

What is Machine Learning?

Anurag Nagar

Motivating example

- I am interested in finding out which UTD students get internships (or jobs).
- Think of it as a function:
 $f: X \rightarrow Y$ (the best function for entire UTD)



X is an instance (student), Y is boolean.

X has certain attributes

$X = \langle X_1, X_2, \dots, X_n \rangle$

e.g.

$X_1 = \text{GPA}$, $X_2 = \text{Taken CS 6375?}$

$X_3 = \text{Years experience, ...}$

It's a binary classification problem.

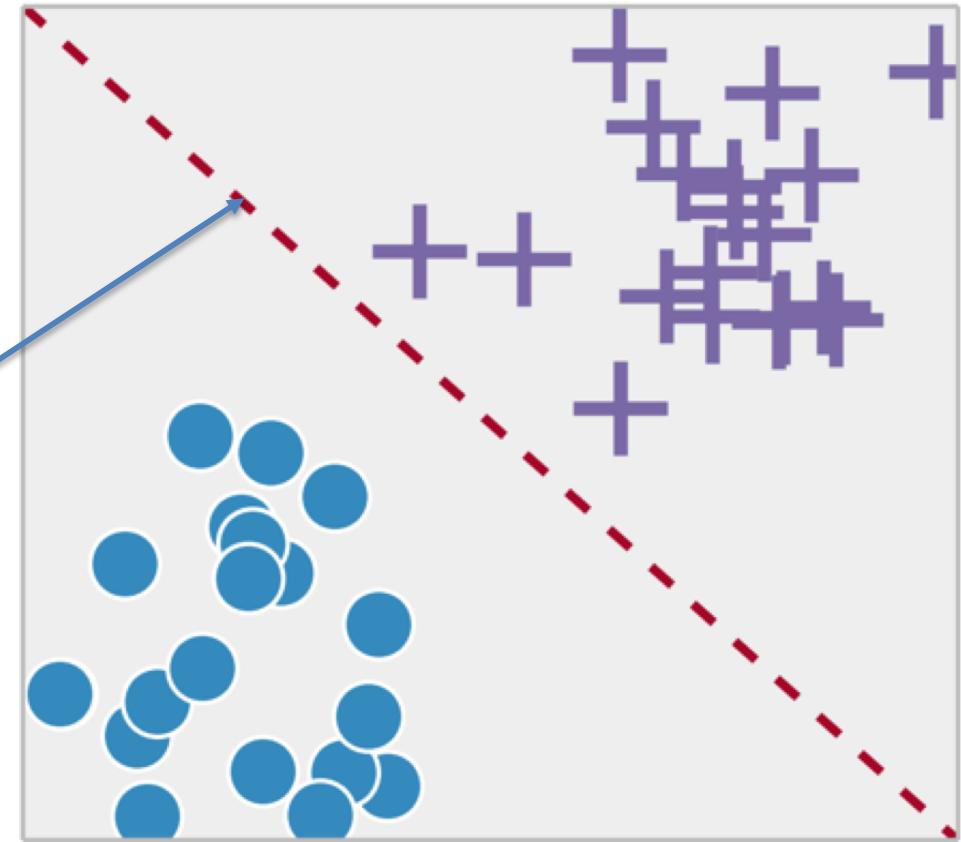
You will see lots of such cases in this course.

Learning Scenario

- You are given access to some training data:
 $\{(\mathbf{X}^1, Y^1), (\mathbf{X}^2, Y^2), \dots, (\mathbf{X}^n, Y^n)\}$
- Using this training data, you try to approximate the function f . This approximation is also called a **hypothesis** (h).
- Our aim is to design a learning model that can approximate f closely.
- If Y is Boolean (or has a fixed number of values), this is called a **classification** problem.

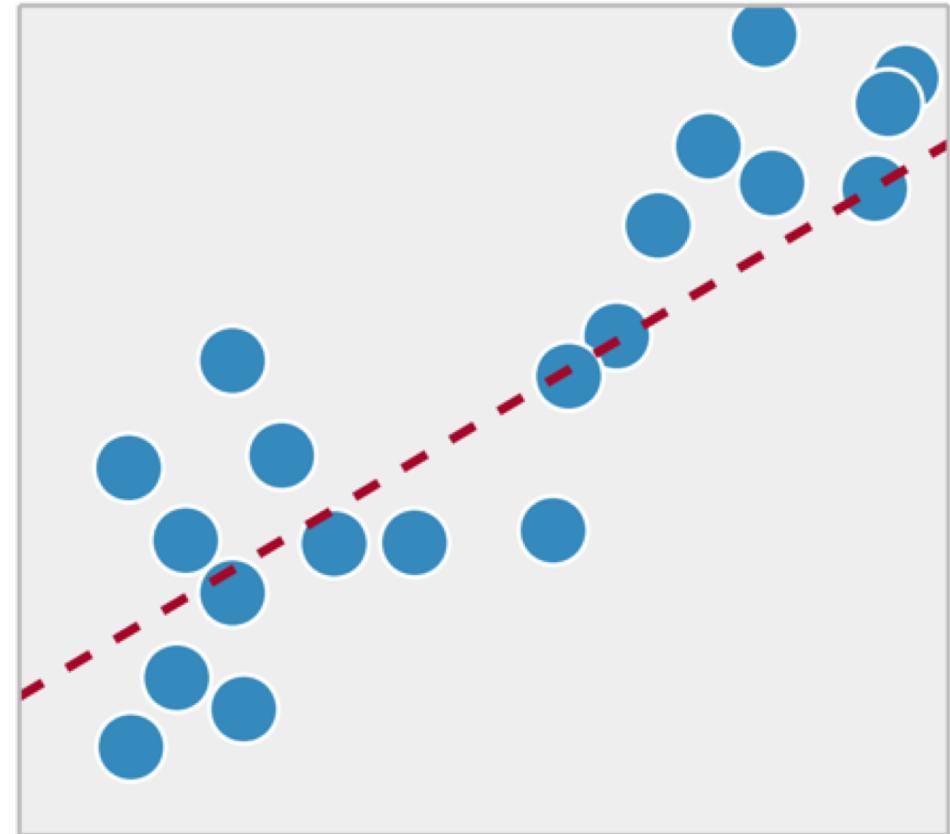
Classification

- Learning to differentiate between two or more classes.
- Learning the **separating boundary**
- Most common problem in ML
 - e.g. classes could be
 - spam or not spam
 - family or sports car
 - apple or orange



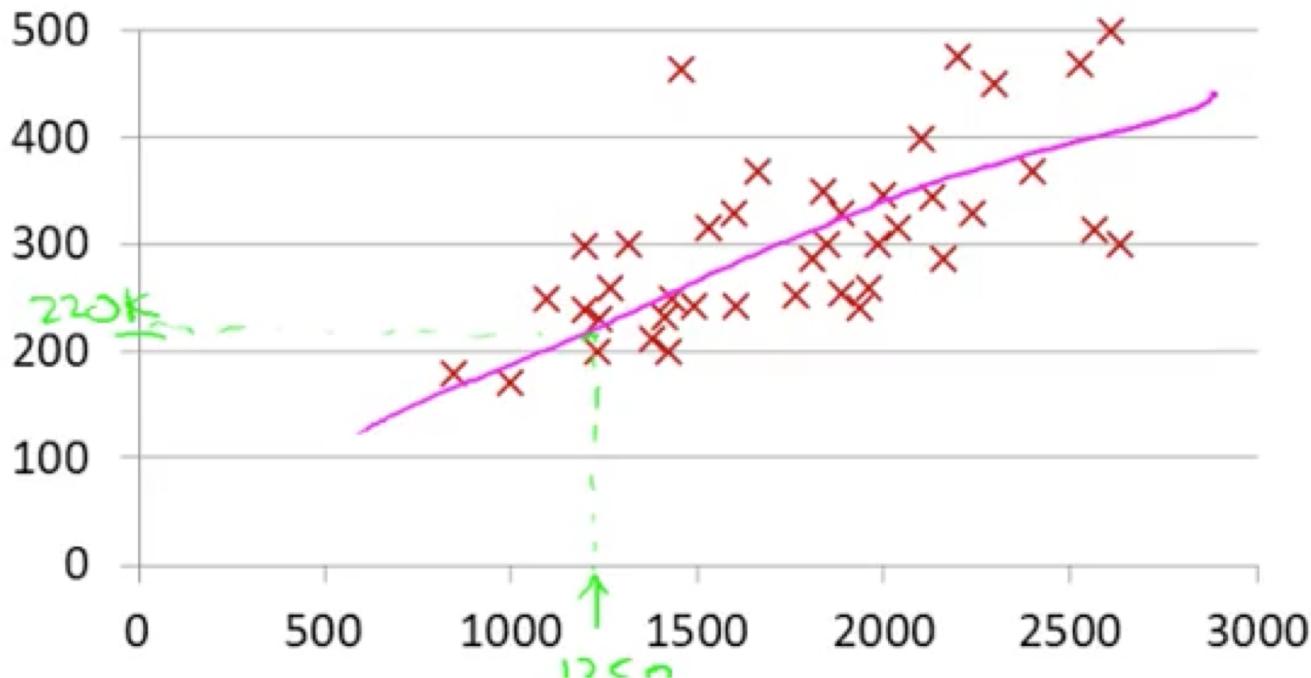
Regression

- Classification is different from regression.
- Regression is trying to fit the best model to approximate some real-valued data.
- Common in statistics, not ML.



Difference in output type

- If the output was real-valued, this problem can be solved by statistics.



Statisticians love regression – linear or non-linear.
It can be used to fit a curve to data points.

Classification

- Computer scientists love all things Boolean
- So, we have Boolean attributes, Boolean output.
- How do we handle this scenario??



0

OR



1

Components of ML

- What do you want for a ML project?
- A well-defined **task** - what are you trying to teach to the machine?
- Training data that will be used by the machine – also called training **experience**.
- A way to check the performance of the machine – called **error** metric.
- A smart machine **learns** from the past errors and improves its performance.

What is ML?

- There is a **task** – generally involving prediction
-e.g.

Will a student at UTD get internship?

Will a stock go up?

Will you play tennis?

Will a person be approved for credit card?

What is the price of a house?

Which digit does a handwritten image represent?

What is ML?

- It has an associated **performance** measure
i.e. how close is your prediction to actual value
- Error metric: If your hypothesis doesn't match the actual value, penalize the model.

$$E_D = \sum_{x \in D} 1(h(x) \neq y)$$

The above is the simplest error function.

$h(x)$ denotes your hypothesis and y is the actual value. D is the dataset.

What is ML?

- The model learns from **experience** i.e. data.
 - More data => Better performance
 - Is it always true?
Only if data is meaningful i.e. it's not **noise**
 - If you could see the **entire population** (entire dataset), you would get the best learning model.
 - Is it possible?
Can you poll the entire US population?
Probably Not!!
- In this class, we will work with samples of data

Definition

“A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.”

-- Tom Mitchell, Carnegie Mellon University

Simpler Definition

- Design of algorithms that **learn from data** and improve performance on a predictive task.
- Development of computational methods **using experience** to improve performance.

Common Theme of this class:

- More **training** data is always good

Provided data is **meaningful** and is **labeled** (in case of supervised learning).

Think:

Why does Google work so well?

When is ML a good choice?

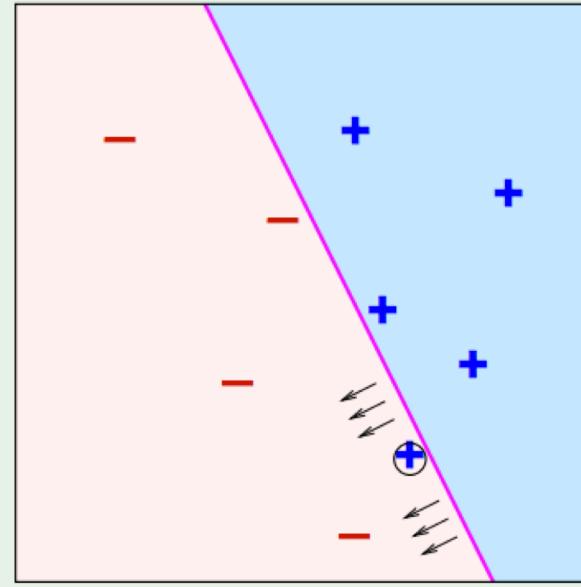
- **Learning used when**

- A pattern exists
- There exists a correlation between predicted variable and features.
- We have data

- **Supervised learning**

- Unknown target function f
 $y = f(x)$
- You get data & labels
 $(x_1, y_1) (x_2, y_2), \dots$
- Learning algorithm picks a function $g \approx f$
(approximation)

Example: Perceptron Learning Algorithm



Simple, but not obvious

- When can we use ML?
 - There is some definite relation/correlation between X and Y.
i.e. there is a deterministic or highly probable function f
 - X comes from a well-defined distribution (we don't need to know the details yet)
=> X is not a set of random variables.

Simple, but not obvious

- When can we NOT use ML?
- There is no correlation between X and Y.
- There is no clear function f
- X is a set of random variables
 - => Can ML predict the lottery?
 - => Can you use it at the casino?
- Can we use it to predict the stock market?

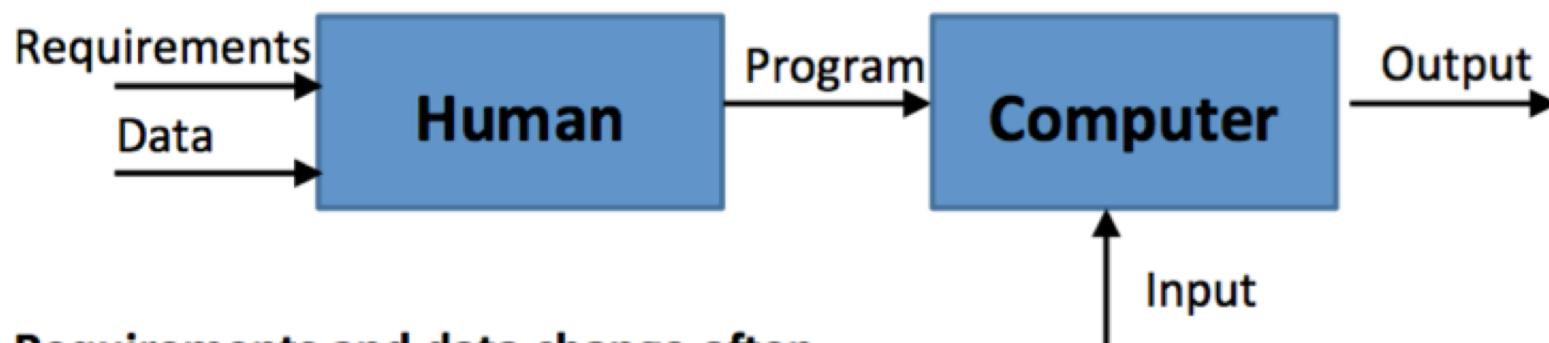


Stock Market



Traditional Programming

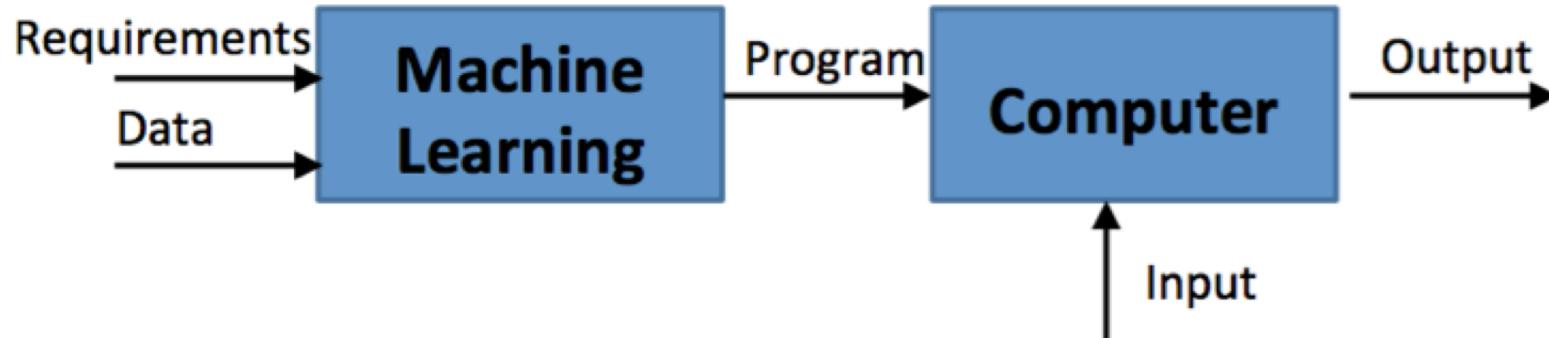
- Getting computers to program themselves
- Writing software is the bottleneck, let data do the work



Requirements and data change often

Machine Learning

In ML, the algorithm learns from data and outputs a program. For example, a self-programming robot.

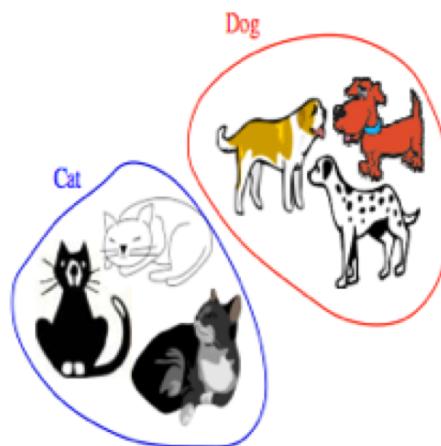


Types of Learning

- **Supervised (inductive) learning**
 - Training data includes desired outputs
- **Unsupervised learning**
 - Training data does not include desired outputs
 - Find hidden/interesting structure in data
- **Semi-supervised learning**
 - Training data includes a few desired outputs
- **Reinforcement learning**
 - the learner interacts with the world via “actions” and tries to find an optimal policy of behavior with respect to “rewards” it receives from the environment

Supervised Learning - Inductive

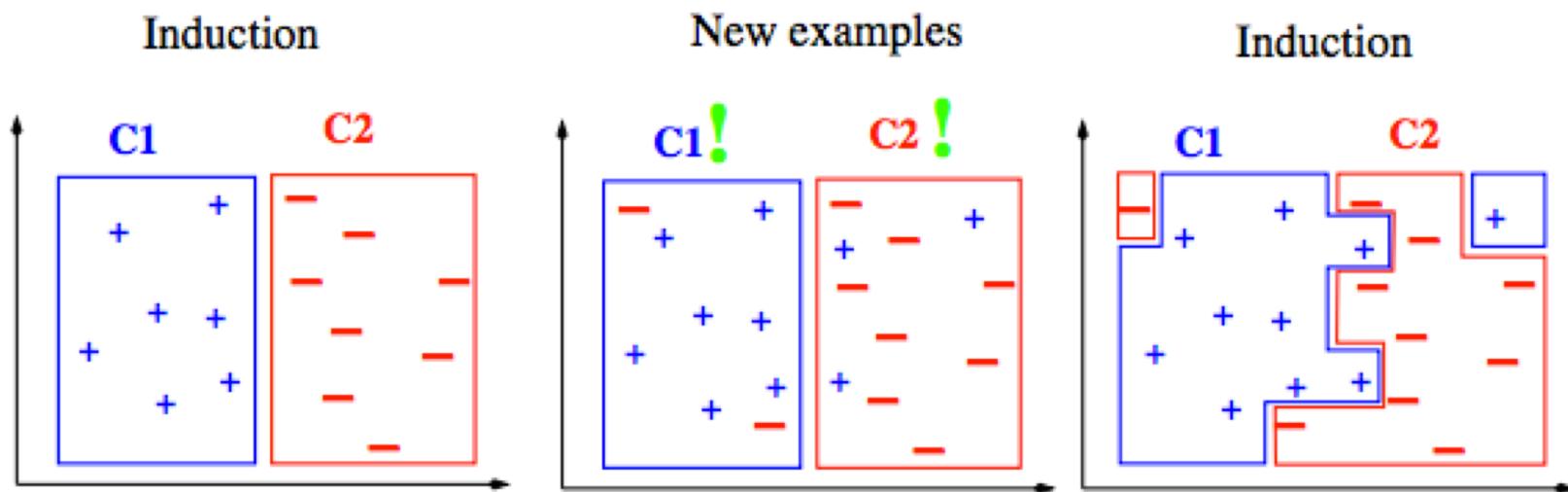
- **Inductive:** Tries to discover general concepts from a limited set of training examples.
=> Generalization
 - Based on search of similar characteristics in different classes of examples.
 - e.g.



Given labeled examples, you find features that are common within each class.

Supervised Learning - Inductive

- **Inductive:** goes from specific to general
 - tries to obtain new knowledge.
 - new data points may force you to change old hypothesis.



Inductive vs Deductive

- **Deductive:** uses given premises and logical arguments to infer conclusions.
- tries to obtain knowledge that is implicit in original knowledge.
- Classic example:
 - * All men are mortal. (major premise)
 - * Socrates is a man. (minor premise)
 - * Socrates is mortal. (conclusion)

Inductive vs Deductive

- In this class, we will focus on inductive learning.
- Can you see any issues with inductive approach?
 - What do you expect the data to be?
 - How do you expect the learner to behave?
 - When you make a conclusion, are you 100% sure or are you probably sure?
 - Is it even possible to obtain 100% accuracy on training and test data?
 - ...



Learning

- Inductive learning is supervised learning because the training data has the **class labels**.
=> That's what you are trying to learn.
=> So you create an algorithm that separates the two classes based on features.
- What if you just want to find patterns in data i.e. how can you find similar items based on their attributes. => **Unsupervised learning**
=> **No labeling provided or you don't care for labels**.
=> You want to create a natural grouping of people based on shared interests on FB.

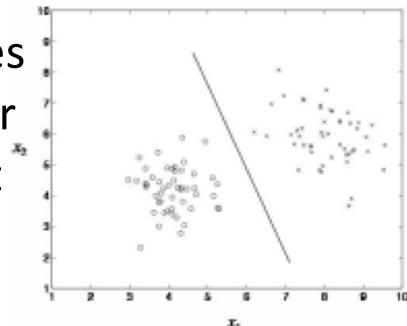
Types of learning task

- Supervised: correct output known for each training example
 - Learn to predict output when given an input vector
 - Classification: 1-of-N output (speech recognition, object recognition, medical diagnosis)
 - Regression: real-valued output (predicting market prices, customer rating)
- Unsupervised learning
 - Create an internal representation of the input, capturing regularities/structure in data
 - Examples: form clusters; extract features
 - How do we know if a representation is good?
- Reinforcement learning
 - Learn action to maximize payoff

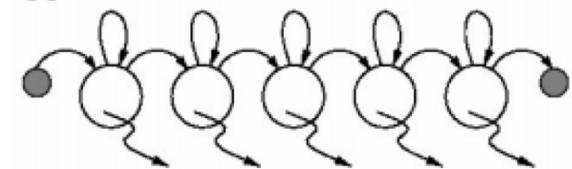
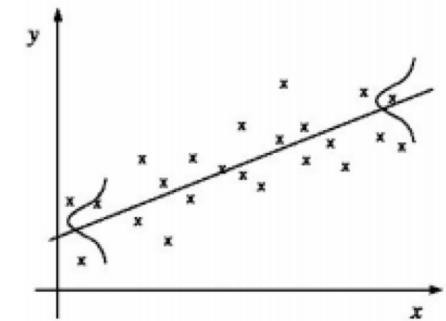
Supervised Learning

- Classification
 - Outputs are categorical (1-of-N)
 - Inputs are anything
 - Goal: select correct class for new inputs
 - Ex: speech, object recognition, medical diagnosis
- Regression
 - Outputs are continuous
 - Inputs are anything (typically continuous)
 - Goal: predict outputs accurately for new inputs
 - Ex: predicting market prices, customer rating of movie
- Temporal Prediction
 - Goal: perform classification/regression on new input sequences values at future time points
 - Given input values and corresponding class labels/outputs at some previous time points

You have N choices
for output, learner
needs to find best
1



You have infinite output
choices.



Applications of ML

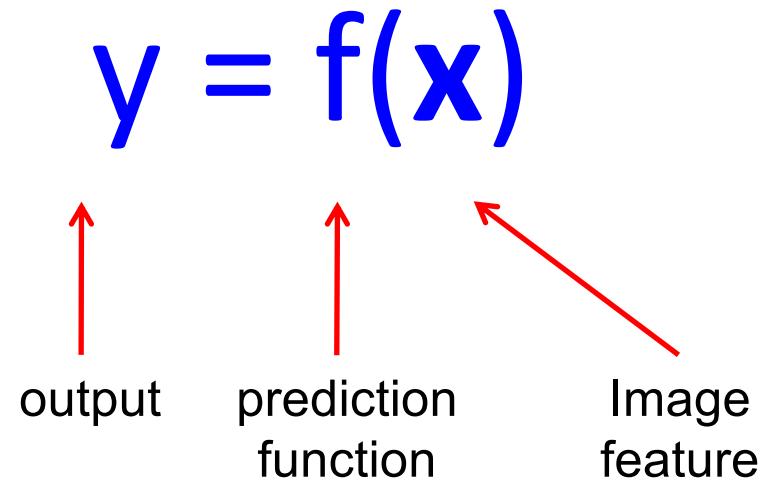
- Pattern Identification:
 - facial, fingerprint, gene sequence
- Differentiation:
 - spam or non-spam
 - cancerous or non-cancerous cells
 - normal traffic or hacker traffic
- Finding association rules
- Recommender Systems
- ... Many more

The machine learning framework

- Apply a prediction function to a feature representation of the image to get the desired output:

$$f(\text{apple}) = \text{"apple"}$$
$$f(\text{tomato}) = \text{"tomato"}$$
$$f(\text{cow}) = \text{"cow"}$$

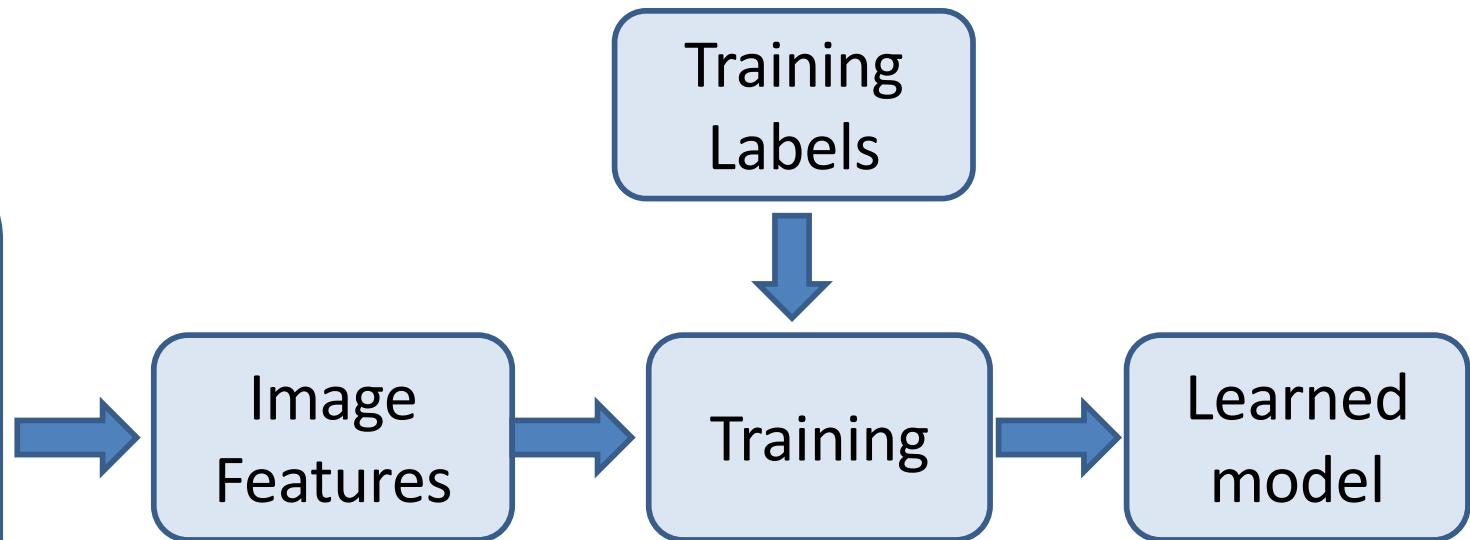
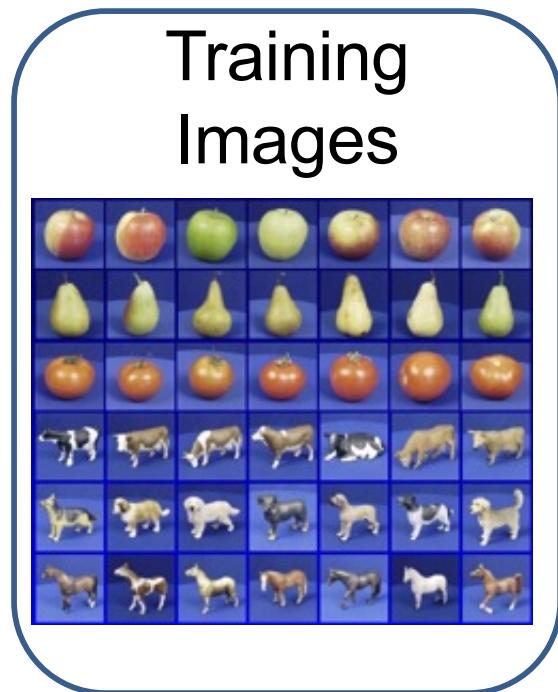
The machine learning framework



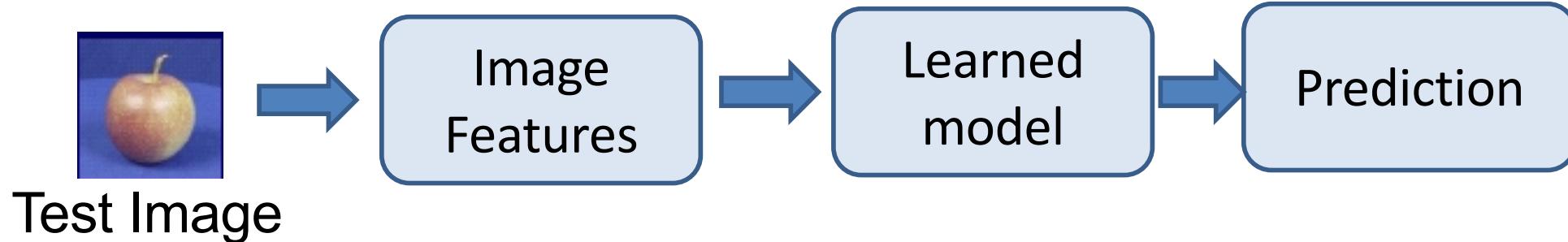
- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* \mathbf{x} and output the predicted value $y = f(\mathbf{x})$

Steps

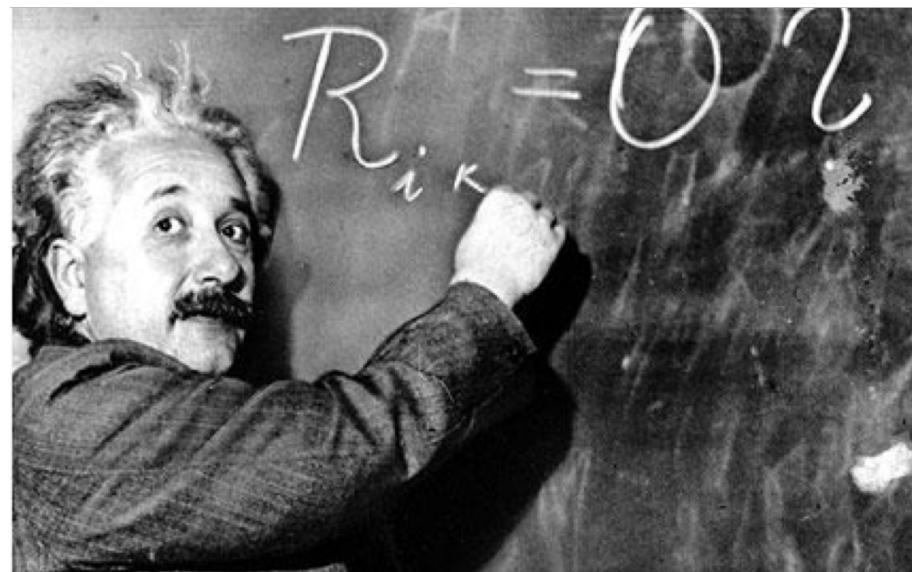
Training



Testing



Let's do some theory



Formal notation of learning

- You own a credit card company.
- You get a lot of applicants, your job is to design the best classifier for approving them.
- You have some historical data to rely upon.

$$x^j = (x_1, x_2, \dots, x_n)^T$$

Input vector for customer x^j

$$X = \{x^1, x^2, \dots, x^N\}$$

set of all customers

$$y = \{0,1\}$$

Binary Output

$$f : X \rightarrow Y$$

Ideal target function i.e. if you had entire data in front of you

Formal notation of learning

$(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)$

You have this data

$H = \{h_1, h_2, \dots, h_N\}$

The set of all possible hypothesis. Doesn't matter if they are meaningful or not

$g : X \rightarrow Y,$

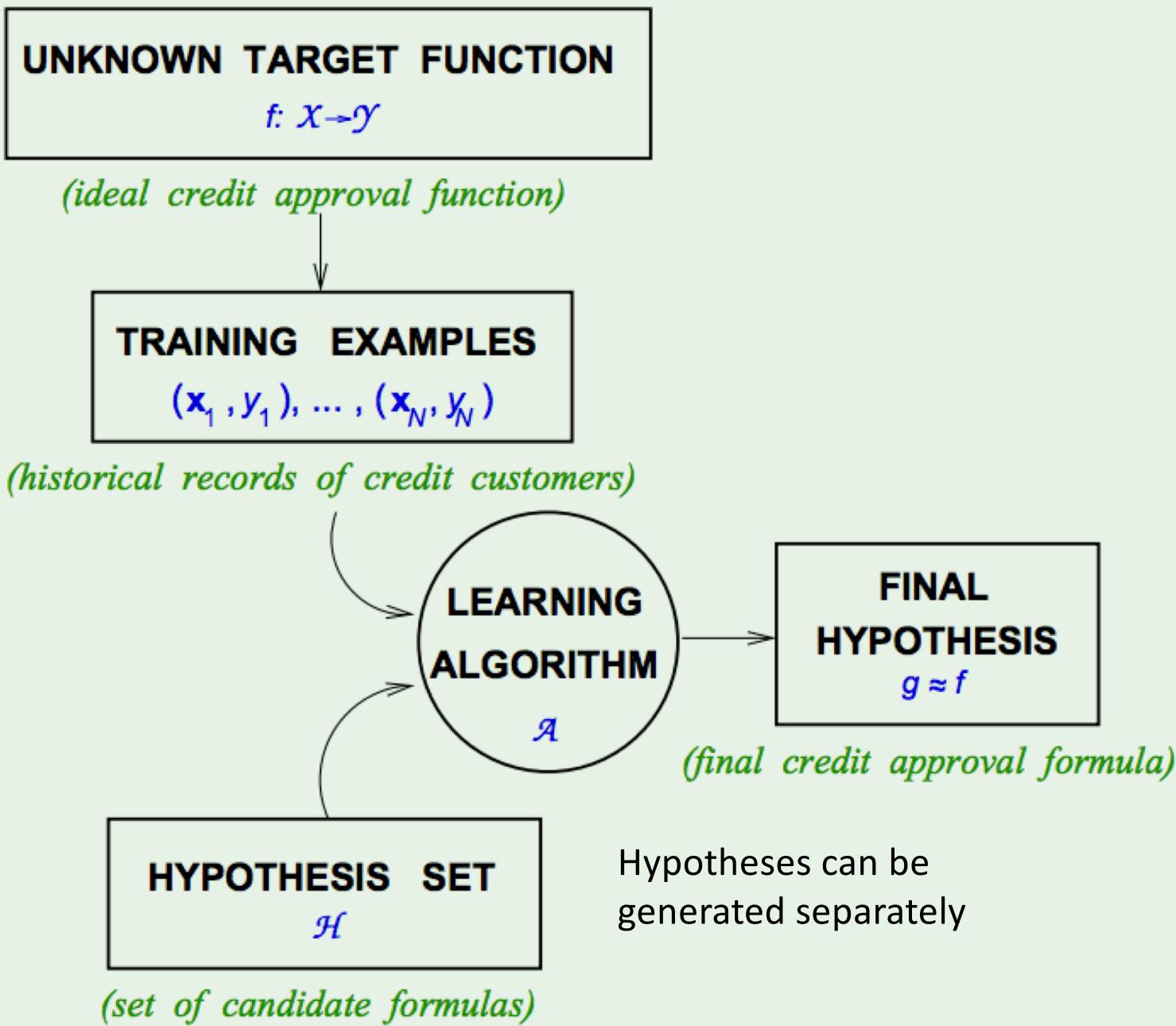
The best hypothesis you can come up with given the data

$g \in H$

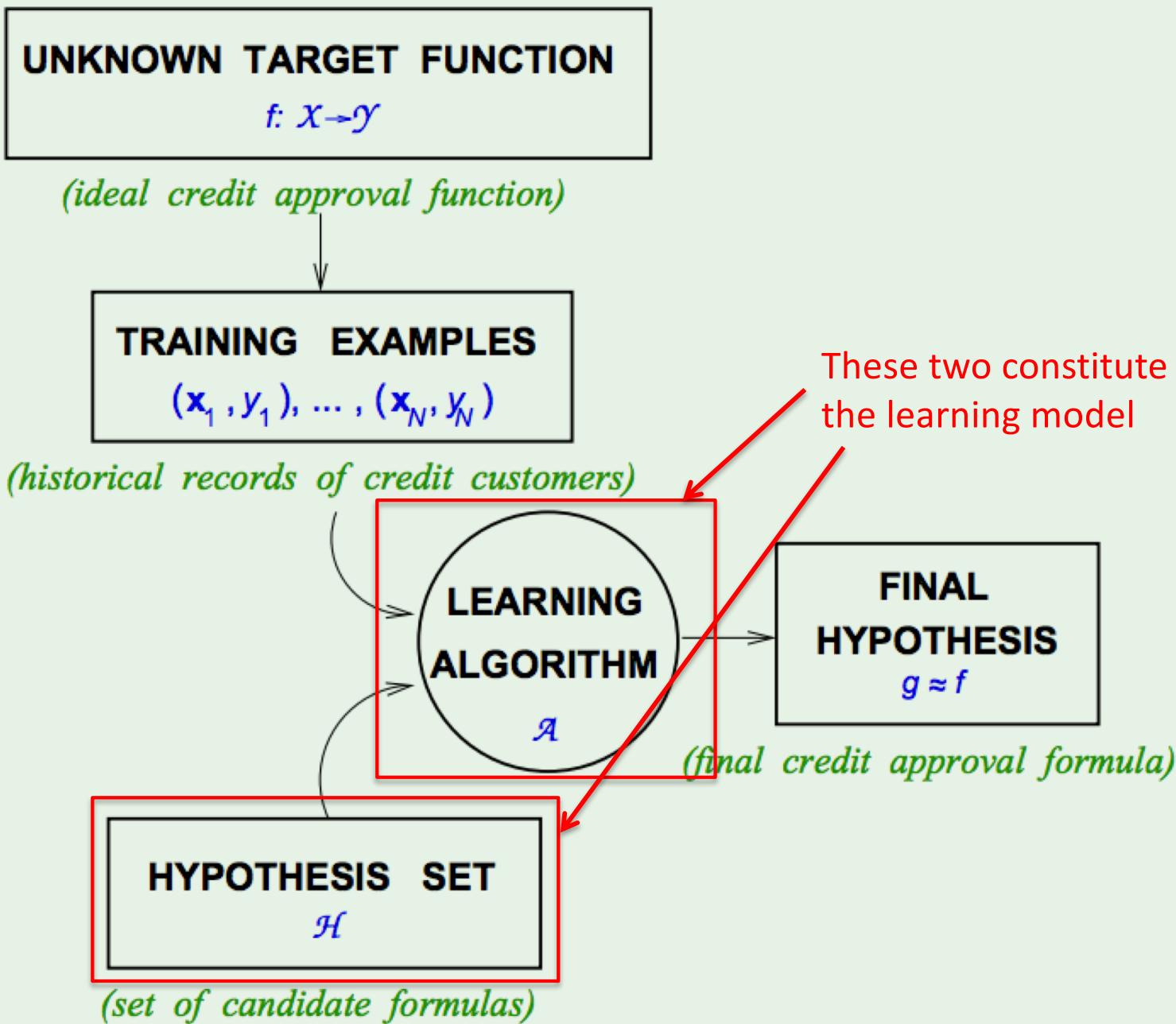
Can you really know f ?

- No, you can only try to approximate it by g
- Your approximation is only as good as the data that you see.

Formal notation of learning



Formal notation of learning



Really simple, right?



- All you have to worry about is the type of learning model and how to generate (and eliminate hypotheses)
- In this class, we will focus on these types:
 - Linear separators (**Perceptron**)
 - Extension of linear to non-linear (**SVM, ANN, etc**)
 - Tree-based classifiers (**Starting with decision trees**)

Linearly Separable Data

- Suppose you have following data:
(This is a toy example ☺)

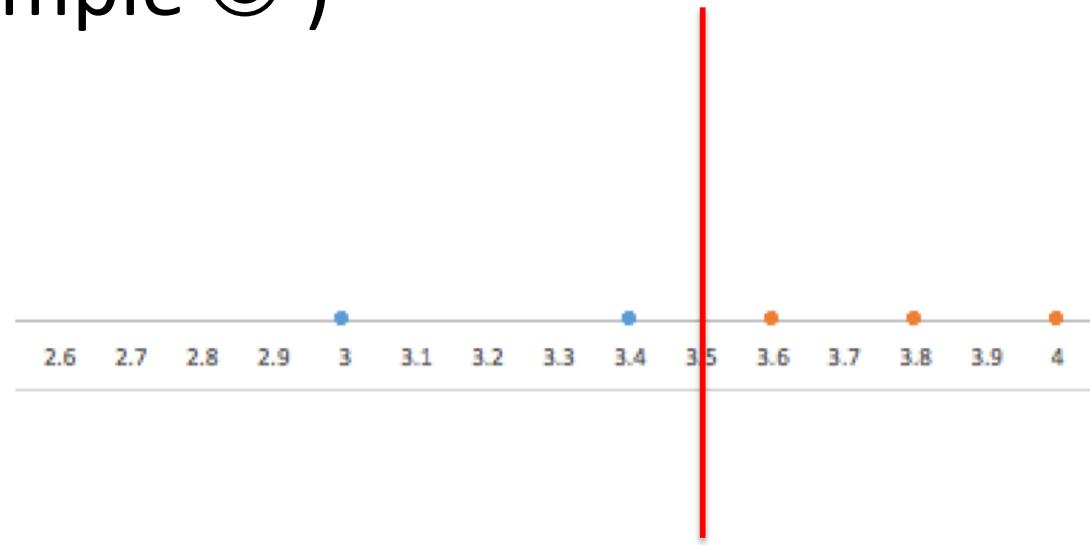
GPA	Internship
4.0	1
3.0	0
3.4	0
3.6	1
3.8	1
2.5	0

What can you
infer?

Linearly Separable Data

- Suppose you have following data:
(This is a toy example ☺)

GPA	Internship
4.0	1
3.0	0
3.4	0
3.6	1
3.8	1
2.5	0

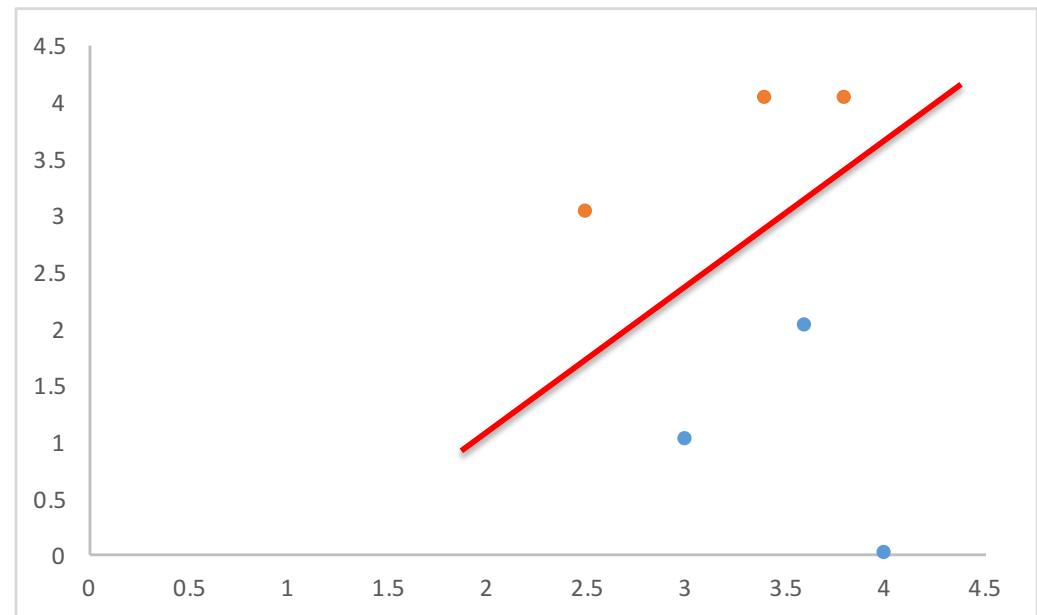


GPA=3.5 can
separate the
points.
Actually, it's one
of many possible
hypothesis.

Linearly Separable Data

- 2-D case

GPA	Years Exp	Internship
4.0	0	0
3.0	1	0
3.4	4	1
3.6	2	0
3.8	4	1
2.5	3	1



Perceptron

- It's a simple linear classifier -> straight line in 2-D and a plane in higher dimensions.
- Suppose each instance is represented by the vector x^j
Example of instance can be a student, a customer, etc. Attributes are x_1, x_2, \dots, x_d
- Each attribute can have different weights
 w_1, w_2, \dots, w_d

Perceptron

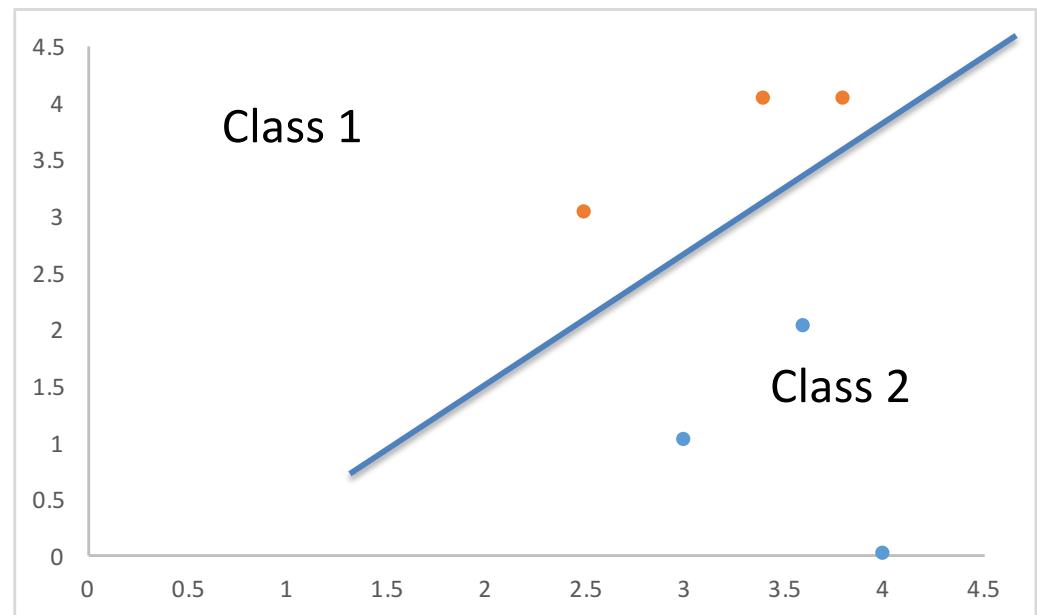
- Equation of separator is:

$$y = \sum_i w_i x_i$$

An easy way to separate classes is:

If $\sum_i w_i x_i > \text{threshold}$ assign class1

else assign class 2.



Perceptron

- An easier way to present this is:

This linear formula $h \in \mathcal{H}$ can be written as

$$h(\mathbf{x}) = \text{sign} \left(\left(\sum_{i=1}^d \mathbf{w}_i x_i \right) - \text{threshold} \right)$$

In vector form, the perceptron implements

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x})$$

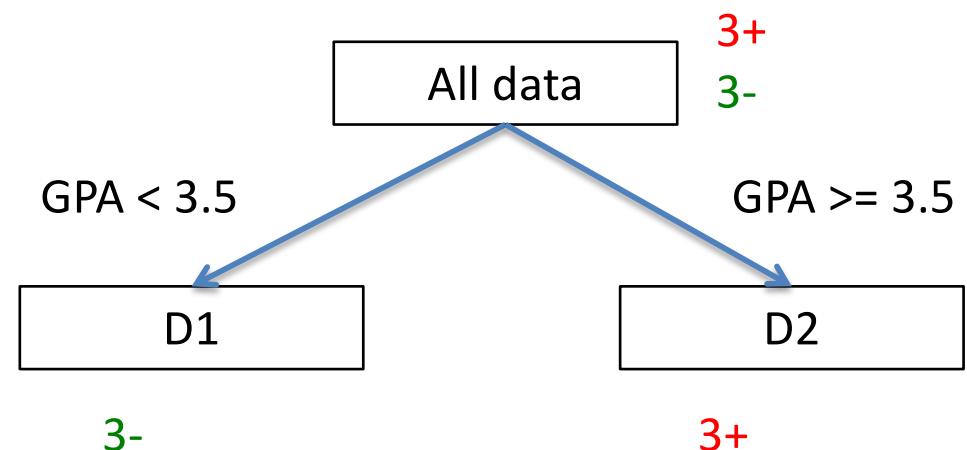
Learning

- Let us look at another way of classification.
- Decision Trees.

Decision Tree

- Toy example again

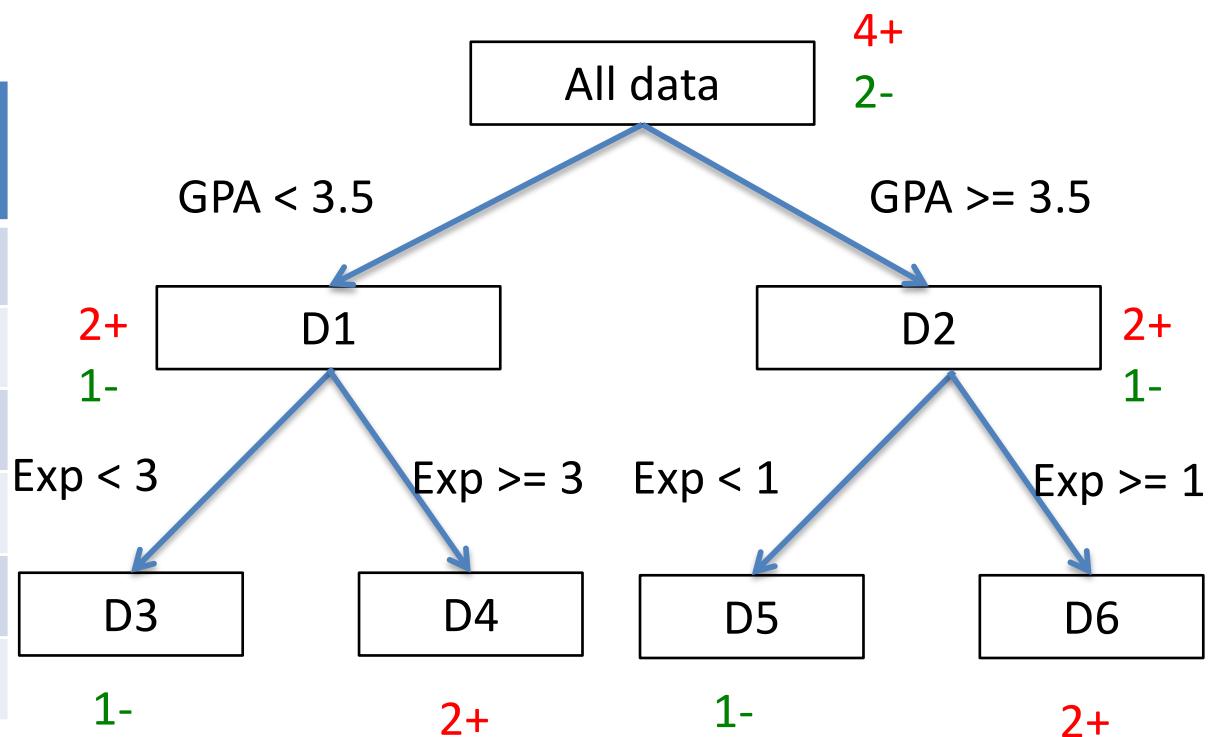
GPA	Internship
4.0	1
3.0	0
3.4	0
3.6	1
3.8	1
2.5	0



Decision Tree

- 2-D case

GPA	Years Exp	Internship
4.0	0	0
3.0	1	0
3.4	4	1
3.6	2	1
3.8	4	1
2.5	3	1



Decision Tree

- What can you infer?
- Think about decision trees as a way to learn classification function.
- Can also be thought of as rules:
eg: $X_1 \wedge X_2 \rightarrow 0$
where x_1 is boolean $GPA < 3.5$
and x_2 is boolean $exp < 3$
- OK...but which attribute should I choose to be at the top i.e. how do I choose attributes??

Sounds great, but ...

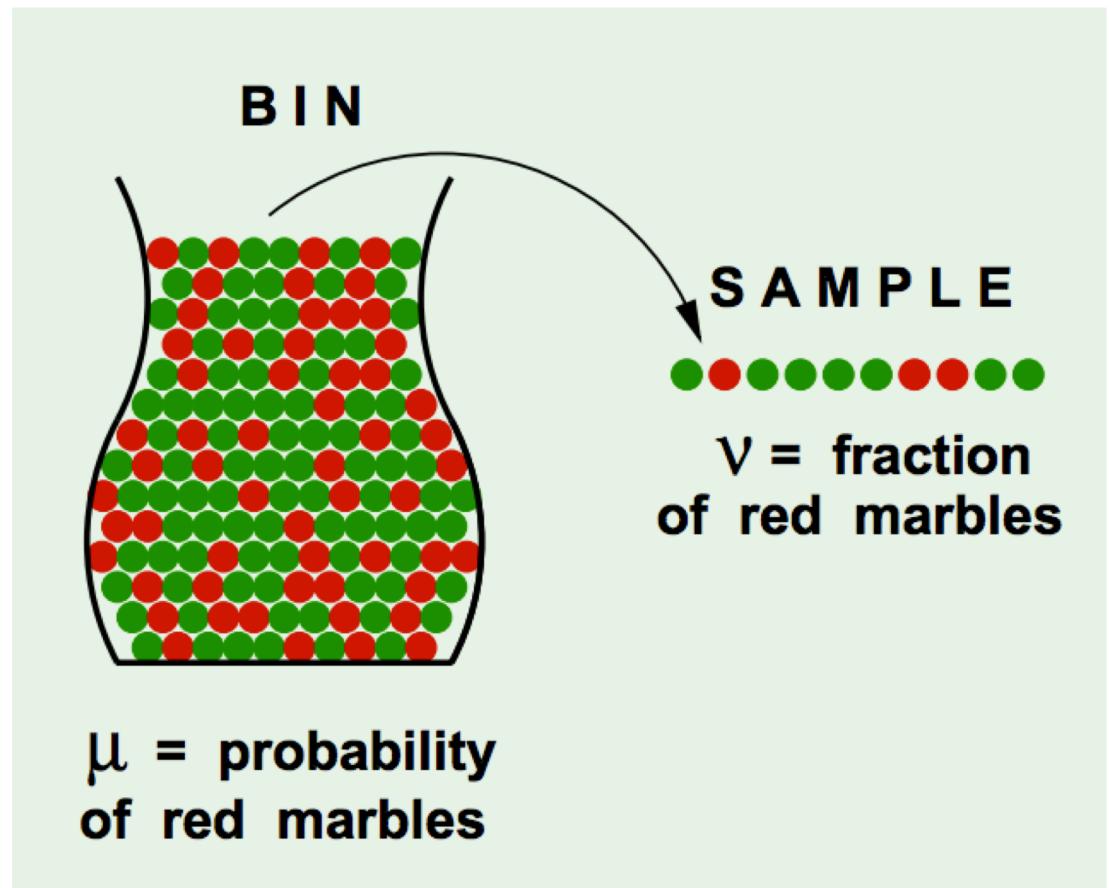
- Can you make some guarantee about the the unknown data / parameters?

Let's do an experiment.

You have a bin and want to estimate the probability of red marbles (μ).

You draw N samples and observe fraction of red marbles = v

Is there any relation between μ , v , and N ?



Hoeffding's Inequality

- It turns out there is a relation:

$$\mathbb{P} [|\nu - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N}$$

For large values of N , the probability of large error (difference) between ν and μ is bounded.

Think:

What happens when N becomes large, and when N is small?

Connection to learning

Bin: The unknown is a number μ

Learning: The unknown is a function $f : \mathcal{X} \rightarrow \mathcal{Y}$

Each marble ● is a point $\mathbf{x} \in \mathcal{X}$

- : Hypothesis got it **right** $h(\mathbf{x})=f(\mathbf{x})$
- : Hypothesis got it **wrong** $h(\mathbf{x})\neq f(\mathbf{x})$

