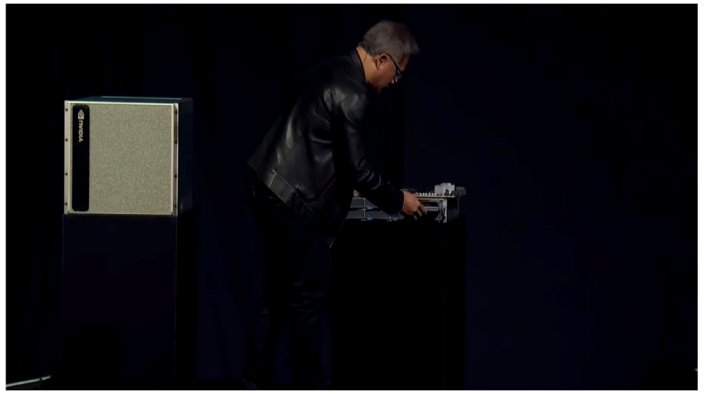
CS 644: Introduction to Big Data

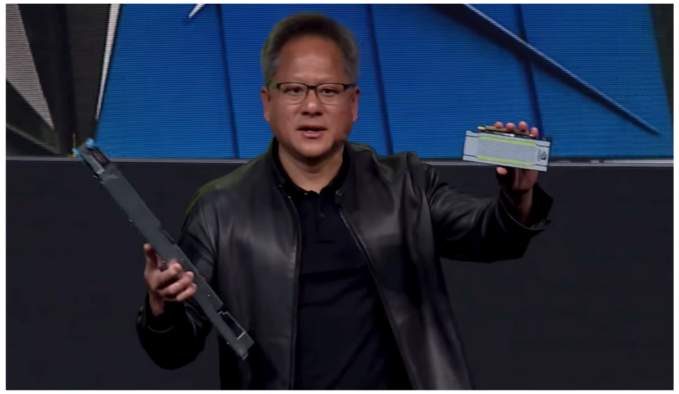
Daqing Yun (daqing.yun@njit.edu)
New Jersey Institute of Technology

Cori SC@LBNL

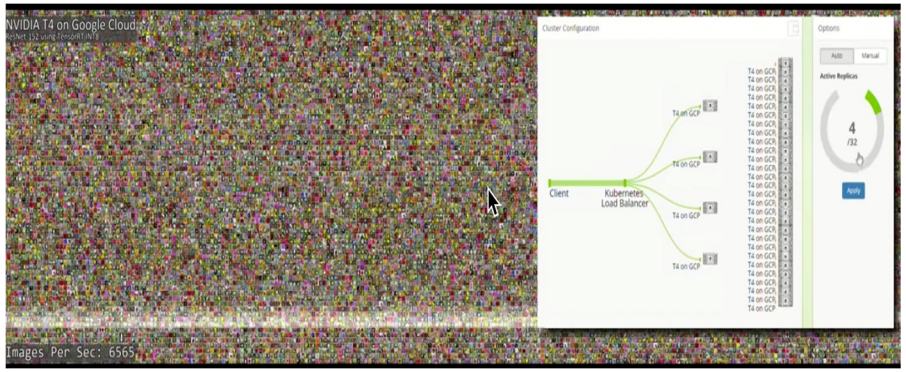
SC '18



NVIDIA T4 GPU
https://youtu.be/c29B-sfzJvY?t=5410

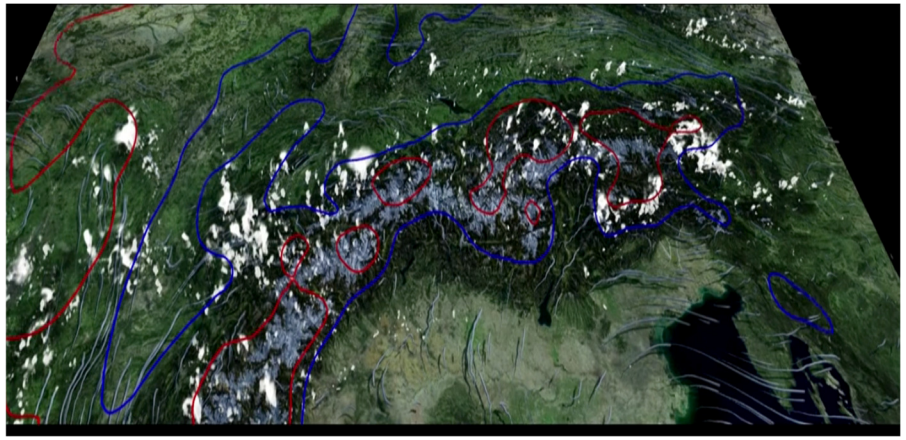


NVIDIA T4 GPU
https://youtu.be/c29B-sfzJvY?t=5450

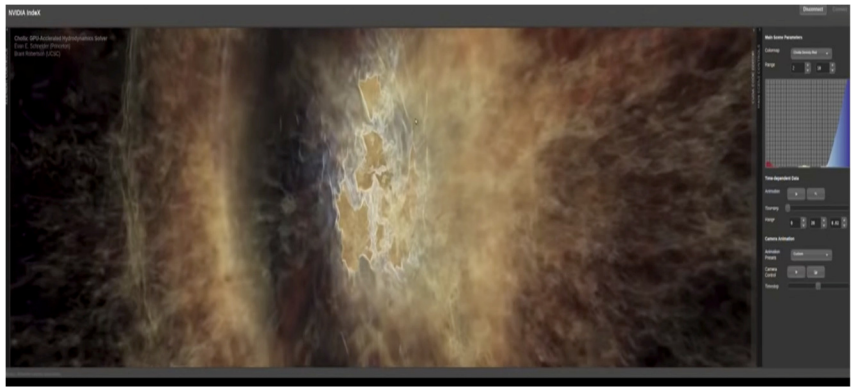


Fast Image Classification
https://youtu.be/c29B-sfzJvY?t=6094

Deployed in Google Cloud



Micro Weather Prediction
https://youtu.be/c29B-sfzJvY?t=5100



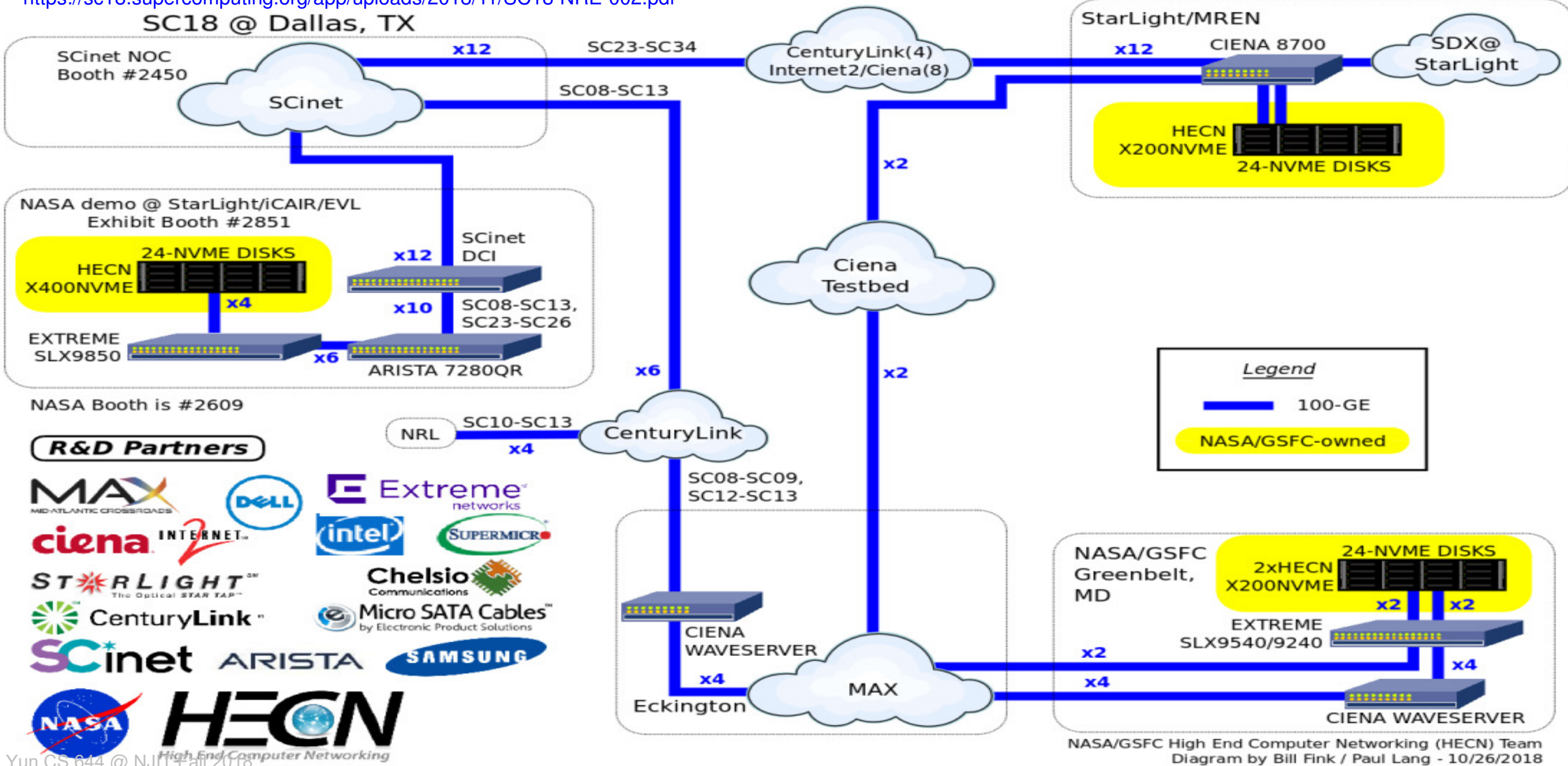
Galaxy Simulation
https://youtu.be/c29B-sfzJvY?t=4036

SC18

Demonstrations of 400 Gbps Disk-to-Disk WAN File Transfers using iWARP and NVMe-oF

An SC18 Collaborative Initiative Among NASA and Several Partners

https://sc18.supercomputing.org/app/uploads/2018/11/SC18-NRE-002.pdf



Top 500 List

Green 500 List

Rank	TOP500 Rank	System
1	375	**Shoubu system B** - ZettaScaler-2.2, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 , PEZY Computing / Exascaler Inc. Advanced Center for Computing and Communication, RIKEN Japan
2	374	**DGX SaturnV Volta** - NVIDIA DGX-1 Volta36, Xeon E5-2698v4 20C 2.2GHz, Infiniband EDR, NVIDIA Tesla V100 , Nvidia NVIDIA Corporation United States
3	1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States
4	7	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2570 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR , Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan

HPCG List

Rank	TOP500 Rank	System
1	1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States
2	2	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States
3	18	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect , Fujitsu RIKEN Advanced Institute for Computational Science (AICS) Japan
4	6	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect , Cray Inc. DOE/NNSA/LANL/SNL United States

https://www.top500.org

Big Data Analytics Algorithms

- Similarity (Distance) Measurements
- Clustering
  - K-Means
- Classification
  - k Nearest Neighbor
  - Logistics Regression (via Gradient Descent)
  - Decision Tree Induction
  - Naïve Bayes
  - Support Vector Machine
- Regression
  - Linear Regression (Single and Multiple) and Non-linear Regression
- Ensemble Methods
  - Random Forest
  - AdaBoost
- Principle Component Analysis
- *Neural Networks*

Hadoop-related Apache Projects

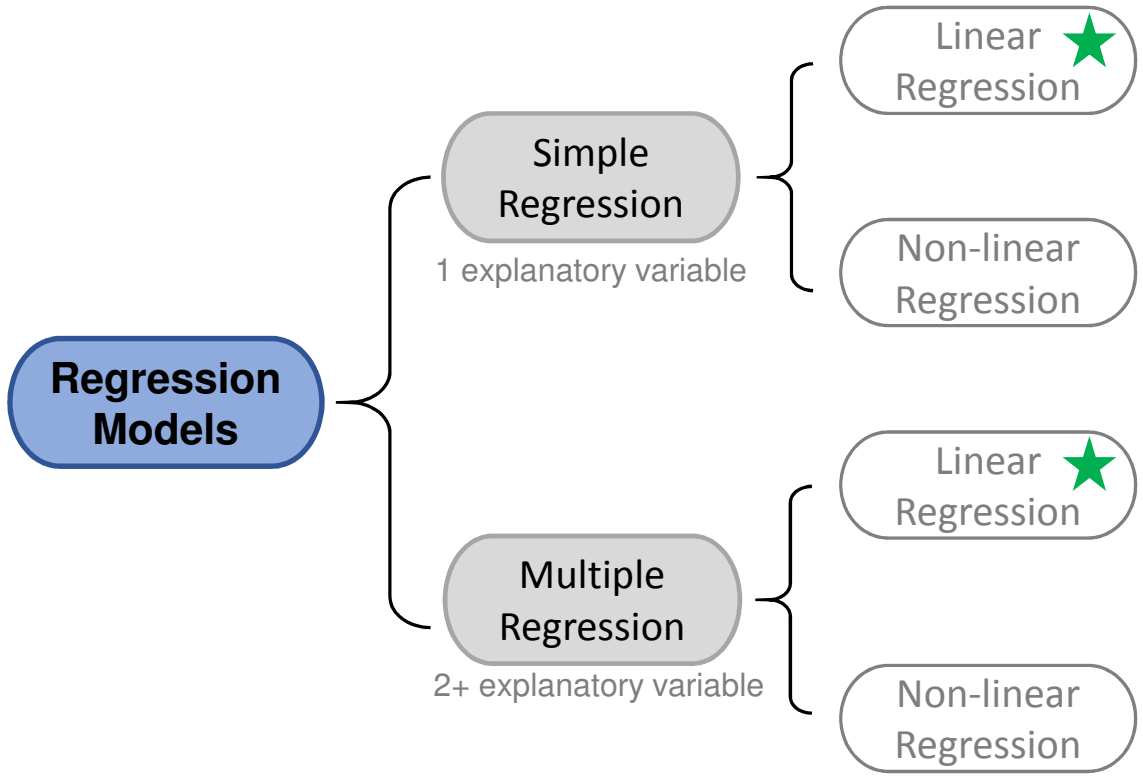
- **Ambari™**: A web based tool for provisioning, managing, and monitoring Hadoop clusters. It also provides a dashboard for viewing cluster health and ability to view MapReduce, Pig, and Hive applications visually.
- **Avro™**: A data serialization system.
- **Cassandra™**: A scalable multi-master database with no single points of failure.
- **Chukwa™**: A data collection system for managing large distributed systems.
- **HBase™**: A scalable, distributed database that supports structured data storage for large tables.
- **Hive™**: A data warehouse infrastructure that provides data summarization and ad hoc querying
- **Mahout™**: A scalable machine learning and data mining library.
- **Pig™**: A high-level data-flow language and execution framework for parallel computation
- **Spark™**: A fast and general compute engine for Hadoop data. Spark provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation.
- **Tez™**: A generalized data-flow programming framework, built on Hadoop YARN, which provides a powerful and flexible engine to execute an arbitrary DAG of tasks to process data for both batch and interactive use-cases.
- **ZooKeeper™**: A high-performance coordination service for distributed applications.

(Non-)Linear Regression

Linear Regression

- ML models for regression problems predict a numeric value.
- Examples
  - What will the temperature be in Seattle tomorrow?
  - For this product, how many units will sell?
  - What price will this house sell for?

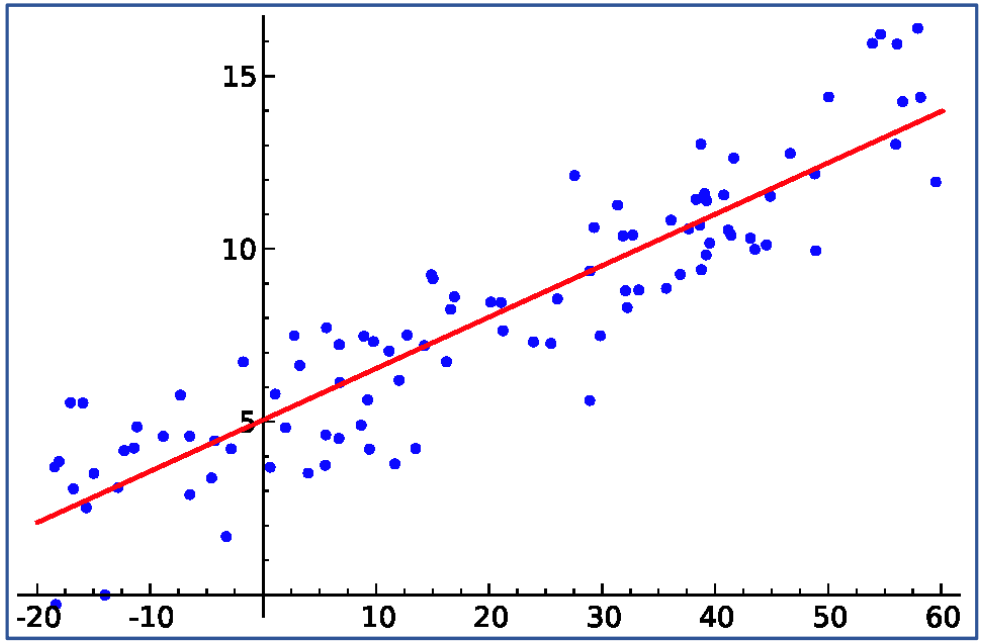
- Type of regression models:



Linear Regression

What is Linear Regression?

- Linear regression is a linear approach for modeling the relationship between a dependent variable *y* and explanatory variables (or independent variables) denoted *x*



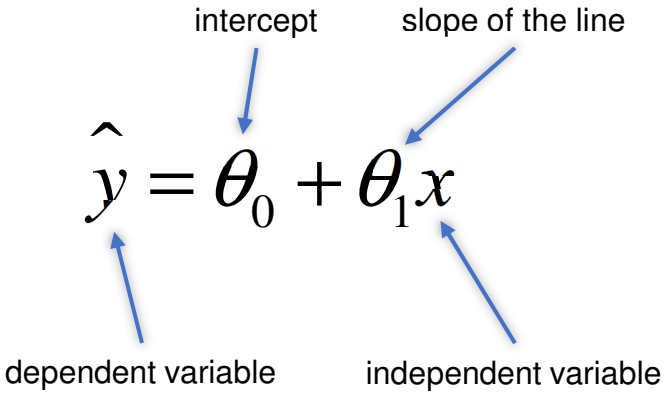
- It is a supervised algorithm that learns from a set of training samples.
- Each training sample has one or more input values and a single output value.
- The algorithm learns the line, plane or hyper-plane that best fits the training samples.
- Use the learned line, plane or hyper-plane to predict the output value for any input sample.

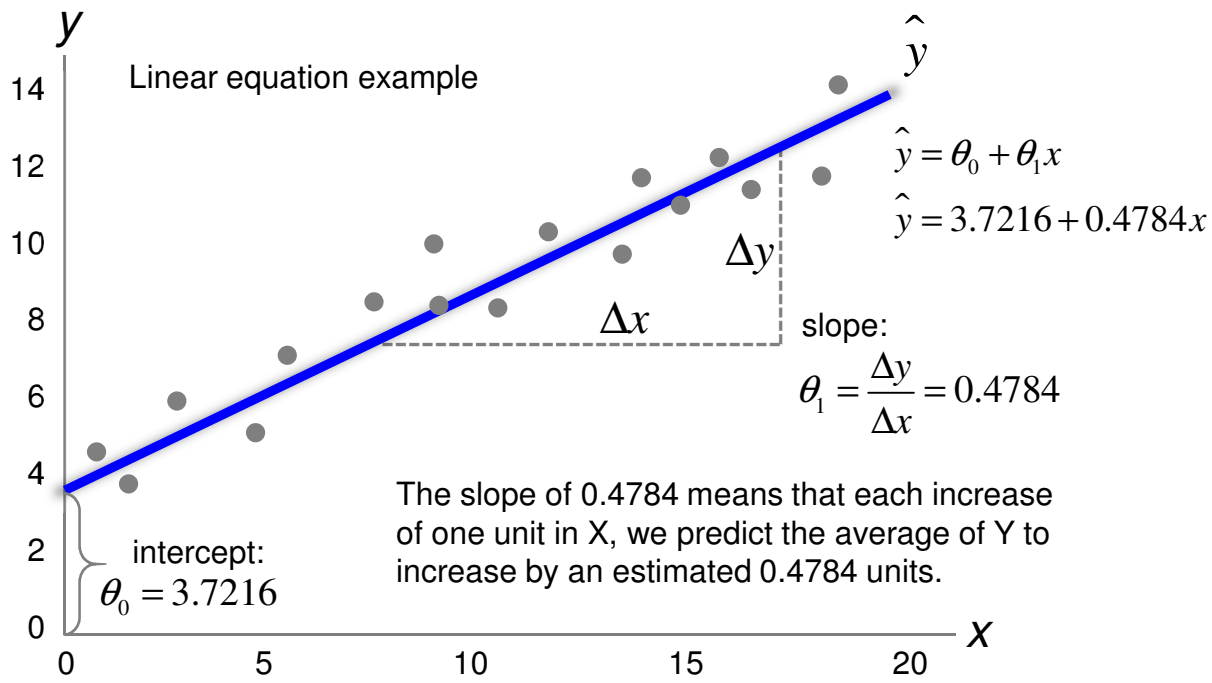
- Because linear regression models are simple to interpret and easy to train, they are often the first model to be fitted to a new dataset.
- Linear regression can be used as a baseline for evaluating other, more complex, regression models

Linear Regression

Simple Linear Regression

- Simple linear regression is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables:
  - One variable, denoted x, is regarded as the **predictor**, **explanatory**, or **independent** variable.
  - The other variable, denoted y, is regarded as the **response**, **outcome**, or **dependent** variable.
- Simple linear regression is termed as simple because there is only independent variable. The relationship between the input x and outcome y can be represented in a form of following equation:



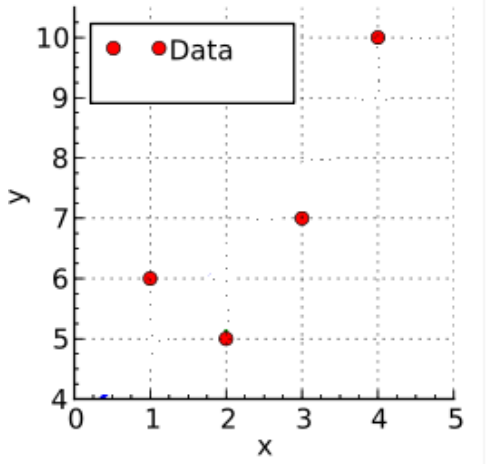


Linear Regression

Example: predict house's price using linear regression

- Suppose X is size of the house in Acres, and Y is the price of houses already sold in hundred thousands, and you have a house that has 2.5 acres size, and you want to know how much you can get for it if you decide to sell it
- Before attempting to fit a linear model to observed data, a modeler should first determine whether or not there is a relationship between the variables of interest. A scatterplot can be a helpful tool in determining the strength of the relationship between two variables. In this example, the price has a positive correlation with the size (0.8367)

X	Y
1	6
2	5
3	7
4	10



$$r_{xy} = \frac{\mathrm{cov}(x, y)}{\sigma_x \sigma_y}$$

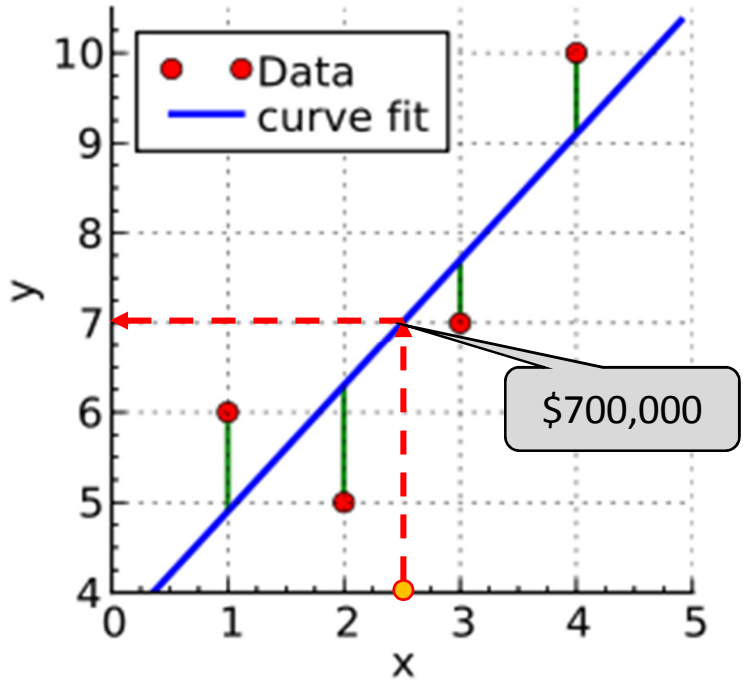
Linear Regression

Example: predict house's price using linear regression

- Then we can identify best-fitting relationship (line) between the variables. The regression line is the "best-fit" line and has the formula $y = \theta_0 + \theta_1 x$. Thus, given a value of X, we can predict a value of Y

Based on this regression line, we can predict the price for a house with a certain size

X	Y
1	6
2	5
3	7
4	10
2.5	?



Linear Regression

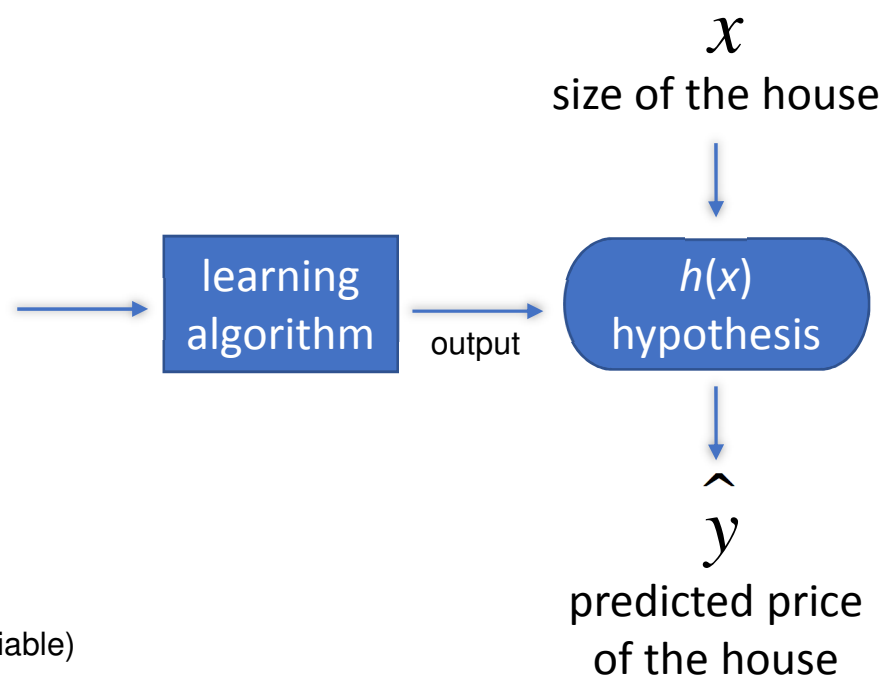
Example: predict house's price using linear regression

- Actually linear regression model guesses a hypothesis function h(x) from training set. Given a training set, the model learn a function h(x) which can be used for predicting the new/unseen data

Training set

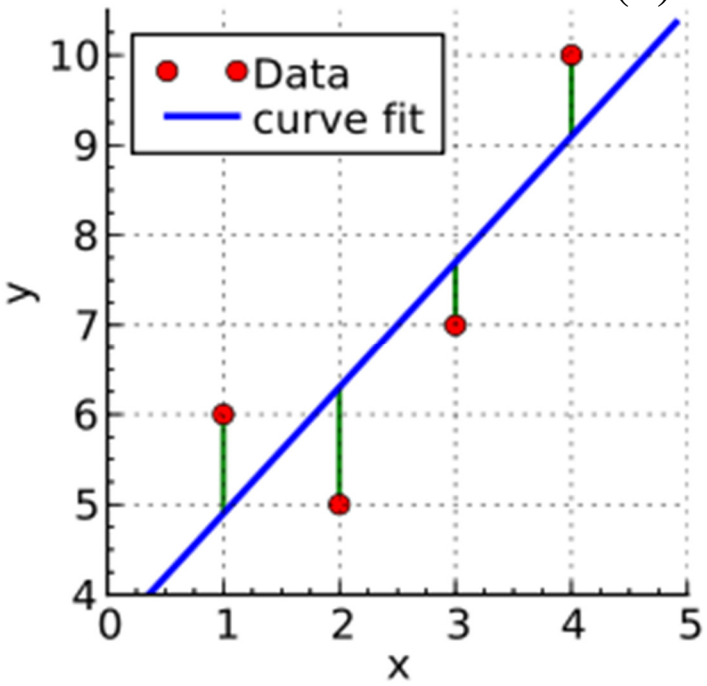
X	Y
1	6
2	5
3	7
4	10
2.5	?

- X is input (feature)
- Y is Output (target variable)



The regression line h(x) is a hypothesis/function

$h(x) = \theta_0 + \theta_1 x$



Linear Regression

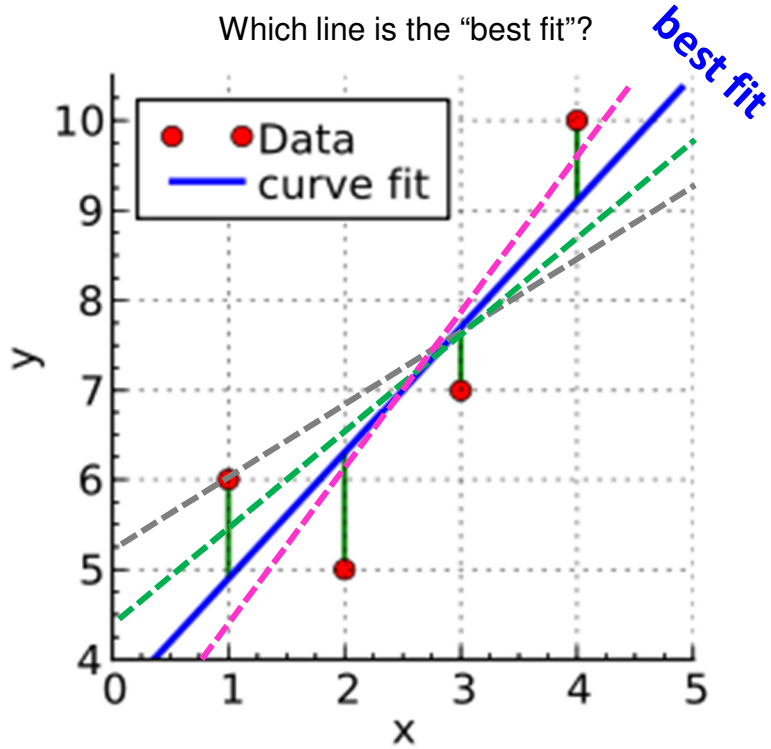
Example: predict house's price using linear regression

- The goal of linear regression is to find the best fit line, i.e., to find estimated value for the parameters θ_0 and θ_1 which would provide the "best" fit for the data points within the training dataset

$$h(x) = \theta_0 + \theta_1 x$$

Different parameters θ_0 and θ_1 will result in different lines

X	Y
1	6
2	5
3	7
4	10
2.5	?

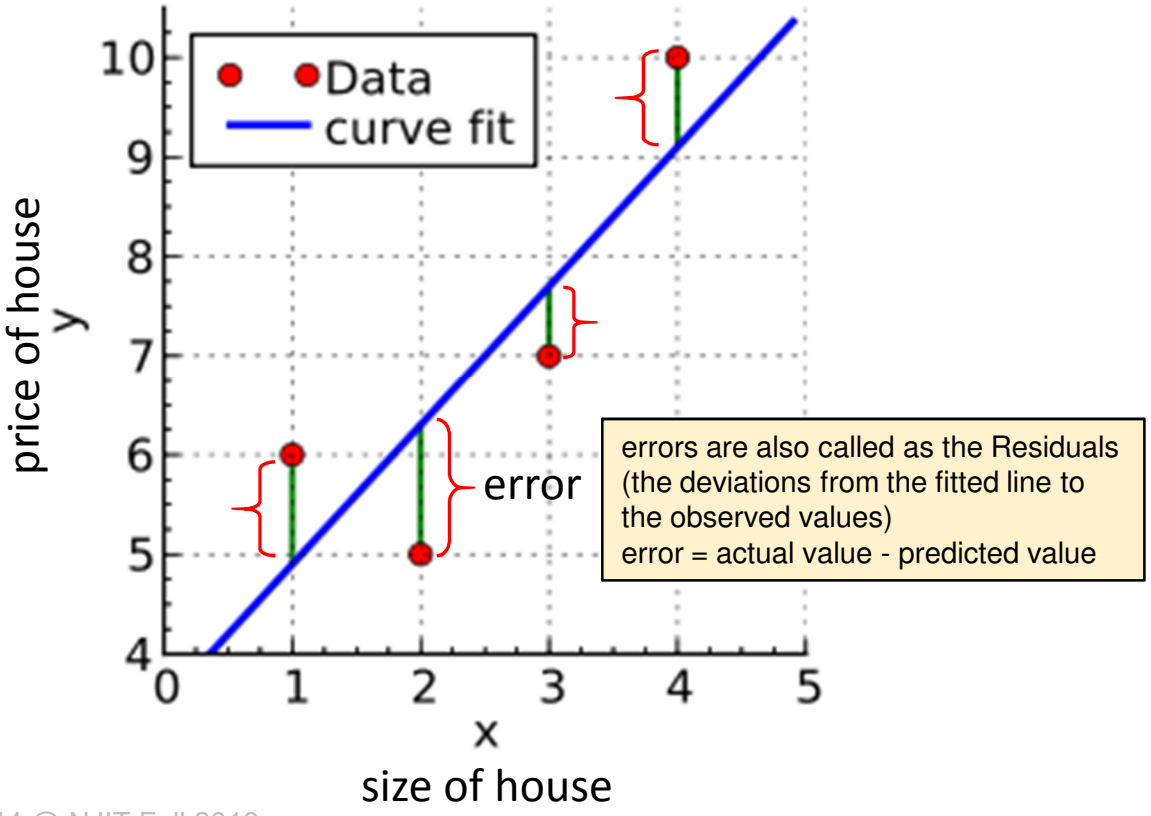


best fit line: $\theta_0 = 3.5, \theta_1 = 1.4$

Linear Regression

Example: predict house's price using linear regression

- How do we know which line is the "best fit"? -- the good line is one that minimizes the sum of the "squared differences" between the points and the regression line. Usually some data points will deviates from the line somehow (if a point lies on the fitted line exactly, then its vertical deviation is 0)



Least Squares Method

- The optimization goal is to minimize the sum of "squared error" (least squares).
- Because the deviations are first squared, then summed, there are no cancellations between positive and negative values.

$$J = \sum_{i=1}^{m} \left(y_i - \widehat{y_i} \right)^2$$

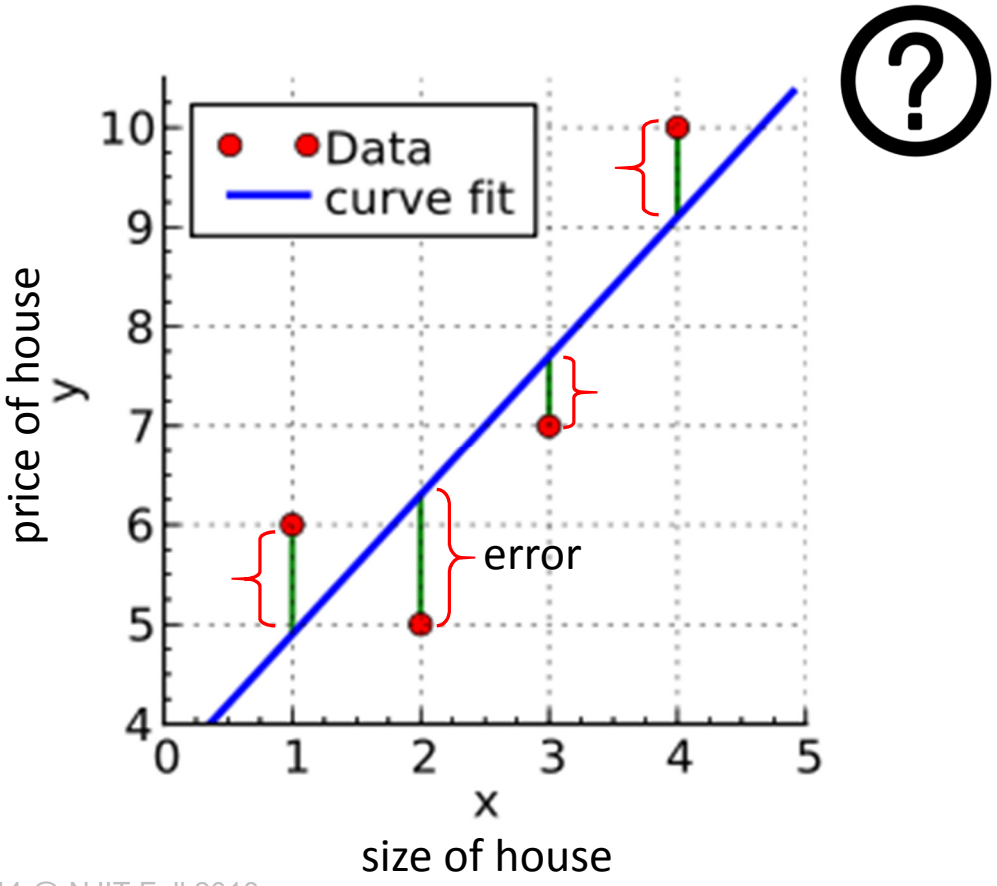
m is the number of training instances, and $\hat{y} = \theta_0 + \theta_1 x$

- The smaller the sum of squared differences are, the better the fit of the line to the data is.

Linear Regression

Example: predict house's price using linear regression

- How do we come up with the "best" parameter value that corresponds to a good fit to the data? -- the idea is to choose the appropriate parameter θ_0 and θ_1 so that $\hat{y}$ is close to y for training example (x, y). In other words, the sum of squared error should be the least



- How to find appropriate parameters θ_0 and θ_1 in order to minimize the error, i.e., the cost function?
- To minimize: $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(y_i - \hat{y}_i\right)^2$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(y_i - (\theta_0 + \theta_1 x_i)\right)^2$$

https://youtu.be/yR2ipCoFvNo

Linear Regression

The computation methods for estimating the parameters

- There are a couple of approaches to find the value of parameter $\boldsymbol{\theta}$ that minimizes the cost function $J(\boldsymbol{\theta})$

- Normal Equation (Closed Form)
  - A method to solve for the parameters analytically
  - Using a direct "closed form" equation that directly computes the model parameters that best fit the model to the training set (i.e., the model parameters that minimize the cost function over the training set)
  - Suitable for small feature set (e.g., > 1000 features)

- Gradient Descent
  - Using an iterative optimization approach, called Gradient Descent (GD), that gradually tweaks the model parameters to minimize the cost function over the training set, eventually converging to the same set of parameters as the first method
  - Gradient Descent is better choice than Normal Equation when there are a large number of features, or too many training instances to fit in memory

A. Géron. Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly, 2017

Linear Regression

The computation methods for estimating the parameters

- There are a couple of approaches to find the value of parameter $\boldsymbol{\theta}$ that minimizes the cost function $J(\boldsymbol{\theta})$

- Normal Equation (Closed Form)
  - A method to solve for the parameters analytically
  - Using a direct "closed form" equation that directly computes the model parameters that best fit the model to the training set (i.e., the model parameters that minimize the cost function over the training set)
  - Suitable for small feature set (e.g., > 1000 features)

- Gradient Des
  - Using an itera … e model parameters to minimize t … meters as the first method
  - Gradient Des … es, or too many training insta

In a simple linear regression the $\theta_0 = \bar{y} - \theta_1 \bar{x}$ coefficients are calculated as:

$$\theta_1 = \frac{\sum_{i=1}^{m}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sum_{i=1}^{m}\left(xi - \bar{x}\right)^2} = \frac{\mathrm{Cov}(X,Y)}{\mathrm{Var}(X)}$$

In a matrix (vector) notation, the coefficients are calculated as: $\boldsymbol{\theta} = \left(\mathbf{x}^{\mathrm{T}}\mathbf{x}\right)^{-1} \cdot \mathbf{x}^{\mathrm{T}} \cdot \mathbf{y}$

A. Géron. Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly, 2017

Linear Regression

The computation methods for estimating the parameters

- Normal Equation (Closed Form)
  - To find the value of $\boldsymbol{\theta}$ that minimizes the cost function, there is a closed-form solution -- in other words, a mathematical equation that gives the result directly. This is called the Normal Equation.

1. Equation of multivariate linear regression

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \ldots + \theta_n x_n$$
$$= h_\theta(\mathbf{x}) = \boldsymbol{\theta}^T \cdot \mathbf{x}$$

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \cdots \\ \theta_n \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_{10} & x_{20} & \cdots & x_{m0} \\ x_{11} & x_{21} & \cdots & x_{m1} \\ \cdots & & \ddots & \cdots \\ x_{1n} & x_{2n} & \cdots & x_{mn} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_m \end{bmatrix}$$

m training instances
n attributes in the feature vector
x_0 is always equal to 1

2. Set derivatives equal to zero and solve for parameters

- Explicitly take its derivatives with respect to each parameter, and set its derivatives to zero, and obtain the normal equations

$$\frac{\partial}{\partial \theta} J(\boldsymbol{\theta}) = 2\mathbf{x}^T \mathbf{x}_\theta - 2\mathbf{x}^T \mathbf{y} = 0 \quad \Rightarrow \quad \mathbf{x}^T \mathbf{x}_\theta = 2\mathbf{x}^T \mathbf{y}$$

- The parameter values that minimize the cost function is given in closed form by

$$\boldsymbol{\theta} = \left(\mathbf{x}^{\mathrm{T}} \mathbf{x}\right)^{-1} \cdot \mathbf{x}^{\mathrm{T}} \cdot \mathbf{y} \qquad //\left(\mathbf{x}^{\mathrm{T}} \mathbf{x}\right)^{-1} \text{ is the inverse of matrix } \mathbf{x}^{\mathrm{T}} \mathbf{x}$$

The Normal Equation method gets very slow when the number of features grows large (e.g., 100,000)

https://youtu.be/B-Ks01zR4HY?t=546
https://youtu.be/EZ1gRqVeMSQ

A. Géron. Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly, 2017

Linear Regression

The computation methods for estimating the parameters

- There are a couple of approaches to find the value of parameter $\boldsymbol{\theta}$ that minimizes the cost function $J(\boldsymbol{\theta})$

- Normal Equation (Closed Form)
  - A method to solve for the parameters analytically
  - Using a direct "closed form" equation that directly computes the model parameters that best fit the model to the training set (i.e., the model parameters that minimize the cost function over the training set)
  - Suitable for small feature set (e.g., > 1000 features)

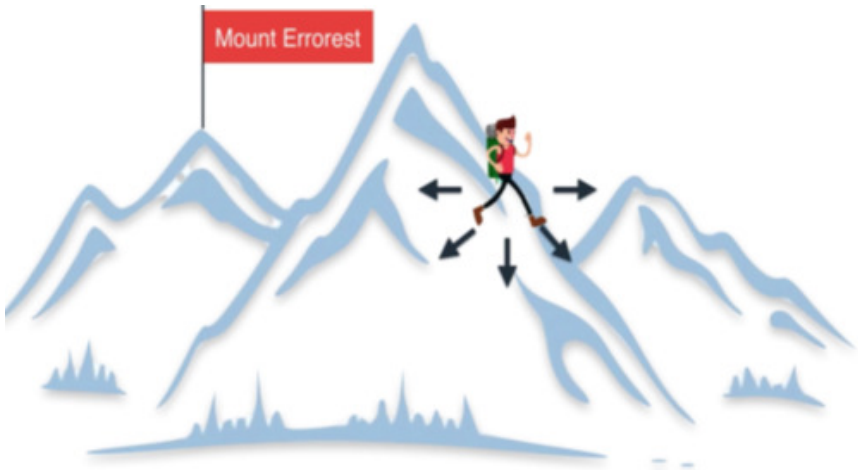
- Gradient Descent
  - Using an iterative optimization approach, called Gradient Descent (GD), that gradually tweaks the model parameters to minimize the cost function over the training set, eventually converging to the same set of parameters as the first method
  - Gradient Descent is better choice than Normal Equation when there are a large number of features, or too many training instances to fit in memory

A. Géron. Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly, 2017

Linear Regression

The computation methods for estimating the parameters

- **Gradient Descent** is one of the most popular optimization algorithms used in ML/DL
- ML algorithms such as linear regression, logistic regression, etc. use Gradient Descent



- Illustrate Gradient Descent optimization as a process that a blindfolded hiker wants to go down a mountain as soon as possible.
- A good strategy to get to the bottom of the valley quickly is to go downhill in the direction of the steepest slope.
- This is exactly what Gradient Descent does:
  - It measures the local gradient of the error function with regards to the parameter vector.
  - It goes in the direction of descending gradient.
  - Once the gradient is zero, you have reached a minimum.

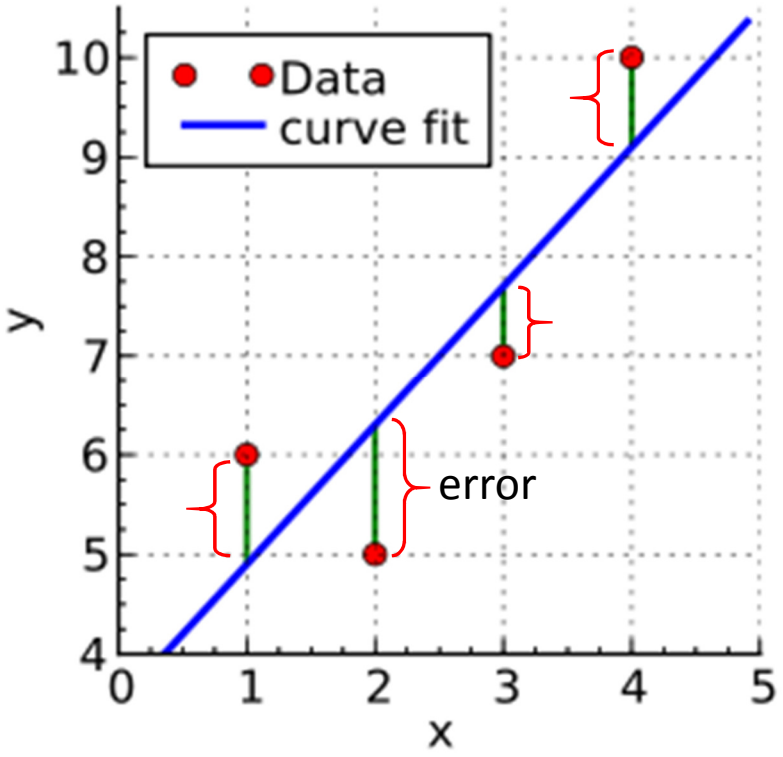
https://youtu.be/BEC0uH1fuGU
A. Géron. Hands-On Machine Learning with Scikit-Learn & TensorFlow. O'Reilly, 2017

Linear Regression

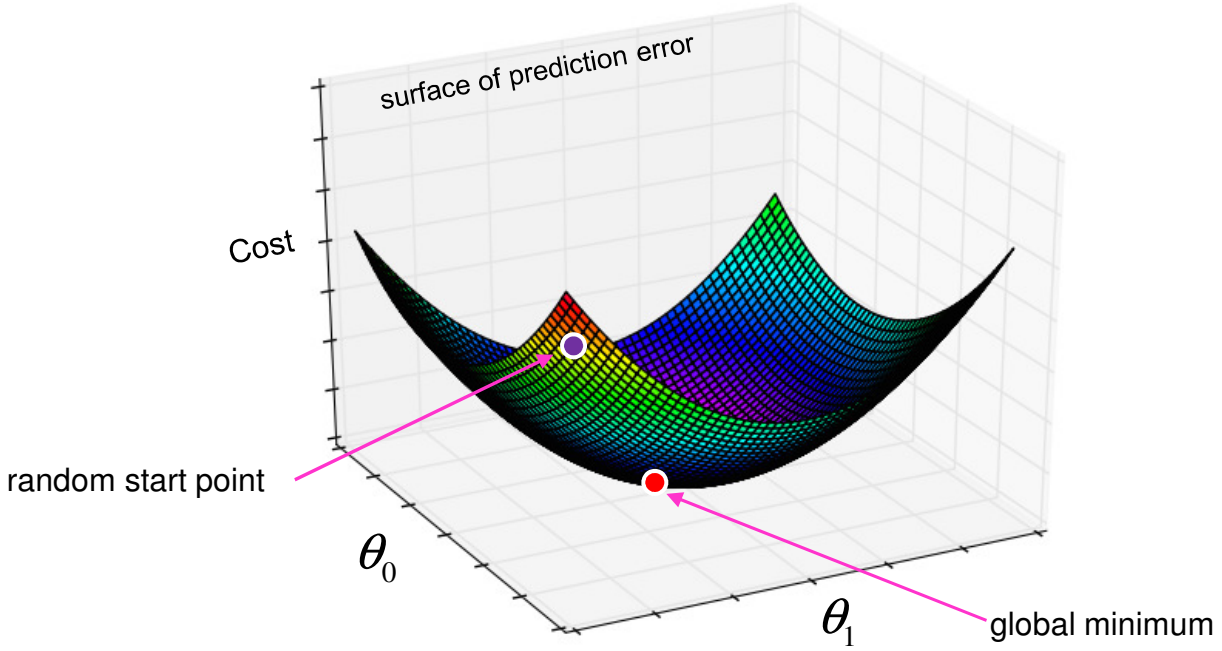
The computation methods for estimating the parameters

- **Gradient descent** is an iterative algorithm to find a minimum of a function
  - The goal is to start at a random point on the error surface, and find out the best parameters values which yield the minimum value of the cost function, i.e., the best parameters values corresponds to the lowest point on the error surface

Cost function: $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(y_i - \widehat{y_i}\right)^2$



The cost function could be visualized as a 3D plot



- The combination of different parameters theta_0 and theta_1 leads to different error (cost function).
- The error surface of the prediction corresponds to the different value of parameters

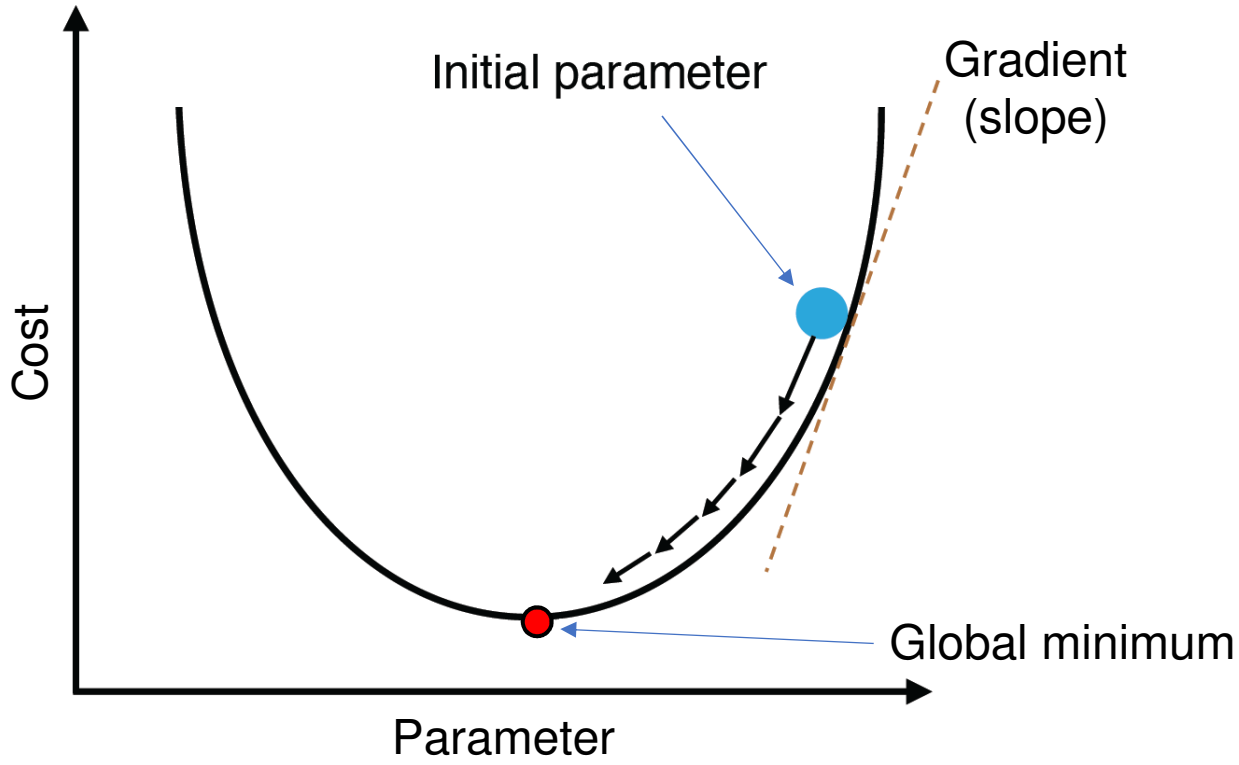
Linear Regression

The computation methods for estimating the parameters

- What's gradient? -- it is just another word for "slope"
  - A gradient is the slope of a function at a specific point.
  - The gradient of a function is the collection of all its partial derivatives into a vector

Cost function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(y_i - \widehat{y_i} \right)^2$$



https://youtu.be/GzphoJOVEcE https://youtu.be/5H7M5Vd3-pk
https://youtu.be/-CG_t8e9Bac https://youtu.be/AXqhWeUEtQU

https://goo.gl/AY3GuX
https://youtu.be/0kns1gXLYg4?t=129

Linear Regression

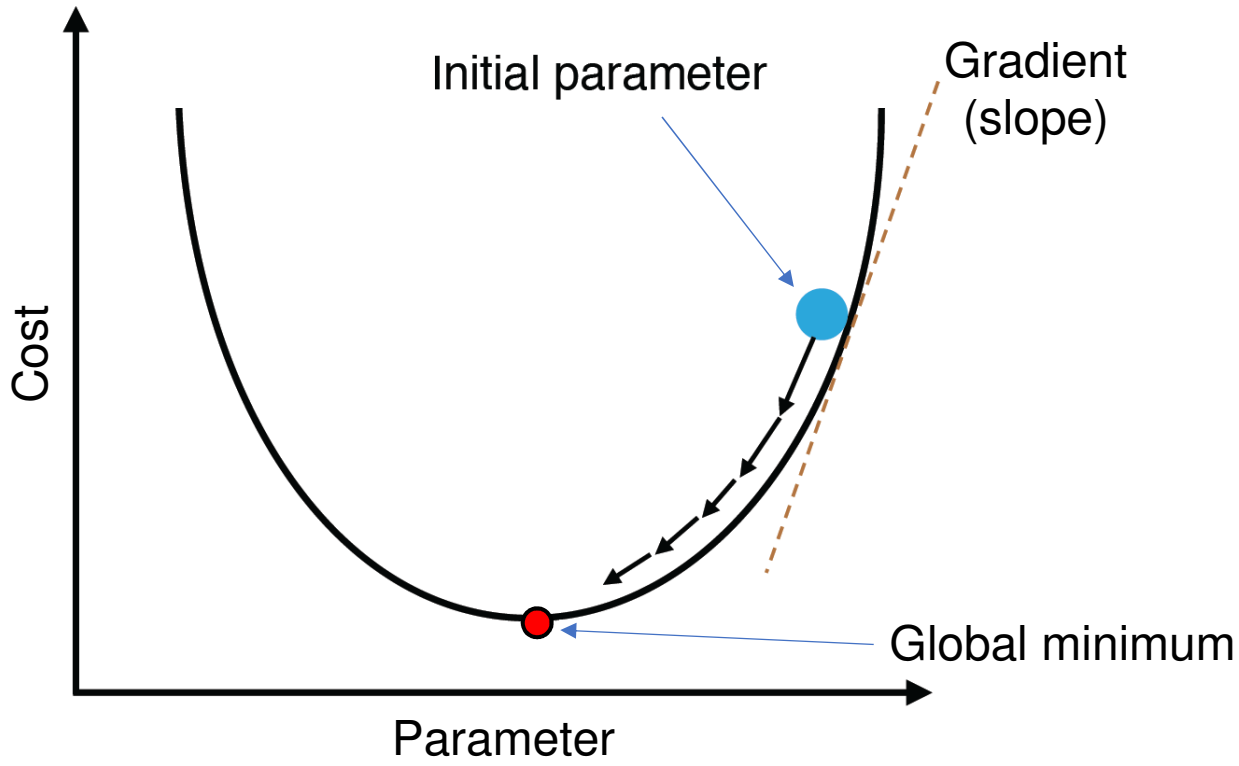
The computation methods for estimating the parameters

- How does Gradient Descent work?
- Consider a cost function that depends on only one parameter
- The parameter is iteratively updated

learning rate (step size) gradient

$$\boldsymbol{\theta}_{\text{new}} := \boldsymbol{\theta}_{\text{old}} - \alpha \cdot \nabla J(\boldsymbol{\theta})$$

1. Pick a value for the learning rate α
2. Start with a random point $\boldsymbol{\theta}$
3. Calculate the gradient $\nabla J(\boldsymbol{\theta})$ at the point $\boldsymbol{\theta}$. Follow the opposite direction of the gradient to get new parameter $\boldsymbol{\theta}_{\text{new}}$
4. Repeat until the cost function converges to the minimum



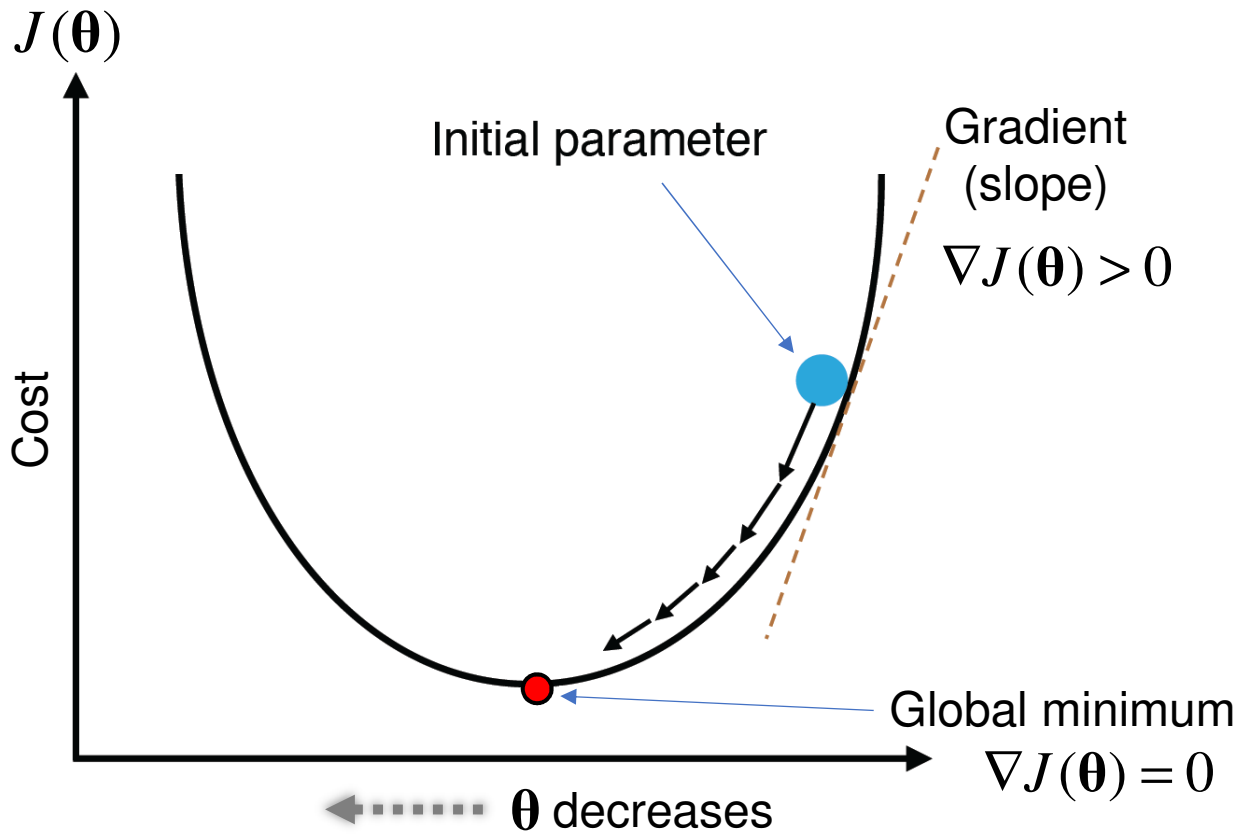
Linear Regression

The computation methods for estimating the parameters

- How does Gradient Descent work?
- Consider a cost function that depends on only one parameter

$$\boldsymbol{\theta}_{\text{new}} := \boldsymbol{\theta}_{\text{old}} - \alpha \cdot \nabla J(\boldsymbol{\theta})$$

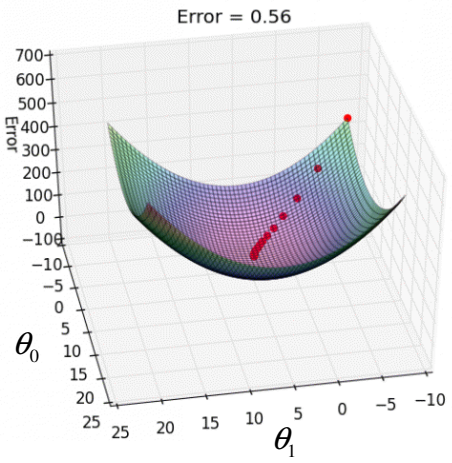
- If the gradient > 0 (positive derivative) parameter decreases $\boldsymbol{\theta}$ ↓
- If the gradient < 0 (negative derivative) parameter increases $\boldsymbol{\theta}$ ↑

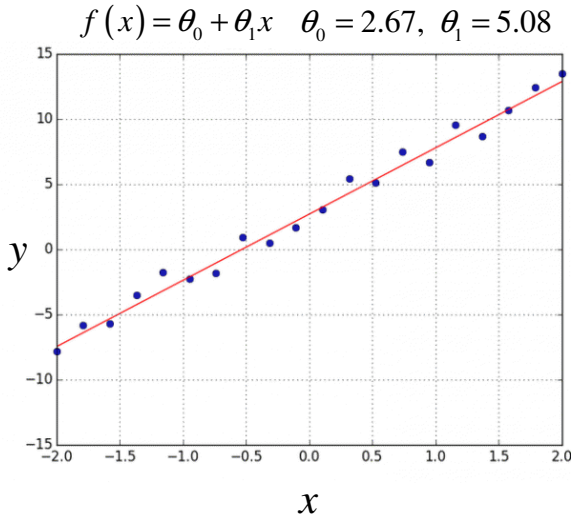


Linear Regression

The computation methods for estimating the parameters

- How does Gradient Descent work?
  - More generally the cost function depends on more than one parameters
  - In Simple Linear Regression, there are 2 parameters θ_0, θ_1





Gradient Descent Algorithm

To minimize the cost function

1. Initialize the parameters with some random values
2. Keep changing these parameters iteratively in such a way it minimizes the cost function $J(\boldsymbol{\theta})$

Repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

Simultaneously update θ_0 and θ_1

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

}

Old position

New position

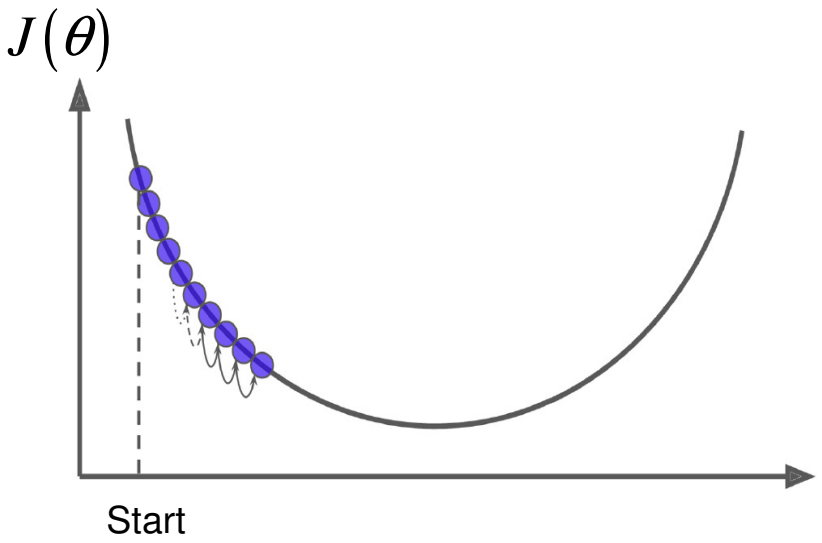
Learning rate

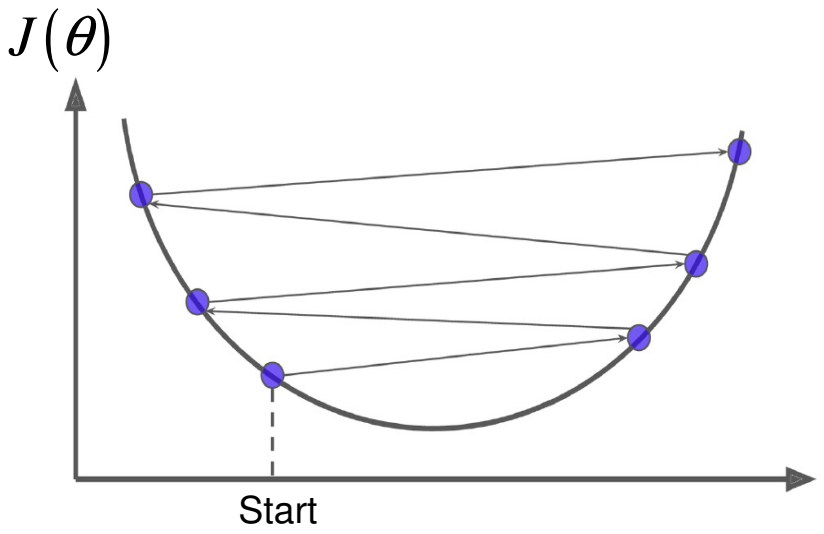
Gradient of function J for θ_1

Linear Regression

The computation methods for estimating the parameters

- How to choose a good learning rate α?
- Finding a good Learning Rate can be tricky
  - If learning rate is too small: slow convergence
  - If learning rate is too large: gradient may not decrease on every iteration; may not converge
  - Sort of art, the most commonly used rates are : **0.001**, **0.003**, **0.01**, **0.03**, **0.1**, **0.3**





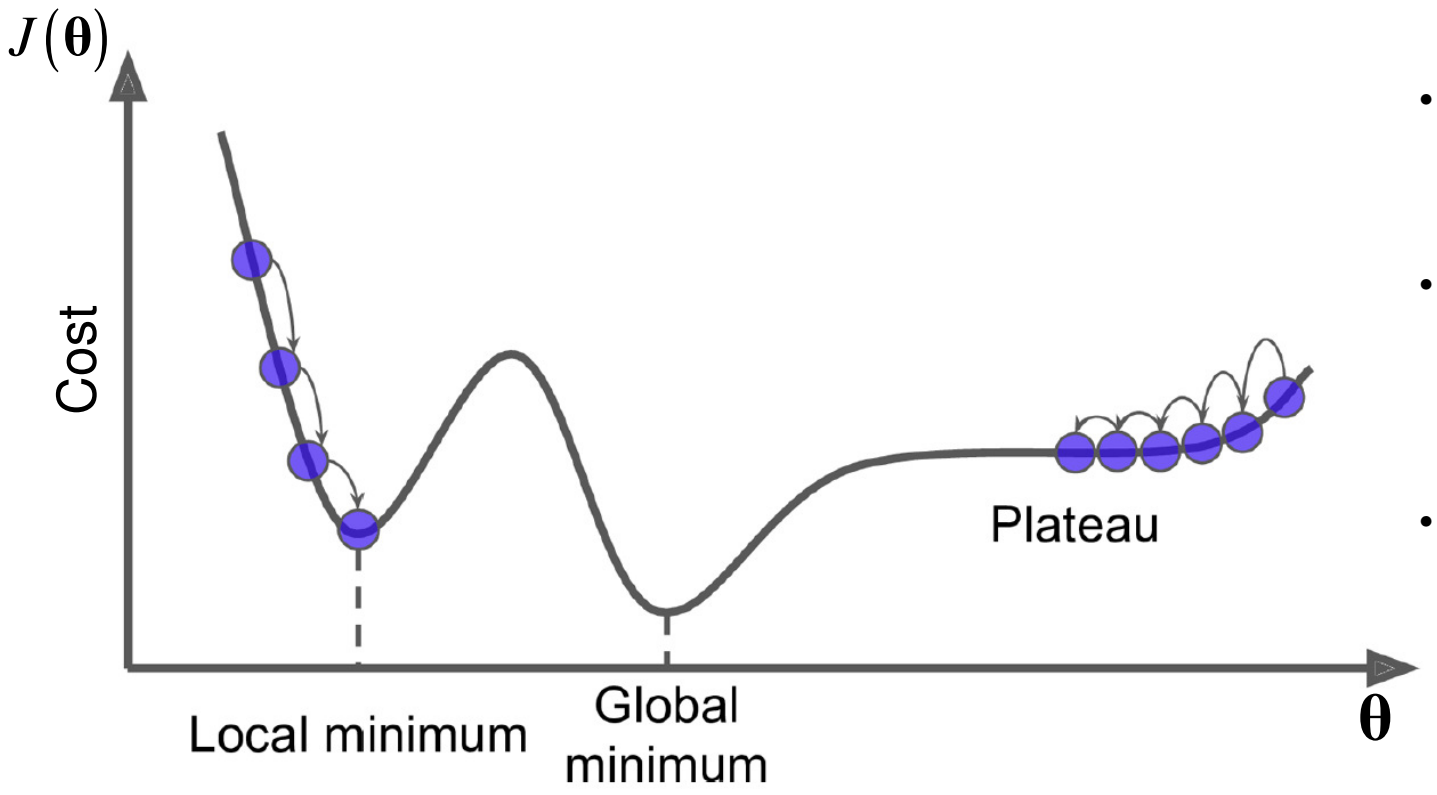
https://youtu.be/CYlR9oYhYuY

A. Géron. Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly, 2017

Linear Regression

The computation methods for estimating the parameters

- Gradient Descent's issue -- not all cost functions look like nice regular bowls. There may be holes, ridges, plateaus, and all sorts of irregular terrains, making convergence to the minimum very difficult



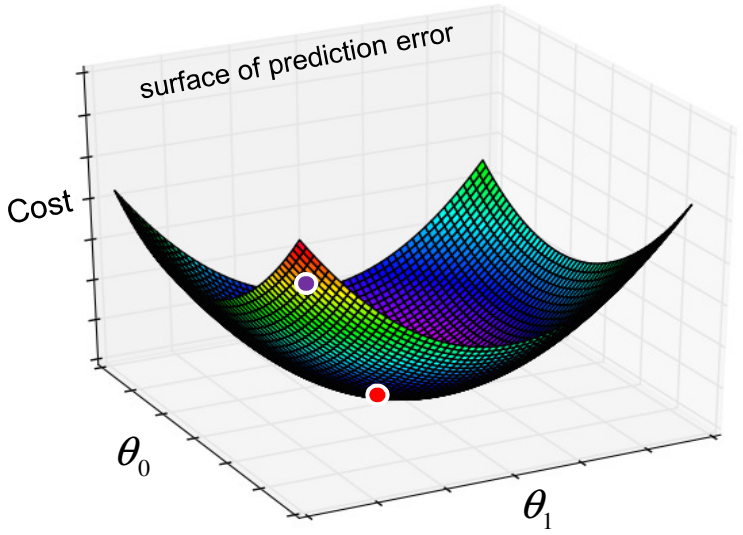
- The gradient (slope) at the local minimum is 0, so the algorithm might get stuck at the local minimum
- If the random initialization starts the algorithm from the left, then it will converge to a local minimum, which is not as good as the global minimum
- If it starts on the right, then it will take a very long time to cross the plateau, and if you stop too early you will never reach the global minimum.

A. Géron. Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly, 2017

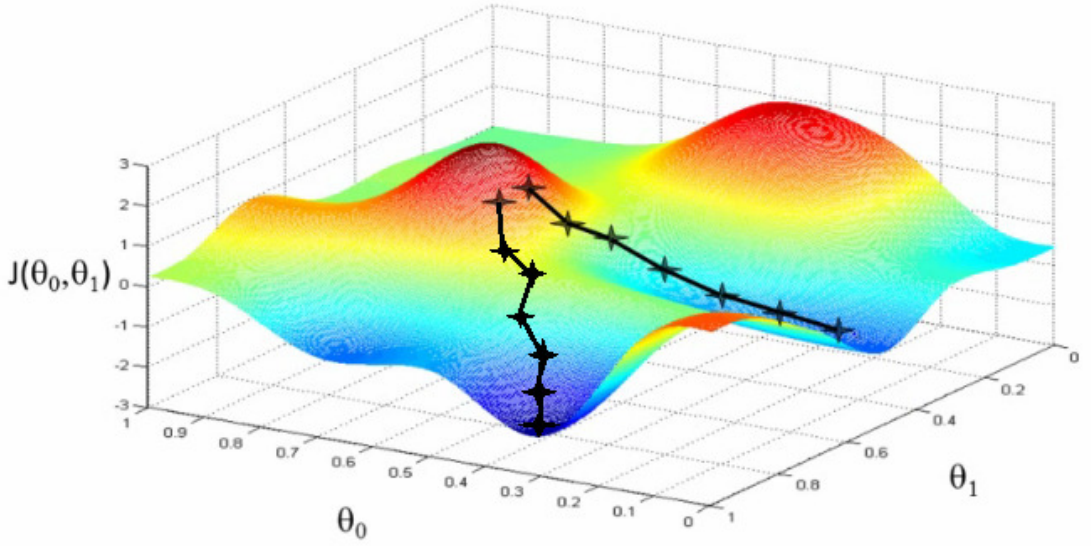
Linear Regression

The computation methods for estimating the parameters

- Gradient Descent's issue
  - If the error surface is convex and with an appropriate learning rate (i.e., not too high), then convergence to the global minima is guaranteed.
  - However, in many real world problems, the error surface is generally highly non-convex, meaning there are a plethora of local minima -- depending on the initially selected parameters, the algorithm could converge to a local minima that is not necessarily the global minima.



Convex Function



Non-Convex Function

A. Géron, Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly, 2017

Linear Regression

Multiple Linear Regression

- Generally one dependent variable depends on multiple factors.
- For example, the price of a house depends on many factors like the neighborhood it is in, size of it, # of bedrooms, attached facilities, distance from the nearest station, distance from the nearest shopping area, etc.

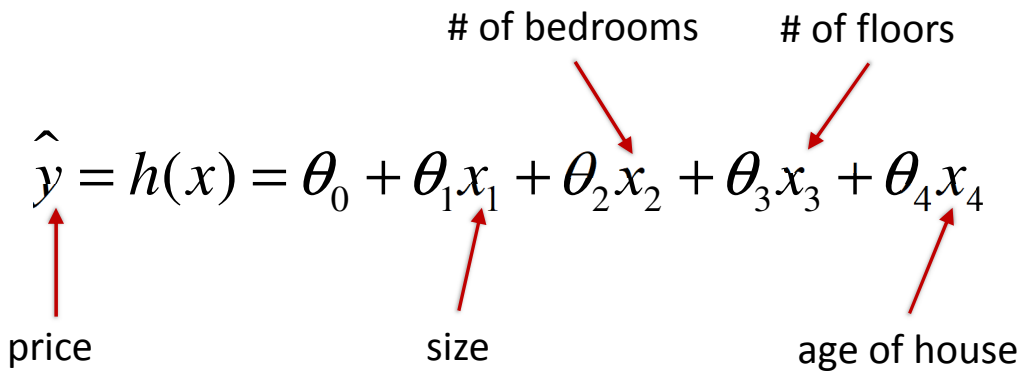
The price of a house may depend on multiple features (variables)

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

Linear Regression

Multiple Linear Regression

- This is quite similar to the previous simple linear regression model, but with multiple independent variables affecting to the dependent variable. Each training sample has an x made up of multiple input values and a corresponding y with a single value. The equation of multivariate linear regression is as follow,
- Simple linear regression: $\hat{y} = h(x) = \theta_0 + \theta_1 x$
- Multiple linear regression: $\hat{y} = h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n$



The price of a house may depend on multiple features (variables)

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

https://onlinecourses.science.psu.edu/stat501/node/382/

Linear Regression

Multiple Linear Regression

- Simplify the equation of multiple linear regression as follows

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n$$

$$x_0 = 1$$

$$\hat{y} = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n$$

In the multiple regression setting, because of the potentially large number of predictors, it is more efficient to use matrices to define the regression model and the subsequent analyses

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \cdots \\ \theta_n \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}$$

$$\hat{y} = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n$$

$$\underbrace{[\theta_0 \;\; \theta_1 \;\; \theta_2 \;\; \cdots \;\; \theta_n]}_{\boldsymbol{\theta}^T} \qquad \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \; \mathbf{x}$$

$$\hat{\mathbf{y}} = \boldsymbol{\theta}^T \mathbf{x}$$

https://youtu.be/Q4GNLhRtZNc
https://onlinecourses.science.psu.edu/stat501/node/382/

Linear Regression

Multiple Linear Regression

- Define the **cost function**:

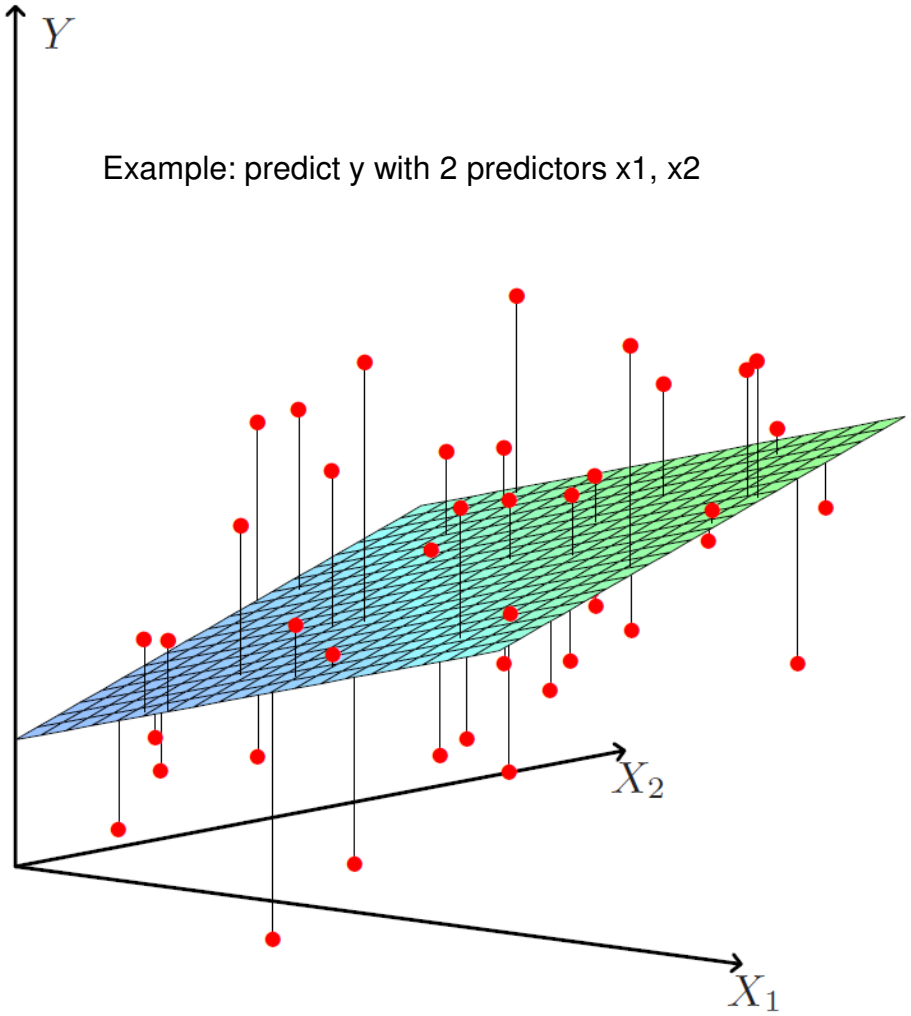
$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(y_i - \hat{y}_i \right)^2$$

where $\hat{y} = h(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n = \sum_{i=0}^{n} \theta_i x_i$

How to find the appropriate parameter $\theta_0, \theta_1, \ldots, \theta_n$

in order to minimize the cost function $J(\boldsymbol{\theta})$?

- Normal Equation
- Gradient Descent



The goal is to find the hyper-plane that "best fits" the training samples, i.e., seek the linear function of X that minimizes the sum of squared residuals (error) from Y

T. Hastie et. al. The Elements of Statistical Learning, Springer 2008

Linear Regression

Multiple Linear Regression

- Use Normal Equation to find the value of the coefficients/parameters $\boldsymbol{\theta}$ that minimizes the cost function

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
…	…	…	…	…	…

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

- The inputs can be represented as an x matrix in which each row is sample and each column is a dimension.
- The outputs can be represented as y matrix in which each row is a sample.

$$\mathbf{x} = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \\ \dots \end{bmatrix}$$

$$\hat{\mathbf{y}} = \mathbf{x}\boldsymbol{\theta}$$

$$\boldsymbol{\theta} = \left(\mathbf{x}^T \mathbf{x}\right)^{-1} \cdot \mathbf{x}^T \cdot \mathbf{y}$$

Value of $\boldsymbol{\theta}$ that minimizes the cost function can be solved by this equation

https://onlinecourses.science.psu.edu/stat501/node/382/

Linear Regression

Multiple Linear Regression

- Use Gradient Descent to find the value of parameters $\boldsymbol{\theta}$ that minimize the cost function

Gradient Descent

for multiple Linear Regression:

Repeat until convergence {

// simultaneously update θ_j for every $j = 0, 1, \ldots, n$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

}

$\nabla_\theta J(\boldsymbol{\theta})$ is the partial derivatives of the cost function with respect to the parameters $\theta_0, \theta_1, \ldots, \theta_n$

Hypothesis: $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$

Parameters: $(\theta_0, \theta_1, \ldots, \theta_n)$

Cost function: $J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

Gradient: $\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$

The jth feature of the ith observation

Gradient Descent:

Repeat until convergence {

// simultaneously update θ_j for every $j = 0, 1, \ldots, n$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

}

Linear Regression

Multiple Linear Regression

- Use Gradient Descent to find the value of parameters $\boldsymbol{\theta}$ that minimize the cost function

Gradient Descent

for multiple Linear Regression:

Repeat until convergence {

// simultaneously update θ_j for every $j = 0, 1, \ldots, n$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) x_j^{(i)}$$

}

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
…	…	…	…	…	…

$$\hat{y} = h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) x_2^{(i)}$$

$$\theta_3 := \theta_3 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) x_3^{(i)}$$

$$\theta_4 := \theta_4 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) x_4^{(i)}$$

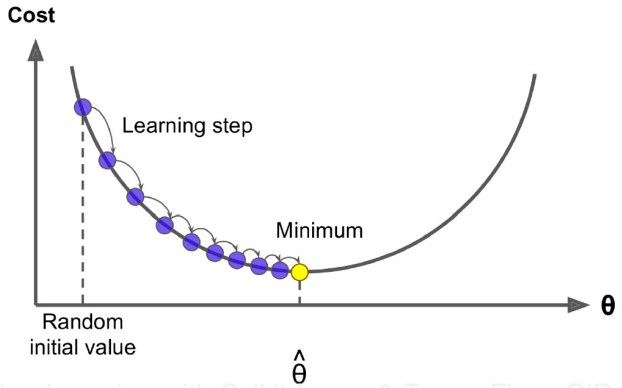
Linear Regression

Summary: Gradient Descent vs. Normal Equation

- Gradient descent is best used when the parameters cannot be calculated analytically (e.g., using linear algebra) and must be searched for by an optimization algorithm
- In Normal Equation approach the values of θ that minimizes the cost function is solved directly rather than iteratively $\theta = \left(\mathbf{x}^{\mathrm{T}}\mathbf{x}\right)^{-1} \cdot \mathbf{x}^{\mathrm{T}} \cdot \mathbf{y}$

Gradient Descent

- Advantages
  - Works well when the number of features is large, e.g., 100,000
- Disadvantages
  - Needs to choose learning rate
  - May need many iterations



Normal Equation

- Advantages
  - No need to choose learning rate
  - No iterations
  - Works well when the number of features is small, e.g., 100
- Disadvantages
  - Need to matrix calculation
  - Slow if the number of features is large

The Normal Equation computes the inverse of a n by n matrix (where n is the number of features). It takes about O(n^(2.4)) to O(n^3) time, if the number of the feature is large, it can be very slow

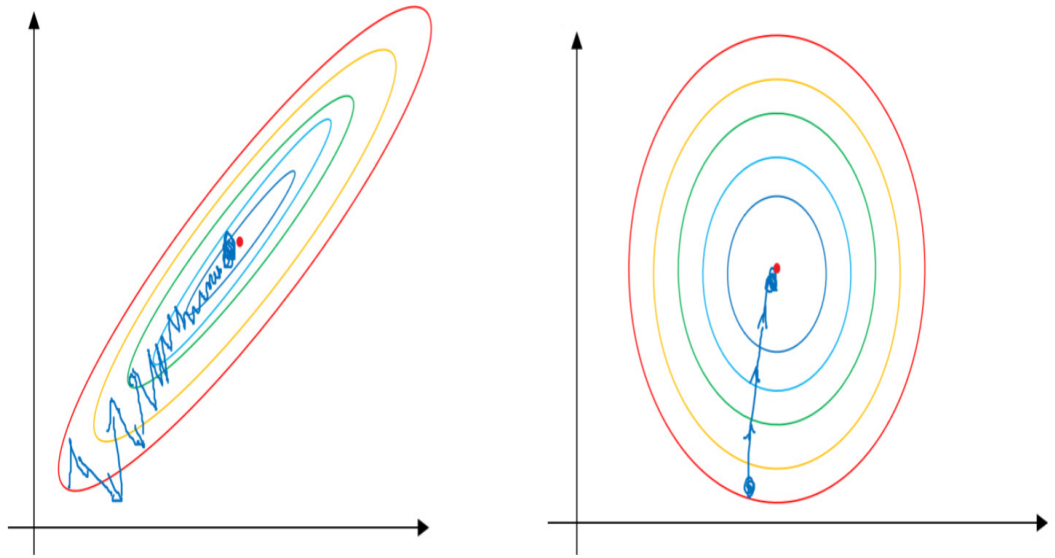
Linear Regression

Feature Scaling

- With few exceptions, Machine Learning algorithms do not perform well when the input numerical attributes have very different scales.
- One of the most important transformations you need to apply to your data is feature scaling.
- When using Gradient Descent, you should ensure that all features have a similar scale, or else it will take much longer to converge.

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

0-2000 1-5



https://goo.gl/tSyRn6

A. Géron. Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly, 2017

Linear Regression

Feature Scaling

- There are two common ways to get all attributes to have the similar scale
  - **Min-Max Scaling (Normalization)**: the data is scaled to a fixed range -- usually from 0 to 1
  - **Standardization**: standardizing the features so that they are centered around 0 with a standard deviation of 1

Min-Max Normalization:

normalized value — original value

$$x_{\text{norm}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

For example:
- x1 = size (100-2000)
- x2 = number of bedrooms (1-5)

$$\begin{cases} x_1 = \dfrac{\text{size} - 100}{1900} \\ x_2 = \dfrac{\#\text{ of bedrooms} - 1}{4} \end{cases}$$

Z-score Normalization (or Standardization):

standard score — original value — mean — standard deviation

$$z = \frac{x - \mu}{\sigma}$$

- Standardization involves rescaling the features such that they have the properties of a standard normal distribution with a mean of zero and a standard deviation of one 1.
- Unlike min-max scaling, standardization does not bound values to a specific range, which may be a problem for some algorithms
  - e.g., neural networks often expect an input value ranging from 0 to 1.
- Standardization is much less affected by outliers

A. Géron. Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly, 2017

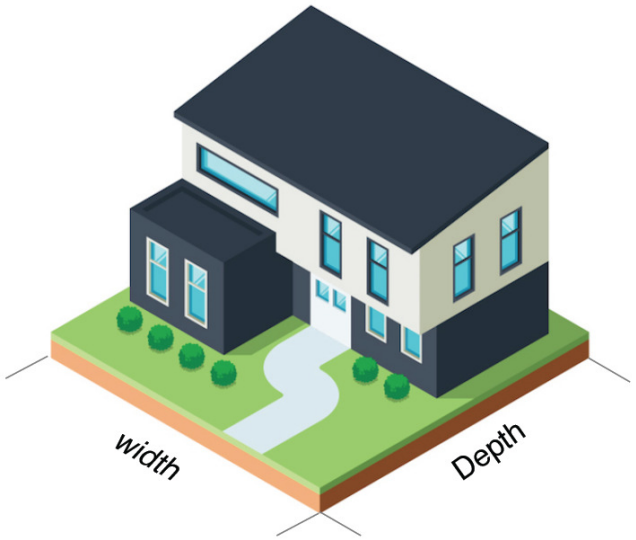
Linear Regression

Create new feature(s)/attribute(s)

- Feature/Variable creation is a process to generate a new variable(s)/feature(s) based on existing variable(s).

$$h_\theta(\theta) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

House's Width Depth



simplified as

Area x

x = Width × Depth

// combine house's width and
// depth as new feature "area"

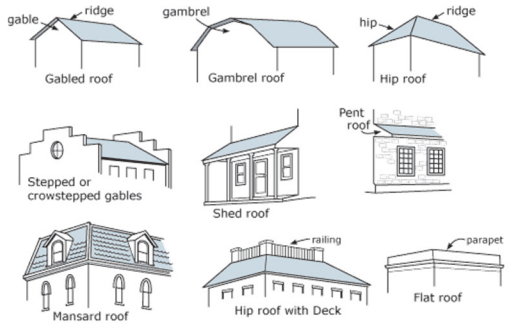
$$h_\theta(\theta) = \theta_0 + \theta_1 x$$

// use one attribute "area" instead of two attributes

Linear Regression

If the inputs are categorical features instead of numbers

- Many ML algorithms cannot work with categorical data directly. The categories must be converted into numbers



$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \ldots + \theta_n x_n$$

Roof Types

Suppose x4 is the roof types (Flat Roof, Gable Roof, Gambrel Roof, Manscard Roof, Shed Roof, etc.), linear regression algorithms cannot work with such categorical data directly.

One Hot Encoding

- Often features are not given as continuous values but categorical. For example, a person's gender is not number. We need to represent the category values as number so that they can be used in our linear regression algorithm. We can use one-hot encoding to convert categorical features into numbers.
- One hot encoding is assigning 1 to working feature and 0's to other idle feature. Mathematically, one hot encoding produces a balanced matrix.

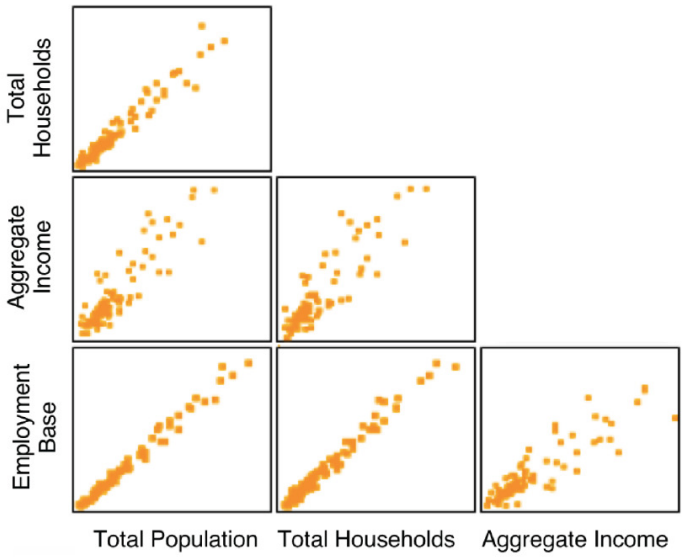
- Red → [1, 0, 0]
- Yellow → [0, 1, 0]
- Green → [0, 0, 1]

http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-categorical-features
https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/
http://www.johnriebli.com/roof-types--house-styles.html

Linear Regression

Multicollinearity

- Multicollinearity is one of the most important issues in regression analysis, as it produces unstable coefficients' estimates and makes the standard errors severely inflated
- "Multicollinearity" refers to predictors that are correlated with other predictors in regression
  - Multicollinearity occurs when your model includes multiple factors that are correlated not just to your response variable, but also to each other, i.e., it results when you have factors that are a bit redundant



Problems caused by multicollinearity:

- The coefficient estimates can swing wildly based on which other independent variables are in the model. The coefficients become very sensitive to small changes in the model.
- Multicollinearity reduces the precision of the estimate coefficients, which weakens the statistical power of your regression model. You might not be able to trust the p-values to identify independent variables that are statistically significant.

Multicollinearity is NOT a big concern in Machine Learning:

- Multicollinearity makes it hard to interpret your coefficients, and it reduces the power of your model to identify independent variables that are statistically significant.
- Although multicollinearity affects the coefficients and p-values, but it does not affect the overall fit or the predictions of the model. In most ML applications, we don't care about coefficients themselves, just the loss of our model predictions. If your primary goal is to make predictions, and you don't need to understand the role of each independent variable, you don't need to reduce severe multicollinearity.

Polynomial Regression

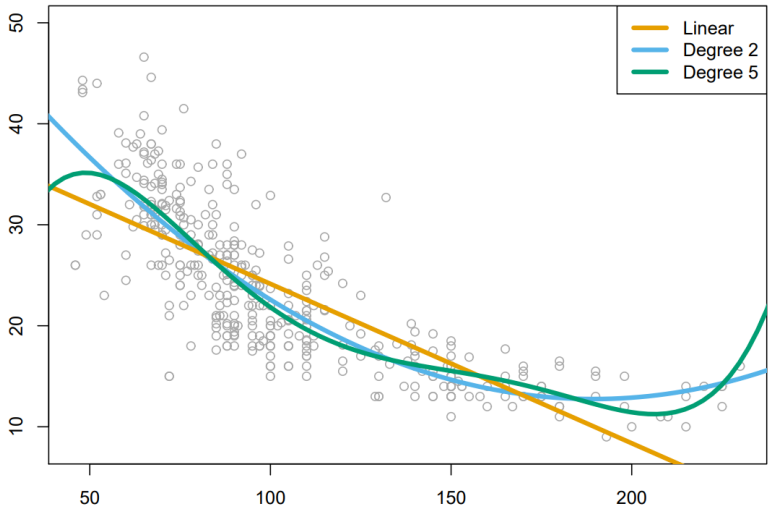
What if your data is actually more complex than a simple straight line?

- Use a linear model to fit nonlinear data -- add powers of each feature as new features, then train a linear model on this extended set of features.

$$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \ldots + \theta_n x^n$$

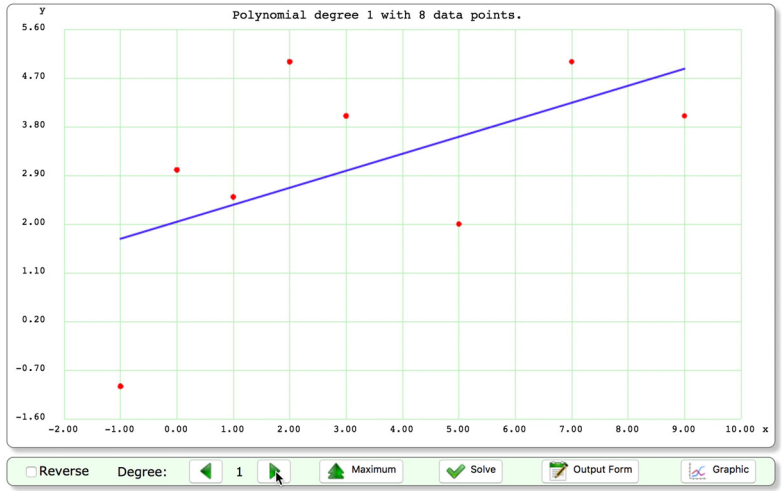
n is called the degree of polynomial

- $n = 1$, it is a **straight line** $\hat{y} = \theta_0 + \theta_1 x$
- $n = 2$, it is **a quadratic** $\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2$
- $n = 3$, it is **cubic** $\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$
- $n = 4$, it is **quartic** $\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$



The polynomial models can be used to approximate a complex nonlinear relationship

Polynomial Regression Demo

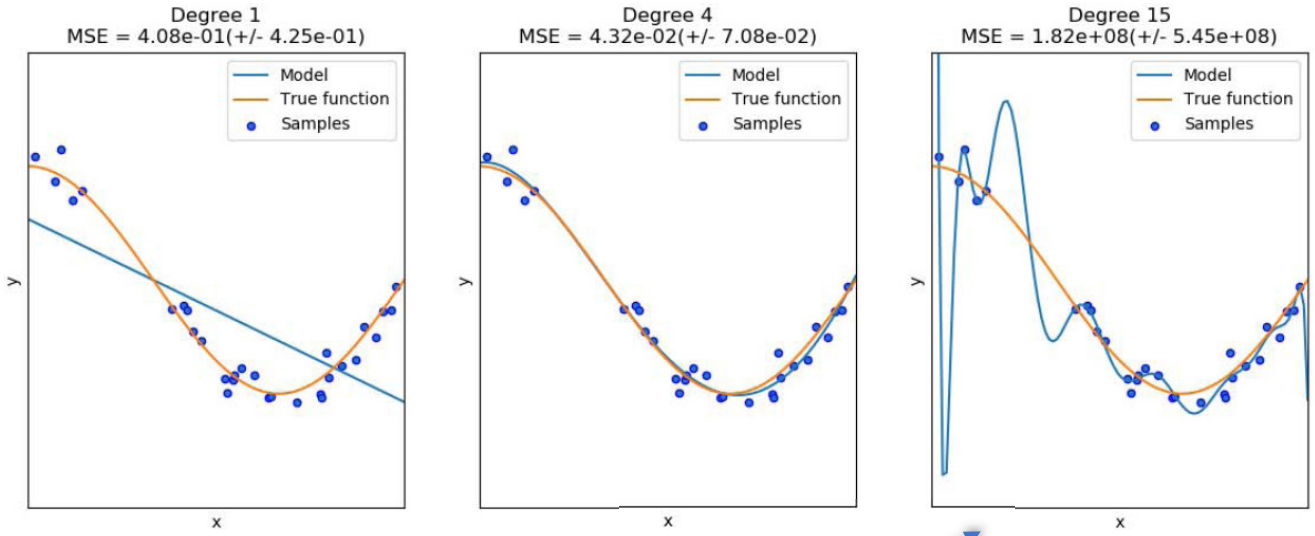


https://arachnoid.com/polysolve/

Polynomial Regression

Overfitting

- The more parameters or features in your model, the more freedom it has to fit the data. However, the model is also more prone to overfitting the training data.



Underfitting

Overfitting

overfitting is due to higher order of polynomials

Plots of polynomials having various orders n
The real function to be approximated a part of the cosine function

- A linear function (polynomial with degree 1) is not sufficient to fit the training samples. This is called **underfitting**.
- A polynomial of degree 4 approximates the true function almost perfectly.
- Higher-degree model will **overfit** the training data, i.e., it learns the noise of the training data.

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

Polynomial Regression

How to Reduce/Avoid Overfitting?

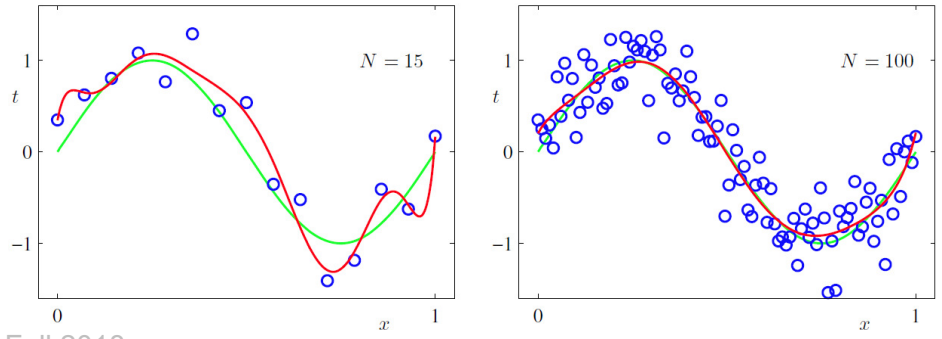
- Reduce model's complexity
  - Reduce the number of features
    - It is prone to overfitting if there are lots of features but with very little training data.
    - Throw away some of the features → $\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \ldots + \theta_n x^n$
  - Reduce the degree of polynomial
    - Try models with lower degree → $\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5$
- Use L1/L2 regularization

$$J(\theta) = \text{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

  - Ridge regression -- a regularized version of Linear Regression. It uses L2-norm penalty. Ridge regression includes all of the features in the model. As the value of alpha/Lambda increases, the coefficient shrinks.
  - Lasso regression -- a regularized version of Linear Regression. It uses L1-norm penalty. L1 will push certain weights to be exactly 0. Lasso Regression automatically performs feature selection and outputs a sparse model

$$J(\theta) = \text{MSE}(\theta) + \alpha \sum_{i=1}^{n} |\theta_i|$$

- Get more data
  - For a given model complexity, the over-fitting problem become less severe as the size of the data set increases

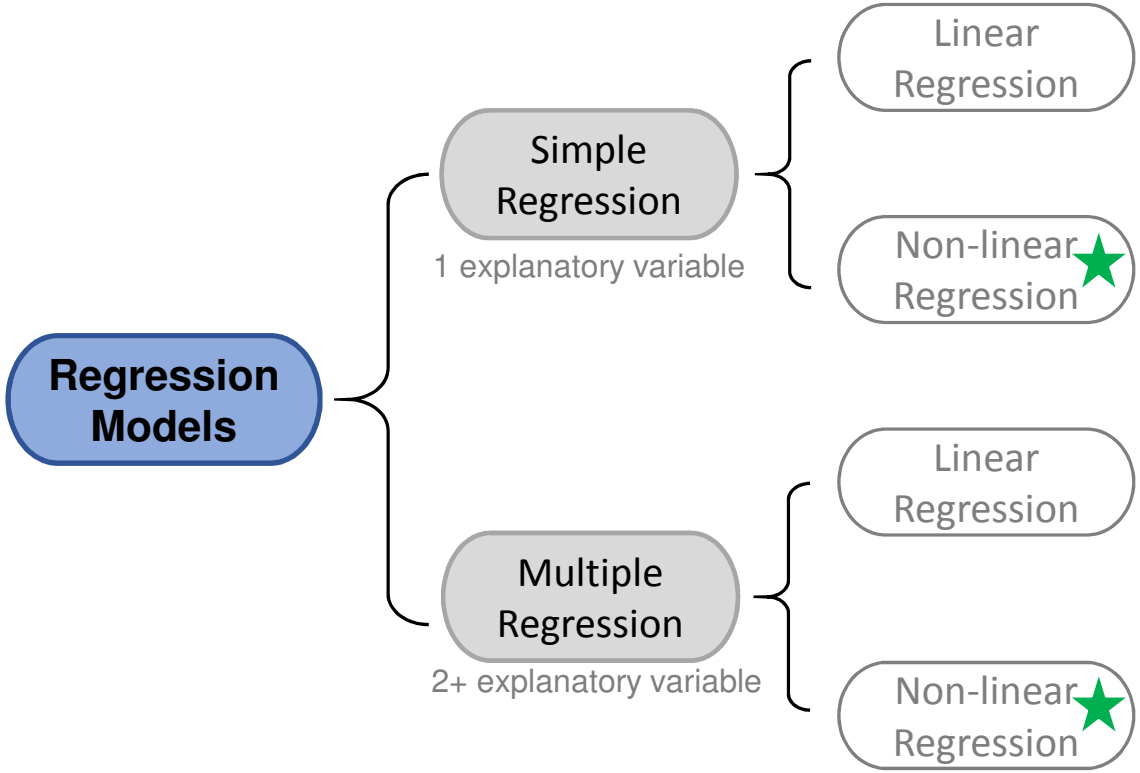


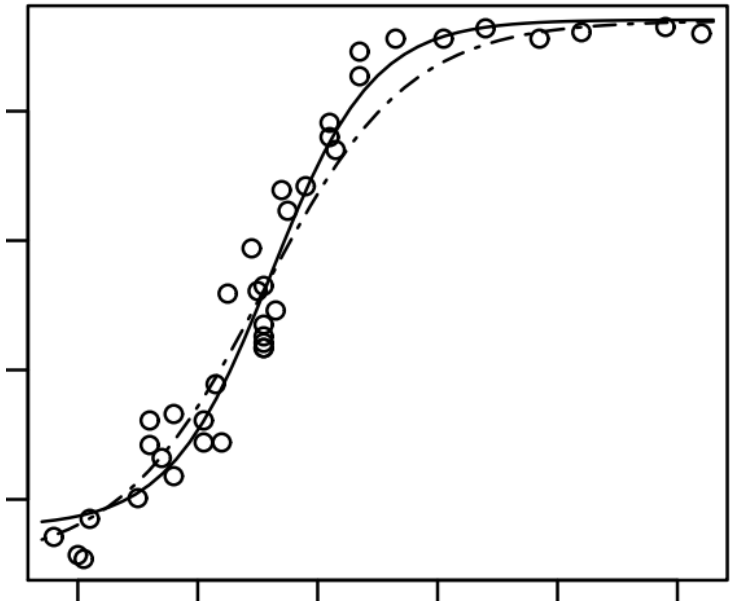
Plots of the solutions obtained by minimizing the sum-of-squares error function using the $M = 9$ polynomial for $N = 15$ data points left plot) and $N = 100$ data points (right plot). Increasing the size of the data set reduces the over-fitting problem

C. Bishop. Pattern Recognition and Machine Learning, Springer, 2006

Nonlinear Regression

- Nonlinear regression is appropriate when the relationship between the dependent and independent variables is not intrinsically linear



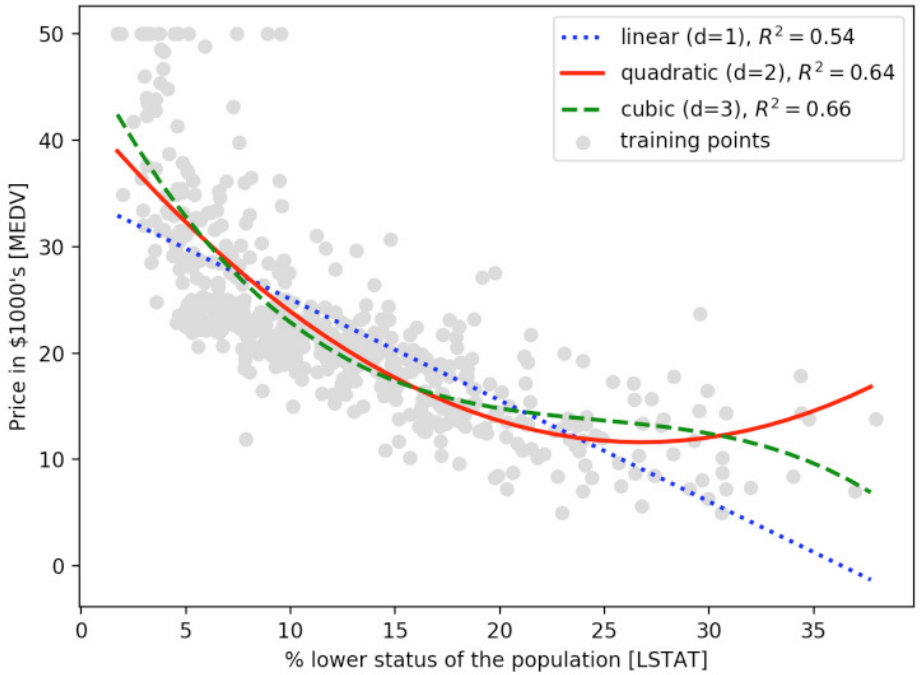


Nonlinear Regression

What's nonlinear regression?

- A method of finding a nonlinear model of the relationship between the dependent variable and a set of independent variables.
- Unlike traditional linear regression, which is restricted to estimating linear models, nonlinear regression can estimate models with arbitrary relationships between independent and dependent variables

Polynomial regression is NOT considered as non-linear regression

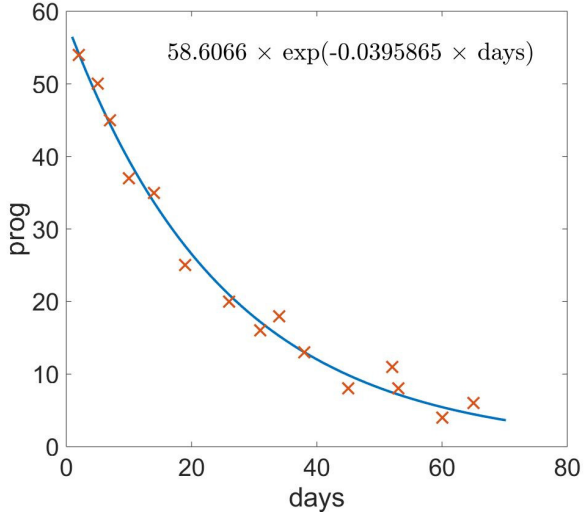


"Nonlinear" refers to a fit function that is a nonlinear function of the parameters

e.g., a nonlinear function of the fitting parameters: $\hat{y} = \dfrac{b_0 x b_1}{x + b_2}$

e.g., exponential regression model:

$$\hat{y} = \theta_0 + \theta_1 \exp\left(\theta_2 x + \ldots \theta_{p+1} x\right)$$



Some other examples of nonlinear models:

$$\hat{y} = \frac{e^{\theta_0 + \theta_1 x}}{1 + e^{\theta_0 + \theta_1 x}}$$

$$\hat{y} = \frac{\theta_0 + \theta_1 x}{1 + \theta_2 e^{\theta_3 x}}$$

$$\hat{y} = \theta_0 + \left(0.4 - \theta_0\right) e^{-\theta_1 (x-5)}$$

https://onlinecourses.science.psu.edu/stat501/node/372/ https://goo.gl/2HQ5xN

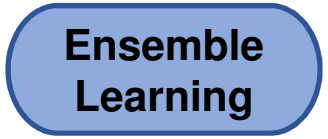
Ensemble Methods

Ensemble methods

- Averaging methods
  - Bagging methods, Random Forest, …
- Boosting methods
  - Ada Boost, Gradient Tree Boosting, …

Ensemble Methods (Random Forest)

Ensemble Learning



Bagging

e.g., **Random Forest**

Independently build several estimators/learners and then average their predictions -- the combined learners on average is usually better than any of the single base learner because its variance is reduced.

Boosting

e.g., **Gradient Boosting**

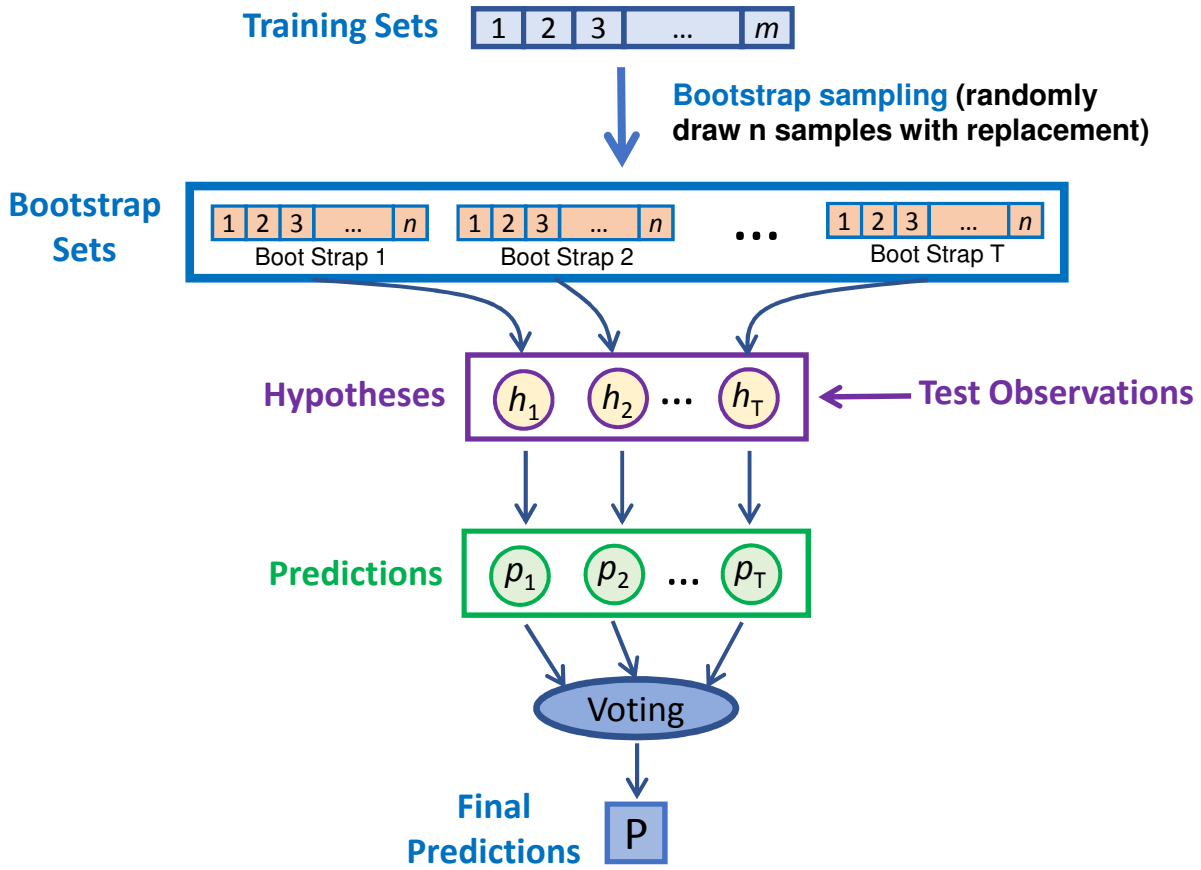
Sequentially build base learners and one tries to reduce the bias of the combined learner -- to combine several weak models to produce a powerful ensemble.

http://scikit-learn.org/stable/modules/ensemble.html

Ensemble Learning

How Bagging works

- Use bootstrapping to train multiple learners and then aggregate the predictions.
- Each individual predictor has a higher bias than if it were trained on the original training set, but aggregation reduces both bias and variance.



1. **Create random subsets of data by bootstrap sampling.**
   - Different training data subsets are randomly drawn – with replacement – from the entire training dataset.
   - By sampling with replacement some observations may be repeated in each new training data set.

2. **Train base learners on each bootstrap sample separately.**

3. **Average predictions from multiple base learners.**
   - The final result is to aggregate the outputs of base learners by
     - Majority voting for classification
     - Averaging for regression

- Various learners/models are built in parallel.
- Bagging is more useful for nonlinear models
- Often used with tree – an extension is random forests.

Image source: http://manish-m.com/?p=794

Random Forest

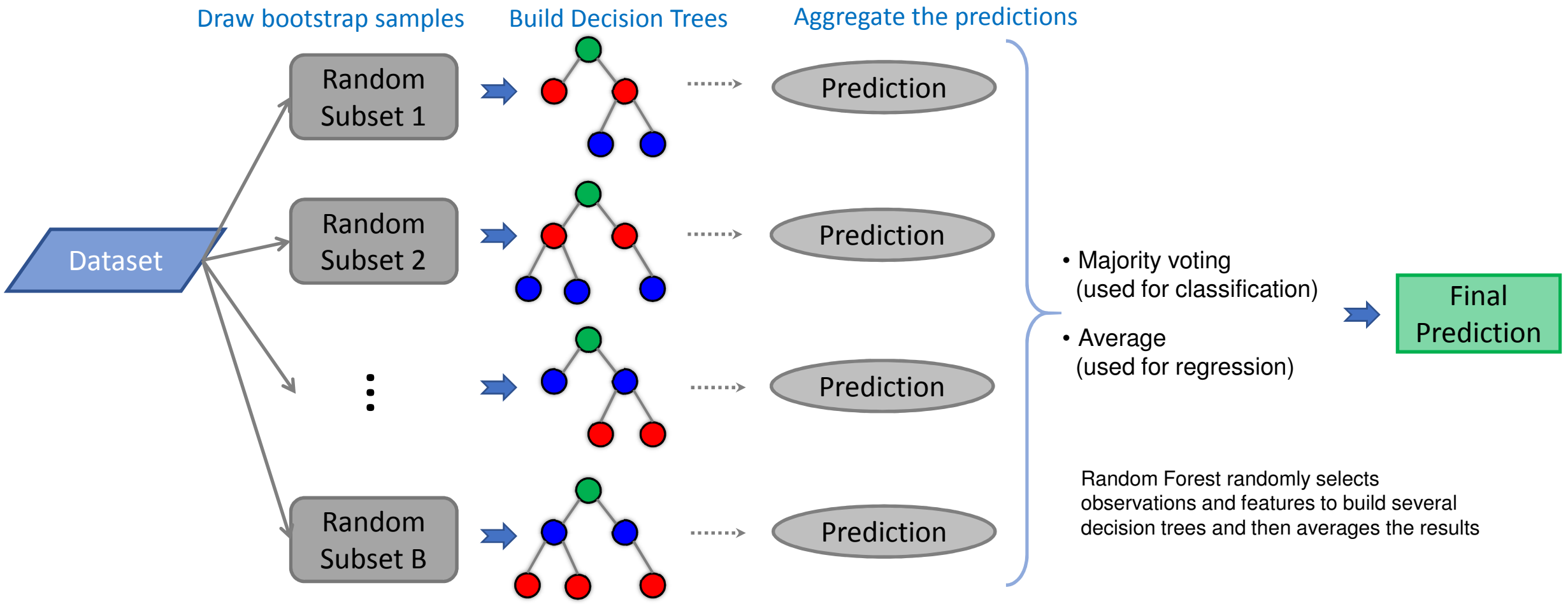
- The essential idea of bagging is to average many noisy but approximately unbiased models, and hence reduce the variance
- **Random Forest** is an extension of bagging
  - One of the most used algorithms that combines the advantages of Bagging and Decision tree
  - An ensemble of Decision Trees, generally trained via the bagging method
  - Implemented in a variety of packages

In many cases an aggregated answer is better
than an experts answer --> wisdom of the crowd

A. Géron, Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly, 2017

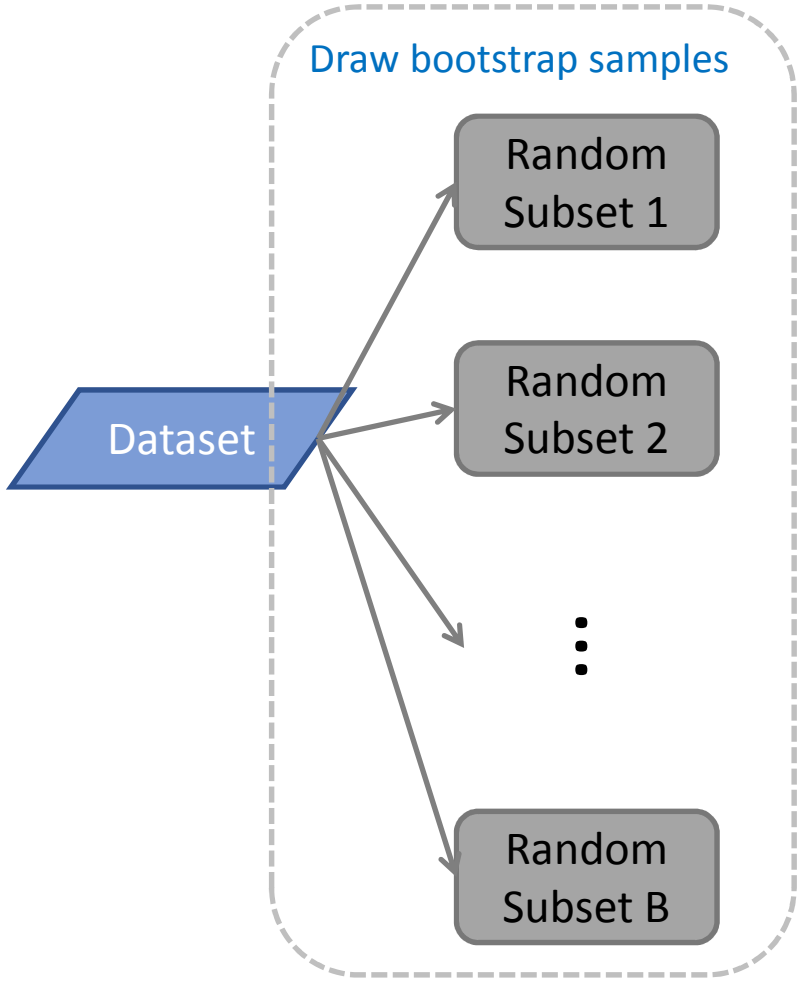
Random Forest

- Random Forest = Bagging + Fully-grown CART Decision Tree
  - Random Forest builds multiple decision trees and merges them together to get a more accurate and stable prediction; it can be used for both classification and regression problems.



Random Forest

- **Step 1: draw bootstrap samples**
  - Use the bootstrap method to produce training subsets. Around 2/3 of the dataset will be drawn into the random subsets, the rest of data will be in “out of the bag”.



Randomly select samples by **sampling with replacement**

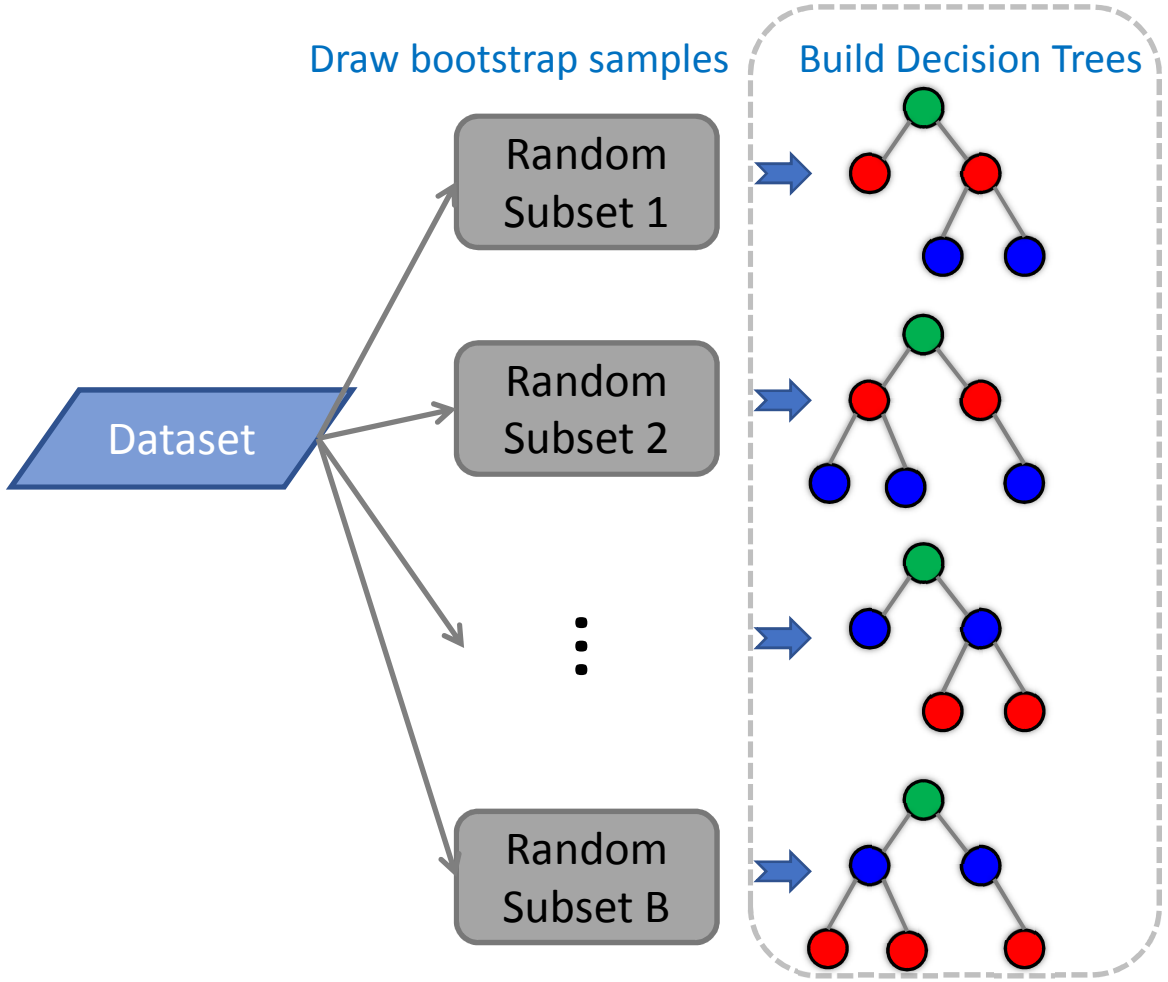
- Use bootstrap sampling to create B subsets of the dataset
  - These sampled subset should be about 2/3 of the total data. They will be the training set for growing the tree.
  - The remaining data are never drawn called “out of the bag” samples. They will be used in evaluation.

- With bootstrap sampling (i.e., sampling with replacement), some instances may be sampled several times for any given predictor, while others may not be sampled at all. After N draws with replacement, the probability of never have been selected is about 37%.

A predictor can be evaluated on the out-of-bag instances, the ones that are never seen by the predictor during training --> no need for a separate validation set or cross-validation. --> the traditional approach to reserve additional data for validation is no longer necessary

Random Forest

- **Step 2: generate multiple decision trees based on bootstrap samples**
  - Random forest builds decision trees using the bootstrapped dataset, but only use a random subset of variables/features --> increases diversity in the forest leading to more robust overall predictions.



Fit decision trees to different bootstrap samples

- When growing a decision tree, select a random sample of m < p predictors to consider in each step.
  - Add extra randomness: instead of searching for the very best feature when splitting a node, Random Forest algorithm searches for the best feature among a random subset of features (e.g., using random thresholds), i.e., if there are p input features, a number m < p is specified such that at each node, m variables are selected at random out of the p.
  - The optimal m is usually around p^(1/2), but this is tunable

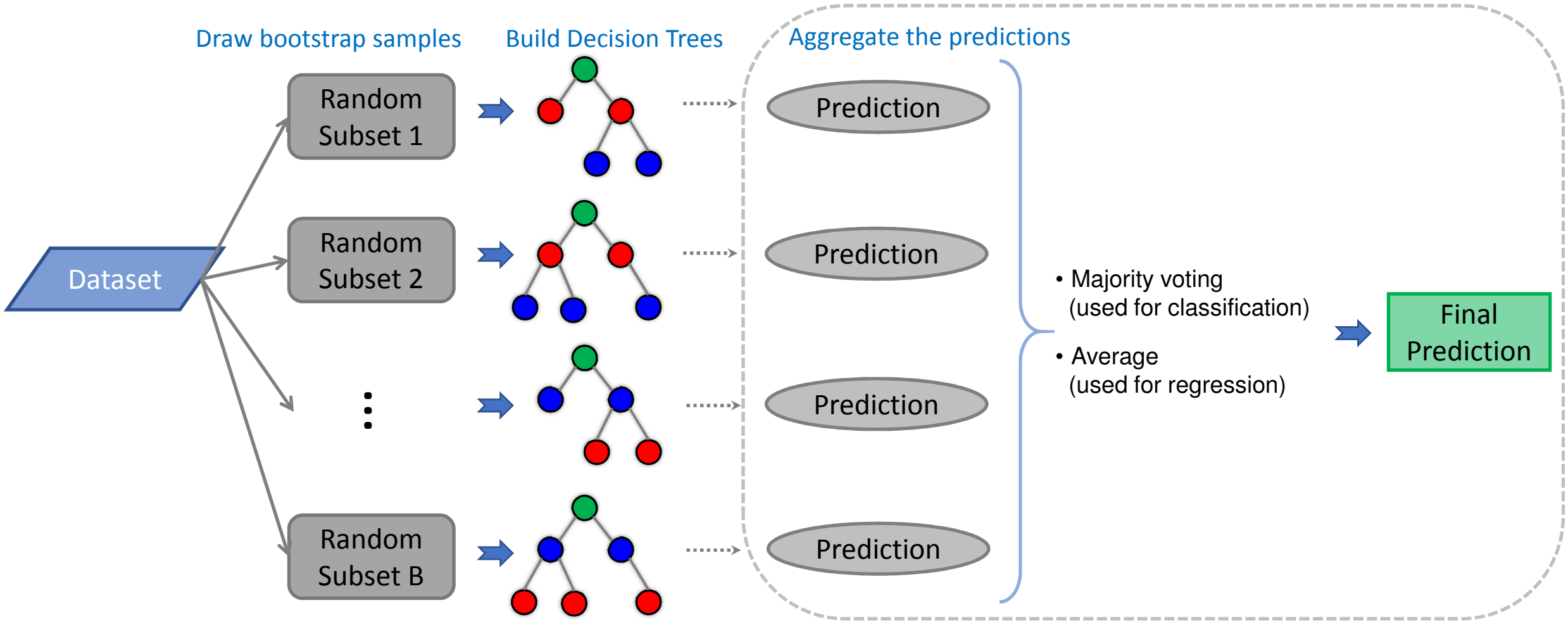
Extra Trees

- Calculate the best split based on these m variables in the training set.
- Grow each tree to the largest extent possible without pruning.

This will lead to very different (or "uncorrelated") trees from each sample.

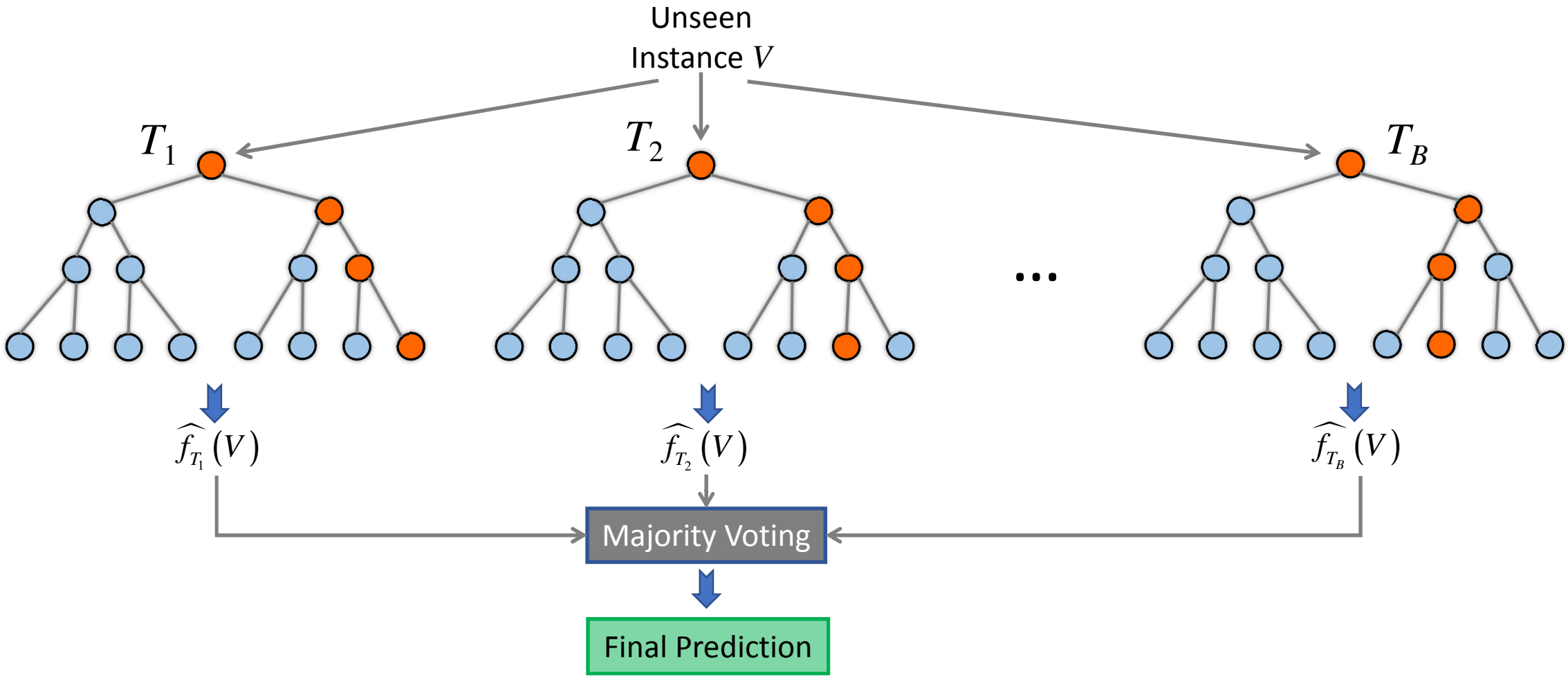
Random Forest

- **Step 3: predict new data by aggregating the predictions of the trees**
  - Use “majority votes” for classification, and “average” for regression problem.



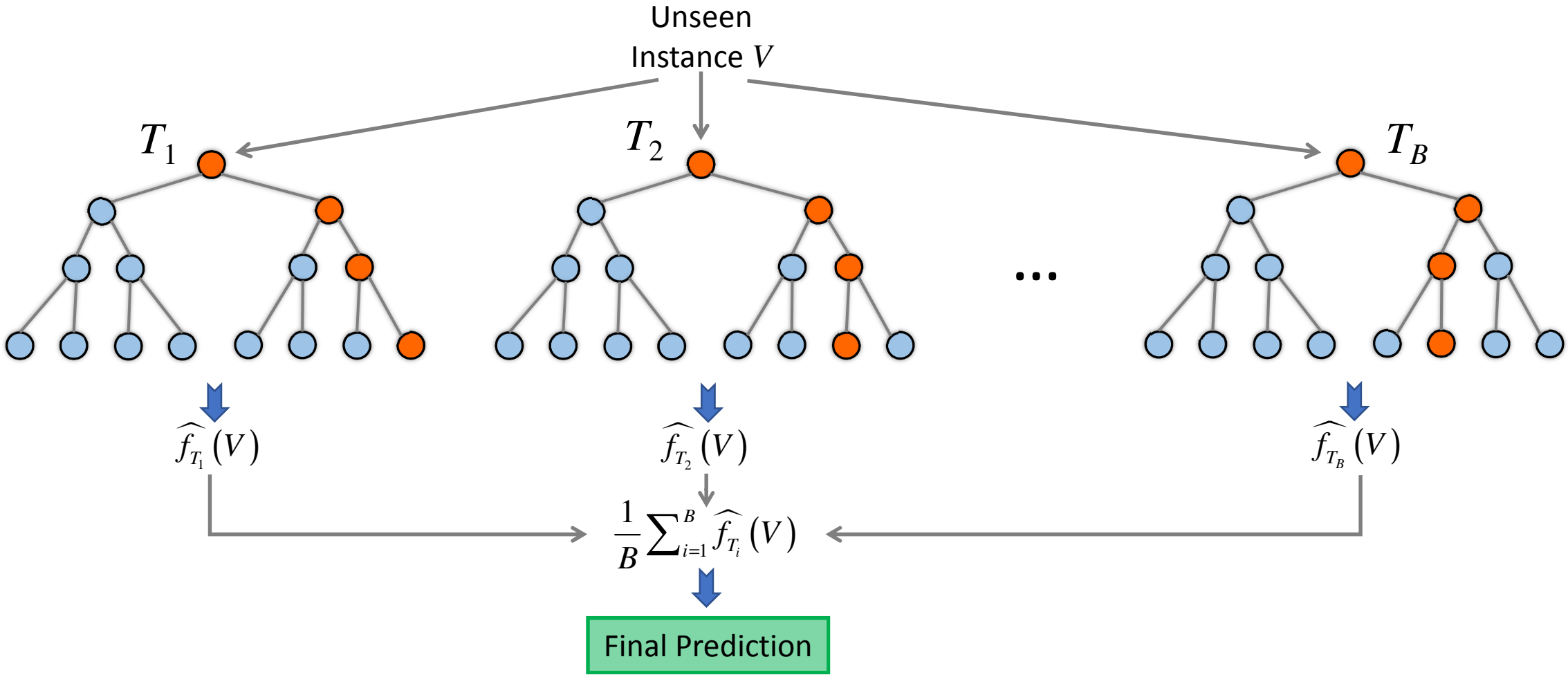
Random Forest

- Use Random Forest to classify data
  - After training, a set of tree $\{T_1, T_2, \ldots, T_B\}$ can be obtained to predict the classes of the unseen samples by taking the majority vote from all individual classification trees.



Random Forest

- Use Random Forest for regression
  - After training, a set of tree $\{T_1, T_2, \ldots, T_B\}$ can be obtained to predict the classes of the unseen samples by taking the average from all individual regression trees.



http://www.saedsayad.com/decision_tree_reg.htm

The Elements of Statistical Learning (2nd Ed.) by T. Hastie et al., Springer

Random Forest

Advantages: Random Forest in general is a fast, simple and flexible algorithm

- Random Forest is less likely to overfit
  - Unlike decision tree algorithms, overfitting is less of an issue with Random Forests
  - No need to pruning trees
- Random Forest algorithms can be grown in parallel
  - Individual decision trees can be trained in parallel
- Random Forest has higher classification accuracy
  - Because of its simplicity, one can easily build a decent model without too much tuning
- Able to deal with the missing value and maintain accuracy in case of missing data
  - Random forest can handle the missing value to some extent
  - It is also not very sensitive to outliers
- Help data scientists save data preparation time
  - It does not requires the input preparation
  - It can handle thousands of input variables without variable deletion

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Random Forest

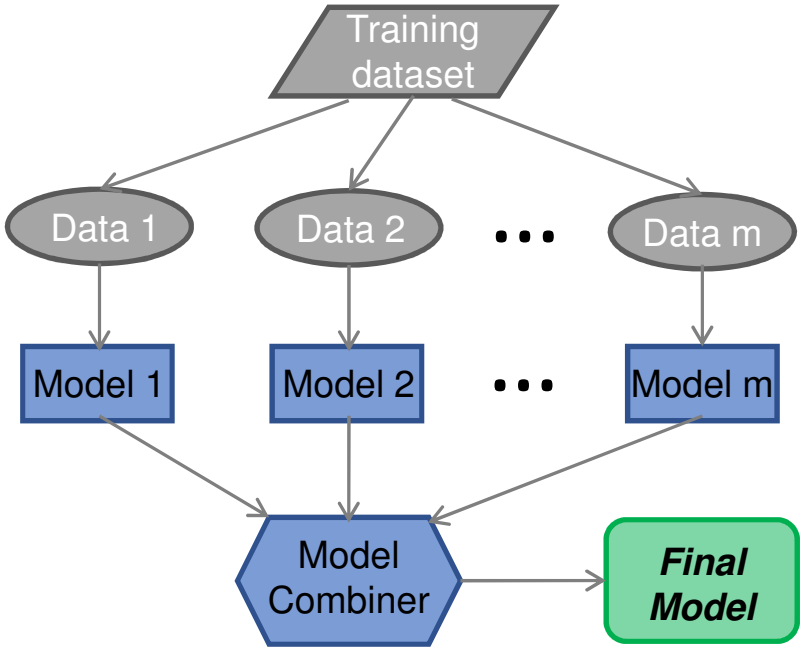
Disadvantages

- Large number of decision trees in the random forest can slow down the algorithm
  - In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation
- Good job at classification but not as good as for regression
  - Good at regression problem as it does not give precise continuous nature predictions
  - In case of regression, it does not predict beyond the range in the training data
- Lack of interpretability -- random forest is more like a black-box approach
  - Random forest is intuitively easy to understand, but it is difficult to get an insight as to what the algorithm actually does
  - Analyzing random forest theoretically is difficult

Ensemble Methods (AdaBoost)

Ensemble Learning

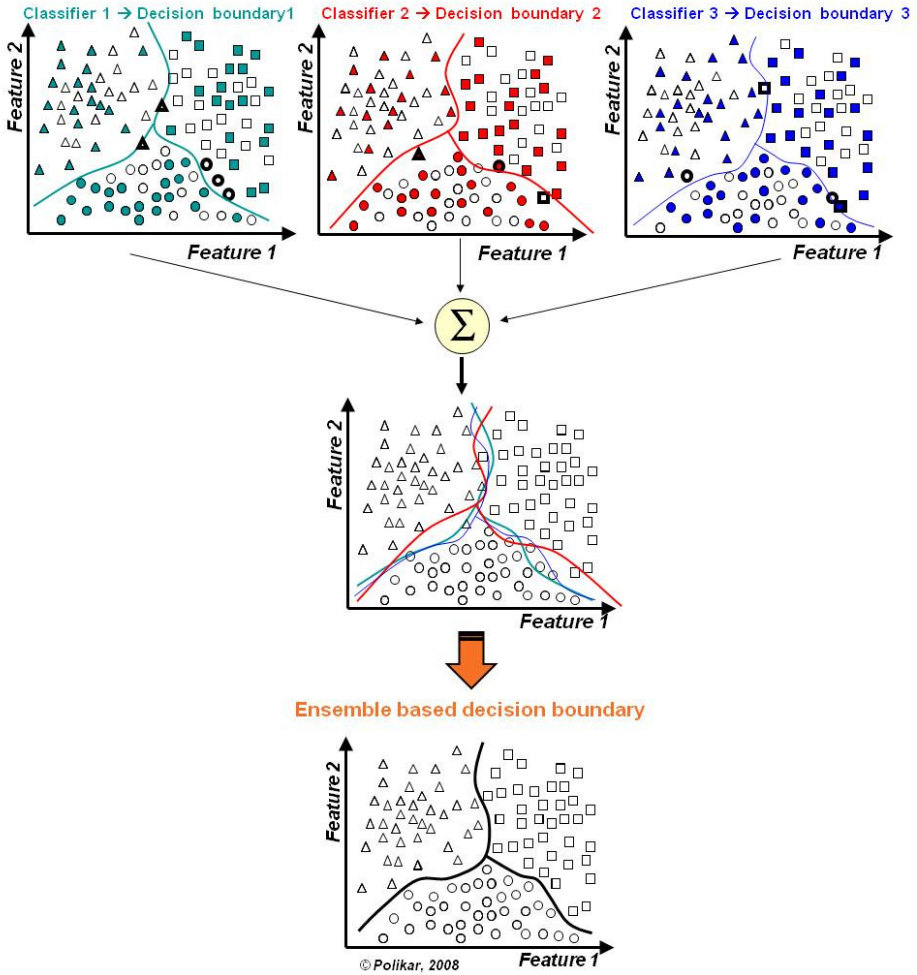
- Ensemble learning is a machine learning paradigm where multiple learners are trained and combined to solve the same problem. By using multiple learners, the generalization ability of an ensemble can be much better than that of a single learner.



In many cases an aggregated answer is better than an experts answer --> wisdom of the crowd

Ensemble Learning

- Combining an ensemble of classifiers could reduce classification error and reduce the overall risk of making selection of a particularly poorly performing classifier



- Each classifier is trained on a different subset of the available training data -- makes different errors (shown as instances with dark borders).
- The combination of the (three) classifiers provides the best decision boundary.
- The diversity in the classifiers -- typically achieved by using different training parameters for each classifier allows individual classifiers to generate different decision boundaries.
- If proper diversity is achieved, a different error is made by each classifier, strategic combination of which can then reduce the total error.

http://www.scholarpedia.org/article/Ensemble_learning

Ensemble Learning

Boosting -- combine a set of weak learners to create a stronger learner

- Iteratively train the next learner based on the mistakes of the previous learners -- after many iterations, the boosting algorithm combines the weak learners into a single strong prediction rule
- Boosting is a popular (successful) machine learning method which has amazing impact.
- It is a simple approach
- Extensively used in computer vision
  - Standard approach for face detection.
- Widely used in industry
- Used by most winners of machine learning competitions (Kaggle, KDD cup, …)
- There are various boosting algorithms
  - AdaBoost (Adaptive Boosting)
  - Gradient Tree Boosting (GBRT)
  - XGBoost
  - …

kaggle

KDD

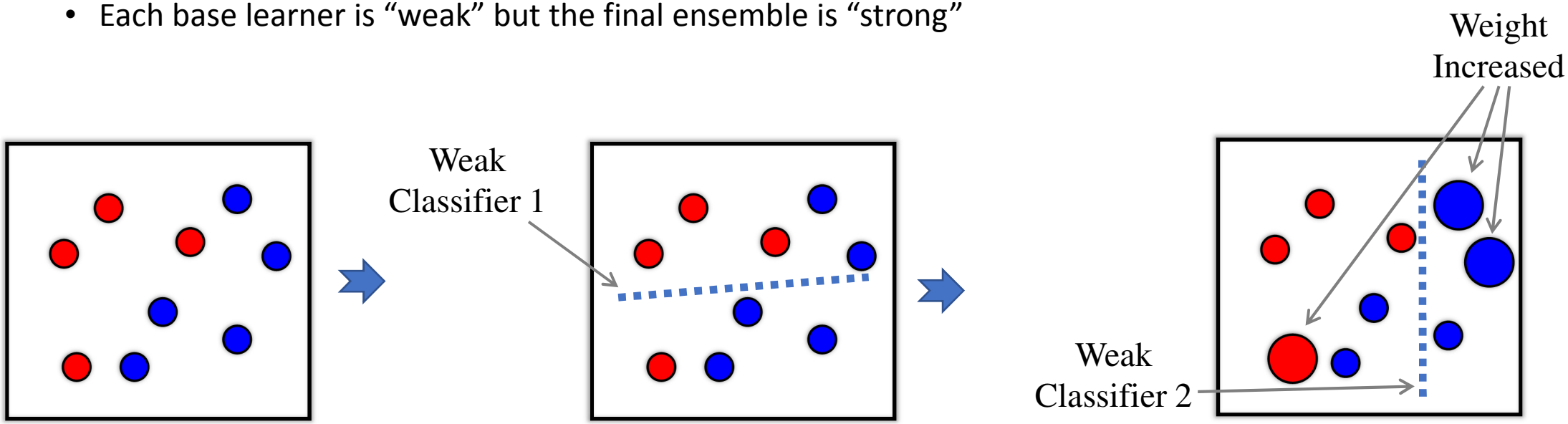
XGBoost
eXtreme Gradient Boosting

http://scikit-learn.org/stable/modules/ensemble.html#gradient-tree-boosting

Ensemble Learning

How Boosting works

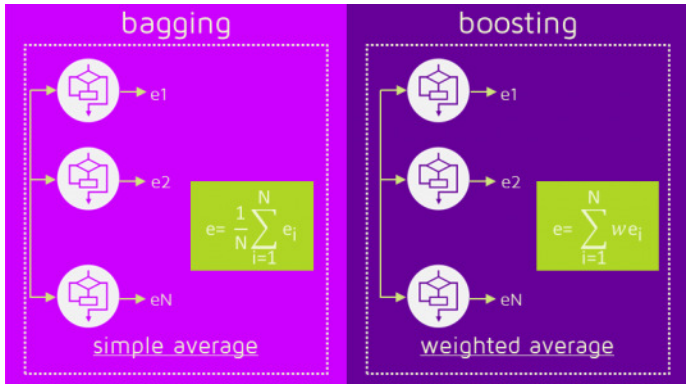
- Generate a sequence of base-learners
  - Each learner is dependent on the previous one, and focus on previous one's error. Examples that are incorrectly predicted in previous learners are chosen more often or weighted more heavily.
- Combine the base learners into a single powerful learner
  - Each base learner is "weak" but the final ensemble is "strong"



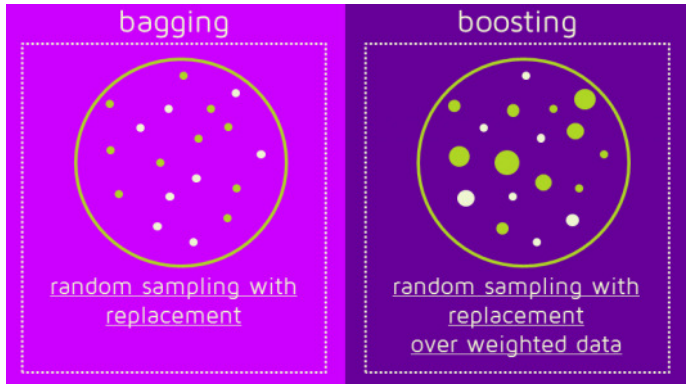
In each boosting round, a new (simple) learner (e.g., classifier) is built on the weighted dataset.

Ensemble Learning

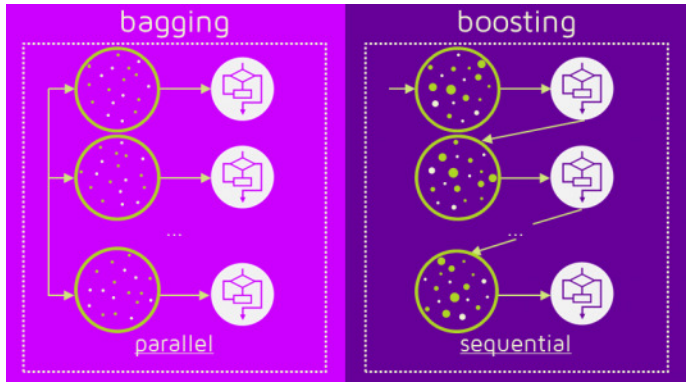
- Bagging vs. Boosting



- **Bagging**
  The result is obtained by averaging the responses of the learners (or majority vote).
- **Boosting**
  The final boosting ensemble uses a weighted average, more weight to those with better performance on training data.



- **Bagging**
  N new training data sets are produced by random sampling with replacement from the original set. Some observations may be repeated in each new training data set.
- **Boosting**
  Predicted observations are weighted and therefore some of them will take part in the new sets more often.

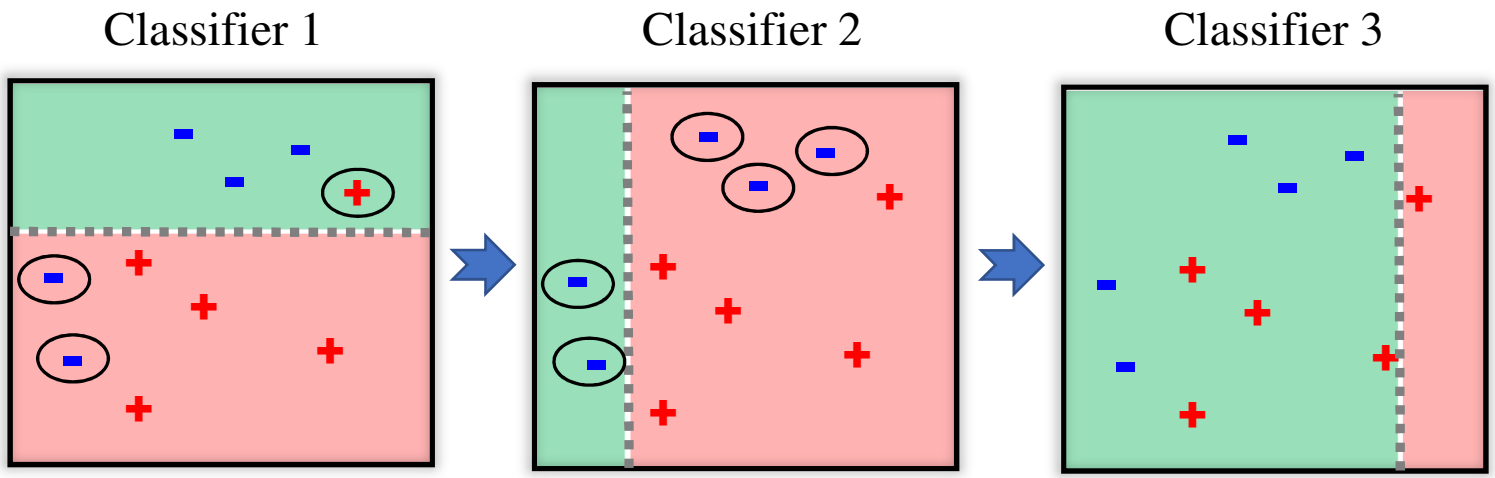


- **Bagging**
  Parallel ensemble methods, where the base learners are generated in parallel.
- **Boosting**
  Sequential ensemble methods, where base learners in boosting methods are generated sequentially (e.g., AdaBoost). Each learner is built on top of the previous one.

https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

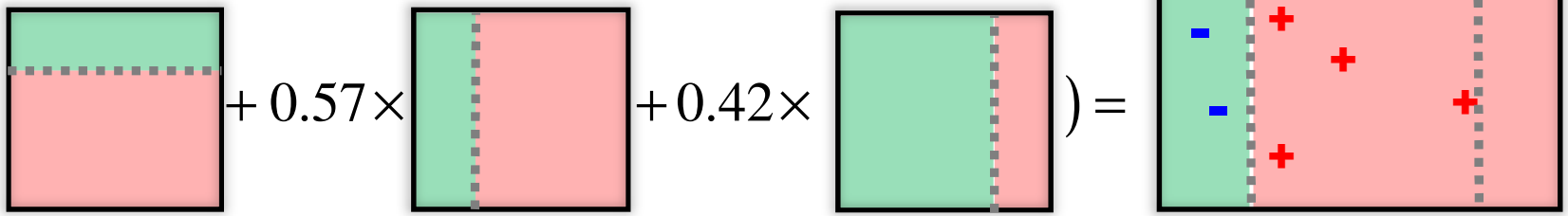
AdaBoost

- AdaBoost aims to convert a set of weak classifiers into a strong one
  - Adaptive Boosting (AdaBoost) is the first practical boosting algorithm (Freund and Schapire, 1996).
  - There are many boosting methods available, but by far the most popular are AdaBoost and Gradient Boosting.
  - AdaBoost can be used to boost the performance of any ML algorithms -- best used with weak learners
  - AdaBoost most commonly uses very simple decision stump (i.e., decision trees with one level) as the weak learners.



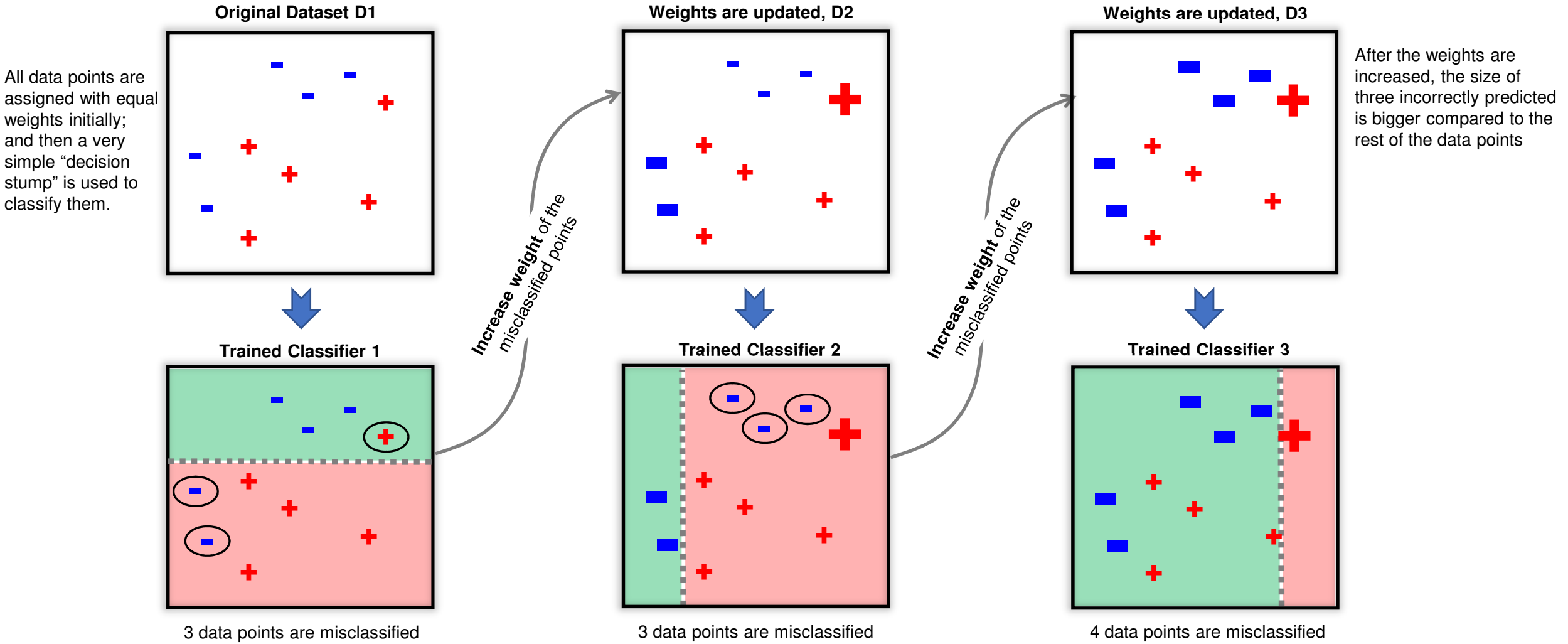
AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers.

$$H(x) = \text{sign}(0.33 \times \quad + 0.57 \times \quad + 0.42 \times \quad) =$$



AdaBoost

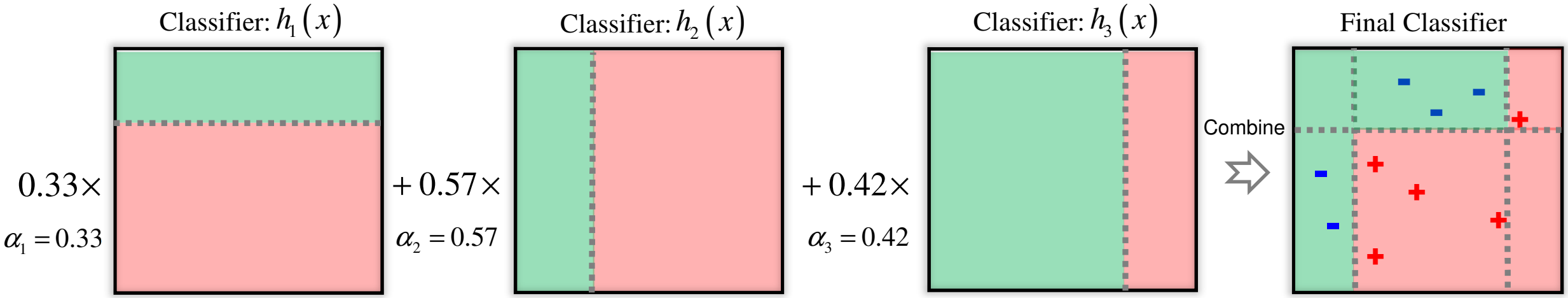
1. Train predictors (classifiers) sequentially, each trying to correct its predecessor



- The weights of misclassified training examples are increased on each round in order to emphasize the most difficult cases.
- The weights are decreased for those that were predicted correctly. Subsequent learners will focus on these mistakes during their training.

AdaBoost

2. Combines the several weak classifiers into one strong learner



α_k : the weight for each classifier

After each classifier is trained, the classifier's weight is calculated based on its accuracy. Higher-accuracy classifiers are given more weight.

$$\alpha_k = \eta \cdot \ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$$

ε_k is the weighted sum error for misclassified points

ε_k > 0.5 indicates that the classifier is not better than random guessing

η is the learning rate parameter (defaults to 1)

The equation for the final classifier:

$$H(x) = \text{sign}\left(\sum_{k=1}^{K} \alpha_k h_k(x)\right)$$

The final output is just a linear combination of all of the weak classifiers $h_k(x)$, and the final decision is decided simply by looking at the sign of this sum

http://mccormickml.com/2013/12/13/adaboost-tutorial/

AdaBoost

AdaBoost Algorithms

1. Initialize weight for each data point $w_i = \frac{1}{N}, i = 1, 2, \ldots, N$
2. For iteration $k = 1, 2, \ldots, K$
   a) Fit a classifier $h_k(x)$ to the training data using weight w_i
   b) Compute the weighted error rate of the k-th predictor

$$\varepsilon_k = \frac{\sum_{i=1}^{N} w_i \cdot I\left(y_i \neq h_k\left(x_i\right)\right)}{\sum_{i=1}^{N} w_i}$$

   c) Compute the predictor's weight α_i

$$\alpha_k = \eta \cdot \ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$$

   d) Update the weights on the training examples

$$w_i^{k+1} = w_i^k \cdot \exp\left\{\alpha_k \cdot I\left(y_i \neq h_k\left(x_i\right)\right)\right\}$$

3. Make predictions using the below final model

$$H\left(x\right) = \text{sign}\left(\sum_{k=1}^{K} \alpha_k h_k\left(x\right)\right)$$

$I\left(y_i \neq h_k\left(x_i\right)\right)$ is the indicator function that equals 1 if y_i is not equal to $h_k(x_i)$ and 0 otherwise

Strength of the hypothesis $h_k(x)$

$\alpha_k, k = 1, 2, \ldots, K$ are computed by the boosting algorithm, and weight the contribution of each respective hypothesis $h_k(x)$: the more accurate the predictor is, the higher its weight will be.

- If it is just random guessing, its weight will be close to zero.
- If it is most often wrong (i.e., less accurate than random guessing), then its weight will be negative.

- The individual weights of each of the observations (data points) are updated for the next iteration.
- Observations misclassified by $h_k(x)$ have their weights scaled by a factor $\exp(\alpha_k)$, increasing their relative influence for inducing the next classifier $h_{k+1}(x)$ in the sequence.

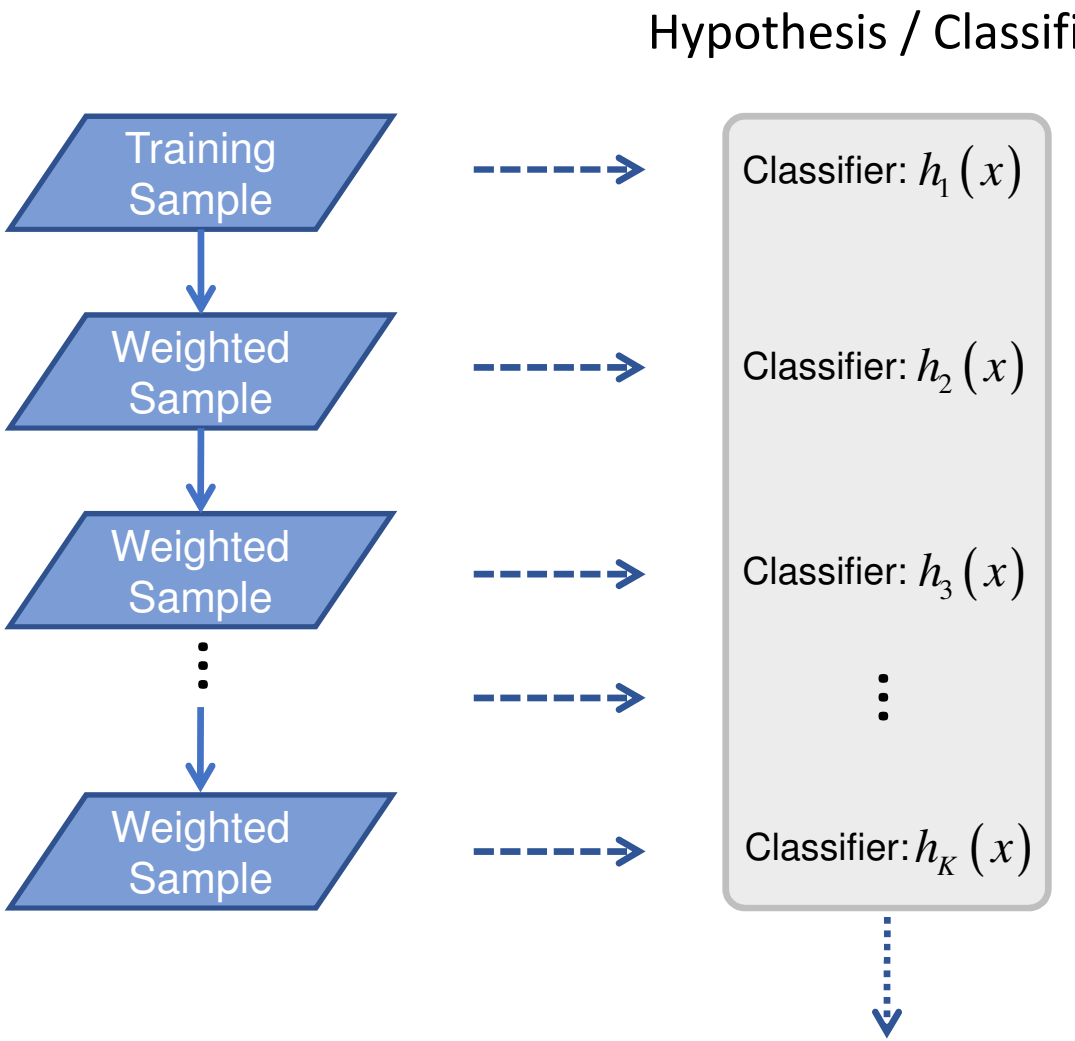
The predictions from all of predictors are then combined through a weighted majority vote to produce the final prediction

A. Géron. Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly, 2017
T. Hastie et. al. The Elements of Statistical Learning, Springer 2008
C. Bishop. Pattern Recognition and Machine Learning, Springer, 2006

AdaBoost

Re-weight each data point

- Weight will be increased for misclassified data points
- Weight will be reduced for correctly classified data points



Train model sequentially

- Classifiers are trained on weighted versions of the dataset

- The classifiers are then combined to produce a final prediction.
- The weighting coefficients give greater weight to the more accurate classifiers when computing the overall output

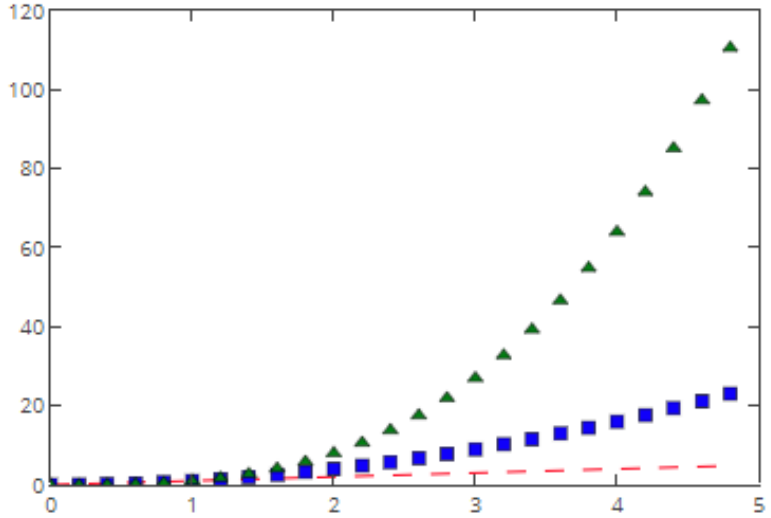
Final Model

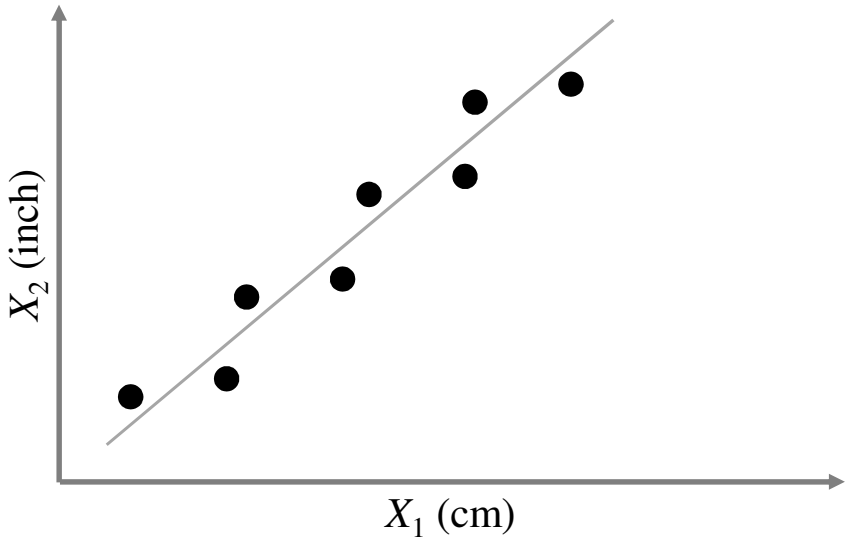
$$H(x) = \text{sign}\left(\sum_{k=1}^{K} \alpha_k h_k(x)\right)$$

Dimensionality Reduction (Principal Component Analysis)

PCA: why reduce dimensions?

- Data Visualization
  - It is difficult to visualize high-dimensional data
  - Dimensionality reduction, e.g., from 50 down to two or three, plot the data and understand it better
- Data Compression
  - Remove redundant and noisy features
  - Reduce memory/disk usage
  - Speed up other learning algorithms because of fewer inputs
- Redundant Information
  - “inch” --> “cm”
  - X1 is the length in centimeters
  - X2 is the same length in inches
  - X1 and X2 are highly correlated (well, actually they are the same but only measured in different units)
  - High redundant --> should reduce it to one-dimensional

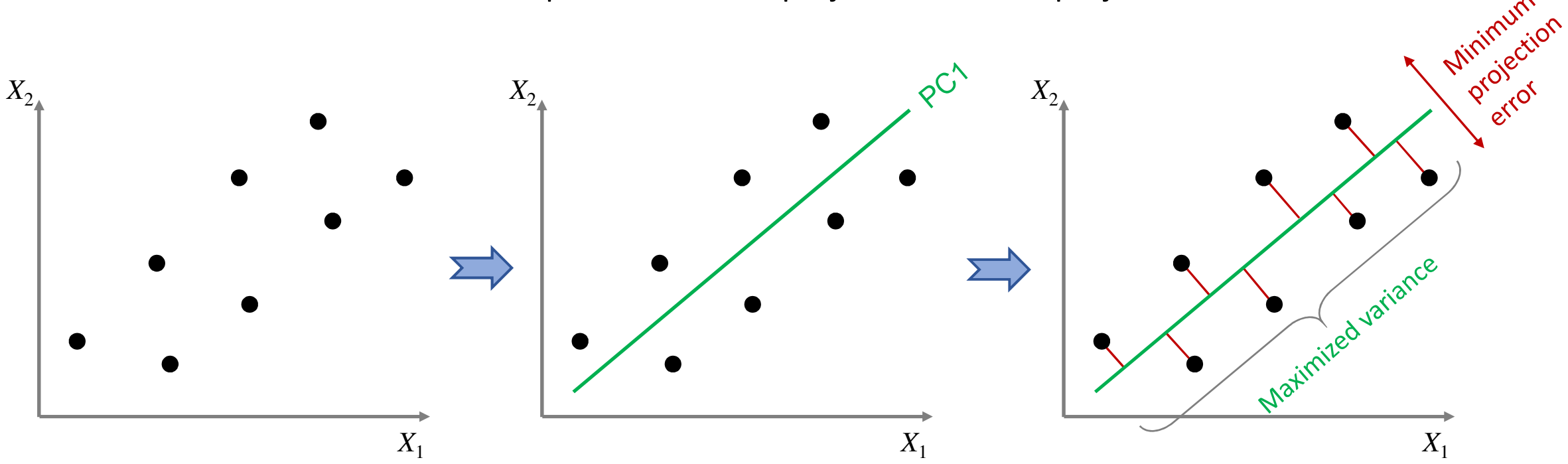




The data points do not perfectly lie on a straight line just due to measurement noise, number round-off, etc.

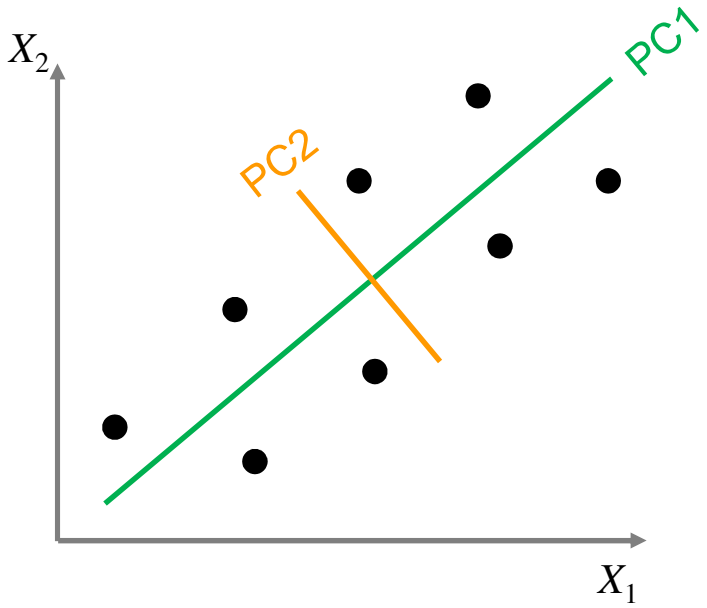
PCA

- What is the criteria for finding the “best-fit” line or (hyper)plane?
  - PCA is to project the data onto lower dimensional linear space, such that the **variance** of the projected data is **maximized**.
  - Pick a line along which the data is spread out the most --> the **variance** of projected data is maximized
  - Linear projection should minimizes the average projection cost, defined as the **mean squared distance** between the data points and their projections --> the projection error is minimized



PCA

- Principal components
  - PCA uses orthogonal projection of highly correlated variables to a set of values of linearly uncorrelated variables called principal components.
  - This linear transformation is defined in such a way that the first principal component has the largest possible variance.



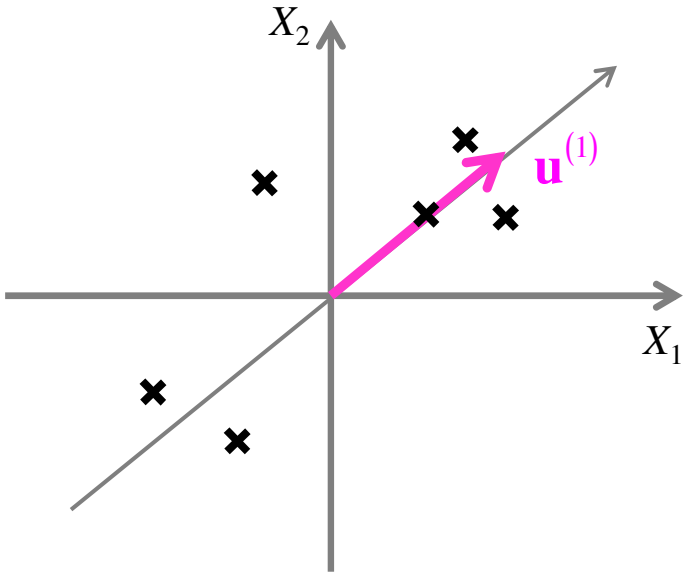
- PC1 (the first principal component) is the direction in the original data that contains the most variance --> PCA identifies the axis that accounts for the largest amount of variance in the training set
- PC2 captures the direction with 2nd most variance --> PCA also finds a second axis (**yellow** line), orthogonal to the first one, that accounts for the largest amount of remaining variance

- For higher-dimensional datasets, PCA would also find a third axis, orthogonal to both previous axes, and a fourth, a fifth, etc. -- as many axes as the number of dimensions in the dataset.

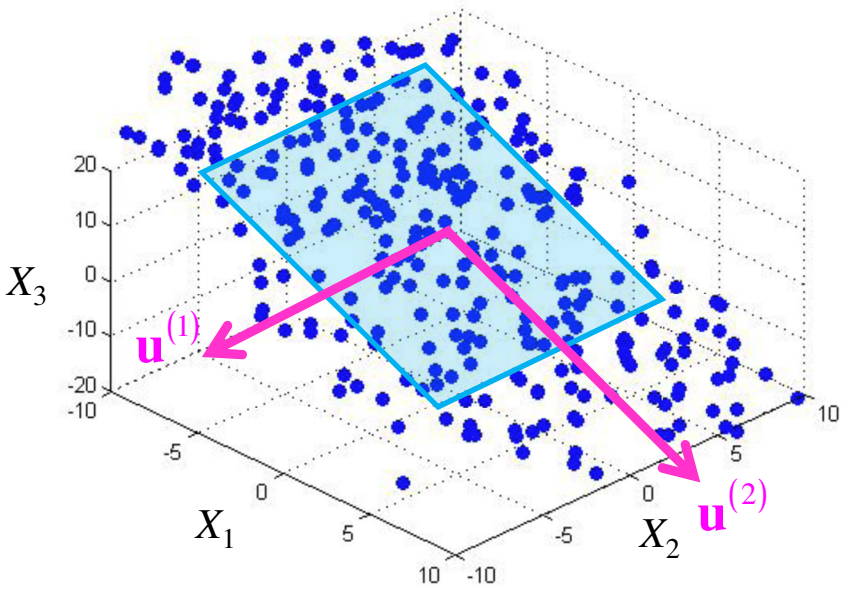
A. Géron. Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly, 2017

PCA

- Formulation
  - PCA finds a lower dimensional surface onto which to project data so as to minimize the **projection squared error**, to minimize the **squared distance** between the each point and the location of where it gets projected



2D --> 1D
(K = 1)



3D --> 2D
(K = 2)

Reduce from 2D to 1D:
Find a direction, a vector $\mathbf{u}^{(1)}$, onto which to project the data so as to minimize the projection error.

Reduce from n-D to k-D:
Find k vectors $\mathbf{u}^{(1)}$, $\mathbf{u}^{(2)}$, …, $\mathbf{u}^{(k)}$ onto which to project the data to minimize the projection error.

Required Statistics

- Variance $Var(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$ The average of the squared differences from the mean

- Covariance $Cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$ how two different variables / dimensions change together

- Covariance Matrix $CM = \begin{bmatrix} Cov(X,X) & Cov(X,Y) & Cov(X,Z) \\ Cov(Y,X) & Cov(Y,Y) & Cov(Y,Z) \\ Cov(Z,X) & Cov(Z,Y) & Cov(Z,Z) \end{bmatrix}$

PCA

1. Subtract the mean

2. Calculate the covariance matrix

3. Calculate the eigenvectors and eigenvalues of the covariance matrix

4. Choosing components and forming a feature vector

5. Deriving the new data set

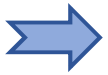
PCA

1. Subtract the mean
2. Calculate the covariance matrix
3. Calculate the eigenvectors and eigenvalues of the covariance matrix
4. Choosing components and forming a feature vector
5. Deriving the new data set

- Subtract the mean from each of the data dimensions --> produces a data set whose mean is zero
- Subtract the mean makes variance and covariance calculation easier by simplifying their equations.
- The variance and covariance value are not affected by the mean value.

Data =

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9



DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

A tutorial on Principal Components Analysis by L.I. Smith

PCA

1. Subtract the mean
2. Calculate the covariance matrix
3. Calculate the eigenvectors and eigenvalues of the covariance matrix
4. Choosing components and forming a feature vector
5. Deriving the new data set

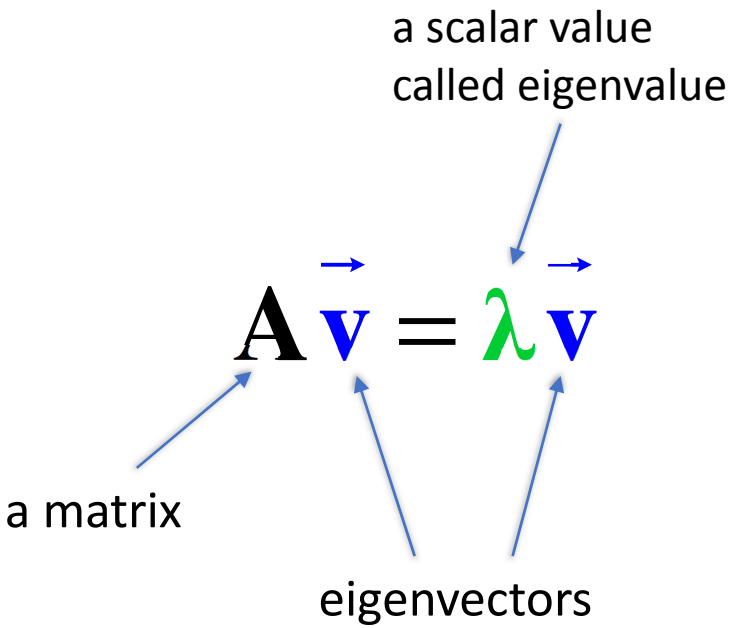
$$Cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

$$CM = \begin{bmatrix} Cov(X,X) & Cov(X,Y) & Cov(X,Z) \\ Cov(Y,X) & Cov(Y,Y) & Cov(Y,Z) \\ Cov(Z,X) & Cov(Z,Y) & Cov(Z,Z) \end{bmatrix}$$

n dimensional data $\Rightarrow n \times n$ covariance matrix

PCA

1. Subtract the mean
2. Calculate the covariance matrix
3. Calculate the eigenvectors and eigenvalues of the covariance matrix
4. Choosing components and forming a feature vector
5. Deriving the new data set



PCA

1. Subtract the mean
2. Calculate the covariance matrix
3. Calculate the eigenvectors and eigenvalues of the covariance matrix
4. Choosing components and forming a feature vector
5. Deriving the new data set

Pick top k principal components

1. Rank the eigenvectors by eigenvalues, highest to lowest
   - Eigenvalues: $\lambda_1 > \lambda_2 > , \ldots , > \lambda_n$
   - Eigenvector with largest eigenvalue captures the most variation in training set
   - Eigenvector with smallest eigenvalue captures the least variation
2. Choose top k eigenvectors which contain most of the variance
   - Select top k eigenvectors ($u_1, u_2, \ldots, u_k$) to form a matrix

$$\mathbf{U} = \begin{bmatrix} u^{(1)} & u^{(2)} & \ldots & u^{(n)} \end{bmatrix} \Rightarrow \mathbf{U}_{\text{reduce}} = \begin{bmatrix} u^{(1)} & u^{(2)} & \ldots & u^{(k)} \end{bmatrix}$$

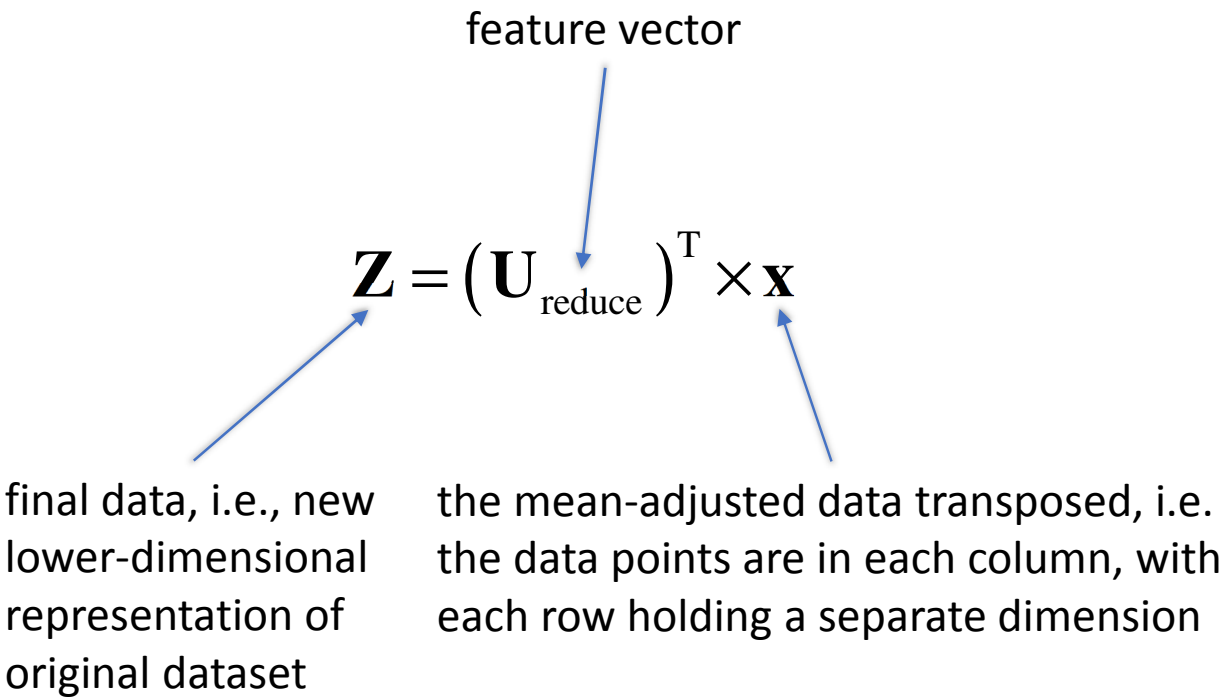
   - This corresponds to choosing a “linear subspace”. The eigenvectors will form the new axes. The eigenvectors with larger eigenvalues correspond to directions in which the data varies more.

Ignore the components of less significance --> some information will be lost, but not too much if the ignored parts are with small eigenvalues

PCA

1. Subtract the mean
2. Calculate the covariance matrix
3. Calculate the eigenvectors and eigenvalues of the covariance matrix
4. Choosing components and forming a feature vector
5. Deriving the new data set

Project original data points onto the hyperplane (subspace) defined by the selected k principal components via the following equation:



PCA

1. Subtract the mean
2. Calculate the covariance matrix
3. Calculate the eigenvectors and eigenvalues of the covariance matrix
4. Choosing components and forming a feature vector
5. Deriving the new data set

Summary

- Principal component analysis involves evaluating the mean and the covariance matrix of the data set and then finding the k eigenvectors of covariance matrix with the k largest corresponding eigenvalues.
- PCA projects the data along the directions where the data varies the most.
- These directions are determined by the eigenvectors of the covariance matrix with the largest corresponding eigenvalues.
- The magnitude of the eigenvalues corresponds to the variance of the data along the eigenvector directions.

PCA

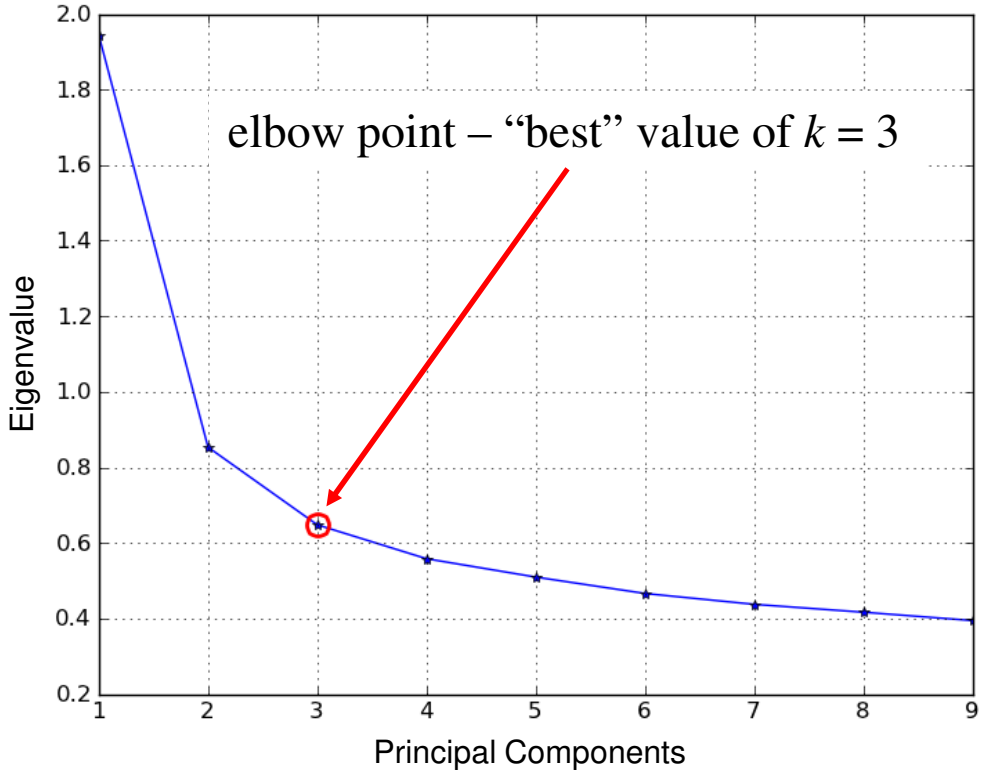
- Challenge: choose parameter ***k*** for PCA, i.e., how to determine the number of components to retain.

$$\mathbf{U} = \left[\underbrace{u^{(1)} \quad u^{(2)} \quad \ldots}_{} \quad u^{(n)}\right] \Rightarrow \mathbf{U}_{\text{reduce}} = \left[u^{(1)} \quad u^{(2)} \quad \ldots \quad u^{(k)}\right]$$

Select top ***k*** eigenvectors ($u_1, u_2, \ldots, u_k$) to form a matrix

Scree test

- look for a "elbow" / "break" (a sharp change in the slopes of adjacent line segments) between the components with relatively large eigenvalues and those with small eigenvalues.
- The components that appearing after the break are assumed to be unimportant and are not retained.
- Determine where exactly in the scree plot a "elbow/break" exists is not straightforward



PCA

- Challenge: choose parameter ***k*** for PCA, i.e., how to determine the number of components to retain.

$$\mathbf{U} = \left[\underbrace{u^{(1)} \quad u^{(2)} \quad \ldots}_{} \quad u^{(n)} \right] \Rightarrow \mathbf{U}_{\text{reduce}} = \left[u^{(1)} \quad u^{(2)} \quad \ldots \quad u^{(k)} \right]$$

Select top ***k*** eigenvectors ($u_1, u_2, \ldots, u_k$) to form a matrix

Percentage of Explained Variance

- A useful measure is the so-called "explained variance", which can be calculated from the eigenvalues.
- Retain components that cumulatively explain a certain percentage of variation. The acceptable level of explained variance depends on how you use Principal Components.
- Use SVD (Singular Value Decomposition), then pick smallest value of k for which:

More details → https://youtu.be/5aHWplWElcc

$$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = svd(\text{Covariance Matrix})$$

$$\mathbf{S} = \begin{pmatrix} S_{11} & \cdots & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & \cdots & S_{nn} \end{pmatrix}$$

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} > \text{Threshold (e.g.,0.9 or 0.95)}$$

If the threshold is 0.99, then 99% variance is retained

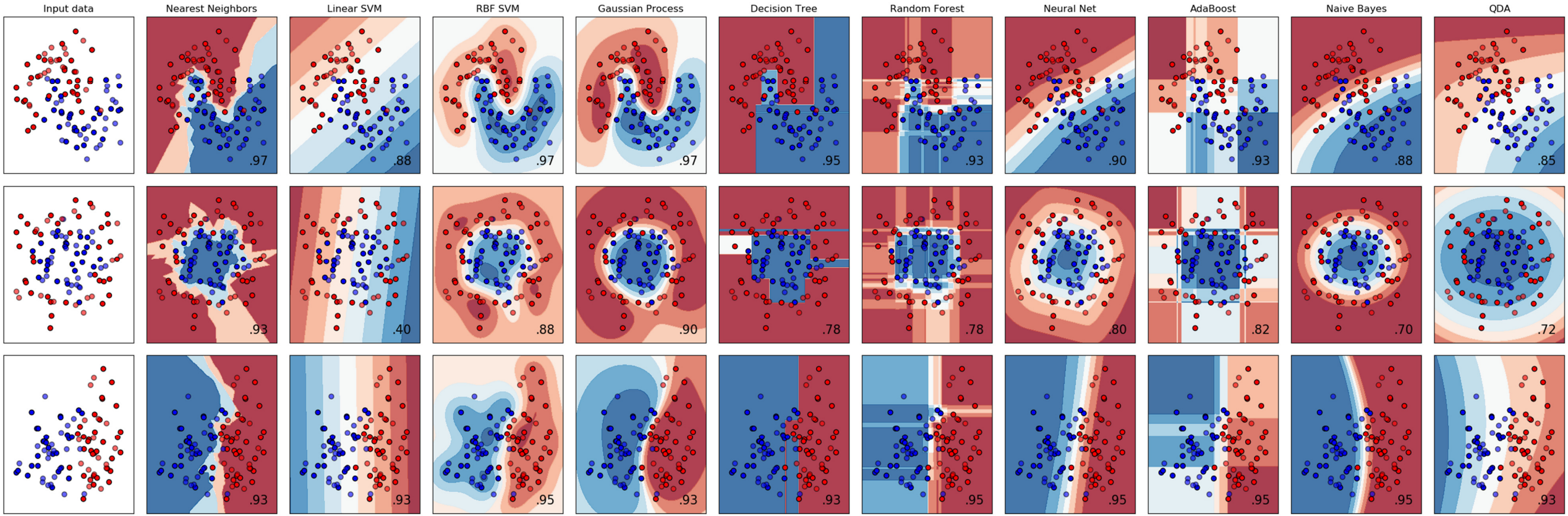
References

- PCA library
  - http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
- A tutorial on Principal Components Analysis by L.I. Smith
  - http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf

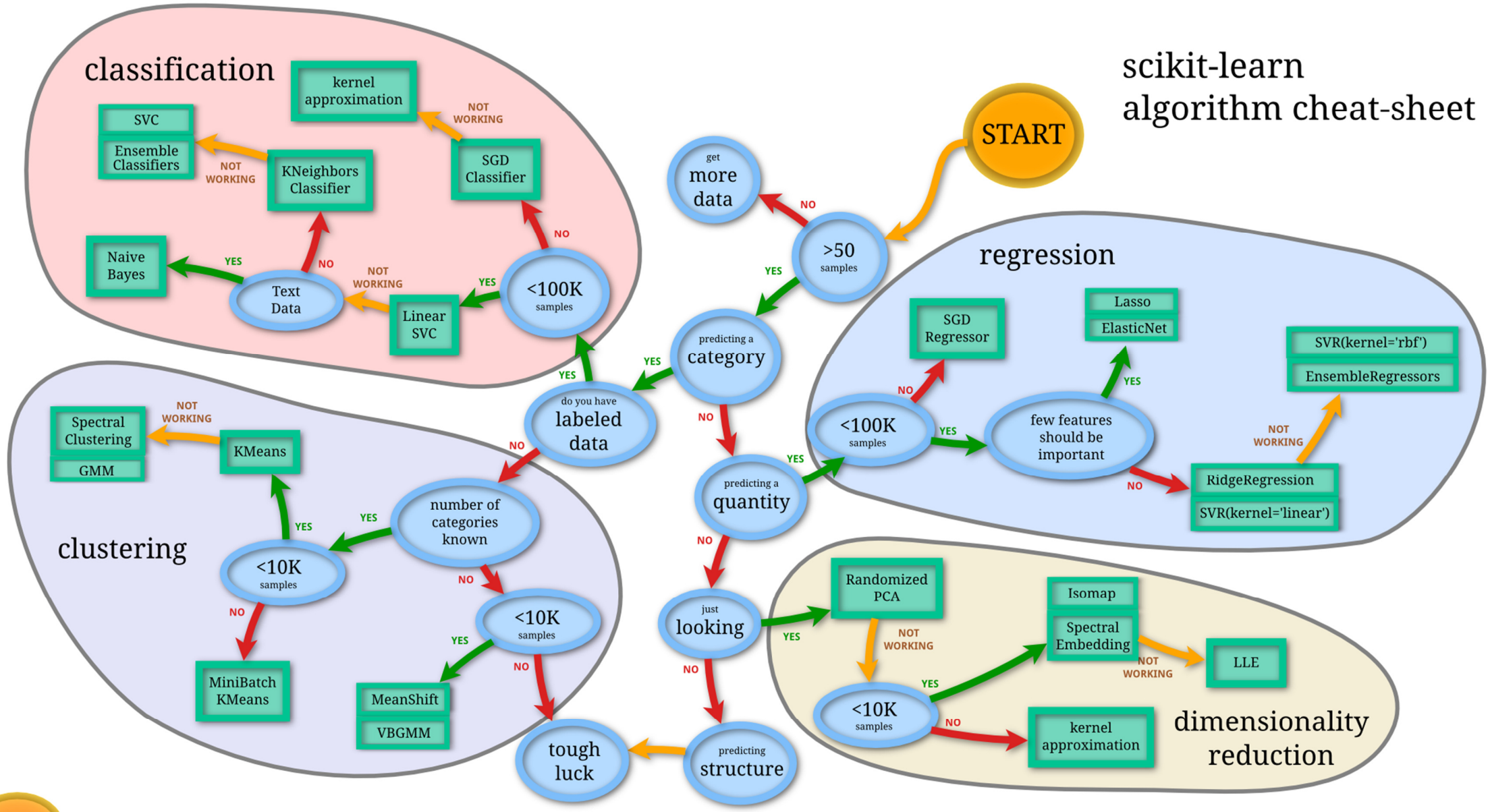
Closing Remarks

A comparison of a several classifiers in scikit-learn on synthetic datasets

The plots show training points in solid colors and testing points semi-transparent.
The lower right shows the classification accuracy on the test set.



http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifi



- Often the hardest part of solving a ML problem is to find the right estimator for the job.
- Different estimators are best suited for different types of data and different problems.
- The flowchart below is designed to give users a bit of a rough guide on how to approach problems with regard to which estimators to try on your data.

http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Microsoft Azure Machine Learning: Algorithm Cheat Sheet

This cheat sheet helps you choose the best Azure Machine Learning algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.

START

Predicting categories

Three or more

Two

Predicting values

Finding unusual data points

Discovering structure

ANOMALY DETECTION

- One-class SVM — >100 features, aggressive boundary
- PCA-based anomaly detection — Fast training

CLUSTERING

- K-means

MULTICLASS CLASSIFICATION

- Fast training, linear model — Multiclass logistic regression
- Accuracy, long training times — Multiclass neural network
- Accuracy, fast training — Multiclass decision forest
- Accuracy, small memory footprint — Multiclass decision jungle
- Depends on the two-class classifier, see notes below — One-v-all multiclass

REGRESSION

- Ordinal regression — Data in rank ordered categories
- Poisson regression — Predicting event counts
- Fast forest quantile regression — Predicting a distribution
- Linear regression — Fast training, linear model
- Bayesian linear regression — Linear model, small data sets
- Neural network regression — Accuracy, long training time
- Decision forest regression — Accuracy, fast training
- Boosted decision tree regression — Accuracy, fast training

TWO-CLASS CLASSIFICATION

- Two-class SVM — >100 features, linear model
- Two-class averaged perceptron — Fast training, linear model
- Two-class logistic regression — Fast training, linear model
- Two-class Bayes point machine — Fast training, linear model

- Accuracy, fast training — Two-class decision forest
- Accuracy, fast training — Two-class boosted decision tree
- Accuracy, small memory footprint — Two-class decision jungle
- >100 features — Two-class locally deep SVM
- Accuracy, long training times — Two-class neural network

Created by the Azure Machine Learning Team

Email: AzurePoster@microsoft.com

Download this poster: http://aka.ms/MLCheatSheet

Microsoft

Machine Learning in ML Studio

Anomaly Detection
- One-class Support Vector Machine
- Principal Component Analysis-based Anomaly Detection
- Time Series Anomaly Detection*

Classification
- Two-class Classification
  - Averaged Perceptron
  - Bayes Point Machine
  - Boosted Decision Tree
  - Decision Forest
  - Decision Jungle
  - Logistic Regression
  - Neural Network
  - Support Vector Machine
- Multi-class Classification
  - Decision Forest
  - Decision Jungle
  - Logistic Regression
  - Neural Network
  - One-vs-all

Clustering
- K-means Clustering

Recommendation
- Matchbox Recommender

Regression
- Bayesian Linear Regression
- Boosted Decision Tree
- Decision Forest
- Fast Forest Quantile Regression
- Linear Regression
- Neural Network Regression
- Ordinal Regression
- Poisson Regression

Statistical Functions
- Descriptive Statistics
- Hypothesis Testing T-Test
- Linear Correlation
- Probability Function Evaluation

Text Analytics
- Feature Hashing
- Named Entity Recognition
- Vowpal Wabbit

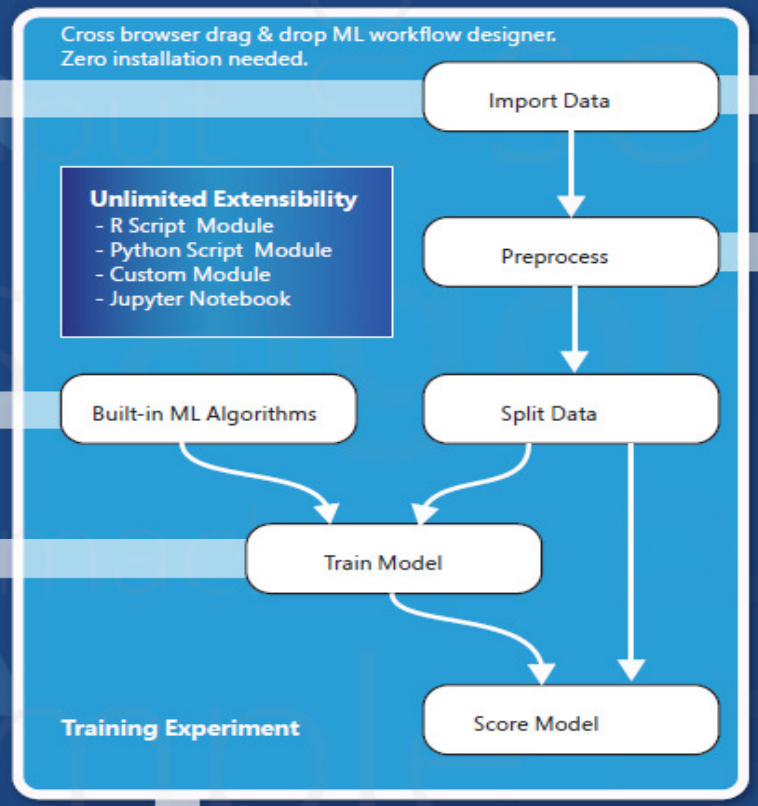
Computer Vision
- OpenCV Library

https://studio.azureml.net

Guest Access Workspace: Free trial access without logging in.
Free Workspace: Free persisted access, no Azure subscription needed.
Standard Workspace: Full access with SLA under an Azure subscription.

Data/Model Visualization
- Scatterplots
- Bar Charts
- Box plots
- Histogram
- R and Python Plotting Libraries
- REPL with Jupyter Notebook
- ROC, Precision/Recall, Lift
- Confusion Matrix
- Decision Tree*

Training
- Cross Validation
- Retraining
- Parameter Sweep



Data Source
- Azure Blob Storage
- Azure SQL DB
- Azure SQL DW*
- Azure Table
- Desktop Direct Upload
- Hadoop Hive Query
- Manual Data Entry
- OData Feed
- On-prem SQL Server*
- Web URL (HTTP)

Data Format
- ARFF
- CSV
- SVMLight
- TSV
- Excel
- ZIP

Data Preparation
- Clean Missing Data
- Clip Outliers
- Edit Metadata
- Feature Selection
- Filter
- Learning with Counts
- Normalize Data
- Partition and Sample
- Principal Component Analysis
- Quantize Data
- SQLite Transformation
- Synthetic Minority Oversampling Technique

Enterprise Grade Cloud Service
- SLA: 99.95% Guaranteed Up-time
- Azure AD Authentication
- Compute at Large Scale
- Multi-geo Availability
- Regulatory Compliance*

One-click Operationalization

Predictive Experiment

Make Prediction with Elastic APIs
- Request-Response Service (RRS)
- Batch Execution Service (BES)
- Retraining API

Community
- Gallery (http://gallery.azureml.net)
- Samples & Templates
- Workspace Sharing and Collaboration
- Live Chat & MSDN Forum Support

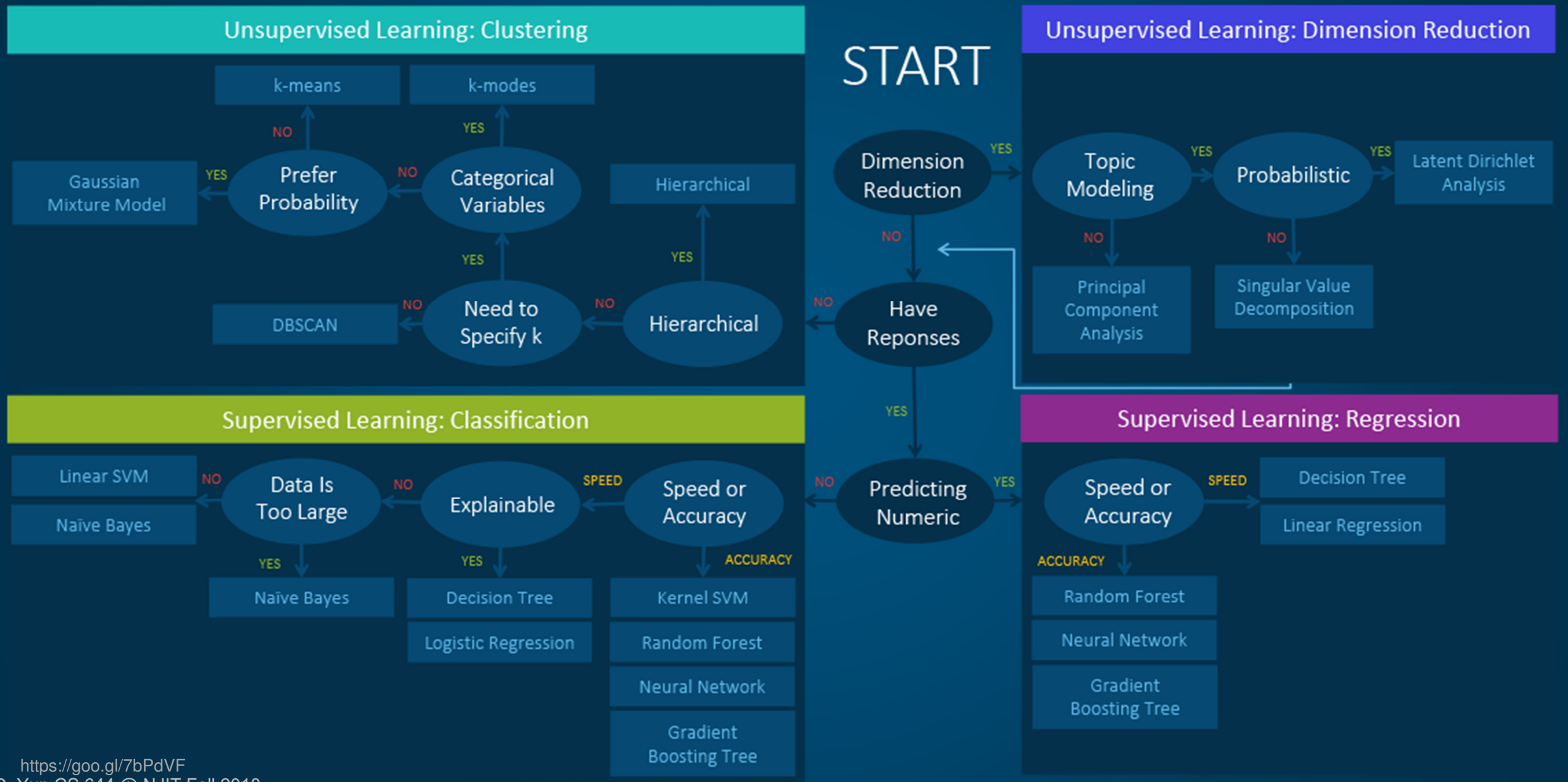
* Feature Coming Soon

Azure Machine Learning Studio Capabilities Overview

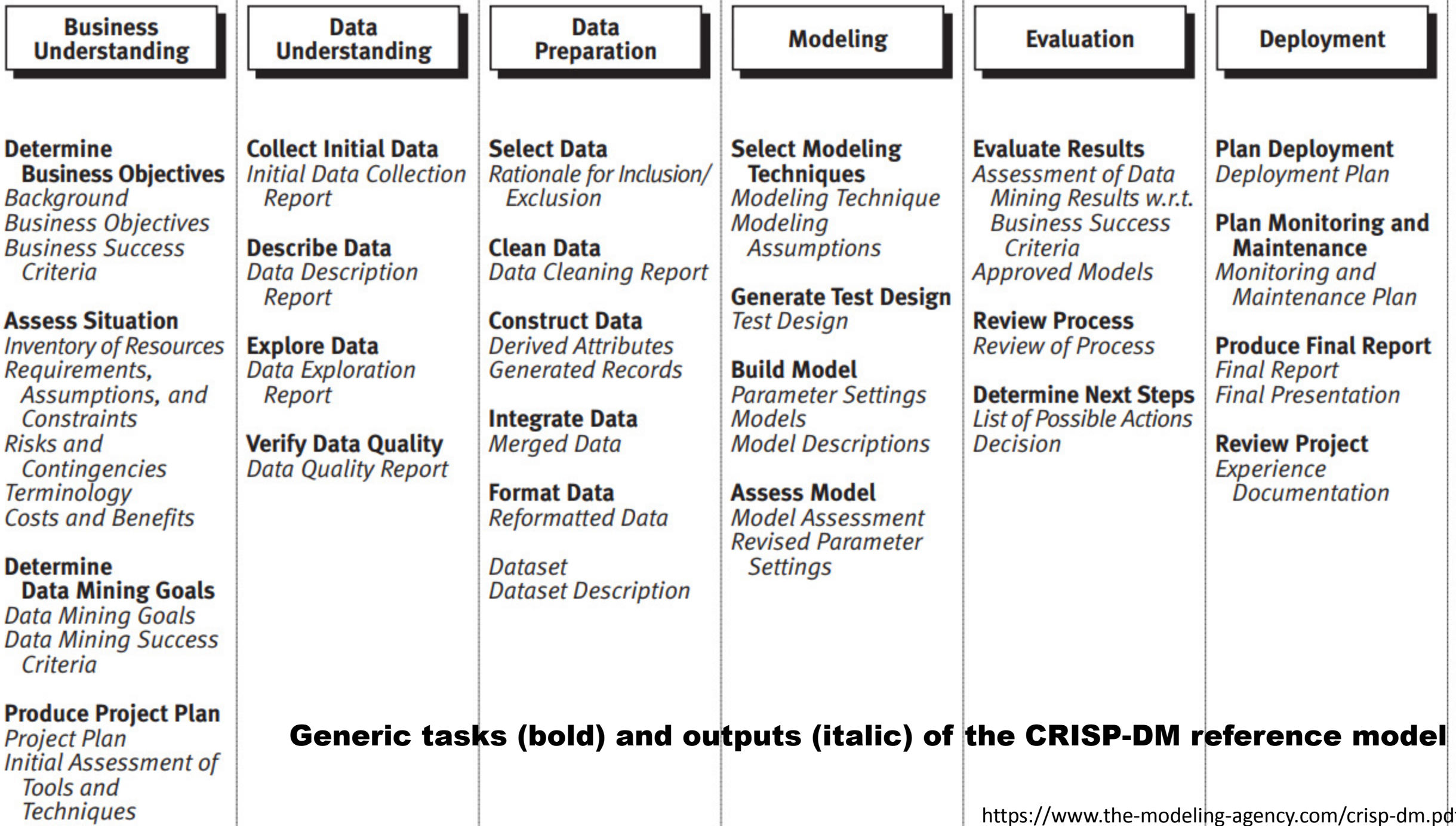
Microsoft

Created by the Azure Machine Learning Team
Email: AzurePoster@microsoft.com
Download this poster: http://aka.ms/MLStudioOverview

Machine Learning Algorithms Cheat Sheet

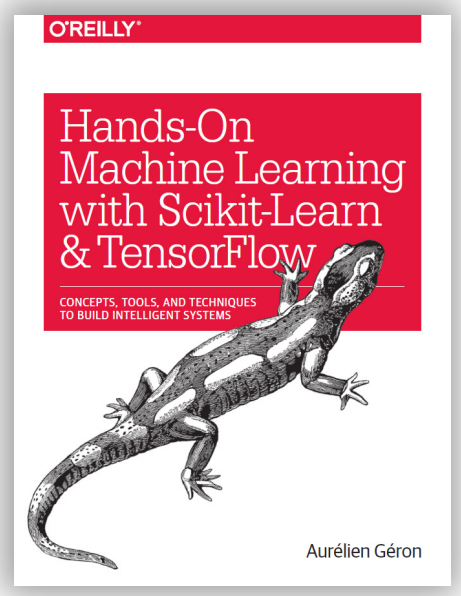


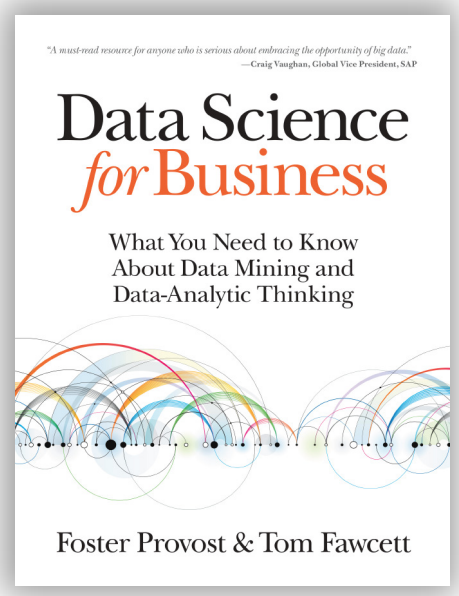
https://goo.gl/7bPdVF

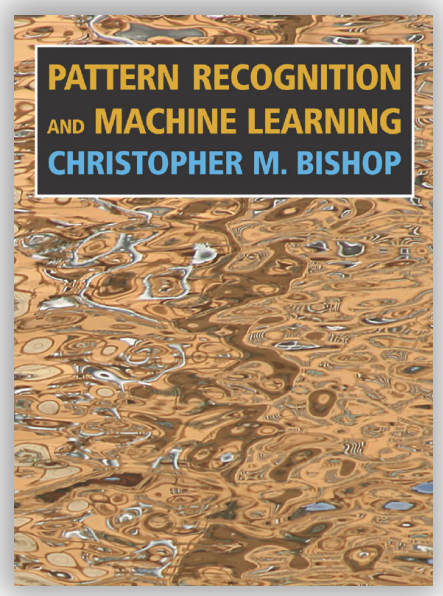


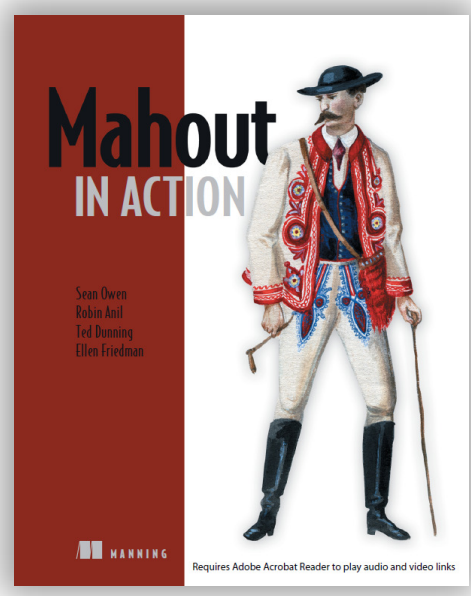
Generic tasks (bold) and outputs (italic) of the CRISP-DM reference model

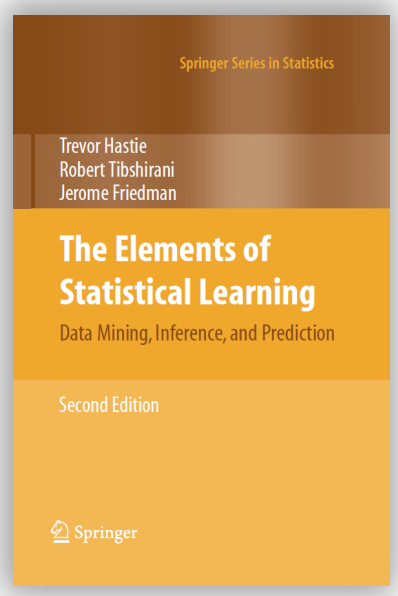
https://www.the-modeling-agency.com/crisp-dm.pdf

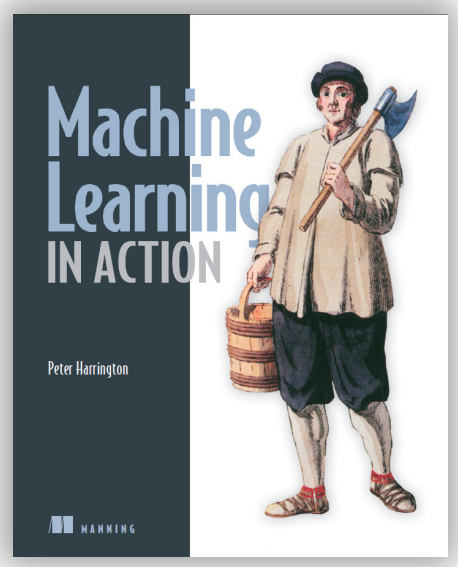

O'REILLY
Hands-On Machine Learning with Scikit-Learn & TensorFlow
CONCEPTS, TOOLS, AND TECHNIQUES TO BUILD INTELLIGENT SYSTEMS
Aurélien Géron

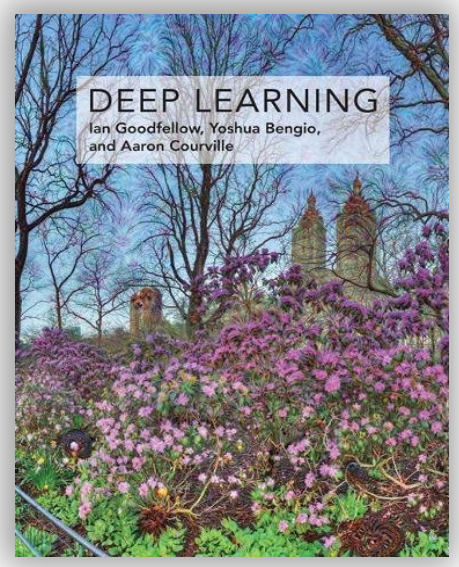

"A must-read resource for anyone who is serious about embracing the opportunity of big data."
—Craig Vaughan, Global Vice President, SAP
Data Science for Business
What You Need to Know About Data Mining and Data-Analytic Thinking
Foster Provost & Tom Fawcett

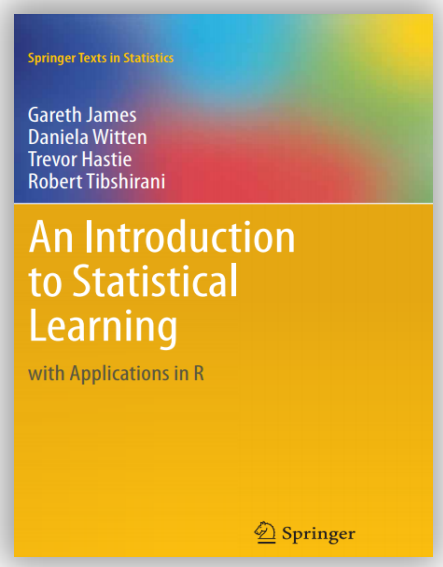

PATTERN RECOGNITION AND MACHINE LEARNING
CHRISTOPHER M. BISHOP

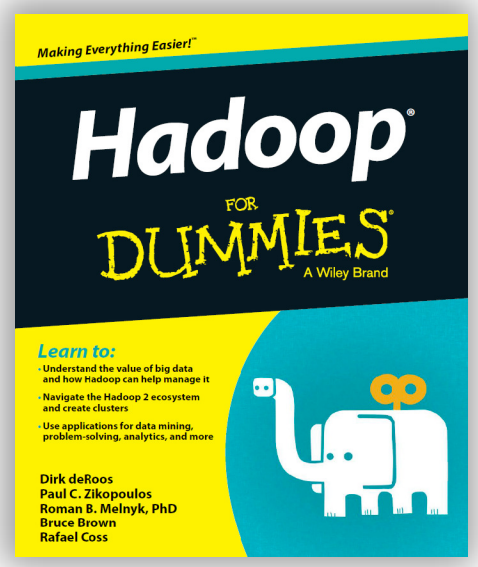

Mahout IN ACTION
Sean Owen
Robin Anil
Ted Dunning
Ellen Friedman
MANNING
Requires Adobe Acrobat Reader to play audio and video links

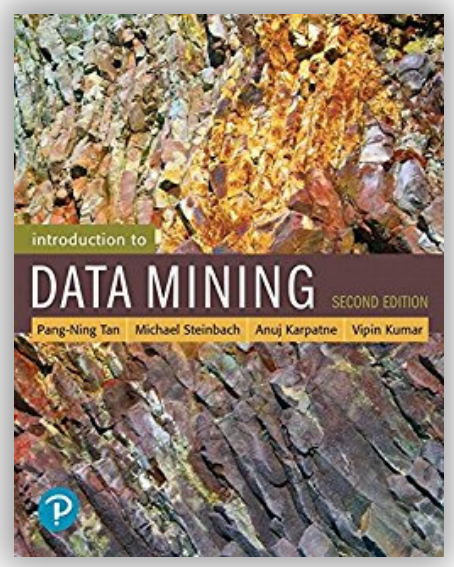

Springer Series in Statistics
Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of Statistical Learning
Data Mining, Inference, and Prediction
Second Edition
Springer


Machine Learning IN ACTION
Peter Harrington
MANNING

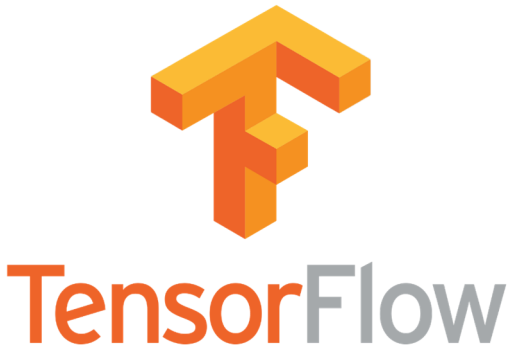

DEEP LEARNING
Ian Goodfellow, Yoshua Bengio, and Aaron Courville


Springer Texts in Statistics
Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani
An Introduction to Statistical Learning
with Applications in R
Springer


Making Everything Easier!™
Hadoop FOR DUMMIES
A Wiley Brand
Learn to:
Understand the value of big data and how Hadoop can help manage it
Navigate the Hadoop 2 ecosystem and create clusters
Use applications for data mining, problem-solving, analytics, and more
Dirk deRoos
Paul C. Zikopoulos
Roman B. Melnyk, PhD
Bruce Brown
Rafael Coss


introduction to
DATA MINING
SECOND EDITION
Pang-Ning Tan
Michael Steinbach
Anuj Karpatne
Vipin Kumar

Popular Frameworks



■■■

Datasets

- https://www.kdnuggets.com/datasets/index.html
- http://archive.ics.uci.edu/ml/index.php
- https://www.kaggle.com/datasets
- https://vision.cornell.edu/se3/coco-text-2/
- http://stat-computing.org/
- …

References

- scikit-learn: Machine Learning in Python
  - http://scikit-learn.org/stable/index.html

Thanks ! ☺

Questions ?