

# Machine Learning

## **Introduction**

Dan Goldwasser

[dgoldwas@purdue.edu](mailto:dgoldwas@purdue.edu)

Goal for Today's class

# ***Machine Learning?***

***What is it?***

***why should I care about it?***

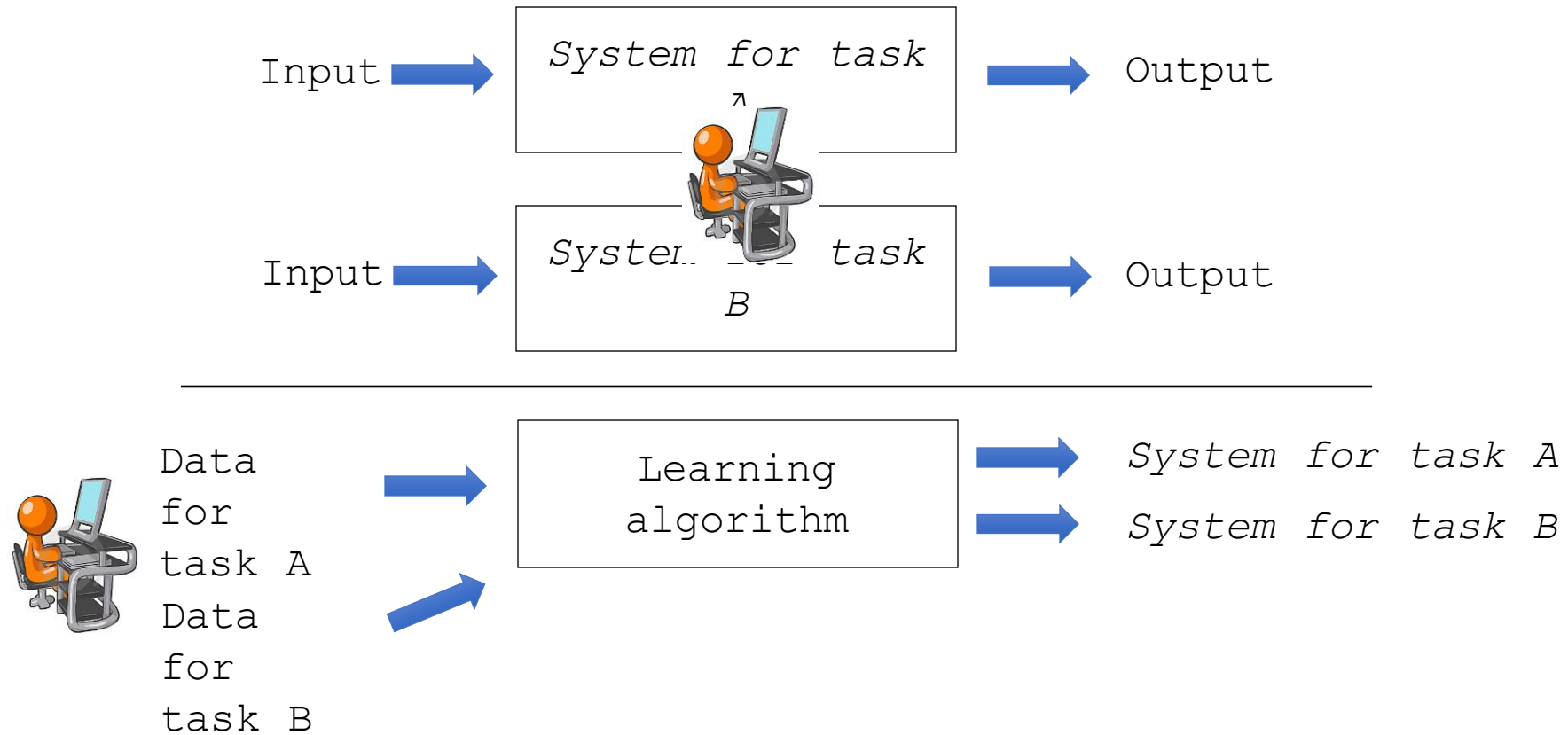
***Also – grading, policies, etc.***

# Machine Learning in a nutshell

**A new paradigm for telling a computer what to do!**

- **Traditionally:**
  - Write lines of code for different tasks
- **Machine learning**
  - Write code once (learning algorithm)
  - Supply data for different tasks
- Key issue: **Generalization**
- *How to generalize from training examples to new data*
  - Identify patterns in the data and abstract from training examples to testing examples

# Machine Learning in a nutshell



# Let's travel back in time.. 90's!

- The year was 1994, the conference was CoLT
  - CoLT = Computational Learning Theory
- Participants got a badge with a +/- label
- Only organizer knew the function generating the labels
- Label is determined only by the participant's name
- **Task:** *look at many examples as you want in the conference, and induce the unknown function*
- A great ice breaker for parties!

# The badges game

Can you figure out the function?

Name	Label
Claire Cardie	-
Peter Bartlett	+
Eric Baum	
Haym Hirsh	
Shai Ben-David	
Michael I. Jordan	

*Do you need more examples?*

# The badges game

Can you figure out the function?

Name	Label
Claire Cardie	-
Peter Bartlett	+
Eric Baum	-
Haym Hirsh	+
Shai Ben-David	-
Michael I. Jordan	+

# Let's travel back in time.. 90's!

- How did you do it?
  - Not a trivial process, even for humans.
  - Requires raising hypotheses, validating those on data, accepting or rejecting them.
- Can you find an algorithm to do the same process?
  - A form of search..
  - *What are you searching over?*
  - *How would you bias the search?*
  - *When would you stop?*

***This is machine learning!***



# What is machine learning?

*“Field of study that gives computers the ability to learn without being explicitly programmed”*

**Arthur Samuel (1959)**

“A computer program is said to learn from **experience E** with respect to some **class of tasks T** and **performance measure P**, if its performance at tasks in T, as measured by P, **improves with experience E.**”

**Tom Mitchell (1999)**

# Why Study Machine Learning?

- Basically :

**if it works well, I can probably use it..**

- Push the limits of existing systems
  - Flexible, personalized computing
- Systems that can react correctly to previously unseen situations
  - Different paradigm for programming and testing
- Make sense of messy data
  - Traditional methods (coding) cannot handle it

# It's everywhere!



Google  
Translate

Break through language barriers



Customers who viewed this item also viewed these products



Dualit Food XL1500  
Processor  
\$560

Add to cart



Kenwood kMix Manual  
Espresso Machine  
★★★★★  
\$250

Select options



Weber One Touch Gold  
Premium Charcoal  
Grill-57cm  
\$225

Add to cart



NoMU Salt Pepper and  
Spice Grinders  
\$3

View options



# It's everywhere!

- Spam detection
- Face recognition
- Content recommendation (e.g., movies, books)
- Stock market prediction
- Handwriting recognition
- Speech recognition
- Weather prediction
- Translation
- Automated Personal Assistants
- Self-driving car
- Disease diagnostic

# In this class..

- We will look into several **learning protocols**, using different **learning algorithms**, for learning **different models**.
  - **Model**: function mapping inputs to outputs
  - **Algorithm**: used for constructing a model, based on data
  - **Protocol**: the settings in which the algorithm learns.
- **Not an introduction to ML tools!**
  - Understand the algorithms, their properties and theory behind them.

# Learning Model

- We think about learning as producing a function mapping input to outputs, based on data
  - E.g., **spam**: email → Boolean
- **Model** is the type of function the learner looks at
  - Linear Classifier
  - Decision Trees
  - Non Linear Classifiers
  - Ensembles of classifiers
- All of these models can be characterized and discussed formally in terms of their expressivity.

# Learning Model Representation

- Learning: producing a function mapping input to outputs, based on data
  - E.g., spam: email → Boolean
- What is the domain and range of that function?
  - Output space defines the learning task
    - Binary, multiclass, continuous, structure, ..
  - What is the function's **domain**? How is the data represented?

# Learning Model Representation

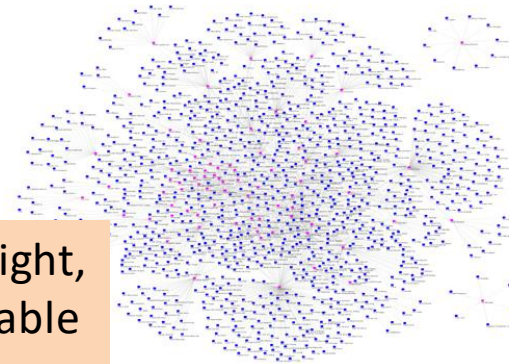
- What is the function's **domain**? **How is the data represented?**

(4.2, 3.5, ..., 1.3)



(1,0,0, 2,..0)

I have never had a worst flight,  
The seats were uncomfortable  
and it was 3 hours late!



0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0
0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0
0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0

Can we use data to figure out what is the best input representation?



# Learning Protocols

- **Supervised** learning
  - Human (teacher) supplies a labeled examples
  - Learner has to learn a model using this data
- **Unsupervised** learning
  - No teacher, learner has only unlabeled examples
  - Data mining: finding patterns in unlabeled data
- **Semi-supervised** learning
  - Learner has access to both labeled and unlabeled examples
- **Active** learning
  - Learner and teacher interact
  - Learner can ask questions
- **Reinforcement** learning
  - Learner learns by interacting with the environment
  - Why is it different/similar to Supervised learning? Active learning?

**Also:** non-traditional forms of interaction, Interactive learning, learning from explanations, advice, etc.

# Learning Algorithms

- **Learning Algorithms** generate a model, they work under the settings of a specific protocol
- **Supervised vs. Semi-Supervised vs. Unsupervised**
- **Online vs. Batch**
  - Online algorithm: learning is done one example at a time
    - Winnow, Perceptron,...
  - Batch algorithm: learning is done over entire dataset
    - SVM, Logistic Regression, Decision Trees, ...
- How can we compare learning algorithms?

# Learning Theory

- ***Will my algorithm work?***
  - Train and then check by testing it
  - What could be a potential problem with this approach?
- Can we reason in a more rigorous, general way?
  - Mathematical definition of learning, guarantees of learning algorithms, and required resources
- Online learning
- PAC

# Machine Learning Tasks

- Supervised Classification
  - Most popular scenario
  - Most of this class is about classification
- Regression
- Clustering (unsupervised)
- Structured Prediction
- Reinforcement learning

# Classification

- **Classification:** mapping data into categories
  - Write a face recognition program
  - Determine if an English sentence is grammatical
  - Distinguish between normal and cancerous cells
- Can't we just write code?
- Provide labeled examples and let a classifier distinguish between the two classes
  - What are the labeled examples in each case?

# Classification dataset

- Three components

- Class label (denoted “y”)
  - Binary, multiclass
- Features (outlook, temperature, windy)
- Feature values (denoted “x”)
  - Can be binary, continuous, categories (Sunny, Rainy,...)

Class	Outlook	Temperature	Windy?
Play	Sunny	Low	Yes
No play	Sunny	High	Yes
No play	Sunny	High	No
Play	Overcast	Low	Yes
Play	Overcast	High	No
Play	Overcast	Low	No
No play	Rainy	Low	Yes
Play	Rainy	Low	No

# Classification Dataset

Class	Outlook	Temperature	Windy?
Play	Sunny	Low	Yes
No play	Sunny	High	Yes
No play	Sunny	High	No
Play	Overcast	Low	Yes
Play	Overcast	High	No
Play	Overcast	Low	No
No play	Rainy	Low	Yes
Play	Rainy	Low	No

- A labeled dataset is a collection of (x,y) pairs
- **Task:** use dataset to predict new instance class

Class	Outlook	Temperature	Windy?
???	Sunny	Low	No

- Generalization: can we make reliable predictions?

# Recipe for building a classifier

- **Step 1: Collect Training Data**

- Can be a bottleneck when labeling is difficult, but in some situations “for free”!
  - Car detector VS. Cancerous cell identification

- **Step 2: Decide Feature representation**

- The classification function domain, captures properties of the input object
  - Trivial decisions in some cases, difficult in others (which?)
- “Art more than science”
- Difficult to learn over bad feature representations
  - What is a bad representation for the weather domain?
  - .. And in general?
- How can we be “more science than art?”



# Recipe for building a classifier

- **Step 3: Model (representation):**

- What “family” of functions to use?
  - Boolean functions, Linear functions (hyperplanes)
- Known as the hypothesis space
- **Terminology:** Classifier = Function = Hypothesis

- **Step 4: Optimization (learning algorithm):**

- Which function to chose?
  - Search over possible functions (i.e., hypothesis space)
- The learning algorithm uses the training data to choose a specific function
  - **Bias the search** (no free lunches, inductive bias, regularization..)
  - **Evaluate** candidate hypothesis (based on...?)

# Clustering

- Clustering: Group *similar* instances.
  - Define a similarity metric
- **Questions:**
  - How can you tell how many groups are in the data?
  - What is the "meaning" of the formed clusters?



**Is clustering the same as classification?**

# Structured Prediction

- Mapping Between **complex objects (structures)**
  - Translation, Protein sequence
- Reduction to classification
  - Multiple **interdependent** decisions
    - We need to model the interactions between decisions!

**Named Entity Recognition:** *decide if a word is a named location, person, organization or neither.*

**Simple approach:**

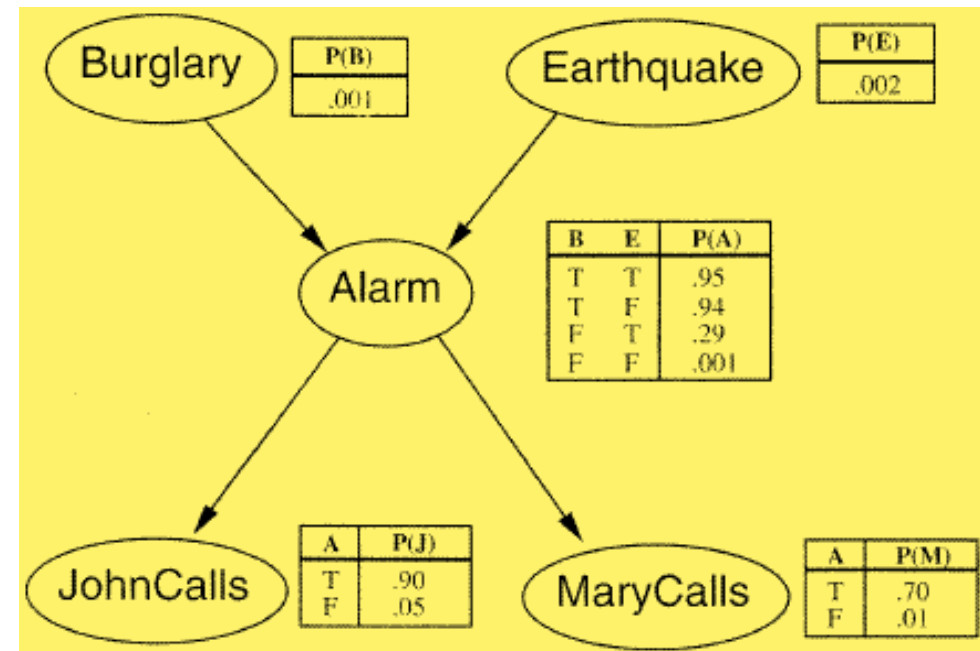
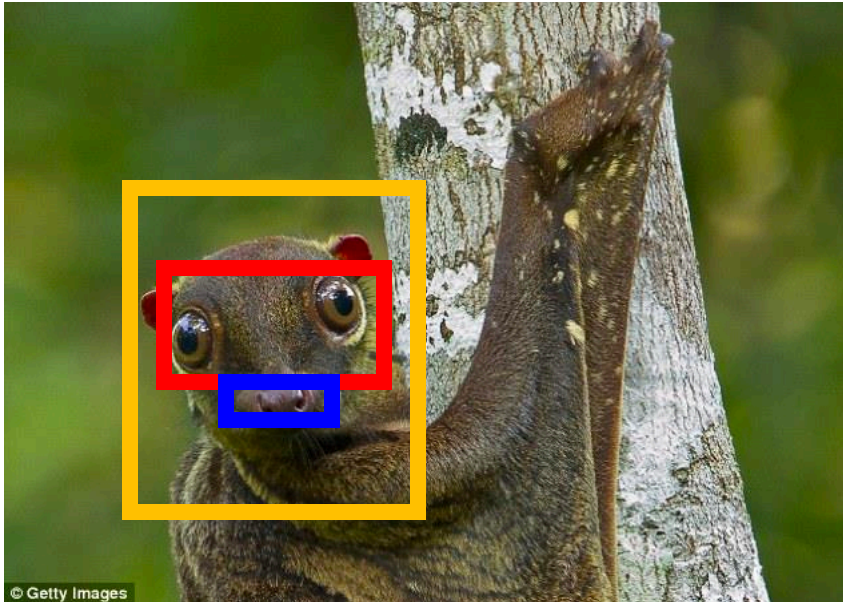
*Each word defines  
a classification problem.*

*Is that the right way to go?*

**B-ORG    I-ORG    I-ORG    O    B-LOC**

**Mount Sinai Hospital in NYC**

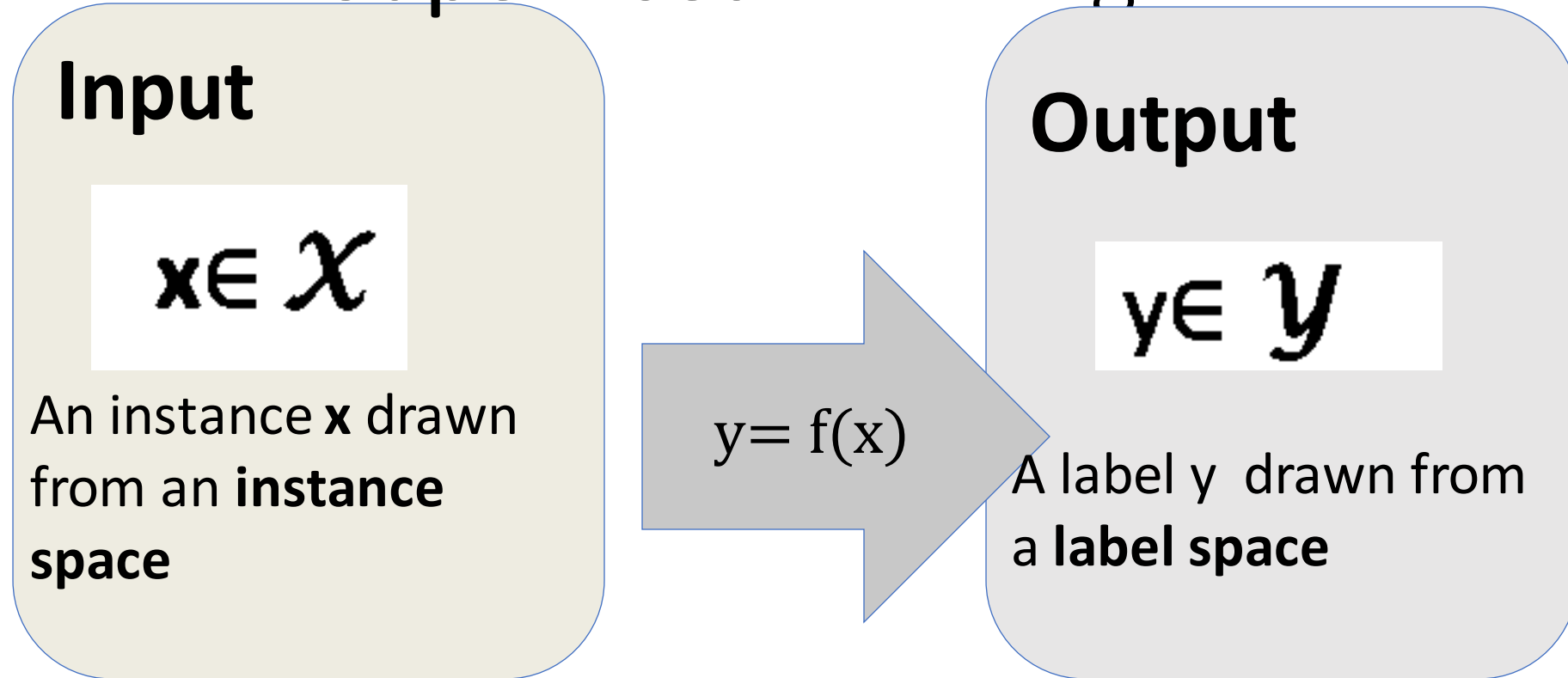
# Structured Prediction



# Our focus: Supervised Learning

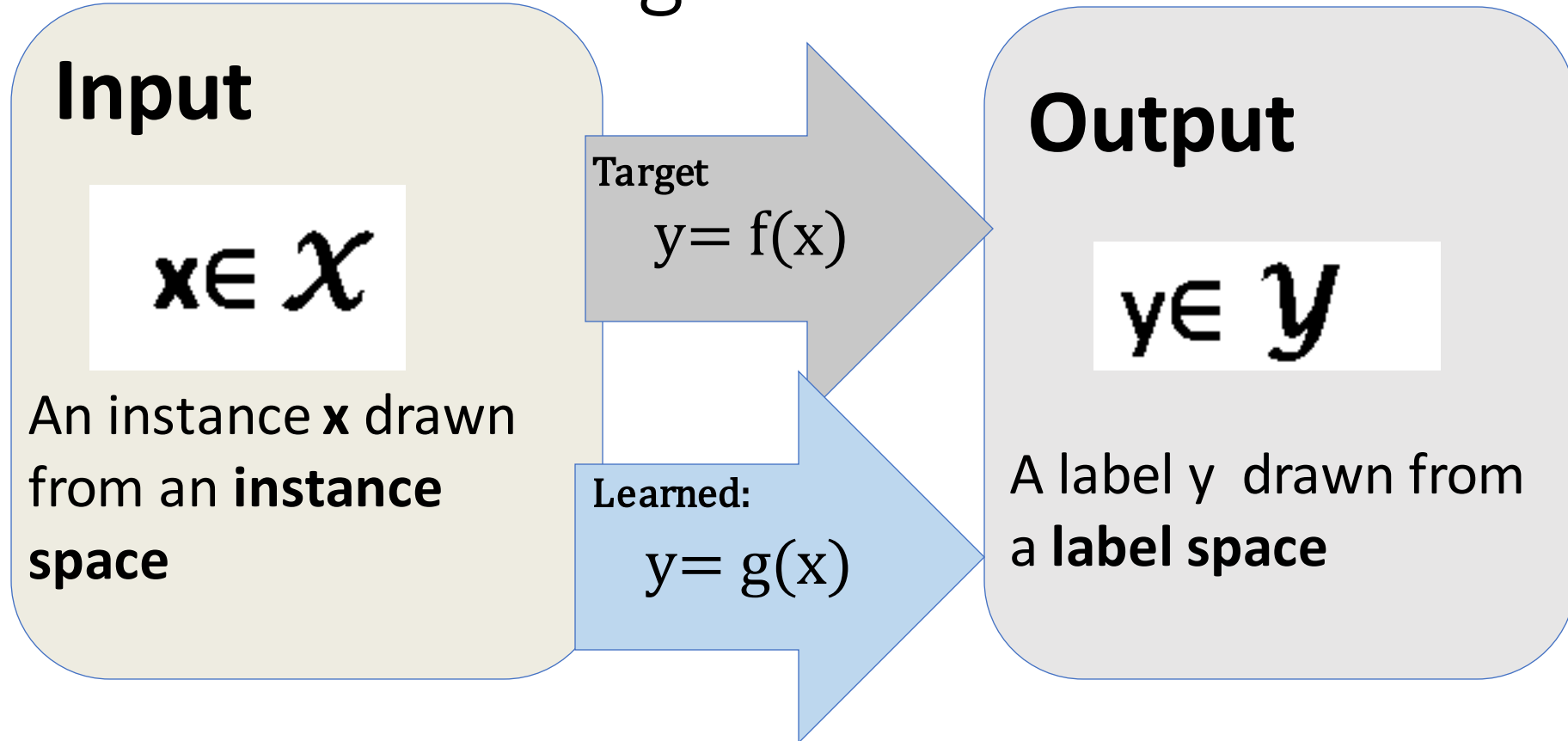
- Recall the “**Badge Game**”
  - Each member gets a badge with a +/- label and a name
- **Assumptions:** *the label is a function of the name*
  - Strong assumption! *Meaning that future data will behave in the same way!*
- **Our goal:** approximate the true function using the available data.
  - Learning is about “removing uncertainty” about what is a good approximation.
- ***Let’s formalize these concepts!***

# Our Focus: Supervised Learning



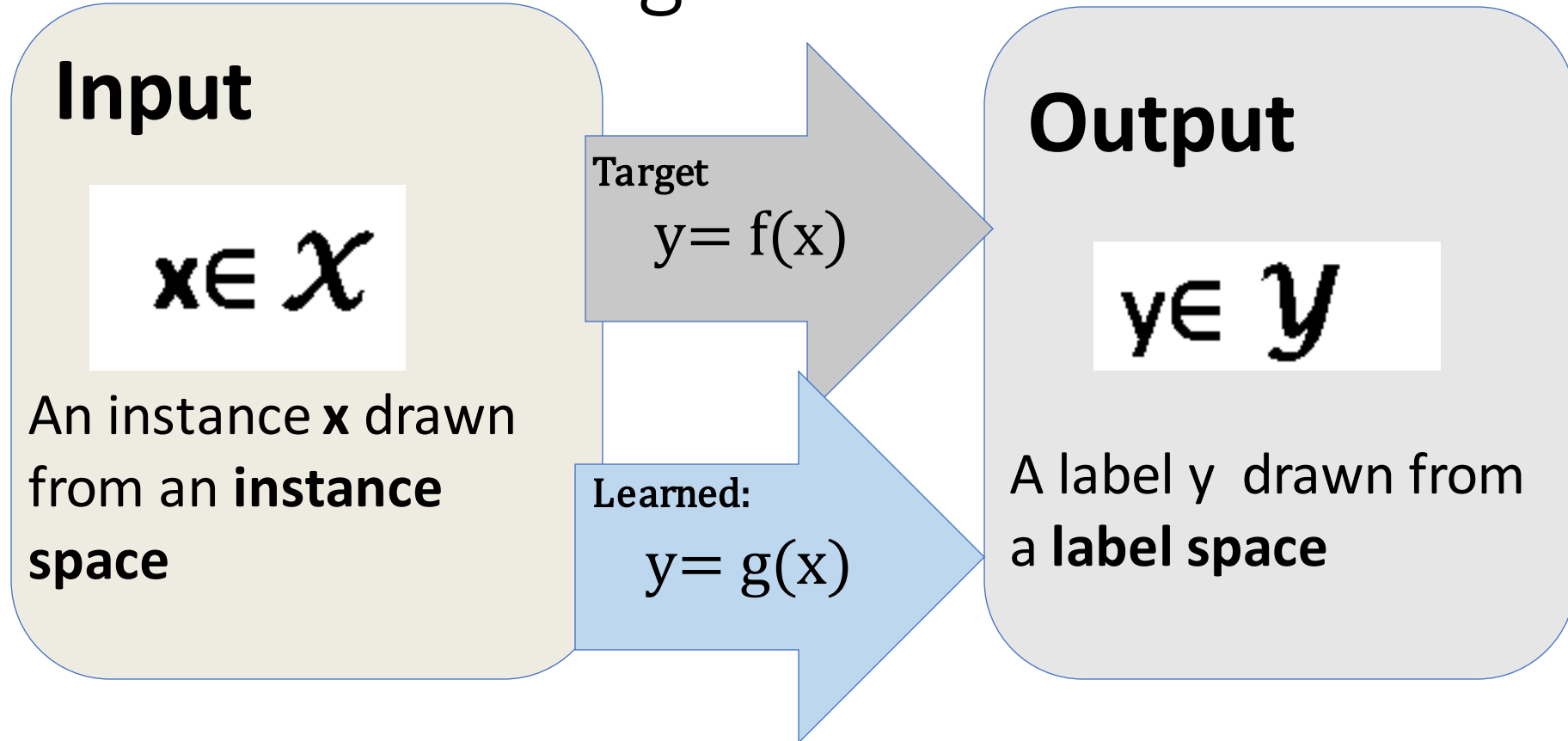
- In the supervised settings we want to find  $f(x)$  based on examples

# Supervised Learning



- The learning algorithm sees a small subset of the instance space, with their labels


# Supervised Learning



- The learned function may not be identical to the target function: for a given  $x$ ,  $f(x) \stackrel{?}{=} g(x)$



# Supervised Learning Settings

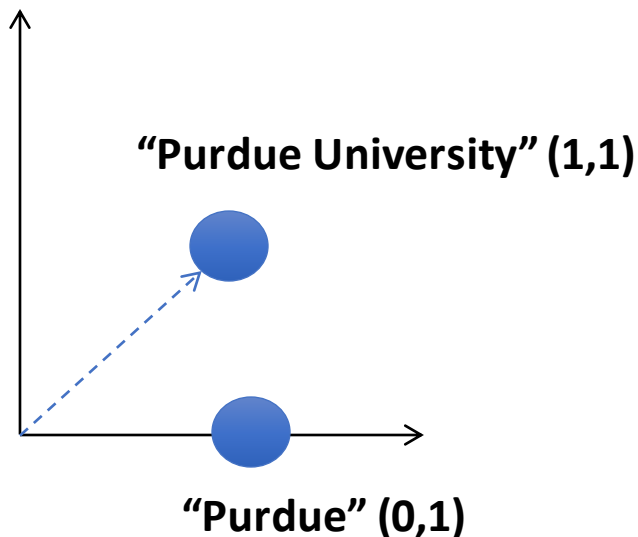
- **Input:** a set of examples  $\langle \mathbf{x}, f(\mathbf{x}) \rangle$ , where  $f(\mathbf{x})$  is the unknown target function
  - $f(\mathbf{x})$  can be binary, categorical, or continuous
- The input  $\mathbf{x}$  is represented in a ***feature space***
  - Typically  $\mathbf{x} \in \{0,1\}^n$ , or  $\mathbf{x} \in \{0,1,\dots,K\}^n$ , or  $\mathbf{x} \in \mathbb{R}^n$
- **Learning:** find  $g(\mathbf{x})$  that approximates  $f(\mathbf{x})$  well.
  - The space from which  $g(\mathbf{x})$  is chosen is called ***hypothesis space***
  - *Learning is searching this space for a “good”  $g(\mathbf{x})$*
- **Question:** *does the hypothesis space of  $f(\mathbf{x})$  and  $g(\mathbf{x})$  has to be similar?* 

# The Instance Space

- ***How you choose to represent examples matters!***
  - Good representations should capture aspect of the input object relevant for classification
- ***What is a good representation for the badge game?***
  - Boolean features: “2<sup>nd</sup> letter is a vowel”
  - Numerical: “how many vowels?”
- This is a domain specific process
  - **Know your problem domain!**

# The Instance Space

- Feature functions map inputs to Boolean/numeric values
- The input objects are represented using a feature vector
  - $X$  is a  $N$ -dimensional vector space
  - Each dimension is one feature
  - Each instance  $\mathbf{x}$  is a point in the vector space



It's convenient to think about templates:

***"what's the 2<sup>nd</sup> character in the name?"***

John →  $\langle 0, 1, 0, \dots \rangle$

James →  $\langle 1, 0, 0, \dots \rangle$

Claire →  $\langle 0, 0, 1, \dots \rangle$

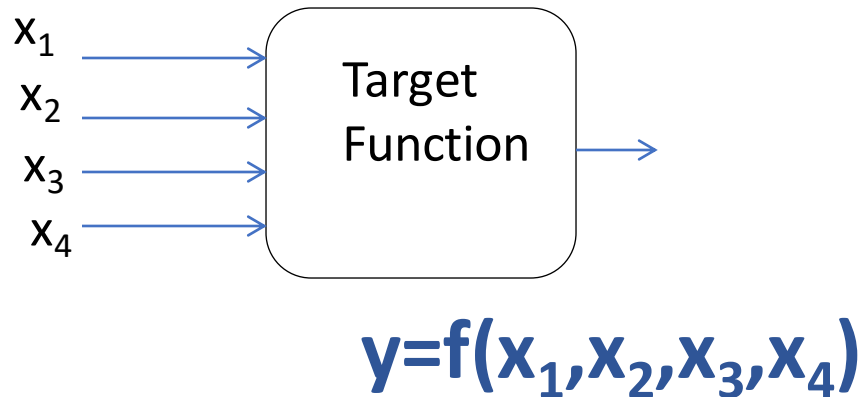
# The Hypothesis Space

- What functions should the learning algorithm consider?
  - **Does not** have to search over the same space as  $f(x)$
  - Should it be a **more** expressive space? **Less** expressive?
- Does it matter?

# The Hypothesis Space

We want to find the target functions based on the training examples

**Can you find the target function?**



$x_1$	$x_2$	$x_3$	$x_4$	$y$
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

# Is learning possible?

- How many Boolean functions are there over 4 inputs?
- $2^{16} = 65536$  functions (**why?**)
  - 16 possible outputs.
  - Two possibilities for each output
- Without any data,  $2^{16}$  options
- *Does the data identify the right function?*
- The training data contains 7 examples
  - We still have  $2^9$  options,

***Is learning even possible?***

$x_1$	$x_2$	$x_3$	$x_4$	$y$
0	0	0	0	?
0	0	0	1	?
0	0	1	0	<b>0</b> ←
0	0	1	1	<b>1</b> ←
0	1	0	0	<b>0</b> ←
0	1	0	1	<b>0</b> ←
0	1	1	0	<b>0</b> ←
0	1	1	1	?
1	0	0	0	?
1	0	0	1	<b>1</b> ←
1	0	1	0	?
1	0	1	1	?
1	1	0	0	<b>0</b> ←
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

# Hypothesis Space

- A ***hypothesis space*** is the set of possible functions we consider
- **The Problem:** *we were looking at the space of **all** Boolean functions*
- **The Solution:** *Instead choose a hypothesis space that is smaller than the space of all Boolean functions:*
  - *Only simple conjunctions (with four variables, there are only 16 conjunctions without negations)*
  - *Simple disjunctions*
  - *m-of-n rules: Fix a set of n variables. At least m of them must be true*
  - *Linear functions*

# Let's try a different Hypothesis space!

- Simple Conjunctions: *very small subset of Boolean functions*
  - Only 16 possible conjunction of the form:  $y = x_i \wedge \dots x_j$

- *Why?*

- *Can you find a consistent Hypothesis in this space?*

$x_1$	$x_2$	$x_3$	$x_4$	$y$
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0



# Simple Conjunctions

- Simple Conjunctions

Rule	Counterexample
$y=c$	
$x_1$	1100 0
$x_2$	0100 0
$x_3$	0110 0
$x_4$	0101 1
$x_1 \wedge x_2$	1100 0
$x_1 \wedge x_3$	0011 1
$x_1 \wedge x_4$	0011 1

$x_1$	$x_2$	$x_3$	$x_4$	$y$
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

*.. let's keep going..*

# Simple Conjunctions

- Simple Conjunctions

Rule	Counterexample
$y=c$	
$x_1$	1100 0
$x_2$	0100 0
$x_3$	0110 0
$x_4$	0101 1
$x_1 \wedge x_2$	1100 0
$x_1 \wedge x_3$	0011 1
$x_1 \wedge x_4$	0011 1

Rule	Counterexample
$x_2 \wedge x_3$	0011 1
$x_2 \wedge x_4$	0011 1
$x_3 \wedge x_4$	1001 1
$x_1 \wedge x_2 \wedge x_3$	0011 1
$x_1 \wedge x_2 \wedge x_4$	0011 1
$x_1 \wedge x_3 \wedge x_4$	0011 1
$x_2 \wedge x_3 \wedge x_4$	0011 1
$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	0011 1

***No simple conjunction can explain this data!***

# Let's try a different Hypothesis space!

- **The class of simple conjunctions is not expressive enough for our functions**
- **How can we pick a better space?**
  - Prior knowledge about the problem
  - Sufficiently “flexible”
- Let's try another space – *m-of-n rules*
  - Rules of the form: *“ $y = 1$  if and only if at least  $m$  of the following  $n$  variables are 1”*
    - How many are there for 4 Boolean variables?
    - **Is there a consistent hypothesis?**

# M-of-N rules

- m-of-n rules

- Examples:

- 1 out of  $\{x_1\}$
- 2 out of  $\{x_1, x_3\}$
- ...

- **Is there a consistent hypothesis?**

- Check!

- For example, let's check: “2 out of  $\{x_1, x_2, x_3, x_4\}$ ”

→ *Exactly one hypothesis is consistent with the data!*

$x_1$	$x_2$	$x_3$	$x_4$	$y$
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	0	0	0
0	1	0	1	0

# Learning

- Learning is removal of remaining uncertainty
  - *If we know that the function is a “ $m$ -out-of- $n$ ”, data can help find a function from that class*
- **Finding a good hypothesis class is essential!**
  - You can start small, and enlarge it until you can find a hypothesis that fits the data

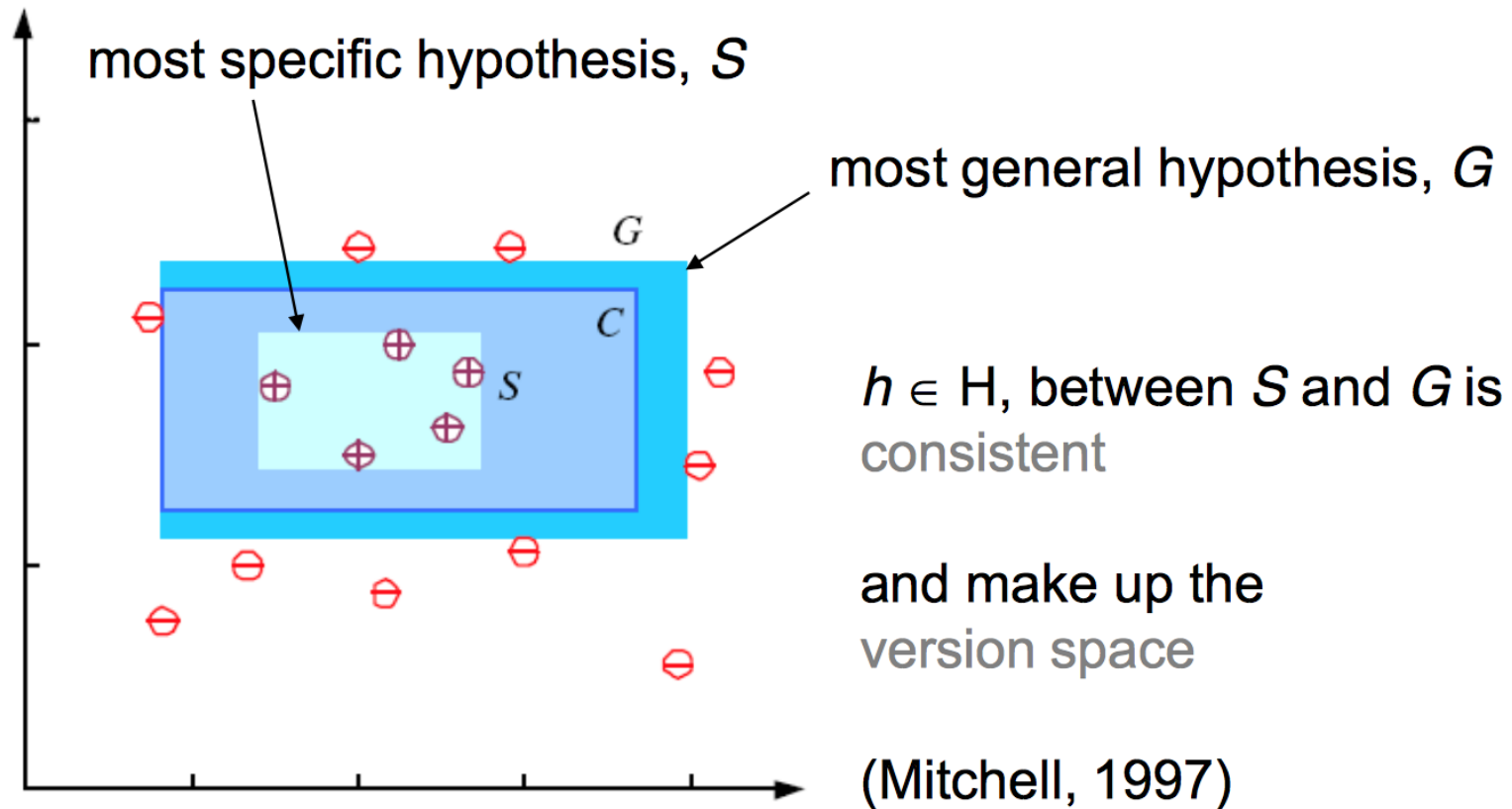
**Question:** *Can there be more than one function that is consistent with the data?*

**How do you choose between them?**

# Terminology

- **Classifier:** discrete-valued function, possible output are called classes.
- **Hypothesis space:** The space of all hypotheses that can be the output of a learning algorithm.
- **Version space:** The space of all hypotheses in the hypothesis space, consistent with the data

# Terminology



# My first Classifier!

- Let's take a look at one of the simplest classifiers
  - *Basically, just maintain the training data (**ideas?**)*
- Assume we have a training set  $(x_1, y_1) \dots (x_n, y_n)$ 
  - **Simplest solution:** given a new sample,  $x$ , find the most similar example in the training data, and predict the same value.
    - If you liked “*Fast and Furious*” you’ll like “*2 fast 2 furious*”
- Key decision: find a good distance metric

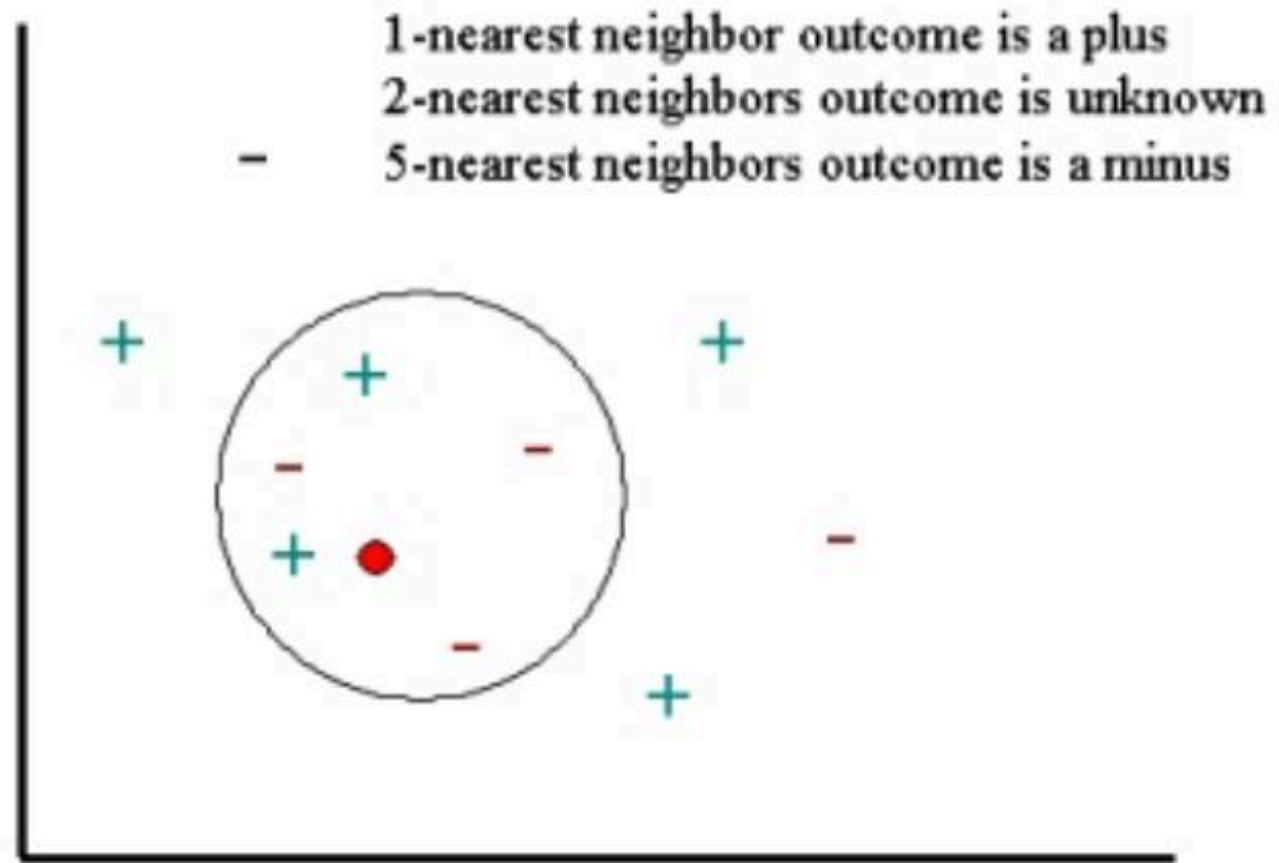
$$d(x_1, x_2) = 1 - \frac{x_1 \cap x_2}{x_1 \cup x_2}$$

$$d(x_1, x_2) = \sqrt{(x_1 - x_2)^2}$$

- ***Can we do better?***
- *We can make the decision by looking at several near examples, not just one. **Why would it be better?***



# KNN



# K Nearest Neighbors

- Learning: just storing the training examples
- Prediction:
  - Find the K training example closest to  $\mathbf{x}$ 
    - Classification: majority vote
    - Regression: mean value
- KNN is a type of *instance based learning*
  - *Learning is requires storing the training set*
  - *Prediction using similar stored examples*
- This is called *lazy* learning, since most of the computation is done at prediction time

# Let's analyze KNN

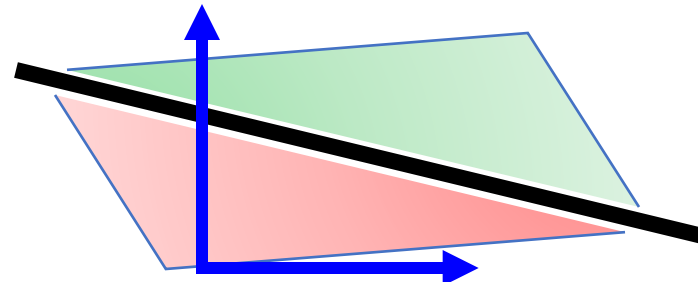
- **What are the advantages/disadvantages?**
  - *What are the important aspects when analyzing learning algorithms?*
- Complexity
  - *Space*
  - *Time (training and testing)*
- Expressivity
  - *What kind of functions can we learn?*

# Let's analyze KNN

- **Time Complexity**
  - *Training is very fast*
  - *Prediction is slower..*
    - *$O(dN)$  for  $N$  examples, of dimensionality  $d$*
    - *Increases with number of training examples!*
- **Space Complexity**
  - *KNN needs to maintain all training examples*
- *Let's Compare to the m-of-n rules*

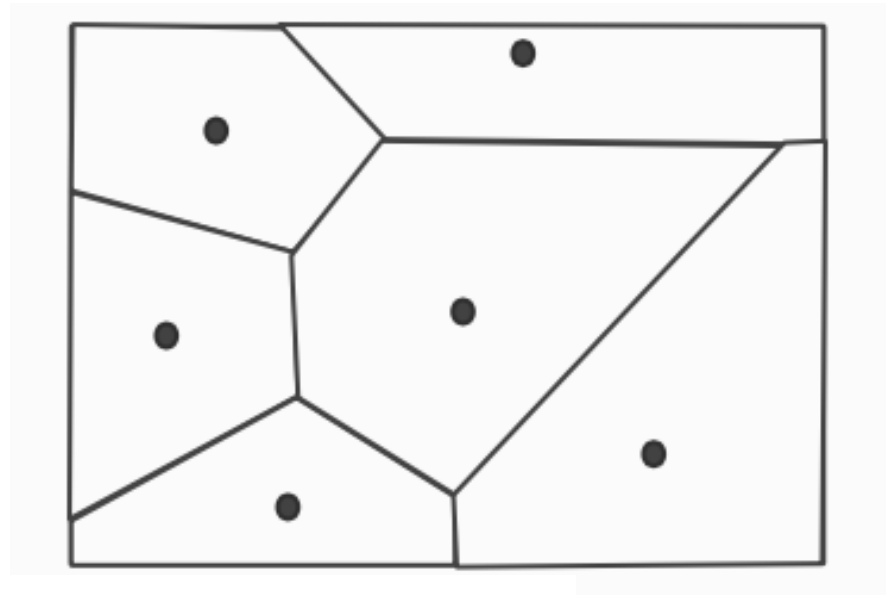
# Let's analyze KNN

- Expressivity
  - Can learn very complex decisions (*more on that later*)
    - Very sensitive to feature representation choice
    - Irrelevant attributes can fool the classifier
- KNN does not construct an explicit hypothesis
- We can try to characterize the learned function using its *decision boundary*
  - *Visualize which elements will be classified as positive/negative*
  - *Decision Boundary is the curve separating the negative and positive regions*



# Decision Boundary: Voronoi Diagram

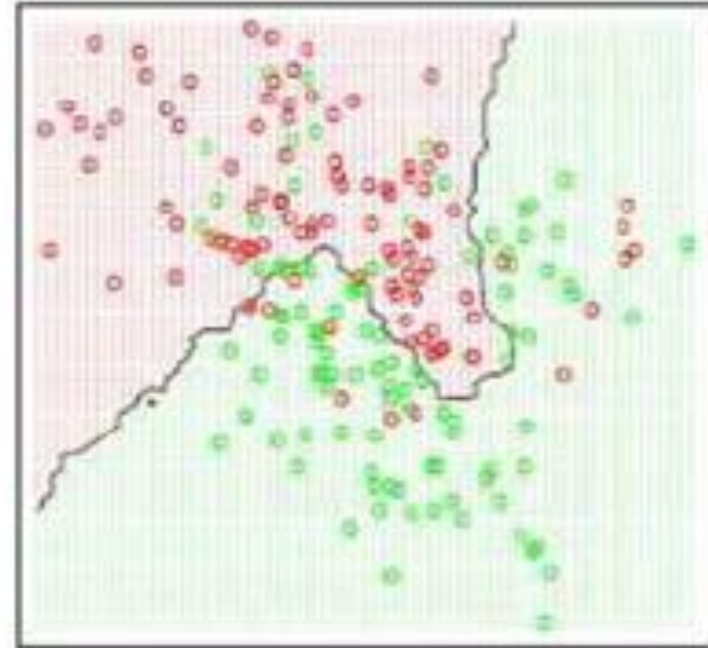
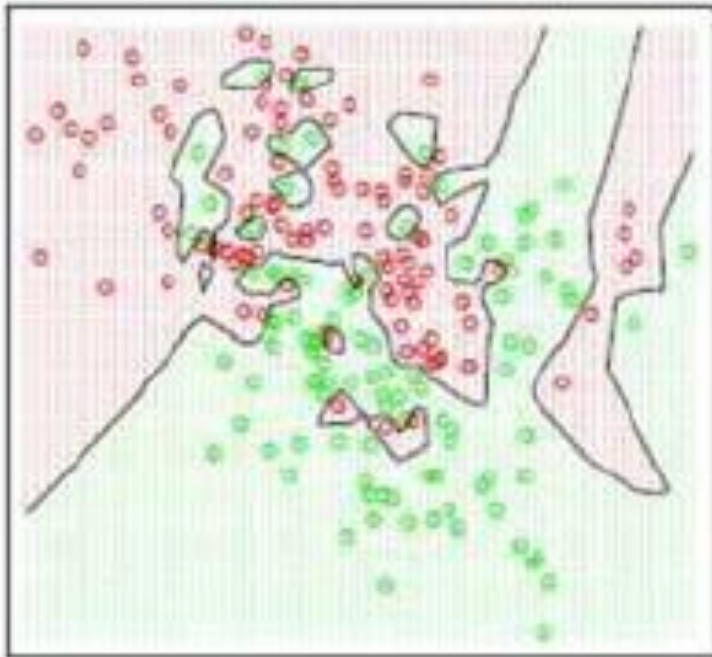
- Each point  $x$  in the training set defines a Voronoi cell
  - The polyhedron consisting of all the points closest to  $x$
- The Voronoi diagram is the union of all these cells
- For 1-NN (with Euclidean distance) this is the decision boundary



# KNN: choosing the right K

Let's take a closer look at the learned function

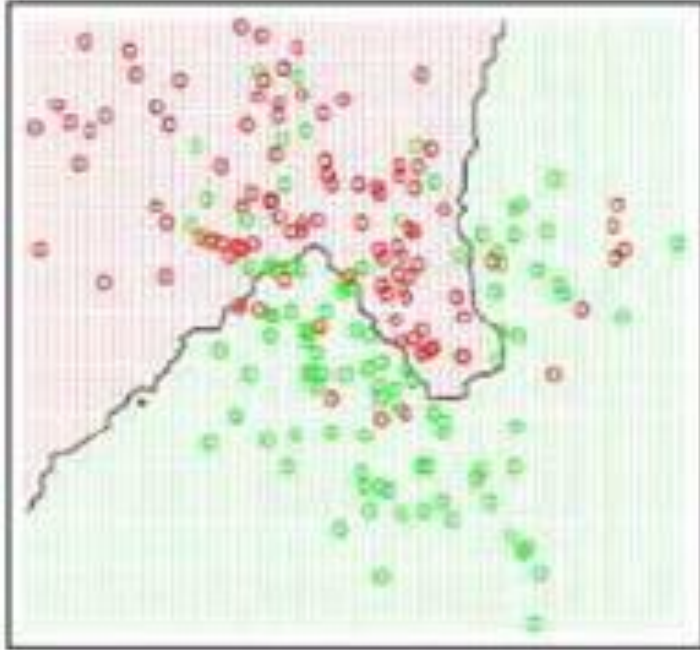
→ ***High sensitivity to noise!***



Higher k values results in smoother decision boundaries

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

# Question



Higher  $k$  values results in smoother decision boundaries

**What will happen if we keep increasing  $K$ ?**

**→ *Extreme scenario: if  $k = N$ ?***

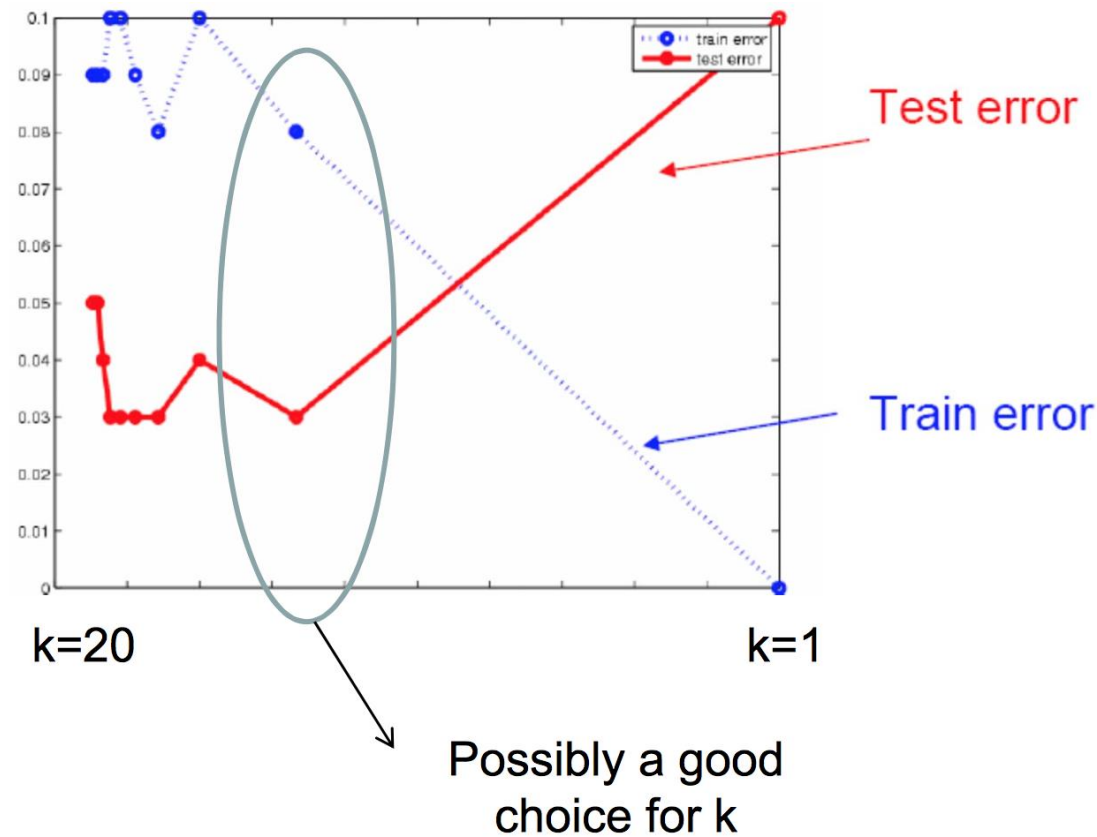


# How should we set the value of K?

- **Option 1**: Pick the K that minimizes the **training error**.
  - **Training Error**: number of errors on the training set, *AFTER* we learned a classifier
- **Question**: *What is the training error of 1-NN? 10-NN?*
- **Option 2**: choose k to minimize mistakes on **test error**
  - **Test Error**: number of errors on the test set, after we learned a classifier
  - **Note**: *It's better to use a validation set, and not the test set. More on that later in the class..*

# How should we set the value of K?

*How would the test and train error change with K?*



In general – using the training error to tune parameters will always result in a more complex hypothesis! **(why?)**

# KNN Practical Considerations

- Very simple to understand and implement
- An odd value for  $k$  is better (why?)
- How can we find the right value for  $k$ ?
  - Using a held out set
- Feature normalization is important! (why?)
  - Different scales for different features

# KNN Practical Considerations

- Choosing the right features is important
  - Very sensitive to irrelevant attributes
  - Sensitive to the number of dimensions
- Choosing the

– Euclidean distance

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_2 = \sqrt{\sum_{i=1}^n (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

– Manhattan distance

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_1 = \sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$

–  $L_p$ -norm

- Euclidean =  $L_2$
- Manhattan =  $L_1$
- Exercise: What is  $L_\infty$ ?

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_p = \left( \sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p \right)^{\frac{1}{p}}$$

# Review Questions

- What is a hypothesis space?
- What is inductive bias?
- When deciding how to design the hypothesis space, what decisions can we make?
- What are the advantages of a simpler, or more complex space?
- How does KNN work? how do we learn and make predictions?
- If the test performance of KNN is low, should we increase or decrease  $K$ ?

# Further reading

**Machine Learning.** Tom Mitchell.

Chapter 8

**A Course in Machine Learning.** Hal Daumé III.

Chapter 2.2 and 2.3

[http://ciml.info/dl/vo\\_9/ciml-vo\\_9-cho2.pdf](http://ciml.info/dl/vo_9/ciml-vo_9-cho2.pdf)

# Course Policies

- **When:** Tuesday, Thursday 12:00
- **Where:** Lawson, 1142
- **Office hours:** Thursday after class (email me)
- **TA:**
  - Tunaz Islam
  - Nikhil Mehta
  - Devulapalli Pramith
  - Tinghan Yang
- See Brightspace page for office hours (will be updated soon)

# Required Background

- **You need a math background**

- Algebra (describe high dimensional data)
- Calculus (finding min/max points of functions)
- Stats and Probability (diggin' in data)

- **It is a CS class**

- Algorithm design, complexity analysis, discrete math
  - Should sound familiar – recursion, dynamic programming, hashing function, graphs, trees, BFS, DFS

- **It's not a theory class**

- You need some programming experience



# My Expectations

- **Academic honesty**

- Submit your own work.
- Load in group project should be evenly distributed

- **Come prepared to class and participate**

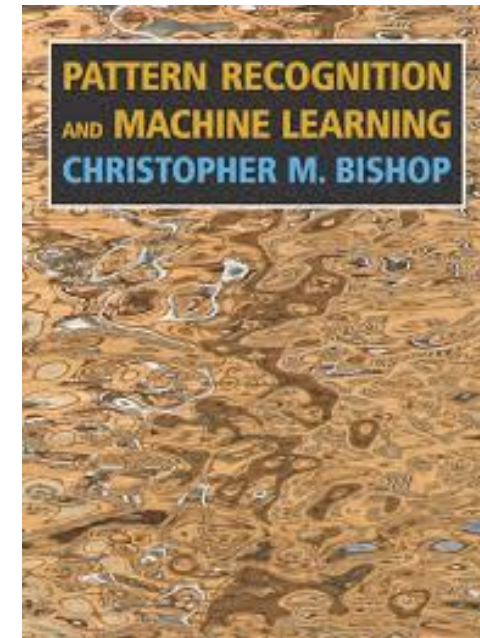
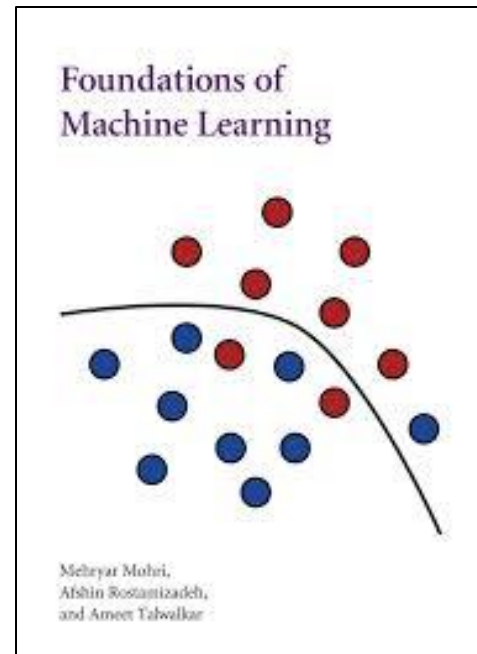
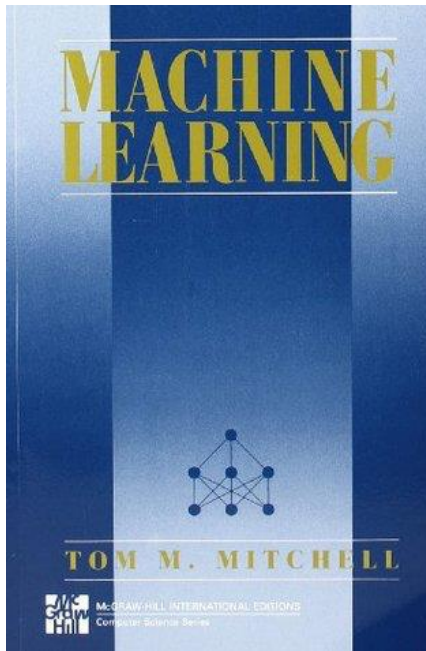
- Review previous lecture notes and other materials
- I encourage class discussions

- **Communicate**

- Inform teaching staff of any problems, or unclear topics
- Interact with your peers, solve problems together

# Textbooks

- No required textbook, several recommended books that I will point to



# Communication

- Most of our interaction will be through Brightspace
  - Dedicated discussion boards for each week
- Part of your grade is determined by your brightspace activity
  - Post questions and respond to questions
  - Express opinions, post related material
- **Communicate with teaching staff and come to office hours!**

# Grading

- Final exam: 35%
- Midterm: 20%
- Homework: 35%
- Attendance and Participation: 10%

# Homework

- Written and Machine problems
  - About 5 assignments
  - Machine problems in Python
    - Other language require TA approval
- Late Policy
  - You have 48 hours of late submission for each assignment.
  - 10% will be deducted for each 24 hours.

# Topics

- Introduction to ML
- Linear classifiers and online learning
- Linear Classifiers using GD and multiclass classification
- Learning theory (PAC)
- Ensemble methods
- Introduction Neural networks and representation learning
- Introduction to Structured prediction
- Application (if we don't run out of time!)