CS 4375
Introduction to Machine Learning

Rishabh Iyer

University of Texas at Dallas

Slides adapted from Nick Rouzzi and Sriraam Natarajan

Course Info.

- Instructor: Rishabh Iyer
  - Office: ECSS 3.405
  - Office hours:
    - Tuesdays, 3 PM – 3:45 PM
    - By Appointment (Extra Office Hours): Thursdays, 3 PM – 3:45 PM
- TA: Will be Announced
- Course website: https://github.com/rishabhk108/MLClass/tree/master/Fall2025

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

Grading

- 3-4 problem sets (25%)
  - Mix of theory and programming (in Python)
  - Available and turned in on eLearning
  - Approximately one assignment every 2-3 weeks
- 2 Midterm Exam (40%)
- Final Project (25%)
- Class Participation (10%)

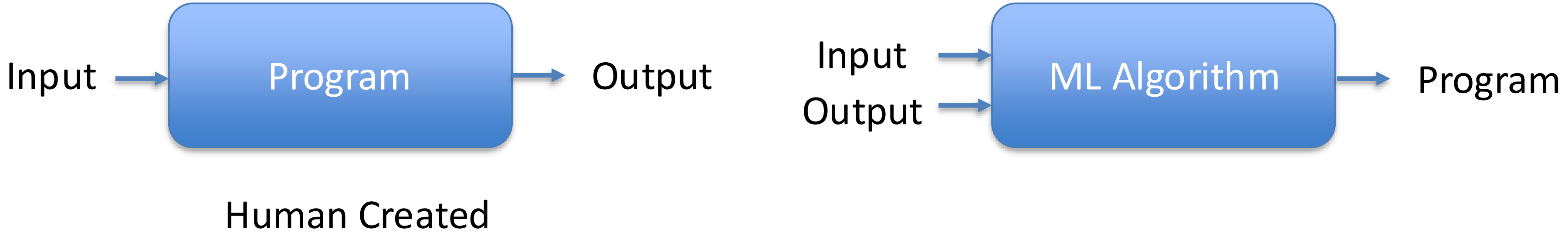
-subject to change-

Course Topics

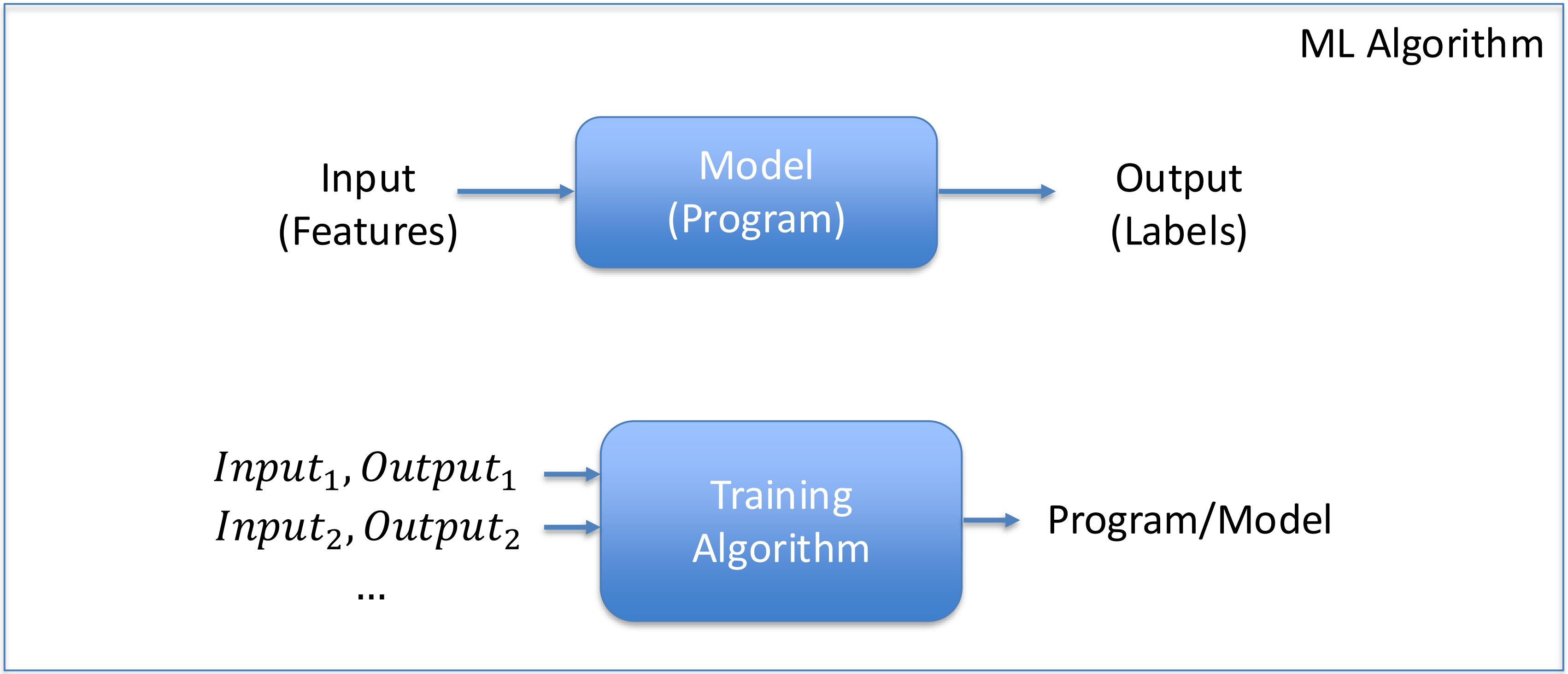
- **Supervised Learning**
  - SVMs & kernel methods
  - Decision trees, Random Forests, Gradient Boosted Trees
  - Nearest Neighbor: KNN Classifiers
  - Logistic Regression
  - Neural networks
  - Probabilistic models: Bayesian networks, Naïve Bayes
- **Unsupervised Learning**
  - Clustering: k-means & spectral clustering
  - Dimensionality reduction
- **Parameter estimation**
  - Bayesian methods, MAP estimation, maximum likelihood estimation, expectation maximization, …
- **Evaluation**
  - AOC, cross-validation, precision/recall
- **Ensemble & Statistical Methods**
  - Boosting, bagging, bootstrapping
  - Sampling
- **Other Forms of Learning**
  - Reinforcement Learning, Semi-supervised Learning, Active Learning, ….

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



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:

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

Matrix Vector Product Example

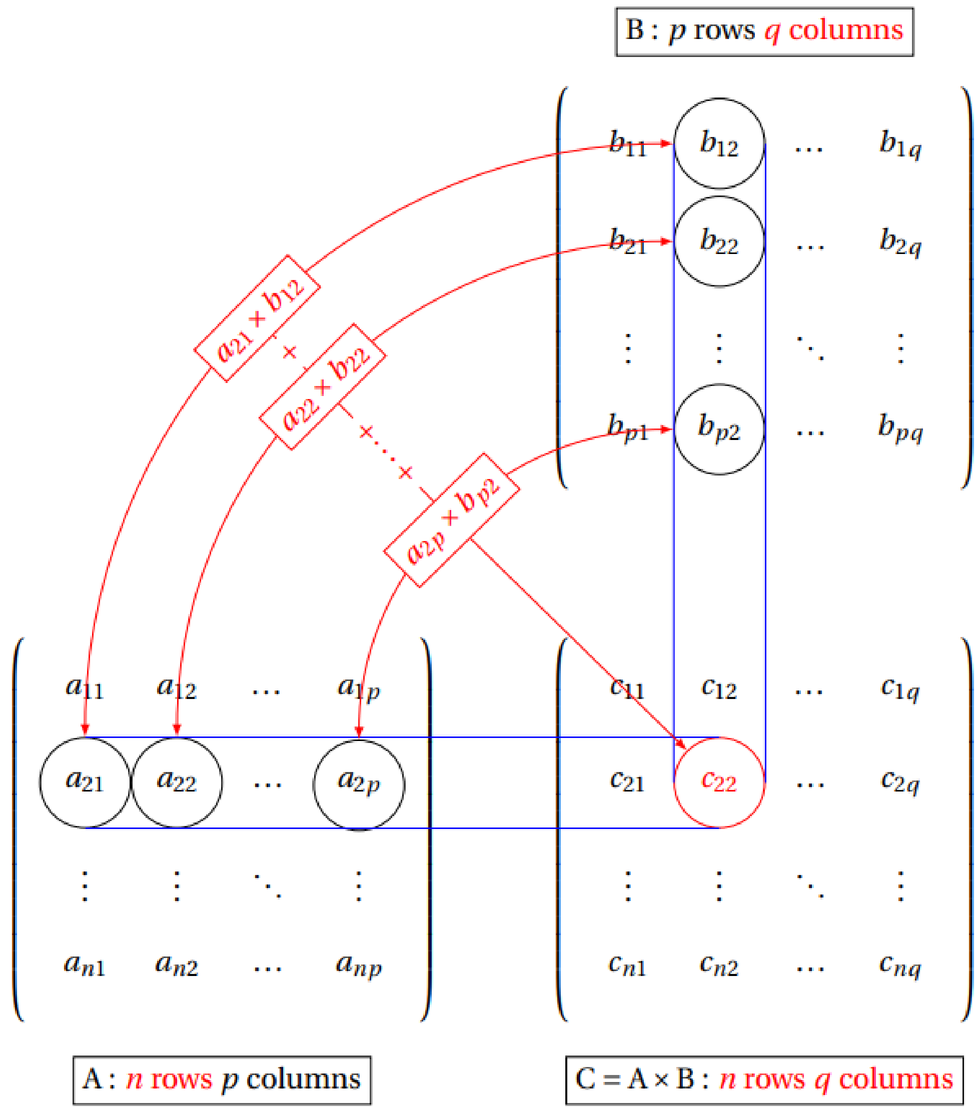
For example, if

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

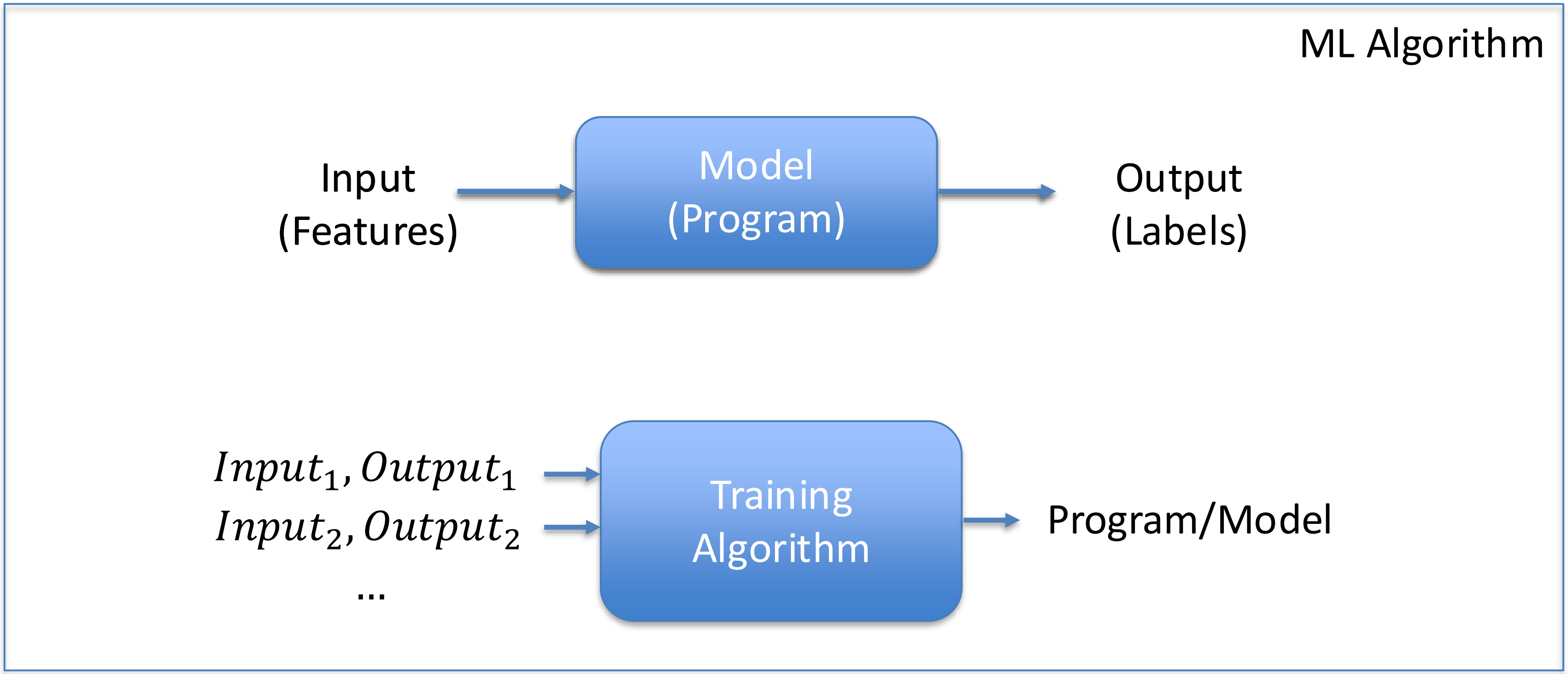
and $\mathbf{x} = (2, 1, 0)$, then

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

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)

Discrete Classes
(**Classification**)

Continuous Value
(**Regression**)

Actions
(**Reinforcement Learning**)

Ranking among items
(**Ranking**)

Recommendations
(**Recommendation Systems**)

Types of Machine Learning

Axis Two

$Input_1, Output_1$ →
$Input_2, Output_2$ →
...

Training Algorithm → Program/Model

Unsupervised
(**No Labels**)

Semi-Supervised
(**Labeled + Unlabeled**)

Active Learning
(**Get Labels Iteratively**)

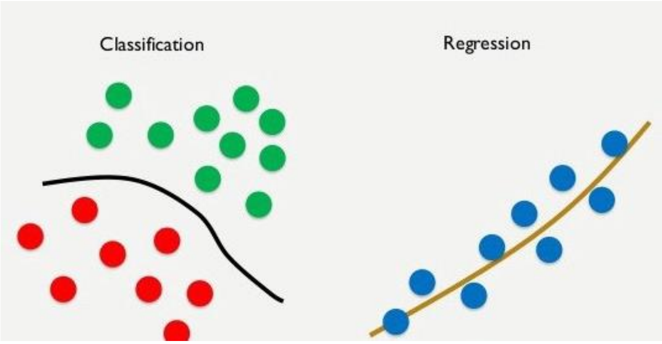
Online
(**Stream**)

Supervised
(**Labeled**)

Supervised Learning

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

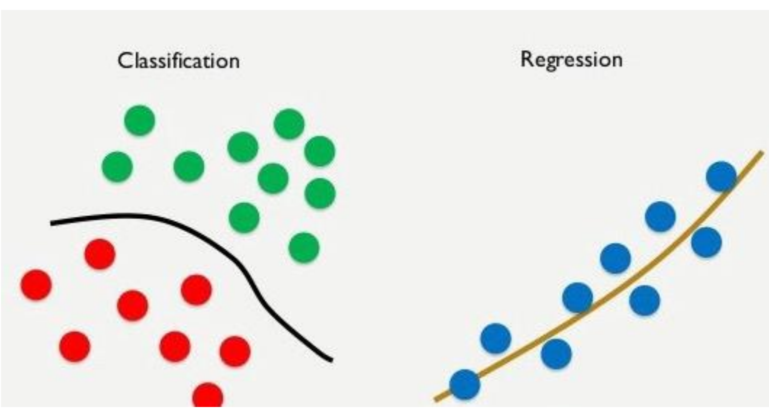
- **Goal:** find a function f such that $f\left(x^{(m)}\right)$ is a "good approximation" to $y^{(m)}$
  - Can use it to predict y values for previously unseen x values

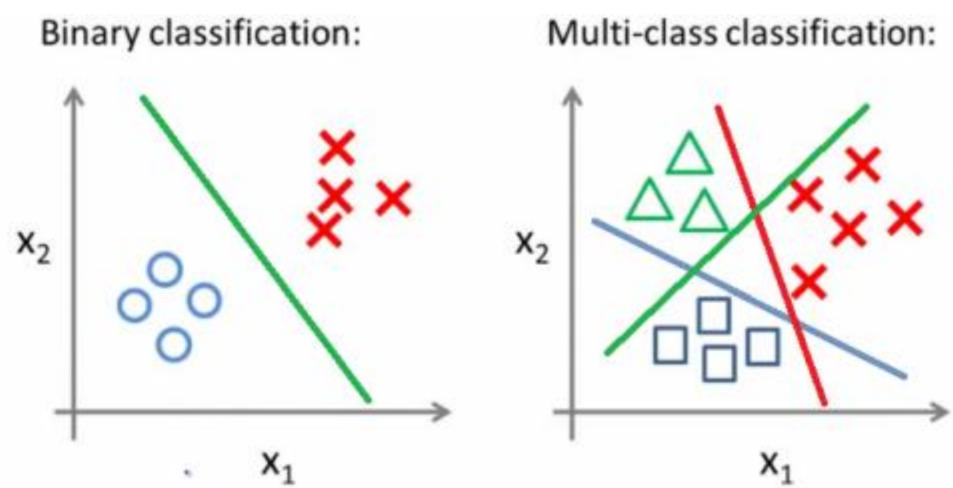


Supervised Learning

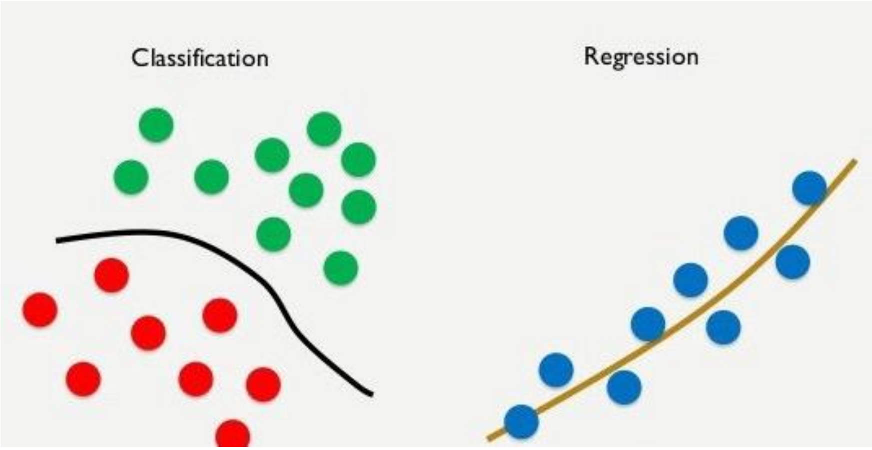
Classification vs Regression

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





Examples of Supervised Learning



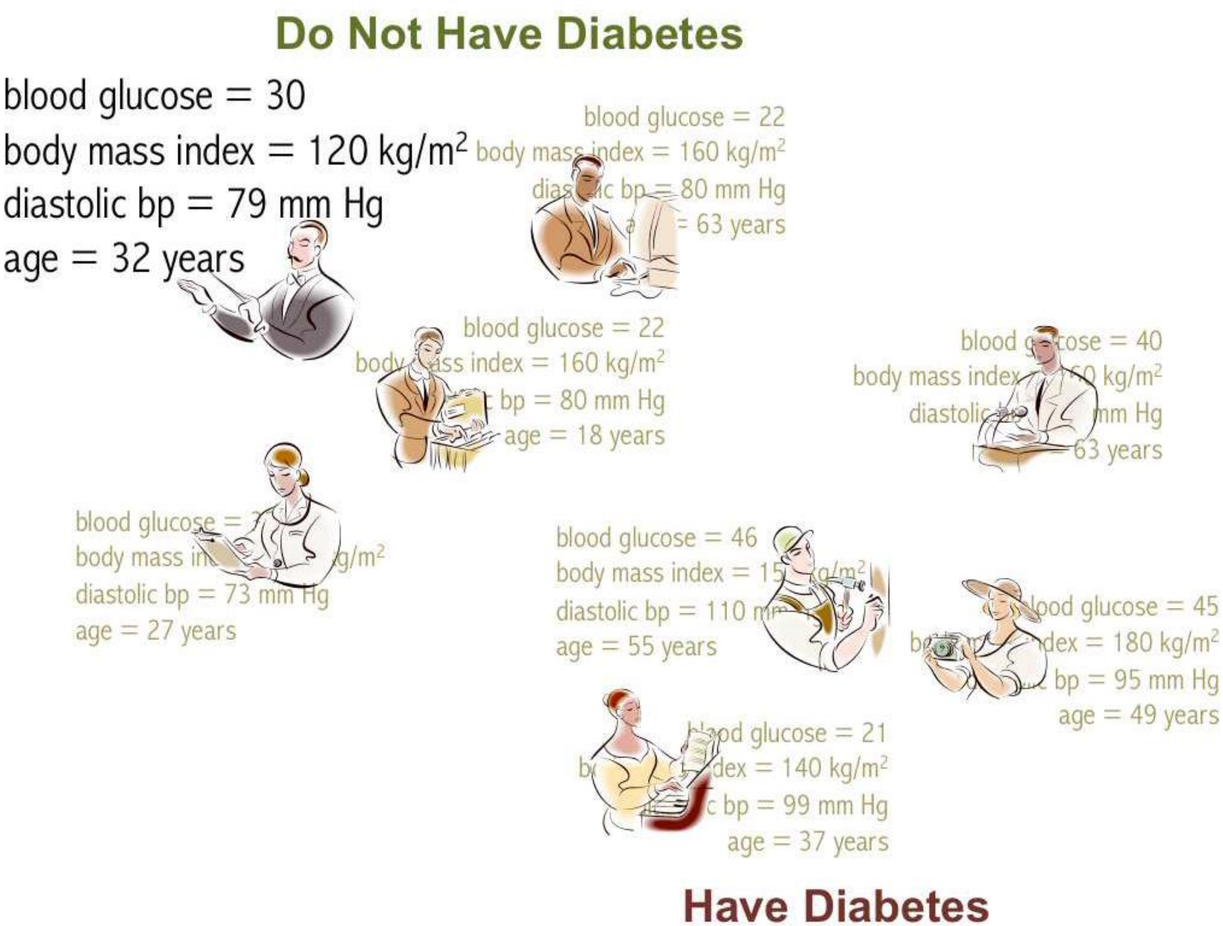
Classification

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

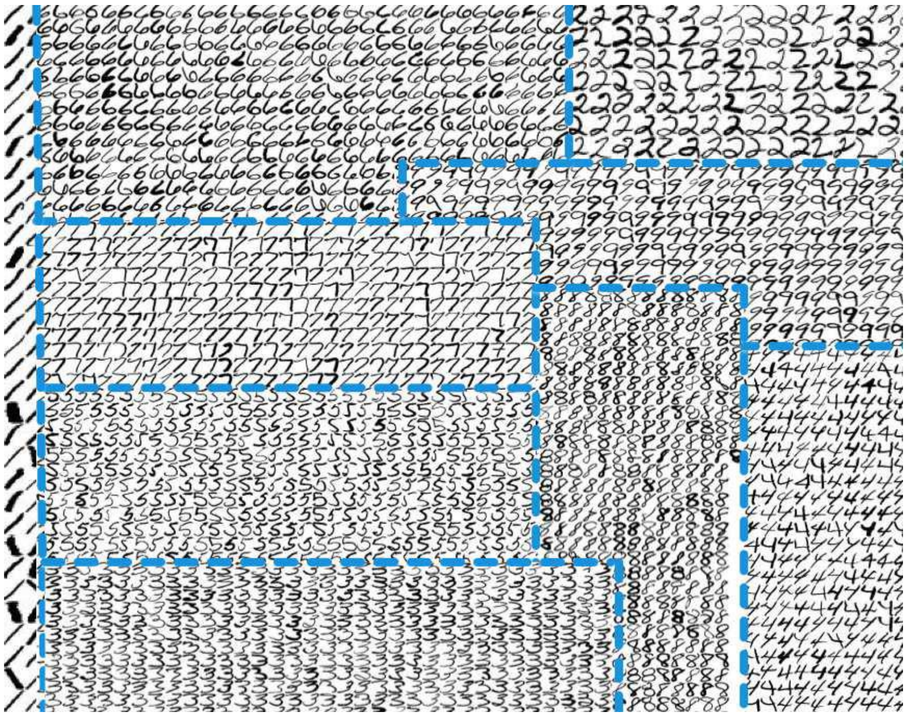
Regression

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

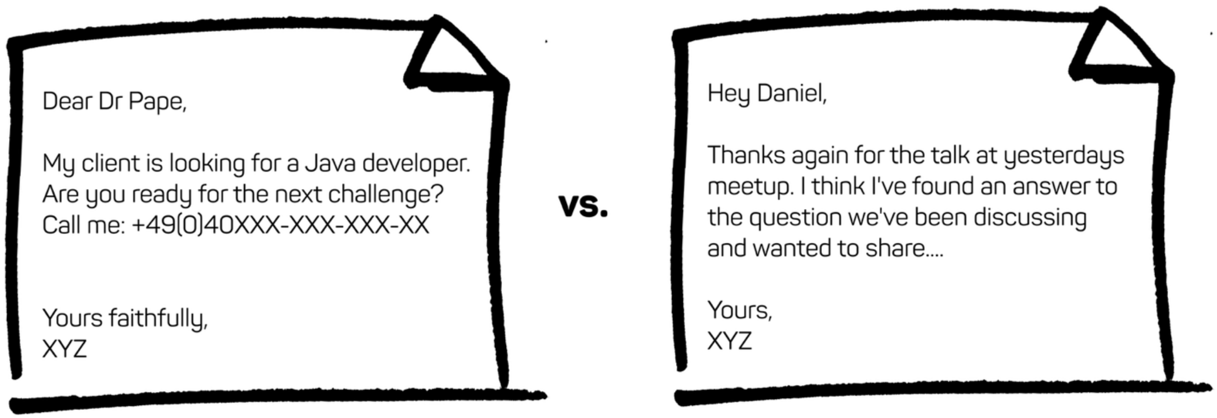
Classification – Medical Diagnosis



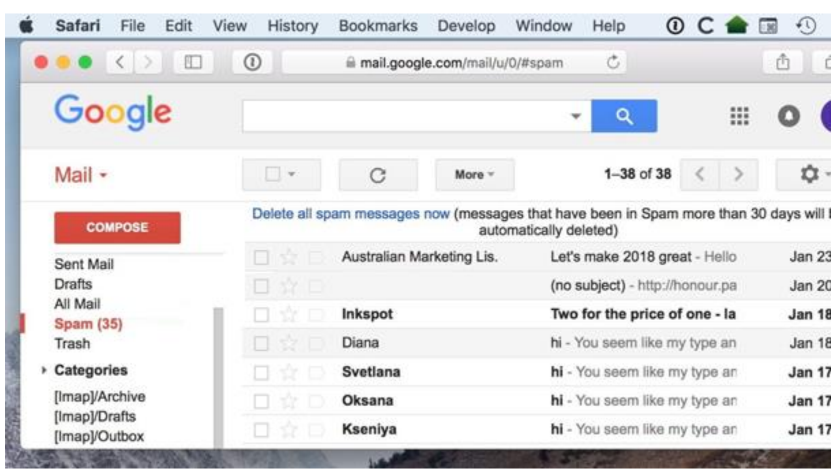
Classification – Digit Recognition



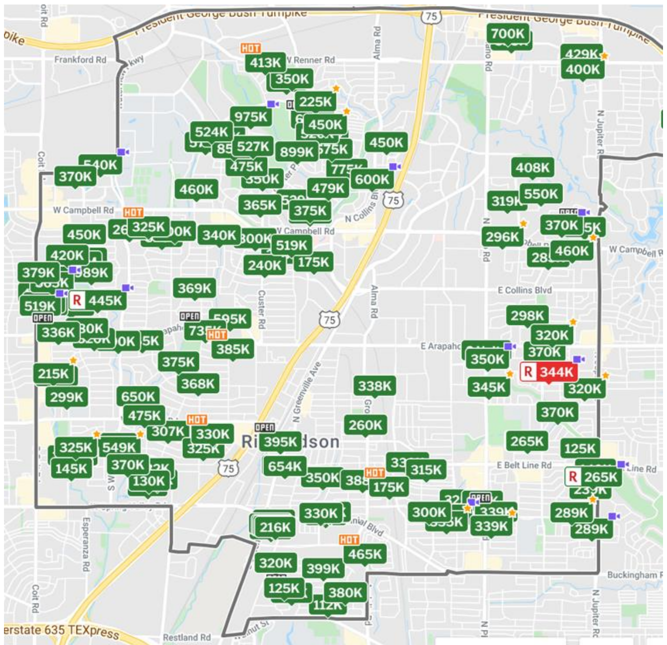
Classification – Spam



SPAM **HAM**



Regression – Housing Price Prediction



1,934 Sq. Ft.
$213 / Sq. Ft.

Status: Active

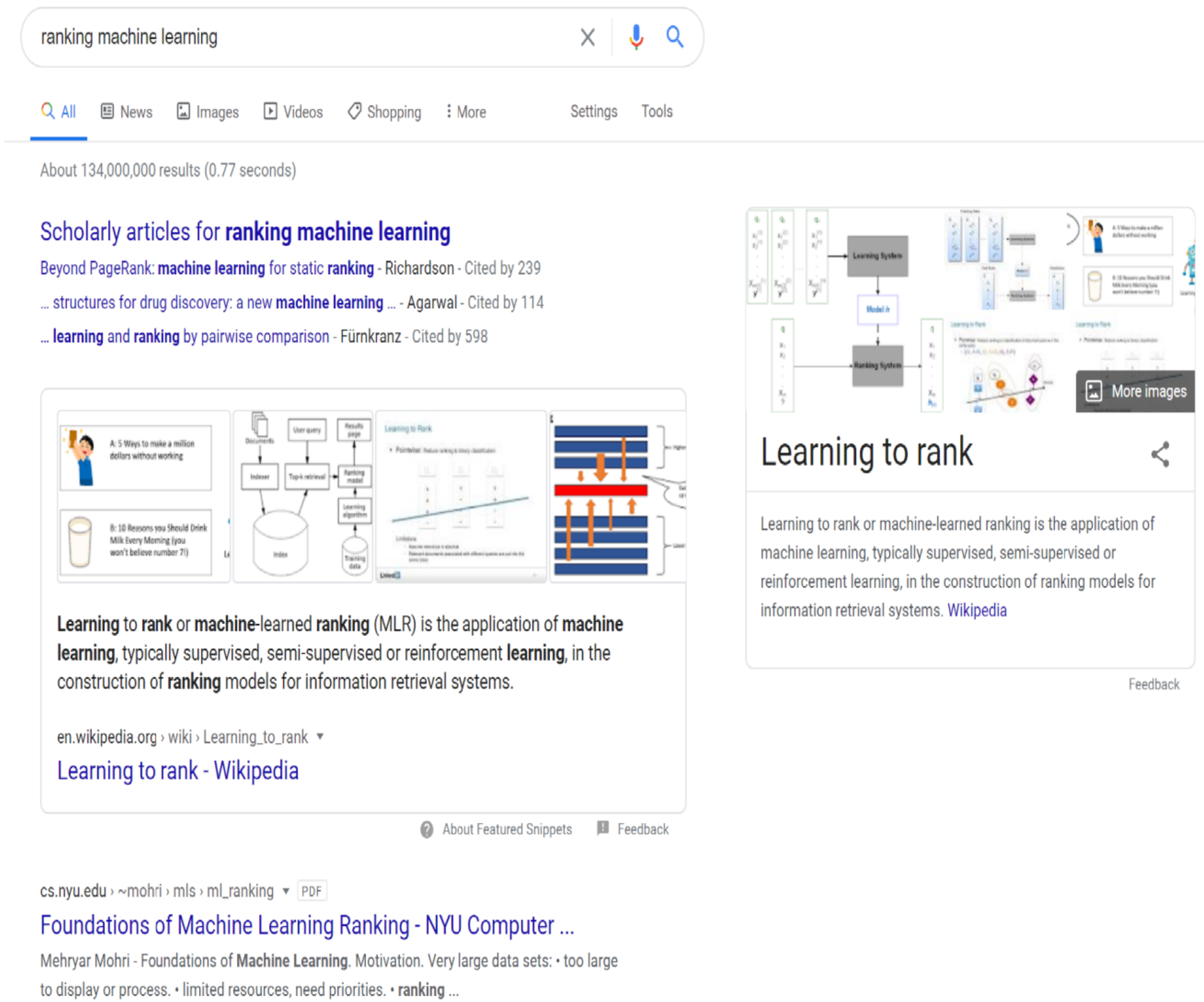
Redfin Estimate: $411,577 On Redfin: 2 days

Overview | Property Details | Property History | Schools | Tour Insights | Public Facts | Redfin

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		

Ranking – Search Engines



ranking machine learning

All | News | Images | Videos | Shopping | More | Settings | Tools

About 134,000,000 results (0.77 seconds)

Scholarly articles for **ranking machine learning**

Beyond PageRank: **machine learning** for static **ranking** - Richardson - Cited by 239

… structures for drug discovery: a new **machine learning** … - Agarwal - Cited by 114

… **learning** and **ranking** by pairwise comparison - Fürnkranz - Cited by 598

Learning to **rank** or **machine**-learned **ranking** (MLR) is the application of **machine learning**, typically supervised, semi-supervised or reinforcement **learning**, in the construction of **ranking** models for information retrieval systems.

en.wikipedia.org › wiki › Learning_to_rank

Learning to rank - Wikipedia

About Featured Snippets | Feedback

cs.nyu.edu › ~mohri › mls › ml_ranking PDF

Foundations of Machine Learning Ranking - NYU Computer ...

Mehryar Mohri - Foundations of **Machine Learning**. Motivation. Very large data sets: • too large to display or process. • limited resources, need priorities. • **ranking** ...

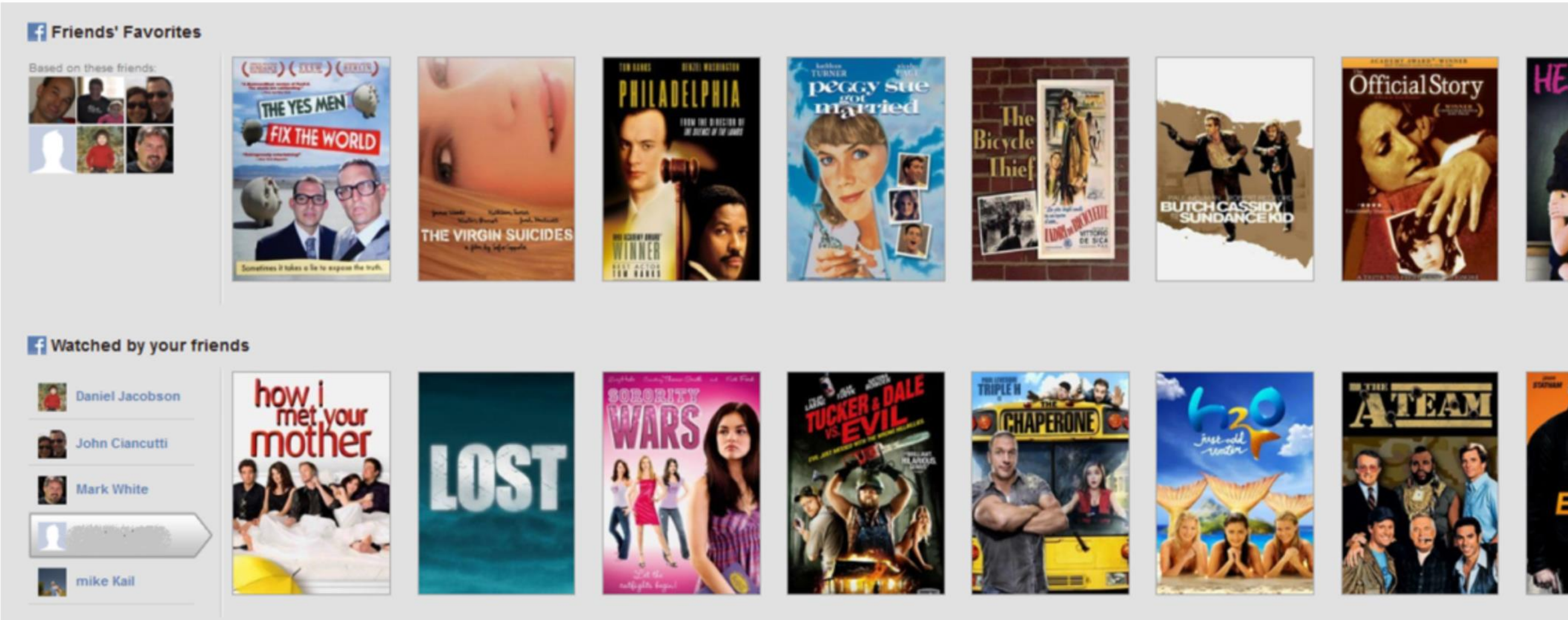
More images

Learning to rank

Learning to rank or machine-learned ranking is the application of machine learning, typically supervised, semi-supervised or reinforcement learning, in the construction of ranking models for information retrieval systems. Wikipedia

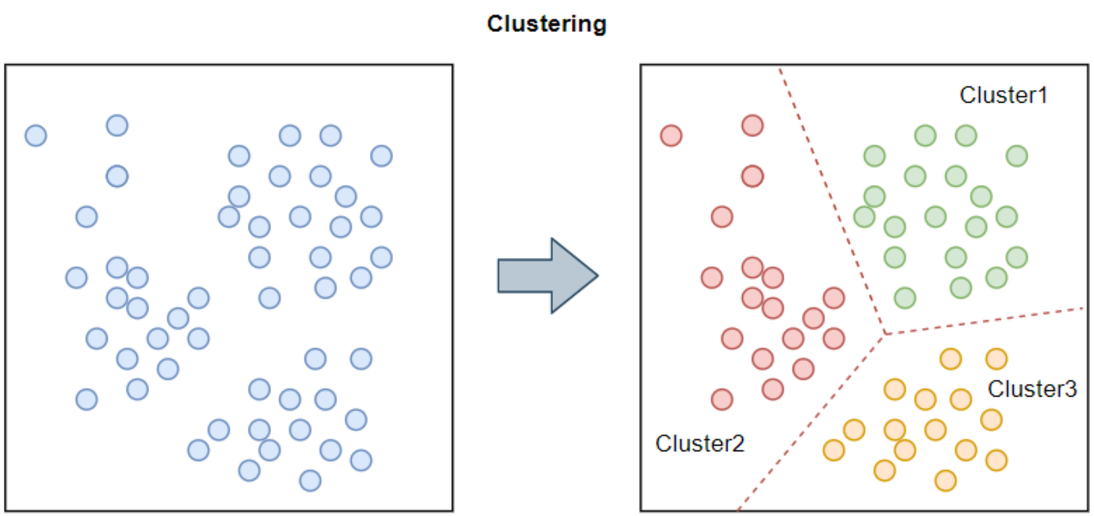
Feedback

Recommendation – Movie Recommendations



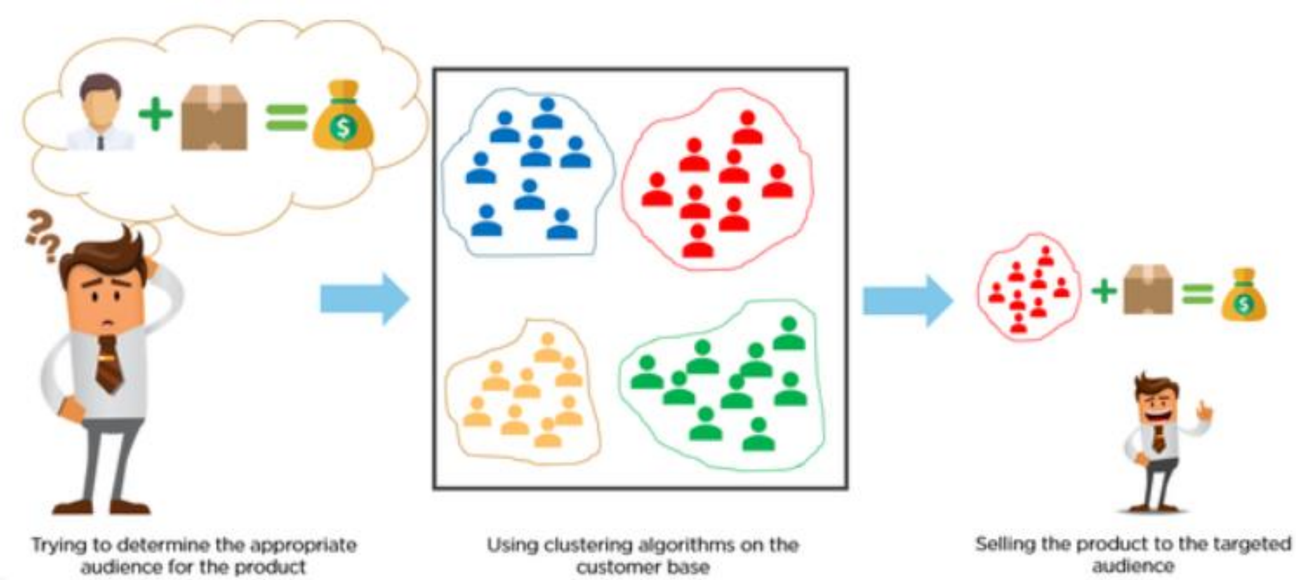
Unsupervised Learning

- **Input:** $x^{(1)}, \dots, x^{(M)}$
  - $x^{(m)}$ 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

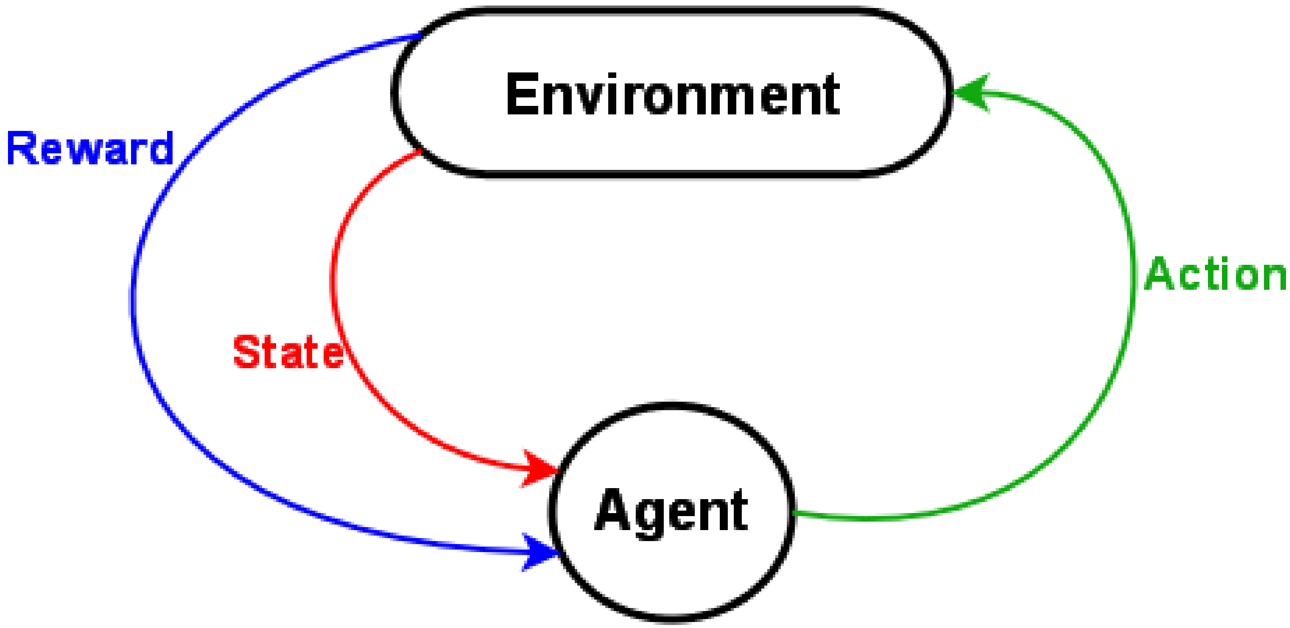


Applications of Unsupervised Learning

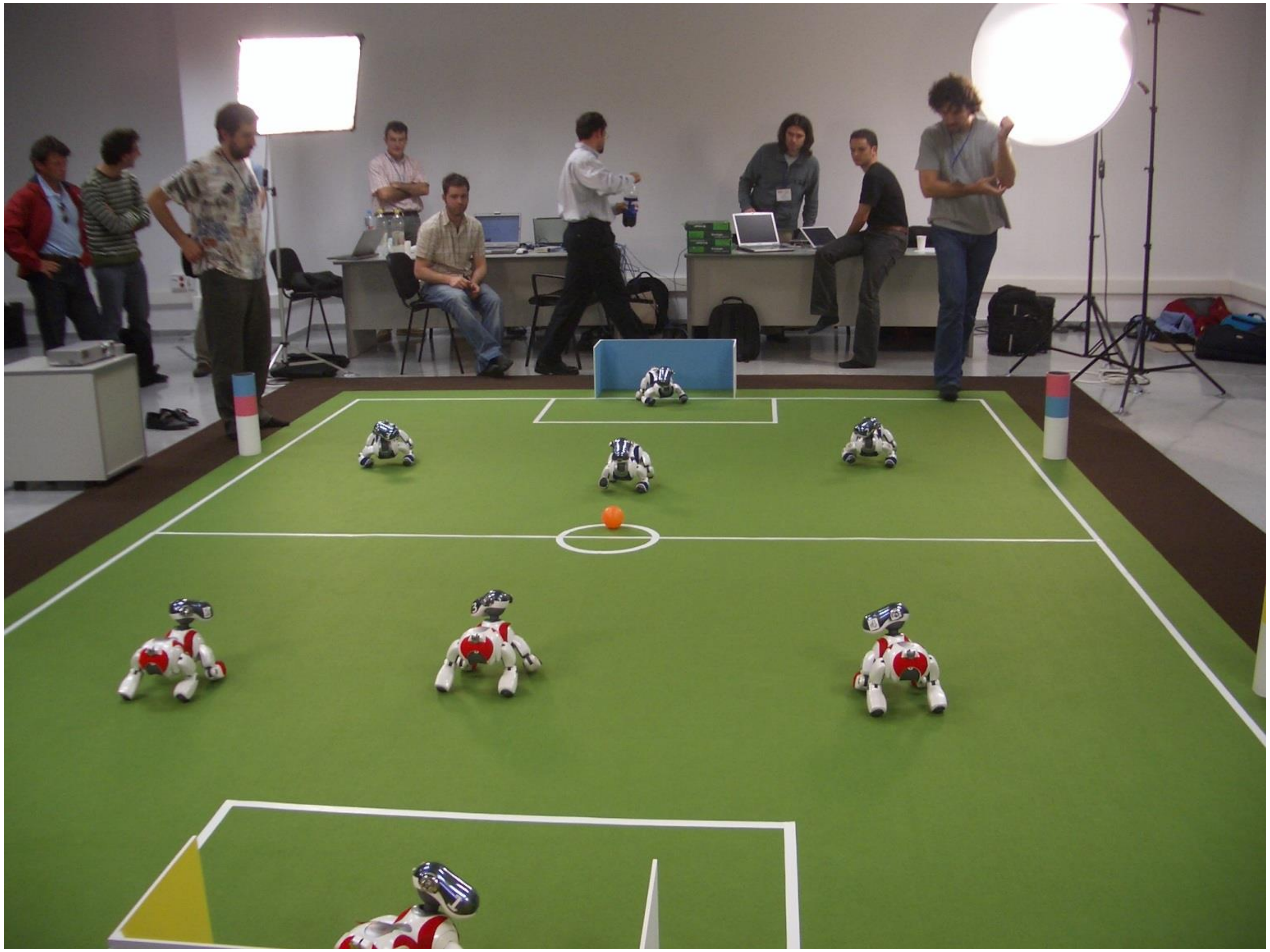
- Item Categorization
- Clustering Customers
- Similar Item Recommendation
- Outlier Detection



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

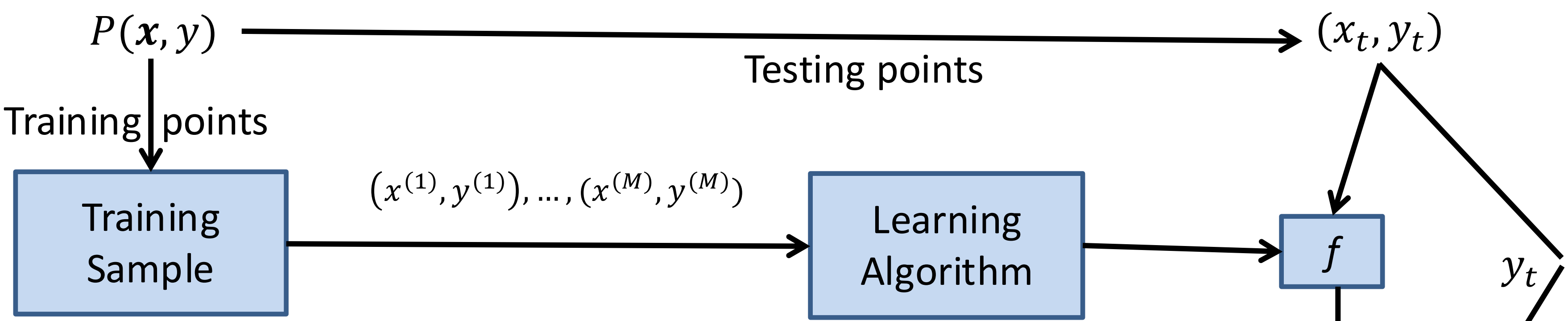
- **Training Example:** <**x**,y>
  - **x:** feature vector (describes the attributes of something)
  - **y:** label (continous values for regression problems: [1,2,…,k] for classification problems)
- **Training set** A set of training examples drawn randomly from P(**x**,y)
  - **Key Assumption:** Independent 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 h 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 h"
- **Hypothesis Space:** The space of all hypotheses that can in principle be considered and returned by the learning algorithm

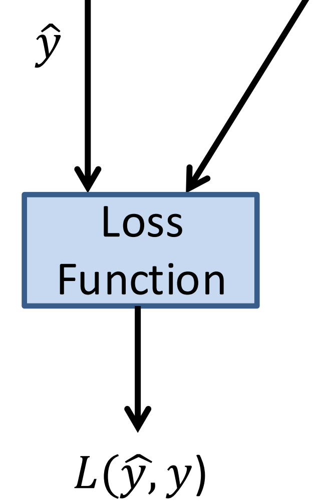
Supervised Learning

- **Given**: Training examples $(x, f(x))$ for some unknown function f.
- **Find**: A good approximation to f.

- Situations where there is no human expert
  - x: bond graph of a new molecule
  - f(x): predicted binding strength to AIDS protease molecule
- Situations where humans can perform the task but can't describe how they do it
  - x: picture of a hand-written character
  - f(x): ascii code of the character
- Situations where the desired function is changing frequently
  - x: description of stock prices and trades for last 10 days
  - f(x): recommended stock transactions
- Situations where each user needs a customized function f
  - x: incoming email message
  - f(x): importance score for presenting to the user (or deleting without presenting)

Supervised Learning Workflow





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

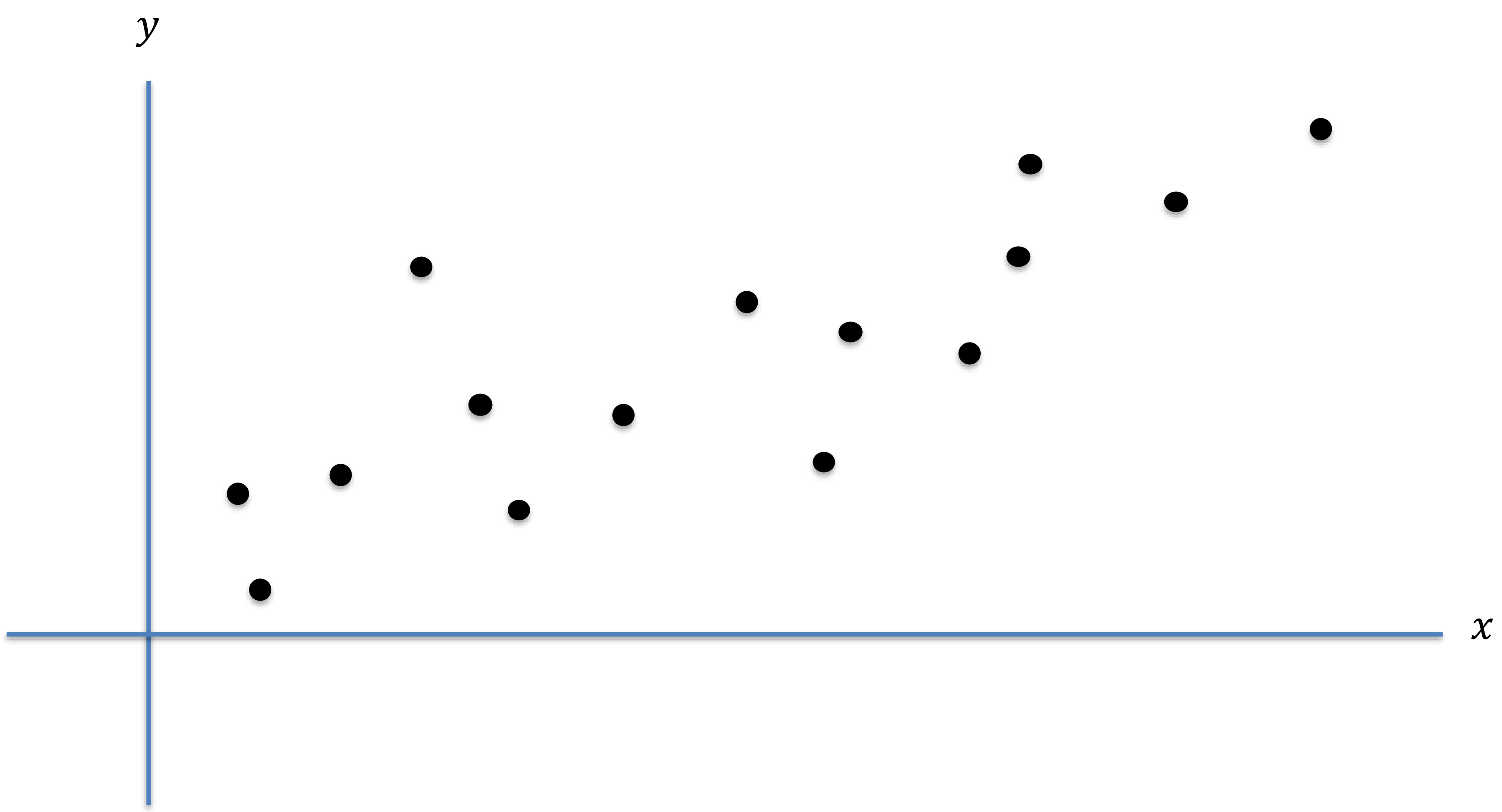
$$\min_f \sum_{i=1:m} L(f(x^i), y^i)$$

- **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

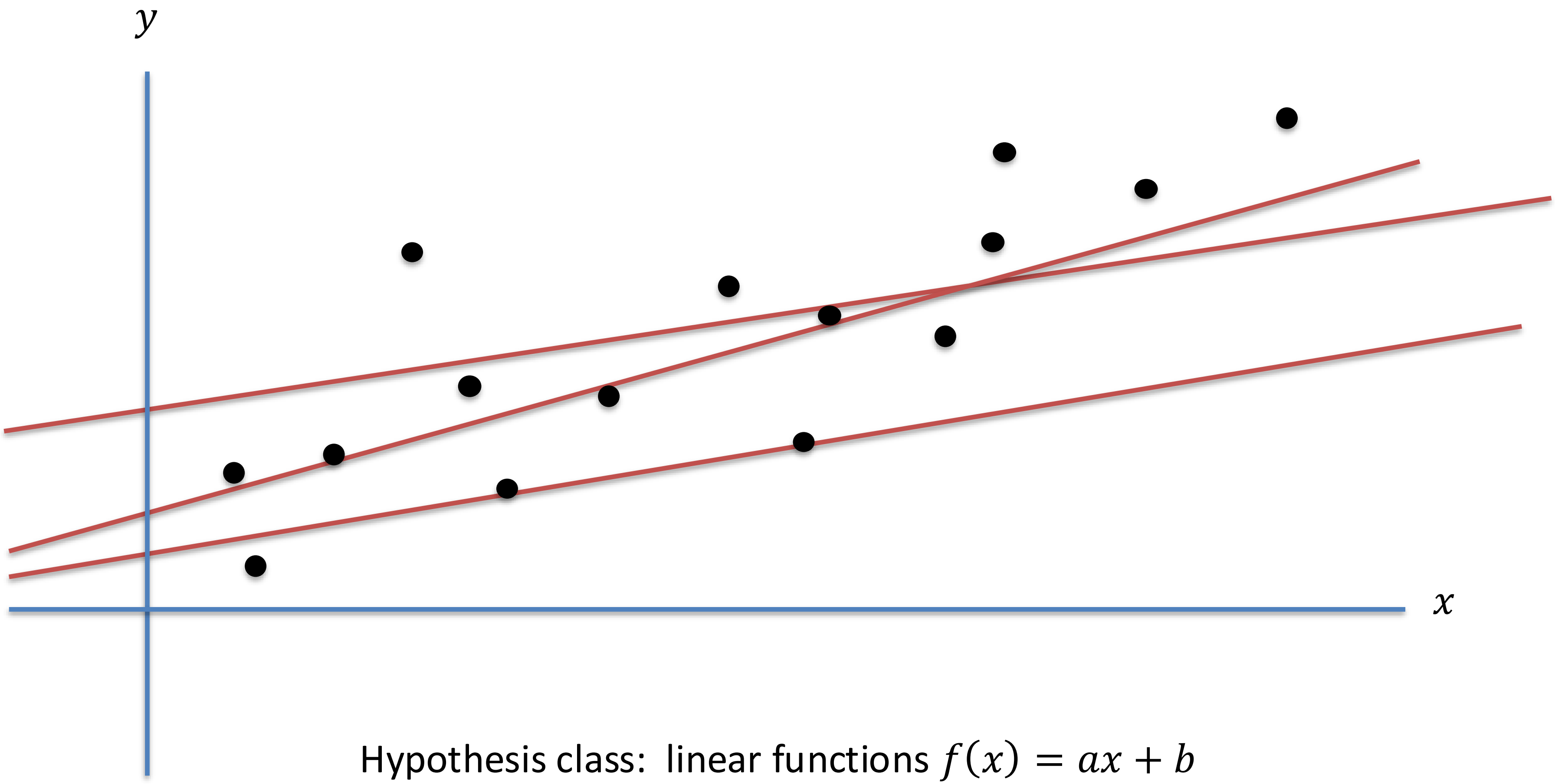
Linear Regression

- Simple linear regression
  - Input: pairs of points $\left(x^{(1)}, y^{(1)}\right), \dots, (x^{(M)}, y^{(M)})$ 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 $a \in \mathbb{R}^d, b \in \mathbb{R}$
  - Error metric and Loss Function: squared difference between the predicted value and the actual value

Regression



Regression



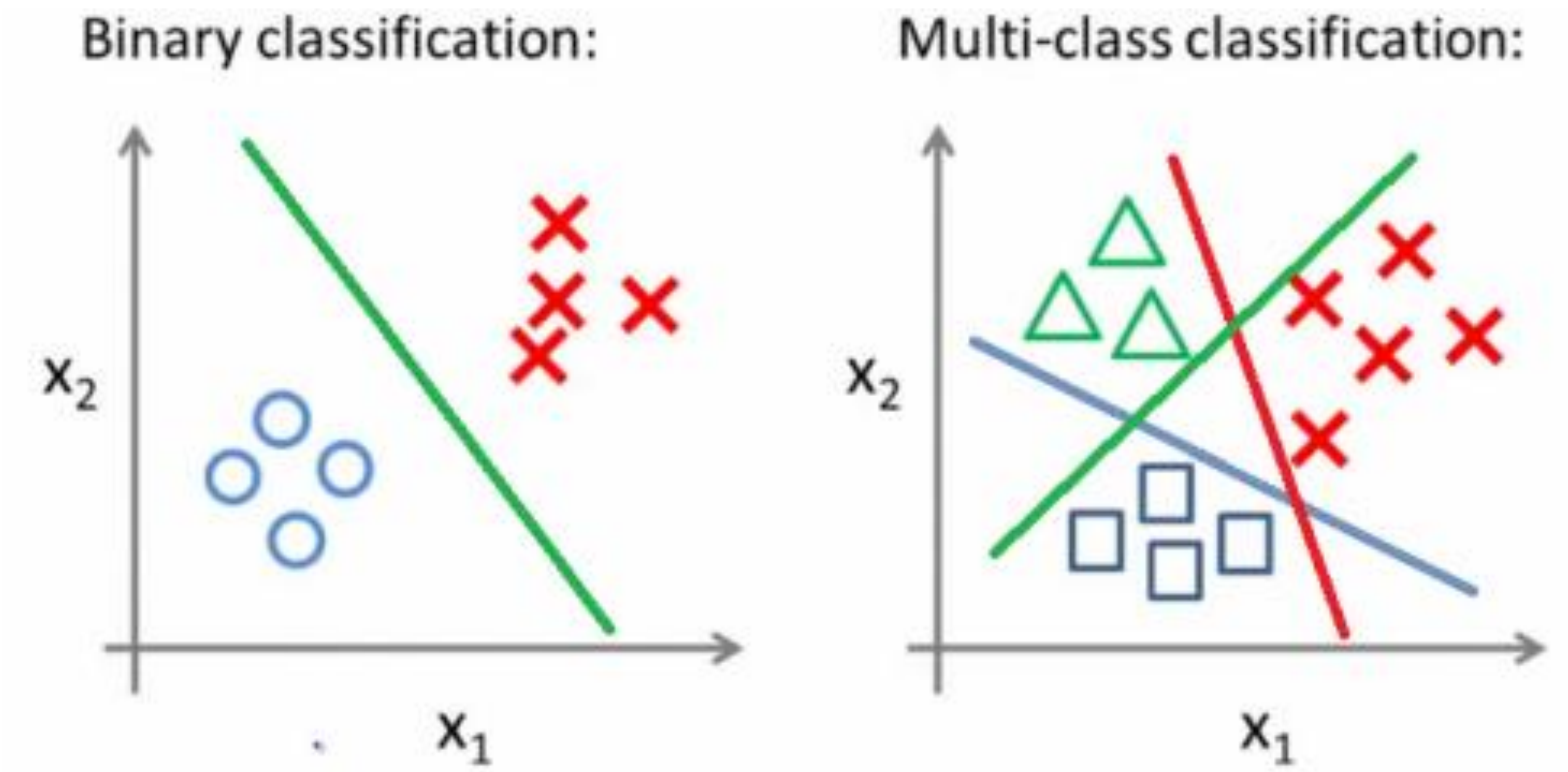
Hypothesis class: linear functions $f(x) = ax + b$

How do we compute the error of a specific hypothesis?

Linear Classification

- Simple linear classification
  - Input: pairs of points $\left(x^{(1)}, y^{(1)}\right), \dots, (x^{(M)}, y^{(M)})$ 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



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

	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