developers

- Go to your profile
- Hire a developer
- Apply as a developer
- Log in

-
- Top 3%
- Why
- Clients
- Enterprise
- Community
- Blog
- About Us
- Go to your profile
- Hire a developer
- Apply as a developer
- Log in
-
    - Questions?
    - Contact Us
    -
    -
    -

- Questions?
- Contact Us
-
-
-

Hire a developer

10 Essential Machine Learning Interview Questions *

- 22shares
-
- f
- in
- G+

Submit an interview questionSubmit a question

Looking for experts? Check out Toptal’s machine learning engineers.

Why do we need a validation set and test set? What is the difference between them?

View the answer → Hide answer

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 valid 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 sco 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.

What is stratified cross-validation and when should we use it?

View the answer → Hide answer

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.

Why do ensembles typically have higher scores than individual models?

View the answer → Hide answer

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.

What is regularization? Can you give some examples of regularization techniques?

View the answer →Hide answer

Regularization is any technique that aims to improve the validation score, sometimes at the cost of reducing the training score.

Some regularization techniques:

- **L1** tries to minimize the absolute value of the parameters of the model. It produces sparse parameters.
- **L2** tries to minimize the square value of the parameters of the model. It produces parameters with small values.
- **Dropout** is a technique applied to neural networks that randomly sets some of the neurons' outputs to zero during training. This forces the network to learn better representations of the data by preventing complex interactions between the neurons: Each neuron needs to learn useful features.
- **Early stopping** will stop training when the validation score stops improving, even when the training score may be improving. This prevents overfitting on the training dataset.

What is the curse of dimensionality? Can you list some ways to deal with it?

View the answer →Hide answer

The curse of dimensionality is when the training data has a high feature count, but the dataset does not have enough samples for a model to learn correctly from so many features. For example, a training dataset of 100 samples with 100 features will be very hard to learn from because the model will find random relations between the features and the target. However, if we had a dataset of 100k samples with 100 features, the model could probably learn the correct relationships between the features and the target.

There are different options to fight the curse of dimensionality:

- **Feature selection.** Instead of using all the features, we can train on a smaller subset of features.
- **Dimensionality reduction.** There are many techniques that allow to reduce the dimensionality of the features. Principal component analysis (PCA) and using autoencoders are examples of dimensionality reduction techniques.
- **L1 regularization.** Because it produces sparse parameters, L1 helps to deal with high-dimensionality input.
- **Feature engineering.** It's possible to create new features that sum up multiple existing features. For example, we can get statistics such as the mean or median.

What is an imbalanced dataset? Can you list some ways to deal with it?

View the answer →Hide answer

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.

Can you explain the differences between supervised, unsupervised, and reinforcement learning?

View the answer →Hide answer

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.

What are some factors that explain the success and recent rise of deep learning?

View the answer →Hide answer

The success of deep learning in the past decade can be explained by three main factors:

1. **More data.** The availability of massive labeled datasets allows us to train models with more parameters and achieve state-of-the-art scores. Other ML algorithms do not scale as well as deep learning when it comes to dataset size.
2. **GPU.** Training models on a GPU can reduce the training time by orders of magnitude compared to training on a CPU. Currently, cutting-edge models are trained on multiple GPUs or even on specialized hardware.
3. **Improvements in algorithms.** ReLU activation, dropout, and complex network architectures have also been very significant factors.

What is data augmentation? Can you give some examples?

View the answer →Hide answer

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.

What are convolutional networks? Where can we use them?

View the answer → Hide answer

Convolutional networks are a class of neural network that use convolutional layers instead of fully connected layers. On a fully connected layer, all the output units have weights connecting to all the input units. On a convolutional layer, we have some weights that are repeated over the input.

The advantage of convolutional layers over fully connected layers is that the number of parameters is far smaller. This results in better generalization of the model. For example, if we want to learn a transformation from a 10x10 image to another 10x10 image, we will need 10,000 parameters if using a fully connected layer. If we use two convolutional layers, the first one having nine filters and the second one having one filter, with a kernel size of 3x3, we will have only 90 parameters.

Convolutional networks are applied where data has a clear dimensionality structure. Time series analysis is an example where one-dimensional convolutions are used; for images, 2D convolutions are used; and for volumetric data, 3D convolutions are used.

Computer vision has been dominated by convolutional networks since 2012 when AlexNet won the ImageNet challenge.

* There is more to interviewing than tricky technical questions, so these are intended merely as a guide. Not every "A" candidate worth hiring will be able to answer them all, nor does answering them all guarantee an "A" candidate. At the end of the day, hiring remains an art, a science — and a lot of work.

Submit an interview question

Submitted questions and answers are subject to review and editing, and may or may not be selected for posting, at the sole discretion of Toptal, LLC.

Name

Email

Enter your question here

Enter your answer here

All fields are required

□ I agree with the Terms and Conditions of Toptal, LLC's Privacy Policy

Submit a Question

Thanks for submitting your question.

Our editorial staff will review it shortly. Please note that submitted questions and answers are subject to review and editing, and may or may not be selected for posting, at the sole discretion of Toptal, LLC.

Looking for Machine Learning experts? Check out Toptal's machine learning engineers.

View full profile »

João Filgueiras

Portugal

João is an accomplished researcher and prototype developer with a special talent for breaking down large problems into solvable pieces. His background in R&D gives him the edge in implementing efficient, effective, and sophisticated yet clean solutions.

Machine Learning EngineerPython+7 more

Hire João

View full profile »

Mahmud Ridwan

Bangladesh

Mahmud is a software developer with a knack for efficiency, scalability, and stable solutions. With years of experience working with a wide range of technologies, he is still interested in exploring, encountering, and solving new and interesting programming problems.

Machine Learning EngineerExpress.jsGo+4 more

Hire Mahmud

View full profile »

Hermina Petric Maretić

Croatia

Hermina is a developer with proven skills in data mining, machine learning, and mathematical optimization. When building a project, she gives special attention to algorithm efficiency, putting an emphasis on creating quick and optimized software.
Machine Learning EngineerPythonC+4 more
Hire Hermina
Toptal connects the top 3% of freelance talent all over the world.

Join the Toptal community.

Hire a developer
or
Apply as a developer

Highest In-Demand Talent

- iOS Developers
- Front-End Developers
- UX Designers
- UI Designers
- Financial Modeling Consultants
- Interim CFOs
- Digital Project Managers

About

- Top 3%
- Clients
- Freelance Developers
- Freelance Designers
- Freelance Finance Experts
- Freelance Project Managers
- About Us

Contact

- Contact Us
- Press Center
- Careers
- FAQ

Social

- Facebook
- Twitter
- Google+
- LinkedIn

Hire the top 3% of freelance talent™

- Privacy Policy
- Website Terms

By continuing to use this site you agree to our Cookie Policy.
Got it

Find a world-class machine learning engineer for your team.Hire Toptal’s machine learning engineers