

INTRODUCTION TO MACHINE LEARNING

RBT 350

Roberto Martin-Martin

Assistant Professor of Computer Science.

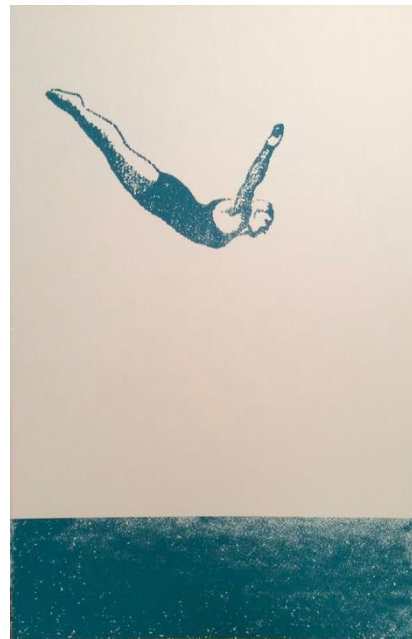
What will you learn today?

- Introduction to Machine Learning
 - What is Machine Learning?
 - Types of Machine Learning problems
 - Elements of a Machine Learning problem
 - Regression
 - Classification
- Next day
 - Clustering
 - Markov Decision Process & Reinforcement Learning
 - Revisiting Regression: Imitation Learning



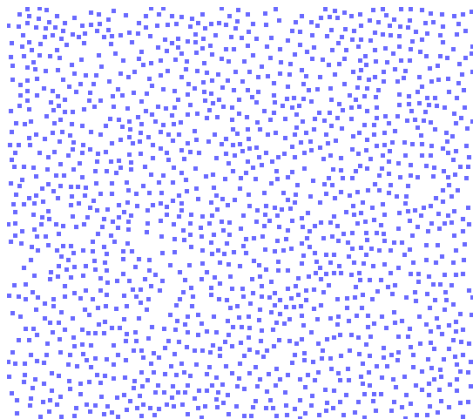
Before We Dive In...

- As with other topics in the class, this lecture is intended to introduce machine learning
- We won't be able to go into all details
- Take related courses/tutorials and read textbooks to learn this subject in depth

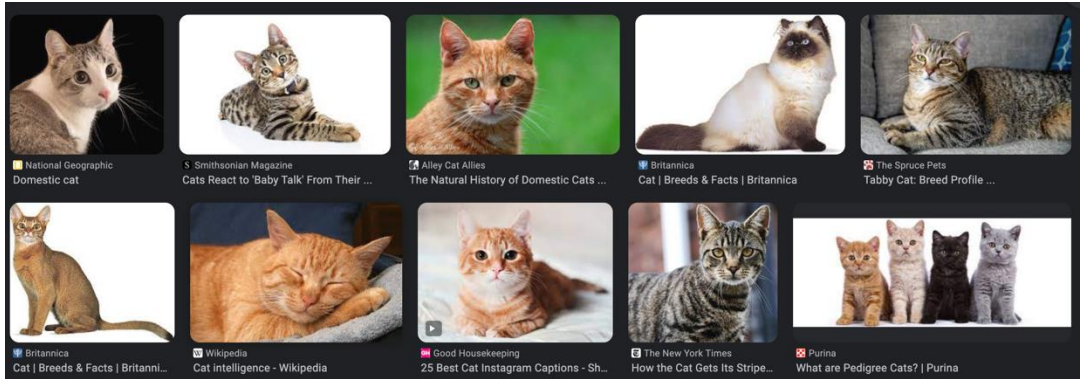


What is Machine Learning?

A set of automatic techniques to discover patterns in data



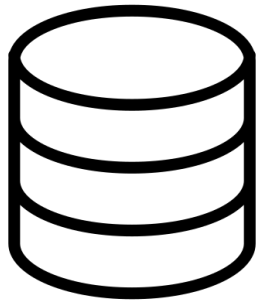
What is a pattern? What is a regularity?



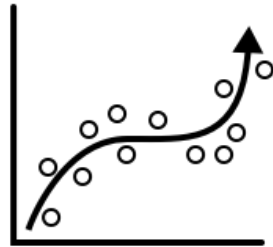
- $X=2, y=4$
- $X=3, y=6$
- $X=4, y=8$
- ...

What do we need to find patterns for?

To make predictions!



data

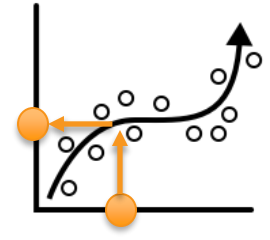
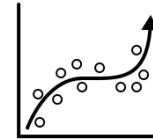


model



new data

+



predictions

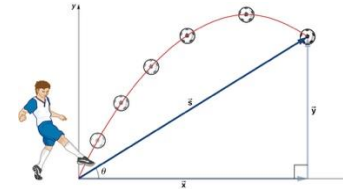
Why do we need ML in Robotics?

- Critical component of modern Artificial Intelligence
- What is AI?
 - The study of how to engineer intelligent systems/machines
- What is intelligence?
 - The ability to see and interpret visual input (CV)
 - The ability to read and understand language (NLP)
 - The ability to move and interact with the world (Robotics)
 - The ability to reason and perform logical deduction (GOFAI)



When **NOT** to Make Robots Learn?

Learning is not a solution to every problem in robotics



Repetitive, non-changing processes



We have very good models of physical processes
Do not “reinvent the wheel”!

When to Make Robots Learn?

Learning is critical to bring robots to unstructured environments



structured

unstructured

When to Make Robots Learn?

Learning is critical to bring robots to unstructured environments



unstructured



learning!



variable

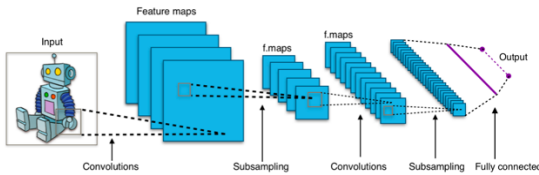


uncertain



dynamic

Great time for ML in Robotics



Artificial Intelligence

Recent breakthroughs in machine learning and computer vision, e.g., deep learning (Turing awards 2018)



Computing Power


Your smartphone is millions of times more powerful than all of NASA's combined computing in 1969.



Robot Hardware

More reliable and affordable cobot hardware that costs around annual salary of American workers

Types of Machine Learning Problems

- Supervised Learning
 - Algorithm learns from a dataset of annotated cases (,  ”)
 - Regression
 - Classification
- Unsupervised Learning
 - No annotated data is provided
 - Clustering
- Reinforcement Learning
 - Agent finds its own data → learns from successes and failures
 - Decision making in Markov Decision Processes (MDP)

Supervised Learning

- We have a dataset of annotated cases → Training data
 - Pairs of (data, label)
 - Examples:
 - (houseSize, price)
 - (image, class)
 - (countrySize, population)
 - (sensorSignal, robotAction)
 - We learn a model that predicts for new data (houseSize, image...) their label
 - Two types of Supervised Learning problems:
 - Labels are “continuous variables” → Regression
 - Labels are “discrete categories” → Classification

Regression

- Predictor values \rightarrow Independent variable(s)
- Numeric output \rightarrow Dependent variable(s)
- Model:
 - Function f from predictors to output
- Goal: Function f applied to training data should produce values as close as possible in aggregate to actual outputs

Training data

$$W_1, X_1, Y_1, Z_1 \rightarrow O_1$$

$$W_2, X_2, Y_2, Z_2 \rightarrow O_2$$

$$W_3, X_3, Y_3, Z_3 \rightarrow O_3$$

.....

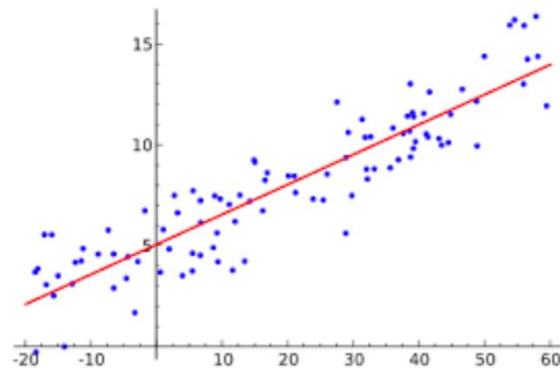
Model

$$f(w, x, y, z) = o$$

$$f(w_1, x_1, y_1, z_1) = o_1'$$

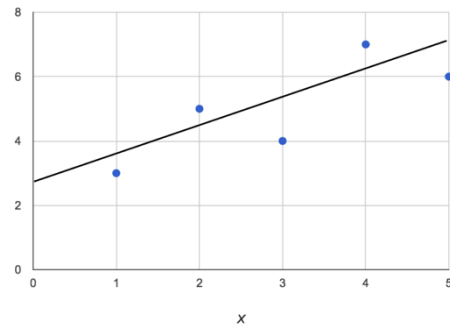
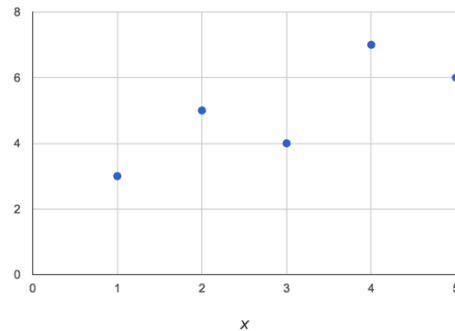
$$f(w_2, x_2, y_2, z_2) = o_2'$$

$$f(w_3, x_3, y_3, z_3) = o_3'$$



Simple Linear Regression

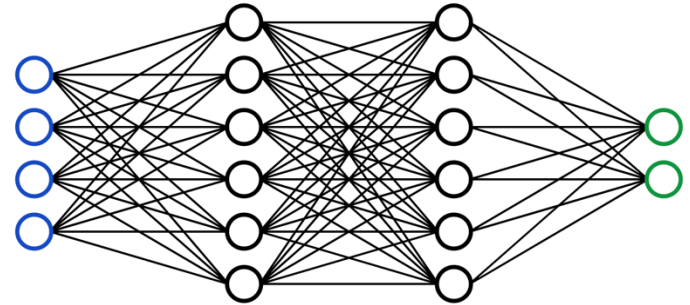
- One numeric predictor value, call it x
- One numeric output value, call it y
- Data items are points in two-dimensional space
- Functions $f(x)=y$ that are lines (for now)



Note: Elements of any ML problem

Any machine learning problem can be analyzed at four levels:

1. Data & application
2. Model
3. Optimization problem
4. Optimization algorithm

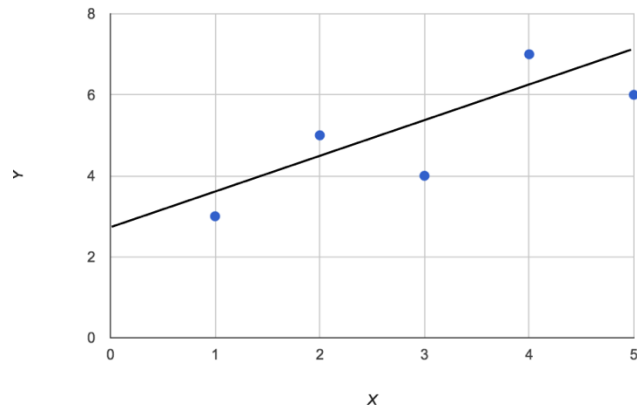


Back to our Linear Regression Problem

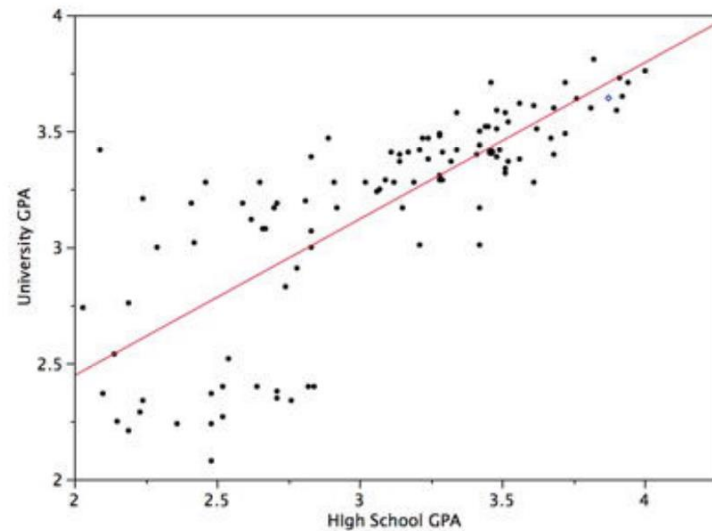
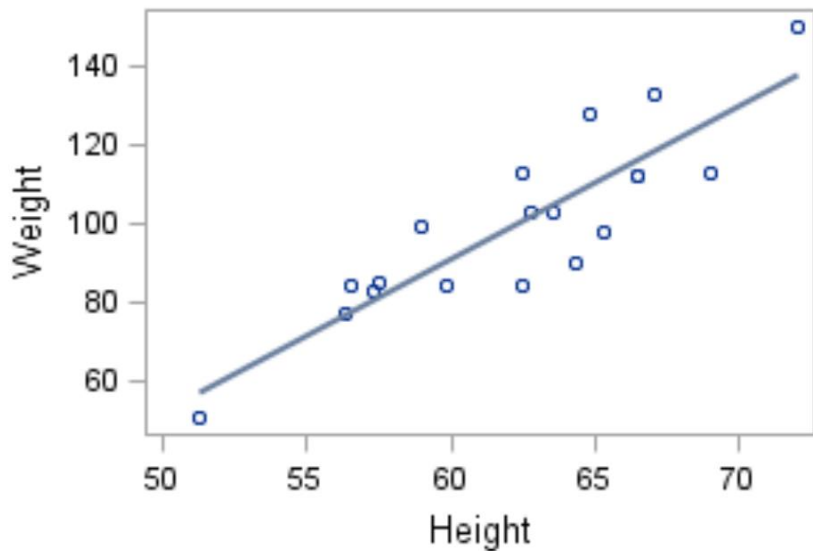
- Predictor= x , Output= y , one dimensional **data**
- We are trying to find the best linear function to “fit” all of our data
- $f(x) = y = \underline{a}x + \underline{b} \rightarrow$ Find a and b

model

$$y = 0.8x + 2.6$$



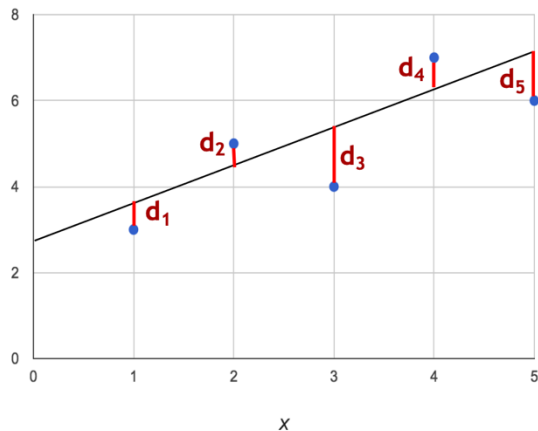
Real world examples



How do we calculate simple linear regression?

- Method of least squares

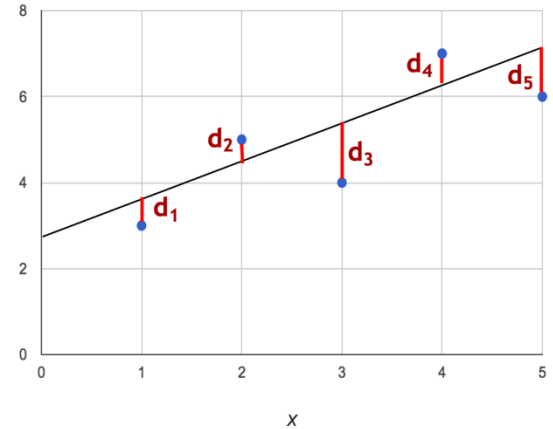
- Given a point and a line, the prediction error d for the point is its vertical distance from the line, and the squared error is d^2
- $d = f(x) - y$ (for each point)
- Given a set of points and a line, the sum of squared error (SSE) is the sum of the squared errors for all the points
- Goal: Given a set of points, find the line that minimizes the SSE



$$SSE = d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2$$

Least Squares → ML elements

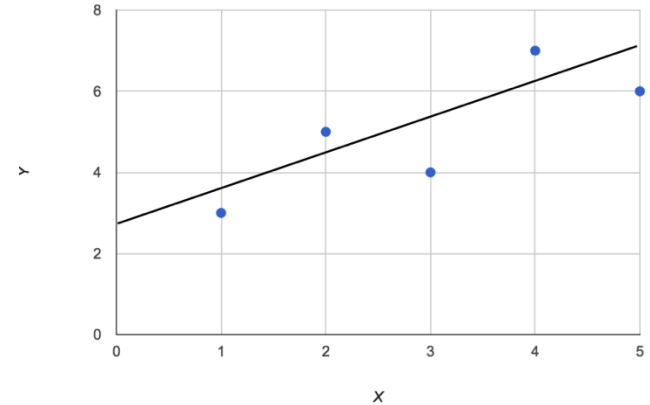
- Data & application → House size - price
- Model → Linear! $y = ax + b$, parameters a, b
- Optimization problem → Minimize the sum of squared errors
 - Loss function:
 - $\mathop{\text{arg min}}_{a,b} L = a^*, L = (y - \hat{y})^2$
- Optimization algorithm → Least squares



$$\text{SSE} = d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2$$

Solving Simple Linear Regression

- Simple Linear Regression: Model is a line with a single set of parameters
- $f(x) = ax + b$
 - $b = \bar{y} - (a \bar{x})$
 - $a = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$
 - $\bar{x}, \bar{y} \rightarrow$ average of x and y
 - $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$
 - $\bar{y} = \frac{\sum_{i=1}^n y_i}{n}$



Solving Ordinary Linear Regression

- Ordinary linear regression: Model is a matrix/vector
 - Allows for x to be multidimensional, dim d
 - Example: \vec{x} = (mother height, father height, current age), y =height

- $f(x) = y = X\omega$

- $X = \begin{pmatrix} \vec{x}_1^T \\ \dots \\ \vec{x}_n^T \end{pmatrix}, \vec{x}_i = \begin{pmatrix} x_i^1 \\ \dots \\ x_i^d \end{pmatrix}$

$$X = \begin{pmatrix} \text{HeightMother1}, \text{HeightFather1}, \text{Age1} \\ \dots & \dots & \dots \\ \text{HeightMotherN}, \text{HeightFatherN}, \text{AgeN} \end{pmatrix}$$

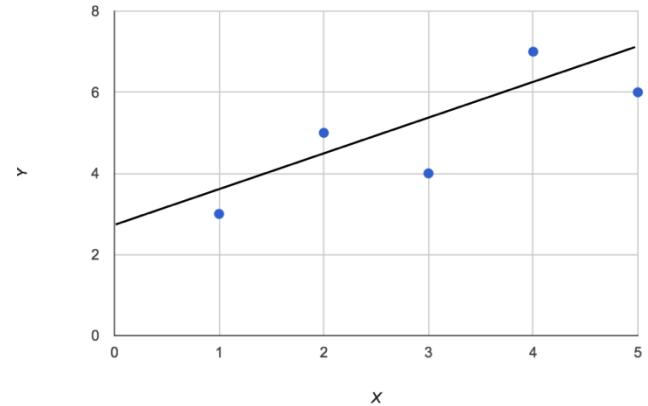
- Optimization: $\arg \min_{\omega} \|X\omega - y\|_2^2 = \arg \min_{\omega} \sum_{i=1}^n (\vec{x}_i^T \omega - y_i)^2$
- Solution: $\omega = (X^T X)^{-1} X^T y$

<https://pollev.com/robertomartinmartin739>

Exercise

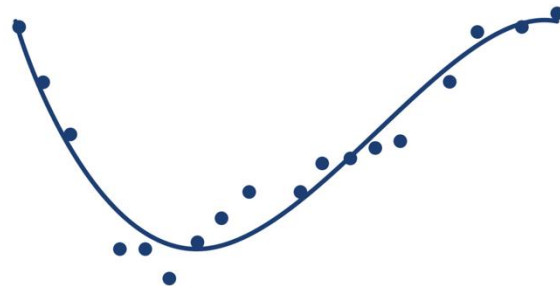
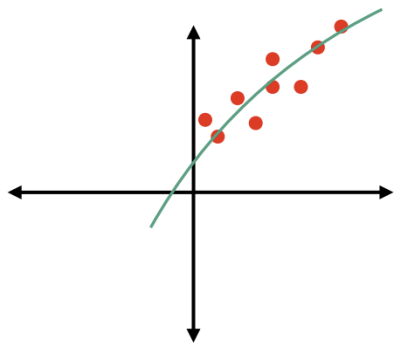
- We assume there is a linear relationship between dog height and weight

height	weight
2	12
3	17
4	22
5	27



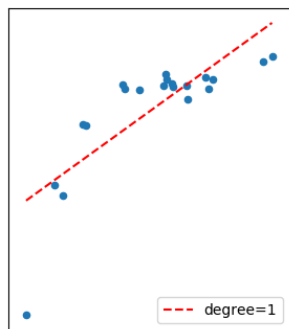
Other models for regression

- Linear regression assumes that the underlying model is linear
- We could try to fit other parametric models:
 - Quadratic
 - Polynomial

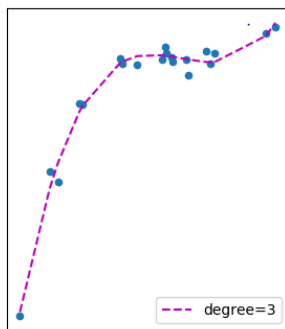


Underfitting, Overfitting

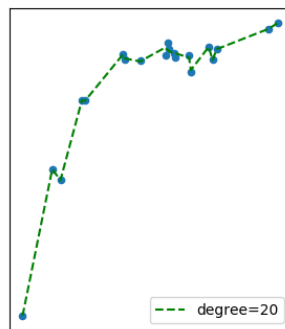
- Issue: the model doesn't generalize (predict) correctly
- Overfitting: we "adapt" too much to our data
- Underfitting: we do not "adapt" enough to our data



Underfit



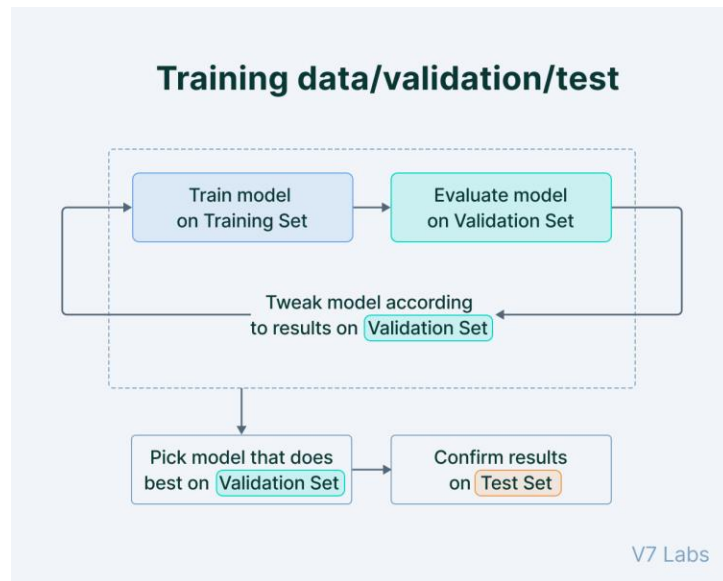
Correct Fit



Overfit

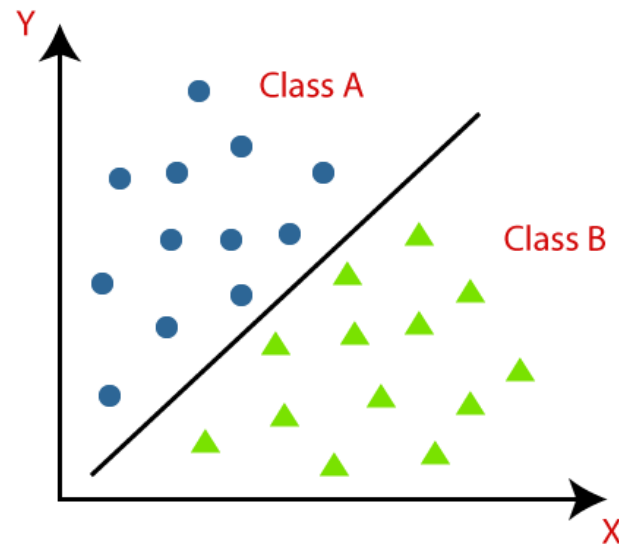
On Training, Test and Validation Splits

- Goal: How do we know if our model is underfitting/overfitting?
 - Fit your data as good as you can but in a way that you still fit some other data (fake new data, validation)
- Problem: we end up “fitting” the validation data too
 - Solution: keep some other data (test) that you do not use when improving your model



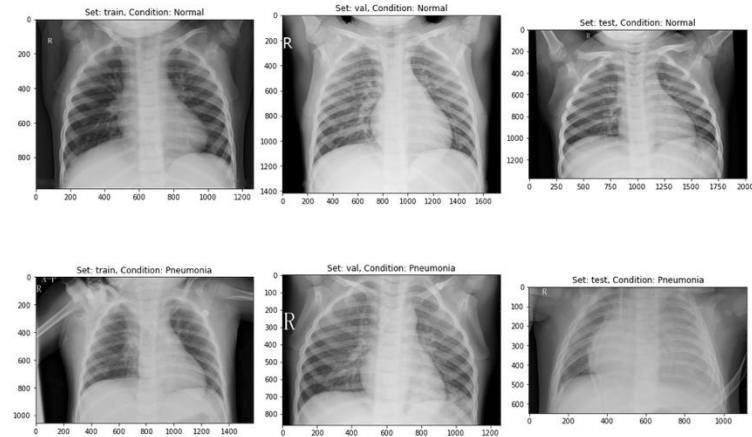
Classification

- Training data, each example:
 - Set of predictor values - Numerical or categorical
 - Categorical output value – “label”
- Model is function from features to label
 - Use model to predict label for new feature values
- Example
 - Features: age, gender, income, profession
 - Label: buyer, non-buyer



Other examples of classification problems

- Medical diagnosis
 - Feature values: age, gender, history, symptom1-severity, symptom2-severity, test-result1, test-result2
 - Label: disease
- Email spam detection
 - Feature values: sender-domain, length,#images, keyword1, keyword2, ..., keywordn
 - Label: spam or not-spam
- Credit card fraud detection
 - Feature values: user, location, item, price
 - Label: fraud or okay
- (A bit harder) Image classification
 - Feature values: image features
 - Label: object in the image



Solutions for Classification

- Very different to the solutions for regression
- Options:
 - K-nearest neighbors
 - Decision trees
 - Deep neural networks
 - Naïve Bayes
 - ...

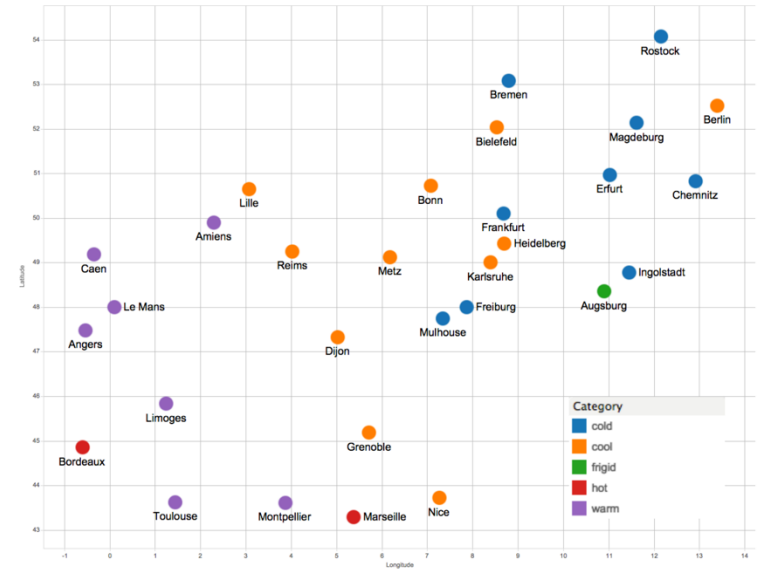
K-Nearest Neighbors (KNN)

- Define a metric that provides distance between any two elements of our dataset
 - $\text{distance}(i_i, i_j) \rightarrow$ For example, Euclidean distance
- For a new item, i , the classification is done by:
 1. Find the k -nearest neighbors of i
 2. Assign the most frequent label

(k is what is called a "hyperparameter", we choose it)

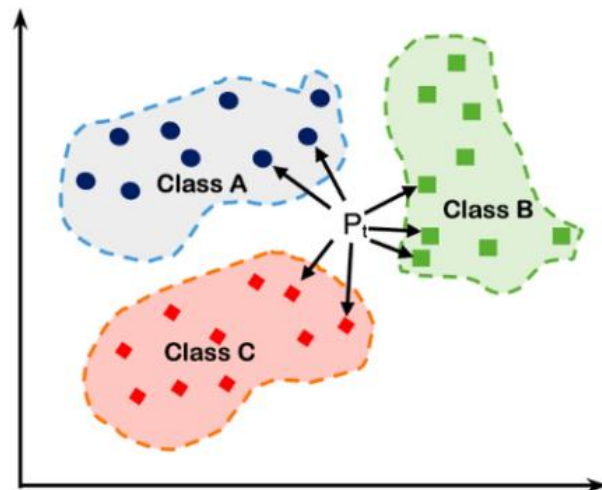
Example

- City temperatures
 - Features: longitude and latitude
 - Label: frigid, cold, cool, warm, hot
 - Distance: Euclidean distance
 - Data:
 - Nice (7.27, 43.72) cool
 - Toulouse (1.45, 43.62) warm
 - Frankfurt (8.68, 50.1) cold
- ...



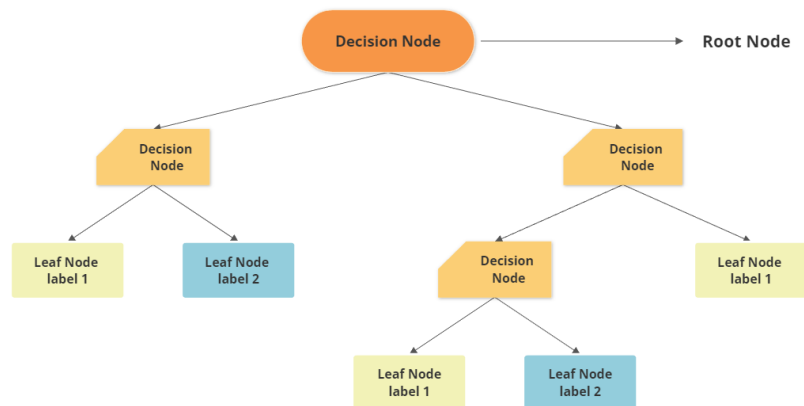
KNN Summary

- No hidden complicated math!
- Once distance function is defined, rest is easy
- Though not necessarily efficient
 - Real examples often have thousands of features
 - Medical diagnosis: symptoms (yes/no), test results
 - Email spam detection: words (frequency)
 - Database of labeled items might (and need to) be enormous



Decision Trees

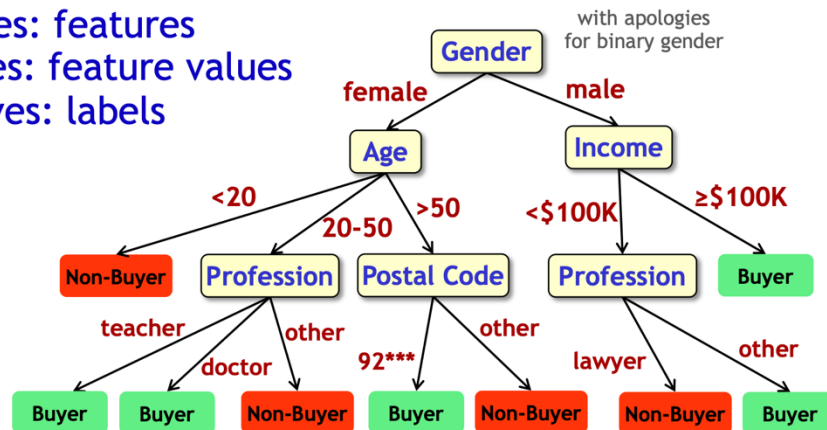
- Decision tree: sequence of discrete decisions
- Use the training data to construct a decision tree
- Use the decision tree to classify new data
- Leaves are classes



Example of Decision Trees

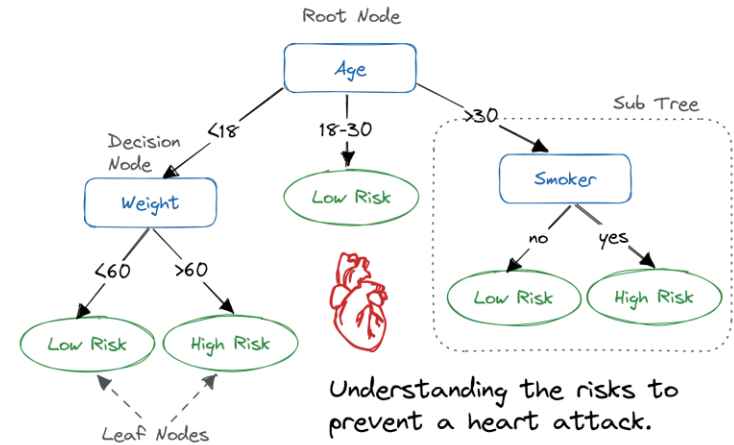
- Features - gender, profession, age, income, postal-code
- Label: buyer / non-buyer
 - person1 = (male, teacher, 47, \$25k, 94305) buyer
 - person2 = (female, teacher, 43, \$28K, 94309) non-buyer

Nodes: features
 Edges: feature values
 Leaves: labels



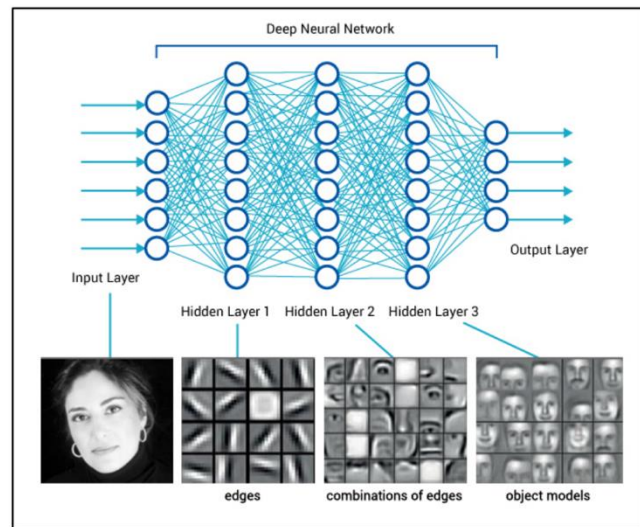
Decision Tree Summary

- Primary challenge is building good decision trees from training data
 - Which features and feature values to use at each choice point
 - HUGE number of possible trees even with small number of features and values
- Common approach: “forest” of many trees, combine the results
 - Still impossible to consider all trees



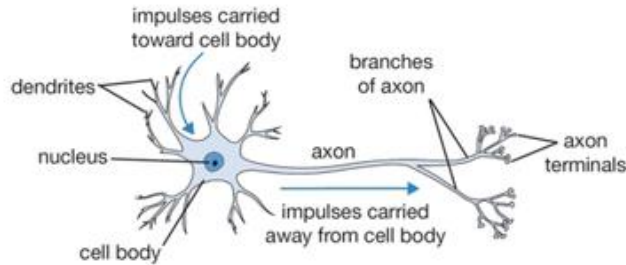
Deep Neural Networks

- Machine learning method modeled loosely after connected neurons in brain
- Invented decades ago but not successful
- Recent resurgence enabled by:
 - Powerful computing that allows for many layers (making the network “deep”)
 - Massive data for effective training
- Huge breakthrough in effectiveness and reach of machine learning
- Accurate predictions across many domains
- Big plus: Automatically identifies features in unstructured data (e.g., images, videos, text)



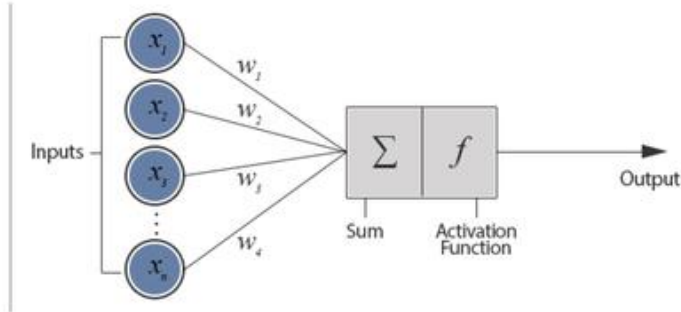
Artificial Neurons

Biological Neuron versus Artificial Neural Network



Biological Neuron

Computational building block for the brain

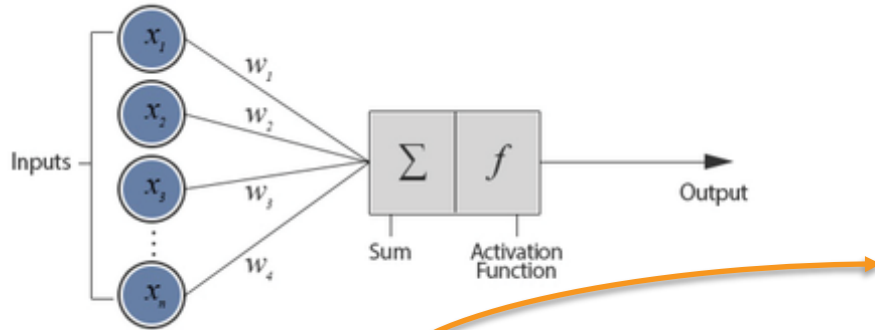


Artificial Neuron

Computational building block for the neural network

Note: Many differences exist – be careful with the brain analogies!

Output of a Neuron



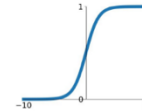
$$o = f\left(\sum_{i=1}^n w_i x_i\right) + b$$

$$o = f\left([w_0 \dots w_n] \begin{bmatrix} x_0 \\ \dots \\ x_n \end{bmatrix}\right) + b$$

Activation Functions

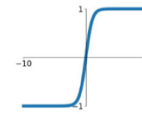
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



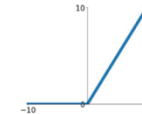
tanh

$$\tanh(x)$$



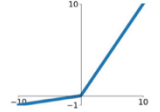
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

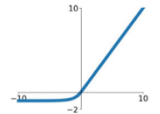


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

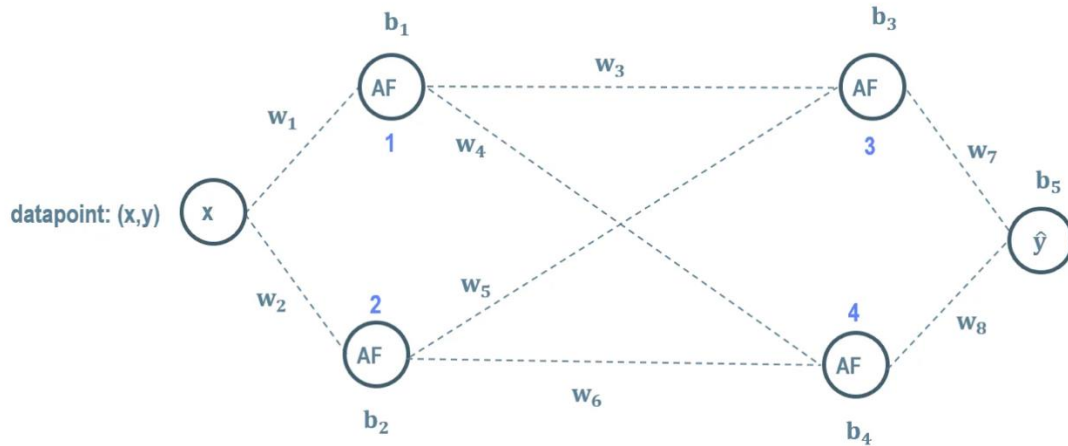
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



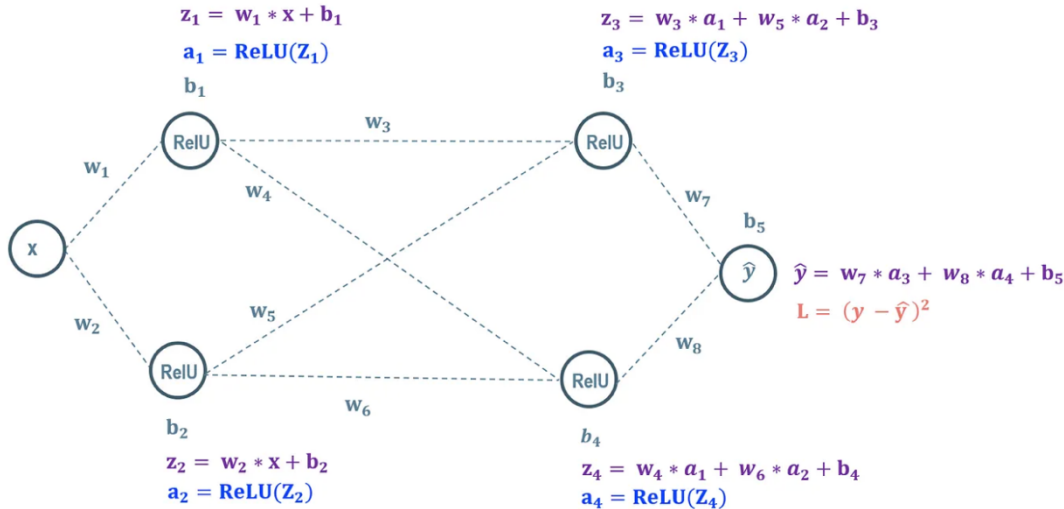
Example of Artificial Neural Network

- Forward pass: Input values and compute output



Example of Artificial Neural Network

- Forward pass: Input values and compute output



$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} w_1 & b_1 \\ w_2 & b_2 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix}$$

(2 X 2)*(2 X 1)

$$\begin{bmatrix} z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} w_3 & w_5 & b_3 \\ w_4 & w_6 & b_4 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ 1 \end{bmatrix}$$

(2 X 2+1)*(2+1 X 1)

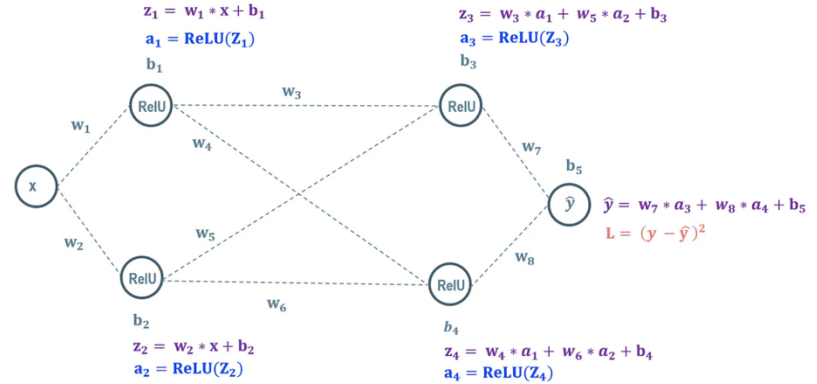
$$\hat{y} = \begin{bmatrix} w_7 & w_8 & b_5 \end{bmatrix} \begin{bmatrix} a_3 \\ a_4 \\ 1 \end{bmatrix}$$

(1 X 2+1)*(2+1 X 1)

How do we train it?

- Backpropagation
 - Compute the way we need to change the weights (w) and biases (b) so that our output gets closer to the given label
 - Propagate the gradient backwards (from output to input)
 - Use gradients:
 - $L = (y - \hat{y})^2$
 - $\frac{\delta L}{\delta \hat{y}} \frac{\delta L}{\delta w_i} \frac{\delta L}{\delta b_i}$
 - Take a small step (size Δ) changing your weights and biases in the opposite direction to the gradients:

$$w'_i = w_i - \frac{\delta L}{\delta w_i} \Delta$$



$$\frac{\delta L}{\delta \hat{y}} = -2(y - \hat{y}) = L_y$$

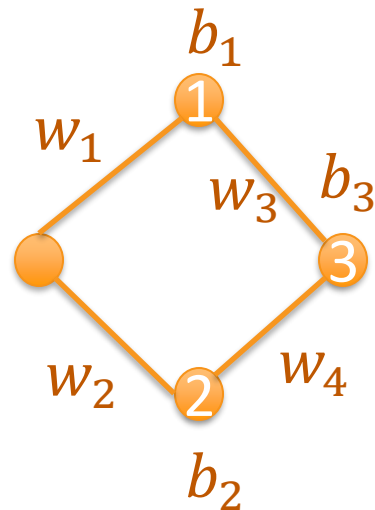
$$\frac{\delta L}{\delta w_7} = \frac{\delta L}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta w_7} = L_y \cdot a_3$$

$$\frac{\partial L}{\partial w_6} = \frac{\partial L}{\partial a_4} \frac{\partial a_4}{\partial z_4} \frac{\partial z_4}{\partial w_6} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_4} \frac{\partial a_4}{\partial z_4} \frac{\partial z_4}{\partial w_6}$$

<https://pollev.com/robertomartinmartin739>

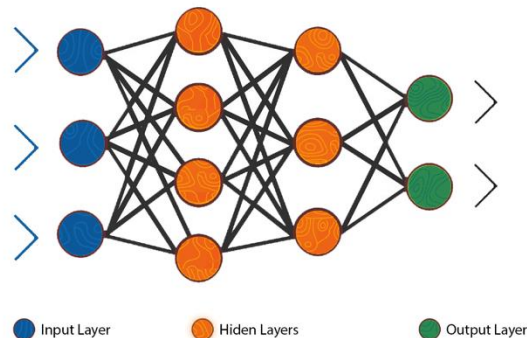
Exercise

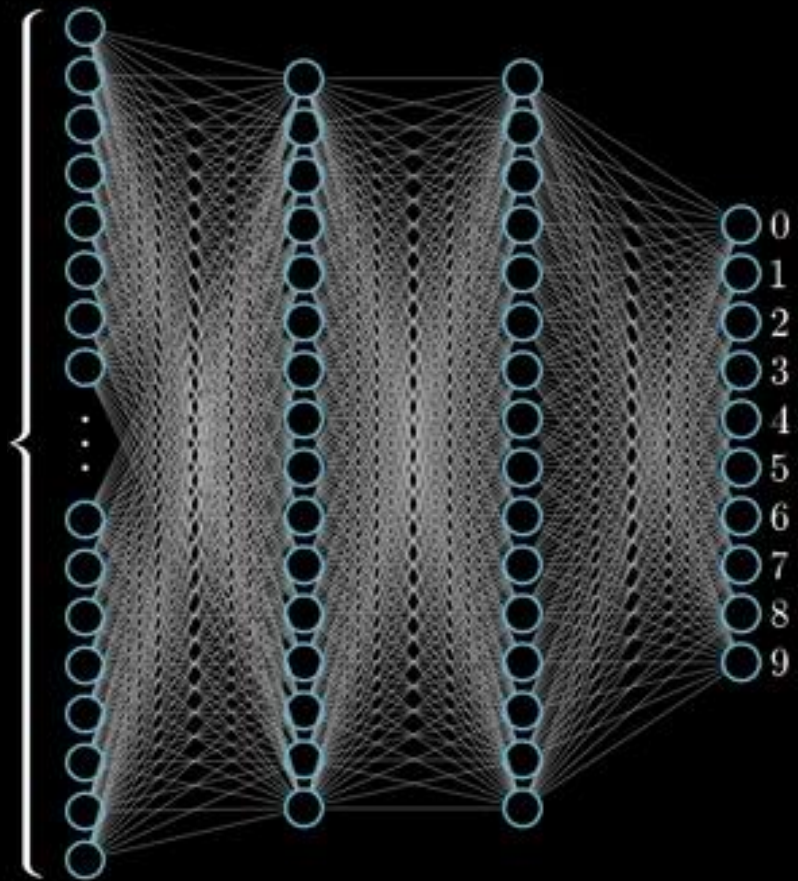
- Current network weights and biases:
 - $w_1 = 3, w_2 = -1, w_3 = 0.5, w_4 = 2$
 - $b_1 = 2, b_2 = 0, b_3 = 1$
- Activation function: ReLU
- Datapoint: $(x=2, y=6)$
- Gradient step: 0.01



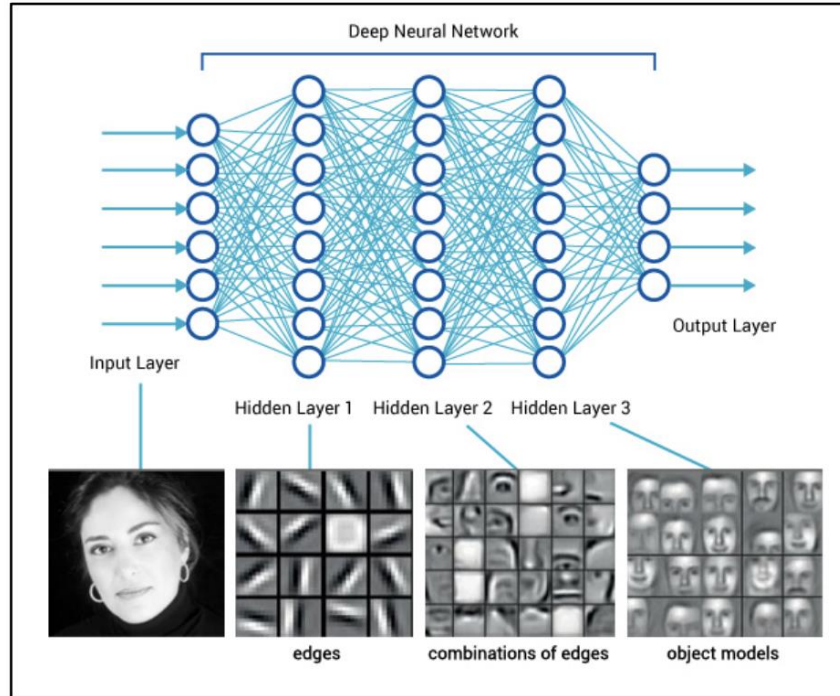
How do we use ANN/DNN for classification?

- We use a neural network to predict the probability of each class given an input (continuous value between 0 and 1) → logistic regression
- We use a output special layer in the neural network: the softmax
 - All outputs of that layer sum to 1
 - Each individual output is between 0 and 1
 - $\hat{y} = softmax(o), \hat{y}_i = \frac{e^{o_i}}{\sum_j e^{o_j}}$
- We combine it with a special loss: cross entropy loss





DNN for image classification



Quick Review of Deep Learning: **Implementation**



```
[ ] import torch
    from torch import nn

    class MNISTClassifier(nn.Module):

        def __init__(self):
            super(MNISTClassifier, self).__init__()

            # mnist images are (1, 28, 28) (channels, width, height)
            self.layer_1 = torch.nn.Linear(28 * 28, 128)
            self.layer_2 = torch.nn.Linear(128, 256)
            self.layer_3 = torch.nn.Linear(256, 10)

        def forward(self, x):
            batch_size, channels, width, height = x.size()

            # (b, 1, 28, 28) -> (b, 1*28*28)
            x = x.view(batch_size, -1)

            # layer 1
            x = self.layer_1(x)
            x = torch.relu(x)

            # layer 2
            x = self.layer_2(x)
            x = torch.relu(x)

            # layer 3
            x = self.layer_3(x)

            # probability distribution over labels
            x = torch.log_softmax(x, dim=1)

            return x
```

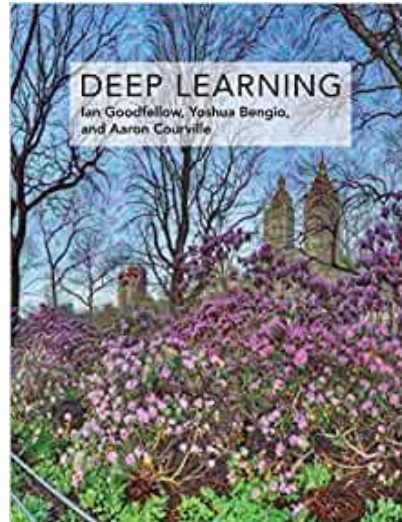
Quick Review of Deep Learning: **Resources**

Online Courses

- CS231N: Convolutional Neural Networks for Visual Recognition
<http://cs231n.stanford.edu/>
- MIT 6.S191: Introduction to Deep Learning <http://introtodeeplearning.com/>

Textbooks:

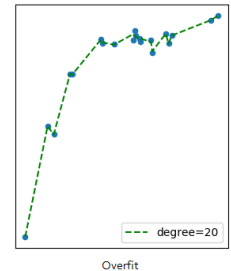
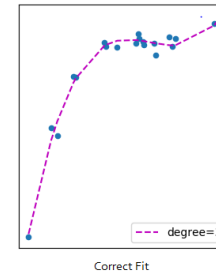
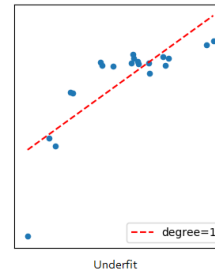
- Deep Learning. Ian Goodfellow, Yoshua Bengio, Aaron Courville
<http://www.deeplearningbook.org/>



Underfitting, Overfitting

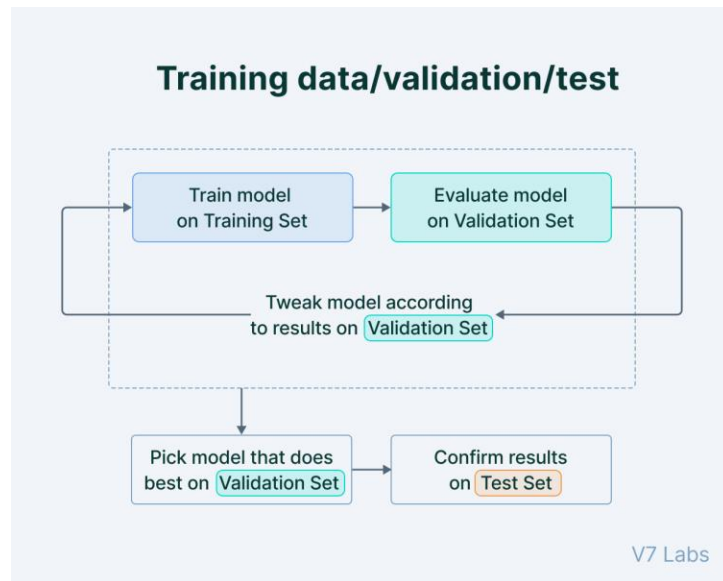
- Issue: the model doesn't generalize (predict) correctly
- Overfitting: we "adapt" too much to our data
- Underfitting: we do not "adapt" enough to our data

To make predictions!



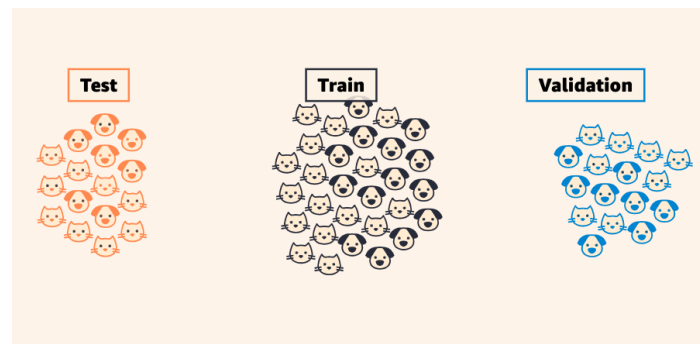
On Training, Test and Validation Splits

- Goal: How do we know if our model is underfitting/overfitting?
 - Fit your data as good as you can but in a way that you still fit some other data (fake new data, validation)
- Problem: we end up “fitting” the validation data too
 - Solution: keep some other data (test) that you do not use when improving your model



Analogy

- Training: teacher teaching students to solve long divisions
- Validation: teaching providing in-class quizzes or homework to assess and improve the understanding of students on long divisions
- Test: students taking final assessment test at the end of the term on long divisions (these are questions they have not seen before but quite similar to they have learned to solve in class)



Summary

- Machine Learning is a critical component of Artificial Intelligence
- AI: The study of how to engineer intelligence in systems/machines
- intelligence:
 - The ability to see and interpret visual input (Computer Vision)
 - The ability to read and understand language (Natural Language Processing)
 - The ability to move and interact with the world (Intelligent Robotics)
 - The ability to reason and perform logical deduction (Good Old-Fashioned AI (GOFAI))
- Any machine learning problem can be analyzed at four levels:
 1. Collect Data for an Application
 2. Develop a Model
 3. Define an Optimization Problem to find best Model Parameters
 4. Solve the Optimization Problem using some Algorithm
- Everything old is new again.
 - AI is big now, not because the theory is new, but because
 - Computation hit inflection points in cost and speed.
 - There is an abundance of data to train on for many applications
 - Co-robotics have hit inflection points in safety and cost
- Regression (linear regression), Classification (knn, decision trees, DNNs)