Foundations + Context

Dimension Reduction

Dimension reduction is a common technique used for working with large data sets with the goal moving data from a high-dimensional feature space to a low-dimensional feature space while retaining key features of the original data. Approaches to dimension reduction can generally broken into one of two categories; feature selection and feature extraction. Feature selection techniques remove unneeded features from the data. For example, feature selection before building a regression model will often include remove predictors with low correlation to the response variable and predictors with high correlation. Feature extraction techniques will create a new set of features that capture relevant information about the original data. For example, a feature extraction technique may create linear combinations of our original predictors. It is important to note that feature selection will return a subset of variables from the original data set while feature extraction will return a set of new variables.

Curse of Dimensionality

At first, dimension reduction may seem like a counter intuitive approach to take when working with data. It seems that the more features we have, the more we should be able to learn about our data. However, when working with large datasets, the curse of dimensionality comes into play. The curse of dimensionality is a term used to refer to a variety of issues that working with high dimensional data presents. As the dimension of feature space increases, data tends to become increasingly sparse making trends in data more difficult to recognize. One way to think of how this problem manifests is to consider how the ratio of the volume of ball and the hypercube containing the ball changes as the dimension d increases.

The volume of a ball, B, of radius r in n dimensions is given by

$$\frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)}r^n$$

The volume of the hypercube,C, containing this ball (with sides of length $2r$) is given by

$$(2r)^n$$

Taking the ratio we have

$$\frac{\text{Vol}(B)}{\text{Vol}(C)} = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)\cdot 2^n} \text{ so } \lim_{n\to\infty}\frac{\text{Vol}(B)}{\text{Vol}(C)} = 0$$

Consider the case where $r = 1$,

```
library(gt)

ratio_df <- data.frame(n = c(1,2,5,10,25,100), Ratio = c('1', '0.7854', '0.1645', '2.49*e-3', '2.854e-1

gt(ratio_df)
```

n	Ratio
1	1
2	0.7854
5	0.1645

10	2.49*e-3
25	2.854e-11
100	1.868e-70

Thus, we can expect the number of points within distance 1 of a point will decrease as the dimension of data increases.

Another problem that arises from high dimensional data is computational time. The computation complexity of fitting a linear regression model to an $n \times p$ matrix of predictors is $O(np^2 + p^3)$ (many programs will not perform the full matrix inversion so the complexity is usually smaller in practice however the general principle of computational complexity increasing holds). This will quickly balloon for large p. More troubling in high dimensions is that for p predictors, there are 2^p possible combinations of predictors making techniques such as *Best Subset Selection* computationally prohibitive.

Principal Component Analysis

Principal component analysis(PCA) is a commonly used unsupervised learning technique. PCA works by finding the principle components of a dataset, these can be thought of as the direction along which the data varies the most in the feature space. For those of you with a background in linear algebra, the principal components will be the eigenvectors of the covariance matrix in order of the norm of the eigenvalues. PCA is one of the most commonly used dimension reduction techniques as it is often able to capture a large amount of the variation in a data set using only a few features. PCA returns a series of principal components, these can be thought of as vectors in the original feature space. Principal components will be orthogonal to each other so the variance explained by each component will be unique. Given an $n \times p$ data set, n principal components can be created where each principal component is a vector of length p. The algorithm for finding the first principal component of a standardized $n \times p$ data matrix $\mathbf{X}$ is as follows:

$$\mathbf{w}_1 = \text{argmax}_{||\mathbf{w}||=1}\left\{\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}\right\}$$

Note that $\mathbf{X}^T\mathbf{X}$ is the covariance matrix for $\mathbf{X}$. For all following components, we will find the matrix $\hat{\mathbf{X}}_k$ such that all variance explained by the first $k-1$ principal components is removed. This is done by finding

$$\hat{\mathbf{X}}_k = \mathbf{X} - \sum_{j=1}^{k-1} \mathbf{X}\mathbf{w}_j\mathbf{w}_j^T$$

We then use the same equation used to find $\mathbf{w}_1$ to find the kth principle component, replacing $\mathbf{X}$ with $\hat{\mathbf{X}}_k$. Note that principal components can often be used for latent variables.

Linear Regression

Linear regression is perhaps the most commonly used regression method given its simplicity and that it often yields fairly well performing models. Many other regression techniques use linear regression as a foundation. Linear regression works to find the line in n space that minimizes the Mean Squared Error (the squared distance between the line and observed data). The MSE is used in order to ensure that there is a unique answer. In linear regression we are looking for terms $\beta_0, ..., \beta_n$ such that

$$\sum_{i=1}^{p}(y_i - \hat{y_i})^2$$

is minimized where

$$\hat{y} = \beta_0 + \beta_1 \cdot x_1 + ... + \beta_n x_n$$

Note that we are working with a $n+1 \times p$ data frame with n predictors, 1 response variable, and p observations.

We can find the vector of regression coefficients $\boldsymbol{\beta}$ using linear algebra by solving the equation

$$\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}\mathbf{Y}$$

where $\mathbf{X}$ is an $n \times p$ matrix of predictors and $\mathbf{Y}$ is a $1 \times p$ matrix of the response variable.

Alternatively, we can find β_i as follows,

$$\beta_i = \frac{\langle \mathbf{x}_i, \mathbf{y} \rangle}{\langle \mathbf{x}_i, \mathbf{x}_i \rangle}$$

Principal Component Regression

Principal Component Regression (PCR) is a regression technique that uses principal components as predictors. To perform PCR, one first finds the desired number of principal components and then performs linear regression using the selected principal components as predictors. Letting $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_k$ be the first k principal components of our predictors $\mathbf{X}$. This will take the form

$$\mathbf{y} = \theta_0 + \sum_{i=1}^{k} \theta_i \mathbf{w}_i$$

where

$$\theta_i = \frac{\langle \mathbf{w}_i, \mathbf{y} \rangle}{\langle \mathbf{w}_i, \mathbf{w}_i \rangle}$$

For high dimensional data, PCR can avoid problems of over fitting that occur with a large number of predictors. Furthermore, it can provide more interpretable models if the principal components can be linked to latent variables. PCR does have some drawbacks, since principal components only depend on variance within predictors, PCR can miss predictors that contribute little to variance among predictors but strongly explain variance among the response variable.

Partial Least Square

Partial Least Square (PLS) is a regression technique with similarities to PCR. Unlike PCR, PLS considers the covariance matrix instead of the variance matrix. By using the covariance matrix, we ensure that the latent variables of each order will have the most possible correlation with the response variable. For example, the first principle component may only be able to explain most of the variance in the predictors but very little in the response whereas the first component in PLS will explain as much of the variance in the response as possible. This allows it to make more accurate predictions than PCR when using models of the same complexity.

Like PCA, PLS also require all variables to be standardized with $\mu = 0$ and $\sigma^2 = 1$. We then find the mth latent variable $\mathbf{z}_m$ using the following formula

$$\mathbf{z}_m = \sum_{i=1}^{p} \hat{\varphi}_{mi} \mathbf{x}_j^{(m-1)}$$

where $\hat{\varphi}_{mi} = \langle \mathbf{x}_j^{(m-1)}, \mathbf{y} \rangle$.

We then find the regression coefficient $\hat{\theta}_m$ for $\mathbf{z}_m$ as follows

$$\hat{\theta}_m = \frac{\langle \mathbf{z}_m, \mathbf{y} \rangle}{\langle \mathbf{z}_m, \mathbf{z}_m \rangle}$$

Note that this is identical to the formula used in classic linear regression and PCR.

Sections to add PCA +Linear Regression? PCR PLS L^1 Sparsification PLS1 and PLS2 + algorithms Multivariate regression

Write about data

Why Partial Least Squares?

Regression models that use latent variables can help with the curse of dimensionality when building models for high dimensional data. When $n < p$ (the number of predictors is larger than the sample size) or $n \approx p$ many traditional regression methods will lose predictive power as they become increasingly susceptible to noise in the data. For example, using single variate least squares regression when $n < p$ is extremely likely to over fit the data. Furthermore, solutions that minimize the squared error will no longer be unique.

Models that use latent variables will be able to avoid this problem by projecting the data into a lower dimensional feature space so that $n > p$. Since we project from the full feature space less information about the data will be lost than by simply performing variable selection.

Sparse partial least squares further mitigates problems of over fitting the data as it...