Advanced Computational Econometrics: Machine Learning

Chapter 3: Model Selection

Thierry Denœux

July 2019

Overview

Need for model selection

- Consider, for instance, a regression problem with a response variable Y and 3 predictors X_1, X_2, X_3.
- We can consider many (an infinity of) models, such as

$$Y = \beta_0 + \beta_1 X_1 + \epsilon$$

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_1^2 + \beta_5 X_2^2 + \beta_6 X_1 X_2 + \beta_7 X_3^2 + \beta_8 X_1 X_3 + \beta_9 X_2 X_3 + \epsilon$$

$$\vdots$$

Which model to choose?

Bias-variance trade-off

- We have seen that a more complex model will not always have a smaller error when applied to test data.
- This is due to the bias-variance trade-off: when the number of parameters increases, the bias of the model decreases, but the variance increases.
- Furthermore, a simpler model often has a distinct advantages in terms of its interpretability.
- In this lecture, we discuss some tools to select models that will be
  - complex enough to fit the data, but
  - not too complex to avoid overfitting and to be interpretable.
- We focus mainly on linear regression, but the tools can be adapted to classification.

Three classes of methods

Subset Selection. We identify a subset of the p predictors that we believe to be related to the response. We then fit a model using the reduced set of variables.

Regularization. We fit a model involving all p predictors, but the estimated coefficients are shrunken towards zero to obtain a smoother prediction function. This regularization (also known as shrinkage) has the effect of reducing variance and can also perform variable selection.

Dimension Reduction. We project the p predictors into a q-dimensional subspace, where $q < p$. This is achieved by computing q different linear combinations, or projections, of the variables. Then these q projections are used as predictors to fit a linear model.

Overview

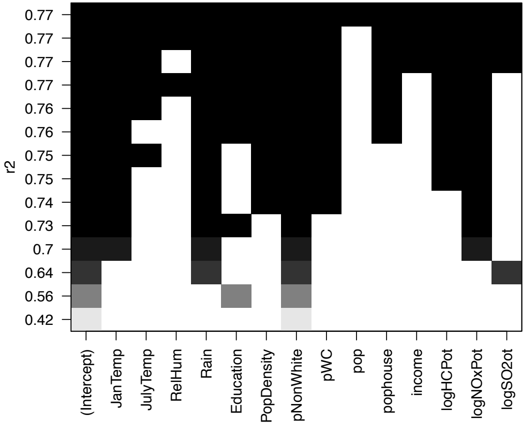
Overview

Best subset selection

1. Let $\mathcal{M}_0$ denote the null model, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, \ldots, p$:
    1. Fit all $\binom{p}{k}$ models that contain exactly k predictors.
    2. Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. Here "best" is defined as having the smallest RSS, or equivalently the largest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$. (How? to be seen later).

Best subset selection in R

```
library('leaps')
reg.fit<-regsubsets(Mortality~.-logNOx,data=pollution,method='exhaustive',nvmax=15)
plot(reg.fit,scale="r2")
```



Extension to other models

- Although we have presented best subset selection here for least squares regression, the same ideas apply to other types of models, such as logistic regression.
- The deviance, $-2\ell(\widehat{\theta})$, plays the role of RSS for a broader class of models.

Stepwise selection

- For computational reasons, best subset selection cannot be applied with very large p.
- Best subset selection may also suffer from statistical problems when p is large: larger the search space, the higher the chance of finding models that look good on the training data, even though they might not have any predictive power on future data.
- Thus an enormous search space can lead to overfitting and high variance of the coefficient estimates.
- For both of these reasons, stepwise methods, which explore a far more restricted set of models, are attractive alternatives to best subset selection.

Forward stepwise selection

- Forward stepwise selection begins with a model containing no predictors, and then adds predictors to the model, one-at-a-time, until all of the predictors are in the model.
- In particular, at each step the variable that gives the greatest additional improvement to the fit is added to the model.

Forward stepwise selection in detail

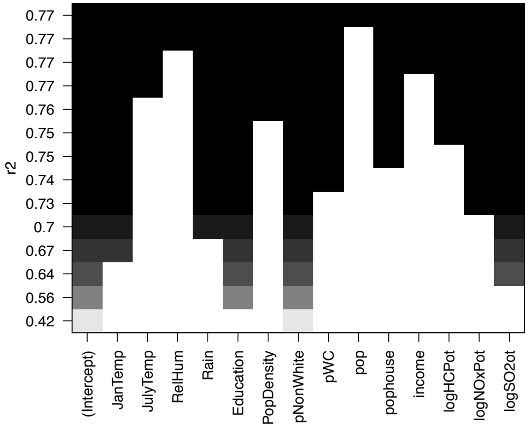
1. Let $\mathcal{M}_0$ denote the null model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
    1. Consider all $p - k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
    2. Choose the best among these $p - k$ models, and call it $\mathcal{M}_{k+1}$. Here "best" is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$. (How? to be seen later).

More on forward stepwise selection

- Computational advantage over best subset selection is clear.
- It is not guaranteed to find the best possible model out of all 2^p models containing subsets of the p predictors.

Forward stepwise selection in R

```
reg.fit<-regsubsets(Mortality~.-logNOx,data=pollution,method='forward',nvmax=15)
plot(reg.fit,scale="r2")
```



Backward stepwise selection

- Like forward stepwise selection, backward stepwise selection provides an efficient alternative to best subset selection.
- However, unlike forward stepwise selection, it begins with the full least squares model containing all p predictors, and then iteratively removes the least useful predictor, one-at-a-time.

Backward stepwise selection in detail

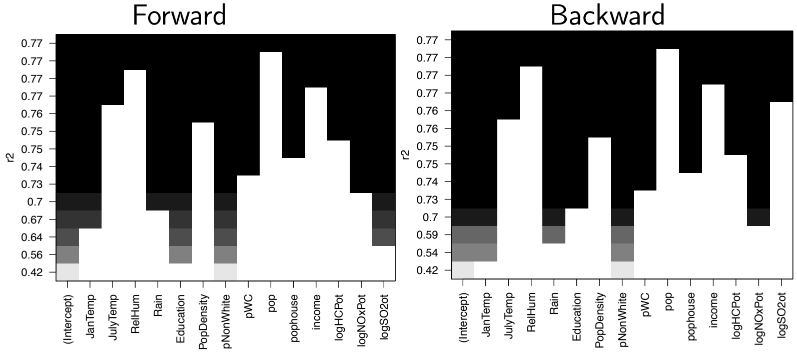
1. Let $\mathcal{M}_p$ denote the full model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
    1. Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
    2. Choose the best among these k models, and call it $\mathcal{M}_{k-1}$. Here "best" is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$. (How? to be seen later).

More on backward stepwise selection

- Like forward stepwise selection, the backward selection approach searches through only $1 + p(p+1)/2$ models, and so can be applied in settings where p is too large to apply best subset selection
- Like forward stepwise selection, backward stepwise selection is not guaranteed to yield the best model containing a subset of the p predictors.
- Backward selection requires that the number of samples n is larger than the number of variables p (so that the full model can be fit). In contrast, forward stepwise can be used even when $n < p$, and so is the only viable subset method when p is very large.

Backward stepwise selection in R

```
reg.fit<-regsubsets(Mortality~.-logNOx,data=pollution,method='backward',nvmax=15)
plot(reg.fit,scale="r2")
```



Overview

Choosing the optimal model

- The model containing all of the predictors will always have the smallest RSS and the largest R^2, since these quantities are related to the training error.
- We wish to choose a model with low test error, not a model with low training error. Recall that training error is usually a poor estimate of test error.
- Therefore, RSS and R^2 are not suitable for selecting the best model among a collection of models with different numbers of predictors.

Estimating test error: two approaches

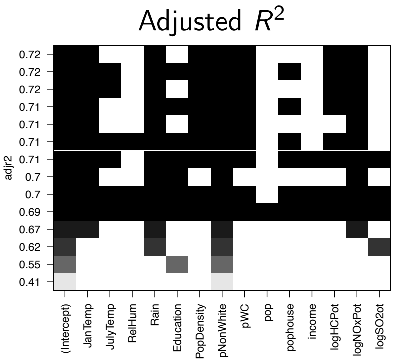
- We can
  1. Indirectly estimate test error by making an adjustment to the training error to account for the bias due to overfitting, or
  2. Directly estimate the test error, using either a validation set approach or a cross-validation approach.
- We illustrate both approaches next.

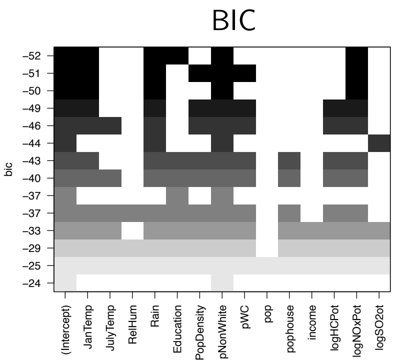
Training error adjustment techniques

- These techniques adjust the training error for the model size, and can be used to select among a set of models with different numbers of variables.
- Three criteria:
  1. Adjusted R^2
  2. Akaike information criterion (AIC)
  3. Bayesian information criterion (BIC)
- The next figure displays BIC, and adjusted R^2 for the best model of each size produced by best subset selection on the pollution data set.

Example

```
reg.fit<-regsubsets(Mortality~.-logNOx,data=pollution,method='exhaustive',nvmax=15)
plot(reg.fit,scale="adjr2") plot(reg.fit,scale="bic")
```





Adjusted R-squared

- Idea: introduce the "population R^2" as

$$R^2_{pop} = 1 - \frac{\sigma^2}{\mathrm{Var}(Y)}$$

- The usual R^2 is

$$R^2 = 1 - \frac{RSS/n}{TSS/n}$$

  It is based on bias estimates of the residual and total variances.

- The adjusted R^2 is based on unbiased estimates:

$$\overline{R}^2 = 1 - \frac{RSS/(n-p-1)}{TSS/(n-1)}$$

  where p is the number of predictors used.

- This criterion is specific to regression.

AIC

- The AIC criterion is defined for a large class of models fit by maximum likelihood:
$$AIC = -2\ell(\widehat{\theta}) + 2r,$$
where $\ell(\widehat{\theta})$ is the maximized value of the log-likelihood function for the estimated model, and r is the number of parameters.
- The best model has the smallest AIC value.
- For linear regression with p variables and a constant term, $r = p + 1$.

BIC

- Definition:
$$BIC = -2\ell(\widehat{\theta}) + r\log(n),$$
where p is the number of parameters.
- Like AIC, BIC will tend to take on a small value for a model with a low test error, and so generally we select the model that has the lowest BIC value.
- Notice that BIC replaces the $2r$ used by AIC with a $r\log(n)$ term, where n is the number of observations.
- Since $\log n > 2$ for any $n > 7$, the BIC statistic generally places a heavier penalty on models with many variables, and hence results in the selection of smaller models than AIC.

Direct estimation of the test error

- Each of the procedures returns a sequence of models $\mathcal{M}_k$ indexed by model size $k = 0, 1, 2, \ldots, p$. Our job here is to select $\widehat{k}$. Once selected, we will return model $\mathcal{M}_{\widehat{k}}$.
- We compute an estimate of the error for each model $\mathcal{M}_k$ under consideration, and then select the k for which the resulting estimated test error is smallest.
- This procedure has an advantage relative to AIC, BIC, and adjusted R^2, in that it provides a direct estimate of the test error.
- It can also be used in a wider range of model selection tasks, even in cases where it is hard to pinpoint the model degrees of freedom.

Direct estimation of the test error

Two methods:

1. Hold-out approach
2. Cross-validation

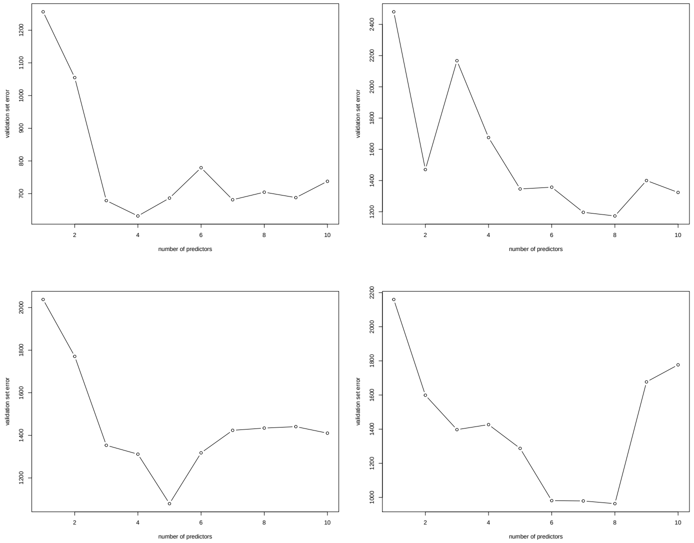
Hold-out approach

- Here we randomly divide the available set of samples into two parts:
  1. a training set and
  2. a validation or hold-out set
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set.
- The resulting validation-set error provides an estimate of the test error. This is typically assessed using MSE in the case of regression and misclassification rate in the case of classification.

Example

```
n<-nrow(pollution)
napp=45
ntst=n-napp
train<-sample(1:n,napp)
pollution.train<-pollution[train,]
pollution.test<-pollution[-train,]

Formula<-c(Mortality ~ pNonWhite,
Mortality ~ Education + pNonWhite,
Mortality ~ Rain + pNonWhite + logSO2ot,
Mortality ~ JanTemp+ Rain +pNonWhite +logNOxPot,
...
)

for(i in 1:10){
reg<-lm(Formula[[i]],data=pollution.train)
pred<-predict(reg,newdata=pollution.test)
err[i]<-mean((pollution.test$Mortality-pred)^2)
}
```

Results with 4 different splits

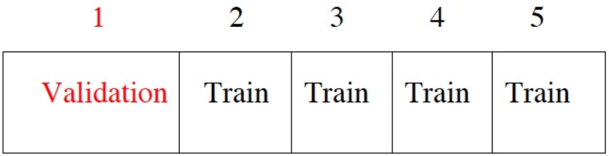


Drawbacks of the hold-out approach

- The validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.
- In the validation approach, only a subset of the observations – those that are included in the training set rather than in the validation set – are used to fit the model.
- This suggests that the validation set error may tend to overestimate the test error for the model fit on the entire data set.

K-fold cross-validation

- Widely used approach for estimating test error.
- Estimates can be used to select best model, and to give an idea of the test error of the final chosen model.
- Idea is to randomly divide the data into K equal-sized parts. We leave out part k, fit the model to the other $K - 1$ parts (combined), and then obtain predictions for the left-out kth part.
- This is done in turn for each part $k = 1, 2, \ldots, K$, and then the results are combined.



K-fold cross-validation in detail

- Let the K parts be $C_1, C_2, \ldots, C_K$, where C_k denotes the indices of the observations in part k. There are n_k observations in part k: if n is a multiple of K, then $n_k = n/K$.
- Compute

  $$CV_{(K)} = \frac{1}{n} \sum_{k=1}^{K} n_k \times \text{MSE}_k,$$

  where

  $$\text{MSE}_k = \frac{1}{n_k} \sum_{i \in C_k} (y_i - \widehat{y}_i^{(-k)})^2$$

  and $\widehat{y}_i^{(-k)}$ is the fit for observation i, obtained from the data with part k removed.
- Setting $K = N$ yields N-fold or leave-one-out cross-validation (LOOCV).

Special case

- With least-squares linear or polynomial regression, a shortcut makes the cost of LOOCV the same as that of a single model fit!
- The following formula holds:

$$CV_{(N)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \widehat{y}_i}{1 - h_i}\right)^2,$$

  where $\widehat{y}_i$ is the ith fitted value from the original least squares fit, and h_i is the leverage (diagonal of the "hat" matrix).
- This is like the ordinary MSE, except the ith residual is divided by $1 - h_i$.

Choice of K

- Since each training set is only $(K-1)/K$ as big as the original training set, the estimates of prediction error will typically be biased upward.
- This bias is minimized when $K = n$ (LOOCV), but this estimate has high variance, because the estimates from each fold are highly correlated.
- $K = 5$ or 10 provides a good compromise for this bias-variance tradeoff.

Standard error of the CV estimate

- We can estimate the standard error (standard deviation) of the CV error by

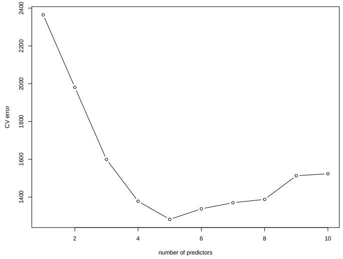
$$\widehat{se}(CV_{(K)}) = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}(\text{MSE}_k - \overline{\text{MSE}})^2}$$

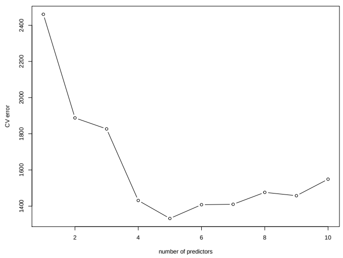
- One-standard-error rule:
  - Calculate the standard error of the estimated test MSE for each model size
  - Select the smallest model for which the estimated test error is within one standard error of the lowest point on the curve.

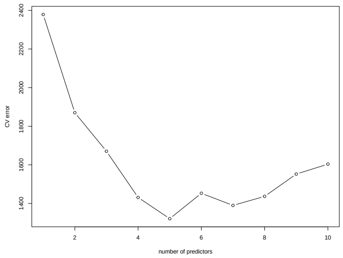
Example of 10-fold cross-validation

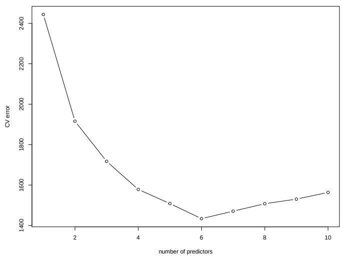
```
K<-10
folds=sample(1:K,n,replace=TRUE)
CV<-rep(0,10)

for(i in (1:10)){
for(k in (1:K)){
reg<-lm(Formula[[i]],data=pollution[folds!=k,])
pred<-predict(reg,newdata=pollution[folds==k,])
CV[i]<-CV[i]+ sum((pollution$Mortality[folds==k]-pred)^2)
}
CV[i]<-CV[i]/n
}
```

Result (4 trials)









Final remarks on cross-validation

- The CV error rates can be averaged over r repetitions of K-fold cross-validation with different random partitions, to reduce the variance of the CV error estimates.
- After the best model has been selected, we usually re-estimate the model parameters using the whole training set.
- To obtain an unbiased estimate of the best model's error, we need an independent test set, or two nested CV loops!

Overview

Overview

Shrinkage methods

- By retaining a subset of the predictors and discarding the rest, subset selection produces a model that is interpretable and has possibly lower prediction error than the full model.
- However, because it is a discrete process – variables are either retained or discarded – it often exhibits high variance, and so does not always reduce the prediction error of the full model.
- Shrinkage methods are more continuous, and do not suffer as much from high variability.
- Two main methods:
  1. Ridge regression
  2. Lasso

Ridge regression

- Ridge regression shrinks the regression coefficients by imposing a penalty on their size. The ridge coefficients minimize a penalized residual sum of squares:

$$\widehat{\beta}^{\text{ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

- Here $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage. The coefficients are shrunk toward zero (and each other), i.e., to the simplest model (with only the constant term).
- Selecting a good value for λ is critical; cross-validation can be used for this.

Equivalent form

- An equivalent way to write the ridge problem is

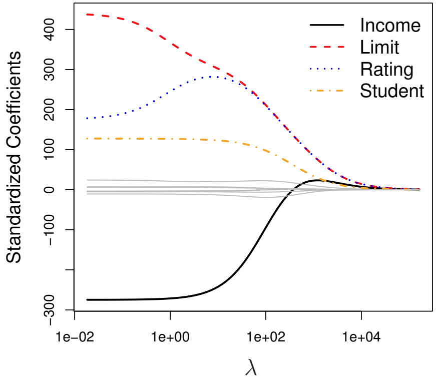
$$\widehat{\beta}^{\text{ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \right\}$$

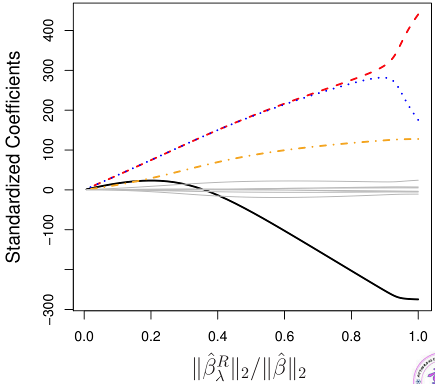
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 \leq t,$$

which makes explicit the size constraint on the parameters.

- There is a one-to-one correspondence between parameters t and λ in the previous formulation.

The effect of ridge regression





Derivation of the ridge regression estimates

- It can be shown that $\widehat{\beta}^{\text{ridge}}$ can be found by separating the minimization problem into two parts, after centering the inputs (replacing x_{ij} by $x_{ij} - \overline{x}_j$):
  1. We estimate β_0 by $\overline{y} = \frac{1}{n}\sum_{i=1}^n y_i$
  2. The remaining coefficients get estimated by a ridge regression without intercept, using the centered x_{ij} and the centered y_i.
- We assume that both the inputs and the output have been centered, so that the input matrix $\mathbf{X}$ has p (rather than $p+1$) columns, and $\mathbf{y}$ is the n-vector of centered outputs.
- The criterion can be written in matrix form

$$\text{RSS}_\lambda(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^T\beta.$$

Derivation of the ridge regression estimates (continued)

- The criterion can be rewritten as

$$\mathrm{RSS}_\lambda(\beta) = \mathbf{y}^T\mathbf{y} - 2\beta^T\mathbf{X}^T\mathbf{y} + \beta^T(\mathbf{X}^T\mathbf{X} + \lambda I)\beta$$

- Differentiating with respect to β we obtain

$$\frac{\partial \mathrm{RSS}_\lambda(\beta)}{\partial \beta} = -2\mathbf{X}^T\mathbf{y} + 2(\mathbf{X}^T\mathbf{X} + \lambda I)\beta$$

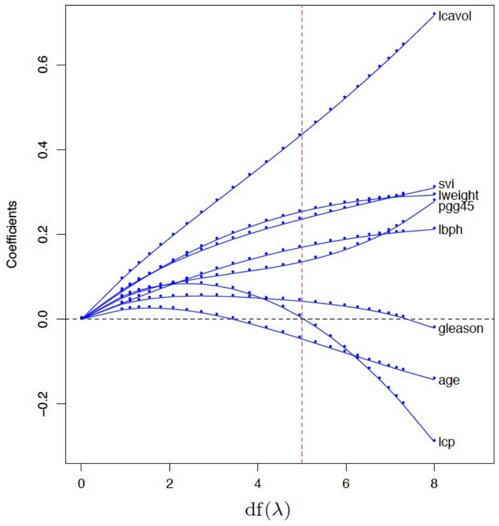
- The solution of the equation $\frac{\partial \mathrm{RSS}_\lambda(\beta)}{\partial \beta} = 0$ is

$$\widehat{\beta}^{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

Effective degrees of freedom

- As with the least-squares method, the fitted values are linear functions of the y_i
$$\widehat{y}^{\text{ridge}} = \mathbf{X}\widehat{\beta}^{\text{ridge}} = \mathbf{S}_\lambda \mathbf{y}$$
with $\mathbf{S}_\lambda = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T$.
- When $\lambda = 0$, $\text{tr}(\mathbf{S}_\lambda) = p$, i.e., the degrees of freedom of the model.
- By analogy, when $\lambda > 0$, we can define the effective degrees of freedom as
$$\text{df}(\lambda) = \text{tr}(\mathbf{S}_\lambda).$$

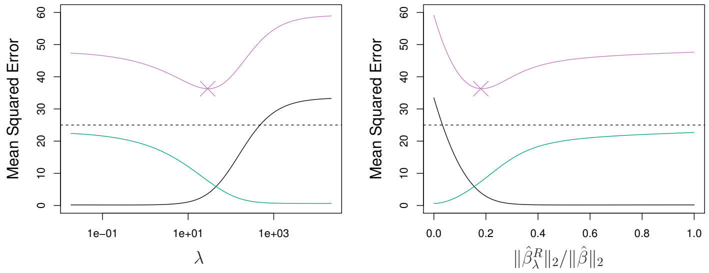
Example


Coefficients
0.6
0.4
0.2
0.0
−0.2
0
2
4
6
8
df(λ)
lcavol
svi
lweight
pgg45
lbph
gleason
age
lcp

Ridge regression: scaling of predictors

- The standard least squares coefficient estimates are scale equivariant: multiplying X_j by a constant c simply leads to a scaling of the least squares coefficient estimates by a factor of $1/c$. In other words, regardless of how the jth predictor is scaled, $X_j\widehat{\beta}_j$ will remain the same.
- In contrast, the ridge regression coefficient estimates can change substantially when multiplying a given predictor by a constant, due to the sum of squared coefficients term in the penalty part of the ridge regression objective function.
- Therefore, it is best to apply ridge regression after standardizing the predictors (dividing each centered variable by its standard deviation).

Why Does Ridge Regression Improve Over Least Squares?



Simulated data with $n = 50$ observations, $p = 45$ predictors, all having nonzero coefficients. Squared bias (black), variance (green), and test MSE (purple) for the ridge regression predictions, as a function of λ and $\|\widehat{\beta}_\lambda^{\text{ridge}}\|^2/\|\widehat{\beta}\|^2$. The horizontal dashed lines indicate the minimum possible MSE.

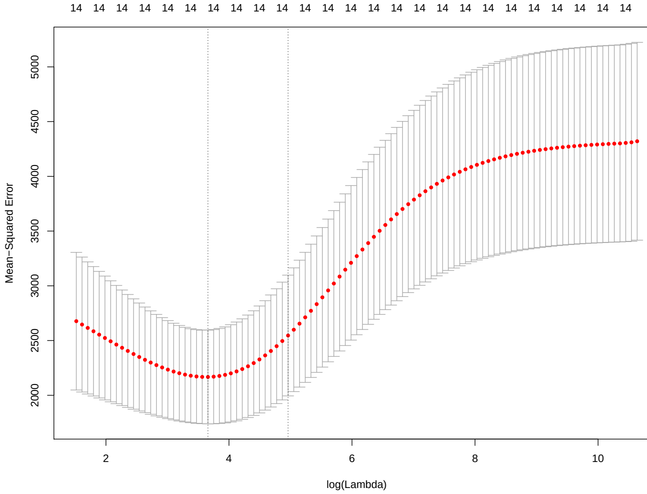
Why does ridge regression reduce variance?

- When there are many correlated variables in a linear regression model, their coefficients can become poorly determined and exhibit high variance.
- A wildly large positive coefficient on one variable can be canceled by a similarly large negative coefficient on its correlated cousin.
- By imposing a size constraint on the coefficients, this problem is alleviated.

Ridge regression in R

```
library(glmnet)

x<-model.matrix(Mortality~.-logNOx,pollution)
y<-pollution$Mortality[-21] # obs 21 has 2 missing values
n<-nrow(x)
napp=45
ntst=n-45
train<-sample(1:n,napp)
xapp<-x[train,]
yapp<-y[train]
xtst<-x[-train,]
ytst<-y[-train]


cv.out<-cv.glmnet(xapp,yapp,alpha=0)
plot(cv.out)

fit<-glmnet(xapp,yapp,lambda=cv.out$lambda.min,alpha=0)
ridge.pred<-predict(fit,s=cv.out$lambda.min,newx=xtst)
print(mean((ytst-ridge.pred)^2))
2421.136
```

CV error as a function of λ



Coefficients

```
fit$beta
s0
(Intercept) .
JanTemp -2.641635e-01
JulyTemp 7.231499e-01
RelHum -1.443636e-01
Rain 9.618201e-01
Education -1.154417e+01
PopDensity 2.066547e-03
pNonWhite 1.478269e+00
pWC -1.105875e+00
pop 2.629839e-06
pophouse 3.057905e+01
income -1.008305e-03
logHCPot 2.311552e+00
logNOxPot 6.616369e+00
logSO2ot 3.966114e+00
```

The Lasso

- Ridge regression has one obvious disadvantage: unlike subset selection, which will generally select models that involve just a subset of the variables, ridge regression will include all p predictors in the final model
- The Lasso is a relatively recent alternative to ridge regression that overcomes this disadvantage. The Lasso coefficients, $\widehat{\beta}^{lasso}$ minimize the quantity

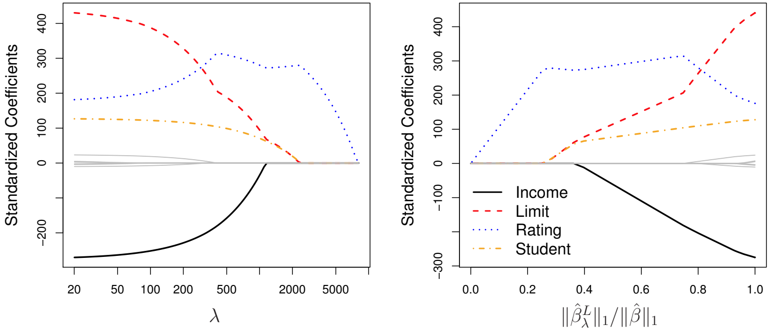
$$\widehat{\beta}^{\text{lasso}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\},$$

i.e., the L_2 norm is replaced by the L_1 norm in the penalty term.

The Lasso (continued)

- As with ridge regression, the lasso shrinks the coefficient estimates towards zero.
- However, in the case of the lasso, the L_1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large.
- Hence, much like best subset selection, the lasso performs variable selection.
- We say that the lasso yields sparse models – that is, models that involve only a subset of the variables.
- As in ridge regression, selecting a good value of λ for the lasso is critical; cross-validation is again the method of choice.

Example


Standardized Coefficients
400
300
200
100
0
−200
20
50
100
200
500
2000
5000
λ
Standardized Coefficients
400
300
200
100
0
−100
−300
0.0
0.2
0.4
0.6
0.8
1.0
$\|\hat{\beta}^L_\lambda\|_1/\|\hat{\beta}\|_1$
Income
Limit
Rating
Student

Equivalent form

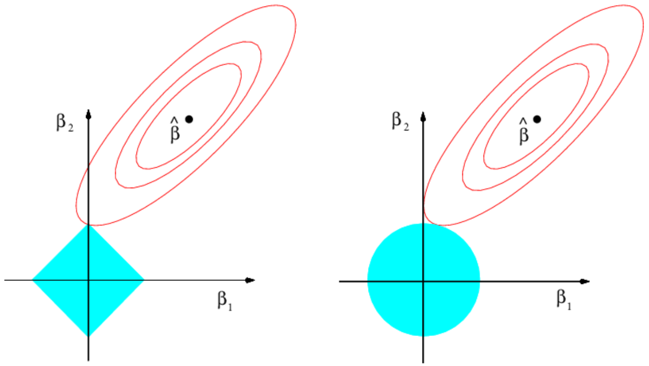
- As in the case of ridge problem, the previous unconstrained optimization problem is equivalent to the following constrained one

$$\widehat{\beta}^{\text{lasso}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \right\}$$

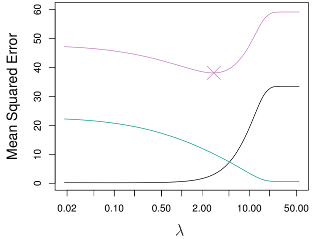
$$\text{subject to } \sum_{j=1}^{p} |\beta_j| \leq t,$$

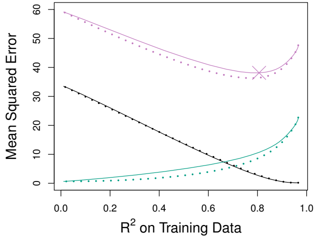
- This problem can be solved using a quadratic programming algorithm.
- Remark: this time, the solution $\widehat{\beta}^{\text{lasso}}$ is a nonlinear function of $\mathbf{y}$. There is no obvious notion of effective degrees of freedom.

Why does the Lasso eliminate variables?



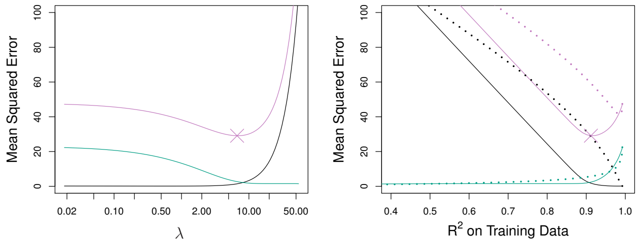
Comparing the Lasso and Ridge Regression





Left: Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso on simulated data set of Slide 53. Right: Comparison of squared bias, variance and test MSE between lasso (solid) and ridge (dashed). Both are plotted against their R^2 on the training data, as a common form of indexing.

Comparing the Lasso and Ridge Regression (continued)

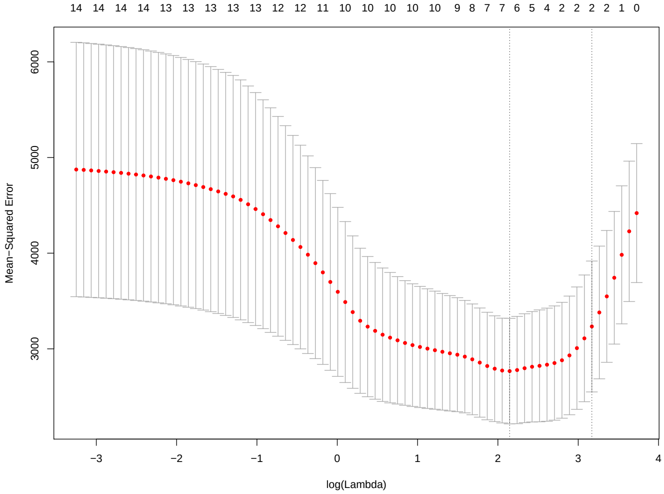


Left: Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso. The simulated data is similar to that in the previous slide, except that now only two predictors are related to the response. Right: Comparison of squared bias, variance and test MSE between lasso (solid) and ridge (dashed). Both are plotted against their R^2 on the training data, as a common form of indexing.

The Lasso in R

```
cv.out<-cv.glmnet(xapp,yapp,alpha=1)
plot(cv.out)

fit.lasso<-glmnet(xapp,yapp,lambda=cv.out$lambda.min,alpha=1)

lasso.pred<-predict(fit.lasso,s=cv.out$lambda.min,newx=xtst)
print(mean((ytst-lasso.pred)^2))
1946.667
```

CV error as a function of λ (Lasso)



Coefficients

```
> print(fit.lasso$beta)
s0
(Intercept) .
JanTemp -1.157095e+00
JulyTemp .
RelHum .
Rain 1.404239e+00
Education -1.796084e+01
PopDensity .
pNonWhite 2.880287e+00
pWC -9.421496e-01
pop 2.141275e-06
pophouse .
income -4.655832e-04
logHCPot .
logNOxPot 1.392387e+01
logSO2ot 3.461564e-01
```

Bayesian interpretation of ridge regression and Lasso

Bayesian inference

- In Bayesian inference, the parameter β is considered as a random variable.
- Inference consists in computing the conditional probability distribution of the parameter given the data, obtained by the Bayes Theorem as

$$p(\beta|\mathbf{y}) = \frac{p(\mathbf{y}|\beta)p(\beta)}{p(\mathbf{y})} \propto p(\mathbf{y}|\beta)p(\beta)$$

- The marginal distribution $p(\beta)$ is called the prior distribution of β. It encodes prior knowledge about β, i.e., information that we have about β before observing the data.

Bayesian interpretation of ridge regression and Lasso

Ridge regression corresponds to Gaussian errors and prior

- Assumptions:
  1. Gaussian errors: $\mathbf{Y} = \mathbf{X}\beta + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_N)$, so
  $$p(\mathbf{y}|\beta) \propto \exp\left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \right\}$$
  2. Gaussian prior: $p(\beta) \propto \exp\left(-\frac{1}{2\sigma_0^2} \sum_{j=1}^{p} \beta_j^2 \right)$,
- Then the log-posterior density of β is
$$\log p(\beta|\mathbf{y}) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 - \frac{1}{2\sigma_0^2} \sum_{j=1}^{p} \beta_j^2 + c$$
- Ridge regression searches for the mode of posterior distribution.

Bayesian interpretation of ridge regression and Lasso

Lasso correspond to a Laplace prior

- With the Gaussian error model as before and an independent Laplace prior

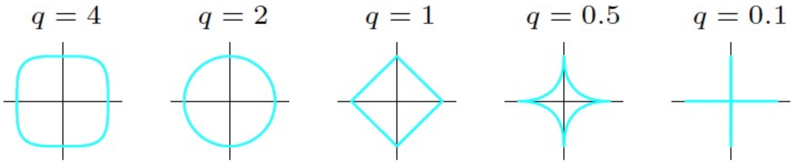
$$p(\beta) \propto \exp\left(-\frac{1}{\tau}\sum_{j=1}^{p}|\beta_j|\right),$$

we get

$$\log p(\beta|\mathbf{y}) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}x_{ij}\beta_j\right)^2 - \frac{1}{\tau}\sum_{j=1}^{p}|\beta_j| + c$$

Generalization

- More generally, a prior of the form
$$p(\beta) \propto \exp\left(-\gamma \sum_{j=1}^{p} |\beta_j|^q\right),$$
leads to minimizing the MSE under a constraint $\sum_{j=1}^{p} |\beta_j|^q \leq t$.
- The case $q = 1$ (lasso) is the smallest q such that the constraint region is convex; nonconvex constraint regions make the optimization problem more difficult.
- The case $q = 0$ corresponds to subset selection.



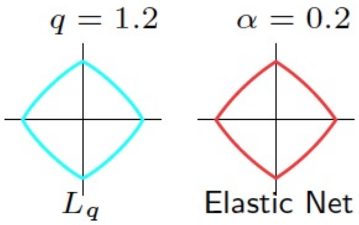
Finding a compromise between lasso and ridge regression

- We might try using other values of q besides 0, 1, or 2. Values of $q \in (1, 2)$ suggest a compromise between the lasso and ridge regression.
- Although this is the case, with $q > 1$, $|\beta_j|^q$ is differentiable at 0, and so does not share the ability of lasso (q = 1) for setting coefficients exactly to zero.
- The elastic net penalty

$$\lambda \sum_{j=1}^{p} (\alpha \beta_j^2 + (1 - \alpha)|\beta_j|)$$

  is a different compromise between ridge and lasso.
- The elastic-net selects variables like the lasso, and shrinks together the coefficients of correlated predictors like ridge.

Elastic net penalty



Contours of constant value of $\sum_{j=1}^{p} |\beta_j|^q$ for $q = 1.2$ (left plot), and the elastic-net penalty $\sum_{j=1}^{p}(\alpha\beta_j^2 + (1-\alpha)|\beta_j|)$ for $\alpha = 0.2$ (right plot). Although visually very similar, the elastic-net has sharp (non-differentiable) corners, while the $q = 1.2$ penalty does not.

Overview

Regularized discriminant analysis

- The idea of regularization can be applied to other models such as QDA and LDA.
- For instance, one can shrink the separate covariances of QDA toward a common covariance as in LDA. The regularized covariance matrices have the form
$$\widehat{\boldsymbol{\Sigma}}_k(\lambda) = (1-\lambda)\widehat{\boldsymbol{\Sigma}}_k + \lambda\widehat{\boldsymbol{\Sigma}}$$
where $\widehat{\boldsymbol{\Sigma}}$ is the pooled covariance matrix as used in LDA.
- Here $\lambda \in [0,1]$ allows a continuum of models between LDA and QDA, and needs to be specified.
- In practice λ can be chosen based on the performance of the model on validation data, or by cross-validation.

Regularized discriminant analysis (continued)

- Similar modifications allow $\widehat{\Sigma}$ itself to be shrunk toward the scalar covariance,
$$\widehat{\boldsymbol{\Sigma}}(\gamma) = (1-\gamma)\widehat{\boldsymbol{\Sigma}} + \gamma\widehat{\sigma}^2\mathbf{I}$$
for $\gamma \in [0,1]$.
- Replacing $\widehat{\boldsymbol{\Sigma}}$ in the previous equation by $\widehat{\Sigma}(\gamma)$ leads to a more general family of covariances $\widehat{\boldsymbol{\Sigma}}_k(\lambda,\gamma)$ indexed by a pair of parameters.
- In R: package `klaR`, function `rda`.

Overview

Feature extraction

- Given p variables (features) $X = (X_1, \ldots, X_p)$, feature extraction consists in finding q new features

$$\begin{aligned} Z_1 &= \phi_1(X_1, \ldots, X_p) \\ &\vdots \\ Z_q &= \phi_q(X_1, \ldots, X_p), \end{aligned}$$

  where $\phi_1, \ldots, \phi_q$ are functions from $\mathbb{R}^p$ to $\mathbb{R}$. These functions may be linear or nonlinear.
- When functions ϕ_j are linear, we can write $Z_j = u_j^T X$, where $u_j \in \mathbb{R}^p$. Only this case will be considered in this chapter.
- Geometrically, Z_j can be seen as the coordinate of the projection of X onto an axis directed by u_j.

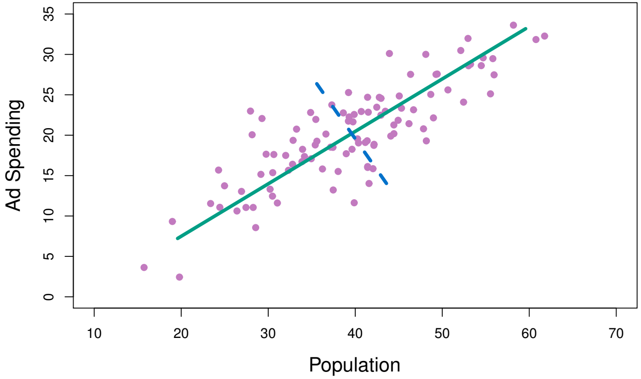
Objectives of feature extraction

- Feature extraction is useful for
  - Representing high-dimensional data in a lower-dimensional feature space
  - Reducing the input dimension (and hence the number of parameters) in prediction (regression or classification) problems
- Vectors $u_1, \ldots, u_q$ are determined in such a way that the new features $Z_1, \ldots, Z_q$ contain as much useful information as possible.
- Feature extraction methods can be supervised, or unsupervised.
- Here, we consider two methods:
  1. Principal Component Analysis (PCA) – unsupervised
  2. Factor Discriminant Analysis (FDA) - supervised

Overview

Principal component analysis

Idea: find orthogonal directions u_j in input space in which the projected data has maximal variance. These directions correspond to features $Z_j = u_j^T X$ (called principal components) that have maximum information content.



Finding the first component

- Let $X = (X_1, \ldots, X_p)$ a random vector with variance matrix $\mathbf{\Sigma}$.
- The first feature $Z_1 = u_1^T X$, called the first component, is chosen such that
$$\text{Var}(Z_1) = \max_{u_1} \text{Var}(u_1^T X) = \max_{u_1} u_1^T \mathbf{\Sigma} u_1$$
subject to $u_1^T u_1 = 1$.
- To solve this constrained optimization problem, we write the Lagrange function as
$$L(u_1, \lambda) = u_1^T \mathbf{\Sigma} u_1 - \lambda(u_1^T u_1 - 1)$$

Finding the first component (continued)

- The solution must verify

$$\frac{\partial L}{\partial u_1} = 2\boldsymbol{\Sigma} u_1 - 2\lambda u_1 = 0 \Leftrightarrow \boldsymbol{\Sigma} u_1 = \lambda u_1$$
$$u_1^T u_1 = 1$$

- Vector u_1 is thus the eigenvector of $\boldsymbol{\Sigma}$ with unit norm and eigenvalue λ_1. (We recall that a symmetric and positive definite $p \times p$ matrix has p orthogonal eigenvectors with real and positive eigenvalues).
- Now,

$$\text{Var}(u_1^T X) = u_1^T \boldsymbol{\Sigma} u_1 = u_1^T (\lambda_1 u_1) = \lambda_1 u_1^T u_1 = \lambda_1.$$

so λ_1 must be the largest eigenvalue.
- Vector u_1 is called the loading vector of the first principal component.

Finding the second component

- The second component $Z_2 = u_2^T X$ is chosen such that

$$\text{Var}(Z_2) = \max_{u_2} \text{Var}(u_2^T X) = \max_{u_2} u_2^T \boldsymbol{\Sigma} u_2$$

  subject to $u_2^T u_2 = 1$ and $\text{Cov}(Z_1, Z_2) = 0$.

- Now,

$$\text{Cov}(Z_1, Z_2) = \text{Cov}(u_1^T X, u_2^T X) = u_1^T \boldsymbol{\Sigma} u_2 = \lambda_1 u_1^T u_2,$$

  so the second constraint can be written $u_1^T u_2 = 0$.

- The Lagrange function is

$$L(u_2, \lambda, \mu) = u_2^T \boldsymbol{\Sigma} u_2 - \lambda(u_2^T u_2 - 1) - \mu u_2^T u_1$$

Finding the second component (continued)

- We solve:
$$\frac{\partial L}{\partial u_2} = 2\mathbf{\Sigma} u_2 - 2\lambda u_2 - \mu u_1 = 0 \tag{1}$$
- Left-multiplying (1) by u_1^T, we get
$$2\underbrace{u_1^T \mathbf{\Sigma} u_2}_{0} - 2\underbrace{u_1^T \lambda u_2}_{0} - \mu u_1^T u_1 = 0 \Rightarrow \mu = 0$$
- So, (1) reduces to
$$\frac{\partial L}{\partial u_2} = 2\mathbf{\Sigma} u_2 - 2\lambda u_2 = 0 \Leftrightarrow \mathbf{\Sigma} u_2 = \lambda u_2$$
- The solution is an eigenvector of $\mathbf{\Sigma}$. We choose the one with the second largest eigenvalue λ_2.

Finding the next components

- Continuing the same line of reasoning, we obtain p uncorrelated components $Z = (Z_1, \ldots, Z_p)$ corresponding to the eigenvalues $\lambda_1 \geq \ldots \geq \lambda_p$ of $\mathbf{\Sigma}$.
- We can write

$$Z = \mathbf{U}^T X,$$

where $\mathbf{U} = (u_1, \ldots, u_p)$ is the $p \times p$ matrix (called the loading matrix) whose columns are the p eigenvectors.

Properties

- The variance matrix of Z is

$$\mathrm{Var}(Z) = \mathbf{U}^T \mathbf{\Sigma} \mathbf{U} = \mathbf{\Lambda},$$

  where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$ is the diagonal matrix containing the p eigenvalues of $\mathbf{\Sigma}$.
- Matrix $\mathbf{U}$ verifies $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, i.e., $\mathbf{U}^{-1} = \mathbf{U}^T$: it is an orthogonal matrix, corresponding to rotation.

Properties (continued)

- Consequently,

$$\mathrm{tr}(\mathbf{\Lambda}) = \mathrm{tr}[\mathbf{U}^T(\mathbf{\Sigma}\mathbf{U})] = \mathrm{tr}[(\mathbf{\Sigma}\mathbf{U})\mathbf{U}^T] = \mathrm{tr}(\mathbf{\Sigma})$$

- Hence, the sum of the eigenvalues is the total variance

$$\sum_{j=1}^{p} \lambda_j = \sum_{j=1}^{p} \mathrm{Var}(X_i)$$

- The proportion of the variance explained by the first q components is

$$\sum_{j=1}^{q} \lambda_j \Big/ \sum_{j=1}^{p} \lambda_j$$

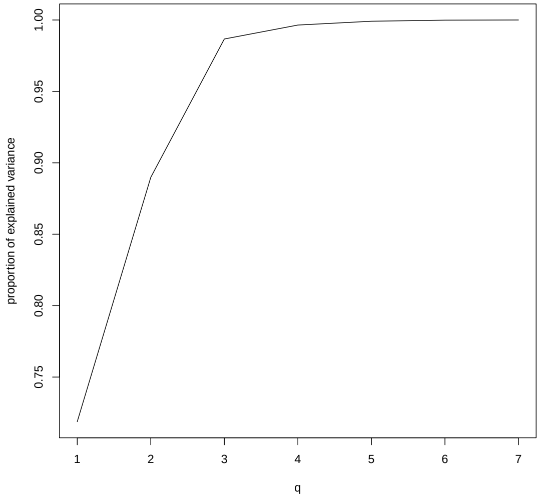
Practical application

- In practice, we center the data, and we estimate $\boldsymbol{\Sigma}$ by the empirical variance matrix
$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n} x_i {x_i}^T$$
- If we also standardize the data, then matrix $\widehat{\boldsymbol{\Sigma}}$ is actually the correlation matrix $\mathbf{R}$ (its diagonal elements equal 1, and its off-diagonal elements are correlation coefficients).
- Typically, we keep only q components $Z_1, \ldots, Z_q$ such that the cumulative proportion of explained variance is close enough to 1.

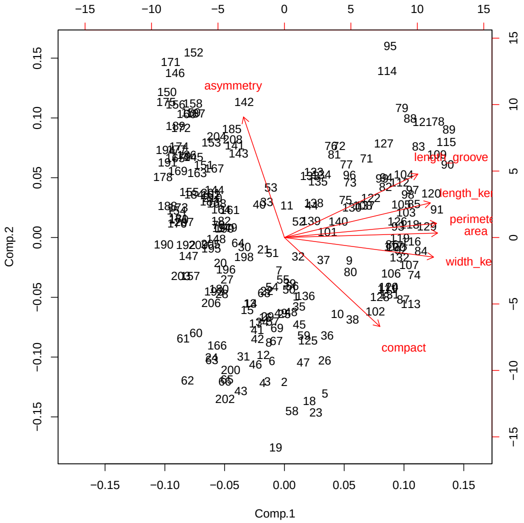
PCA in R: example with the Seeds data

```
pca<-princomp(x,cor=TRUE)
Z<-pca$scores
lambda<-pca$sdev^2

plot(cumsum(lambda)/sum(lambda),type="l",xlab="q",ylab="proportion
of explained variance")

biplot(pca)

km<-kmeans(x,centers=3,nstart=10)
pairs(Z[,1:3],col=km$cluster)
```

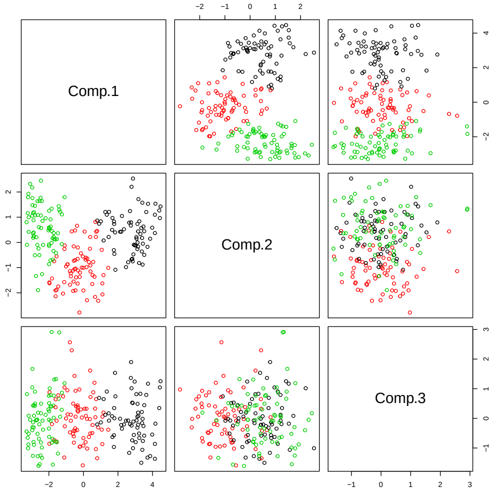
Proportion of explained variance



First 2 principal components



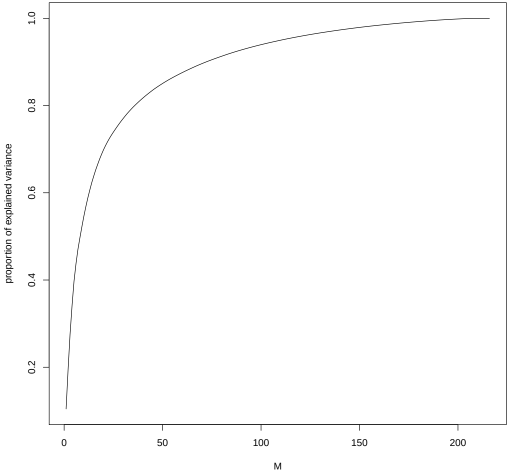
First 3 principal components and HCM partition



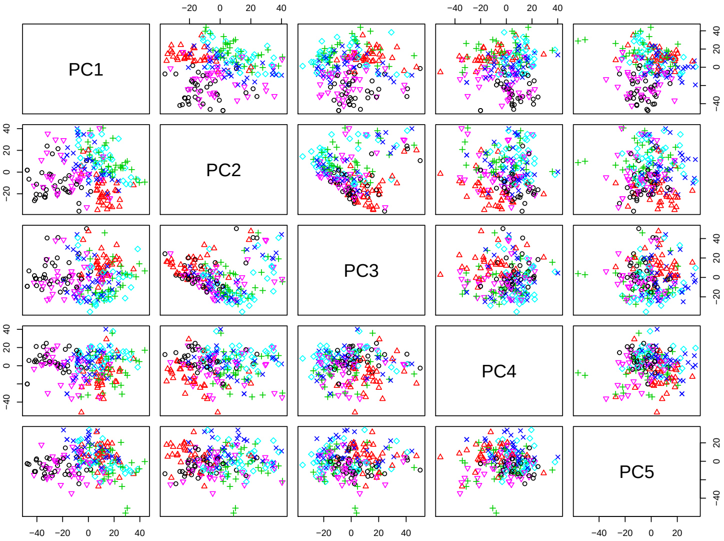
PCA in R: example 2

```
# Expressions dataset
load('data_expressions.RData')
ii<-which(colMeans(X)==0)
X1<-X[,-ii]
X1<-scale(X1)

# princomp doesn't work because the number of colums of X1
# exceeds the number of rows

pca<-prcomp(X1)
lambda<-pca$sdev^2

pairs(pca$x[,1:5],col=y)
plot(cumsum(lambda)/sum(lambda),type="l",xlab="M",
                 ylab="proportion of explained variance")
```

Proportion of explained variance



First five principal components



Principal component regression (PCR)

- The idea is to fit a regression model using least squares, taking as predictors $M < p$ principal components:

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

- We have

$$\sum_{m=1}^{M} \theta_m z_{im} = \sum_{m=1}^{M} \theta_m \sum_{j=1}^{p} u_{mj} x_{ij} = \sum_{j=1}^{p} \sum_{m=1}^{M} \theta_m u_{mj} x_{ij} = \sum_{j=1}^{p} \beta_j x_{ij}$$

with

$$\beta_j = \sum_{m=1}^{M} \theta_m u_{mj}$$

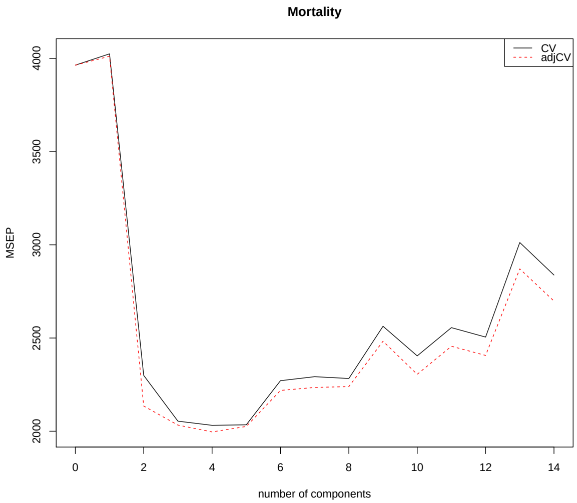
Principal component regression (continued)

- Hence, the PCR model can be thought of as a special case of the original linear regression model.
- Dimension reduction serves to constrain the estimated β_j coefficients, which can yield a good bias-variance tradeoff.
- As with ridge regression, principal components depend on the scaling of the inputs, so typically we first standardize them.
- The value of M can be determined by cross-validation.

Principal component regression in R

```
library(pls)

pcr.fit<-pcr(Mortality ~.-logNOx,data=pollution,scale=TRUE,
                                    validation="CV")

summary(pcr.fit)
validationplot(pcr.fit,val.type = "MSEP",legendpos = "topright")
```

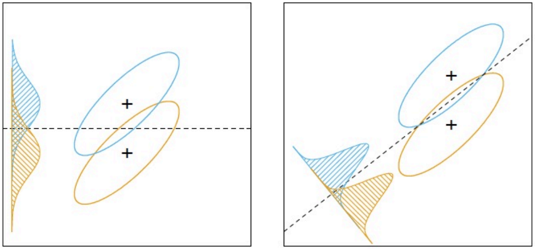
Cross-validation MSE as a function of M



Overview

Factor discriminant analysis

- Factor discriminant analysis (FDA) is a supervised dimension reduction techniques, suitable for classification problems.
- It finds linear combinations of the original predictors, such that the between-class variance is maximized with respect to the within-class variance (i.e., such that the overlap between the classes is minimized).



Decomposition of the variance

It can be shown that the variance matrix $\mathbf{\Sigma} = \mathrm{Var}(X)$ can be decomposed as

$$\mathbf{\Sigma} = \mathbf{B} + \mathbf{W},$$

where $\mathbf{B}$ and $\mathbf{W}$ are, respectively, the between-class and within-class matrices:

$$\mathbf{W} = \sum_{k=1}^{c} \pi_k \mathbf{\Sigma}_k$$

$$\mathbf{B} = \sum_{k=1}^{c} \pi_k (\mu_k - \mu)(\mu_k - \mu)^T,$$

where $\mu_k = \mathbb{E}(X|Y = k)$, $\mu = \mathbb{E}(X)$ and $\mathbf{\Sigma}_k = \mathrm{Var}(X|Y = k)$

Decomposition of the variance (continued)

- For a new variable $Z = u^T X$, its variance is

$$\mathrm{Var}(Z) = u^T \boldsymbol{\Sigma} u = u^T(\mathbf{W}+\mathbf{B})u = u^T\mathbf{W}u + u^T\mathbf{B}u.$$

- Here,

$$u^T\mathbf{W}u = u^T\left(\sum_{k=1}^{c}\pi_k\Sigma_k\right)u = \sum_{k=1}^{c}\pi_k u^T\Sigma_k u = \sum_{k=1}^{c}\pi_k \mathrm{Var}(Z|Y=k)$$

and

$$\begin{aligned} u^T\mathbf{B}u &= \sum_{k=1}^{c}\pi_k u^T(\mu_k-\mu)(\mu_k-\mu)^T u \\ &= \sum_{k=1}^{c}\pi_k(u^T\mu_k - u^T\mu)^2 = \sum_{k=1}^{c}\pi_k\left\{\mathbb{E}(Z|Y=k)-\mathbb{E}(Z)\right\}^2 \end{aligned}$$

Maximization of class separation

- Problem, find Z such that the ratio

$$J(u) = \frac{u^T \mathbf{B} u}{u^T \mathbf{W} u}$$

  is maximized.
- Solution: u is the eigenvector associated to the largest eigenvalue of $\mathbf{\Sigma}^{-1}\mathbf{B}$. Variable Z is called a discriminant coordinate.
- Case $c = 2$: matrix $\mathbf{B}$ has rank one, and so has $\mathbf{\Sigma}^{-1}\mathbf{B}$. We can show that

$$u = \mathbf{\Sigma}^{-1}(\mu_1 - \mu_2).$$

  There is only one discriminant coordinate $Z = u^T X$.

Discriminant coordinates: case $K > 2$

- When $c > 2$, matrix $\Sigma^{-1}\mathbf{B}$ has rank $c-1$.
- There are $M = \min(c-1, p)$ discriminant coordinates $Z_1, \ldots, Z_M$, obtained from the eigenvectors of $\Sigma^{-1}\mathbf{B}$ corresponding to its M eigenvalues $\lambda_1 \geq \ldots \geq \lambda_M$.
- Remark: matrix $\Sigma^{-1}\mathbf{B}$ is not symmetric: its eigenvectors are not orthogonal, and variables Z_j are correlated.
- Interpretation: each coordinate $Z_m = u_m^T X$ for $m \geq 2$ maximizes the ratio

$$J(u_m) = \frac{u_m^T \mathbf{B} u_m}{u_m^T \mathbf{W} u_m}$$

under the constraints $u_m^T \mathbf{W} u_\ell = 0$, for $\ell = 1, \ldots, m-1$.

FDA in practice

- Matrices $\mathbf{\Sigma}$ and $\mathbf{B}$ are estimated by

$$\widehat{\mathbf{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \widehat{\mu})(x_i - \widehat{\mu})^T$$

and

$$\widehat{\mathbf{B}} = \sum_{k=1}^{c}\frac{n_k}{n}(\widehat{\mu}_k - \widehat{\mu})(\widehat{\mu}_k - \widehat{\mu})^T$$

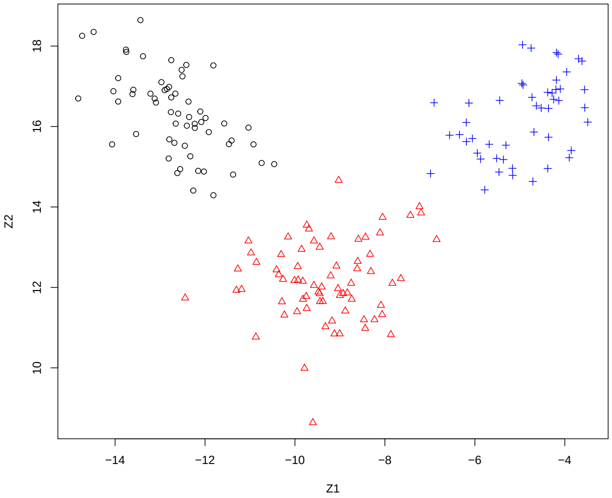
Example: Wine data

- Results of a chemical analysis of 178 wines grown in the same region in Italy but derived from three different cultivars.
- The analysis determined the quantities of 13 constituents found in each of the three types of wines.

FDA in R: Wine data

```
wine<-read.csv('wine.data',header=FALSE)

lda.wine<-lda(V1~. ,data=wine)
U<-lda.wine$scaling
X<-as.matrix(wine[,2:14])
Z<-X%*%U

plot(Z[,1],Z[,2],pch=wine$V1,col=wine$V1,xlab='Z1',ylab='Z2')
```

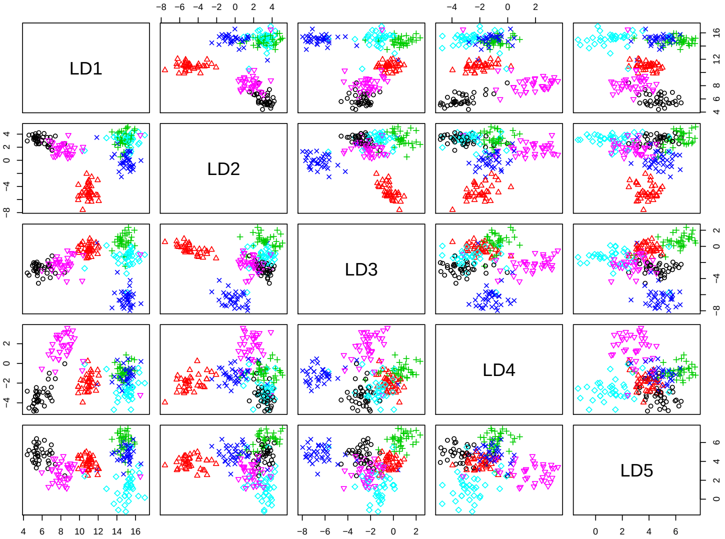
Plot in the 2-D space of discriminant coordinates



FDA in R: Expressions data

```
load('data_expressions.RData')
ii<-which(colMeans(X)==0)
X1<-X[,-ii]

lda.expr<-lda(y~ X1)
U<-lda.expr$scaling
Z<-X1%*%U

pairs(Z,col=y,pch=y)
```

Plot in the 5-D space of discriminant coordinates


LD1
LD2
LD3
LD4
LD5