[NTI/VietAI] Zero-to-Hero Program - Entrance Test

2711vedant@gmail.com Switch account

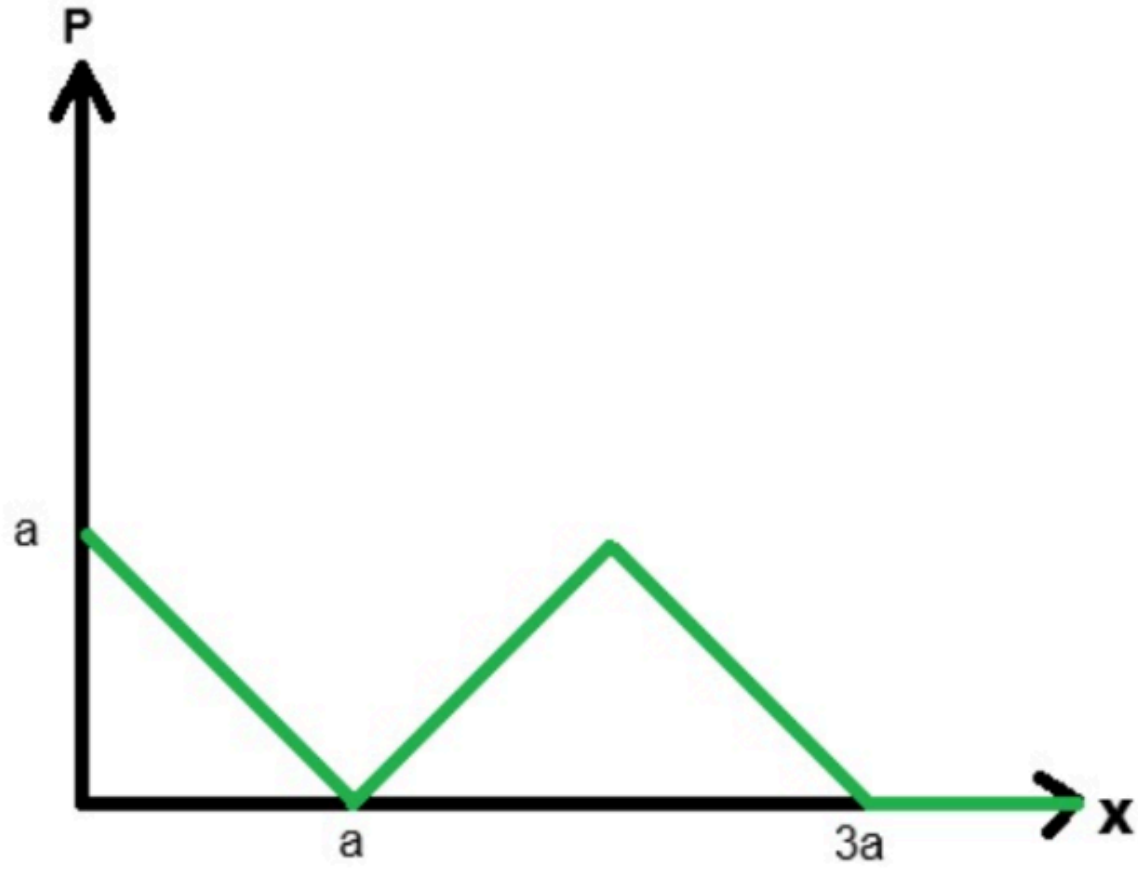
Not shared

Draft saved

A. Mathematics

Problem A.1

Given a random variable X≥0 and its probability distribution shown in the formula below, note that P(X)=0 for X>3a. Find the value of the variable a, and explain your answer.



Area under the probability curve must sum to one. Therefore, 3a^2/2 = 1 , so a = sqrt(2/3) i .e. ~ 0.816

Problem A.2

Find the derivative of the function, then determine the **maximum value** of f’(x).

$$f(x) = \frac{1}{1 + e^{-x}}$$

f'(x) = -1 / ((1 + e^(-x))^2) * -1 * e^(-x) which is f'(x) = e^(-x)/ ((1 + e^(-x))^2). Max value of f'(x) can be found out by f''(x) = 0. Consider y = f(x), then f'(x) = y * (1-y), now to find max value of f'(x) we set d/dy of (y - y^2) to 0 using which we get y = 0.5, therefore the max value of f'(x) is 0.5(1 - 0.5) which is 0.25.

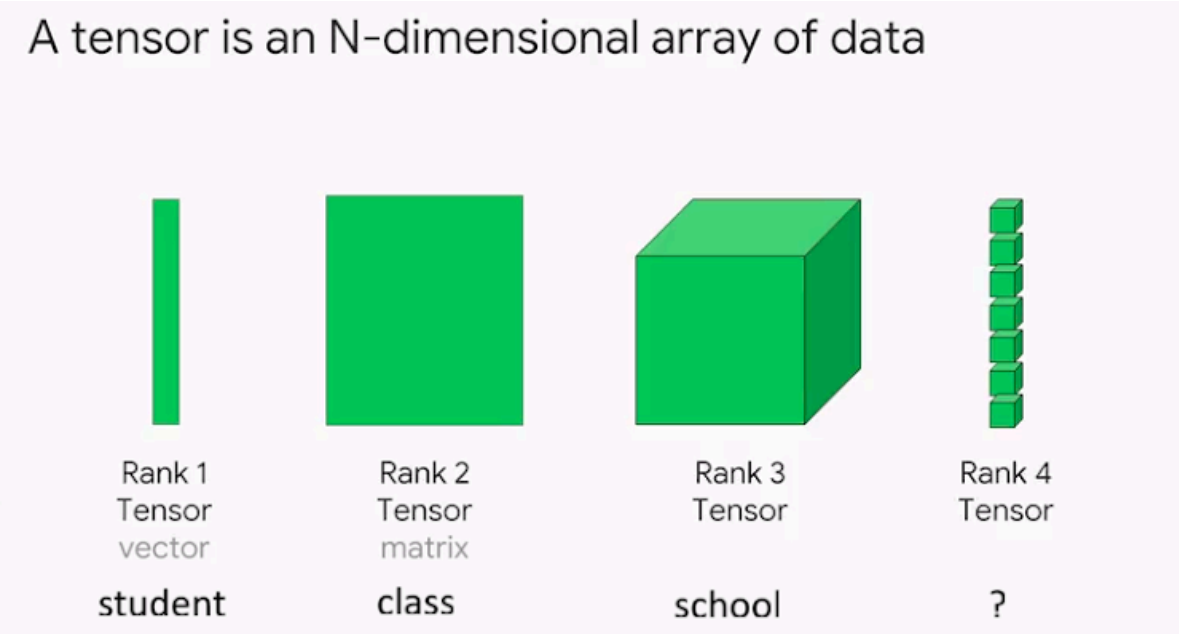
Problem A.3

Three players each roll a fair six-sided die (faces numbered 1–6) in every round. After each round, the smallest number among the three rolls is recorded. After 10,000 rounds, approximate the total of all recorded smallest values? Please explain your answer.

Need to calculate the expected smallest value on rolling all three dice at once in one try.
For a single roll, we can say that for k from 0 to 6 , P(X > k) = (6 - k)/6
probability of smallest value being exactly 1 in three independent rolls (P(S = 1)) = P(X > 0) - P(X> 1) = ((6-0)/6)^3 - ((6-1)/6)^3 = 91/216
similarly, get P(S = 2) = ((6-1)/6)^3 - ((6-2)/6)^3 = 61/216
P(S = 3) = ((6-2)/6)^3 - ((6-3)/6)^3 = 37/216
P(S = 4) = ((6-3)/6)^3 - ((6-4)/6)^3 = 19/216
P(S = 5) = ((6-4)/6)^3 - ((6-5)/6)^3 = 7/216
P(S = 6) = ((6-5)/6)^3 - ((6-6)/6)^3 = 1/216
E(X) = sum(each possible smallest value * probability of that smallest value) = 1 *(91/216) + 2 (61/216) + 3(37/216) + 4(19/216) + 5(7/216) + 6(1/216) = 441/216 ~ 2.04166
In 10000 rounds, the expected smallest rolled value would be 10000* 2.04166 = 20416.6 ~20417.

Problem A.4

A rank-4 tensor can be used to represent which subject?



system of school in a district/city

B. Machine Learning

Problem B.1

You are given two datasets:

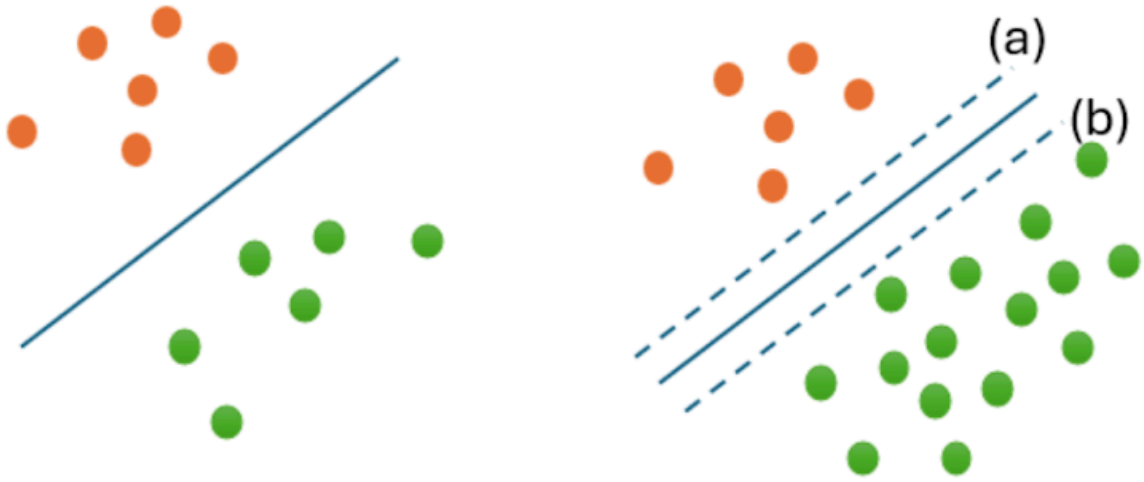
* Balanced dataset: Equal number of red and green points
* Imbalanced dataset: Unequal number of red and green points

A logistic regression classifier is used to separate red and green points. When training the classifier on the **imbalanced dataset**, how does the decision boundary behave compared to training on the **balanced dataset**?

A) Stay the same (unchanged)

B) Shift towards red points (a)

C) Shift towards greenpoints (b)



B) Shift towards red points (a)

In the objective function for logistic regression, we would get many green point error terms and lesser red point error terms. In order to minimise the total error, the model would try more to classify the green points correctly than red points. So the decision boundary would move further away from green points in w's direction, towards red points.

Problem B.2

You need to estimate the price of your house in preparation for a transaction. You conduct a survey of several nearby houses but are only able to collect data on about 10 of them, as this information is sensitive and not widely shared. The data you gather is also limited, including only the number of floors, area, and transaction price. Which of the following models would be suitable for this scenario?

- [ ] Naive Bayesian Regressor
- [ ] K-Nearest Neighbors Regressor
- [ ] Neural Network with more than 2 layers and non-linear activation function
- [x] Linear Regression
- [ ] Logistic Regression

Problem B.3

In regression tasks, we often use Mean Squared Error (MSE) or Mean Absolute Error (MAE) as the loss function. Why don't we use a linear loss instead, as shown below?

$$L(y, \tilde{y}) = (y - \tilde{y})$$

In the above loss function, there is no absolute function involved. suppose there are two predictions for y =5 , one is 1 and one is 9. In this case the total error would be (5-1) + (5-9) i.e. 0 .Even though both predicitons are wrong, the loss is zero, and model cannot learn well, therefore we cannot use this loss function for model training.

Problem B.4

A compressed CSV from a nationwide grocery chain contains ≈ 1.7 million completed checkouts.

- Receipt_uid - uuid
- Seq_no - increment by 1 for each new transaction
- Number of items_scanned: heavy right-tail
- Use promotion: boolean
- Loyalty_tier | Bronze / Silver / Gold

- Total_spend_usd: target variable - float - (USD)

Given the nature of the heavy right-tail data, how would you do train / test / split?

since one of the features, number of items scanned has heavy right tailed distribution i.e. most values are on the lower end, but there are still some outliers on the far right, we would do a train test split that preserves the overall class frequency in the train/test split like a stratified split.

C. Deep learning

Problem C.1

When fine-tuning a pretrained model, how can you prevent overfitting? List at least two methods you can use and explain.

- [x] Transform data with multiple perspectives
- [ ] Add skip connection
- [ ] Change the activation from sigmoid to ReLU / LeakyReLU
- [ ] Random select a minibatch to train
- [x] Random select a subset of neural to train

Problem C.2

Please explain in detail, why the Residual Module helps deep learning models converge better than original feature extraction function:

a) The original feature extraction: *F(X) = feat(X)*

b) The feature extraction with residual module: *F(X) = X + feat(X)*
where *X* is the input feature, *F(X)* is the output feature, ***feat*** is a function to extract features like Convolution / Self-Attention / Multi-layer Perceptron.

By using residual modules, we can be assured that the
1. model is not learning information from any distortions in input after passing through multiple layers and
2. with the inclusion of X ensures that gradients calculated in deep networks do not get smaller and not interfere with the model learning properly.

Problem C.3

After reading the original Transformer paper (*Attention is all you need*), you may find the positional embedding formula too complicated. Why don't we use the following simpler embeddings instead?

$$p_i = \begin{bmatrix} i \\ i \\ \vdots \\ i \\ i \end{bmatrix} \text{ where } p_i \text{ has the same dimension as } x_i, \text{ the embedding of token } i$$

p_i in this case has only one information for a token which is i, its index position, which is not enough information to learn semantic information about the token. Even if the same token repeated after a while, its embedding would be different, which is not desirable.
Also, it is static in the context of current collection of tokens ,meaning it wont be able to generalise to new unseen tokens.

Problem C.4

```
import torch, torch.nn as nn, torch.optim as optim

class LReLU(nn.Module):
    def forward(self, x):
        return x

def build_net(input_dim=20, hidden=64, output_dim=1):
    return nn.Sequential(
        nn.Linear(input_dim, hidden),
        LReLU(),
        nn.Linear(hidden, hidden),
        LReLU(),
        nn.Linear(hidden, output_dim)
)

net = build_net()
loss_fn = nn.MSELoss()
opt = optim.Adam(net.parameters(), lr=1e-3)

# toy training loop (optional)
for _ in range(100):
    xb, yb = torch.randn(128, 20), torch.randn(128, 1)
    opt.zero_grad(); loss_fn(net(xb), yb).backward(); opt.step()
```

We trained the three-layer network shown above on a synthetic dataset where the target is

y = Σ (x_i**3 + 3x_i**2 + 3) (sum taken over every input feature x_i)
Can the deep learning model above overfit (training error close to zero) on such data?
What training-vs-validation pattern (under-fit, good fit, or over-fit) would you predict, and why?

The activation function involved is linear, therefore no matter how many times we combine it, we won't be able to reasonably approximate the cubic target function....
The model would not overfit on such data, and the overall training-validation pattern would expected to be a under-fit.

D. Natural Language Processing

Problem D.1

You are required to build a sentiment classification system for a sensitive domain (e.g., finance, banking). The customer has provided a training set (~100,000 samples), and the test dataset has been preprocessed into an encoded word format (e.g., integer or token IDs instead of raw text). As a machine learning engineer, which of the following models is best suited for this scenario?

- [ ] Naive Bayesian Classifier
- [ ] Training RNN with Word2Vec embedding
- [x] Fine-tuning BERT for Text Classification
- [ ] Training RNN with LSTM cells and Word2Vec embedding

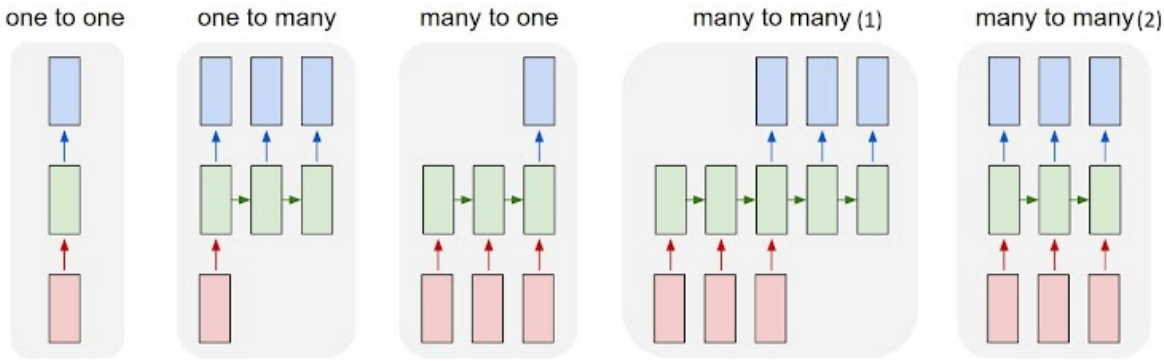
Problem D.2

Which is a type of word embedding?

- [x] Word2Vec
- [x] GloVe
- [ ] TF-IDF
- [ ] One-hot encoding
- [x] BERT

Problem D.3

Which variant of RNN is suitable for text summarization task in NLP?



- [ ] one-to-one
- [ ] one-to-many
- [ ] many-to-one
- [ ] many-to-many (1)
- [x] many-to-many (2)

Problem D.4

When fine-tuning a pretrained language model (like BERT or GPT) for a specific downstream NLP task (e.g., sentiment analysis or question answering), which of the following challenges may arise?

- [x] Catastrophic forgetting – the model may lose its general language understanding during task-specific training
- [x] Overfitting – especially when the downstream dataset is small
- [x] Domain mismatch – when the downstream task differs significantly from the pretraining domain
- [ ] Need for full retraining from scratch.

Submit Clear form

Never submit passwords through Google Forms.