Data Science: Deep Learning Prerequisites – Linear Reg in Python Notes

**Section 4: Practical Machine Learnining Issues**

**What do all these letters mean?**

Data

comes in pairs of inputs and ouputs

goal of predicting output given input

outputs (targets)

usually recognized by Y

can also be recognized by T

Y is a N-length vector (1-d matrix)

can also be thought of as an Nx1 column fector

inputs

usually recognized by X

X is a 2-D matrix of Size NxD

N=number of samples

D=Number of features

y-hat

same size as Y

issues with naming targets

if Y is used as target

y\_hat is used as predition

if T is used as target

Y is used as prediction

context should be immediately evident from context

Parameters of linear regression

form of y=mx+b

y=estimate

m=slope

b=intercept

could also be set up as y=ax+b

could also be done using Greek letters

in multiple linear regression

parameters are called weights and are represented by w

for example

wtx

each weight is indexed by a subscript

wi

error/cost functions

represented by E or C

goal of minimizing these values

if we have negative E (-E), then we will want to mazimize it

J is used to represent objective

what our goal is

Likelihood

maximizing log likelihood is the same as maximizing likelihood

can use both to minimize (or maximize the objective)

A>B then log(A)>log(B) and vice versa

‘L’ represents the likelihood

‘l’ represents the log-likelihood

indexing

X is a 2-d matrix

we need to use 2 indexes to get an elemnt of a

X(i,j)

need to use a nested for-loop structure to get

for i in range(N):

for j in range(D):

do something

if we have too many letters, best to use lowercase ‘n’ and ‘d’ (X(n,d) or Xnd

lower case n is index

updercase n is length

interpreting the weights

**Interpreting the weights**

interpreting the weights learned from linear regression

example of Ohm’s law

V=IR

R=resistance

V=voltage

current=I or input

means that everytime the current is increased by 1, then the voltage is increased by R

now thinking in terms of y,x,w

start with 1-d

y=mx+b

every unit increase in x results in a increase of y by w

b does not enter picture

interpreting b

this is why y is when all the x’s are zero

in amultidimentional case

example

houseprice=-0.1\*crimerate+2.5\*median income+1

for every unit increase in median income, house price increases by 2.5

same for the other inputs

**Generalizing Error: overfitting, train, and test sets**

adding more Xs as inputs can more perfectly approximate a function

why don’t we add infinite

more complex code and longer runtimes

machine learning doesn’t just try to fit past data, we want to approximate future data

generalization error

how well does model handle data that it has not yet seen before

we can simulate a situation that requires generalization

assign 20% of inputs as training data

then assign 80% as test data

train/test error curves

if you make your model so complex to fit the training data

but then the test set error increases

**this is the definition of overfitting**

Example: overfitting.py

mocking up a sin wave and make polynomial regression

make\_poly

creates a polynomial that has a degree and a certain number of Xs to create the function

fit function

finds the weights according to the function

fit\_and\_display function(X,Y,sample, deg)

sample is the number of fumctions to incorporate into training data

we use this to calculate weights

then we plot the polynomial and the sin wave in addition to the training samples

this is to figure out the how well the thing fits

get\_mse function

this is the get mean squared error for a function

plottrainvs test curves function

part 1: takes a random sample to X and y and makes it training set to make test set

part 2: fits polynomicals from degree 1 to 20 then plots the test error curves

part 3: plots the training error curves

**determines ways to get a more exact measurement**

1. then we do a for loop to test the function according to the different degrees

see which one makes the best sense

2. calculate mean squared erro for the training and test data at the same time

as a result they will stay at

can see where things generalize well and where they don’t

according to how close the functions are

one should really analyze the areas where there are no samples or gaps

train mse is always decreasing as more complexity is added

but test mean squared error increases after a certain point

what did we learn

a high degree polynomial does not always results in overfitting

it all depends on the training data

if your training data is fully representative

your fit will be good no matter what

the more data we have the better it represents reality

if training data is not representative and is bunched in certain places

then the polynomial will go wild in areas where there is no test data

**Categorical inputs-how to deal with things that aren’t numbers**

eexamples

inputs of:

gender

degree type

Solution #1: One-hot encoding

degreetype=bachelors, master, PHd

bachelors=[1,0,0]

masters=[0,1,0]

PhD=[0,0,1]

you will never see 1,1,0

if you have multiple

then you will have that many more dimensions

so for example if you have gender and degree type

you will have 5 dimensions

degree will take up 3 dimensions

gender will take up 2 dimentions

each is represented as a separate variable with a separate weight of 1 or zero,

**You will have 5 dimensions to your X even if you only have 2 “characteristics”**

example of salary

y=salary

x1=1 if male

x2=1 if male

y=50000-5000x1+5000x2

we would run a linear regression on this set

explanation

if you are female, we subtract 5000 from yhat

if male we add 5000 to yhat

solution # 2 for gender

x=1 if male

x=0 if female

y=45000+10000x

**you really don’t want to do this if you have more than one variable**

**one-hot encoding quiz**

question

want to predict return on company stock using 2 inputs

1. whether or not the company was mentioned in the news last week

2. emotional sentiment of tweets (categorized as happy, angry, sad)

What is the dimentionality of the model?

5 demenstions

1. company was mentioned

2. company was not mentioned

3. sentimate of tweets was happy

4. sentimate of tweets was angry

5. sentimate of tweets was sad

**Probabilistic interpretation of squared error**

we will take a deeper look at squared error cost function

we want to show that linear regression is the maximum likelihood solution of the line of best fit

maximum likelihood

example-plotting height of all students in school

assuming gausian distribution

we would collecte everyone’s hights (x1,x2,x32)

we know the average which is summation of Xs/N

there is a more systematic way of arriving at this answer

finding the true mean of the cuasian distribution

**see notes pages 13-15**

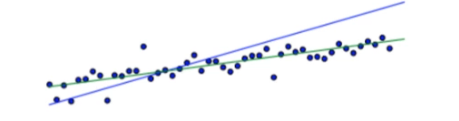
**l2 regularization-also called ridge regression**

helps us avoid overffitgin

we don’t want to have any overly large weights

might make us want to fit to outliers to minimize squared error

sourced from <https://www.udemy.com/data-science-linear-regression-in-python/learn/v4/t/lecture/6183988?start=0>



above diagram shows how outliers could screw up a regression line

How does it work?

modify original cost function by penalizing large weights

**see notes page 16-18**

**The dummy variable trap**

one-hot encoding is good for categorical variables

leads to good interpretablity because weights tell us how much that category value affects the output

if a category has K different values, then there will be K columns in X to represent it

k-1 encoding: alternative for categorical variables

save 1 column of space with all 0s or 1s to represent the value

undesirable because the effect of one value gets absorbed intothe bias term (weight)

cannot see the effect of the input variable on the output

so instead of K columns we have k-1 columns

however, there is a good reason to use k-1 encoding

moral of the story XTX is not invertable so we can use k-1 encoding directly, but there are workarounds

note: inverting a singular matrix is the equivalent of divisiding by 0

adding lambdaI is the matrix equivalent of adding a small number lambda to the denominator so that it is not zero

**look at notes page 19**

several ways to deal with dummy variable trap

1. just use k-1 encoding

2. remove all of 1s so there is no bias term

3. use l2 regularation: XTX

4. gradient descent (the most general and preferred method and used throughout deep learning)

linear regression is the only real closed form solution for weights

multicollinearity

definition: one problem of one column being the combination of several other columsn

simple case

when 2 columsn are perfectly correlated with one another

**gradiant descent is good because you generally can’t guarantee that your data isn’t correlated**

image data has strong correlations

it is used throughout deep learning

**Gradient Descent Tutorial: Hidden Markov Models in Python**

looking at gradient descent

used extensively in deep learning and other seituations

it is an optimization method

your goal is to minimize a function such as J(w) = cost of error

we can also maximize things (such as likelihood of some probability distribution) and all we have to do is reverse the signs

Example: See notes page 20

Why is gradient descent important?

as we progress functions will get more complicated

regular neurla networks with softmax

might take you a few hours or days to get derivatives the first time if you do it on paper

convolution or recurrent neural networks

possible, but definitely don’t want to spend time doing this on paper

our time is best spent testing different architectures and parameters without having to worry about gradients

we can use Theano/TensoFlow to calculate gradients for us

**good to understand what is going on so that we can use it on anything!!!!**

**Explanation of how to solve for a gradient mathematically**

<https://betterexplained.com/articles/vector-calculus-understanding-the-gradient/>

notes on page 21,22

**Gradient Descent for Linear Regression:**

see notes on page 23,24

**L1 Regularization (LASSO Regression)-Theory**

have seen adding a column of completely random noise can improve R2

but we generally want D (dimentionality) of X to be less and not close to N (number of samples)

otherwise we overfit to the training data

when we look at X matrix,

we want it to be skinny not fat

goal

select a small number of important features that predict the trend and eliminate noise

“sparsity”

most weights will be zero and a few of the weights will be non-zero

L1 regularization is like L2 regularlization

it uses a penalty term

explanation of L1 regularization

see notes on page 25

**L1 Regularization (LASSO Regression)-Code**

see the code

**L1 vs L2 Regularization**

what is the difference between L1 and L2 regularization

L1 regularization

encourages a sparse solution (fewer non-zero ws with many equal to 0)

L2 regularziation

encourages small weights (all w’s close to 0 but not exactly zero)

both help prevent overfitting by not fitting to noise

L1 does this by choosing most important features

L2 regularizations ensures weights are not exceedingly large

look at notes on page 26

you can include L1 and L2 simultaneously in your model

This is called ElasticNet

Note!!!: J is sepearte from the penalty (penalty is not included in the summation it sits in the corner