Dimensionality Reduction / Curse of Dimensionality Review

Akhil Vasvani

March 2019

1 Questions

Exercise 1. Why do we need dimensionality reduction techniques?

Proof. A classic unsupervised learning task is to find the "best" representation of the data. By 'best' we can mean different things, but generally speaking we are looking for a representation that preserves as much information about $\boldsymbol{x}$ as possible while obeying some penalty or constraint aimed at keeping the representation simpler or more accessible than $\boldsymbol{x}$ itself.

There are multiple ways of defining a simpler representation. Three of the most common include **lower dimensional representations**, **sparse representations** and **independent representations**. Low-dimensional representations attempt to compress as much information about x as possible in a smaller representation. Sparse representations (Barlow, 1989; Olshausen and Field, 1996; Hinton and Ghahramani, 1997) embed the dataset into a representation whose entries are mostly zeroes for most inputs. The use of sparse representations typically requires increasing the dimensionality of the representation, so that the representation becoming mostly zeroes does not discard too much information. This results in an overall structure of the representation that tends to distribute data along the axes of the representation, which helps better visualize the data. Independent representations attempt to disentangle the sources of variation underlying the data distribution such that the dimensions of the representation are statistically independent, which speeds up the learning algorithm. □

Exercise 2. Why do we need PCA and what does it do?

Proof. **Principal Component analysis** (PCA) learns a representation that has lower dimensionality than the original input. It also learns a representation whose elements have no linear correlation with each other. This is a first step toward the criterion of learning representations whose elements are statistically independent. To achieve full independence, a representation learning algorithm must also remove the nonlinear relationships between variables.

TL;DR. PCA tries to find a lower dimensional surface such the sum of the squared projection error is minimized.

(https://stats.stackexchange.com/questions/32174/pca-objective-function-what-is-the-connection-between-maximizing-variance-and-m)

□

Exercise 3. What is the difference between logistic regression and PCA?

Proof. We can generalize linear regression to the classification scenario by defining a different family of probability distributions. If we have two classes, class 0 and class 1, then we need only specify the probability of one of these classes. The probability of class 1 determines the probability of class 0, because these two values must add up to 1.

The normal distribution over real-valued numbers that we used for linear regression is parametrized in terms of a mean. Any value we supply for this mean is valid. A distribution over a binary variable is slightly more complicated, because its mean must always be between 0 and 1. One way to solve this problem is to use the logistic sigmoid function to squash the output of the linear function into the interval (0, 1) and interpret that value as a probability:

$$P(y = 1|\boldsymbol{x}; \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}) \tag{1}$$

This approach is known as **logistic regression** (a somewhat strange name since we use the model for classification rather than regression).

The key difference between two approaches: PCA will NOT consider the response variable but only the variance of the independent variables and **logistic regression** will consider how each independent variable impact on response variable.

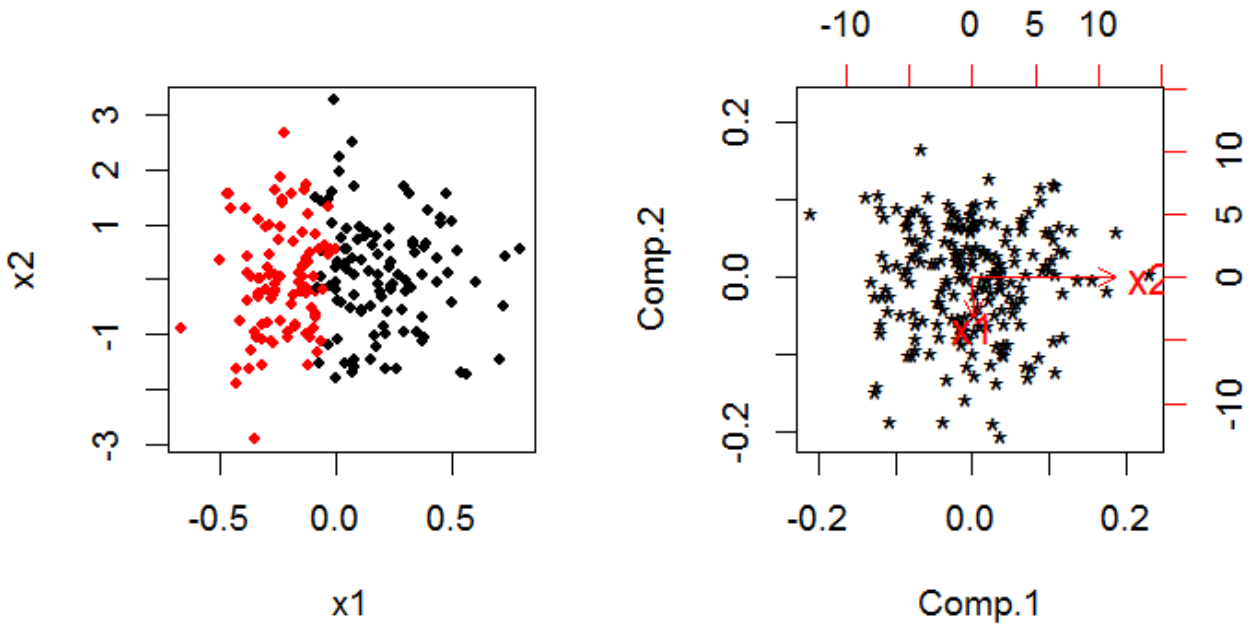

Figure 1: Direction of the two maximum eigenvales yield the maximum variance

□

Exercise 4. What are the two preprocessing steps that should be applied before doing PCA?

Proof. In order to use PCA to its fullest, you need to center your data (subtract data points via the mean μ) then scale your features.

Example. Suppose you have a dataset arranged as a set of n data vectors $\mathbf{x}_1 \dots \mathbf{x}_n$ with each $\mathbf{x}_i$ representing a single grouped observation of the p variables. So, the row vectors are placed into a single matrix $\boldsymbol{X}$ of dimensions $n \times p$.

Next, we find the empirical mean along each column $j = 1, \dots, p$ and place the mean values into a vector $\boldsymbol{\mu}$ of dimensions $p \times 1$:

$$\mu_j = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{X}_{ij}.$$

Now, we subtract the mean of each column from the original matrix $\boldsymbol{X}$ in order to yield matrix $\boldsymbol{B}$:

$$\boldsymbol{B} = \boldsymbol{X} - \boldsymbol{\mu}^\top.$$

Note that matrix $\boldsymbol{\mu}^\top$ is broadcasted so that its number of rows matches matrix $\boldsymbol{X}$.

The next step is to scale our features so their unit norm is 1 and so that each dimension is in the same range. There are a few methods for feature scaling, but these two are the most common:

One is mean normalization: divide each column i of matrix $\boldsymbol{B}$ by the difference between the max value and the minimum value of i.

$$\boldsymbol{A} = \frac{\boldsymbol{B}_{:,i}}{\max(\boldsymbol{B}_{:,i}) - \min(\boldsymbol{B}_{:,i})}. \tag{2}$$

The other is standardization: divide each column i of matrix $\boldsymbol{B}$ by the standard deviation of i.

$$\boldsymbol{A} = \frac{\boldsymbol{B}_{:,i}}{\sqrt{\mathrm{Var}[\boldsymbol{B}_{:,i}]}}, \tag{3}$$

which is just another way of saying

$$\boldsymbol{A} = \mathrm{Var}[\boldsymbol{B}] = \frac{1}{n-1} \boldsymbol{B}^\top \boldsymbol{B}, \tag{4}$$

which is the unbiased sample covariance matrix associated with $\boldsymbol{B}$. After these two steps, PCA can now be used.

Note, that if $\boldsymbol{B}$ is composed of complex number, use conjugate transpose instead of just transpose.

□

Exercise 5. Describe the curse of dimensionality with examples.

Proof. Many machine learning problems become exceedingly difficult when the number of dimensions in the data is high. This phenomenon is known as the **curse of dimensionality**. Of particular concern is that the number of possible distinct configurations of a set of variables increases exponentially as the number of variables increases.

□

Reduction.png

Figure 2: As the number of relevant dimensions of the data increases (from left to right), the number of congurations of interest may grow exponentially. *(Left)* In this one-dimensional example, we have one variable for which we only care to distinguish 10 regions of interest. With enough examples falling within each of these regions (each region corresponds to a cell in the illustration), learning algorithms can easily generalize correctly. A straightforward way to generalize is to estimate the value of the target function within each region (and possibly interpolate between neighboring regions). *(Center)* With two dimensions, it is more dicult to distinguish 10 dierent values of each variable. We need to keep track of up to $1010 = 100$ regions, and we need at least that many examples to cover all those regions. *(Right)* With three dimensions, this grows to $10^3 = 1,000$ regions and at least that many examples. For d dimensions and v values to be distinguished along each axis, we seem to need $O(v^d)$ regions and examples. This is an instance of the curse of dimensionality. Figure graciously provided by Nicolas Chapados

Exercise 6. What is local constancy prior or smoothness prior or regularization?

Proof. In order to generalize well, machine learning algorithms need to be guided by prior beliefs about what kind of function they should learn. Previously, we have seen these priors incorporated as explicit beliefs in the form of probability distributions over parameters of the model. More informally, we may also discuss prior beliefs as directly influencing the function itself and only indirectly acting on the parameters via their effect on the function. Additionally, we informally discuss prior beliefs as being expressed implicitly, by choosing algorithms that are biased toward choosing some class of functions over another, even though these biases may not be expressed (or even possible to express) in terms of a probability distribution representing our degree of belief in various functions. Among the most widely used of these implicit "priors" is the smoothness prior or local constancy prior. This prior states that the function we learn should not change very much within a small region. □