

How to Prepare for a Machine Learning Interview

A comprehensive guide to a Machine Learning interview: the things you have to master to become a Machine Learning expert and pass an interview

Posted by Josh (https://twitter.com/semanti_ca) on 02-08-2018

At [semanti.ca](https://www.semanti.ca) (<https://www.semanti.ca>), we believe that Machine Learning is a skill that any software developer needs to have. The extent to which Machine Learning has to be mastered can vary, of course, depending on the applicative domain of the developer.

However, if you want to make building Machine Learning systems your day-to-day activity, you probably want to apply to a Machine Learning Engineer or Machine Learning Developer role. The interview process for such roles is quite hard. Partially, this is due to the absence of Machine Learning specialization in most university computer science departments.

As a consequence, the range of questions that can be asked during an interview for an ML role can vary a lot depending on a company.

In this tutorial, we gathered the most important points that are common to almost any ML interview. If you want to pass an interview, or just become a Machine Learning expert, you will find almost everything you need to know below.

Programming is the Key

Choose one programming language, master it, and be ready to answer practical questions by writing code in this language.

The lack of knowledge of a particular programming language will not be a dealbreaker: any language can be learned fast enough, but it takes years to learn to program.

We recommend focussing on learning Python. It is a de-facto standard in the Machine Learning community. However, C++, C#, Java, Kotlin, Scala, Clojure, Lua or even R could be possible alternatives. We don't recommend JavaScript, Perl, Ruby, and PHP.

We don't recommend either such languages as Matlab or SAS/SPSS, as they are proprietary and quite niche. The modern ML community is open-source oriented: the best tools and frameworks that are used in the state-of-the-art ML systems (TensorFlow (<https://github.com/tensorflow/tensorflow>), Keras (<https://keras.io/>), scikit-learn (<http://scikit-learn.org>), numpy (<http://www.numpy.org>), scipy (<https://www.scipy.org/>), and pandas (<https://pandas.pydata.org/>)) are all open source, well-documented and of an excellent stability. You can rely upon them to build your AI systems.

There's an almost infinite amount of Python code online you can inspire from to build your own Machine Learning solutions.

The most important things to know about any programming language are:

- How to work with sets, lists, and dictionaries (maps) and when to use which;
- How to handle exceptions;
- Being capable of building specialized data structures such as linked lists, binary or prefix trees;
- Being capable of using highly optimized vectorized operations instead of loops.

Linux is your OS

Modern Machine Learning ecosystem is hard to imagine without Linux. Of course, it's possible to be an effective Machine Learning engineer on Windows. Mac is also a good alternative. We recommend, however, to start learning Linux at the same time as Python. A successful modern ML Engineer is supposed to know how to work with the Linux file system, how to install Linux and Python packages, and how to move data from and to a Linux system.

Know your Machine Learning Routine

During the interview, you will most likely be asked to describe the typical routine of a machine learning project:

- Data gathering (identifying the data you need, the sources you can get it from);
- Data cleanup (removing noise);
- Identifying and fixing missing values;
- Feature engineering (transforming your examples into a vector format);
- Train/validation/test split;
- Choice of the algorithm;
- Addressing the problems of high bias (underfitting) or high variance (overfitting);
- Hyperparameter tuning;
- Model deployment in production.

Know your Problems

Machine Learning problems can be separated into three major domains:

- Supervised learning;
- Unsupervised learning;
- Reinforcement learning.

Supervised learning deals with situations when you have labeled training examples, such as messages with the label either spam or not spam. Such subproblems as classification or regression belong to this domain.

Unsupervised learning deals with cases when all your data is not annotated, but you have to find some structure in it. Such subproblems as clustering or topic modeling belong to this domain.

Reinforcement learning deals with sequential decision problems. Usually, you don't have examples, but rather an environment and an agent that can act in this environment. Every time the agent executes an action it gets a reward. The goal is to learn a policy that permits the agent to maximize the reward in the long term.

Know your Algorithms

You have to be able to explain how at least one algorithm from each Machine Learning domain works.

For supervised learning, you have to be able to explain (at least conceptually) how the following algorithms work:

- Decision Trees;
- Linear and Logistic Regression,
- Support Vector Machines;
- Perceptron;
- Multilayer Perceptron (Feedforward Neural Network).

Specifically, in the SVM context, you have to be able to explain how kernels are used. In Multilayer Perceptron case you have to be able to talk about activations, loss functions, as well as forward and backward propagation.

For unsupervised learning, you have to be able to talk about such algorithms as K-Means clustering (and its difference from Expectation Maximization), Latent Dirichlet Allocation (and its difference from Latent Semantic Indexing). You have to be able to explain how to find the right number of clusters in the data or topics in the collection of documents.

For reinforcement learning, you can learn Q-learning, the algorithm most frequently cited in the literature. However, the knowledge of one of the modern deep reinforcement learning algorithms will play in your favor.

Know your Features

Feature extraction is one of the most important parts of any machine learning project. You have to be able to explain how text or sound is converted into features, how one-hot encoding works and how you can transform continuous attributes into categorical (binning) or the other way around (embedding).

Know How to Visualize

In Machine Learning, data is often very high-dimensional. Visualizing data points that have more than three dimensions can be challenging for humans.

You have to know several dimensionality reduction algorithms and be able to explain how they work and how they are different from one another. We recommend to master at least these three algorithms:

- Principal Component Analysis,
- t-SNE, and
- UMAP.

Know What to Do When Something Goes Wrong

Be able to answer the following questions:

- You train your model but it fails to predict the **training** data correctly. What would you do?

Possible solutions include:

- The data is possibly non-linearly separable, so use kernels;
- The data is probably too high dimensional, so a dimensionality technique, such as PCA (<http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>) or UMAP (<https://github.com/lmcinnes/umap>), could be used;
- The features are not informative enough, so add additional features or combine the existing ones into meta-features;

- The neural network is too small, so increase the number of units or layers.
- You train your model but it fails to predict the **test** data correctly. What would you do?

Possible solutions include:

- If the training data is not predicted correctly either, then first make sure it is predicted correctly, then fix the issue with the test data.
- If the training data is predicted correctly, then you can try:
 - use regularization (L1, L2, dropout);
 - get more training data (even if labeling more examples by hand).

Know how to Deploy your Model

As part of a bigger application, the Machine Learning model can be deployed in various ways. The most frequently used ones are:

- Docker (<https://www.docker.com/>) containers;
- RESTful web services (flask (<http://flask.pocoo.org/>) or falcon (<https://github.com/falconry/falcon>)).

When you deploy your model you have to be able to explain when the model consists of. In Python, usually, it's a [pickle](https://docs.python.org/3/library/pickle.html) (<https://docs.python.org/3/library/pickle.html>) file that contains the model object itself as well as additional objects such as feature extractors, data normalizers, and dimensionality reducer).

Know Statistics

You have to be able to explain basic probability distributions and when each is applied. Among the most important ones are Normal, Poisson, Binomial, Multinomial and Uniform distributions.

Be ready to answer the following questions:

- Name a probability distribution other than Normal and explain how to apply this probability?
- How best to select a representative sample of search queries from 5 million?
- The mean heights of men and women in a population were calculated to be **mM** and **mW**. What is the mean height of the total population?
- Three friends in Seattle told you it's rainy. Each has a probability of 1/3 of lying. What's the probability of Seattle is rainy?
- How do you detect if a new observation is an outlier?
- What is the Central Limit Theorem? Why it's important?
- Explain the difference between mean and median.
- Define variance.
- What is covariance? Where is it used?
- What is the goal of A/B Testing?
- What is sampling? Why we need it? What is stratified sampling?

Programming Challenges

Be ready to write the programming code on the whiteboard or on a computer. Usually, the challenges are not ML related because it would be quite long to accomplish. Here some examples of coding challenges:

- Find all palindromic substrings in a given string.
- How would you check if a linked list has cycles?

- Merge k (in this case $k=2$) arrays and sort them.
- Find max sum subsequence from a sequence of integers.
- Create a function that checks if a word is a palindrome.
- You have a 'csv' file with ID and Quantity columns, it doesn't fit in memory. Write a program in any language of your choice to aggregate the Quantity column.
- Given a list A of objects and another list B which is identical to A except that one element is removed, find the removed element.
- Given a list of integers (positive & negative), write an algorithm to find whether there's at least a pair of integers that sum up to zero.

Other Things to Know

While we didn't mention it in the previous sections, these questions could be asked in the interview and you have to be prepared to answer them:

- What are your go-to packages? (scikit-learn, xgboost, Pandas, numpy/scipy, Keras, matplotlib)
- What techniques do you use to fine-tune hyperparameters? (Cross-validation, grid-search, random-search, tree of Parzen estimators, Bayesian optimization.)
- How to quickly find if a text contains one of a million substrings. (Prefix trees.)
- Explain word embeddings. How are they learned?
- What is Stochastic Gradient Descent. Describe it in your own words?
- Explain the difference between Gradient Descent and Stochastic Gradient Descent. When use which?

- How can dropout be useful in a neural network?
- Explain Batch Normalization. What benefit does it bring?
- What is the use for a 1x1 convolution?
- How to represent a text document for machine learning?
- How to predict the next value in a time series?
- How to measure the similarity of two words or two documents?
- You have a search engine. How would you generate related searches for a query?
- How would you suggest followers on Twitter?
- What are support vectors in Support Vector Machines?
- What's the difference between Logistic Regression and SVM?
- When training an SVM, what value are you optimizing for?
- What is an unbalanced classification problem and how to deal with it?
- What is data wrangling? Explain its main steps.
- What would you do to summarize a Twitter feed?
- Explain the problem of vanishing gradient. How to deal with it?
- Explain the problem of exploding gradient. How to deal with it?
- Explain the need of the bias term.
- Explain the difference between unsupervised, semi-supervised and self-supervised learning.


- How Generative Adversarial Neural Networks (GANs) work?
- What is an auto-encoder? Why do we "auto-encode" something?
- What is a variational auto-encoder? Where can it be useful?
- When do we use sigmoid for an output function? What is the problem with sigmoid during backpropagation?
- What is transfer learning? How to do it in neural networks?
- How to combine two different kinds of input (ex: image and text or sequence and vector) in a neural network?
- How to make a neural network predict two different kinds of output?
- How does a logistic regression model know what the optimal coefficients are?
- Why use feature selection? What are the main techniques of feature selection?
- Explain ensemble learning. What techniques do you know?
- Why are ensemble methods superior to individual models?
- Explain bagging.
- Explain boosting.
- How to combine predictions of several different learning algorithms?
- Explain the difference between parametric and non-parametric models?
- Why SVM doesn't give a probabilistic output? Can we make transform it to get probabilities?


- How to build a multiclass classifier? Name two different strategies.
- How to build a multilabel classifier?
- What's the trade-off between bias and variance?
- How is KNN different from k-means?
- What is the difference between hard and soft clustering?
- Define precision and recall.
- State the Bayes Theorem? Why it's so important?
- How AlphaGo/AlphaZero work?
- Why Naive Bayes is called "naive"?
- What's the difference between a generative and discriminative model?
- How do you handle missing or corrupted data in a dataset?
- How to measure the difference between two probability distributions?
- What is cross-entropy? How is it useful in Machine Learning?
- How to detect outliers?
- Explain the difference between a test set and a validation set.


If you know the answers to most of the above questions, we are sure that you are well-equipped to pass a Machine Learning interview. Let us know (<https://www.semanti.ca/contacts>) if this tutorial helped you to pass an interview. Also, let us know if you think that some important aspect of an ML job is missing in our tutorial.


And do not be discouraged if you don't succeed the first time. The key to success is constant learning and persistence!


Like it? Share it!

 (<https://twitter.com/home?status=https://www.semanti.ca/blog/?how-to-prepare-for-a-machine-learning-interview>)

 (<https://www.facebook.com/sharer/sharer.php?u=https://www.semanti.ca/blog/?how-to-prepare-for-a-machine-learning-interview>)

 (<https://plus.google.com/share?url=https://www.semanti.ca/blog/?how-to-prepare-for-a-machine-learning-interview>)

 (<https://www.linkedin.com/shareArticle?mini=true&url=https://www.semanti.ca/blog/?how-to-prepare-for-a-machine-learning-interview>)

 (<http://www.reddit.com/submit?url=https://www.semanti.ca/blog/?how-to-prepare-for-a-machine-learning-interview>)