Supervised Learning: Regression, Part II

Noah Simon & Daniela Witten

July 15-17, 2015
Summer Institute for Statistics of Big Data
University of Washington

Linear Models in High Dimensions

- When p is large, least squares regression will lead to very low training error but terrible test error.
- We will now see some approaches for fitting linear models in high dimensions, $p \gg n$.
- These approaches also work well when $p \approx n$ or $n > p$.

Motivating example

- We would like to build a model to predict survival time for breast cancer patients using a number of clinical measurements (tumor stage, tumor grade, tumor size, patient age, etc.) as well as some biomarkers.
- For instance, these biomarkers could be:
  - the expression levels of genes measured using a microarray.
  - protein levels.
  - mutations in genes potentially implicated in breast cancer.
- How can we develop a model with low test error in this setting?

Remember

- We have n training observations.
- Our goal is to get a model that will perform well on future test observations.
- We'll incur some bias in order to reduce variance.

Variable Pre-Selection

The simplest approach for fitting a model in high dimensions:

1. Choose a small set of variables, say the q variables that are most correlated with the response, where $q < n$ and $q < p$.
2. Use least squares to fit a model predicting y using only these q variables.

This approach is simple and straightforward.

Variable Pre-Selection in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
cors <- cor(xtr,ytr)
whichers <- which(abs(cors)>.2)
mod <- lm(ytr~xtr[,whichers])
print(summary(mod))
```

How Many Variable to Use?

- We need a way to choose q, the number of variables used in the regression model.
- We want q that minimizes the test error.
- For a range of values of q, we can perform the validation set approach, leave-one-out cross-validation, or K-fold cross-validation in order to estimate the test error.
- Then choose the value of q for which the estimated test error is smallest.

Estimating the Test Error For a Given q

This is the right way to estimate the test error using the validation set approach:

1. Split the observations into a training set and a validation set.
2. Using the training set only:
   a. Identify the q variables most associated with the response.
   b. Use least squares to fit a model predicting y using those q variables.
   c. Let $\hat{\beta}_1, \ldots, \hat{\beta}_q$ denote the resulting coefficient estimates.
3. Use $\hat{\beta}_1, \ldots, \hat{\beta}_q$ obtained on training set to predict response on validation set, and compute the validation set MSE.

Estimating the Test Error For a Given q

This is the wrong way to estimate the test error using the validation set approach:

1. Identify the q variables most associated with the response on the full data set.
2. Split the observations into a training set and a validation set.
3. Using the training set only:
   a. Use least squares to fit a model predicting y using those q variables.
   b. Let $\hat{\beta}_1, \ldots, \hat{\beta}_q$ denote the resulting coefficient estimates.
4. Use $\hat{\beta}_1, \ldots, \hat{\beta}_q$ obtained on training set to predict response on validation set, and compute the validation set MSE.

Frequently Asked Questions

- **Q:** Does it really matter how you estimate the test error?
  A: Yes.
- **Q:** Would anyone make such a silly mistake?
  A: Yes.

A Better Approach

- The variable pre-selection approach is simple and easy to implement – all you need is a way to calculate correlations, and software to fit a linear model using least squares.
- But it might not work well: just because a bunch of variables are correlated with the response doesn't mean that when used together in a linear model, they will predict the response well.
- What we really want to do: pick the q variables that best predict the response.

Best Subset Selection

- We would like to consider all possible models using a subset of the p predictors.
- In other words, we'd like to consider all 2^p possible models.
- This is called best subset selection.
- Unfortunately, this is computationally intractable:
  - When $p = 3$, $2^p = 8$.
  - When $p = 6$, $2^p = 64$.
  - When $p = 250$, there are $2^{250} \approx 10^{80}$ possible models. According to www.universetoday.com, this is around the number of atoms in the known universe.
  - Not feasible to consider so many models!
- Need an efficient way to sift through all of these models: forward stepwise regression.

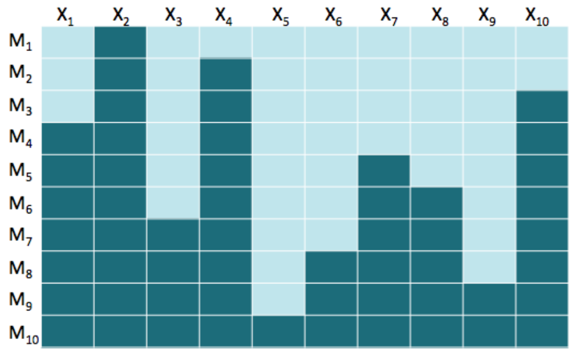
Forward Stepwise Regression

1. Use least squares to fit p univariate regression models, and select the predictor corresponding to the best model (according to e.g. training set MSE).
2. Use least squares to fit $p-1$ models containing that one predictor, and each of the $p-1$ other predictors. Select the predictors in the best two-variable model.
3. Now use least squares to fit $p-2$ models containing those two predictors, and each of the $p-2$ other predictors. Select the predictors in the best three-variable model.
4. And so on....

This gives us a nested set of models, containing the predictors

$$\mathcal{M}_1 \subseteq \mathcal{M}_2 \subseteq \mathcal{M}_3 \subseteq \ldots .$$

Forward Stepwise Regression With $p = 10$



Example in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
library(leaps)
out <- regsubsets(xtr,ytr,nvmax=30,method="forward")
print(summary(out))
print(coef(out,1:10))
```

Which Value of q is Best?

- This procedure traces out a set of models, containing between 1 and p variables.
- The qth model contains q variables, given by the set $\mathcal{M}_q$.
- **Q:** Which value of q is best?
  A: *The one that minimizes the test error!*
- We can select the value of q using cross-validation or the validation set approach.

Drawback of Forward Stepwise Selection

- Forward stepwise selection isn't guaranteed to give you the best model containing q variables.
- To get the best model with q variables, you'd need to consider every possible one; computationally intractable.
- For instance, suppose that the best model with one variable is

  $$y = \beta_3 X_3 + \epsilon$$

  and the best model with two variables is

  $$y = \beta_4 X_4 + \beta_8 X_8 + \epsilon.$$

  Then forward stepwise selection will not identify the best two-variable model.
- **Q:** Does this really happen in practice?
  A: Yes.

How To Do Forward Stepwise?

Wrong: Split the data into a training set and a validation set. Perform forward stepwise on the training set, and identify the model with best performance on the validation set. Then, refit the model (using those q variables) on the full data set.

Right: Split the data into a training set and a validation set. Perform forward stepwise on the training set, and identify the value of q corresponding to the best-performing model on the validation set. Then, perform forward stepwise selection in order to obtain a q-variable model on the full data set.

Bottom Line: We estimate the test error in order to choose the correct level of **model complexity**. Then we refit the model on the full data set.

Let's Try It Out in R!

Chapter 6 R Lab, Part 1
www.statlearning.com

Ridge Regression and the Lasso

- Forward stepwise selection does a discrete search through model space, considering subsets of the predictors, and fitting each of the resulting models using least squares. Model complexity is controlled by using subsets of the predictors.
- Ridge regression and the lasso instead control model complexity by using an alternative to least squares, by shrinking the regression coefficients.
- This is known as regularization or penalization.
- Hot area in statistical machine learning today.

Crazy Coefficients

- When $p > n$, some of the variables are highly correlated.
- Why does correlation matter?
  - Suppose that X_1 and X_2 are highly correlated with each other... assume $X_1 = X_2$ for the sake of argument.
  - And suppose that the least squares model is
  $$\hat{y} = X_1 - 2X_2 + 3X_3.$$
  - Then this is also a least squares model:
  $$\hat{y} = 100000001X_1 - 100000002X_2 + 3X_3.$$
- Bottom Line: When there are too many variables, the least squares coefficients can get crazy!
- This craziness is directly responsible for poor test error.
- It amounts to too much model complexity.

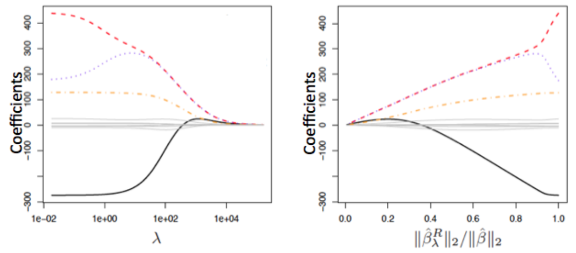
A Solution: Don't Let the Coefficients Get Too Crazy

- Recall that least squares involves finding β that minimizes
$$\|\mathbf{y} - \mathbf{X}\beta\|^2.$$
- Ridge regression involves finding β that minimizes
$$\|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda \sum_j \beta_j^2.$$
- Equivalently, find β that minimizes
$$\|\mathbf{y} - \mathbf{X}\beta\|^2$$
subject to the constraint that
$$\sum_{j=1}^{p} \beta_j^2 \leq s.$$

Ridge Regression

- Ridge regression coefficient estimates minimize
$$\|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda \sum_j \beta_j^2.$$
- Here λ is a nonnegative tuning parameter that shrinks the coefficient estimates.
- When $\lambda = 0$, then ridge regression is just the same as least squares.
- As λ increases, then $\sum_{j=1}^{p} (\hat{\beta}_{\lambda,j}^R)^2$ decreases – i.e. coefficients become shrunken towards zero.
- When $\lambda = \infty$, $\hat{\beta}_\lambda^R = 0$.

Ridge Regression As λ Varies



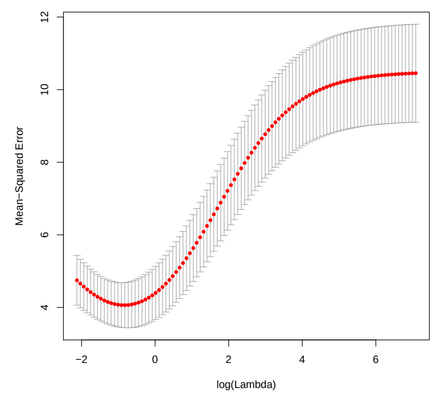
Ridge Regression In Practice

- Perform ridge regression for a very fine grid of λ values.
- Use cross-validation or the validation set approach to select the optimal value of λ – that is, the best level of model complexity.
- Perform ridge on the full data set, using that value of λ.

Example in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
library(glmnet)
cv.out <- cv.glmnet(xtr,ytr,alpha=0,nfolds=5)
print(cv.out$cvm)
plot(cv.out)
cat("CV Errors", cv.out$cvm,fill=TRUE)
cat("Lambda with smallest CV Error",
cv.out$lambda[which.min(cv.out$cvm)],fill=TRUE)
cat("Coefficients", as.numeric(coef(cv.out)),fill=TRUE)
cat("Number of Zero Coefficients",
sum(abs(coef(cv.out))<1e-8),fill=TRUE)
```

R Output



Drawbacks of Ridge

- Ridge regression is a simple idea and has a number of attractive properties: for instance, you can continuously control model complexity through the tuning parameter λ.
- But it suffers in terms of model interpretability, since the final model contains all p variables, no matter what.
- Often want a simpler model involving a subset of the features.
- The lasso involves performing a little tweak to ridge regression so that the resulting model contains mostly zeros.
- In other words, the resulting model is sparse. We say that the lasso performs feature selection.
- The lasso is a very active area of research interest in the statistical community!

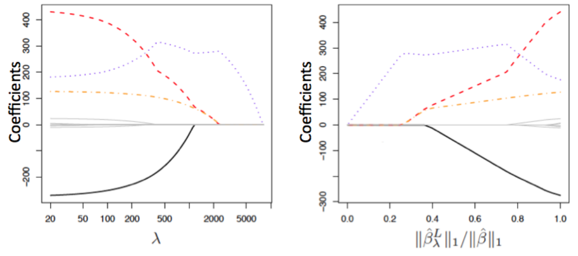
The Lasso

- The lasso involves finding β that minimizes
$$\|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda \sum_j |\beta_j|.$$
- Equivalently, find β that minimizes
$$\|\mathbf{y} - \mathbf{X}\beta\|^2$$
subject to the constraint that
$$\sum_{j=1}^{p} |\beta_j| \leq s.$$
- So lasso is just like ridge, except that β_j^2 has been replaced with $|\beta_j|$.

The Lasso

- Lasso is a lot like ridge:
  - λ is a nonnegative tuning parameter that controls model complexity.
  - When $\lambda = 0$, we get least squares.
  - When λ is very large, we get $\hat{\beta}_\lambda^L = 0$.
- But unlike ridge, lasso will give some coefficients exactly equal to zero for intermediate values of λ!

Lasso As λ Varies



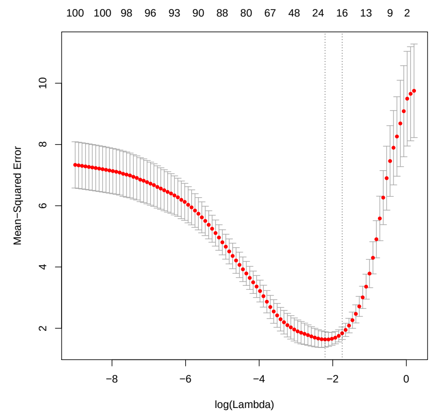
Lasso In Practice

- Perform lasso for a very fine grid of λ values.
- Use cross-validation or the validation set approach to select the optimal value of λ – that is, the best level of model complexity.
- Perform the lasso on the full data set, using that value of λ.

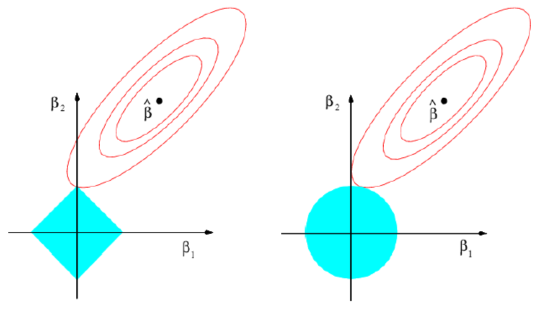
Example in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
library(glmnet)
cv.out <- cv.glmnet(xtr,ytr,alpha=1,nfolds=5)
print(cv.out$cvm)
plot(cv.out)
cat("CV Errors", cv.out$cvm,fill=TRUE)
cat("Lambda with smallest CV Error",
cv.out$lambda[which.min(cv.out$cvm)],fill=TRUE)
cat("Coefficients", as.numeric(coef(cv.out)),fill=TRUE)
cat("Number of Zero Coefficients",sum(abs(coef(cv.out))<1e-8),
fill=TRUE)
```

R Output



Ridge and Lasso: A Geometric Interpretation



Let's Try It Out in R!

Chapter 6 R Lab, Part 2
www.statlearning.com

Review

- So far we have seen two approaches that select subsets of the features and fit a least squares model:
  - Variable Pre-Selection
  - Forward Stepwise Selection
- And we have seen two approaches that fit a shrunken model instead of using least squares:
  - Ridge regression
  - Lasso
- Now we see one final approach, principal components regression, that first finds a low-dimensional subspace of the data and then fits a model on that low-dimensional subspace, using least squares.

Principal Components Regression

- Our data consist of n observations in a p-dimensional space.
- However, not all of those p dimensions are equally useful, especially when $p \gg n$.
- Many are either completely redundant (correlated features) or uninformative (noise features).
- Can we find a low-dimensional representation of the variables that captures most of the variability in the data?
- This is a dimension reduction approach.

PCR

- Let $Z_1, Z_2, \ldots, Z_M$ represent $M < p$ linear combinations of the p predictors:

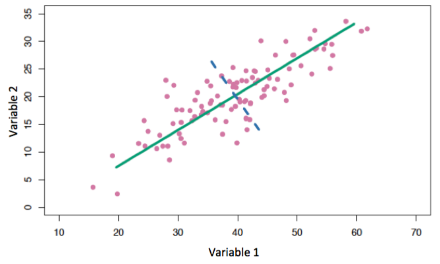
$$Z_m = \sum_{j=1}^{p} \phi_{mj} X_j.$$

- Use least squares to fit the model

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m Z_{im} + \epsilon_i, \quad i = 1, \ldots, n.$$

- In other words, we perform least squares using M new predictors, $Z_1, \ldots, Z_M$.
- $Z_1, \ldots, Z_M$ chosen to be the principal components of the data.

Principal Components, Conceptually



- PCs are the linear combinations of the variables that contain as much as possible of the variability in the features.
- Will be discussed further in SISBID Module 4 — Unsupervised Learning.

PCR

Our final model is linear in the original predictors:

$$\begin{aligned} y_i &= \theta_0 + \sum_{m=1}^{M} \theta_m Z_{im} + \epsilon_i \\ &= \theta_0 + \sum_{m=1}^{M} \theta_m \sum_{j=1}^{p} \phi_{mj} X_{ij} + \varepsilon_i \\ &= \theta_0 + \sum_{j=1}^{p} \left(\sum_{m=1}^{M} \theta_m \phi_{mj} \right) X_{ij} + \varepsilon_i \end{aligned}$$

More on PCR

- PCR doesn't yield feature selection – all of the original predictors are involved in the final model.
- But when M is small, then PCR can avoid overfitting and can give good results.
- Choose M by cross-validation or validation set approach.
- With $M = p$, we just get least squares regression: no dimension reduction occurs!
- Turns out that PCR is closely related to ridge regression.
- Shortcoming of PCR: the first M principal components are guaranteed to explain a lot of the variation in the features, but that doesn't mean that they are predictive of the response!
- In SISBID Module 4, will see how principal components can be used for unsupervised learning.

Example in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
library(pls)
out <- pcr(ytr~xtr,scale=TRUE,validation="CV")
summary(out)
validationplot(out,val.type="MSEP")
```

Let's Try It Out in R!

Chapter 6 R Lab, Part 3
www.statlearning.com

Pros/Cons of Each Approach

Approach	Simplicity?*	Sparsity?**	Predictions?***
Pre-Selection	Good	Yes	So-So
Forward Stepwise	Good	Yes	So-So
Ridge	Medium	No	Great
Lasso	Bad	Yes	Great
PCR	Medium	No	Great

* How simple is this model-fitting procedure? If you were stranded on a desert island with pretty limited statistical software, could you fit this model?
** Does this approach perform feature selection, i.e. is the resulting model sparse?
*** How good are the predictions resulting from this model?

No “Best” Approach

- There is no “best” approach to regression in high dimensions.
- Some approaches will work better than others. For instance:
  - Lasso will work well if it’s really true that just a few features are associated with the response.
  - Ridge will do better if all of the features are associated with the response.
- If somebody tells you that one approach is “best”... then they are mistaken. Politely contradict them.
- While no approach is “best”, some approaches are wrong (e.g.: there is a wrong way to do cross-validation)!

Predicting Age Using DNA Methylation Data

- Comparison on 6 data sets
- SPC: Like principal components regression, but using a subset of features most associated with response. Between 1 and 3 principal components were used.
- Elastic Net: A hybrid between ridge and lasso.
- SVM: We’ll see it next lecture in the classification context.
- Citation: Zhuang et al., BMC Bioinformatics, 2012

Didn’t I Tell You? No Best Method!

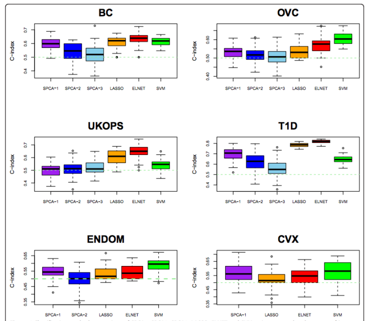


Figure 6 Classification method comparison of SPCA1, SPCA2, SPCA3, LASSO, ELNET and SVM, using age as the phenotype of interest, for each of the 6 DNAm studies with available age information. A total of 50 training-test set partitions were used, so each boxplot contains 50 data points. C-index (y-axis) measures the concordance index between the DNAm age predictor score and age itself. Dashed green line denotes the null hypothesis of no association.

High C-index indicates a low test error.

Bottom Line

Much more important than what model you fit is how you fit it.

- Was cross-validation performed properly?
- Did you select a model (or level of model complexity) based on an estimate of test error?