Machine Learning Training

Kevin Lee MS, Director of Data Science

1

Agenda

- Introduction of Machine Learning
- Machine Learning Concepts – Hypothesis function, Cost function, Gradient Descent, Learning Rate
- Machine Learning Types – Supervised and Unsupervised
- Machine Learning Algorithms – Classification, Regression, Linear Regression, Decision Tree, Clustering
- Artificial Neural Network (ANN)
  - Definition
  - Impact
  - Structures – input layers, hidden layers, output layers
  - Parameters – weights, biases
  - Hyperparameters - activation function, learning rate
- Deep Neural Network (DNN)
- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)
- Feature Engineering
- Current/Future Machine Learning Implementation

2

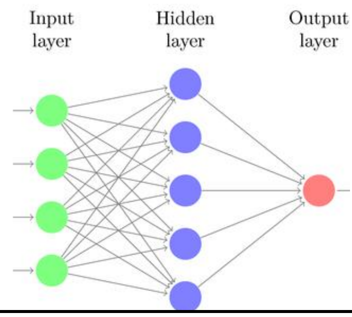
Machine Learning Introduction

- What is Machine Learning? - An application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.
- Difference

Explicit programming:
rule-based programming

```
** Derivation of 1st Age group **;
if age < 65 then do;
   agegr1n=1; agegr1="<65";
end;
if 65<=age<=69 then do;
   agegr1n=2; agegr1="65 - 69";
end;
else if 70<=age<=74 then do;
   agegr1n=3; agegr1="70 - 74";
end;
else if 75<=age<=79 then do;
   agegr1n=4; agegr1="75 - 79";
end;
else if 80<=age<=84 then do;
   agegr1n=5; agegr="80 - 84";
end;
else if age ge 85 then do;
   agegr1n=6; agegr1=">=85";
end;
```

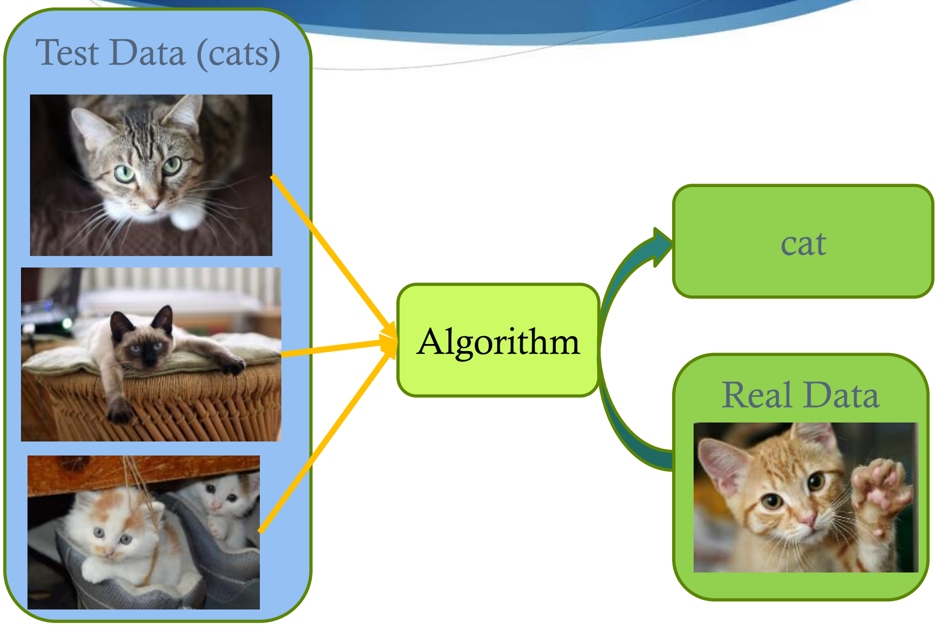
Machine Learning:
using data and algorithm



3

4

How does Machine learn?



5

How does Machine Learning work?

Input & output

X_0	X_1	X_2	…	Xn	Y
0	17	250		29	1
34	134	22		57	0
72	169	167		89	0
22	34	220		9	0
52	227	235		69	1

Algorithm

- Hypothesis Function - $h_\theta(x) = \theta x + b$
- Minimize Cost Function to find optimum parameters (θ, b), $J(\theta) = h_\theta(x) - Y$, using Gradient Descent

6

Machine Learning Functions

- Hypothesis function
  - Algorithm/Model for data
  - Linear Regression, Logistic Regression, Support Vector Machine, Decision Tree, Artificial Neural Network
  - $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 +,, + \theta_n x_n$ (e.g., $Y = 2x + 30$)
- Parameters : $\theta_0, \theta_1, \theta_2 , \theta_3,,,\theta_n$
- Cost function
  - To measure how well hypothesis function fits into data.
  - Difference between actual data point and hypothesized data point. (e.g., $Y - h_\theta(x)$)
  - $J(\theta_{0,} \theta_{1,} \theta_{2,,} \theta_n) = (1/2m)$*sum[(Y - H)^2]
- Gradient Descent
  - Engine that minimizes cost function
  - Repeat
    - $\Theta_0 := \theta_0 -$ alpha* $d/d\theta_0$ * $J(\theta_{0,} \theta_{1,} \theta_{2,,} \theta_n)$
    - $\Theta_1 := \theta_1 -$ alpha* $d/d\theta_1$ * $J(\theta_{0,} \theta_{1,} \theta_{2,,} \theta_n)$
- Learning rate (alpha) – size of learning step in gradient descent

Machine Learning Project Workflow

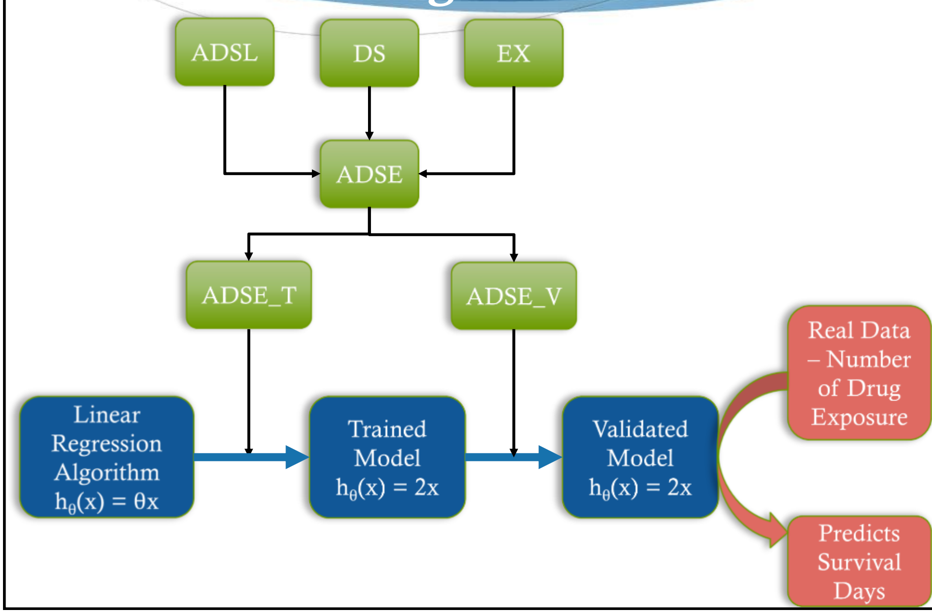
1. Identify the problems to solve
2. Acquire necessary data
3. Transform and clean data
4. Prepare train data and validation data
5. Select an algorithm
6. Train an algorithm with train data
7. Validate the trained model with validation data
8. Solve the problems/predict with the validated model

Machine Learning Project Workflow

Step	Project Workflow	Example
1	Identify the problems to solve	Find the pattern between survival days vs drug exposures
2	Acquire necessary data	Obtain ADSL, DS and EX
3	Transform and clean data	Prepare ADSE
4	Prepare train data and validation data	Prepare ADSE_T and ADSE_V
5	Select an algorithm	Import a linear regression algorithm (sklearn.linear_model)
6	Train an algorithm with train data	Train selected a linear regression algorithm with ADSE_T
7	Validate the trained model with validation data	Validate the trained model with ADSE_V
8	Solve the problems/predict with the validated model	Use the validated model to predict survival days vs drug exposures

9

ML Project Workflow of Data and Algorithm

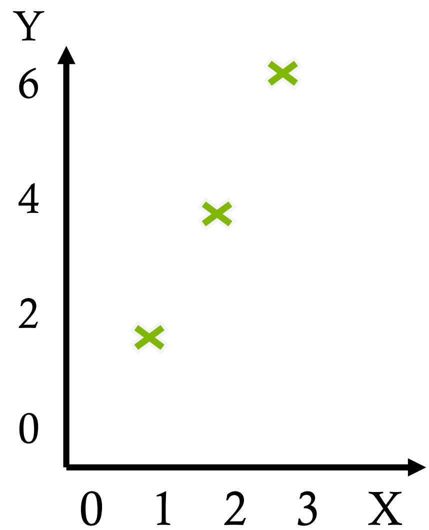


10

ML Algorithm Training with Test Data

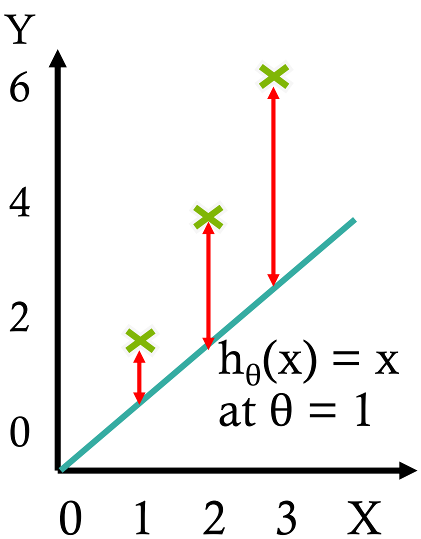
In ADSE_T,
- Y = Survival Days in months
- X = number of drug exposures

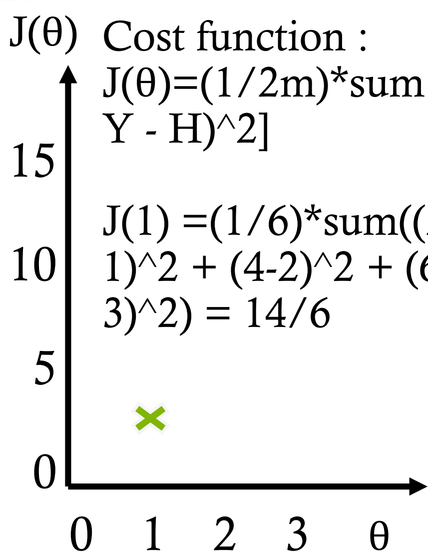
Algorithms – Linear Regression ($h_\theta(x) = \theta x$)



11

Hypothesis Function and Cost Function



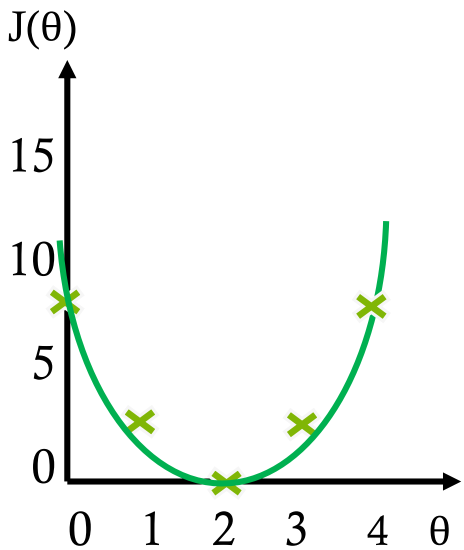


Cost function :
J(θ)=(1/2m)*sum[(Y - H)^2]

J(1) =(1/6)*sum((2-1)^2 + (4-2)^2 + (6-3)^2) = 14/6

12

Cost Function in different Parameters



- J(0) =56/6 = 9.333
- J(1) =14/6 = 2.333
- J(2) =0/6 = 0
- J(3) =14/6 = 2.333
- J(4) =56/6 = 9.333

Optimum θ is 2 –Best fitted model is H = 2X.

Question – How can Machine find the optimum θ?

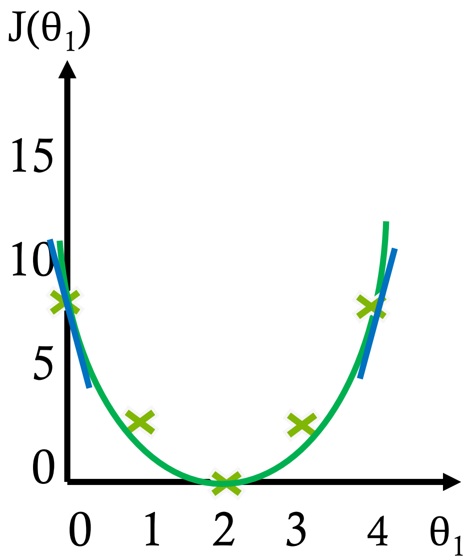
13

Gradient Descent

- Definition – Engine that minimize cost function.
- The process of getting to the lowest error value (optimum cost function)
- It is like walking down to the valley in the mountain to find the gold located in valley.

14

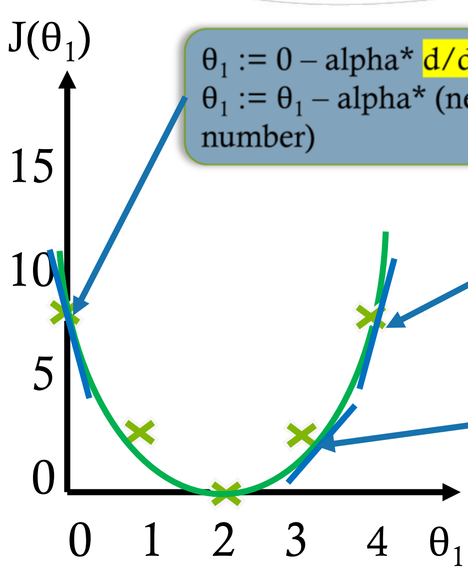
Cost function with Gradient Descent



- Take the derivative of cost function since it will give the direction to move forward : $d/d\theta_1\ J(\theta_1)$
- Make steps down the cost function : $\theta_1 := \theta_1 - d/d\theta_1\ J(\theta_1)$
- The size of steps is determined by learning rate : $\theta_1 := \theta_1 -$ alpha* $d/d\theta_1\ J(\theta_1)$

15

Cost function with Gradient Descent



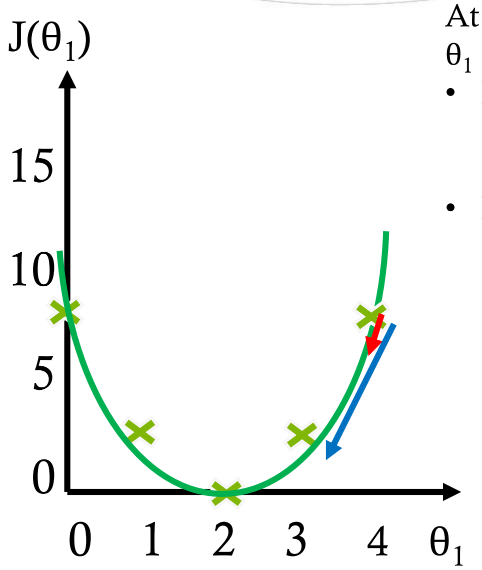
$\theta_1 := 0 -$ alpha* $d/d\theta_1\ J(0)$
$\theta_1 := \theta_1 -$ alpha* (negative number)

$\theta_1 := 4 -$ alpha* $d/d\theta_1\ J(4)$
$\theta_1 := \theta_1 -$ alpha* (positive number)

$\theta_1 := 3 -$ alpha* $d/d\theta_1\ J(3)$

16

Learning Rate

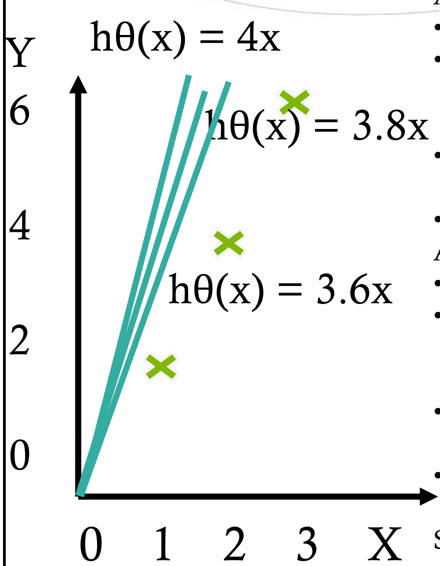


At $\Theta_1 := \theta_1 - \text{alpha}^* \, d/d\theta_1 J(\theta_1)$ at $\theta_1 = 4$.

- If the learning rate (alpha) = 0.1, $\Theta_1 := 4 - 0.1^*2 = 3.8$ assuming $d/d\theta_1 J(4) = 2$.
- If the learning rate (alpha) = 0.5, $\Theta_1 := 4 - 0.5^*2 = 3$ assuming $d/d\theta_1 J(4) = 2$.

17

Machine Learning in each iteration



At first iteration,

- Initial Hypothesis function - $h_\theta(x) = 4x$
- Cost function - $J(\theta_1) = (1/2m)^*\text{sum}[(Y - h)^\wedge 2] = 1/(2^*3)^* \text{sum}(((2-4)^\wedge 2 + (4-8)^\wedge 2 + (6-12))^\wedge 2)) = 56/6 = 9.333$
- Gradient Descent – $\theta_1 := \theta_1 - \text{alpha}^* \, d/d\theta_1 \, J(\theta_1) = 4 - 0.1^*2 = 3.8$
- Final Hypothesis function - $h_\theta(x) = 3.8x$

At second iteration,

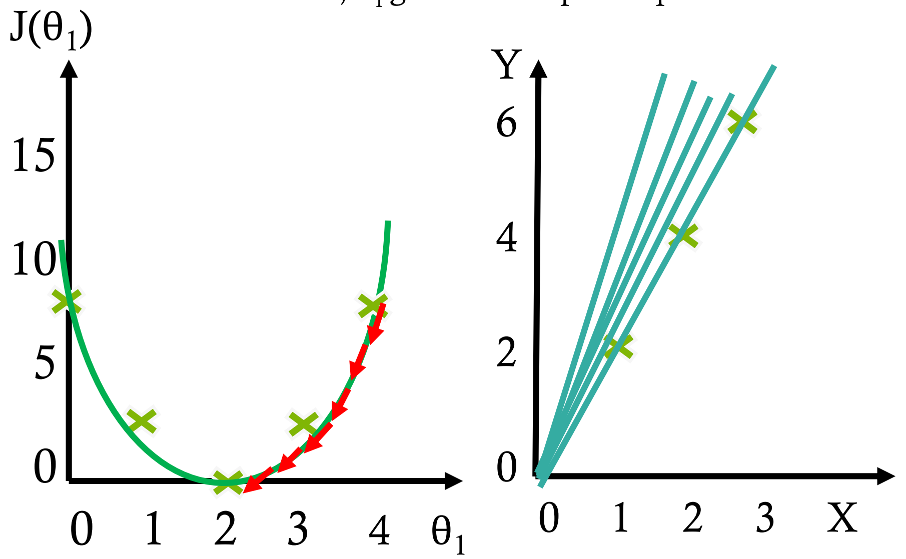
- Initial Hypothesis function - $h_\theta(x) = 3.8x$
- Cost function - $J(\theta) = 1/(2^*3)^* \text{sum}(((2-3.8)^\wedge 2 + (4-7.6)^\wedge 2 + (6-11.4)^\wedge 2)) = 45.36/6 = 7.56$
- Gradient Descent – $\theta_1 := \theta_1 - \text{alpha}^* \, d/d\theta_1 \, J(\theta_1) = 3.8 - 0.1^*2 = 3.6$
- Final Hypothesis function - $h_\theta(x) = 3.6x$

So, the cost function decrease and slope (θ_1) decrease as well.

18

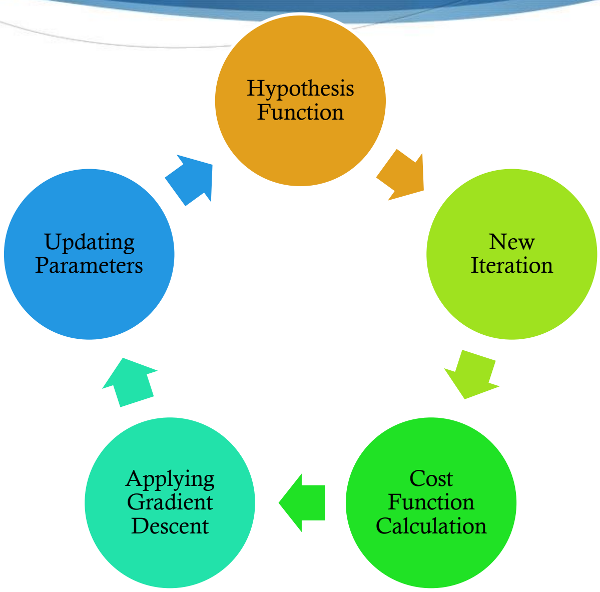
How Cost function and Gradient Descent find the best model

At each iteration, θ_1 goes to the optimal point.



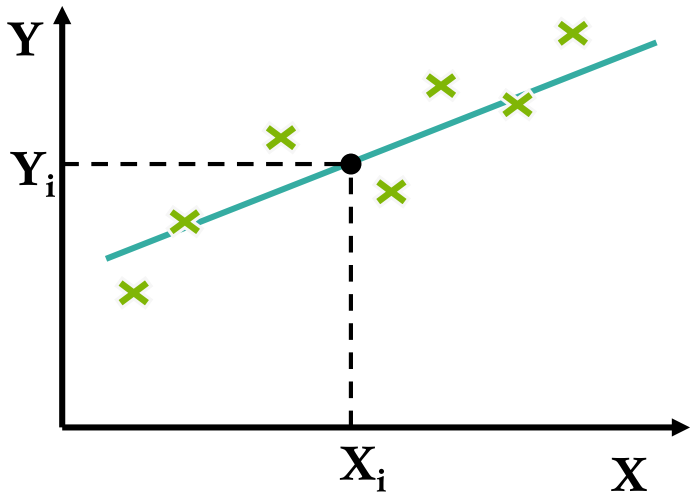
19

Model building in Machine Learning

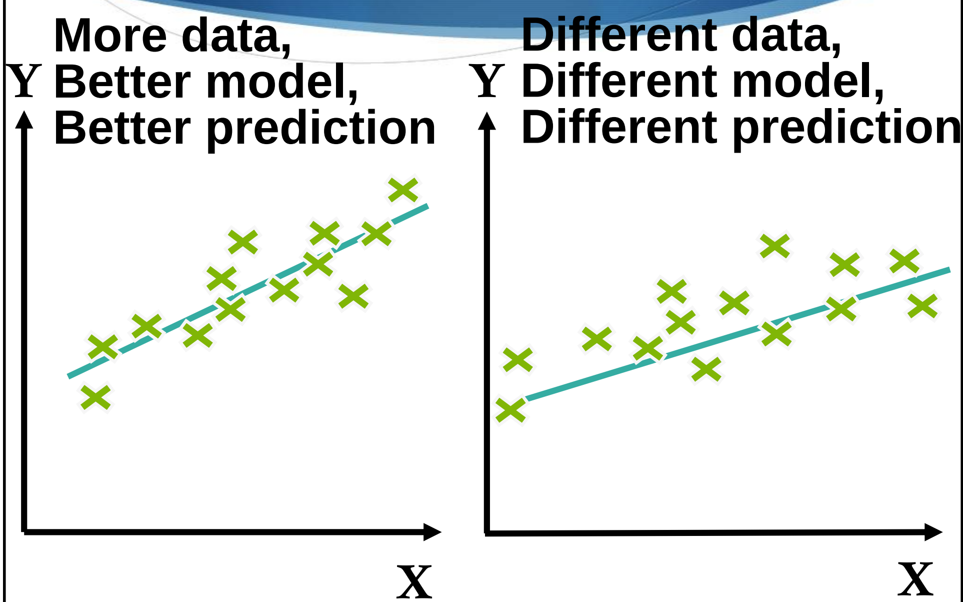


20

Best model can provide best predicted value

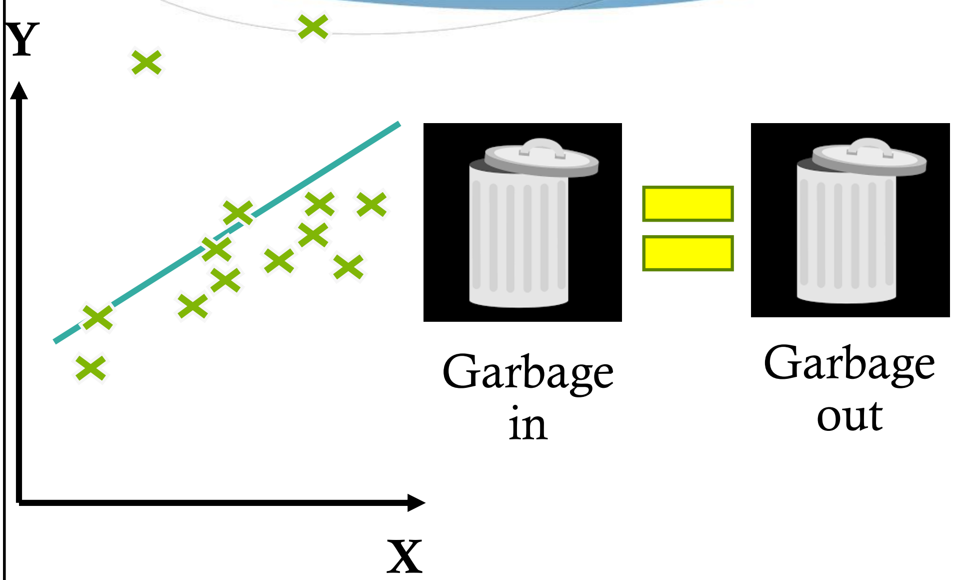


21



22

Bad data, Bad model, Bad prediction



23

Machine Learning Type

- Supervised
  - Input data labeled : has correct answers
  - Specific purpose
  - Types
    - Classification for distinct output values
    - Regression for continuous output values
- Unsupervised
  - Input data not-labeled : No correct answers
  - Exploratory
  - Types : Clustering (clusters)

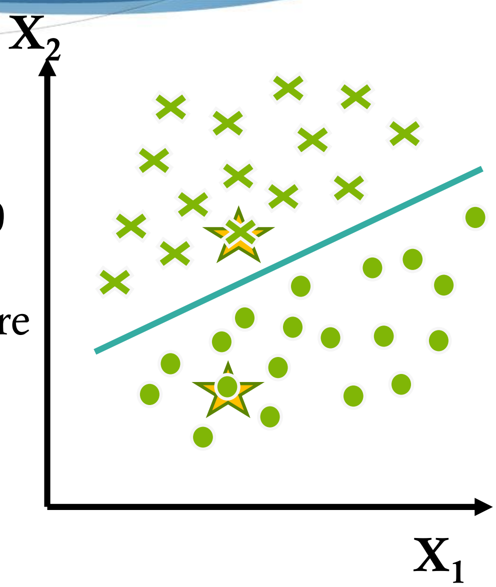
X_0	X_1	X_2	…	Xn	Y

X_0	X_1	X_2	…	Xn

24

Classification

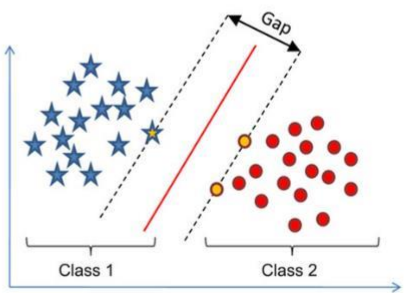
- Categorical target
- Binary or Classification
- Example : Yes/No, 0 to 9, mild/moderate/severe
- Logistic Regression, SVM, Decision Tree, Random Forests

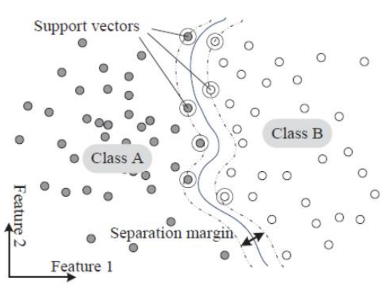


25

Support Vector Machine (SVM)

- SVM is one of the most powerful classification model, especially for complex, but small/mid-sized datasets.
- Best for binary classification
- Easy to train
- It finds out a line/ hyper-plane (in multidimensional space that separate outs classes) that maximizes margin between two classes.
- Hyperparameters – Kernel (linear, polynomial), gamma, margin(a separation of line to the closest class points)

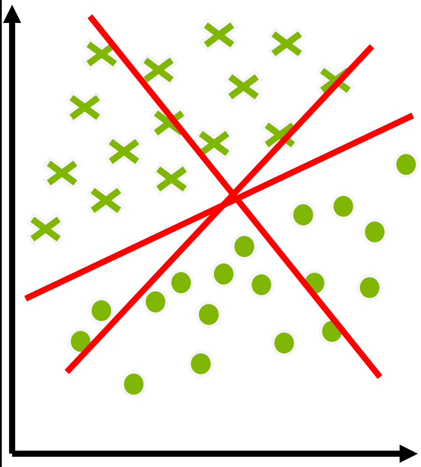




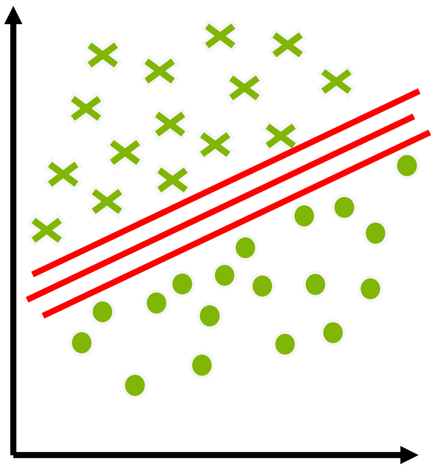
26

How SVM works

Identify right hyperplane (line)



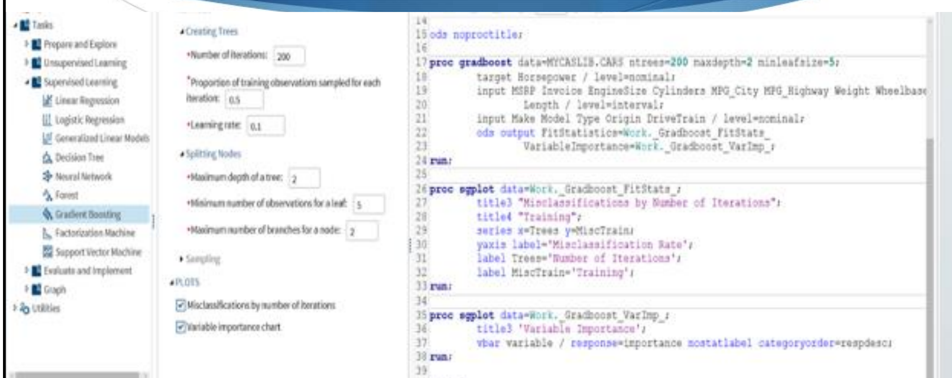
Identify right hyperplane (line)



Goals – to minimize distance between support vectors and hyperplane.

27

SAS Machine Learning Packages

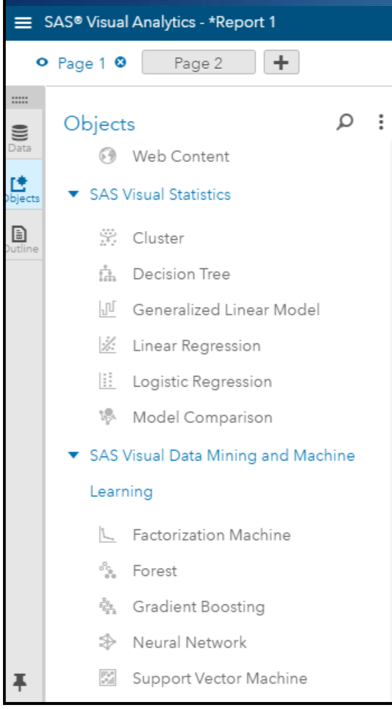


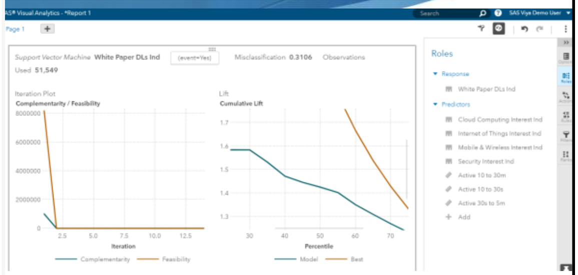
SAS Visual Data Mining and Machine Learning

- Linear Regression
- Logistic Regression
- Support Vector Machine
- Artificial Neural Networks (limited layers)

28

SVM in SAS Visual Data Mining and Machine Learning





```
*** SVM SAS codes;
proc svmachine data=x_train C=1.0;
        kernel linear;
        input x1 x2 x3 x4 / level=interval;
        target  y;
run;
```

29

Python in Machine Learning

Most popular programming language in Machine Learning

- Scikit-Learn
  - simple and efficient ML tool, open-source
  - Classification, Regression, SVM, Clustering, Dimensionality Reduction, Decision Trees, Random Forests
- TensorFlow
  - developed by Google, open source since Nov.'2015
  - Artificial Neural Network, Convolutional NN, Recurrent NN, Autoencoder, Reinforcement Learning
- Keras
  - The official high-level API of TensorFlow
  - Easier and more friendly than TensorFlow
  - Developed by Google and contributed by MS, AWS

30

Python codes for SVM

```
#import ML algorithm
from sklearn.svm import SVC

#prepare train and test datasets
x_train = ...
y_train = ....
x_test = ....

#select and train model
svm = SVC(kernel='linear', C=1.0, random_state=1)
svm.fit(x_train, y_train)

#predict output
predicted = svm.predict(x_test)
```

31

SVM implementation

Pros:

- It works really well with clear margin of separation.
- It works well in high dimensional spaces.
- It works well where number of dimensions is greater than the number of samples.

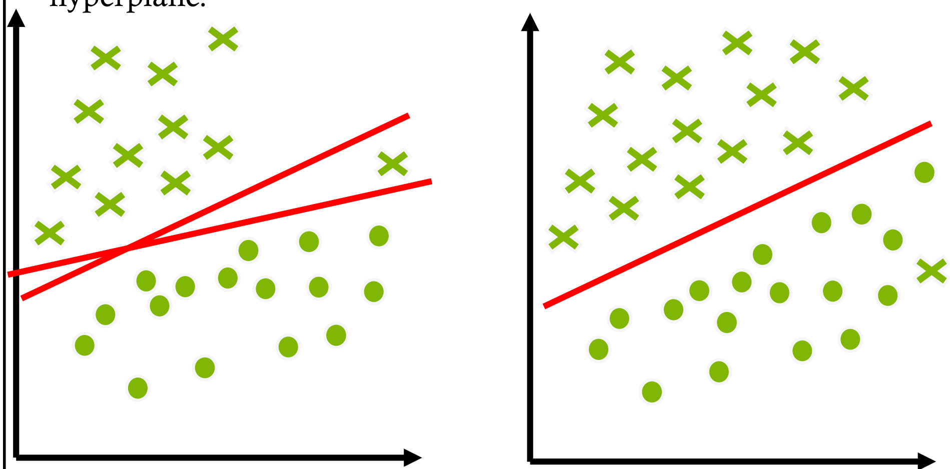
Cons:

- It requires higher computation power, so it does not work well with larger data.
- It also doesn't perform very well when the data set has a lot of noises.

32

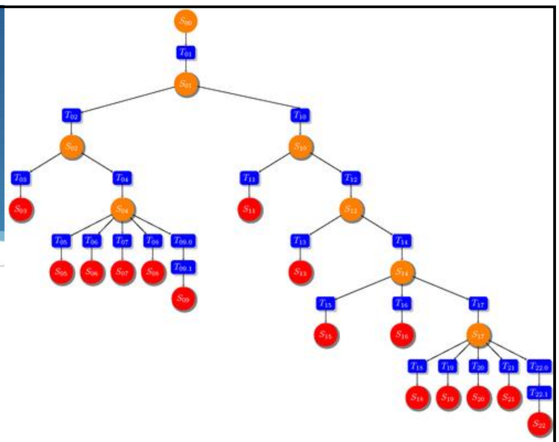
Questions : What is hyperplane for SVM?

Goals – to minimize distance between support vectors and hyperplane.



33

Decision Trees



The widely used machine learning algorithm that use tree to visually and explicitly represent decision and its conditions.

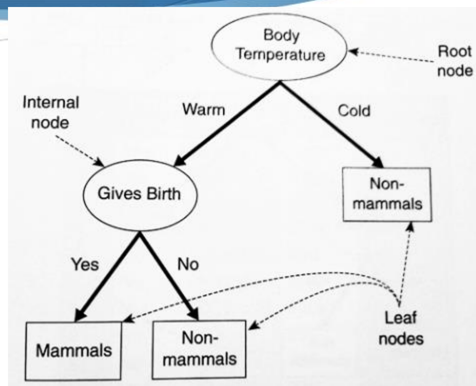
Why Decision Trees?

- Easy to understand and interpret
- Require a very little data preparation
- Possible to validate model using statistical tests.

34

Decision Trees Architecture

- Root node – the topmost node in a tree
- Internal nodes - condition
- Branches – results from condition
- Splitting - a process of dividing a node into two or more sub-nodes.
- Leaf nodes (decision) – the end of the branch that does not split any more



Hyperparameters that you need to set in Decision Trees Models
- Maximum number of features(variables)
- Depth of branches – min / max
- Criteria – the measure of quality of a split (gini / information gain)

35

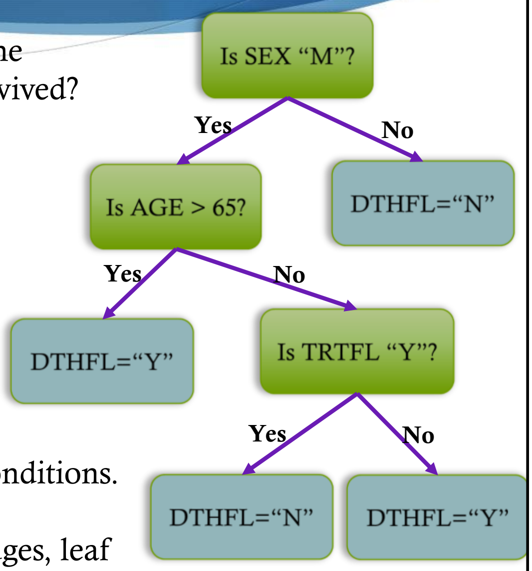
Decision Trees Use Cases

Problems to solve – What is the condition of patients who survived?

ADSL dataset
- features (input variables)
  - SEX
  - AGE
  - TRTFL
- labels (results)
  - DTHFL

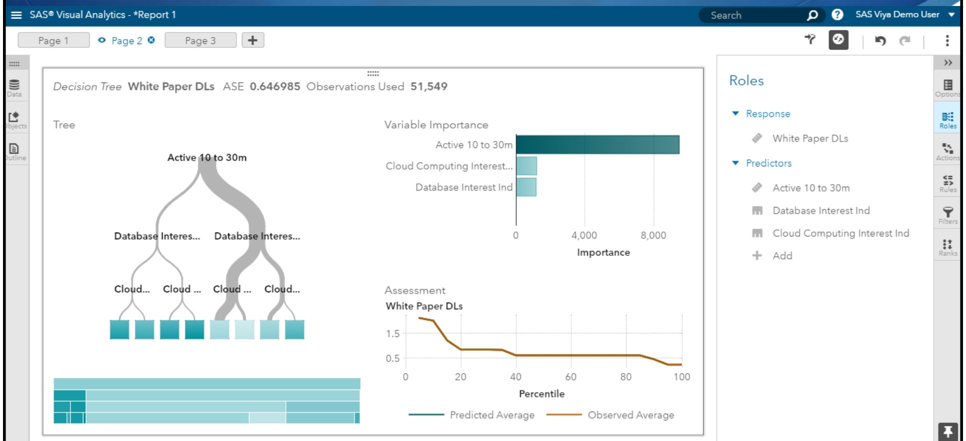
Decision Trees find out the conditions.

Questions?: internal nodes, edges, leaf nodes



36

Decision Tree in SAS Visual Data Mining and Machine Learning




```
PROC HPSPLIT data = ADAE maxleaves=100
      maxbranch = 4 leafsize=1 ;
      model Y(event='y') = x1 x2 x3 x4;
Run;
```

37

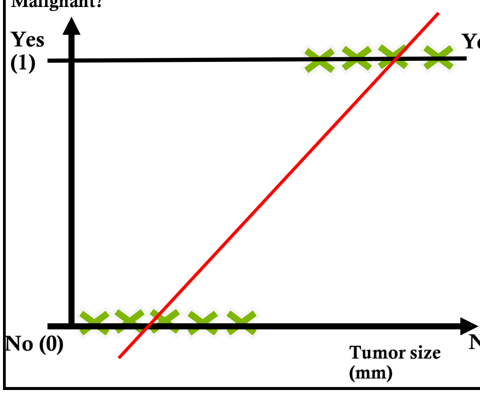
Python codes for Decision Tree

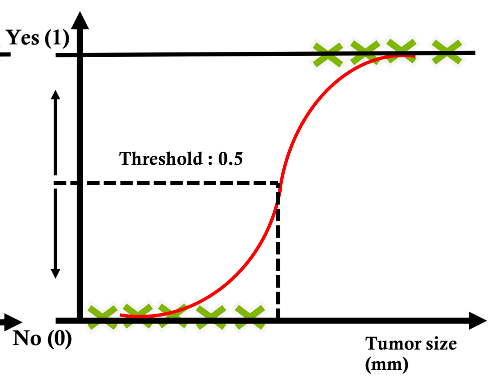
```
#import ML algorithm
from sklearn.tree import DecisionClassifier

#prepare train and test datasets
x_train = …
y_train = ….
x_test = ….

#select and train model
d_tree = DecisionClassifier(max_depth=4)
d_tree.fit(x_train, y_train)

#predict output
predicted = d_tree.predict(x_test)
```

38

Logistic Regression

- Output: Binary target (Yes/No)
- Input: Continuous/categorical variables
- Example : predicting if the tumor is malignant based on tumor size



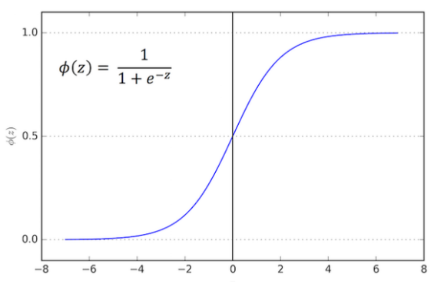


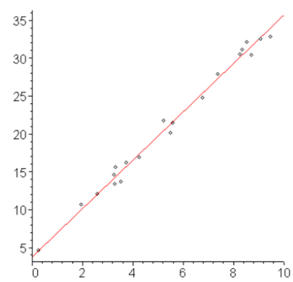
39

Logistic Regression (Logistic + Regression)

- Logistic function: Sigmoid function, which describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. Any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

- Regression function: a set of statistical processes which estimate the relationships among variables. It indicates the relationship how the dependent variable changes when any one of the independent variables is varied, while the other independent variables are held fixed. $z = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 +,, + \theta_n x_n$

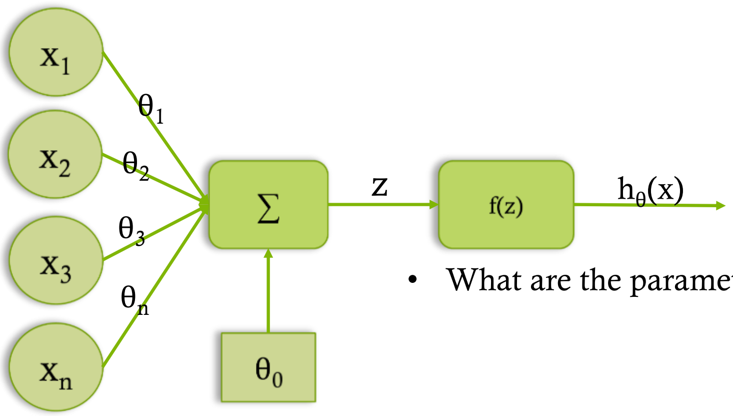




40

Logistic Regression Architecture

- Hypothesis function : $h_\theta(x) = 1 / (1 + e^{-z})$
  where $z = (\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 +,, + \theta_n x_n)$



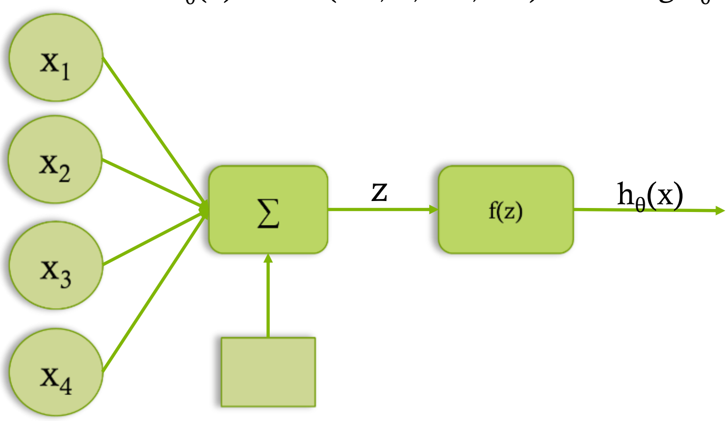
- What are the parameters?

41

Logistic Regression Exercise

Hypothesis function : $h_\theta(x) = 1 / (1 + e^{-z})$
where $z = (2x_0 + x_1 + 5x_2 + 6x_3 + 0.5x_4)$

- Put the parameters
- Calculate z and $h_\theta(x)$ if $x = (0.1, 2, 0.2, 0.4)$ assuming $x_0 = 1$



42

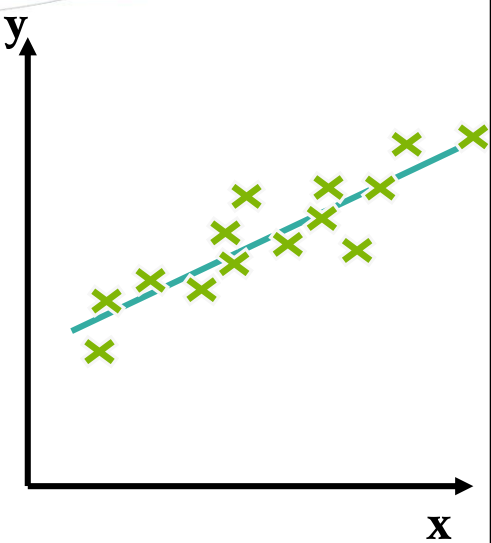
Python codes for Logistic Regression

```
#import ML algorithm
from sklearn.linear_model import LogisticRegression

#prepare train and test datasets
x_train = …
y_train = ….
x_test = ….

#select and train model
lr = LogisticRegression()
lr.fit(x_train, y_train)

#predict output
predicted = lr.predict(x_test)
```

43

Regression

- To predict values of a desired target quantity when the target quantity is continuous.
- Output: Continuous variables
- Example : predicting house price per sqft
- Types: Linear Regression, Polynomial Regression
- Hypothesis function:
  - $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 +,, + \theta_n x_n$
  - $y = 10 + 2x$



44

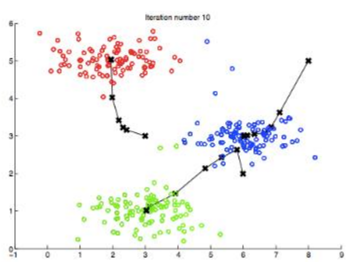
Python codes for ML Linear Regression

```
#import ML algorithm
from sklearn import linear_model

#prepare train and test datasets
x_train = …
y_train = ….
x_test = ….

#select and train model
linear = linear_model.LinearRegression()
linear.fit(x_train, y_train)

#predict output
predicted = linear.predict(x_test)
```

45

Unsupervised Machine Learning

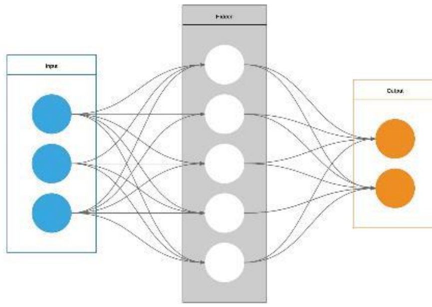
- Input data not-labeled – no correct answers
- Exploratory
- Clustering – the assignment of set of observations into subsets (clusters) so that data in the same clusters are more similar to each other than to those from different clusters.
- Algorithms – k-means
- Industry implementation – the grouping of documents, music, movies by different topics, finding customers that share similar interests based on common purchase behaviors.



46

Artificial Neural Network (ANN)

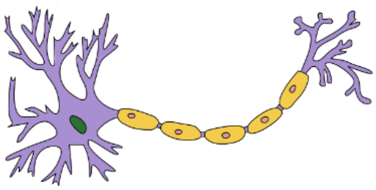
- Most powerful ML algorithm
- Game Changer
- Works very similar to human brain – Neural Network
- Types – DNN, CNN, RNN
- Architecture
  - Artificial Neurons
  - Input Layers
  - Hidden Layers
  - Output Layers
  - Activation Functions

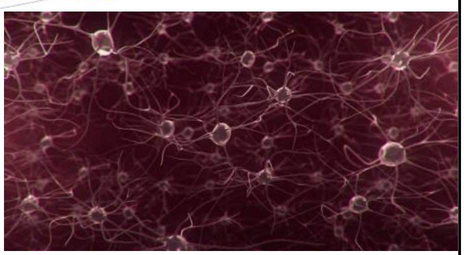


47

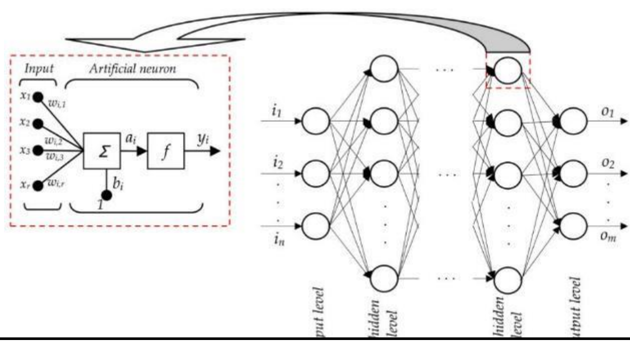
Human Neural Network vs Artificial Neural Network (ANN)

- Human Neurons : Neural Network – 100 billions



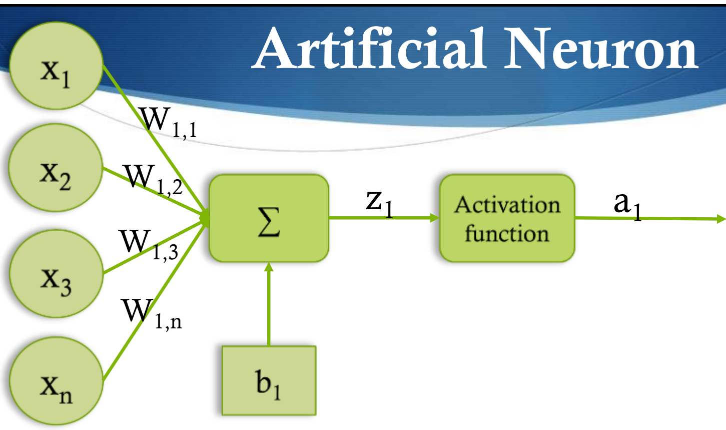


- Artificial Neurons : Artificial Neural Networks



48

Artificial Neuron



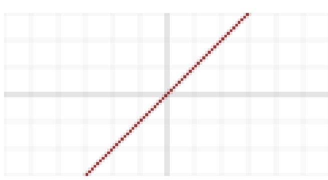
- Input (x_1, x_2, x_3,,,,x_n) – numeric values that goes into neuron model
- Weight (w_1, w_2, w_3,,,,w_n) – weight of each input values
- Bias (b_1) – bias of input* weight
- Artificial neuron
  - Model (input values * weights + bias): $z_1 = x_1w_{1,1} + x_2w_{1,2} + x_3w_{1,3} +,,,, + x_nw_{1,n} + b_1$
  - Activation function : $a_1 = f(z_1)$

49

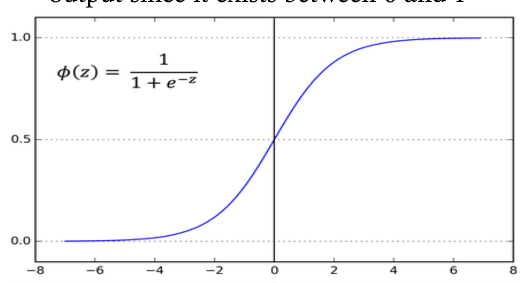
Activation Functions

- Introduction – a node that is added to the output of NNs. Sometime, it could be added between two NNs.
- Purpose – to provide boundary of outputs.

- Linear function
  - f(x) = ax + b
  - Ex. Regression



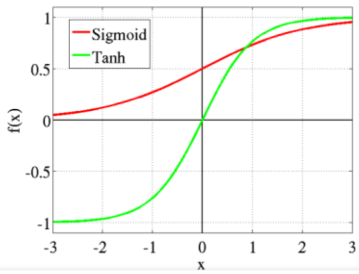
- Sigmoid or Logistic function
  - $f(x) = 1 / (1 + e^{-x})$
  - Ex. Logistic Regression, DNN, CNN
  - Purpose – to predict the probability as an output since it exists between 0 and 1



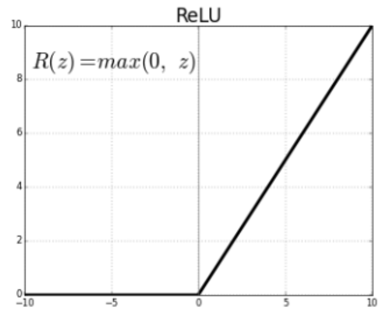
50

Activation Functions

- Tanh or Hyperbolic Tangent function
  - f(x) = tahn(x) = 2 / (1 + e^{-x}) – 1
  - Ex. RNN
  - mainly used for classification between two classes



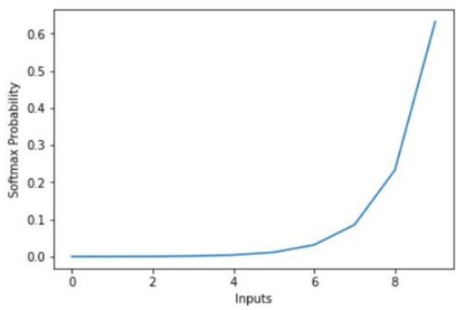
- ReLU (Rectified Linear Unit) function
  - f(x) = 0 for x < 0 ; x for x>= 0
  - Ex – CNN, DNN
  - the most used activation function in the world right now
  - Speeding up the computation



51

Activation Functions

- Softmax function
  - f(x_i) = e^{-xi} / sum(e^{-x})
  - Unlike Sigmoid, the sum does not need to add up to 1
  - Usage – RNN
  - To provide the probability of a given output. The model will select the best probability.
  - Text Sensitivity Analysis – read texts and predict the mood of text



Output	Sad	Fine	Happy	Very Happy
Softmax function	0.02	0.2	0.89	0.43

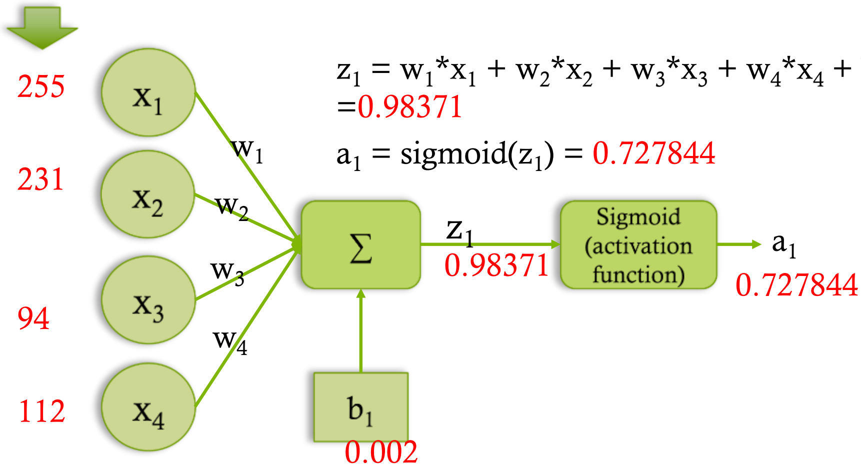
52

Artificial Neuron - example

W = [0.00045, 00056, 0.00077, 0.00606]

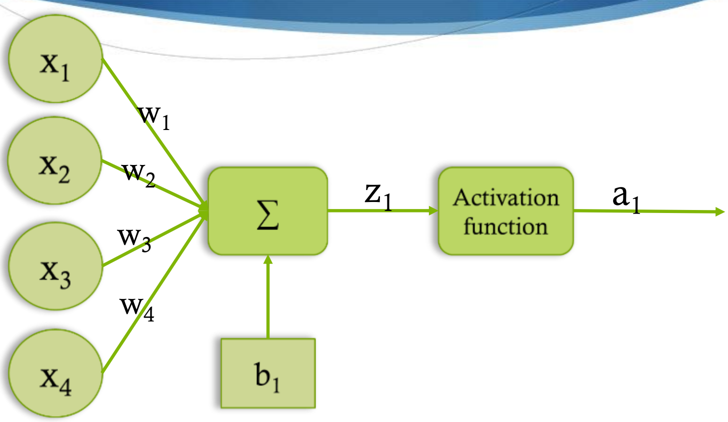
$z_1 = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + w_4 * x_4 + b_1$
$= 0.98371$

$a_1 = \text{sigmoid}(z_1) = 0.727844$



53

What "Artificial Neuron" learns

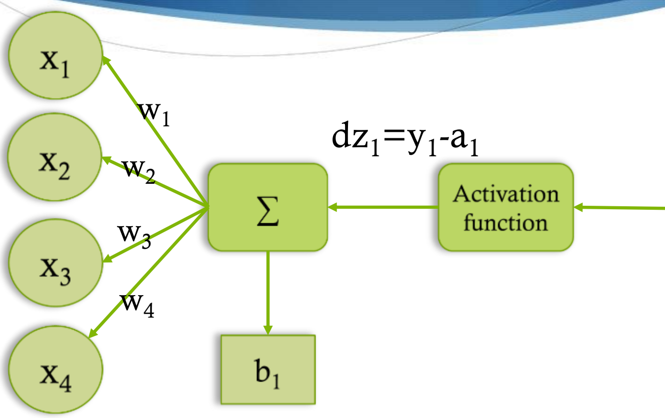


Machine Learning – To learn parameters
- Weights (w_1, w_2, $w_{3,} w_4$)
- Bias (b_1)

How many parameters in this neuron?

54

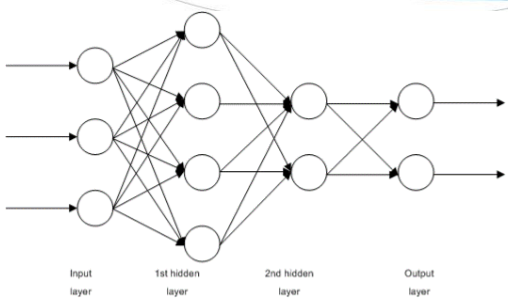
How "Artificial Neuron" learns



- Cost function of ANN (actual output – predicted value) is $J(w ,b) = (1/m)^* \text{ sum } [\ell(y_1 - a_1)]$
- Gradient descent/optimization
  - w := w – alpha*dJ/dw
  - b := b – alpha*dJ/ db

55

ANN Architecture



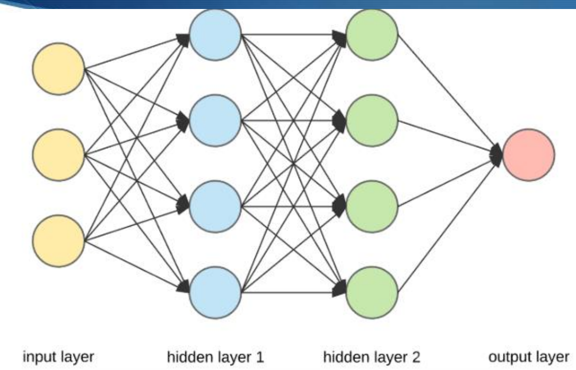
- Input layer - 3 features (variables)
- Hidden layer
  - Hidden layer1 - 4 neurons
  - Hidden layer2 - 2 neurons
- Output layer – 2 outputs

- How many weights at hidden layer 1? 3*4 = 12
- How many weights at hidden layer 2? 4*2 = 8
- How many weights at output layer? 2*2 = 4
- How many biases at hidden layer 1? 4
- How many biases at hidden layer 2? 2
- How many biases at output layer? 2

- How many parameters all together? (12+4) + (8+4) + (4+2) = 34

56

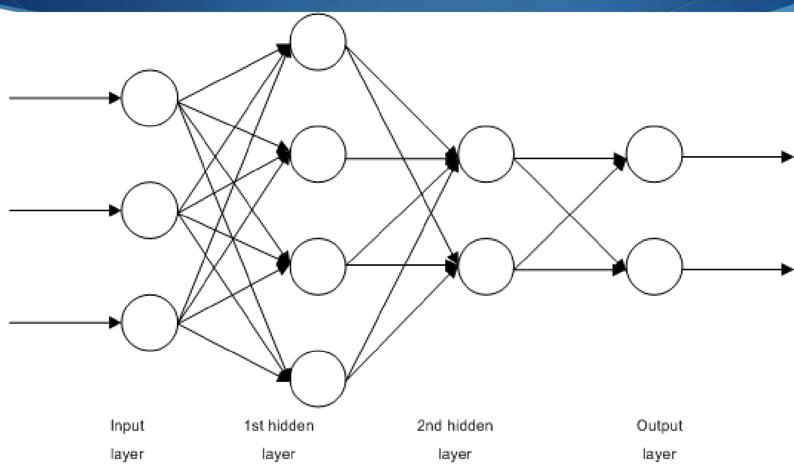
ANN Architecture Exercise



- How many input variables?
- How many weights at hidden layer 1?
- How many weights at hidden layer 2?
- How many weights at output layer?
- How many biases at hidden layer 1?
- How many biases at hidden layer 2?
- How many biases at output layer?
- How many parameters all together?

57

Deep Neural Network (Deep Learning)



- DNN is the aggregates of individual artificial neurons.
- Basic architecture - Input layer, Hidden layer, Output layer.
- Every layer is made up of a set of neurons, which each layer is fully connected to all neurons in the layer before.

58

DNN example

To detect whether patients will have cancer or not
- Input variables– age, sex, race, weight, height, family history
- Outcome – yes/no

To detect numbers from 0 to 9 from image
- Input variables – 28 by 28 pixel image
- Outcome – 0, 1,2,,,,, 9

To detect AE events from text message
- Input variables – texts, comments from social media
- Outcome - AE or not

To detect sensitivity from text message
- Input variables – blogs, comments from social media
- Outcome - “Very unhappy”, “Unhappy”, “Happy”, “Very happy”

59

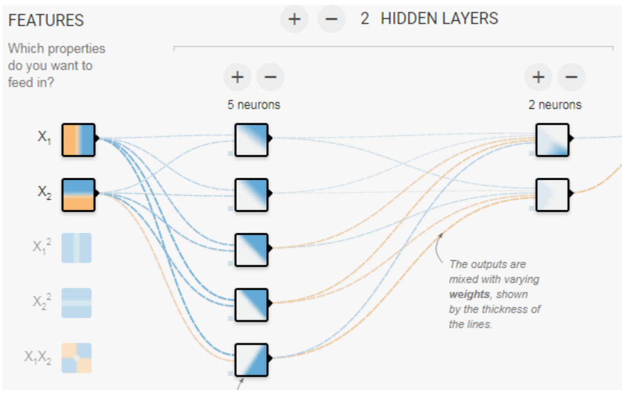
Deep Learning Project Workflow

Step	Project Workflow	Example
1	Identify the problems to solve	Find the pattern of data
2	Acquire necessary data	Obtain data
3	Transform and clean data	Prepare data for Deep Learning
4	Prepare train data and validation data	Prepare train data and validation data
5	Select an algorithm and hyperparameters	Select features(variables), number of hidden layers, number of neurons in each layer, activation function, learning rate and other parameters.
6	Train an algorithm with train data	Train the selected model with train data
7	Validate the trained model with validation data	Validate the trained model with test data
8	Solve the problem/predict with the validated model	Use the validated model to predict.

60

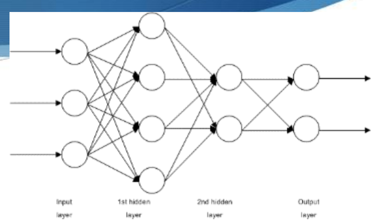
Tensor Flow Demo

http://playground.tensorflow.org



61

Python codes for DNN




```
#import ANN - TensorFlow
import tensorflow as tf

X = tf.placeholder(..)
Y = tf.placeholder(..)
hidden1 = tf.layer.dense(X, 4, activation=tf.nn.relu)
hidden2 = tf.layer.dense(hidden1, 2, activation=tf.nn.relu)
logits = neuron_layer(hidden2, 2)
….
loss = tf.reduce_mean(….)
optimizer = tf.train.GradientDescentOptimezer(0.1)
training_op = optimizer.minimizer(loss)

tf.Session.run(training_op, feed_dict={X:x_train, Y:y_train})
```

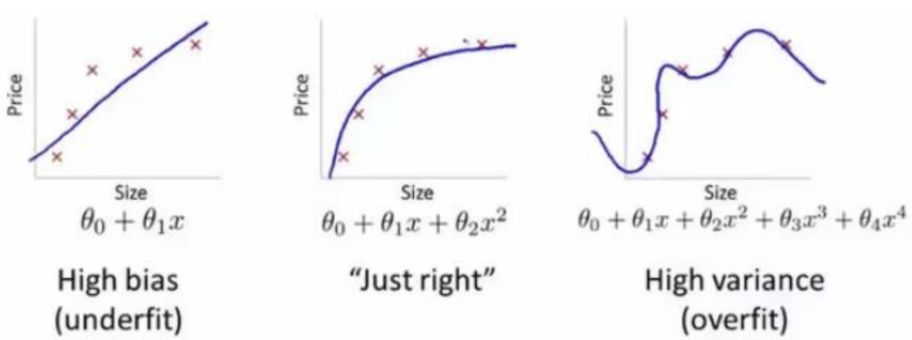
62

DNN improvement – Bias vs Variance

- Bias
  - The assumption made by a model to make target function easier to learn.
  - Kind
    - Low Bias - Less assumption to the target function.
    - High Bias - More assumption to the target function. It leads to "Underfitting" to model. Training data underperforms.
  - It is the same as MSE (Mean Square of Error)
- Variance
  - The amount that the estimate of the target function will change if different training data was used
  - Kind
    - Low Variance – Small changes for the estimated target function with changes to the data.
    - High Variance – Big changes for the estimated target function with changes to the data. It leads to "Overfitting" to model. Training data performs well, but Validation data underperforms.

63

DNN improvement – Bias vs Variance



64

DNN improvement – Bias-Variance Tradeoff

- The goals - to achieve low Bias and low Variance.
- General trends
  - Parametric or linear machine learning algorithms often have a high bias but a low variance.
  - Non-parametric or non-linear machine learning algorithms often have a low bias but a high variance
- General relationship between Bias and Variance
  - Increasing the bias will decrease the variance.
  - Increasing the variance will decrease the bias
- General resolution
  - To fix High Bias (Underfitting)
    - Adding more features
    - Adding polynomial features
    - Decreasing Lambda
  - To fix High Variance (Overfitting)
    - Getting more training examples
    - Trying smaller set of features
    - Increasing Lambda

65

Regularization in Machine Learning

- The goals - to discourage more complex or flexible model, so as to avoid the risk of overfitting.
- Regularization term in ML model
  - the cost function is regulated by the regularization term.
  - Lasso (L1) Regularization

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \boxed{\lambda \sum_{i=1}^{k} |w_i|}$$

  - Ridge (L2) Regularization

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \boxed{\lambda \sum_{i=1}^{k} w_i^2}$$

  - Lambda is tuning parameter that decides how much we want to penalize the flexibility of our model.

66

L1 vs L2 Regularization

- Lasso (L1) Regularization
  - To prevent weights from rising too high.
  - As Lambda increase (to infinity), the weight parameters goes to 0.
  - To add “Squared magnitude” of coefficient as penalty term to the loss function
  - To shrink the less important features’ coefficient to zero, removing some features all together – works well for feature selection.

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

Cost function

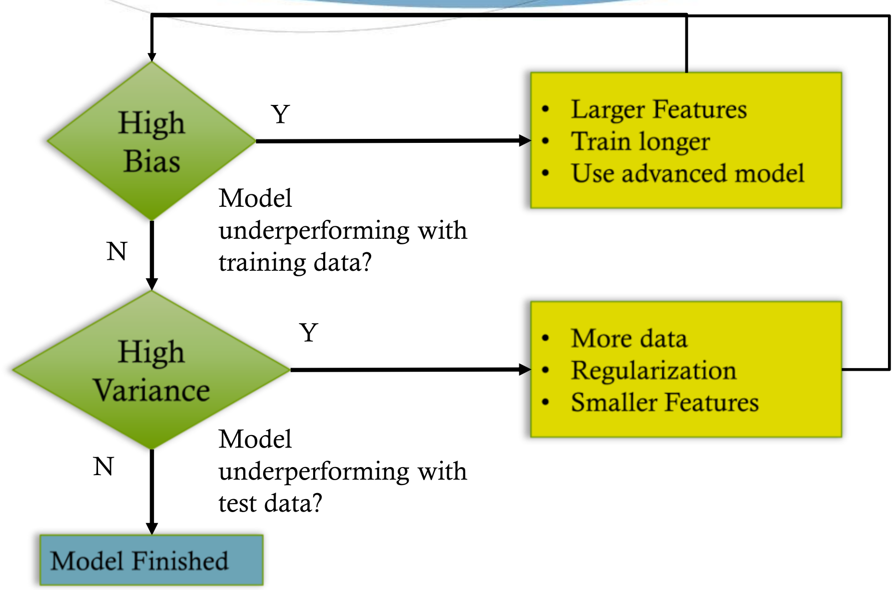
- Ridge (L2) Regularization
  - To add “Absolute value of magnitude” of coefficient as penalty term to the loss function.
  - Work well to avoid over-fitting issue.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

Cost function

67

The General procedures to remove high Bias & Variance



68

Convolutional Neural Network (CNN)

Image-specific Artificial Neural Network – image search/recognition, face recognition, face verification, self-driving and more

Its concept comes from brain visual cortex where many neurons have a small region of the visual field (in CNN, filter)

The name of CNN comes from "convolution", one of the most important operation in CNN.

The first Convolutional Neural Network is LeNet-5, which can classify digits from hand-written number.

Advantage compared to DNN
- 2 or 3 dimensional read (image rather than vector)
- Less parameters, much less computing power

Image Data

When the computer see the image, it sees pixel values.

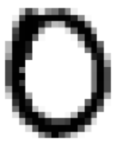
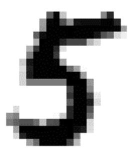


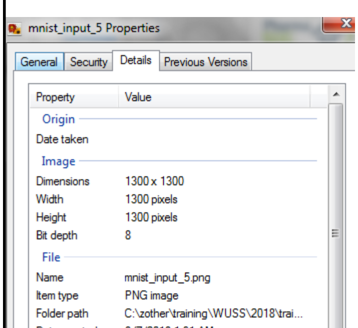
Image data in .png is “0”. It is 28 by 28 pixels. This can be converted into 28 by 28 number.

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  64 191 118  24 118 138 191 118  17   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  17 193 253 252 252 252 252 253 252 227 120   5   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  93 252 253 252 233 141  69  79 227 252 252 137   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 187 252 253 252  89   0   0   0  29 154 252 252  95   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 207 252 253 157   6   0   0   0   0   7 158 252 220   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  70 253 253 231   0   0   0   0   0   0   0  43 241 255 154   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  70 252 252 230   0   0   0   0   0   0   0   0 135 253 206   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0 101 252 252 167   0   0   0   0   0   0   0   0   0 222 240  50   0   0   0   0   0]
 [  0   0   0   0   0   0   0 184 252 252   0   0   0   0   0   0   0   0   0   0 138 252 111   0   0   0   0   0]
 [  0   0   0   0   0   0   0 184 252 147   0   0   0   0   0   0   0   0   0   0 138 252 183   0   0   0   0   0]
 [  0   0   0   0   0   0   0 185 253 253   0   0   0   0   0   0   0   0   0   0 139 253 184   0   0   0   0   0]
 [  0   0   0   0   0   0   0 184 252 200   0   0   0   0   0   0   0   0   0   0 138 252 183   0   0   0   0   0]
 [  0   0   0   0   0   0   0 101 252 252   0   0   0   0   0   0   0   0   0   0 138 252 183   0   0   0   0   0]
 [  0   0   0   0   0   0   0  44 236 252  95   0   0   0   0   0   0   0   0   0 233 252  89   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 207 252 220   0   0   0   0   0   0   0   0 158 253 210   6   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 145 253 244  65   7   0   0   0   0   0  62 243 252 135   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  34 217 253 252 154  30   0   0   9  78 236 252 157   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  84 222 252 252 227 184 185 197 252 252 221  32   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  36 177 252 252 252 253 252 252 176  35   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   5 107 179 252 190 137  54   4   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

71

Example



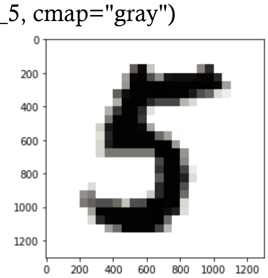


Python codes

image_5 =imread(fname='./images/mnist_input_5.png')
print('Shape of Image 5 : ', image_5.shape)
print(image_5)

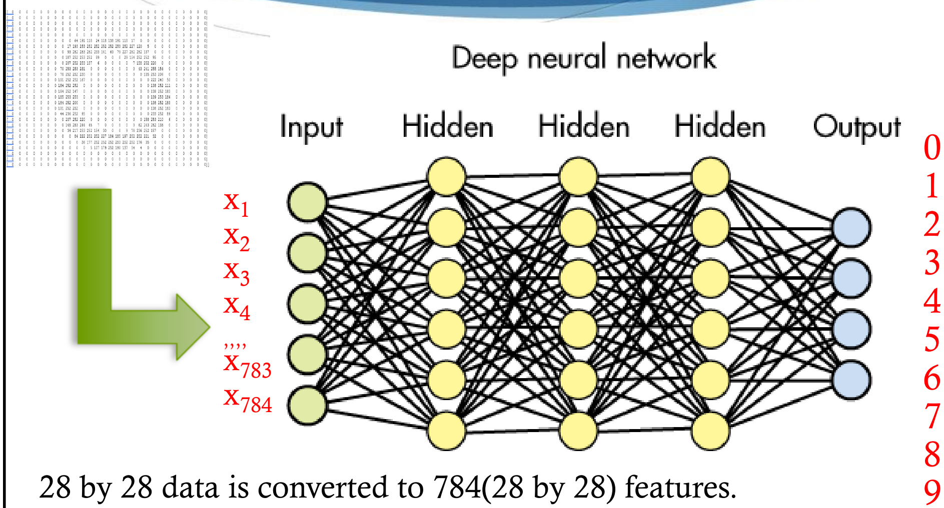
```
Shape of Image 5 : (1300, 1300)
[[255 255 255 ... 255 255 255]
 [255 255 255 ... 255 255 255]
...
 [255 255 255 ... 255 255 255]]
```

plt.imshow(image_5, cmap="gray")
plt.show()



72

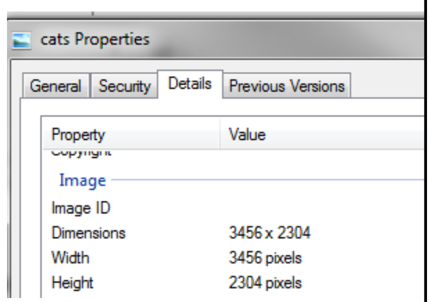
DNN on Images data



28 by 28 data is converted to 784(28 by 28) features.
784 neurons in input layer
neurons in hidden layer 1
10 neurons in output layer

73

Complex Image data




```
### import imread function
from skimage.io import imread

image_cat = imread(fname='./images/cats.jpg')
print('Shape of Image cat : ', image_cat.shape)
```

Shape of Image cat : (2304, 3456, 3), which means 2304 * 4356 * 3 = 30,108,672 features goes into DNN, a way too much computation.

74

CNN image

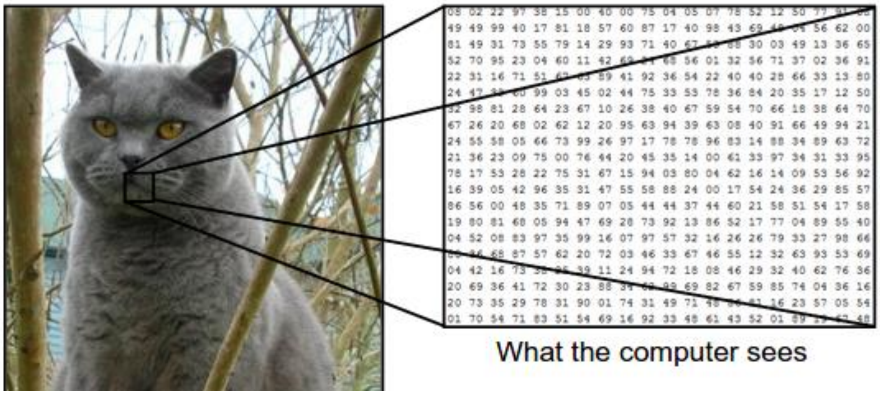
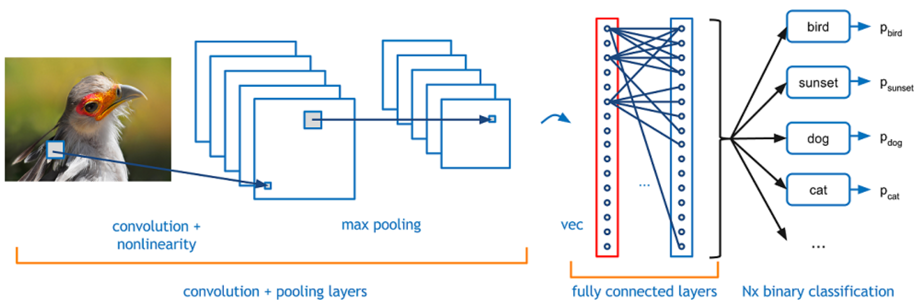


Image-specific Artificial Neural Network

75

CNN Architecture



CNN Architecture
- Input (3 dimensional vector)
- Convolution – Finding local pattern
- Pooling
- Fully-connected layers – Finding global pattern
- Classification (outputs)

76

Convolution in CNN

Convolution – the mathematical combination of two functions to produce a third function.

3*1 + 1*1 + 2*1 + 0*0 + 5*0 + 7*0 + 1*-1 + 8*-1 + 2* -1 = -5

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

input - 6 x 6

*

1	0	-1
1	0	-1
1	0	-1

parameter - 3 x 3 filter

=

-5	-4	0	8
-10	-2	-4	-7
0	-2	-4	-7
-3	-2	-3	-16

output - 4 x 4

77

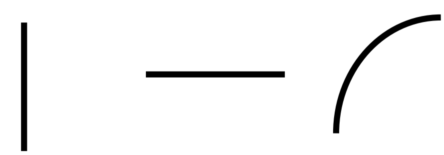
Filters in Convolution

- Filters are feature identifiers (e.g., straight edges and curves)
- Input Image * Filter
  - If high number, it has that feature identifier.
  - If low number, it does not have that feature identifier.

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

0	1	1
1	0	0
1	0	0



3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

1	0	-1
1	0	-1
1	0	-1

parameter - 3 x 3 filter

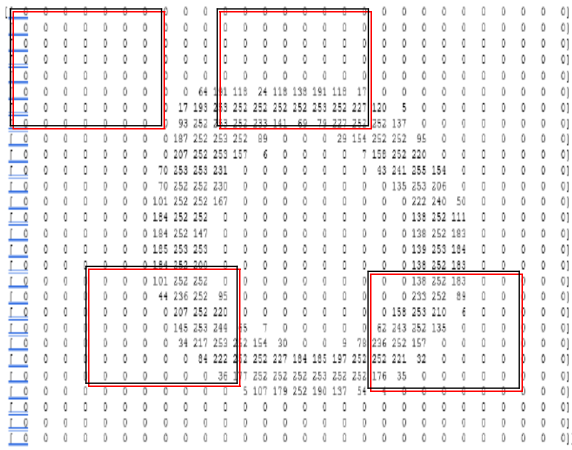
=

-5	-4	0	8
-10	-2	-4	-7
0	-2	-4	-7
-3	-2	-3	-16

output - 4 x 4

78

Convolution in image perspective



Pooling in CNN

Pooling with filters of 2 X 2

- Max Pooling – more popular than average pooling

-5	-4	0	8
-10	-2	-4	-7
0	-2	-4	-7
-3	-2	-3	-16

-2	8
0	-3

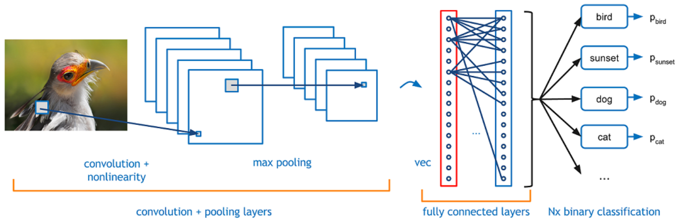
- Average Pooling
  (-5 -4 -10 -2) / 4 = -21/4= -5.25

-5	-4	0	8
-10	-2	-4	-7
0	-2	-4	-7
-3	-2	-3	-16

-5.25	-0.75
-1.75	-7.5

Pooling progressively reduces the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting.

Fully-connected layer in CNN



This layer basically takes an input volume (e.g., 7x7x32) and outputs an N (e.g., 10) dimensional vector where N is the number of classes that the program has to choose from.

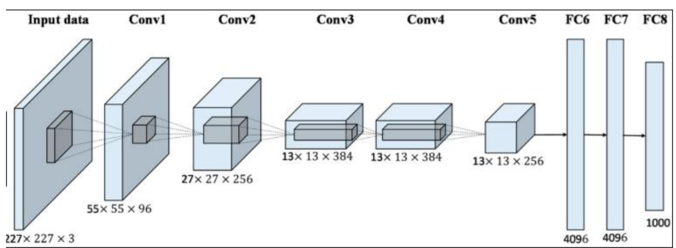
For example,

1. The resulting vector for a digit classification program is [0,.1, .1, .75, 0, 0, 0, 0, 0, .05] for number 0 to 9
2. It represents a 10% probability that the image is a 1, a 10% probability that the image is a 2, a 75% probability that the image is a 3, and a 5% probability that the image is a 9.
3. Model will predict it is "3".

81

CNN implementation History

- AlexNet
  - Famous winner of the 2012 ImageNet LSVRC-2012 competition by a large margin (15.3% VS 26.2% (second place) error rates)
  - Architecture



- This layer basically takes an input volume (e.g., 7x7x32) and outputs an N (e.g., 10) dimensional vector where N is the number of classes that the program has to choose from.

82

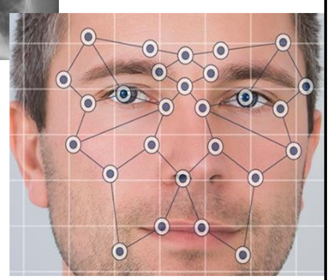
Famous CNN Architecture competition winner

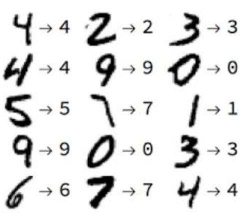
- LeNet (developed in 1998)
  - Number of Parameters : 60K
- AlexNet (developed in 2012)
  - Famous winner of the 2012 ImageNet LSVRC-2012 competition by a large margin (15.3% VS 26.2% (second place) error rates)
- ZFNet (in 2013)
  - Top 5 error rate: 14.8%
- GoogLeNet(19) - 2014
  - Top 5 error rate: 6.67%
  - Number of Parameters : 4 million
- VGG Net(16) - 2014
  - Top 5 error rate: 6.67%
  - Number of Parameters : 4 million
- ResNet(152) - 2015
  - Top 5 error rate: 3.6%

83

CNN Implementation

- One of the most popular DNN
- Go-to-model for all the image-related problems
- Image Classification
- X-ray diagnosis
- Face Recognition
- Face Verification
- Object detection
- Autonomous vehicles





84

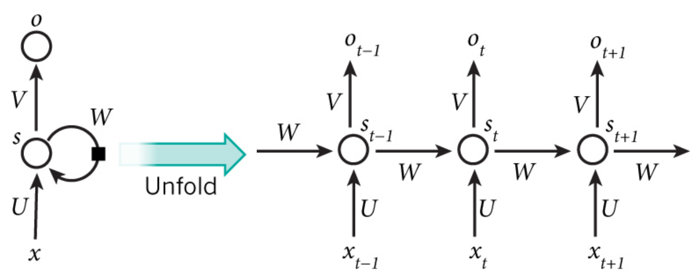
Python codes for CNN

```
From keras.models import Model
From keras.layers import Input, Conv2D, Activation, MaxPooling2D, Flatten, Dense

X = Input(64,64,3)

X1 = Conv2D(64, (7,7))(X) # (32,32,64)
X2 = Activation('relu')(X1)
X3 = MaxPooling2D((2,2))(X2) # (16,16,64)
X4 = Flatten()(X3) # 16*16*64

X5 = Dense(128)(X4) # 128
X6 = Activation('relu')(X5)
Y = Dense(1, activation='sigmoid')(X6) # 1

model = Model(inputs=X, outputs=Y)
model.compile(optimizer='adam', loss='binary_crossentropy')

model.fit(X_train, Y_train, epochs=50)
```

85

Recurrent Neural Network (RNN) Introduction

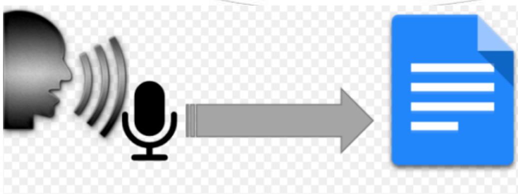
Introduction – recurrent neural network model to use sequential information.



Why RNN?
In traditional ANN, all inputs and outputs are independent of each other. But, in some case, they could be dependent.
Some problems such as text analysis and translation, we need to understand which words come before.
RNN has a memory which captures previous information about what has been calculated so far.

86

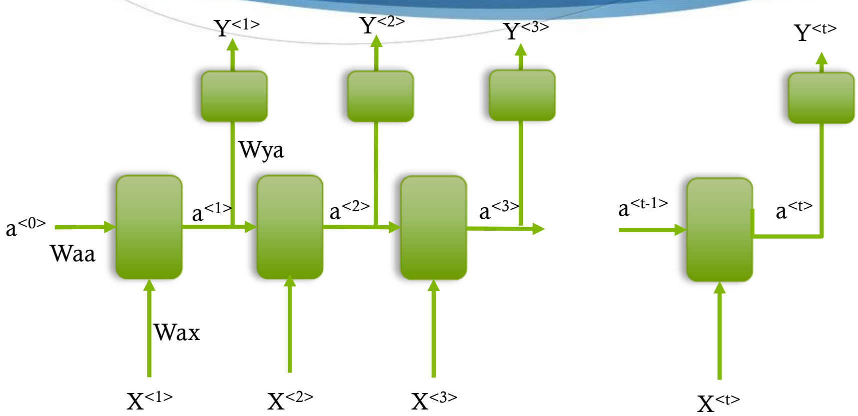
Recurrent Neural Network (RNN) Implementation



- Speech Recognition
- Music Generation
- Sentiment Classification
- Machine Translation
- Video Activity Recognition
- Name Entity Recognition
- DNA Sequence Analysis

87

Basic RNN Structure and algorithm



$a^{<t>} = g(Waa*a^{<t-1>} + Wax*X^{<t>} + ba)$ when g is an activation function (e.g., tahn, ReLU)

$Y^{<t>} = g(Wya*a^{<t>} + by)$ when g is an activation function (e.g., sigmoid, softmax)

Note that $a^{<t>}$ includes the information from previous X.

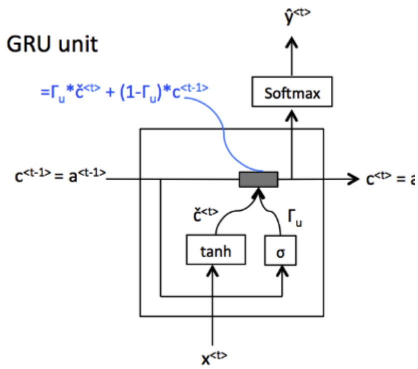
If t = 10, how many parameters?

88

More Complex RNN Structure and algorithm

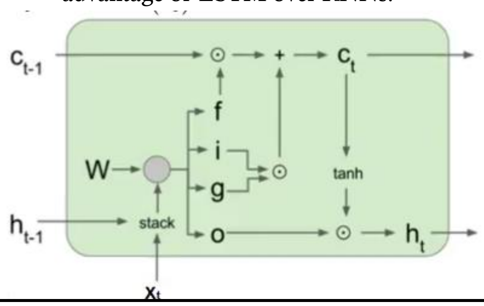
GRU (Gated Recurrent Unit) – A gating mechanism in RNN

- GRUs have been shown to exhibit better performance on smaller datasets.
- GRUs have fewer parameters than LSTM, as they lack an output gate.



LSTM (Long Short-Term Memory Unit)

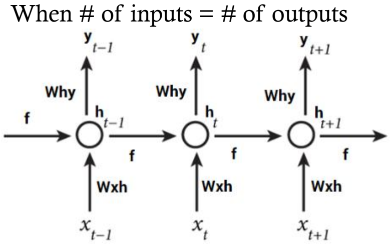
- It is composed of 3 gates – input, forget and output.
- LSTM remembers values over arbitrary time intervals and the 3 gates regulate the flow of information into and out of LSTM unit.
- LSTMs were developed to deal with the vanishing gradient problems.
- Relative insensitivity to gap length is an advantage of LSTM over RNNs.



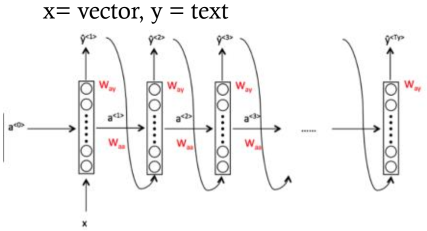
89

Popular RNN Architecture

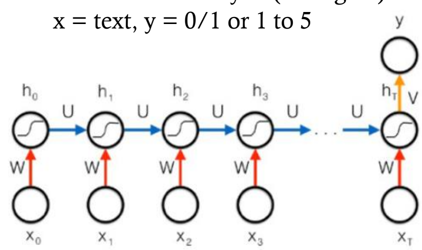
- Training on Embedding Matrix. Text notation. When # of inputs = # of outputs



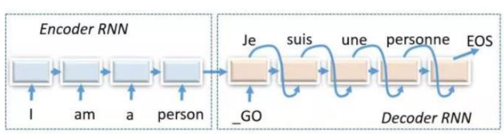
- Music generation. Picture Description. x= vector, y = text



- Sentimental Analysis (PV signal) x = text, y = 0/1 or 1 to 5

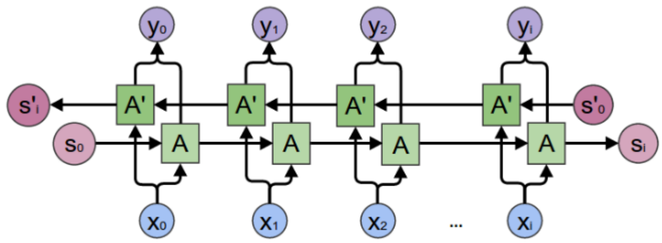


- Machine translation. x = text in English, y = text in French



90

Bidirectional Recurrent Neural Network (BRNN)



- Introduction of BRNN– RNN which learns both directions.
- Forward and backward learning of the same input.
- It reads the whole sentence. A great model of NLP.
- Examples
  - He said, "Teddy bears are on sale!"
  - He said, "Teddy Roosevelt was a great President!"
- Traditional RNN can't tell the difference between two sentences until Teddy.
- BRNN can tell the difference.

91

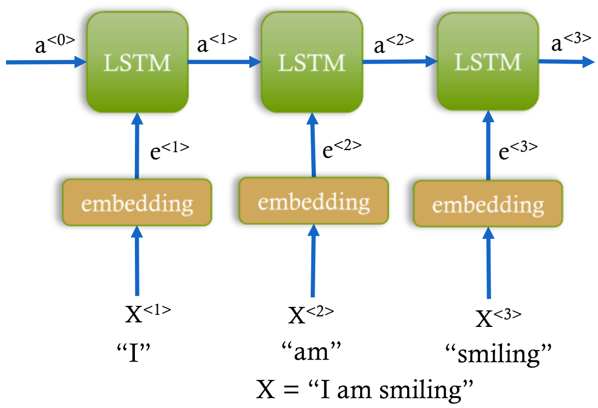
Natural Language Processing (NLP) using RNN

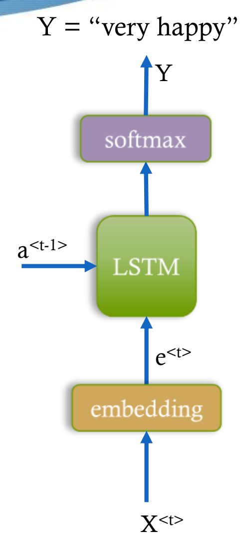
- Introduction of NLP – An area of artificial intelligence on an interaction between computer and human natural language. ML which programs computers to process and analyze natural language data.
- Input data
  - Embedding – representing each word to vectors of numbers
    - Glove (Global vectors for word representation)
      - 400K words
      - Each word represented by 50 dimensional vector
      - e_{the} - "the" [0.418, 0.24968, -0.41242, 0.1217, 0.34527, -0.044457, -0.49688, -0.17862, -0.00066023,,,,,]
    - Encoding – using word embedding (e.g., Glove), convert words to 50 dimensional vector.
      - "I am happy" to [e_I, e_{am}, e_{happy}]
- Output data
  - Softmax output [0.01, 0.33, 0.20, 0.73] to outputs ['very unhappy', 'unhappy', 'happy', 'very happy'] of "very happy"

92

Simple RNN architecture on NLP

- Input data – "I am smiling", "I laugh now", "I am crying", "I feel good", "I am not sure now"
- Embedding – to convert words to vector number
- LSTM – to learn language
- Softmax – to provide probability of output
- Output data - "sad", "fine", "happy", "very happy"





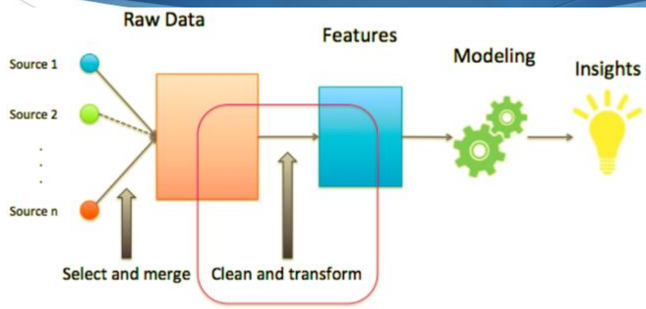
93

Python codes for RNN

```
From keras.models import Model
From keras.layers import Dense, Input, LSTM, Activation
From keras.layers.embeddings import Embedding

X = Input(10, dtype='int32)

embedding_layer = Embedding(400000, 50) # size of word-vector
embedding_layer.set_weights([emb_matrix]) # import from glove.400k.50d.txt

embedding = embedding_layer(X)
X1 = LSTM(10)(embedding)
X2 = Dense(4)(X1) # 4 possible outputs
Y = Activation('softmax')(X2)

model = Model(inputs=X, outputs=Y)
model.compile(loss='categorical_crossentropy', optimizer='adam')

model.fit(X_train, Y_train, epochs=50)
```

94

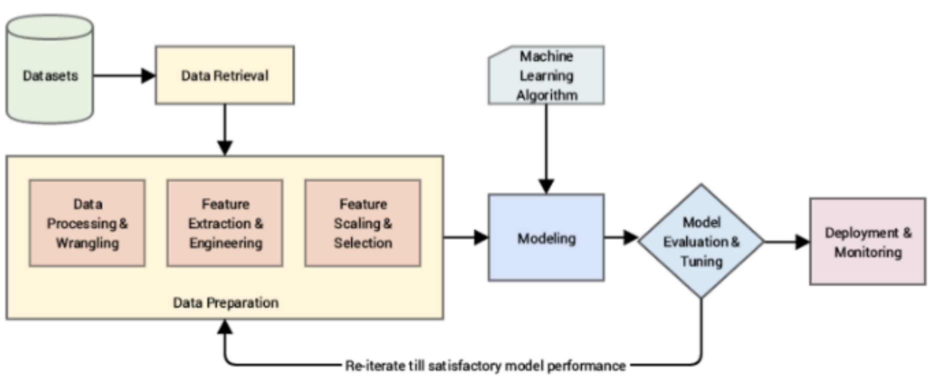
Feature Engineering



- Introduction of Feature Engineering
- Purpose of Feature Engineering
- Types of Feature Engineering

95

Feature Engineering in Standard Machine Learning Pipeline



A standard machine learning pipeline (source: Practical Machine Learning with Python, Apress/Springer)

96

Introduction of Feature Engineering

- Feature – numeric representation of raw data
- Feature Engineering
  - Not extracting process
  - The process of formulating the most appropriate features given the data, model, and task

Columns (Features)

Rows (Observations)

	HP	Attack	Defense
0	45	49	49
1	60	62	63
2	80	82	83
3	80	100	123
4	39	52	43

- Number of features
  - Too few – the model can not achieve the appropriate accuracy for the task.
  - Too many – expensive to train the model
- Examples
  - Normalization of number
  - 'Mild'/'Moderate'/'Severe' to 1/2/3 or [1,0,0]/ [0,1,0]/ [0,0,1]
  - Images to numeric vectors
  - Words to numeric vectors

97

Feature Engineering Types

- Numeric Featuring
  - Easy to impute
  - Types : floats, counts, integer
  - Examples : Age, Weight, House Price
- Categorical Featuring
  - Examples :
    - Race ['White', 'Black', 'Asian']
    - Subject ID : 01-0001, 01-0002, 02-0001, 02-0011
- Text Featuring
  - Examples: "The subject of 01-0001 experienced some issues after taking the study drug"
- Image Featuring
  - Examples : Cat image, X-ray
- Sound Featuring
  - Examples: Voice

98

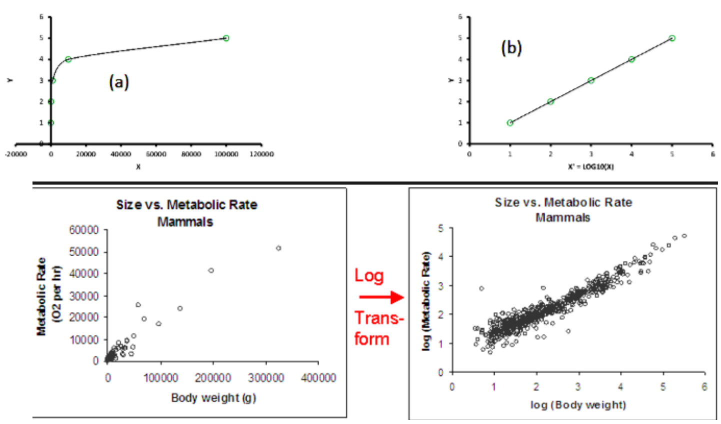
Numeric Featuring - Types

- Counting
  - the counting of certain values
  - Ex - how many times the subject has AE
- Binarization
  - Yes/No
  - Ex – Does a subject take study drug?
- Quantization - grouping the counts into bins
  - Fixed-width binning : 1900 to 2000 – (1900 – 1910], (1910 – 1920], (1920 – 1930],,, (1990 – 2000]
  - Quantile binning : 0 to 100 -> (0 – 25], (25 – 50], (50 – 75], (75-100]
- Interaction
  - To increase the complexity (e.g., non-linear feature) to the current linear model
  - Linear model : $y = w_1x_1 + w_2x_2$
  - Linear model with interaction feature: $y = w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2$

99

Numeric Featuring - Types

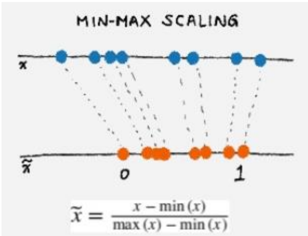
- Log Transformation
  - log of x
  - To transform Skewed data to linear data

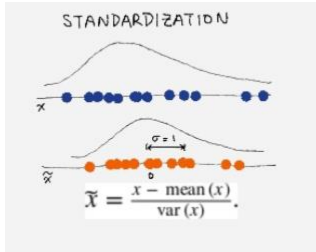


100

Numeric Featuring - Types

- Scaling - limiting the scale of input features since your model is sensitive to the scale.
  - Min-max scaling – squeezing all values to be within 0 and 1.
  - Variance scaling (standardization) – mean of 0 and a variance of 1





101

Categorical Featuring - Introduction

- Categorical data
  - To represent categories or labels.
  - The number of categorical data is always finite.
  - Types
    - Ordinal – have a meaning of ordering (AE severity ['Mild','Moderate','Sever'])
    - Nominal - no meaning of ordering (SEX ['Male', 'Female'])

102

Categorical Featuring - Types

- One-hot Encoding
  - To transform the m attributes to m binary features with 0 or 1
  - For nominal categorical data that should not be considered in a meaning of ordering.
  - Pros
    - Easy to implement
    - Potentially most accurate
  - Cons
    - Not feasible for anything other than linear models
    - A lot of features – computationally inefficient

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

103

Categorical Featuring - Types

- Dummy Coding
  - To transform the m attributes to (m -1) binary features with 0 or 1
  - Compared to one-hot coding, it is a unique and interpretable model.
  - Can't handle missing data very well

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow
1	0
1	0
0	1
0	0

- Effect Coding
  - To transform the m attributes to (m -1) binary features with 0 , 1 or -1
  - Better representation due to -1
  - -1 will require more storage and computation power

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow
1	0
1	0
0	1
~~0~~ -1	~~0~~ -1

104

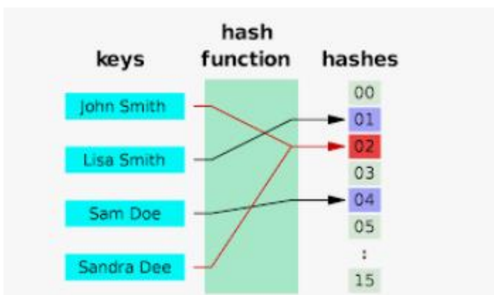
Categorical Featuring - Types

- Issues on Encoding Featuring
  - Huge number of categorical data
    - A lot of features
    - Expensive storage and computation
  - Examples –"user id" could yield huge number.
- How to solve issues on huge number of categorical data
  - Feature Hashing
    - Mapping unbounded integers to a finite range[1, m]
    - Feasible with linear model
  - Bin Counting
    - Using the probability based statistical information of the category.
    - Example: selecting # of AE rather than SUBJID
    - Feasible with linear model and trees
  - Use a simple model (linear model, logistic regression , SVM)

105

Categorical Featuring - Types

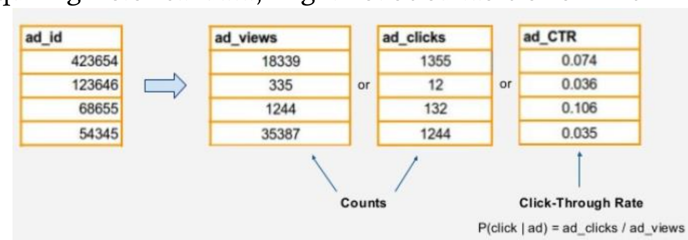
- Feature Hashing
  - To hash categorical values into vectors
  - Pros
    - Easy to implement
    - Fewer features – easier and cheaper to train the model
    - Easily handle new categories and rare categories
  - Cons
    - Only suitable for linear or kernelized models
    - Not accurate



106

Categorical Featuring - Types

- Bin Counting
  - Using the probability based statistical information of the category rather than actual categorical value
  - Example: selecting # of clicks rather than ad_id
  - Pros
    - Less expensive computation
    - Feasible with linear model and trees
    - Easily handle new categories and rare categories
  - Cons
    - Requires historical data
    - Since requiring historical data, might not be suitable of on-line learning



107

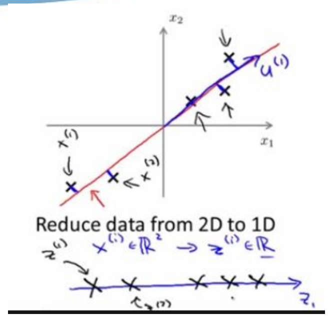
Feature Dimensionality Reduction

- Introduction – reducing random variables (input data) under consideration by obtaining a set of principle variables (new features)
- Definition – removing "uninformative information" while retaining the crucial information.
- Types
  - Principal Component Analysis (PCA)
    - Main linear techniques for dimensional reduction
    - A linear mapping of the data to a lower-dimensional space
  - K-Means Model Stacking
    - Main non-linear techniques for dimensional reduction
    - A data point is to be represented by its clusters.

108

PCA Introduction and Implementation

- Principal Component Analysis (PCA)
  - Use linear projection to transform data in the new feature space.
  - Examples : reducing data from 2D to 1D.



Implementation Steps

- Find the principal component : pca_comp = PCA(n_component=0.8) #80% variance
- Transform the data
  - pca_data = pca_comp.fit_transform(input_data)
  - Fit the model with input data and transform with 80% variance. This will provide reduced features (e.g., 64 to 13). These 13 feature's variance will sum up to 0.8.
  - One can choose top 3 or 5.

109

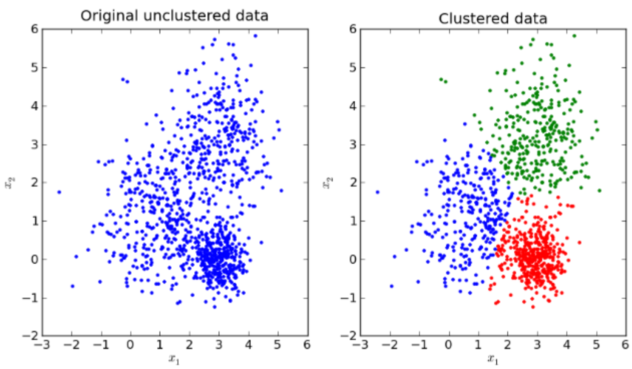
PCA Final Thought

- Linear Featurization / Linear Dimentionality Reduction
- Mechanism - Linear Projection
- Objective – To maximize the variance of projected data
- Model-driven feature engineering
  - Since features are linear-projected, they are good for linear Machine Learning model.
- Great dimensional reduction method
- Good use cases
  - Abnormality detection
  - Correlated features reduction
  - Feature Engineering for Deep Learning
- Limitation
  - High computation cost – expensive for more than a few thousand features
  - Since PCA transform removes information from data, downstream model is cheaper to train, could be less accurate
  - Uninterpretable outcome

110

K-Means Non-linear Featurization

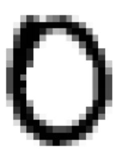
- Main non-linear techniques for dimensional reduction
- Data will be grouped into clusters and represent “closeness” in the same cluster.
- If the number of clusters < features, features reductions.
- If a label is present, one can include label in K-means featurization.

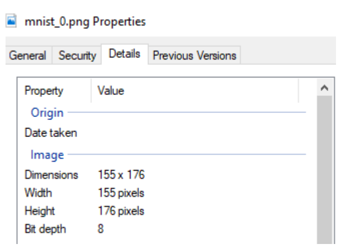


111

Image Featuring

- Converting image files (.png) to numeric representation (1300x1300)
- Color – 3D
- Blank/white – 2D






```
image_0 =imread(fname='./images/mnist_0.png')
print('Shape of Image 50: ', image_0.shape)
```

```
Shape of Image 0 : (155, 176)
```

```
[[ 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0   0  64 191 118  24 118 138 191 118  17   0   0   0   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0  17 193 253 252 252 252 252 253 252 227 120   5   0   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0  93 252 253 252 233 141  69  79 227 252 252 137   0   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0 187 252 253 252  89   0   0   0  29 154 252 252  95   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0 207 252 253 157   6   0   0   0   0   7 158 252 220   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0  70 253 253 231   0   0   0   0   0   0   0  43 241 255 154   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0  70 252 252 230   0   0   0   0   0   0   0   0 135 253 206   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0 101 252 252 167   0   0   0   0   0   0   0   0   0 222 240  50   0   0   0   0   0]
 [ 0   0   0   0   0   0   0 184 252 252   0   0   0   0   0   0   0   0   0   0 138 252 111   0   0   0   0   0]
 [ 0   0   0   0   0   0   0 184 252 147   0   0   0   0   0   0   0   0   0   0 138 252 183   0   0   0   0   0]
 [ 0   0   0   0   0   0   0 185 253 253   0   0   0   0   0   0   0   0   0   0 139 253 184   0   0   0   0   0]
 [ 0   0   0   0   0   0   0 184 252 200   0   0   0   0   0   0   0   0   0   0 138 252 183   0   0   0   0   0]
 [ 0   0   0   0   0   0   0 101 252 252   0   0   0   0   0   0   0   0   0   0 138 252 183   0   0   0   0   0]
 [ 0   0   0   0   0   0   0  44 236 252  95   0   0   0   0   0   0   0   0   0 233 252  89   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0 207 252 220   0   0   0   0   0   0   0   0 158 253 210   6   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0 145 253 244  65   7   0   0   0   0   0  62 243 252 135   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0  34 217 253 252 154  30   0   0   9  78 236 252 157   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0   0  84 222 252 252 227 184 185 197 252 252 221  32   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0   0   0  36 177 252 252 252 253 252 252 176  35   0   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0   0   0   0   5 107 179 252 190 137  54   4   0   0   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [ 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

112

Image Featuring

- Image Data Pre-Processing for pre-trained model
  - Same Size (1200, 1200, 3) for all the input image data
  - Normalizing
    - Number ranges from[0, 255]
    - Normalize data into [0, 1] by dividing by 255
- Featuring extraction from pre-trained model
  - Process image through pre-trained model
  - Get output and use them for another model like SVM, Logistic Regression

	Input	Output (Number of features)
ResNet-18	224 by 224	512
ResNet-50	224 by 224	2048
ResNet-101	224 by 224	2048
AlexNet	227 by 227	4096

113

Text Featuring

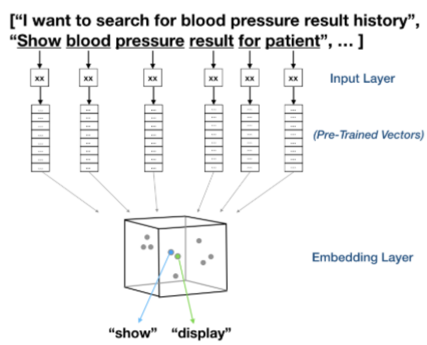
- Converting words to numeric representation
- Word embedding - representing each word to vectors of numbers
  - Use embedding matrix, Glove (400K of 50 dimensional vector)
  - Convert words to numeric numbers

Examples of converting word, “the” to numeric representation using Glove

```
glove['the']

array([ 4.1800e-01,  2.4968e-01, -4.1242e-01,  1.2170e-01,  3.4527e-01,
       -4.4457e-02, -4.9688e-01, -1.7862e-01, -6.6023e-04, -6.5660e-01,
        2.7843e-01, -1.4767e-01, -5.5677e-01,  1.4658e-01, -9.5095e-03,
        1.1658e-02,  1.0204e-01, -1.2792e-01, -8.4430e-01, -1.2181e-01,
       -1.6801e-02, -3.3279e-01, -1.5520e-01, -2.3131e-01, -1.9181e-01,
       -1.8823e+00, -7.6746e-01,  9.9051e-02, -4.2125e-01, -1.9526e-01,
        4.0071e+00, -1.8594e-01, -5.2287e-01, -3.1681e-01,  5.9213e-04,
        7.4449e-03,  1.7778e-01, -1.5897e-01,  1.2041e-02, -5.4223e-02,
       -2.9871e-01, -1.5749e-01, -3.4758e-01, -4.5637e-02, -4.4251e-01,
        1.8785e-01,  2.7849e-03, -1.8411e-01, -1.1514e-01, -7.8581e-01])
```

NLP process : train embedding and LSTM layers with trained data

converting process of text and putting embedding layer for the training



114

Machine Learning Tools

Program
- SAS
  - SAS tools: SAS Visual Data Mining and Machine Learning in https://www.sas.com/en_us/software/visual-data-mining-machine-learning.html - interactive, visualized, easy to use tools. Paid packages
  - Procedures in SAS Enterprise Miner : REG, HPSPLIT, NEURAL, HPSVM, HPCLUS,
- R
  - A great scripting language for data manipulation, data visualization and Machine Learning
  - Packages : caret, nnet, keras, mlr
- Python
  - The most popular Machine Learning scripting language
  - Packages : sklearn, tensorflow, Keras

Interactive easy to use Tool
- IBM Watson
- Microsoft Azure Machine Learning Studio
- Google Cloud

115

Difference between Statistics and Machine Learning

- Statistics
  - Sample Size
  - P-value
  - Specific models
  - Mathematically /statistically proven
- Machine Learning
  - Big Data
  - Accuracy
  - Any models
  - Black Box

116

How Machine Learning is being used in our daily lives

- Voice Recognition – Apple's Siri, Google's Home Assistant, Amazon's Alexa
- Recommendation – Amazon, Netflix, Spotify
- Chatbot – Online customer service
- Google Duplex – AI System for accomplishing Real-World Tasks over the phone
- Amazon Go – Cashless Grocery Store
- AlphaGO – beat "Go" world champion
- Terminator " I will be back"
- FDA first approval on ML/AI: Artery's medical imaging platform to diagnosis heart problem

117

Why is AI(ML) so popular now?

- Cost effective
  - Automate a lot of works
  - Can replace or enhance human labors
  - "Pretty much anything that a normal person can do in <1 sec, we can now automate with AI" Andrew Ng
- Accurate
  - Better than humans
- Can solve a lot of complex business problems

118

How Biometrics can utilize ML/AI in Pharma

Gold mines - Clinical trial data in Pharmaceutical industries

- Clean – Pharma companies spent a lot of hours to clean the data.
- Unbiased - Prospective study, randomized
- Blinded – double-blinded
- Standards – CDISC
- Structured
- Metadata

Pharmaceutical companies already owned the data.

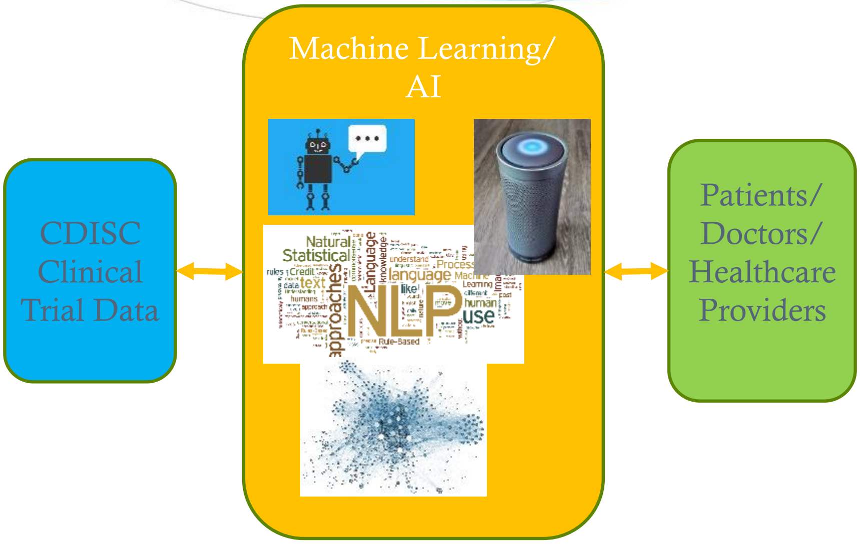
119

What is the reality of clinical trial data?

- Main purpose - for the submission
- No or limited analysis after submission
- Do not know exactly where clinical trial data is
- Limited/No access
- No CDR (Central Data Repository)

120

Clinical Trial Data with ML/AI implementation for Patients/Doctor



121

Machine Learning Implementation to solve problems

- Disease Identification/Diagnosis
- Personalized Treatment
- Drug Discovery
- Manufacturing Automation / Optimization
- Clinical Trial Research
- Radiology and Radiotherapy
- Site Selection
- Patient Recruitments
- Pharmacovigilance

122

Adoption of Machine Learning / AI in Pharma

- Slow
- Regulatory restriction
- Machine Learning Black Box challenge – need to build ML models, statistically or mathematically proven and validated, to explain final results.
- Status Quo / Change Management
- Big investment in Healthcare and a lot of AI Start up aiming Pharma

123

Now, how Pharma goes into AI/ML market

- GSK sign $43 million contract with Exscientia to speed drug discovery
- J&J (Surgical Robotics) – partners with Google. Leverage AI/ML to help surgeons by interpreting what they see or predict during surgery
- Roche - With GNS Healthcare, use ML to find novel targets for cancer therapy using cancer patient data
- Pfizer - With IBM, utilize Watson for drug discovery. Watson has accumulated data from 25 million articles compared to 200 articles a human researcher can read in a year.

124

Healthcare AI/ML market

- US - 320 million in 2016
- Europe – 270 million in 2016
- 40% annual rate
- 10 billion in 2024

- Short in talents
- Great opportunities

125

How can I start/learn "Machine Learning"?

- Embrace changes
- Starting reading about Machine Learning
- Take courses (online or school)
- Exercises
- Mimic others
- Compete with others (Kaggle)
- Start implementing in your organization even if it is a simple implementation

126

Conclusion

"Medicines and Data Science" company.
Novartis CEO Vas Narasimhan

"The hardest part is starting. Once you get that out of the way, you will find the rest of journey much easier."

"Secret of getting ahead is getting started"

127

Thanks!!!
Please contact
kevin.kyosun.lee@gmail.com
for any questions

128