**Theoretical Data Science Concepts**

**- Learning Notes for reminder and concept intuition**

**Author : Anderson Chau**

**Email : ansonchau2007@hotmail.com**

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

**Deep Learning, Statistical Learning, Optimization, Learning Theory, Information, Time Series, Reinforcement 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**

**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/Cross Entropy/KL divergence

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

Association Rule Mining(Apriori)

Maximum Likelihood Estimation (MLE)

Maximum a Posteriori(MAP)

Generative vs Discrminative Learning Model for classification(Naive Bayes/Lapace Smoothing(in

progress))

Another comparison of G vs D Learning model, MLE, MAP, Bayesian way(todo)

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

Bayes Error

No Free Lunch Theorem

K-Nearest Neighbour classifier(KNN)

Perceptron

Kernel Tricks

Neural Network(In progress)

ML project strategy

EM algorithm (Todo)

Improving NN(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/Multinominal/Binominal)(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 preduction) 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 discribe 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 {1,2,3,4,5,6} , each having probability 1/6

**Independent, Identically distributed(iid):**

Independent -- different samples have no correlations , formal P(x and y) = P(x)\*P(y)

Identically distributed -- have the same distribution

**Central Limit Theorem (proof omitted):**

When sample size is large enough, the sample means will comes 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 means 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 his case transformed vectors will lose the min. consistuent 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 it 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/Gini Coefficient/Cross Entropy/KL divergence ( 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 change 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-p1^2 - p2^2 ...... ,

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

**Cross Entropy :**

Measure the difference between 2 distributions ( e.g. sample distribution and model distribution )

machine learning - What is cross-entropy? - Stack Overflow

MLE Interpretation :

machine learning - The cross-entropy error function in neural networks - Data Science Stack

Exchange

Being used as a common loss function used in DL,for classification error.

Sparsecategoricalcrossentropy() in tensorflow

KL divergence : measure difference between 2 distributions for difference of bits.

CE as loss function → minimize the loss function → tune the model to be same as sampled data.

**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(mean > 25 min | H0 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 =/= population mean

2-sample-test: test wether 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>=5,

sample is simple random sampleing

-- case similar to GoF test, just to check whether 2 distribution are depenent 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 occurence rate, count(I)/count(total transaction), > minsup called freq. itemset

**Confidence** : I1 and I2 = NULL, definition : support(I1 & I2)/support(I1) = P(I2|I1)

**Lift** : support(I1 & I2) / support(I2)\*support(I1)

if Lift > 1 , I1 and I2 , I1 and I2 are likely to be bought together.

if lift = 1, independent

if 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->I2, having support > minsup , confidence > 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 freq => B

freq , B not freq, A not freq

(2) support(A&B) < 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->BC ... BC->A....

by property of following : conf ({beer, bread} → {milk}) ≥ conf ({beer} → {milk, bread}). (by

conf definition)

Since support(beer,milk,bread)/support(beer, bread) < support(beer,milk,bread)/support(milk)

if abc → d not a rule , then ab → cd ,ac → bd , a->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(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 theta

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

x is multivariate , and x1,... xn are independent, we have L(Theta) = p(x1|theta)\*p(x2|theta).... p(xn|

theata) , if every increasing, can apply

log likelihood.

We target to find theta that make the observed sample most likely (this theta is most probable for

the observed data), i.e. to maximize p(Data|Theta)

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

sampled infinitely.

**Linear Discriminant Analysis :**

**Generative vs Discrminative Learning Model for classification**

**(ref : CS229 notes2 Generative )**

D - example : Logistic Regression.

G - example : Naive Bayes Classifier , Gaussian Discriminant Analysis

suppose 2 classes y1, y2 cases, x be features

Discriminative : [1] construct model (p(y|x) using selected algorithm ) [2] to classify y based on x

<=> to see p(y1|x) or p(y2|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 argmax y p(y|x) = argmax y p(x|y)p(y) = argmax y (p(x 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 , variating the class instead of

parameter variation )

it is the same as finding max value of (p(x 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 = elephant ,man, x = height

What does "argmax y p(y|x) = 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 argmax y.... is 1 ,

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

Assume we have a new case to classify

we consider p(x1, . . . , x50000|y) ( some x's one , some x's zero )

= p(x1|y)p(x2|y, x1)p(x3|y, x1, x2)・ ・ ・ p(x50000|y, x1, . . . , x49999) (by Bayes rule)

= p(x1|y)p(x2|y)p(x3|y)・ ・ ・ p(x50000|y) (by Naive Bayes assumption)

=> classified y = argmax y p(y) p(xi|y)..... p(xn|y)

**Bayesian Statistics (Concept)**

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

Bayesian Learning :

p(Theta | Data ) {Posterier}

= p(Data | Theta ){likelihood} \* p(Theta) {prior} /p(Data) {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 = Posterier = 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 optimizating 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 : vt = beta\*vt-1+(1-beta)(current value), have smoothing effect,

reduce fluctuation. (optionally there is bias correction, for correcting initial values.) , used in

minibatch/

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 <=> weight given to history gradient.

RMSprop also for smoothing,

Adam = RMSprop + Momentum.

Learning Rate decay→ 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

decorrelate

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.

– why resampling with replacement ?

There are 2 ways to find the estimators(mean/SD) :

(A) make assumption on population distribution (e.g. Normal) and just use samples to guess them.

(B) Assume sample best represent population, replacement make sure that resampled data are

independent. Another view : when sample size is not large, without replacement => reduced

denominator. The repeated case in BS is just simulating the population’s characteristics closed to

that of the repeated case.

**Bagging ( not limited to Decision Tree )**

–**Boostrap + Aggregration.**

-- By averaging (aggregrating ) the output of models, or voting for classification, we reduce

variance with on p and M(CS229 notes the Var(Xs/m) computation. Different models are trained by

different samples sampled from same population for N times.

( the Var formula in Aman's AI Journal • CS229 • Ensemble Methods )

– Ensemble Methods: Bagging and Boosting (iitk.ac.in)

-- 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 rougly give the generalization error, OOB erorr 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.

**Random Forest**

– Different samples are created by randomly choosing samples with D^1/2 features for each DT.

- Average/vote output of each trained DT.

**Boosting :**

-- Choose a weak learner first(prediction rate just barely > 50% ),

create more stronger learners by adding more weights to misclassified data points, 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

best reference : ISL – Hastie chapter 10

-- create stronger learners by increasing the weight of mis-classified data , then

-- don't know why additive model => next model dependent on preivous 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 ?

**FSAM (Forward Stagewise Additive Modeling)**

-- Adaboost is a special case of FASM

-- note it is an abstract model, not implementation like Adaboost.

**Gradient Boosting :**

-- use residue (e.g. residual function y-h(x)) to train the next weak learner, which we believe it to be

a stronger learner.

– imagine that this is similar to applying GD to learners.

**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 => f +ve , otherwise f -ve

w have to be normalized -> w/|w| for fair comparison

=> 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 => +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-afunctionalmargin)

Simplified Derivation :

For a particular sample i (xi,yi)

Geometric Margin = min (ri)

where ri = functional margin / |w| = yi( (w/|w|).xi + b/|w| )

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

ie maximize M subject to ri >= M for all i

(w,b)

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

minimize 1/2 \* |w|^2

subject to yi(w.xi)+b>=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

cleanining 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 different models to choose, e.g. value of k in KNN , or another example : ax+b

vs ax^2+bx+c , we use validation set to choose model(parameters),

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.

( yes, we can use test set for choosing param, but here we will have no “unseen” data for

checking” )

**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, eash slice of 20 data is used to evaluate CV error(e1,.... e10) , final CV

error is the avg(e1....e10)

used when when data is small

Pick model with best average performance across K trials

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

where K = number of data. ( good when very small data, but generally higher variance )

Hold out/Bootstrap/K Cross-validation/Leave one out

Stratified Cross Validation

/Balanced Stratified Cross Validation

/Stratified Hold out

/Stratified Bootstrap

**Linear Regression (ref: CS229 lecture notes)**

– model for approximation.

– Hypothesis Function : Theta\*X( multivariate) , guess Theta t1,t2… tn 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) = sigmoid function for classification, H>=0.5 → 1 , H<0.5 → 0

- Notes : CS229 notes uses MLE proof approach.

Coursera uses Penalty function and minimize it using GDA, 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)

**Bias Variance Analysis**

– In general :

model too simple(underfitting) → large training error, large bias

model too complicated (overfitting) → small training error, but large test error, large variance

but decrease V → 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-mean)/(max-min)

or

x = (xi-mean)/(standard deviation) -- called standardization, lesser affected by outlier

or just normalize :

x = (x-min)/(max-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 , Total Sum of Square Error

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)/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 : occurrance

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

We increase h boundary to predict 1 => higher precision , lower recall (P/Rinverse relationship)

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

when in fact have cancer.

=> 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 beta for F1 score to tune the importance of P or R.

beta = 0 => precision

beta = +inf => 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

**No Free Lunch Theorem**

each ML algo must make its own assumption (e.g. KNN assumption) => no single ML algo works

for every settings.

**Bayes Error**

– **Assume we KNOW the true distribution of samples( note : not population )P(y|x) (which**

**we don’t practically, we can only find h(x) which approximate P(y|x) )**

**=>** the best we can do is to classify the sample using P(y|x) , but still we cannot achive zero error.

There is still Bayes Error, which is irreducible error, from noise in sampling.

**K-Nearest Neighbour classifier(KNN), with demonstration of curse of dimensionality**

ref : https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02\_kNN.html

KNN assumption : data similar to each other has smaller distance.

It is simply for a test point choosing K nearest ( e.g. min. minkowski distance ) neighbor points and

vote the largest occurrence of the classes.

Other facts :

When number of samples is very large, 1-NN algorithm approximately equals 2\*bayes error

i.e. When number of data very large, KNN is accurate, but algo is very slow.

When dimension of data is very large >> number of samples, the smallest space to find k nearest

point is roughly equals to whole space(or say data not similar to each other anymore, all data has

very large distance from others ) <=> which means KNN assumption breakdown

**Perceptron**

ref : https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote03.html

Note : Perceptron only applies to linear separable data.

Adjust the boundary according when found misclassified data point with a shift of distance

applies to linear separable data for classification, it can be proved that the algo always converge for

a limited number of trials wrt to margin.

**Kernel Tricks**

ref : CS229 notes kernel methods

Some algo originally just work for linearly / quadratic separable data,

By using feature mapping K we can make complex classifier for non-linearly separable data.

In LMS(Least Mean squares) gradient descent update steps(or may be others algos), update can be

rewritten in terms of φ(x (j) ) T \* φ(x (i) ) , where φ is the feature mapping.

Kernel some properties : <φ(x (j) ), φ(x (i) )> is much more efficient than computing the values

element by elements.

From the above , we can precompute all the <φ(x (j) ), φ(x (i) )>’s and do update based on precomputed

result.

K is a valid kernel iff K is PSD. By Mercer Theorem.

Kernels are measures of similarity, i.e. s(a, b) > s(a, c) if objects a and b are considered “more

similar” than objects a and c

Example: Gaussian Kernel.

**DEEP LEARNING**

Neural Network ( and concepts associated with NN)

**Introduction :**

**Application** : photo recognition(cat/not cat, voice recognition, CNN image recognition NLP (Using RNN) , price prediction,

Advertisement..),

* **Performance :** NN generally has higher performance compared to other ML algorithms like SVM

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

\* **Neuron :** to introduce non-linearity to NN.

Each output of a neuron represent a new feature , but not all output features are interpret-able, feature in deeper layer generally means more complicated condition.

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)→ output (family size)

input(zip code/wealth) → output(school quality)

input(family size/school quality) → 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 → drive NN.

Idea → experiment → code → idea → experiment → code

common activation function ( ReLU , Sigmoid,tanh )

Why **ReLU** better than **Sigmoid** ?

Ref :

<https://stats.stackexchange.com/questions/126238/what-are-the-advantages-of-relu-over-sigmoid-function-in-deep-neural-networks>

<http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf>

With reference to their shape : ReLU : constant gradient, Sigmoid : diminishing gradient at flattened area => more difficult to converge in GDA. ReLU easier to compute gradient in GDA because Sigmoid is a function of exponents.

**General Steps :**

**Forward Propagation step ( in each neuron, ANN )**

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

a = sigmoid/tanh/ReLU .. , 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] …

non-zero random initialization for w[L] needed , otherwise all layer zero, suggested technique :

**xaiver/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 :**

An Example (must-read, notice the vectorization of data in the NN)

coursera-deep-learning-specialization/Planar\_data\_classification\_with\_onehidden\_layer\_v6c.ipynb

at master · amanchadha/coursera-deep-learning-specialization (github.com)

random initialization(**xaiver/he initialization.)**

for 1 to n\_iterations(epoch value) :

forward\_propagation // computer 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

**Intuition of a neuron and activation function.**

– each output of neuron simulate a step function, which represent a particular characteristics of a

class, the step function is triggered once the input contains that characteristic. This also explains

why we don’t use 4 neurons to represent 16 classes, but # of neurons in output layer = #classes.

– The more deeper the layer → more complicated chacteristics

**The activation Function :**

**Why not just step function ?**

- Step function not differentiable, cannot be incrementally tuned by variation **w** and **b,** we want to a

smooth(differentiable) function which simulate step function (e.g. sigmoid), for GD in optimization.

(Δ output of activation function) roughly = summation of ΔW’s and Δb’s.

Ref : Neural networks and deep learning

**Backpropagation Intuition :**

Example : <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>

- We want to find all w’s and b’s that minimize the loss function (e.g. least square, cross entropy).

- This loss function depends on all w’s and b’s. Therefore we use GD find the last layer’s w’s and

b’s that minimize the loss function. Then fix w’s and b’s and use GD again to find w’s , b’s of

second last layer that min. loss function.

- it is similar to coordinate descent algorithm.

**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 → bigger network , more advanced opt. Algo.

High Variance → More data, regularization (L2/Dropout)

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

**Dropout regularization**=> shutting down/cancel certain portion of neurons, simpler network →

increase bias.

Other regularization : Data Augmentation,Early stopping

Normalize input → 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 → 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 => faster convergence, applied with mini match. 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 ) ← avoidable bias err → train error ← variance → Dev error.

Focus on which gap is larger, Human error should be == 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 Analsysis (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

mislabeled

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 → variance

Gap between train-dev , dev data → dis mismatch

Gap between dev, test data → 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

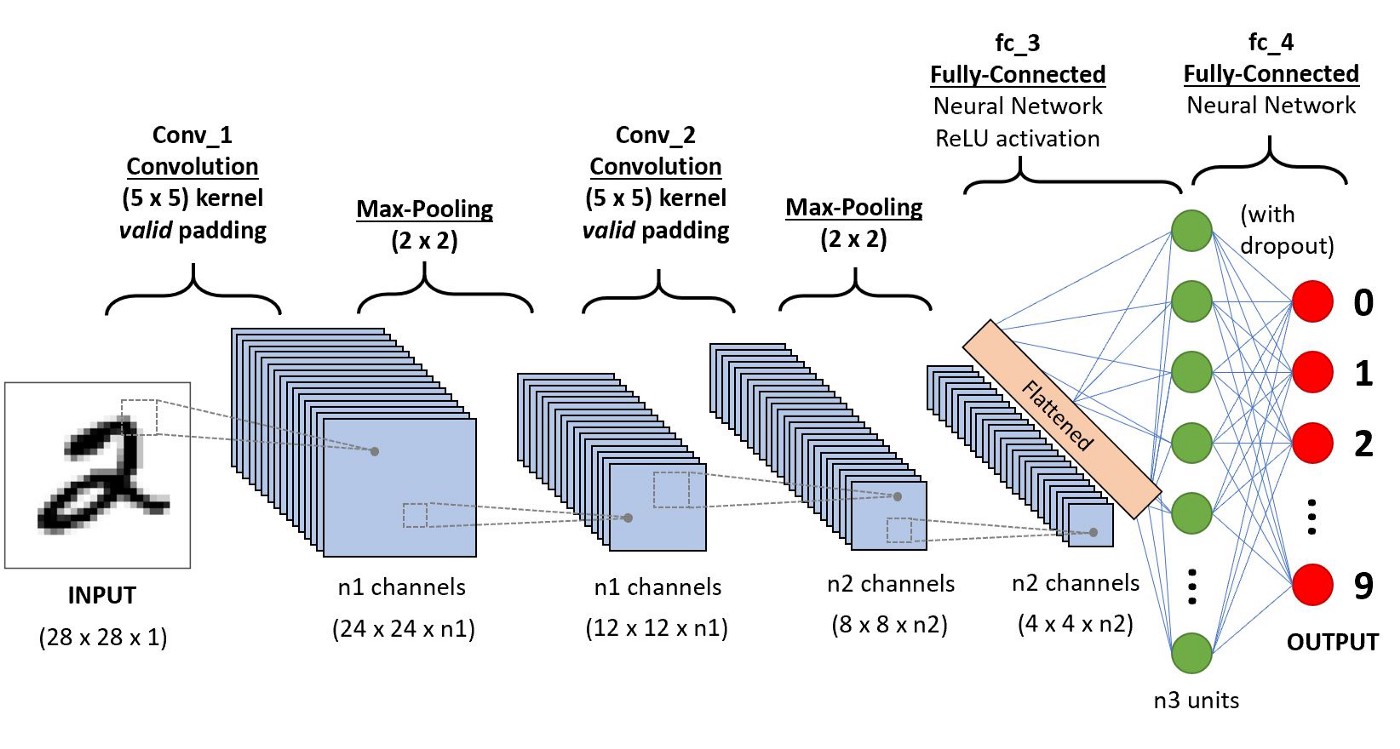
tremendous amount of data(which provides enough data for complexity).

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.

**Convolutional Neural Network (CNN)**

* Application : Image classification, Object detection etc.
* Image size may be large, impractical to use one pixel as one input.



* Convolution : apply n kernels (therefore n channels) to every pixel(with paddings), purpose : to extract high level feature (e.g. edge/contour detection)
* Pooling : decrease computational power to process data/ extract dominant features which are little rotational and positional invariant(that means positional feature may be lost). (AVG/MAX pooling),Max pooling perform noise Supressant,MAX pooling perform better than AVG.
* Softmax Classification
* Famous CNN : Alexnet

**Books for reference :**

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-learningandstatisticsspring-

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 redudant 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/

Deep Learning Books :

Neural networks and deep learning

Dive into Deep Learning — Dive into Deep Learning 0.16.5 documentation (d2l.ai)

Deep Learning MIT.

CS230,CS231, MIT Deep Learning 6.S191

Coursera Deeplearning.AI – Deep Learning Specialization.