Cross Entropy-

1. How to improve the entropy function?

ans: We know,

entropy = $-\log(P) - \log(1-P)$ [P = Probability]

Let's consider a seenerio where there are 10 people classified into good people or bad people.

good	bad	Probability of good	Probability of bad
10	0	10/10	0/10
9	1	9/10	1/10
8	2	8/10	2/10
7	3	7/10	3/10
6	4	6/10	4/10
5	5	5/10	5/10
4	6	4/10	6/10
3	7	3/10	7/10
2	8	2/10	8/10
1	9	1/10	9/10
0	10	0/10	10/10

violence increasing ↓

maximum violence →

violence increasing ↑

From the above table, when the number of bad people is increasing, violence increasing. When the number of bad and good people is equal there is maximum violence. If we want to calculate entropy for $P(good) = \frac{8}{10}$ and $P(bad) = \frac{2}{10}$,

$$\text{Entropy} = -\log\frac{8}{10} - \log\frac{2}{10} = 0.7958800173 \approx 0.8$$

Again if we calculate entropy for $P(good) = \frac{2}{10}$, $P(bad) = \frac{8}{10}$.

$$\text{Entropy} = -\log\frac{2}{10} - \log\frac{8}{10} = 0.7958800173 \approx 0.8$$

In both cases, entropy is equal. But in actual, when there are 8 good people and 2 bad people, violence won't be the same when there are 2 good people and 8 bad people. In order to get rid of this problem, we can introduce a weighting factor.

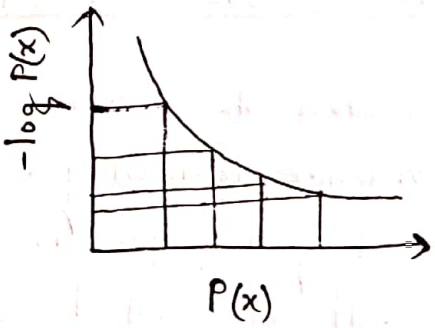
$$\therefore \text{Entropy} = -P \log(P) - (1-P) \log(1-P)$$

(P → w_1, $(1-P)$ → w_2)

Here, w_1 and w_2 are weighting factors. Now for $P(\text{good}) = \frac{8}{10}$ and $P(\text{bad}) = \frac{2}{10}$,

$$\text{Entropy} = -\frac{8}{10} \log\left(\frac{8}{10}\right) - \frac{2}{10} \log\left(\frac{2}{10}\right)$$

By adding weighting factor, we are improving entropy function as we can see if P is high, (1-P) will be low. But when P is high log(P) will be low and when (1-P) is low log(1-P) will be high. So that, the fareness will be decreased.

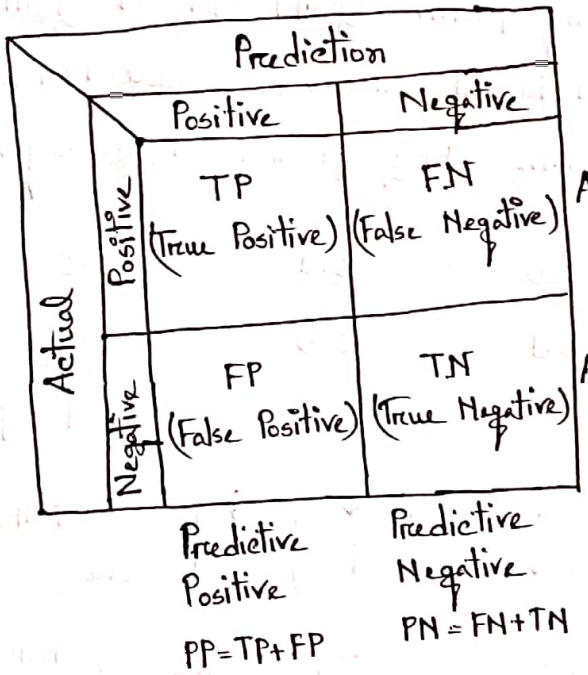


We can see a rectangular hyperbolic figure. So, we have to improve weighting factor to improve entropy function. We have to research on data and find out the maximum fareness in order to get better entropy function.

Performance Evaluation

Suggestion from sir:
1) Scenario based question
2) Specificity, precision, recall, F1-score, accuracy কখন কোথায় বসবি?

A confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one.



$\therefore$ Total/Sum = P+N / PP+PN = S

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

imbalaneed dataset

100 data

95 positive 5 negative

যদি data imbalanced হয় মানে data distribution equal না হয়, তখন accuracy দিয়ে model এর performance বুঝা যায় না।

$\therefore$ Precision = $\frac{TP}{PP} = \frac{TP}{TP+FP}$

আমার predict করা positive গুলোর মধ্যে আসলেই positive কোনগুলো তা বের করাই precision. যেমন:

10টা MCQ তে,

1) 6 টা ans করল → 5 টা correct, 1 টা incorrect
   4 টা বাদ দিল

2) 10 টা ans করল → 5 টা correct, 5 টা incorrect
   0 টা বাদ দিল

more precized, এটার precision বেশি

$$\therefore \text{Recall} = \frac{TP}{P} = \frac{TP}{TP+FN}$$

Recall হচ্ছে actually positive গুলোর মধ্যে কতগুলো positive find out করছে। Recall = discovering rate। যেমন:

Covid positive কিন্তু predict করছে negative

Precision	: 0.9	0.1	0.6
Recall	: 0.1	0.9	0.6 (best option)

Precision ↑ Recall ↓
Precision ↓ Recall ↑
} balance করার জন্য f1-score

$$\therefore \text{f1-score} = \frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} \times 2$$

: harmonic mean of precision and recall

at the same time discover করা + true predict করা both important তখন f1-score লাগে।

$$\therefore \text{Specificity} = \frac{TN}{TN+FP} = \text{True Negative Rate (TNR)}$$

actual negative গুলোর মধ্যে কতগুলো negative ঠিকমত predict হলো।

Previous : Fall-2020: 4(b)

Data Preprocessing

Important topics :

1) missing data handling
2) normalization কেন দরকার? কীভাবে করা হয়?
3) binarization, discretization কেন দরকার? কীভাবে হয়?

suggestion from sir :

1) কোন step এ কী হচ্ছে? কেন হচ্ছে?
2) math

missing data handing :

i) ignore the tuple : যেই data missing, ঐ row বাদ দিয়ে দেয়া which is not a good solution.

ii) Use the attribute mean to fill in the missing value : সব attribute এর value যোগ করে total no. দিয়ে ভাগ করে mean বের করে missing space এ বসানো। যেমন: 20, 30, 40, 50 এর mean 35। So, missing data = 35.

iii) Use the attribute mean for all samples belonging to the same class : যে class এর data missing শুধু ঐ class এর mean বের করা। যেমন: class 1 = 20, 30, 40; class 2 = 50, 60। class 1 এর missing data হবে (20+30+40)/3 = 30

iv) Predict the missing value by using a learning algorithm : missing data বাদে বাকি data গুলো কোন learning algorithm এর সাহায্যে learn করিয়ে missing data কি হতে পারে তা predict করে নেয়া। Regression এর সাহায্যে predict করা যেতে পারে।

normalization :

Suppose কোন marks এর distribution 0-20 or 0-100 or 250-300। এরকম different range এর data নিলে model কে

learn করানো কষ্ট হয়ে যায়। model কে আরও resource দেয়া লাগে বোঝানোর জন্য। তাই model যেন overfit বা underfit না হয় তাই data গুলোকে normalize করে ফেলা হয়। Normalize করা মানে data কে একটা particular range এর মধ্যে আনা।
যেমন: to transform V in [min, max] to V' in [0,1]

$$V' = \frac{V - Min}{Max - Min}$$ [linear transformation formula]

Data normalization এর standard way হচ্ছে mean/average কে subtract করে standard deviation দিয়ে ভাগ করা। মানে,

$$V' = \frac{V - Mean}{Standard\ Deviation}$$

Normal distribution এর মধ্যে রাখার জন্য এই technique। outliers যখন থাকে + min/max দেয়া না থাকে তার জন্য এই way।

Discretization:
Discretization is useful to increase the generalization and accuracy of discovered knowledge. It's the process of dividing the range of the continuous attribute into intervals. Every interval is labeled a discrete value and then the original data will be mapped to the discrete values.

unsupervised discretization:
→ equal interval (equiwidth) binning
→ equal frequency (equidepth) binning

Data : 0, 4, 12, 16, 16, 18, 24, 26, 28

equal width:
bin 1 : 0, 4 [0, 10)
bin 2 : 12, 16, 16, 18 [10, 20)
bin 3 : 24, 26, 28 [20, 30)

} value গুলোকে same range এ রাখলাম। তা ভাগ করে প্রতি ভাগে 10 এর মধ্যে range। e.g. $= \frac{0+28}{3} \approx 10$ ← $\frac{1^{st}\ value + last\ value}{no.\ of\ bin}$

equal frequency:

bin 1 : 0, 4, 12 [0, 14)
bin 2 : 16, 16, 18 [14, 21)
bin 3 : 24, 26, 28 [21, 30)

} ৩টা data ছিল। equally 3 টা bin করে 3 টা করে value দেয়া হয়েছে each bin এ

supervised discretization : (check slide)

math : (sir বলছে এটা আসবেই, না পড়লে fail)

$$E_p = -\frac{3}{10}\log\frac{3}{10} - \frac{7}{10}\log\frac{7}{10} = 0.2661357085 \approx 0.266$$

For 55: $E_{c1} = -\frac{0}{0}\log\frac{0}{0} - \frac{0}{0}\log\frac{0}{0} = 0$

$$E_{c2} = -\frac{3}{10}\log\frac{3}{10} - \frac{7}{10}\log\frac{7}{10} = 0.2661357085$$

$$\therefore E_c = \frac{0}{10}E_{c1} + \frac{10}{10}E_{c2} = 0\times0 + 1\times0.2661357085 \approx 0.266$$

$$\therefore E_{gain} = E_p - E_c = 0.266 - 0.266 = 0$$

For 65: $E_{c1} = -\frac{0}{1}\log\left(\frac{0}{1}\right) - \frac{1}{1}\log\left(\frac{1}{1}\right) = 0$

$$E_{c2} = -\frac{3}{9}\log\frac{3}{9} - \frac{6}{9}\log\left(\frac{6}{9}\right) = 0.2764345909 \approx 0.276$$

$$\therefore E_c = \frac{1}{10}E_{c1} + \frac{9}{10}E_{c2} = \frac{1}{10}\times0 + \frac{9}{10}\times0.276 \approx 0.2488$$

$$\therefore E_{gain} = E_p - E_c = 0.266 - 0.2488 \approx 0.017$$

For 72: $E_{c1} = -\frac{0}{2}\log\left(\frac{0}{2}\right) - \frac{2}{2}\log\left(\frac{2}{2}\right) = 0$

$$E_{c2} = -\frac{3}{8}\log\left(\frac{3}{8}\right) - \frac{5}{8}\log\left(\frac{5}{8}\right) = 0.2873132638 \approx 0.287$$

$$\therefore E_c = \frac{2}{10}E_{c1} + \frac{8}{10}E_{c2} = \frac{1}{5}\times0 + \frac{4}{5}\times0.287 \approx 0.2299$$

$$\therefore E_{gain} = E_p - E_c = 0.266 - 0.2299 \approx 0.0363$$

For 80: $E_{c1} = -\frac{0}{3}\log\left(\frac{0}{3}\right) - \frac{3}{3}\log\left(\frac{3}{3}\right) = 0$

$$E_{c2} = -\frac{3}{7}\log\left(\frac{3}{7}\right) - \frac{4}{7}\log\left(\frac{4}{7}\right) = 0.2965832215 \approx 0.2966$$

$$\therefore E_c = \frac{3}{10}E_{c1} + \frac{7}{10}E_{c2} = \frac{3}{10}\times0 + \frac{7}{10}\times0.2966 \approx 0.208$$

$$\therefore E_{gain} = E_p - E_c = 0.266 - 0.208 \approx 0.0585$$

For 87: $E_{c1} = -\frac{1}{4}\log\left(\frac{1}{4}\right) - \frac{3}{4}\log\left(\frac{3}{4}\right) = 0.2442190503 \approx 0.244$

$E_{c2} = -\frac{2}{6}\log\left(\frac{2}{6}\right) - \frac{4}{6}\log\left(\frac{4}{6}\right) = 0.2764345909 \approx 0.276$

$\therefore E_c = \frac{4}{10}E_{c1} + \frac{6}{10}E_{c2} = \frac{2}{5} \times 0.244 + \frac{3}{5} \times 0.276 \approx 0.264$

$\therefore E_{gain} = E_p - E_c = 0.266 - 0.264 \approx 0.0026$

For 92: $E_{c1} = -\frac{2}{5}\log\left(\frac{2}{5}\right) - \frac{3}{5}\log\left(\frac{3}{5}\right) = 0.2922852532 \approx 0.292$

$E_{c2} = -\frac{1}{5}\log\left(\frac{1}{5}\right) - \frac{4}{5}\log\left(\frac{4}{5}\right) = 0.2173220113 \approx 0.217$

$\therefore E_c = \frac{5}{10}E_{c1} + \frac{5}{10}E_{c2} = \frac{1}{2} \times 0.292 + \frac{1}{2} \times 0.217 \approx 0.2548$

$\therefore E_{gain} = E_p - E_c = 0.266 - 0.2548 \approx 0.0113$

For 97: $E_{c1} = -\frac{3}{6}\log\left(\frac{3}{6}\right) - \frac{3}{6}\log\left(\frac{3}{6}\right) = 0.3010299957 \approx 0.301$

$E_{c2} = -\frac{0}{4}\log\left(\frac{0}{4}\right) - \frac{4}{4}\log\left(\frac{4}{4}\right) = 0$

$\therefore E_c = \frac{6}{10}E_{c1} + \frac{4}{10}E_{c2} = \frac{3}{5} \times 0.301 + \frac{2}{5} \times 0 \approx 0.18062$

$\therefore E_{gain} = E_p - E_c = 0.266 - 0.18062 \approx 0.0855$

For 110: $E_{c1} = -\frac{3}{7}\log\left(\frac{3}{7}\right) - \frac{4}{7}\log\left(\frac{4}{7}\right) = 0.2965832215 \approx 0.297$

$E_{c2} = -\frac{0}{3}\log\left(\frac{0}{3}\right) - \frac{3}{3}\log\left(\frac{3}{3}\right) = 0$

$\therefore E_c = \frac{7}{10}E_{c1} + \frac{3}{10}E_{c1} = \frac{7}{10} \times 0.297 + \frac{3}{10} \times 0 \approx 0.208$

$\therefore E_{gain} = E_p - E_c = 0.266 - 0.208 \approx 0.05853$

For 122: $E_{c1} = -\frac{3}{8}\log\left(\frac{3}{8}\right) - \frac{5}{8}\log\left(\frac{5}{8}\right) = 0.2873132638 \approx 0.287$

$E_{c2} = -\frac{0}{2}\log\left(\frac{0}{2}\right) - \frac{2}{2}\log\left(\frac{2}{2}\right) = 0$

$\therefore E_c = \frac{8}{10}E_{c1} + \frac{2}{10}E_{c2} = \frac{4}{5} \times 0.287 + \frac{1}{5} \times 0 \approx 0.2299$

$\therefore E_{gain} = E_p - E_c = 0.266 - 0.2299 \approx 0.0363$

$\therefore$ Division point = $\max(E_{gain}) = 0.0855$ [For 97]

(ans:)

Optimizers

Suggestion from sir :
1) direct ques থাকবে : যেকোন একটা optimizer এর advantage/disadvantage

Stochastic Gradient Descent :

$$\hat{g} \leftarrow + \frac{1}{m} \nabla_\theta \sum_i L(f(x_i;\theta), y_i)$$

$$\theta \leftarrow \theta - \epsilon \hat{g}$$

advantage : reduce dependency
disadvantage : non convex problem, slow convergence

Momentum :

$$\hat{g} \leftarrow + \frac{1}{m} \nabla_\theta \sum_i L(f(x_i;\theta), y_i)$$

$$v \leftarrow \alpha v - \epsilon \hat{g}$$

$$\theta \leftarrow \theta + v$$

advantage : faster convergence, reduced oscillation
disadvantage : blindly follow slopes

Nestercov momentum :

$$\hat{\theta} \leftarrow \theta + \alpha v$$

$$\hat{g} \leftarrow + \frac{1}{m} \nabla_{\hat{\theta}} \sum_i L(f(x_i;\hat{\theta}), y_i)$$

$$v \leftarrow \alpha v - \epsilon \hat{g}$$

$$\theta \leftarrow \theta + v$$

advantage : faster convergence, know where it's going
disadvantage : not adaptive

AdaGrad :

$$\hat{g} \leftarrow + \frac{1}{m} \nabla_\theta \sum_i L(f(x_i;\theta), y_i)$$

$$r \leftarrow r + \hat{g} \odot \hat{g}$$

$$\Delta\theta \leftarrow - \frac{\epsilon}{\delta + \sqrt{r}} \odot \hat{g}$$

$$\theta \leftarrow \theta + \Delta\theta$$

advantage : adaptive
disadvantage : keeps going, learning rate shrinks

RMSProp :

$$\hat{g} \leftarrow +\frac{1}{m} \nabla_\theta \Sigma_i L\left(f(x_i;\theta), y_i\right)$$

$$r \leftarrow \rho r + (1-\rho)\, \hat{g} \odot \hat{g}$$

$$\Delta\theta \leftarrow -\frac{\epsilon}{\delta + \sqrt{r}} \odot \hat{g}$$

$$\theta \leftarrow \theta + \Delta\theta$$

advantage : recursive, learning rate optimally high
disadvantage : average of past gradients

RMSProp with Nesterov Momentum :

$$\hat{\theta} \leftarrow \theta + \alpha v$$

$$\hat{g} \leftarrow +\frac{1}{m} \nabla_{\hat{\theta}} \Sigma_i L\left(f(x_i;\hat{\theta}), y_i\right)$$

$$r \leftarrow \rho r + (1-\rho)\, \hat{g} \odot \hat{g}$$

$$v \leftarrow \alpha v - \frac{\epsilon}{\delta + \sqrt{r}} \odot \hat{g}$$

$$\theta \leftarrow \theta + v$$

Adam :

$$\hat{g} \leftarrow +\frac{1}{m} \nabla_\theta \sum_{\substack{i \\ t \leftarrow t+1}} L\left(f(x_i;\theta), y_i\right)$$

$$s_t \leftarrow \rho_1 s_t + (1-\rho_1)\, \hat{g}$$

$$r_t \leftarrow \rho_2 r_t + (1-\rho_2)\, \hat{g} \odot \hat{g}$$

$$\hat{s}_t \leftarrow \frac{s_t}{1-\rho_1^t}$$

$$\hat{r}_t \leftarrow \frac{r_t}{1-\rho_2^t}$$

$$\Delta\theta \leftarrow -\frac{\epsilon}{\delta + \sqrt{\hat{r}_t}} \odot \hat{s}_t$$

$$\theta \leftarrow \theta + \Delta\theta$$

advantage : slows down near local minima, adaptive
disadvantage : computationally costly

Bag of Words

Important topics:
1) Definition
2) Math

Suggestion from sir:
1) Drawbacks of BoW

Drawbacks of BoW:

1) If the new sentences contain new words, then our vocabulary size would increase and thereby the length of the vectors would increase too.
2) Additionally, the vectors would also contain many 0s, thereby resulting in a sparse matrix (which is what we like to avoid)
3) We are retaining no information on the grammar of the sentences nore on the ordering of the words in the text.

TF-IDF: (Term Frequency-Inverse Document Frequency)

Suggestion from sir:
1) Math (Lab Final এর মতো)

Math:

Spring: 2020:

5) a) i) Vocabulary = 'a', 'man', 'is', 'throwing', 'frisbee', 'in', '@', 'park', 'holding', 'Frisbee', '##', 'his', 'hand', 'm@n', 'standing', 'the', 'gra$s', 'with'

iii)

Term	Line:1	Line:2	Line:3	TF 1	TF 2	TF 3	IDF	TF-IDF 1	TF-IDF 2	TF-IDF 3
a	2	1	1	2/9	1/10	1/9	0	0	0	0
man	1	1	0	1/9	1/10	0	0.18	0.02	0.018	0
is	1	1	1	1/9	1/10	1/9	0	0	0	0
throwing	1	0	0	1/9	0	0	0.48	0.053	0	0
Frisbee	1	0	1	1/9	0	1/9	0.18	0.02	0	0.02
in	1	1	1	1/9	1/10	1/9	0	0	0	0
@	1	1	0	1/9	1/10	0	0.18	0.02	0.018	0
park	1	0	0	1/9	0	0	0.48	0.053	0	0
holding	0	1	0	0	1/10	0	0.48	0	0.048	0
Frisbee	0	1	0	0	1/10	0	0.48	0	0.048	0
##	0	1	0	0	1/10	0	0.48	0	0.048	0
his	0	1	0	0	1/10	0	0.48	0	0.048	0
hand	0	1	0	0	1/10	0	0.48	0	0.048	0
m@n	0	0	1	0	0	1/9	0.48	0	0	0.053
standing	0	0	1	0	0	1/9	0.48	0	0	0.053
the	0	0	1	0	0	1/9	0.48	0	0	0.053
grafs	0	0	1	0	0	1/9	0.48	0	0	0.053
with	0	0	1	0	0	1/9	0.48	0	0	0.053

$$IDF = \log\left(\frac{\text{number of document}}{\text{number of document containing the word}}\right)$$

Hyperparameters & Parameters

Important topics:

1) Tuning Techniques

Suggestion from sir:

1) কোনটা parameter? কোনটা hyperparameter? কীভাবে deal করব?

Difference between parameter and hyper parameter:

Parameter	Hyper Parameter
1) Internal to the model	1) External to the model
2) Value can be estimated from data	2) Value can't be estimated from data
3) Required by model when making predictions	3) Tuned for a given predictive modeling problem
4) Often not set manually by the practitioner	4) Set before training
5) Example: weight matrix of a model, coefficients in a linear regression, depth of a decision tree	5) Example: dropout, number of hidden layer, active function, epoch, batch size, optimizer, learning rate

Tuning Techniques:

Random search:

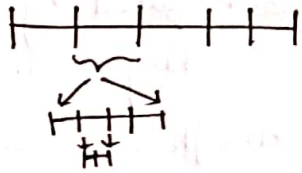
We create a grid of possible values for hyperparameters. Each iteration tries a random combination of hyperparameters from this grid, records the performance and lastly returns the combination of hyper parameters that provided the best performance.

Grid Search:

In the grid search method, we create a grid of possible values for hyperparameters. Each iteration tries a combination of hyperparameters in a specific order. It fits the model on each and every combination of hyperparameters possible and records the model performance. Finally it returns the best model with the best hyperparameters.

Exhaustive Search:

We create a grid of possible values for hyper parameters. Then finds the range for which the performance is best. Then again makes a grid inside that grid and search for the lowest loss. That's how it works.



Bayesian Optimizer:

Bayesian optimization helps us find the minimal point in the minimum number of steps. It explore the space of potential choices of hyper parameters by deciding which combination to explore next based on previous observations.

Gradient Based:

For specific learning algorithms, after computing gradient, we optimize hyperparameter using gradient descent by adopting a continuous relaxation of parameters.

Overfitting + Underfitting

Important Topics:
1) Relation with model complexity

Suggestion from sir:
1) bias + variance + trade off
2) detection + reduction technique
3) quiz এর মত question

Reduction Technique:

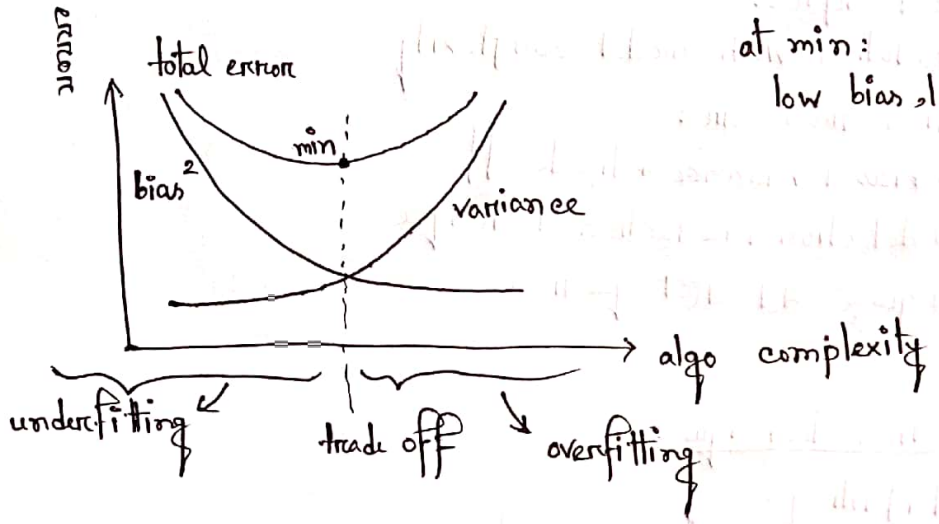
Underfitting:
1) increase model complexity
2) increase number of features
3) remove noise from data
4) increase number of epochs

Overfitting:
1) reduce model complexity
2) early stopping during the training phase
3) Ridge regularization and Lasso regularization
4) use dropout for neural networks

Detection Technique:

1) Train-test split : if training performance is good but testing performance is bad then it's overfitting.
If both training and testing performance is bad then it's underfitting.
2) Holdout cross validation

Bias + variance + trade-off:



at min:
low bias, low variance

Total error = bias² + variance + irreducible error

mean থেকে কত দূরে : bias
কতটা scattered : variance

underfitting : high bias + low variance
overfitting : high variance + low bias

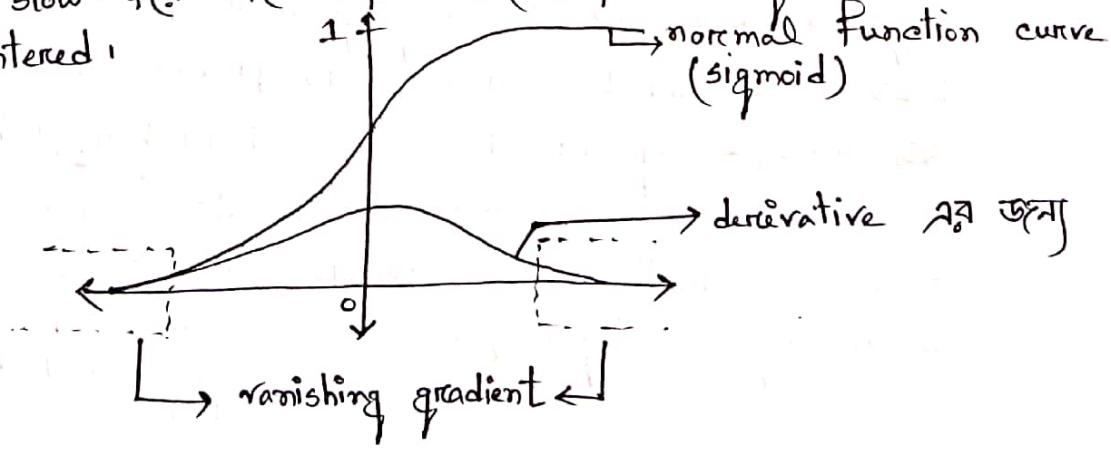
important ques : cost function কেন hyperbolic?

Activation Function

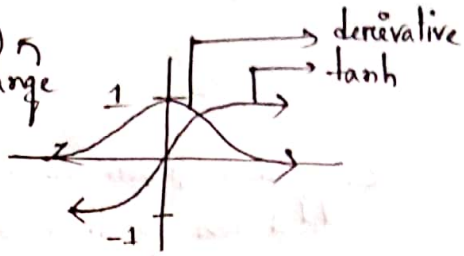
Suggestion from sir :
1) এটা question থাকবে
2) ReLU, Leaky ReLU, Sigmoid, Tanh, Softmax এর কাজ কী? কোনটা কখন use হয়?

Sigmoid/Logistic : function = $\frac{1}{1+e^{-x}}$ = ∅ (0-1) ← range

এটার max value 1, min value 0। এই function apply করলে data smooth হয়ে যায়। x এর যেকোন value কে ∅ এর মধ্যে transform করে যার range 0-1. Advantage হচ্ছে বড় value গুলোকে short range এ আনে। x এর $-\infty$ to $+\infty$ এর value কে ∅ এর 0 to 1 এ নিয়ে আসে। Back propagation এ function টার derivative use হয়। derivative এর curve টাও smooth + continuous। এটার both side এই value 0। মানে এটা certain time পর value 0 হয়ে যায়। value maximum হয় midpoint এ। কি পরিমান learn করব তা depend করে back propagation এর উপর। So, derivative যত ভালো, learning তত smooth হবে। So, sigmoid function এর advantage হচ্ছে continuous curve দেয়। Problem হচ্ছে Lower point এ gradient নাই। ফলে lower point এ learning একদম কম হবে। Lower point = 0 এর কাছাকাছি point। আবার derivation curve এর maximum point ও খুব ছোট। এটা vanishing gradient. Non-convex problem এ (যেখানে multiple minima থাকে : local, global) local minima এর মধ্যে আসার পর মনে করতে পারে এটাই best solution। এই point এ learning rate slow হয়ে যাবে। Expensive (computationally) function। Not 0 centered।



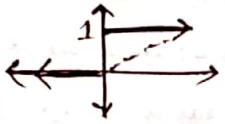
Hyperbolic Tangent: $y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ (-1,1) range



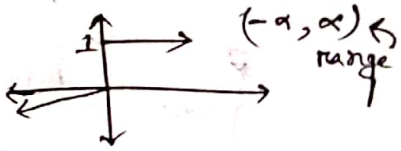
আগেরটাই just range হচ্ছে -1 থেকে +1। advantage হচ্ছে derivative টা এখন +1 এ চলে এসেছে। ফলে Learning rate বেশি থাকবে। Zero center পাওয়া যাবে। range বড়। Learning rate একটু faster হবে। কিন্তু slow learning rate still exists। application: sigmoid এ শুধু reward এ use করা যেতে পারে, tanh এ যেহেতু +1, -1 তাই reward + punishment এ use হতে পারে।

Rectified Linear Unit: $y = \begin{cases} 0 & x<0 \\ x & x \geq 0 \end{cases}$ (piecewise linear function) (0-∞ range)

Computation সবচেয়ে easy। Sigmoid এ Loss এর ক্ষেত্রে each point এ একটা value পাব। কিন্তু ReLU তে each point এ loss এর value similar হবে। output layer এ তাই ReLU used হয়না। অনেক neuron থাকলে sigmoid/tanh use করলে slow হয়ে যাবে। প্রত্যেকটার জন্য এত calculation efficient না। ReLU সব neuron কে activate করেনা at the same time। ReLU এর performance better। তাই hidden layer এ use হয়।



Leaky ReLU: $y = \begin{cases} x & x \geq 0 \\ 0.01 & x<0 \end{cases}$



non positive এর জন্য 0 এর জায়গায় কিছু একটা value দেয়।

Softmax:

Sigmoid + tanh এর vanishing gradient problem কে দূর করে। ReLU তে 0 আর 1 return করার problem টাও solve করে। এটা multiple sigmoid কে combine করে। Then probabilistic value দেয় as an output। softmax প্রায়ই output layer এ used হয়। Multiclass classification এ softmax use হয়। disadvantage: costly। $a_i = \frac{e^{z_i}}{\sum_{k=1}^{c} e^{z_k}}$ value টা always same for all calc। তাই normalize করে cost কমাতে পারি।

RNN + LSTM

Suggestion from sir :
1) Derivation
2) Quiz:03

RNN:
DNN model গুলো কয়েকটা লাইনের মধ্যের সামঞ্জস্য খুঁজে পায়না।

A recurrent neural network can be ... ...

DNN প্রতিটা word কে আলাদা consider করে
CNN suppose 3টা করে word নিয়ে তাদের মধ্যে connection বানানোর try করবে কিন্তু Line এর শুরুর A এর সাথে শেষের দিকের কোন word এর connection থাকবে না। যেটা solve করবে RNN.

RNN এটা input নিয়ে feature extract করে। মানে Learn করে। যেটা Learn করে সেটা পরের state এ send করে। share করা info + new input নিয়ে new state আবার learn করবে। এভাবে চলতে থাকবে। advantage : each stage এ learn করা info পরের stage এ pass হয়। ফলে কি কতটুকু জানা আছে আগেই সেটার basis এ কাজ করা যায়।

$h_t = f_w(h_{t-1}, x_t)$

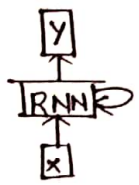
- h_t: new state
- f_w: some functions with parameter w
- h_{t-1}: old state
- x_t: input vector at some time step

Quiz ques: How does the RNN model reduce the number of learnable parameter?

ans: The same function and the same set of parameters are used at every time step. That's how it reduces the number of learnable parameter. $h_t = f_w(h_{t-1}, x_t)$

RNN derivation:

there will be hidden state and output for each input.



$$h_t = f_W(h_{t-1}, x_t)$$
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$ } hidden state

$$y_t = W_{hy}h_t$$ } output

h_t use হয় যেন information remember করব কিনা decide করতে। tanh এর range -1 to 1. So, -1 হলে remember করবনা, +1 হলে করব। y_t = sigmoid function use হয়।
3 টা parameter learn করতে হয়: W_{hh}, W_{xh}, W_{hy}.

প্রত্যেক phase এ error multiplication করতে হয়।
ফলে error 0 এর কাছাকাছি চলে আসে। এটা vanishing gradient descent problem। এই problem solve এর জন্য : LSTM (Long short Term Memory)

LSTM derivation: (most important)

3টা gate : input, forget, output

C_{t-1} = previous state থেকে যা learn করল (মানে আগের সব state থেকে learn করা)

h_{t-1} = ঠিক্‌ঠিক্‌ আগের state থেকে যা learn করল

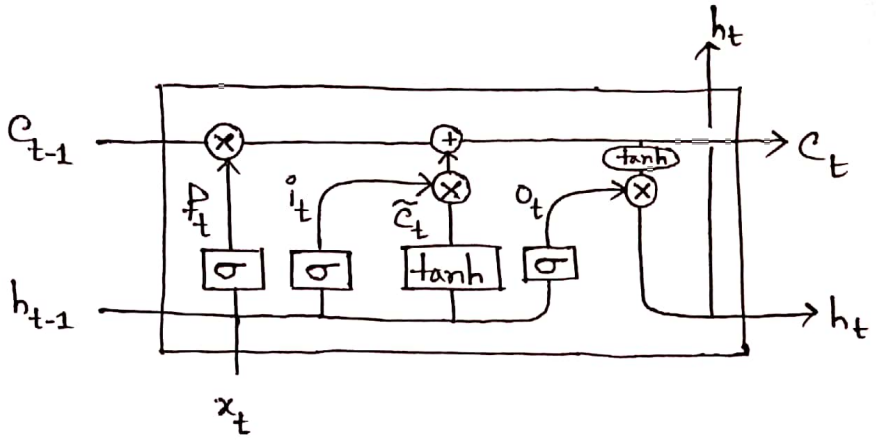
f_t = h_{t-1} আর input (x_t) থেকে যে info টা important আর যেটা forget করতে হবে সেটা denote করে
f_t mainly is how much it have to forget

$i_t \otimes \tilde{C}_t$ = current input এর significance generation using cosine similarity
কতটুকু নতুন learn করব তা ঠিক্‌ করানোর

$\oplus$ = f_t দ্বারা কতটুকু forget করব সেটা আর $i_t \otimes \tilde{C}_t$ দ্বারা কতটুকু learn করব সেটা merge করা হয়

o_t = local output (normal RNN এর মতো)

h_t = o_t আর long term sequence টা ⊗ করে hidden state generate করে + output ও দেয়।



1) $f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$

2) $i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$

3) $\tilde{C}_t = \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right)$

4) $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$

5) $o_t = \sigma\left(W_o [h_{t-1}, x_t] + b_o\right)$

6) $h_t = o_t * \tanh(C_t)$

Quiz ques: 'LSTM model improves long term dependencies by implementing a direct gradient flow line' - Do you agree?

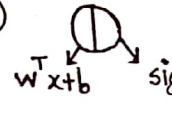
ans: Yes I agree. $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$ দ্বারা long term dependency improve করা হচ্ছে। diagram draw করতে হবে।

Logistic Regression + Vectorizing Logistic Regression

Important topics:

1) disadvantage কি LR এর? কীভাবে solve করা যায়?

Suggestion from sir:

1) 

$w \in R^D$ [R = real number, D = dimension]

এরকম বিস্তারী ques থাকবে but এটার ans অনেক সোজা

Fuzzy Neural Network

Suggestion from sir:

1) 3টা ques থেকে ১টা আসবে

Genetic algorithm

Suggestion from sir:

1) Mutation/Cross over যেকোন ১টা আসবে

Computational graph

Suggestion from sir:

1) Math

* Prototype এর sem final এর solve

More Suggestion

1) Shallow neural network কি? কিভাবে কাজ করে? কেন popular?
2) Loss function RMS কেন বাজে?
3) Gradient দিয়ে convex problem কীভাবে solve করে?
4) CNN এর pooling, filtering কীভাবে কাজ করে?
5) Dropout কীভাবে কাজ করে?
6) Skip gram model টা কীভাবে কাজ করে?