Practical Machine Learning

Practical Machine Learning

Data Scientist (n.):

Person who is better at statistics than any software engineer, and better at software engineering than any statistician.

Overview

- Course Overview
- Objective
- What the course will not cover / common misconceptions
- Expectations for participation (This is very important.)

Course Objective

Students will experience the most common types of machine learning algorithms and have a sense of their business applications.

Expectations for Participation

What’s not covered

Daily Structure

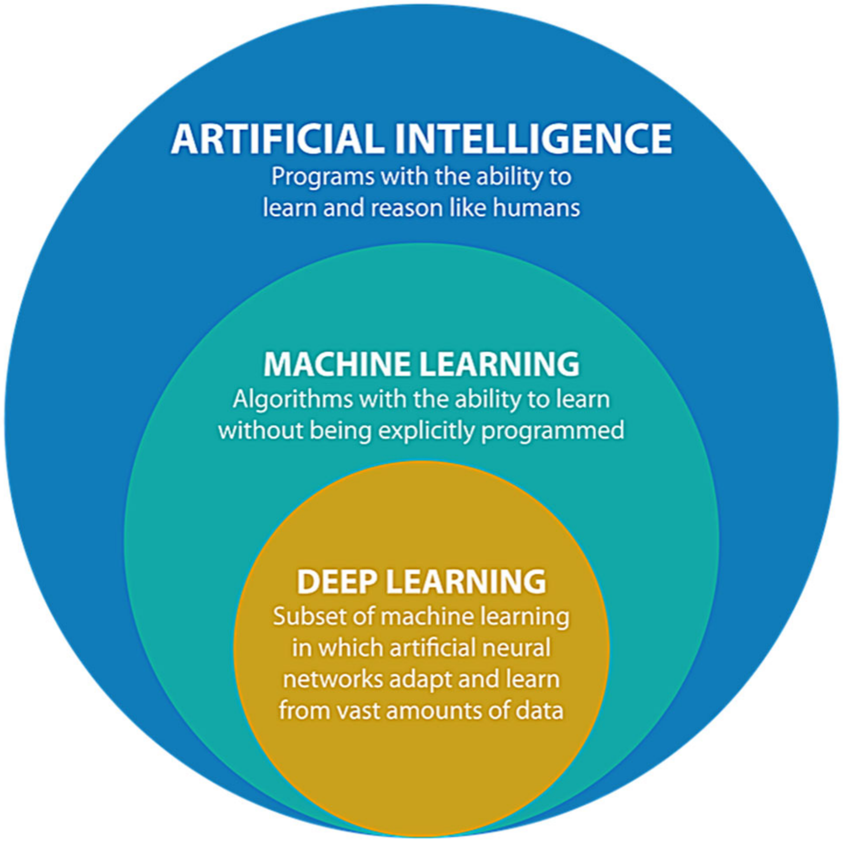
- Morning challenge review (15)
- Period Topic Introduction (15)
- Lecture/Demo (60-90)
- Group Recap (30-50)
- Assessment (15)
- Lab (60-120)

Introductions

- Icebreaker and intros per person

Definitions & Relationships

- AI (Artificial Intelligence)
  A vague umbrella with hopes for the future.
- ML (Machine Learning) - A part of AI
- Data Science – Practical ML



Familiar Process

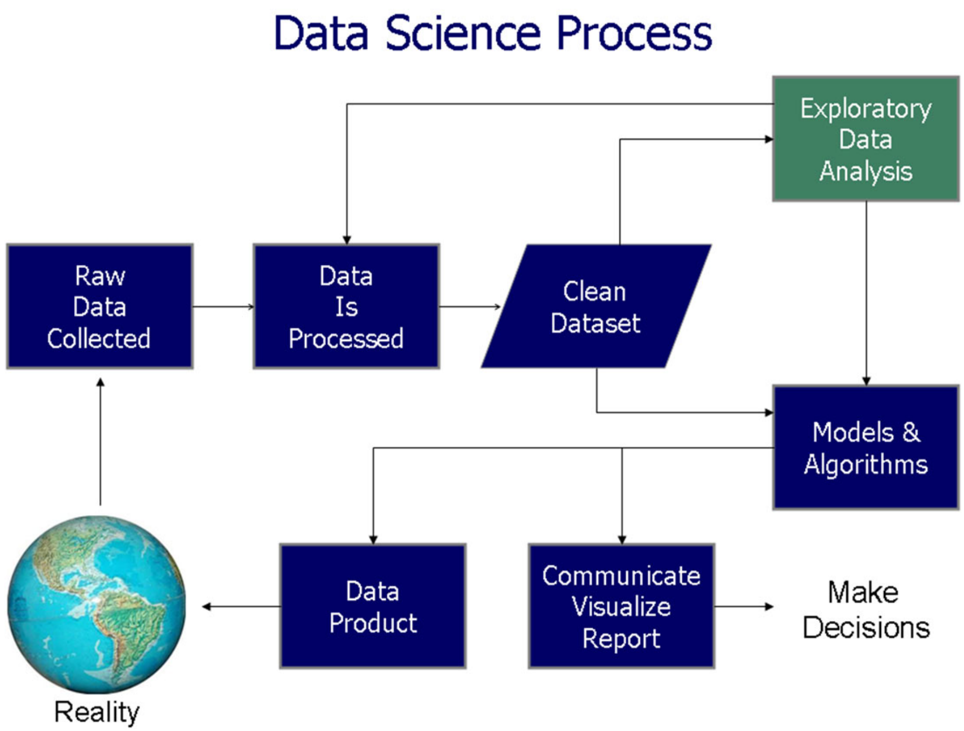
- ETL
- Reporting
- Where have you implemented this in your work before?

```
Excel?

SQL?

Data Warehouse?

Command-Line Log Analysis?
```

EDA Objectives

- The objectives of EDA are to:

  - Suggest hypotheses about the causes of observed phenomena
  - Assess assumptions on which statistical inference will be based
  - Support the selection of appropriate statistical tools and techniques
  - Provide a basis for further data collection through surveys or experiments[5]



Using a Pre-built Cloud Environment

- Self Hosted Jupyter Notebooks
- Google
  - Kaggle – Kernals
- Microsoft Azure

Lab: Python and SQL review

- What languages have you used?
- Let's talk about differences and similarities then we will build something simple from the ground up.

Python Review

- Terminology
- Python 2 vs 3 Differences

Initial Pure Python Projects

- Don’t use libraries in these review projects.
- Use only the simple python concept we reviewed.

Statistics from scratch

- You may not complete all of these, but we all should complete at least the first one. Screen share help for anyone struggling is available. Share your issues with your class-mates; many folks may be having the same ones.

Total

- Write a function called ***total*** that takes a List/Array of numbers as an argument
- Returns the total sum.
- Call it and verify and print the results.
- Then do the same thing for ***minimum***, ***maximum***, ***mean***, ***mode***, and ***median***.

What’s Normal

- **Random Numbers (Generate a data set with normal distribution) and Normal Distribution**

Roll

- Write a function named ***roll*** that takes a parameter called ***sides*** (defaults to 6) return generates a random number between 1 and sides (e.g. 6).

Roll Many

- Write a function named ***roll_many*** that takes a parameter called ***quantity*** (defaults to 3) and calls roll that many times then returns the total. Return values with defaults will be in range 3-18 as if totaling thee dice.

Generate

- Write a function named ***generate*** that takes a parameter called ***epochs*** (default to 100)
- Call ***roll_many*** once for each epoch.
- Store in a Dict/Hash the number of times each return value from roll_many occurs.
- The keys will be 3-18 and resulting quantities should be higher in the middle near 10 and lower at the extremes of 3 and 18.
- Return the final populated Dict/Hash.

```
{
 3:2,
 4:4,
…
10:13,
11:14,
…
17:3,
18:1
}
```

Chart

- Write a function called ***chart*** that takes a parameter called data that is a Dict/Hash Returned from Generate.
- Prints an ascii chart with hashes.

```
 3|## (2)
 4|##### (4)
…
10|############# (13)
11|############## (14)
…
17|### (3)
18|# (1)
```

Linear Algebra

- **Describe how Vector is just another name for a List/Array**
- **Matrix is just a List of Lists (A two-dimensional Array)**
- **And a Tensor is just an Array with three or more dimensions.**
- **Review what a *dot product* is with short examples.**

Dot

- Write a function called dot that takes two lists called A and B as parameters and returns the dot product.

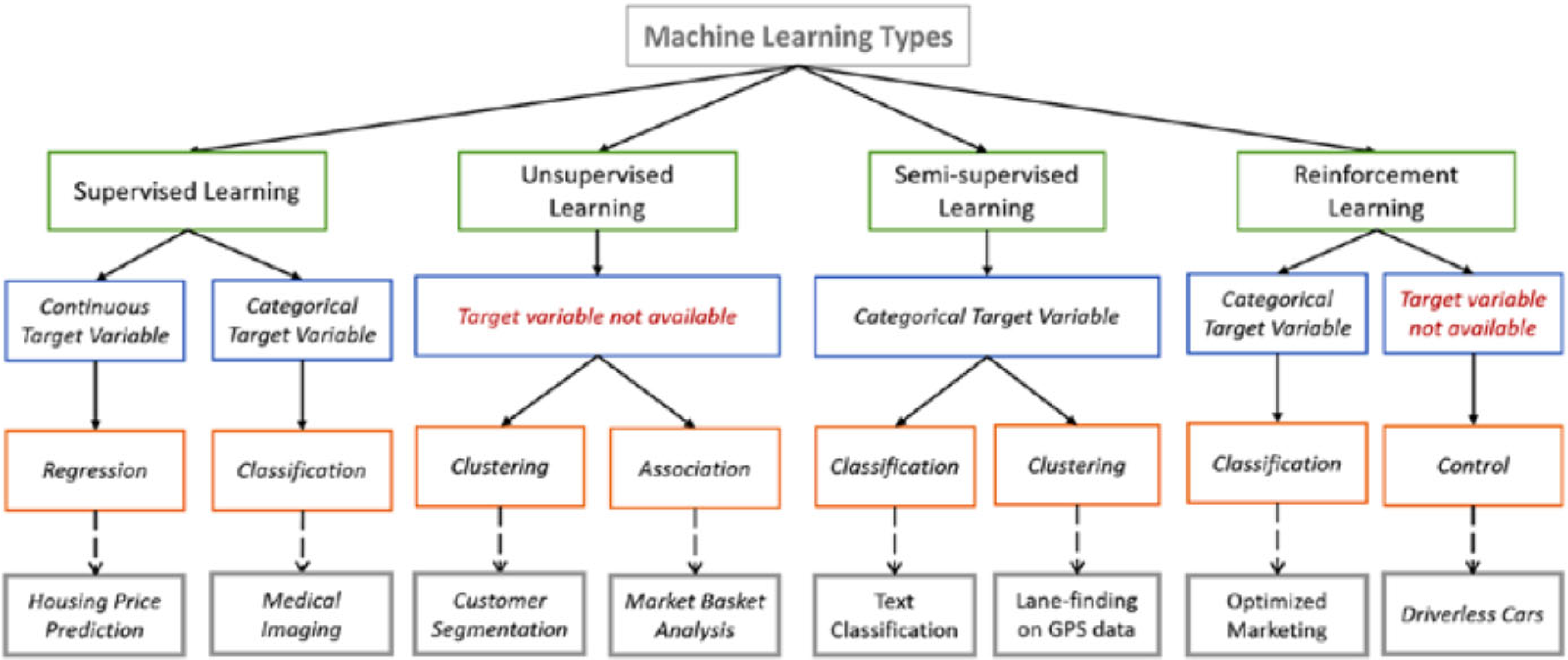
- **Extra Credit**
- Write a guessing game that guesses a number between 1 and 100 that a human has picked. The human will respond with one of three responses. H for higher, L for lower, or C for correct.

Machine Learning Algorithms

- Classification and Regression
- Clustering
- Principal Components Analysis
- Time Series

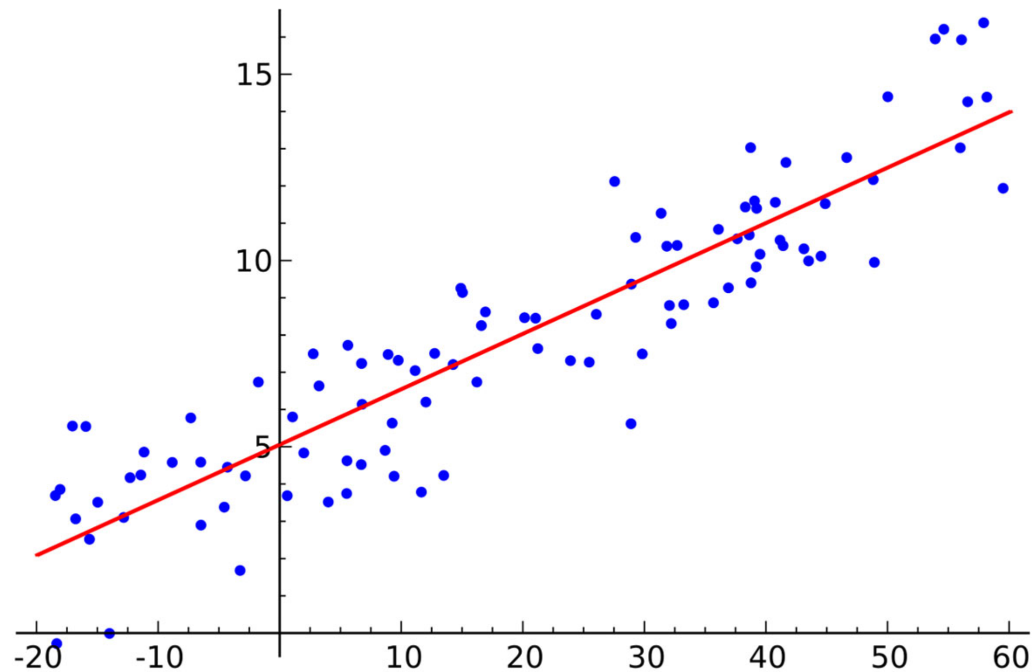
- Machine Learning
  - Supervised Learning
    - Classification/Regression
      - Binary Classifiers
        - Perceptron
        - Decision trees
        - Random forests
        - Bayesian networks
        - Support vector machines
        - Neural networks
        - Logistic regression

Machine Learning Types



Linear Regression

- Multiple Linear Regression



Period 3 – Classifiers (Statistical Classification)

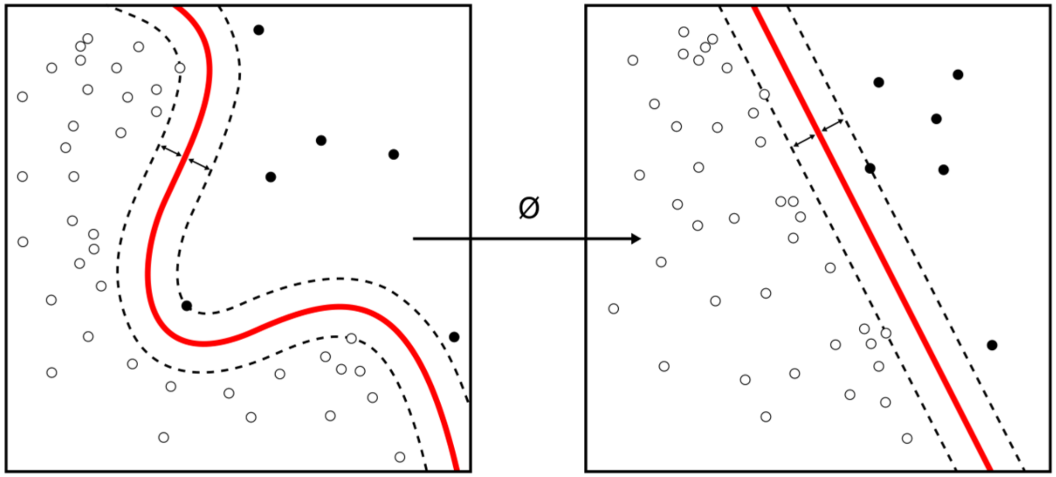
- **Evaluating a classifier**
  - **Holdout method**
  - Cross-validation

Common Learning Algorithms:

- Support Vector Machines
- linear regression
- logistic regression
- naive Bayes
- linear discriminant analysis
- decision trees
- k-nearest neighbor algorithm
- Neural Networks (Multilayer perceptron)
- Similarity learning

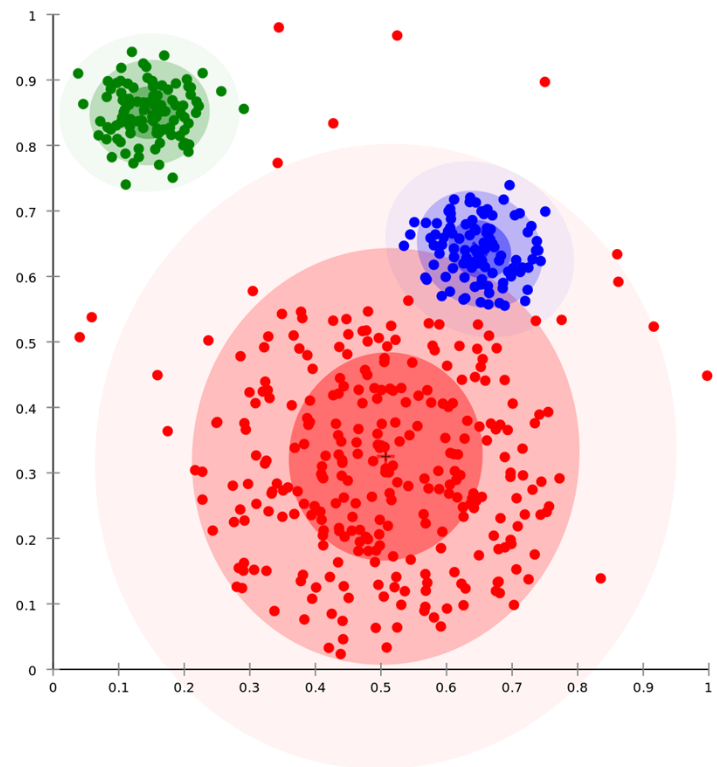
Naive Bayes (Supervised Learning)

- **Naive Bayes**
- Others
  - Decision Trees
  - ANN – Artificial Neural Networks
  - k-Nearest Neighbor (KNN)
- **Evaluating a classifier**
  - **Holdout method**
  - Cross-validation



Clustering (Cluster Analysis – Unsupervised Learning)

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.



LAB – Demo: Regression

Classification

In this section we will examine BitCoin data and see if we can predict a buy or sell. The data comes from a set of Coinbase trades from December of 2014 to January of 2018 and is available from Kaggle. We will also examine a set of data that describes wheat seeds of various geometries and their attributes.

Logistic Regression

Logistic Regression is a common algorithm (and amongst the simplest used for classification tasks). To build a classifier, the algorithm attempts to find the line that best splits the data into the target classes.

This generally happens by:

Picking a parameter value at random and placing a random line through the distribution.

Measure how well the line separates the two classes (statistical deviance is used for the metric).

Guess the new values of the parameters and measure the separation.

Repeat until there are no better guesses. Gradient descent is typically used for the optimization.

jupyter

LAB - Demo: Regression

Can we predict tomorrow's close based on today's info?

We will use a row of data for input. We will call the input X and the prediction y. This is called "supervised learning" as we will feed in both X and y to train the model.

Let's use a model called Linear Regression. This performs better if we *standardize* the data (0 mean, 1 std).

```
For 2 dimensions this takes the form of:
```

$y=m*x+b$y=m*x+b

```
M is the slope (or coefficient) and b is the
intercept.

Let's see if we can predict the open price from
the ts component.
```

Exercise: Regression

```
    Use linear regression to predict area from the other columns.
    Calculate the predictive model's score.
        om the ts component.
```

LAB - Demo: **Visualize Performance of the Model**

Visualize Performance of the Model:

Actual and Predicted Values

You can plot the actuals and the predicted values.

It looks like the model does a pretty poor job of describing the data.

Exercise:

Visualize the Errors

Plot the actual y and predicted y against one another to compare the accuracy of the model

LAB - Demo: **Visualize Performance of the Model**

Improve the Accuracy of the Model:

Try More Features

In an attempt to get a better model we are going to use more features to make a prediction. Many machine language estimators require "standardization" of the data and will perform badly if the individual features do not more or less look like normally distributed data: Gaussian distributions with a zero mean and unit variance.

`Exercise:`

`Regression`

`Try scaling the input and using the log of the area and see if you get a better score.`

`Examine the coefficients`

LAB - Demo: **Training/Test Split**

Training/Test Split

In fact we were cheating, predicting things that we already saw serves little purpose. The model could just memorize the data and get a perfect score. But it wouldn't *generalize* to unseen data.

To see how it will perform in the real world we will train on a portion of the data and test on a portion that it hasn't seen.

Exercise:

Regression with Train/Test Split

Split the data into test and training data. What is the score on the test data?

jupyter

LAB - Demo: **Visualize Errors with Residual Plots**

Visualize Errors with Residual Plots

A residual is the difference between the prediction and the actual.

If we plot predicted value against residuals, we should get a random distribution. If not, a different model would be better given the data. *A pattern in the residuals implies that there is a non-parametric relationship at play.*

Exercises:

Residual Plot

Make a residual plot of your test and train data

LAB - Demo: **Other Models**

Other Models:

- **SVM**
- **Random Forest**
- **Huber**

```
Logistic Regression is not the
only model that can be used for
classification!
```

LAB - Demo: **SVM - (**Support Vector Machines)

SVM - (Support Vector Machines)

Support vector machines include both linear and non-linear variations.

Like logistic regression, the main idea is to find the line (or plane or dividing shape) that separates the targets/classes optimally.

Instead of measuring the distance to all points, VCMs try to find the largest margin between only the points on either side of the decision line but rather than worry about points that are far away to the boundary of a decision (e.g., the obvious ones), the algorithm focuses on the points that the closest to the line.

It then seeks to place the line in such a way so that the distance of those points is as great as possible.

SVMS uses a trick to map points that are non-linear in nature to a coordinate plane that is non-linear.

The algorithm then tries to find a linear boundary in the warped space.

jupyter

LAB - Demo: **Random Forest**

Random Forest

Random forests rely upon the use of a decision tree.

Decision trees are based on a series of branch points that help to make a decision.

When using a decision tree algorithm, you allow the computer to figure out (based on the training data) which variables are the most important.

It then puts these at the top of the tree and gradually uses less important variables in subsequent branches until a path to target outcomes has been plotted.

In decision trees, the topmost level branches have an enormous influence on the quality of the tree. If new data doesn't follow the same distribution as the training set, then the model doesn't generalize quite as well.

Random forests build a collection of decision trees and apply these to new observations. It then uses a set of "votes".

To weight the outputs of several trees and apply them to the new observation. It provides the majority vote in the case of classification or the mean value when performing regression.

- Random forests have a degree of immunity to unimportant features.
- They are also able to cope with noisy datasets or those with missing values.

LAB - Demo: **Huber**

Huber

A regression algorithm that is useful with datasets with outliers.

It does this by scoring the outliers and weighting their scores appropriately.

Exercises:

Other Models

1. Try using another model (Random Forest or SVR)

2. assess the accuracy of the new model.

jupyter

LAB - Exercise: Regression

Predicting the size of forest fires based on meteorological data.

Data Source:

`https://archive.ics.uci.edu/ml/datasets/Forest+Fires`

First Steps:

1. Read the data into a ***DataFrame***
2. Examine the types
3. Describe the data

```
Attributes:
1. X - x-axis spatial coordinate within the Montesinho park
   map: 1 to 9

2. Y - y-axis spatial coordinate within the Montesinho park
map: 2 to 9

3. month - month of the year: "jan" to "dec"
4. day - day of the week: "mon" to "sun"
5. FFMC - FFMC index from the FWI system: 18.7 to 96.20
6. DMC - DMC index from the FWI system: 1.1 to 291.3
7. DC - DC index from the FWI system: 7.9 to 860.6
8. ISI - ISI index from the FWI system: 0.0 to 56.10
9. temp - temperature in Celsius degrees: 2.2 to 33.30
10. RH - relative humidity in %: 15.0 to 100
11. wind - wind speed in km/h: 0.40 to 9.40
12. rain - outside rain in mm/m2 : 0.0 to 6.4

13. area - the burned area of the forest (in ha): 0.00 to
1090.84

The `area` output variable is very skewed towards 0.0, thus
it may make sense to model with the logarithm transform.

Tasks:
1. Load the data into the frame
2. Inspect the data types that were loaded
3. Describe/summarize the data values
4. Plot the general distribution
```

LAB - Demo: Decision Tree

Exercise: Load Data

Exercises associated with this example look at predicting the whether a mushroom is poisonous.

Load the mushroom data

Decision Tree

Decision tree models construct a set of rules based on the desired outcome

The process of training classifier is to get X and y and call .fit.

To predict values of y (y hat), call .predict(X)

To get the accuracy call .score(X, y)

LAB - Exercise: Regression

Questions

- What trends do we see in the data?
- What do we know about the frequency of wildfires? Does the distribution conform to what might be

Regression

- Use linear regression to predict area from the other columns.

  (If you have object data columns, you can create *dummy columns* using pd.get_dummies, pd.concat, and pd.drop)

- What is your score?
- Expected?

LAB - Demo: **Feature Engineering**

Feature Engineering

Only using historical price data results in a poor model. We need to be a little more intelligent about what we are basing our decisions on.

- What might a predictive model based purely on price be a poor predictor?
- How might we derive additional insight from the data?

Feature engineering is the practice of using a transformation of raw input data to create new features that can be used in an ML model. It can be used to add "additional insight" (usually derived from a procedure provided by a domain expert) that can help the machine model find more accurate predictions.

Examples:

- dividing a stock price by its earnings in order to get a ratio of how much an equity costs to how much money it makes
- counting the occurrence of a particular word across a text document
- joining data across tables (for example data describing cardiac events with neurological events) to get a better feel for a patient's overall health
- applying signal-processing tools to an image and summarizing the output, for example transform functions to an EKG signal or a histogram to a medical image

Why use feature engineering?

1. Transform original data relative to the target
2. Bring in external data sources
3. Use unstructured data sources
4. Create features which are more easily interpreted

As the relative predictive accuracy of the model is assessed, it can be updated over time.

LAB - Demo: **Feature Engineering**

Feature Engineering

- Exercises associated with this example look at creating a classifier from the (wheat) seed dataset.
- Does the classification score improve if a feature engineered column is included?

ROC Curve

Many machine learning predictions involve a degree of uncertainty and classification algorithms output not only the zero-one predictions, but the full probabilities. These probabilities can be summarized as a probabilistic classifiers (also called probability vectors or class probabilities). When evaluating a test data set, there is generally a number from 0 to 1 which describes the probability of a particular target. Generally the machine learning algorithm picks a threshold which is used to assign a particular prediction.

The probabilities can be visualized as an ROC curve to determine if there are "accuracy tradeoffs" for a specific dataset. By convention, you plot the false positive rate on the x-axis and the true-positive rate on the y-axis. A perfectly predictive model is a right angle with no false positives and no missed detections.

The area under the ROC curve is also used as an evaluation metric. The larger the area, the better the classification performance. Using both the visualization and the area provides a powerful way to gauge accuracy versus mis-classification tradeoffs.

If classifying for a disease, it is better to classify some healthy patients as sick rather than miss truly healthy patients. *Though this comes with a cost as well.*

LAB - Demo: Confusion Matrix

Confusion Matrix

A Confusion Matrix is another way to evaluate performance. You can see where false positives (lower left) and false negatives (upper right) are. A confusion matrix is a two-by-two diagram where each element shows the class-wise accuracy or confusion between the negative and positive classes.

Confusion matrixes provide ways to evaluate model performance. They provide a way to see where false positives (lower left) and false negatives (upper right) appear.

Exercise: Confusion Matrix

1. Exercises associated with this example look at creating a classifier from the (wheat) seed dataset.
2. Plot a confusion matrix for the seed model

LAB - Demo: **Classification Report**

Classification Report

Precision - Correct positive over all positive - True positives / (false + true positives) - How many selected items are relevant?

Recall - Correct positive over positive that should have been returned - True positives / (true positives + false negatives) - How many relevant items are selected?

F1 - Harmonic mean of above

`Exercise: Create a classification report`

`Exercises associated with this example look at creating a classifier from the (wheat) seed dataset.`

`1. Create a classification report.`

LAB - Demo: **Calibration Curve**

Calibration Curve

From http://scikit-learn.org/stable/auto_examples/calibration/plot_calibration_curve.html and https://jmetzen.github.io/2015-04-14/calibration.html.

When performing classification, we want not only to predict the class label but also obtain a probability of the respective label. This gives a degree of confidence on the prediction. Some models can give you poor estimates of the class probabilities and some even do not support probability prediction.

A well calibrated binary classifier should be able to pick among samples that approximates 80% (0.8). Some of the implementations in sklearn struggle, however. The sklearn.calibration module adds additional support for managing calibration in a uniform fashion. It also helps to assess the calibration of a specific model.

In a calibration curve, a perfectly calibrated curve will be a straight line. Logistic regression returns a well calibrated curve by default as it directly optimizes log-loss.

jupyter

LAB - Demo: **Optimizing Models**

Optimizing Models

Models have *hyperparameters* that we can tune. These allow for different variations of the model to be explored for which is the most accurate.

Grid search cross validation will hold out some of the data for testing purposes, so we can pass in the full X and y into it.

Exercise:

Optimize Model

Exercises associated with this example look at creating a classifier from the (wheat) seed dataset.

Optimize the classifier.

LAB - Demo: **Learning Curves**

Learning Curves

Do we have enough data?¶

http://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html

An important question that often needs to be addressed in machine learning is "Do we have enough data?" Learning curves can be helpful in assessing the answer. Every estimator has advantages and drawback with three general sources of error: bias, variance, and noise:

bias:

average error between different training sets

variance:

how sensitive a model is to different data sets

noise:

property of the data that can be used to describe how much samples may deviate from the underlying relationship. Some distributions adhere very closely to predicted values while others deviate wildly.

A highly biased model will describe the training data well, but offers poor predictions on testing data even if it is from the same sample or distribution. A highly variable model will describe training and testing data well (if the data is from the same sample/distribution), but offers poor predictions on new data from a different sample/distribution.

It is common for different of estimators to describe data differently. For example a simple model may provide a poor fit because it is too simple (and for that reason, highly biased). Or it is possible that a complex model may fit the training data too well, and is not able to make good predictions on new data (high variance).

When training and assessing models, the goal is to make both bias and variance as low as possible.

LAB - Demo: **Learning Curves**

Learning Curves

Do we have enough data?¶

http://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html

An important question that often needs to be addressed in machine learning is "Do we have enough data?" Learning curves can be helpful in assessing the answer. Every estimator has advantages and drawback with three general sources of error: bias, variance, and noise:

bias:

average error between different training sets

variance:

how sensitive a model is to different data sets

noise:

property of the data that can be used to describe how much samples may deviate from the underlying relationship. Some distributions adhere very closely to predicted values while others deviate wildly.

A highly biased model will describe the training data well, but offers poor predictions on testing data even if it is from the same sample or distribution. A highly variable model will describe training and testing data well (if the data is from the same sample/distribution), but offers poor predictions on new data from a different sample/distribution.

It is common for different of estimators to describe data differently. For example a simple model may provide a poor fit because it is too simple (and for that reason, highly biased). Or it is possible that a complex model may fit the training data too well, and is not able to make good predictions on new data (high variance).

When training and assessing models, the goal is to make both bias and variance as low as possible.

jupyter

LAB - **Exercise : Learning Curves**

Exercise:

Learning Curves

Exercise: Learning Curves

Exercises associated with this example look at creating a classifier from the (wheat) seed dataset.

1. Run a learning curve against the seed data.

How much data do we need to train on?

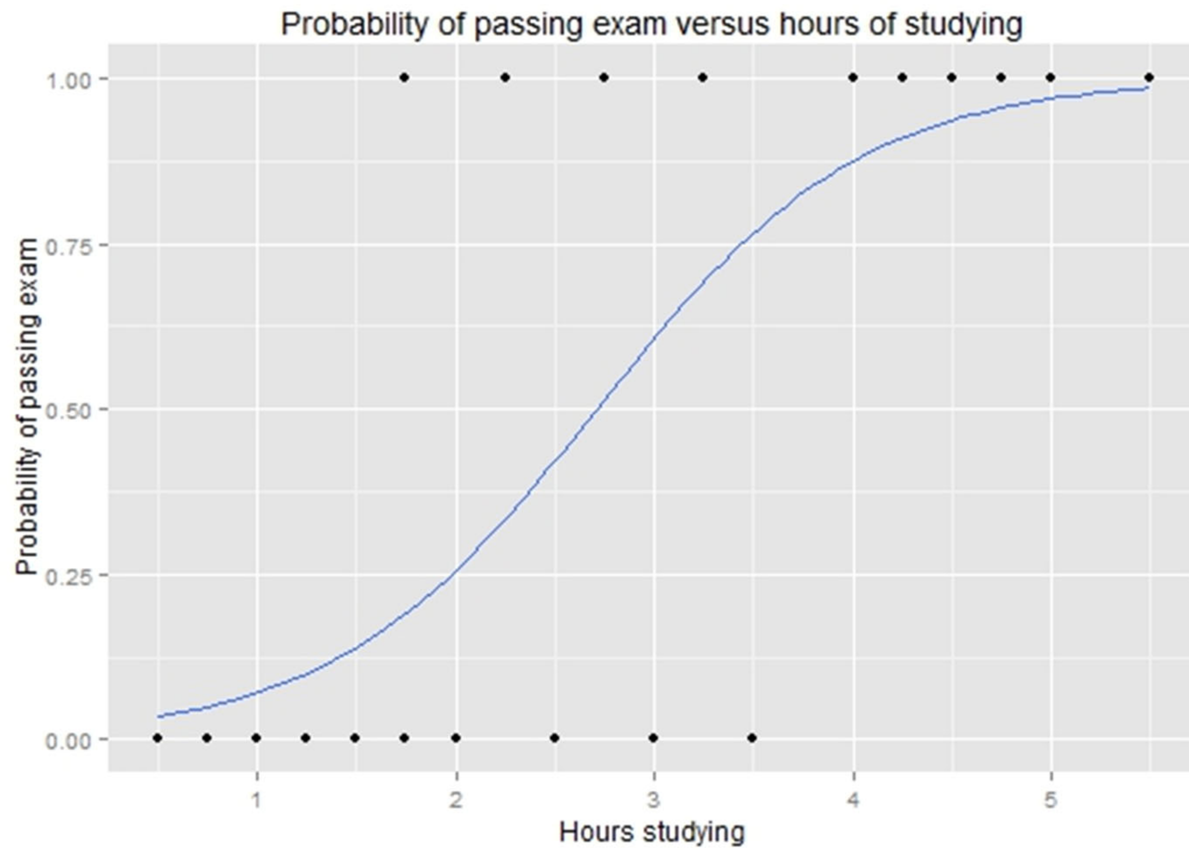
Linear Regression

Linear regression is the simplest and most widely used algorithm for building regression models. The algorithm plots the dataset as a set of points with the target variable on the y axis. It then attempts to fit a straight line (or plane) to the points using a variant of the equation $y = m * x + b$y=m∗x+b.

jupyter

Logistic Regression

- Logistic Regression
  - Boolean
    - 0/1
    - True/False
    - In/Out



jupyter

NLP Natural Language Processing

- NLP
  - (Natural Language Processing)

Extracting meaning from unstructured text is an example of "summarization". It is a classic problem in NLP and has been looked at from a variety of different perspectives. In this example, we are going to use a common algorithm to summarize and then expand and normalize the terms using a medical thesaurus.

Clustering

Clustering

Clustering is an *unsupervised method*.

You tell the computer to create groups without giving it labels.

Useful in:

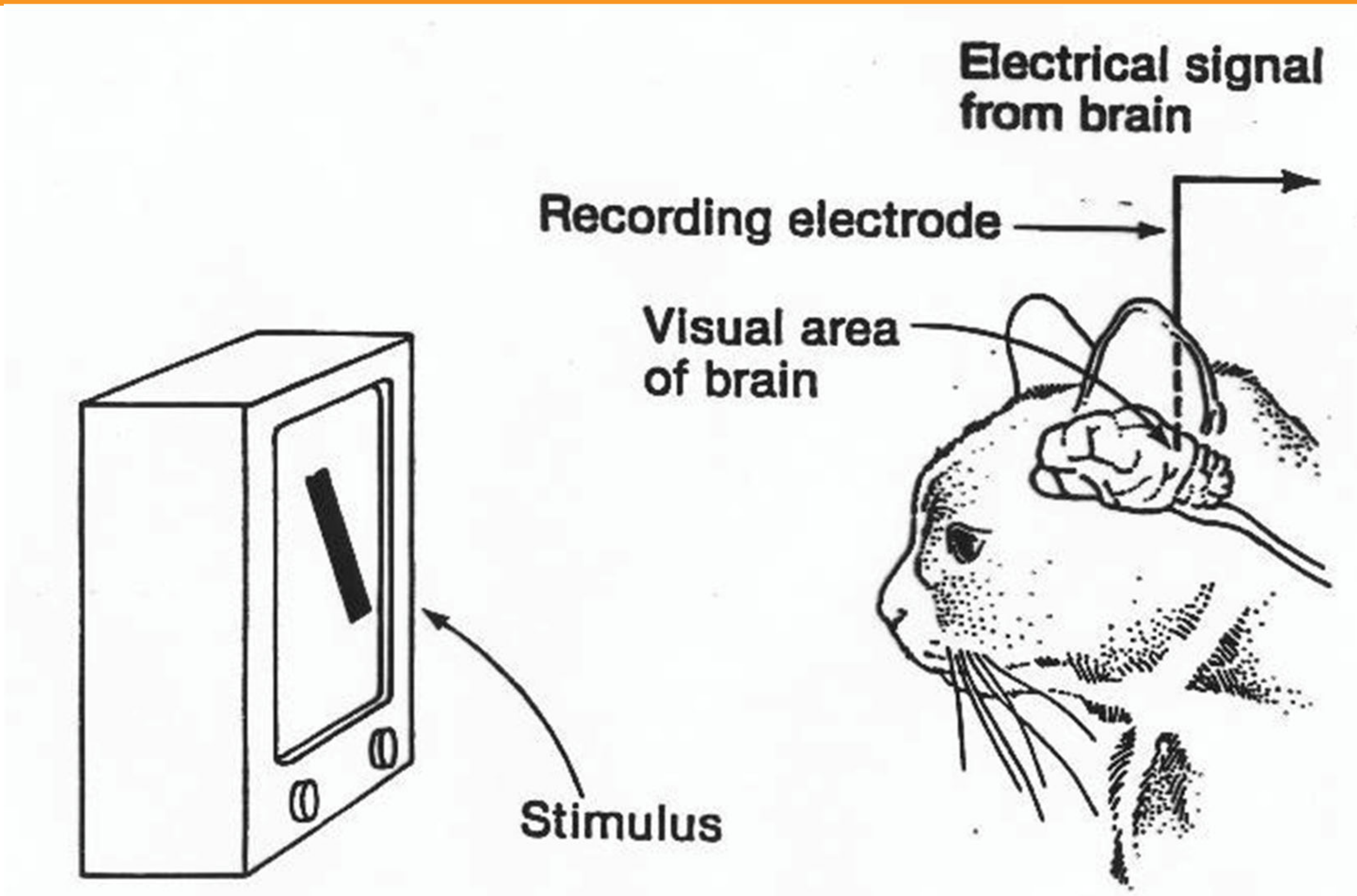
- recommendation systems
- cohort grouping
- determining supervised learning labels and features

Deep Learning with ANN

- TensorFlow
- Keras

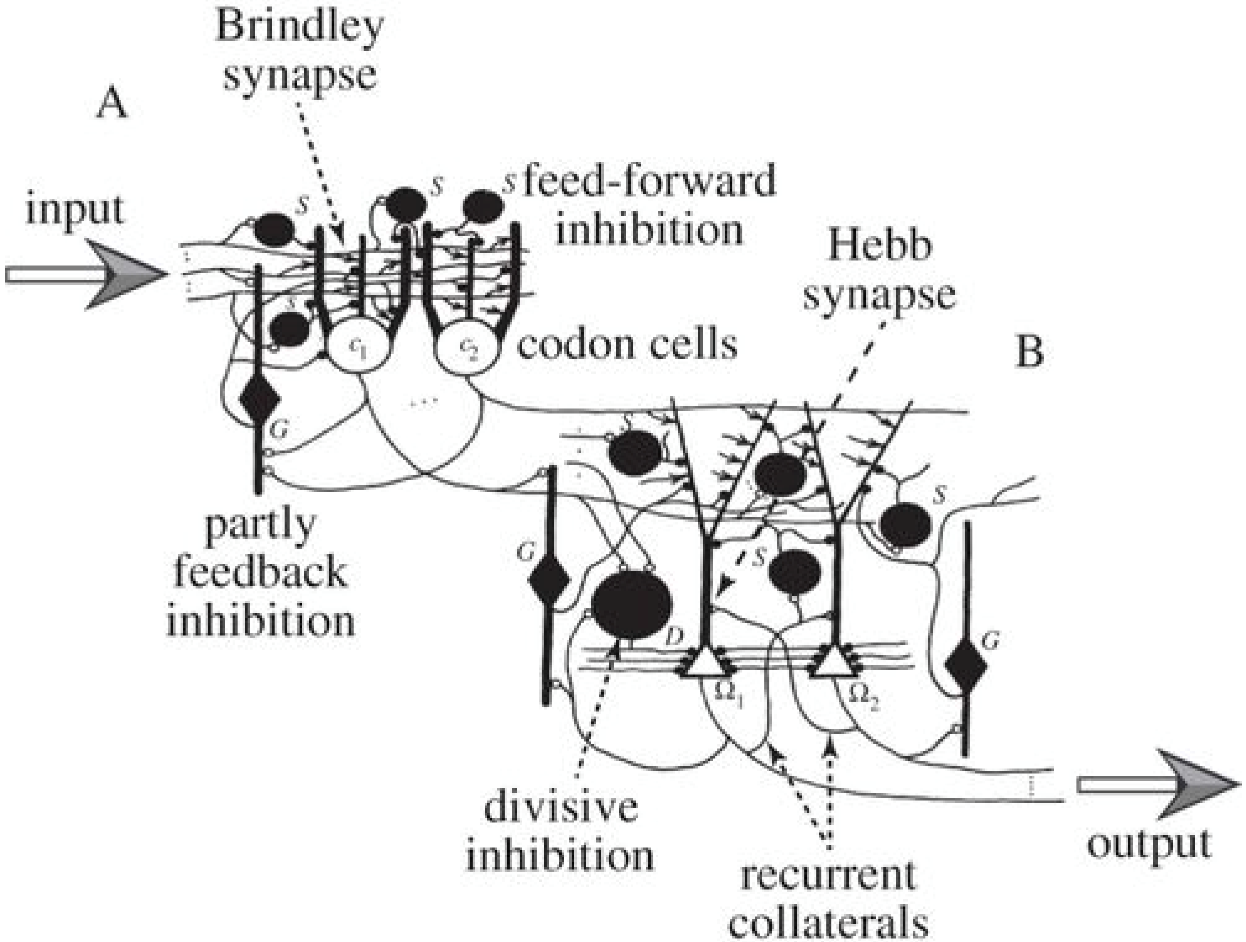
- Approach
- New Territory

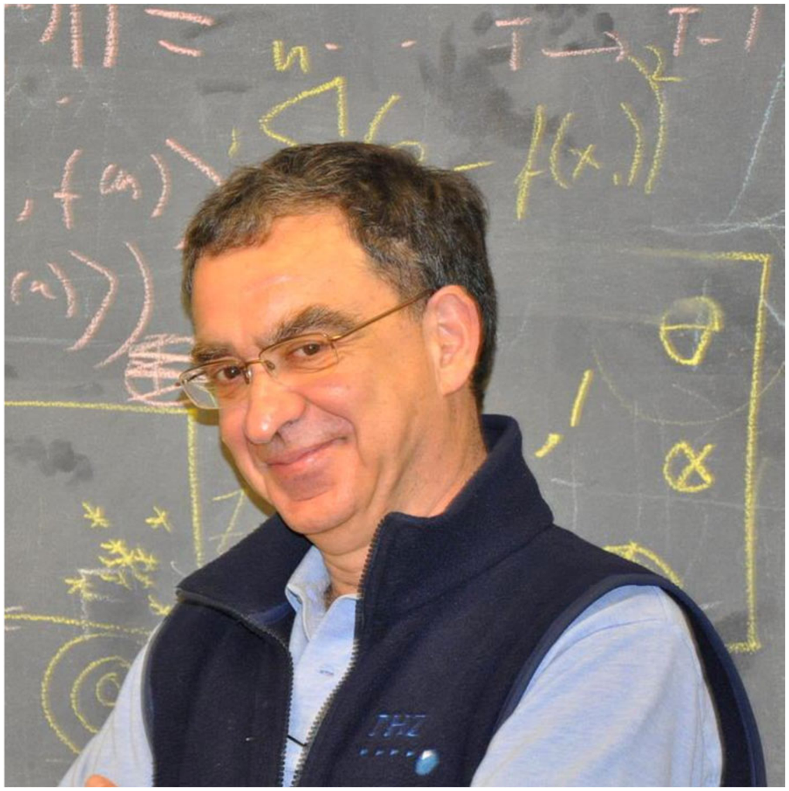
Natural Cat Vision 1959



Computer Vision 1959 pt 2

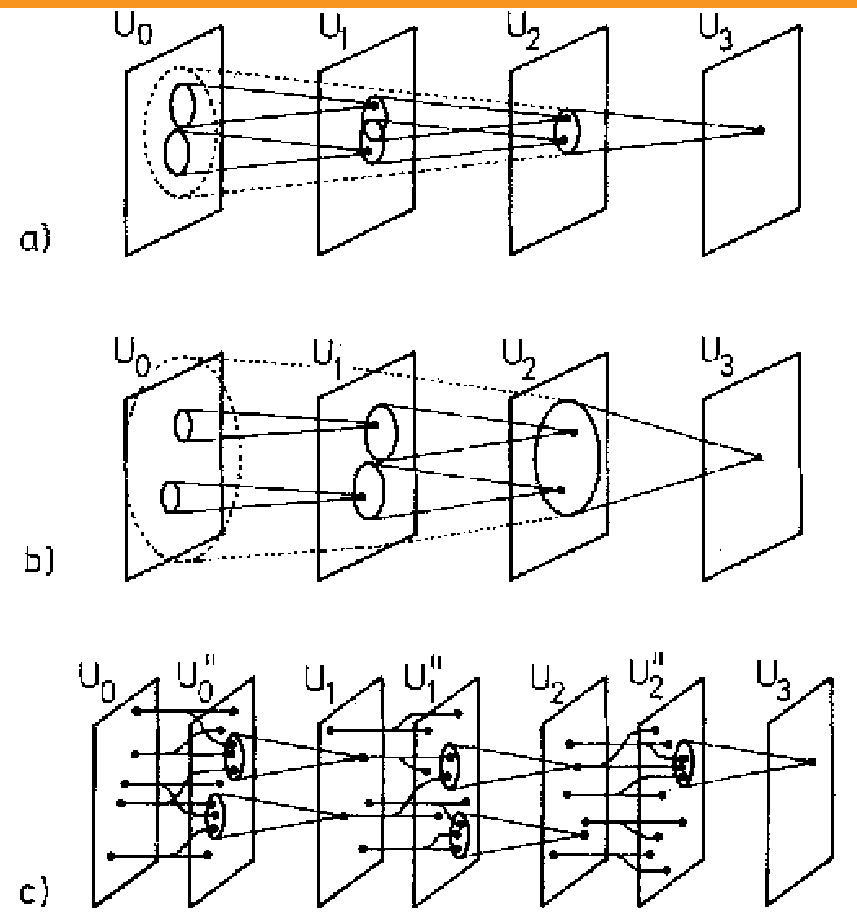
Computer Vision 1982 – Deeper than Cats





Cognitron – First MultiLayered ANN

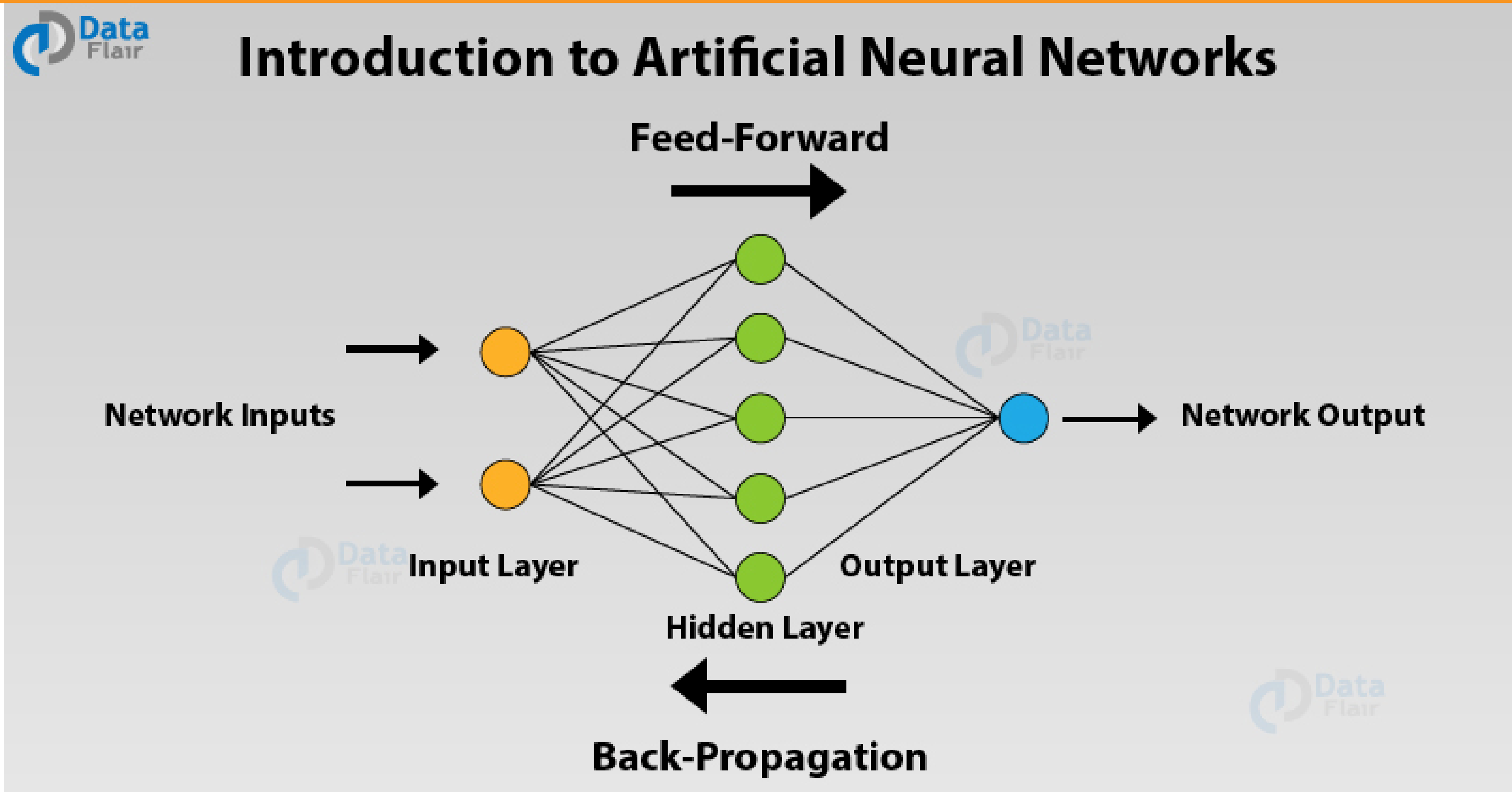
- Fukushima, Kunihiko



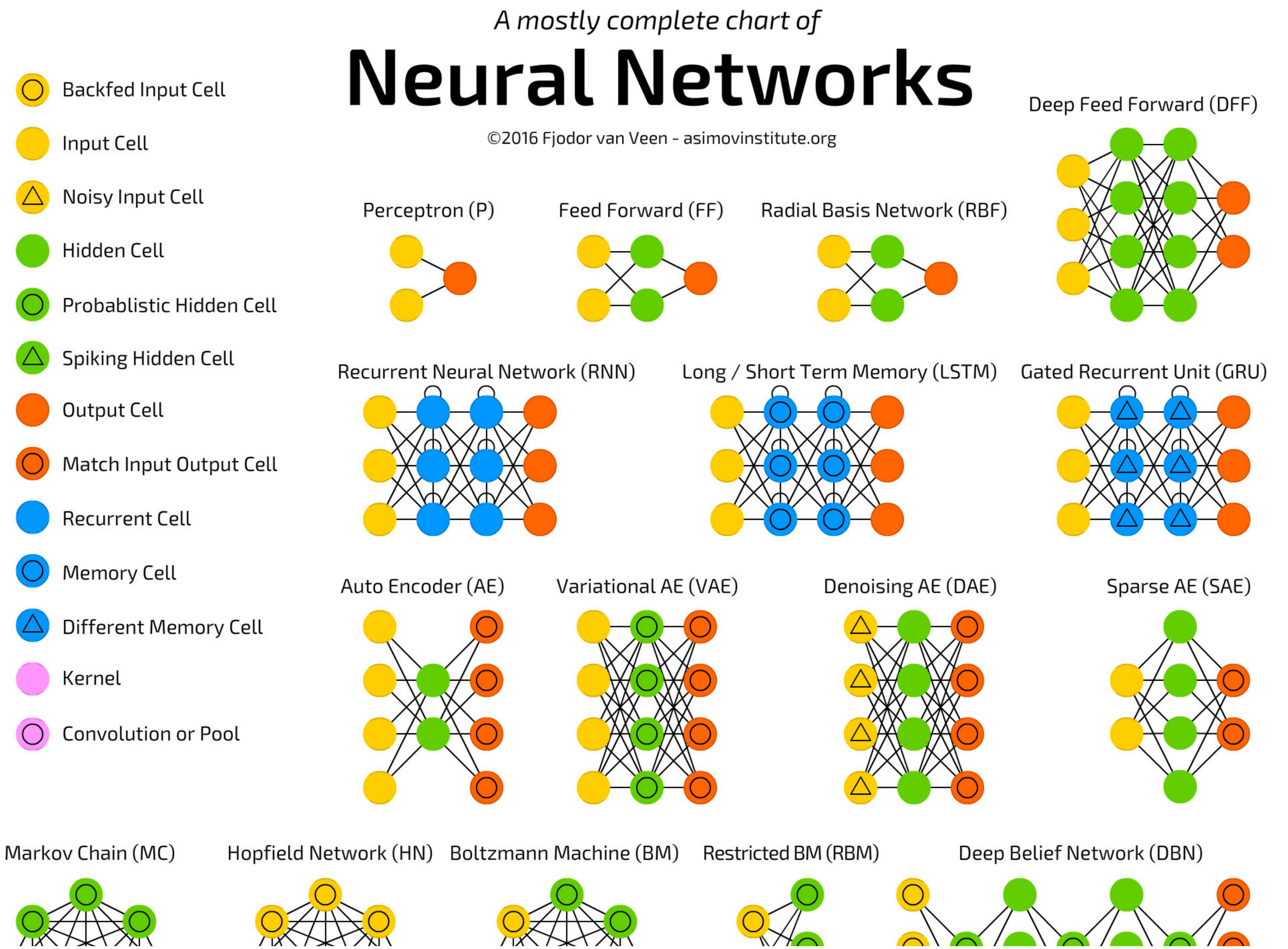
AlexNet

- AlexNet is the name of a convolutional neural network (CNN), designed by Alex Krizhevsky,[1] and published with Ilya Sutskever and Krizhevsky's doctoral advisor Geoffrey Hinton.[2][3]

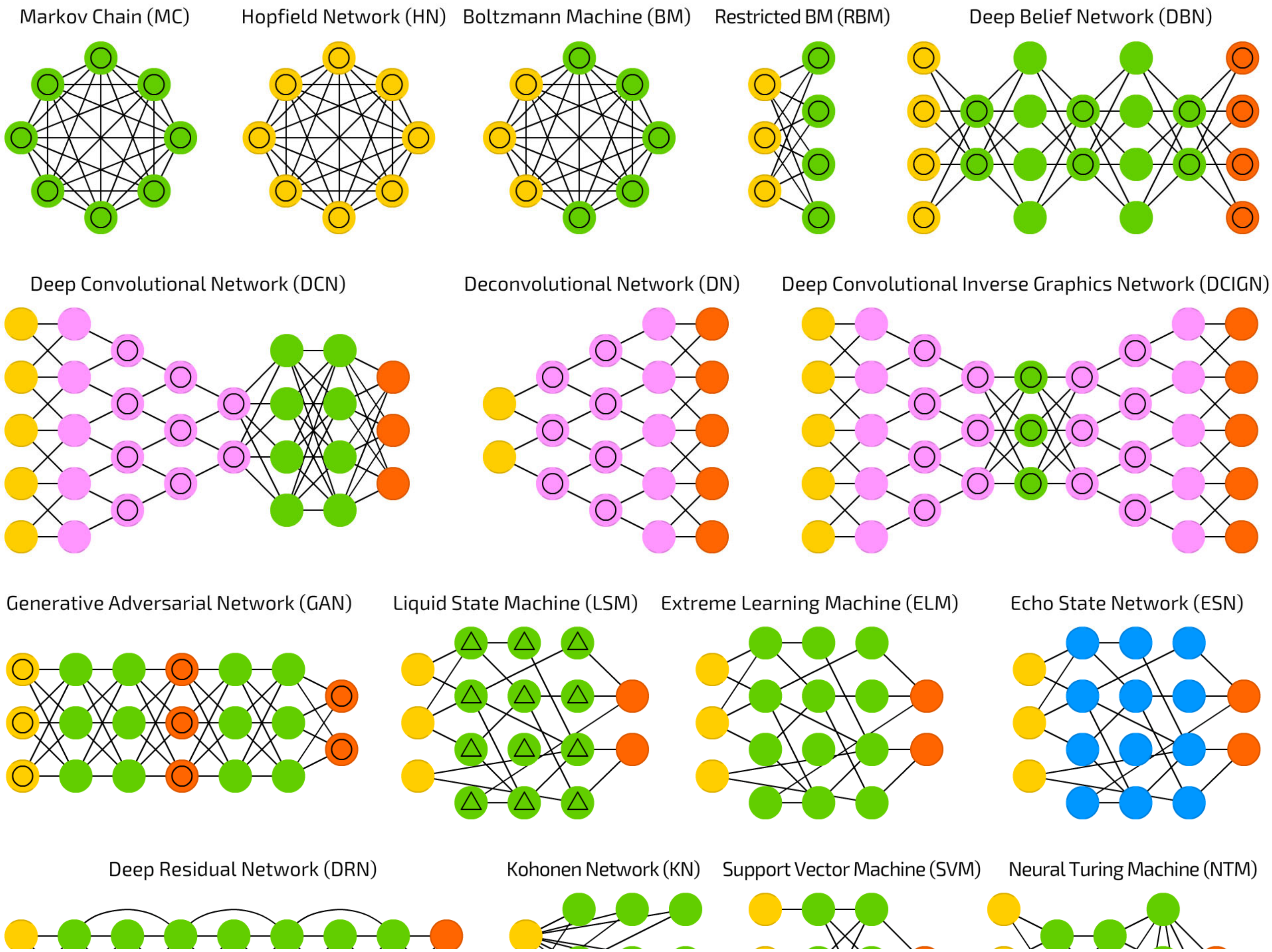
Artificial Neural Networks



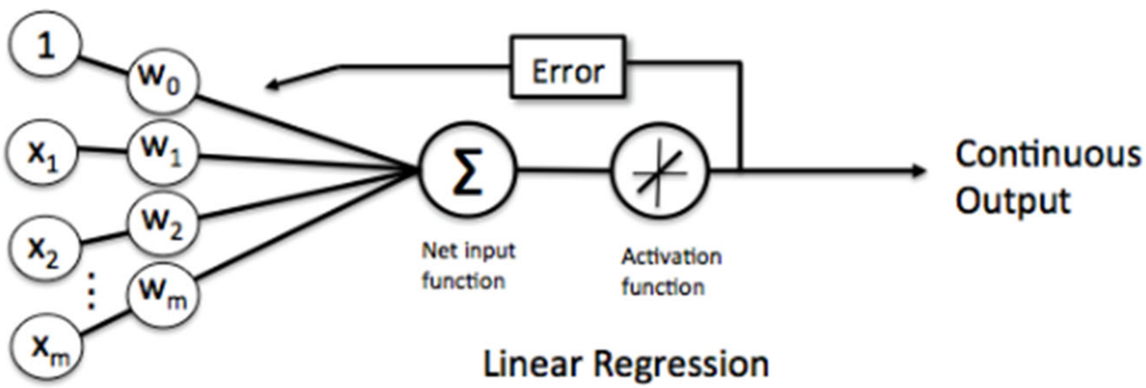
ANN Archite


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)

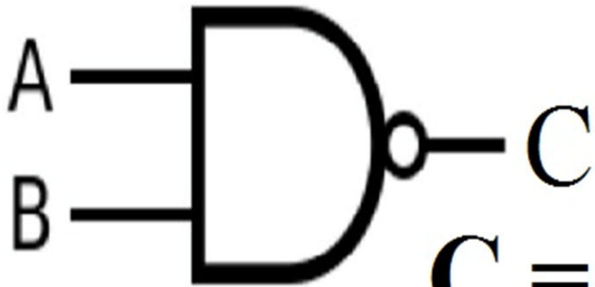
ANN Archite


Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Linear Regression Formula is Same as One Node



NAND GATE

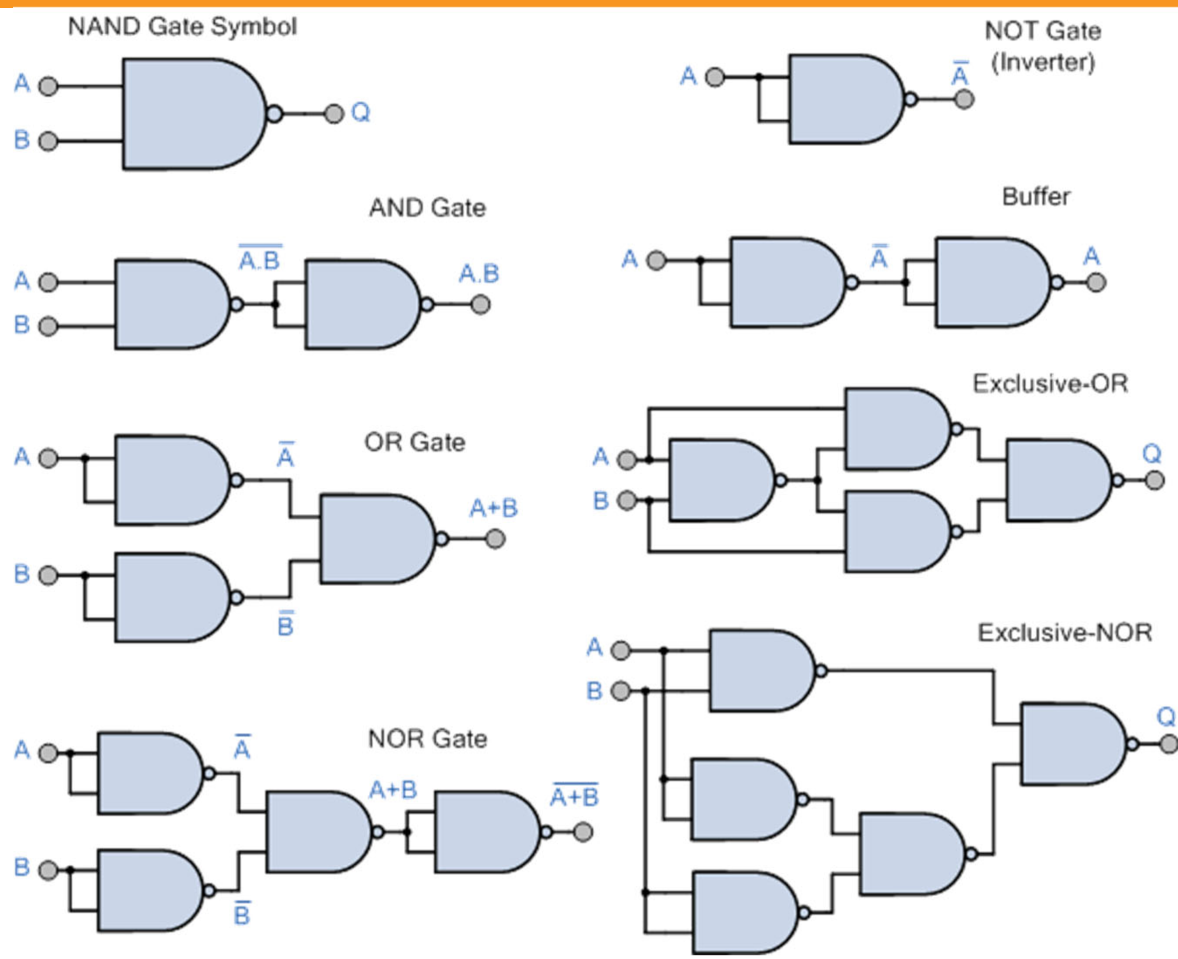


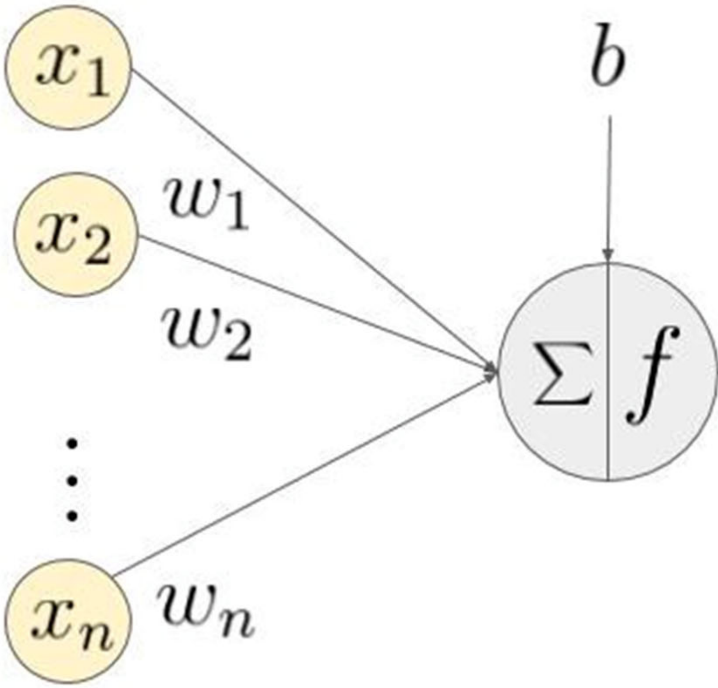
$$C = \overline{A.B}$$

Truth Table

INPUT		OUTPUT
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

Build Any Logic By Chaining them Together

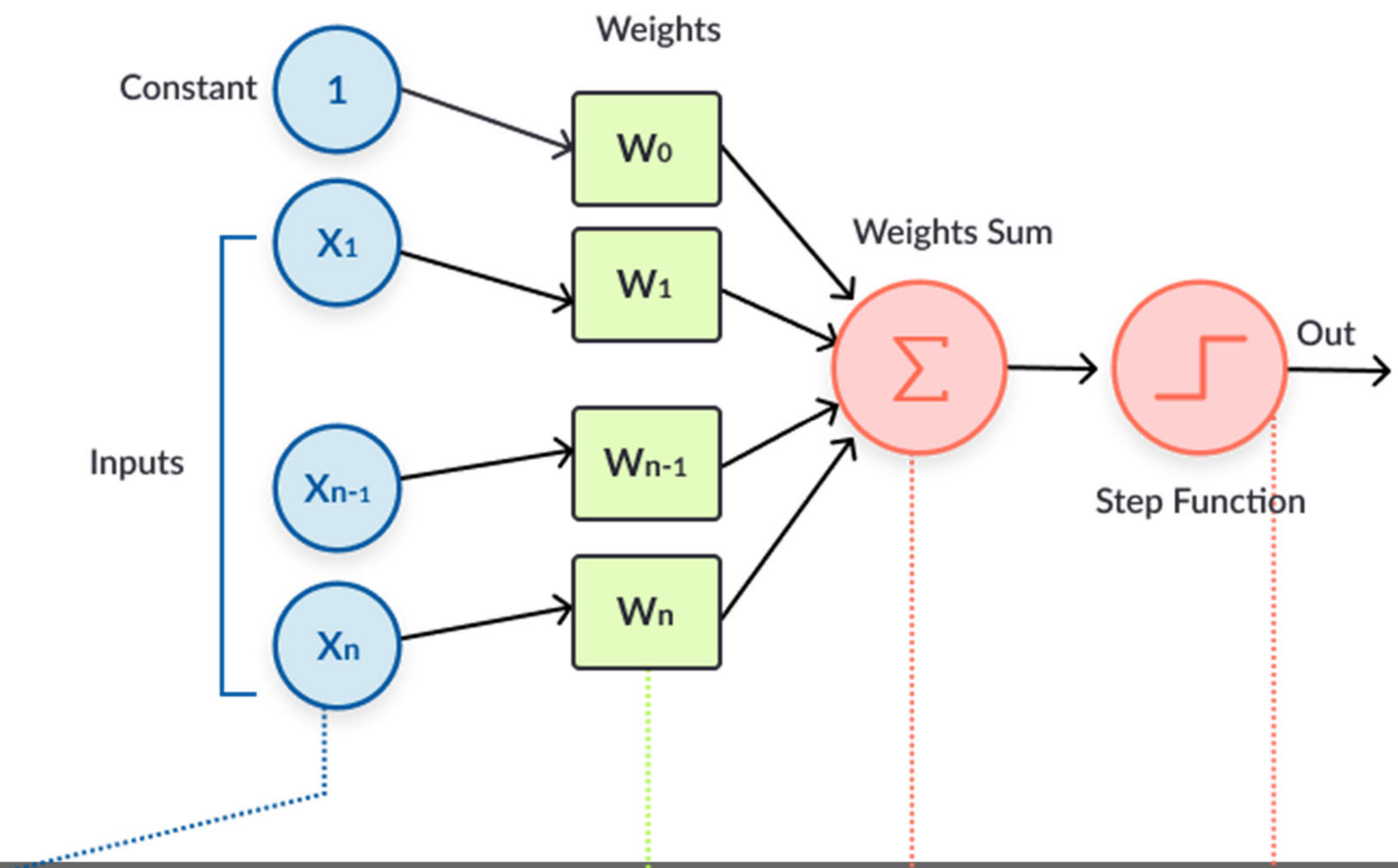




$$f\left(b + \sum_{i=1}^{n} x_i w_i\right)$$

An example of a neuron showing the input (x_1 - x_n), their corresponding weights (w_1 - w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.

Each Neuron now has multiple inputs and a Threshold Output

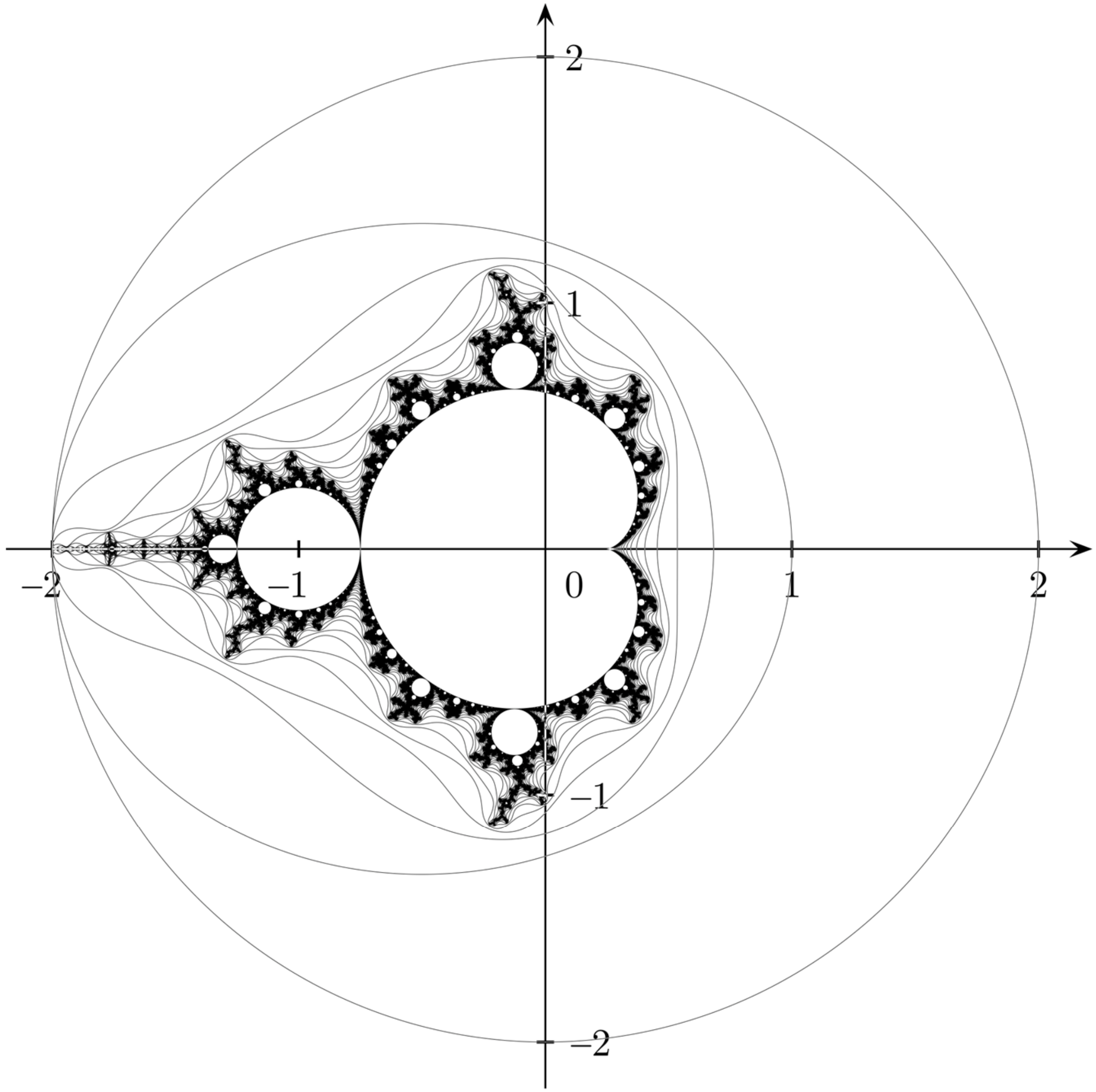


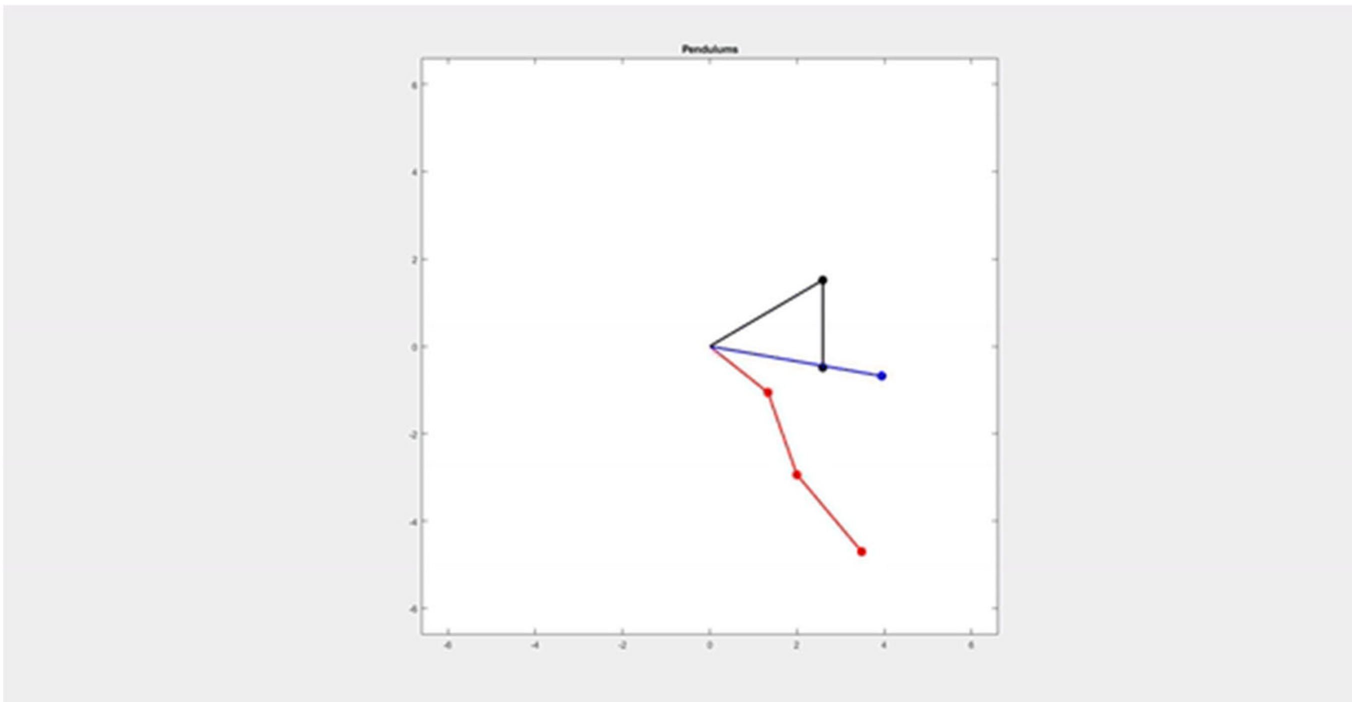
The Mandelbrot Set

- The Mandelbrot set is a set of points in the complex plane, the boundary of which forms a fractal.
- Mathematically, the Mandelbrot set can be generated using a very simple iterative formula, called the quadratic recurrence equation, applied to points in complex plane

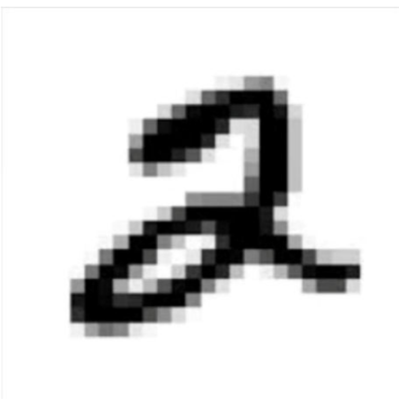
$$z_{n+1} = z_n^2 + c$$

That is, a complex number c, is in the Mandelbrot set if the absolute value of z_n never exceeds a certain number.



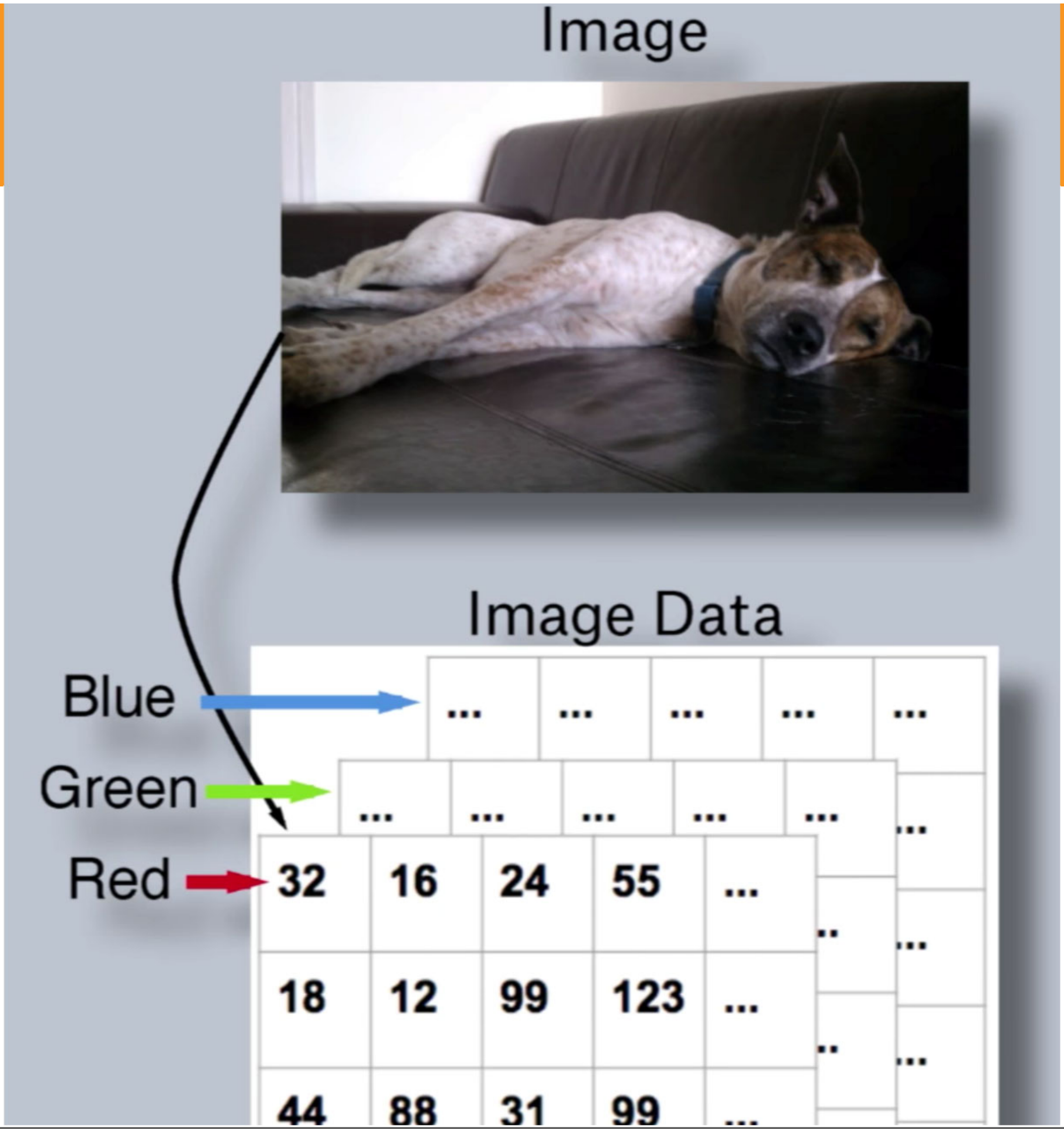

Pendulums

How Images Are Stored



0	0	200	150	0	0
0	143	55	99	222	0
0	188	0	0	181	0
0	0	0	200	0	0
0	0	149	0	0	0
0	245	202	140	225	0
0	0	0	0	0	0

Color Images



2D

0	0	200	150	0	0
0	143	55	99	222	0
0	188	0	0	181	0
0	0	0	200	0	0
0	0	149	0	0	0
0	245	202	140	225	0
0	0	0	0	0	0

Convolutions

1.5	1.5
-1.5	-1.5

Convolution Example 1

1.5	1.5
-1.5	-1.5

Data

200	200	...	...	...
200	200	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

$$1.5 * 200 + 1.5 * 200 - 1.5 * 200 - 1.5 * 200 = 0$$

Example 2

1.5	1.5
-1.5	-1.5

Data

0	0	...	...	...
0	0	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

Example 3 Dark over Light.

1.5	1.5
-1.5	-1.5

Data

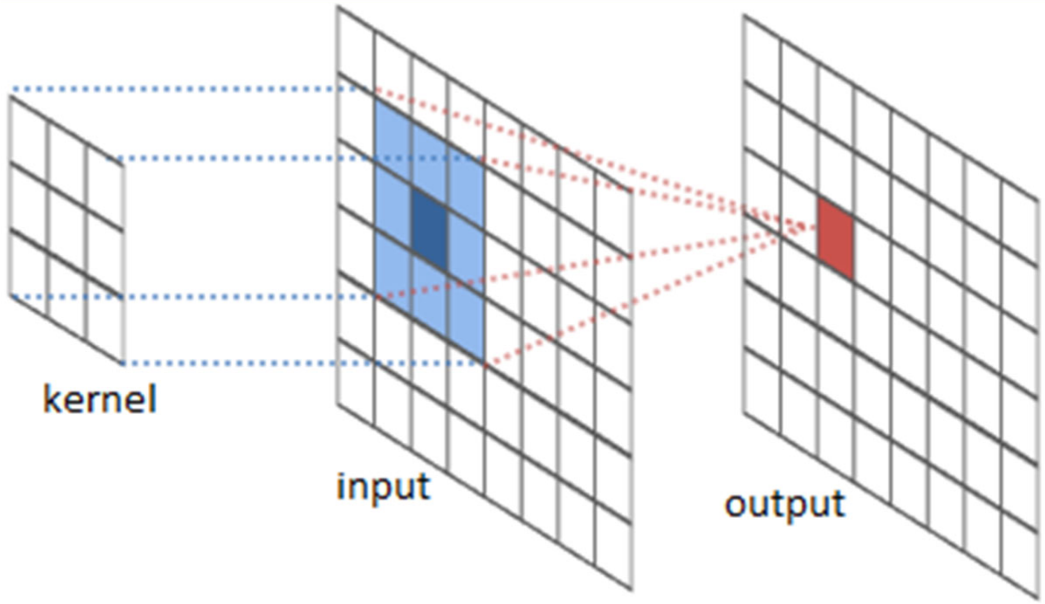
200	200	...	...	...
0	0	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

$$1.5 * 200 + 1.5 * 200 + 0 + 0 = 600$$

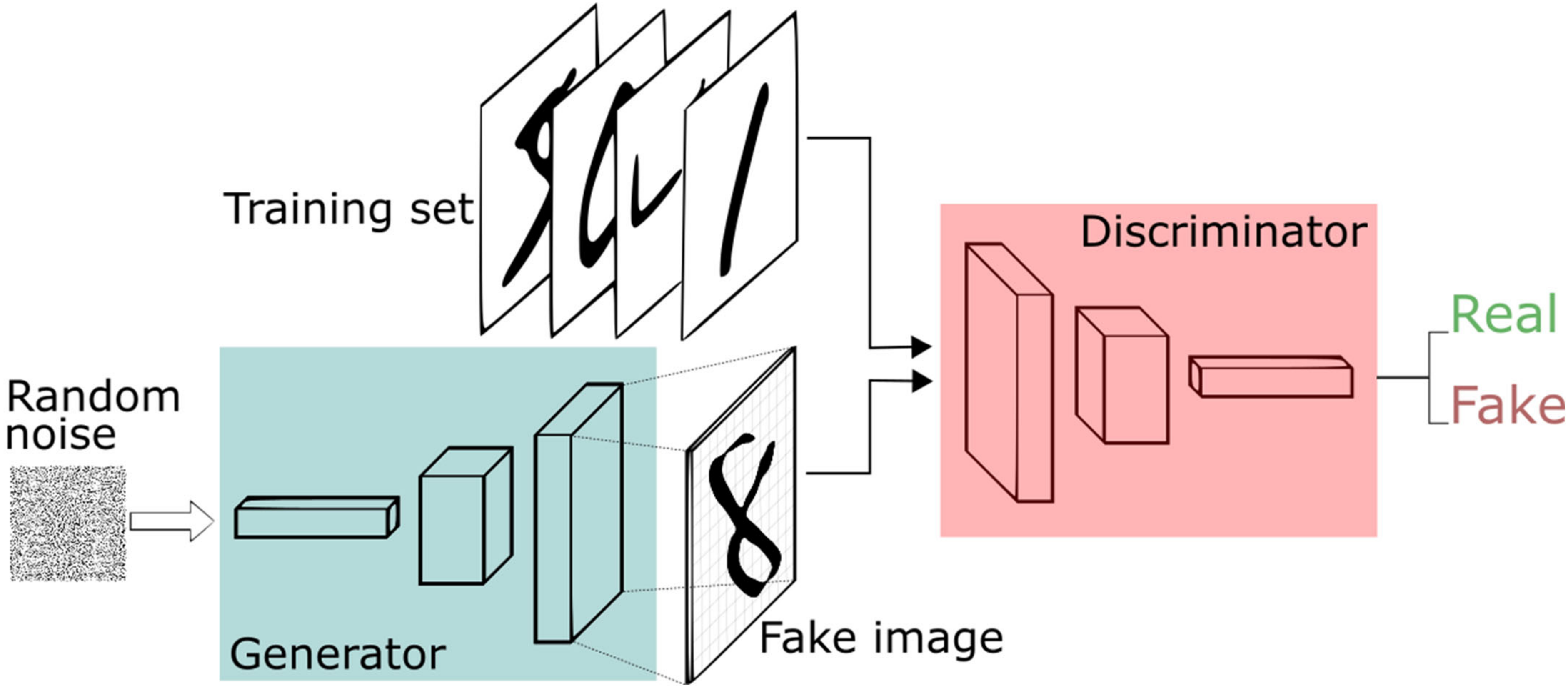
Exercise - Computer Vision

- You saw a convolution that detected horizontal lines. That convolution shows up again in this code.

- What do we need to change to complete the vertical line portion?



GAN (Generative Adversarial Networks



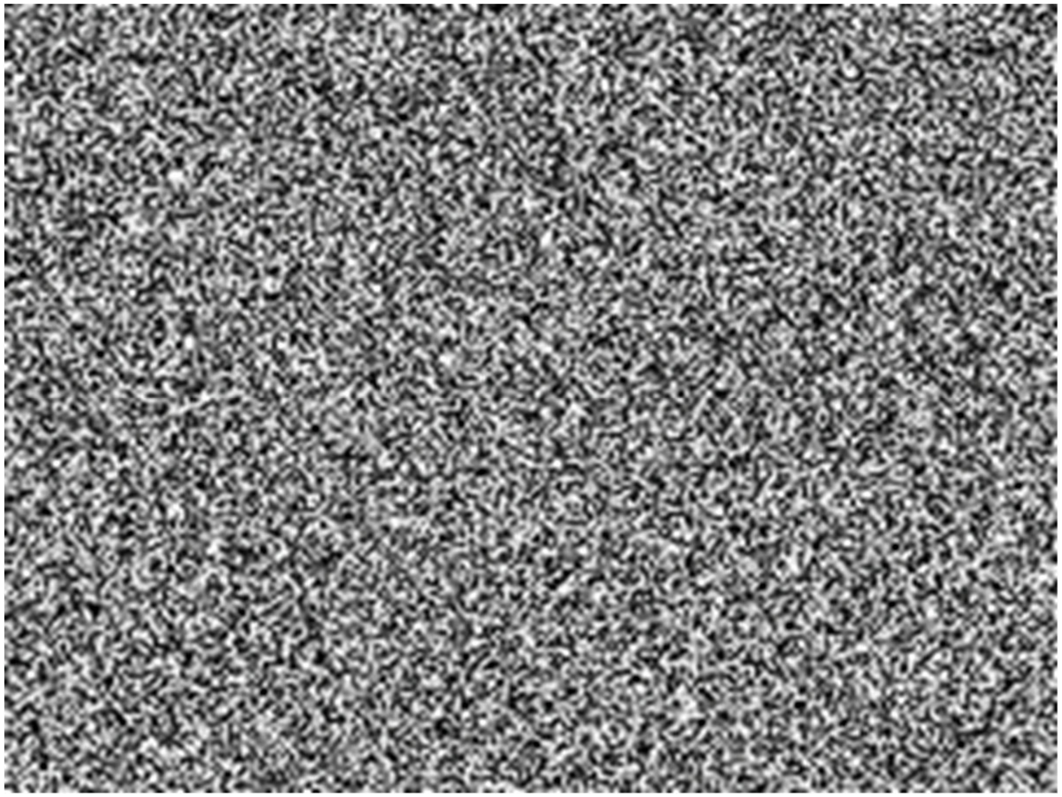
GAN (Joke)

"They don't appear to want to take over. They just want to dance."

How GANs Work

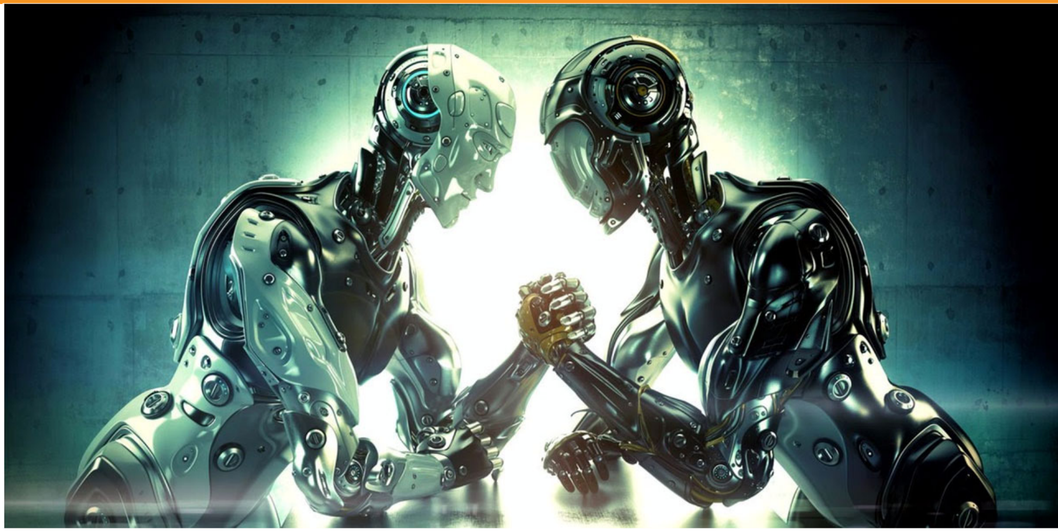
Generative

- The **Generator** is a model that takes an input as a random noise signal and then outputs an image.



Adversarial

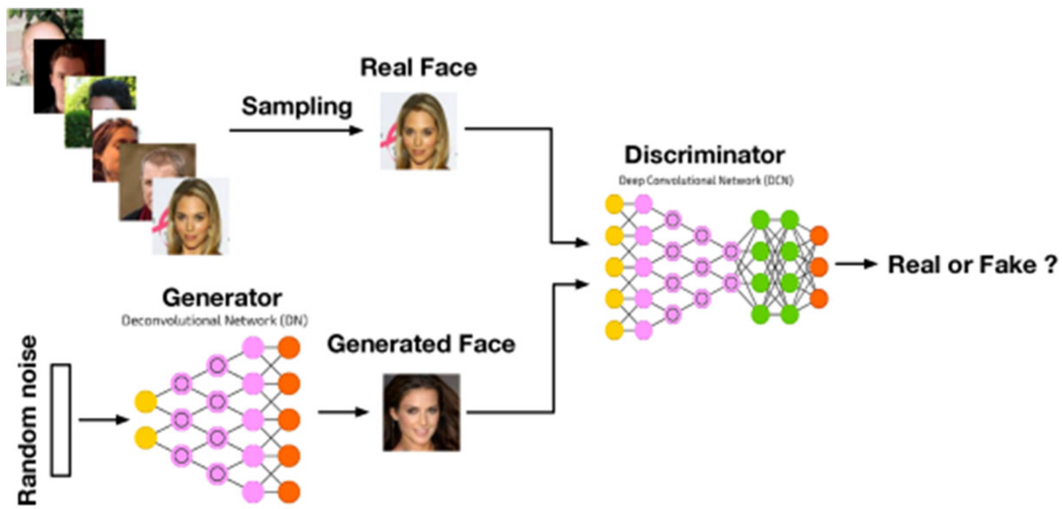
- The Discriminator is the adversary of the Generator.
- The Discriminator is capable of learning about objects, animals or other features specified. For example: if you supply it with pictures of dogs and non-dogs, it would be able to identify the difference between the two.



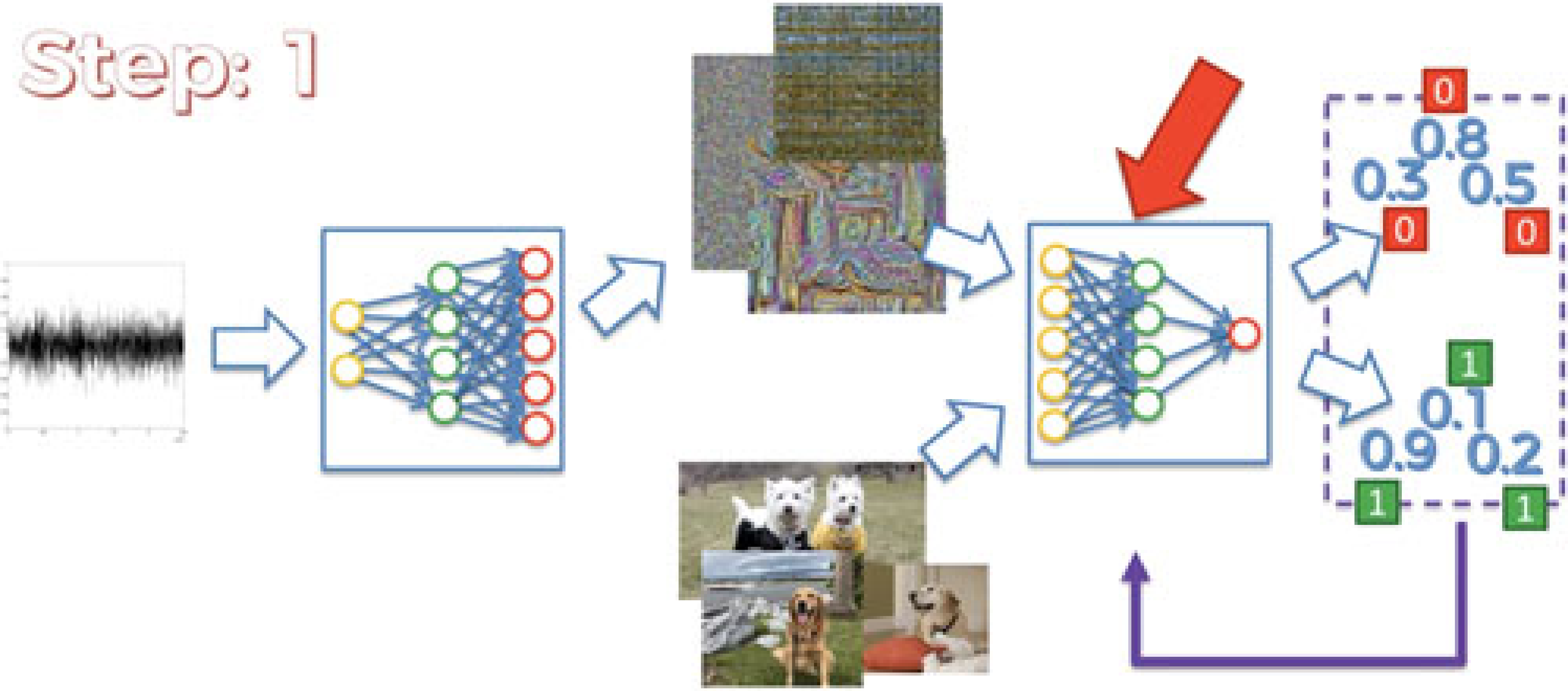
- Using this example, once the discriminator has been trained, showing the discriminator a picture that isn't a dog, it will return a 0. Whereas, if you show it a dog it will return a 1.

Network

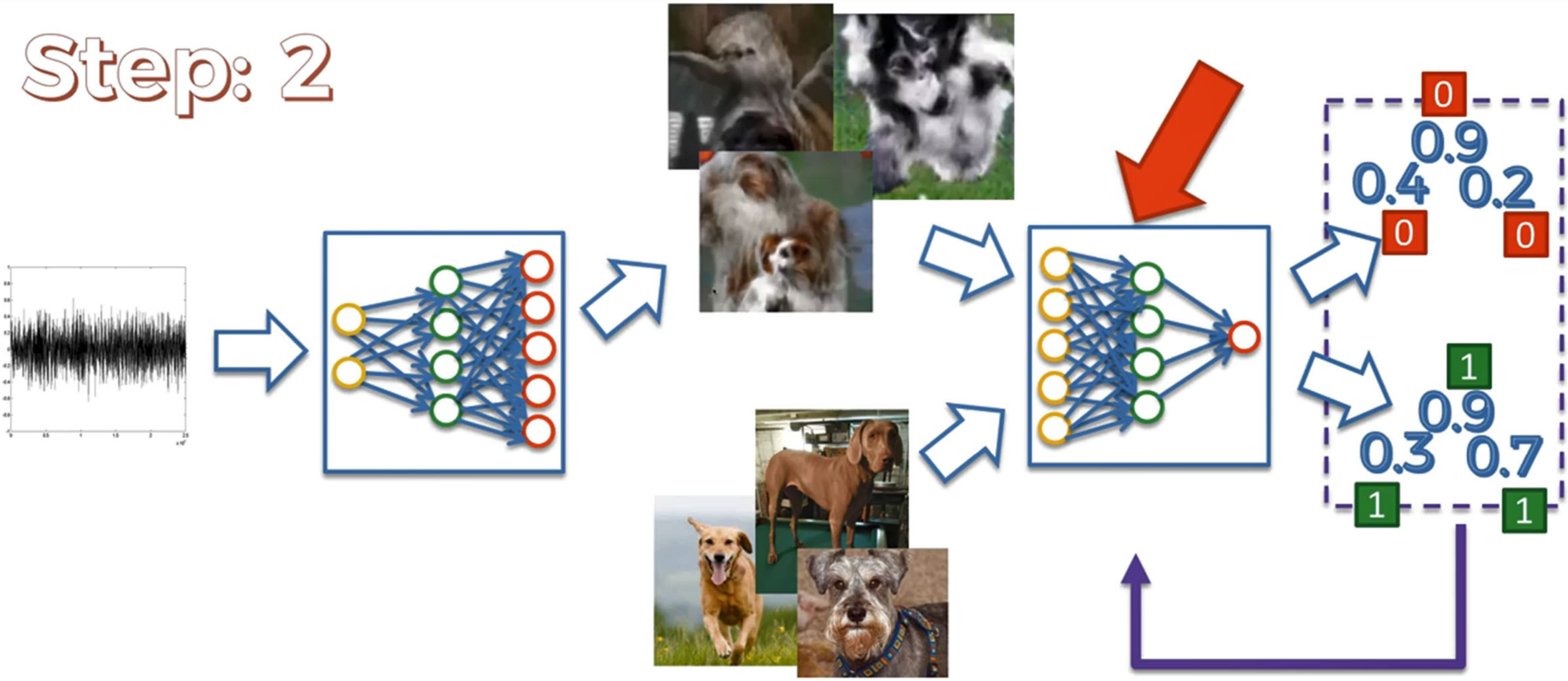
- Network
  - Meaning the generator and discriminator are both neural networks.



First Step



Second Step



Examples

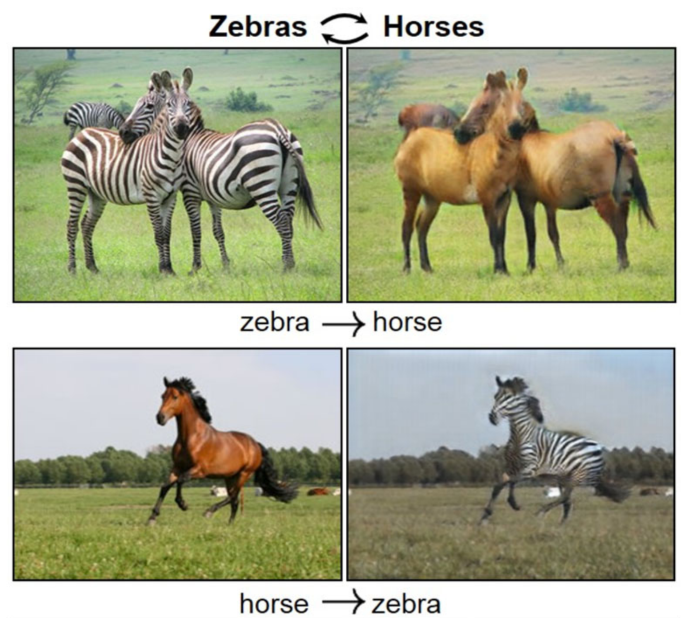
GANs can be used for the following:

- Generating Images
- Image Modification
- Super Resolution
- Assisting Artists
- Photo-Realistic Images
- Speech Generation
- Face Ageing

NVIDIA

It's Training Cats and Dogs:

NVIDIA Research Uses AI to Turn Cats Into Dogs, Lions, and Tigers, Too!

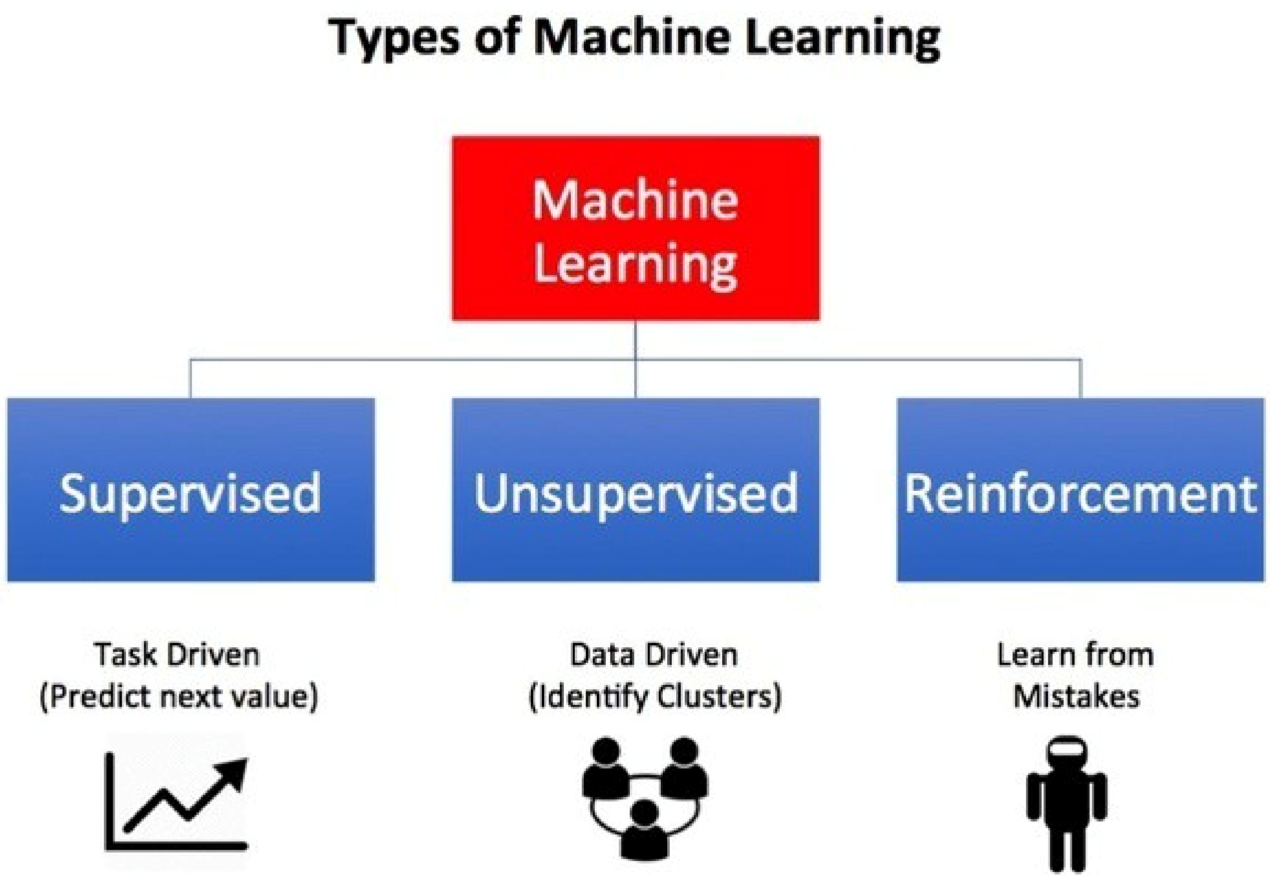


GAN Project Demo

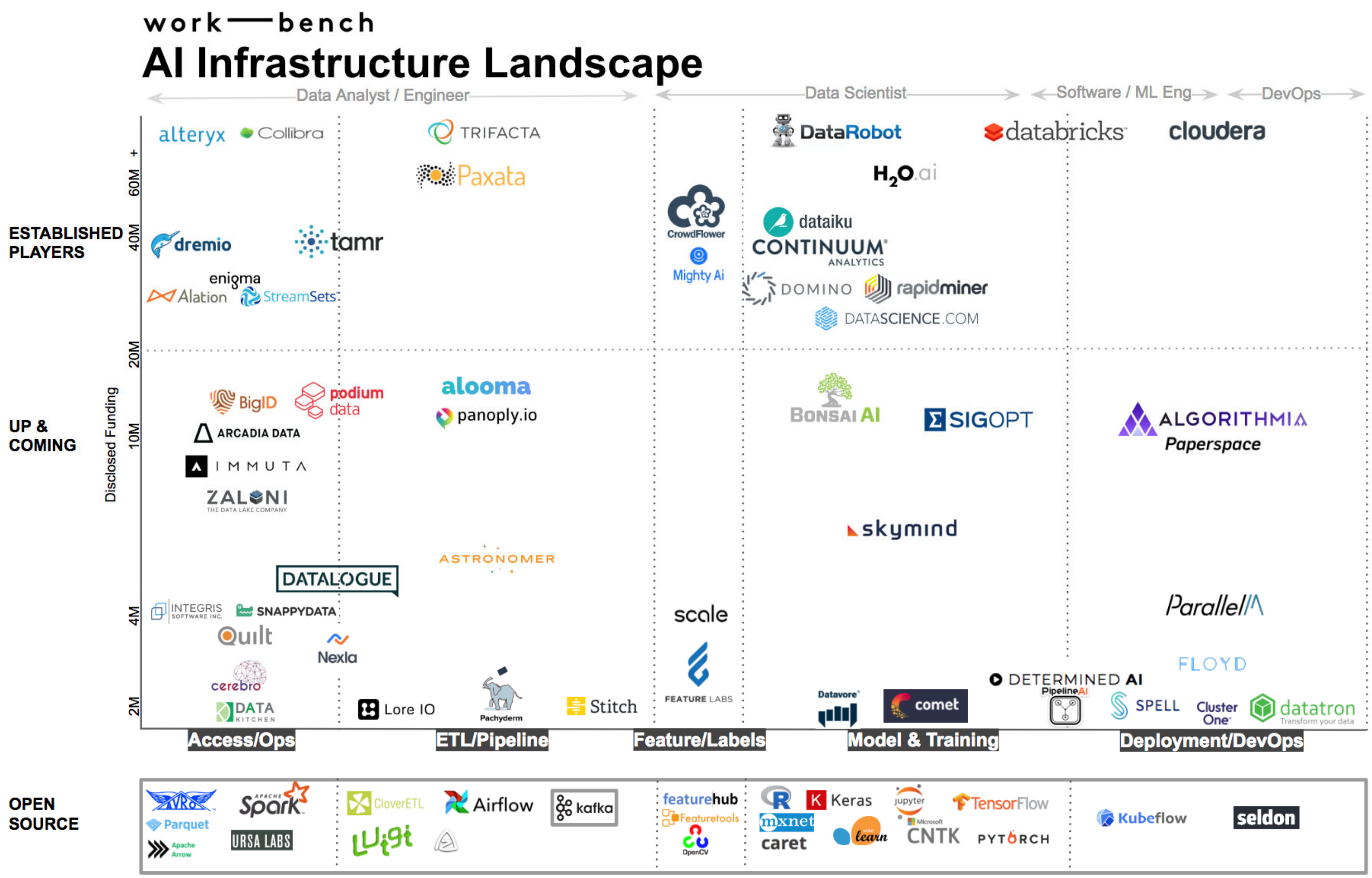
- Let's generate fake digits!

Reinforcement Learning

- Strategy
- Thinking Ahead
- Delayed Reward



Tool Landscape



Resources for Further Study

- **Scientific Python Ecosystem**
  - NumPy User Guide
  - SciPy Tutorial
  - Matplotlib beginner's guide
  - pandas tutorials
  - SymPy tutorial

- **General Machine Learning**
  - https://towardsdatascience.com/https://www.kaggle.com/
  - https://www.tensorflow.org/resources/learn-ml
- Awesome
  - https://www.youtube.com/channel/UCbfYPyITQ-7l4upoX8nvctg