ECEN 5053-002

Developing the Industrial Internet of Things

Week 10 - Lecture

Machine Learning

Dave Sluiter - Spring 2018

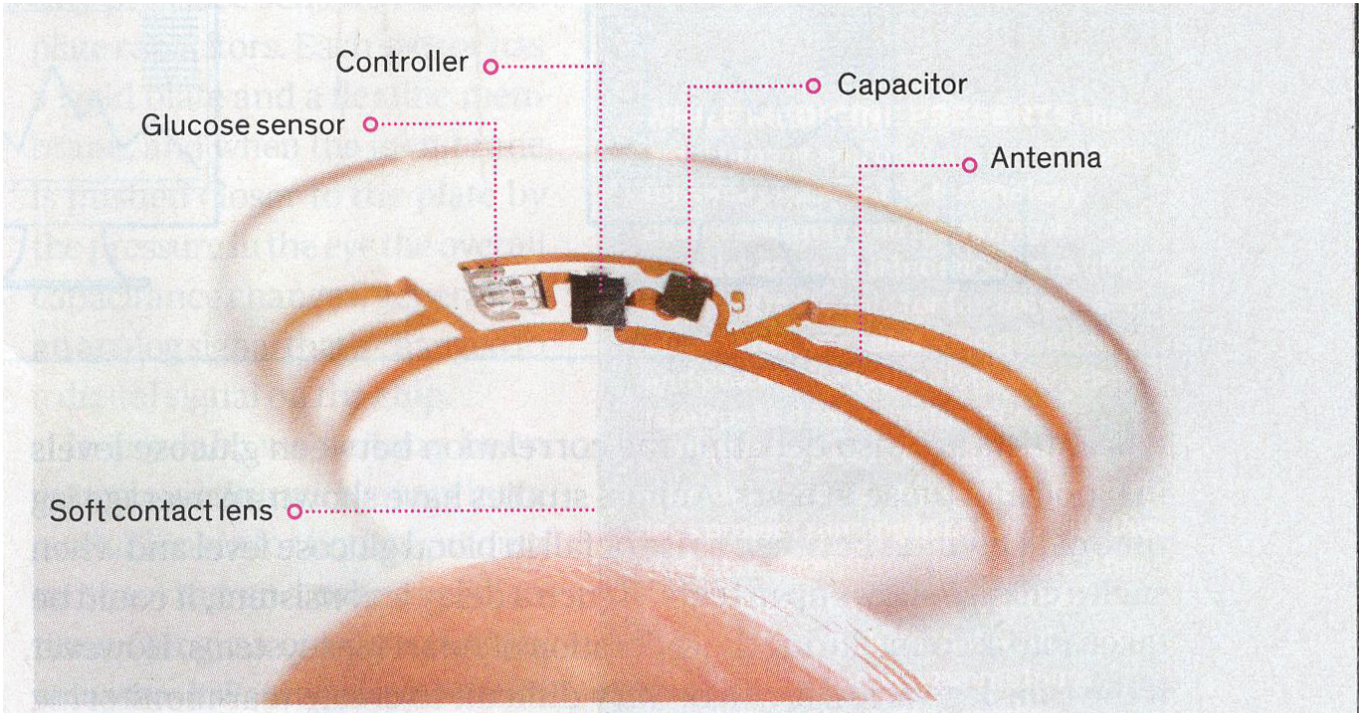
Material

- Machine Learning
  - History and background
  - Categories of learning algorithms
- Algorithms
  - Linear Regression
  - Naive Bayes
  - Support Vector Machines (SVM)
  - K-Means

Learning Outcomes

- Become aware of the variety of machine learning algorithms
- Understand the types of problems that machine learning algorithms can solve today
- Articulate how these algorithms are fundamentally different from traditional programming algorithms
- Understand the role that *humans* play in crafting effective machine learning solutions
- Understand the role that *machine learning* algorithms can play in the IIoT space

But first, a cool embedded system



Verily-Alcon smart-lens project. This sensor uses an enzyme that catalyzes glucose to create hydrogen peroxide, which is further oxidized at the electrode to release electrons, thus generating an electrical current proportional to the glucose concentration.

TELLTALE TEARS: This diabetes-monitoring contact lens, still under development by researchers at Google and Novartis, measures glucose levels in the wearer's eye fluids.

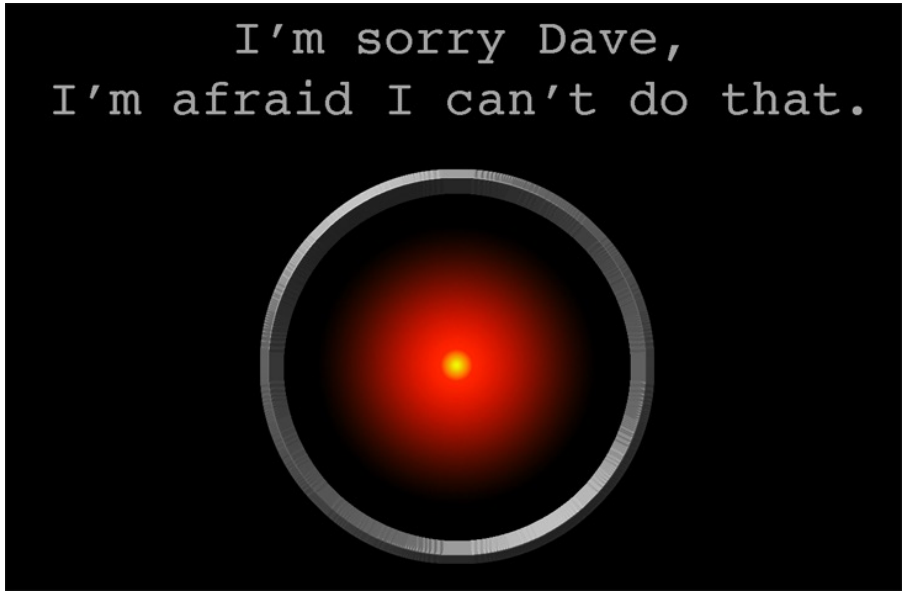
Source: IEEE Spectrum August 2017

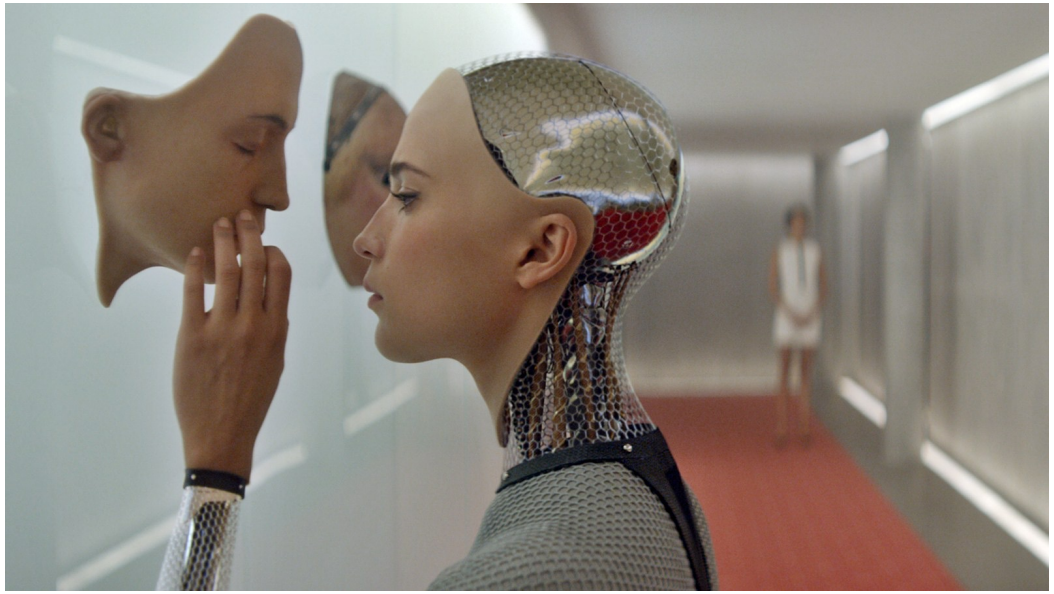
AI Backgrounder

- AI is built on the hypothesis that *mechanizing thought* is possible
- First century: Greek, Indian and Chinese philosophers all worked on ways to perform this task
- 17th century: Gottfried Leibniz, Thomas Hobbes and Rene Descartes discussed the potential for *rationalizing all thought* as math symbols
- 1950: Alan Turing published "*Computing Machinery and Intelligence*"
  - This paper led to the **Imitation Game**: Player A is a computer, player B is a human. Player C is human, and can't see players A and B. Players A and B must convince player C it is human. If Player C cannot tell the difference, the computer wins.

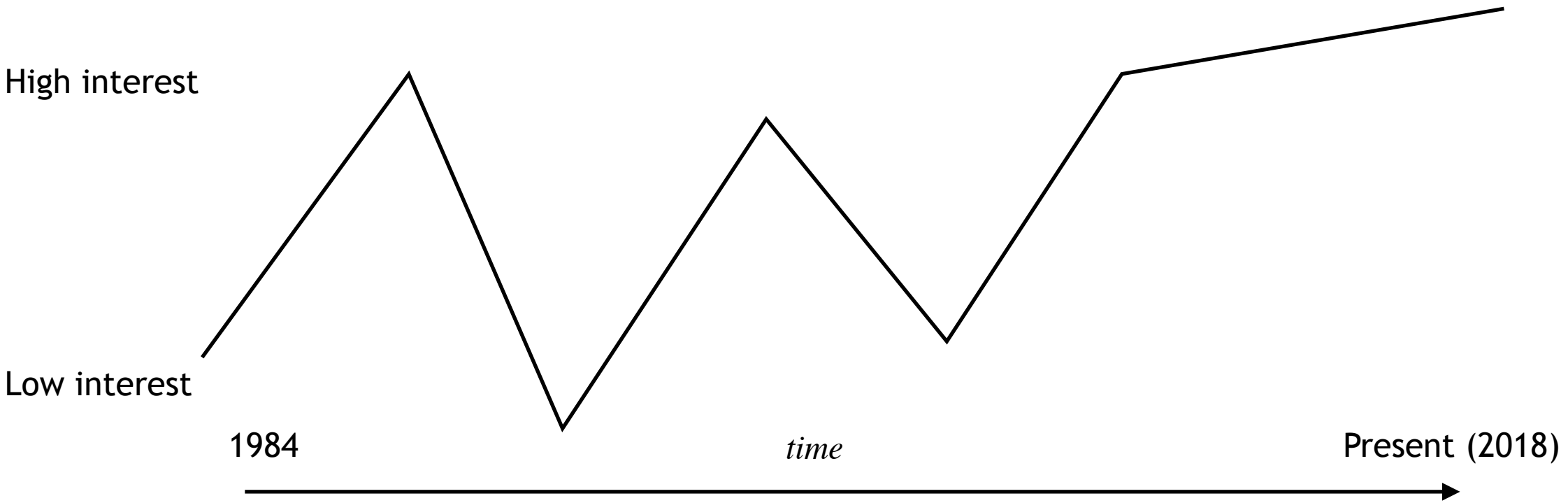
AI Backgrounder

- Machine learning relies on algorithms to analyze huge data sets
- These algorithms can perform predictive analytics far faster than any human can
- Too much hype in films and TV shows, setting the public's expectations too high





Rise and Fall of AI



3 contributing factors:
- Improved algorithms
- Fast computing & GPUs
- Large data sets

AI Backgrounder

- People confuse a machine getting better at its job (learning) with awareness (intelligence/consciousness) - *not true*.
- Machine learning provides the "*learning*" part of AI, and is primitive in comparison to what we see in films.
- A true AI may eventually emerge in the future when computers can emulate the clever combination used by nature:
  - Genetics: Slow learning from one generation to the next
  - Teaching: Fast learning from organized sources
  - Exploration: Spontaneous learning from media and interactions with people

Machine Learning, What is it?

- 1959 - Arthur Samuel: "Field of study that gives computers the ability to learn without being explicitly programmed"
- **Technical proposition**: Machines are better than humans at accurately and consistently:
  - Capturing data
  - Communicating data
  - Analyzing data (identifying structure/trends)

- AI encompasses many disciplines
  - Natural language processing and understanding
  - Image/pattern recognition
  - Learning
  - Knowledge representation
  - Reasoning
  - Planning
  - Perception
  - Social intelligence
  - Creativity

Shallow Autoencoder is a neutral network that finds structure in data

MLP = multilayer perception, a neural network

Source: “Deep Learning”, Ian Goodfellow

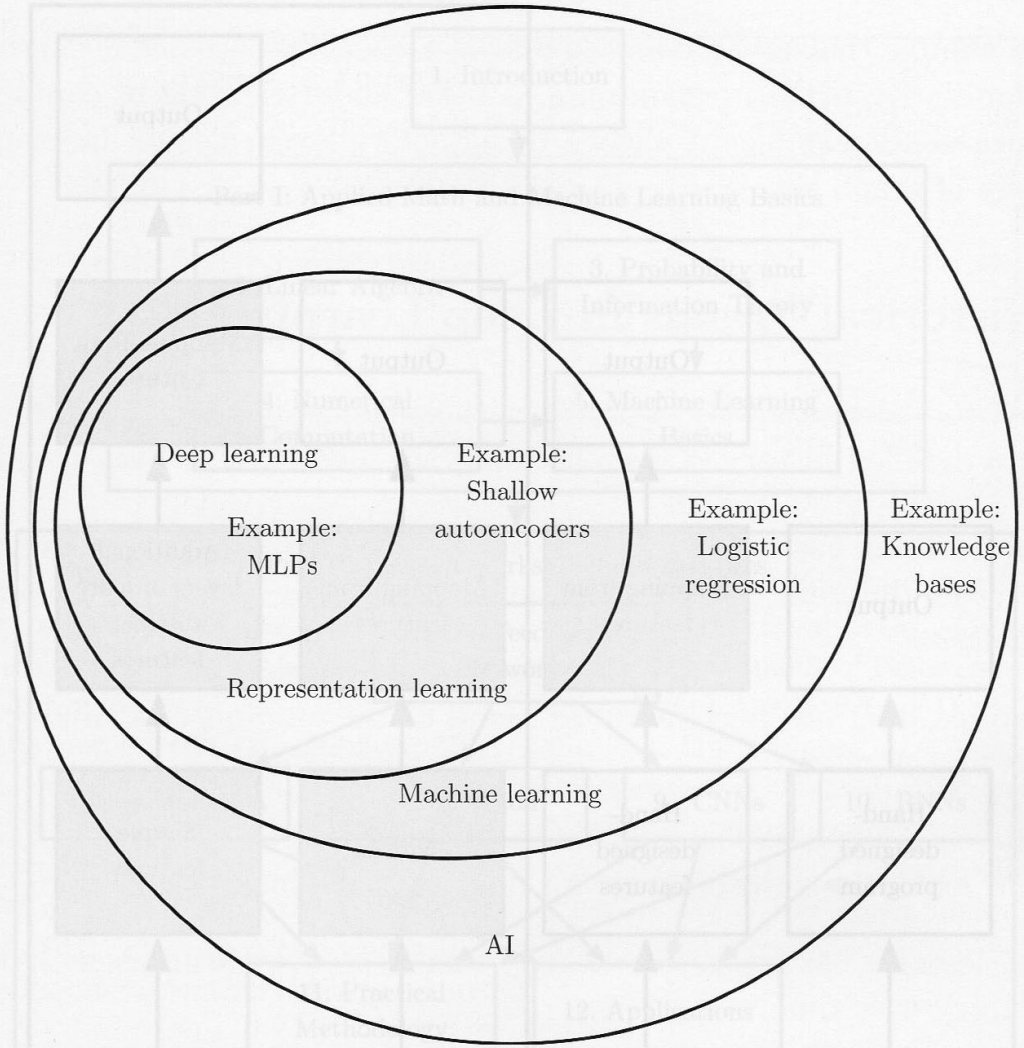


Figure 1.4: A Venn diagram showing how deep learning is a kind of representation learning, which is in turn a kind of machine learning, which is used for many but not all approaches to AI. Each section of the Venn diagram includes an example of an AI technology.

Examples

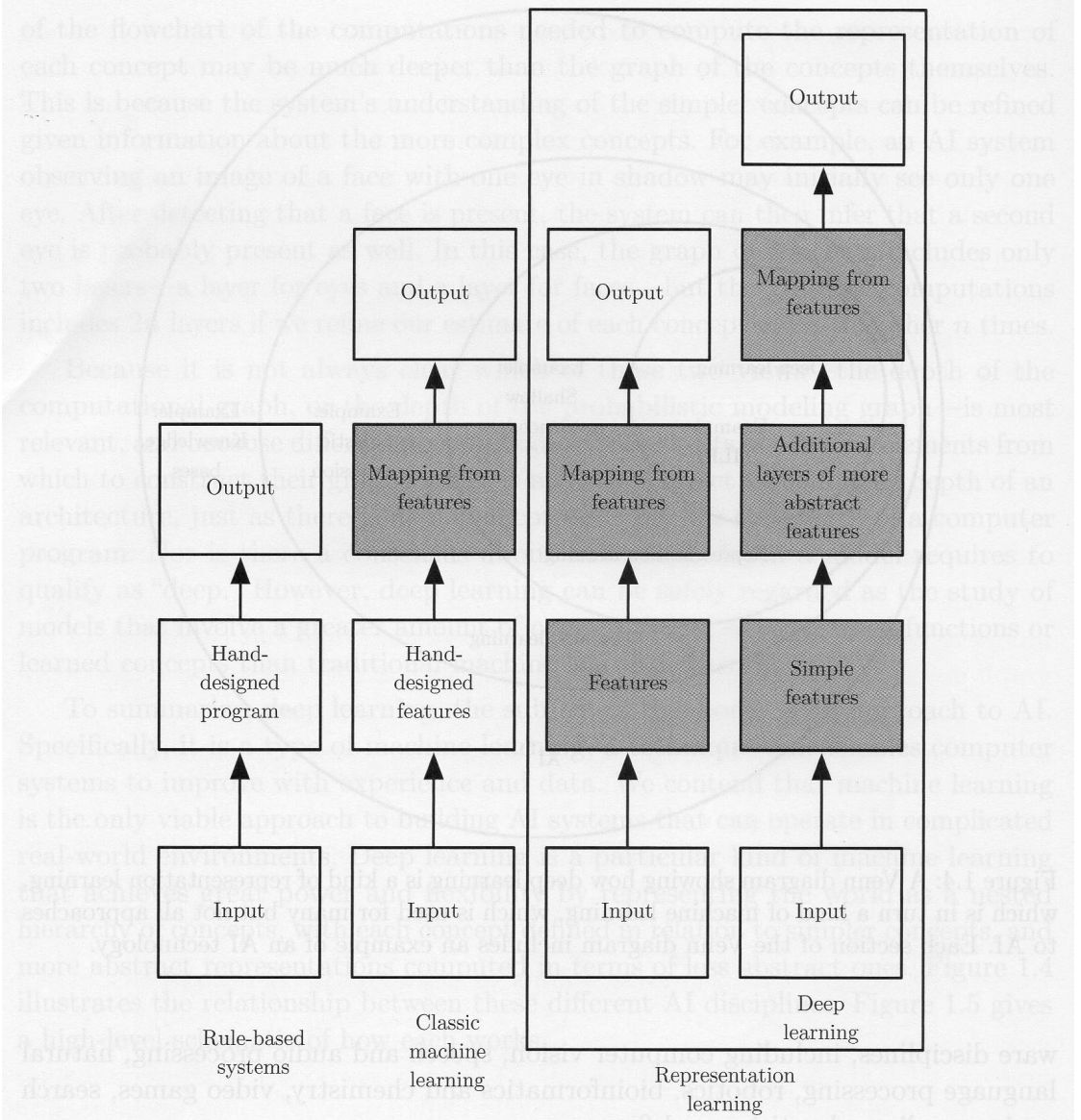


Figure 1.5: Flowcharts showing how the different parts of an AI system relate to each other within different AI disciplines. Shaded boxes indicate components that are able to learn from data.

Source: "Deep Learning", Ian Goodfellow

Examples of Machine Learning we encounter every day

- Postal service, ZIP code reader
- Bank check deposits, reading the numerical amount
- Credit card fraud detection
- Amazon, eBay and Netflix recommendations

A good point to remember

- Failure is more common than success when working through a **machine learning** experience. Your intuition adds the "*art*" to the machine learning experience, but sometimes intuition is wrong and you have to revisit your assumptions.

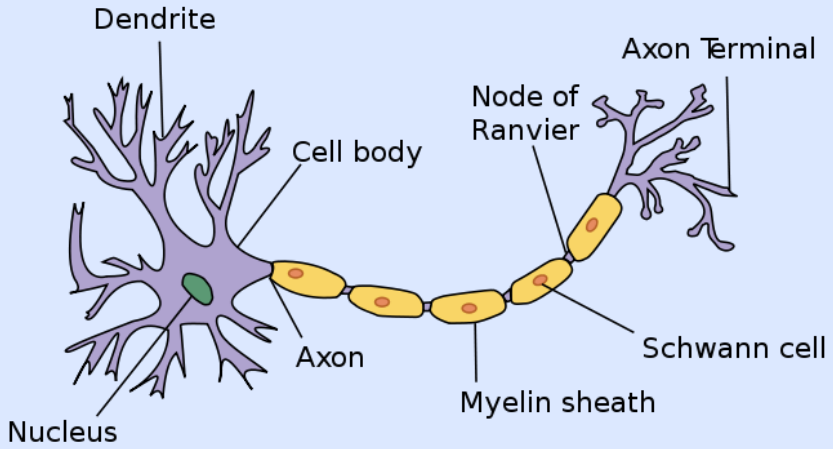
Source: "Machine Learning for Dummies", John Paul Mueller

Machine Learning Schools of Thought

- **The 5 Tribes**
  - Symbolists
    - The origin of this tribe is in logic and philosophy
  - Connectionists
    - The origin of this tribe is in neuroscience
  - Evolutionaries
    - The origin of this tribe is in evolutionary biology
  - **Bayesians**
    - The origin of this tribe is in statistics. **We will focus on this.**
  - Analogizers
    - The origin of this tribe is in psychology

Source: “Machine Learning for Dummies”, John Paul Mueller

Connectionists



- My personal opinion of where the breakthroughs will eventually occur
  - Neuromorphic[1] computing
  - Based on neurons and brain function
  - Our neocortex has about 30 billion neurons
    - Each neuron receives connections from 5,000 to 10,000 other neurons (synaptic connections)
    - Making more than 100 trillion synaptic connections

1 - Concept developed by Carver Mead

Image source: https://simple.wikipedia.org/wiki/Neuron

Source: IEEE Spectrum, June 2017, article by Jeff Hawkins

Connectionists (con't)

- Largely missing from today's AI
  - **Rewiring** - we learn quickly, a fews touches or glances, and **incrementally**: Adds to what we know already know
  - **Sparse distributed representations (SDR)**
    - A brain uses thousands of "bits" to represent a piece of information
    - Two properties
      - **Overlap property**: To distinguish between how things are similar or different
      - **Union property**: To represent multiple ideas simultaneously
      - Example: One SDR for "cat" and another for "bird". They share an SDR for "pet" and "clawed", but would not share the SDR for "furry"
      - SDR's are inherently fault tolerant
  - **Embodiment**: Use of movement to learn about the world. Sensorimotor integration occurs in every part of the neocortex

Source: IEEE Spectrum, June 2017, article by Jeff Hawkins

Connectionists (con't)

- The hunt is "on" to determine the neuron's "**master algorithm**"
- Power efficiency:
  - Data centers in the US draw megawatts, ~2% of electricity is consumed by US data centers
  - Human brain runs quite well on ~20 watts, a fraction of the food a persons eats each day
- 2000 kilocalories = ~97 watts, so ~1/5 of our energy consumption

Source: IEEE Spectrum, June 2017

Get the Tools

- We will use **python** for our examples
  - **Action**: Download Anaconda for python 2.7
- https://www.continuum.io/downloads
- Installs a high-performance version of python
- 720 library packages
- Demo Anaconda
  - Jupyter notebook
  - Spyder
- R is an alternative to python. Used extensively for machine learning.

Categories of Machine Learning

- **Supervised Learning**
  - You have example data, outputs are known.
  - **Regression** problems, **continuous value** problems: predicting housing prices
  - **Classification problems**, **discrete outputs**, what type of iris (flower) is this? Does the patient have cancer?
- **Unsupervised Learning**
  - Outputs are unknown
  - Market segmentation
  - Social network analysis
  - Cocktail party problem - filter out all the noise and listen to one conversation. Find the underlying structure in the data
- **Reinforcement Learning**
  - Make a sequence of decisions over time, to maximize a cumulative reward

Categories of Machine Learning

- Researchers are looking into algorithms generating algorithms

Supervised Learning

- Generalize steps
  - Load a data set
  - Choose an algorithm
  - Train the model
  - Visualize the model
  - Test the model
  - Evaluate the model

Supervised Learning

- **Linear Regression**
- We want to predict housing prices based on:
  - Square footage
  - Number of bedrooms

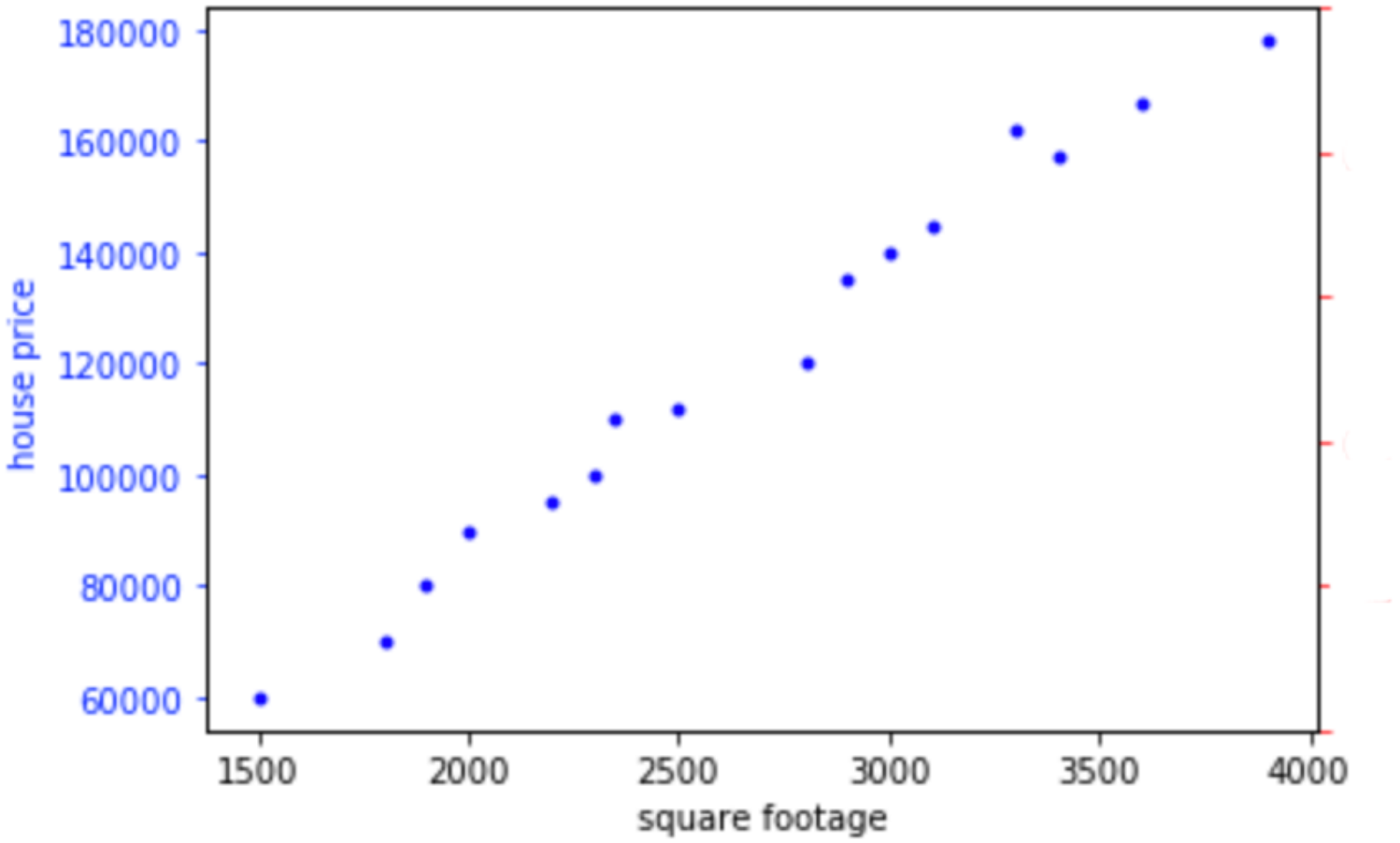
Linear Regression

- We have some example data
  - Training data
- Each row is an example from real estate sales data
- Contains a number of features, x_n
  - x_1 is square footage
  - x_2 is number of bedrooms
- And output y, the price
  - Is what we are trying to predict, also known as the **target value**

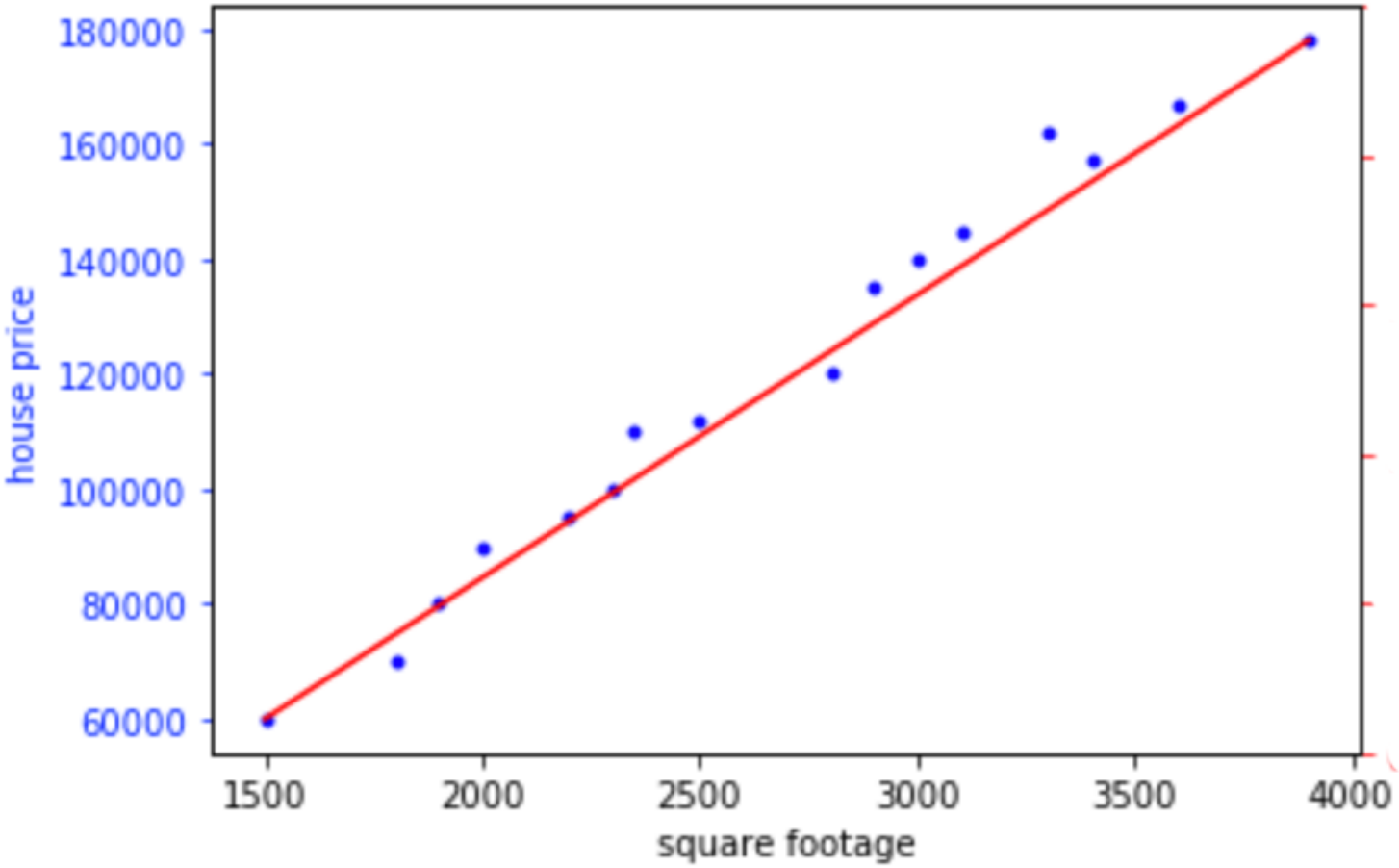
```
train_data = np.array(
    [
    # sqft, #bedrooms, price
    [1500, 2,  60000],
    [1800, 2,  70000],
    [1900, 2,  80000],
    [2000, 3,  90000],
    [2200, 3,  95000],
    [2300, 2, 100000],
    [2350, 3, 110000],
    [2500, 3, 112000],
    [2800, 4, 120000],
    [2900, 3, 135000],
    [3000, 4, 140000],
    [3100, 4, 145000],
    [3300, 5, 162000],
    [3400, 4, 157000],
    [3600, 5, 167000],
    [3900, 5, 178000]
    ]
) # end training data
```

Training Data

of bedrooms is not plotted



We can manually draw a best-fit curve



Linear Regression

- We want to create a hypothesis function $h()$ that will make predictions
- To perform supervised learning, we have to decide how we will represent the hypothesis function $h()$
- As an initial choice, let's say we decide to approximate $h()$ as a linear combination of features x and weights θ, so we have:
  - $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2$
  - where we set $x_0 = 1$, and θ_0 becomes the intercept term
    - like in the equation for a line $y = ax + b$, b is the intercept term

Linear Regression

- m = number of training examples, 16
- n = number of features, 2
- We can rewrite $h_\theta(x)$ as :
  - $h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \dots \\ \theta n \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

$$\theta^T x = [\theta_0, \theta_1, \theta_2 \,..\, \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \mathbb{R} = h_\theta(x)$$

a real number

Linear Regression

- How do we pick/calculate/learn the values for the θ_i's?
  - As a starting point, we can make $h() \sim= y$
- We can define a **cost function**:
  - $J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$
  - The superscript i's refer to training examples, not raise to a power!
- We want to choose θ's to minimize $J(\theta)$
- To do so, let's use a search algorithm that starts with some initial guess for θ, and repeatedly changes θ to make $J(\theta)$ smaller, until we converge at a minimum $J(\theta)$ value.

Linear Regression

- **Search algorithm = Gradient Descent**
  - Start with some initial guess at θ
  - and then repeatedly perform the update:
  - $\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta), j = 0 \ldots n$
    - this rule is called the LMS update rule (least mean squares rule)
    - α (alpha) is the learning rate (hyper parameter)
    - Great care needs to be taken choosing alpha

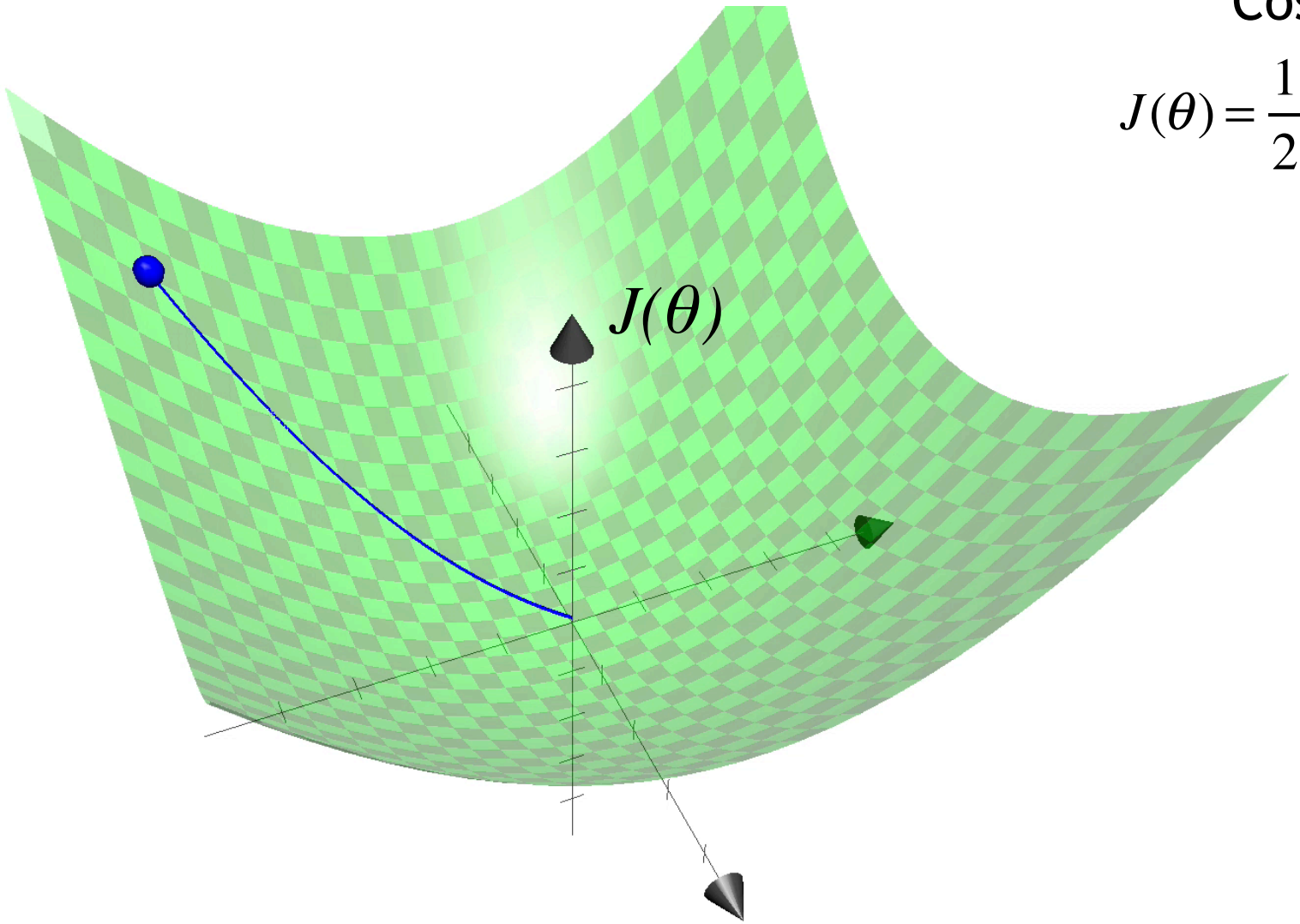
Linear Regression

- **Gradient Descent**
  - Calculating the derivative, $\frac{\partial}{\partial \theta_j} J(\theta)$ and for all n, we get:
  - *repeat until convergence {*

$$\theta_j = \theta_j + \alpha \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)}))x_j^i, j = 0 \ldots n$$

  - *}*
    - for each θ_j cycle through all m training examples
  - This method is called **batch gradient descent**
  - This $J(\theta)$ equation is a convex quadratic function, and has only 1 global minimum. Beware of functions that may have local minima, because gradient descent could get "stuck" in a local minima.

Gradient Descent



Cost function:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Linear Regression

- **Gradient Descent**
  - Test for convergence is a **user defined function**
  - Downside to **batch gradient descent** is that every θ update needs to run through all m training samples
    - Computationally expensive for large training sets

Linear Regression

- Second method
  - **Stochastic gradient descent**
  - For each training example m we update the θ_i's
  - Faster for large training sets, but potentially less accurate
- *for i=1 to m {*

$$\theta_j = \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^i, j = 0...n$$

- *}*

stochastic: randomly determined; having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely.

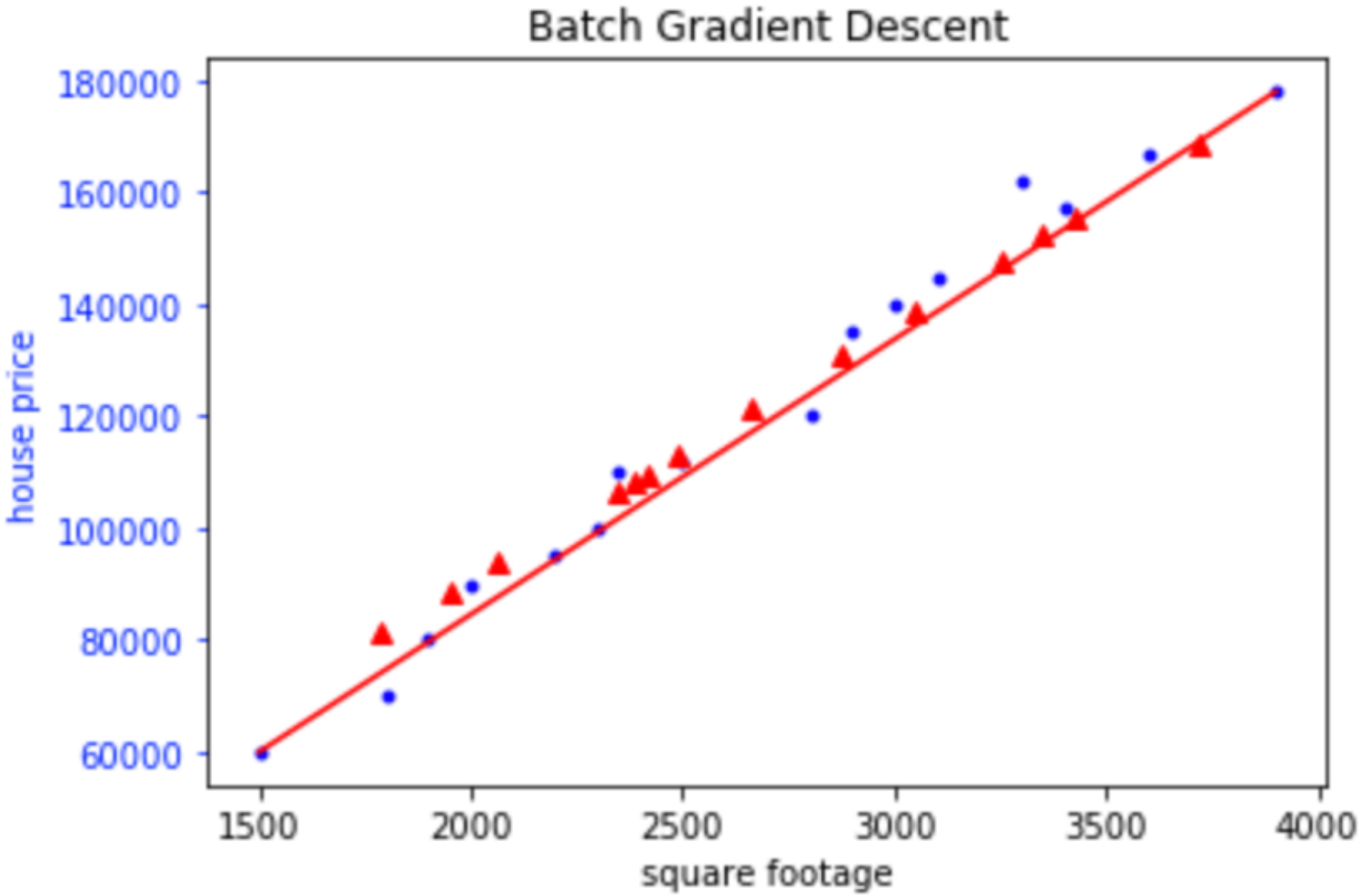
Linear Regression

- Third method
- Is it possible to directly calculate the $\theta j's$?
  - Turns out the answer is yes
  - **Closed form equation**, no iterations
  - In this method we minimize $J(\theta)$ by explicitly taking the derivatives of $\theta j's$ and setting them to 0.
  - After some crazy linear algebra, we get:

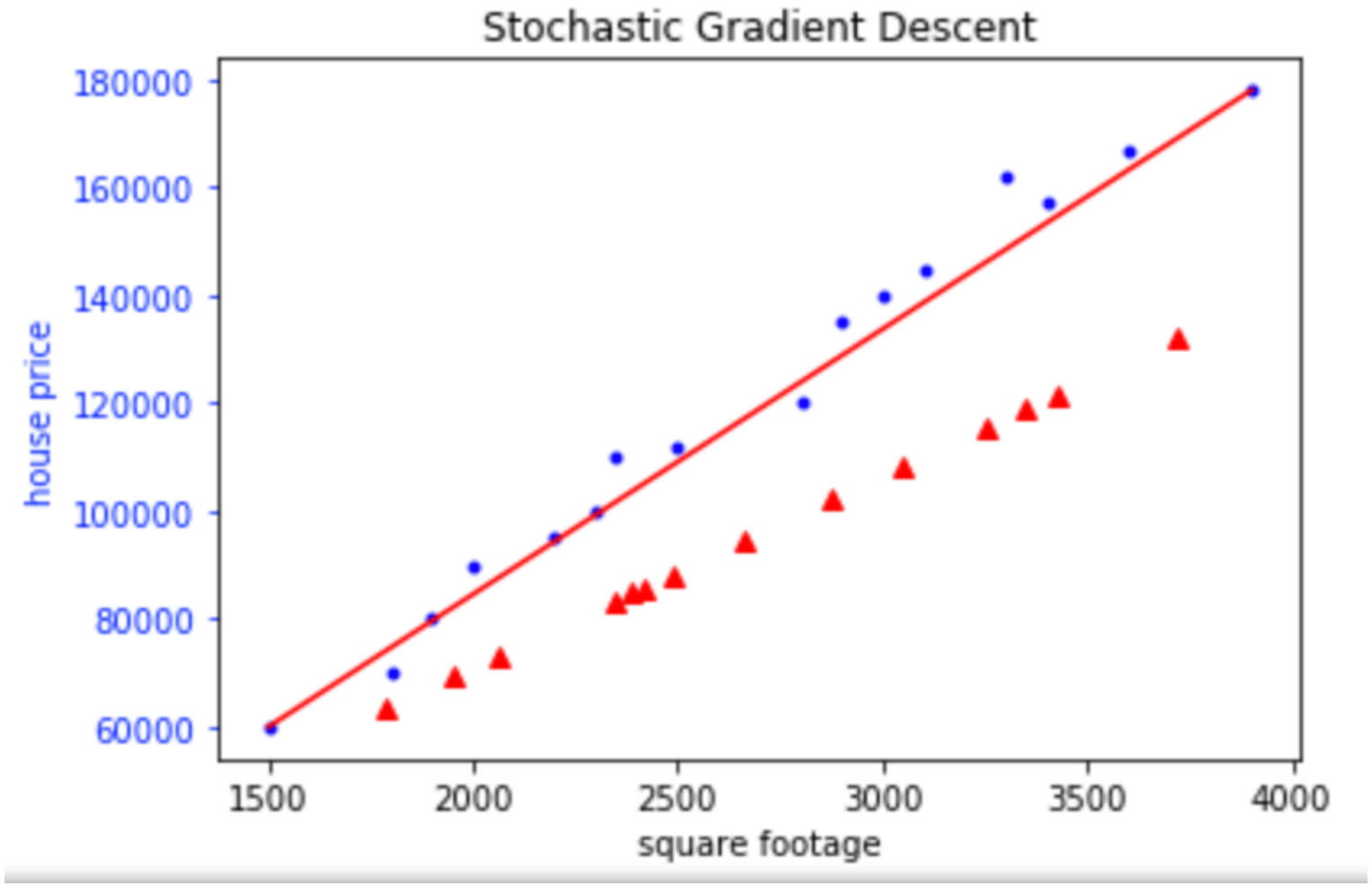
$$\theta = (X^T X)^{-1} X^T \vec{y}$$

Linear Regression Results

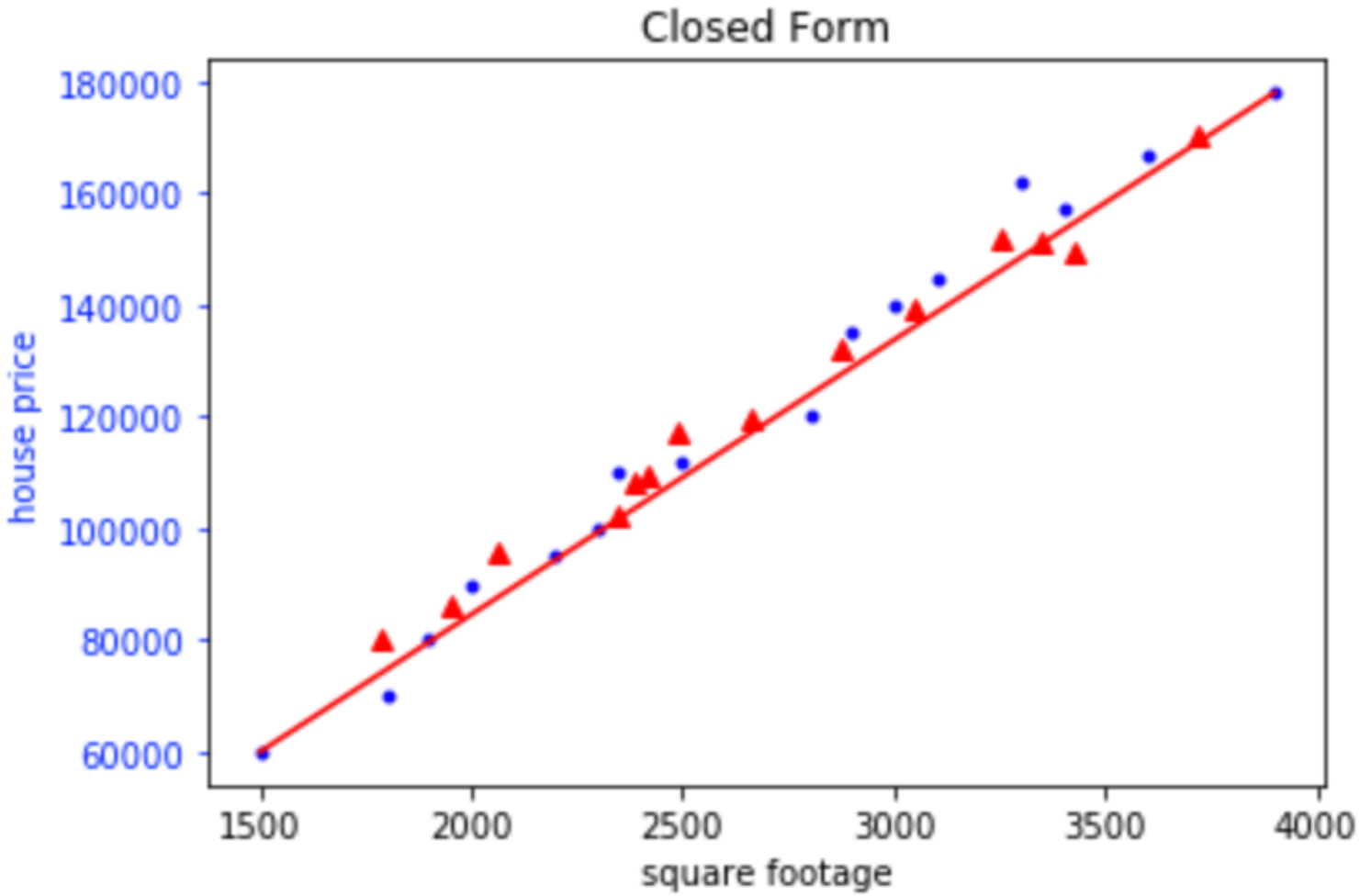
Example Batch Gradient Descent



Example Stochastic Gradient Descent



Example Closed Form



A Look at the θj's

```
('>>> iteration count=', 4)
-----------------------------------------------------
('Theta vector=', array([  1.01536624,  45.91706257,   1.05785631]))
Theta's converged
('Batch trained theta=', array([  1.0150378 ,  45.45597599,   1.05740841]))
('Error function called', 192, 'Cost function called', 12)

('>>> iteration count=', 9)
-----------------------------------------------------
('Theta vector=', array([  1.0083882 ,  33.71396924,   1.04194099]))
Theta's converged
('Stochastic trained theta=', array([  1.00885329,  35.52783201,   1.04426645]))
('Error function called', 432, 'Cost function called', 432)


('Closed form theta=', array([  1.00000000e+00,   3.95580633e+01,   4.67540178e+03]))
('Error function called', 0, 'Cost function called', 0)
```

Choosing Alpha

- Choosing alpha to small and we don't make significant progress towards convergence at the global minima
- Choosing alpha too big causes the iterations to over/ under-shoot, possibly running away to very large/small values. This happened to me! ☹️

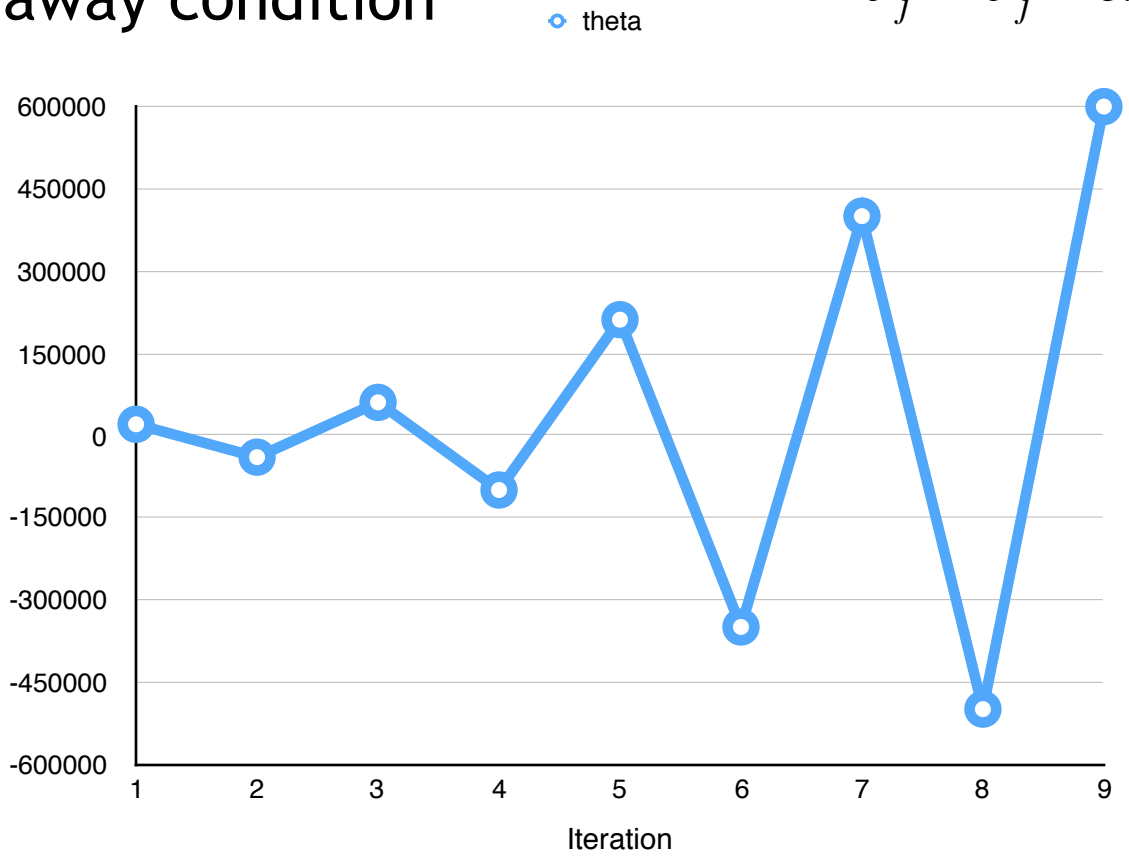
Detecting Convergence

```
THETA_OUT_OF_RANGE_VALUE = 5.0e+20
THETA_DELTA_THRESHOLD    = 2
 ALPHA                   = 0.00000001 # learning rate

keep_going = 0
theta_out_of_range = 0
for k in range(n) :
   if (abs(new_theta[k] - theta[k]) > THETA_DELTA_THRESHOLD) :
       keep_going += 1 # we found 1 theta delta > than our threshold, so keep going
       if (abs(new_theta[k]) > THETA_OUT_OF_RANGE_VALUE) :
           print ("-----------------------------------------")
           print ("Error: Theta out-of-range=", new_theta[k])
           print ("-----------------------------------------")
           theta_out_of_range = 1

theta[:] = new_theta[:] # update the theta values
```

Detecting Convergence

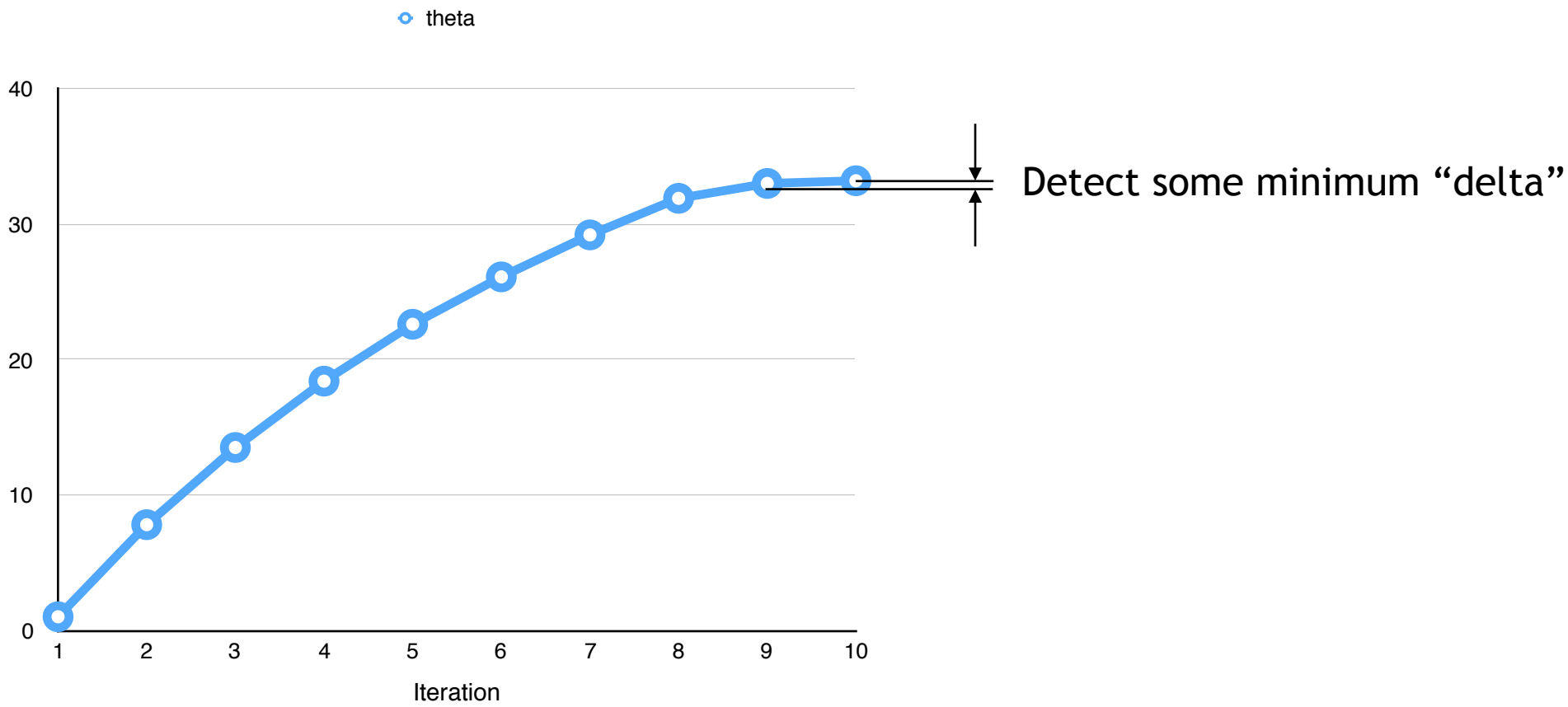
Alpha too large, run-away condition

$$\theta_j = \theta_j + \alpha \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)}))x_j^i, j = 0...n$$



Detecting Convergence

A good Alpha

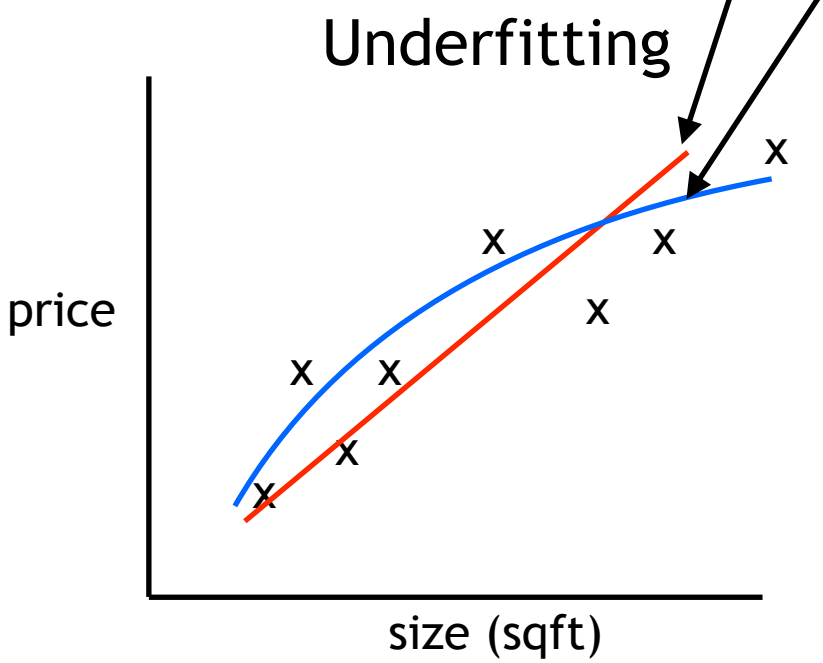


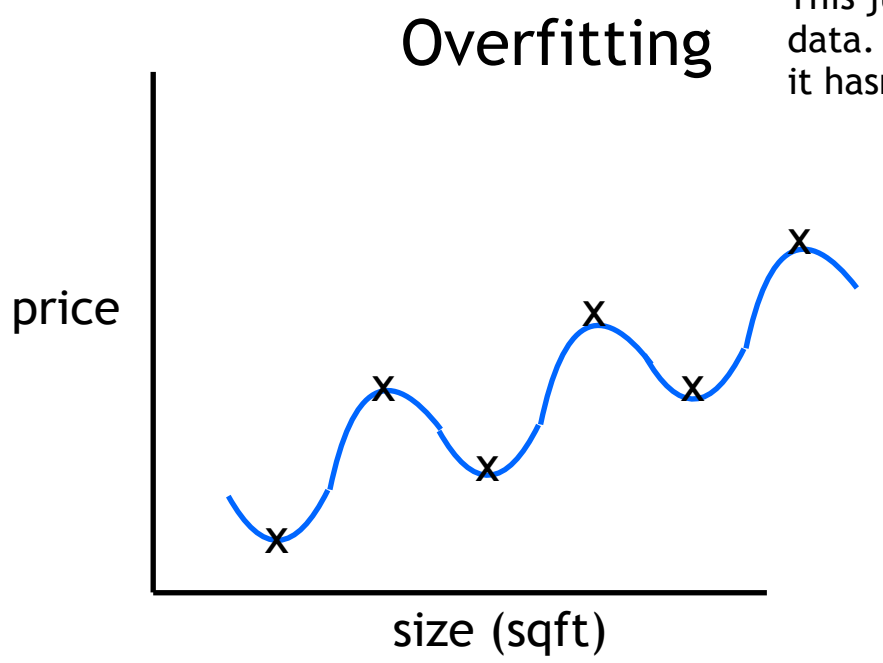
Problems That Can Occur

- Underfitting
- Overfitting

linear: x_i = sqft

exponential: $x_i = k(1 - e^{-sqft})$





This just memorizes the data. Performs poorly for data it hasn't seen yet

A look at the code

- See code example, jupyter notebook file
  - CS229 - Housing Prices - Logistic Regression.ipynb

Supervised Learning

- **Naive Bayes**
- We want to detect spam emails

but first, Bayes theorem…

Bayes Theorem

- Named after Rev. Thomas Bayes (1701-1761)
- An equation that allows new evidence to update beliefs:

$$P(A \mid B) = \frac{P(B \mid A)}{P(B)} P(A)$$

A and B are events
P(B) != 0
P(A) = Probability of A
P(B) = Probability of B
P(B|A) = Probability of B given A
P(A|B) = Probability of A given B

Bayes Theorem

- In this scheme, probability measures a "*degree of belief*"
- The theorem links the degree of belief in a proposition *before and after* accounting for evidence
- For proposition A and evidence B,
  - P(A) - the initial degree of belief in A, *prior*
  - P(A|B) - the degree of belief in A having accounted for B, *posterior*
  - $\frac{P(B \mid A)}{P(B)}$ - represents the support B provides for A

Bayes Theorem Example

- The entire output of a factory is produced by 3 machines, accounting for *20%*, *30%* and *50%* of the output respectively
- % of defective items produced is: *5%*, *3%* and *1%* for each machine respectively
- An output item is chosen at random and is **found to be defective**. *What is the probability that it was produced on the 3rd machine*?

Source: https://en.wikipedia.org/wiki/Bayes%27_theorem

Bayes Theorem Example

- Let A_i denote that a randomly chosen item was made on the ith machine ($i=1,2,3$). Let B denote that a randomly chosen item was defective.
  - $P(A_1)$=0.2, $P(A_2)$=0.3, $P(A_3)$=0.5 (*prior proposition*)
  - $P(B|A_1)$=0.05, $P(B|A_2)$=0.03, $P(B|A_3)$=0.01
  - Compute $P(B) = \sum_i P(B \mid A_i)P(A_i)$
    - (.05 * .2) + (.03 * .3) + (.01 * .5) = 0.24
    - 2.4% of the factory output is defective

Bayes Theorem Example

- We are given that B has occurred, and we asked what is the conditional probability $P(A_3/B)$?

$$P(A_3 \mid B) = \frac{P(B \mid A_3)}{P(B)} P(A_3)$$

$$P(A_3 \mid B) = \frac{0.01}{0.024} 0.50 = 0.2083333 \quad posterior$$

Given the knowledge that the item selected was defective allows us to replace the prior probability $P(A_3)$ = 0.5 with $P(A_3/B)$ = 0.2083333

Naive Bayes

- There are several Naive Bayes versions. We will look at the **Bernoulli** version. Features x_i are discrete values {0, 1} for the Bernoulli version and indicate presence/absence of a feature.
- Algorithms that try to learn $P(y|x)$ (or y given x) such as **Linear Regression** are called **discriminative** learning algorithms. Algorithms that instead try to model $P(x|y)$ are called **generative** algorithms.
  - For instance, if y indicates an example is a dog (0) or an elephant (1), then $P(x|y) = 0$ models the distribution of dogs' features, and $P(x|y) = 1$ models the distribution of elephants' features.
- Naive Bayes is an example of a **generative** learning algorithm.

Naive Bayes

- It is called **naive** because it makes **a strong assumption** that all of the x_i's are independent (rarely true in the real world). Yet given this seemingly bold assumption, the algorithm often produces excellent results.
- We want to determine if a given email is *spam* or *not spam*. This is one of many text classification problems

Naive Bayes

After modeling $P(y)$ (called the **class priors**) and $P(x|y)$, the algorithm can use Bayes theorem to derive a posterior prediction on y given x.

$$P(y \mid x) = \frac{P(x \mid y)}{P(x)} P(y)$$

A look at the code

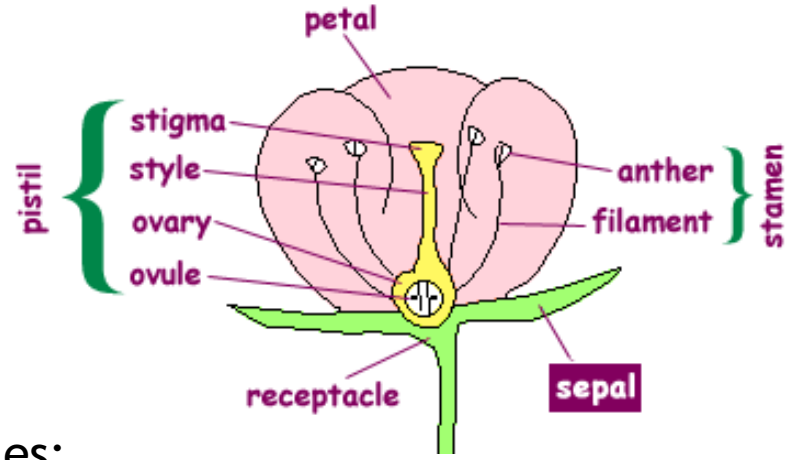
- See code example, jupyter notebook file
  - CS229 - Spam Filter - Naive Bayes Bernoulli.ipynb

Supervised Learning

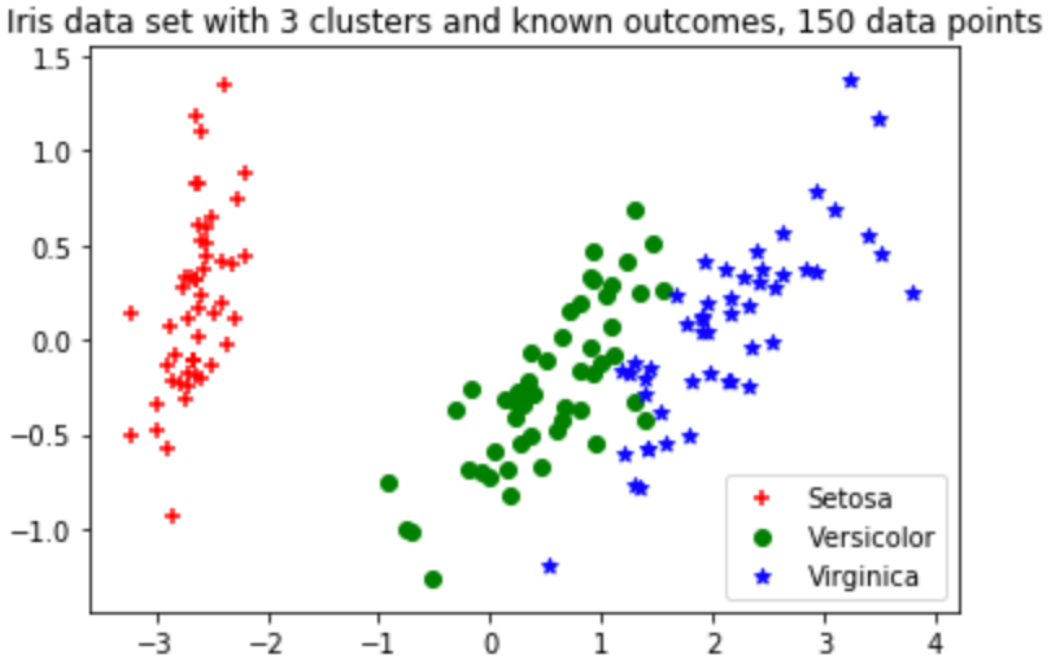
- **Support Vector Machines (SVM)**
- We want to classify Iris flowers

Support Vector Machines

- Iris flower example
- Multivariate dataset of three classes of the Iris flower:
  - *setosa*, *virginica*, *versicolor*
- Introduced in 1936 by Ronald Fisher
- 150 instances, 50 of each flower type, 4 features:
  - length and width of petals
  - length and width of the sepals

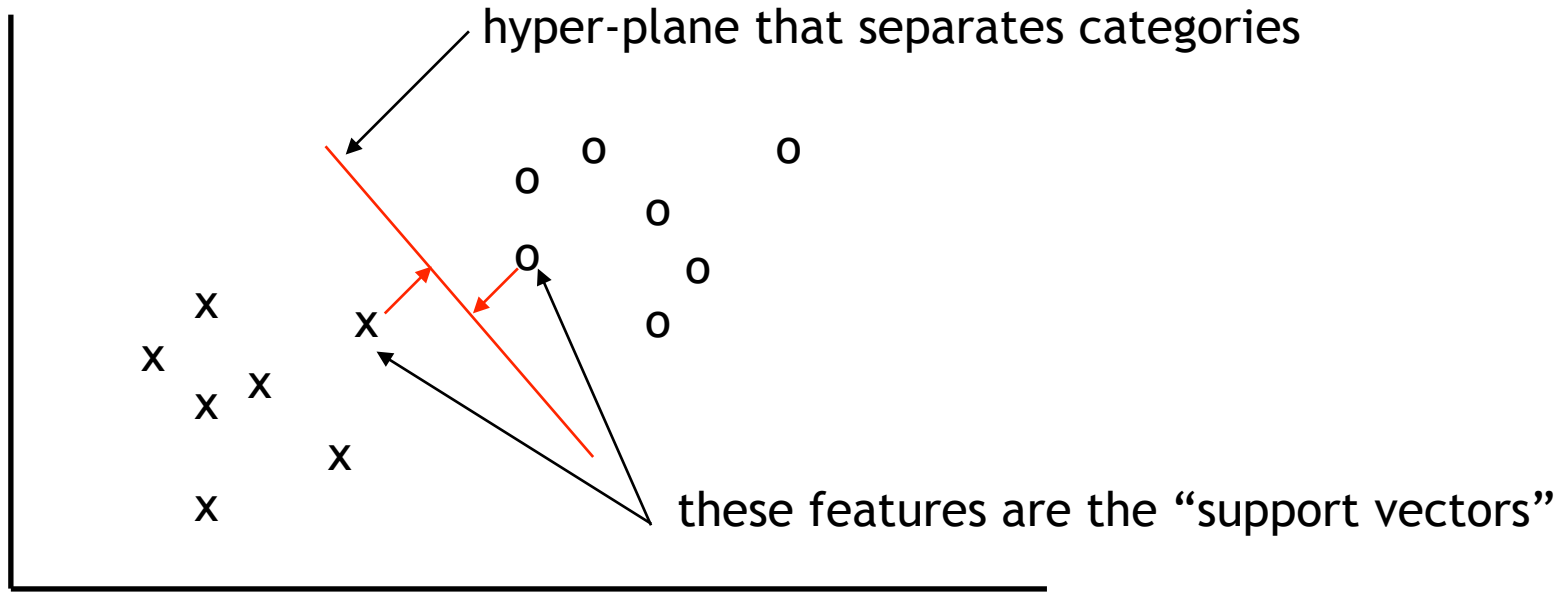


135 examples:



SVM

- A linear SVM is a type of **classifier**, where features fall into 1 of several categories, 2 categories are shown below
- SVM also supports non-linear boundaries



SVM Example

- You can try on the web
- https://www.csie.ntu.edu.tw/~cjlin/libsvm/

SVM - Back to the Iris Example

```
# load a data set
from sklearn.datasets import load_iris
iris = load_iris()

from sklearn.svm import LinearSVC
svmClassifier = LinearSVC(random_state=111)

# separate the data set into: training examples and
test samples
from sklearn import cross_validation
X_train, X_test, y_train, y_test =
cross_validation.train_test_split(iris.data,
    iris.target, test_size=0.10, random_state=111)

# train the model with the training set
svmClassifier.fit (X_train, y_train)

# run the test data, make the predictions
predicted = svmClassifier.predict (X_test)
```

```
# measure performance of the prediction
from sklearn import metrics
accuracy = metrics.accuracy_score (y_test, predicted)
print "Accuracy:"
print accuracy
# correct outcomes/test_size = 14/15 = 0.933333

# create confusion matrix (error matrix)
cm = metrics.confusion_matrix(y_test, predicted)
print cm
```

Outputs

```
Predicted:
[0 0 2 2 1 0 0 2 2 1 2 0 1 2 2]
Expected:
[0 0 2 2 1 0 0 2 2 1 2 0 2 2 2]
```

```
Accuracy:
0.933333333333
```

```
                    Predicted:
                    setosa versicolor virginica
Expected:
setosa              Confusion Matrix:
                    [[5 0 0]
versicolor           [0 2 0]
virginica           [0 1 7]]
```

A look at the code

- See code example, jupyter notebook file
  - Predictive Analytics - Iris Flower - SVM.ipynb

Unsupervised Learning

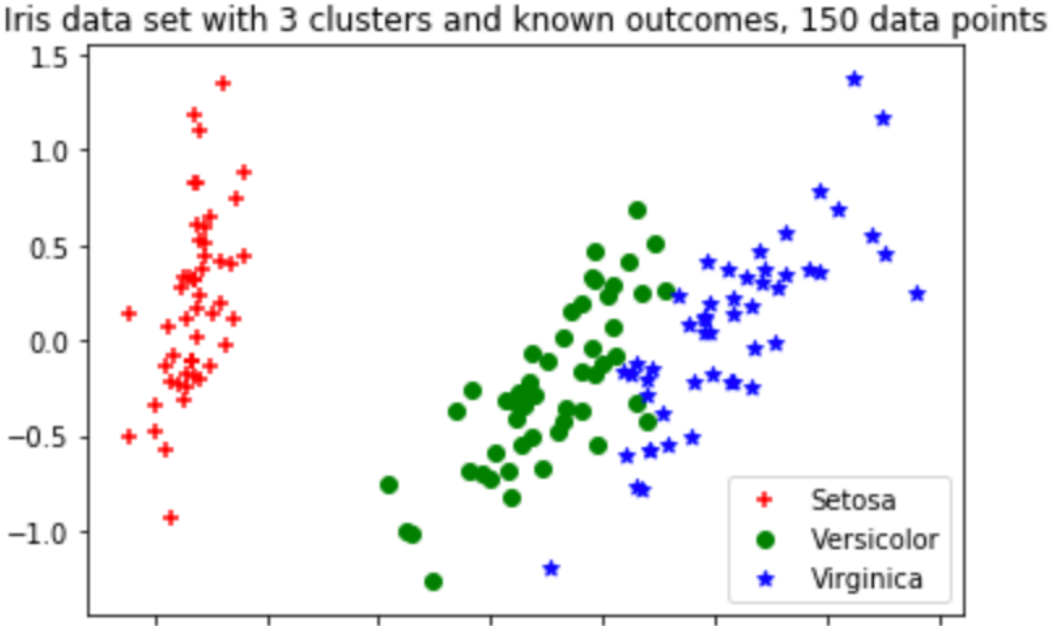
- Generalize steps
  - Load a data set
  - Fit data to the model
  - Visualize the model/clusters
  - Tune parameters
  - Repeat steps 2-4
  - Evaluate the model

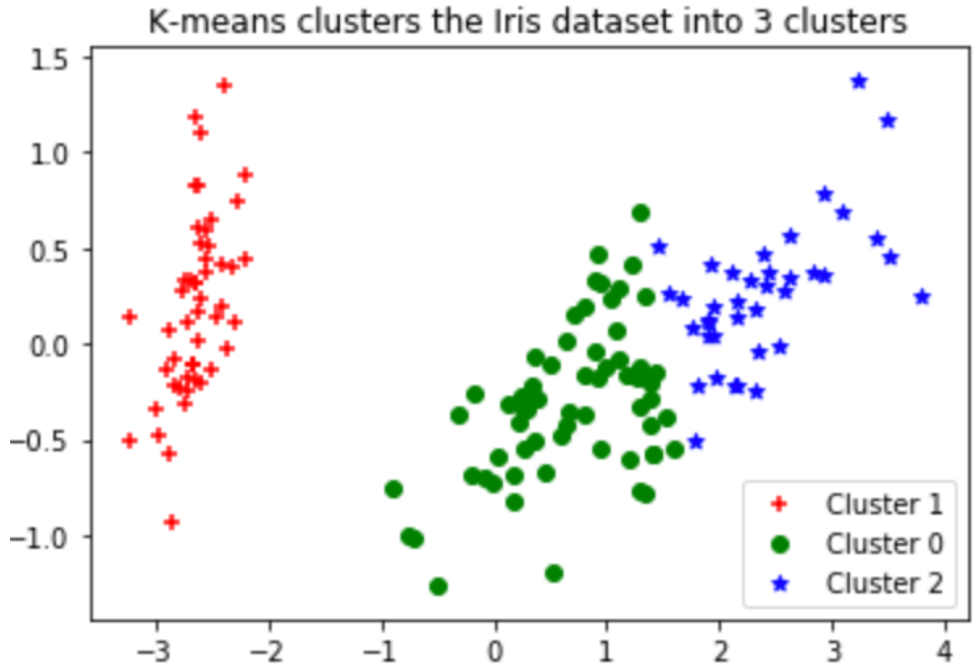
Unsupervised Learning

- **K-means**
  - Looking to extract structure from data
  - Applied to problems where we **don't know the outcomes (targets)** and we want to **cluster similar samples** together into **"k"** clusters
  - Biggest challenge/problem is to select the "correct" number of clusters
  - Good for problems:
    - With a small number of clusters with proportional sizes
    - Clusters are linearly separable

K-Means Example on the Iris Dataset

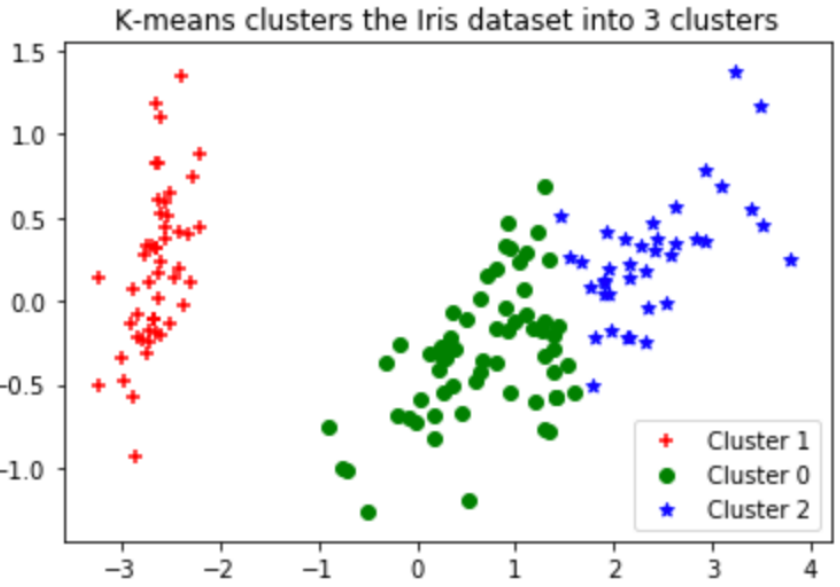
```
# load a data set
from sklearn.datasets import load_iris
iris = load_iris()

# first 50 samples are: setosa
# second 50 samples are: versicolor
# third 50 samples are: virginica

# Run the full dataset
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, random_state=111)
kmeans.fit(iris.data)

print "Clusters:"
print kmeans.labels_
```



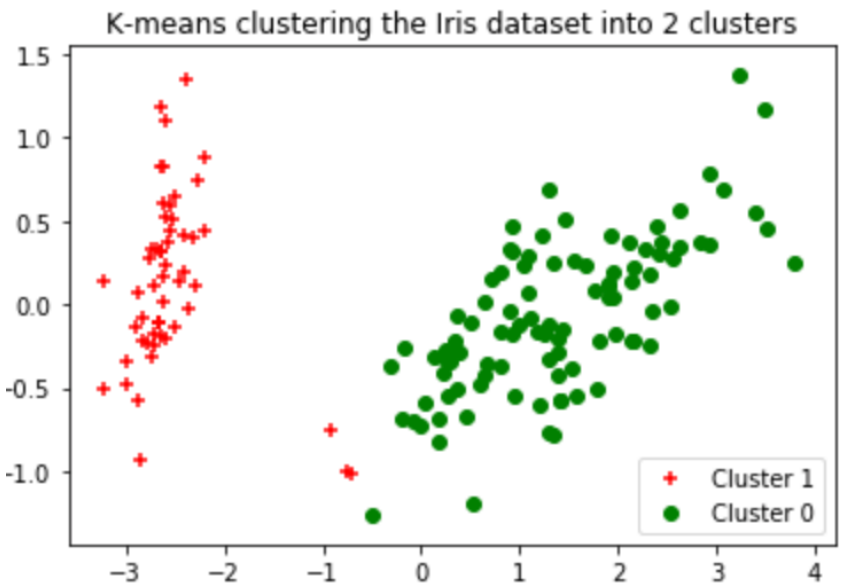


K-Means - 2, 3 & 4 clusters compared

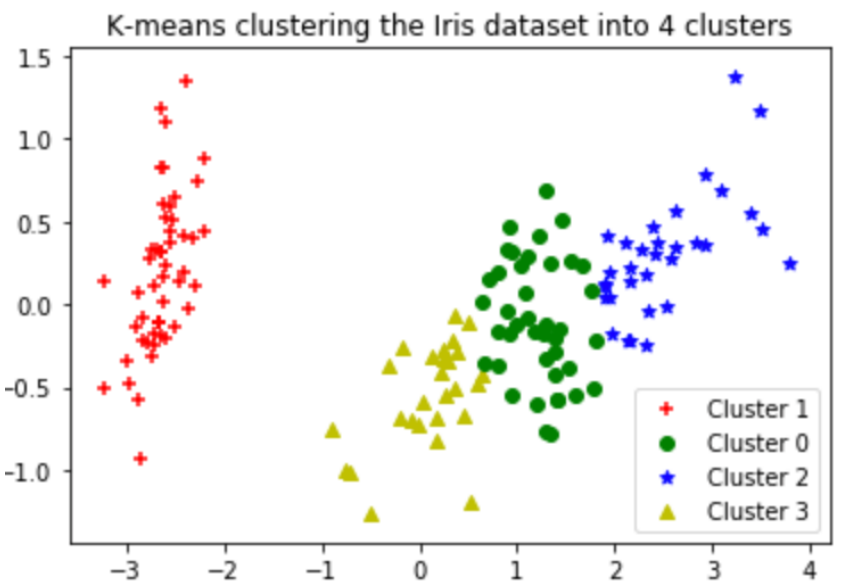
3 clusters



2 clusters



4 clusters



K-Means

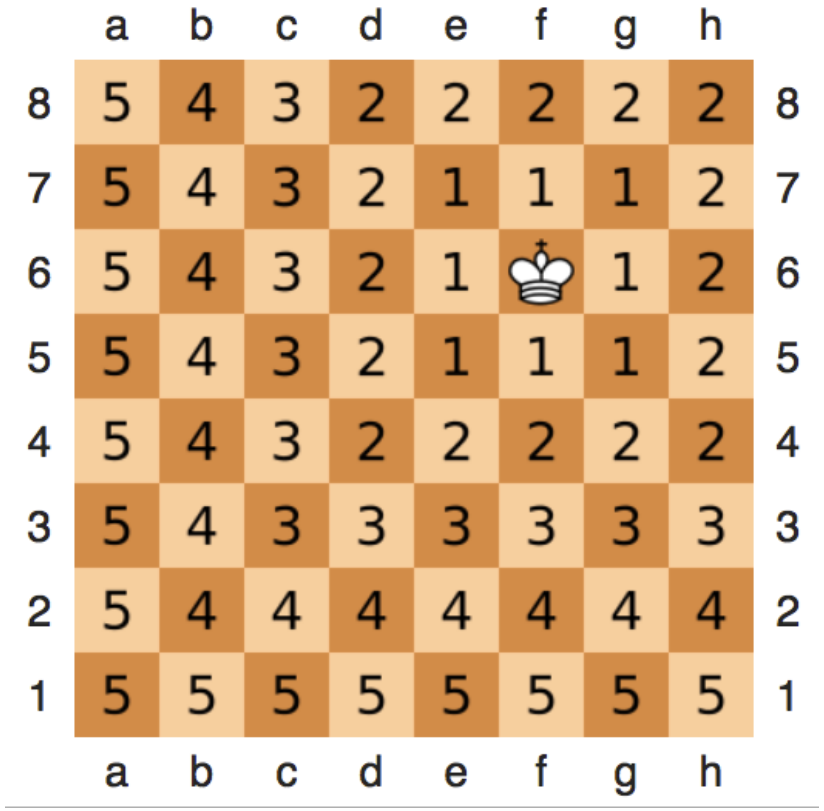
- Works by leveraging similarities among examples
- Data points in a multidimensional space have a **distance** between them. Distance has to follow certain rules:
  - No negative distances
  - Distance from point1 to point2 is the same as the distance from point2 to point1 (symmetry)
  - Distance between an initial point and a farthest point is >= the distance from the initial point, to a 2nd point, then to the farthest point (triangle inequality, i.e. no short cuts)

K-Means Distances

2 points, a and b, in 3 dimensions

- Euclidean Distance (shortest)
  - $= \sqrt{(x^a - x^b)^2 + (y^a - y^b)^2} + (z^a - z^b)^2$
- Manhattan Distance (longest)
  - $= |x^a - x^b| + |y^a - y^b| + |z^a - z^b|$
- Chebyshev Distance
  - $= \max(|x^a - x^b|, |y^a - y^b|, |z^a - z^b|)$

Example of Chebyshev distance



K-Means : Assumptions

- The data “has” clusters
- The clusters are made up of similar examples with the starting example, called the *protoype* or the *centroid*, in the center of the cluster
- The clusters have a “spherical” shape
  - Which isn’t always true for real-world data
- K-Means uses the Euclidean distance
  - Can work with ordinal numbers (1,2,3), binary (0,1) as well

Centroid: is the mean position of all the points in all of the coordinate directions.

K-Means : To be aware of

- Because it uses Euclidean distance, your numbers need to have roughly the same scale. Otherwise features with large values will dominate, potentially throwing off your results
- One solution is to transform your data before K-Means by statistically standardizing all the features and transforming them into new features by a dimensionality reduction process, such as principal component analysis (PCA, more on this later in Data Analytics)
- It expects you to know a-priori how many clusters are in your data.
- Always do a reality check to see whether the result is reproducible under different conditions and makes sense

K-Means : How it works

- 1) User provides K clusters as input. The algorithm picks k random samples as the original centroids
- 2) The algorithm assigns all of the features to each of the k clusters based on their euclidean distance to each centroid. The feature becomes part of the cluster.
- 3) After all features have been assigned to a centroid, the algorithm recalculates the new centroid for each cluster based on the features within the centroid
- 4) The algorithm checks how much the position of each centroid has changed. If the change is under a certain threshold, the algorithm assumes a stable solution and returns the result

Alternate K-Means

- We used sklearn.cluster.KMeans above
- The library contains another method: MiniBatchKMeans
  - Differs from KMeans because it can operate on portions of the data
  - The advantage of MiniBatchKMeans is that it can process data that won't fit in main memory by fetching examples in small portions from disk
    - Perhaps from a Hadoop or Lustre file system disk
  - It can take more time, but produces results similar to KMeans

Monitoring KMeans Centroids

- See code example, jupyter notebook file
  - Predictive Analytics - Iris Flower - K-means.ipynb

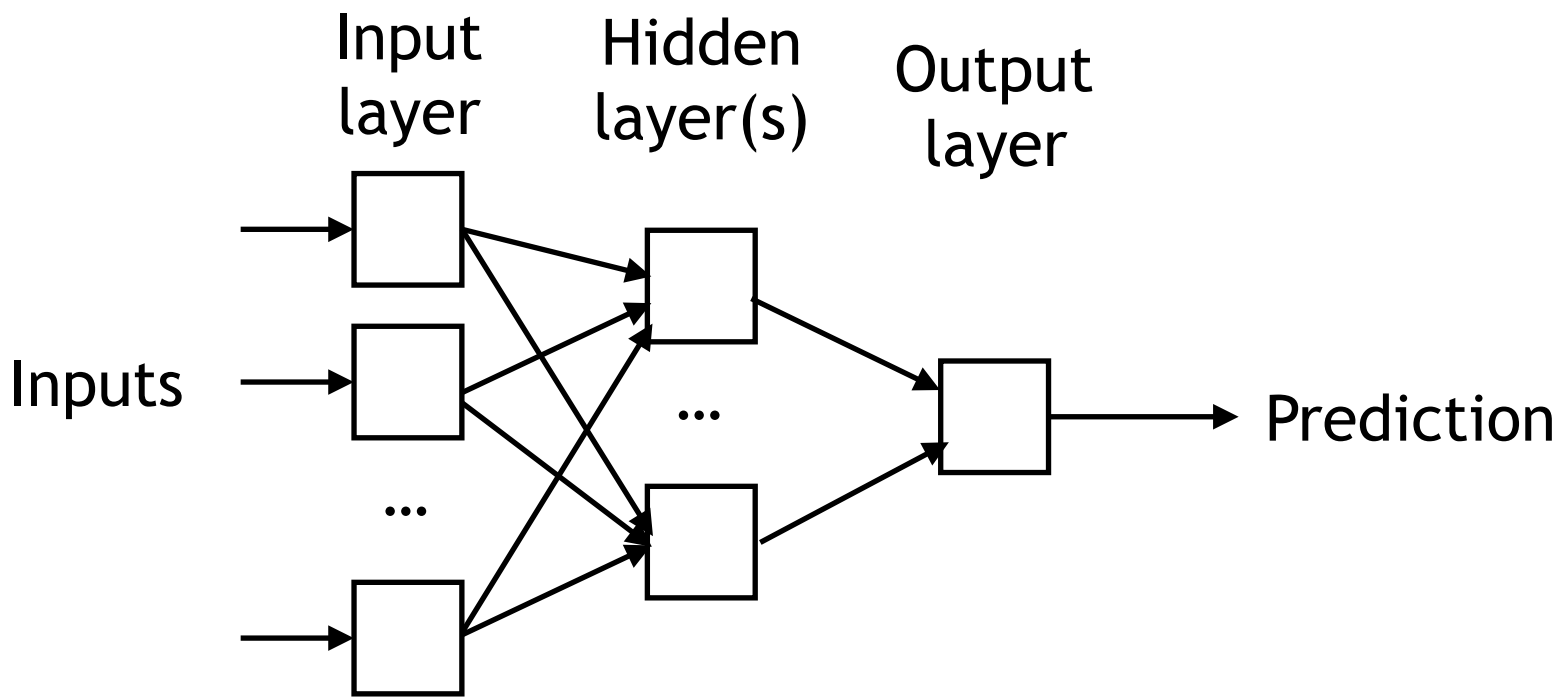
Reinforcement Learning

- It is like unsupervised learning. However you can provide positive and negative feedback with the examples
  - It is like learning by trial and error
- These algorithms make decisions over time and those decisions effect subsequent decisions, working towards maximizing a reward.
- *I have no examples at this time*

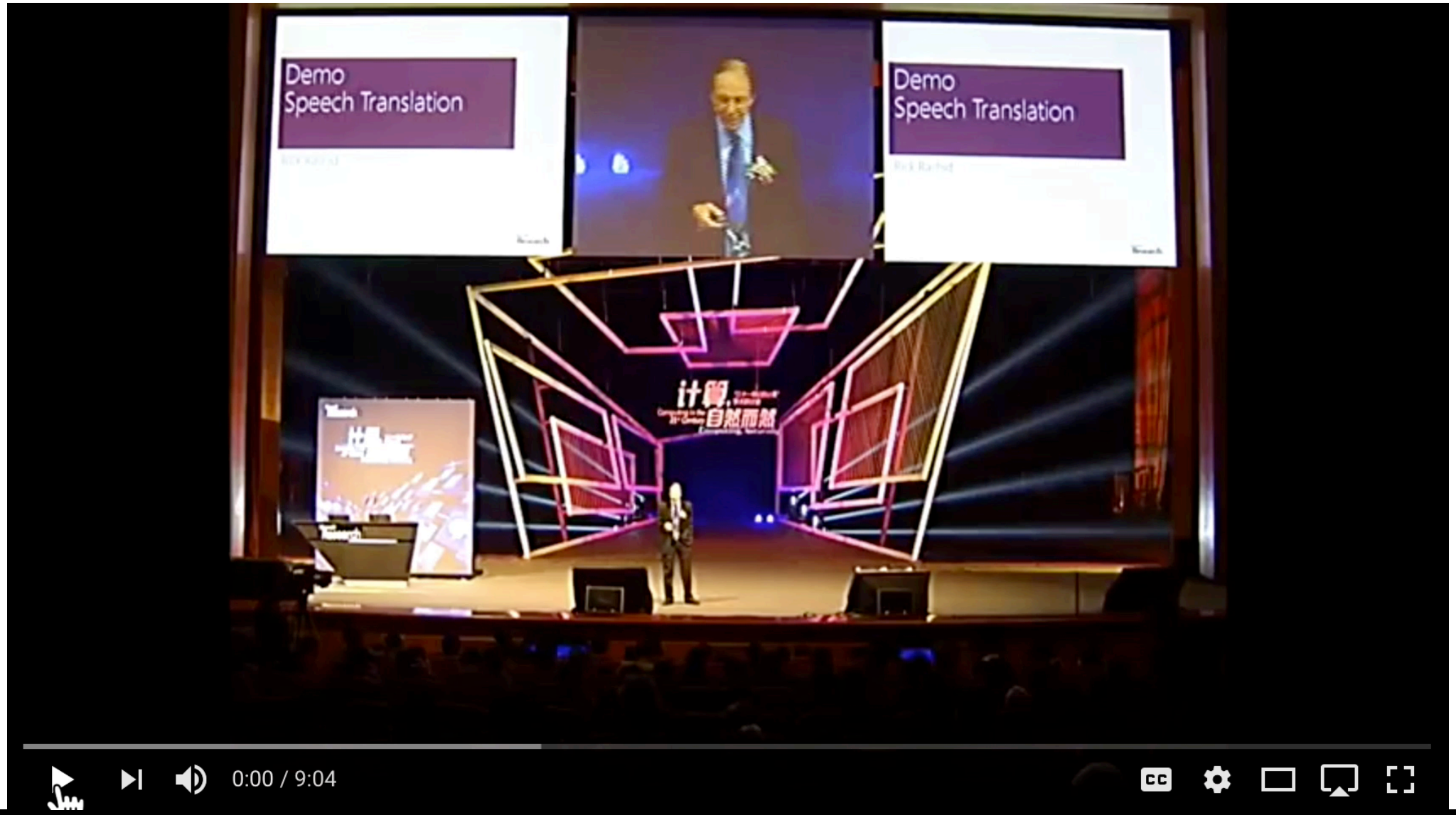
Deep Learning

- Consists of multiple processing layers. See:
  - http://numenta.com
  - https://www.pddnet.com/news/2015/09/neuromemristive-processor-breaks-boundaries-machine-learning
  - http://deeplearning.net
  - https://research.google.com/teams/brain
- Microsoft demo, Chief Research Officer, Rick Rashid
  - Spoken language -> Chinese text -> Chinese speech : https://www.youtube.com/watch?v=Nu-nlQqFCKg
- *Beyond the scope of this course*

Neural networks, they come in many different flavors



Rick Rashid Video



Deep Learning

- See the hearing aid PDF, solving the “cocktail party” problem

Machine Learning Summary

- Is part *art* and part *computer science*
- Failure is frequent
- Requires a great deal of human intervention
  - Asking the right questions
  - Choosing the right “data”
  - Choosing the right algorithm(s)
  - Tuning the algorithm(s) - the hyper parameters
  - Testing/validating the algorithm(s)

Machine Learning in IIoT

- What can machine learning do in the 5 application areas we looked at earlier in the semester?
  - **Automotive and Transportation**
    - Autonomous navigation, safety. Identify road and weather conditions.
  - **Industrial**
    - Predicting equipment failures, identifying opportunities for improved operational efficiency
  - **Building Automation**
    - Biometric based physical access, energy use optimization
  - **Oil and Gas**
    - Predicting hydrocarbon locations, predicting equipment failures
  - **Agriculture**
    - Predicting consumables (seed, water, herbicides, fertilizer) usage

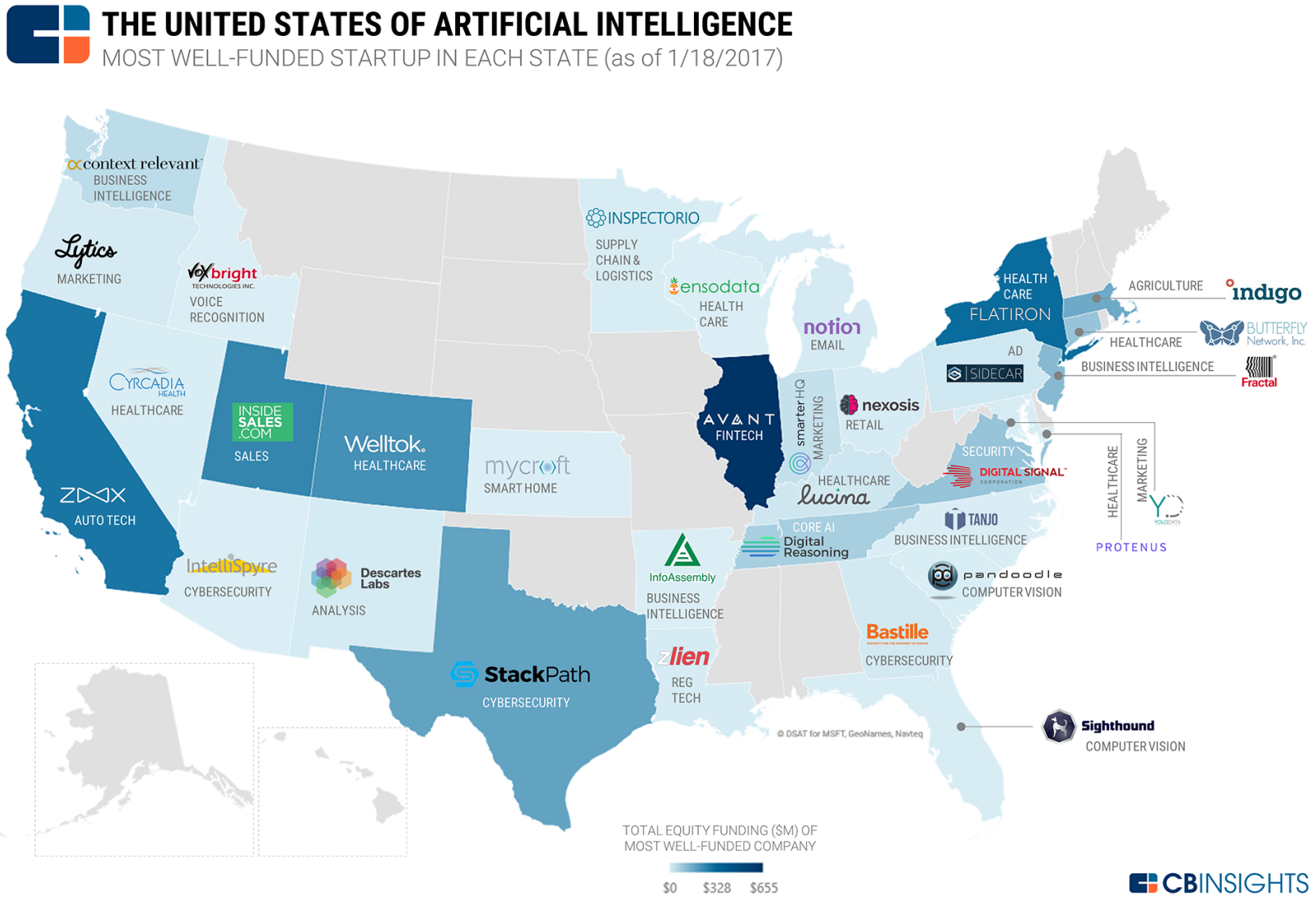
Applying the Algorithms

- How can I apply these algorithms in my embedded system?
- Two basic approaches:
  - Train offline in a lab, then deploy the trained algorithm(s) in the field
  - Train in the field

Sources

- Standford CS 229: http://cs229.stanford.edu
  - Videos of the lectures are free on iTunes U
- "*Machine Learning for Dummies*", John Paul Mueller
- "*Predictive Analytics for Dummies*", Anasse Bari, Ph.D.
- "*Deep Learning*", Ian Goodfellow
- Anaconda: https://www.continuum.io/downloads
- http://scikit-learn.org
- https://hackernoon.com/ten-machine-learning-algorithms-you-should-know-to-become-a-data-scientist-c11a3e735f93

Business Info

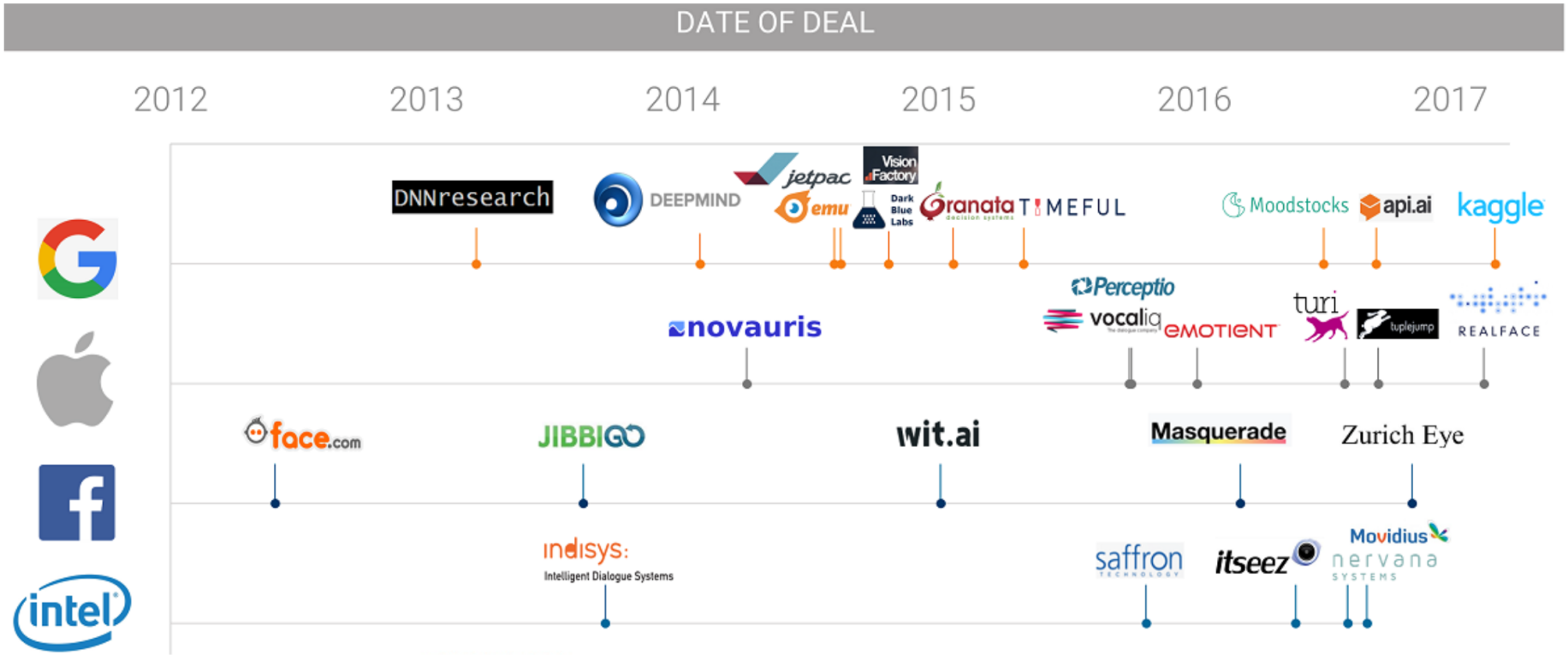


Source: CBInsights

Business Info

MARCH 30, 2017

The Race For AI: Google, Twitter, Intel, Apple In A Rush To Grab Artificial Intelligence Startups



Source: CBInsights

Elon Musk's AI Efforts

OpenAI : https://openai.com
OpenAI is a non-profit AI research company, discovering and enacting the path to safe artificial general intelligence.

FutureOfLife : https://futureoflife.org
Mission: To catalyze and support research and initiatives for safeguarding life and developing optimistic visions of the future, including positive ways for humanity to steer its own course considering new technologies and challenges.

NeuraLink : https://www.neuralink.com
Neuralink is developing ultra high bandwidth brain-machine interfaces to connect humans and computers.
Also known as brain-computer-interfaces (BCI)

End