1. What are the key reasons for reducing the dimensionality of a dataset? What are the major disadvantages?

Motivations and drawbacks: • The main motivations for dimensionality reduction are: — To speed up a subsequent training algorithm (in some cases it may even remove noise and redundant features, making the training algorithm perform better). — To visualize the data and gain insights on the most important features. — Simply to save space (compression). • The main drawbacks are: — Some information is lost, possibly degrading the performance of subsequent training algorithms. — It can be computationally intensive. — It adds some complexity to your Machine Learning pipelines. —Transformed features are often hard to interpret.

1. What is the dimensionality curse?

The curse of dimensionality refers to the fact that many problems that do not exist in low-dimensional space arise in high-dimensional space. In Machine Learning, one common manifestation is the fact that randomly sampled highdimensional vectors are generally very sparse, increasing the risk of overfitting and making it very difficult to identify patterns in the data without having plenty of training data.

3. Tell if its possible to reverse the process of reducing the dimensionality of a dataset? If so, how can you go about doing it? If not, what is the reason?

Once a dataset’s dimensionality has been reduced using one of the algorithms we discussed, it is almost always impossible to perfectly reverse the operation, because some information gets lost during dimensionality reduction. Moreover, while some algorithms (such as PCA) have a simple reverse transformation procedure that can reconstruct a dataset relatively similar to the original, other algorithms (such as T-SNE) do not.

4. Can PCA be utilized to reduce the dimensionality of a nonlinear dataset with a lot of variables?

PCA can be used to significantly reduce the dimensionality of most datasets, even if they are highly nonlinear, because it can at least get rid of useless dimensions. However, if there are no useless dimensions—for example, the Swiss roll—then reducing dimensionality with PCA will lose too much information. You want to unroll the Swiss roll, not squash it.

5. Assume you're running PCA on a 1,000-dimensional dataset with a 95 percent explained variance ratio. What is the number of dimensions that the resulting dataset would have?

That’s a trick question: it depends on the dataset. Let’s look at two extreme examples. First, suppose the dataset is composed of points that are almost perfectly aligned. In this case, PCA can reduce the dataset down to just one dimension while still preserving 95% of the variance. Now imagine that the dataset is composed of perfectly random points, scattered all around the 1,000 dimensions. In this case all 1,000 dimensions are required to preserve 95% of the variance. So the answer is, it depends on the dataset, and it could be any number between 1 and 1,000. Plotting the explained variance as a function of the number of dimensions is one way to get a rough idea of the dataset’s intrinsic dimensionality.

6. Will you use vanilla PCA, incremental PCA, randomized PCA, or kernel PCA in which situations?

Regular PCA is the default, but it works only if the dataset fits in memory. Incremental PCA is useful for large datasets that don’t fit in memory, but it is slower than regular PCA, so if the dataset fits in memory you should prefer regular PCA. Incremental PCA is also useful for online tasks, when you need to apply PCA on the fly, every time a new instance arrives. Randomized PCA is useful when you want to considerably reduce dimensionality and the dataset fits in memory; in this case, it is much faster than regular PCA. Finally, Kernel PCA is useful for nonlinear datasets.

7. How do you assess a dimensionality reduction algorithm's success on your dataset?

Intuitively, a dimensionality reduction algorithm performs well if it eliminates a lot of dimensions from the dataset without losing too much information. One way to measure this is to apply the reverse transformation and measure the reconstruction error. However, not all dimensionality reduction algorithms provide a reverse transformation. Alternatively, if you are using dimensionality reduction as a preprocessing step before another Machine Learning algorithm (e.g., a Random Forest classifier), then you can simply measure the performance of that second algorithm; if dimensionality reduction did not lose too much information, then the algorithm should perform just as well as when using the original dataset.

8. Is it logical to use two different dimensionality reduction algorithms in a chain?

It can absolutely make sense to chain two different dimensionality reduction algorithms. A common example is using PCA to quickly get rid of a large number of useless dimensions, then applying another much slower dimensionality reduction algorithm, such as LLE. This two-step approach will likely yield the same performance as using LLE only, but in a fraction of the time.