Math 7243--Machine Learning and Statistical Learning Theory

Section 1 Introduction

Instructor: He Wang
Department of Mathematics
Northeastern University

- **Prerequisite:**

1. Linear Algebra
2. Multivariant calculus
3. Probability and Statistics
4. Basic programming skills

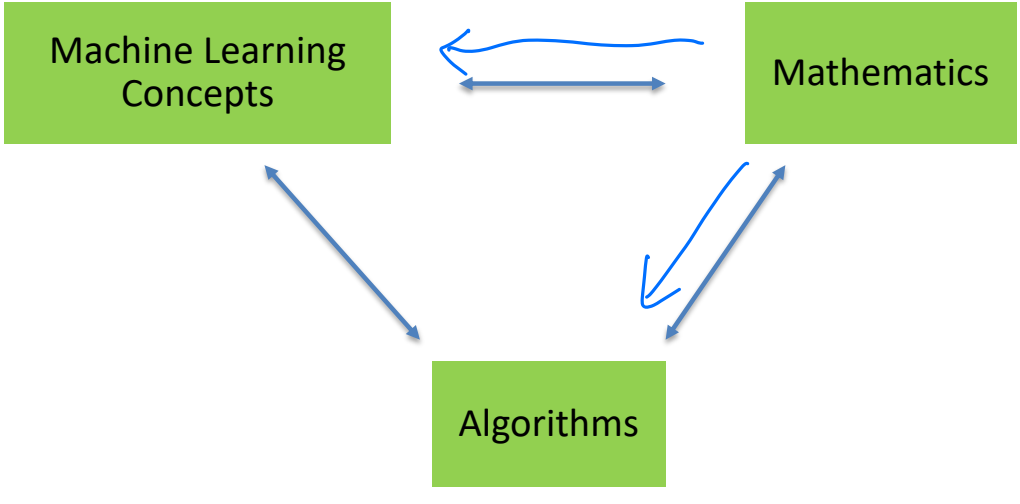
before 19 ∞

Algebraic Topology

Topological Data Analysis (TDA)

• Jose Perea

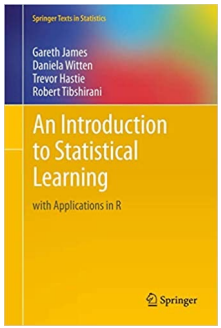
- **Goal of this class:**

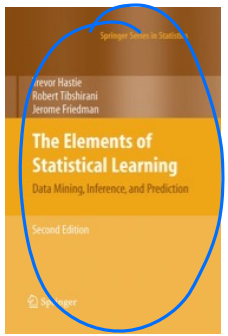


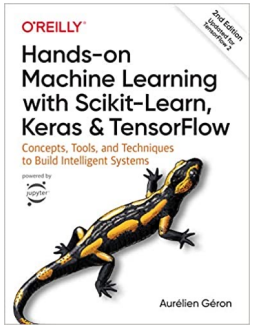
Recommended Textbooks:

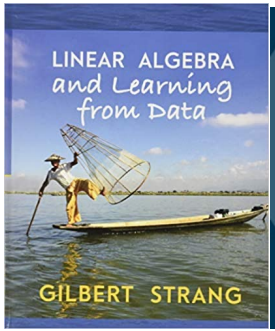
1. *An Introduction to Statistical Learning, with applications in R, by G. James, D. Witten, T. Hastie, R. Tibshirani.* https://www.statlearning.com/
2. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* by Trevor Hastie, Robert Tibshirani, Jerome Friedman https://web.stanford.edu/~hastie/ElemStatLearn/
3. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* - by Aurélien Géron
4. *Linear algebra and learning from data* by Gilbert Strang
5. *Machine Learning: A Probabilistic Perspective* by Kevin P. Murphy
6. *Pattern Recognition and Machine Learning* by Christopher Bishop

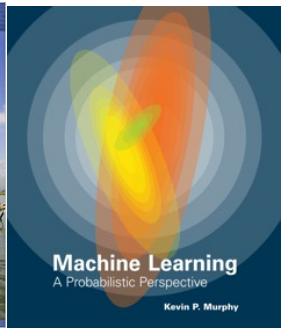
My recommended book list https://docs.google.com/spreadsheets/d/1j8LcyTxBTZ5Nh0-P1rb6NT2x_2iXFp1j8ptpvHYQWGA/edit?usp=sharing

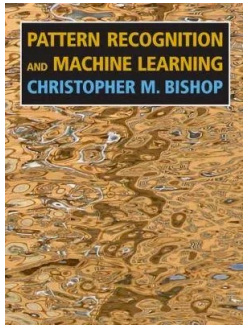












More online sources:

Relevant materials will be linked or uploaded to our Canvas page.

Nate Bade taught this course before at Northeastern:
https://tipthederiver.github.io/Math-7243-2020/index.html

There are several ML open courses from Stanford/Cornell/BU/MIT/Carnegie Mellon, etc.
https://cs229.stanford.edu/syllabus.html
http://www.cs.cornell.edu/courses/cs4780/2018fa/page18/
http://math.bu.edu/people/mkon/MA751/index.html
https://people.eecs.berkeley.edu/~jrs/189/
https://www.cs.cmu.edu/~tom/10701_sp11/lectures.shtml
http://people.csail.mit.edu/dsontag/courses/ml16/
http://introtodeeplearning.com/2021/index.html
http://www.mit.edu/~9.520/fall19/
...
Tremendous lecture materials/code sources are on github or other online websites.
I will add more extra notes on Canvas along with the class process.

Grades distribution:

Written Homework (25%) - There will be three written assignments which will focus on theory.

Labs (30%) - There will be roughly 6 labs. Labs will focus on the implementation of algorithms on real world data sets. Class time will be allotted for labs, but students may finish labs at home. In each lab, we will fit a real world data set using the algorithms of techniques introduced in that weeks' theory lecture. Labs will be graded out of 10 pts, 6 pts for completion, 4 pts for communication. There will also be a standing bonus of 2 pts for going above and beyond and exploring an interesting aspect of the parameter space, or getting a really good fit.

Midterm (15%)

Final Project (30%) - The final project will consist of a proposal (1 page), middle stage progress report(2-3 pages), project report (roughly 5 pages depending on the format) and presentation (roughly 20 minutes with poster or slides). Project groups should contain 3-5 people.

More on the **project**:

- Students are encouraged to participate in our in-class XN project.
  (More XN project see https://careers.northeastern.edu/experiential-network/)
- If you would like to propose your own project, it can take one of three forms:
  1. A computational analysis of a data set using sufficiently complicated or novel techniques from this course.
  2. A theoretical presentation of a topic not covered in this course with a case study.
  3. Thesis or Lab project.

Excellent project can consider to submit the poster to RISE
(https://www.northeastern.edu/rise/about/)
I am more than happy to discuss possible projects in any of these categories with you.

Rough project timeline: (Exact due day and time will be announced on Canvas.)

- Group Selection: As early as possible
- Project Proposal Deadline: Early Feb
- Milestone progress Report(extended proposal): Mid March
- First Draft: Mid April
- Final Draft: Late April
- Presentations: Last class

The project can be any topic relating to the class, however it can't be something you did it the end of the semester. It can't be a project you did in another course.

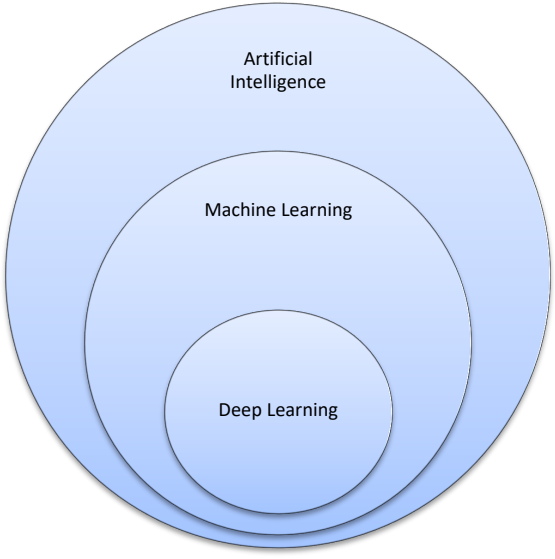
➢ Python Programing:

This course requires Python, Jupyter Notebook Server and github, all of which are free and open source.

After today's class:

1. Install **Python** from here: https://www.python.org/about/gettingstarted/
2. Install **Jupyter Notebook** from here: https://jupyter.org/
3. Try the tutorial examples using Jupyter-Notebook in the lab on Canvas.
4. Register a github account. https://github.com/

- Homework/Lab/Project Questions can be asked on Piazza.
- You should ask/answer a question with your name.
- When you sent me an email, please tell me which section (2:50 or 7:40) are you in.
- Teaching Assistant: Yujia Shi <shi.yuji@northeastern.edu>
- Office Hours: see Canvas/Syllabus



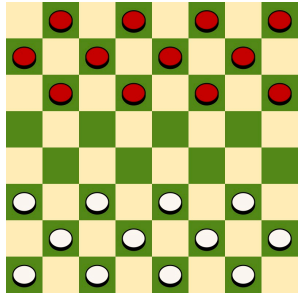
Artificial intelligence (AI) is wide-ranging branch of computer science concerned with building smart machines capable of performing tasks that typically require human intelligence.

Patrick Winston, the Ford professor of artificial intelligence and computer science at MIT, defines AI as "algorithms enabled by constraints, exposed by representations that support models targeted at loops that tie thinking, perception and action together."

Definition of Machine Learning

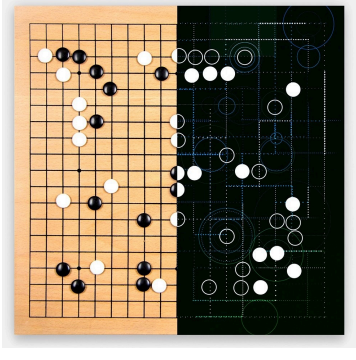
Arthur Samuel (1959): Machine Learning is the field of study that gives the computer the ability to learn without being explicitly programmed.

(Samuel's Checkers Player Program.)



Tom Mitchell (1998): 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.

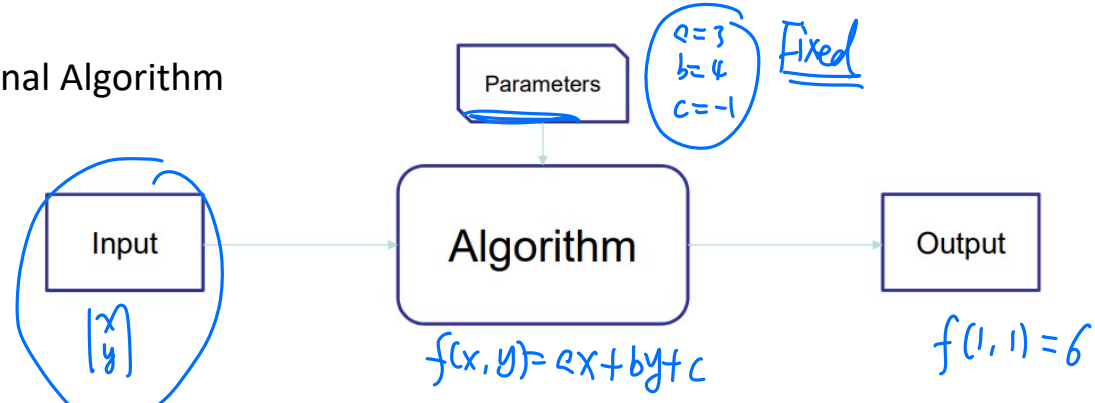
Experience E (data): games played by the program (with itself).
Task T: decisions the software need to make.
Performance measure P: winning rate.



AlphaGo

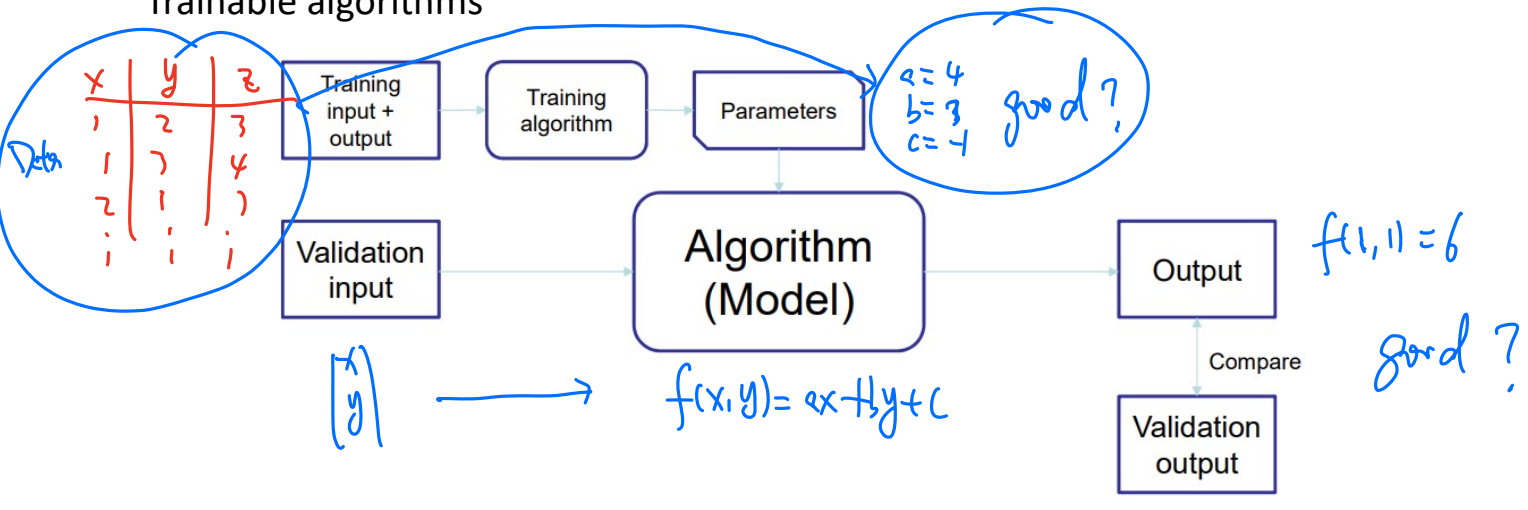
Algorithms that improve on some tasks with experience.

Traditional Algorithm



M.L.

Trainable algorithms



➢ **1. Supervised Learning:**

Given a sample set of **labeled** data, can we predict the labels on new unlabeled data from the same domain?

- Regression Examples: predict house price, predict birth rate, etc.
- Classification Examples: Image classification, classification by features, Fraud detection, Email Spam Detection, etc.

➢ **2. Unsupervised Learning:**

Given a set of **unlabeled** data, can we find structure within the data?

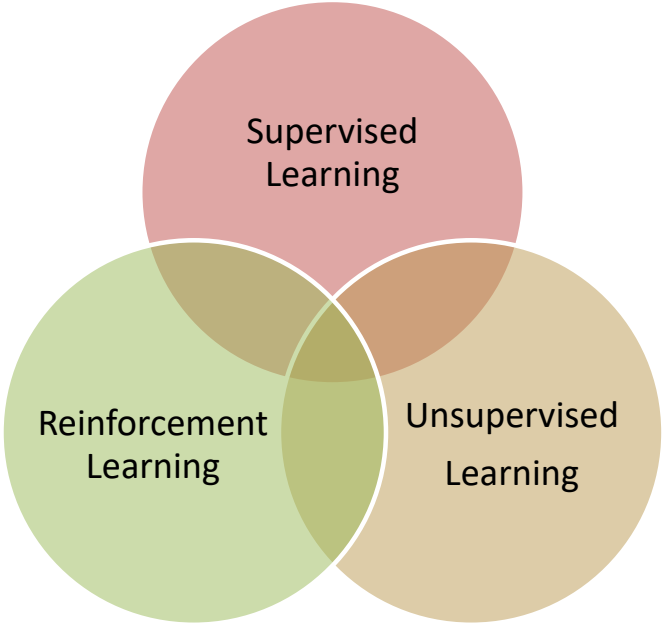
- Clustering Examples: Biology
- Dimensional reduction: face/image recognition, big data visualization.

➢ **3. Reinforcement Learning:**

An agent that performs certain actions in an environment so as to maximize the reward.

- Examples: Gaming, Robot Navigation, Finance, etc.

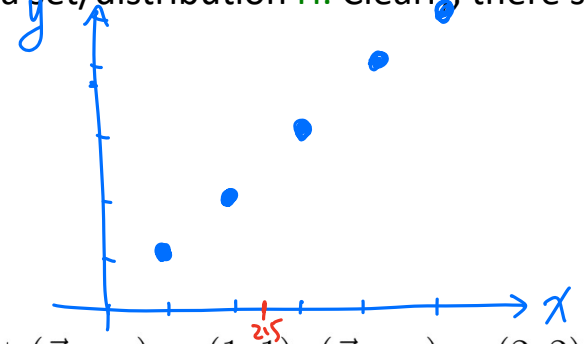
1. Supervised Learning (Most in our course)
2. Unsupervised Learning (a few in our course)
3. Reinforcement Learning (for future study)



- **No free lunch theorem.**

The No Free Lunch Theorem states that every successful Machine Learning (ML) algorithm must make assumptions. This also means that there is no single ML algorithm that works for every setting.

Every ML algorithm has to make **assumptions** on which hypothesis class H should you choose? This choice depends on the data, and encodes your assumptions about the data set/distribution H. Clearly, there's no one perfect H for all problems.



- Example:

Assume that $(\vec{x}_1, y_1) = (1, 1)$, $(\vec{x}_2, y_2) = (2, 2)$, $(\vec{x}_3, y_3) = (3, 3)$, $(\vec{x}_4, y_4) = (4, 4)$, $(\vec{x}_5, y_5) = (5, 5)$.

Question: what is the value of y if $\vec{x} = 2.5$?

- $y = 2.5$ ⇐ Assume "linear" $y = x$ ($y = ax + b$)
- $y = 0$ ⇐ Assume $y \neq x$ for integers, 0 for others
- $y =$ ⇐ Assume $y = ax^2 + bx^2 + cx + d$

1. Supervised Learning

Data is pre-categorized or numerical.

- Training Data Set : $\mathcal{D} = \{(\vec{x}^{(i)}, y^{(i)}) \,|\, i = 1 \ldots n\} \subset \mathbb{R}^d \times \mathcal{C}$
- Label Space: $\mathcal{C} = \mathbb{R}$, or {0,1}, or {-1, 1}, or {1,2,3,...}

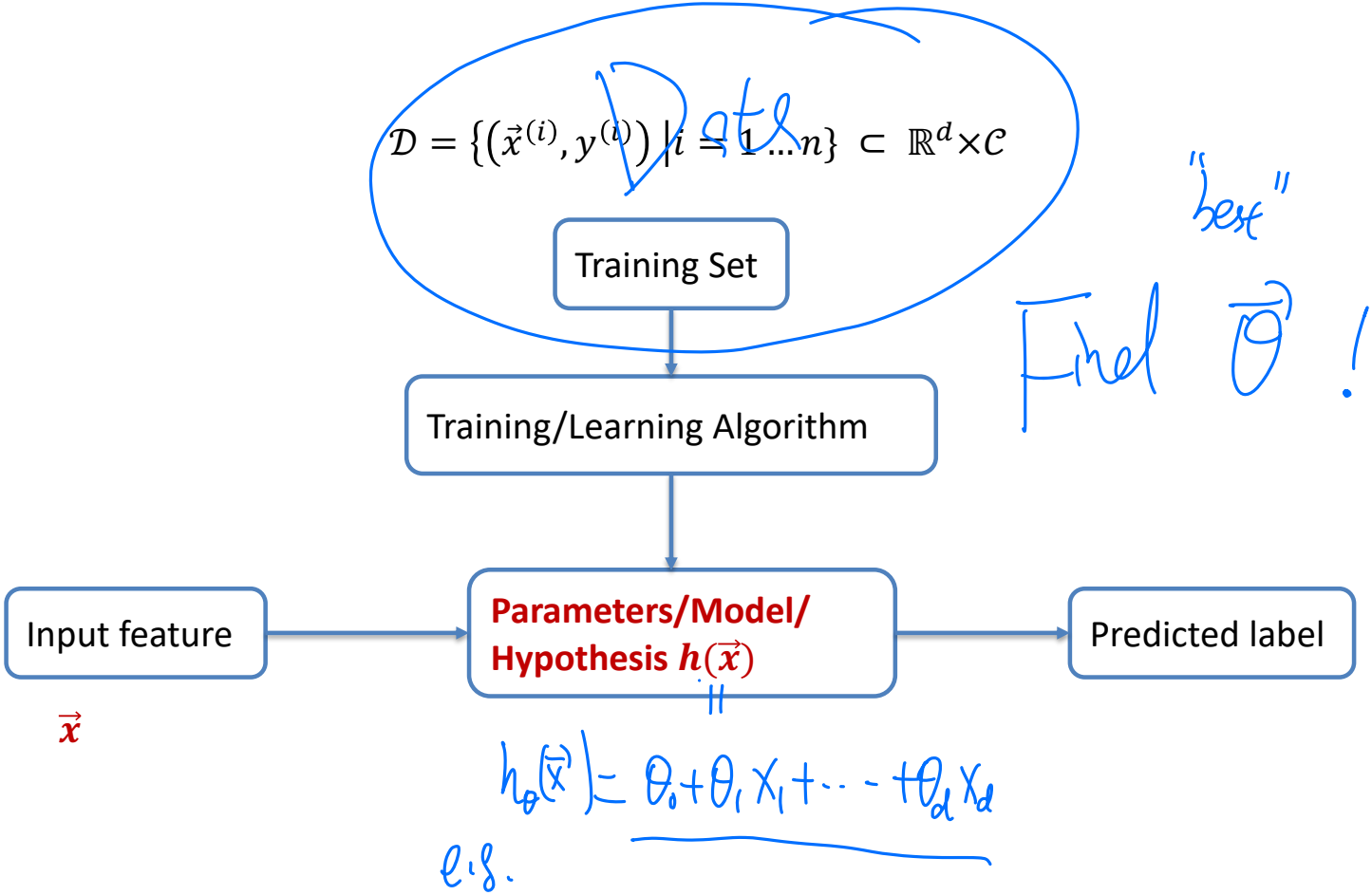
The goal in supervised learning is to **make *predictions from data*.**

- **Regression:** Predict a number. Predict house price, future temperature or the height of a person, etc.

Quantitative Variable $y \in \mathbb{R}$: Variables that take quantitative values, with some values larger than others and close values designating similar characteristics. Also called numerical.

- **Classification:** Predict a category. (Spam or not spam, dog or cat or ,)

Qualitative Variable $y \in$ {0,1}, or {-1, 1}, or {1,2,3,...}: Variables that only take discrete values, usually in a set of descriptive classes. Also called categorical or discrete variables, or factors. There could be no additional ordering or structure on the classes.


$\mathcal{D} = \{(\vec{x}^{(i)}, y^{(i)}) | i = 1 \ldots n\} \subset \mathbb{R}^d \times \mathcal{C}$
Data
Training Set
Training/Learning Algorithm
Input feature
$\vec{x}$
Parameters/Model/
Hypothesis $h(\vec{x})$
Predicted label
"best"
Find $\vec{\theta}$!
$h_\theta(\vec{x}) = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$
e.g.

➢ **Regression**:

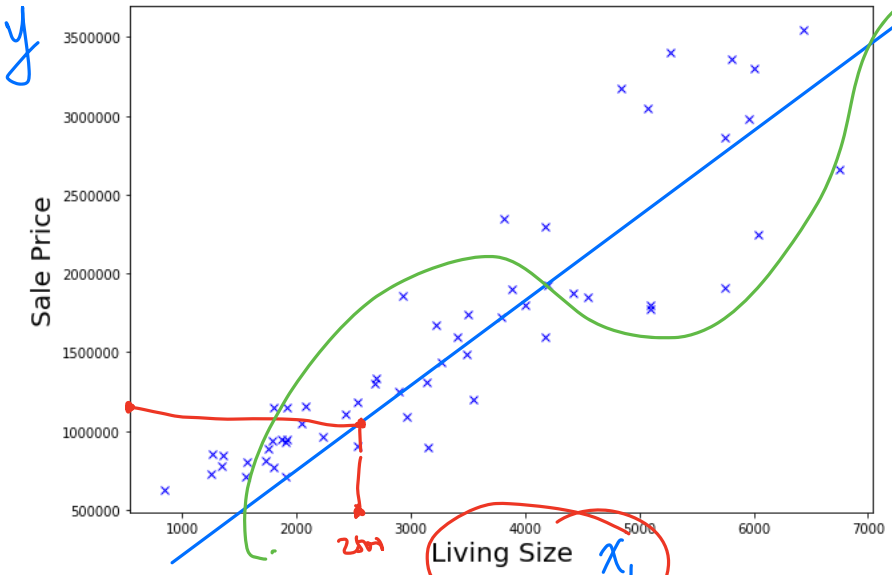
- Data Set: $\mathcal{D} = \{(\vec{x}^{(i)}, y^{(i)}) \mid i = 1 \dots n\} \subset \mathbb{R}^d \times \mathcal{C}$
- Label space $\mathcal{C} \subset \mathbb{R}$

Goal: Find a good model $h(\vec{x})$ from the data.

➢ Example: Predict house price.

Input: a dataset that contains n samples with d=1.

Task: if a house has x square feet, predict its price?



Assumption:

$h_{\vec{\theta}} = \theta_0 + \theta_1 x_1$ or $h_{\vec{\theta}} = \theta_0 + \theta_1 x_{i1}$

$+\theta_2 x_1^2$

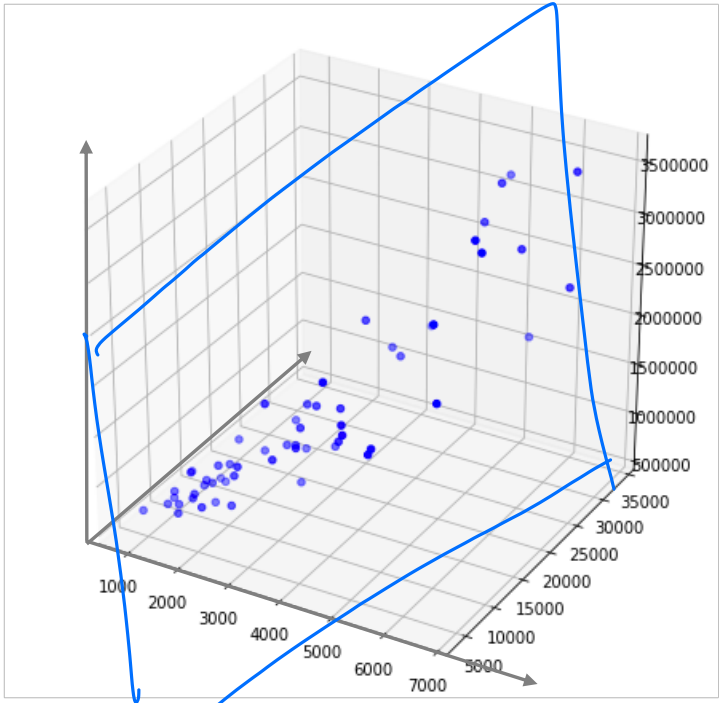
$+\theta_3 x_1^3$

Find the "best" $\vec{\theta}$.

Input: a dataset that contains n samples $\mathcal{D} = \{(\vec{x}^{(i)}, y^{(i)}) \,|\, i = 1 \ldots n\} \subset \mathbb{R}^2 \times \mathcal{C}$

Task: if a house has x_1 feet^2 living size and x_2 feet^2 lot size, predict its price?

$\vec{x} \in \mathbb{R}^2$



Assume $h_{\vec{\theta}}(\vec{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

- Find $\vec{\theta}$ from data.

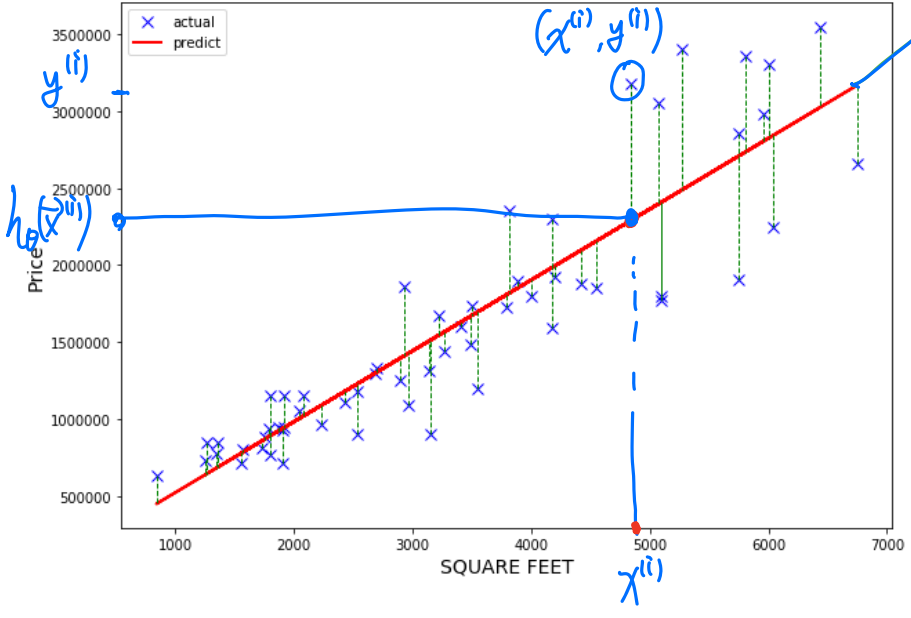
- Predict house price.

BEDS (x_1)	BATHS (x_2)	LOCATION (x_3)	SQUARE_FEET (x_4)	LOT_SIZE (x_5)	YEAR_BUILT (x_6)	PRICE (y)
3	3	Newton	2969	15014	1967	1090000
3	2.5	Newton	1566	5582	1922	805000
4	2.5	Newton Corner	2532	6273	1953	905000
7	4.5	Newton Center	6748	26607	1902	2660000
4	4	West Newton	4200	20446	2007	1925000
4	2.5	Newton	2232	3966	1870	965000
2	1.5	Newton Corner	1344	5559	1851	775000
3	2.5	Newton	2898	12420	1943	1250000
2	2	West Newton	1729	4171	1953	815000
6	3	West Newton	3149	12616	1953	900000
5	3.5	West Newton	4000	12006	1912	1800000
4	3.5	West Newton	6430	30600	1920	3550000
4	1.5	Auburndale	1750	8222	1893	885000
2	2	Newton	840	5548	1955	630000
…	….	…	…	…	…	…

Input: a dataset that contains n samples $\mathcal{D} = \{(\vec{x}^{(i)}, y^{(i)}) \,|\, i = 1 \ldots n\} \subset \mathbb{R}^6 \times \mathcal{C}$.

Task: predict its price, if a house has $\vec{x}$

- **Evaluation: Cost/Loss/Error Functions**



True.

$(x^{(i)}, y^{(i)})$

$y^{(i)}$

$h_\theta(\vec{x}^{(i)})$

$h_\theta(\vec{x})$ || e.g. $\theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$

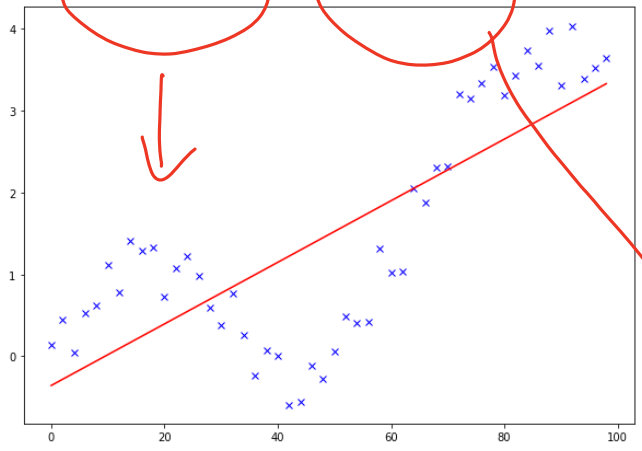
$x^{(i)}$

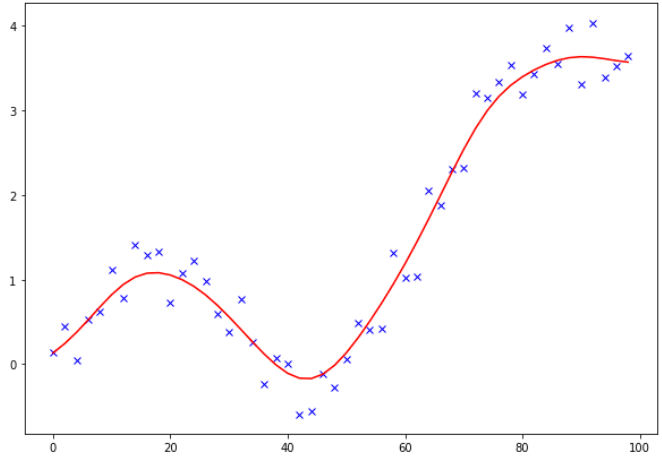
e.g.

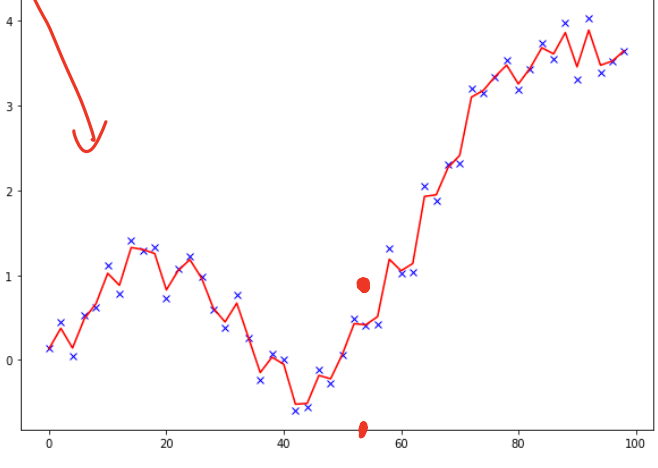
$$L(\vec{\theta}) = \sum_{i=1}^{n} \left[y^{(i)} - h_{\vec{\theta}}(\vec{x}^{(i)}) \right]^2$$

Goal: Find $\vec{\theta}$ to Minimize $L(\vec{\theta})$

- Underfitting v.s. Overfitting. --- Bias-Variance Tradeoff







➢ Supervised Classification.

- Training Data Set : $\mathcal{D} = \{(\vec{x}^{(i)}, y^{(i)}) \,|\, i = 1 \ldots n\} \subset \mathbb{R}^d \times \mathcal{C}$
- Label Space: $\mathcal{C}$ = {0,1}, or {-1, 1}, or {1,2,3,...}

$P(y=1 \mid \begin{bmatrix} 8 \\ 8 \end{bmatrix}) = 0.9$

$\vec{x} = \begin{bmatrix} 8 \\ 8 \end{bmatrix}$

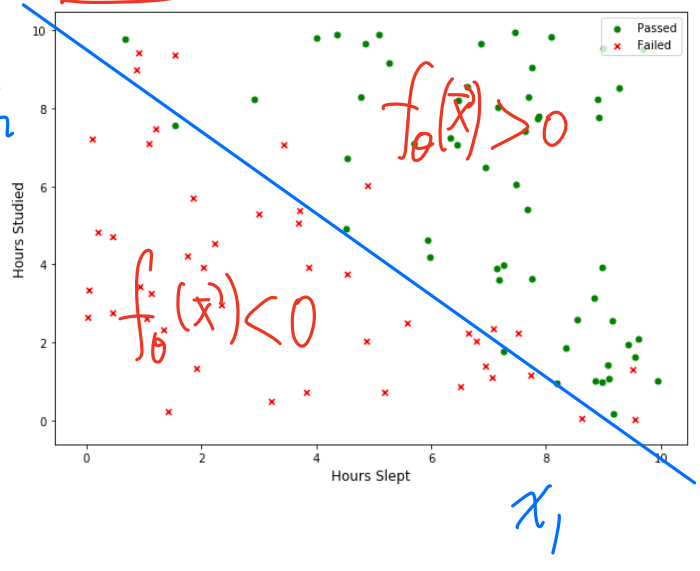
· Test $\vec{x}$ · Want $P(y=1 \mid \vec{x})$

$f_\theta(\vec{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_2^2 + \ldots = 0$ Boundary

1. **Logistic Regression** Method:

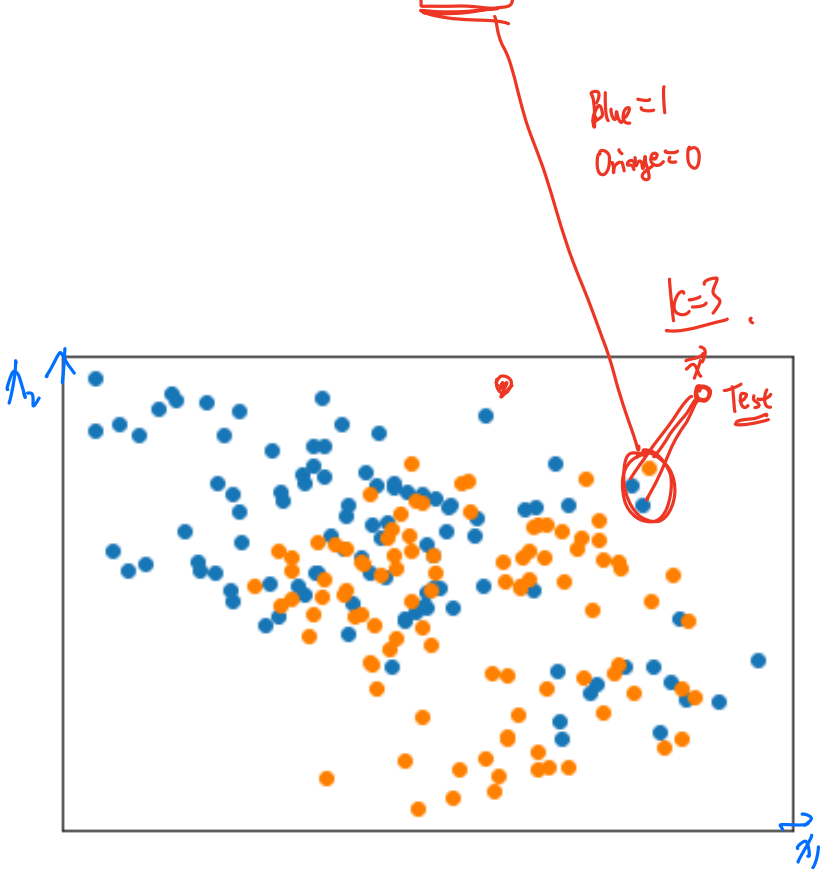
Assume:

$$P(y=1 \mid \vec{x}) = \frac{1}{1 + e^{-f_\theta(\vec{x})}}$$

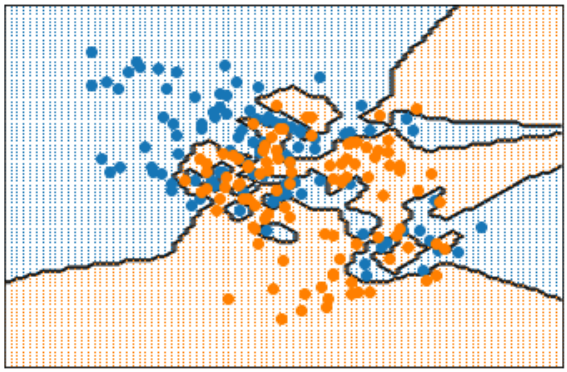


2. **k-nearest neighbors** method:

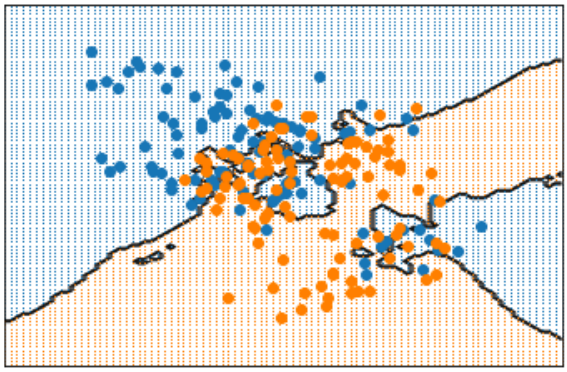
$$\hat{Y}(\vec{x}) = \frac{1}{k} \sum_{x_i \in N_k(\vec{x})} y_i$$



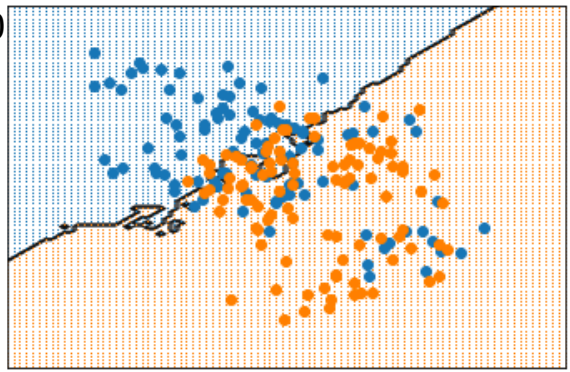
k=1



k=10



k=50



- More binary classification:

Spam filtering. Here, an email (the data instance) needs to be classified as *spam* or *not-spam*.

Dear Good Friend

I am Abdoul Issouf, I work for BOA bank Ouagadougou Burkina Faso. I have a business proposal which concerns the transfer of ($13.5 Million US Dollars) into a foreign account. Everything about this transaction shall be legally done without any problem. If you are interested to help me, Please keep this transaction as a Top Secret to your self till the Money get into your account in your Country OK. and I will give you more details as soon as I receive your positive response. You will be Entitled to 50%, 50% will be for Me If you are willing to work with me send me immediately the information listed bellow.

Your Name .....

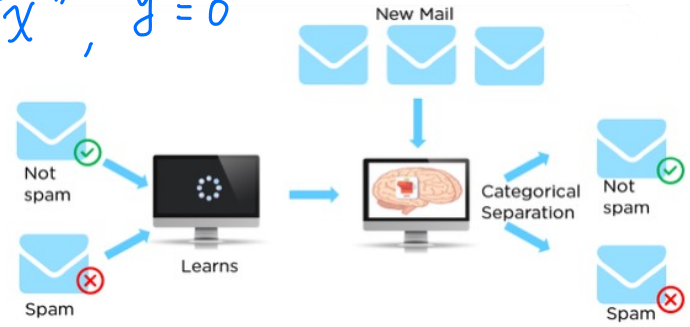
Your Nationality .....

Your Age .....

Female Or Male .....

Your Occupation .....

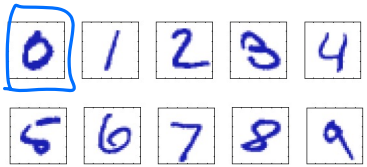
Your Private Telephone.....



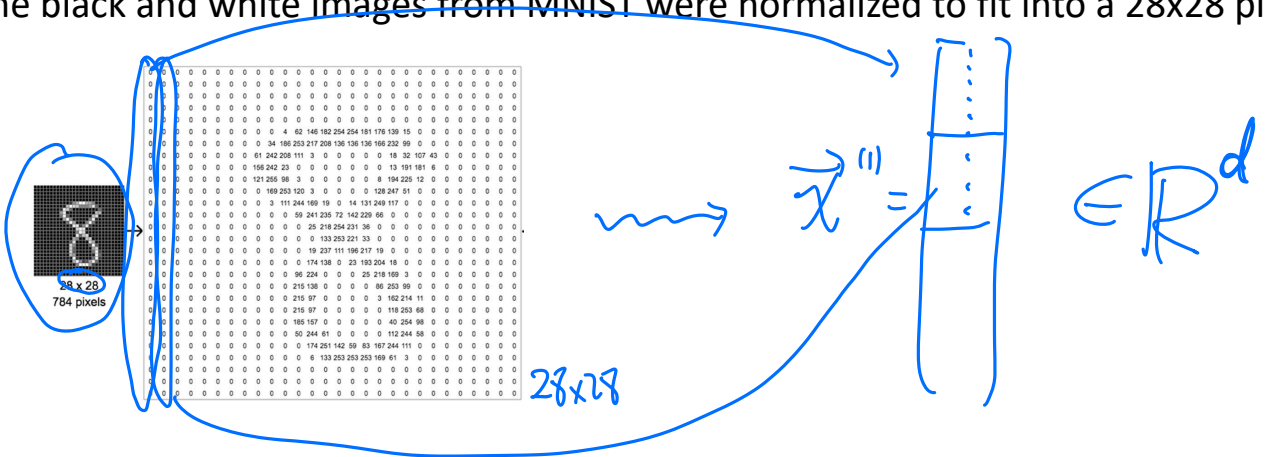
We will represent an email via a feature vector, whose length d is equal to the number of words in the dictionary.

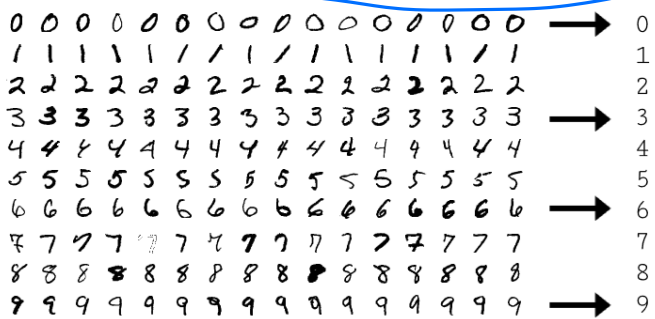
- Multi classes classification. Examples of hand-written digits taken from US zip codes.

The MNIST (Modified National Institute of Standards and Technology) data set of handwritten numbers. It contains 60,000 training images and 10,000 testing images.



The black and white images from MNIST were normalized to fit into a 28x28 pixels.





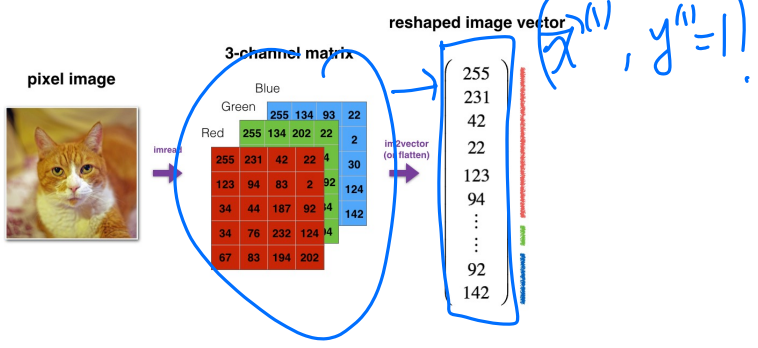
- Multi-class classification: Image Classification

Relationship → "Dog" = 0

Relationship → "Cat" = 1

Relationship → "Rabbit" = 2

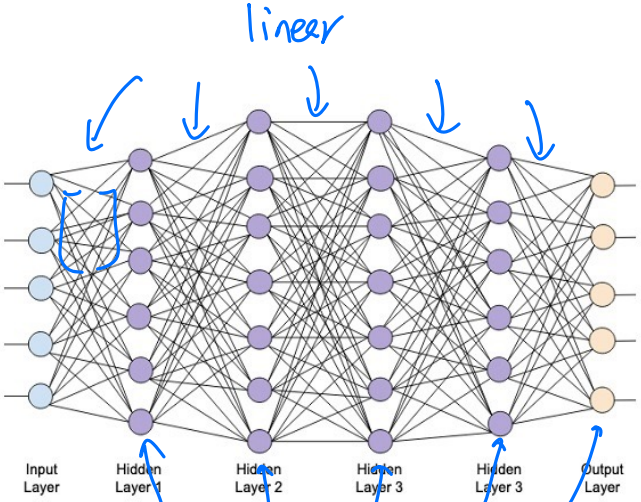
The features represent pixel values.



Artificial Neural Network

Human Neural Networks was introduced in 1943 by neurophysiologist Warren McCulloch and mathematician Walter Pitts to model neurons in the brain using electrical circuits.

Artificial Neural networks are a series of algorithms that mimic the operations of a human brain to recognize relationships between vast amounts of data. it's a very broad term that encompasses any form of Deep Learning model.

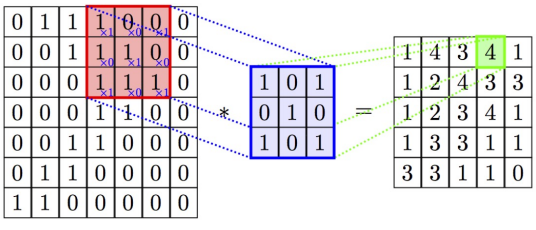


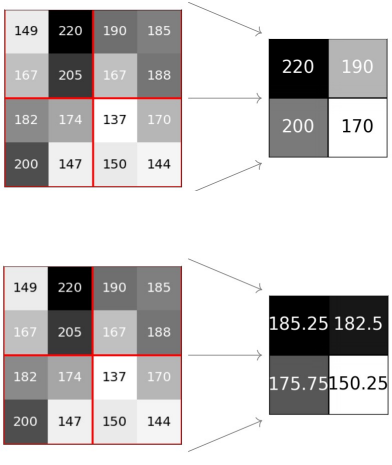
$$f(z) = \frac{1}{1+e^{-z}}$$

or $$f(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$$

- **Convolution Neural Networks (CNN)**

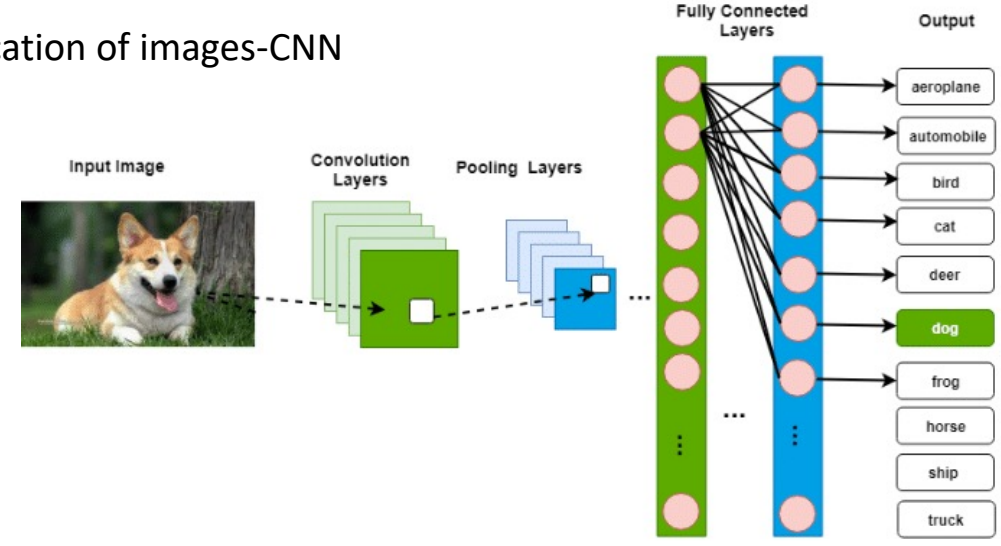
  CNNs are a specific type of neural networks that are generally composed of convolution layers and pooling layers.



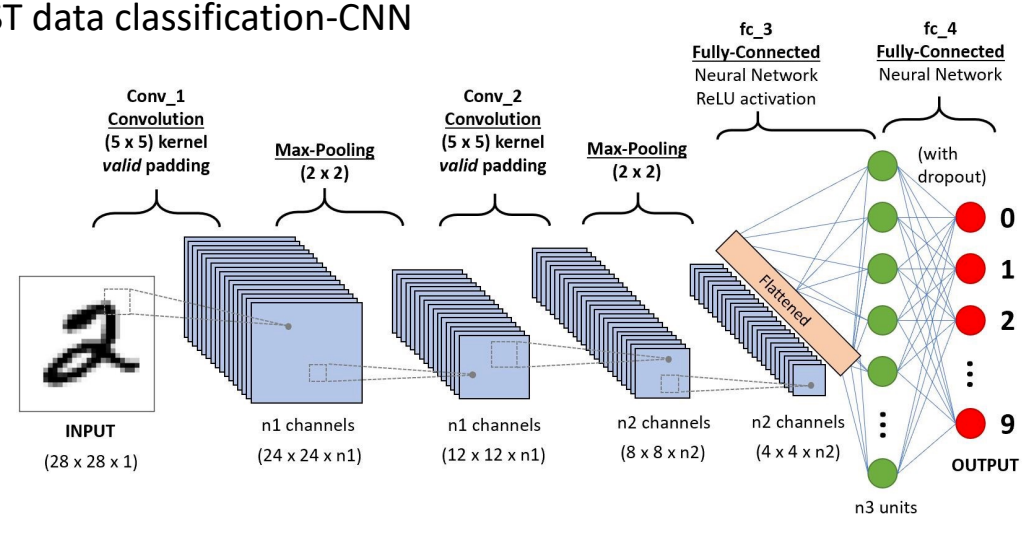


An Interactive Node-Link Visualization of Convolutional Neural Networks
https://www.cs.ryerson.ca/~aharley/vis/conv/

- Classification of images-CNN

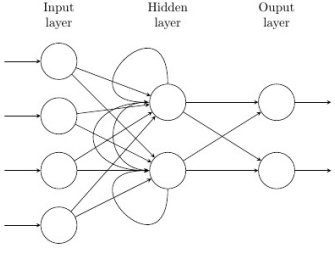


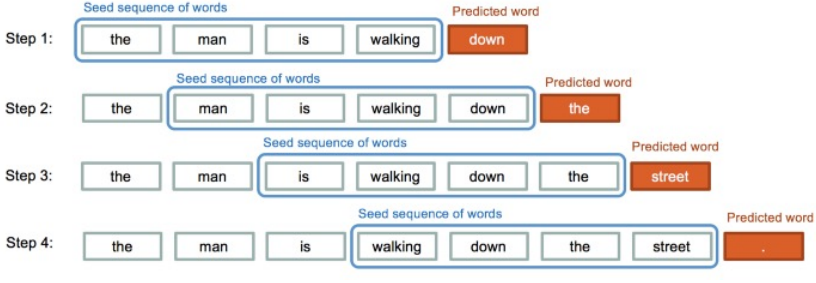
- MNIST data classification-CNN

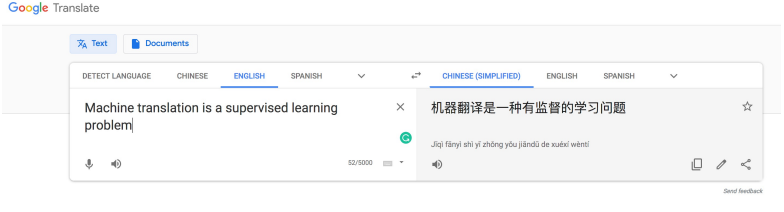


- **Recurrent Neural Networks (RNN)**

In RNN, each of the neurons in hidden layers receives an input with a specific delay in time. It allow previous outputs to be used as inputs while having hidden states. (RNN usually has a short-term memory. Long-term memory is also used in some research problems.)



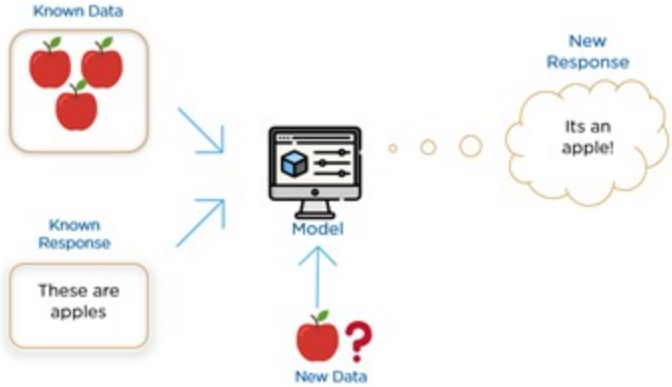


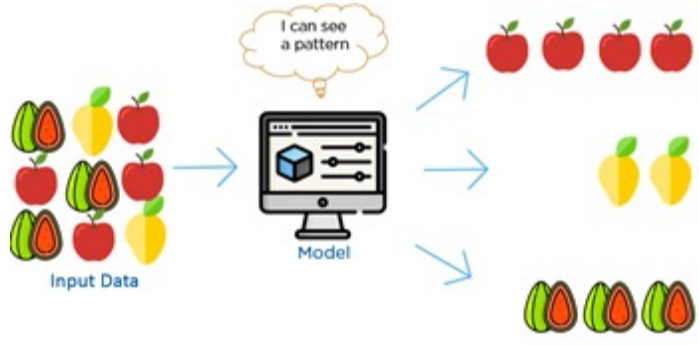


Other supervised learning methods in our class:

1. Linear/quadratic discriminant analysis
2. Naive Bayes
3. Tree based methods (random forest)
4. Support Vector Machine (SVM)
5. Kernel methods
6. etc.

Supervised Learning v.s. Unsupervised Learning

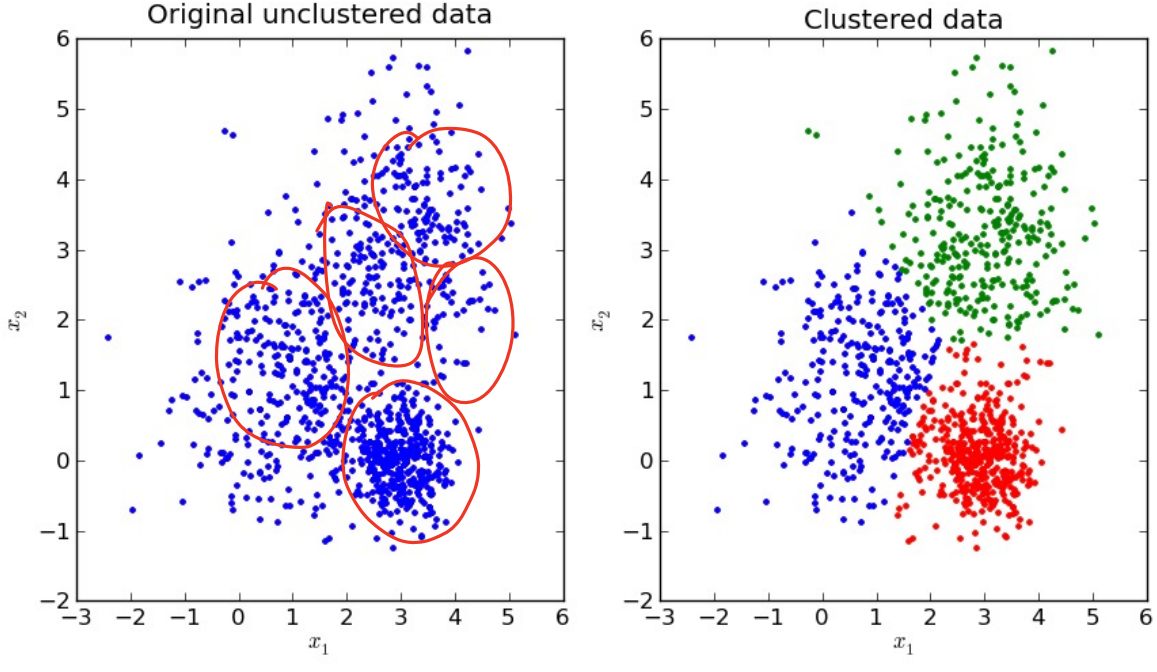




2. Unsupervised Learning

- Clustering

k-mean clustering



- principal component analysis (PCA)

$$\vec{x} = \begin{bmatrix} x_1 \\ x_i \\ \vdots \\ x_d \end{bmatrix} \in \mathbb{R}^d \qquad d = 1000$$

$$\updownarrow$$

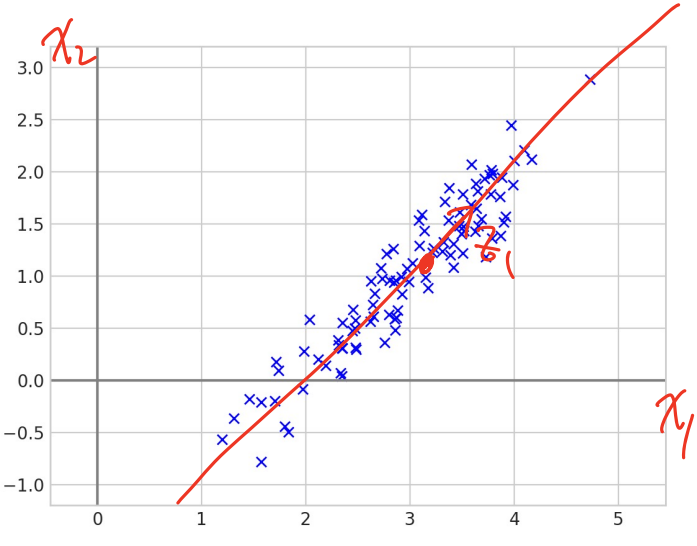
$$\mathbb{R}^3$$

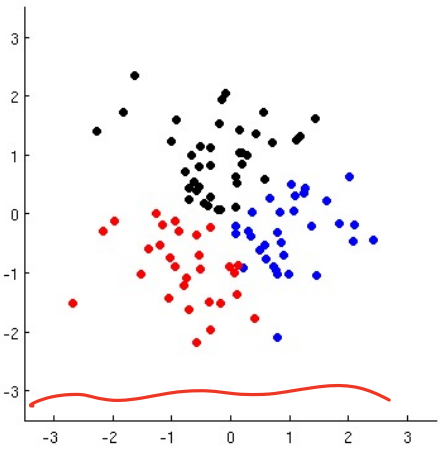
$z_1 = c_1 x_1 + c_i x_i + \dots + c_d x_d$

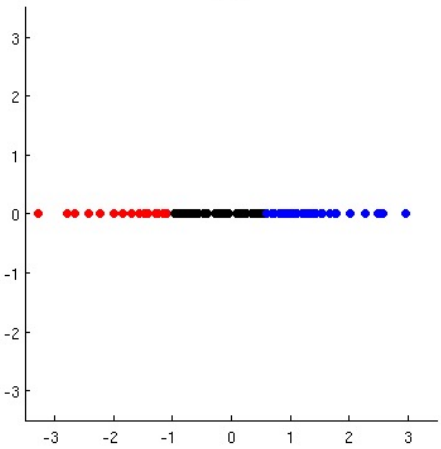
z_2

z_3

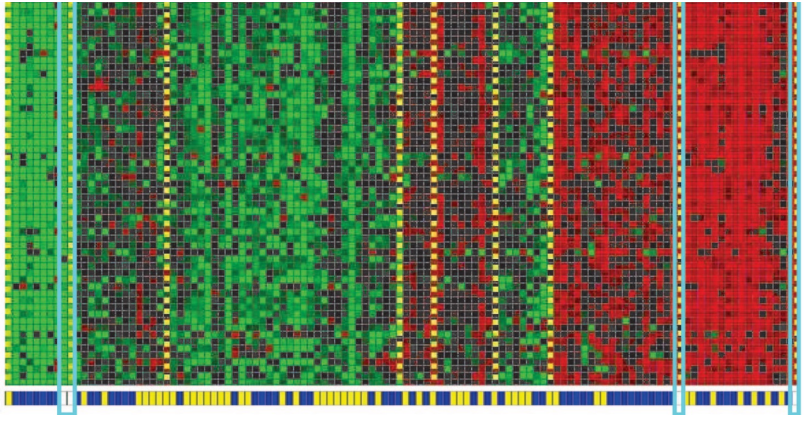
z_{100}

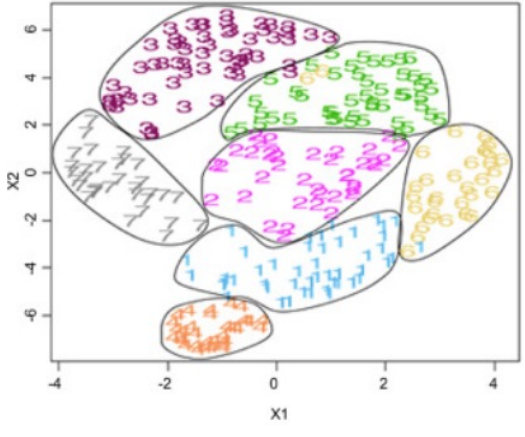






Application: Clustering Genes

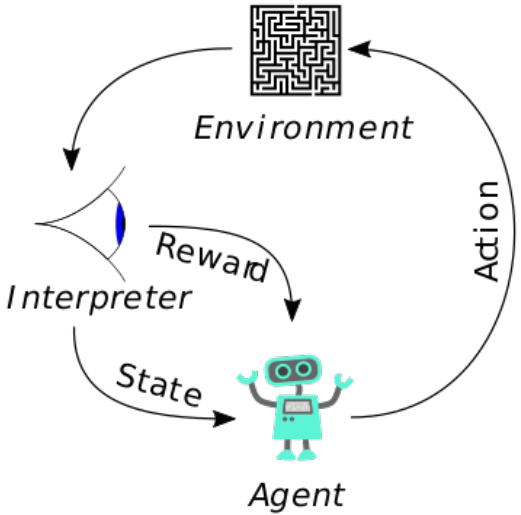




3.Reinforcement Learning (For future study)

Reinforcement Learning enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences. An agent that performs certain actions in an environment so as to maximize the reward.

Examples: Robotics, Games, self-driving cars, etc.



- **Agent:** It is an assumed entity which performs actions in an environment to gain some reward.
- **Environment (e):** A scenario that an agent has to face.
- **Reward (R):** An immediate return given to an agent when he or she performs specific action or task.
- **State (s):** State refers to the current situation returned by the environment.

In our Math7243 class

- We will NOT do work like:

https://www.youtube.com/watch?v=fn3KWM1kuAw
https://www.youtube.com/watch?v=tF4DML7FIWk

- We will do work like:

Training Data $D = \{(\vec{x}^{(1)}, y^{(1)}), \ldots, (\vec{x}^{(n)}, y^{(n)})\}$

Model Assumption: $h(\vec{x}) = \vec{\theta}^T \vec{x} = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$

Find $\vec{\theta}$ to minimize $RSS(h) = \sum_{i=1}^{n} \left(h(\vec{x}^{(i)}) - y^{(i)}\right)^2$

Least Squares solution $\vec{\theta} = (X^T X)^{-1} X^T \vec{y}$

;ression
:y

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & x_d^{(n)} \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Lemma. The covariance matrix of $\vec{z} = A\vec{x}$ is

$$\mathrm{Cov}(\vec{z}) = A\ \mathrm{Cov}(\vec{x})\ A^T$$

Proof of the Proposition:

$$\begin{aligned} \mathrm{Cov}(\vec{\theta}) &= \mathrm{Cov}((X^T X)^{-1} X^T \vec{y}) \\ &= \mathrm{Cov}((X^T X)^{-1} X^T (X\theta_* + \epsilon)) \\ &= \mathrm{Cov}(\theta_* + (X^T X)^{-1} X^T \epsilon) \\ &= \mathrm{Cov}(\ (X^T X)^{-1} X^T \epsilon) \\ &= (X^T X)^{-1} X^T\ \sigma^2 I\ ((X^T X)^{-1} X^T)^T \\ &= \sigma^2 (X^T X)^{-1} \end{aligned}$$

Proof of the

$\mathrm{Cov}(\vec{z}) =$

=

=

=

jupyter Ex2StudyPass Last Checkpoint: 12/28/2020 (autosaved)

Edit View Insert Cell Kernel Widgets Help

Run Code

```
labels.shape = (len(features),1)
```

In [9]:
```
def plot_scatter(trues,falses):
    fig = plt.figure()
    ax1 = fig.add_subplot(111)

    ax1.scatter(trues['X1'],trues['X2'], s=25, c='g', marker="o", label='Passed')
    ax1.scatter(falses['X1'],falses['X2'], s=25, c='r', marker="x", label='Failed')
    #ax1.plot([x for x in range(0,11,1)],[x for x in range(11,0,-1)], c='b')
    plt.legend(loc='upper right');
    ax1.set_xlabel('Hours Slept')
    ax1.set_ylabel('Hours Studied')
    #ax1.grid(True)
    fig.set_size_inches(10, 7)
    #plot_decision_boundary()
    plt.show()
```

In [10]:
```
def plot_scatterl(trues,falses):
    fig = plt.figure()
    ax1 = fig.add_subplot(111)

    ax1.scatter(trues['X1'],trues['X2'], s=25, c='g', marker="o", label='Passed')
    ax1.scatter(falses['X1'],falses['X2'], s=25, c='r', marker="x", label='Failed')
    ax1.plot([x for x in range(0,11,1)],[x for x in range(11,0,-1)], c='b')  # guessed boundary
    plt.legend(loc='upper right');
    ax1.set_xlabel('Hours Slept')
    ax1.set_ylabel('Hours Studied')
    #ax1.grid(True)
    fig.set_size_inches(10, 7)
    #plot_decision_boundary()
    plt.show()
```

In [11]:
```
plot_scatter(trues,falses)
plot_scatterl(trues,falses)
```