CUNY School of Professional Studies

SPS.CUNY.EDU

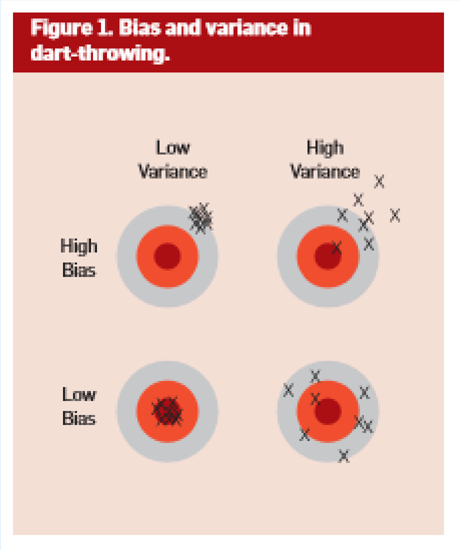
Lecture 04
2020 Spring Data-622
Bias Variance Tradeoff

Raman Kannan

Instructor Email Address: Raman.Kannan@sps.cuny.edu

Acknowledgements:
Generous support from IBM Power Systems Academic Initiative
IBM PSAI provides computing infrastructure for free

Bias/Variance



Error, because we dont have representative sample to learn from. Need more data. Estimating required data size is an important exercise in statistics.

Error because the model is too simple, underfitting, Model is so simple that it cannot fit even when labels are given, training data.
Manifests as Error during training is significant
Increase model complexity.

Error because the model captures even noise from the training data and is unable to determine class with never seen before data. Very small error during Training phase and <u>much larger</u> error during Testing phase (never seen before data).
Note error(training) is always lesser than error(testing),

https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

Variance Reduction Strategy

Cross Validation
Instead of 80% for training and 20% for testing,

Create equally sized subsets of data,
Iterate (train,test) over all the subsets, keeping one subset as test data
In each iteration.
Use the average of the model. (model parameter average)
10-fold CV or 5-fold CV or 3-fold CV.
Or Leave one out (LOO) CV.

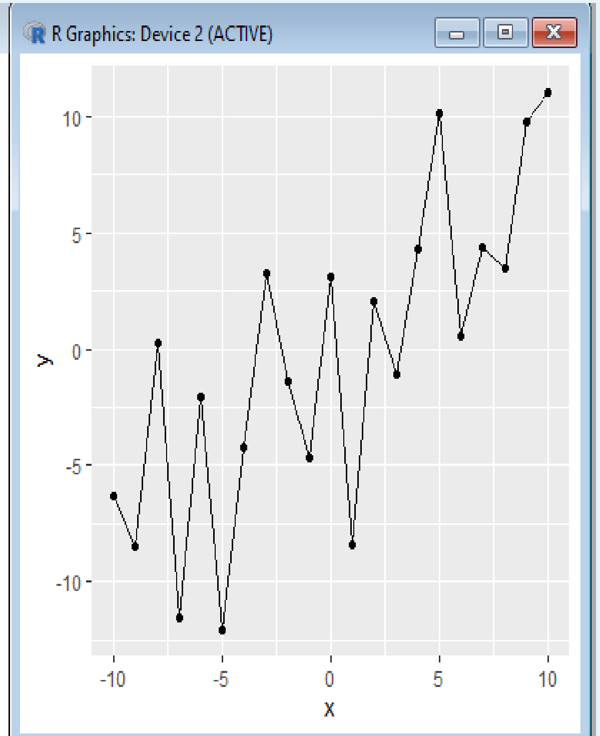
LOO CV: For each observation, exclude that observation, train on the
Rest of the data, test on the excluded observation
Finally take the average. Compute intensive.

Note with any type of CV, every available observation is used to train and
Test unlike the traditional process – where some% of data is excluded
from training and training.

Overfitting – high variance

R Console

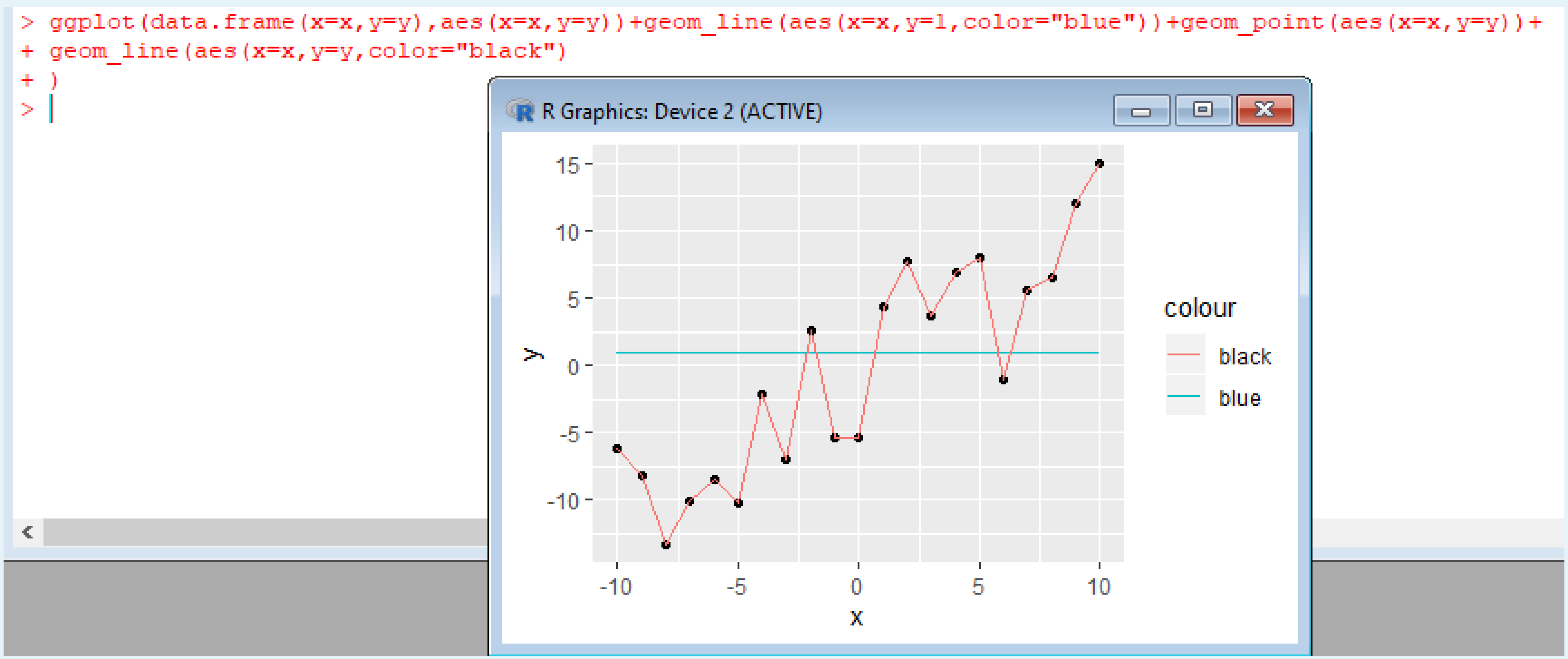
```
R version 3.5.2 (2018-12-20) -- "Eggshell Igloo"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> require(ggplot2)
Loading required package: ggplot2
Warning message:
package 'ggplot2' was built under R version 3.5.3
> x<- (-10):10
> n<-length(x)
> y<-rnorm(n,x,4)
> ggplot(data.frame(x=x,y=y),aes(x=x,y=y))+geom_point(aes(x=x,y=y))+geom_line(a$
> 
```



x<- (-10):10
n<-length(x)
y<-rnorm(n,x,4)
ggplot(data.frame(x=x,y=y),aes(x=x,y=y))+geom_point(aes(x=x,y=y))+geom_line(aes(x=x,y=y))

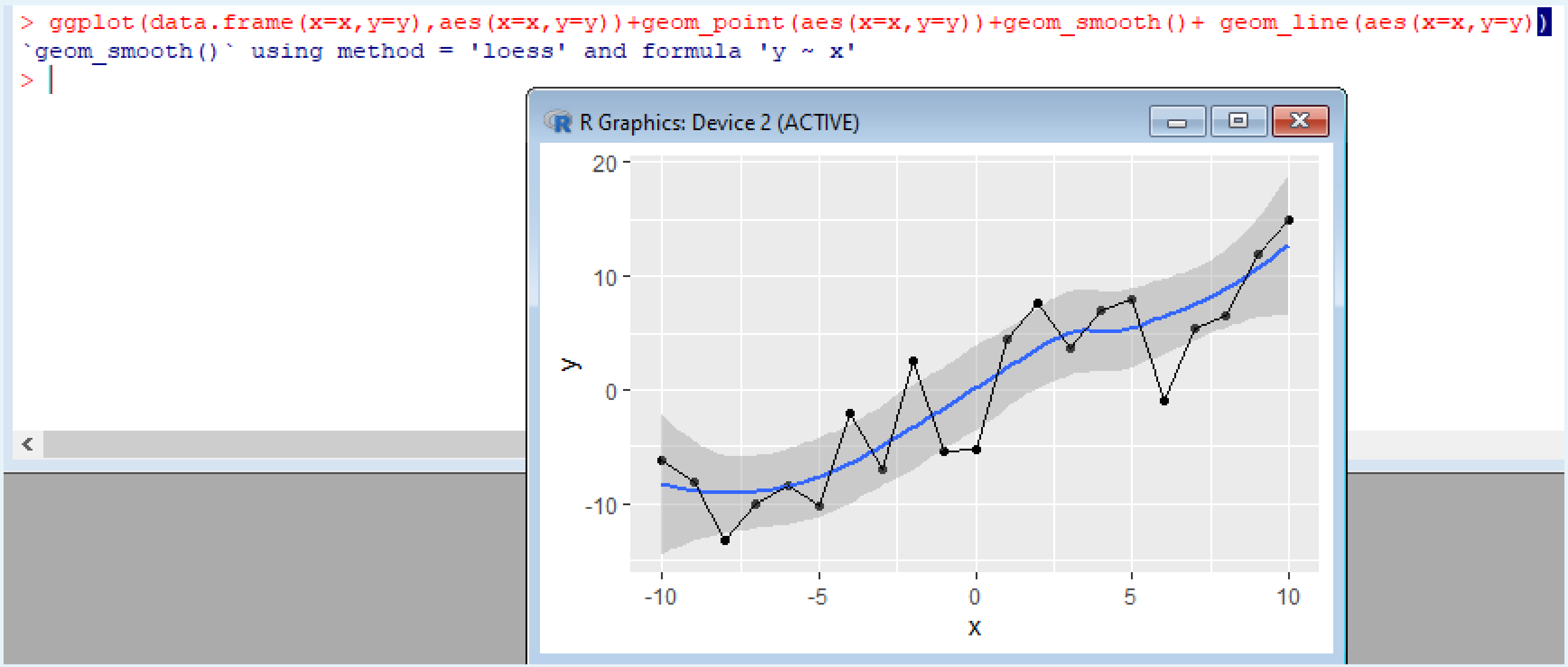
Underfitting – high bias

```
x<- (-10):10
n<-length(x)
y<-rnorm(n,x,4)
ggplot(data.frame(x=x,y=y),aes(x=x,y=y))+geom_line(aes(x=x,y=1))+geom_point(aes(x=x,y=y))+
geom_line(aes(x=x,y=y))
```



Smooth – less variability and lower bias

```
x<- (-10):10
n<-length(x)
y<-rnorm(n,x,4)

ggplot(data.frame(x=x,y=y),aes(x=x,y=y))+geom_point(aes(x=x,y=y))+geom_smooth()+
geom_line(aes(x=x,y=y))
```



https://towardsdatascience.com/bias-variance-and-regularization-in-linear-regression-lasso-ridge-and-elastic-net-8bf81991d0c5

Shrinkage (aka Penalization)

https://towardsdatascience.com/bias-variance-and-regularization-in-linear-regression-lasso-ridge-and-elastic-net-8bf81991d0c5

https://towardsdatascience.com/bias-variance-and-regularization-in-linear-regression-lasso-ridge-and-elastic-net-8

CUNY EMAIL ISL Statistics How to Stop Skype Fr Attributes: Defining C Fundry & Yesco - Firm Module 8 - Google Dr CS-

Towards Data Science
Sharing concepts, ideas, and codes.
Follow

11

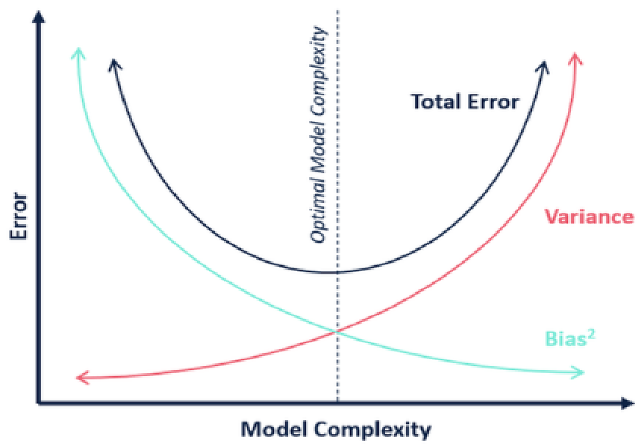


Image from Sydney Firmin

The bias-variance tradeoff is visualized above. The total error of the model is composed of three terms: the $(\text{bias})^2$, the variance, and an irreducible error term. As we can see in the graph, our optimal solution in which total error is minimized is at some intermediate model complexity, where neither bias nor variance is high.

L-1 Lasso

https://towardsdatascience.com/bias-variance-and-regularization-in-linear-regression-lasso-ridge-and-elastic-net-8bf81991d0c5

Lasso

$$\min_{\beta\in\mathbb{R}^p}\left\{\frac{1}{N}\|y-X\beta\|_2^2+\lambda\|\beta\|_1\right\}$$

The Lasso cost function, from Wikipedia

Lasso (sometimes stylized as LASSO or lasso) adds an additional term to the cost function, adding the sum of the coefficient values (the L-1 norm) multiplied by a constant lambda. This additional term penalizes the model for having coefficients that do not explain a sufficient amount of variance in the data. It also has a tendency to set the coefficients of the bad predictors mentioned above 0. This makes Lasso useful in feature selection.

Lasso however struggles with some types of data. If the number of predictors (p) is greater than the number of observations (n), Lasso will pick at most n predictors as non-zero, even if all predictors are relevant. Lasso will also struggle with colinear features (they're related/correlated strongly), in which it will select only one predictor to represent the full suite of correlated predictors. This selection will also be done in a random way, which is bad for reproducibility and interpretation.

It is important to note that if lambda=0, we effectively have no regularization and we will get the OLS solution. As lambda tends to infinity, the coefficients will tend towards 0 and the model will be just a constant function.

Feature selection

L-2 Ridge

https://towardsdatascience.com/bias-variance-and-regularization-in-linear-regression-lasso-ridge-and-elastic-net-8bf81991d0c

Ridge Regression

$$\hat{\beta}^{ridge} = \underset{\beta \in \mathbb{R}}{argmin} \|y - XB\|_2^2 + \lambda \|B\|_2^2$$

Thanks to Kyoosik Kim

Ridge regression also adds an additional term to the cost function, but instead sums the squares of coefficient values (the L-2 norm) and multiplies it by some constant lambda. Compared to Lasso, this regularization term will decrease the values of coefficients, but is unable to force a coefficient to exactly 0. This makes ridge regression's use limited with regards to feature selection. However, when p > n, it is capable of selecting more than n relevant predictors if necessary unlike Lasso. It will also select groups of colinear features, which its inventors dubbed the 'grouping effect.'

Much like with Lasso, we can vary lambda to get models with different levels of regularization with lambda=0 corresponding to OLS and lambda approaching infinity corresponding to a constant function.

Interestingly, analysis of both Lasso and Ridge regression has shown that neither technique is consistently better than the other; one must try both methods to determine which to use (Hou, Hastie, 2005).

Combining L1 and L2 norms

Elastic Net

$$\hat{\beta} \equiv \underset{\beta}{\operatorname{argmin}}(\|y - X\beta\|^2 + \lambda_2\|\beta\|^2 + \lambda_1\|\beta\|_1).$$

Thanks to Wikipedia

Elastic Net includes both L-1 and L-2 norm regularization terms. This gives us the benefits of both Lasso and Ridge regression. It has been found to have predictive power better than Lasso, while still performing feature selection. We therefore get the best of both worlds, performing feature selection of Lasso with the feature-group selection of Ridge.

Elastic Net comes with the additional overhead of determining the two lambda values for optimal solutions.

https://medium.com/@jayeshbahire/lasso-ridge-and-elastic-net-regularization-4807897cb722

OLS Basic Regression

Load data
Split Training
Split Testing
Create Model
Find Training Error
Predict Testdata
Estimate testing error
Run Regularization
Compare

```
rsquare <- function(given, predicted) {
    sse <- sum((predicted - given)^2)
    sst <- sum(given^2)
    rsq <- 1 - sse / sst

    # For this post, impose floor...
    if (rsq < 0) rsq <- 0

    return (rsq)
  }
msd<-function(given,predicted) {
sqrt(mean((given-predicted)^2))
}
 mtcarshp<-mtcars[,c("mpg","wt","drat","hp")]

 ntrain<-round(0.8*nrow(mtcarshp))
 ntest<-nrow(mtcarshp)-ntrain
 allidx<-1:nrow(mtcarshp)
 trainidx<-sample(allidx,ntrain,rep=FALSE)
 testidx<-allidx[-trainidx]

 traindata<-mtcarshp[trainidx,]
 testdata<-mtcarshp[testidx,]
```

https://medium.com/@jayeshbahire/lasso-ridge-and-elastic-net-regularization-4807897cb722

OLS Basic Regression

Load data
Split Training
Split Testing
Create Model
Find Training Error
Predict Testdata
Estimate testing error
Run Regularization
Compare

```
lm.model<-lm(hp~mpg+wt+drat,data=traindata)
traindata$train.predicted.hp<-predict(lm.model,
traindata[,c("mpg","wt","drat")])
#training error
train_error<-
rsquare(traindata$hp,traindata$train.predicted.hp)
train_msd<-
sqrt(mean((traindata$hp-traindata$train.predicted.hp)^2))
#test error for lm
testdata$test.predicted.hp<-
predict(lm.model,testdata[,c("mpg","wt","drat")])
test_error<-rsquare(testdata$hp,testdata$test.predicted.hp)
test_msd<-s
qrt(mean((testdata$hp-testdata$test.predicted.hp)^2))
```

```
model_stats<-data.frame(#run_names=c("proportion","rsquare","msd"),
+ lm.train=c(1,train_error,train_msd),
+ lm.test=c(1,test_error,test_msd),stringsAsFactors=F)
```

```
> model_stats
    lm.train    lm.test
1  1.0000000  1.0000000
2  0.9395461  0.8616764
3 41.3058453 48.1315269
```

Model captured 94%/86% of the variation during train/test correspondingly

https://medium.com/@jayeshbahire/lasso-ridge-and-elastic-net-regularization-4807897cb722

OLS Basic Regression

Is it possible to improve?

Load data
Split Training
Split Testing
Create Model
Find Training Error
Predict Testdata
Estimate testing error
Run Regularization
Compare

Let us regularization, we need lambda.
If we know the best lambda, we are done.
But to find that we create a sequence of lambdas
And calculate Y for all of them.

```
require(ggplot2)
require(glmnet)
y<-traindata$hp

x<-as.matrix(traindata[,c("mpg","wt","drat")])
lambdas<-10^seq(3,-2,by=-0.1)
glm.fit<-glmnet(x,y,alpha=0,lambda=lambdas)
all_coef<-coef(glm.fit)
betas<-all_coef[2:4,]
fitval<-x%*%betas
```

Fitval is all the y values fitted by the coefficients generated by glmnet.
We have to find the best fit -- very laborious work

https://medium.com/@jayeshbahire/lasso-ridge-and-elastic-net-regularization-4807897cb722

Cross Validation

Load data
Split Training
Split Testing
Create Model
Find Training Error
Predict Testdata
Estimate testing error
Run Regularization
Compare

```
cv.glm.fit<-cv.glmnet(x,y,alpha=0,lambda=lambdas,nfolds=5)

cv.glm.fit$lambda.min
```

We can start with the lambda and find the best fit.

```
traindata$train.penalized.hp<-predict(
glm.fit,newx=as.matrix(traindata[,c("mpg","wt","drat")]),
s=cv.glm.fit$lambda.min,type='response')

# store results
model_stats<-cbind(model_stats,
train.penalized.hp=c(proportion=1,
rsquare=rsquare(traindata$hp,traindata$train.penalized.hp),
msd=msd(traindata$hp,traindata$train.penalized.hp)))
```

https://medium.com/@jayeshbahire/lasso-ridge-and-elastic-net-regularization-4807897cb722

OLS – Linear Model

```
> t(model_stats)
                       proportion   rsquare      msd
lm.train                     1.00 0.9317804 43.97826
lm.test                      1.00 0.9299146 33.68764
train.penalized.hp           1.00 0.9310128 44.22497
tr.p.hp2                     0.98 0.9310381 44.21685
tr.p.hp3                     0.88 0.9311583 44.17832
tr.p.hp4                     0.80 0.9312467 44.14993
tr.p.hp5                     0.78 0.9312689 44.14280
tr.p.hp6                     0.76 0.9312911 44.13566
tr.p.hp7                     0.66 0.9313948 44.10236
tr.p.hp8                     0.50 0.9315417 44.05511
tr.p.hp9                     0.40 0.9316201 44.02987
tr.p.hp10                    0.20 0.9317353 43.99278
tr.p.hp11                    0.10 0.9317680 43.98223
tr.p.hp12                    0.05 0.9317771 43.97932
tr.p.hp13                    0.00 0.9317803 43.97827
tr.p.hp14                   -0.05 0.9317803 43.97827
tr.p.hp15                   -0.80 0.9317803 43.97827
tr.p.hp16                   -0.90 0.9317803 43.97827
tr.p.hp17                   -1.00 0.9317803 43.97827
tr.p.hp18                   -2.00 0.9317803 43.97827
tr.p.hp19                   -3.00 0.9317803 43.97827
```

Run same experiments over test

```
> t(model_stats)
                   proportion  rsquare      msd
lm.train                 1.00 0.9317804 43.97826
lm.test                  1.00 0.9299146 33.68764
train.penalized.hp       1.00 0.9310128 44.22497
tr.p.hp2                 0.98 0.9310381 44.21685
tr.p.hp3                 0.88 0.9311583 44.17832
tr.p.hp4                 0.80 0.9312467 44.14993
tr.p.hp5                 0.78 0.9312689 44.14280
tr.p.hp6                 0.76 0.9312911 44.13566
tr.p.hp7                 0.66 0.9313948 44.10236
tr.p.hp8                 0.50 0.9315417 44.05511
tr.p.hp9                 0.40 0.9316201 44.02987
tr.p.hp10                0.20 0.9317353 43.99278
tr.p.hp11                0.10 0.9317680 43.98223
tr.p.hp12                0.05 0.9317771 43.97932
tr.p.hp13                0.00 0.9317803 43.97827
tr.p.hp14               -0.05 0.9317803 43.97827
tr.p.hp15               -0.80 0.9317803 43.97827
tr.p.hp16               -0.90 0.9317803 43.97827
tr.p.hp17               -1.00 0.9317803 43.97827
tr.p.hp18               -2.00 0.9317803 43.97827
tr.p.hp19               -3.00 0.9317803 43.97827
```

```
test.penalized.hp        1.00 0.9435833 30.22462
tst.p.hp2                0.98 0.9433878 30.27694
tst.p.hp3                0.88 0.9423940 30.54154
tst.p.hp4                0.80 0.9415793 30.75673
tst.p.hp5                0.78 0.9413614 30.81406
tst.p.hp6                0.76 0.9411375 30.87281
tst.p.hp7                0.66 0.9399995 31.16983
tst.p.hp8                0.50 0.9380092 31.68259
tst.p.hp9                0.40 0.9366343 32.03200
tst.p.hp10               0.20 0.9335746 32.79624
tst.p.hp11               0.10 0.9318559 33.21781
tst.p.hp12               0.05 0.9309258 33.44374
tst.p.hp13               0.00 0.9300303 33.65982
tst.p.hp14              -0.05 0.9300303 33.65982
tst.p.hp15              -0.80 0.9300303 33.65982
tst.p.hp16              -0.90 0.9300303 33.65982
tst.p.hp17              -1.00 0.9300303 33.65982
tst.p.hp18              -2.00 0.9300303 33.65982
tst.p.hp19              -3.00 0.9300303 33.65982
tst.p.hp20              -4.00 0.9300303 33.65982
```

WE get to review the R-Squared and MSD

the test cases...

```
testdata$tst.p.hp18<-predict(glm.fit,newx=as.matrix(testdata[,c("mpg","wt","drat")]),
s=-2*cv.glm.fit$lambda.min,type='response')
model_stats<-cbind(model_stats,tst.p.hp18=c(-2.0,rsquare=rsquare(testdata$hp,
testdata$tst.p.hp18),msd=msd(testdata$hp,testdata$tst.p.hp18)))
proportion<-c(proportion,-2.0)
sum((testdata$tst.p.hp18-testdata$hp)^2)


testdata$tst.p.hp19<-predict(glm.fit,newx=as.matrix(testdata[,c("mpg","wt","drat")]),
s=-3*cv.glm.fit$lambda.min,type='response')
model_stats<-cbind(model_stats,tst.p.hp19=c(-3.0,rsquare=rsquare(testdata$hp,
testdata$tst.p.hp19),msd=msd(testdata$hp,testdata$tst.p.hp19)))
proportion<-c(proportion,-3.0)
sum((testdata$tst.p.hp19-testdata$hp)^2)

testdata$tst.p.hp20<-predict(glm.fit,newx=as.matrix(testdata[,c("mpg","wt","drat")]),
s=-4*cv.glm.fit$lambda.min,type='response')
model_stats<-cbind(model_stats,tst.p.hp20=c(-4.0,rsquare=rsquare(testdata$hp,
testdata$tst.p.hp20),msd=msd(testdata$hp,testdata$tst.p.hp20)))
proportion<-c(proportion,-4.0)
sum((testdata$tst.p.hp20-testdata$hp)^2)
```

Regularization and CV are important steps to reduce overfitting.

Other Penalization Methods

– Ridge completed beta-squared all predictors are kept
– Lasso –linear and so some betas are shrunk to zero.
– combination of Ridge and Lasso using elasticnet

Concluding Remarks

Let us take stock

We have run linear regression
And we tried to improve using regularization and cross validation. To minimize overfitting....
Both CV and Penalization minimize variation not bias.

Nowwe will veer off into logistic regression and deveop formalism for various classifiers.

RMSE and R-Squared are not applicable.

We have to AUC etc to compare Classifier performance.