andrewekhalel / **MLQuestions**

Machine Learning and Computer Vision Engineer - Technical Interview Questions

README.md

Machine Learning Interview Questions

A collection of technical interview questions for machine learning and computer vision engineering positions.

1) What's the trade-off between bias and variance? [src]

If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data. [src]

2) What is gradient descent? [src]

[Answer]

3) Explain over- and under-fitting and how to combat them? [src]

[Answer]

4) How do you combat the curse of dimensionality? [src]

- Manual Feature Selection
- Principal Component Analysis (PCA)
- Multidimensional Scaling
- Locally linear embedding
  [src]

5) What is regularization, why do we use it, and give some examples of common methods? [src]

A technique that discourages learning a more complex or flexible model, so as to avoid the risk of overfitting. Examples

- Ridge (L2 norm)
- Lasso (L1 norm)
  The obvious *disadvantage* of **ridge** regression, is model interpretability. It will shrink the coefficients for least important predictors, very close to zero. But it will never make them exactly zero. In other words, the *final model will include all predictors*. However, in the case of the **lasso**, the L1 penalty has the effect of forcing some of the coefficient estimates to be *exactly equal* to zero when the tuning parameter λ is sufficiently large. Therefore, the lasso method also performs variable selection and is said to yield sparse models. [src]

6) Explain Principal Component Analysis (PCA)? [src]

[Answer]

7) Why is ReLU better and more often used than Sigmoid in Neural Networks? [src]

Imagine a network with random initialized weights (or normalised) and almost 50% of the network yields 0 activation because of the characteristic of ReLu (output 0 for negative values of x). This means a fewer neurons are firing (sparse activation) and the network is lighter. [src]

8) Given stride S and kernel sizes for each layer of a (1-dimensional) CNN, create a function to compute the receptive field of a particular node in the network. This is just finding how many input nodes actually connect through to a neuron in a CNN. [src]

9) Implement connected components on an image/matrix. [src]

10) Implement a sparse matrix class in C++. [src]

11) Create a function to compute an integral image, and create another function to get area sums from the integral image.[src]

12) How would you remove outliers when trying to estimate a flat plane from noisy samples? [src]

13) How does CBIR work? [src]

14) How does image registration work? Sparse vs. dense optical flow and so on. [src]

15) Describe how convolution works. What about if your inputs are grayscale vs RGB imagery? What determines the shape of the next layer? [src]

16) Talk me through how you would create a 3D model of an object from imagery and depth sensor measurements taken at all angles around the object. [src]

17) Implement SQRT(const double & x) without using any special functions, just fundamental arithmetic. [src]

18) Reverse a bitstring. [src]

19) Implement non maximal suppression as efficiently as you can. [src]

20) Reverse a linked list in place. [src]

21) What is data normalization and why do we need it? [src]

Data normalization is very important preprocessing step, used to rescale values to fit in a specific range to assure better convergence during backpropagation. In general, it boils down to subtracting the mean of each data point and dividing by its standard deviation. If we don't do this then some of the features (those with high magnitude) will be weighted more in the cost function (if a higher-magnitude feature changes by 1%, then that change is pretty big, but for smaller features it's quite insignificant). The data normalization makes all features weighted equally.

22) Why do we use convolutions for images rather than just FC layers? [src]

Firstly, convolutions preserve, encode, and actually use the spatial information from the image. If we used only FC layers we would have no relative spatial information. Secondly, Convolutional Neural Networks (CNNs) have a partially built-in translation in-variance, since each convolution kernel acts as it's own filter/feature detector.

23) What makes CNNs translation invariant? [src]

As explained above, each convolution kernel acts as it's own filter/feature detector. So let's say you're doing object detection, it doesn't matter where in the image the object is since we're going to apply the convolution in a sliding window fashion across the entire image anyways.

24) Why do we have max-pooling in classification CNNs? [src]

for a role in Computer Vision. Max-pooling in a CNN allows you to reduce computation since your feature maps are smaller after the pooling. You don't lose too much semantic information since you're taking the maximum activation. There's also a theory that max-pooling contributes a bit to giving CNNs more translation in-variance. Check out this great video from Andrew Ng on the benefits of max-pooling.

25) Why do segmentation CNNs typically have an encoder-decoder style / structure? [src]

The encoder CNN can basically be thought of as a feature extraction network, while the decoder uses that information to predict the image segments by "decoding" the features and upscaling to the original image size.

26) What is the significance of Residual Networks? [src]

The main thing that residual connections did was allow for direct feature access from previous layers. This makes information propagation throughout the network much easier. One very interesting paper about this shows how using local skip connections gives the network a type of ensemble multi-path structure, giving features multiple paths to propagate throughout the network.

27) What is batch normalization and why does it work? [src]

Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. The idea is then to normalize the inputs of each layer in such a way that they have a mean output activation of zero and standard deviation of one. This is done for each individual mini-batch at each layer i.e compute the mean and variance of that mini-batch alone, then normalize. This is analogous to how the inputs to networks are standardized. How does this help? We know that normalizing the inputs to a network helps it learn. But a network is just a series of layers, where the output of one layer becomes the input to the next. That means we can think of any layer in a neural network as the first layer of a smaller subsequent network. Thought of as a series of neural networks feeding into each other, we normalize the output of one layer before applying the activation function, and then feed it into the following layer (sub-network).

28) Why would you use many small convolutional kernels such as 3x3 rather than a few large ones? [src]

This is very well explained in the VGGNet paper. There are 2 reasons: First, you can use several smaller kernels rather than few large ones to get the same receptive field and capture more spatial context, but with the smaller kernels you are using less parameters and computations. Secondly, because with smaller kernels you will be using more filters, you'll be able to use more activation functions and thus have a more discriminative mapping function being learned by your CNN.

29) Why do we need a validation set and test set? What is the difference between them? [src]

When training a model, we divide the available data into three separate sets:

- The training dataset is used for fitting the model's parameters. However, the accuracy that we achieve on the training set is not reliable for predicting if the model will be accurate on new samples.
- The validation dataset is used to measure how well the model does on examples that weren't part of the training dataset. The metrics computed on the validation data can be used to tune the hyperparameters of the model. However, every time we evaluate the validation data and we make decisions based on those scores, we are leaking information from the validation data into our model. The more evaluations, the more information is leaked. So we can end up overfitting to the validation data, and once again the validation score won't be reliable for predicting the behaviour of the model in the real world.
- The test dataset is used to measure how well the model does on previously unseen examples. It should only be used once we have tuned the parameters using the validation set.

So if we omit the test set and only use a validation set, the validation score won't be a good estimate of the generalization of the model.

30) What is stratified cross-validation and when should we use it? [src]

Cross-validation is a technique for dividing data between training and validation sets. On typical cross-validation this split is done randomly. But in stratified cross-validation, the split preserves the ratio of the categories on both the training and validation datasets.

For example, if we have a dataset with 10% of category A and 90% of category B, and we use stratified cross-validation, we will have the same proportions in training and validation. In contrast, if we use simple cross-validation, in the worst case we may find that there are no samples of category A in the validation set.

Stratified cross-validation may be applied in the following scenarios:

- On a dataset with multiple categories. The smaller the dataset and the more imbalanced the categories, the more important it will be to use stratified cross-validation.
- On a dataset with data of different distributions. For example, in a dataset for autonomous driving, we may have images taken during the day and at night. If we do not ensure that both types are present in training and validation, we will have generalization problems.

31) Why do ensembles typically have higher scores than individual models? [src]

An ensemble is the combination of multiple models to create a single prediction. The key idea for making better predictions is that the models should make different errors. That way the errors of one model will be compensated by the right guesses of the other models and thus the score of the ensemble will be higher.

We need diverse models for creating an ensemble. Diversity can be achieved by:

- Using different ML algorithms. For example, you can combine logistic regression, k-nearest neighbors, and decision trees.
- Using different subsets of the data for training. This is called bagging.
- Giving a different weight to each of the samples of the training set. If this is done iteratively, weighting the samples according to the errors of the ensemble, it's called boosting. Many winning solutions to data science competitions are ensembles. However, in real-life machine learning projects, engineers need to find a balance between execution time and accuracy.

32) What is an imbalanced dataset? Can you list some ways to deal with it? [src]

An imbalanced dataset is one that has different proportions of target categories. For example, a dataset with medical images where we have to detect some illness will typically have many more negative samples than positive samples—say, 98% of images are without the illness and 2% of images are with the illness.

There are different options to deal with imbalanced datasets:

- Oversampling or undersampling. Instead of sampling with a uniform distribution from the training dataset, we can use other distributions so the model sees a more balanced dataset.
- Data augmentation. We can add data in the less frequent categories by modifying existing data in a controlled way. In the example dataset, we could flip the images with illnesses, or add noise to copies of the images in such a way that the illness remains visible.
- Using appropriate metrics. In the example dataset, if we had a model that always made negative predictions, it would achieve a precision of 98%. There are other metrics such as precision, recall, and F-score that describe the accuracy of the model better when using an imbalanced dataset.

33) Can you explain the differences between supervised, unsupervised, and reinforcement learning? [src]

In supervised learning, we train a model to learn the relationship between input data and output data. We need to have labeled data to be able to do supervised learning.

With unsupervised learning, we only have unlabeled data. The model learns a representation of the data. Unsupervised learning is frequently used to initialize the parameters of the model when we have a lot of unlabeled data and a small fraction of labeled data. We first train an unsupervised model and, after that, we use the weights of the model to train a supervised model.

In reinforcement learning, the model has some input data and a reward depending on the output of the model. The model learns a policy that maximizes the reward. Reinforcement learning has been applied successfully to strategic games such as Go and even classic Atari video games.

34) What is data augmentation? Can you give some examples? [src]

Data augmentation is a technique for synthesizing new data by modifying existing data in such a way that the target is not changed, or it is changed in a known way.

Computer vision is one of fields where data augmentation is very useful. There are many modifications that we can do to images:

- Resize
- Horizontal or vertical flip
- Rotate
- Add noise
- Deform
- Modify colors Each problem needs a customized data augmentation pipeline. For example, on OCR, doing flips will change the text and won't be beneficial; however, resizes and small rotations may help.

35) What is Turing test? [src]

The Turing test is a method to test the machine's ability to match the human level intelligence. A machine is used to challenge the human intelligence that when it passes the test, it is considered as intelligent. Yet a machine could be viewed as intelligent without sufficiently knowing about people to mimic a human.

36) What is Precision?

Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances
Precision = true positive / (true positive + false positive)
[src]

37) What is Recall?

Recall (also known as sensitivity) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. Recall = true positive / (true positive + false negative)
[src]

38) Define F1-score. [src]

It is the weighted average of precision and recall. It considers both false positive and false negative into account. It is used to measure the model's performance.
F1-Score = 2 * (precision * recall) / (precision + recall)

39) What is cost function? [src]

Cost function is a scalar functions which Quantifies the error factor of the Neural Network. Lower the cost function better the Neural network. Eg: MNIST Data set to classify the image, input image is digit 2 and the Neural network wrongly predicts it to be 3

40) List different activation neurons or functions. [src]

- Linear Neuron
- Binary Threshold Neuron
- Stochastic Binary Neuron
- Sigmoid Neuron
- Tanh function
- Rectified Linear Unit (ReLU)

41) Define Learning rate.

Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient. [src]

42) What is Momentum (w.r.t NN optimization)?

Momentum lets the optimization algorithm remembers its last step, and adds some proportion of it to the current step. This way, even if the algorithm is stuck in a flat region, or a small local minimum, it can get out and continue towards the true minimum. [src]

43) What is the difference between Batch Gradient Descent and Stochastic Gradient Descent?

Batch gradient descent computes the gradient using the whole dataset. This is great for convex, or relatively smooth error manifolds. In this case, we move somewhat directly towards an optimum solution, either local or global. Additionally, batch gradient descent, given an annealed learning rate, will eventually find the minimum located in it's basin of attraction.

Stochastic gradient descent (SGD) computes the gradient using a single sample. SGD works well (Not well, I suppose, but better than batch gradient descent) for error manifolds that have lots of local maxima/minima. In this case, the somewhat noisier gradient calculated using the reduced number of samples tends to jerk the model out of local minima into a region that hopefully is more optimal. [src]

44) Epoch vs Batch vs Iteration.

Epoch: one forward pass and one backward pass of **all** the training examples
Batch: examples processed together in one pass (forward and backward)
Iteration: number of training examples / Batch size

45) What is vanishing gradient? [src]

As we add more and more hidden layers, back propagation becomes less and less useful in passing information to the lower layers. In effect, as information is passed back, the gradients begin to vanish and become small relative to the weights of the networks.

46) What are dropouts? [src]

Long Short Term Memory – are explicitly designed to address the long term dependency problem, by maintaining a state what to remember and what to forget.

47) Define LSTM. [src]

As we add more and more hidden layers, back propagation becomes less and less useful in passing information to the lower layers. In effect, as information is passed back, the gradients begin to vanish and become small relative to the weights of the networks.

48) List the key components of LSTM. [src]

- Gates (forget, Memory, update & Read)
- tanh(x) (values between -1 to 1)
- Sigmoid(x) (values between 0 to 1)

49) List the variants of RNN. [src]

- LSTM: Long Short Term Memory
- GRU: Gated Recurrent Unit
- End to End Network
- Memory Network

50) What is Autoencoder, name few applications. [src]

Auto encoder is basically used to learn a compressed form of given data. Few applications include

- Data denoising
- Dimensionality reduction
- Image reconstruction
- Image colorization

51) What are the components of GAN? [src]

- Generator
- Discriminator

52) What's the difference between boosting and bagging?

Boosting and bagging are similar, in that they are both ensembling techniques, where a number of weak learners (classifiers/regressors that are barely better than guessing) combine (through averaging or max vote) to create a strong learner that can make accurate predictions. Bagging means that you take bootstrap samples (with replacement) of your data set and each sample trains a (potentially) weak learner. Boosting, on the other hand, uses all data to train each learner, but instances that were misclassified by the previous learners are given more weight so that subsequent learners give more focus to them during training. [src]

53) Explain how a ROC curve works. [src]

The ROC curve is a graphical representation of the contrast between true positive rates and the false positive rate at various thresholds. It's often used as a proxy for the trade-off between the sensitivity of the model (true positives) vs the fall-out or the probability it will trigger a false alarm (false positives).

54) What's the difference between Type I and Type II error? [src]

Type I error is a false positive, while Type II error is a false negative. Briefly stated, Type I error means claiming something has happened when it hasn't, while Type II error means that you claim nothing is happening when in fact something is. A clever way to think about this is to think of Type I error as telling a man he is pregnant, while Type II error means you tell a pregnant woman she isn't carrying a baby.

55) What's the difference between a generative and discriminative model? [src]

A generative model will learn categories of data while a discriminative model will simply learn the distinction between different categories of data. Discriminative models will generally outperform generative models on classification tasks.

Contributions

Contributions are most welcomed.

1. Fork the repository.
2. Commit your *questions* or *answers*.
3. Open **pull request**.