NASA
SPACE APPS
CHALLENGE

Intro to Artificial Intelligence, & Machine Learning

October 10, 2018

Huntsville AI - Facebook Group

What we cover:

- Application - how to solve a problem with a technology
- Theory - for those times when knowing “How” something works is necessary
- Social / Ethics - Human / AI interaction
- Brainstorming - new uses for existing solutions
- Hands on Code - for those times when you just have to run something for yourself
- Coworking Night - maybe have combined sessions with other groups to discuss application in their focus areas

https://www.facebook.com/groups/390465874745286/

About me...

J. Langley

Chief Technical Officer at CohesionForce, Inc & Founder of Session Board

Involved in Open Source (Eclipse & Apache Foundations)

Started playing with AI about 15 years ago when Intelligent Agents were all the rage.

Developed a Naive Bayes approach for text classification, a Neural Network for audio classification, heavily into NLP.

Intro to a REALLY BIG TOPIC

So…

Artificial Intelligence and Machine Learning are pretty big topics to cover.

This session will attempt to help you categorize the types of problems that AI is best suited to solve.

We will cover a bit of history to give an idea of how fast things are changing.

We will also cover several of the tools and resources used by people working in this field.

Intro to a REALLY BIG TOPIC

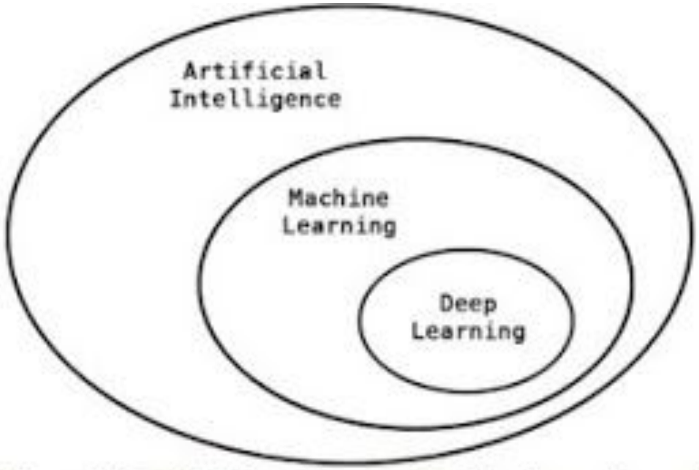


Figure 1.1 Artificial Intelligence, Machine Learning and Deep Learning

Here’s a useful way to think about the relationship between AI, ML, and Deep Learning.

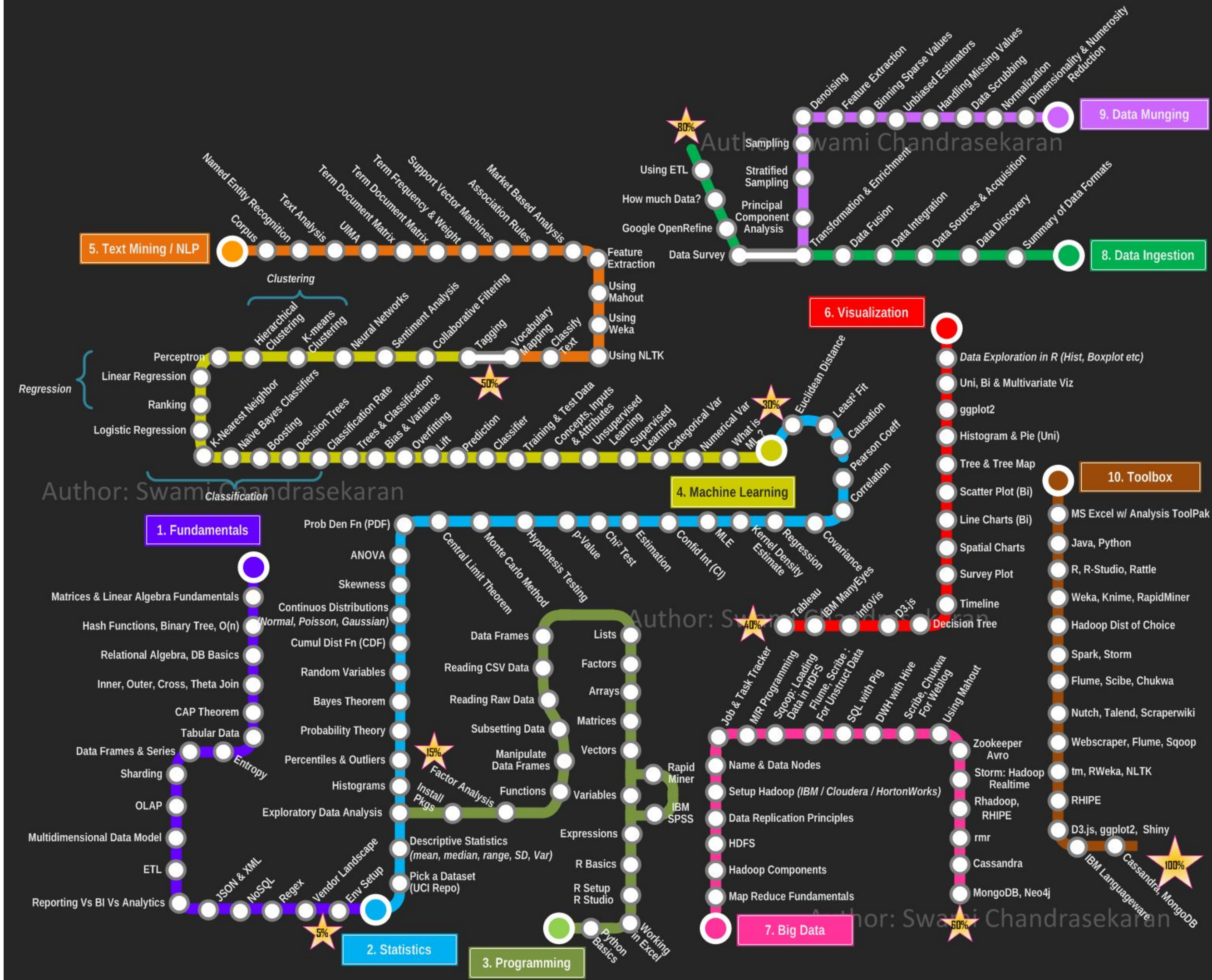
Another way is to break it down into two approaches:

1. Supervised Learning
2. Unsupervised Learning

Graphic from:
https://sqlandsiva.blogspot.com/2017/07/day-70-machine-learning-deep-learning.html

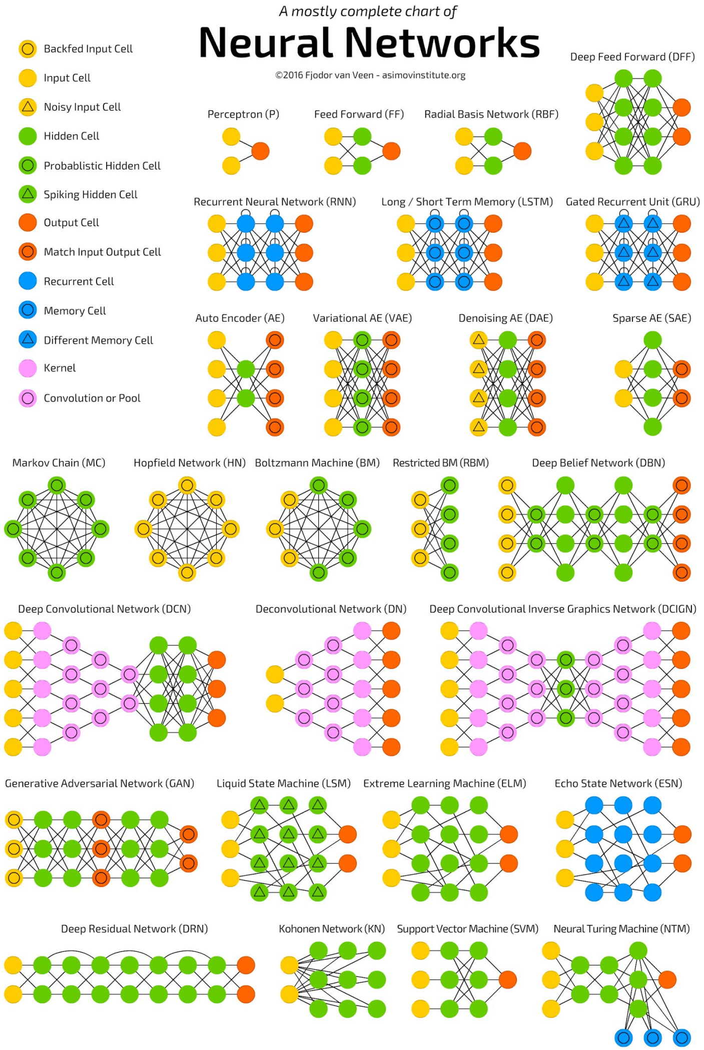
Here’s the part where I put up an eye chart and everyone pretends that they can read it.

Each node on this subway map is marked with an AI/ML topic that you may want to learn.



You will soon learn that you are no match for my eye charts...

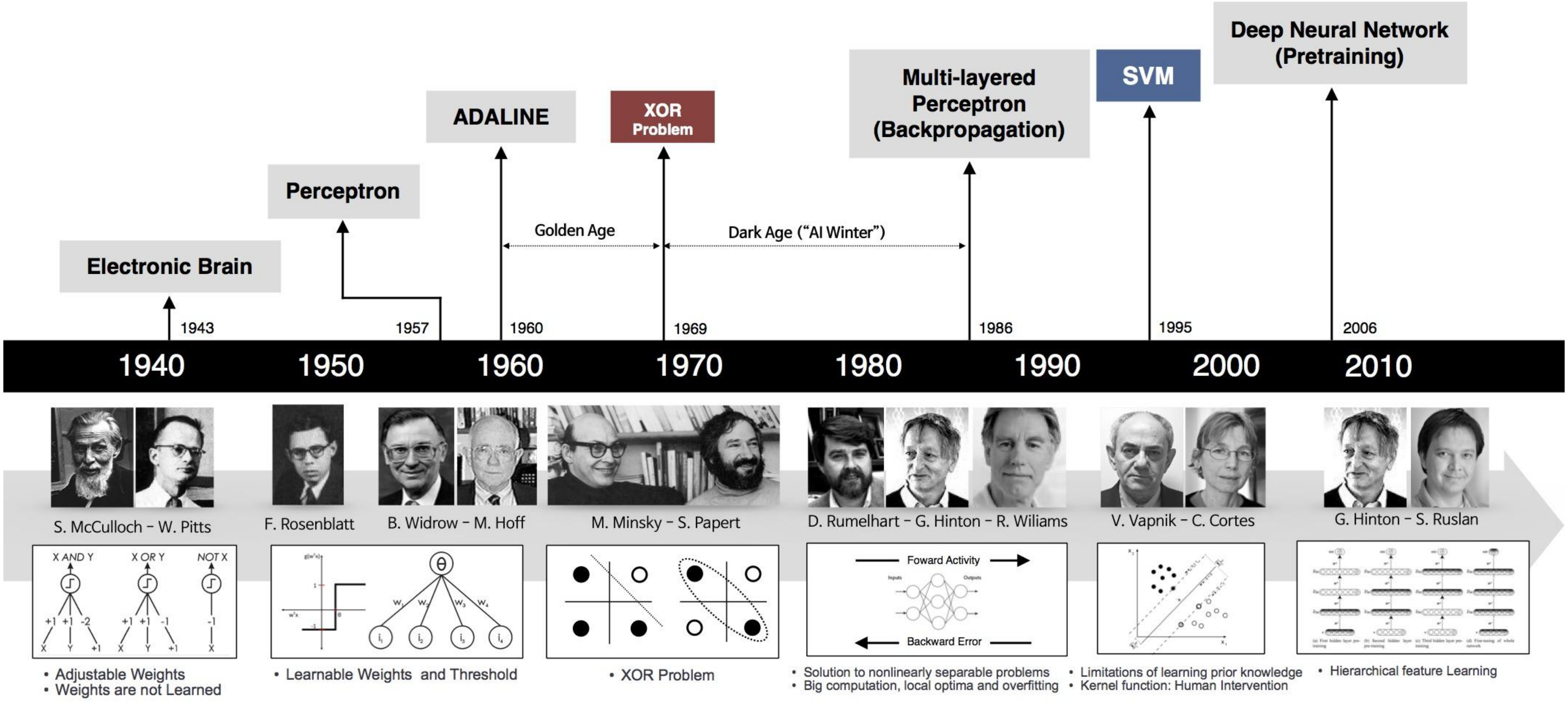
Here’s a diagram of existing Neural Network architectures from 2016.


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)
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)

But first, let’s time travel!

How about a shallow history of deep learning?

Deep Learning History



Electronic Brain — 1943

Perceptron — 1957

ADALINE — 1960

XOR Problem — 1969

Multi-layered Perceptron (Backpropagation) — 1986

SVM — 1995

Deep Neural Network (Pretraining) — 2006

Golden Age

Dark Age ("AI Winter")

1940 1950 1960 1970 1980 1990 2000 2010

S. McCulloch - W. Pitts

F. Rosenblatt

B. Widrow - M. Hoff

M. Minsky - S. Papert

D. Rumelhart - G. Hinton - R. Wiliams

V. Vapnik - C. Cortes

G. Hinton - S. Ruslan

- Adjustable Weights
- Weights are not Learned

- Learnable Weights and Threshold

- XOR Problem

- Solution to nonlinearly separable problems
- Big computation, local optima and overfitting

- Limitations of learning prior knowledge
- Kernel function: Human Intervention

- Hierarchical feature Learning

Beginnings

Alan Turing - Computing Machinery and Intelligence (1950)

https://www.csee.umbc.edu/courses/471/papers/turing.pdf

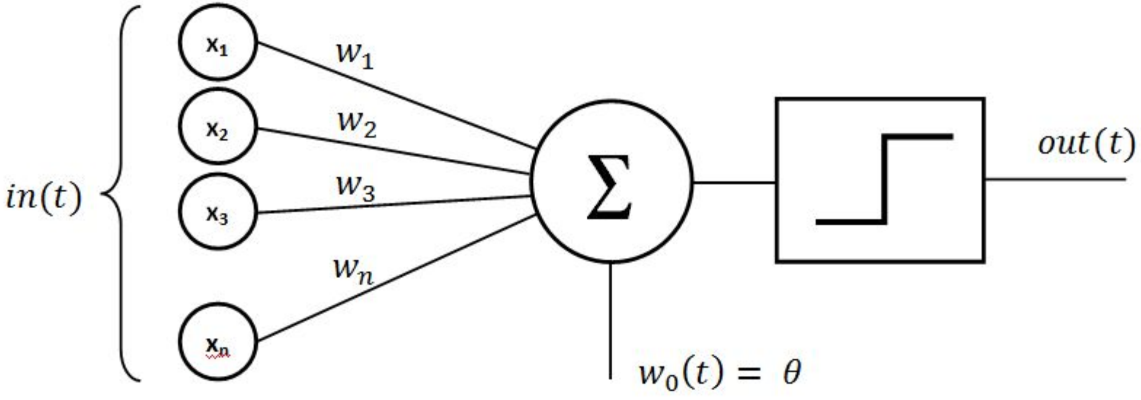
This paper introduce the concept of what is now known as the “Turing Test”

The state of the art at the time was the Threshold Logic Unit - mostly based on current knowledge of how neurons were thought to work.

Perceptron

The perceptron algorithm was invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt,funded by the United States Office of Naval Research The perceptron was intended to be a machine, rather than a program, and while its first implementation was in software for the IBM 704, it was subsequently implemented in custom-built hardware as the "Mark 1 perceptron". This machine was designed for image recognition: it had an array of 400 photocells, randomly connected to the "neurons". Weights were encoded in potentiometers, and weight updates during learning were performed by electric motors

.



AI Winter (1969)

Cause of AI Winter

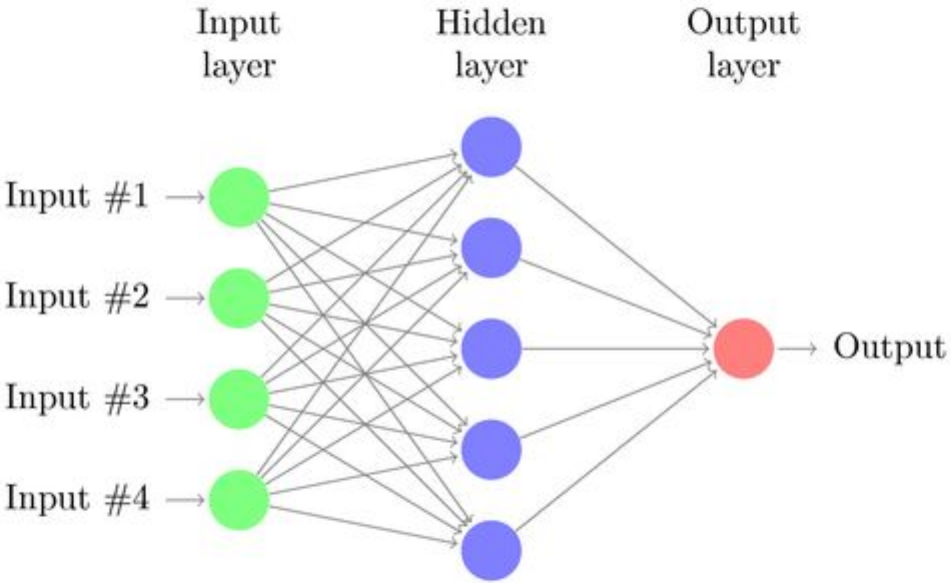
Perceptrons: an introduction to computational geometry is a book written by Marvin Minsky and Seymour Papert and published in 1969.

It offered a mathematical proof that the perceptron could not approximate an XOR function given an infinite training set.

I never liked XOR anyway.

Multilayer Perceptrons

By stacking several layers of perceptrons, researchers were able to overcome the XOR problem.



Backpropagation (1986)

Geoff Hinton, along with David Rumelhart and Ronald Williams, published a paper entitled “Learning representations by back-propagating errors”

https://www.iro.umontreal.ca/~vincentp/ift3395/lectures/backprop_old.pdf

This provided a mechanism for training multilayer perceptron networks.

Also about this time, the **universal approximation theorem** states that a feed-forward network with a single hidden layer containing a finite number of neurons (i.e., a multilayer perceptron), can approximate continuous functions on compact subsets of $\mathbf{R}^n$, under mild assumptions on the activation function.

Learning before “Deep”

Gradient based learning (1998) - Yann Lecun -

http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

CNN from Yann Lecun (AT&T Bell Labs) could recognize handwritten digits.

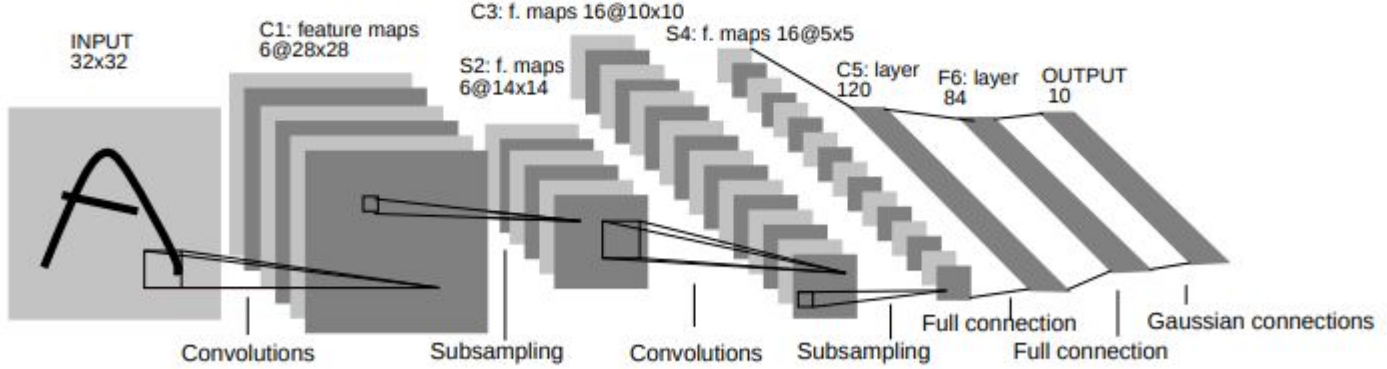


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Rolling in the Deep

Deep Learning (2006) - Again with Geoff HInton. The idea was to train a simple 2-layer unsupervised model like a restricted boltzman machine, freeze all the parameters, stick on a new layer on top and train just the parameters for the new layer.

Using this strategy, people were able to train networks that were deeper than previous attempts, prompting a rebranding of ‘neural networks’ to ‘deep learning’.

Filling the void with Hardware and Data

Imagenet (2009) - millions of labeled images created and published by

Fei-Fei Li at Stanford

MNIST - Handwritten digits

Google House Numbers from street view

Flickr 30k Image dataset

GPU - used for multi-core floating point calculation

Custom chipsets from Intel (Nervana) https://ai.intel.com/ and NVIDIA

Exponential Improvements

Alexnet (2012) - Won the Large Scale Visual Recognition Challenge(LSVRC) with an error rate 10% lower than the previous year. Used dropout to reduce overfitting and a rectified linear activation unit (ReLU)

Generative Adversarial Networks (2014) - Ian Goodfellow

Speech recognition:

https://venturebeat.com/2017/05/17/googles-speech-recognition-technology-now-has-a-4-9-word-error-rate/

https://techcrunch.com/2017/08/20/microsofts-speech-recognition-system-hits-a-new-accuracy-milestone/

https://hacks.mozilla.org/2017/11/a-journey-to-10-word-error-rate/

Future History

Open source libraries are allowing new products to be developed very quickly.

For instance:
https://towardsdatascience.com/hey-google-where-is-my-pet-tensorflow-object-detection-contribution-9c1d1fdd0443

Google’s TensorFlow
Microsoft’s CNTK
Amazon’s DSSTNE
Theano
Torch
Caffe
Deeplearning4J

Machine Learning Overview

We'll cover several areas of Machine Learning algorithms / techniques along with some sample applications.

- Regression
- Classification
- Clustering

Regression

Regression is a way to predict an outcome based on a set of variables, after training using a set of labeled data.

It is often the ML method that people learn - also may be the one fallback used.

Regression

Regression is used to answer questions like:

“Given current income, savings, # of late payments, # of kids, marital status, etc—will this person default on a loan of $$$ amount?”

“Given square footage, age, school district, zip code, price of nearby homes—what is the value of a house?”

Regression

There are two parts of the question that will determine which type of regression would be the most useful:

1. The expected answers—Is this a yes/no question? If you plotted the actual answers from historical data, do they look like a straight line or some type of curve? The type of regression method is generally dependent on the type of answer expected.

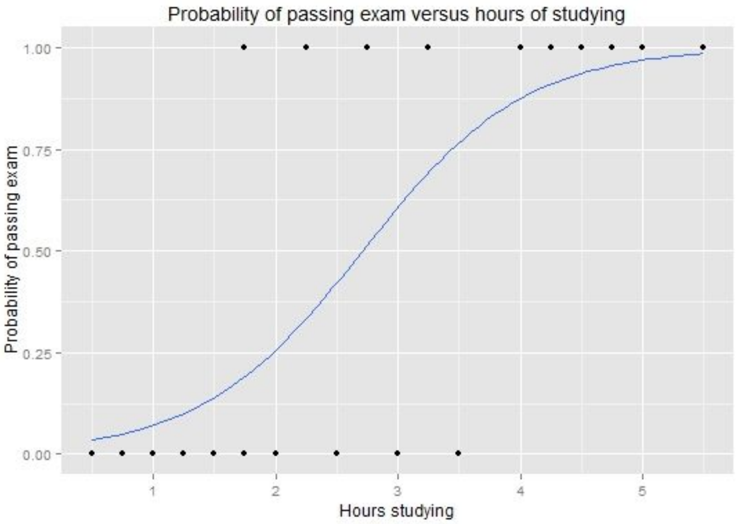
2. The input values—Are they independent of each other (square footage and age)? Are they highly related (zip code and school district)? Based on the dependence between input values, special cases of regression methods can take this into account.

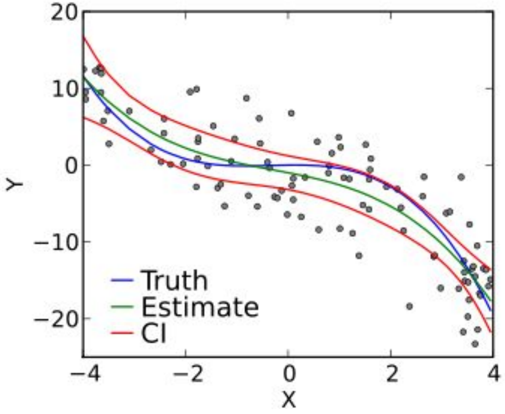
Regression

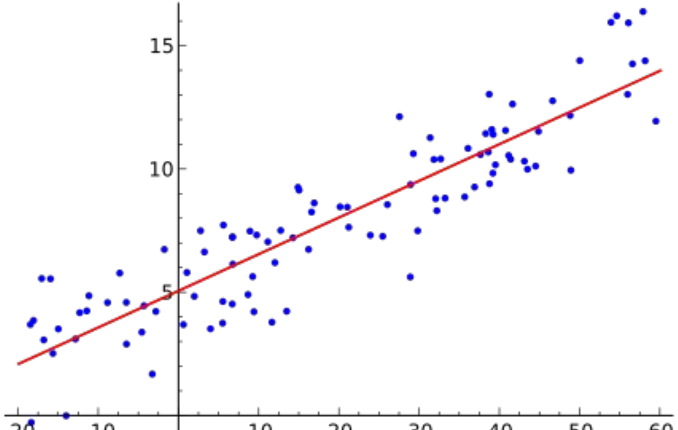
Logistic Regression is used for yes/no answers.

Linear Regression is used when answers follow a straight line if you put them on a graph.

Polynomial Regression is used when answers follow a curve if you put them on a graph.







Classification

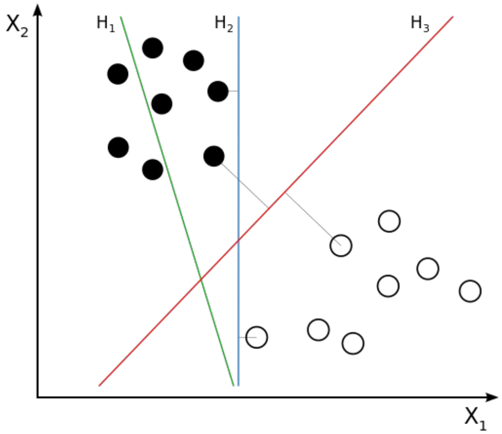
Classification is used to take a piece of data and determine which classification it should belong to, from a set of known classifications.

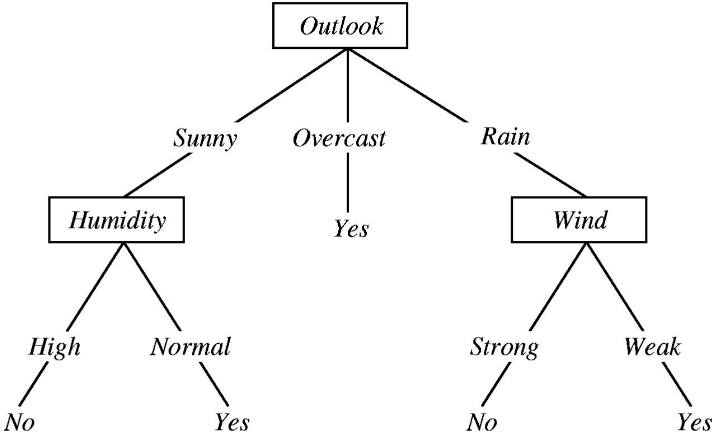
Typically, there are two types of classification problems:

- Binary - there are only two classes - like “Spam or Not Spam”. This is the same as a yes or no answer. This can usually be solved through Logistic Regression.
- Multiple Classes - potentially a large number of classes to choose from. This is used to answer questions like “What part of speech is the word ‘blue’ in the sentence ‘Roses are red, violets are blue’ ?”

Classification

- **Linear Regression** - if the classifications can be grouped separately along some arbitrary line, then linear regression may be the best and most efficient way to solve the problem.
- **Support Vector Machine** - attempts to find a line between two groups in training data.
- **Decision Tree** - used to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Thank you Wikipedia!
- **Naive Bayes** - looks a lot like probability. Actually, it is probability - make sure variables are independent when using it though.





Clustering

Clustering can be thought of as the opposite of classification. The classifications (or groupings) are not known ahead of time, and are discovered by using machine learning.

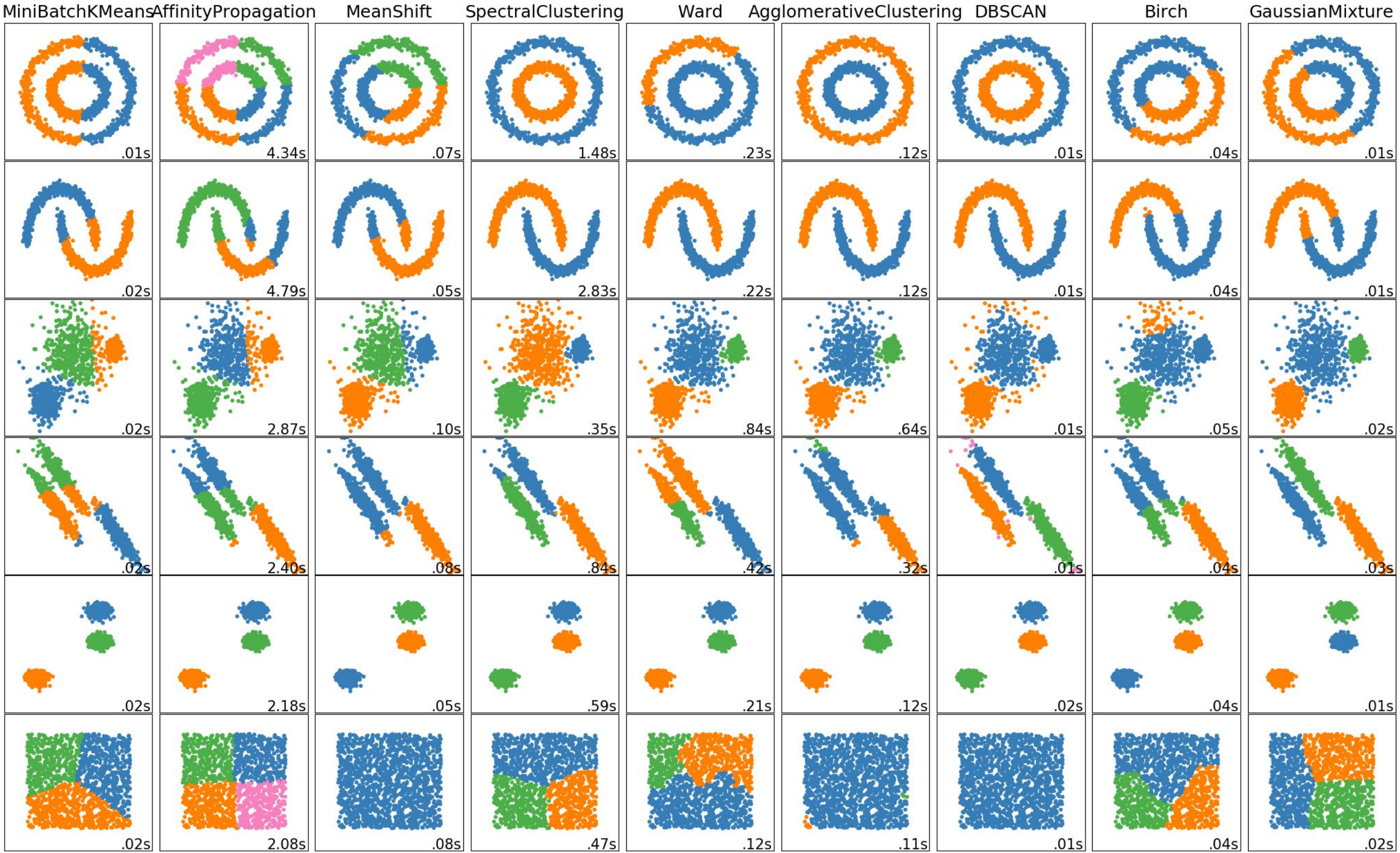
Some algorithms used here need to be given the number of clusters to break the data into, while others take other parameters and identify some set of clusters based on them.

Clustering

Here are a few of the many algorithms used for clustering:

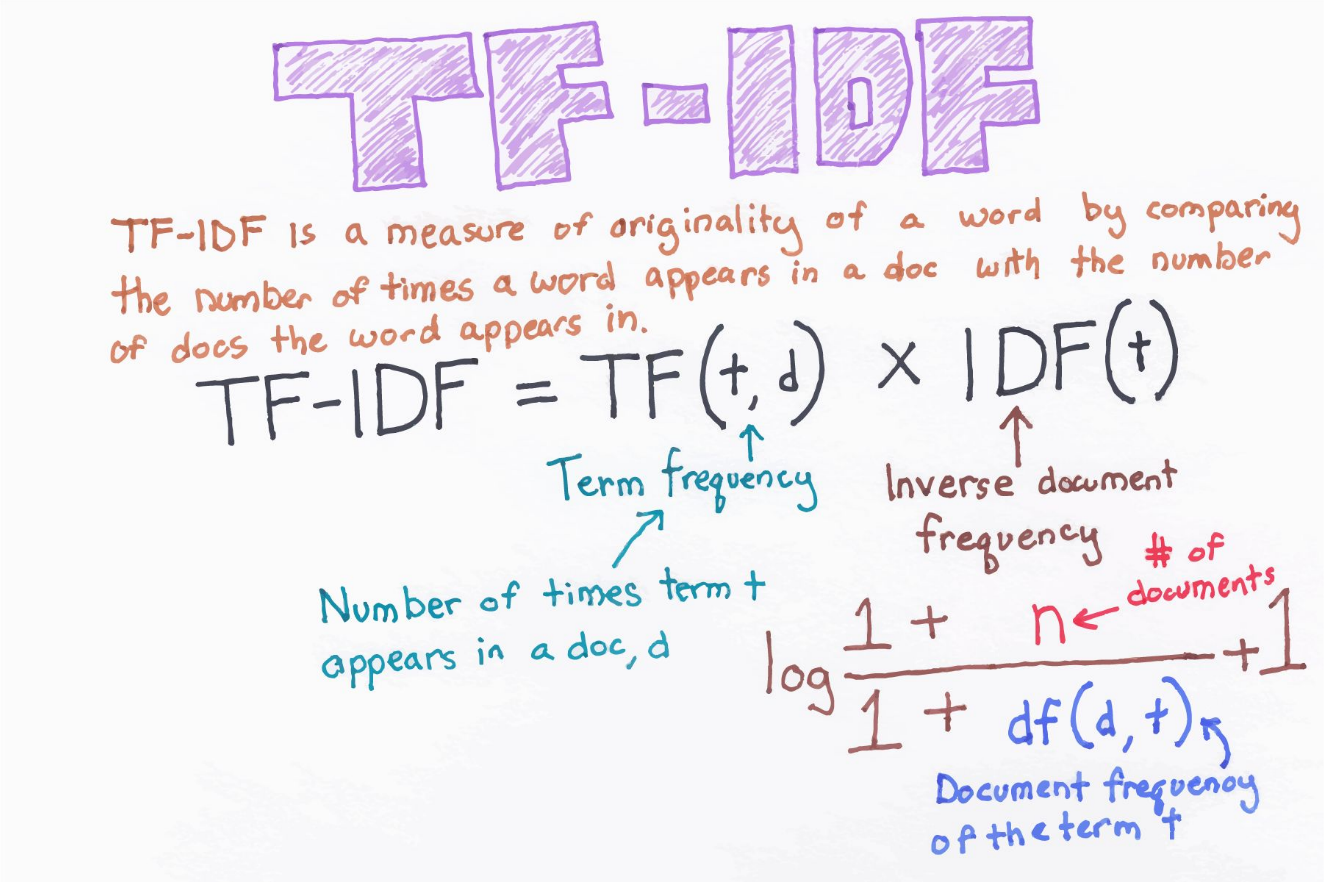
- **K-Means** - takes a number of clusters as an input, and computes the cluster boundaries based on the distance between points.
- **Affinity Propagation** - also known as “Nearest Neighbor”. Takes a set of parameters and computes the cluster boundaries based on the nearest neighbor graph distance.

There are a lot of other algorithms as well, but all seem to use either the distance between points or nearest neighbor graph distance.



From Scikit-learn Website

Text Analysis Approaches



TF-IDF

TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.

$$\text{TF-IDF} = \text{TF}(t, d) \times \text{IDF}(t)$$

$$\log \frac{1 + n}{1 + df(d, t)} + 1$$

Term frequency
Number of times term t appears in a doc, d
Inverse document frequency
# of documents
Document frequency of the term t

Text Analysis Problems

- Detecting Sentence Boundaries
- Text Search
- Stemming
- Document Clustering
- Grammar Check
- Summarization
- Sentiment Analysis
- Named Entity Recognition
- Language Translation
- Text Completion

Natural Language Processing

Note - This is NOT Neuro-Linguistic Programing (unless you're into psychotherapy)

Natural-language processing (**NLP**) is an area of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to fruitfully process large amounts of natural language data.

In linguistics, a **corpus** (plural *corpora*) or **text corpus** is a large and structured set of texts (nowadays usually electronically stored and processed). They are used to do statistical analysis and hypothesis testing, checking occurrences or validating linguistic rules within a specific language territory.

MASC - http://www.anc.org/data/masc/ - Over 500K words of written and spoken data, including 25K words from each of 19 genres, all or parts of the data annotated for 17 different annotation types.

The New York Times Annotated Corpus - https://catalog.ldc.upenn.edu/ldc2008t19

Penn Treebank Tags

Part-of-speech tags are assigned to a single word according to its role in the sentence. Traditional grammar classifies words based on eight parts of speech: the verb (**VB**), the noun (**NN**), the pronoun (**PR**+**DT**), the adjective (**JJ**), the adverb (**RB**), the preposition (**IN**), the conjunction (**CC**), and the interjection (**UH**).

Chunk tags are assigned to groups of words that belong together (i.e. phrases). The most common phrases are the noun phrase (**NP**, for example *the black cat*) and the verb phrase (**VP**, for example *is purring*).

Relations tags describe the relation between different chunks, and clarify the role of a chunk in that relation. The most common roles in a sentence are **SBJ** (subject noun phrase) and **OBJ** (object noun phrase). They link **NP** to **VP** chunks. The subject of a sentence is the person, thing, place or idea that is *doing* or *being*something. The object of a sentence is the person/thing affected by the action.

https://www.clips.uantwerpen.be/pages/mbsp-tags

NLP Libraries

- NLTK - Natural Language Toolkit - starting point for most people interested in NLP. Easy to use Python library with a lot of examples (Google NLTK Book).
- Stanford CoreNLP - State of the art NLP library from your friends at Stanford. Check the license before use in a commercial product.
- OpenNLP - NLP library similar to CoreNLP, but under the Apache License.
- spaCy - Python based industrial scale NLP library. Easy to set up and use.
- Spark NLP (John Snow Labs) - Cluster compute for NLP tasks.

Text Comparison

Naive Bayes Approach -

This was my first introduction to text comparison in Grad school at F.I.T. The project was intended to allow a user to enter a question, and then determine the best newsgroup to post the question to get a response.

$P(q \mid ng) = P(ng \mid q)P(q) / P(ng)$

Probability computed simply based on how many times the words in the question appeared. Did not take context or order of the words into consideration.

Worked much better than I expected.

Text Comparison

Text Frequency Inverse Document Frequency - TF/IDF -

Numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.[1] It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling.

Normally used as an N*M matrix where

N = number of words in vocabulary

M = number of documents in corpus

Demo - Jupyter Notebook for Bugzilla Analysis

Text Analysis

Word2Vec - based on a paper from Google - https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

Great explanation of how it is built:
http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

List of Pre-trained models:
http://ahogrammer.com/2017/01/20/the-list-of-pretrained-word-embeddings/

Text Analysis

Word Mover's Distance - WMD

Based on a paper from Washington University in St. Louis - http://proceedings.mlr.press/v37/kusnerb15.pdf

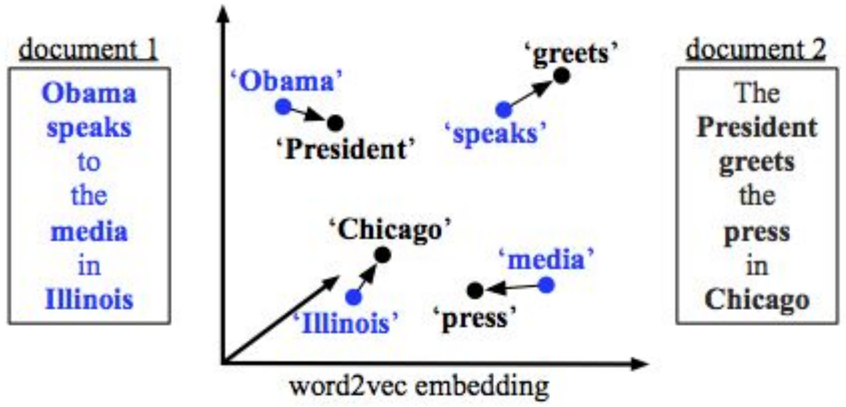
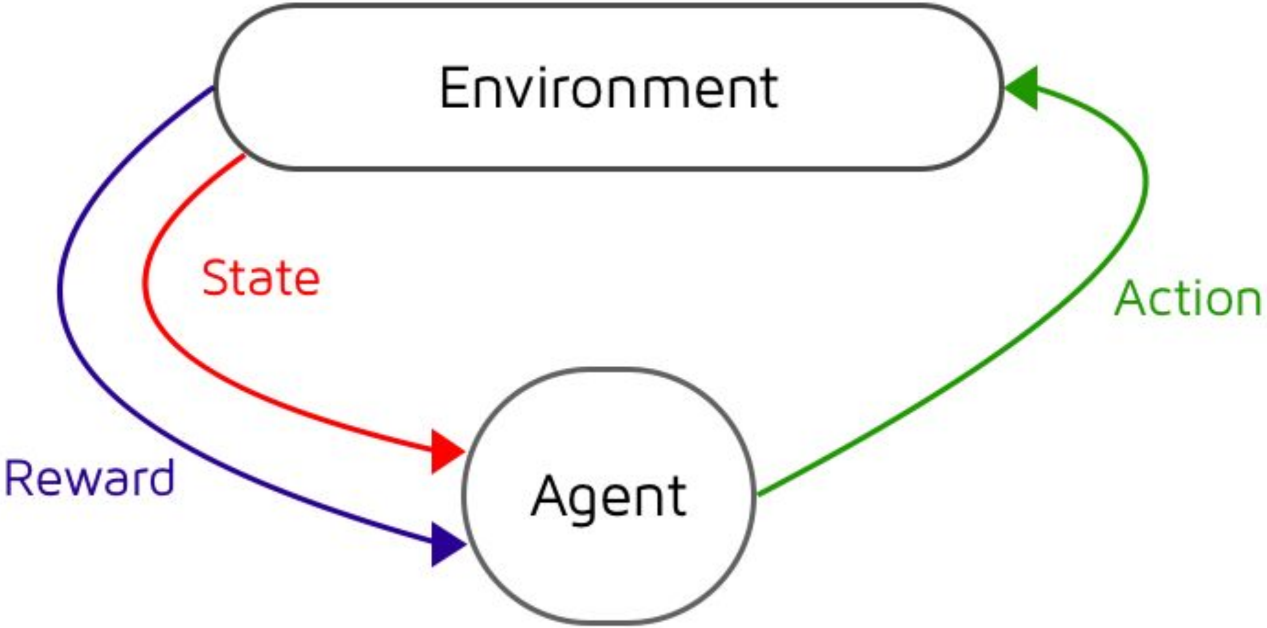


Figure 1. An illustration of the *word mover's distance.* All non-stop words (***bold***) of both documents are embedded into a *word2vec* space. The distance between the two documents is the minimum cumulative distance that all words in document 1 need to travel to exactly match document 2. (Best viewed in color.)

Deep Learning Deep Dive - Reinforcement Learning



What is it?

Wikipedia:

Reinforcement learning (RL) is an area of machine **learning** inspired by behaviourist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.

Russell & Norvig:

The task of reinforcement learning is to use observed rewards to learn an optimal (or nearly optimal) policy for the environment.

Beginnings

Alan Turing - Computing Machinery and Intelligence (1950) - I didn't even have to change this line from our last presentation.

The basic concept of "learning by trial and error" goes back to the "Law of Effect" from Edward Thorndike (search & memory)

The term "Reinforcement" and "Reinforcement Learning" were used in 1965 in a paper by Waltz & Fu

"Optimal Control" - Richard Bellman - mid 1950's

- Dynamic Programming
- Markovian Decision Processes

Intelligent Agents

Markov Decision Processes

1. **A finite set of states.**
2. **A set of actions available in each state.**
3. **Transitions between states.**
4. **Rewards associated with each transition.**
5. **A discount factor γ between 0 and 1.**
6. **Memorylessness.** Once the current state is known, the history can be erased because the current Markov state contains all useful information from the history. In other words, “the future is independent of the past given the present”.

Q Learning

Q Learning - evaluates which action to take based on an **action-value**

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

As the program chooses an action to take when changing state, the Q value of that state is updated based on the reward received, the learning rate, the discount factor, and the old value.

Discount factor = the difference between an immediate reward, and future rewards.

Temporal Difference (TD)

Temporal Difference (TD) Learning methods can be used to estimate these value functions. If the value functions were to be calculated without estimation, the agent would need to wait until the final reward was received before any state-action pair values can be updated. Once the final reward was received, the path taken to reach the final state would need to be traced back and each value updated accordingly.

On the other hand, with TD methods, an estimate of the final reward is calculated at each state and the state-action value updated for every step of the way.

The TD method is called a "bootstrapping" method, because the value is updated partly using an existing estimate and not a final reward.

Policy Learning

Another option - Policy Learning - learns a policy function, which is a map from each state to the best corresponding action at that state.

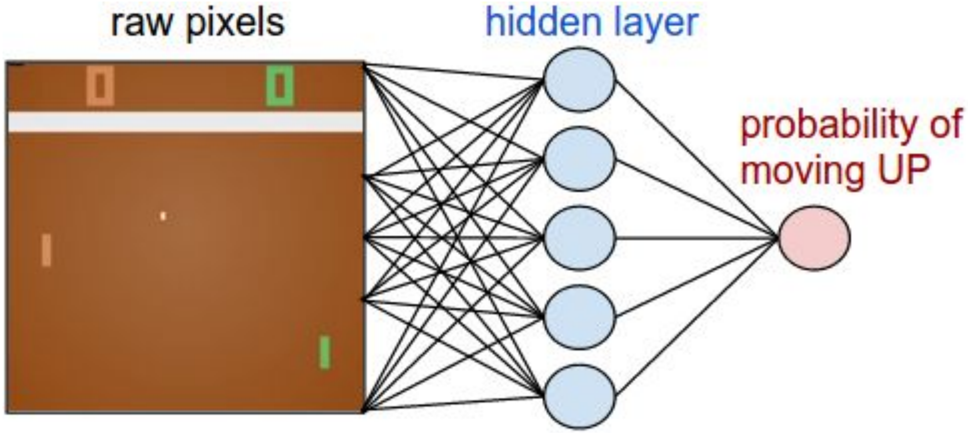
Most of the material I have been consuming lately references learning a “policy” with respect to Reinforcement Learning.

Any thoughts on a mechanism to learn a complex function?

Deep Reinforcement Learning

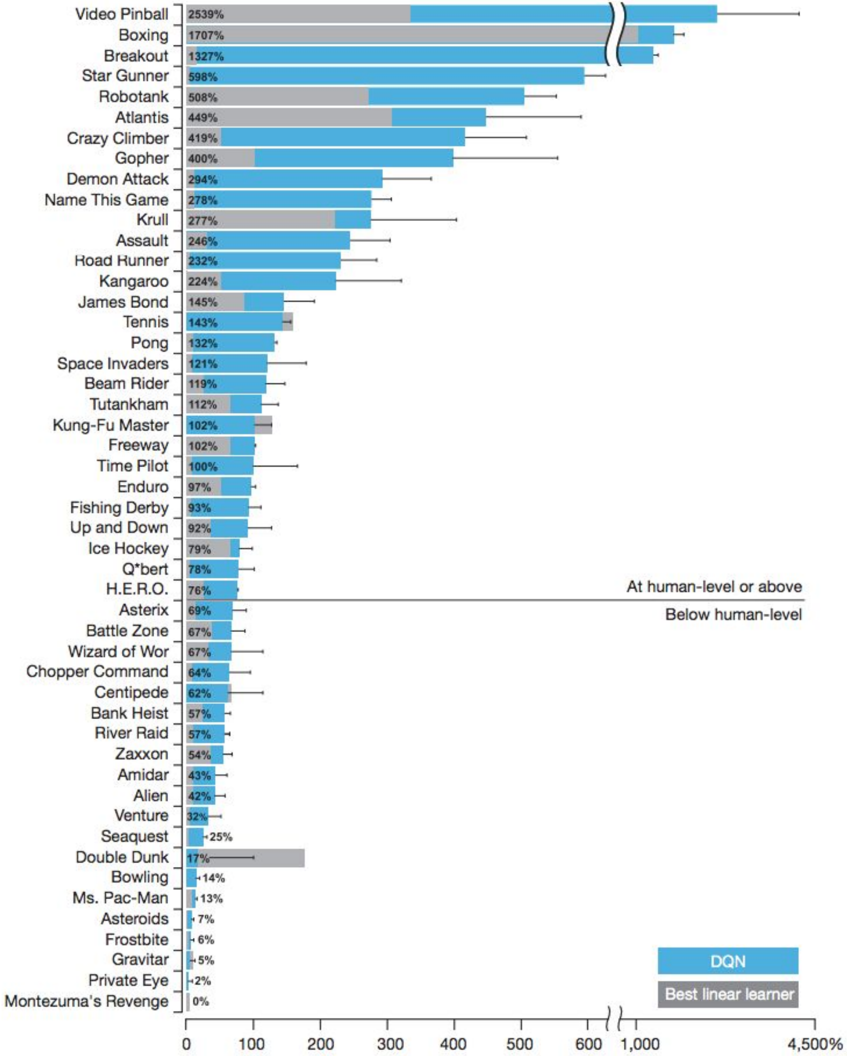
Deep Neural Networks began entering the conversation of Reinforcement Learning some time around 2014/2015.

For example - here’s an architecture for Andrej Karpathy’s Pong from Pixels implementation:



Deep Q Networks (DQN)

In 2015, DeepMind used a method called **deep Q-networks (DQN)**, an approach that approximates Q-functions using deep neural networks, to beat human benchmarks across many Atari games


Video Pinball 2539%
Boxing 1707%
Breakout 1327%
Star Gunner 598%
Robotank 508%
Atlantis 449%
Crazy Climber 419%
Gopher 400%
Demon Attack 294%
Name This Game 278%
Krull 277%
Assault 246%
Road Runner 232%
Kangaroo 224%
James Bond 145%
Tennis 143%
Pong 132%
Space Invaders 121%
Beam Rider 119%
Tutankham 112%
Kung-Fu Master 102%
Freeway 102%
Time Pilot 100%
Enduro 97%
Fishing Derby 93%
Up and Down 92%
Ice Hockey 79%
Q*bert 78%
H.E.R.O. 76%
Asterix 69%
Battle Zone 67%
Wizard of Wor 67%
Chopper Command 64%
Centipede 62%
Bank Heist 57%
River Raid 57%
Zaxxon 54%
Amidar 43%
Alien 42%
Venture 32%
Seaquest 25%
Double Dunk 17%
Bowling 14%
Ms. Pac-Man 13%
Asteroids 7%
Frostbite 6%
Gravitar 5%
Private Eye 2%
Montezuma's Revenge 0%
At human-level or above
Below human-level
DQN
Best linear learner
0
100
200
300
400
500
600
1,000
4,500%

Other Cool Stuff!

https://twitter.com/eron_gj/status/967672260147470336

Other Cool Stuff!