MACHINE LEARNING & DEEP LEARNING

Carlos A Castro, PhD
Data Scientist
Intel Costa Rica

About: Carlos A Castro

- Education:
  - PhD in Computer Science – DePaul University, Chicago IL
  - MS in Software Engineering – DePaul University, Chicago IL
  - MBA in Banking and Finance – FUNDEPOS, Costa Rica
  - BS in Computer Science – Universidad de Costa Rica, Costa Rica

- Work experience:
  - Intel: Data Scientist
  - Google: Software Engineer
  - Central Bank of Costa Rica: Software Engineer
  - DePaul and University of Costa Rica: researcher and instructor

Agenda

- Overview of Machine learning
- Foundations of Machine learning
- Neural networks
- Deep learning
- Tools
- Parting thoughts
- Optional: Walk through the math

OVERVIEW OF MACHINE LEARNING

Think of the possibilities

Is this a tree?









How would you program an algorithm to identify trees?
What rules would you need?

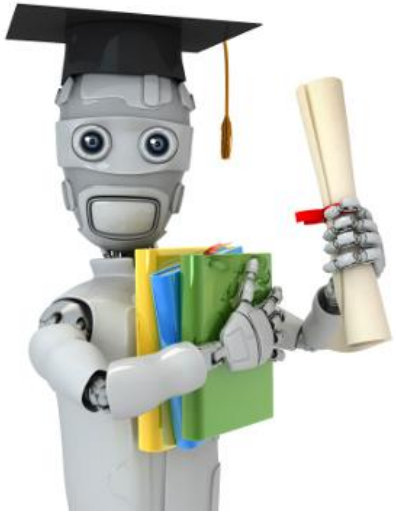
Machine Learning

- "Field of study that gives computers the ability to learn without being explicitly programmed" – Arthur Samuel, 1959

- Computational methods using *experience* to improve and make accurate predictions [1]
  - *Experience* refers to past information available

- Closely related to:
  - Probability, Statistics, Linear Algebra, and Optimization

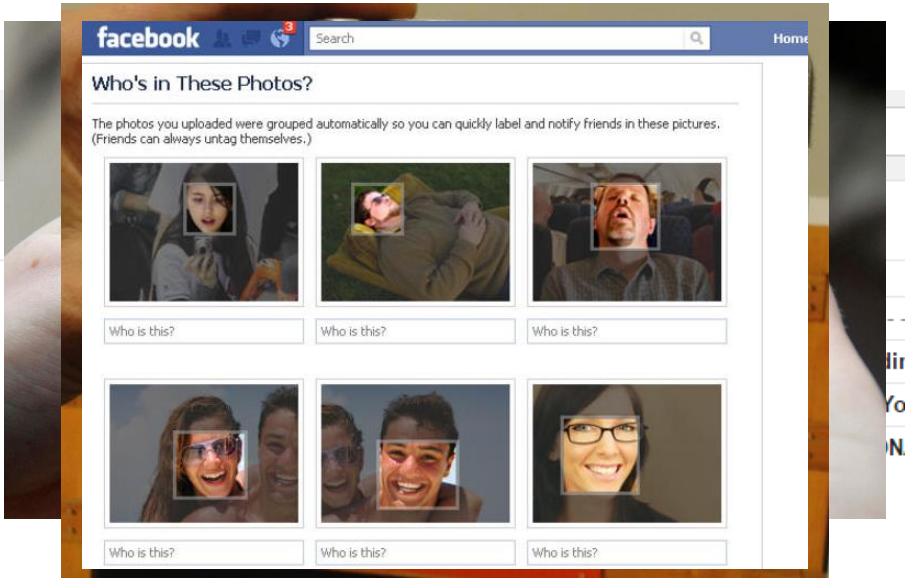
- It has been around for a while, but is getting a lot of attention lately



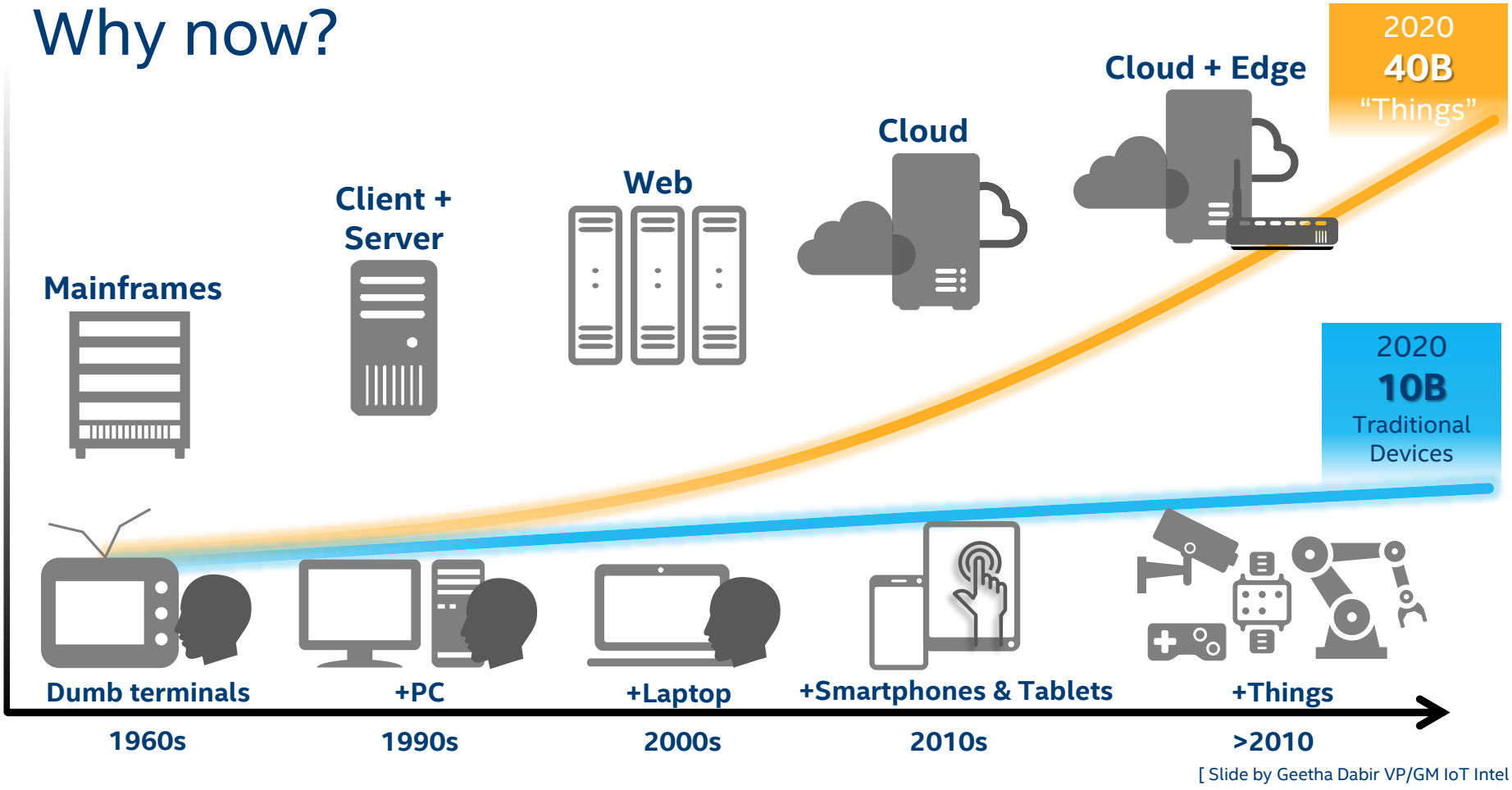
Applications

We will see a couple of examples in a bit…

- Text or document classification
- Speech recognition
- Optical character recognition
- Computer vision
- Natural language processing
- Fraud detection
- Games
- Medical diagnosis



Why now?



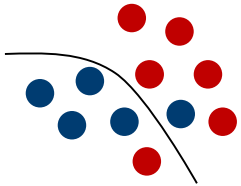
[Slide by Geetha Dabir VP/GM IoT Intel]

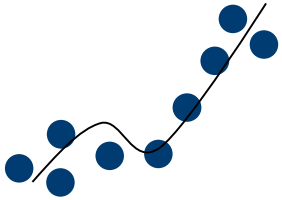
FOUNDATIONS OF MACHINE LEARNING

A bit of theory

Different Goals / Classes of problems

- Classification:
  - Assign a category to each item.
  - Binary & multiclass
- Regression:
  - Predict a real value for each item
- Ranking:
  - Order items according to some criteria



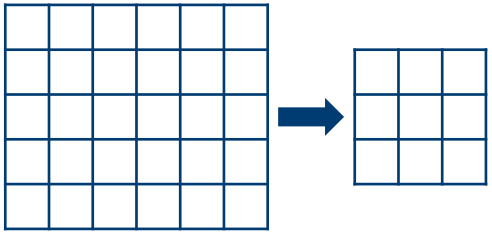


[*Foundations of Machine Learning*, by Mohri, Rostamizadeh andTalwalkar]

Different Goals / Classes of problems

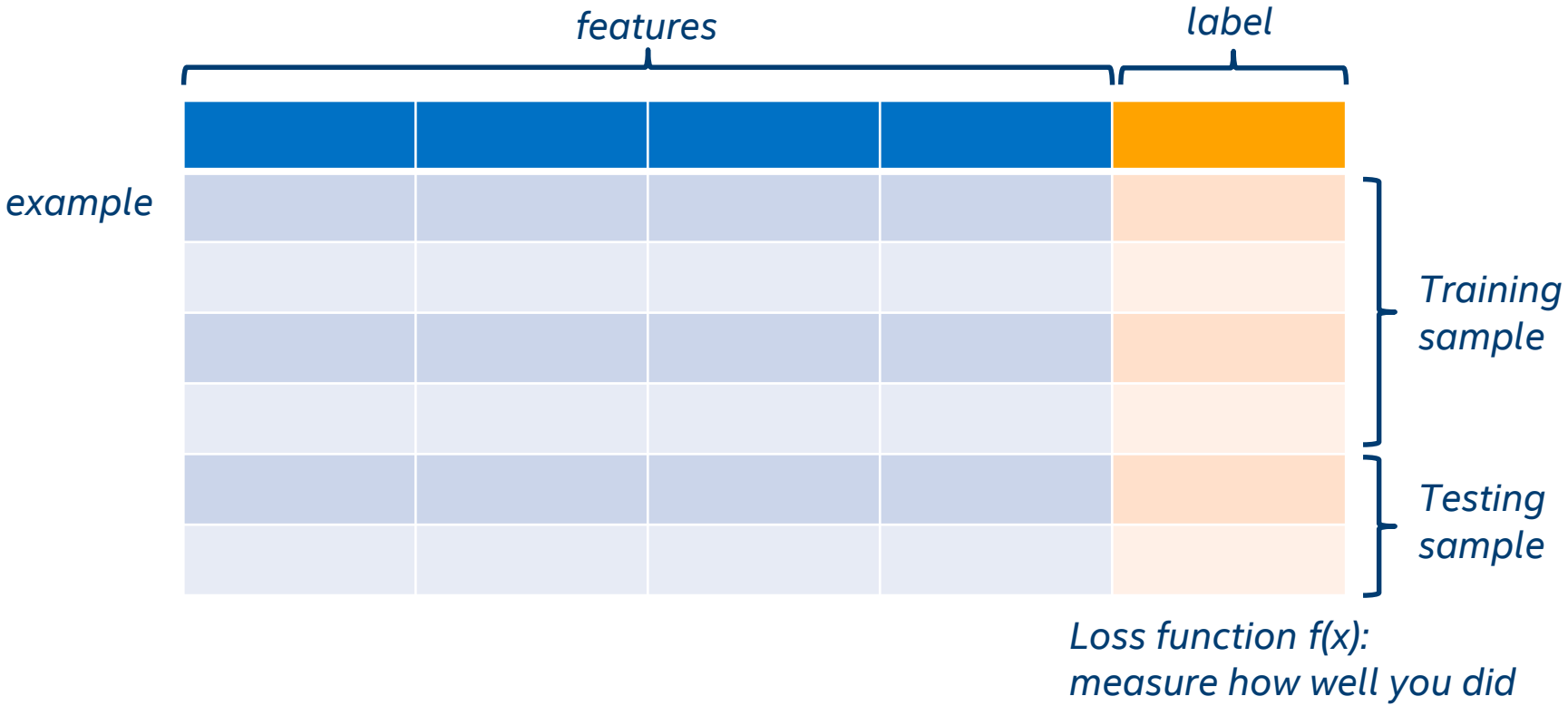
- Clustering:
  - Partition items into homogeneous regions

- Dimensionality Reduction:
  - Transform an initial representation into a lower dimensional representation



[*Foundations of Machine Learning*, by Mohri, Rostamizadeh andTalwalkar]

Terminology



Common terms – Ex. Spam detection

example:
a single email

features *label*

to	from	subject	text	...	spam?
alice@foo.com	ben@getrich.com	Get rich in 24 hours	Dear Ms, I am the lawyer of a rich deceased...		Yes
alice@foo.com	charlie@foo.com	Here is the presentation	Alice, attached you will find the presentation for Monday...		No

Loss function f(x):
1: if correctly classified,
0: otherwise

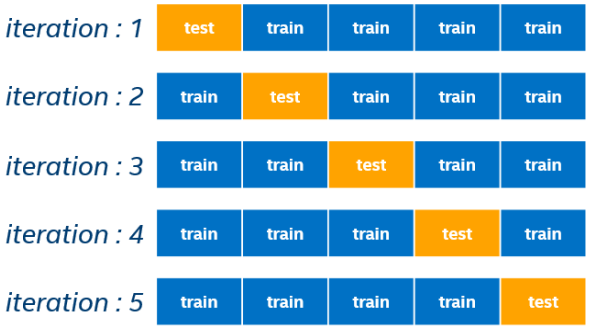
Different Learning Approaches (most common)

- Supervised Learning
  - Learner gets a set of labeled examples for training, and makes predictions for all unseen points
- Un-supervised Learning
  - Learner gets unlabeled data and makes predictions for all unseen points

Evaluation

- Hold-out:
  - Divide your examples into training and testing
  - Evaluate the performance on the testing subset
    - Increases fairness, reduces over-fitting

- Cross validation
  - Divide your examples into n folds or subsets
  - At each iteration you use n-1 folds for training, and the remaining fold for testing
  - Considered a more robust approach
  - Good when the amount of labeled data is small



Evaluation Metrics

- Depending on your *goal*, your *loss function*, and the *learning approach*, you can use different evaluation metrics:

- Example:
  - If your *goal* is: Regression
  - your *loss function*: the difference between the predicted and the real label
  - and your *learning approach* is: supervised

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- You can use metrics such as:
  - RMSE: Root Mean Square Error → an average of the error
  - R^2: Coefficient of determination → how well the data fits the model

Evaluation Metrics

- Example:
  - If your *goal* is: Classification
  - your *loss function*: 1 if correctly classified, 0 otherwise
  - and your *learning approach* is: supervised

- You can use the confusion matrix metrics such as:
  - Sensitivity
  - Specificity
  - Precision
  - Accuracy

		Predicted Class	
		Yes	No
Actual Class	Yes	True Positive	False Negative (Type II Error)
	No	False Positive (Type I Error)	True Negative

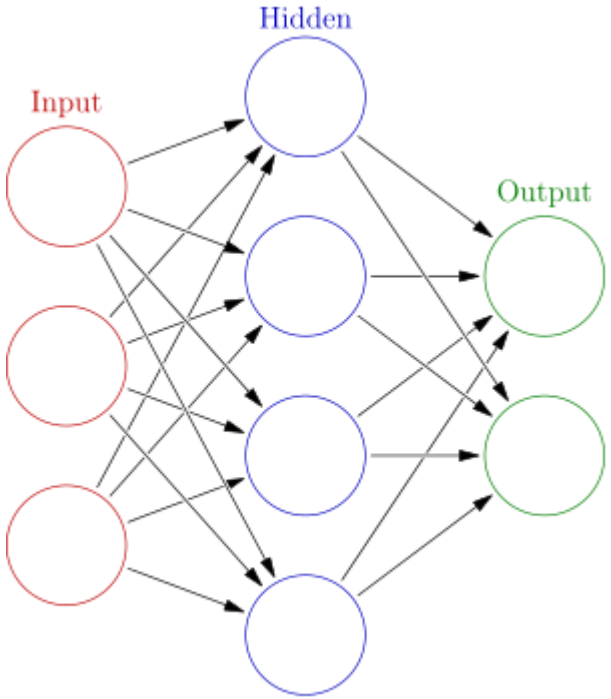
Many different techniques

- Linear regression
- Logistic regression
- Decision trees
- Association rule mining
- Support vector machines
- K-nearest neighbors
- Naive Bayes
- K-means clustering
- Hierarchical clustering
- Linear discriminant analysis
- Matrix factorization
- **Neural networks**
- **Deep learning**
- …

NEURAL NETWORKS

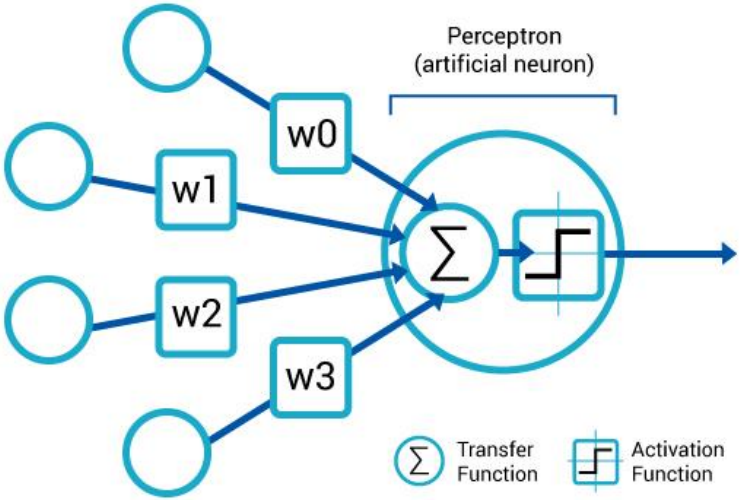
Neural Networks

- *Inspired* by the human neural networks
  - Nodes/Vertices → Neurons: processing units
  - Edges → Connections between neurons
- The connection strengths between the edges are adaptive
  - These are tuned by a learning algorithm
- The Nodes have a function that will fire when the input is above a certain level



[https://en.wikipedia.org/wiki/Artificial_neural_network]

Perceptron - Artificial Neuron

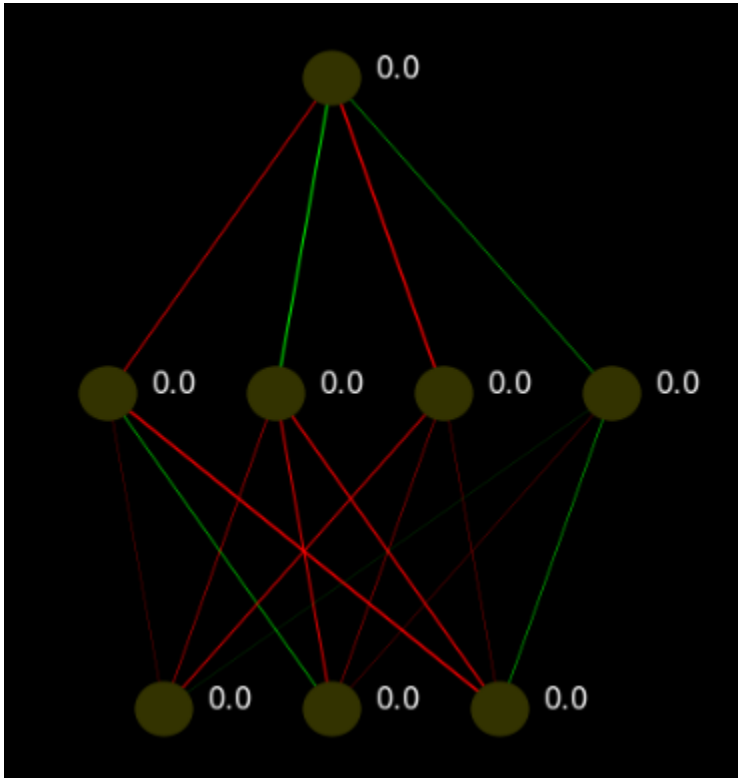


[http://www.amax.com/blog/?p=804]

Example

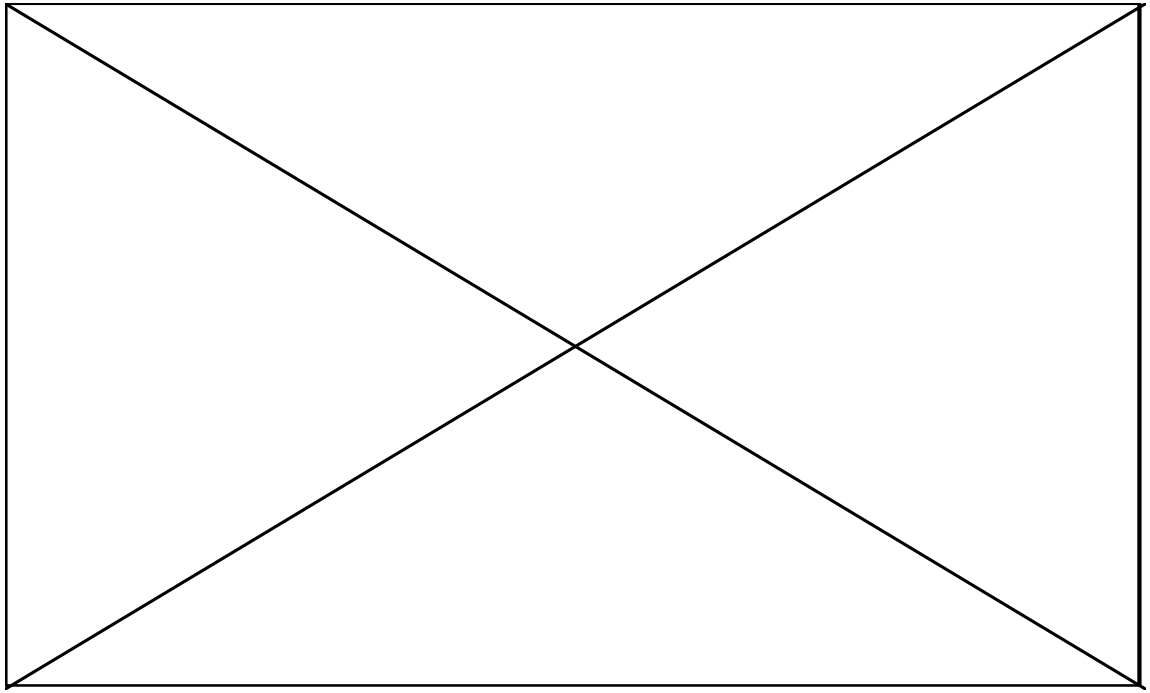
	Input			Output
Example 1	0	0	1	0
Example 2	0	1	1	1
Example 3	1	0	1	1
Example 4	0	1	0	1
Example 5	1	0	0	1
Example 6	1	1	1	0
Example 7	0	0	0	0

New situation	1	1	0	**?**



[https://medium.com/deep-learning-101/how-to-generate-a-video-of-a-neural-network-learning-in-python-62f5c520e85c#.q3w82nhnb]

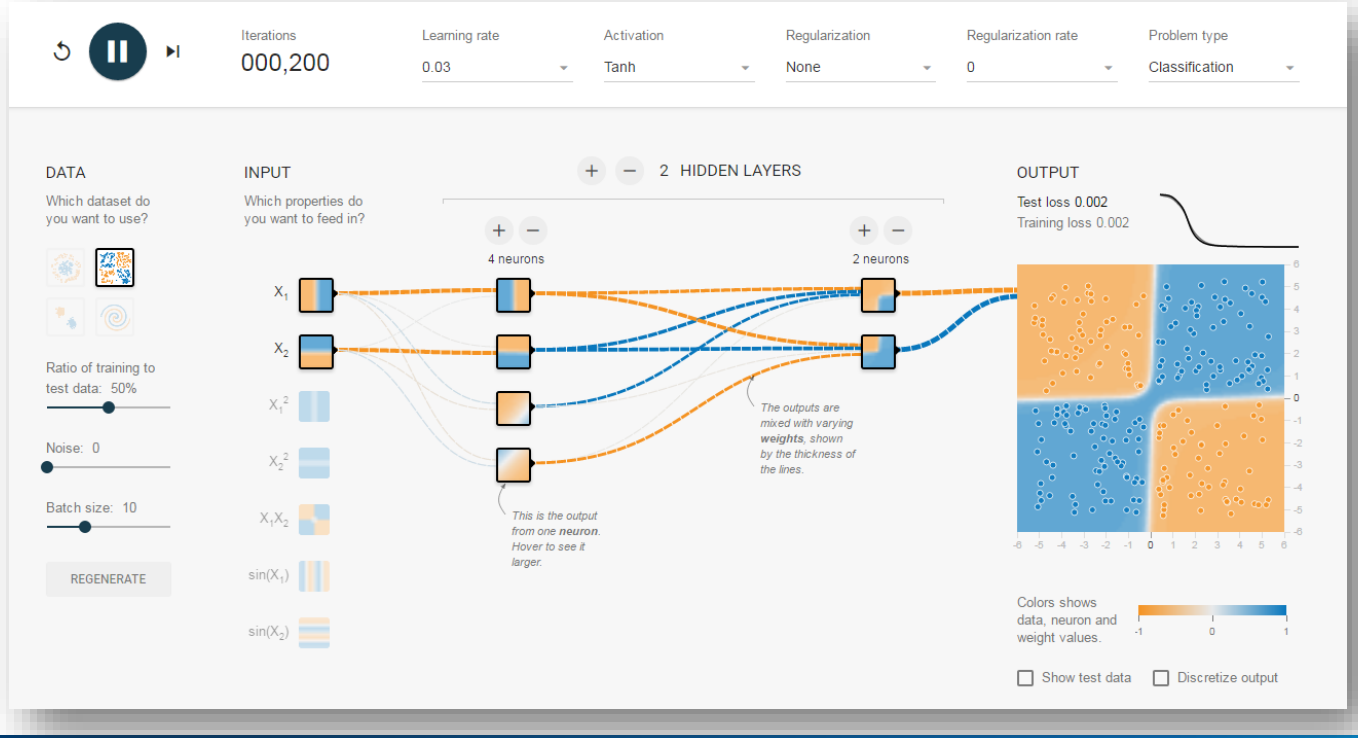
Animation



[https://youtu.be/nrnxZVEHZCo]

Play with a Neural Network

http://playground.tensorflow.org/



DEEP LEARNING

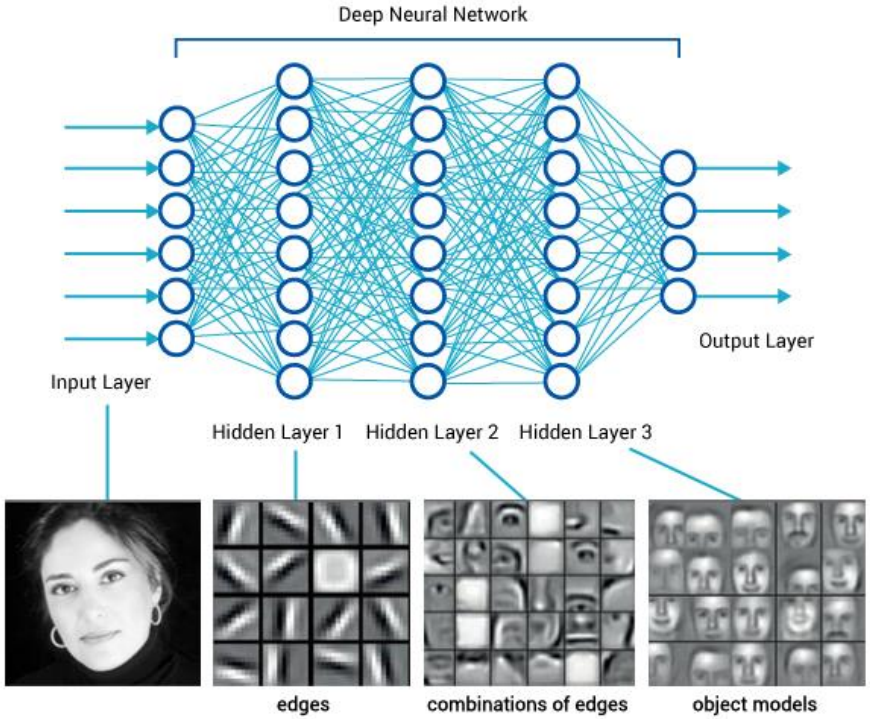
Deep Learning

- Class of machine learning algorithms that:
  - Cascade **multiple layers** of nonlinear processing units, where each layer uses the output of the previous layers as input.
  - The layers form a hierarchy from low-level to high-level features
    - More abstract concepts are learned from lower level ones
  - Typically use neural networks – but this is not a requirement

[https://en.wikipedia.org/wiki/Deep_learning#Artificial_neural_networks]

Deep Neural Network

But with **many many many** more layers!

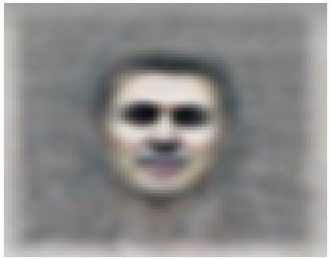


This requires significant processing power!

[http://www.amax.com/blog/?p=804]

Example: Google Face Recognition

- Problem: Build a face classifier from only **un-labeled** data
- Dataset: 10 million 200x200 pixel images taken from YouTube
- Deep neural network: 9 layers, over 1 billion connections
- Hardware: cluster with 1,000 machines (16,000 cores) trained for 3 days



[http://static.googleusercontent.com/media/research.google.com/en//archive/unsupervised_icml2012.pdf]

Example: AlphaGo

- Problem: Play the ancient Chinese game of Go.
- Complexity:
  - The search space is huge 10^{100}.
  - Cannot be solved by brute force
- Three main parts:
  - A Monte-Carlo tree search where it plays out the remainder of the game
  - A deep neural network "*policy network*" to predict the next move
  - A deep neural network "*value network*" estimate winner at each position
- Dataset: 30 million moves for the initial training. Then it was left to play itself - through reinforced learning.
- In March 2016 it beat the world's top player Lee Sedol (4 games to 1)

[http://googleresearch.blogspot.com/2016/01/alphago-mastering-ancient-game-of-go.html]

Example: Google Voice transcription

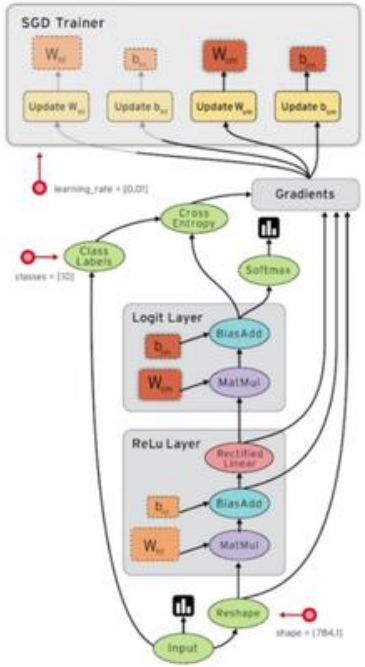
- Problem: Recognize speech and transcribe it to text
- Original models used state of the art Gaussian Mixture models.
- In 2012 they started using Deep Neural Networks with a twist:
  - Long Short-term Memory Recurrent Neural Networks (LSTM RNNs)
  - They have connections that allow them to *remember* the data they've seen
- Dataset: Millions of voice mails donated by the users
  - Without ground truth!
- Trained acoustic, language, and punctuation neural networks.

[http://googleresearch.blogspot.com/2015/08/the-neural-networks-behind-google-voice.html]

A COUPLE OF TOOLS

Google's TensorFlow

- Open source machine learning library
- Uses data flow graphs:
  - Nodes: mathematical operations
  - Edges: multidimensional data arrays
- Allows deployment in CPUs or GPUs:
  - Phone, PC, servers, data centers
- Udacity class:
  - https://www.udacity.com/course/deep-learning--ud730



[https://www.tensorflow.org/]

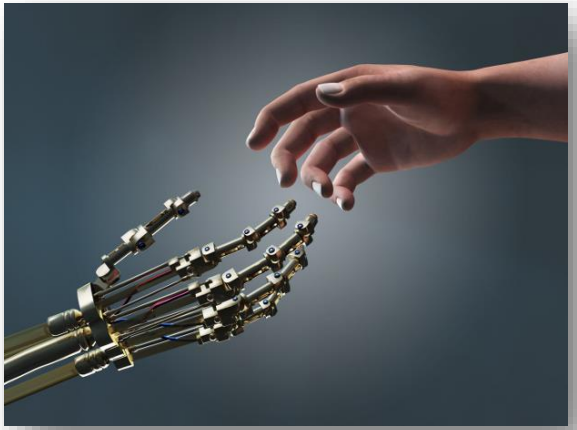
Caffe + NVIDIA cuDNN

- Caffe is a open source deep learning framework
  - Developed at UC Berkeley
  - Models are defined by configuration not code
- Integrated with NVIDIA's cuDNN
  - GPU-accelerated library of primitives for DNN
  - Includes routines such as:
    - Convolution, Softmax, Activaitons (sigmoid, RELUs, TanH), …

[https://devblogs.nvidia.com/parallelforall/accelerate-machine-learning-cudnn-deep-neural-network-library/]
[http://caffe.berkeleyvision.org/]

PARTING THOUGHTS

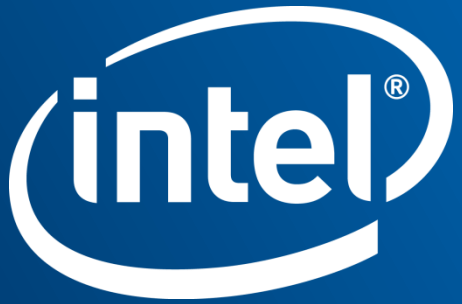
The impact on society



- We are teaching computers to:
  - See, Read, Speak, Drive, …
- This will have a profound effect on society
- Examples:
  - Self driving cars with near perfect driving records
  - Software that can detect cancer from medical images with a higher precision than a radiologist
- What will the impact be on employment? What will the impact be on leisure?
- It will be a different world!

Questions?

Thank you!


intel®

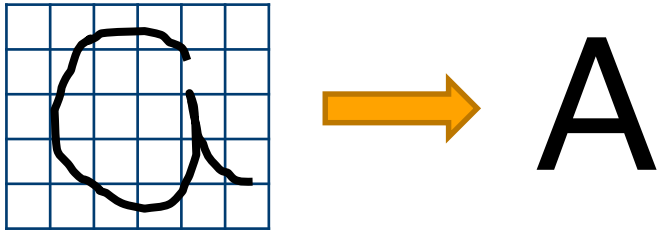
BACKUP

A WALK THROUGH THE MATH BEHIND DEEP LEARNING

Example from Udacity's Deep Learning course:
https://www.udacity.com/course/deep-learning--ud730

Problem: Identify letters

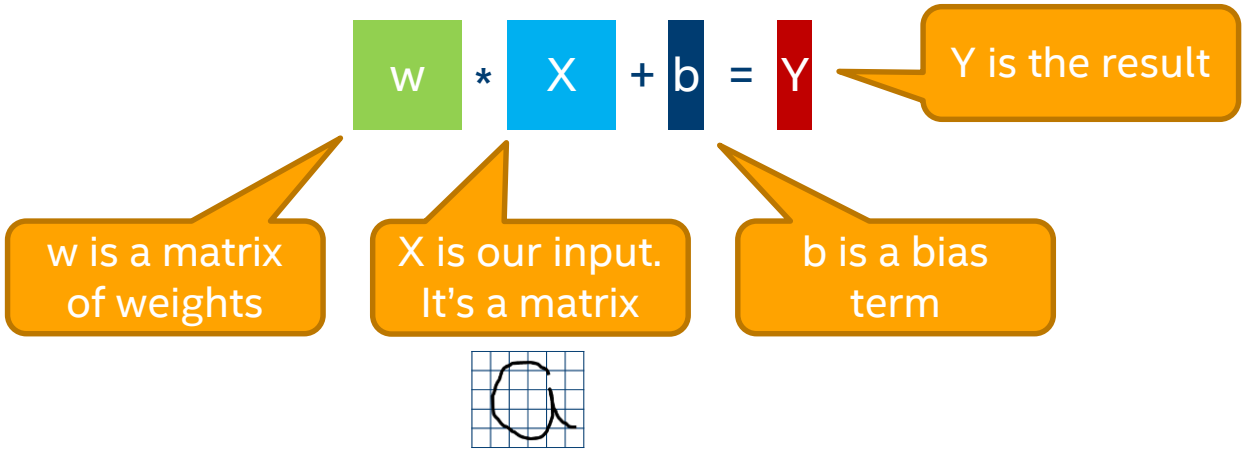
- Classification problem:
  - Input: image (matrix of pixels)
  - Output: one of the 26 letters of the English alphabet



Let's build a linear classifier

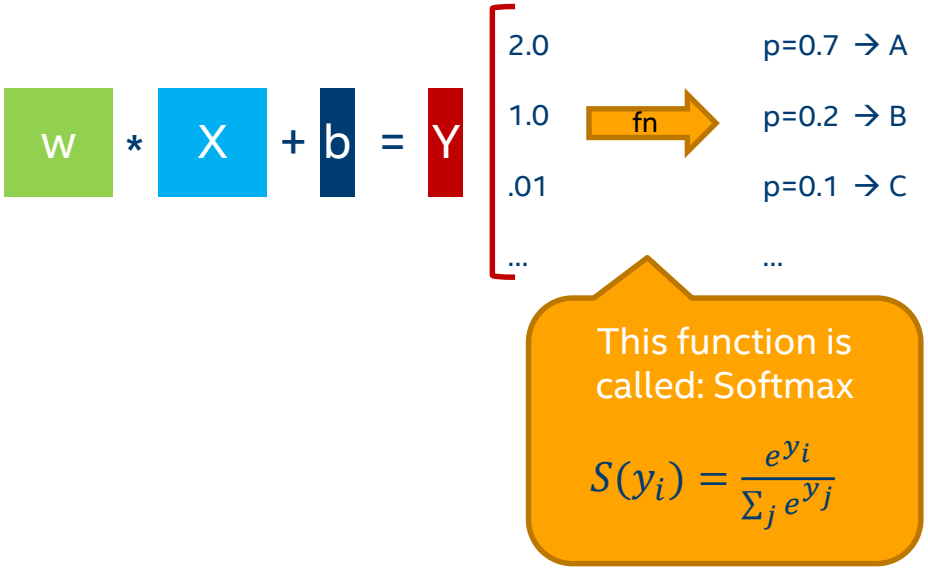
Remember the equation of the line?

$$Y = mX + b$$



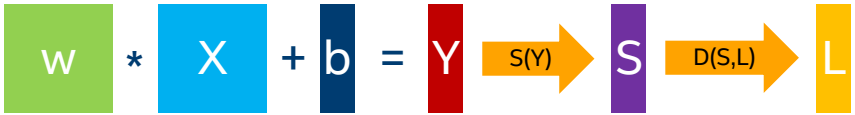
We need it to output probabilities, not scores

*This makes it a **logistic classifier***



How do we know how well it did?

Compare it against the answer vector

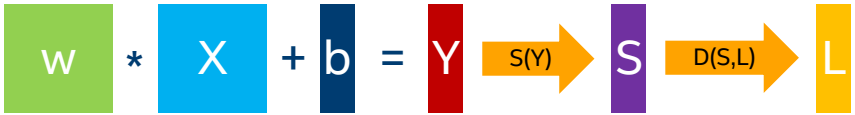


*L is called '**one-hot encoding**' vector:*

A	B	C	...
1.0	0.0	0.0	
0.0	1.0	0.0	
0.0	0.0	1.0	
...	...	...	

How do we calculate this distance?

*One way is to use **Cross-Entropy***



This function is called:
Cross-Entropy

$$D(S, L) = -\sum_i L_i \log(S_i)$$

But how do we determine the values of **w** and **b** ?

- You want w and b so that you minimize the overall error. Ex:
  - D(𝒶, A) should be low
  - D(𝒷, A) should be high

- Measure and average the distance across all of your training set:
  - This is called the **Loss** (Average Cross-Entropy):

$$\mathcal{L} = \frac{1}{N}\sum_{i} D(S(wX_i + b), L_i)$$

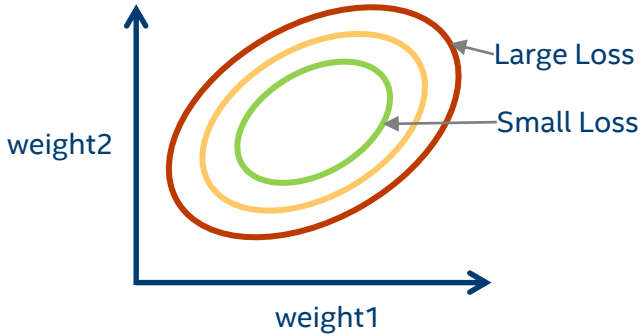
This is VERY expensive!

The i^{th} element of the training set

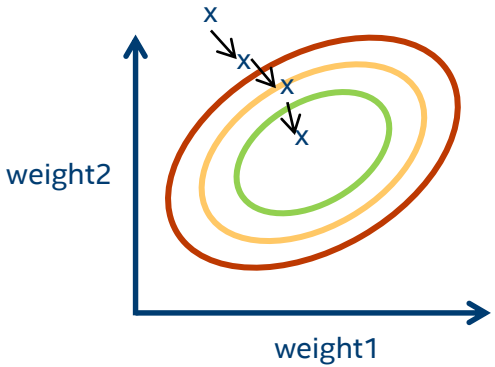
You need to find weights that minimize the Loss

- Numerical optimization problem.
- Technique: **Stochastic Gradient Descent**

- Ex. If you only had 2 weights…



Stochastic Gradient Descent



1. Calculate Loss:
$\mathcal{L}(w_1, w_2)$

2. Calculate derivative:
$\Delta\mathcal{L}(w_1, w_2)$

3. Follow the derivative by taking a step backwards:
$-\alpha\Delta\mathcal{L}(w_1, w_2)$

4. Repeat

Remember this is if for 2 parameters, you will likely have 1000s of parameters in your weights

Great!

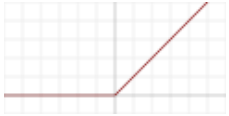
At this point you have built a multi-linear logistic classifier

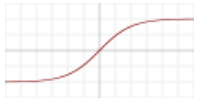
But…

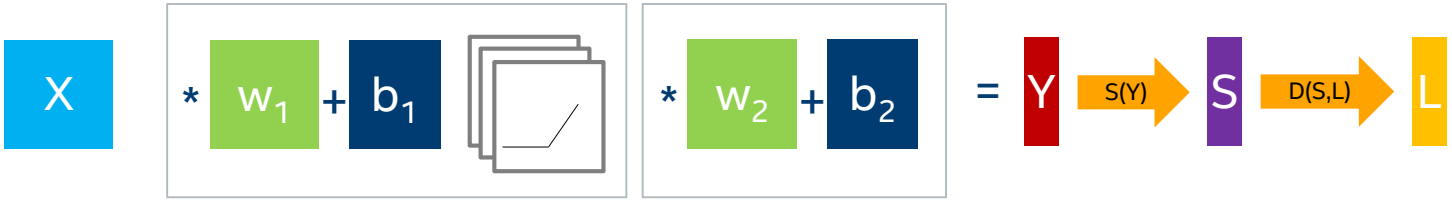
This is linear, it will only work for some types of problems!

Lets add some non-linearity

- We can add an activation function, and chain the process
- There are many activation functions:
  - Rectified Linear Unit (RELU)
  - Sigmoid







We now have a 2-layer Neural Network !

Repeat

- Overall this is complex, but all the parts are pretty simple:
  - Multiply, Add, RELU, …, Softmax
- Computationally the training can be set up efficiently and in a distributed manner

Back propagation