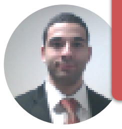


Apr 25 · 8 min read

Data Science and Machine Learning Interview Questions

Ah the dreaded machine learning interview. You feel like you know everything… until you're tested on it! But it doesn't have to be this way.

Over the past few months I've interviewed with many companies for entry-level roles involving Data Science and Machine Learning. To give you a bit of perspective, I was in graduate school in the last few months of my masters in machine learning and computer vision with most of my previous experience being research/academic, but with 8 months at an early stage startup (unrelated to ML). The roles included work in Data Science, general Machine Learning, and specializations in Natural Language Processing or Computer Vision. I interviewed with big companies like Amazon, Tesla, Samsung, Uber, Huawei, but also with many startups ranging from early-stage to well established and funded.

Today I'm going to share with you all of the interview questions I was asked and how to approach them. Many of the questions were quite common and expected theory, but many others were quite creative and curious. I'm going to simply list the most common ones since there's many resources about them online and go more in depth into some of the less common and trickier ones. I hope in reading this post that you can get great at Machine Learning interviews and land your dream job!

Let's dive in:

- What's the trade-off between bias and variance?
- What is gradient descent?
- Explain over- and under-fitting and how to combat them?
- How do you combat the curse of dimensionality?
- What is regularization, why do we use it, and give some examples of common methods?

- Explain Principal Component Analysis (PCA)?
- Why is ReLU better and more often used than Sigmoid in Neural Networks?
- **What is data normalization and why do we need it?** I felt this one would be important to highlight. 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.
- **Explain dimensionality reduction, where it's used, and it's benefits?** Dimensionality reduction is the process of reducing the number of feature variables under consideration by obtaining a set of principal variables which are basically the important features. Importance of a feature depends on how much the feature variable contributes to the information representation of the data and depends on which technique you decide to use. Deciding which technique to use comes down to trial-and-error and preference. It's common to start with a linear technique and move to non-linear techniques when results suggest inadequate fit. Benefits of dimensionality reduction for a data set may be: (1) Reduce the storage space needed (2) Speed up computation (for example in machine learning algorithms), less dimensions mean less computing, also less dimensions can allow usage of algorithms unfit for a large number of dimensions (3) Remove redundant features, for example no point in storing a terrain's size in both sq meters and sq miles (maybe data gathering was flawed) (4) Reducing a data's dimension to 2D or 3D may allow us to plot and visualize it, maybe observe patterns, give us insights (5) Too many features or too complex a model can lead to overfitting.
- **How do you handle missing or corrupted data in a dataset?** You could find missing/corrupted data in a dataset and either drop those rows or columns, or decide to replace them with another value. In Pandas, there are two very useful methods: isnull() and

dropna() that will help you find columns of data with missing or corrupted data and drop those values. If you want to fill the invalid values with a placeholder value (for example, 0), you could use the fillna() method.

- **Explain this clustering algorithm?** I wrote a popular article on the The 5 Clustering Algorithms Data Scientists Need to Know explaining all of them in detail with some great visualizations.
- **How would you go about doing an Exploratory Data Analysis (EDA)?** The goal of an EDA is to gather some insights from the data before applying your predictive model i.e gain some information. Basically, you want to do your EDA in a *coarse to fine manner*. We start by gaining some high-level global insights. Check out some imbalanced classes. Look at mean and variance of each class. Check out the first few rows to see what it's all about. Run a pandas `df.info()` to see which features are continuous, categorical, their type (int, float, string). Next, drop unnecessary columns that won't be useful in analysis and prediction. These can simply be columns that look useless, one's where many rows have the same value (i.e it doesn't give us much information), or it's missing a lot of values. We can also fill in missing values with the most common value in that column, or the median. Now we can start making some basic visualizations. Start with high-level stuff. Do some bar plots for features that are categorical and have a small number of groups. Bar plots of the final classes. Look at the most "general features". Create some visualizations about these individual features to try and gain some basic insights. Now we can start to get more specific. Create visualizations between features, two or three at a time. How are features related to each other? You can also do a PCA to see which features contain the most information. Group some features together as well to see their relationships. For example, what happens to the classes when A = 0 and B = 0? How about A = 1 and B = 0? Compare different features. For example, if feature A can be either "Female" or "Male" then we can plot feature A against which cabin they stayed in to see if Males and Females stay in different cabins. Beyond bar, scatter, and other basic plots, we can do a PDF/CDF, overlayed plots, etc. Look at some statistics like distribution, p-value, etc. Finally it's time to build the ML model. Start with easier stuff like Naive Bayes and Linear Regression. If you see that those suck or the data is highly non-linear, go with polynomial regression,

decision trees, or SVMs. The features can be selected based on their importance from the EDA. If you have lots of data you can use a Neural Network. Check ROC curve. Precision, Recall

- **How do you know which Machine Learning model you should use?** While one should always keep the "no free lunch theorem" in mind, there are some general guidelines. I wrote an article on how to select the proper regression model here. This cheatsheet is also fantastic!

- **Why do we use convolutions for images rather than just FC layers?** This one was pretty interesting since it's not something companies usually ask. As you would expect, I got this question from a company focused on Computer Vision. This answer has 2 parts to it. 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.

- **What makes CNNs translation invariant?** 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.

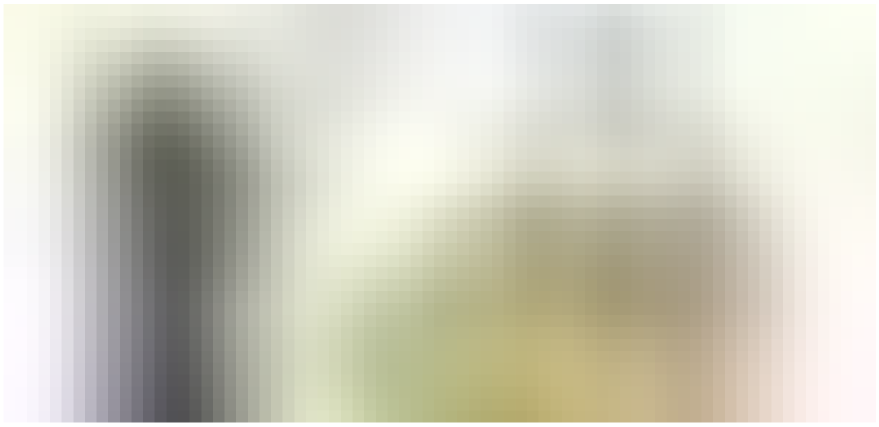
- **Why do we have max-pooling in classification CNNs?** Again as you would expect this is for a role in Computer Vision. Max-pooling in a CNN allows you to reduce computation since your feature maps are smaller after the pooling. You don't lose too much semantic information since you're taking the maximum 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.

- **Why do segmentation CNNs typically have an encoder-decoder style / structure?** 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.

- **What is the significance of Residual Networks?** 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.
- **What is batch normalization and why does it work?** 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).
- **How would you handle an imbalanced dataset?** I have an article about this! Check out #3 :)
- **Why would you use many small convolutional kernels such as 3x3 rather than a few large ones?** 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.
- **Do you have any other projects that would be related here?** Here you'll really draw connections between your research and

their business. Is there anything you did or any skills you learned that could possibly connect back to their business or the role you are applying for? It doesn't have to be 100% exact, just somehow related such that you can show that you will be able to directly add lots of value.

- **Explain your current masters research? What worked? What didn't? Future directions?** Same as the last question!



Conclusion

There you have it! All of the interview questions I got when apply for roles in Data Science and Machine Learning. I hope you enjoyed this post and learned something new and useful! If you did, feel free to hit the clap button.