Avoid overfitting

- Increase size & quality of your data
- Data Augmentation
- Early stopping
- Regularization
- Dropout
- Decrease model complexity
- Feature selection
- Ensemble
- Cross Validation
- Transfer learning

Gradient accumulation:
- with.zero_grad() → grad_zero()

Avoid underfitting

- Increase model complexity
- increase amount of training data
- Feature engineering
- Increase # of epochs
- Use regularization (reduce regularization strength)
- ~~Cross validation~~
- Ensemble learning

(cont'd)

high variance - fits training data very closely, but it may not generalize well to unseen data

⇒ L1 and L2 regularization:

L1 regularization penalizes sum of absolute values of parameters. Model encourages to have a smaller number of large weights rather than large # of smaller weights. This leads to more sparse model where many weights are zero.

L1 - large # of features (there is need to perform feature selection)
L2 - small " " " to increase model stability.

L2 regularization:

Penalizes the sum of squares of model's weights. Model is encouraged to have a large # of smaller weights. Dense model with all nonzero weights.

$Err(x) = Bias^2 + Variance + Irreducible\ error$

↳ measure of amount of noise in data

Evaluation Metrics

Accuracy - measures how often classifier correctly predicts

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

predicted actual

static - value, retains value across fn calls

const - immutable

Unique key can include NULL

Precision - how much of positive predicted are actually positive

$$P = \frac{TP}{TP + FP}$$

Recall - how much of actual true cases we were able to predict correctly

$$Recall_2 \quad \frac{TP}{TP + FN}$$

F1 Score

$$F1 = \frac{2 \cdot Precision \times Recall}{Prec + recall}$$

ROC - true true rate & false true rate

2)

Stemming - multiple words have root

Strips prefixes & suffixes from words.

Is not word doesn't have to be word

Lemmatization

Is aim to identify actual word from dictionary

⟷

POS (part of speech) tags

Helps and confusion b/w two same words that have diff meaning

⟷

TF * IDF → Doesn't consider context → Give more weightage to rare words

↙ how often words appear in specific doc.

↘ how common word is in entire doc collection

Out-of-Vocab (OOV) - word not in vocab

NLTK / SpaCy

CBOW. (NN)

↳ NN based algo that predicts a target word given context words.

She is a [predict (target)] dancer

↳ focuses on meaning of words from its context.

Skip-gram (NN)

↳ predicts surrounding words given a single target word. It focuses on understanding how a word related to other words
↳ predict context words using distributed representation of input word.

Glove Embeddings (not NN)

↳ focuses on Co-occurrence

↳ Coocur mean syntactically similar

↔

Chunking / Shallow Parsing

↳ split sentence into chunks

~~In BoW, to score words~~ | embedding vector = vocab length

BoW score words

- → ① count
- → ② frequency (count / total words in doc)

Language models – that assign probs to sequence of words

"N-GRAM" → tokenization happens → context window

- ↳ continuous sequence of n-items
- ↳ n-gram captures the context information & relationship
- ↳ sliding a window of n words across a text corpus
- ↳ Analyze frequency of certain word sequences.

TOKENIZATION ALWAYS HAPPENS

There can be overlapping in N-gram

↳ WINDOW will be SLIDE ONE WORD AT a TIME

⇒ One-hot-encoding is applied on category

To

one-hot encoding = category data

⇒ BoW (we look a histogram of text words within the text. considering word count as a feature

Similar docs have similar contents

- ignores grammar & word order but keeps track of frequency of words.
- ↳ set vocab of known words and a frequency of known words

Feature extracting from text = count encoding words to numbers

Corpus is full Text including repetition

In BoW, next step is to score the words in each doc.

As vocab has 10 words

vocab: [It, was, the, best, of, times, worst, age, wisdom, fool.]

for "It was the best of the times" vector would be

[1 | 2 | 1 | 1 | 0 0 0 0]

↳ the - 2 times

⇒ If word is ~~encountered~~ unknown, it would be ignored

Sparse - Not of zeros

Scanned with CamScanner

SQL 2

- Select DISTINCT(Name) from tbl where CITY IN ('Paris', London') ORDER BY Price DESC
  - between
  - like (%) → 's%' - n chars after S.
  - in
- SELECT * FROM Customers Where ID = 1 And (name like '4%' OR NOT ID='F')
  - without bracket AND will get precidence
- NOT LIKE, Not BETWEEN. NOT IN
  - _ : single character
  - % : n-chars
- INSERT INTO table (col1, col2) ~~values~~
  VALUES ('x', 'y')
  VALUES ('c', 'b')
- SELECT * ~~where col-name~~ from table where col-name is NULL
- UPDATE table
  SET col1 = 'a', col2 = 'c'
  where ID = 1.
  - without where, it will be applied to whole data
- Delete from table where ID=2
- DROP Table

Aggregate fn : perform calculation on set of values

|— MIN, MAX, AVG, COUNT, SUM.

→ SELECT MIN(PRICE) AS SmallestPrice FROM table

→ SELECT MIN(PRICE) as SmallestPrice, CategoryID
FROM table
GROUP by CategoryID

IN (SELECT)
sub query

Select * FROM Customers
where CustomerID IN (SELECT CUSTOMER FROM ORDERS)

⟹ OR

Select C.* From Customers C
INNER JOIN ORDERS O ON C.CustomerID = O.CustomerID

⟸ UNION to join results of two SELECT

Select * from t1
UNION
SELECT * From t2

(Doc 2)

Where can not be used with aggregate fn
& HAVING

↓
acts before aggregate happens
↳ where acts on rows

SELECT *
FROM table
where cond
GROUP by colname
Having condt
ORDER BY name

SELECT COUNT(ID) From Customers
Group by cat where last_name = "D"
Group by last_name
Having COUNT(order.order) > 2

← EXISTS

SELECT * From Table
→ where EXISTS (SELECT)
→ ANY EXISTS (

→ SELECT P From T where P = ANY (SELECT)

STORED Procedure

```
CREATE PROCEDURE NAMEOFSP @city varchar(30),
                          @Pin varchar(6)
AS
    SELECT ....
GO;
```

→ EXEC NAMEOFSP @CITY = 'L', @PIN = 'P'

Alter table x,

Add column name datatype,
DROP COLUMN C1

CONSTRAINT, To limit type of data that goes inside table

SQL 3)

UNIQ - allow NULL

Foreign key - link b/w tables

[INDEXES help in retrieving data from DB

[Create index on

CHAR - fixed byte

VARCHAR

CHAR - 1 byte
INT - 4 byte
float - 4
Double - 8 byte
bool - 1 byte

Python - interpretation lang
↳ doesn't have to compile before run

→ FUNCTIONS/classes are first class. They can be assigned to variables, returned from other fns and passed into fn.

↳ COMPILE mean to convert into machine code. Python is not in machine but code like runtime

→ Name space → Each obj is created its name & address of object is created. object
↳ name which is assigned to each class & python.

↳ object are fns/ classes

→ Python Decorator
↳ are used to add some design pattern to a function without changing it structure

SQL 4

MVC

- Model represents data. – data retrieval & all the
- → view handles display
- → Receives input, interacts with model, updates view

MVT

- → Controller part is big take care of by framework
- → Templates acts as a presentation layer and are basically HTML code that render data. Templates are typically HTML files that contain placeholders for dynamic data

---

break exits loop

continue skips iteration

serialize → Pickling : convert to string by stream

Function that returns iterable of 1 item are Generators

Ternary operator

x if ___ else ___

Package : namespace containing multiple modules

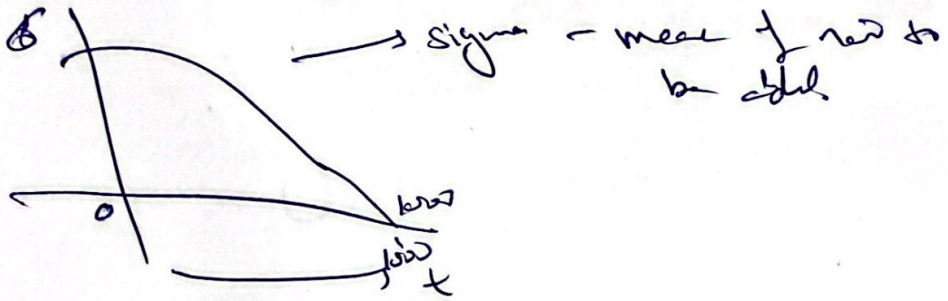
Shallow copy

changes will be reflected

Sol 6)

we added different lvl of noise at different steps

Let's create a noising schedule which is a monotonically decreasing fn



randomly pick a few levels of the added noise according to

At start t=0, pure noise

Moneith:

Inference same parameter multiple times is multiple steps. You should just increase amount you change the

So, in the offline we have [J], we add noise to it and then try to predict the noise so we can get back [J]

$$[\epsilon] = [\epsilon] + [\epsilon_\theta]$$

$$\text{MSE} = \frac{(\epsilon - \hat{\epsilon})^2}{2}$$ noise diff

Unet predict

⇒ it would be much easier if we can already input a text '3' to our denoising ↓ by clip has idea how it looks

Guidance :-

↳ if we feed in '3' & noise
↳ it will say noise is everything except 3.
this is guidance

3

$\frac{\nabla p(x_3)}{\nabla x_3}$ → we want gradient of

⇒ x_3 is a handwritten digit with

⇐ to pixels of x_3

↓

change in ~~pixel of x_3~~ pixels

if x_3 want change in pixels of x_2

can't do for every pixel, so we'll change pixel values according to gradient

For —

[So instead of changing ~~input to~~ weights of model we are changing input to model (image pixel)

So, we'll take each pixel and subtract from it a little bit of gradient

$$x_3 - c \times \frac{\nabla p(x_3)}{\nabla x_3}$$

Loss fn

Classification
- BCE
- Categ cross entropy
  - → Argmax
    - Index of largest element in a vector
  - → Softmax
    - Numbers which sum to 1

Regression
- MSE
- MAE

Overfitting (reduce techniques)
- L1 loss, L2 ridge, Dropout, Early stopping, Batch Normalization, gradient penalty, cross validation, Data Augmentation, Ensemble

validation set for hyperparameter tuning
Test set: model doesn't see during training

Lasso shrinks less important features

sparse model

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum |\theta|$$

↙

large λ mean underfit

$\lambda = 0$, mean overfitting least

fit

Ridge

→ dense model

$$+\lambda \sum |\theta|^2$$

↙

encourage smaller weights

SGD/GD → to compute gradient of loss fn.

Gradient accumulation

» An epoch means we have passed each sample of training set one time through the network to update the parameters.

» No. of epochs is hyper parameter that defines no. of times that GD will pass the entire dataset

- In batch grad, one epoch corresponds to a single iteration
- In stochastic GD, one epoch corresponds to n iterations where n is no. of training samples
- In mini batch one epoch corresponds to $\frac{n}{b}$ iterations where b is size of mini-batch

Gradient Accumulation:

↳ ~~does~~ only calculate but doesn't update model parameter in time. ∴ gradients & finally use accumulated gradients to update parameter.

Split batch size into mini batches, so all mini batches use same model weights to calculate the gradient, ~~to~~ on which is equivalent to using original batch without splitting

- zero_grad to reset gradients
- .grad : to compute gradient

3) # RNN

RNN component
- input Gate
- output Gate
- hidden state

RNN - to process sequence of data

$x(t) = x(1), \ldots, x(\tau)$ with with time step t ranging from (1 to τ)

→ RNN perform same task for every element of sequence, with output being dependent on previous computations

weight matrix
- → input matrix
- → hidden weight matrix → captures temporal dependencies. It multiplies the previous hidden state (h_{t-1}) at each time step.

Hidden state $(h_t)_2 f(W_x x_t + W_h h_{t-1} + b)$

ReLU is best
- not correct so derivate is not possible bcz fn is not defined there

by GDescent won't work

Gradient Descent is optimization technique

Back propagation is to calculate gradient for GD

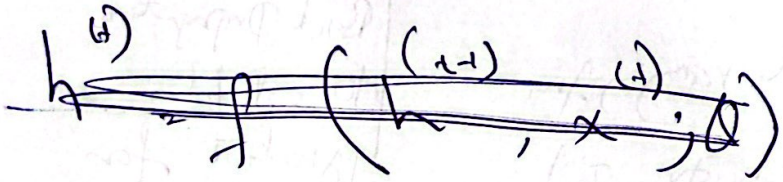
Hidden state $h_t = f(W_x * X_t + W_h * h_{t-1} + b)$

↓

activation fn

W_x - input weight matrix - ~~to convert X_t~~

It converts X_t (input matrix) into format suitable for combining with previous previous hidden state



LSTM - three gate structure

- forget
- input
- output

GRU

- update
- reset

R-CNN

↳ You can not use standard CNN for detect

is fixed Fully conn layer at end

- No info about localisation,
- Multiple object

image → 2k region proposals → Compute CNN feats → classify

- → classification loss — Softmax
- → bbox regression loss

Fast RCNN

img → CNN map → region proposals

↳

Faster RCNN

instead of selective search, CNN gives region proposals

Scanned with CamScanner

YOLO

↳ single CNN predict box, & da

↳ image into S×S grid

↳ works on bounding boxes

⟵

SVM → find hyperplane in n-dimens space

⟵

mAP - det detect

5)

Sequence to sequence

↳ ~~Machine trans~~ translate

↳

↳ convert & create & decoder

UNET

CLIP - Multimodal.

↳

torch.no_grad - disables computing gradients

model. eval discard

↳ dropout layers are bypassed

↳ batch normalization uses pre computed mean & var.

Unet consist of
- contracting path (encoder)
- Expansive method (decoder)
  - use skip connection to generate segmentation map

Skip connections
- are concatenate

Depth allows network to capture high-level feature

Each pixel in output img represent a label that correspond to a class in ~~output~~ input

loss:
- pixel wise cross cross entropy
- SLD.

Max pool
- dimensionality reduction
- noise suppression
- translation invariance.
- robustness to small transformation

Unet consist of
- contracting path (encoder)
- Expansive method (decoder)
  - use skip connection to generate segmentation map

Skip connections
- are concatenate

Depth allows network to capture high-level feature

Each pixel in output img represent a label that correspond to a class in ~~output~~ input

loss:
- pixel wise cross cross entropy
- SLD.

Max pool
- dimensionality reduction
- noise suppression
- translation invariance
- robustness to small transformation

Coursera LLMs (1)

FCNN

↳ deep ~~con~~ features can be obtained from going deeper but spatial info's lost.

mean

Shallow layer have

↳ high spatial info
↳ low level features

& we need to use concatn b/w them

↳ ⇒ output of conv layer have same depth as # of filters

⇒ Filter has same depth as input

Dilated conv has gaps ~~b/w~~ when applied to input. results in layers receptive area

↳ Gap is introduced in kernel.

1×1 Conv

↳ to reduce # of channels while introduces non-linearity

↳ It mean a single number (1×1) as opposed to matrix.

This 1×1 pixel will convolve one entire input image pixel-by-pixel

⇒ 1×1 filter will only have a single parameter for each channel in input

* Depth wise Conv
  ↳ single channel at a time unlike CNN where all in at once
* Point conv
  ↳ 1×1

Coursera IIM. (1)

IoU - loss fn. - how well img is segmented

↳ area of overlap / area of union

DICE as loss fn

overlap

DICE - loss fn for pixel segments

↳ for unbalanced dataset.

↳ measure of disparity b/w predicted & the actual segment

$$DL(y,\hat{y}) = 1 - \frac{2y\hat{y}+1}{y+\hat{y}+1}$$

$$R^2 = 1 - \frac{\text{sum of square residuals}}{\text{Total sum of square}}$$

R2 - eval of regression model

$R2 = 1$ - perfect fit

fitting mean

actual variance total from regression line

$R2 = 0$ - no effect in target value

actual value from Mean

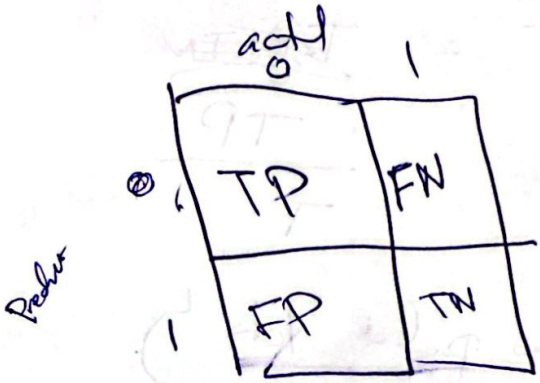
bias & variance

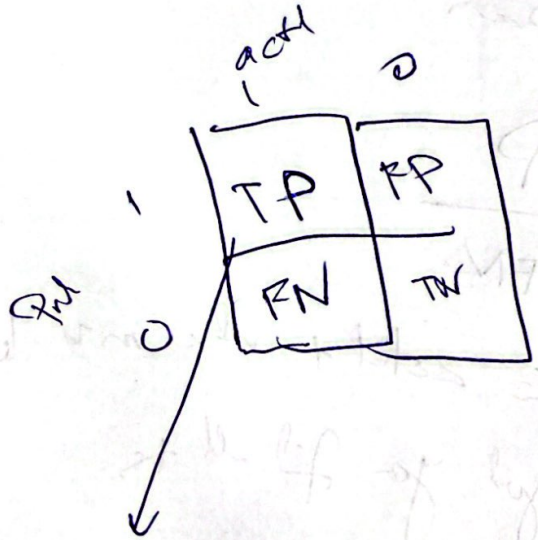
$$\frac{\text{total var explained by model}}{\text{~~total~~ actual total var}}$$

If two variables are correlated,

no variance at all.

Confusion matrix





TP: Model predicted +ve, actl: pos

FN: model " -ve, actl: +ve

TN: " " -ve, " " -ve

FP: " +ve, " " -ve

Precision = ~~TP+TN~~

$$\frac{TP}{T}$$

→ saare predict kiye }

→ right pred

$$\frac{TP}{TP+FN}$$

Recall

→ kitne actual true value mein kiye

→ how good you find all true

$$\frac{TP}{TP+FN}$$

$$\text{accuracy} = \frac{TP+TN}{\text{total}}$$

$$F_1 = \frac{2 \cdot \text{Prec} \cdot \text{Recall}}{\text{prec} + \text{recall}}$$

Scanned with CamScanner

P — are ~~sk~~ under pres. recall acc

Preci = $\frac{2}{3}$ = 0.67

recall = 2/5 = 0.4

T	1.0	, 0.2
T	1.0	, 0.4
R	0.6	, 0.4

sigmoid weight

Swish?

swish

Swiglu

x tanh ≠ x sigmoid

BERT SCORE → generative

→ evaluate similarity b/w generated text & contextual text.

↳ scaling LLM

BLEU — bilingual

↳ machine translate text by comparing it with human translation

GLUE — how well ~~hum~~ machine understand human lang.

Perplexity

GELU — monotonic

↳ $x\phi(x)$

↓

cumulative dist fn

ARC , AI2 Reasoning challenge. - reasoning [illegible]

TruthfulQA - coherent & well structured response
is answer [illegible]

SQL

HumanEval , functionally correct code

MT bench - multi turn dialogue

chain of thought, you give reasoning step-by-step

Tree of thought. → React based on
chain of thought

PEFT - identify most
significant params & finetune

LoRA input transfer
low rank decomposition

R-CNN

- ↳ Y can not use standard CNN for detect
  - ↳ fixed fully conn layer at end
    - No info about localiz.
    - Multiple object

image → 2k region proposals → Compute CNN feat. → classify

- ↳ classification loss — softmax
- ↳ bbox regre loss

⟵⟶

Fast RCNN

Img → CNN map → region proposals

↳

Faster RCNN

instead of selective search, CNN gives region proposals

YOLO

↳ style CNN predict Box, & class

↳ image into S×S grid

↳ work on bounding boxes

SVM → find hyperplane in n-dimensional space

mAP - detection

Coursera LLMs (1)

completion - output of model.

Zero shot inference - No examples in prompt

One shot - One " " "

Few shot - Multiple examples
inference

↳ Small models might need one/few shot inferences

⇒ * Gen configs - inference Parameters

→ max_tokens

→ top k → # of tokens to choose from
→ select an output from top k results
then apply random weighted strategy

→ top p using prob
↳ prob to choose from (cumulative prob)

→ Temperature.