

Data Science Concepts - Learning Notes for reminder and concept intuition

(includes Statistics, Linear Algebra, Practical Data Analysis, Machine Learning, Data Mining, Deep Learning)

purpose :

Some concepts learnt from University lecture notes, textbooks , blogs and papers.

Get myself easier to grasp the idea when going back to those materials

(todo : insert formula/proof after being good in Latex)

Index :

ML/DM difference

ML, Stat. Learning Difference

Different type of statistics

Different Errors

Random Variable

Independent, Identically distributed

Central Limit Theorem

Pearson Correlation's Coefficient

Principal Component Analysis

Shannon Entropy/Gini Coefficient

Hypothesis Testing (with t-test,z-test,sign-test,Chi-Square-test, ANOVA)

Association Rule Mining(Apriori)

Maximum Likelihood Estimation (MLE)

Generative vs Discriminative Learning Model for classification(Naive Bayes/Laplace Smoothing(in progress))

Bayesian Statistics

Gradient Descent Algorithm

Newton's method(todo)

Coordinate Ascent(ToDo)

Ensembling methods, Bagging, Boosting

Support Vector Machine(in progress)

Data leakage

Training set , Validation(DEV), Testing Set

Linear Regression

Logistic Regression

Softmax Regression

Bias Variance Analysis

Data Cleaning

Error Metric report

Neural Network(In progress)

ML project strategy

EM algorithm (ToDo)

Perceptron (ToDo)

Improving NN(todo)

K-Nearest Neighbour classifier(todo)

K-Means clustering(todo)

Reinforcement Learning(todo)

Time Series (ToDo)

Markov Model(todo)

NLP (todo)

RNN(Todo)
Tokenization(todo)
Recommender System(todo)
XGBoost(todo)
MapReduce(todo)
Common Statistical Distribution(Poisson/Gaussian/Multinomial/Binomial)(todo)
Convex optimization(Todo)
ICA(todo)
Generalized Linear Model
Data characteristics

Weighted least square.
Learning Theory.

ML/DM difference :

-- can share the same techniques (SVM, Lin R, Logistic R, neural network ..), Both can be used for prediction.

ML - we program something (predict a model for future prediction) using data

DM - we use data to build model to create knowledge (knowledge discovery), to discover pattern from data

ML, Stat. Learning Difference :

SL studies on stat. Inference of the parameters, ML focus on prediction.

Statistics -- how to collect data, interpret data, reach conclusion about data.

Descriptive Stat -- mean/var/skewness to describe data, visualization

Inferential Stat -- estimation , hypothesis testing

ML(Predictive modeling) vs Stat(Stat learning) -- use algo to model data vs studies of maths of model and goodness of fit.

Exploratory data analysis -- summarization and visualization of data.

Data Mining -- finding patterns/relationship in data.

Error :

Generalization error: errors rate that a model worked on unseen data. i.e. test error
in ML, Stat.L, GE = measure of how accurate an algorithm to predict previously unseen data.
sometimes = test error

It includes irreducible error (Noise), Bias : models' too simple(underfitting) , Variance : model too complicated(overfitting), captured too many noise in training data.

Standard error : Standard error of mean.

Training Error : error (rate) obtained during training. e.g. misclassification rate during training

DEV error : error rate obtained during parameter tuning of the trained model.

Test Error : for validation.

Random Variable :

A numerical variable, having it values comes in an associated probability.

e.g. Faired Dice rolling value $X \in \{1,2,3,4,5,6\}$, each having probability $1/6$

Independent, Identically distributed(iid):

Independent -- different samples have no correlations , formal $P(x \text{ and } y) = P(x) \cdot P(y)$

Identically distributed -- have the same distribution

Central Limit Theorem (proof omitted):

When sample size is large enough, the sample means will come in Normal distribution with (Population of any distribution)

sample mean = population mean

and sample SD = Population SD/sqrt(n)

Pearson coefficient :

- measure RV's correlation.

- range from -1 to 1, standardized value of correlation, 1 large, LINEAR relationship between 2 RVs, 0 independent.

- but easily affected by outliers.

Correlation vs Causation

Correlation does not mean causation, sales of ice-cream has high correlation with opening hour of aircon, but there is a third factor (weather, temperature)

PCA(Principal Component Analysis) :

- problem : too many features(too high dimension) , we want to combine some features into one by linear combination

- But we still want to maximize the variance of new feature, so that the one component still collects the most "uniqueness" from the data set.

- (to minimize the reconstruction error)

- in this case transformed vectors will lose the min. constituent elements characteristics (compressed lesser other-dimension)

- it is found by finding the project of "principle axis" of the data in features and features are projected into this axis to form the new feature.

- it can be proved that max var. is equivalent to finding max. eigenvalue and its associated eigenvector (principal axis).

Greedy Algorithm :

- We have no algo to find an optimal solution, and brute force is too costly.

- We define an objective(e.g. largest reduction in entropy in decision tree, which we conform to and make the (most/best) of it.)

- Greedy Algorithm generally not finding optimal solution, but based on an objective(e.g. personal interest/Information Gain)

Dynamic programming is used with Greedy Algorithm to break down bigger problem into smaller problem and fix it.

Shannon Entropy Concept (for decision tree ID3 · C4.5 · C5.0) :

(ref : <https://arxiv.org/abs/1405.2061>)

How Entropy formula relates to certainty.

- Surprise (events of low probability),

- (1) Measure the EXPECTED average min. amount of information (number of bits) used to store a distribution of event,

to save bits, we generally use smaller number of bits to indicate higher chance event (e.g. 01) and larger number

of bits to denote smaller chance event (e.g. 00100101). This can be calculated by Shannon Entropy formula.

- (2) evenly distributed events(10*10%) (no surprise/uncertain) generally has highest entropy

(we cannot do optimization to assign higher number of bits to lesser chance events), more certain cases(80%,10%,10%..) use lesser bytes.

=> Entropy formula measure the uncertainty of event distribution. => higher entropy, higher uncertainty

How certainty relates to Decision Tree.

IG (Information Gain) => amount reduction in entropy => amount of uncertainty removed.

With this splitting, we are in a more certain position make decision.

Some Decision tree are the greedy algorithm aiming to minimize the largest entropy (greatest information gain) first.

=> DT is to do splitting on the feature which provides highest promotion of certainty.

e.g. $P(C_1|A_1) = 0.4/P(C_2|A_1) = 0.6/P(C_1|A_2) = 0.4/P(C_2|A_2) = 0.6$

<-- less certain , even after answering this question, we do not gain much information.

$P(C_1|B_1) = 0.01, P(C_2|B_1) = 0.99, P(C_1|B_2) = 0.99, P(C_2|B_2) = 0.01$

=> We will choose Splitting on B instead of A first, and we probably will discard the splitting on A.

e.g. features, M/F , Tall/Short, muscular/thin => play basketball

M/F less IG , T/S highest IG, M/T middle IG

(same percentage of M/F player basketball, short people generally wont play)

Tree may be

Short -> not player , Tall -> next

Tall+Thin -> not player , Tall+muscular -> player

M/F too low IG , not considered

Gini impurity (CART) , usage : similar to Entropy

This is the probably of a class of object randomly put into the distribution, what is the probability of misclassification

higher => more uncertain

in this case : $1-p_1^2 - p_2^2$,

https://www.jmp.com/en_hk/statistics-knowledge-portal/t-test/two-sample-t-test.html

Hypothesis Testing (with t-test,z-test,sign-test) :

H0(Null Hypothesis) - Assumption failed to be rejected

H1(Alternative Hypothesis) - Assumption rejected at some level significance

one-tail-test vs 2-tail-test, 95% confidence interval as example,

one tail → whether the value has increased/decreased. (Question : data fall in last max/min 5% of distribution)

2-tail → whether the value has changed. (Question : data fall in 2 ending 2.5% section of distribution)

p-value :

The probability that the sample gives such a (or more extreme) result, given the condition that H0 is true.

It is tested against significant level (e.g. 95%/99%) for Hypothesis testing

p-value obtained by sample is used to compare against a critical value of a sig. level for H-Test

p-value < critical value => not sig. result , failed to reject H0

p-value > critical value => sig. result , reject H0

e.g. sample = 100 , mean = 25 min , p-value = $P(\text{mean} > 25 \text{ min} \mid H_0 \text{ true})$

Assume Sig. level = 95%

Type I Error : Incorrect rejection of a true H0, false positive

Type II Error : H1 is in fact true but we failed to reject H0, false negative

T-test : (sample size < 30 , or (unknown population var AND sample size large) or known normal distribution)

1-sample-test : test whether mean of single population = target value (is mean this sample of this population = 30)

2-sample-test : test whether means of 2 population are different (does mean of height of female diff from mean of height of male)

Paired T : Test whether difference between 2 dependent = target value (does mean of weight diff after taking weight loss pill)

python : `ttest_rel(data1, data2)` // from `scipy.stats` import `ttest_rel`

Z-test(assumed normal distribution , used when known population variance or sample size ≥ 30)

1-sample-test: test whether sample mean \neq population mean

2-sample-test: test whether means of 2 samples are different.

t-test use sample SD , z-test used population SD

ANOVA

one-way : H0: all sample independent distribution are equals , H1: one or more sample mean(s) is/are not equal

python : `stat, p = f_oneway(data1, data2, data3) // from scipy.stats import f_oneway`

Repeated Measures ANOVA Test : H0: aAll paired sample distributions are equal , H1: One or more paired sample distributions are not equal.

Chi square test (Goodness of fit)

full example : <https://stattrek.com/chi-square-test/goodness-of-fit.aspx>

Given categorical variable , it is used in Hypothesis test whether the sample data consistent(H0) with a specified distribution or not (Ha)

e.g. Merchant claimed that the draw have 10% gold, 20% silver, 70% bronze coin , we have 2% gold, 8 % silver, 92% bronze coin in sample

Chi square test (Independence) :

full example : <https://stattrek.com/chi-square-test/independence.aspx>

-- applicable to X2 test apply to categorical variable, each sample size happens at least $n \geq 5$, sample is simple random sampling

-- case similar to GoF test, just to check whether 2 distribution are dependent or not.

One-way Test : whether in a group of distribution , whether means are different by examining the variance.

ref : <https://online.stat.psu.edu/stat500/lesson/10/10.1>

Association Rule Mining , basket analysis (Apriori algo)

Original Paper : <https://www2.cs.duke.edu/courses/spring02/cps296.1/papers/AS-VLDB94.pdf>

Support : Itemset occurrence rate, $\text{count}(I)/\text{count}(\text{total transaction})$, $> \text{minsup}$ called freq. itemset

Confidence : $I1 \text{ and } I2 = \text{NULL}$, definition : $\text{support}(I1 \ \& \ I2)/\text{support}(I1) = P(I2|I1)$

Lift : $\text{support}(I1 \ \& \ I2) / \text{support}(I2) * \text{support}(I1)$

if $\text{Lift} > 1$, $I1$ and $I2$, $I1$ and $I2$ are likely to be bought together.

if $\text{lift} = 1$, independent

if $\text{Lift} < 1$, adverse relationship

Why lift? (why conf not enough?) conf may give illusive result when $P(I1)$ is small and $P(I2)$ is large.

<- This is equivalent to saying, If I buy $I1$ then I will also by $I2$ at a confident rate.

ARM is to find all rules : $I1 \rightarrow I2$, having support $> \text{minsup}$, confidence $> \text{minconf}$

NOTE:

(1) $I1$ and $I2$ are disjoint

(2) meaning: if transaction includes Itemset1 then it also has Itemset2

Important properties :

(1) Every subset of a frequent set is frequent set , more generally if A is superset of B : $A \text{ freq} \Rightarrow B \text{ freq}$, $B \text{ not freq}$, $A \text{ not freq}$

(2) $\text{support}(A \& B) < \text{support}(B)$ (anti-monotone property)

Step I : Freq. items computation(using property (1))

Step II : rule generation

Simple Demo :

Step I:

1. Assume : A,B,C freq, generate AB,AC,BC (by property 1) , ABC (only if AB,AC,BC are frequent)

STEP II:

2. if ABC freq, we have about 2^k possible rules. e.g. $A \rightarrow BC \dots BC \rightarrow A \dots$

by property of following : $\text{conf}(\{\text{beer}, \text{bread}\} \rightarrow \{\text{milk}\}) \geq \text{conf}(\{\text{beer}\} \rightarrow \{\text{milk}, \text{bread}\})$. (by conf definition)

Since $\text{support}(\text{beer}, \text{milk}, \text{bread}) / \text{support}(\text{beer}, \text{bread}) < \text{support}(\text{beer}, \text{milk}, \text{bread}) / \text{support}(\text{milk})$

if $abc \rightarrow d$ not a rule , then $ab \rightarrow cd, ac \rightarrow bd, a \rightarrow bcd$ also not a rule.

Maximum Likelihood Estimation (MLE):

When we have a set of sample data, we want to find the best parameter of a model (e.g. mean, standard deviation) which is best fit to this observation (sampled data). best chance (max. likelihood) that fit this model. $f(X|\theta)$

maximize the $L(\theta) = p(\text{Data}|\theta)$

Interpreting Likelihood function :

$L(\theta)$ {or $L(\theta|X)$ } is defined as $p(X|\theta)$, the distribution of x (which is observed sample, note in classification samples

notice in GLM MAP is used instead of MLE) given the variation of parameters

In interpreting L , we have to think in the context of parameter estimation, therefore it is the variation of θ

it is $L(\theta|x)$ because we are working based on the observed samples x .

x is multivariate , and x_1, \dots, x_n are independent, we have $L(\theta) = p(x_1|\theta) * p(x_2|\theta) \dots p(x_n|\theta)$, if every increasing, can apply log likelihood.

We target to find θ that make the observed sample most likely (this θ is most probable for the observed data), i.e. to maximize $p(\text{Data}|\theta)$

Unbias Estimator : sample parameters (e.g.mean,sd) that will = the population value when sampled infinitely.

Linear Discriminant Analysis :

Generative vs Discriminative Learning Model for classification
(ref : CS229 notes2 Generative)

D - example : Logistic Regression.

G - example : Naive Bayes Classifier , Gaussian Discriminant Analysis

suppose 2 classes y_1, y_2 cases, x be features

Discriminative : [1] construct model $p(y|x)$ using selected algorithm) [2] to classify y based on x
 \Leftrightarrow to see $p(y_1|x)$ or $p(y_2|x)$ is greater for hypothesis.

Generative : Approach to find $p(y|x)$, instead of finding $p(y|x)$ directly ,
model $p(x|y)$ and $p(y)$ (or $p(x,y)$) (given known classes, observe the distribution of its features) ,then
use $p(x|y)p(y)/p(x)$ to find $p(y|x)$

$p(x)$ can be omitted because $\text{argmax}_y p(y|x) = \text{argmax}_y p(x|y)p(y) = \text{argmax}_y (p(x \text{ AND } y))$

which means : In the modeling process , if want to find max. $p(y|x)$

(Note in here, the concept is very similar (in fact the same) to MLE for parameter estimation , but
in fact here, we are having trying $y=1/y=0$, varying the class instead of
parameter variation)

it is the same as finding max value of $(p(x \text{ AND } y))$ during our observation.

$p(y|x)$ means given features, the distribution of $p(y)$ which is $h(x)$

$p(x|y)$ found by we have samples which is classified , we observe their features.

$y = \text{elephant ,man}$, $x = \text{height}$

What does " $\text{argmax}_y p(y|x) = \text{argmax}_y p(x|y)p(y)$ " means ?

e.g.

for case where $y = \{0,1\}$

we find $p(x|y=1)p(y=1) > p(x|y=0)p(y=0)$, then $\text{argmax}_y \dots$ is 1 ,

same as when given x , we classify $y = 1$ because $p(y=1|x) > p(y=0|x)$ { finding $\text{argmax}_y p(y|x)$ }

Assume we have a new case to classify

we consider $p(x_1, \dots, x_{50000}|y)$ (some x 's one , some x 's zero)

$= p(x_1|y)p(x_2|y, x_1)p(x_3|y, x_1, x_2) \cdots p(x_{50000}|y, x_1, \dots, x_{49999})$ (by Bayes rule)

$= p(x_1|y)p(x_2|y)p(x_3|y) \cdots p(x_{50000}|y)$ (by Naive Bayes assumption)

\Rightarrow classified $y = \text{argmax}_y p(y) p(x_i|y) \dots p(x_n|y)$

Bayesian Statistics (Concept)

Bayesian Theorem : $p(x,y) = p(y|x)p(x) = p(x|y)p(y)$

Bayesian Learning :

$p(\text{Theta} | \text{Data})$ {Posterior}

$= p(\text{Data} | \text{Theta}) \{\text{likelihood}\} * p(\text{Theta}) \{\text{prior}\} / p(\text{Data}) \{\text{Normalization}\}$

A view : (<https://www.stat.auckland.ac.nz/~brewer/stats331.pdf>)

Prior : initial guess

Likelihood : given initial guess, we have this sample data

Then final estimation = Posterior = prior + likelihood update

Gradient Descent Algorithm(details : Coursera StandfordOnline - ML, CS229 notes, e.g. finding
min. of least square, though it already has simple, closed form solution in matrix representation)

-- purpose : numerical method to find **local max/min** of a multivariate function.

-- follow the gradient of a function , gradient is proved to the direction of steepest descent (negative
direct of steepest ascent).

- can apply to non-convex function, but may fall to local min instead of global min.

- Used in Linear/logistic regression(for optimizing least square error function), neural network

Batch GD -- use all samples for each step of GD (ref: definition least square derivative function in
GD)

mini-batch GD -- use shuffled subset of sample for each step , lesser computing cost

Stochastic GD -- special case of mini-batch GD , just use one sample for each step, claimed to be as good as Batch GD. (how?)

alpha -- step size, large -> faster , but leads to steps oscillation around the minimum, need some detection algo.

Exponentially weighted average : $v_t = \beta v_{t-1} + (1-\beta)(\text{current value})$, have smoothing effect, reduce fluctuation. (optionally there is bias correction, for correcting initial values.) , used in mini-batch/stochastic GD, due to randomness of

GD with momentum (Intuition) :

In NN case, $w(t+1) = w(t) - (\alpha)dL/dw$,

with momentum , in minibatch/SGD case.

$z(k+1) = \beta z(k) + dL/dw$

$w(t+1) = w(t) - \alpha * dL/dw$

smoothing effect made, weight given to the gradient in previous steps as smoother, we believe that the function wont change that much \Leftrightarrow weight given to history gradient.

RMSprop also for smoothing,

Adam = RMSprop + Momentum.

Learning Rate decay \rightarrow some formula applied to Learning rate alpha, make it larger steps initially, smaller step in later steps.

Ensembling methods : (ref : proof with a special case found in cs229 ensemble, no general case proof available)

good : decrease in var, better accuracy , free validation set (Out of Bag data), support missing value

bad : increase bias, hard to interpret , more expensive ,not additive.

-- It can be proved that variance can be reduced by "averaging error"

-- by increasing n (number of models, but increase cost), and decrease correlation of models), to de-correlate models :

use diff algorithm(SVM,logistic) , use diff training set, Bagging , Boosting

Bootstrap : to create multiple samples (size of full samples) through replacement

-- pro : got many different sample sets. Since we assume $S = P$, replacement can resemble duplication in P .

Bagging (not limited to DT)

-- By averaging (aggregating) the output of models, or voting for classification, we reduce variance with on p and M .

-- reduced correlations term w.r.t. single sample(Bootstrap), reduced VAR by increasing number of models

-- using bootstrap, on avg. only about 2/3 of data is used for training for each tree, remaining data (out-of-bag data) can be used for test set.

(<https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710>).

<https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710>

OOB error estimation can roughly give the generalization error, OOB error gives an equivalent result to LOOCV

-- If our test sample does not contain a feature, we can just exclude those model(tree) used that feature in split.

Boosting :

-- Choose a weak learner first(prediction rate just barely $> 50\%$),
create more stronger learners,
then predict results using the averaging of learners.

example : Adaboost <https://blog.paperspace.com/adaboost-optimizer/> note the alpha variation and how alpha is applied to data

-- create stronger learners by increasing the weight of mis-classified data , then
-- don't know why additive model \Rightarrow next model dependent on previous model (which will increase variance).

"Each new weak learner is no longer independent of the previous models in the sequence, meaning that increasing

M leads to an increase in the risk of overfitting." -- just because the data weight change ?

FASM (Forward Stagewise Additive Modeling)

-- use lost function L (e.g. residual function $y-h(x)$) to train the next weak learner
-- Adaboost is a special case of FASM
-- note it is an abstract model, not implementation like Adaboost.

Gradient Boosting :

Support Vector Machine : (SVM Succinctly,CS229 notes)

First assume linearly separable case , (for non-linear separable case, apply kernel for feature mapping)

Hyperplane : subspace with $n-1$ dimension, mathamatically : $w.x+b$, for simplicity , just let $y = \{-1,1\}$

General Idea : We find a hyperplane wich min. distance of data point to the plane (geometric margin) is greatest. (***) , for all samples

$f = y(w.x + b)$, f : functional margin, $w.x+b$ is the prediction , y is the actual sampled data,

if prediction is correct $\Rightarrow f +ve$, otherwise $f -ve$

w have to be normalized $\rightarrow w/|w|$ for fair comparison

$\Rightarrow f = y((w/|w|).x+(b/|w|))$

Function Margin vs Geometric Margin

GM is just scaled version of FM by $|w|$, $GM = FM/|w|$

FM only tell whether properly classified (properly classified $\Rightarrow +ve$ or missclassified $-ve$), GM can tell the the confidence.

FM is introduced to provide intuition to distance maximization (thinking this way, FM is more like geometric, but in actual computation ,

we need GM for correctness of confidence. It is because can always scale up FM to increase the distance of data point to hyperplane, which is

meaningless.) (ref : <https://stackoverflow.com/questions/20058036/svm-what-is-a-functional-margin>)

Simplified Derivation :

For a particular sample $i (x_i, y_i)$

Geometric Margin = $\min(r_i)$

where r_i = functional margin / $|w| = y_i((w/|w|) \cdot x_i + b/|w|)$

Optimal separating Hyperplane (w,b) for data is the plane where GM is largest.

ie

maximize M subject to $r_i \geq M$ for all i

(w,b)

It is finally a convex optimization, or a quadratic programming) problem :

minimize $\frac{1}{2} * |w|^2$

subject to $y_i(w \cdot x_i) + b \geq 1$ for all i

Main Steps : see SVM.doc in andersonchau's github

Data leakage :

target leakage , some features are in fact 'result' of target, it happens after the target=> do good in train and test but poor in actual

train-test data contamination : data cleaning on CV or test data (which is wrong), only do cleaning on train data

Training set , Validation, Testing Set : Why need validation set.

In general , Train Set => fit model param , Test Set -> Verification

But when we have models to choose e.g. $ax+b$, ax^2+bx+c , we use validation set to choose model, then use Test set for verification.

Why not just use test set to choose (or parameter tuning, parameter that are not fitted at training) model ?

-- Validation set is a part of training set, for finding optimal model, we still need test set for verification, we still need

test set as unseen data for verification.

K-fold Cross validation :

Training on $k-1$ folds and assessment on the remaining one, Generally $k=5$ or 10

e.g. 200 data, for 10 fold, each slice of 20 data is used to evaluate CV error(e_1, \dots, e_{10}) , final CV error is the avg(e_1, \dots, e_{10})

used when data is small

Leave-x-out(leave one out, LOOCV), training on $n-x$ data, x data used for calculate avg. cv error.

Linear Regression (ref: CS229 lecture notes)

- model for approximation.
- Hypothesis Function : $\Theta^T X$ (multivariate) , guess $\Theta_1, \Theta_2 \dots \Theta_n$ for N dimension data which will minimize least square error $J(\Theta)$, why not Absolute error ? AE function not differentiable, LSE is convex, guarantee to min. to global min.
- Minimization can be done by Gradient Descent Algorithm
- Linear Regression's optimal parameter can be written in vectorized closed form with

Logistic Regression (ref: CS229 lecture notes, coursera ML course)

- use $H(\theta) = \text{sigmoid function}$ for classification, $H \geq 0.5 \rightarrow 1$, $H < 0.5 \rightarrow 0$
- CS229 notes uses MLE proof, Coursera uses Penalty function to minimize the penalty, both gives the same result. Penalty function can also be minimized by GD (ref : Coursera NN course).
- Logistic regression can be extended to multi-classification using one-vs-all scheme. Assume A,B,C class, Train (A,B vs C),Train (A vs BC),Train (B vs AC) and compare which class has largest value, but inefficient, see softmax.
- sigmoid function much less susceptible to outlier (than linear function).

Softmax Regression

- SR is the multi-class classification version of Logistic Regression used in Deep Learning(neural network)
- ref : [Unsupervised Feature Learning and Deep Learning Tutorial \(stanford.edu\)](https://stanford.edu/~jlmiller/stanford-mlcourse2/unsupervised.html)

Bias Variance Analysis

- In general :
 - model too simple(underfitting) \rightarrow large training error, large bias
 - model too complicated (overfitting) \rightarrow small training error, but large test error, large variance
 - but decrease V \rightarrow increase B, or vice versa, (proof : Hastie ISLR), we generally find an optimal point to where B and V are in acceptable range.

Data Cleaning

Null value :

- If there is very small proportion of N/A data, have a look into why, or just drop those row.
(df.drop(<column name>,inplace=True)
- If larger proportion :
 - (1) impute mean value (or SimpleImputer)
 - (2) impute mean value according to class (which class? need domain knowledge)

e.g.

```
def impute_age(cols):
```

```
    Age = cols[0]
```

```
    Pclass = cols[1]
```

```
    if pd.isnull(Age):
```

```
        if Pclass == 1:
```

```
            return 37
```

```
        elif Pclass == 2:
```

```
            return 29
```

```
        else:
```

```
            return 24
```

```
    else:
```

```
        return Age
```

```
df['Age'] = df[['Age','Pclass']].apply(impute_age,axis=1)
```

-- If very large portion :

just drop the column : df.drop(<column name>,axis=1,inplace=True) // axis = 0 for row, 1 for whole column, default 0

```
my_imputer = SimpleImputer()
```

```
imputed_X_train = pd.DataFrame(my_imputer.fit_transform(X_train))
imputed_X_valid = pd.DataFrame(my_imputer.transform(X_valid))
```

Data Cleaning - Categorical Variable Handling

(1) One-Hot Encoding :

Sex : Male / Female

`sex = pd.get_dummies(df['Sex'],drop_first=True)` # makes it numerical , `drop_first =>` avoid duplicate

`df.drop(['Sex'],axis=1,inplace=True)` # drop those string based variable

`df = pd.concat([sex],axis=1)` # add those numerical column back.

(2) drop the columns with object type (non-num)

`drop_X_train = X_train.select_dtypes(exclude=['object'])`

`drop_X_valid = X_valid.select_dtypes(exclude=['object'])`

(3) LabelEncoding :

`label_encoder = LabelEncoder()`

for col in object_cols:

`label_X_train[col] = label_encoder.fit_transform(X_train[col])`

`label_X_valid[col] = label_encoder.transform(X_valid[col])`

Generally One-hot encoding perform best

Feature Scaling : Mean Normalization :

$x = (xi - \text{mean}) / (\text{max} - \text{min})$

or

$x = (xi - \text{mean}) / (\text{standard deviation})$ -- called standardization, lesser affected by outlier

or just normalize :

$x = (x - \text{min}) / (\text{max} - \text{min})$

When to use ?

K-Means, KNN -- distance based algo,

PCA -- need variance computation

NN -- use GD internally

GD -- converge faster, (Visualize circle(normalized direction) , elliptic contour curve Gradient direction, elliptic curve direction not pointpoing to the min. point's direction).

Error Metric report

R2_Score (for Linear Regression) :

1-RSS/TSS

ASE : avg standard err

MSE : mean standard error

RMSE : root mean standard error

For classification :

`classification_report(y_test,predictions)`

True Positive : Predicted yes, infact yes

True Negative : Predicted no, in fact no

False Positive : Predicted yes, in fact no (Type I error)

False Negative : Predicted no, in fact yes (Type II error)

Note in logistic regression , h's output is p , not 1/0, we just set the decision boundary as $p > 0.5$ by default.

Accuracy : $(TP+TN)/\text{Total Trials}$

Precision : $TP/(TP+FP)$ = true positive / number of predicted positive

recall : $TP/(TP+FN)$ = true positive / no. of actual positive

F1-score : $2/(1/P+1/R) = 2(PR)/(P+R)$

support : occurrence

In security recognition, we want to open door (predict 1) only when very confident.

We increase h boundary to predict 1 \Rightarrow higher precision , lower recall (P/R inverse relationship)

In cancer prediction case, we decrease h's boundary, because very bad when reported no cancer when in fact have cancer.

\Rightarrow lower precision, higher recall.

The Metrics to find the "best" P/R combination : find the algo (if we have > 1 algos) with highest F1 score (with $\beta = 1$, P/R same importance)

Can adjust β for F1 score to tune the importance of P or R.

$\beta = 0 \Rightarrow$ precision

$\beta = +\infty \Rightarrow$ recall

in python classification report

precision + 1 means when the model say 1, % the model is correct.

recall + 1 means of all 1s , the % which the model can predict 1

Neural Network (and concepts associated with NN)

Application : photo recognition(cat/not cat, voice recognition, NLP, price prediction,

Advertisement..), NN generally has higher performance

N layer = #hidden layers + output layer (input layer not count)

Each output of a neuron represent a new feature , but not all output feature interpretable

e.g. in housing price prediction, in each neuron1

input feature (x1 size, x2 number of bedroom, x3 zip code, x4 wealth)

input (size/number of bedroom) \rightarrow output (family size)

input(zip code/wealth) \rightarrow output(school quality)

input(family size/school quality) \rightarrow output(housing price)

Standard NN, CNN(data with very high dimension e.g. RGB image), RNN(sequence)

Improvement to algo, computation power , availability of larger amount of data \rightarrow drive NN.

Idea \rightarrow experiment \rightarrow code \rightarrow idea \rightarrow exp \rightarrow code

common activation function (ReLU , Sigmoid, tanh)

Forward Propagation step (in each neuron)

$z = WX + b$ (X is input, may be vectorized) , then apply activation $y = a(z)$

$a = \text{sigmoid/tanh/ReLU} \dots$, $y' = 0$ or 1 depend on y

in first layer : $z[1] = W[1]x + b[1]$, $a[1]$

in next layer : $z[2] = W[2]a[1] + b[2] \dots$

non-zero random initialization for $w[L]$ needed , otherwise all layer zero, suggested technique :
xavier/he initialization.

Back propagation : (to optimize L (loss function) wrt to W) , using GD algorithm by computing $dL/dW[l]$, $dL/db[l]$

Full training :

random initialization

for 1 to n_iterations :

 forward_propagation // compute $a[L]$

 compute_cost // compute $L[y, a[L]]$

 back_propagation // use GD to optimize w 's, b 's

 update_param // update w 's/ b 's for next forward propagation

Hyperparameters : #layers, #hidden units, learning rate, activation function.

Train/dev/test set : small data 60/20/20, large data 9x/1/1 , better from Same distribution

not having test set may be OK

comparing Train and dev set, it can be any combination of bias and variance.

Strategy

High bias \rightarrow bigger network , more advanced opt. Algo.

High Variance \rightarrow More data, regularization (L2/Dropout)

Intuition of regularization , Lambda large, w small, z lies in linear region, linear fcn.

Dropout regularization \Rightarrow shutting down/cancel certain portion of neurons, simpler network \rightarrow increase bias.

Other regularization : Data Augmentation, Early stopping

Normalize input \rightarrow faster GD convergence.

Vanishing/Exploding gradients, e.g. gaussian, Xavier initialization

Numerical approximation of gradient, Gradient checking , within 10^{-5} is okay

Please check improvement scheme on GD.

Hyperparameter tuning importance

(1st) learning rate

2nd momentum term beta ($0.9 \rightarrow 0.999$, decay way) , epsilon (10^{-8}), #number hidden units, mini batch size

3rd # layers, learning rate decay.

Do not use grid search for hyperparameter , use random , then coarse to fine.

Pandas model (train one model at a time , limited computing power, tune this specifically)

Train different models with different hyperparam at same time.

Use softmax regression instead of logistic regression for multi-class classification in NN.

Use batch norm to normalize z , then scale it to avoid zero mean, use mean and SD obtained in train time for test time \Rightarrow faster convergence, applied with mini batch. 2nd reason can avoid covariate shift problem : ref : <https://zhuanlan.zhihu.com/p/39918971>

Objective for tuning NN: do well on training set, performance may be comparable to human, then to dev then test.

Must have single evaluation metric(e.g. F1 score, or complicated : precision * speed/(k*memory requirement) or weighted performance on different samples of different counties)

Satisficing and Optimizing metric:

e.g.

accuracy : Optimizing , as much as possible

Satisficing : just OK if met with certain min. requirement

dev set and test set should come from same distribution.

Size of test size : objective to give confidence to the system.

Performance should take all elements of confusion matrix into consideration.

Bayes error is the best error that can be achieved by any algo, some algo work better than human error.

If ML error > human error, get labeled data from human and check why human OK but ML algo not working on this data, better analysis of Bias, Variance.

Human Error(proxy for Bayes error) \leftarrow avoidable bias err \rightarrow train error \leftarrow variance \rightarrow Dev error.

Focus on which gap is larger, Human error should be \approx expert's error.

ML performace may be > human performance , probably in case involving non-interpretable/many data(online advertisement, loan approval)

Avoidable bias :

bigger model, better algo(Momentum,RMSProp,Adam), Hyperparam search

Variance :

more data, regularization, Hyperparam search.

ML project strategy

Human Error Analysis (When Human analysis on ML error) :

worth it ? (depends on the occurrence rate of this "type" of ML error, and after improved, how much total system improvement vs Time).

not necessarily to correct mis-labeled data if number is small, NN is robust enough, but note mis-labeled data implied that data might come from different distribution,make sure dev/test data come from SAME distribution, split data for train/dev set of different distribution.

e.g. orig 500K data, new 20K data from new dist. , 10K train, 5K dev, 5K test.

1. setup metric

2. build system quickly

3. use B/V analysis to prioritize next step.

Data mismatch problem checking :

Train data

train-dev data (same dist with train data, but not used for training)

dev data

test data

Gap between train , train-dev data \rightarrow variance

Gap between train-dev , dev data \rightarrow dis misatch

Gap between dev, test data \rightarrow degree of variance btw dev/test set.

Address dist mismatch problem : manual analysis, create more data similar to dev/test set, e.g. voice vs voice with background noise, then synthesize voice with noise with audio software.

Transfer learning : reuse previous training for similar projects as previous layers as input layers.

Multi-task learning : do learning at a time for different tasks, different similar object.

End-to-End deep learning : no need engineered feature creation , just let Deep NN do it, by tremendous amount of data(which provides enough data for complexity).

When without tremendous amount of data for end-to-end, combine 2 learning algo.

e.g. face recognition => face portion as input for person identification.

Books :

Introduction to Statistical Learning

SVM Succinctly (step-by-step intro to SVM)

Elements of Statistical Learning (ISL is much easier, haven't started reading this yet)

Machine Learning, Tom Mitchell

Bookmarked University UG/PG DS courses :

<http://cs229.stanford.edu/> -- (the best , not pure ML : Sometimes mix with Stat. Learning / Bayesian Stat.)

<http://cs109.github.io/2015/pages/videos.html>

<http://web.stanford.edu/class/cs109/> -- first half parts are too easy, later part is good for understanding some concepts like Likelihood, Boosting.

<https://ocw.mit.edu/courses/sloan-school-of-management/15-097-prediction-machine-learning-and-statistics-spring-2012/index.htm>

-- the lecture notes are very good for stating basic concepts

CUHK Data Mining :

www1.se.cuhk.edu.hk/~hcheng/seg4630/index.html -- CUHK data mining courses,

others (ECLT 5810 E-Commerce Data Mining Technique/

CMSC5724 Data Mining and Knowledge Discovery)

Data Mining seems to ousted by ML courses these years, but I think those materials are still very worth reading.

General Statistics (t-test,z-test,chi-square test,sign test,ANOVA)

<https://stattrek.com/>

<https://online.stat.psu.edu/stat500/>

Khan Academy

Linear Algebra / Matrix computation :

<http://www.ee.cuhk.edu.hk/~wkma/engg5781/> -- ENGG 5781 Matrix Analysis and Computations (PSM, Least square , Matrix Diagonalization parts are very useful for understanding CS229 notes)

Online Bootcamp-type course :

<https://www.coursera.org/learn/machine-learning> -- (Best for beginner , not maths intensive)

<https://www.udemy.com/course/python-for-data-science-and-machine-learning-bootcamp/> -- best how-to for python beginners

<https://www.kaggle.com/learn/overview> -- free courses, step-by-step practical tutorial.

Python for DS :

<https://jakevdp.github.io/PythonDataScienceHandbook/>

<https://github.com/ageron/handson-ml>

<https://github.com/fengdu78/>

Books : Python for Data Analysis

Dashboard

Qlik Sense :

I used QS in working, My comment :

-- the GUI is beautiful/professional and responsive, boss/businessmen would like it

But

- association model is quite redundant in my opinion
- Weak ETL / data cleaning tools , Load script has to be written very complicated to support
- It is just a BI tools for data display , very weak integration with analytic tools of (ML,python)

Learning Resources :

- "Mastering Qlik Sense"
- Udemy QS certificate course
- QS official tutorial and newsgroup.

R

Other ML courses :

<http://www.cs.cornell.edu/courses/cs4780/2018fa/page18/>