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. The feature space is the mathematical space which our data resides in. For example, if our data consists of observations of 10 variables that take real values, our feature space will be $\mathbb{R}^{10}$. By moving data to a lower dimensional feature space, we make the data more interpretable and easier to work with. Furthermore, we can also remove features of the data that aren't relevant as these features may interfere with relevant information. 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. Commonly used dimension reduction tehcniques include best subset selection, LASSO, PCA, and adhoc variable selection. There is no single standard for deciding when and how to perform dimension reduction as it depends on a number of factors including but not limited to the size of the dataset, the methods being used, and the amount of precision needed in results. Dimension reduction is commonly used with large datasets to remove features that are mostly noise and to make computations less costly. In addition, dimesnion reduction can be used to make models more interpretable. For example, when performing linear regression with a data set originally containing 25 predictors one may select only 5 of the predictors to include in the model in order for the model to be simpler and easier to interpret.

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 issues that arise due to the size of the feature space. 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 the ball has radius 1,

n	Ratio
1	1
2	0.7854
5	0.1645
10	2.49*e-3
25	2.854e-11
100	1.868e-70

Table showing how the ratio of the volume of a ball to the volume of a hybercube decreases as the number of dimensions increases.

Thus, we can expect the number of points within distance 1 of a point will decrease to 0 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.

Working with high dimensional data presents a number of issues which make it more difficult to extract meaningful finding from data. By using dimension reduction techniques we can mitigate many of these issues however many feature selection techniques will be less successful as they require us to discard a large amount of the data and may have a hard time finding the most meaningful features. Instead, we may want to turn to extraction methods that project the data into a lower dimensional space while still maintaining as much of the structure as we can.

Principal Component Analysis

Principal component analysis(PCA) is a commonly used unsupervised learning technique that can be used to reduce the dimension of data. 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. These principal components will preserve as much of the structure of the data as possible while projecting it into lower dimensions. 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.

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

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.

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.

L_1 Sparsification

L_1 sparsification is a common shrinkage method used to avoid over fitting models. This method is referred to as adding an L_1 penalty, as a new term is introduced that penalizes the model for being overly complex. This penalty is generated using the L_1 norm, commonly known as the absolute value. The added penalty encourages the model to only give high weight (usually in the term of coefficients) to terms that have high predictive power in the model. The weighting of terms with lower predictive power is lowered ro shrunk thus this is called a shrinkage method.

Unlike many other common shrinkage methods such as the L_2 norm, using the L_1 norm will also perform variable selection. This is due to the fact that the L_1 norm will sometimes send the weighting of terms to 0, completely ignoring them in the model. Since some terms are eliminated from the model, model built using the L_1 norm will often be simpler as they depend on fewer terms. This is useful when working with high dimensional data sets as removing predictors will help with the curse of dimensionality.

One difficulty with using the L_1 norm is that it introduces another parameter, λ into the model that must be selected for. λ is often referred to as the tuning parameter and decides how much model complexity should be penalized. Higher values of λ will more heavily penalize the model causing fewer terms to be included and more shrinkage. The tuning parameter is usually selected using cross validation.

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 will perform variable shrinkage and selection. Features of the data that do not have much predictive power won't be included in the final model or their impact will be greatly reduced.

Literature Review

Partial Least Squares was first proposed by Herman Wold in 1975 under the name Non-linear Iterative Partial Least Squares (NIPALS) in order to model complex datasets. Later models built using the framework of NIPALS would more simply be referred to by the name Partial Least Squares(PLS). Wold would later suggest that the model be referred to as Projection onto Latent Structures as he found this name to more accurately describe the model. Parameters for the model are found as the result of bivariate least squares linear regression using a latent variable believed to describe the underlying relation between predictor and response variables. These latent variables are then iterated over to estimate model parameters.

PLS models always contain one parameter, K, the number of latent variables included in the model. Some variants allow for manual selection of K while others use their own criteria to select the number of latent variables included in the model. Latent variables corresponding to higher values of K will explain less of the relationship between the predictors and response. Ideally K is low as this means that the relation between predictors and the response can be described using a small number of latent variables.

PLS and variants have shown to be some of the best performing models for working with high dimensional data. This has lead to its widespread use in fields such as chemometrics, genomics, spectrometry, as well as others that work with data where the number of predictors is close to or greater than the number of observations.

PLS models generally follow a five step process as outlined by Lorenzo et al.

a. Estimate the covariance matrix between the predictor ($\mathbf{X}$) and response ($\mathbf{Y}$) variables.

b. Estimate the singular-space associated with the largest singular-value of the previous matrix.

c. Project the covariate and response parts on the previously defined subspace.

d. Estimate the linear regression matrix of the response part on the covariate part in the subspace.

e. Remove the information carried by the current subspace from the covariate and response parts

Although not a problem unique to PLS, model consistency is shown to decrease as the ratio of predictors to observations increases. This is due to models tendency to overfit the data as they are unable to distinguish between noise and the underlying relationships of variables. In order to improve on these problems of overfitting, L_1 penalization of regression coefficients was introduced by Tibshirani in 1995 with the Lasso method. The Least Absolute Shrinkage and Selection Operator(Lasso) model is based on Ordinary Least Squares Regression(OLS) and introduces a penalty term to enforce sparsity among predictors.

The basic idea of penalization is to discourage models from including larger coefficients for terms that only explain a small amount of variance. This prevents models from overfitting data as the model will give less weight to terms that only explain noise in the data when trained on a sufficiently large sample. Unlike similar models with a penalty term such as Ridge Regression, Lasso also performs variable selection.

The Lasso model contains one parameter, λ, which takes positive real numbers as values and decide how heavily the model will be penalized. When 0 is chosen, the model is identical to partial least squares. As λ increases, the model will become increasingly sparse. When $\lambda \to \infty$, the model will become $\hat{y} = \beta_0 = \bar{X}$ as all coefficients other than β_0 are penalized. Unlike other shrinkage methods, Lasso also performs variable selection as the L_1 norm will send coefficients to 0 for sufficiently large λ. This means that Lasso tends to select simpler more interpretable models than similar techniques. Parameter selection is usually performed through cross validation, a method originally proposed by Tibshirani.

Sparse partial least squares(SPLS) is a PLS method that uses the L_1-norm to penalize PLS models and thus avoid overfitting the data. SPLS was first formulated in 2007 by Chun and Keles and published in 2010. Their formulation was heavily inspired by the recently proposed technique of sparse principal component analysis. Their model works by finding a surrogate vector close to the original direction vector and imposing sparsity through the L_1-norm on this vector. The SPLS model has four parameters, $(K, \kappa, \lambda_1, \lambda_2)$. K is the number of components, $\kappa \in [0, 1]$ and decides how far the starting surrogate vector can be from the original direction vector, λ_1 determines the severity of the L_1 penalty, and λ_2 determines the severity of the L_2 penalty.

Functionally, the SPLS only has two parameters that need to be tuned, K and λ_1. Values of $\kappa < \frac{1}{2}$ avoid local solution problems while offering little variation in the final model. Chun and Keles suggested that several values of $\kappa < \frac{1}{2}$ be tried but did not offer a formal proposal for selecting an ideal value. λ_2 is always tuned to ∞ in order to preserve the soft-thresholding solution. λ_1 and K are tuned using cross validation.

Referring to the 5 step process of PLS algorithms outline by Lorenzo, we can write the algorithm for SPLS such that only the second step(b) is different depending on our formulation.

Another sparse partial least squares(this time denoted as sPLS) algorithm was suggested by Lê Cao, Rossow, Robert-Granié, and Besse. The sPLS algorithm depends on initializing the the singular vector $\mathbf{v}$ and iterating until the relation between the two singular vectors, $\mathbf{u}$ and $\mathbf{v}$, converges. Each of these singular vectors has a corresponding parameter, $\lambda_{\mathbf{u}}$ and $\lambda_{\mathbf{v}}$. This leads to a total of $2K + 1$ parameters(note that K is one of the parameters) as there are two additional parameters that must be tuned for each component. We should note that sPLS only performs variable selection among predictors.

Data-driven sparse partial least squares(ddsPLS) is a new method introduced in 2021 by Lorenzo, Cloarec, Thiébaut, and Saracco. Unlike previous sparse PLS methods, ddsPLS directly penalizes the empirical covariance matrix of the data. This is where the "data-driven" part of the name comes from the fact that the covariance matrix is more closely related to the original data than the eigenvectors of the covariance matrix which are normally penalized.

Unlike other sparse PLS methods, ddsPLS only has $K + 1$ parameters, K and $\Lambda_K = \lambda_k, k = 1, ..., K$. Since the covariance matrix is directly penalized, we only need one tuning parameter per component with λ_k being the tuning parameter for the kth component. Each λ_k is tuned through bootstrapping, we can either select using $R^2 - Q^2$ or Q^2 metric with the former being recommended.

The use of bootstrapping for parameter selection in the ddsPLS model differs from practice for similar models. Lorenzo justifies the choice by saying that "use of bootstrap is interesting to enrich the underlying information of the available data". Bootstrapping was developed by Efron in 1979 and has become one of the most important techniques in statistics with the increased availability of high speed computers. Typically K-fold cross validation is used for parameter selection with K varying based on the sample size and other features of the data. In his study of model selection techniques, Kohavi recommended 10-fold stratified cross validation for model selection.

ddsPLS and other PLS models often use the Q^2 metric to assess model performance. Q^2 is closely related to the more common R^2 metric as both are calculated using the formula $1 - \frac{\text{RSS}}{\text{TSS}}$ where RSS is the residual sum of squares and TSS is the total sum of squares. R^2 is calculated using the data the model is built using while Q^2 is calculated using data that the model isn't built on. Typically this is done as part of cross validation with Q^2 calculated using the excluded data. The Q^2 metric was introduced by Stone and Gessier while Tennehaus first suggested it uses to assess PLS models.

Despite the similarity of the R^2 and Q^2 metrics, the two metrics will improve under different conditions for the model. Maximizing R^2 will favor more complex models that closely fit the training data while maximizing Q^2 will favor less complex models that address the underlying structure of the data. Initially, increasing model complexity will improve both metrics. However, past a certain threshold, increasing model complexity will cause Q^2 to decrease while R^2 to continues to increase. Due to this, minimizing $R^2 - Q^2$ is recommended for selecting values of λ_k for the ddsPLS model.