COMP 562

Introduction to Machine Learning

Jorge Silva

jsilva@cs.unc.edu

SN 014   T/H 5:00-6:15pm

Course website: Sakai

Welcome to COMP 562

Textbooks:

- Machine Learning – A Probabilistic perspective. Kevin P. Murphy
- Deep Learning. Ian Goodfellow, Yoshua Bengio and Aaron Courville

Optional:

- The Elements of Statistical Learning
  http://statweb.stanford.edu/~tibs/ElemStatLearn/
- Pattern Recognition and Machine Learning
  https://www.microsoft.com/en-us/research/people/cmbishop/#prml- book

Prerequisites and Languages

COMP 410, MATH 233, and STOR 435

Basic Programming and Data Structures

Discrete Math and Probability or Stats

Language: Python (NumPy and SciPy)

Grading and Exams

- 4 homework assignments (35%)
- Midterm (20%)
- Final exam (30%)
- Final project (15%)

- Final exam: The course final is given in compliance with UNC final exam regulations and according to the UNC Final Exam calendar.

- Late policy: 0 points for late submissions unless you contact me in advance.

TA and Office Hours

Your TA is Ric Poirson (poirson@cs.unc.edu)

My office hours: after class until 7:15pm
Please send an e-mail to let me know you are coming

Schedule

Week	Topic	Reading	Deadlines
1	Introduction	Chapter 1	
2	Probability Distributions and Optimization	Chapter 2	Assignment 1
3	Linear Models for Regression	Chapter 7	
4	Linear Models for Classification	Chapter 8	Assignment 2
5	Directed Graphical Models	Chapter 10	
6	Mixture Models and EM	Chapter 11	
7-8	Latent Linear Models	Chapter 12	Midterm
9-10	Sparse Linear Models	Chapter 13	Assignment 3
11	Kernel Methods and SVM	Chapter 14	
12	Neural networks and Deep Learning	Chapter 28	
13-14	Neural networks and Deep Learning	Chapter 28	Assignment 4
15-16	Neural networks and Deep Learning	Chapter 28	

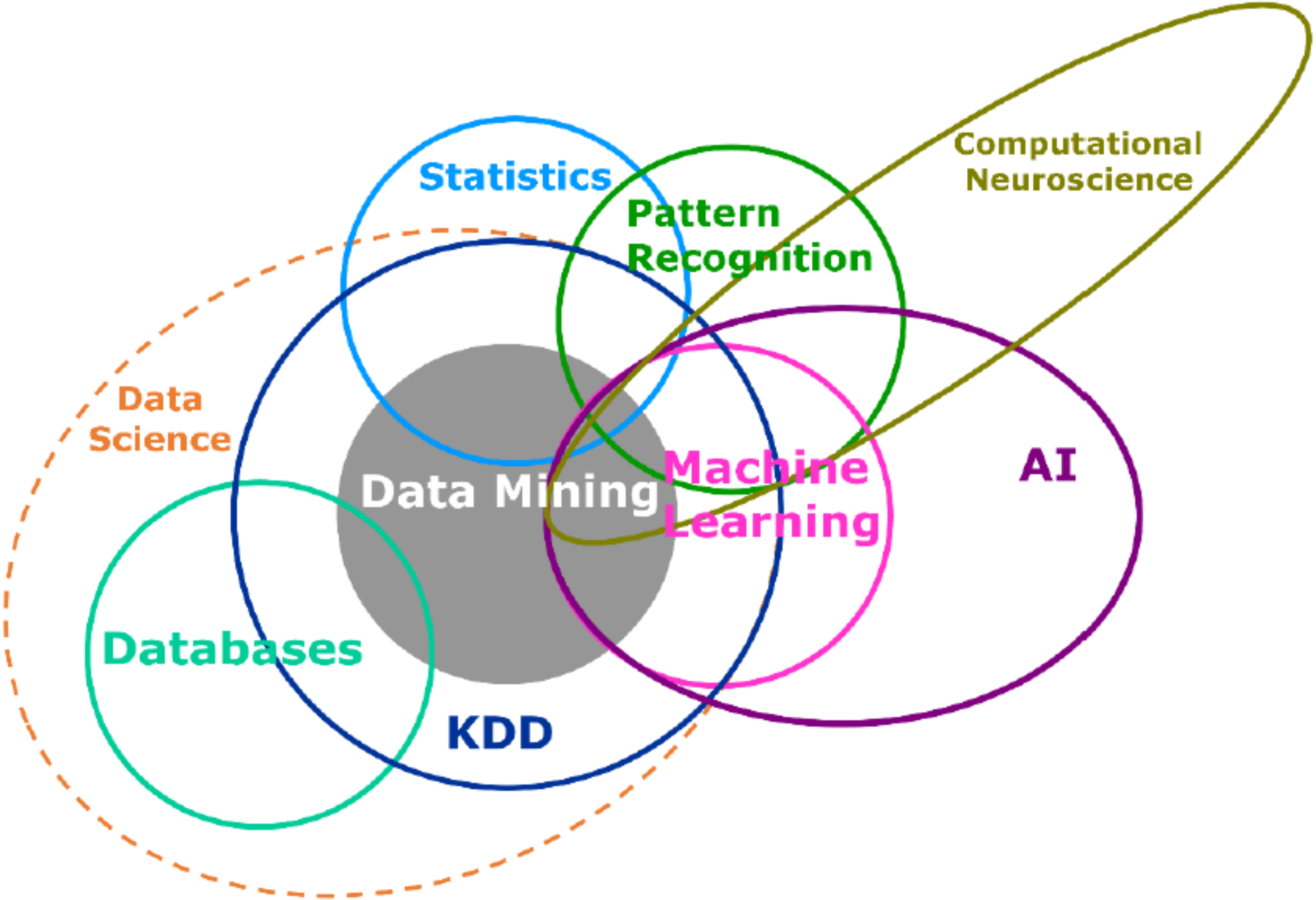
Goals

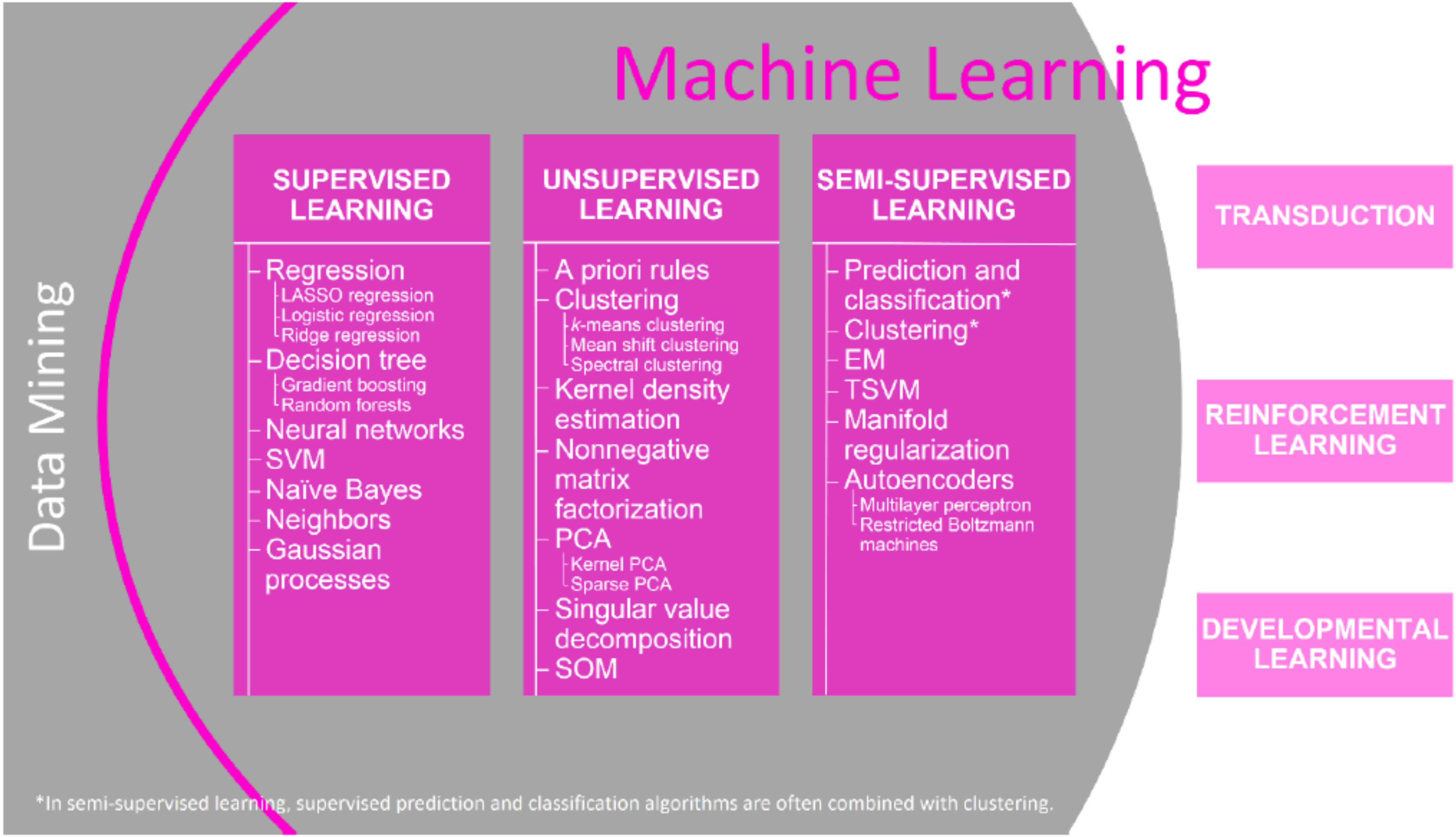
Develop an understanding of basic machine learning algorithms, their efficient implementations and their applicability to different tasks.

What is Machine Learning?

Machine learning is a branch of artificial intelligence that is concerned with building systems that require minimal human intervention in order to learn from data and make accurate predictions.

“All models are wrong, but some are useful“ –George Box


Statistics
Pattern Recognition
Computational Neuroscience
Data Science
Data Mining
Machine Learning
AI
Databases
KDD



Data Mining

Machine Learning

SUPERVISED LEARNING

- Regression
  - LASSO regression
  - Logistic regression
  - Ridge regression
- Decision tree
  - Gradient boosting
  - Random forests
- Neural networks
- SVM
- Naïve Bayes
- Neighbors
- Gaussian processes

UNSUPERVISED LEARNING

- A priori rules
- Clustering
  - *k*-means clustering
  - Mean shift clustering
  - Spectral clustering
- Kernel density estimation
- Nonnegative matrix factorization
- PCA
  - Kernel PCA
  - Sparse PCA
- Singular value decomposition
- SOM

SEMI-SUPERVISED LEARNING

- Prediction and classification*
- Clustering*
- EM
- TSVM
- Manifold regularization
- Autoencoders
  - Multilayer perceptron
  - Restricted Boltzmann machines

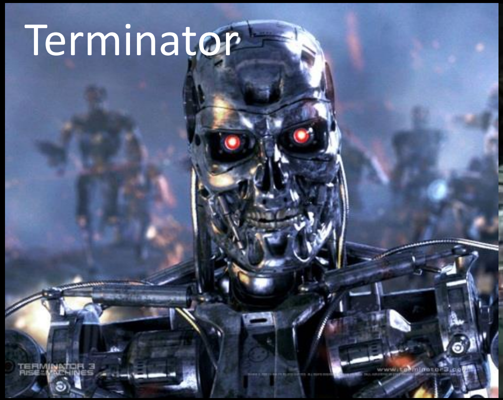
TRANSDUCTION

REINFORCEMENT LEARNING

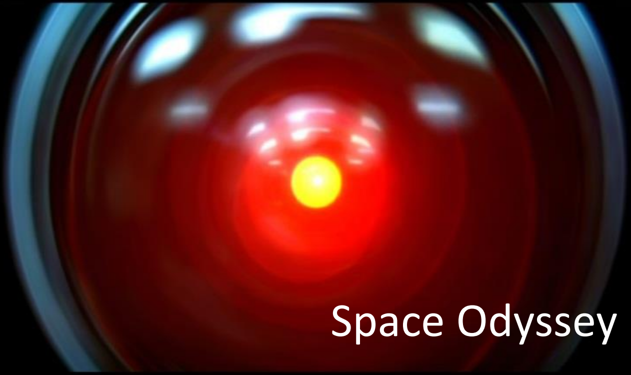
DEVELOPMENTAL LEARNING

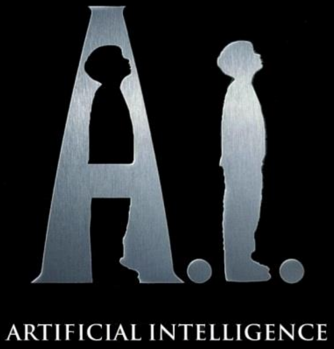
*In semi-supervised learning, supervised prediction and classification algorithms are often combined with clustering.

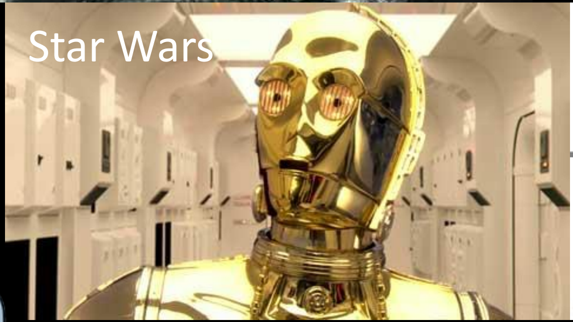
Artificial Intelligence in Art

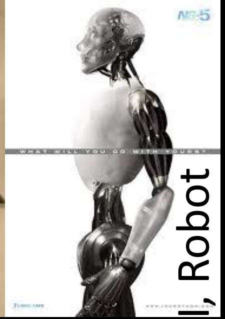

Terminator

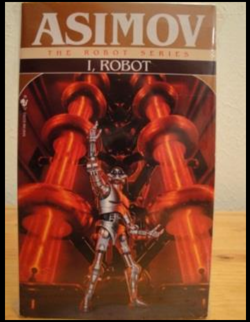
Matrix


Space Odyssey

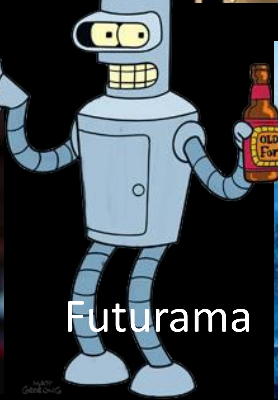



Star Wars


I, Robot



Star Trek


Futurama

Wall-E

Metropolis (1927)

Mass Effect

Reality

Siri. Beta

Your wish is its command.

Speech Recognition

Natural Language Processing

Ontology-based task delegation

Cognitive Assistant that Learns and Organizes (CALO) DARPA funded project carried out at SRI

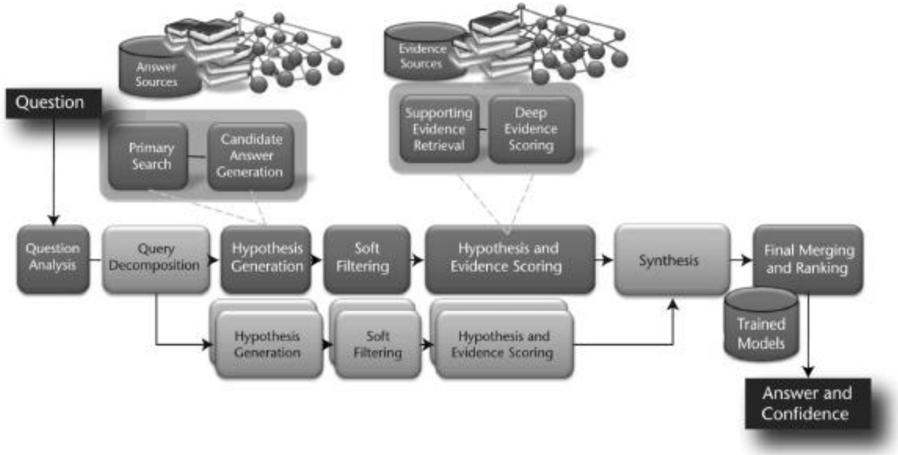
Reality

IBM Watson

Natural Language Processing

Question Answering

Decision theory

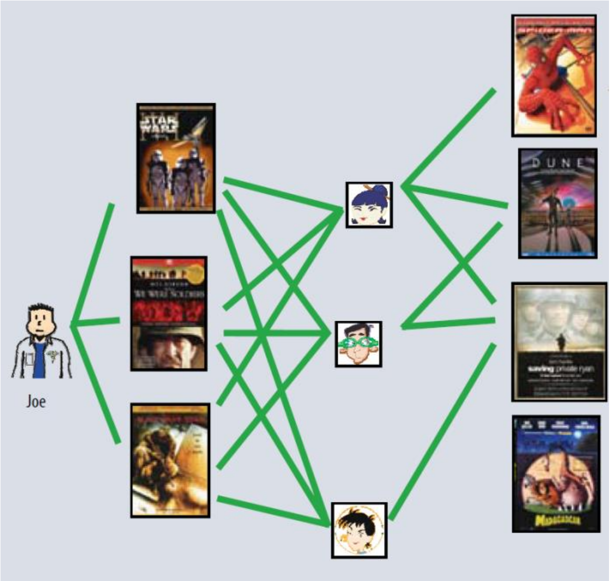


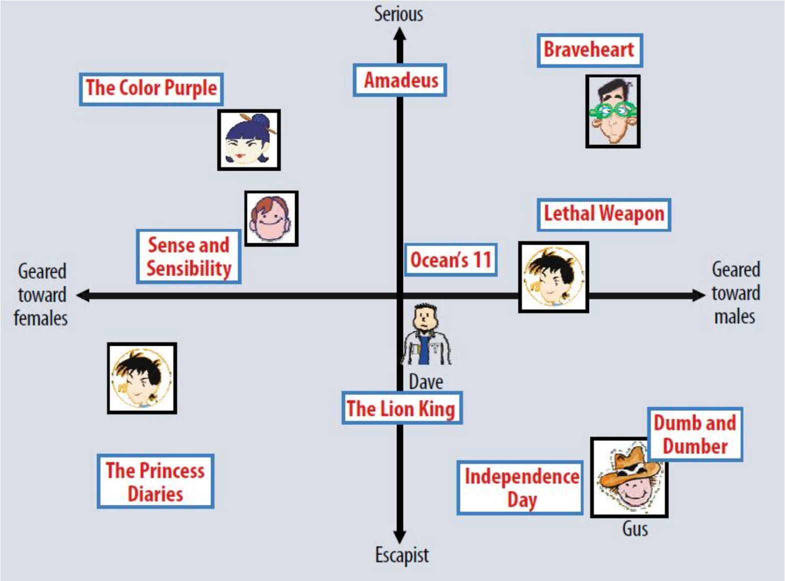
Reality

amazon.com

NETFLIX

PANDORA





Reality

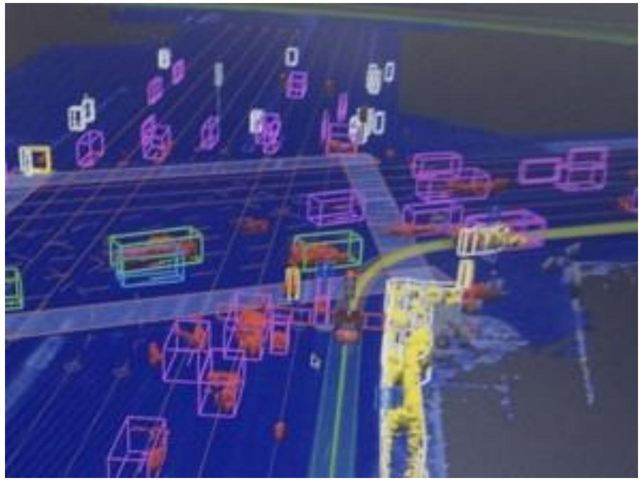
Google autonomous car

Toyota autonomous car

Computer Vision

Path/motion planning

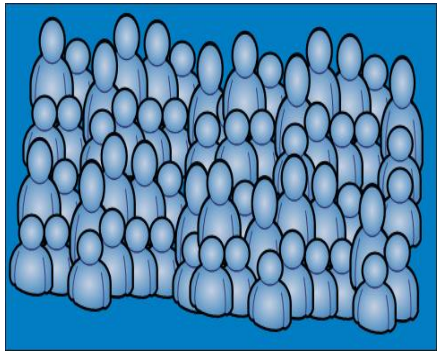
Visualization of an autonomous car's world representation



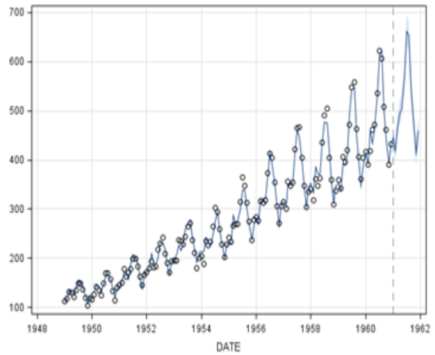
Data types

Type		Example
Numeric	Integer	-1, 0, 1, 2, 56, 100001
	“Float”, “Real”	2.335, -6792.29388675465435687
Categorical	Binary	{0, 1}, {yes, no}, {alive, dead}
	Nominal	{cat, dog, mouse}, {red, blue, green, yellow}
	Ordinal	lowest, lower, middle, higher, highest

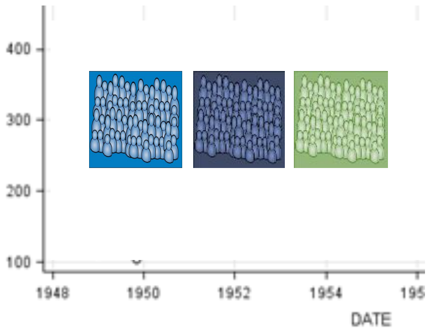
DATA Galore



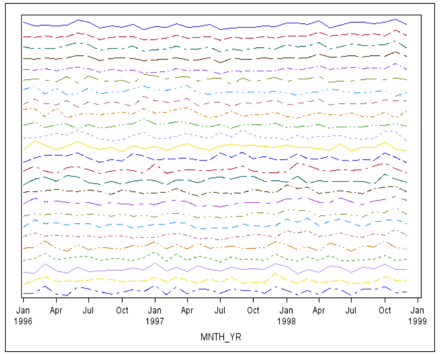
Cross-Sectional



Time Series

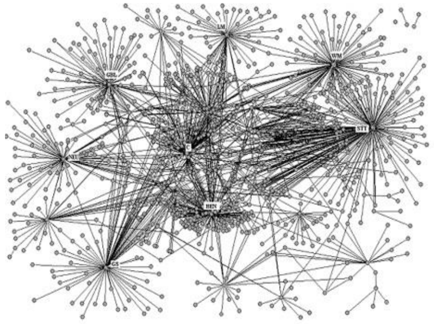


Panel

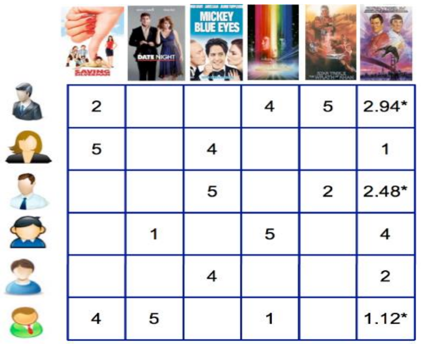


Streaming

Spatial



Network



MICKEY BLUE EYES

2			4	5	2.94*
5		4			1
		5		2	2.48*
	1		5		4
		4			2
4	5		1		1.12*

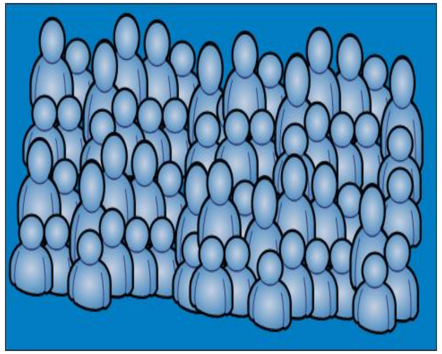
Link

Text

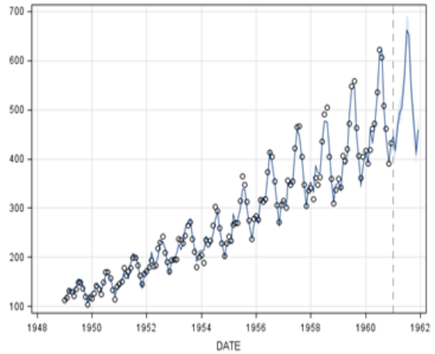
Sound

Image/Video

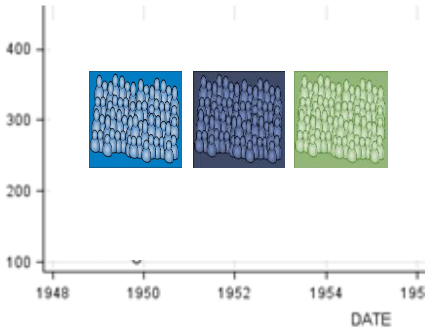
Business problems



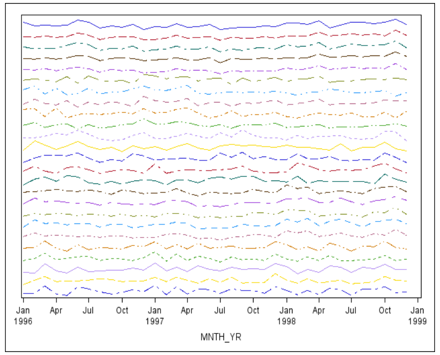
Who will respond to a campaign?



What will future demand look like?

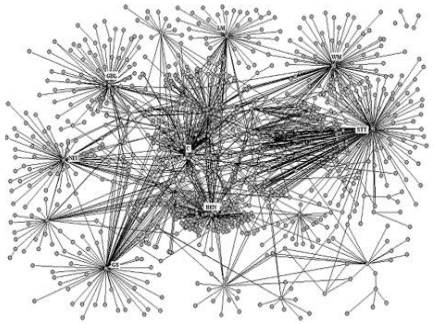


Will regulations have an impact?

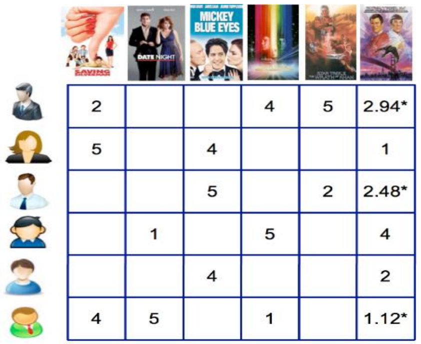


Are there anomalies?

Where should we look for oil?



Who are influencers?


MICKEY BLUE EYES
DATE NIGHT

2			4	5	2.94*
5		4			1
		5		2	2.48*
	1		5		4
		4			2
4	5		1		1.12*

Which item should I recommend?

How is our product perceived?

Emerging problem in our product?

Can we prevent insurance claims?

Classical programming …

TASKs like sorting an array

1,4,6,5,2,3 → 1,2,3,4,5,6

are easy to formalize, relatively easy to implement, and straightforward to test.

Array arr is sorted if and only if

$$\mathrm{arr}[i] \geq arr[i+1]\ \forall i\ \{0,..\mathrm{length}(\mathrm{arr}) - 2\}.$$

Write a program that takes an array as input and returns sorted array containing the same elements.

Traditional approach is not always feasible

Write a program that detects whether there is a dog in an image …

```
bool hasDog(Image im) {
    return random()>0.5;
}
```

Some **TASK**s are difficult to formalize.

Why do we need machine learning?

Object recognition

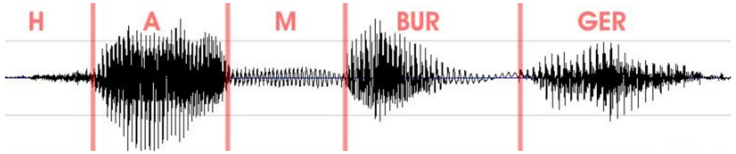
Product recommendation

HAMLET
WILLIAM SHAKESPEARE
CliffsNotes
They're still going to know you didn't read the book

OR

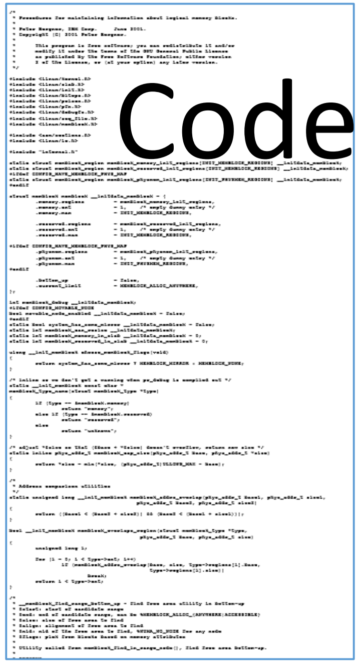
HAMLET
WILLIAM SHAKESPEARE
Beowulf
On the Island of Harty in Kent
Paul Wilkinson

Speech-to-Text



Election result prediction

Traditional Programming Approach

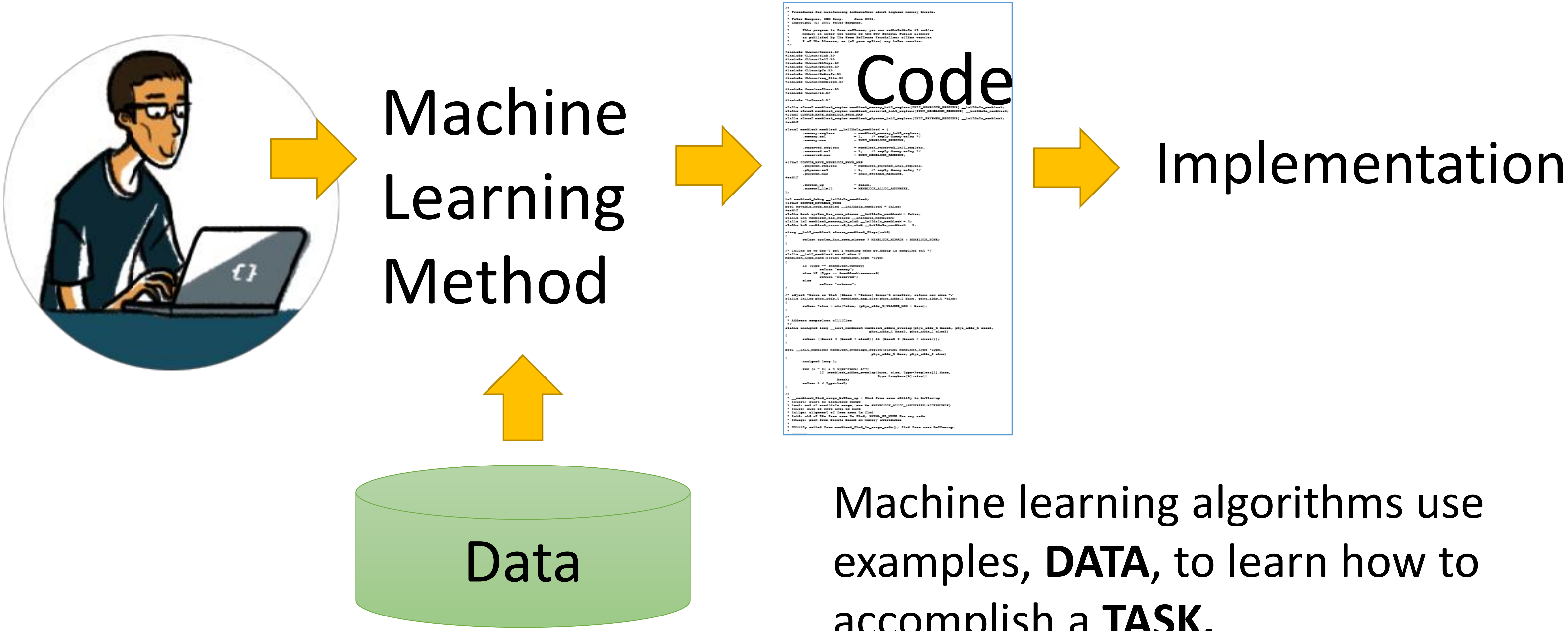


```
/*
 * Procedures for maintaining information about logical memory blocks.
 *
 * Peter Bergner, IBM Corp.     June 2001.
 * Copyright (C) 2001 Peter Bergner.
 *
 *      This program is free software; you can redistribute it and/or
 *      modify it under the terms of the GNU General Public License
 *      as published by the Free Software Foundation; either version
 *      2 of the License, or (at your option) any later version.
 */

#include <linux/kernel.h>
#include <linux/slab.h>
#include <linux/init.h>
#include <linux/bitops.h>
#include <linux/poison.h>
#include <linux/pfn.h>
#include <linux/debugfs.h>
#include <linux/seq_file.h>
#include <linux/memblock.h>

#include <asm/sections.h>
#include <linux/io.h>

#include "internal.h"

static struct memblock_region memblock_memory_init_regions[INIT_MEMBLOCK_REGIONS] __initdata_memblock;
static struct memblock_region memblock_reserved_init_regions[INIT_MEMBLOCK_REGIONS] __initdata_memblock;
#ifdef CONFIG_HAVE_MEMBLOCK_PHYS_MAP
static struct memblock_region memblock_physmem_init_regions[INIT_PHYSMEM_REGIONS] __initdata_memblock;
#endif

struct memblock memblock __initdata_memblock = {
        .memory.regions         = memblock_memory_init_regions,
        .memory.cnt             = 1,    /* empty dummy entry */
        .memory.max             = INIT_MEMBLOCK_REGIONS,

        .reserved.regions       = memblock_reserved_init_regions,
        .reserved.cnt           = 1,    /* empty dummy entry */
        .reserved.max           = INIT_MEMBLOCK_REGIONS,

#ifdef CONFIG_HAVE_MEMBLOCK_PHYS_MAP
        .physmem.regions        = memblock_physmem_init_regions,
        .physmem.cnt            = 1,    /* empty dummy entry */
        .physmem.max            = INIT_PHYSMEM_REGIONS,
#endif

        .bottom_up              = false,
        .current_limit          = MEMBLOCK_ALLOC_ANYWHERE,
};

int memblock_debug __initdata_memblock;
#ifdef CONFIG_MOVABLE_NODE
bool movable_node_enabled __initdata_memblock = false;
#endif
static bool system_has_some_mirror __initdata_memblock = false;
static int memblock_can_resize __initdata_memblock;
static int memblock_memory_in_slab __initdata_memblock = 0;
static int memblock_reserved_in_slab __initdata_memblock = 0;

ulong __init_memblock choose_memblock_flags(void)
{
        return system_has_some_mirror ? MEMBLOCK_MIRROR : MEMBLOCK_NONE;
}

/* inline so we don't get a warning when pr_debug is compiled out */
static __init_memblock const char *
memblock_type_name(struct memblock_type *type)
{
        if (type == &memblock.memory)
                return "memory";
        else if (type == &memblock.reserved)
                return "reserved";
        else
                return "unknown";
}

/* adjust *size so that (*base + *size) doesn't overflow, return new size */
static inline phys_addr_t memblock_cap_size(phys_addr_t base, phys_addr_t *size)
{
        return *size = min(*size, (phys_addr_t)ULLONG_MAX - base);
}

/*
 * Address comparison utilities
 */
static unsigned long __init_memblock memblock_addrs_overlap(phys_addr_t base1, phys_addr_t size1,
                                       phys_addr_t base2, phys_addr_t size2)
{
        return ((base1 < (base2 + size2)) && (base2 < (base1 + size1)));
}

bool __init_memblock memblock_overlaps_region(struct memblock_type *type,
                                        phys_addr_t base, phys_addr_t size)
{
        unsigned long i;

        for (i = 0; i < type->cnt; i++)
                if (memblock_addrs_overlap(base, size, type->regions[i].base,
                                           type->regions[i].size))
                        break;
        return i < type->cnt;
}

/*
 * __memblock_find_range_bottom_up - find free area utility in bottom-up
 * @start: start of candidate range
 * @end: end of candidate range, can be %MEMBLOCK_ALLOC_{ANYWHERE|ACCESSIBLE}
 * @size: size of free area to find
 * @align: alignment of free area to find
 * @nid: nid of the free area to find, %NUMA_NO_NODE for any node
 * @flags: pick from blocks based on memory attributes
 *
 * Utility called from memblock_find_in_range_node(), find free area bottom-up.
 *
```

Code

Implementation

Machine Learning Approach



Machine learning algorithms use examples, **DATA**, to learn how to accomplish a **TASK.**

Machine Learning – a toy example

Learning to detect dogs in images.

Data: {(, 1), (,0), (,0) …}

Each example is composed of an image and a label (, 1)

Typically we denote such pairs as $(\boldsymbol{x}, y)$ where $\boldsymbol{x}$ is a **feature** vector and y a **label.**

Feature vector **-> [0.25,0.3,….]** vector of color intensities

Label **1**

The **TASK**: predict label y from feature vector $\boldsymbol{x}$

Machine Learning – a toy example

Data: {(, 1), (,0), (,0) …}

The TASK: predict label y from feature vector $\boldsymbol{x}$

A machine learning algorithm produces a function $f(\cdot)$ such that $f(\boldsymbol{x})$ frequently matches the true label y.

Machine Learning – a toy example

Machine learning algorithm produces a function $f(\boldsymbol{x})$ such that $f(\boldsymbol{x})$ frequently matches the true label y.

$\boldsymbol{x}^1$= y^1=1 $= f(\boldsymbol{x}^1) = 1$

$\boldsymbol{x}^2$= y^2=0 $= f(\boldsymbol{x}^2) = 0$

$\boldsymbol{x}^3$= y^3=0 $\neq f(\boldsymbol{x}^3) = 1$

Q: How do we measure performance of function $f(\boldsymbol{x})$?

Q: Let’s say that 75% of the images in the dataset are dogs and 25% are cats. If $f(x) = 1$ (always predict that dog is in the image) how well are we doing?

Machine learning – a toy example

One way to measure performance of your prediction function $f(x)$ is to count fraction of mistakes.

$$\text{Error} = \frac{\sum_{\{t=1\}}^{T}[f(x^t) \neq y^t]}{T}$$

where

$$[v] = \begin{cases} 1, & \text{if v} = \text{true} \\ 0, & \text{otherwise.} \end{cases}$$

$\boldsymbol{x}^1$= y^1=1 $= f(\boldsymbol{x}^1) = 1$

$\boldsymbol{x}^2$= y^2=0 $= f(\boldsymbol{x}^2) = 0$

$\boldsymbol{x}^3$= y^3=0 $\neq f(\boldsymbol{x}^3) = 1$

$\text{Error} = 1/3$

Machine Learning – a toy example

Machine learning algorithm produces a function $f(\boldsymbol{x})$ such that $f(\boldsymbol{x})$ frequently matches the true label y.

We want to find a function that minimizes the error.

Direct optimization of the count of mistakes is very hard – there is no feedback on how to adjust the function to improve prediction.

Machine Learning – probabilistic view

Instead of making hard predictions our prediction functions can compute probabilities

$$P(y = 1|x, \theta)$$

Probability that there is a dog in an image given pixels

$P(y = 1|$ $, \theta) = 0.75, P(y = 1|$ $, \theta) = 0.25, \ldots$

Since our predictions are no longer discrete we can get back feedback on how to adjust them.

Probabilistic view enables use of optimization, and it allows us to quantify uncertainty.

Recap: Tasks, Data, Optimization

TASKs are defined implicitly through DATA. For example,

Data = {(, 1), (,0), (,0) …}

Machine Learning Algorithms minimize ERROR to find a mapping $f(\boldsymbol{x})$ that performs well on the TASK.

Optimization and probabilistic view of the learning allows us to accomplish this.

Unsupervised Machine Learning

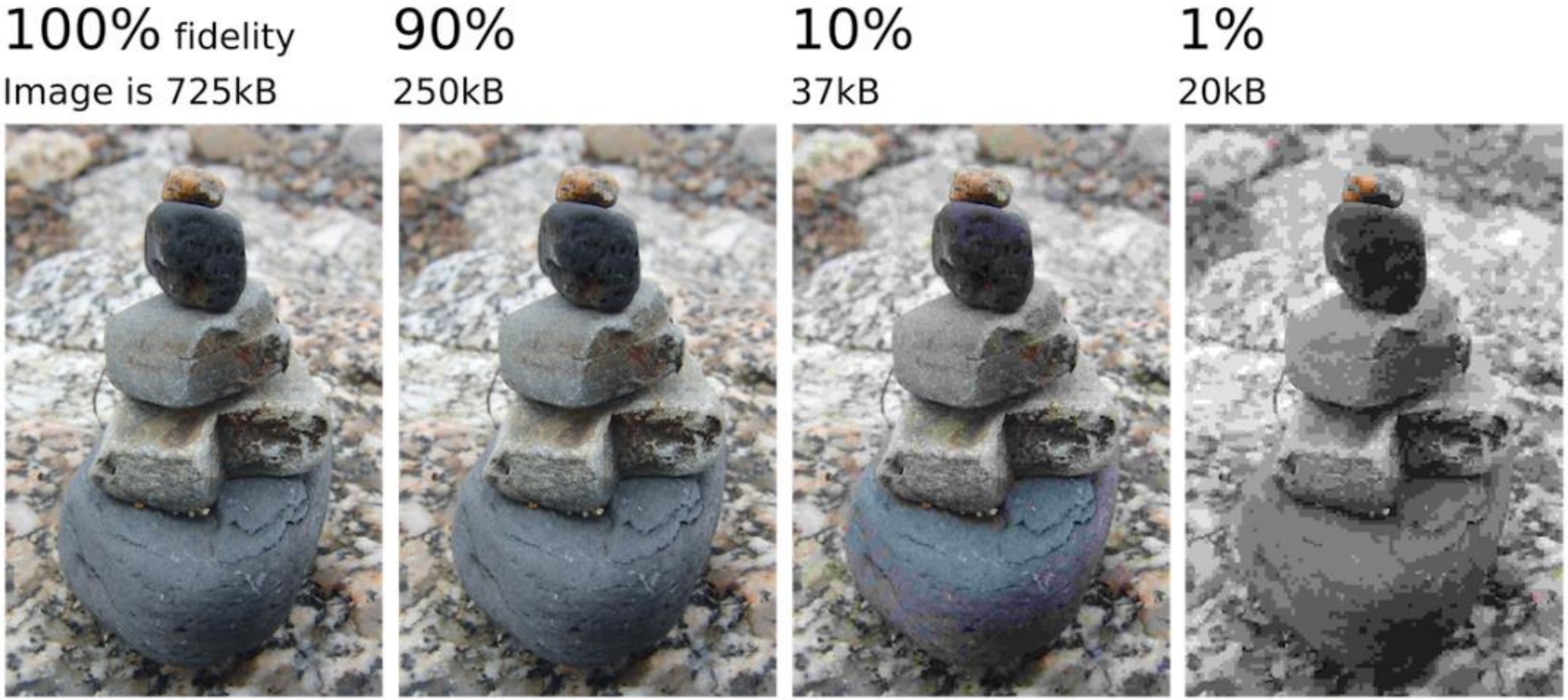
TASKs thus far have had a defined input and output (, 0)

This is called SUPERVISED Machine Learning.

UNSUPERVISED Learning does not have a pre-specified output.

Unsupervised Machine Learning

UNSUPERVISED learning seeks to find compact data representations from which original data can be reconstructed.



Unsupervised Machine Learning

Compression is an example of **UNSUPERVISED** machine learning.

ZIP, RAR, BZIP2 find a **REPRESENTATION** for the data.

More formally find function such that:

$\mathrm{h} = f(x)$ such that $\mathrm{f}^{-1}(h) \approx x$ (Lossy reconstruction)

$\mathrm{h} = f(x)$ such that $\mathrm{f}^{-1}(h) = x$ (Lossless reconstruction)

Here, h **REPRESENTS** the **DATA** x.

Error for Unsupervised Learning

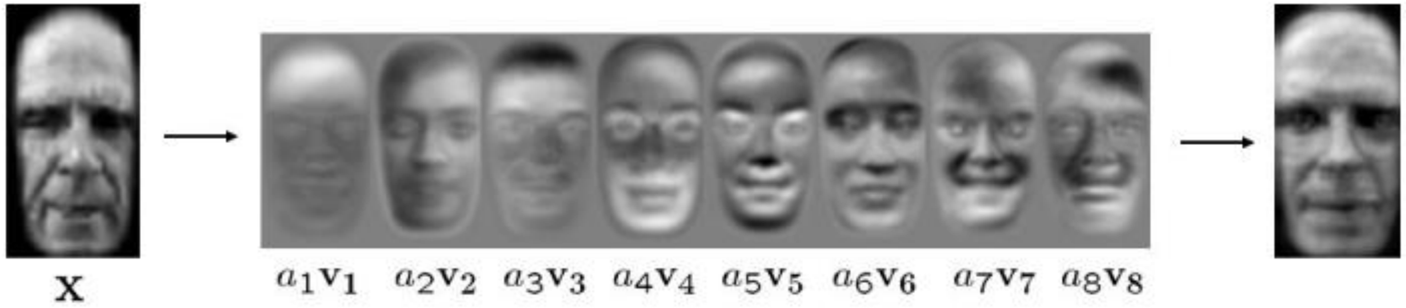
In unsupervised learning reconstruction error is optimized:

$$\text{Error } = \frac{1}{T}\sum_{\{t=1\}}^{T} \text{Distance}(\text{f}^{-1}\big(f(x^t)\big), x^t)$$

Typically the Distance is a smooth function enabling optimization.

Examples of Unsupervised Learning

Dimensionality reduction

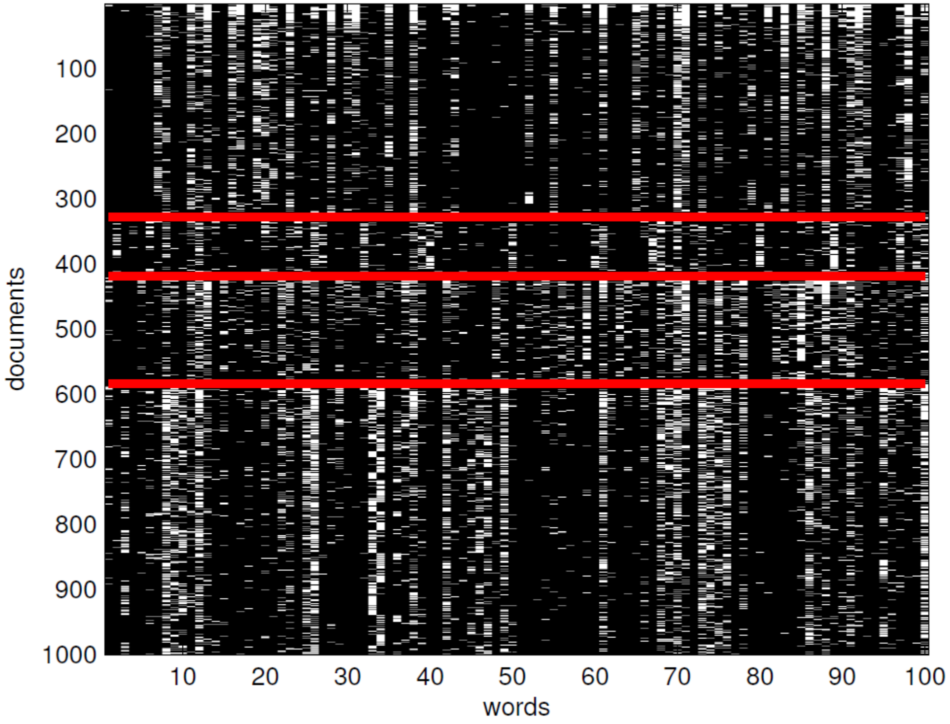


A face on the left is represented by eight real values corresponding to weight of each of the eight components.

Reconstructed face is shown on the right.

Examples of Unsupervised Learning

Clustering documents



Row is a text document; Column is a word.

White means that the document contains the word.

Red lines separate clusters of documents with similar vocabulary.

You can represent each document with the indicator of its cluster (1,2,3 or 4).

Pitfalls in machine learning

DATA is used to learn mappings $f(\cdot)$ by minimizing errors:

$$\text{Error } = \frac{\sum_{\{t=1\}}^{T} [f(x^t) \neq y^t]}{T}$$

$$\text{Error } = \frac{1}{T} \sum_{\{t=1\}}^{T} \text{Distance}(\mathrm{f}^{-1}\left(f(x^t)\right), x^t)$$

Q: Given two machine learning methods how do we tell which one is better?

Training Data

Imagine the following scenario:

1) Teacher works out a quadratic equation $x^2 - 1 = 0$ in class
2) Teacher gives final exam with one of the problems being

$$x^2 - 1 = 0$$

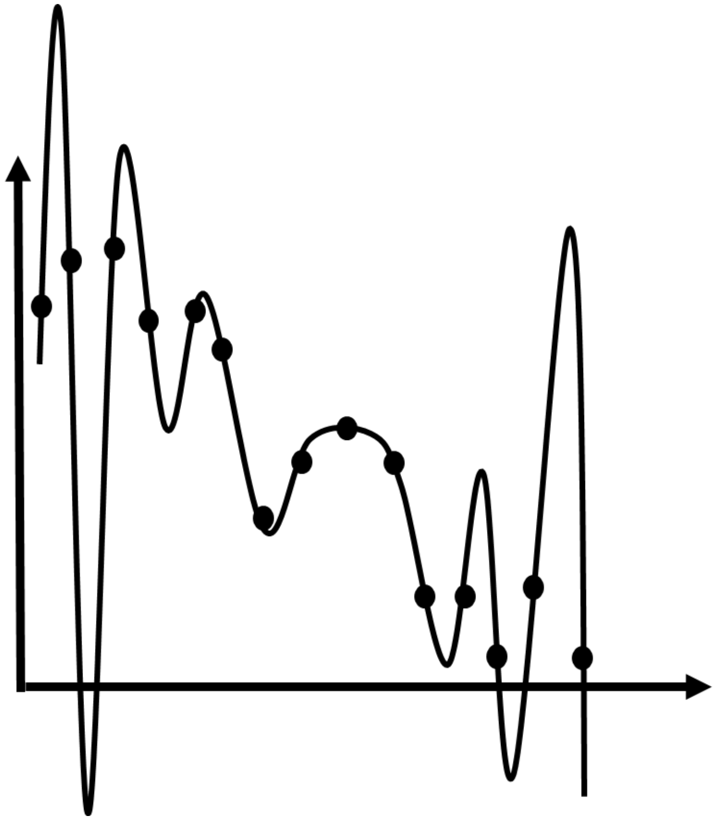
Q: Is this a good idea? Is this a bad idea? What should the teacher do?

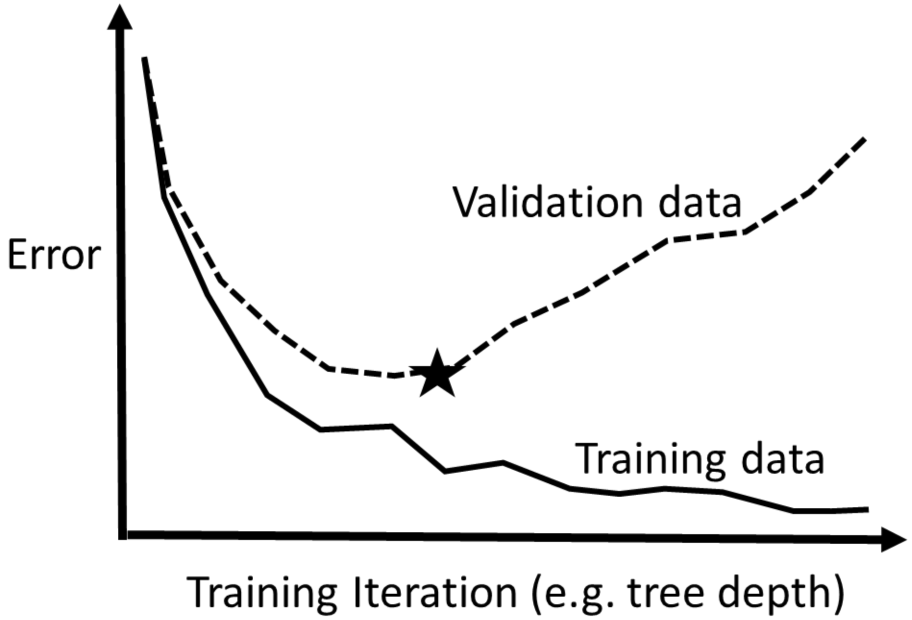
Q: Would training a machine learning algorithm run into the similar problem? How should we fix it?

Q: Given two machine learning methods how do we tell which one is better?

Best practice for machine learning

- Do not overfit!





Pitfalls

From the reading assignments:

- (1) Statistics in the big data era: Failures of the machine: http://dx.doi.org/10.1016/j.spl.2018.02.028

- (2) Adversarial Examples that Fool both Computer Vision and Time-Limited Humans: https://arxiv.org/pdf/1802.08195.pdf

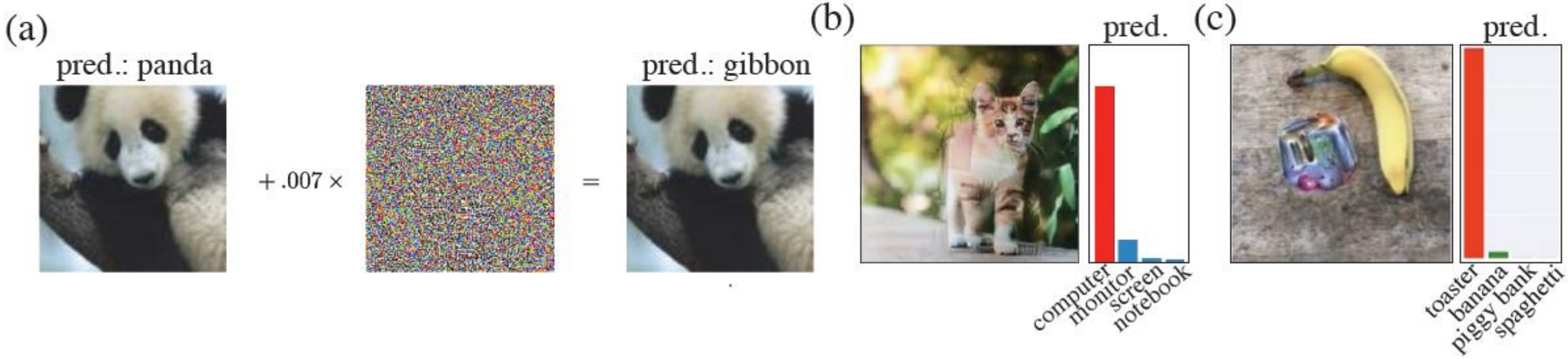
Algorithmic bias: from (1)

2.2. Fair decisions and predictive algorithms

There are numerous decisions that are made by various authorities based on their own judgment and experience; these decisions can have an enormous impact on society as a whole and on individuals. Some examples include whether and how to regulate car emissions, patrol locations and decisions of who to stop and search in policing, sentencing and bail decisions in the criminal justice system, hiring decisions, salary levels, selection of grants to fund, and tenure decisions in academics.

Of course whenever there is an individual or a small group of individuals in charge of making such decisions, there is substantial room for the decisions to not be entirely ‘‘fair’’ and objectively based on the data at hand but instead driven in part by implicit or explicit biases. Such biases may lead to under-regulation of pollution, policing that targets certain minority communities, more severe sentencing for individuals within those communities, and hiring/salary/tenure decisions driven in part by demographic factors […]

Adversarial attacks: from (2)



Today

Recap:

1. In machine learning, data implicitly encodes tasks
2. Supervised learning discovers a mapping from features to labels
3. Supervised learning error is measured by comparing mapped output to the true output
4. Unsupervised learning finds compact representations of the data
5. Training and test data should be kept separate to enable accurate assessment of the method’s performance