

---

## 65 Machine Learning Interview Questions 2023

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

### Recently added: Natural Language Processing (NLP) Interview Questions 2023

**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]

Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost).

Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimization algorithm.

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

ML/DL models essentially learn a relationship between its given inputs(called training features) and objective outputs(called labels). Regardless of the quality of the learned relation(function), its performance on a test set(a collection of data different from the training input) is subject to investigation.

Most ML/DL models have trainable parameters which will be learned to build that input-output relationship. Based on the number of parameters each model has, they can be sorted into more flexible(more parameters) to less flexible(less parameters).

The problem of Underfitting arises when the flexibility of a model(its number of parameters) is not adequate to capture the underlying pattern in a training dataset. Overfitting, on the other hand, arises when the model is too flexible to the underlying pattern. In the later case it is said that the model has "memorized" the training data.

An example of underfitting is estimating a second order polynomial(quadratic function) with a first order polynomial(a simple line). Similarly, estimating a line with a 10th order polynomial would be an example of overfitting.

---

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

- Feature Selection(manual or via statistical methods)
- 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  $\lambda$  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]

Principal Component Analysis (PCA) is a dimensionality reduction technique used in machine learning to reduce the number of features in a dataset while retaining as much information as possible. It works by identifying the directions (principal components) in which the data varies the most, and projecting the data onto a lower-dimensional subspace along these directions.

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

- Computation Efficiency: As ReLU is a simple threshold the forward and backward path will be faster.
- Reduced Likelihood of Vanishing Gradient: Gradient of ReLU is 1 for positive values and 0 for negative values while Sigmoid activation saturates (gradients close to 0) quickly with slightly higher or lower inputs leading to vanishing gradients.
- Sparsity: Sparsity happens when the input of ReLU is negative. This means fewer neurons are firing ( sparse activation ) and the network is lighter.

[src1] [src2]

#### 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]** The receptive field are defined portion of space within an inputs that will be used during an operation to generate an output.

Considering a CNN filter of size  $k$ , the receptive field of a peculiar layer is only the number of input used by the filter, in this case  $k$ , multiplied by the dimension of the input that is not being reduced by the convolutional filter  $a$ . This results in a receptive field of  $k \cdot a$ .

More visually, in the case of an image of size  $32 \times 32 \times 3$ , with a CNN with a filter size of  $5 \times 5$ , the corresponding receptive field will be the filter size, 5 multiplied by the depth of the input volume (the RGB colors) which is the color dimension. This thus gives us a receptive field of dimension  $5 \times 5 \times 3$ .

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

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

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

**12) How would you remove outliers when trying to estimate a flat plane from noisy samples? [src]** Random sample consensus (RANSAC) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates. [src]

**13) How does CBIR work? [src]** [Answer] Content-based image retrieval is the concept of using images to gather metadata on their content. Compared to the current image retrieval approach based on the keywords associated to the images, this technique generates its metadata from computer vision techniques to extract the relevant informations that will be used during the querying step. Many approach are possible from feature detection to retrieve keywords to the usage of CNN to extract dense features that will be associated to a known distribution of keywords.

With this last approach, we care less about what is shown on the image but more about the similarity between the metadata generated by a known image and a list of known label and or tags projected into this metadata space.

**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?**[\[src\]](#)

In a convolutional neural network (CNN), the convolution operation is applied to the input image using a small matrix called a kernel or filter. The kernel slides over the image in small steps, called strides, and performs element-wise multiplications with the corresponding elements of the image and then sums up the results. The output of this operation is called a feature map.

When the input is RGB (or more than 3 channels) the sliding window will be a sliding cube. The shape of the next layer is determined by Kernel size, number of kernels, stride, padding, and dialation.

[\[src1\]](#)[\[src2\]](#)

**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.** [\[src\]](#) There are two popular methods for 3D reconstruction: \* Structure from Motion (SfM) [\[src\]](#)

- Multi-View Stereo (MVS) [\[src\]](#)

SfM is better suited for creating models of large scenes while MVS is better suited for creating models of small objects.

**17) Implement `SQRT(const double & x)` without using any special functions, just fundamental arithmetic.** [\[src\]](#) The taylor series can be used for this step by providing an approximation of `sqrt(x)`:

[\[Answer\]](#)

**18) Reverse a bitstring.** [\[src\]](#) If you are using python3 :

```
1 data = b'\xAD\xDE\xDE\xC0'
2 my_data = bytearray(data)
3 my_data.reverse()
```

**19) Implement non maximal suppression as efficiently as you can.** [\[src\]](#) Non-Maximum Suppression (NMS) is a technique used to eliminate multiple detections of the same object in a given image. To solve that first sort bounding boxes based on their scores ( $N \log N$ ). Starting with the box with the highest score, remove boxes whose overlapping metric (IoU) is greater than a certain threshold. ( $N^2$ )

To optimize this solution you can use special data structures to query for overlapping boxes such as R-tree or KD-tree. ( $N \log N$ ) [\[src\]](#)

---

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

**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.

$$F1\text{-Score} = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

**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 its 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]** Dropout is a simple way to prevent a neural network from overfitting. It is the dropping out of some of the units in a neural network. It is similar to the natural reproduction process, where the nature produces offsprings by combining distinct genes (dropping out others) rather than strengthening the co-adapting of them.

**47) Define LSTM. [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.

#### **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? [src]** 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.

**56) Instance-Based Versus Model-Based Learning.**

- **Instance-based Learning:** The system learns the examples by heart, then generalizes to new cases using a similarity measure.
- **Model-based Learning:** Another way to generalize from a set of examples is to build a model of these examples, then use that model to make predictions. This is called model-based learning. [src]

**57) When to use a Label Encoding vs. One Hot Encoding?** This question generally depends on your dataset and the model which you wish to apply. But still, a few points to note before choosing the right encoding technique for your model:

We apply One-Hot Encoding when:

- The categorical feature is not ordinal (like the countries above)
- The number of categorical features is less so one-hot encoding can be effectively applied

We apply Label Encoding when:

- The categorical feature is ordinal (like Jr. kg, Sr. kg, Primary school, high school)
- The number of categories is quite large as one-hot encoding can lead to high memory consumption

[src]

**58) What is the difference between LDA and PCA for dimensionality reduction?** Both LDA and PCA are linear transformation techniques: LDA is a supervised whereas PCA is unsupervised – PCA ignores class labels.

We can picture PCA as a technique that finds the directions of maximal variance. In contrast to PCA, LDA attempts to find a feature subspace that maximizes class separability.

[src]

---

**59) What is t-SNE?** t-Distributed Stochastic Neighbor Embedding (t-SNE) is an unsupervised, non-linear technique primarily used for data exploration and visualizing high-dimensional data. In simpler terms, t-SNE gives you a feel or intuition of how the data is arranged in a high-dimensional space.

[src]

**60) What is the difference between t-SNE and PCA for dimensionality reduction?** The first thing to note is that PCA was developed in 1933 while t-SNE was developed in 2008. A lot has changed in the world of data science since 1933 mainly in the realm of compute and size of data. Second, PCA is a linear dimension reduction technique that seeks to maximize variance and preserves large pairwise distances. In other words, things that are different end up far apart. This can lead to poor visualization especially when dealing with non-linear manifold structures. Think of a manifold structure as any geometric shape like: cylinder, ball, curve, etc.

t-SNE differs from PCA by preserving only small pairwise distances or local similarities whereas PCA is concerned with preserving large pairwise distances to maximize variance.

[src]

**61) What is UMAP?** UMAP (Uniform Manifold Approximation and Projection) is a novel manifold learning technique for dimension reduction. UMAP is constructed from a theoretical framework based in Riemannian geometry and algebraic topology. The result is a practical scalable algorithm that applies to real world data.

[src]

**62) What is the difference between t-SNE and UMAP for dimensionality reduction?** The biggest difference between the output of UMAP when compared with t-SNE is this balance between local and global structure - UMAP is often better at preserving global structure in the final projection. This means that the inter-cluster relations are potentially more meaningful than in t-SNE. However, it's important to note that, because UMAP and t-SNE both necessarily warp the high-dimensional shape of the data when projecting to lower dimensions, any given axis or distance in lower dimensions still isn't directly interpretable in the way of techniques such as PCA.

[src]

**63) How Random Number Generator Works, e.g. rand() function in python works?** It generates a pseudo random number based on the seed and there are some famous algorithm, please see below link for further information on this. [src]

---

**64) Given that we want to evaluate the performance of ‘n’ different machine learning models on the same data, why would the following splitting mechanism be incorrect**

```
1 def get_splits():
2     df = pd.DataFrame(...)
3     rnd = np.random.rand(len(df))
4     train = df[ rnd < 0.8 ]
5     valid = df[ rnd >= 0.8 & rnd < 0.9 ]
6     test = df[ rnd >= 0.9 ]
7
8     return train, valid, test
9
10 #Model 1
11
12 from sklearn.tree import DecisionTreeClassifier
13 train, valid, test = get_splits()
14 ...
15
16 #Model 2
17
18 from sklearn.linear_model import LogisticRegression
19 train, valid, test = get_splits()
20 ...
```

The `rand()` function orders the data differently each time it is run, so if we run the splitting mechanism again, the 80% of the rows we get will be different from the ones we got the first time it was run. This presents an issue as we need to compare the performance of our models on the same test set. In order to ensure reproducible and consistent sampling we would have to set the random seed in advance or store the data once it is split. Alternatively, we could simply set the ‘`random_state`’ parameter in sklearn’s `train_test_split()` function in order to get the same train, validation and test sets across different executions.

[src]

**65) What is the difference between Bayesian vs frequentist statistics? [src]** Frequentist statistics is a framework that focuses on estimating population parameters using sample statistics, and providing point estimates and confidence intervals.

Bayesian statistics, on the other hand, is a framework that uses prior knowledge and information to update beliefs about a parameter or hypothesis, and provides probability distributions for parameters.

The main difference is that Bayesian statistics incorporates prior knowledge and beliefs into the analysis, while frequentist statistics doesn’t.

---

## Contributions

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

## Preparation Resources

1. All of Statistics: A Concise Course in Statistical Inference by Larry Wasserman
2. Machine Learning by Tom Mitchell
3. Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications by Chip Huyen