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

Answers:

Advantages of Dimensionality Reduction

* It helps in data compression, and hence reduced storage space.
* It reduces computation time.
* It also helps remove redundant features, if any.

Disadvantages of Dimensionality Reduction

* It may lead to some amount of data loss.
* PCA tends to find linear correlations between variables, which is sometimes undesirable.
* PCA fails in cases where mean and covariance are not enough to define datasets.
* We may not know how many principal components to keep- in practice, some thumb rules are applied.

2. What is the dimensionality curse?

Answers: The curse of dimensionality basically means that the error increases with the increase in the number of features. It refers to the fact that algorithms are harder to design in high dimensions and often have a running time exponential in the dimensions.

Example: It's easy to catch a caterpillar moving in a tube (1 dimension). It's harder to catch a dog if it were running around on the plane (two dimensions). It's much harder to hunt birds, which now have an extra dimension they can move in.

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?

Answers:

Dimensionality reduction (compression of information) is reversible in auto-encoders. Auto-encoder is regular neural network with bottleneck layer in the middle. You have for instance 20 inputs in the first layer, 10 neurons in the middle layer and again 20 neurons in the last layer. When you train such network you force it to compress information to 10 neurons and then uncompress again minimizing error in the last layer(desired output vector equals input vector). When you use well known Back-propagation algorithm to train such network it performs PCA - Principal Component Analysis. PCA returns uncorrelated features. It's not very powerful.

By using more sophisticated algorithm to train auto-encoder you can make it perform nonlinear ICA - Independent Component Analysis. ICA returns statistically independent features. This training algorithm searches for low complexity neural networks with high generalization capability. As a byproduct of regularization you get ICA.

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

Answers:

PCA (Principal component analysis) 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 — as in a Swiss roll dataset — then reducing dimensionality with PCA will lose too much information. We 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?

Answers:

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 roughly 950 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 950. 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?

Answers:

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 we should prefer regular PCA. Incremental PCA is also useful for online tasks, when we need to apply PCA on the fly, every time a new instance arrives. Randomized PCA is useful when we 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?

Answers:

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 we are using dimensionality reduction as a pre-processing step before another Machine Learning algorithm (e.g., a Random Forest classifier), then we 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?

Answers:

It can make sense to combine two DR methods. One can use a fast projection methods (PCA) to first get rid of useless dimensions (i.e. dimensions that have no variance), and then use a slow manifold learning methods (LLE) to ‘unfold’ then remaining dataset to even lower dimensions.