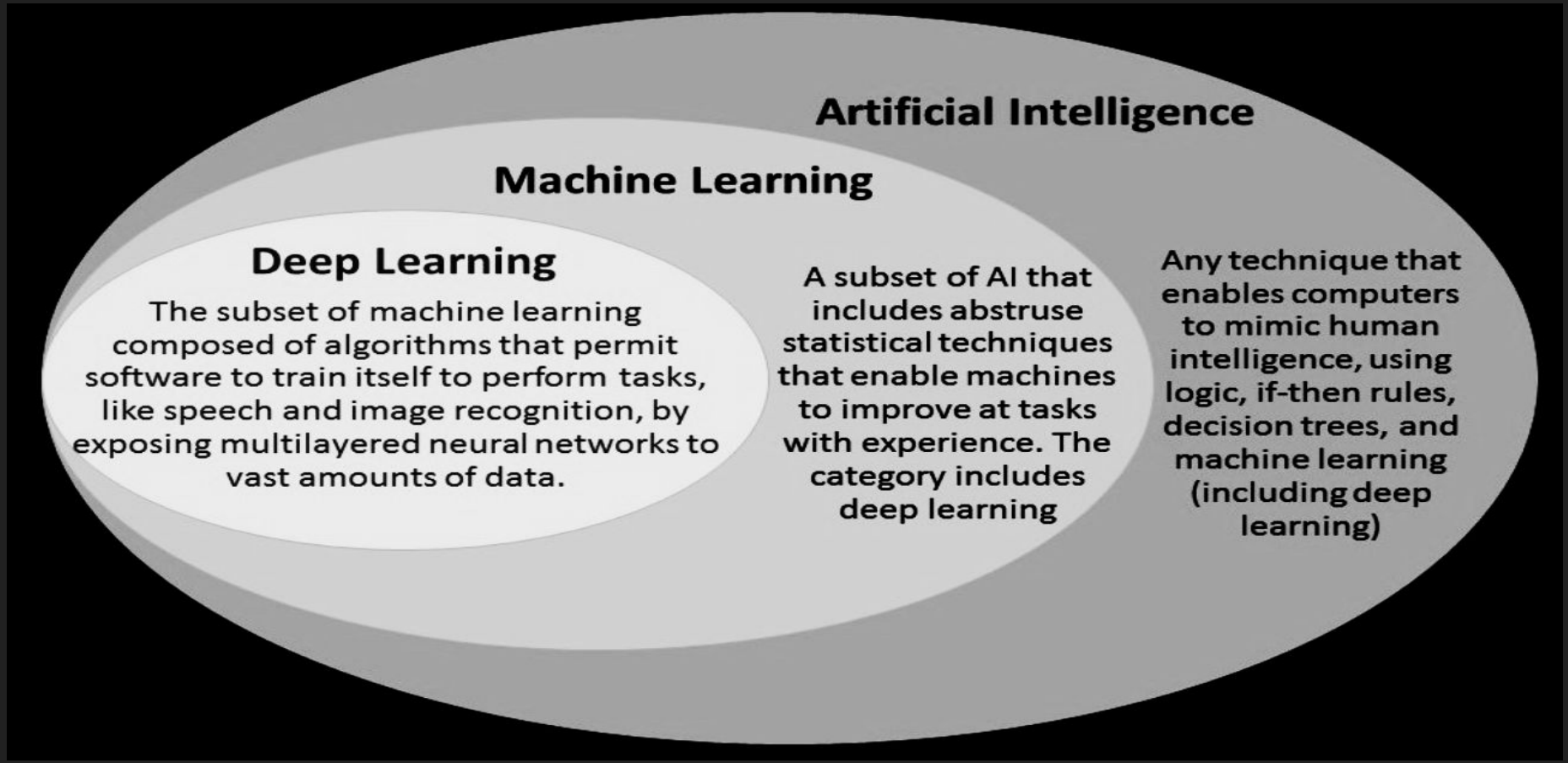
# AI and Deep Learning in Finance

Sanjiv R. Das
Santa Clara University

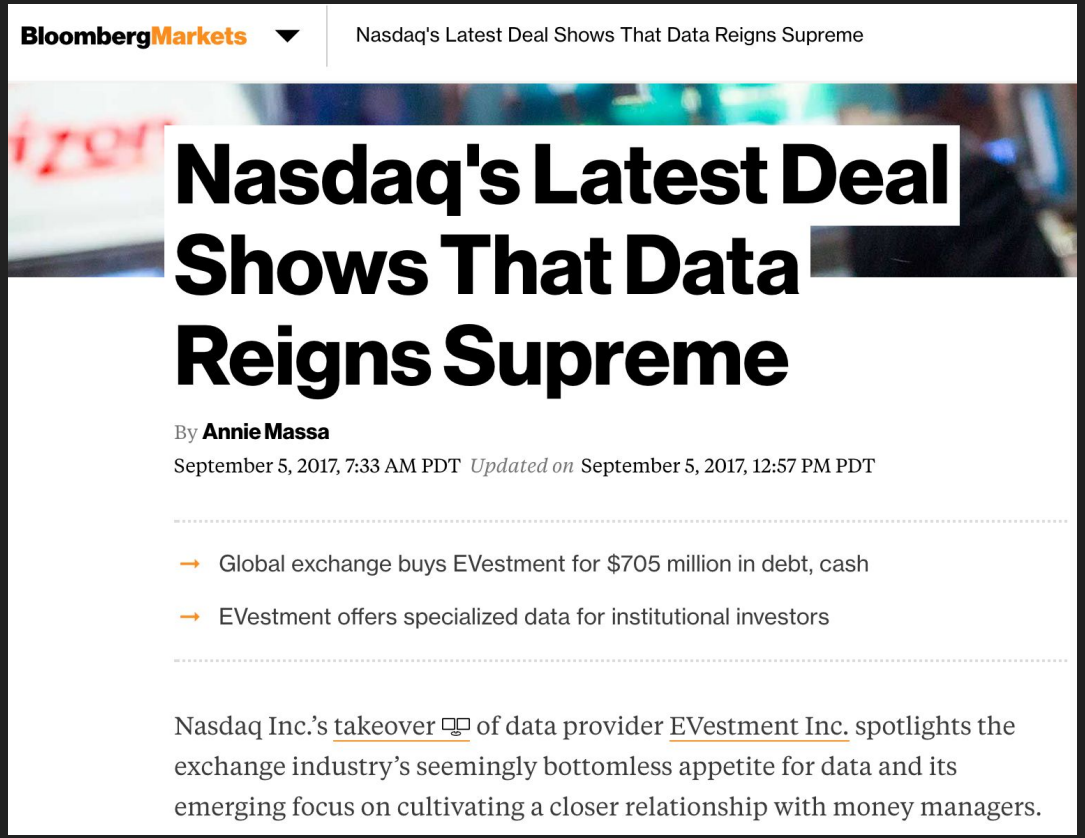
ISB | FinTech | 2018

# AI, ML and DL


Artificial Intelligence
Machine Learning
Deep Learning
The subset of machine learning composed of algorithms that permit software to train itself to perform tasks, like speech and image recognition, by exposing multilayered neural networks to vast amounts of data.
A subset of AI that includes abstruse statistical techniques that enable machines to improve at tasks with experience. The category includes deep learning
Any technique that enables computers to mimic human intelligence, using logic, if-then rules, decision trees, and machine learning (including deep learning)

# Game Changers

- Mathematical innovations
- Hardware
- Big Data



BloombergMarkets | Nasdaq's Latest Deal Shows That Data Reigns Supreme

## Nasdaq's Latest Deal Shows That Data Reigns Supreme

By **Annie Massa**

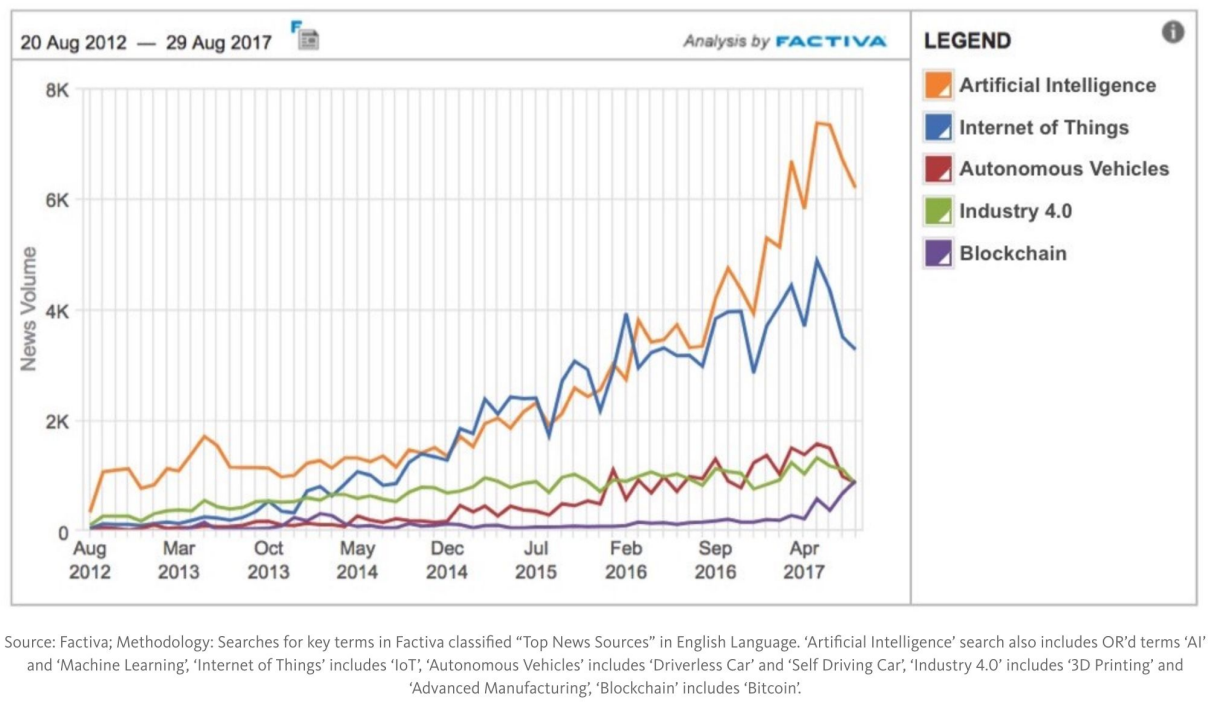
September 5, 2017, 7:33 AM PDT *Updated on* September 5, 2017, 12:57 PM PDT

- Global exchange buys EVestment for $705 million in debt, cash
- EVestment offers specialized data for institutional investors

Nasdaq Inc.'s takeover of data provider EVestment Inc. spotlights the exchange industry's seemingly bottomless appetite for data and its emerging focus on cultivating a closer relationship with money managers.

# AI and the Technological Singularity

https://hackernoon.com/why-ai-is-now-on-the-menu-at-dinner-even-with-my-non-tech-friends-44c666348de4




Source: Factiva; Methodology: Searches for key terms in Factiva classified "Top News Sources" in English Language. 'Artificial Intelligence' search also includes OR'd terms 'AI' and 'Machine Learning', 'Internet of Things' includes 'IoT', 'Autonomous Vehicles' includes 'Driverless Car' and 'Self Driving Car', 'Industry 4.0' includes '3D Printing' and 'Advanced Manufacturing', 'Blockchain' includes 'Bitcoin'.

## Transformation vs Change

Kurzweil claims that AI will pass the Turing test in 2029, and the singularity will come in 2045.

https://futurism.com/kurzweil-claims-that-the-singularity-will-happen-by-2045/

# Huge Demand for AI Talent

https://www.nytimes.com/2017/10/22/technology/artificial-intelligence-experts-salaries.html

Home The New York Times TECHNOLOGY Share 149

## Tech Giants Are Paying Huge Salaries for Scarce A.I. Talent

Nearly all big tech companies have an artificial intelligence project, and they are willing to pay experts millions of dollars to help get it done.

查看简体中文版 | 查看繁體中文版

By CADE METZ OCT. 22, 2017

# Definition of AI

- Intelligence exhibited by machines
- Narrow or Weak AI: “Expert systems that match or exceed human intelligence in a narrowly defined area, but not in broader areas” ( Dvorsky G., 2013) e.g. Siri.
- Artificial General Intelligence: An artificial neural network not preprogrammed with fixed rules. Rewire itself to reflect patterns in the data, adaptable to its environment, in which (hopefully) advanced skills emerge organically.
- “Humans don’t learn to understand language by memorizing dictionaries and grammar books, so why should we possibly expect our computers to do so?” ( LEWIS-KRAUS G, 2016).
- And, Super AI?

http://io9.gizmodo.com/how-much-longer-before-our-first-ai-catastrophe-464043243 ( Dvorsky G., 2013)https://mobile.nytimes.com/2016/12/14/magazine/the-great-ai-awakening.html?_r=0&referer= ( LEWIS-KRAUS G, 2016)

# Two Types of AI

- Rule-Based AI
- Data-driven AI
- Example: Checkers. Albert Samuel @IBM began writing code for a checkers game program in 1949. In 1956, the program was demonstrated to the public on live television. In 1962, the computer beat checkers master player Robert Nealey, and IBM's stocks rose 15 percent overnight.
- Rule-based AI can never be more intelligent than its creators, but data-driven AI can!

# AlphaGoZero

https://www.quantamagazine.org/artificial-intelligence-learns-to-learn-entirely-on-its-own-20171018/

ABSTRACTIONS BLOG

# Artificial Intelligence Learns to Learn Entirely on Its Own

2

*A new version of AlphaGo needed no human instruction to figure out how to clobber the best Go player in the world — itself.*

A mere 19 months after dethroning the world's top human Go player, the computer program AlphaGo has smashed an even more momentous barrier: It can now achieve unprecedented levels of mastery purely by teaching itself. Starting with zero knowledge of Go strategy and no training by humans, the new iteration of the program, called AlphaGo Zero, needed just three days to invent advanced strategies undiscovered by human players in the multi-millennia history of the game. By freeing artificial intelligence from a dependence on human knowledge, the breakthrough removes a primary limit on how smart machines can become.

Still it seems at least plausible to expect artificial intelligence in investing to outstrip human intelligence in discomfiting ways. If your computer figures out for itself how to invest using "advanced strategies undiscovered by human players," and if those strategies are so advanced that the computer can't even explain them to you, and if they make money, then what do you do? Do you trust that the computer knows what it's doing better than you do, and stand aside and give it all your money? Or do you worry that the computer's success reflects a particular market regime, or that the computer's strategy will break down once it starts working with a lot of money? Do you trust that the *computer* is thinking about these things, that it can see the big picture rather than just grabbing whatever works right now?

# An Early History of AI

- Warren McCulloch and Walter Pitts (1943): proposed a model of artificial neurons in which each neuron is characterized as being "on" or "off" ; suggested that suitably defined networks could learn.
- The Turing Test (Alan Turing, 1950):  designed to provide a satisfactory operational definition of intelligence. Intelligent behavior: the ability to achieve human-level performance in all cognitive tasks, sufficient to fool an interrogator.
- John McCarthy proposed the term  ARTIFICIAL INTELLIGENCE (1956);  defined the high-level language Lisp.
- Frank Rosenblatt (1962) - “perceptron convergence theorem”: his learning algorithm could adjust the connection strengths of a perceptron to match any input data.

# An Early History of AI

- Japanese"Fifth Generation" project (1981) , a 10-year plan to build intelligent computers code;  proposed to achieve full-scale natural language understanding.
- Fueled interest in the field in United States, led to the formation of The Microelectronics and Computer Technology Corporation i (MCC) to counter the Japanese project.
- Reinvention of the back-propagation learning algorithm (1969) in mid 1980s.
- HITECH:  the first computer program to defeat a grandmaster in a game of chess (Berliner, 1989).
- MARVEL: a real-time expert system that monitored massive data transmitted by a spacecraft, handling routine tasks and alerting the analysts to more serious problems (Schwuttke, 1992).
- PEGASUS: A speech understanding program to manage travel transactions (Zue et al., 1994).

Ref: Artificial intelligence : a modern approach/ Stuart Russell, Peter Norvig. 1995

# Machine Learning

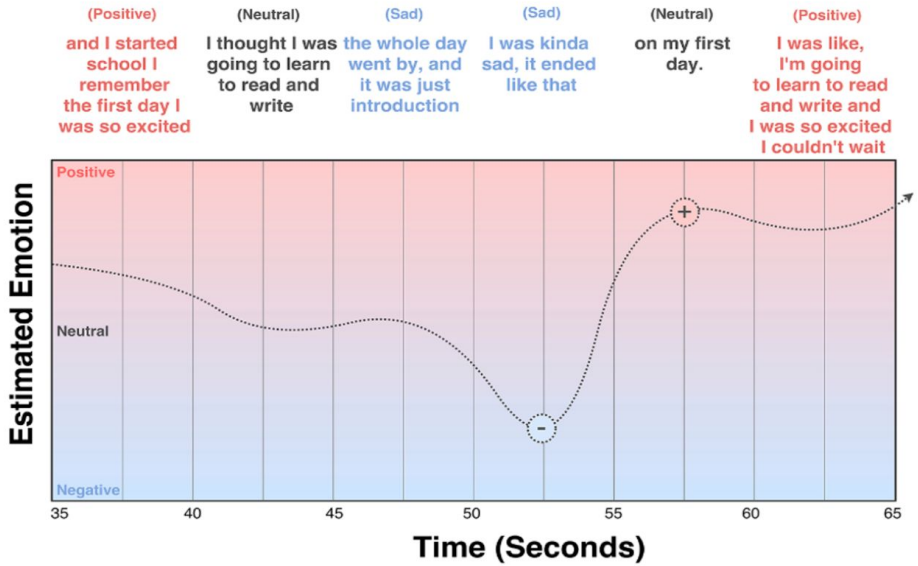
- Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed." (Arthur Samuel)

- Definition (Tom Mitchell): "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*."

- Supervised vs Unsupervised.

# AI in Wearables

## Emotion detectors Could Tell You When The Conversation Is Getting Awkward!




Graph showing real-time emotion detection *[Graphic: courtesy of MIT CSAIL]*


https://www.fastcompany.com/3066867/wearables-that-detect-emotion-could-tell-you-when-the-conversation-is-getting-awkward

# FaceApp

**FaceApp:** AI Selfie Transformation App Can Even Make Mona Lisa Smile !

Uses neural networks to edit your selfie via photo-realistic filters


https://news.developer.nvidia.com/this-ai-selfie-transformation-app-can-even-make-mona-lisa-smile/?_lrsc=a06a736b-bf67-4a30-b40f-368d4225b35b&ncid=so-lin-lt-798

# Conversational AI

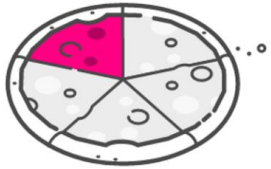
# Chat Bots


AMAZON'S ALEXA
GOOGLE'S ASSISTANT
What can I help you with?
APPLE'S SIRI
MICROSOFT'S CORTANA

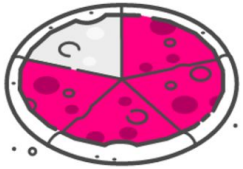
# Insurance

Lemonade

- Improving fairness of the insurance industry through Maya, their artificial intelligence bot
- Social Impact is the legal mission and business model
- Faster, easier



A **transparent** 20% fee to run everything



We pay claims **super fast**

If there's money leftover, we **give it back to causes**

https://www.lemonade.com/

# Healthcare

## Remedy

- Improving primary health care treatment
- Remy: The world's smartest medical AI
- Taught by physicians, driven by data
- No insurance required

## Deep Patient:

- **An Unsupervised Representation**
- **to Predict the Future of Patients**
- **from the Electronic Health Records**
- discovered patterns hidden in the hospital data to indicate
- people's disposition to a wide range of ailments,
- including cancer of the liver

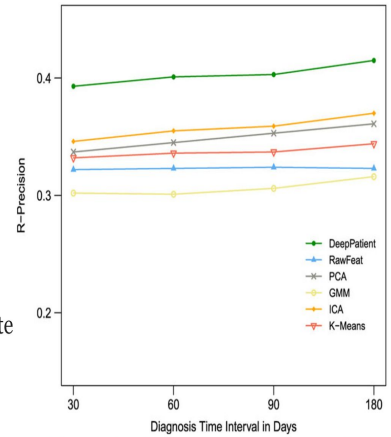



Figure 3: R-precision obtained in the disease tagging experiment by the different patient representations over several prediction time intervals (expressed as number of days).

https://www.remedymedical.com/

https://www.nature.com/articles/srep26094

# Estimating the effects of technology

- (Roy) Amara's Law: "We tend to overestimate the effect of a technology in the short run and underestimate the effect in the long run."

- Arthur C. Clarke's Three Laws:
  a. When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.
  b. The only way of discovering the limits of the possible is to venture a little way past them into the impossible.
  c. Any sufficiently advanced technology is indistinguishable from magic.

# Deep Learning in Finance

- Bridgewater Associates: World’s largest hedge fund has a project to automate decision-making to save time and eliminate human emotion volatility.

- Goldman Sachs: Two out of the 600 equity traders left. Found that four traders can be replaced by one computer engineer.

# Deep Learning in Finance

- Transactions: By 2020 at least five percent of all economic transactions will be handled by autonomous software. AI will process payment functions and learn from customer behaviours, through Intelligent Payment Management (IPM).

- Savings: AI will help consumers make daily financial decisions and monitor spending. New Personal Financial Management apps use contextual awareness, which measures spending habits and online footprints to create personalised advice. Combining pooled financial data with end-user control to offer tailor-made services is a classic AI solution.

# Deep Learning in Finance

- Cross-selling: Categorization-as-a-Service (Caas), for understanding customer transactions for cross selling.

- Fraud prevention: Mining user data to detect abnormal behavior, anomalies, and unusual transactions.

# Deep Learning in Finance

- Mizuho Financial Group sent Pepper, its humanoid robot into its Tokyo branch to handle customer inquiries. Partnering with IBM to enable Pepper to understand human emotions, and build interaction into apps.

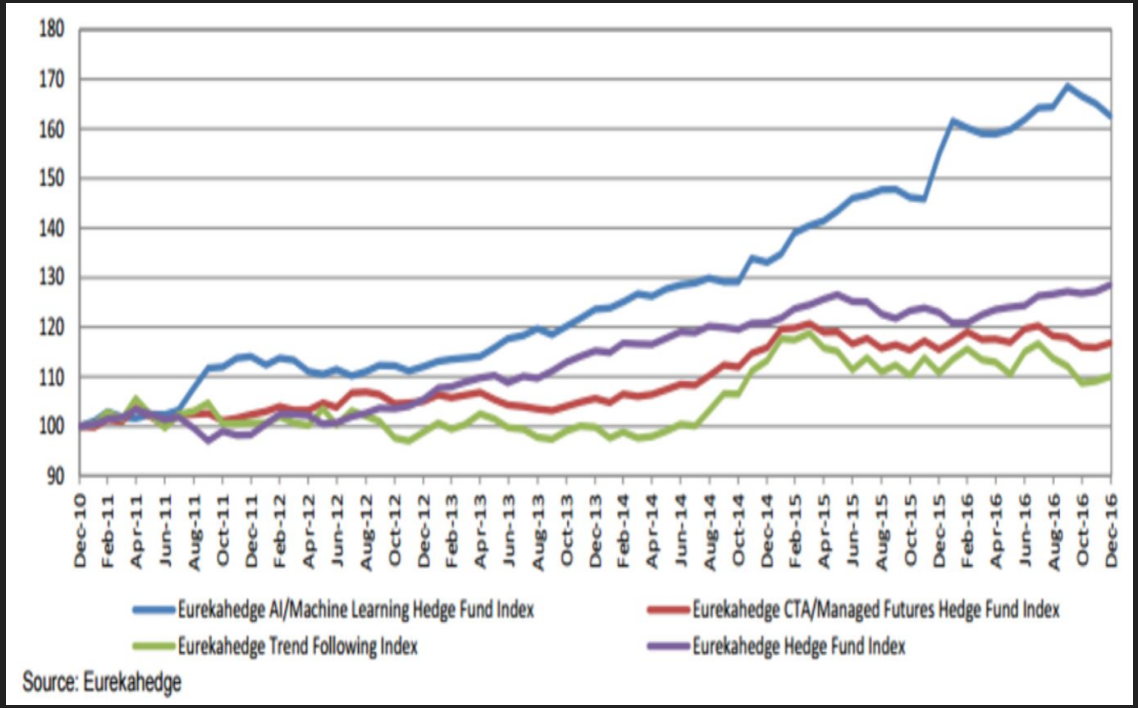
- RBS is trialing Luvo AI, a customer service assistant to interact with staff and customers.

# Deep Learning in Finance

- AXA (insurer) has an app-based bot called Xtra, it engages in bespoke conversations with customers about healthy living.

- AI is used in peer2peer lending.

# Hedge Funds use Machine Learning




Very little data about the track record of these hedge funds, as the business remains secretive.

Investor reluctance to turn over money completely to a machine.

Blackrock: replacing human stock pickers with machine algorithms.

Sentient Inc: Hedge fund run entirely using AI. Secret algo with adaptive learning. Uses thousands of machines.

Numerai: Hedge fund makes trades by aggregating trading algorithms submitted by anonymous contributors, prizes are awarded in cryptocurrency.

Emma: Evolved a hedge fund using a software that writes news articles.

# Financial Automation

BloombergMarkets

JPMorgan Software Does in Seconds What Took Lawyers 360,000 Hours

## JPMorgan Software Does in Seconds What Took Lawyers 360,000 Hours

by **Hugh Son**

February 27, 2017, 4:31 PM PST *Updated on* February 28, 2017, 4:24 AM PST

At JPMorgan Chase & Co., a learning machine is parsing financial deals that once kept legal teams busy for thousands of hours.

The program, called COIN, for Contract Intelligence, does the mind-numbing job of interpreting commercial-loan agreements that, until the project went online in June, consumed 360,000 hours of work each year by lawyers and loan officers. The software reviews documents in seconds, is less error-prone and never asks for vacation.

# Deep Learning is Pattern Recognition

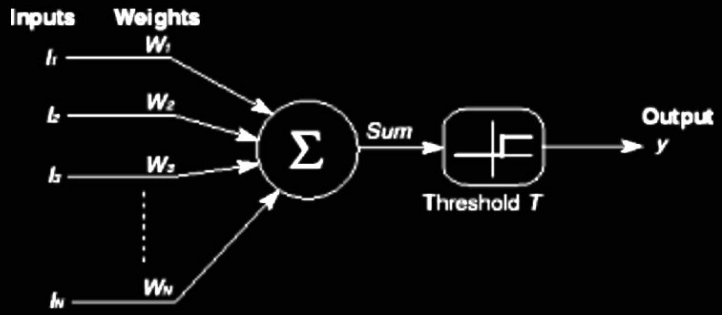



Figure 1.1: McCullough Pitts neuron

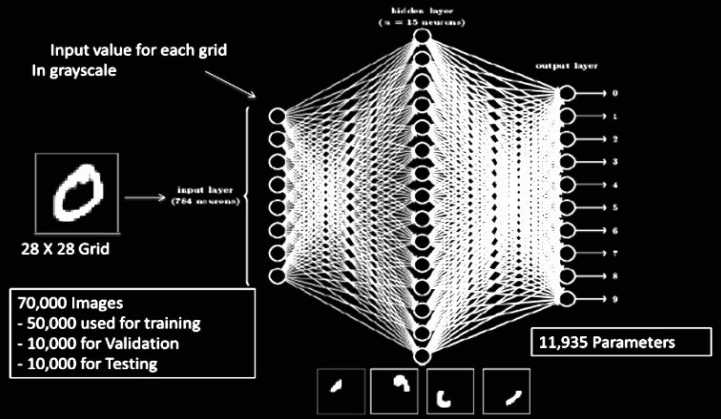



Figure 2.2: The NN used for Solving the MNIST Digit Recognition Problem

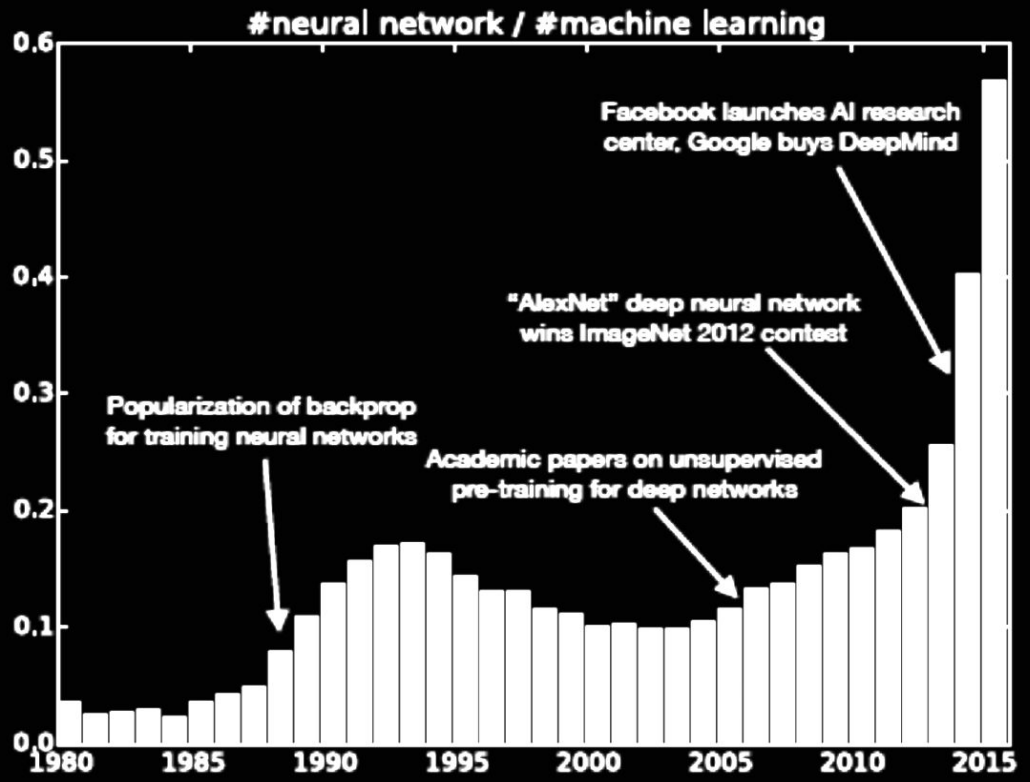
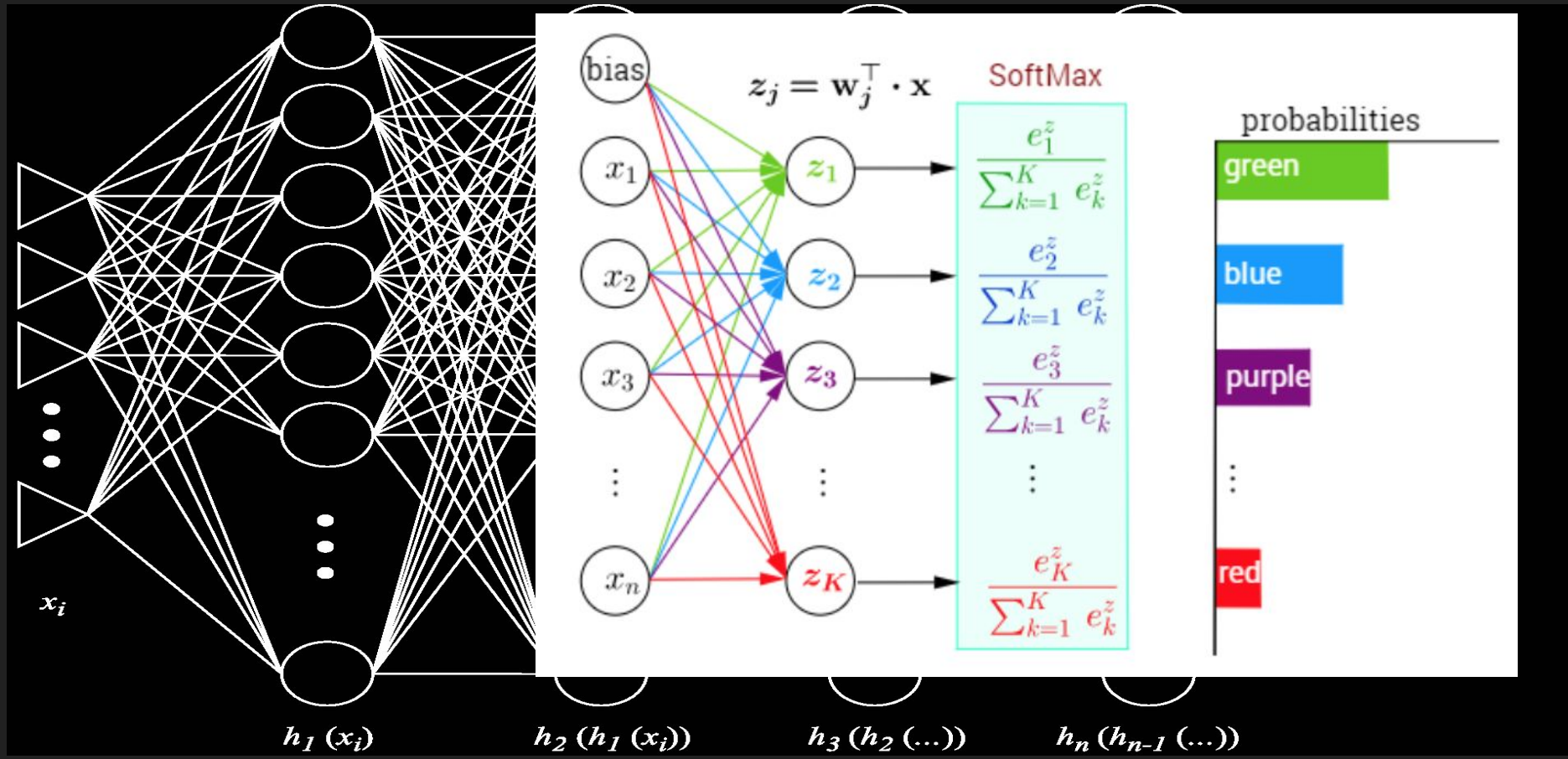
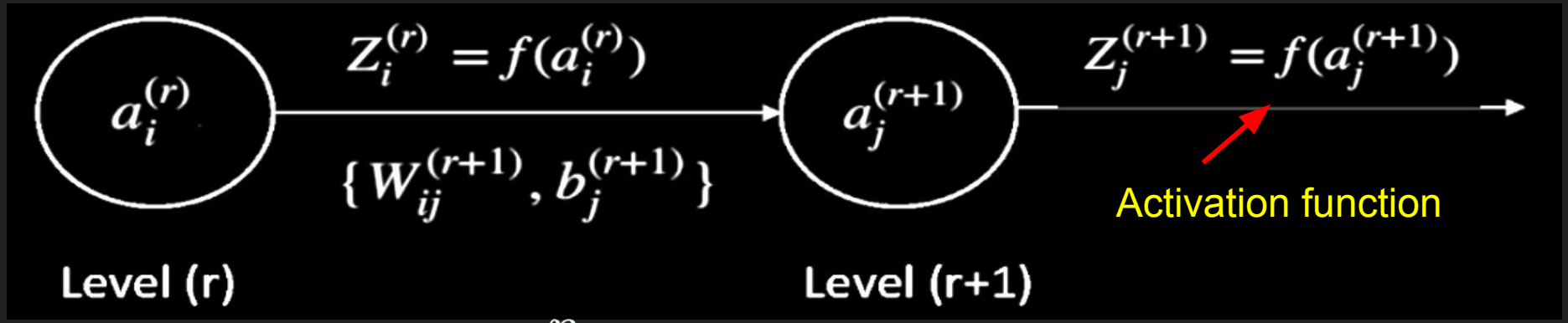



Figure 1.2: Number of NN research reports

# The Mathematics of Deep Learning (http://srdas.github.io/DLBook/)



# Subset of the Net

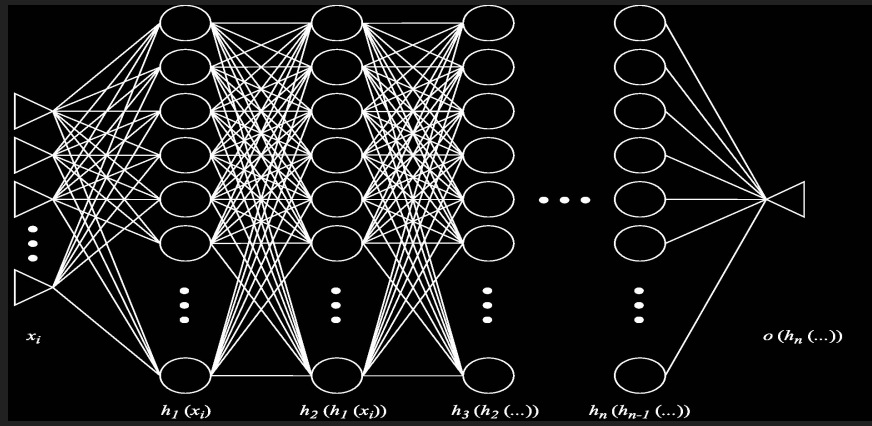



$$a_j = b_j + \sum_{I=1}^{n} w_{ij} z_i$$

Net input

$$\sigma(a_j) = \frac{1}{1 + \exp(-a_j)}$$

Sigmoid function



# Activation Functions

https://en.wikipedia.org/wiki/Activation_function

| Activation function | Equation | Example | 1D Graph |
|---|---|---|---|
| Unit step (Heaviside) | $\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$ | Perceptron variant |  |
| Sign (Signum) | $\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$ | Perceptron variant |  |
| Linear | $\phi(z) = z$ | Adaline, linear regression |  |
| Piece-wise linear | $\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$ | Support vector machine |  |
| Logistic (sigmoid) | $\phi(z) = \frac{1}{1 + e^{-z}}$ | Logistic regression, Multi-layer NN |  |
| Hyperbolic tangent | $\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$ | Multi-layer NN |  |

# Loss Function

Fitting the DLN is an exercise where the best weights $\{W, b\} = \{W_{ij}^{(r+1)}, b_j^{(r+1)}\}, \forall r$ for all layers are determined to minimize a loss function generally denoted as

$$\min_{W,b} \sum_{m=1}^{M} L_m[h(X^{(m)}), T^{(m)}]$$

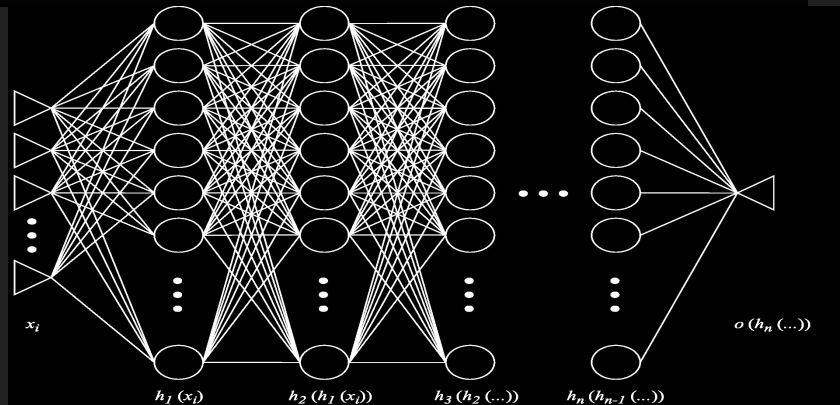
where $M$ is the number of training observations (rows in the data set), $T^{(m)}$ is the true value of the output, and $h(X^{(m)})$ is the model output from the DLN. The loss function $L_m$ quantifies the difference between the model output and the true output.

Cross Entropy

$$C = -\frac{1}{n} \sum_x [y \ln a + (1 - y) \ln(1 - a)]$$

Quadratic Loss

$$C = \frac{(y - a)^2}{2}$$



# Gradient Descent

Fitting the DLN requires getting the weights $\{W, b\}$ that minimize $L_m$. These are done using gradient descent, i.e.,

$$W_{ij}^{(r+1)} \leftarrow W_{ij}^{(r+1)} - \eta \cdot \frac{\partial L_m}{\partial W_{ij}^{(r+1)}}$$

$$b_j^{(r+1)} \leftarrow b_j^{(r+1)} - \eta \cdot \frac{\partial L_m}{\partial b_j^{(r+1)}}$$

Here $\eta$ is the learning rate parameter. We iterate on these functions until the gradients become zero, and the weights discontinue changing with each update, also known as an **epoch**.

Total Gradient

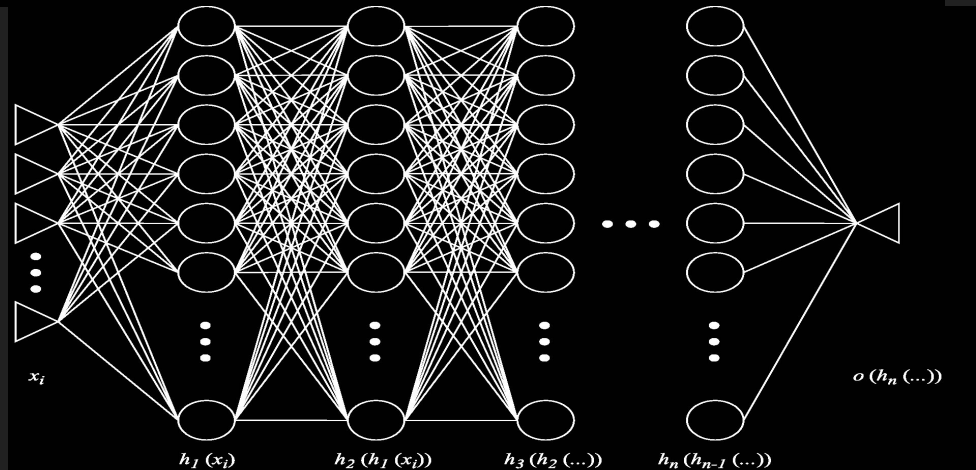
$$w_{ij}^{(r)} \leftarrow w_{ij}^{(r)} - \frac{\eta}{M} \sum_{m=1}^{M} \frac{\partial L_m(w, b; X(m), T(m))}{\partial w_{ij}^{(r)}}$$

$$b_i^{(r)} \leftarrow b_i^{(r)} - \frac{\eta}{M} \sum_{m=1}^{M} \frac{\partial L_m(w, b; X(m), T(m))}{\partial b_i^{(r)}}$$

Stochastic Batch Gradient

$$w_{ij}^{(r)} \leftarrow w_{ij}^{(r)} - \frac{\eta}{s} \sum_{m=1}^{s} \frac{\partial L_m(w, b; X(m), T(m))}{\partial w_{ij}^{(r)}}$$

$$b_i^{(r)} \leftarrow b_i^{(r)} - \frac{\eta}{s} \sum_{m=1}^{s} \frac{\partial L_m(w, b; X(m), T(m))}{\partial b_i^{(r)}}$$



# Gradients and the Chain Rule

To solve this minimization problem, we need gradients for all $W, b$. These are denoted :

$$a_j^{(r+1)} = \sum_{i=1}^{n_r} W_{ij}^{(r+1)} Z_i^{(r)} + b_j^{(r+1)}$$

$$\frac{\partial L_m}{\partial W_{ij}^{(r+1)}}, \quad \frac{\partial L_m}{\partial b_j^{(r+1)}}, \quad \forall r+1, j$$

We write out these gradients using the chain rule:

$$\frac{\partial L_m}{\partial W_{ij}^{(r+1)}} = \frac{\partial L_m}{\partial a_j^{(r+1)}} \cdot \frac{\partial a_j^{(r+1)}}{\partial W_{ij}^{(r+1)}} = \delta_j^{(r+1)} \cdot Z_i^{(r)}$$

where we have written

$$\delta_j^{(r+1)} = \frac{\partial L_m}{\partial a_j^{(r+1)}}$$

Likewise, we have

$$\frac{\partial L_m}{\partial b_j^{(r+1)}} = \frac{\partial L_m}{\partial a_j^{(r+1)}} \cdot \frac{\partial a_j^{(r+1)}}{\partial b_j^{(r+1)}} = \delta_j^{(r+1)} \cdot 1 = \delta_j^{(r+1)}$$

# Delta Values

we need the following intermediate calculation

$$a_j^{(r+1)} = \sum_{i=1}^{n_r} W_{ij}^{(r+1)} Z_i^{(r)} + b_j^{(r+1)}$$

$$= \sum_{i=1}^{n_r} W_{ij}^{(r+1)} f(a_i^{(r)}) + b_j^{(r+1)}$$

$$\frac{\partial a_j^{(r+1)}}{\partial a_i^{(r)}} = W_{ij}^{(r+1)} \cdot f'(a_i^{(r)})$$

$$\delta_i^{(r)} = \frac{\partial L_m}{\partial a_i^{(r)}}$$

$$= \sum_{j=1}^{n_{r+1}} \frac{\partial L_m}{\partial a_j^{(r+1)}} \cdot \frac{\partial a_j^{(r+1)}}{\partial a_i^{(r)}}$$

$$= \sum_{j=1}^{n_{r+1}} \delta_j^{(r+1)} \cdot W_{ij}^{(r+1)} \cdot f'(a_i^{(r)})$$

$$= f'(a_i^{(r)}) \cdot \sum_{j=1}^{n_{r+1}} \delta_j^{(r+1)} \cdot W_{ij}^{(r+1)}$$

# Output Layer

$$a_j^{(R+1)} = \sum_{i=1}^{n_R} W_{ij}^{(R+1)} Z_j^{(R)} + b_j^{(R+1)}$$

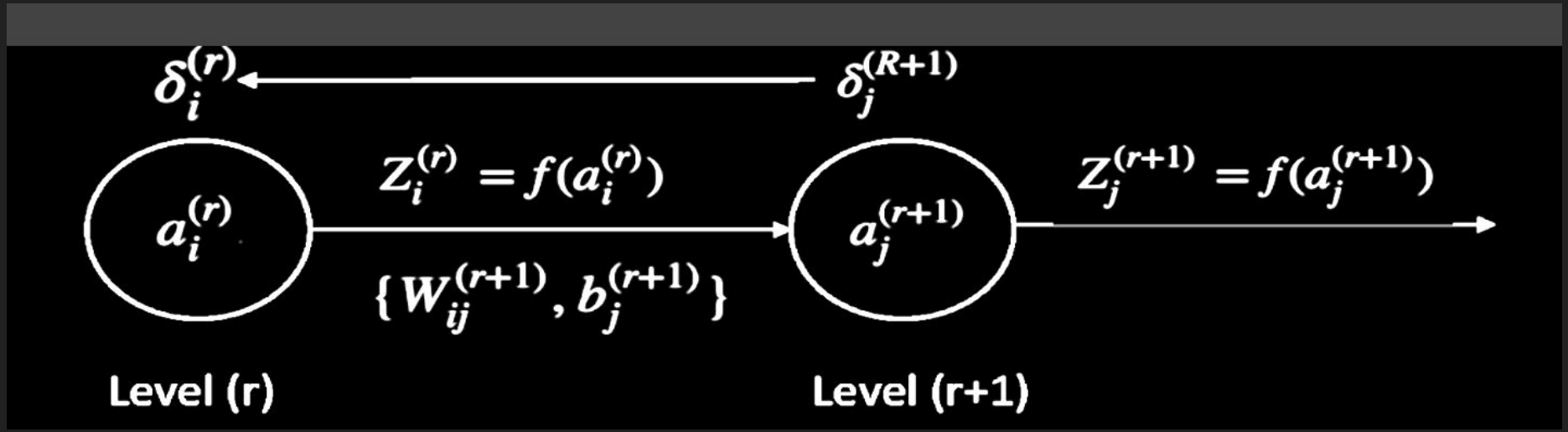
$$\text{Final output} = h(a_j^{(R+1)})$$

$$\delta_j^{(R+1)} = \frac{\partial L_m}{\partial a_j^{(R+1)}} = h'(a_j^{(R+1)})$$

## Iterative Process

1. Start with an initial set of weights $\{w, b\}$.
2. Feedforward the initial data and weights into the DLN, and find the $\{a_i^{(r)}, Z_i^{(r)}\}, \forall r, i$.
3. Then, using backpropagation, compute all $\delta_i^{(r)}, \forall r, i$.
4. Use these $\delta_i^{(r)}$ values to get all the new gradients.
5. Apply gradient descent to get new weights.
6. Keep iterating steps 2-5, until the chosen number of epochs is completed.

# Feedforward and Backprop



# Recap

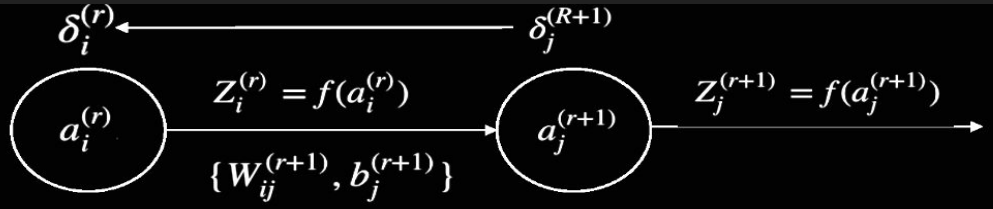
$$\frac{\partial L_m}{\partial W_{ij}^{(r+1)}} = \frac{\partial L_m}{\partial a_j^{(r+1)}} \cdot \frac{\partial a_j^{(r+1)}}{\partial W_{ij}^{(r+1)}} = \delta_j^{(r+1)} \cdot Z_i^{(r)}$$

$$\delta_j^{(r+1)} = \frac{\partial L_m}{\partial a_j^{(r+1)}}$$

$$= f'(a_i^{(r)}) \cdot \sum_{j=1}^{n_{r+1}} \delta_j^{(r+1)} \cdot W_{ij}^{(r+1)}$$

$$\frac{\partial L_m}{\partial b_j^{(r+1)}} = \frac{\partial L_m}{\partial a_j^{(r+1)}} \cdot \frac{\partial a_j^{(r+1)}}{\partial b_j^{(r+1)}} = \delta_j^{(r+1)} \cdot 1 = \delta_j^{(r+1)}$$

# The Magic of Backpropagation




$$\min_{W,b} \sum_{m=1}^{M} L_m[h(X^{(m)}), T^{(m)}]$$

Loss Function

$$a_j^{(r+1)} = \sum_{i=1}^{n_r} W_{ij}^{(r+1)} Z_i^r + b_j^{(r+1)} = \sum_{i=1}^{n_r} W_{ij}^{(r+1)} f(a_i^{(r)}) + b_j^{(r+1)}$$

$$\frac{\partial a_j^{(r+1)}}{\partial a_i^{(r)}} = W_{ij}^{(r+1)} \cdot f'(a_i^{(r)})$$

Gradients

$$\frac{\partial L_m}{\partial W_{ij}^{(r+1)}} = \frac{\partial L_m}{\partial a_j^{(r+1)}} \cdot \frac{\partial a_j^{(r+1)}}{\partial W_{ij}^{(r+1)}} = \delta_j^{(r+1)} \cdot Z_i^{(r)}$$

$$\frac{\partial L_m}{\partial b_j^{(r+1)}} = \frac{\partial L_m}{\partial a_j^{(r+1)}} \cdot \frac{\partial a_j^{(r+1)}}{\partial b_j^{(r+1)}} = \delta_j^{(r+1)} \cdot 1 = \delta_j^{(r+1)}$$

$$\begin{aligned}
\delta_i^{(r)} &= \frac{\partial L_m}{\partial a_i^{(r)}} \\
&= \sum_{j=1}^{n_{r+1}} \frac{\partial L_m}{\partial a_j^{(r+1)}} \cdot \frac{\partial a_j^{(r+1)}}{\partial a_i^{(r)}} \\
&= \sum_{j=1}^{n_{r+1}} \delta_j^{(r+1)} \cdot W_{ij}^{(r+1)} \cdot f'(a_i^{(r)}) \\
&= f'(a_i^{(r)}) \cdot \sum_{j=1}^{n_{r+1}} \delta_j^{(r+1)} \cdot W_{ij}^{(r+1)}
\end{aligned}$$

Deltas

$$\delta_j^{(R+1)} = \frac{\partial L_m}{\partial a_j^{(R+1)}} = h'(a_j^{(R+1)})$$

Gradient Descent

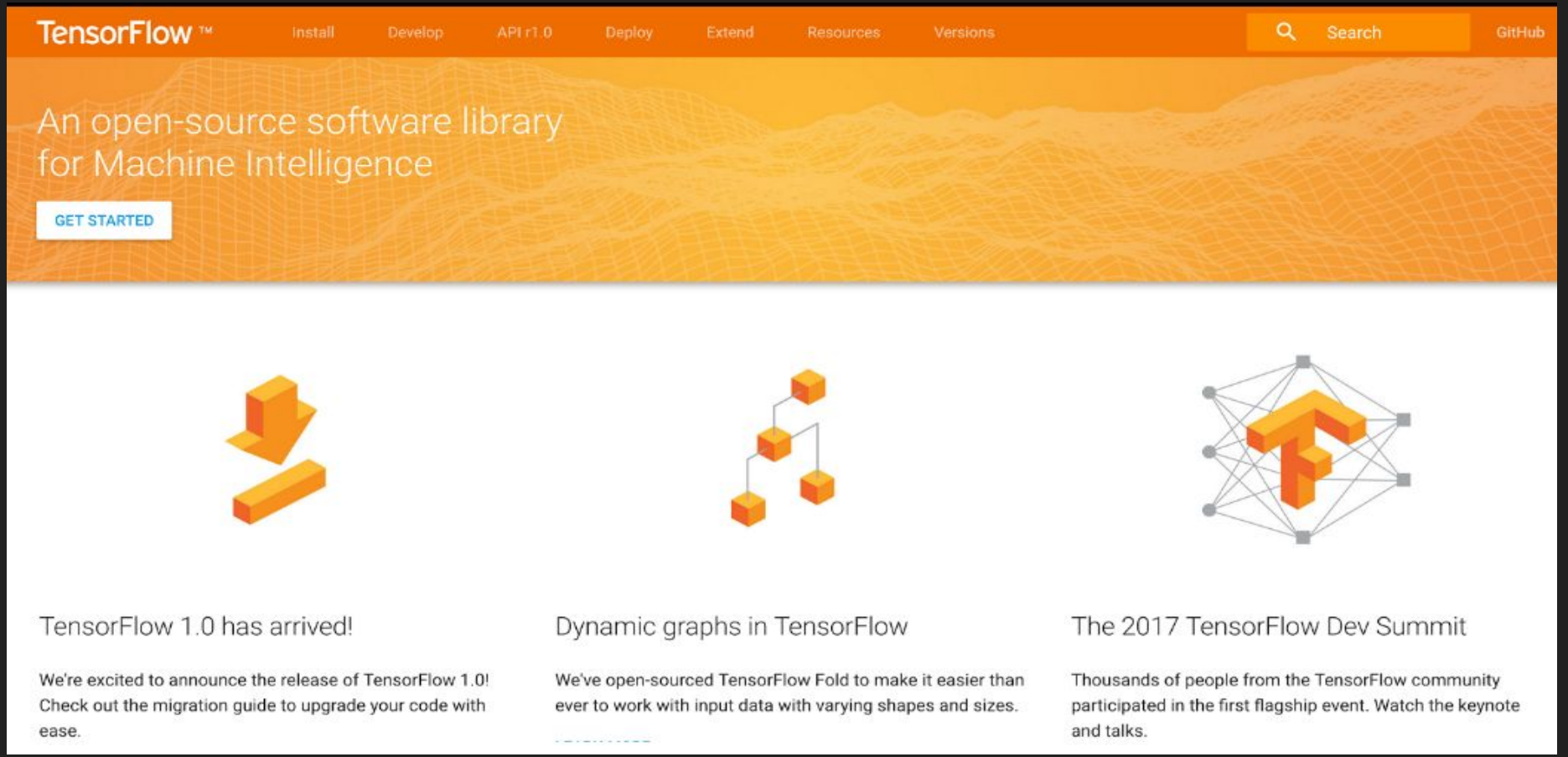
$$W_{ij}^{(r+1)} \leftarrow W_{ij}^{(r+1)} - \eta \cdot \frac{\partial L_m}{\partial W_{ij}^{(r+1)}}$$

$$b_j^{(r+1)} \leftarrow b_j^{(r+1)} - \eta \cdot \frac{\partial L_m}{\partial b_j^{(r+1)}}$$

# Genesis: Breast Cancer Detection

- We use a data set of breast cancer cell measurements and assess whether we can predict whether a tumor is malignant or benign.
- The data used comprises nine cell measurements such as cell thickness, dimension, etc., and from these measurements we wish to attain a high level of accuracy in classification of cells.
- A system such as this can potentially be the first line of diagnosis and may be able to improve the doctor's diagnosis, in a Bayesian sense, as there are now two readings of the cell, by the deep learning net and the doctor.
- The data set is the Wisconsin breast cancer data.

# TensorFlow

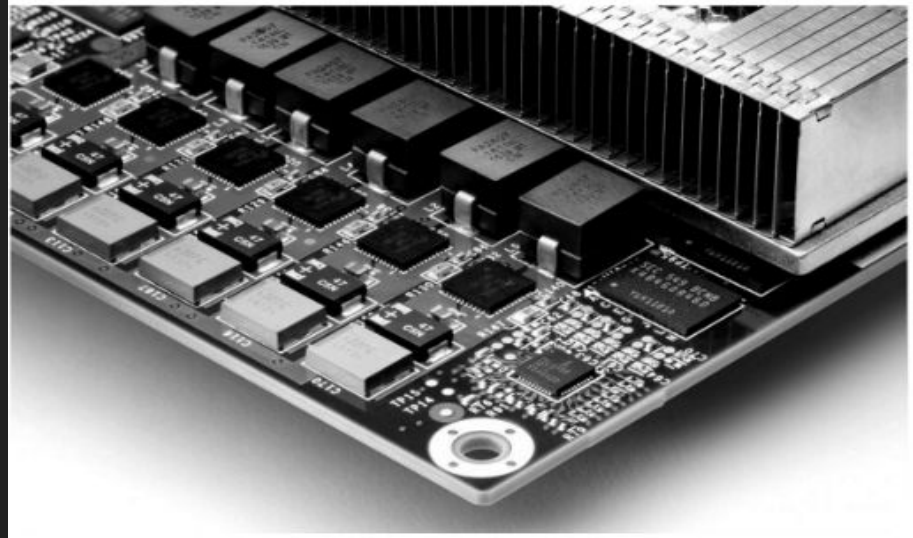

TensorFlow ™
Install
Develop
API r1.0
Deploy
Extend
Resources
Versions
Search
GitHub
An open-source software library
for Machine Intelligence
GET STARTED
TensorFlow 1.0 has arrived!
We're excited to announce the release of TensorFlow 1.0! Check out the migration guide to upgrade your code with ease.
Dynamic graphs in TensorFlow
We've open-sourced TensorFlow Fold to make it easier than ever to work with input data with varying shapes and sizes.
The 2017 TensorFlow Dev Summit
Thousands of people from the TensorFlow community participated in the first flagship event. Watch the keynote and talks.

# Special Purpose Chips

## Google's Tensor Processing Unit explained: this is what the future of computing looks like

By Joe Osborne August 22, 2016 Processors

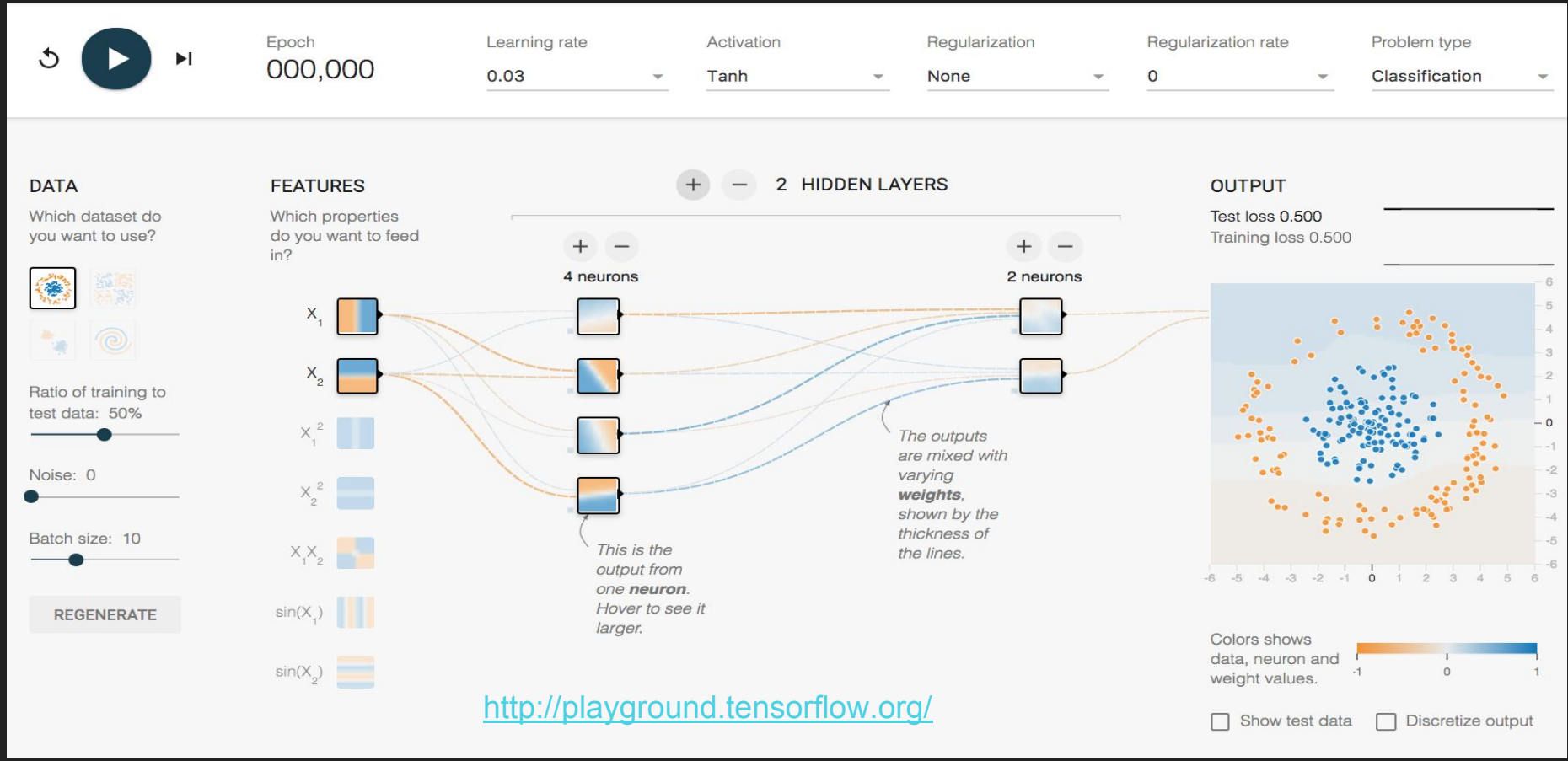
Specialized silicon will lead the way forward



## Google's Big Chip Unveil For Machine Learning: Tensor Processing Unit With 10x Better Efficiency (Updated)

by Lucian Armasu May 19, 2016 at 9:10 AM - Source: Google Cloud Platform Blog

# TensorFlow Playground




http://playground.tensorflow.org/

# Genesis: Breast Cancer Detection

- We use a data set of breast cancer cell measurements and assess whether we can predict whether a tumor is malignant or benign.
- The data used comprises nine cell measurements such as cell thickness, dimension, etc., and from these measurements we wish to attain a high level of accuracy in classification of cells.
- A system such as this can potentially be the first line of diagnosis and may be able to improve the doctor's diagnosis, in a Bayesian sense, as there are now two readings of the cell, by the deep learning net and the doctor.
- The data set is the Wisconsin breast cancer data.

# Cancer Data

In [2]: Slide Type: Skip

```
%pylab inline
import pandas as pd
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [4]: Slide Type: Slide

```
## Read in the data set
data = pd.read_csv("BreastCancer.csv")
data.head()
```

Out[4]:

| | Id | Cl.thickness | Cell.size | Cell.shape | Marg.adhesion | Epith.c.size | Bare.nuclei | Bl.cromatin |
|---|---|---|---|---|---|---|---|---|
| 0 | 1000025 | 5 | 1 | 1 | 1 | 2 | 1 | 3 |
| 1 | 1002945 | 5 | 4 | 4 | 5 | 7 | 10 | 3 |
| 2 | 1015425 | 3 | 1 | 1 | 1 | 2 | 2 | 3 |
| 3 | 1016277 | 6 | 8 | 8 | 1 | 3 | 4 | 3 |
| 4 | 1017023 | 4 | 1 | 1 | 3 | 2 | 1 | 3 |

```
x = data.loc[:,'Cl.thickness':'Mitoses']
print(x.head())
y = data.loc[:,'Class']
print(y.head())
```

```
   Cl.thickness  Cell.size  Cell.shape  Marg.adhesion  Epith.c.size
0             5          1           1              1             2
1             5          4           4              5             7
2             3          1           1              1             2
3             6          8           8              1             3
4             4          1           1              3             2

   Bare.nuclei  Bl.cromatin  Normal.nucleoli  Mitoses
0            1            3                1        1
1           10            3                2        1
2            2            3                1        1
3            4            3                7        1
4            1            3                1        1
0    benign
1    benign
2    benign
3    benign
4    benign
Name: Class, dtype: object
```

https://futurism.com/machine-learning-identifies-breast-lesions-likely-to-become-cancer/

# TensorFlow and One-Hot Encoding

```
## Convert the class variable into binary numeric
ynum = zeros((len(x),1))
for j in arange(len(y)):
    if y[j]=="malignant":
        ynum[j]=1
ynum[:10]
```

```
array([[ 0.],
       [ 0.],
       [ 0.],
       [ 0.],
       [ 0.],
       [ 1.],
       [ 0.],
       [ 0.],
       [ 0.],
       [ 0.]])
```

```
## Define the neural net and compile it
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()
model.add(Dense(32, activation='relu', input_dim=9))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
```

```
## Make label data
from keras import u
y.labels = utils.to
x = x.as_matrix()
print(y.labels[:10]
print(shape(x))
print(shape(y.label
print(shape(ynum))
```

```
Using TensorFlow back
```

```
[[ 1.  0.]
 [ 1.  0.]
 [ 1.  0.]
 [ 1.  0.]
 [ 1.  0.]
 [ 0.  1.]
 [ 1.  0.]
 [ 1.  0.]
 [ 1.  0.]
 [ 1.  0.]]
(683, 9)
(683, 2)
(683, 1)
```

```
## Accuracy
yhat = model.predict_classes(x, batch_size=32)
acc = sum(yhat==ynum)
print("Accuracy = ",acc/len(ynum))

## Confusion matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(yhat,ynum)
```

```
 32/683 [>.............................] - ETA: 0sAccuracy =  0.978038067
35

array([[431,   2],
       [ 13, 237]])
```

# Canonical Example: Digit Recognition

## THE MNIST DATABASE

### of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

| | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

60,000 x 784

Showing 1 to 25 of 60,000 entries

- Futile to run a multinomial regression on 784 variables
- Extensible to many finance problems

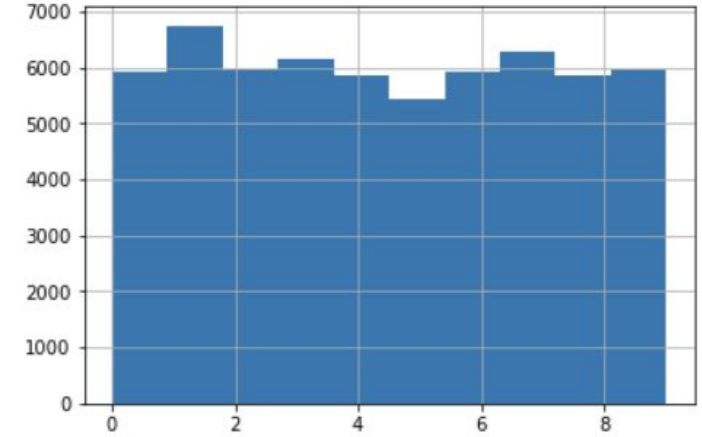
https://www.cs.toronto.edu/~kriz/cifar.html

# MNIST Data

```
## Read in the data set
train = pd.read_csv("train.csv", header=None)
test = pd.read_csv("test.csv", header=None)
print(shape(train))
print(shape(test))
```

```
(60000, 785)
(10000, 785)
```

```
hist(Y_train); grid()
```




```
## Reformat the data
X_train = train.as_matrix()[:,0:784]
Y_train = train.as_matrix()[:,784:785]
print(shape(X_train))
print(shape(Y_train))
X_test = test.as_matrix()[:,0:784]
Y_test = test.as_matrix()[:,784:785]
print(shape(X_test))
print(shape(Y_test))
y.labels = utils.to_categorical(Y_train, num_classes=10)
print(shape(y.labels))
print(y.labels[1:5,:])
print(Y_train[1:5])
```

```
(60000, 784)
(60000, 1)
(10000, 784)
(10000, 1)
(60000, 10)
[[ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]]
[[3]
 [0]
 [0]
 [2]]
```

# TF Model

```
## Define the neural net and compile it
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD

data_dim = shape(X_tra

model = Sequential()
model.add(Dense(100, a
model.add(Dropout(0.25
model.add(Dense(100, a
model.add(Dropout(0.25
model.add(Dense(100, a
model.add(Dropout(0.25
model.add(Dense(100, a
model.add(Dropout(0.25
model.add(Dense(10, ac
model.compile(optimize
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
## Fit/train the model (x,y need to be matrices)
model.fit(X_train, y.labels, epochs=5, batch_size=32,verbose=2)
```

```
Epoch 1/5
13s - loss: 0.3134 - acc: 0.9170
Epoch 2/5
13s - loss: 0.3083 - acc: 0.9197
Epoch 3/5
14s - loss: 0.3073 - acc: 0.9200
Epoch 4/5
13s - loss: 0.3048 - acc: 0.9211
Epoch 5/5
13s - loss: 0.2982 - acc: 0.9228
```

# Accuracy

```
## In Sample
yhat = model.predict_classes(X_train, batch_size=32)

## Confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(yhat,Y_train)
print(" ")
print(cm)

##
acc = sum(diag(cm))/len(Y_train)
print("Accuracy = ",acc)
```

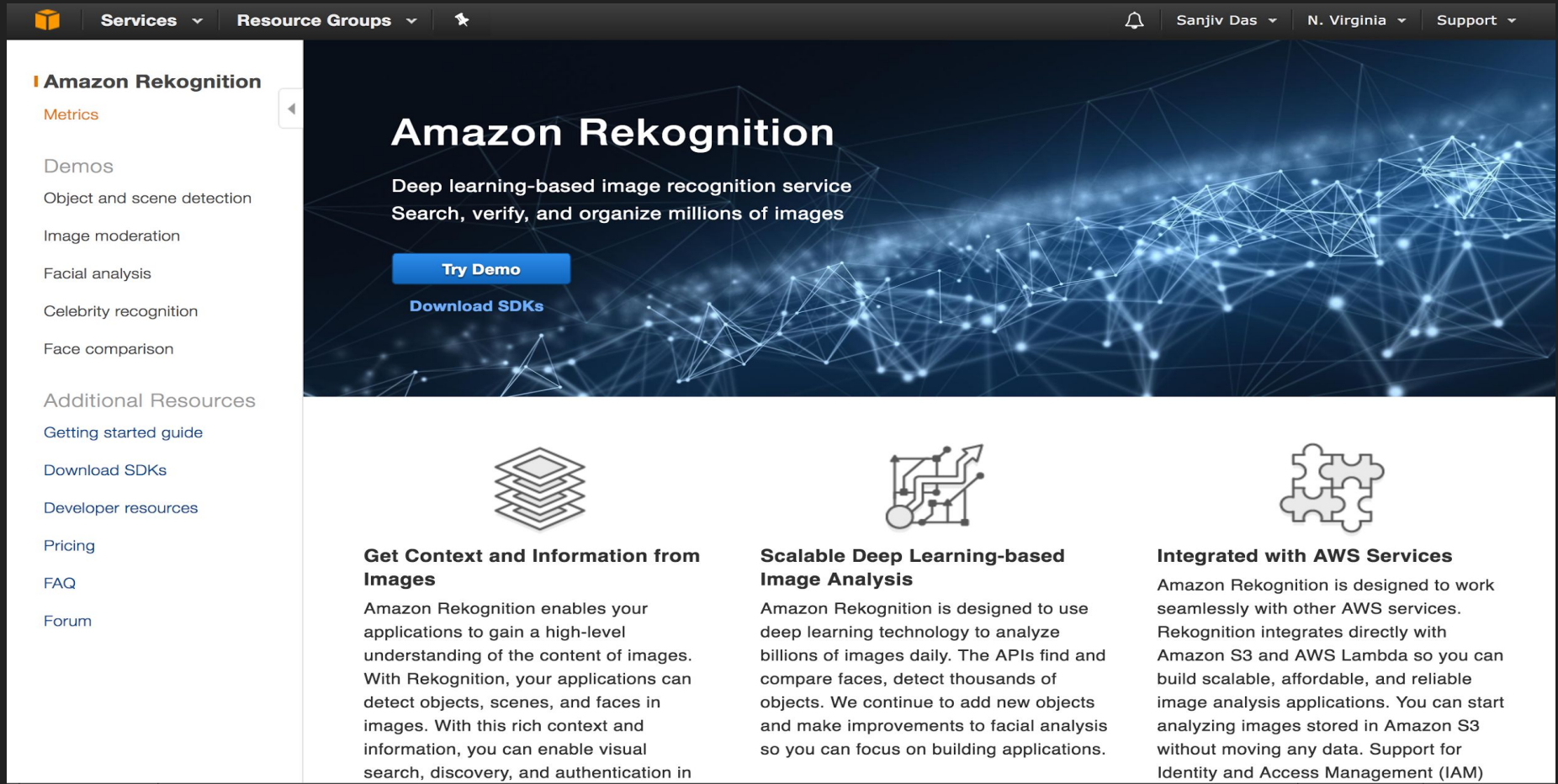
```
59392/60000 [============================>.] - ETA:
[[5749    0   45   16   11   39   29   23   20   42]
 [   0 6533   23   12   15    4   11   24   61    6]
 [  14   47 5568  167   19   27   14   59   55    5]
 [   8   22   48 5505    2  116    0   17   72   71]
 [  16    4   45    4 5522   21   20   48   26  203]
 [  26   23   11  185    5 5004   71    4   84   40]
 [  42    8   87   16   70   84 5747    1   39    4]
 [   3   19   41   70    4   16    0 5982   11   76]
 [  62   76   83  130   25   77   26    7 5435   89]
 [   3   10    7   26  169   33    0  100   48 5413]]
Accuracy =  0.940966666667
```

```
## Out of Sample
yhat = model.predict_classes(X_test, batch_size=32)

## Confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(yhat,Y_test)
print(" ")
print(cm)

##
acc = sum(diag(cm))/len(Y_test)
print("Accuracy = ",acc)
```

```
 9728/10000 [============================>.] - ETA: 0
[[ 959    0   11    1    2   12   16    2    7   10]
 [   0 1115    0    1    3    0    2   14    6    4]
 [   3    2  964   30    7    6    9   16   16    1]
 [   1    4   12  922    0   43    1    3   18    9]
 [   1    0    9    0  923    5    9    5   12   41]
 [   6    0    1   23    0  772   10    0   19    4]
 [   6    3   10    0   11   15  908    0   10    1]
 [   2    2   11   14    1    7    0  961    7   15]
 [   2    9   13   14    3   27    3    0  871    9]
 [   0    0    1    5   32    5    0   27    8  915]]
Accuracy =  0.931
```

# Amazon Rekognition

https://console.aws.amazon.com/rekognition/home?region=us-east-1#/label-detection



# Learning the Black-Scholes Equation (Culkin & Das, 2017)

THE JOURNAL OF FINANCE • VOL. XLIX, NO. 3 • JULY 1994

## A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks

JAMES M. HUTCHINSON, ANDREW W. LO, and TOMASO POGGIO*

**ABSTRACT**

We propose a nonparametric method for estimating the pricing formula of a derivative asset using learning networks. Although not a substitute for the more traditional arbitrage-based pricing formulas, network-pricing formulas may be more accurate and computationally more efficient alternatives when the underlying asset's price dynamics are unknown, or when the pricing equation associated with the no-arbitrage condition cannot be solved analytically. To assess the potential value of network pricing formulas, we simulate Black-Scholes option prices and show that learning networks can recover the Black-Scholes formula from a two-year training set of daily options prices, and that the resulting network formula can be used successfully to both price and delta-hedge options out-of-sample. For comparison, we estimate models using four popular methods: ordinary least squares, radial basis function networks, multilayer perceptron networks, and projection pursuit. To illustrate the practical relevance of our network pricing approach, we apply it to the pricing and delta-hedging of S&P 500 futures options from 1987 to 1991.

```
from scipy.stats import norm
def BSM(S,K,T,sig,rf,dv,cp):  #cp = {+1.0 (calls), -1.0 (puts)}
    d1 = (math.log(S/K)+(rf-dv+0.5*sig**2)*T)/(sig*math.sqrt(T))
    d2 = d1 - sig*math.sqrt(T)
    return cp*S*math.exp(-dv*T)*norm.cdf(d1*cp) - cp*K*math.exp(-rf*T)*no

df = pd.read_csv('/Users/srdas/GoogleDrive/Papers/DeepLearning/DLinFinance
```

## Normalizing spot and call prices

$C$ is homogeneous degree one, so

$$aC(S,K) = C(aS,aK)$$

This means we can normalize spot and call prices and remove a variable by dividing by $K$.

$$\frac{C(S,K)}{K} = C(S/K,1)$$

## Data, libraries, activation functions

Slide Type: Fragment

```
n = 300000
n_train =  (int)(0.8 * n)
train = df[0:n_train]
X_train = train[['Stock Price', 'Maturity', 'Dividends', 'Volatility', 'R
y_train = train['Call Price'].values
test = df[n_train+1:n]
X_test = test[['Stock Price', 'Maturity', 'Dividends', 'Volatility', 'Ris
y_test = test['Call Price'].values
```

Slide Type: Fragment

```
#Import libraries
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, LeakyReLU
from keras import backend

def custom_activation(x):
    return backend.exp(x)
```

# Fit the Model

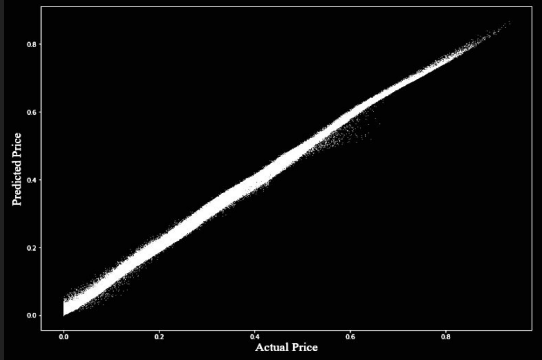
```
nodes = 120
model = Sequential()

model.add(Dense(nodes, input_dim=X_train.shape[1]))
model.add(LeakyReLU())
model.add(Dropout(0.25))

model.add(Dense(nodes, activation='elu'))
model.add(Dropout(0.25))

model.add(Dense(nodes, activation='relu'))
model.add(Dropout(0.25))

model.add(Dense(nodes, activation='elu'))
model.add(Dropout(0.25))

model.add(Dense(1))
model.add(Activation(custom_activation))

model.compile(loss='mse',optimizer='rmsprop')

model.fit(X_train, y_train, batch_size=64, epochs=10, validation_split=0.
```

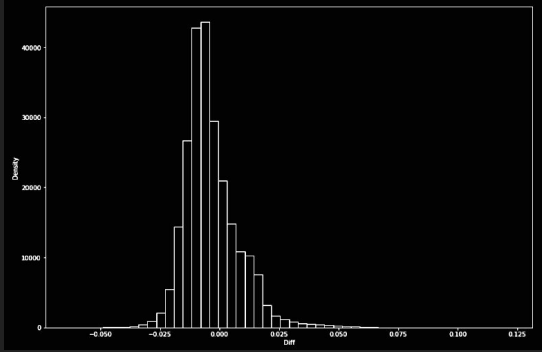
```
Train on 216000 samples, validate on 24000 samples
Epoch 1/10
22s - loss: 0.0051 - val_loss: 0.0013
Epoch 2/10
20s - loss: 0.0013 - val_loss: 1.3056e-04
Epoch 3/10
21s - loss: 0.0010 - val_loss: 7.0662e-04
Epoch 4/10
20s - loss: 8.4915e-04 - val_loss: 2.9489e-04
Epoch 5/10
21s - loss: 7.4240e-04 - val_loss: 6.5513e-04
Epoch 6/10
21s - loss: 6.7416e-04 - val_loss: 4.9628e-04
Epoch 7/10
21s - loss: 6.3666e-04 - val_loss: 3.7541e-04
Epoch 8/10
21s - loss: 5.9715e-04 - val_loss: 1.2603e-04
Epoch 9/10
26s - loss: 6.0532e-04 - val_loss: 1.3601e-04
Epoch 10/10
29s - loss: 6.1739e-04 - val_loss: 1.3957e-04

<keras.callbacks.History at 0x10ba4cef0>
```

## In-Sample

```
y_train_hat = model.predict(X_train)
#reduce dim (240000,1) -> (240000,) to match y_train's dim
y_train_hat = squeeze(y_train_hat)
CheckAccuracy(y_train, y_train_hat)
```

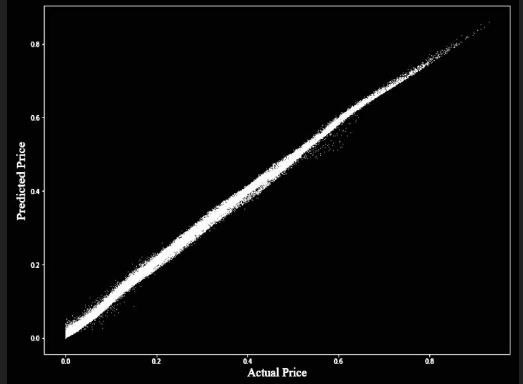
```
Mean Squared Error:       0.000141080530002
Root Mean Squared Error:  0.0118777325278
Mean Absolute Error:      0.00954217479875
Mean Percent Error:       0.0444015623019
```

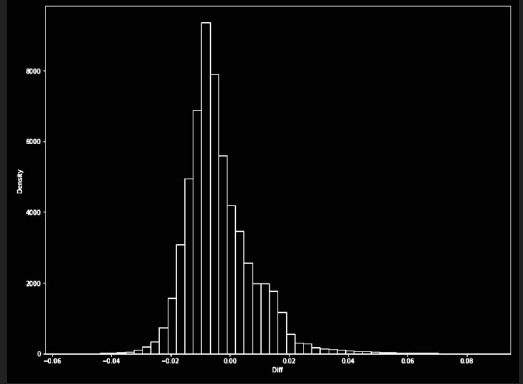


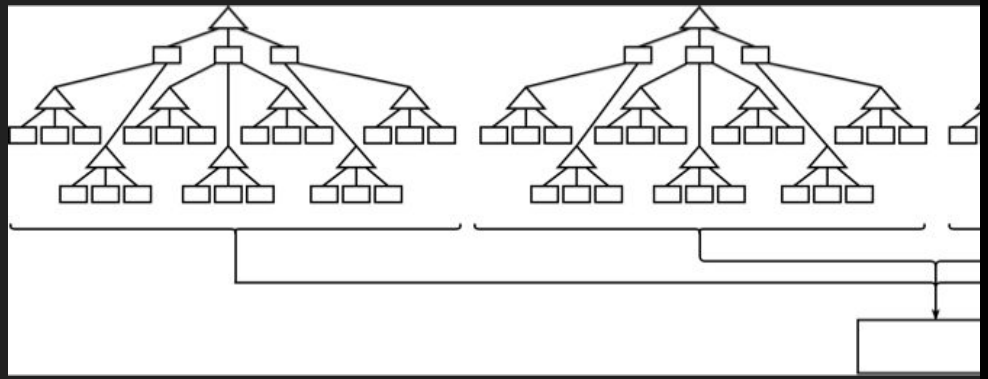

## Out-of-Sample

```
Mean Squared Error:       0.00014218354892
Root Mean Squared Error:  0.0119240743423
Mean Absolute Error:      0.00958741885268
Mean Percent Error:       0.0446387882149
```



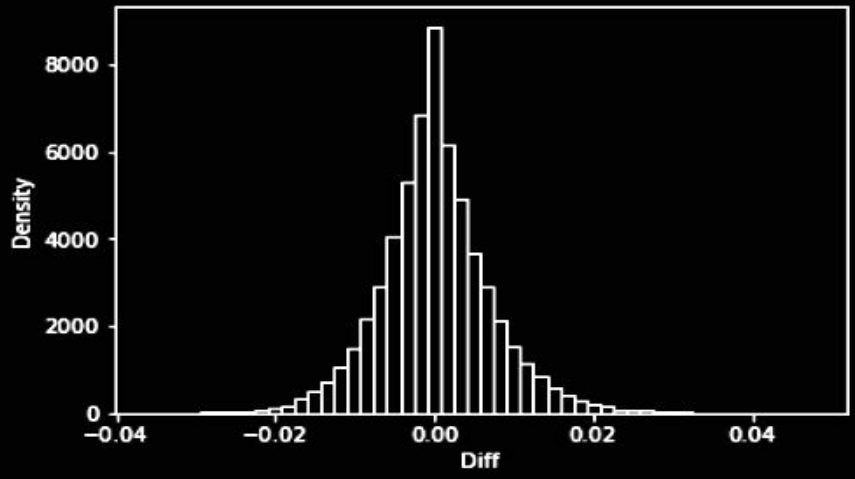
# Random Forest



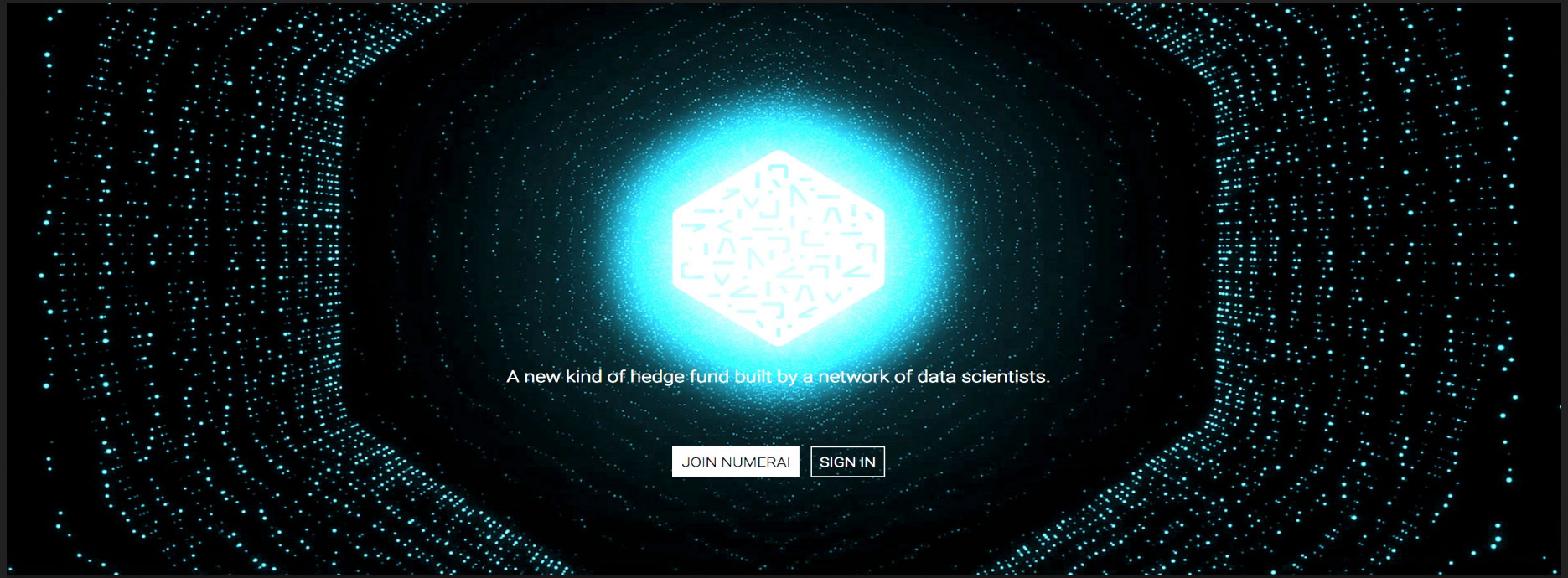

```
from sklearn.ensemble import RandomForestRegressor

forest = RandomForestRegressor()
forest = forest.fit(X_train, y_train)
y_test_hat = forest.predict(X_test)
```

```
stats = CheckAccuracy(y_test, y_test_hat)
```

```
Mean Squared Error:         4.97733713185e-05
Root Mean Squared Error:    0.00705502454415
Mean Absolute Error:        0.00516501148658
Mean Percent Error:         0.0264110854593
```



# Are Markets Still Efficient?




The stock market is inefficient with respect to new developments in machine learning — only a fraction of the world's data scientists have access to its data. Numerai is changing this.

# Predicting the Direction of the S&P500

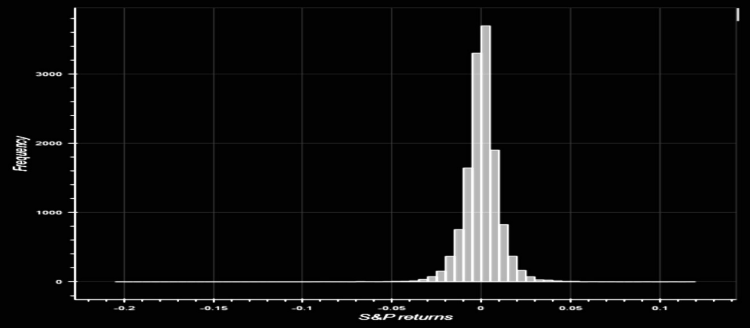
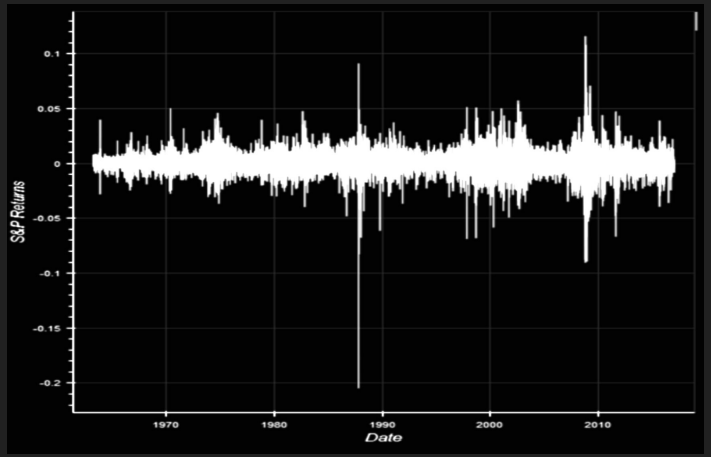



Figure 2: The distribution of daily S&P 500 index returns from 1963-2016. The mean return is 0.00031 and the standard deviation is 0.01. The skewness and kurtosis are −0.62 and 20.68, respectively.




Culkin, Das, Mokashi (2017)

## Data, lookback, lookforward

```
n = 30                  #Lookback period
nfwd = 30               #Look ahead forecast period
train_size = 10000      #Training size

m = shape(df)[0]
print(m) #Number of rows

df2 = df[n:m][df.columns[[0,1,21]]]
print(shape(df2))
df2.head()
```

```
13532
(13502, 3)
```

| | date | SPX | Sign |
|---|---|---|---|
| 30 | 1963-05-14 | -0.003831 | 0.0 |
| 31 | 1963-05-15 | 0.003133 | 1.0 |
| 32 | 1963-05-16 | -0.002556 | 0.0 |
| 33 | 1963-05-17 | 0.000569 | 1.0 |
| 34 | 1963-05-20 | -0.004695 | 0.0 |

# Organize data and fit the model

```
## Fit the model to the first 10,000 rows of data
X_train = df2[x:x+train_size].as_matrix()
X_train = X_train[:,3:len(df2.columns)]
print(shape(X_train))
Y_train = df2[x:x+train_size].as_matrix()
Y_train = Y_train[:,2:3].astype(int32)
print(shape(Y_train))

data_dim = shape(X_train)[1]
print(data_dim)

X_test = df2[x+train_size:(x+trai
X_test = X_test[:,3:len(df2.colum
print(shape(X_test))
Y_test = df2[x+train_size:(x+trai
Y_test = Y_test[:,2:3].astype(int
print(shape(Y_test))
```

```
(10000, 570)
(10000, 1)
570
(30, 570)
(30, 1)
```

```
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.layers.advanced_activations import LeakyReLU
from keras.utils import to_categorical

Y_train2 = to_categorical(Y_train,2)
```

## Model Accuracy

$$parameters = 570 \times 65 + 64 \times 65 + 64 \times 65 + 65 = 45435$$

```
model.evaluate(X_train, Y_train2, verbose=0)
```

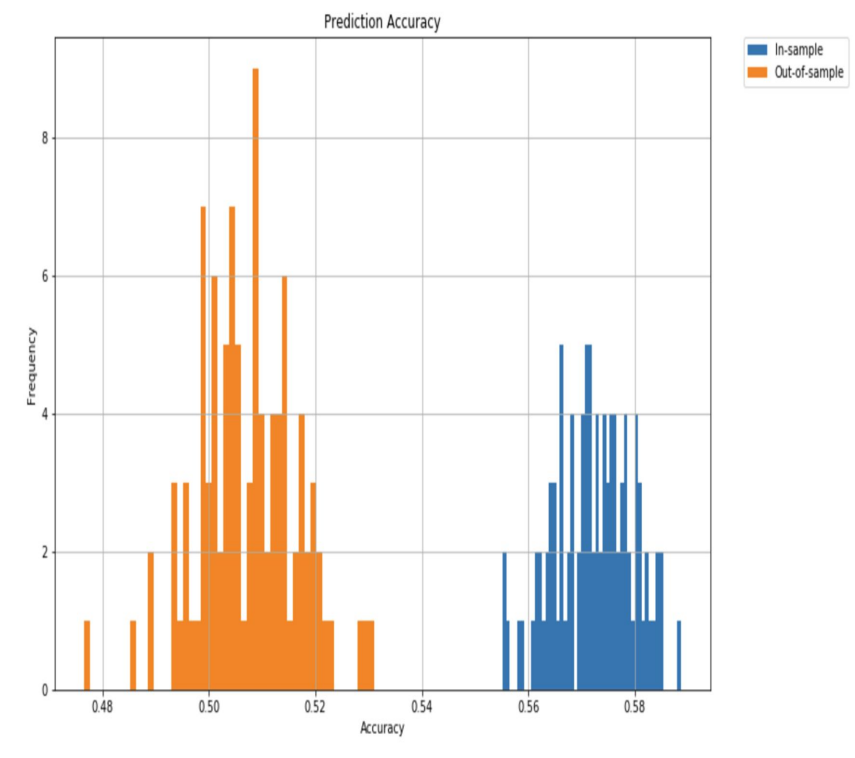
```
[0.68329057350158695, 0.55654999999999999]
```

```
model = Sequential()
n_units = 64 #200

model.add(Dense(n_units, input_dim=data_dim))
model.add(LeakyReLU())
model.add(Dropout(0.25))
```

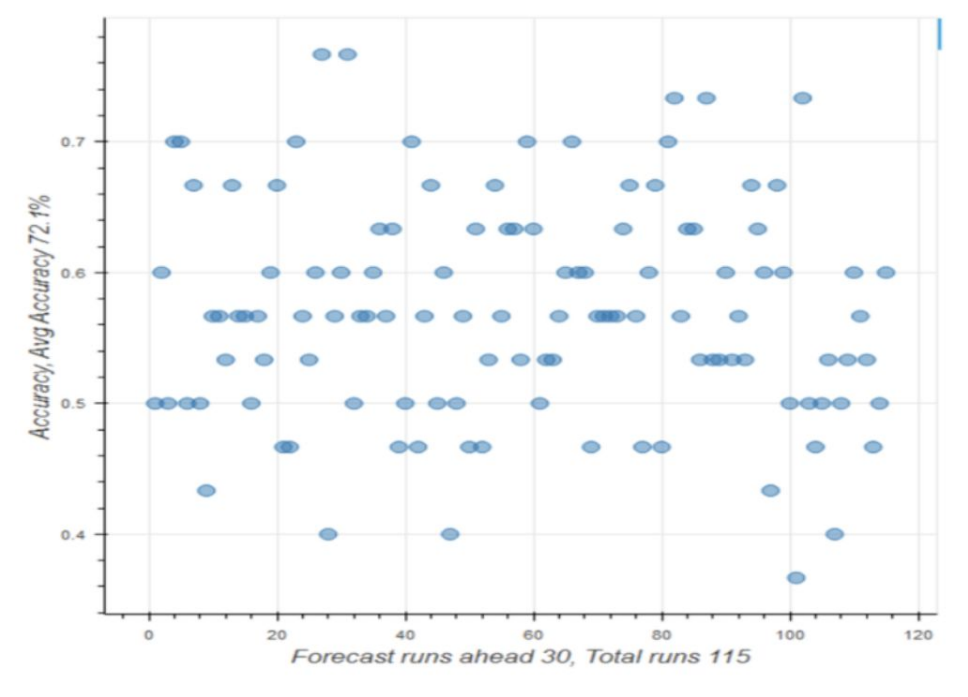
```
                metrics=['accuracy'])
bsize = 32
model.fit(X_train, Y_train2, batch_size=bsize, epochs=25, validation_
```

```
Train on 9000 samples, validate on 1000 samples
Epoch 1/25
2s - loss: 0.6939 - acc: 0.5124 - val_loss: 0.6924 - val_acc: 0.5180
Epoch 2/25
2s - loss: 0.6935 - acc: 0.5171 - val_loss: 0.6899 - val_acc: 0.5430
Epoch 3/25
```

# Predictions



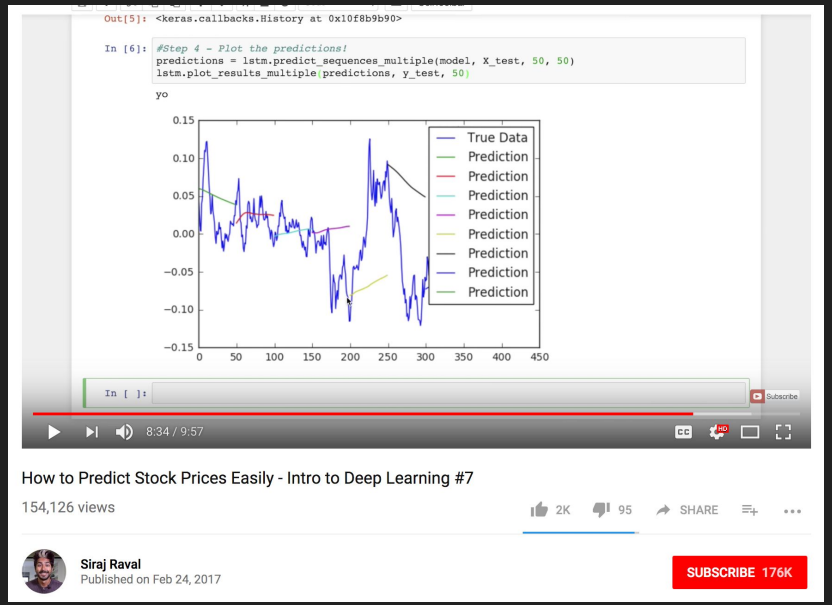

TensorFlow



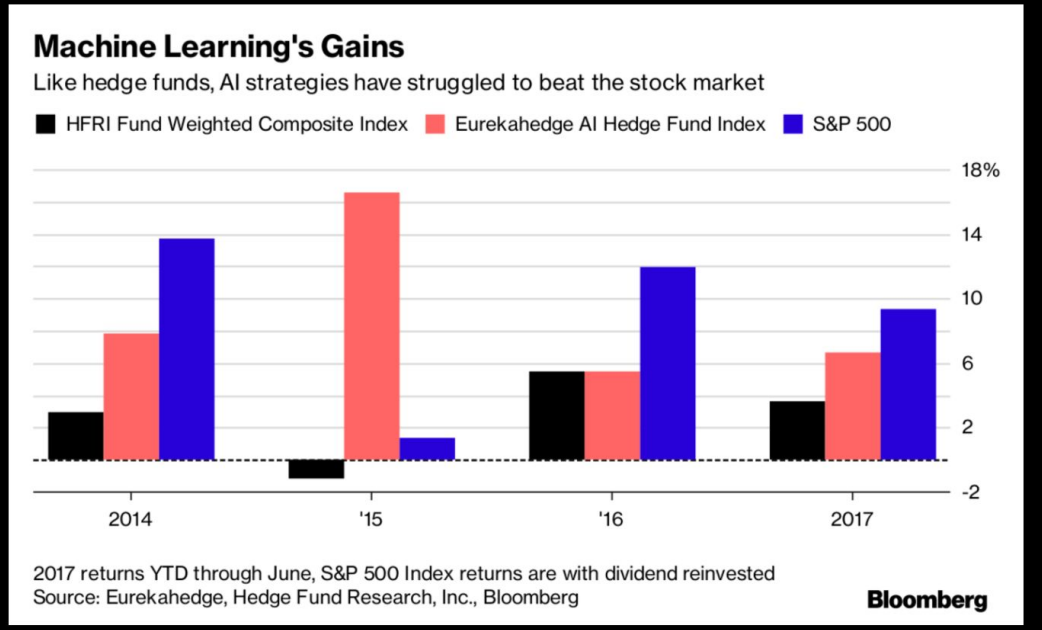

H20.deeplearning : lookback 30 days, forward predict 30 days, total experiments 115.

- More than 50% right 72.1% of the time
- Overall accuracy 54.4%

# Everyone's trying to deep learn market prediction




https://www.youtube.com/watch?v=ftMq5ps503w



# Two Curses of Predictive Analytics

## Non-stationarity

- Strong: Joint distribution of all variables (Y,X) remains the same over time.
- Weak: only mean and autocorrelation need to be same over time. If we are predicting the first moment, then this works.

## Randomness

- If noise swamps the signal, then we get poor predictions.

In short, stock-picking is hard:
https://www.wsj.com/articles/the-future-is-bumpy-high-tech-hedge-fund-hits-limits-of-robot-stock-picking-1513007557
Or is it?
https://www.bloomberg.com/news/articles/2017-12-11/hedge-fund-founder-and-amazon-researcher-build-a-better-ai-stock-picker

# Estimating the effects of deep learning

- (Roy) Amara's Law: "We tend to overestimate the effect of a technology in the short run and underestimate the effect in the long run."

- Arthur C. Clarke's Three Laws:
  a. When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.
  b. The only way of discovering the limits of the possible is to venture a little way past them into the impossible.
  c. Any sufficiently advanced technology is indistinguishable from magic.

# Results of Market Prediction Analysis

Add more slides here on using Python for market prediction analysis.

# AI as a Service

- Commoditized Platforms: Amazon's AWS Machine Learning VMs.
- Commoditized Services: e.g., Image Rekognition. Document analysis.
- Commoditized Models: Model zoos. NLP offerings.
- Encrypted Crowdsourcing of AI. e.g., Numerai. Managing data and privacy.
- Federated Machine Learning. Distributed Learning and data privacy.

# Federated Machine Learning

INNOVATION

## How Google wants to crowdsource machine learning with smartphones and Federated Learning

Google's new approach to machine learning relies on data stored on mobile devices to train models, without the need to centrally store the data.

By Conner Forrest | April 7, 2017, 11:22 AM PST





# Other applications of deep learning in finance

- Forecasting the VIX curve.
- Predicting Credit Card Default, see Khandani, Lee, Lo (2016).
- Portfolio optimization.
- Text analytics.
- Text generation (e.g., Narrative Science).

# What happens to financial sector employment?

**Elon Musk**
@elonmusk

Follow

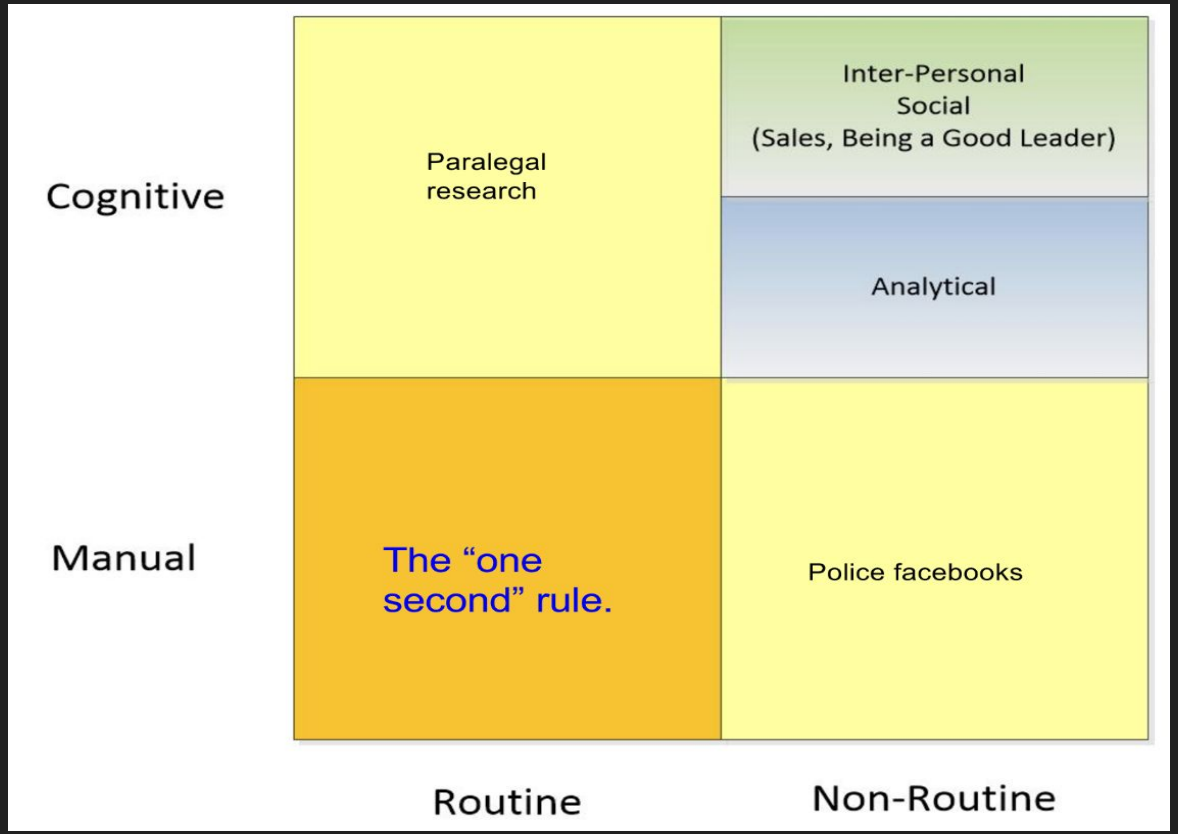
Replying to @elonmusk

China, Russia, soon all countries w strong computer science. Competition for AI superiority at national level most likely cause of WW3 imo.

2:33 AM - Sep 4, 2017

3,540 19,169 46,908

# The Atomic Level of Work




What tasks get automated first?

David Beyer

The End !!

## Readings on AI and Deep Learning

1. http://srdas.github.io/DLBook/
2. https://medium.com/@katherinebailey/hashtag-artificial-intelligence-47ff35e6a9cc
3. https://www.linkedin.com/pulse/ai-deep-learning-explained-simply-fabio-ciucci
4. https://hackernoon.com/why-ai-is-now-on-the-menu-at-dinner-even-with-my-non-tech-friends-44c666348de4
5. https://futurism.com/kurzweil-claims-that-the-singularity-will-happen-by-2045/
6. https://www.quantamagazine.org/artificial-intelligence-learns-to-learn-entirely-on-its-own-20171018/
7. https://www.technologyreview.com/s/603984/googles-ai-explosion-in-one-chart/?utm_campaign=add_this&utm_source=twitter&utm_medium=post
8. http://www.zerohedge.com/news/2017-02-13/goldman-had-600-cash-equity-traders-2000-it-now-has-2
9. https://futurism.com/an-ai-completed-360000-hours-of-finance-work-in-just-seconds/
10. https://www.theguardian.com/technology/2016/dec/22/bridgewater-associates-ai-artificial-intelligence-management
11. https://www.mizuhobank.com/mizuho_fintech/news/pepper/index.html
12. http://www.businessinsider.com/royal-bank-of-scotland-launches-ai-chatbot-luvo-using-ibm-watson-2016-9?r=UK&IR=T&IR=T
13. https://www.the-digital-insurer.com/dia/xtra-by-axa-ai-driven-personal-wellness-coaching-app/
14. http://www.nanalyze.com/2017/04/ai-fintech-startups-loans-new-credit/
15. https://www.americanbanker.com/news/this-is-how-financial-services-chatbots-are-going-to-evolve
16. http://fortune.com/2017/03/30/blackrock-robots-layoffs-artificial-intelligence-ai-hedge-fund/
17. https://en.wikipedia.org/wiki/Sentient_Technologies
18. https://medium.com/numerai/an-ai-hedge-fund-goes-live-on-ethereum-a80470c6b681
19. https://venturebeat.com/2017/05/06/ai-powered-trading-raises-new-questions/
20. http://www.wired.co.uk/article/how-ai-is-transforming-the-future-of-fintech
21. https://medium.com/numerai/encrypted-data-for-efficient-markets-fffbe9743ba8
22. https://www.bloomberg.com/news/features/2017-09-27/the-massive-hedge-fund-betting-on-ai
23. https://www.wired.com/2017/02/ai-threat-isnt-skynet-end-middle-class/?mbid=social_twitter_onsiteshare
24. https://www.wired.com/2008/06/pb-theory/
25. https://www.gartner.com/smarterwithgartner/gartner-predicts-our-digital-future/