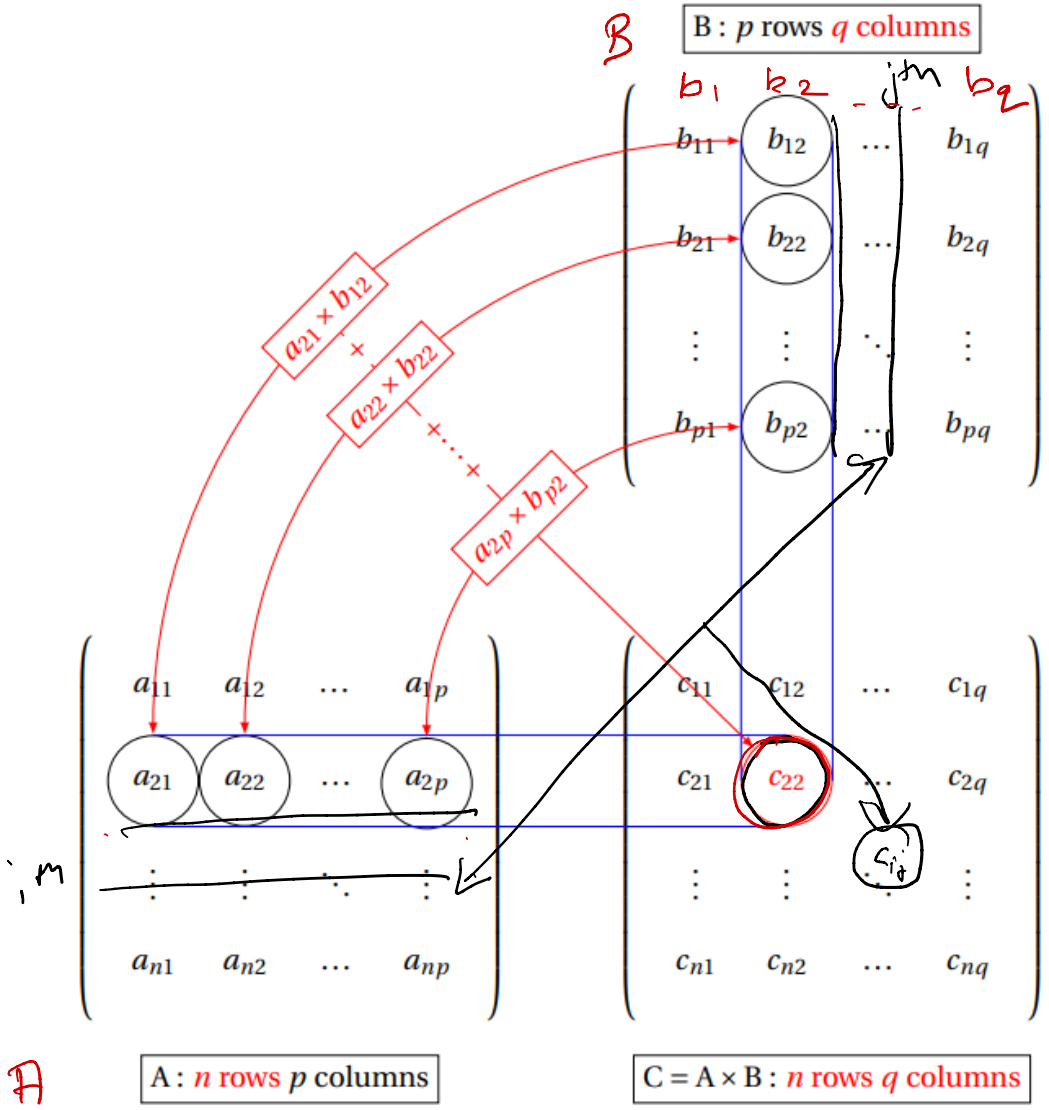
$$A = \begin{bmatrix} 1 & -1 & 2 \\ 0 & -3 & 1 \end{bmatrix}$$

and $\mathbf{x} = (2, 1, 0)$, then

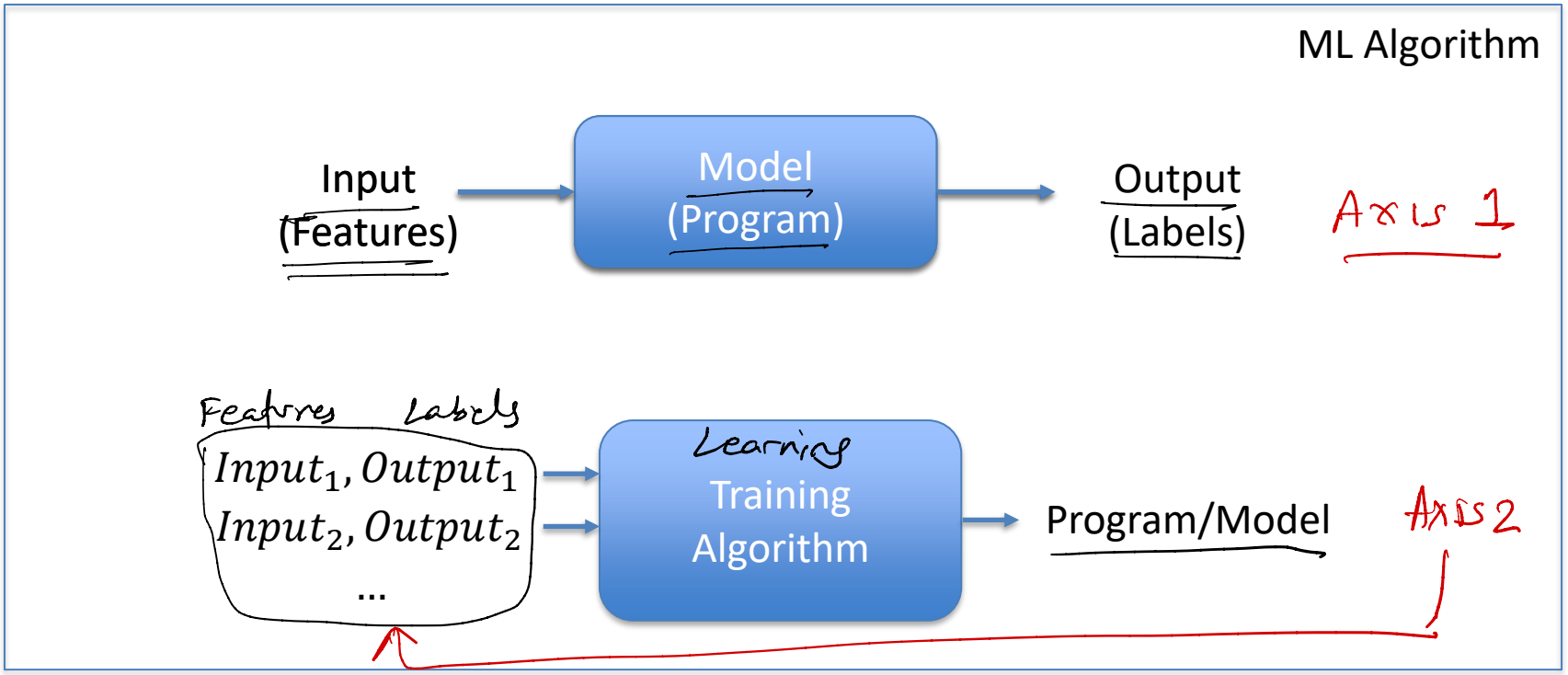
$$\begin{aligned} A\mathbf{x} &= \begin{bmatrix} 1 & -1 & 2 \\ 0 & -3 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 2 \cdot 1 - 1 \cdot 1 + 0 \cdot 2 \\ 2 \cdot 0 - 1 \cdot 3 + 0 \cdot 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ -3 \end{bmatrix}. \end{aligned}$$

Matrix Matrix Product





Types of Machine Learning



Axis One: What is the Output?

Axis Two: Amount of Labeled Data for training and how is it available to us

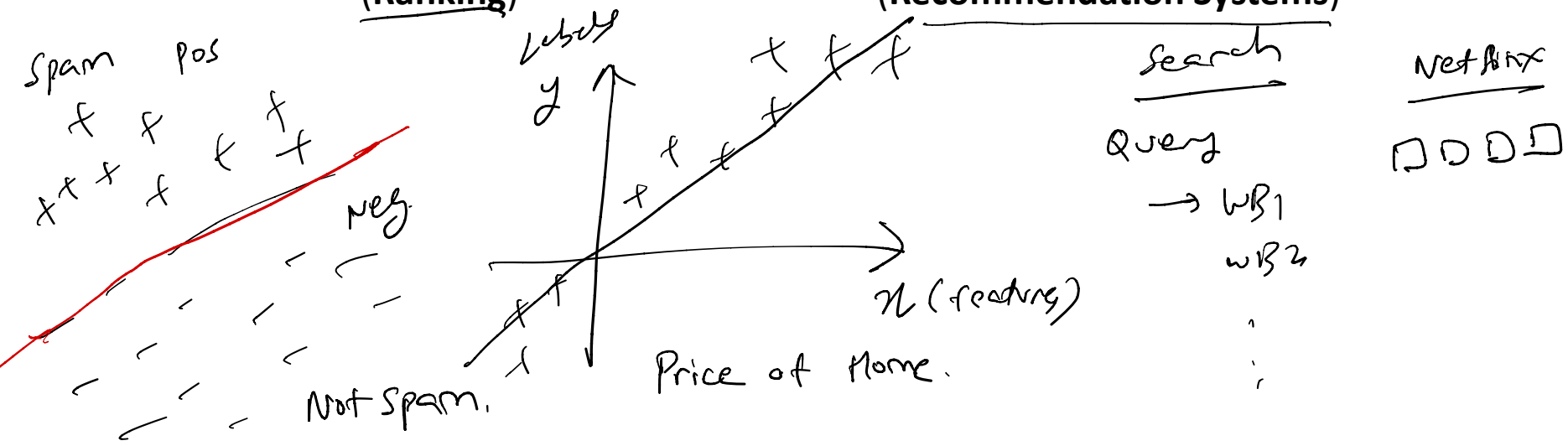
Types of Machine Learning

Axis One

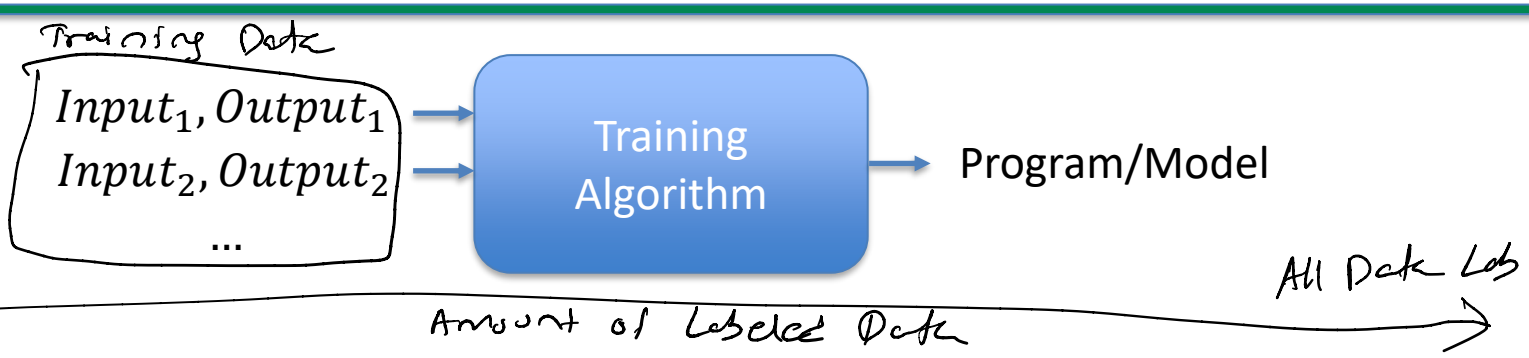
Input (Features) → Model (Program) → Output (Labels)

Labels

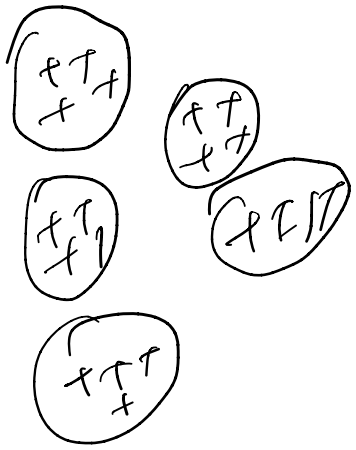
- Discrete Classes (**Classification**)
  - K- Classes
- Continuous Value (**Regression**)
- Actions (**Reinforcement Learning**)
- Ranking among items (**Ranking**)
- Recommendations (**Recommendation Systems**)

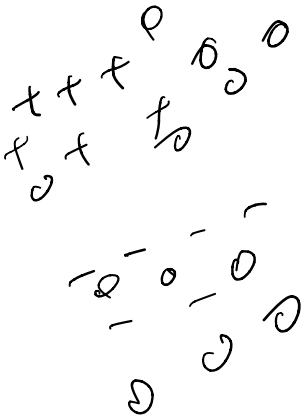


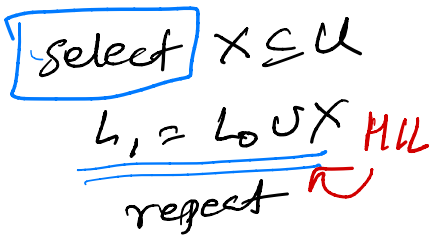
Types of Machine Learning

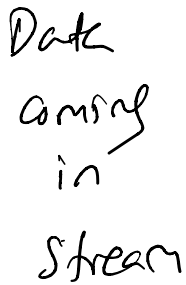


Unsupervised	Semi-Supervised	Active Learning	Online	Supervised
(No Labels)	**(Labeled + Unlabeled)**	**(Get Labels Iteratively)**	**(Stream)**	**(Labeled)**
No training Data		Human In Loop		









+ +ive Class } Labels
− −ive Class }
0 unlabeled

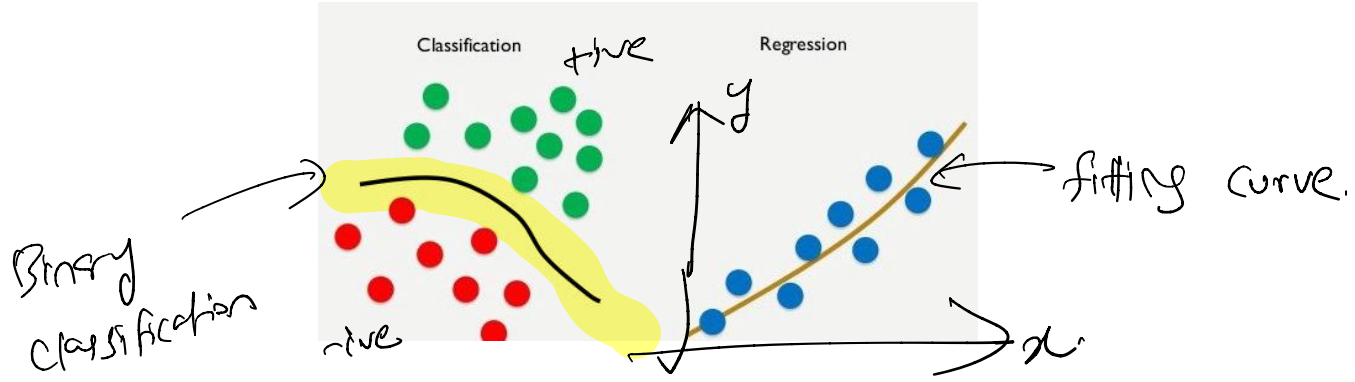
Supervised Learning

M training data points
All labeled

- **Input:** $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(M)}, y^{(M)}\right)$
  - $x^{(m)}$ is the m^{th} data item and $y^{(m)}$ is the m^{th} **label**

Features Labels

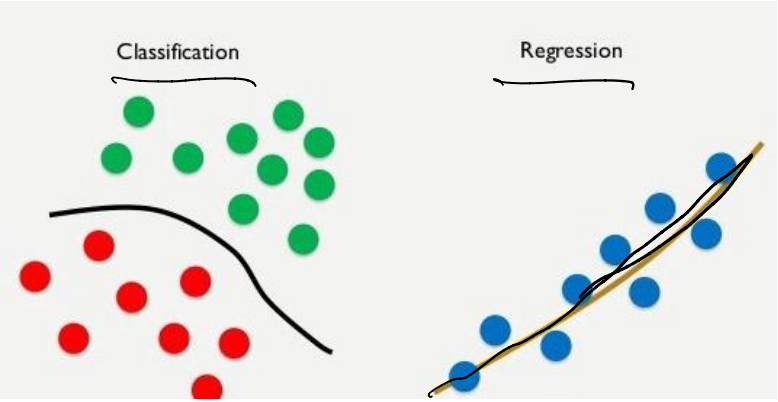
- **Goal:** find a function f such that $f\left(x^{(m)}\right)$ is a "good approximation" to $y^{(m)}$
  - Can use it to predict y values for previously unseen x values

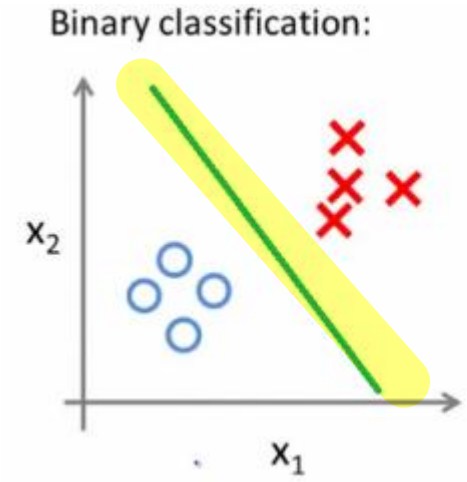


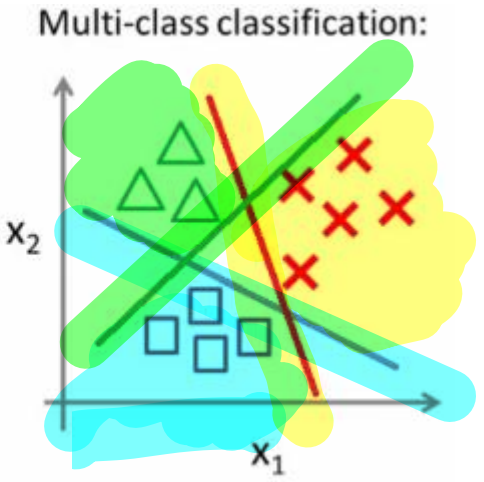
Supervised Learning

Classification vs Regression

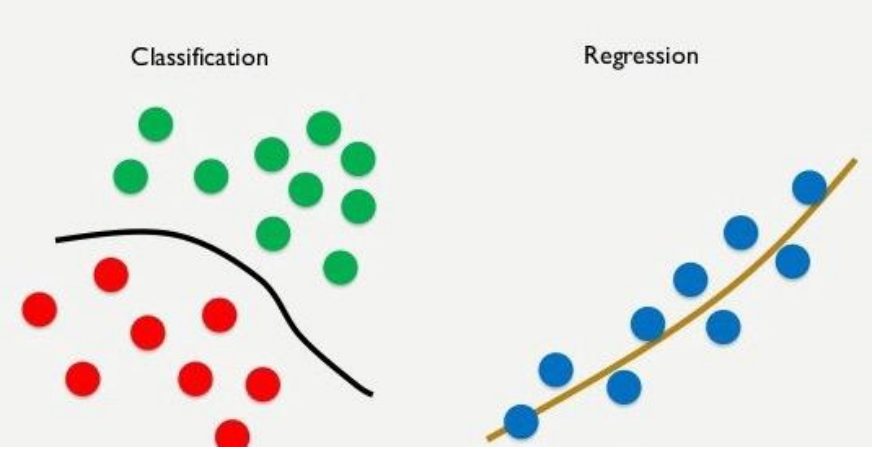
- Input: pairs of points $(x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^d$
- Regression case: $y^{(m)} \in \mathbb{R}$ (cont)
- Classification case: $y^{(m)} \in [0, k-1]$ [k-class classification]
- If k = 2, we get Binary classification (Discrete)







Examples of Supervised Learning



Classification

- Spam email detection
- Handwritten digit recognition
- Medical Diagnosis
- Fraud Detection
- Face Recognition

Regression

- Housing Price Prediction
- Stock Market Prediction
- Weather Prediction
- Market Analysis and Business Trends

Classification – Medical Diagnosis

Do Not Have Diabetes

blood glucose = 30
body mass index = 120 kg/m²
diastolic bp = 79 mm Hg
age = 32 years

blood glucose = 22
body mass index = 160 kg/m²
diastolic bp = 80 mm Hg
age = 63 years

blood glucose = 22
body mass index = 160 kg/m²
bp = 80 mm Hg
age = 18 years

blood glucose = 40
body mass index = 160 kg/m²
mm Hg
63 years

blood glucose =
body mass in g/m²
diastolic bp = 73 mm Hg
age = 27 years

blood glucose = 46
body mass index = 15 g/m²
diastolic bp = 110 m
age = 55 years

blood glucose = 45
dex = 180 kg/m²
bp = 95 mm Hg
age = 49 years

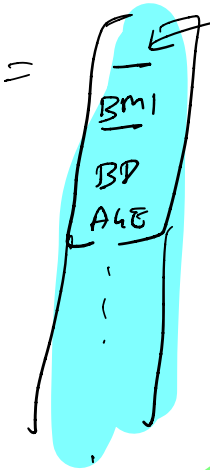
blood glucose = 21
dex = 140 kg/m²
c bp = 99 mm Hg
age = 37 years

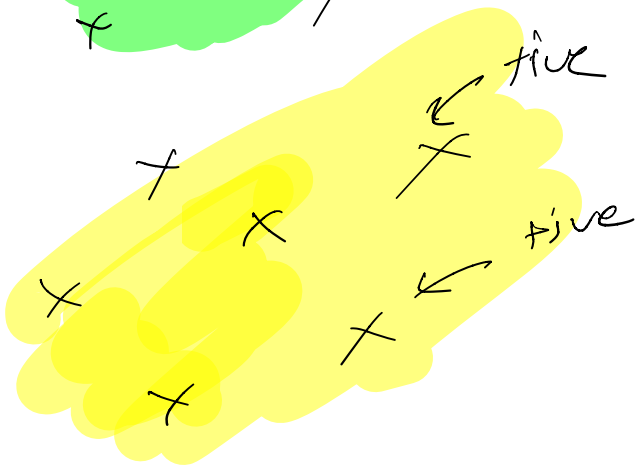
Have Diabetes

1. Feature Eng
2. Collected Lab data
3. Train a ML Model.

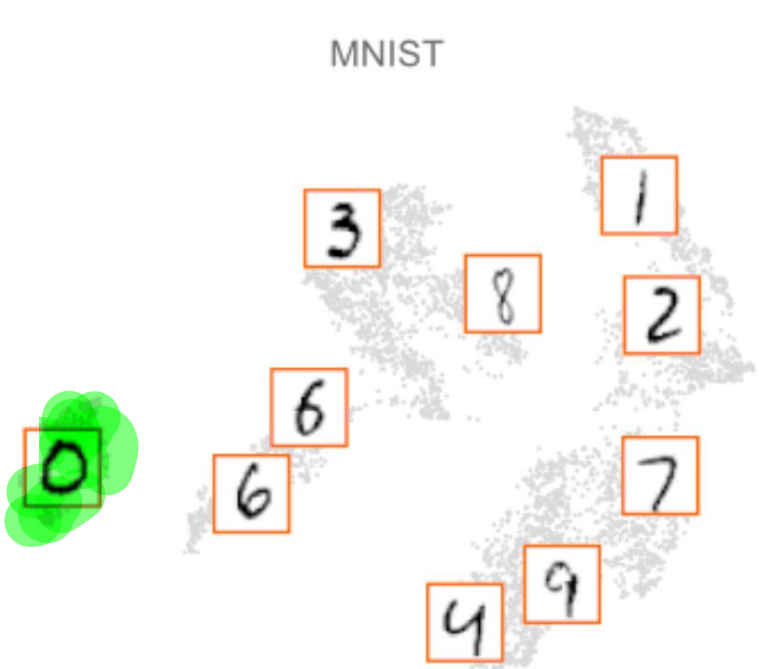
0 → Not Diab
1 → Diab

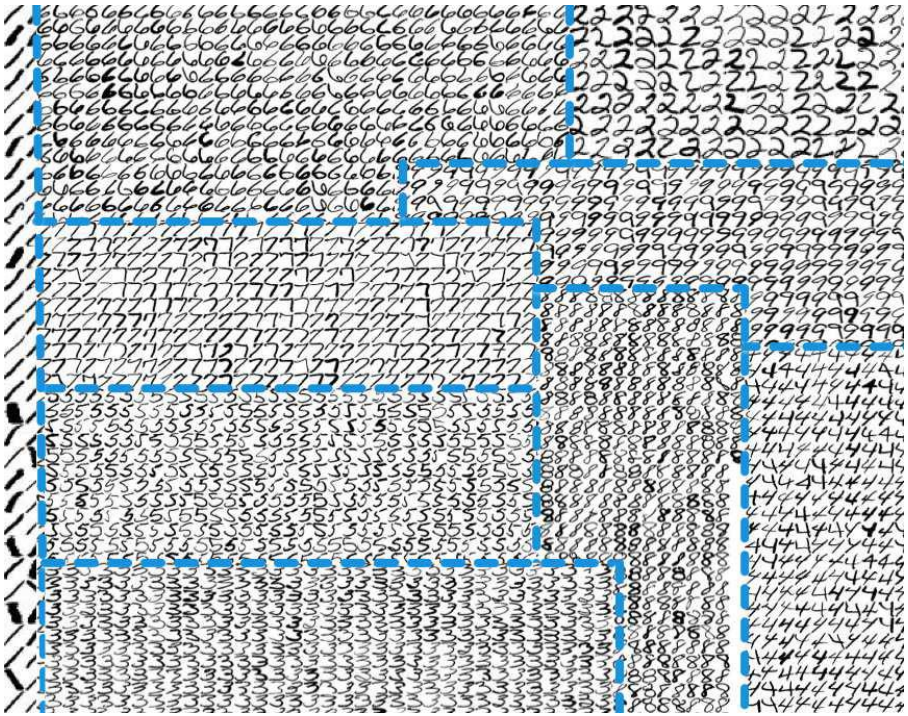
Person = [BG, BMI, BP, Age, ...] ∈ $\mathbb{R}^d$

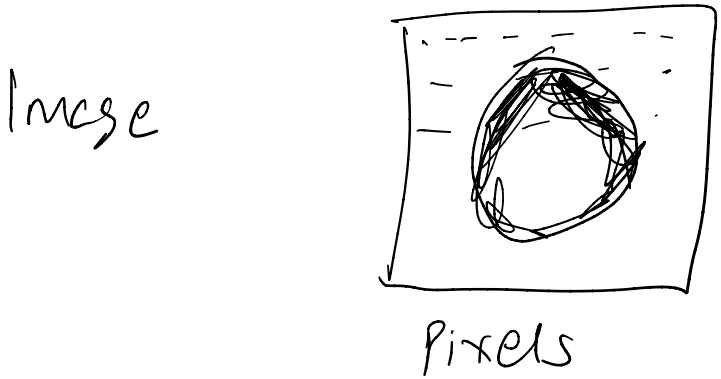




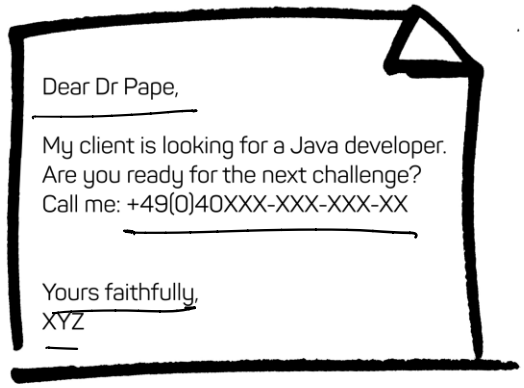
Classification – Digit Recognition



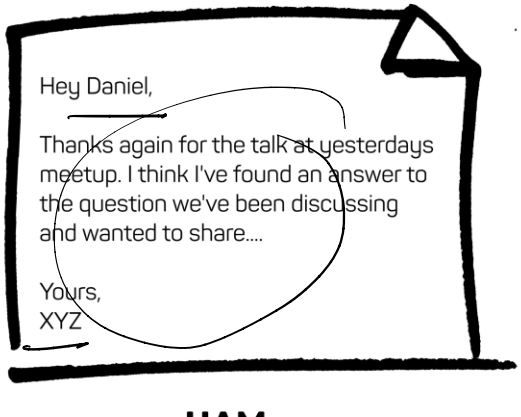


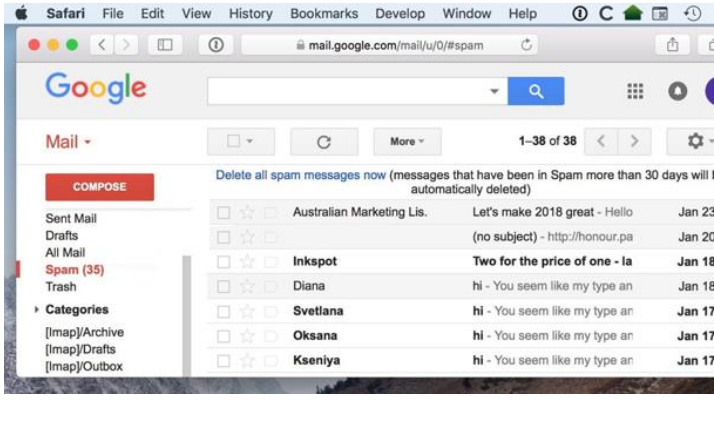


Classification – Spam



vs.



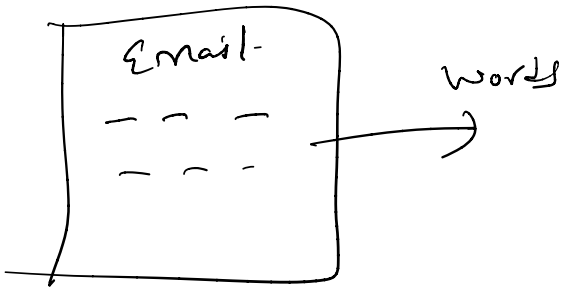


SPAM

HAM

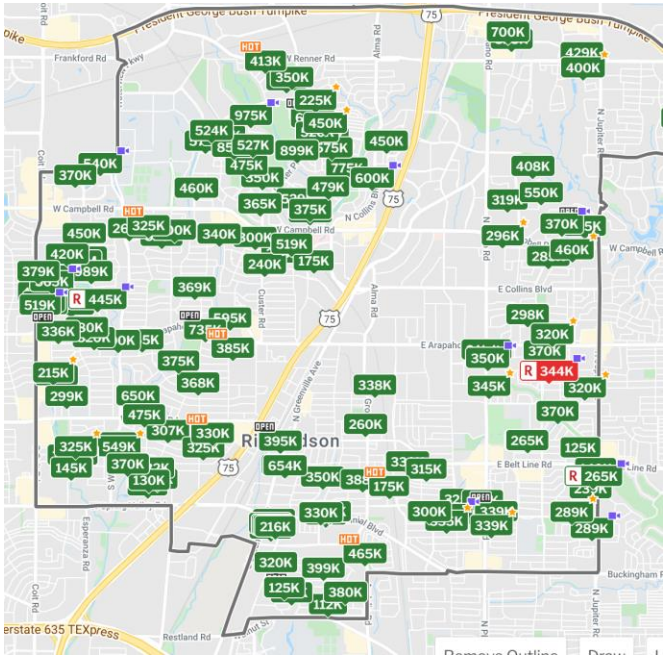
Not Spam

Hi!!!, FREE!!!



Label: Spam (+1)
Not Spam (-1)

Regression – Housing Price Prediction

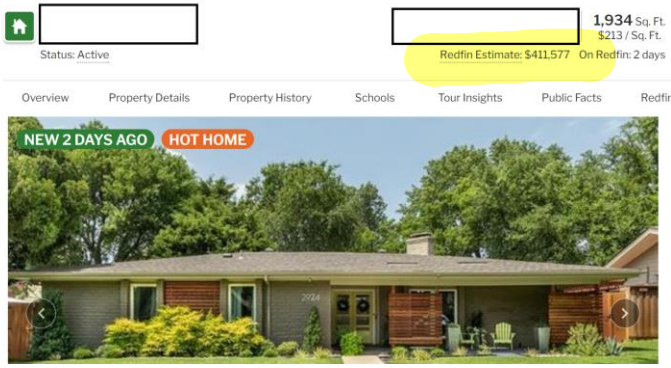


Home price = y

Features = x

$y = f(x)$

~ 411000 $



y

Home Facts

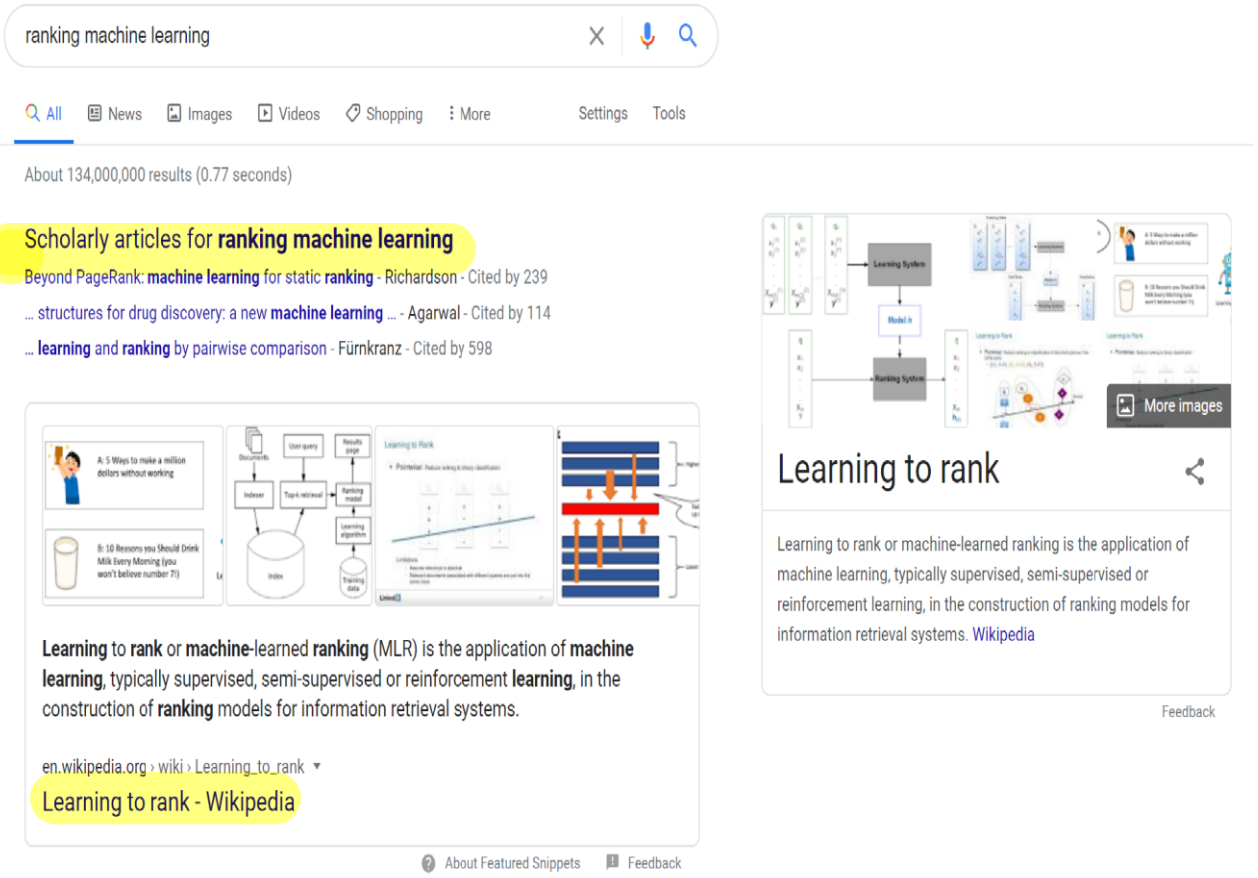
Status	Active	Time on Redfin	2 days
Property Type	Residential, Single Family	HOA Dues	$4/month
Year Built	1969	Style	Single Detached, Mid-Century Modern, Ranch, Traditional
Community	Canyon Creek Country Club 9	Lot Size	10,019 Sq. Ft.
MLS#	14375892		

y SF TH

Size : 1974 Sq. (VVIMP)

No Bed / No Baths

Ranking – Search Engines



URL
TITLE
Text
:
:

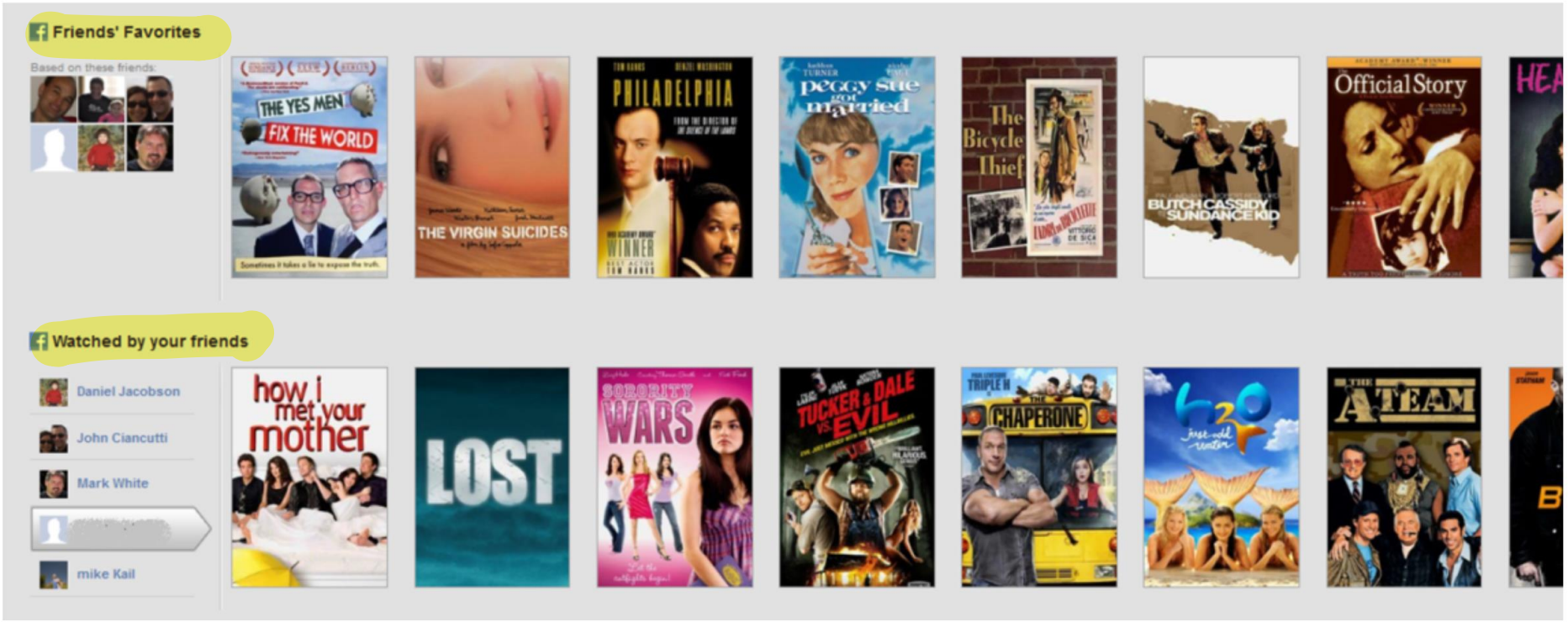
} features

Label = { P_1, P_2, ., ., P_{10}

Rank 1

Rank 2

Recommendation – Movie Recommendations



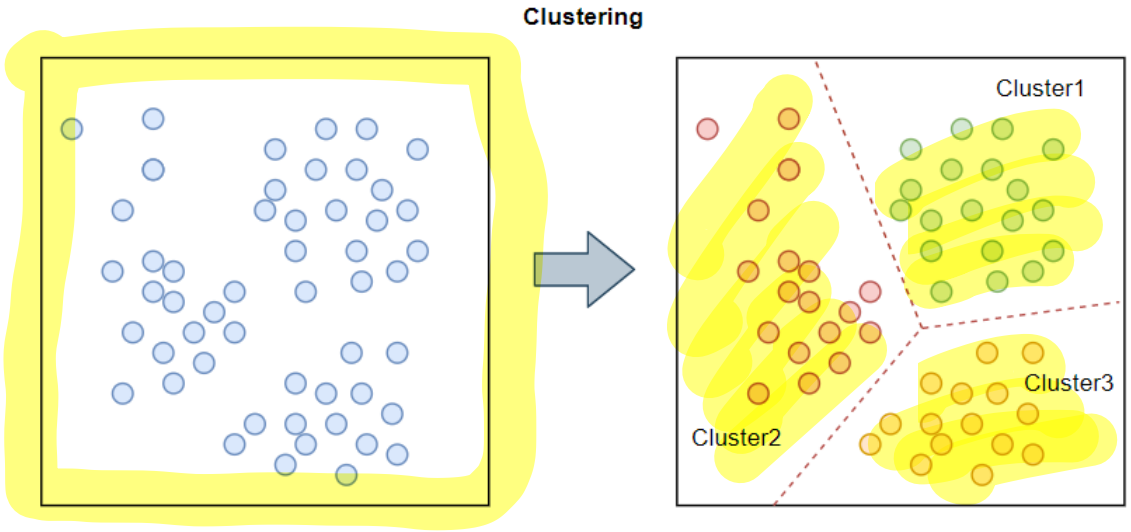
(1) My Past (History)

(2) Friends (similar Movies)

} Recommendations

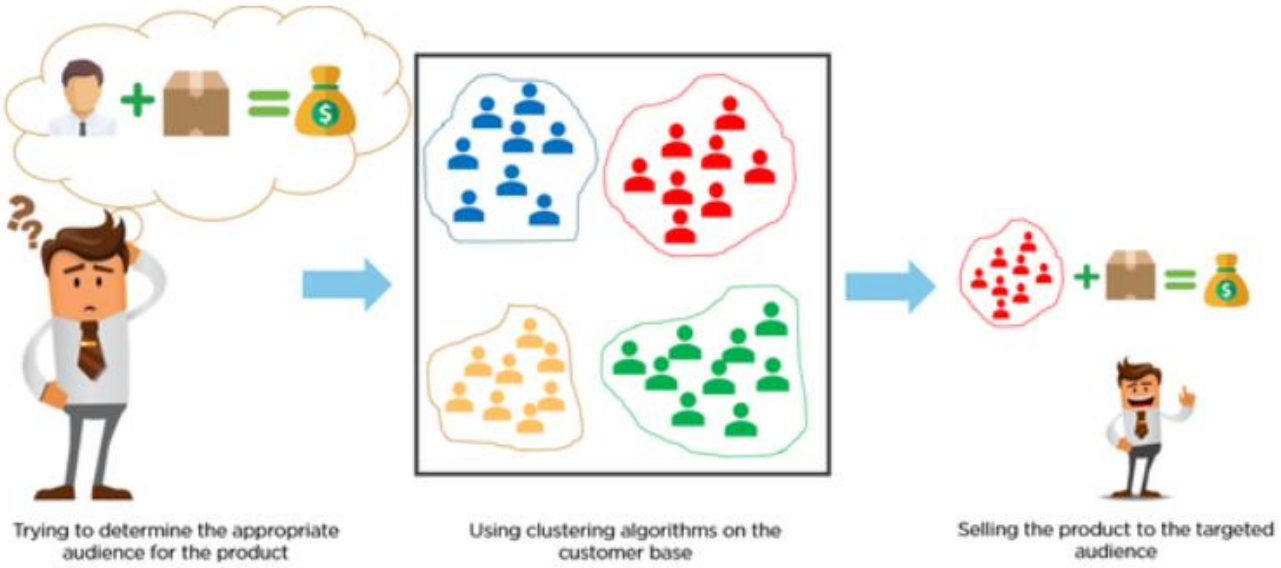
Unsupervised Learning

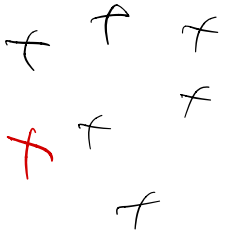
- **Input:** $x^{(1)}, \ldots, x^{(M)}$
  - $x^{(m)}$ is the m^{th} data item
  - **No Label!**
- **Goal:** find a clustering/grouping of data points into k clusters so that each cluster consists of similar points

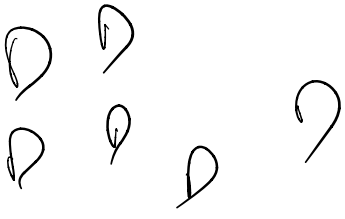


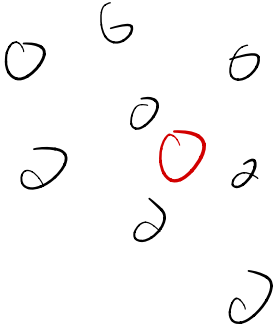
Applications of Unsupervised Learning

- Item Categorization
- Clustering Customers
- Similar Item Recommendation
- Outlier Detection

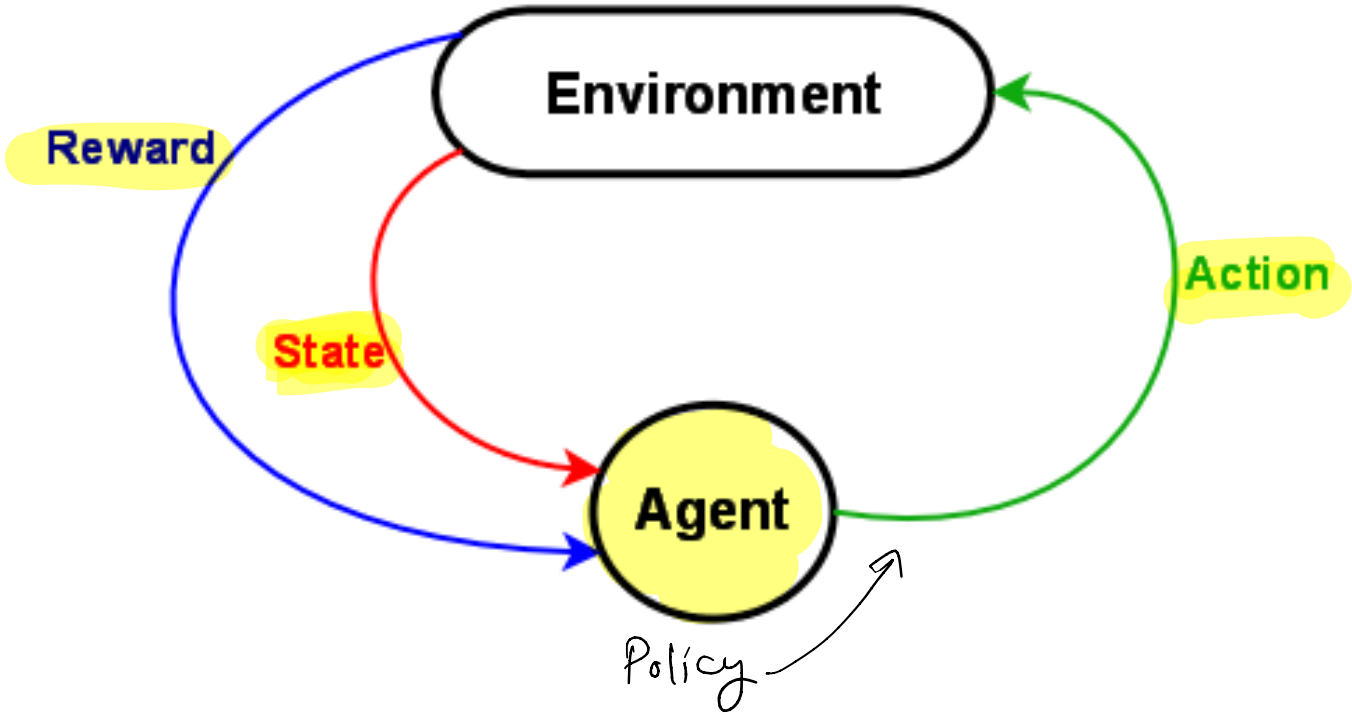




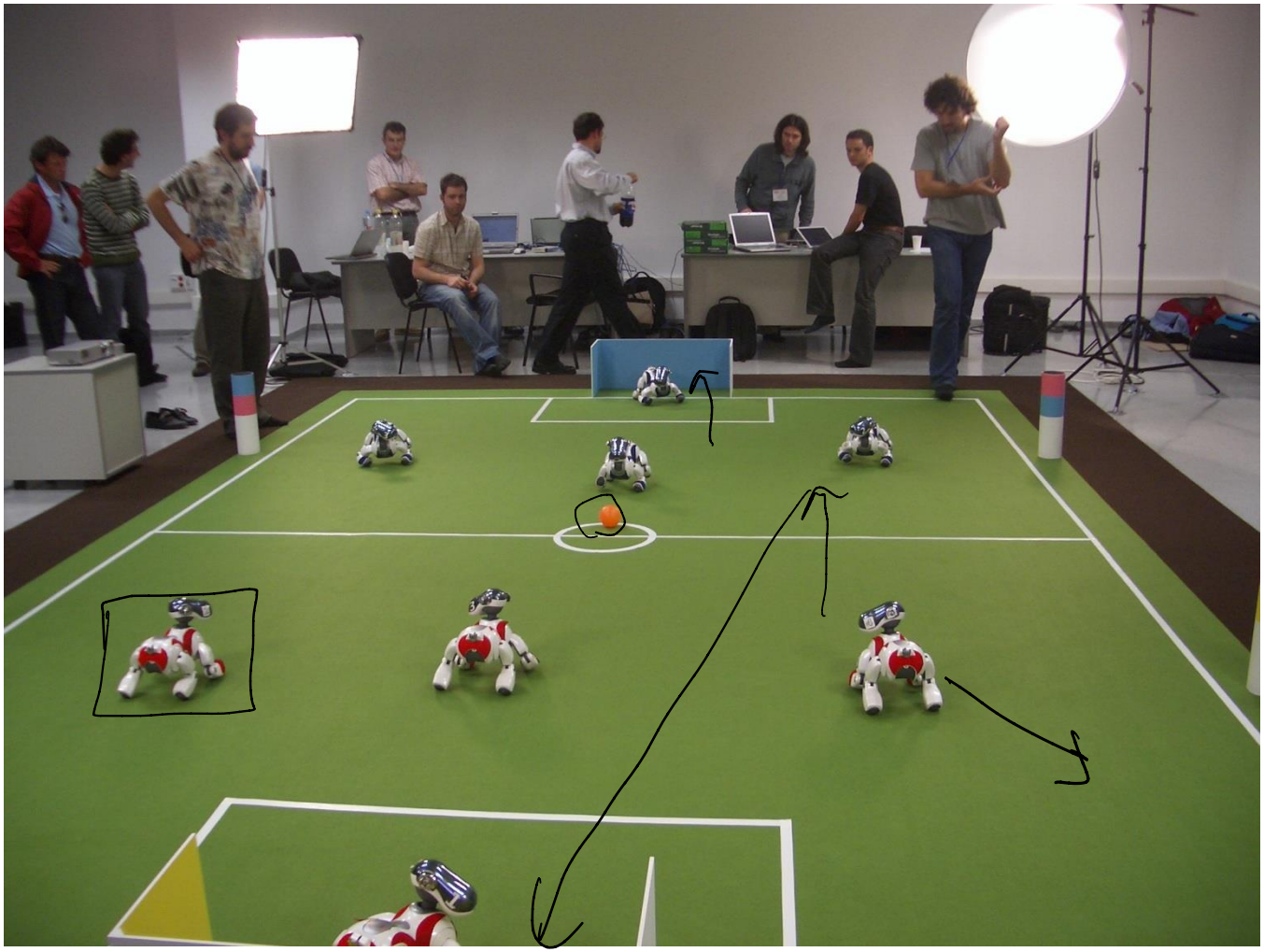




Reinforcement Learning



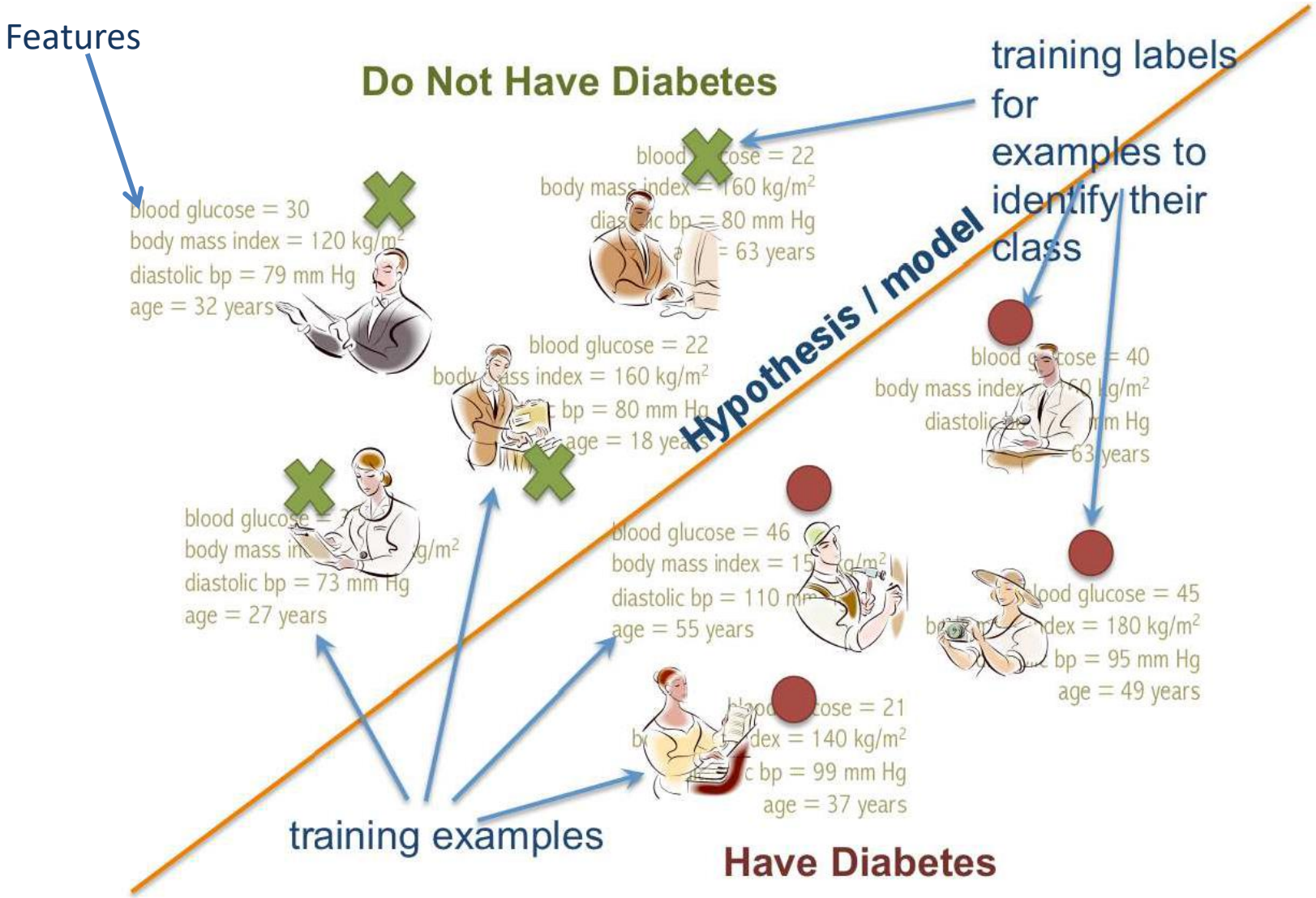
Reinforcement Learning – Robocup Soccer



Other Types of Learning

- Semi-supervised
  - Training Labeled + Unlabeled Data Jointly
- Active learning
  - Semi-supervised learning where the algorithm can ask for the correct outputs for specifically chosen data points
- Online Learning
  - Data and Labels coming in a stream
- Reinforcement learning
  - The learner interacts with the world via allowable actions which change the state of the world and result in rewards
  - The learner attempts to maximize rewards through trial and error

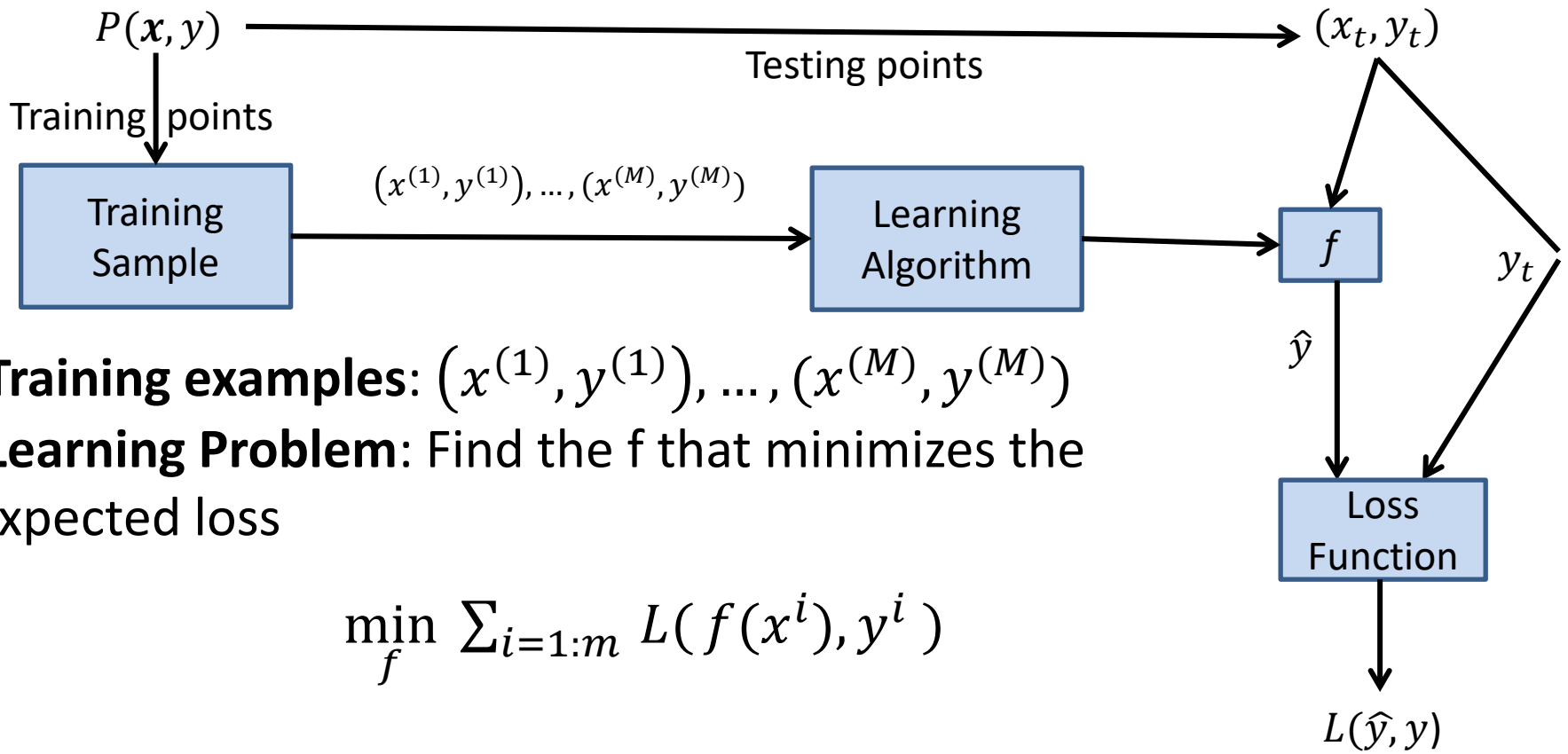
Terminology



Terminology

- **Training Example:** <**x**,y>
  - **x:** feature vector (describes the attributes of something)
  - **y:** label (continous values for regression problems: [1,2,…,k] for classification problems)
- **Training set** A set of training examples drawn randomly from P(**x**,y)
  - **Key Assumption:** Independent and identically distributed. i.e., all the examples are drawn from the same distribution but are drawn independent of one another
- **Target function** True mapping from **x** to y
- **Hypothesis:** A function ~~f~~ considered by the learning algorithm to be similar to the target function
- **Test set:** A set of examples drawn from P(**x**,y) to evaluate the "goodness of ~~f~~"
- **Hypothesis Space:** The space of all hypotheses that can in principle be considered and returned by the learning algorithm

Supervised Learning Workflow



- **Training examples**: $(x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})$
- **Learning Problem**: Find the f that minimizes the expected loss

$$\min_f \sum_{i=1:m} L(f(x^i), y^i)$$

- **Testing:** Given a new point (x_t, y_t) drawn from P, the classifier is given x and predicts $\hat{y}_t = f(x_t)$
- **Evaluation:** Measure the error $Err(\hat{y}_t, y_t)$ – often same as L

Train phase

① Loss function L: [e.g. $L(y, \hat{y}) = [y - \hat{y}]^2$]

② Hyp function $f(x)$ [e.g. $f(x) = w^T x + b$]

③ Train Data: $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

$$\min_f \sum_{i=1}^{m} L(f(x^{(i)}), y^{(i)})$$

Alg is solving $\longrightarrow$ f_{train}

Test:

Held out test set (x_t, y_t)

$$\sum_t Err(y_t, f_{train}(x_t))$$

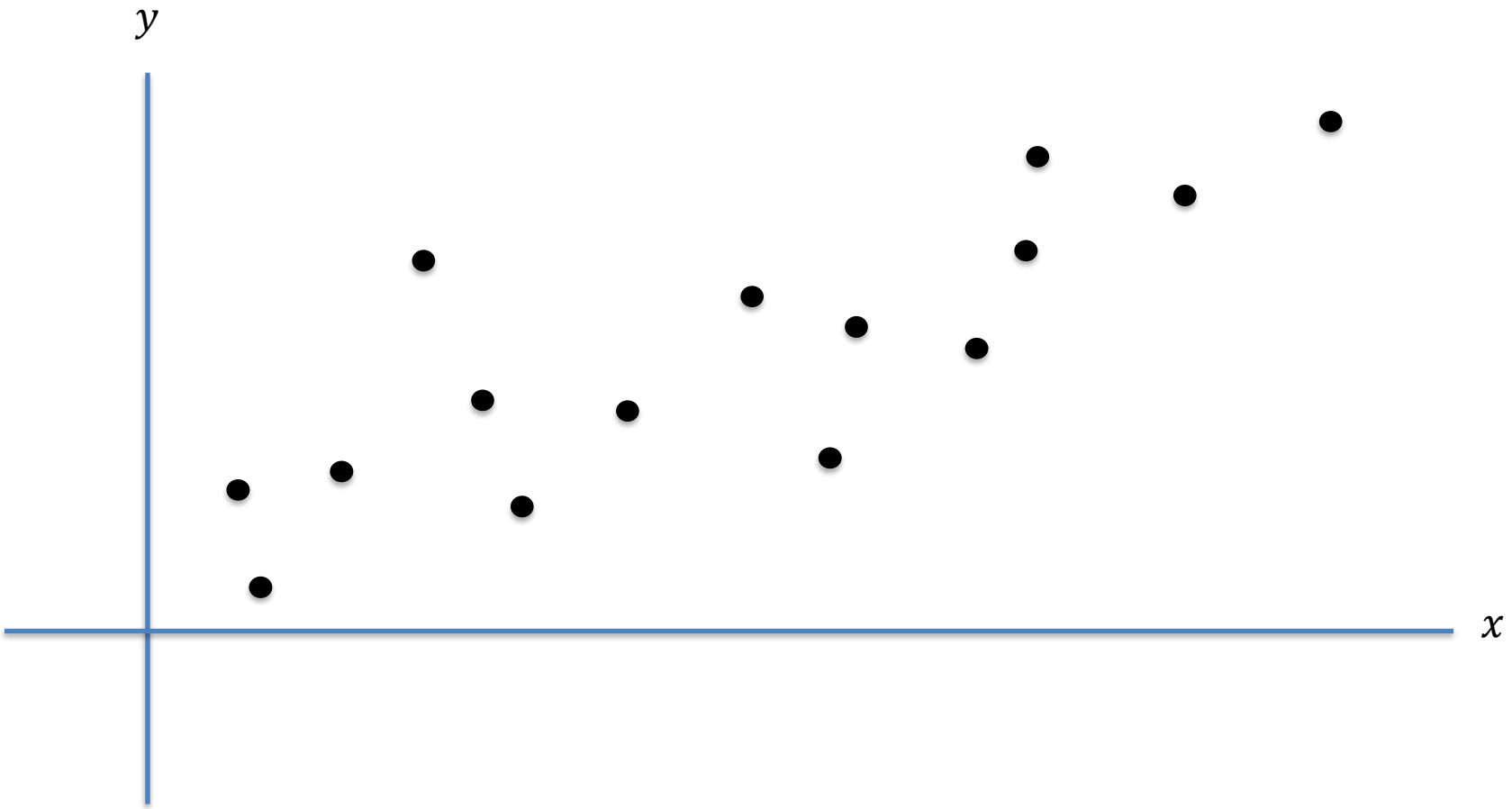
Linear Regression

- Simple linear regression
  - Input: pairs of points $(x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^d$ and $y^{(m)} \in \mathbb{R}$ (Regression)
  - Hypothesis space: set of linear functions $f(x) = a^T x + b$ with $a \in \mathbb{R}^d, b \in \mathbb{R}$

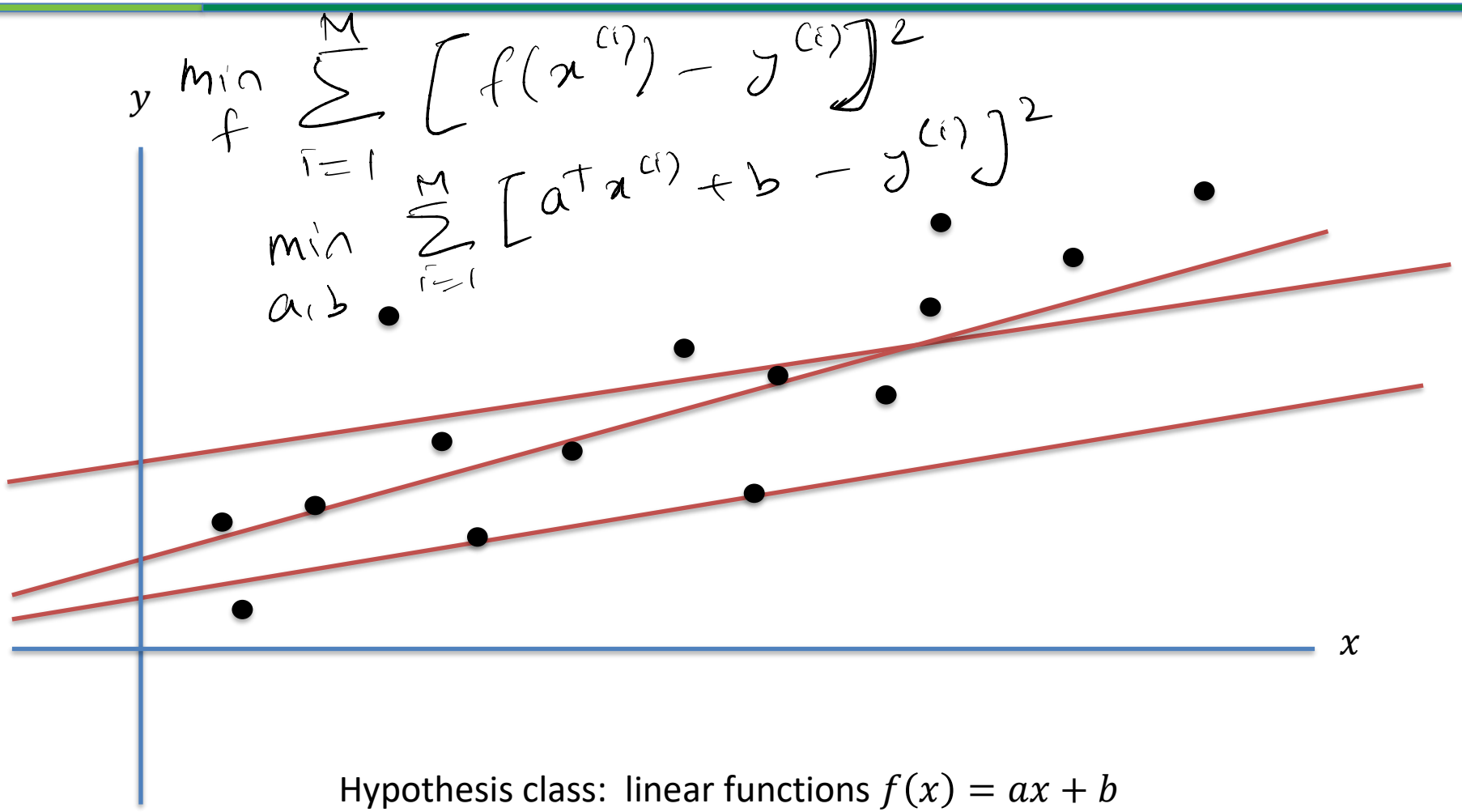
    $a \cdot x$
  - Error metric and Loss Function: squared difference between the predicted value and the actual value

$$L(y, \hat{y}) = [y - \hat{y}]^2$$

Regression



Regression



Hypothesis class: linear functions $f(x) = ax + b$

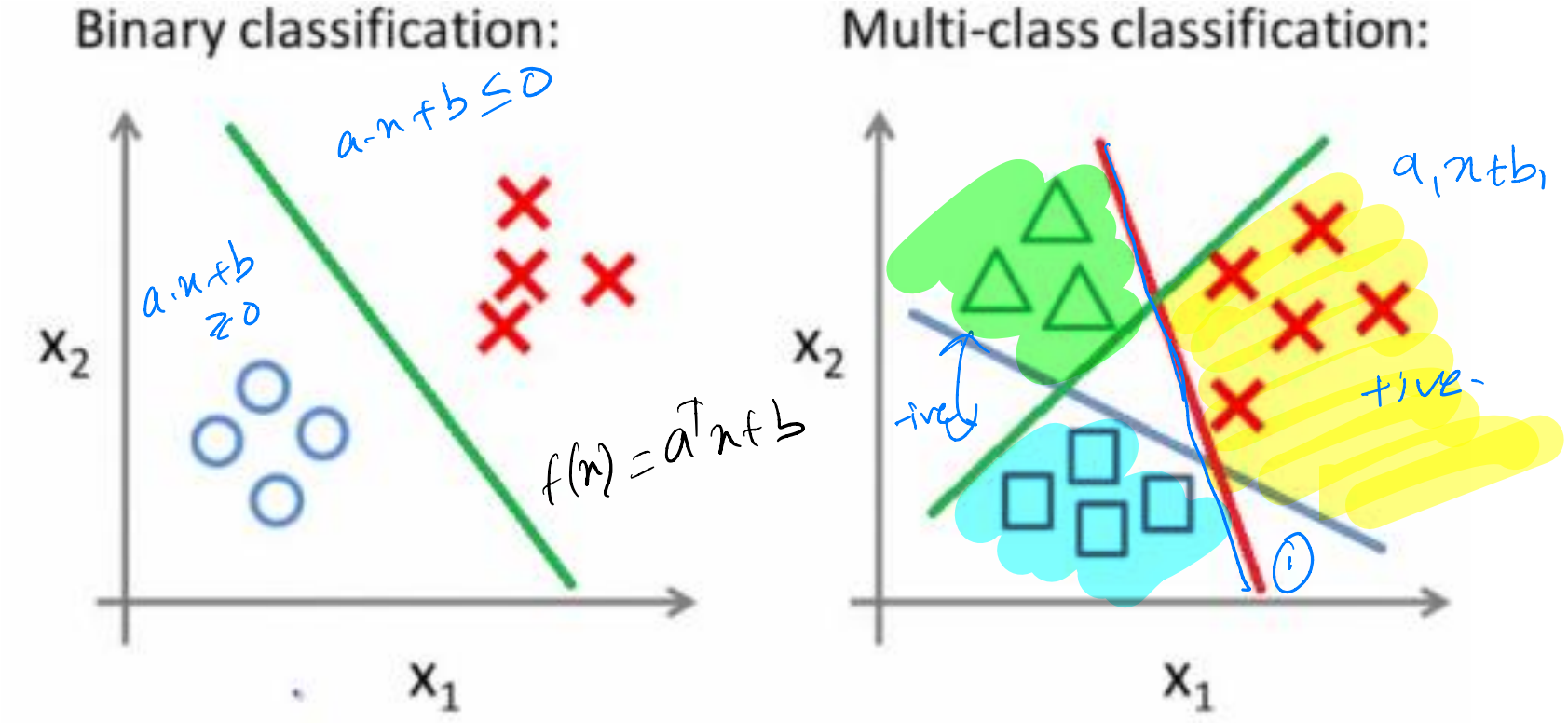
How do we compute the error of a specific hypothesis?

Linear Classification

- Simple linear classification
  - Input: pairs of points $\left(x^{(1)}, y^{(1)}\right), \ldots, (x^{(M)}, y^{(M)})$ with $x^{(m)} \in \mathbb{R}^d$ and $y^{(m)} \in [0, k-1]$ (Classification)
  - Hypothesis space: set of linear functions $f(x) = sign(a^T x + b)$ with $a \in \mathbb{R}^d, b \in \mathbb{R}$
  - Error metric: Accuracy (or more complex like AUC, …)
  - Loss Function: Log Loss, Hinge Loss, Perceptron Loss…

Linear Classification

$y \in [-1, +1]$



Binary Classification

- Regression operates over a continuous set of outcomes
- Suppose that we want to learn a function $f: X \rightarrow \{0,1\}$
- As an example:

	x_1	x_2	x_3	y
1	0	0	1	0
2	0	1	0	1
3	1	1	0	1
4	1	1	1	0

How many functions with three binary inputs and one binary output are there?

Binary Classification

4 data points

	x_1	x_2	x_3	y
	0	0	0	?
1	0	0	1	0
2	0	1	0	1
	0	1	1	?
	1	0	0	?
	1	0	1	?
3	1	1	0	1
4	1	1	1	0

2^8 possible functions

2^4 are consistent with the observations

How do we choose the best one?

What if the observations are noisy?

Challenges in ML

- How to choose the right hypothesis space?
  - Number of factors influence this decision: difficulty of learning over the chosen space, how expressive the space is, …
- How to evaluate the quality of our learned hypothesis?
  - Prefer "simpler" hypotheses (to prevent overfitting)
  - Want the outcome of learning to generalize to unseen data
- Computational Tractability
- Can we trust the results? Explainability!

Challenges in ML

- How do we find the best hypothesis?
  - This can be an NP-hard problem!
  - Need fast, scalable algorithms if they are to be applicable to real-world scenarios