Introduction to Machine Learning

Nicholas Mattei, *Tulane University*

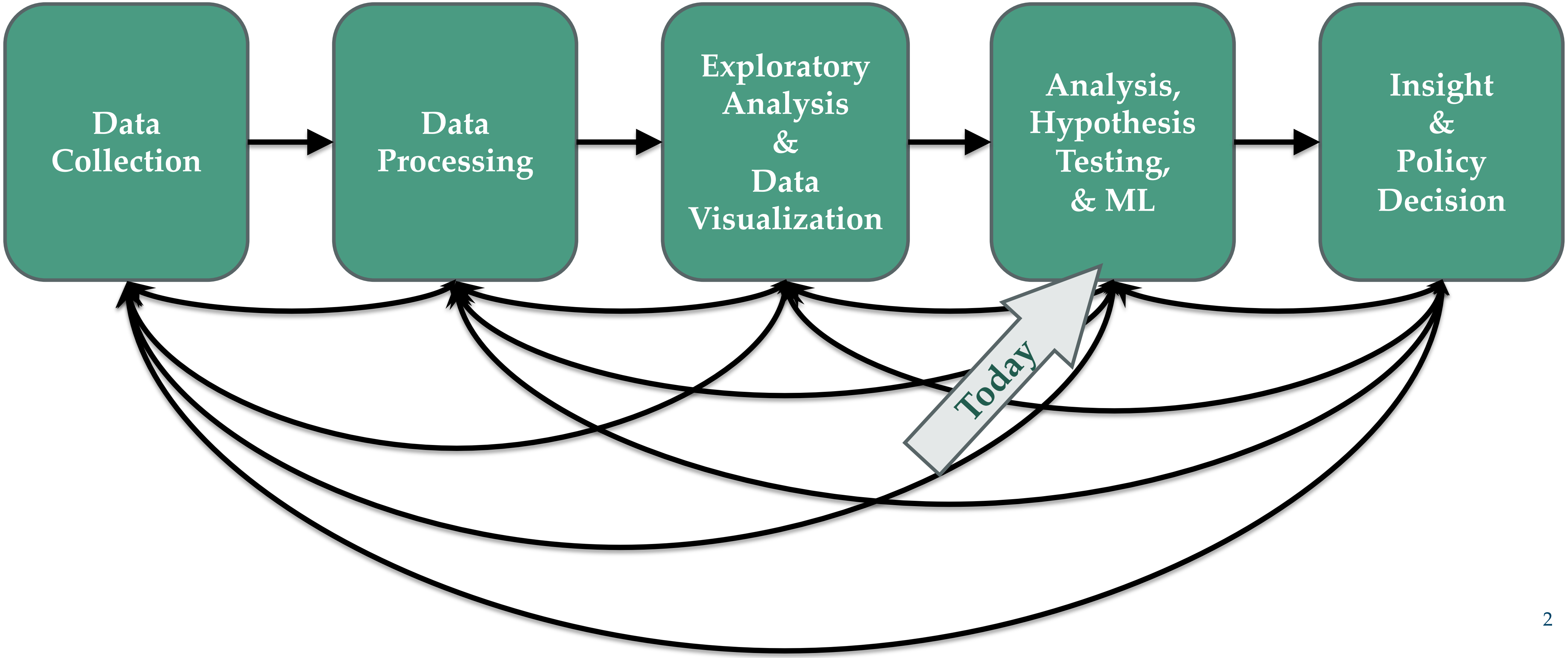
CMPS3660 – Introduction to Data Science – Fall 2019

https://rebrand.ly/TUDataScience

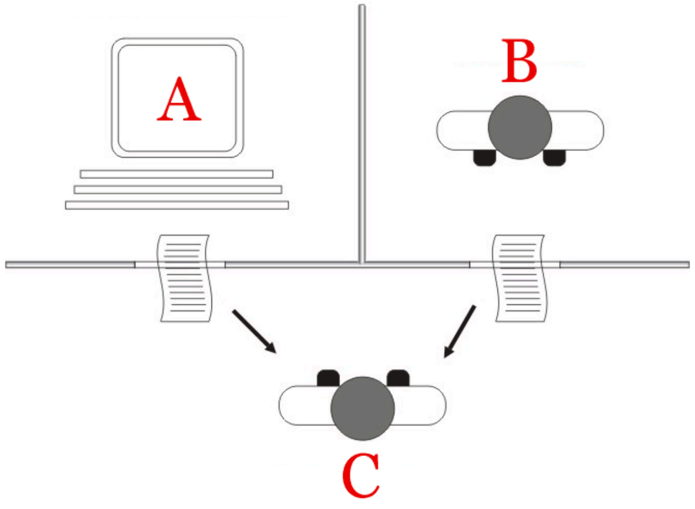
Thanks to Bart Selman [Cornell] and Zico Kolter [CMU]

Many Thanks
Slides based off Introduction to Data Science from John P. Dickerson - https://cmsc320.github.io/

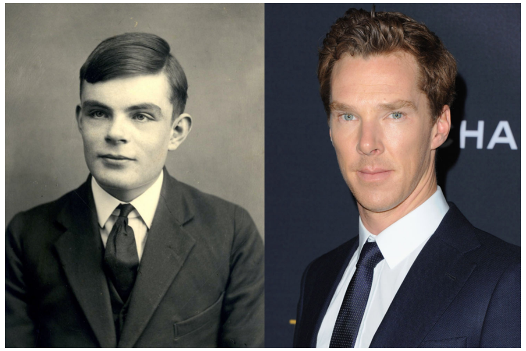
The Data LifeCycle

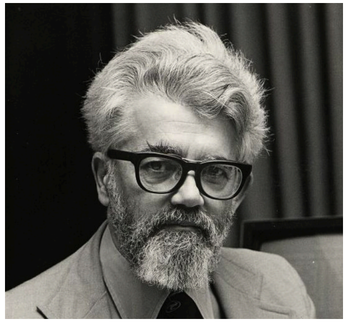


Wait, What is AI? What is ML?



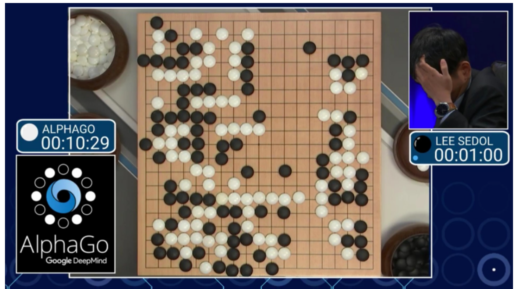
- **Alan Turing** 1950, *"I propose to consider the question, 'Can machines think?' … I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words."*
  - **The Imitation Game:** "… a man (A), a woman (B) and an interrogator (C) … The object of the game for the interrogator is to determine which of the other two is the man and which is the woman.





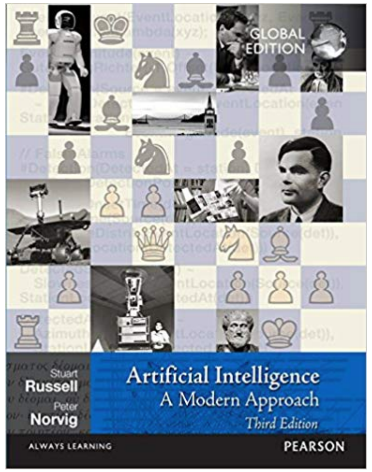
- **John McCarthy** et al. 1955, "… *every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it."*

- **Tessler's Theorem:** "*Intelligence is what machines haven't done yet."*
  - I.e., "If machines can do it, it isn't intelligence.



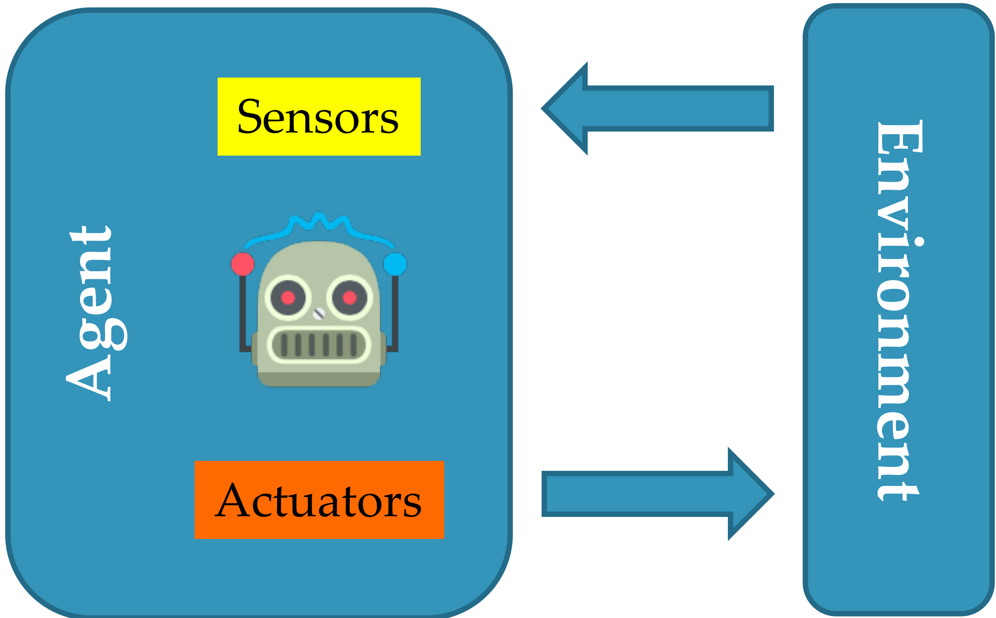
AI for AI Researchers

- **Russell and Norvig**, *"'artificial intelligence' is when a machine mimics 'cognitive functions' that humans associate with other human minds such as 'learning' and 'problem solving'"*
  - Broadly responsible for the agent based approach: "… *rational agents that acts so as to achieve the best outcome, or, when there is uncertainty, the best expected outcome."*



- Traditional Goals:
  - Reasoning, Planning, and Knowledge Representation;
  - Machine Learning and Natural Language Processing;
  - Perception (vision), and Embodiment (Robotics).

  - All of these are in service of (for some) creating Artificial General Intelligence (AGI).

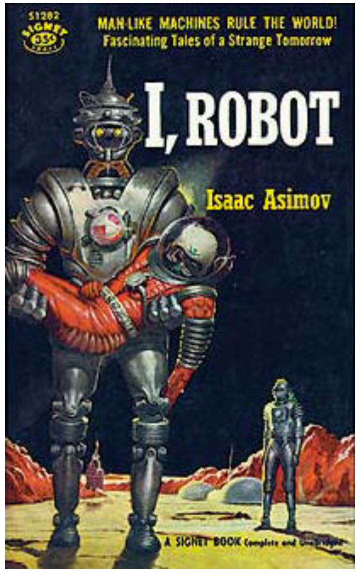


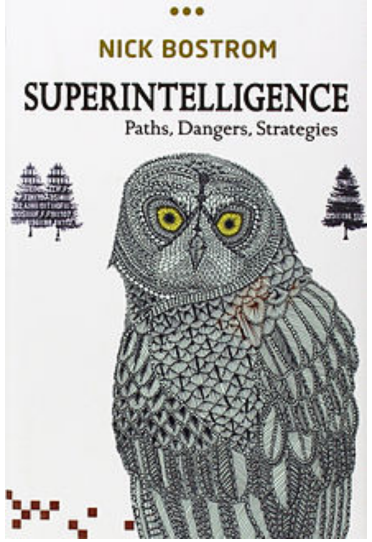
Artificial General Intelligence?

- What you think of when you think of *AI* most of the time.
  - **Strong AI:** program for the whole range of human cognition.
  - **Weak AI:** narrow or domain specific tasks.

```
> Hello, I am Eliza.
* I am afraid.
> How long have you been afraid?
* Since midterms.
> Tell me more...
* I am afraid that I will fail
my course.
> Do you enjoy being afraid that
you will fail your course?
```

- *If* we can create a big enough system, we can simulate a human … or close enough.
  - Intelligence is just a function mapping inputs to outputs.
  - Is there a difference between faking it and making it?



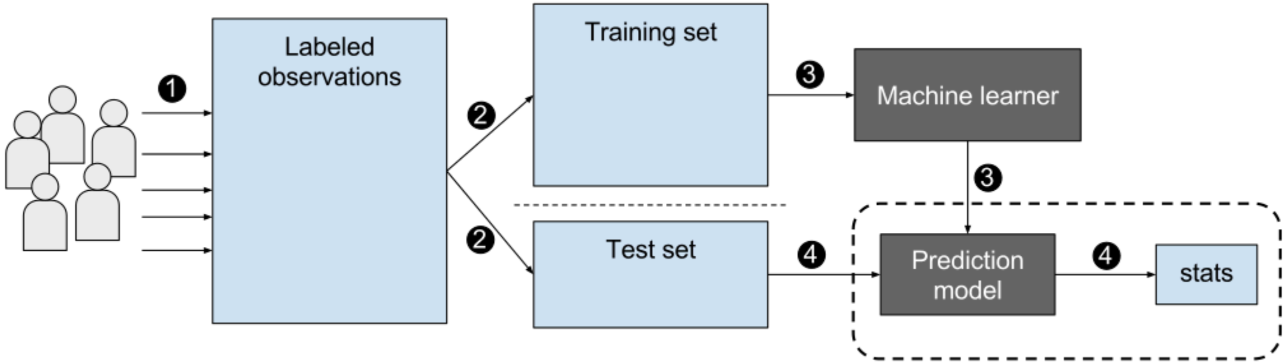


Tulane University

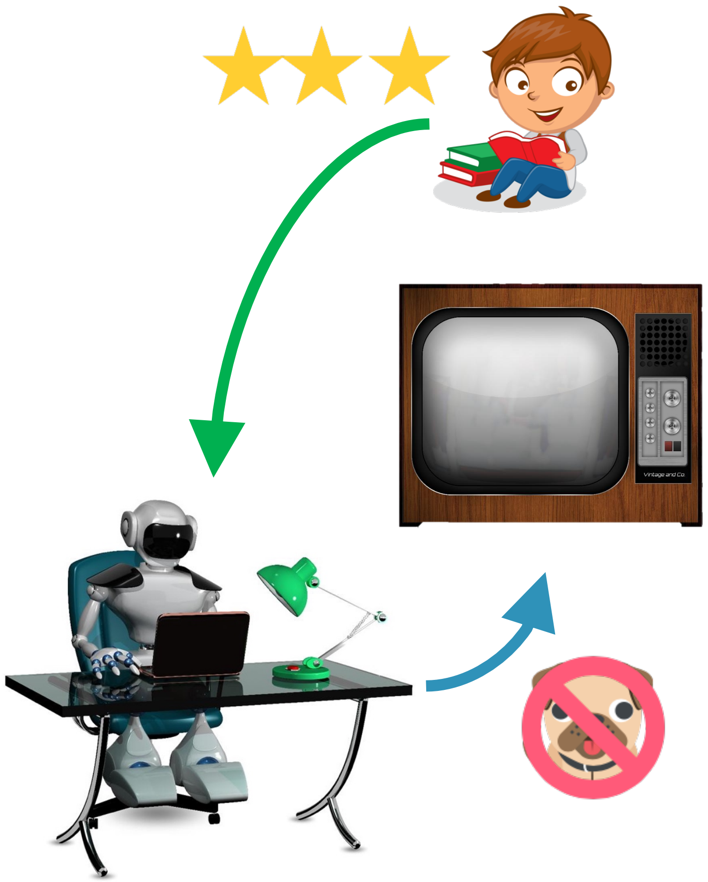
Machine Learning

- *(Depending on whom you ask, either is AI or is a subfield of AI.)*
- **Arthur Samuel 1959**, *"… give[s] computers the ability to learn without being explicitly programmed."*

- **Tom M. Mitchell**, "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."



A Concrete Example: Supervised Learning



- In many domains it's **hard to build a predictive model but easy to collect data!**
- Machine learning gives us a way to automatically infer a predictive model from the data.
- Given many many many examples consisting of a vector of *features* (x) and their output label (y): $([x_1, x_2, \ldots x_n]^{(1)}, (y^{(1)}))$.

Training Data → **Machine Learning Algorithm** → **Prediction (Model)**

Training Data:

$([x_1, x_2, \ldots x_n]^{(1)}, (y^{(1)}))$
$([x_1, x_2, \ldots x_n]^{(2)}, (y^{(2)}))$
$([x_1, x_2, \ldots x_n]^{(3)}, (y^{(3)}))$
…

Machine Learning Algorithm:

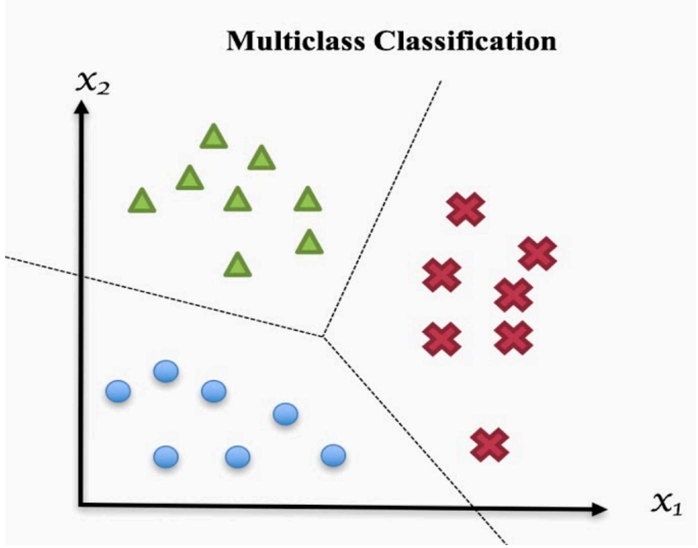
Hypothesis Function
$y^{(i)} \approx h(x^{(i)})$

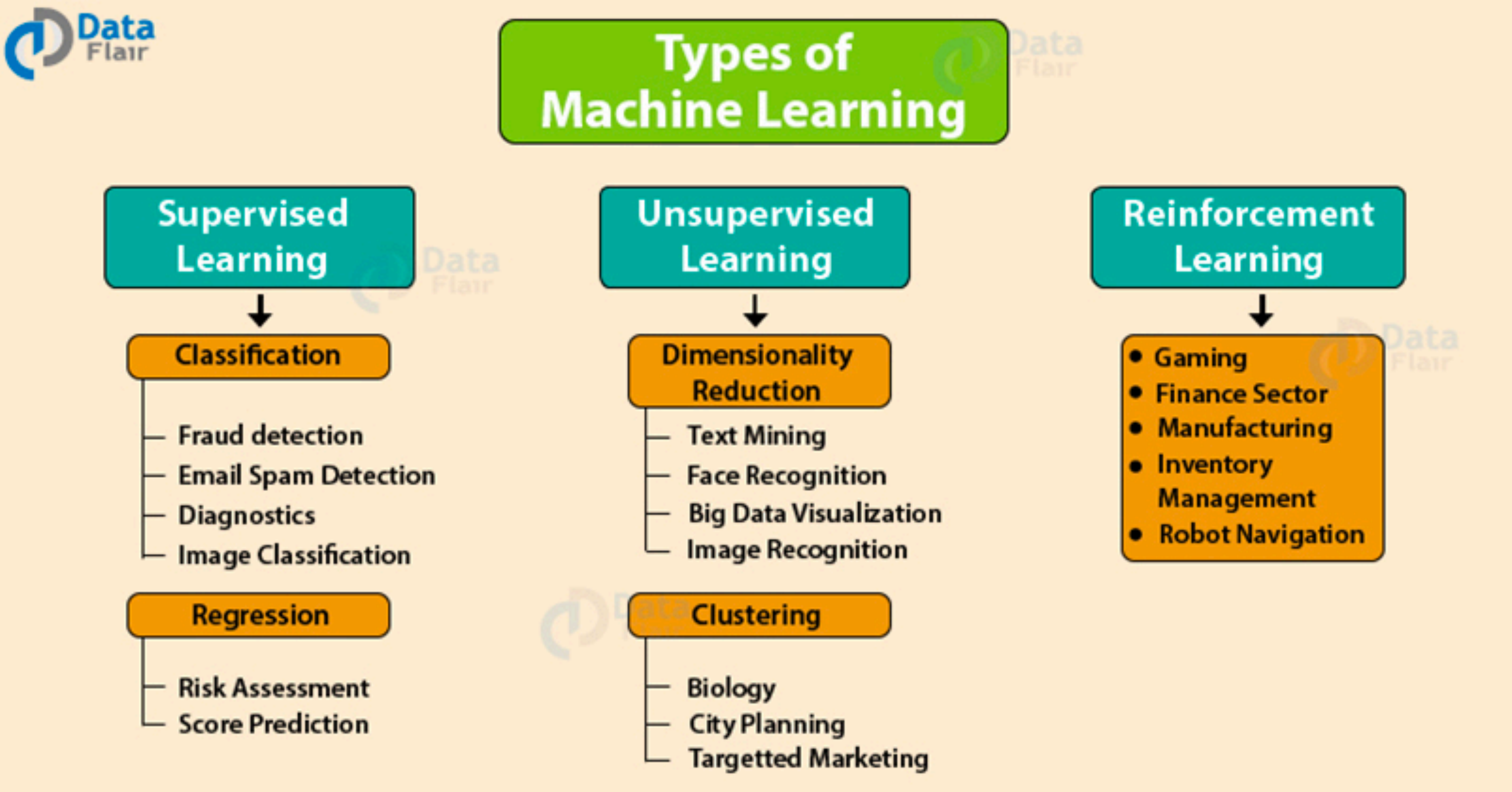
Prediction (Model):

New example x…
$\hat{y} = h(x)$

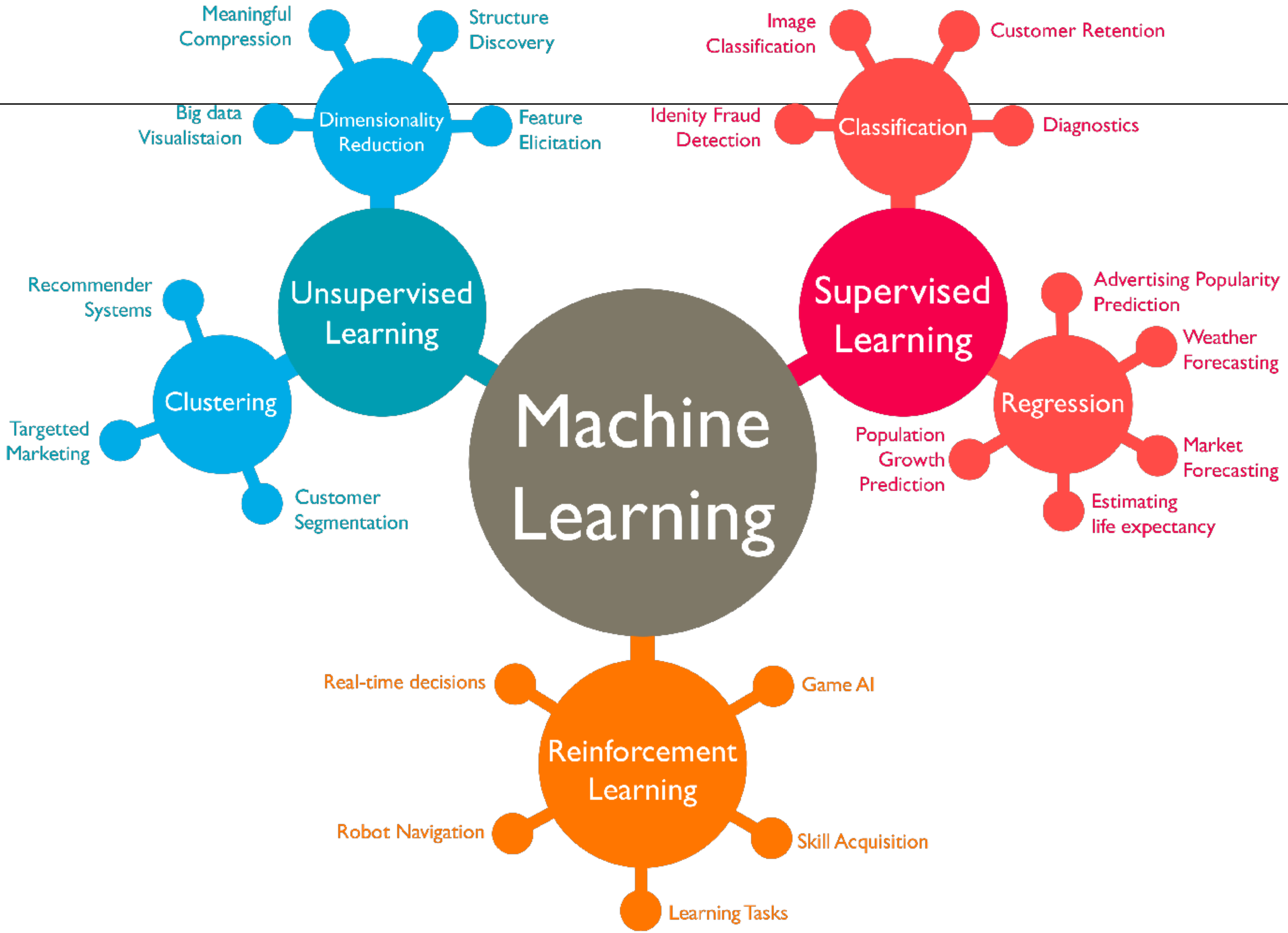
Learning: Types of Feedback

- Supervised Learning.
  - Learn a **function** from examples of its inputs and outputs.
  - E.g., An agent is presented with many camera images and is told to learn which ones contain busses.
  - Agent learns to map from images to Boolean output 0/1 of bus not present/present.
  - Learning decision trees is a form of supervised learning.
- Unsupervised Learning.
  - Learn patterns in the input with no output values supplied.
  - E.g,: Identify communities on the Internet.
- Reinforcement Learning.
  - Learn from reinforcement (occasional rewards).
  - E.g., An agent learns how to play backgammon or go or chess against itself.





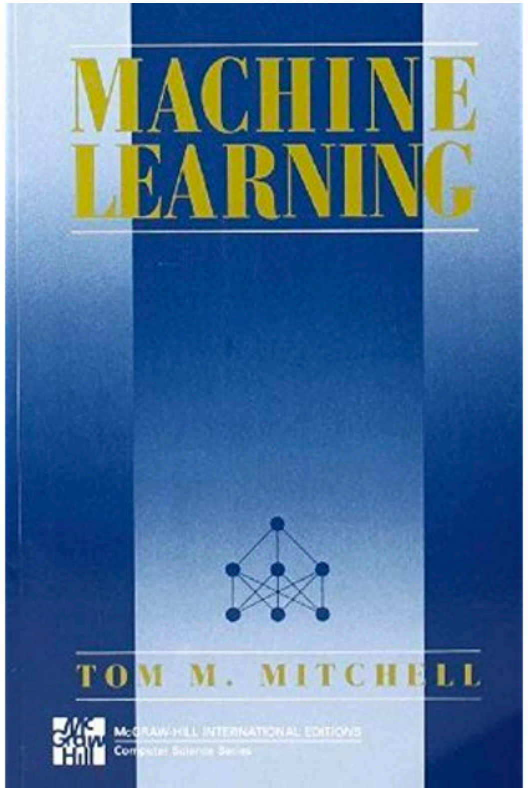
Types of Machine Learning
Supervised Learning
Classification
Fraud detection
Email Spam Detection
Diagnostics
Image Classification
Regression
Risk Assessment
Score Prediction
Unsupervised Learning
Dimensionality Reduction
Text Mining
Face Recognition
Big Data Visualization
Image Recognition
Clustering
Biology
City Planning
Targetted Marketing
Reinforcement Learning
Gaming
Finance Sector
Manufacturing
Inventory Management
Robot Navigation


Machine Learning
Unsupervised Learning
Supervised Learning
Reinforcement Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

Learning: Mitchell's Definition

- 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*.

- We're going to focus on a specific of learning:
  - Learning from Examples: Special case of inductive learning



Examples

- Spam Filtering:
  - T: Classify Emails (HAM/SPAM)
  - E: Examples (e1, HAM), (e2, SPAM), (e3, HAM), (e4, SPAM)…
  - P: Prob. of mis-classification on new emails.
- Personalized Retrieval
  - T: Find Documents the user wants for query.
  - E: Watch documents people click on (query/click pairs).
  - P: Number of Relevant Docs in Top-10
- Play Checkers:
  - T: Play Checkers
  - E: Games against self
  - P: Winning Percentage.

Inductive Learning Example

Food (3)	Chat (2)	Fast (2)	Price (3)	Bar (2)	BigTip
great	yes	yes	normal	no	yes
great	no	yes	normal	no	yes
mediocre	yes	no	high	no	no
great	yes	yes	normal	yes	yes

Instance Space X: Set of all possible objects described by attributes (often called features).

Target Function f: Mapping from Attributes to Target Feature (often called label) (f is unknown)

Hypothesis Space H: Set of all classification rules h_i we allow.

Training Data D: Set of instances labeled with Target Feature

Inductive Learning / Concept Learning

- Task:
  - Learn (to imitate) a function $f: X \rightarrow Y$
- Training Examples:
  - Learning algorithm is given the correct value of the function for particular inputs.
  - An example is a pair ($[x_1, x_2, \ldots x_n]^{(1)}, (y^{(1)})$) where x is the input vector of *features* and y is the output of the function f applied to x.
- Goal:
  - Learn a function $h: X \rightarrow Y$ that approximates $f: X \rightarrow Y$ as well as possible.

Training Data

($[x_1, x_2, \ldots x_n]^{(1)}, (y^{(1)})$)
($[x_1, x_2, \ldots x_n]^{(2)}, (y^{(2)})$)
($[x_1, x_2, \ldots x_n]^{(3)}, (y^{(3)})$)
…

Machine Learning Algorithm

Hypothesis Function
$y^{(i)} \approx h(x^{(i)})$

Prediction (Model)

New example x…
$\hat{y} = h(x)$

Tulane University

Classification v. Regression

- Naming:
  - If Y is a discrete set, then we call it **Classification.**
  - If Y is not a discrete set, then we call it **Regression.**
- Examples:
  - Steering a vehicle…
    - road images -> direction to turn wheel (distance).
  - Medical Diagnosis…
    - patient symptoms -> has/does not have disease.
  - Forensic Hair Comparison…
    - Image of two haris -> match or not.
  - Stock Market Prediction:
    - closing price of last few days -> market will go up or down (how much?)
  - Noun Phrase Coreference:
    - description of two things in document -> same entity?

Inductive Learning Algorithm

Task:

– Given: collection of examples
– Return: a function *h* (*hypothesis*) that approximates *f*

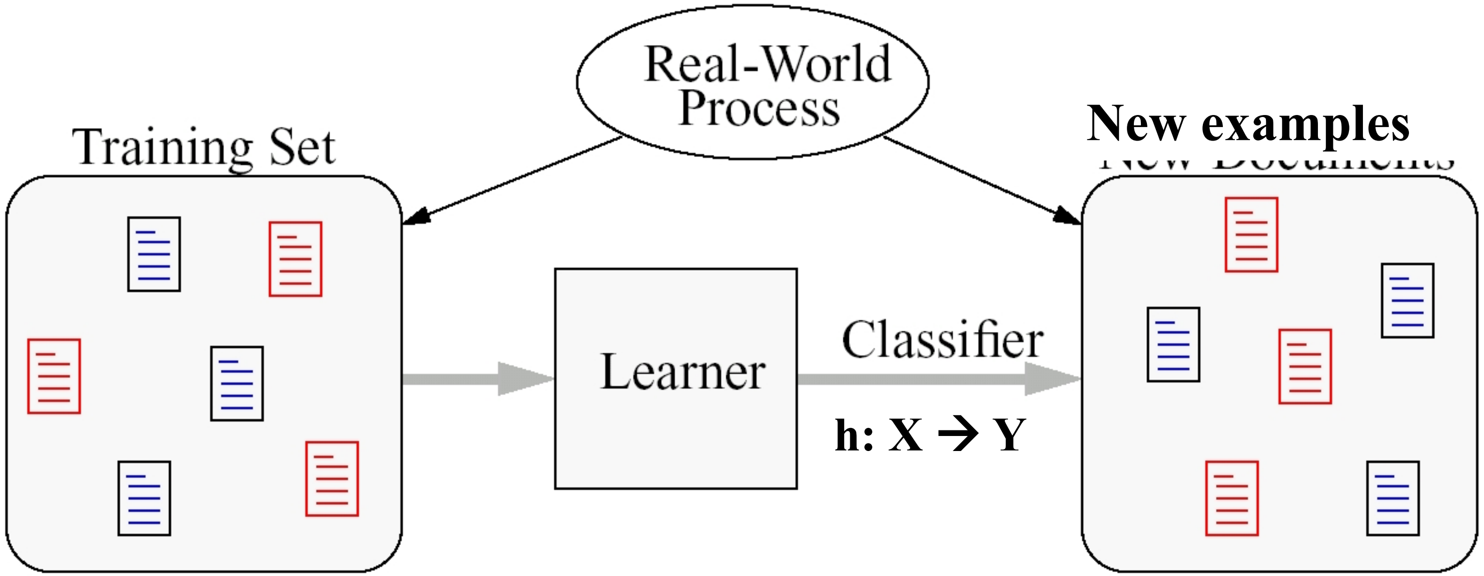
Inductive Learning Hypothesis:

Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over any other unobserved examples.

Assumptions of Inductive Learning:

– The training sample represents the population
– The input features permit discrimination

Inductive Learning Setting



Task:

Learner (or inducer) induces a general rule h from a set of observed examples that classifies new examples accurately. An algorithm that takes as input specific instances and produces a model that generalizes beyond these instances.

Classifier - A mapping from unlabeled instances to (discrete) classes.

Classifiers have a form (e.g., decision tree) plus an interpretation procedure (including how to handle unknowns, etc.)

Inductive learning method

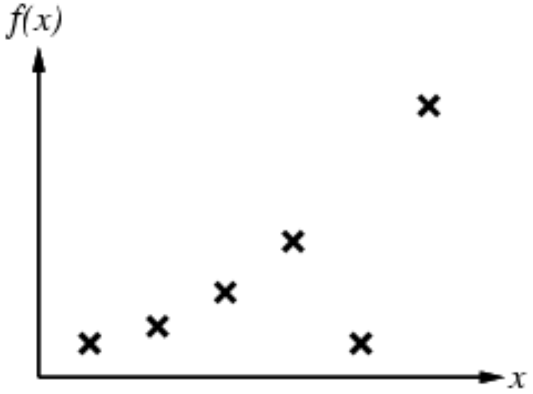
Fitting a function of a single variable to some data points

Examples are (x, f(x)) pairs;

Hypothesis space H – set of hypotheses we will consider for function f, in this case **polynomials of degree at most k**

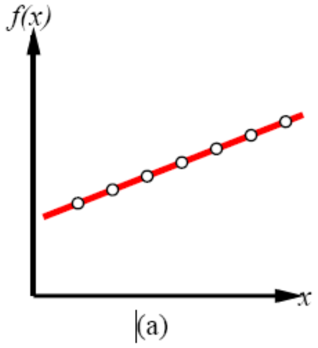
Construct/adjust *h* to agree with *f* on training set

(*h* is consistent if it agrees with *f* on all examples)

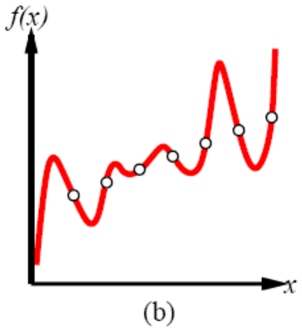


Multiple consistent hypotheses?

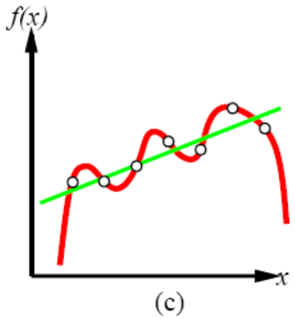
Polynomials of degree at most k



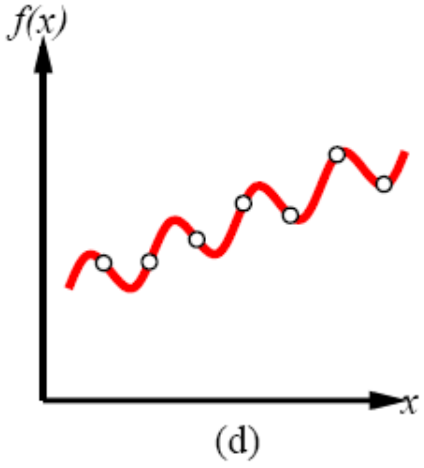
Linear hypothesis



Degree 7 polynomial hypothesis



Degree 6 polynomial and approximate linear fit



Sinusoidal hypothesis

How to choose from among multiple consistent hypotheses?

Ockham's razor: ***maximize a combination of consistency and simplicity***

Preference Bias: Ockham's Razor

Aka Occam's Razor, Law of Economy, or Law of Parsimony

Principle stated by William of Ockham (1285-1347/49), an English philosopher, that

- "*non sunt multiplicanda entia praeter necessitatem*"
- or, entities are not to be multiplied beyond necessity.

The simplest explanation that is consistent with all observations is the best.

- E.g, the smallest decision tree that correctly classifies all of the training examples is the best.
- Finding the provably smallest decision tree is NP-Hard, so instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small.

Different Hypothesis Spaces

Learning can be seen as fitting a function to the data. We can consider different functions as the target function and therefore different hypothesis spaces. Examples:

Propositional if-then rules
Decision Trees
First-order if-then rules
First-order logic theory
Linear functions
Polynomials of degree at most k
Neural networks
Java programs
Etc

Tradeoff in expressiveness and complexity

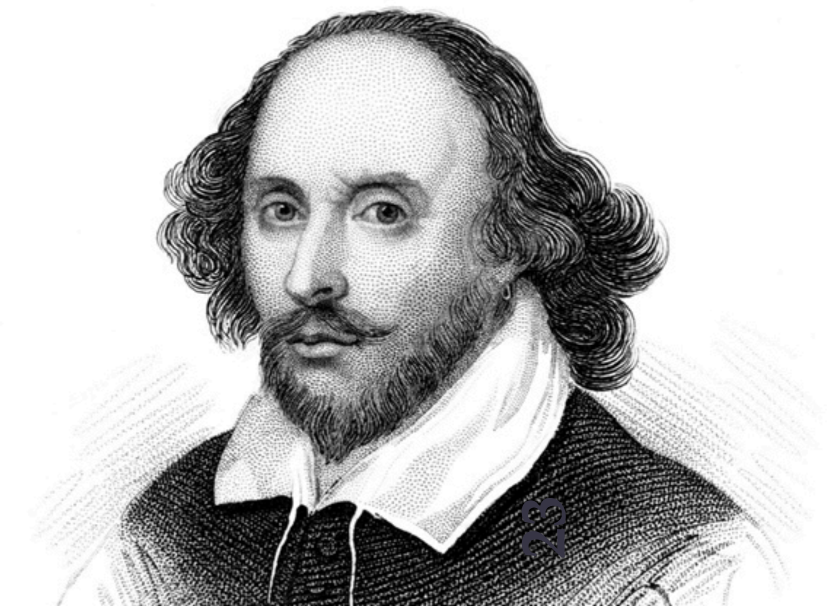
A learning problem is realizable if its hypothesis space contains the true function.

Why not pick the largest possible hypothesis space, say the class of all Turing machines?

Tradeoff between expressiveness of a hypothesis space and the complexity of finding simple, consistent hypotheses within the space (also risk of "overfitting").
Extreme overfitting: Just remember all training examples.

Semantics: Text classification

- Is it spam?
- Who wrote this paper? (Author identification)
- https://en.wikipedia.org/wiki/The_Federalist_Papers#Authorship
- https://www.uwgb.edu/dutchs/pseudosc/hidncode.htm
- ¡Identificación del idioma!
- Sentiment analysis
- What type of document is this?
- When was this document written?
- Readability assessment



Text classification

- Input:
  - A document w
  - A set of classes $Y = \{y_1, y_2, \ldots, y_J\}$

- Output:
  - A predicted class $y \in Y$

- (You will spend much more time on classification problems throughout the program, this is just a light intro!)

Text classification

- Hand-coded rules based on combinations of terms (and possibly other context)
- If email w:
- Sent from a DNSBL (DNS blacklist) OR
- Contains "Nigerian prince" OR
- Contains URL with Unicode OR …
- Then: y_w = spam
- Pros: ?????????
- Domain expertise, human-understandable
- Cons: ?????????
- Brittle, expensive to maintain, overly conservative

Text classification

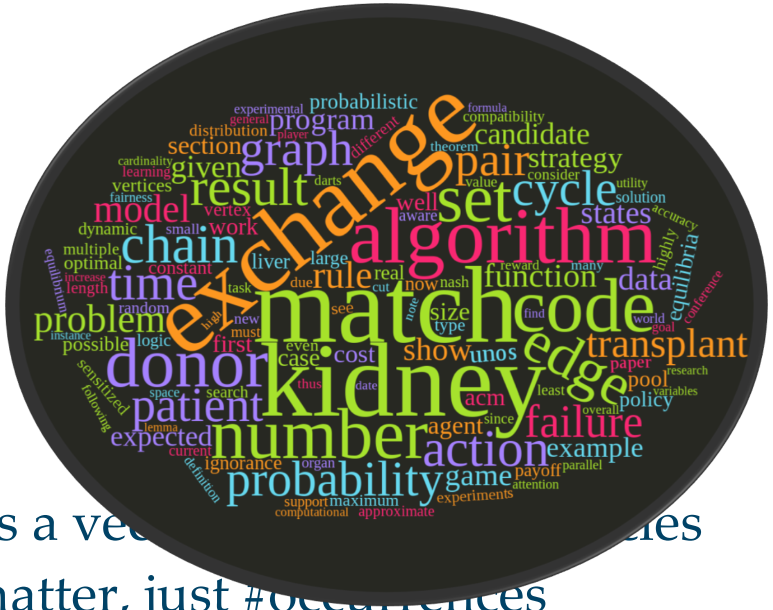
- Input:
  - A document w
  - A set of classes $Y = \{y_1, y_2, \ldots, y_J\}$
  - A training set of m hand-labeled documents
    $\{(w_1, y_1), (w_2, y_2), \ldots, (w_m, y_m)\}$

- Output:
  - A learned classifier $w \rightarrow y$

- This is an example of supervised learning

Representing a document “in math”

- Simplest method: bag of words



- Represent each document as a ve… …es
- Order of words does not matter, just #o…

Bag of words Example

- the quick brown fox jumps over the lazy dog
- I am he as you are he as you are me
- he said the CMSC320 is 189 more CMSCs than the CMSC131

	the	CMSC320	you	he	I	quick	dog	me	CMSCs	…	than
Document 1	2	0	0	0	0	1	1	0	0		0
Document 2	0	0	2	2	1	0	0	1	0	…	0
Document 3	2	1	0	1	0	0	0	0	1		1

Term Frequency

- Term frequency: the number of times a term appears in a specific document
  - tf_{ij}: frequency of word *j* in document *i*
- This can be the raw count (like in the BOW in the last slide):
  - $\mathrm{tf}_{ij} \in \{0,1\}$ if word *j* appears or doesn't appear in doc *i*
  - $\log(1 + \mathrm{tf}_{ij})$ – reduce the effect of outliers
  - $\mathrm{tf}_{ij} / \max_j \mathrm{tf}_{ij}$ – normalize by document i's most frequent word
- What can we do with this?
  - Use as features to learn a classifier $w \rightarrow y$ …!

Defining features From Term Frequency

- Suppose we are classifying if a document was written by The Beatles or not (i.e., binary classification):
- Two classes $y \in Y = \{ 0, 1 \} = \{$ not_beatles, beatles $\}$
- Let's use $tf_{ij} \in \{0,1\}$, which gives:

	the	CMSC320	you	he	I	quick	dog	me	CMSCs	…	than	
$x_1^T =$	1	0	0	0	0	1	1	0	0		0	$y_1 = 0$
$x_2^T =$	0	0	1	1	1	0	0	1	0	…	0	$y_2 = 1$
$x_3^T =$	1	1	0	1	0	0	0	0	1		1	$y_3 = 0$

- Then represent documents with a feature function:

$f(x, y = \text{not\_beatles} = 0) = [x^T, 0^T, 1]^T$

$f(x, y = \text{beatles} = 1) = [0^T, x^T, 1]^T$

Linear classification

- We can then define weights θ for each feature
- θ = { <CMSC320, not_beatles> = +1,
  <CMSC320, beatles> = -1,
  <walrus, not_beatles> = -0.3,
  <walrus, beatles> = +1,
  <the, not_beatles> = 0,
  <the, beatles>, 0, … }
- Write weights as vector that aligns with feature mapping
- Score $\boldsymbol{\psi}$ of an instance x and class y is the sum of the weights for the features in that class:
- $\boldsymbol{\psi}_{xy} = \Sigma\, \theta_n f_n(\mathrm{x}, y)$
- $= \theta^T\, \mathrm{f}(\mathrm{x}, y)$

Linear classification

- We have a feature function f(x, y) and a score $\boldsymbol{\psi}_{xy} = \theta^{\mathrm{T}}$ f(x, y)

And return the class with highest score!

Compute the score of the document for that class

$$\hat{y} = \arg\max_{y} \theta^{\intercal} \mathbf{f}(\mathbf{x}, y)$$

For each class y ∈ { not_beatles, beatles }

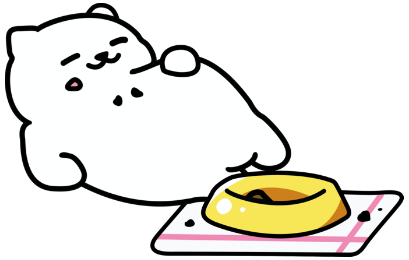
(… and also this whole "linear classifier" thing.)

Where did these weights come from? We'll talk about this in the ML lectures …

Explicit Example

- We are interested in classifying documents into one of two classes $y \in \mathcal{Y} = \{ 0, 1 \} = \{$ hates_cats, likes_cats$\}$
- Document 1: I like cats
- Document 2: I hate cats

	I	like	hate	cats
$x_1^T =$	1	1	0	1
$x_2^T =$	1	0	1	1



$y_1 = ?$

$y_2 = ?$

- Now, represent documents with a feature function:

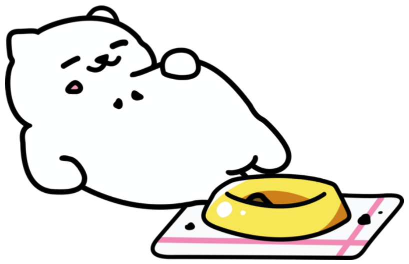
$$f(x, y = \text{hates\_cats} = 0) = [x^T, 0^T, 1]^T$$
$$f(x, y = \text{likes\_cats} = 1) = [0^T, x^T, 1]^T$$

Explicit Example

$f(\mathbf{x}, y = 0) = [\mathbf{x}^T, 0^T, 1]^T$

$f(\mathbf{x}, y = 1) = [0^T, \mathbf{x}^T, 1]^T$

	I	like	hate	cats
$x_1^T =$	1	1	0	1
$x_2^T =$	1	0	1	1



$y_1 = ?$

$y_2 = ?$

	y=0: hates_cats				*y=1: likes_cats*				(1)
	I	like	hate	cats	I	like	hate	cats	:
$f(\mathbf{x}_1, y = \text{hates\_cats} = 0) =$	1	1	0	1	0	0	0	0	1
$f(\mathbf{x}_1, y = \text{likes\_cats} = 1) =$	0	0	0	0	1	1	0	1	1
$f(\mathbf{x}_2, y = \text{hates\_cats} = 0) =$	1	0	1	1	0	0	0	0	1
$f(\mathbf{x}_2, y = \text{likes\_cats} = 1) =$	0	0	0	0	1	0	1	1	1

Explicit Example

- Now, assume we have weights θ for each feature
- θ = { <I, hates_cats> = 0, <I, likes_cats> = 0,
- <like, hates_cats> = -1, <like, likes_cats> = +1,
- <hate, hates_cats> = +1, <hate, likes_cats> = -1,
- <cats, hates_cats> = -0.1, <cats, likes_cats = +0.5> }
- Write weights as vector that aligns with feature mapping:

	y=0: hates_cats				*y=1: likes_cats*				(1)
Parameter vector $\boldsymbol{\theta}^{T}$ =	0	-1	1	-0.1	0	1	-1	0.5	1
	I	like	hate	cats	I	like	hate	cats	--
f($\mathbf{x}_1$, y = hates_cats = 0) =	1	1	0	1	0	0	0	0	1
f($\mathbf{x}_1$, y = likes_cats = 1) =	0	0	0	0	1	1	0	1	1
f($\mathbf{x}_2$, y = hates_cats = 0) =	1	0	1	1	0	0	0	0	1
f($\mathbf{x}_2$, y = likes_cats = 1) =	0	0	0	0	1	0	1	1	1

Explicit example

- Score $\boldsymbol{\psi}$ of an instance x and class y is the sum of the weights for the features in that class:
- $\boldsymbol{\psi}_{xy} = \Sigma\ \theta_n f_n(\mathrm{x}, y)$
- $= \theta^{\mathrm{T}}\ \mathrm{f}(\mathrm{x}, y)$
- Let's compute $\boldsymbol{\psi}_{x1, y=hates\_cats}$ …
- $\boldsymbol{\psi}_{x1, y=hates\_cats} = \theta^{\mathrm{T}}\ \mathrm{f}(\mathrm{x}_1, y = \mathrm{hates\_cats} = 0)$
- $= 0*1 + -1*1 + 1*0 + -0.1*1 + 0*0 + 1*0 + -1*0 + 0.5*0 + 1*1$
- $= -1 - 0.1 + 1 = -0.1$

$\boldsymbol{\theta}^{\mathrm{T}} =$

0	-1	1	-0.1	0	1	-1	0.5	1

●

1	I	*hates_cats*
1	like	
0	hate	
1	cats	
0	I	*likes_cats*
0	like	
0	hate	
0	cats	
1	–	(1)

$\mathrm{f}(\mathbf{x_1}, y = 0)$

Explicit example

- Saving the boring stuff:
- $\boldsymbol{\psi}_{x1,y=hates\_cats} = -0.1$; $\boldsymbol{\psi}_{x1,y=likes\_cats} = +2.5$
- $\boldsymbol{\psi}_{x2,y=hates\_cats} = +1.9$; $\boldsymbol{\psi}_{x2,y=likes\_cats} = +0.5$
- We want to predict the class of each document:

Document 1: I like cats

Document 2: I hate cats

$$\hat{y} = \arg\max_{y} \theta^{\top} \mathbf{f}(\mathbf{x}, y)$$

- Document 1: argmax{ $\boldsymbol{\psi}_{x1,y=hates\_cats}$, $\boldsymbol{\psi}_{x1,y=likes\_cats}$ } ????????
- Document 2: argmax{ $\boldsymbol{\psi}_{x2,y=hates\_cats}$, $\boldsymbol{\psi}_{x2,y=likes\_cats}$ } ????????

Inverse Document Frequency

- Recall:
- tf_{ij}: frequency of word *j* in document *i*
- Any issues with this ??????????
- Term frequency gets overloaded by common words
- Inverse Document Frequency (IDF): weight individual words negatively by how frequently they appear in the corpus:

$$\mathrm{idf}_j = \log\left(\frac{\#\text{documents}}{\#\text{documents with word } j}\right)$$

- IDF is just defined for a word j, not word/document pair j,i

Inverse Document Frequency

	the	CMSC320	you	he	I	quick	dog	me	CMSCs	…	than
Document 1	2	0	0	0	0	1	1	0	0		0
Document 2	0	0	2	2	1	0	0	1	0	…	0
Document 3	2	1	0	1	0	0	0	0	1		1

$$\mathrm{idf}_{\mathrm{the}} = \log\left(\frac{3}{2}\right) = 0.405 \qquad \mathrm{idf}_{\mathrm{you}} = \log\left(\frac{3}{1}\right) = 1.098$$

$$\mathrm{idf}_{\mathrm{CMSC320}} = \log\left(\frac{3}{1}\right) = 1.098 \qquad \mathrm{idf}_{\mathrm{he}} = \log\left(\frac{3}{2}\right) = 0.405$$

TF-IDF

- How do we use the IDF weights?
- Term frequency inverse document frequency (TF-IDF):
- TF-IDF score: tf_{ij} x idf_j

	the	CMSC320	you	he	I	quick	dog	me	CMSCs	…	than
Document 1	0.8	0	0	0	0	1.1	1.1	0	0		0
Document 2	0	0	2.2	0.8	1.1	0	0	1.1	0	…	0
Document 3	0.8	1.1	0	0.4	0	0	0	0	1.1		1.1

- This ends up working better than raw scores for classification and for computing similarity between documents.