High dimensional data

PCA, PLS

Ramesh Srinivasan

October 24, 2024

High Dimensional Data

- When the number of features p is as large or larger than the number of observations n you are going to overfit the data.
- For example with p = 1 and n = 2 a simple least squares regression will perfectly fit the data, but is unlikely to fit test data very well (see Figure 6.22)
- Some form of dimensionality reduction such that p < n will lead to better predictive models.
- PCA and PLS are two forms of dimensionality reduction

Regression Model - Centered Data

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

- $\boldsymbol{X}$ is an n $\times$ (p) matrix
- $\boldsymbol{\beta}$ is an p-dimensional vector
- $\boldsymbol{Y}$ is an n-dimensional vector
- $\boldsymbol{\epsilon}$ is an n-dimensional vector
- where n is the number of data points and p is the number of predictors

$$\boldsymbol{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ x_{2,1} & x_{2,2} & \dots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

PC Regression Model

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{V}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

- $\boldsymbol{X}$ is an n $\times$ (p) matrix
- $\boldsymbol{V}$ is an p $\times$ (c) matrix, which are the eigenvectors of PCA
- $\boldsymbol{\beta_v}$ is an c-dimensional vector
- $\boldsymbol{Y}$ is an n-dimensional vector
- ϵ is an n-dimensional vector
- where n is the number of data points and p is the number of predictors

$$\boldsymbol{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ x_{2,1} & x_{2,2} & \dots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{bmatrix}, \quad \boldsymbol{V} = \begin{bmatrix} v_{1,1} & v_{1,2} & \dots & v_{1,c} \\ v_{2,1} & v_{2,2} & \dots & v_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ v_{p,1} & v_{p,2} & \dots & v_{p,c} \end{bmatrix}, \quad \boldsymbol{\beta_v} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_c \end{bmatrix}$$

Although rarely done, its worth noting,

$$\hat{\boldsymbol{\beta}} = \boldsymbol{V}\boldsymbol{\beta_v}$$

PLS Regression Model

The PLS Regression model is based on the concept of cross-decomposition.
It can be used to predict multivariate Y in addition to univariate Y.
A related method that has sadly falled into some disfavor is canonical correlation.
There are 3 steps to a PLS Regression model

- Compute a predictor/target cross-correlation matrix, i,e,.

$$\boldsymbol{C} = \boldsymbol{X}^T \boldsymbol{Y}$$

- Find the direction with the strongest covariance, usually by a SVD of the cross correlation matrix.
- The left singular vectors (u) will give you a loading on X and the right singular vectors give you a loading on Y.
- Deflate (remove the dimension) corresponding to the first left singular vector in X. Depending on approach to Y, perhaps deflate Y also.
- Rinse and repeat.

PLS Regression

$$\boldsymbol{Y} = \boldsymbol{XU}\beta_{\boldsymbol{PLS}} + \epsilon$$

- $\boldsymbol{X}$ is an n $\times$ (p) matrix
- $\boldsymbol{U}$ is an p $\times$ (c) matrix which are the left singular vectors of the cross-covariance matrix between X and Y
- $\beta_{\boldsymbol{PLS}}$ is an c-dimensional vector
- $\boldsymbol{Y}$ is an n-dimensional vector
- ϵ is an n-dimensional vector
- where n is the number of data points and p is the number of predictors

$$\boldsymbol{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ x_{2,1} & x_{2,2} & \dots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,c} \\ u_{2,1} & u_{2,2} & \dots & u_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ u_{p,1} & u_{p,2} & \dots & u_{p,c} \end{bmatrix}, \quad \beta_{\boldsymbol{PLS}} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_c \end{bmatrix}$$

Although rarely done, its worth noting,

$$\hat{\beta} = \boldsymbol{U}\beta_{\boldsymbol{PLS}}$$

Closing Thoughts

- if $n > p$ Ridge Regression/Classifier will always produce the most stable models with the best prediction performance. If all you want to do is predict, do this.
- If $n > p$ Lasso Regression/Classifier produces the most interpretable model with a reduced set of predictors. Unless your underlying model is genuinely sparse, its unlikely you will outperform Ridge with Lasso.
- if $n < p$ PCA Regression/Classification gets you a model that works. I do not believe you can meaningfully interpret these components.
- if $n < p$ PLS Regression/Classifier takes advantage of cross-decomposition to produce a more compact model than PC Regression. The components may be more interpretable.