Date
Page

* Machine Learning Crash course.

* Simple Linear Regression

$y = mx + b$

- inevitable margin of error called loss.
- Difference between predicted y & actual y is called residual.

- minimize residuals using a loss function

Best approach is to square & sum them

we can find m & b values that will find the sum of least squares.

- Solving for a fit

we randomly Increase / decrease m & b with random values from a standard normal distribution or even better a T-distribution which has fatter tails.

Note:- Diffrent ways to perform linear regressings

(5 ways are given in book)

* multiple linear regression

Solving for function $y = mx + b$ is elementary as its only has one independent variable x.

We can also solve for multiple independent variables like x_1, x_2, x_3, and so on....

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 .$$

Hill climbing - A simple optimization Algorithm.

1. Start with random or initial solution, even if its poor quality

2. Repeat the following steps for a number of iterations and/or until the solution cannot improve anymore

- Select a random part of the solution
- If that results in improvement, keep it.

* clustering

Let's start with 4 Centroids.

Distance between a point & centroid.

$$d=\sqrt{(\;)^2}$$

* K-means clustering

1) Take average x & y values for points nearest to each ~~ee~~ centroid. (hence k-means).

2) Set each centroid to that average x & y

3) Repeat until the centroids do no not move anymore & are at the average of their points.

- Overfitting

Overfitting means that our ML model works great with training data but fails to predict correctly with new data.

High variance & low bias

Fixes →

Splitting the data into two sets training & testing

we can also train with more data as well as utilize cross-validation, regularization, bagging, boosting & other techniques.

* Logistic Regression

Logistic regression is a classification tool that predicts true or false value for one or more variables.

- An S shaped curve (logistic curve) is fit to the points & then used to predict probability

- If predicted value is less than 0.5 it is typically categerized as false (0) & if the predicted value is greater than/equal to 0.5 it is typically categerized as true (1)

$$y = \frac{1}{1.0 + e^{-(B_0 + B_1 x)}}$$

```
def predict_probability(x):

    p= 1.0/(1.0+ math.exp(-(b0+b1 x))
    return p
```

you may also see this logistic function.

$$y = \frac{e^{B_0 + B_1 x}}{1.0 + e^{B_0 + B_1 x}}$$

Notice ~~how~~ the expression $B_0 + B_1 x$ is linear, & this is known as log odds function. which is translated logarithamically into a probability

- Maximum Likelihood.

- Get true (1) data, calculate the likelihood y for each x value, and multiply them toghether.

- Get the false (0) data, Calculate likelihood $(1.0 - y)$ for each x value & multiply them toghether.

- multiply the two products above toghether, & that is your total likelihood.

To avoid floating point underflow

we can use logarithmic addition instead of multiplication

So modified function is

- pass the value through log() & add.

- pass the value through log() & add.

- sum the two values above toghether pass it through exp() function to undo logarithm & that is total likelihood.

$$y = \log\left(\frac{1.0}{1.0 + e^{-(-3.17 + 0.69x)}}\right)$$

classmate
Date
Page

* Naive Bayes

Naive Bayes is a machine learning application of bayes theorem that merges probability of multiple features to predict a given category.

P_{fx} = Probability of feature x; with a fudged constant in numerator & denominator

$$\text{occur product} = \exp\left(\log(P_{f_1}) + \log(P_{f_2}) + \log(P_{f_n})\right)$$

$$\text{Not occur product} = \exp\left(\log(1-P_{f_1}) + \log(1-P_{f_2}) + \dots + \log(1-P_{f_n})\right)$$

$$\text{Combined probability} = \frac{(\text{occur product})}{(\text{occur product}) + (\text{Not occur product})}$$

classmate

Date

Page

* Decision Trees.

Decision trees are a powerful machine learning tool & work well for a lot of machine learning problems.

- Decision trees are notorious ~~of~~ for overfitting
- This can be remedid by Random Forest.
- Decision trees can also be improved with gradient boosting
- Other flavours of Decision trees exists like regression trees.

Note:- XGBoost is considered best in prediction type of problems.

- Gini Impurity

Gini Impurity is a common way to measure impurity using a function (for events A&B)

$$1 - (\text{Probability of } A)^2 - (\text{probablity of } B)^2$$

The Gini Impurity should never exceed 0.5

- weighted Average of Gini Impurity

At every step we chose the property that provides the least weighted average Gini Impurity.

classmate

Date
Page

- split continuous variables.

→ Calculate 2-value average of each candidate

- Then choose the 2-value average that produces the best Gini impurity that splits on that value.

- when do we stop?

when next property's weighted Gini is inferior to the previous Gini we stop the branch there.

* Random Forest

Random Forests are machine learning that generates hundred of decision trees, where each one build off partial random data & properties, rather than all data

- Typically each decision trees will learn with only 2/3 of the randomly sampled data, which is known as bootstrapping

- ~~It~~ can use the 1/3rd date. Known as out-of-bag data. as the test data.

- Each tree will vote on prediction, the prediction with highest vote wins.