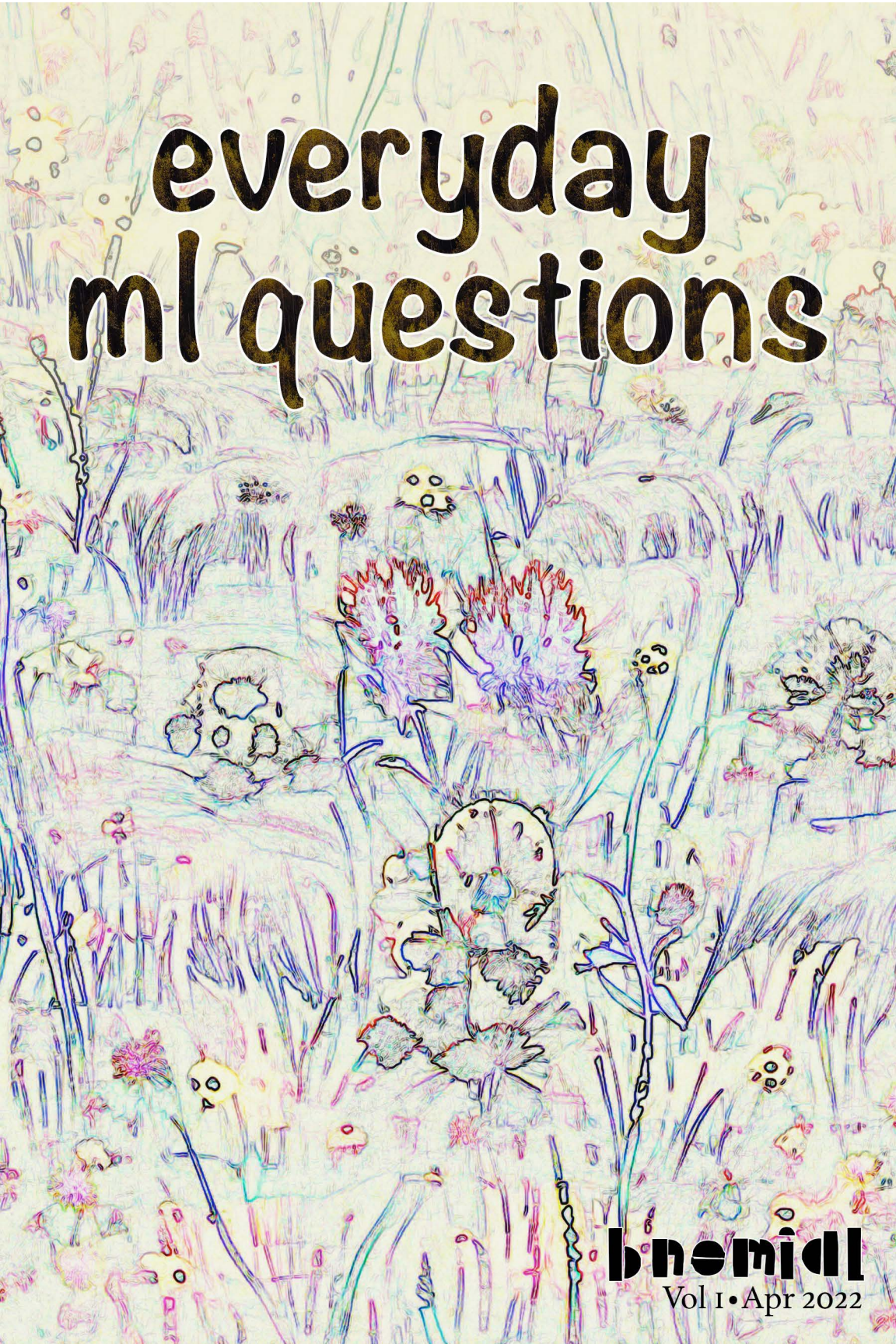
everyday ml questions



bnemidl
Vol 1 • Apr 2022

Questions and explanations: Santiago Valdarrama and Vladimir Haltakov
Cover design: VQGAN+CLIP. “Flowers on a meadow”
Book design: Yasmin Lemus

First published: May 2022

www.bnomial.com

Contents

Answers and explanations

Preface

This book was inevitable.

I started toying with the idea of multi-choice questions back at the start of the century. I wanted my undergraduate thesis to center around a mechanism to better evaluate these questions, but I decided on something simpler after a while.

Over the years, I brought back the idea in different flavors but never got past writing a few questions and thinking about a way to publish them.

It wasn't until late 2021 that I decided to make it happen. I reused a domain I had bought with a different purpose and recruited Vladimir to help me put this out there.

That's how Bnomial was born. A site that publishes one machine learning question every day. Technically simple, but the perfect avenue for us to share the content.

And one thing led to another. This book is a compilation of every question we published in April. We reviewed them, extended the explanations, and worked hard to make each question as informative as possible.

It was hard work but here we are.

How to answer the questions

This book aims to teach you something new, one question at a time.

Getting a question right or wrong only serves as feedback about where you should make emphasis, but the score is not important; what you take away from each question is.

Here is a good approach to answering the questions in this book:

After you read the question, pause for a second and write down a few critical ideas that you think will answer the question. This will help as you go through each possible choice.

Starting from the top and using the above list, determine whether each choice seems a plausible answer to the question. Remember that every question may have more than one correct answer.

Eventually, you'll find a choice you don't understand or can't outright discard. Don't guess! Instead, do some research focusing on that particular choice.

People look at me in funny ways whenever I recommend they research before answering the question. Years in the school system have trained our brains to think we are cheating if we do that. Well, you aren't. Remember, your score doesn't matter at all.

After you answer, find the solution and go through the explanation. Even if you get it right, try

and read through the notes we collected. I've found that many times—although I know the answer to something—the reasons surprise me, so those extra minutes of reading could help.

I like to finish by writing down the most important lesson I'm taking away from the question. I find it helpful when revising the material or as a guide for further research.

Finally, make sure to have fun! Our goal is to give you a different way to learn what otherwise could be a dry subject, so take it slow, show up every day, and you'll come out in much better shape than you started.

There are plenty more questions!

This book is a compendium of the 30 questions we published online from April 1 to April 30, 2022. But that's just the beginning!

Every day we publish a new question. You can go online and answer them right as they go live!

We are planning to keep collecting and writing new books. This one is just the first.

Scan the code below and you'll go straight to today's question!

QUESTIONS

April 1, 2022

Hot dog or not hot dog

A popular TV show made fun of a pretty useless startup idea: an application to take pictures and determine whether they showed a hot dog or not.

Useless—I know—but let's go with it for a second and imagine we are in charge of building it.

We have plenty of emails to train the model, so we will use deep learning to build a binary classifier. The only question remaining is how to structure the final layer of the network.

Which of the following activation functions could be a good candidate for the output layer?

1. Rectifier Linear Unit (ReLU)
2. Leaky ReLU
3. Sigmoid
4. Softmax

Difficulty: **Hard**
Answer and explanation: page 43

April 2, 2022

Daniella is looking at ReLU

Daniella knows that she can improve her deep learning model by replacing the activation function of some of the layers with Rectified Linear Units (ReLU).

Unfortunately, she knows there's going to be pushback from the team. There always is!

Her best path forward is to prepare a list of some of the advantages of ReLU to try and convince everyone to make the change.

Daniella prepares an initial draft, but she would like some help reviewing it right before sending it.

Which of the following are advantages of using the ReLU?

1. ReLU saturates around extreme values, allowing backpropagation to converge faster and helping the network learn.

2. ReLU provides better representational sparsity than the sigmoid and tanh activation functions because it can produce an actual zero output.

3. ReLU is computationally straightforward to implement.

4. Neural networks are easier to optimize when their behavior is as linear as possible. ReLU acts mainly as a linear function, improving our ability to optimize neural networks.

Difficulty: **Hard**
Answer and explanation: page 45

April 3, 2022

A few regression models to start

Regression analysis is a Supervised Learning technique that investigates the relationship between a continuous target variable based on the value of one or multiple predictor variables.

We have access to many regression models with slightly different characteristics but the same overall purpose in machine learning.

The team knew that.

After studying the problem before them for a few days, they knew they needed to select a few regression models to start experimenting.

Which of the following are regression models that the team could consider?

1. Linear Regression
2. Polynomial Regression
3. Ridge Regression
4. Lasso Regression

Difficulty: **Hard**
Answer and explanation: page 48

April 4, 2022

Odd one out

Here is a simple exercise.

The list below contains four different functions. Almost every one of them is related to the rest somehow.

Your goal is to determine which one doesn't belong on this list.

Can you select the odd one out?

1. ReLU
2. Sigmoid
3. Dropout
4. Softmax

Difficulty: **Easy**
Answer and explanation: page 49

April 5, 2022

Eliana knew the answer immediately

Eliana showed up early to the phone call.

Her agency has been advising a client who started building a deep learning model. Unfortunately, they've been stuck for a while, and it's Eliana's job to get them back on track.

Eliana realized what was happening ten minutes into the call.

When training their network and during the backpropagation process, the gradients quickly decrease until they approach zero, leaving the weights of the lower layers essentially unchanged.

Eliana knows this is the vanishing gradient problem. All that's left is to speculate about why this was happening.

Which of the following could be causing the model to suffer from the vanishing gradient problem?

1. The hidden layers of the model use the sigmoid activation function.

2. The hidden layers of the model use the ReLU activation function.

3. The hidden layers of the model use the tanh activation function.

4. The model uses batch normalization.

Difficulty: **Hard**
Answer and explanation: page 50

April 6, 2022

The ideas behind Deep Learning aren't new

Over the last few years, deep learning has become very popular.

But you know what's wild? Many of the core ideas we use today in deep learning were already around in the '90s!

We had the raw materials back then, but the conditions weren't adequate for it to explode.

Which of the following do you think likely caused deep learning to gain so much popularity?

1. The gaming industry's massive investment in developing fast, parallel chips.

2. The amount of positive press at the beginning of the latest AI Winter brought new investment into the field.

3. The proliferation of the Internet, allowing the collection and distribution of massive amounts of data.

4. The development and improvement of algorithms, making possible the training of deep networks.

Difficulty: **Medium**
Answer and explanation: page 52

April 7, 2022

Histogram head and tails

We are trying to predict the price of a car.

Our company has a website where users sell and buy used cars. Deciding how much we should charge for a used car is complicated and error-prone.

We want to build an automatic price recommendation system.

The Manufacturer's Suggested Retail Price (MSRP) of the car is our target variable. To check its distribution, we can plot its histogram.

Which of the following statements are true about the distribution plot of our target variable?

1. The head of the distribution is a range where many of the values are concentrated.

2. The head of the distribution is the area where values fall right at the center of the histogram.

3. The tail of the distribution is the part of the histogram that tapers off on one side.

4. A long tail is an area where many values are spread far from the head.

Difficulty: **Medium**
Answer and explanation: page 54

April 8, 2022

We need more capacity

As the business grew, Amelia saw their primary dataset balloon in size.

The neural network model they had running in production was starting to show severe issues. Amelia knew they had to make a change.

The issue was straightforward: the company has been collecting more features from a more varied customer base. Their current model was barely using a sliver of the available information, and it was clearly underfitting.

Amelia knew they had to increase the capacity of the model.

Which of the following are steps that Amelia can take to increase the capacity of her neural network model?

1. Amelia should increase the learning rate to train the model.

2. Amelia should increase the number of hidden layers the neural network uses.

3. Amelia should increase the regularization she is applying to the model.

4. Amelia should increase the batch size to train the model.

Difficulty: **Medium**
Answer and explanation: page 55

April 9, 2022

Helping Madeline remember

Madeline is trying to complete her class assignment.

She just joined a machine learning program, and over the first few weeks, they have been talking about structured data and the different feature types we usually encounter.

Her task seems very simple: remove every ordinal feature from a dataset.

There's only one problem: Madeline doesn't remember what ordinal features are.

Which of the following definitions correctly summarizes what an ordinal feature is?

1. An ordinal feature is a categorical variable with ten or more possible values.

2. An ordinal feature is a categorical variable with fewer than ten possible values.

3. An ordinal feature is a categorical variable with a meaningful order.

4. Any feature used in a machine learning model is an ordinal feature.

Difficulty: **Easy**
Answer and explanation: page 57

April 10, 2022

Peter needs a better balance

Peter is having a bad day.

He is working on a deep learning model that uses MRI images to classify a rare disease. Unfortunately, Peter has many more pictures of healthy people than people having the disease, and his model classifies every sample as healthy.

Collecting more data for such a rare disease is very difficult, and strict privacy regulations make the whole process even more problematic.

Peter needs another solution.

Select every option that Peter can pursue to improve the model.

1. Give higher weight to the samples showing the disease.

2. Replace the deep learning model with a Decision Tree because the latter is more robust to imbalanced data.

3. Augment the dataset with slightly modified copies of the images showing the disease.

4. Reduce the learning rate to allow the model to learn underrepresented samples.`

Difficulty: **Medium**
Answer and explanation: page 59

April 11, 2022

I'm frustrated with my training loss

I'm sure you've been here before.

I've been working on this neural network for a while. I'm using Mini-Batch Gradient Descent, and the results show the training loss oscillating up and down.

It is so frustrating!

I've been expecting it to go down consistently as my model learns, but I haven't had any luck so far.

Honestly, I don't have too much time left to deal with this, so I'll try one or two more things before I give up.

Which of the following do you think I should try next? Select everything that applies.

1. This is clearly a problem with the data. I should go back to the dataset and ensure it's appropriately balanced.

2. I should increase the learning rate to take larger steps in the direction of the gradient.

3. I should decrease the learning rate to avoid missing the local minima.

4. I should increase the batch size to increase the variability of samples on every batch.

Difficulty: **Medium**
Answer and explanation: page 61

April 12, 2022

Ted wants to sell more cars

Ted is just starting his first job.

He is working at a major car dealership. Hundreds of cars go out the door every month, and Ted plans to help the company understand its customers a little bit more.

After a week of obsessing over all the information they capture about every customer, Ted starts thinking about a good way to slice them into segments based on their characteristics.

The problem is that Ted is unsure about the best approach. Should he segment customers based on their purchase history, or should it be better to do it by age? What about the type of vehicle or purchase power?

How would you approach this problem if you were in Ted's shoes?

1. Use an unsupervised learning algorithm to produce potentially interesting ways to segment the customers.

2. Define beforehand a few segments, and train a supervised learning algorithm to classify every customer into one of them.

3. Use a supervised learning algorithm to produce potentially interesting ways to segment the customers.

4. Use a semi-supervised learning algorithm to process the data, thus taking advantage of both supervised and unsupervised techniques.

Difficulty: **Medium**
Answer and explanation: page 63

April 13, 2022

Handling missing values

The customer data the team received was not in great shape.

After some analysis, they found that many values were missing, entire fields were incomplete, and some of the data was corrupted.

Everyone knew they had to fix this before building the machine learning application they had been planning for a year.

The team lead called a special meeting to discuss their next steps, and after some time, they narrowed it down to a few possibilities.

Which of the following are valid techniques they could use to handle the problems with their data?

1. Replace missing values with the mean, median, mode, or other imputation techniques.

2. Drop any rows or columns that contain missing or corrupted data.

3. Predict the missing values using a separate machine learning model.

4. Use a machine learning algorithm that is robust to missing values.

Difficulty: **Hard**
Answer and explanation: page 64

April 14, 2022

The mysterious case of the strange loss

Jena has finally finished building a neural network!

It took her some time to deal with the dataset, but she is finally ready to train the model.

Fifteen minutes later, Jena notices that her training loss is not decreasing as expected; it stays much higher than she hoped for.

Jena got back to the drawing board to investigate what could be happening.

After a couple of hours, Jena has come up with a few potential reasons that could explain the problem.

Select every potential reason that could be causing the loss to behave this way.

1. The regularization that Jena is using is too aggressive.

2. Jena is using a learning rate that's too low.

3. The neural network is getting stuck at local minima.

4. Jena is using a learning rate that's too high.

Difficulty: **Hard**
Answer and explanation: page 66

April 15, 2022

Looking behind François's Tweet

Recently, François Chollet mentioned that you should never try to use a learning-rate schedule from an old dataset to train a new one.

Imagine a scenario where you create a deep learning classification model to train on images from a security camera. During the exploration phase, you set up a specific training schedule that works well for your problem and gives you good performance.

When training a new version of the model—even when using the same architecture—with images from a different security camera, François' advice is to avoid using the same training schedule.

Which of the following could be François' thinking behind his advice?

1. The learning-rate schedule depends on the number of samples in the dataset. Therefore the exact schedule won't work with a different size dataset.

2. The learning-rate schedule should change whenever we have a dataset from a different target distribution. This won't be the case if the target distribution is the same.

3. A new dataset changes the tradeoff between optimization and regularization. The learning-rate schedule is dataset-specific.

4. Learning-rate schedules are specific to the optimization mechanisms used by the model.

A new dataset usually requires that we change the optimization process, which will invalidate the learning-rate schedule.

Difficulty: **Hard**
Answer and explanation: page 68

April 16, 2022

Patricia is building a logistic classifier

Patricia has been working on a logistic classifier for a new project.

She knows that the key to getting successful predictions depends on the error function she decides to use. She wants this function to have the following characteristics:

- The function should return a small number if the sample is correctly classified.
- The function should return a large number if the sample is incorrectly classified.
- The error for a set of samples should be the sum or average of the errors for all the samples.

Patricia has several choices but would like to hear your opinion.

Which of the following would be the best error function for a logistic classifier?

1. Absolute error: a function that returns the absolute value of the difference between the prediction and the label.

2. Square error: a function that returns the square of the difference between the prediction and the label.

3. Log loss: a function that returns the negative logarithm of the product of probabilities.

4. Mean percentage: a function that returns the average error of the differences between predicted and actual values.

Difficulty: **Hard**
Answer and explanation: page 70

April 17, 2022

The birthday paradox

After a lot of begging, Lucia convinced Anna to go to a party together.

Anna likes to spend most of her time with math books. The change of scenery was good, so she showed up.

After an hour or so, Lucia noticed that Anna was not having fun, so she decided to cheer her up with an interesting problem.

"Anna, there are 23 people at this party right now. What is the probability that two of them share the same birthday?"

What is the correct answer to Lucia's question?

1. There's about a 1% probability of two people sharing the same birthday.

2. There's about a 6% probability of two people sharing the same birthday.

3. There's about a 17% probability of two people sharing the same birthday.

4. There's about a 50% probability of two people sharing the same birthday.

Difficulty: **Medium**
Answer and explanation: page 71

April 18, 2022

Keeping pedestrians safe

A team has been working on a self-driving car model to detect pedestrians crossing the street on images captured from the car cameras.

Their model will help cars navigate busy areas while keeping everyone around safe. It's a critical and delicate piece to get to full autonomy.

Despite initial promising results, they haven't settled on the best way to evaluate the model's performance.

A discussion starts with some initial ideas.

Which of the following evaluation metrics would be the best for this problem?

1. The team should use the recall of the model, as defined by the percentage of detected pedestrians with respect to every image containing a pedestrian.

2. The team should use the precision of the model, as defined by the percentage of legitimate detected pedestrians with respect to every detected pedestrian.

3. The team should use the Fβ-Score of the model with a high value of β.

4. The team should use the Fβ-Score of the model with a low value of β.

Difficulty: **Hard**
Answer and explanation: page 73

April 19, 2022

Thinking about softmax

Remember the last time you built a machine learning multi-class classification model?

You probably used a softmax activation on the output layer of your network. It's a widespread practice, so there's a good chance you've done it.

It's usually fun to think about why we do things the way we do them, so let's get into it.

Which of the following statements is true about the softmax activation function when used in the output layer of a neural network?

1. The softmax function turns the network's input into a vector of probabilities that sum to 1.

2. The softmax function is a probabilistic or "smooth" version of the function that returns the index of the vector's largest value.

3. The softmax function turns a vector of real values into a sorted vector of probabilities that sum to 1.

4. The softmax function is a probabilistic or "smooth" version of the function that returns the vector's maximum value.

Difficulty: **Hard**
Answer and explanation: page 75

April 20, 2022

Alice can't remember how One-Hot Encoding works

When building a machine learning model, much of the work happens well before starting training.

Alice knows that she needs to prepare the data before it's ready. She has been looking into encoding some of the features on her dataset. One-Hot Encoding seems like a good candidate.

It's been quite a while since Alice used One-Hot Encoding, and she can use some help.

Which of the following statements explains how One-Hot Encoding works?

1. One-Hot Encoding encodes a numerical feature into its categorical representation.

2. One-Hot Encoding creates additional features based on the number of unique values in a categorical feature.

3. One-Hot Encoding encodes a string-encoded feature into its numerical representation.

4. One-Hot Encoding encodes a string-encoded feature into its categorical representation.

Difficulty: **Medium**
Answer and explanation: page 77

April 21, 2022

The anatomy of ReLU

The Rectified Linear Activation Function, or ReLU, returns its input if it's positive or a zero otherwise.

In other words: ReLU turns any negative values into zero and leaves everything else unmodified.

ReLU has become a popular activation function when training neural networks. Since it's a piecewise linear function and computationally straightforward to implement, models become easier to optimize and achieve better performance.

There's something very interesting about combining Gradient Descent with ReLU, and to get there, we have to think a bit about the mathematical properties of ReLU.

Which of the following is true about the Rectified Linear Activation Function (ReLU)?

1. The function is neither continuous nor differentiable.
2. The function is differentiable but not continuous.
3. The function is both continuous and differentiable.
4. The function is continuous but not differentiable.

Difficulty: **Medium**
Answer and explanation: page 79

April 22, 2022

Australian birds

Marc was excited about the performance of his model.

He's been working on an app to classify photos of birds, and his model is performing very well on all metrics.

Marc knows what he is doing and followed all the best practices: he split training and test data, used the proper evaluations metrics, balanced his dataset, and reviewed examples regularly to ensure there were no labeling errors.

Finally, Marc launched his app and positive feedback started rolling in except from one place: users in Australia complained the performance was awful.

Marc was baffled.

What is the most likely reason for the problem?

1. Marc didn't train the model long enough to capture all the necessary details of different bird species.

2. Marc's model is too simple, and it couldn't learn the entire dataset of birds, leaving out those from Australia.

3. Marc's model suffers from sampling bias. He probably didn't include enough examples of Australian birds.

4. Marc's model is suffering from data or concept drift.

Difficulty: **Easy**
Answer and explanation: page 81

April 23, 2022

Breaking a function down into pieces

We are trying to predict the price of a car.

Our company has a website where users sell and buy used cars. Deciding how much we should charge for a used car is complicated and error-prone.

We want to build an automatic price recommendation system using supervised learning.

The Manufacturer's Suggested Retail Price (MSRP) of the car is our target variable. Since its distribution has a very long tail, we apply a log transformation before training the model.

If our model for a single observation is y = f(x), what are x and y for this project?

1. x is a feature vector containing the variables that describe the car, and y is the logarithm of the price of the car.

2. x is a feature vector containing the variables that describe the car, and y is the price of the car.

3. y is a feature vector containing the variables that describe the car, and x is the logarithm of the price of the car.

4. y is a feature vector containing the variables that describe the car, and x is the price of the car.

Difficulty: **Medium**
Answer and explanation: page 83

April 24, 2022

Mia and the feedback loop

Mia was crushing her thesis!

She was about to release a new neural network architecture that promised to raise the bar on image classification problems.

Mia did not start from scratch. She modified an existing model but added a key ingredient: feedback loops.

A feedback loop is when connections between units form a directed cycle, thus creating loops in the network. This gave Mia's network the ability to save information in the hidden layers.

Mia did a lot of research before deciding in favor of this architecture. She knew the advantages of her decision.

Which was the architecture that Mia studied to learn about feedback loops?

1. Recurrent Neural Networks
2. Convolutional Neural Network
3. Multilayer Perceptron
4. Radial Basis Function Network

Difficulty: **Easy**
Answer and explanation: page 85

April 25, 2022

Harper and the small gradients

Harper's team is struggling with the deep neural network they have been building.

Unfortunately, during backpropagation, the gradient values of their network decrease dramatically as the process gets closer to the initial layers, preventing them from learning at the same pace as the last set of layers.

Harper knows their model suffers from the vanishing gradient problem. She decides to research every possible option to improve their model.

Which of the following techniques will make Harper's model more robust to the vanishing gradient problem?

1. Harper should try ReLU as the activation function since it's well-known for mitigating the vanishing gradient problem.

2. Harper should modify the model architecture to introduce Batch Normalization.

3. Harper should make sure they are initializing the weights properly. For example, using He initialization should help with the vanishing gradient problem.

4. Harper should increase the learning rate to avoid getting stuck in local minima and thus reduce the chance of suffering vanishing gradients.

Difficulty: **Hard**
Answer and explanation: page 86

April 26, 2022

Exploring data before anything else

An essential step in any machine learning project is the Exploratory Data Analysis process.

Before we can train a model, we need to understand our data. As the name suggests, Exploratory Data Analysis allows us to explore the data to discover potential problems or patterns that we can use.

Which of the following are some of the steps we take during this process?

1. Learn the distribution of the target variable.

2. Understand the features in the dataset and the distribution of their values.

3. Evaluate the performance of our models on this data.

4. Assess the data quality, including missing or corrupt values.

Difficulty: **Easy**
Answer and explanation: page 88

April 27, 2022

Susan needs to make a decision

The deadline is approaching, and Susan still hasn't decided which version of her classification model to deploy to production.

She experimented with different hyperparameters, and now she has two models that perform pretty well.

Her problem is that none of these models is better than the other in every situation. One model has a higher recall but worse precision than the other. Susan can improve the precision by playing with different thresholds, but now the recall decreases.

How can Susan decide which is the best overall model?

1. Susan should tune the thresholds until both have a recall of 95% and choose the one with higher precision.

2. Susan should tune the thresholds until both have a precision of 95% and choose the one with a higher recall.

3. Susan should compute the area under the ROC curve for both models and choose the one with the higher value.

4. There's no objective way to decide which model is best. Susan should pick either one of them.

Difficulty: **Medium**
Answer and explanation: page 90

April 28, 2022

Linear regression by hand

The best way to learn something new is to rip the band-aid and tackle a problem from scratch.

Imagine you get a dataset with thousands of samples of houses sold in the U.S. over the last five years. We know the value of a few different features of each home and the price it was sold for. The goal is to build a simple model capable of predicting the price of a new house given those features.

A linear regression model seems like an excellent place to start.

But you are not writing any code yet. You want to do this manually, starting with a matrix X containing the value of the features and a vector w containing the weights.

The next step is to multiply X and w, but you aren't sure about the result of this operation.

Which of the following better describes the result of multiplying X and w?

1. The result will be a vector y containing the actual price of each house as provided in the dataset.

2. The result will be a vector y containing the predicted price of each house.

3. The result will be a matrix y containing the actual price of each house and the features from the matrix X.

4. The result will be a matrix y containing the predicted price of each house and the features from the matrix X.

Difficulty: **Easy**
Answer and explanation: page 92

April 29, 2022

20,000 sunny and cloudy samples

Today is your very first day.

You get access to weather data. Twenty thousand samples with the weather of sunny and cloudy days. You want to build a model to predict whether a future day will be sunny or cloudy.

You already know this is a binary classification problem, and now it's time to pick a model.

Which of the following techniques can you use to build a binary classification model?

1. Logistic Regression
2. k-Nearest Neighbors
3. Neural Networks
4. Decision Trees

Difficulty: **Medium**
Answer and explanation: page 94

April 30, 2022

The true meaning of hyperparameter tuning

Marlene is trying to build an audience.

Writing content seems easy, but taking a complex subject and boiling it down to its essence is not an obvious task.

Marlene wants to start from the basics and write as much as possible about the fundamentals of machine learning.

She picked her first topic: hyperparameter tuning.

If you were trying to summarize the core idea of hyperparameter tuning, which one of the following sentences would you use?

1. Hyperparameter tuning is about choosing the set of optimal features from the data to train a model.

2. Hyperparameter tuning is about choosing the set of optimal samples from the data to train a model.

3. Hyperparameter tuning is about choosing the optimal parameters for a learning algorithm to train a model.

4. Hyperparameter tuning is about choosing the set of hypotheses that better fit the goal of the model.

Difficulty: **Easy**
Answer and explanation: page 95

ANSWERS AND EXPLANATIONS

April 1, 2022

Hot dog or not hot dog

The correct answers to this question are:

- Sigmoid
- Softmax

We are building a binary classifier, which means we want our output layer to produce a result that we can easily interpret to determine the correct answer.

The first two choices are ReLU and Leaky ReLU. ReLU returns its input if positive or a zero otherwise, while Leaky ReLU is based on ReLU but with a slight slope for negative values.

Now imagine our deep learning model did a good job. The previous to last layer of the network captures all of the necessary information we need to make a prediction. How would ReLU or Leaky ReLU on the output layer help?

Let's assume the input to the output layer is a positive value. Both ReLU and Leaky ReLU will leave that value untouched. They will act as a passthrough for any positive values coming from the previous layer and turn any negative values into zero—Leaky ReLU will turn negative values into a tiny positive result. That is not helpful because we won't be able to use the magnitude of the values to differentiate between different inputs.

Sigmoid can squash its input into a range from 0 to 1. That's good! We could set an arbitrary threshold to interpret the result of the model.

We could assume that anything equal to or under 0.5 doesn't show a hot dog, while anything above 0.5 does. Sigmoid lets us turn the output into a "probabilistic" result, making it a very popular option for binary classification models.

Finally, softmax converts a vector of numbers into a vector of probabilities. In our binary classifier, we care about two possible states: do we have a hot dog or not? We could use softmax to output a vector with two values: the first representing the negative case and the second the positive case.

Later, we can get the most significant value outputted by softmax to determine the final prediction. If the first value is larger, we assume there is no hot dog—because that's the value representing the negative case. If the second value is larger, we presume the picture shows a hot dog.

Something interesting to note is that the softmax function is an extension of the sigmoid function to cover cases where we care about more than two classes. Softmax reduces to sigmoid when we use it in a binary classification context, so we should get the same result in both cases.

Therefore, the third and fourth choices are correct answers to this question.

References

- "Logistic function"
- "Softmax function"
- "A Gentle Introduction To Sigmoid Function"
- "The Differences between Sigmoid and Softmax Activation Functions"

April 2, 2022

Daniella is looking at ReLU

The correct answers to this question are:

- ReLU provides better representational sparsity than the sigmoid and tanh activation functions because it can produce an actual zero output.
- ReLU is computationally straightforward to implement.
- Neural networks are easier to optimize when their behavior is as linear as possible. ReLU acts mainly as a linear function, improving our ability to optimize neural networks.

Let's start by remembering that ReLU is a piecewise linear function that outputs the input directly if it is positive or outputs zero otherwise:

$$f(x) = \max(x, 0)$$

The first choice argues that ReLU saturates around extreme values. This is partially true: whenever the input value is negative, ReLU converts it to zero, so any negative values will cause the function to saturate. However, ReLU doesn't saturate for positive input values because it keeps them untouched. In comparison, sigmoid saturates around 0 and 1, and tanh around -1 and 1 for positive and negative inputs.

This reason alone is enough to discard the first choice as a correct answer. However, the choice continues by arguing that the saturation allows backpropagation to converge faster and helps the network learn, which is also not true. Saturation is problematic for neural networks. It causes problems like the "dying ReLU" or the "vanishing gradient problem."

The second choice is correct since ReLU can turn any negative inputs into zero, unlike sigmoid and tanh, which only learn to approximate a zero value. When using ReLU, we can have layers with one or more nodes containing zero values, which we call "sparse representation." This simplifies the model and can significantly save computing resources while representing the data in low-dimensional space.

The third choice is also correct since ReLU is very simple to implement. In comparison, sigmoid and tanh require exponentiation which is more expensive to compute.

Finally, neural networks are easier to optimize when their behavior is linear, and ReLU acts as a linear function for the most part. We call it a "piecewise linear function" because it's linear for half of the input domain and nonlinear for the other half. Here is a quote from *Deep Learning*:

> Because rectified linear units are nearly linear, they preserve many of the properties that make linear models easy to optimize with gradient-based methods. They also preserve many of the properties that make linear models generalize well.

In summary, the last three choices are correct answers to this question.

References

- "A Gentle Introduction to the Rectified Linear Unit (ReLU)"
- "Rectifier (neural networks)"

- "Dying ReLU and Initialization: Theory and Numerical Examples"
- "How to Fix the Vanishing Gradients Problem Using the ReLU"
- *Deep Learning*

April 3, 2022

A few regression models to start

The correct answers to this question are:

- Linear Regression
- Polynomial Regression
- Ridge Regression
- Lasso Regression

Other than knowing that we need to predict a continuous target variable, the problem doesn't give us specific information about what the team is facing.

There are many different regression models, including all four choices in this question. Depending on the problem, one or more of these techniques could have the edge over the rest, but we can't discard any of them based on what we know.

Therefore, the short answer is that every one of these methods is a regression technique the team could use, so they are all valid answers to this question.

References

- "6 Types of Regression Models in Machine Learning You Should Know About"
- "Regression Analysis"
- "7 Regression Techniques you should know!"

April 4, 2022

Odd one out

The correct answer to this question is:

› Dropout

ReLU, sigmoid, and softmax are all activation functions used to train neural networks. They all take in the weighted sum of the inputs from the previous layer and generate an output value for the next layer.

Dropout is a regularization method used to train neural networks. It's not an activation function, and its goal is to ignore units during the training phase of the network. This makes it very different from the other three functions.

Dropout is the correct answer for this question.

References

- "A Gentle Introduction to Dropout for Regularizing Deep Neural Networks"
- "A Gentle Introduction to the Rectified Linear Unit (ReLU)"
- "A Gentle Introduction To Sigmoid Function"
- "Softmax Activation Function with Python"

April 5, 2022

Eliana knew the answer immediately

The correct answers to this question are:

- The hidden layers of the model use the sigmoid activation function.
- The hidden layers of the model use the tanh activation function.

If the gradients of the loss function approach zero, the model will stop learning because the network will stop updating the weights. This phenomenon is known as the Vanishing Gradient Problem, and it's very common when using the sigmoid and tanh activation functions in deep neural networks.

The sigmoid and tanh functions squeeze a large input space into a value between [0..1] and [-1..1], respectively. Therefore, large changes in the input of these functions cause small changes in the output. On top of that, both functions saturate when their input grows extremely large or small. Sigmoid saturates at 0 and 1, tanh saturates at -1 and 1. The derivatives at these extremes are very close to zero.

This is not a problem if we are building a shallow network, but as we add more and more layers using these activation functions, the gradients will eventually become too small, and the network will stop learning.

ReLU, on the other hand, is a way to solve the vanishing gradient problem. ReLU is much less

likely to saturate, and its derivative is 1 for values larger than zero. This means that the second choice is an incorrect answer.

Finally, batch normalization is another way to mitigate the vanishing gradient problem. If we normalize the input to a layer that's using a sigmoid activation function, the values won't reach the edges and will stay instead around the area where the derivative isn't too small. Therefore, the fourth choice is also incorrect.

In summary, the correct answers are the first and third choices.

References

- "Can ReLU Cause Exploding Gradients if Applied to Solve Vanishing Gradients?"
- "How to Fix the Vanishing Gradients Problem Using the ReLU"
- "Vanishing gradient problem"

April 6, 2022

The ideas behind Deep Learning aren't new

The correct answers to this question are:

- The gaming industry's massive investment in developing fast, parallel chips.
- The proliferation of the Internet, allowing the collection and distribution of massive amounts of data.
- The development and improvement of algorithms, making possible the training of deep networks.

In *Deep Learning with Python, Second Edition*, François Chollet writes:

> [...] typical deep learning models used in computer vision or speech recognition require orders of magnitude more computational power than your laptop can deliver. Throughout the 2000s, companies like NVIDIA and AMD invested billions of dollars in developing fast, massively parallel chips (graphical processing units, or GPUs) to power the graphics of increasingly photorealistic video games [...]

This investment massively benefitted the development of deep learning: we found ourselves with as much computing power as we ever needed. This makes the first choice a correct answer to this question.

As with many other things, the Internet has played a fundamental role in popularizing deep learning.

Thanks to it, we have collected and distributed the data we use to power these algorithms. Therefore, the third choice is also correct.

Also, a few important algorithmic improvements allowed the community to start training deep neural networks reliably. Again, from *Deep Learning with Python*, Second Edition:

- Better activation functions for neural layers
- Better weight-initialization schemes
- Better optimization schemes

This makes the fourth choice also correct.

Finally, the second choice argues about "positive press at the beginning of the latest AI Winter" which is incorrect. The AI Winter is a period of reduced funding and interest in artificial intelligence, and "positive press" wasn't part of that equation.

In summary, every choice except the second one is correct.`

References

- *Deep Learning with Python*, Second Edition
- "The unreasonable effectiveness of deep learning in artificial intelligence"
- "Deep Learning"

April 7, 2022

Histogram head and tails

The correct answers to this question are:

- The head of the distribution is a range where many of the values are concentrated.
- The tail of the distribution is the part of the histogram that tapers off on one side.
- A long tail is an area where many values are spread far from the head.

Looking at a picture of a histogram definitely helps.

The head of the distribution is the area where most of the values are concentrated. If the histogram is skewed in one direction, this area will not necessarily fall right at the center of the histogram. This makes the first choice correct and the second choice incorrect.

The distribution's tails are the areas of the graph that start tapering off. A long-tail happens when many values are spread far from the head of the distribution. Here is an example of a long tail.

Therefore, the only incorrect choice for this question is the second one..

References

- "A Complete Guide to Histograms"
- "Histogram"
- "Histogram"

April 8, 2022

We need more capacity

The correct answer to this question is:

› Amelia should increase the number of hidden layers the neural network uses.

Here is a helpful excerpt from "How to Control Neural Network Model Capacity With Nodes and Layers":

> The capacity of a deep learning neural network model controls the scope of the types of mapping functions that it is able to learn. [...] The capacity of a neural network model is defined by configuring the number of nodes and the number of layers.

There are three ways that Amelia can use to increase the capacity of her network:

- She can add more hidden layers.
- She can increase the number of nodes on each layer.
- She can go with a combination of these two strategies.

Increasing the network's capacity will give the model more generalization power, so Amelia should see an improved performance as the new model accommodates more data. Therefore, the second choice is the correct answer to this question.

The learning rate, regularization methods, and batch size do not influence the network's capacity, so none of the these choices are correct.

References

- "How to Control Neural Network Model Capacity With Nodes and Layers"
- "The capacity of feedforward neural networks"

April 9, 2022

Helping Madeline remember

The correct answer to this question is:

› An ordinal feature is a categorical variable with a meaningful order.

Here is Jason Brownlee in his "Ordinal and One-Hot Encodings for Categorical Data" article:

> [An ordinal variable is a] variable that comprises a finite set of discrete values with a ranked ordering between values.

Having a meaningful order between the values of an ordinal feature is very important. For example, you could have a feature representing a person's economic status with three possible values: "low," "medium," and "high." Notice how there's a clear order among these values: "low" comes first, then "medium," and finally "high." Compare this with a nominal variable, a categorical feature that doesn't have a meaningful order between values. For example, the values of a feature "color" will not have any discernible order among them.

The number of possible values does not limit an ordinal feature, so the first two choices aren't correct answers. The fourth choice is also not correct since machine learning models could use features that aren't ordinal—for example, there are numerical and nominal features.

In summary, only the third choice is correct.

References

- "Ordinal and One-Hot Encodings for Categorical Data"
- "Here's All you Need to Know About Encoding Categorical Data"
- "Ordinal data"

April 10, 2022

Peter needs a better balance

The correct answers to this question are:

- Give higher weight to the samples showing the disease.
- Augment the dataset with slightly modified copies of the images showing the disease.

Dealing with imbalanced datasets is a common problem in machine learning, and there are many strategies to deal with it. The solution is usually a combination of several approaches.

An approach that is usually easy to implement is giving the underrepresented classes higher weight in the loss function. This way, the model will be penalized more if it fails to detect one of the rare samples. In other words, increasing the weight of some classes is a way to make them "more important" to the eyes of the loss function. Every major machine learning framework supports a way to control the weight assigned to each class, so the first choice is a correct answer.

Another good approach to solve the problem is to augment the underrepresented class. Data augmentation is about creating new training data by applying slight modifications to existing samples. For instance, rotation, translation, skewing, and color changes, among others.

Using data augmentation, we can increase the size of the training dataset and make the model more robust to these variations. This makes the third choice a correct answer as well.

These aren't the only ways to deal with imbalanced datasets: We can also oversample the underrepresented classes, downsample the dominant class, or generate additional synthetic data.

Decision Trees suffer from the same problems when dealing with imbalanced data, and the learning rate will not affect how the model handles the underrepresented class. Therefore, none of the other two options are correct.

In summary, the first and third choices are the correct answer to this question.

References

- "A Gentle Introduction to Imbalanced Classification"
- "Cost-Sensitive Decision Trees for Imbalanced Classification"
- Data augmentation with PyTorch
- Data augmentation with TensorFlow

April 11, 2022

I'm frustrated with my training loss

The correct answers to this question are:

- I should decrease the learning rate to avoid missing the local minima.
- I should increase the batch size to increase the variability of samples on every batch.

Let's start by focusing on the first option. The problem with the training loss is unlikely to be related to an imbalanced dataset. If that were the case, we would see a model struggling to learn anything, but we wouldn't see the loss oscillating back and forth.

The second and third choices focus on the learning rate. Should we increase it or decrease it?

As the second option suggests, increasing the learning rate will allow us to take larger steps in the gradient's direction, but we may miss the local minima! If we miss the local minima, we have to get back. If the learning rate is too large, the same thing will happen: we'll miss the local minima and start oscillating back and forth. Therefore, the second choice is not going to help at all.

In fact, there's a very good chance that the oscillation is happening because we are using a learning rate that's too high. Reducing the learning rate is a great experiment to see whether it fixes the problem. This makes the third choice a correct answer to this question.

Finally, let's suppose we are using a very small batch. We might see bad samples cause significant shifts in the training loss from one batch to another. This could also explain the oscillating loss. Increasing the batch size and ensuring the data is distributed correctly is also a great experiment to run. This makes the fourth choice a correct answer as well.

References

- "What could an oscillating training loss curve represent?"
- "Why is my training loss fluctuating?"
- Mini-Batch Gradient Descent

April 12, 2022

Ted wants to sell more cars

The correct answer to this question is:

› Use an unsupervised learning algorithm to produce potentially interesting ways to segment the customers.

Customer segmentation is a popular field where you try to find similar characteristics among your customers.

Although there are many different ways to frame a problem, customer segmentation is a perfect opportunity to use unsupervised learning techniques, specifically a clustering method.

For example, you could use K-Means to find interesting patterns and group together the customers that share them. A critical distinction to keep in mind is that you don't need to consider the segments preemptively; the clustering algorithm will find them for you.

Therefore, the first choice is the correct answer to this question.

References

- "Customer Segmentation with Machine Learning"
- *Pattern Recognition and Machine Learning* (PDF)
- Clustering by scikit-learn
- "10 Clustering Algorithms With Python"

April 13, 2022

Handling missing values

The correct answers to this question are:

- Replace missing values with the mean, median, mode, or other imputation techniques.
- Drop any rows or columns that contain missing or corrupted data.
- Predict the missing values using a separate machine learning model.
- Using machine learning algorithms that are robust to missing values.

Replacing missing values with the mean, median, mode, or other imputation techniques is an excellent approach to handling this problem. Remember that, for imputation to work, we need to apply it to those features where most of the data is in good shape and only a few values are missing. If the dataset contains features with that characteristic, imputation will help. Therefore, the first choice is correct.

Sometimes, a row or column is in such poor shape that the best approach is to drop it altogether. Imagine a feature where only a small percentage of rows have values or most of the data is corrupted. In those cases, getting rid of the data is the appropriate approach. This means that the second choice is also correct.

A more advanced technique commonly used is to predict missing values using a separate model. For example, you could use a linear regression model to fill in the blanks on one specific column

of data. This approach considers the correlation between other features and the column with missing values, potentially producing good results. This makes the third choice also correct.

Finally, the fourth choice is also correct. We can use a particular implementation of a machine learning algorithm that is robust to missing values. For example, we could run the k-nearest neighbors algorithm and ignore a column with missing values when computing the distance. This, however, depends on the implementation of the algorithm, so you need to watch out for that.

In summary, all four choices are correct.

References

- "How to Handle Missing Data with Python"
- Imputation of missing values by scikit-learn
- "Handling Missing Values"
- "Imputation"

April 14, 2022

The mysterious case of the strange loss

The correct answers to this question are:

- The regularization that Jena is using is too aggressive.
- Jena is using a learning rate that's too low.
- The neural network is getting stuck at local minima.

Many factors can cause problems during training. Let's discuss some of the more common ones.

Regularization is a helpful technique to avoid overfitting, but it may prevent the network from learning when it is too aggressive. This happens because regularization imposes a penalty on the network's weights, preventing them from becoming too large. If we aren't careful, the weights may stop changing, the network will stop learning, and the loss will stay high. Therefore, the first choice is a potential explanation for this problem.

Using a learning rate that's too low might also have the effect of keeping the training loss high. The network weights will be updated very slowly with a low learning rate. Unless we run the training process for many iterations, the network will struggle to get to where the loss is sufficiently low. Therefore, the second choice could also be a potential explanation for the problem.

Finally, the optimization may be stuck at a local minimum. This will cause the loss to stop decreasing altogether. Strategies to overcome this problem include better initializing the network's parameters or increasing the learning rate or momentum to overcome the local minimum. Therefore, the third choice is another correct answer to this question.

It's improbable that the problem is caused by a learning rate that's too high. If this were the case, Jena would see rather significant changes in the loss. With a high learning rate, it's also common to see the loss oscillating after some time.

In summary, the first three choices are correct answers to this question.

References

- "What should I do when my neural network doesn't learn?"
- "A Recipe for Training Neural Networks"
- "Regularization"

April 15, 2022

Looking behind François's Tweet

The correct answer to this question is:

› A new dataset changes the tradeoff between optimization and regularization. The learning-rate schedule is dataset-specific.

Here is François's Tweet:

> PSA: never attempt to use a training schedule from an old dataset on a new dataset. Even if the model, the number of samples, and the target distribution are the same, the intrinsic difficulty of the problem may have changed (tradeoff between optimization and regularization). Learning rate schedules and regularization are fundamentally dataset-specific, far more than model architecture.

Notice that he mentions that even when the dataset size and the target distribution are the same, the learning-rate schedule might not be helpful anymore. Therefore, neither the first nor second choices are correct.

The fourth choice argues that the issue is because of a likely change in the optimization process, which is not the case. Even when we don't change anything about the model architecture or the method we use to train it, a new dataset may change the intrinsic difficulty of the problem.

By using a different dataset, the problem's difficulty will change. We may need more regularization to avoid

overfitting, or we may need less of it to ensure the model learns the specifics of the problem. A different dataset leads to a different set of tradeoffs. Therefore, the third choice is the correct answer.

References

- François's Tweet
- *Deep Learning with Python*, Second Edition

April 16, 2022

Patricia is building a logistic classifier

The correct answer to this question is:

› Log loss: a function that returns the negative logarithm of the product of probabilities.

Out of the four choices in this question, the first three error functions satisfy Patricia's three characteristics. The absolute error, the square error, and the log loss are good choices based on this. The "mean percentage" is a made-up error function that we threw in there for fun, so we can quickly discard it as a valid answer.

Now, out of the three functions that satisfy Patricia's requirements, we need to analyze whether they can be used to build a binary classification model.

Neither the absolute nor square error functions are used in binary classification problems. They don't penalize mistakes as harshly as log loss does, so they aren't a good choice for this type of problem.

Log loss, however, is one of the most popular error functions and a perfect fit for a binary classifier like the one Patricia is trying to build. Therefore, the third choice is the correct answer.

References

- "Logistic Regression for Machine Learning"
- "Logistic Regression: Loss and Regularization"
- "Logistic Regression"

April 17, 2022

The birthday paradox

The correct answer to this question is:

› There's about a 50% probability of two people sharing the same birthday.

This is an interesting question.

Most people go with 22/365 and end up with a 6% probability. This answer, unfortunately, is wrong.

Let's think about the answer by simplifying the problem: given two people, what is the probability they share the same birthday?

Sometimes, it's useful to turn the problem around and focus on the opposite case: can we compute the probability of 2 people not sharing the same birthday? Let's see.

The first person's birthday is any day of the year, so there are 365 possibilities. The second person's birthday is any day of the year except the first person's birthday, so there are 364 possibilities.

In total, we have 365 * 364 = 13,2860 combinations where two people don't share a birthday. Since we are interested in the probability of two people sharing birthdays, we can use the following formula where DO refers to "desired outcomes" and PO to "possible outcomes":

```
P = 1 - (DO / PO)
P = 1 - ((365 * 364) / (365 * 365))
```

```
P = 1 - 99.72%
P = 0.0028
P = 0.28%
```

There's less than a 1% probability of 2 people sharing birthdays. Using this, we can generalize our process to the 23 people at the party.

```
P = 1 - (365! / ((365 - 23)! * 365^23))
```

The math is rough at this point, but you can use Wolfram Alpha to solve it for you.

```
P = 0.5072
P = 50%
```

So there you have it: There's about a 50% probability of two people sharing the same birthday at the party.

References

- "Birthday problem"
- "The Birthday Paradox Experiment"
- "Let's test your probabilistic intuition!"

April 18, 2022

Keeping pedestrians safe

The correct answer to this question is:

› The team should use the Fβ-Score of the model with a high value of β.

Imagine mounting a video camera on your car and exporting every frame after a long day of driving.

Most likely, the vast majority of images will not show pedestrians. Well, I guess there's a chance that you only drive on busy city streets and get to see many more pedestrians, but for most drivers, this won't be the case.

This insight is essential. Detecting pedestrians on images captured from a car is an imbalanced problem; only a few photos will contain the class we care about: a person.

When working with imbalanced problems where the class you want to detect represents the minority of samples, neither recall nor precision will be a good metric by itself to evaluate the model.

For example, you can use the recall to understand how many pedestrians the model is detecting, but if the model flags every image as showing pedestrians, the recall will be 100%. This, of course, it's not helpful so we can discard the first and second choices of this question.

Using the Fβ-Score, however, is a good choice.

The Fβ score lets us combine precision and recall into a single metric. When using β = 1, we place

equal weight on precision and recall. For values of $\beta > 1$, recall is weighted higher than precision; for values of $\beta < 1$, precision is weighted higher than recall.

You are probably familiar with F1-Score. F1-Score is just Fβ-Score with $\beta = 1$.

Now, to answer this question correctly, we need to determine if we want to prioritize recall or precision. This, however, is not always an easy task.

Let's say the team would instead report a pedestrian that is not there than miss somebody crossing the street. If the former happens, the car will probably brake for no reason, but something worse could happen if we don't stop the car.

This seems to be a sensible choice, and if the team agrees, they should prioritize building a system with high recall, so using a higher value of β is the way to go.

This makes the third choice the correct answer.

References

- "What is the F-Score?"
- "A Gentle Introduction to the Fbeta-Measure for Machine Learning"
- "F-score"

April 19, 2022

Thinking about softmax

The correct answer to this question is:

> The softmax function is a probabilistic or "smooth" version of the function that returns the index of the vector's largest value.

I'm not good with formal definitions, but here is a simplified explanation of what the softmax function does when used in the output layer of the network: the softmax function turns a vector of real values into another vector of probabilities that sum to 1. These probabilities are proportional to the relative scale of each value in the vector.

When used as the activation function of the output layer of a neural network, softmax converts the scores coming from the previous layer to a normalized probability distribution. This is a very convenient way to interpret the results of a multi-class classification model.

Let's look at the first choice for this question. It's a close one, but it's incorrect: softmax doesn't convert the network's input into a vector of probabilities; it converts the vector from the previous layer.

The third choice is also close, but it talks about a "sorted vector of probabilities." Softmax doesn't sort the output vector, so this option is incorrect.

The remaining two options seem to be similar, but they aren't if you read them carefully.

The second choice talks about a function that returns the index of the largest value in the vector. We call this function argmax.

If we run a vector through the argmax function we will get the index of the largest value in the vector. From a probabilistic perspective, the largest value will be assigned 1 while every other value will be assigned 0.

Softmax is a softer version of this function. Instead of returning 1 for the largest value and 0 for everything else, softmax returns a probability proportional to each value in the vector. This makes the second choice the correct answer to this question.

Finally, notice that the fourth choice is similar to the second one, but it talks about the "vector's maximum value." This would be a max function, not an argmax.

Running a vector through the max function returns the maximum value in that vector, not the index of where that value is located. From a probabilistic perspective, having the maximum value will not help create a vector of probabilities that sum up to 1, so the fourth choice is not a correct answer.

References

- "Softmax Activation Function with Python"
- "What is the Softmax Function"

April 20, 2022

Alice can't remember how One-Hot Encoding works

The correct answer to this question is:

› One-Hot Encoding creates additional features based on the number of unique values in a categorical feature.

Before analyzing this question, we need to understand what "categorical data" means.

Categorical data are variables that contain label values rather than numeric values. For example, a variable representing the temperature with values "hot," "warm," and "cold" is a categorical variable.

Although some algorithms can use categorical data directly, the majority can't: they require the data to be numeric. One-Hot Encoding is one of the techniques we can use to turn categorical data into a numerical representation.

For example, assume we have a dataset with a single feature called "temperature" that could have the values "hot," "warm," and "cold." Applying One-Hot Encoding will get us a new dataset with three features, one for each value of the original "temperature" column.

A sample that had the value "warm" in the previous column will now have the value 0 for both "hot" and "cold" and the value 1 under the "warm" feature.

This means that the second choice is the correct explanation of how One-Hot Encoding works.

The first choice is the opposite of how One-Hot Encoding works. The third and fourth choices talk about "string-encoded" features, which is not how it works either. None of these three choices are correct.

References

- "Why One-Hot Encode Data in Machine Learning?"
- "Here's All you Need to Know About Encoding Categorical Data"
- "One-hot"

April 21, 2022

The anatomy of ReLU

The correct answer to this question is:

› The function is continuous but not differentiable.

Let's cut right to the chase: ReLU is continuous but not differentiable, which means the fourth choice is the correct answer.

While the function is indeed differentiable for values of $x>0$ and $x<0$, it is not differentiable for $x=0$.

Think about it: what is the derivative of the ReLU function at $x=0$? Is it constant? Is it rising? The derivative is not defined at $x=0$, so the function is not differentiable.

This may be surprising because we know that Gradient Descent needs a differentiable function to work. How come it works with ReLU, then?

In a really good post about this topic, Sebastian Raschka wrote about this:

> In practice, it's relatively rare to have $x=0$ in the context of deep learning, hence, we usually don't have to worry too much about the ReLU derivative at $x=0$. Typically, we set it either to 0, 1, or 0.5.

So there you have it. The correct answer to this question is the fourth choice.

References

- "Why is the ReLU function not differentiable at x=0?"
- "The anatomy of ReLU"
- "A Gentle Introduction to the Rectified Linear Unit (ReLU)"

April 22, 2022

Australian birds

The correct answer to this question is:

› Marc's model suffers from sampling bias. He probably didn't include enough examples of Australian birds.

A common problem in machine learning is that a model that shows promising results during evaluation doesn't perform well when deployed in production.

While there may be various reasons for that, the story above points us in one particular direction.

Not training the model long enough is unlikely to be a problem here. Marc wouldn't be getting good results anywhere if this was the case. Notice that problems seem to only happen in Australia, so we can easily discard the first choice.

Insufficient model capacity can't either be a valid reason for what Marc is seeing. As we mentioned before, if this were the case, the model would be underfitting and give poor results across the board, not localized to a specific region.

Data and concept drift are indeed common problems with models in production. However, they arise when the environment changes over time, and so does the input to the model. In this case, the problem appeared straight after deployment.

This leaves us with the only correct answer: The most likely reason for this problem is that Marc didn't have enough data from Australia, so his model is struggling to recognize birds from that region.

This issue is called "sampling bias." It explains why the problem occurred in one particular country. Sampling bias is difficult to detect during development because the data is missing from the training and test datasets, so we can't notice it while evaluating the model.

In conclusion, the third choice is the correct answer to this question.

References

- "Sampling bias"
- Bird classification

April 22, 2022

Breaking a function down into pieces

The correct answer to this question is:

› x is a feature vector containing the variables that describe the car, and y is the logarithm of the price of the car.

Here we have a model that predicts the price of the car. We can think of the "model" as a function, albeit a complex one.

The good news is that we don't care what the actual function looks like. We can think of it as y = f(x).

This function has two components: the function's output —or y value— and the function's input —or x value.

In our example, the result of that function is what we are looking to predict: the car's price. That means that y represents the car's price. On the other hand, x is a feature vector containing all of the other variables that we use to train our model. These could include the color of the car, the make, model, year, etc.

We can rule out this question's third and fourth choices by understanding this.

We have two possibilities left: the result is the car price or a logarithmic transformation of the price.

Remember that we trained this model using the logarithm of the target variable, which means our model can't output the price directly. Instead, the model's output will always be the logarithm of the car's price.

Therefore, the first choice is the correct answer to this question.

References

- "How Machine Learning Algorithms Work"
- *Machine Learning Bookcamp*
- "Log Transformation: Purpose and Interpretation"

April 24, 2022

Mia and the feedback loop

The correct answer to this question is:

› Recurrent Neural Networks

Recurrent Neural Networks have an advantage over traditional feed-forward networks when working with time series or data points that depend upon previous samples.

The magic ingredient of Recurrent Neural Networks is the ability to store the information of previous inputs to generate the following output of the sequence. Recurrent Neural Networks do this by implementing the concept of a "feedback loop" or "feedback connection." These are connections feeding the hidden layers of the neural network back into themselves.

Neither Convolutional Neural Networks, Multilayer Perceptrons, nor Radial Basis Function Networks supports the concept of feedback loops. Only Recurrent Neural Networks do.

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network very popular for sequence prediction problems.

References

- "An Introduction To Recurrent Neural Networks And The Math That Powers Them"
- "Recurrent neural network"
- "A Gentle Introduction to Long Short-Term Memory Networks by the Experts"

April 25, 2022

Harper and the small gradients

The correct answers to this question are:

- Harper should try ReLU as the activation function since it's well-known for mitigating the vanishing gradient problem.
- Harper should modify the model architecture to introduce Batch Normalization.
- Harper should make sure they are initializing the weights properly. For example, using He initialization should help with the vanishing gradient problem.

If the gradients of the loss function approach zero, the model will stop learning because the network will stop updating the weights. This phenomenon is known as the Vanishing Gradient Problem, and it's very common when using the sigmoid and tanh activation functions in deep neural networks.

In contrast, ReLU is a way to solve the vanishing gradient problem. ReLU is much less likely to saturate, and its derivative is 1 for values larger than zero.

Batch normalization is another way to mitigate the vanishing gradient problem. Suppose we have a layer that uses a sigmoid activation function. We can normalize the input to that layer to ensure that the values don't reach the edges and stay around the area where derivatives aren't too small. By modifying the input to this layer with

batch normalization, we will be mitigating the vanishing gradient problem.

Randomly initializing the weights of the deep network could also be problematic and lead to the vanishing gradient problem. If we are using sigmoid or tanh as our activation functions, and many of the weights are initialized with values too small or too large, we will end up with derivatives close to zero. Using He initialization should prevent this from happening.

Finally, the vanishing gradient problem has nothing to do with the learning rate used to train the network, so the final choice is not a correct answer.

In summary, the first three choices are the correct answer to this question.

References

- "On weight initialization in deep neural networks"
- "The Vanishing Gradient Problem. The Problem, Its Causes, Its Significance, and Its Solutions"
- "How to Fix the Vanishing Gradients Problem Using the ReLU"
- "Vanishing gradient problem"

April 26, 2022

Exploring data before anything else

The correct answers to this question are:

- Learn the distribution of the target variable.
- Understand the features in the dataset and the distribution of their values.
- Assess the data quality, including missing or corrupt values.

Exploratory Data Analysis is a process that happens before we have a model. It's precisely the step that helps us create that model.

We focus on learning as much as we can about the data during this process. Here are some examples of activities that happen at this time:

- Investigate the dataset to discover valuable patterns.
- Spot any potential anomalies that may exist in the data.
- Use statistics and plots to test different hypotheses and check assumptions.
- Understand the distribution of the variable that we are trying to predict —the target variable.
- Determine whether there are missing or corrupt values in the data.

Exploratory Data Analysis doesn't include building a model, so the third choice is incorrect. We can't possibly evaluate a model that doesn't exist at this time.

All the other three choices are correct.

References

- "What is Exploratory Data Analysis?"
- "Exploratory data analysis"
- *Feature Engineering for Machine Learning*

April 27, 2022

Susan needs to make a decision

The correct answer to this question is:

› Susan should compute the area under the curve for both models and choose the one with the higher value.

Which model is better usually depends on the application. For some use cases, high recall is more important than precision, while in others, it is not.

However, contrary to the fourth choice, there is an objective way to determine which model is better overall.

The first two options suggest fixing one particular metric and choosing the model that performs the best on the other one. This is a valid approach, but it's not what Susan needs. Susan wants to determine which model is the best, but fixing either recall or precision won't return the best overall model since we would always be prioritizing one of the metrics.

For example, imagine that we tune both models to a recall above 90% and then pick the one with the higher precision. There's no guarantee that the model we choose is the best possible model overall—the one that better balances recall and precision. Instead, we just ensured that the model we picked was the best model with a recall above 90%.

To find the best overall model, Susan should compute the area under the ROC curve (Receiver Operating Characteristic) and choose the model with the higher value.

A ROC curve is a graph showing the True Positive Rate and the False Positive Rate at different classification thresholds. The area under this curve measures the performance of the model. A perfect model will have an area of 1.0, while a model that only makes mistakes will have an area of 0.0. Therefore, choosing the model with the higher area will give Susan the best overall model.

Therefore, the third choice is the correct answer to this question.

References

- ROC and AuC
- ROC metrics
- "Area under the curve"

April 28, 2022

Linear regression by hand

The correct answer to this question is:

› The result will be a vector y containing the predicted price of each house.

Let's start by looking at the mathematical properties of the multiplication of a matrix and a vector: Whenever we multiply a m × n matrix with a n × 1 vector, the result will always be a column vector of m × 1 dimensions.

The third and fourth choices suggest that we'll get a matrix as the result of the operation, but we know the multiplication will return a vector, so we can safely remove them as candidate answers.

The first choice proposes that the result will contain the actual price of each house, while the second choice suggests that the result will be the predicted price. Let's think about this for a moment.

If you look at the multiplication components, we have a matrix containing the features of every house and a vector of weights. The price we are trying to predict isn't part of those features. If you think about it, it won't make sense to predict a price by showing the model that same price (that would be cheating!) The algorithm estimates the vector of weights, so the actual price of the house is not there either.

Looking at the multiplication components, we can conclude that the result is the predicted price. But if you think about the problem logically, you

should arrive at the same answer: the entire goal of our model is to predict the price of a house, so that must be the result of the function.

Therefore, the correct choice is the second one: The multiplication result is a vector containing the price predictions.

References

- "Understanding Linear Regression with Mathematical Insights!"
- "Linear Regression for Machine Learning"

April 29, 2022

20,000 sunny and cloudy samples

The correct answers to this question are:

- Logistic Regression
- k-Nearest Neighbors
- Neural Networks
- Decision Trees

The four choices are examples of supervised learning techniques that you could use to build a binary classification model.

Although they all work in theory, they are very different approaches, and some might lead to better results on this dataset than others.

However, the problem doesn't give us any information about the characteristics of the data, so it's hard to disqualify any of the choices based on what we know.

Therefore, every choice is a correct answer.

References

- "Binary classification"
- "4 Types of Classification Tasks in Machine Learning"

April 30, 2022

The true meaning of hyperparameter tuning

The correct answer to this question is:

› Hyperparameter tuning is about choosing the optimal parameters for a learning algorithm to train a model.

We use the term "hyperparameter" to refer to the parameters and settings that we can use to control the learning process. These are the "knobs" and "levers" that we can control from a learning algorithm.

In contrast, we use "parameters" to refer to variables internal to a model whose values we estimate during the learning process using data.

A good way of thinking about this:

- Parameters: We can't control them manually. We learn their values during training.
- Hyperparameters: We do control them manually. They act like configuration settings for our models.

Each model has different hyperparameters: You can control the depth of a decision tree or the step size during the optimization process of a neural network.

Understanding this should be enough to analyze the four choices for this question.

Hyperparameters have nothing to do with the data, so the first and second choices are incorrect answers. They aren't about features or samples.

The fourth choice is somewhat attractive. A good set of hyperparameters will indirectly lead to a better-fitted model, but "tuning hyperparameters" is not about "choosing a better hypothesis." Therefore, this choice is not correct.

The third choice is the one that best describes what hyperparameter tuning is about: choosing the best set of parameters to tune a learning algorithm.

References

- "Overview of hyperparameter tuning"
- "Hyperparameter optimization"
- "What is the Difference Between a Parameter and a Hyperparameter?"