ELEMENTS OF DATA SCIENCE AND STATISTICAL LEARNING

FALL 2017

Week 5

OUTLINE

- Importance of physical model for feature selection
- Best subset selection
- Backward/forward/alternating feature selection
- Model independent feature selection
- Shrinkage methods (ridge and lasso)
- Principal components regression
- Wrong and right way to perform cross-validation

LINEAR MODELS SO FAR

- We have examined linear regression in great detail
  - (Relatively) simple, interpretable models
  - Efficient implementation, many analytical results, properties are well understood
  - Great flexibility, especially when variable transformations, polynomial regression and interactions are considered
  - May be more than enough to describe all the trends that can be gleaned from the data, given the amount of data points available and strength of the noise, regardless of what the "true" underlying dependence might be
    - Remember our simulated toy example where the second-order term in the "true" dependence on a predictor variable was washed out by the noise
  - Linear model may (and will) overfit when large number of predictors is considered (technically, applies to any model)
    - We examined the metrics for accuracy of the fit and, most importantly for the predictive accuracy of the model
  - How do we choose the "right" model? Does the right one even exist? How to select few predictors out of many?
- Today's discussion:
  - Model selection and regularization

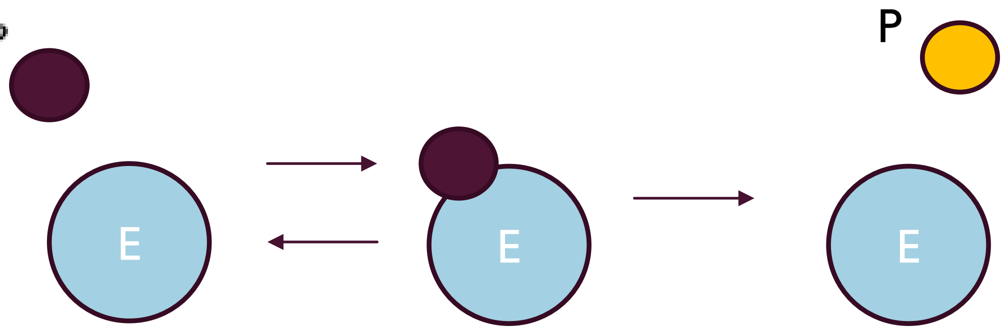
IMPORTANCE OF PHYSICAL MODEL

- It is clear that the best possible method of model selection is *understanding the problem*
- Example: let's assume we are studying air drag experienced by a car
  - We figure out that alongside with the drag (output) we should measure speed *V*, car's frontal cross-section *S* and the air density ρ. We can try building *a* model based on those variables, but…
  - Even without knowing much Physics, we can make a simple argument based on the dimensionality analysis:
  - Let's assume $F_{\text{drag}} \propto V^a \rho^b S^c$; drag is force (g · cm/$s^2$), hence
  - g · cm/s^2 = $cm^a/s^a \cdot g^b/cm^{3b} \cdot cm^{2c}$, equating the dimensionalities we get: (g) *b=1, (s) a*=2, (cm) a+2c-3b=1, i.e. c=1
  - Now we *know* that drag is proportional to $S \cdot V^2 \cdot \rho$ (try finding such interaction effect just from the data!)
  - The coefficient of proportionality of course depends on car's aerodynamics and might further depend on the predictor variables (and it does, for instance it is ~1/V at small speeds) – but we still have a much better chance of modeling it properly even without knowing any more Physics if we use correct variables (e.g. normalize measured drag by $SV^2\rho$)

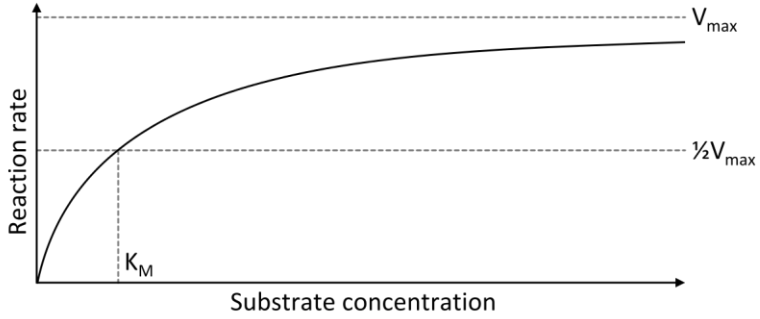
IMPORTANCE OF PHYSICAL MODEL: CONTINUED

- Example 2: Enzymatic kinetics

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} ES \xrightarrow{k_2} E + P$$



- Michaelis-Menten (1913): reaction rate $v = \dfrac{d[P]}{dt} = \dfrac{V_{\max}[S]}{K_M + [S]}$
- How to fit (for coefficients)? The dependence is beyond linear model domain!
- With simple algebra: $\dfrac{S}{v} = \dfrac{K_M + S}{V_{\max}}, \quad \Rightarrow \quad \dfrac{1}{v} = \dfrac{1}{V_{\max}} + \dfrac{K_M}{V_{\max}} \cdot \dfrac{1}{S}$



- Moral: knowledge of the subject domain and understanding of relevant phenomena beats (or at least greatly improves) any statistical model! Many advanced, domain-specific models include some statistical fitting but are also heavily based on physical principles:
  - http://www.huffingtonpost.com/michael-e-mann/nate-silver-climate-change_b_1909482.html

MANUAL FEATURE SELECTION

- Use domain knowledge (or bright insight) to preprocess raw data
- Many examples in image classification: before attempting to train a model for finding a cat in an image, we can for instance:
  - Convert to grayscale (unless we believe additional information carried by color outweighs huge increase in the predictor variable domain)
  - Downsample/smooth the image (do we need fine details at 25 Megapixel resolution or a cat is a cat is a cat even on a much more crude pixel grid of 512x512?) – note that each pixel *is* an independent predictor variable! Of course the feasibility of downsampling depends on whether we want to be able to discern a tiny cat in a window in a huge panoramic image.
  - Edge detection: instead of interpreting raw individual pixels, find less primitive objects first such as lines (this is in fact how human visual cortex works). Theoretically, some models (e.g. neural networks) can accomplish the discovery of such relevant features automatically, but at very high computational expense, *and* the more you need to discover the more data you must collect!
  - And many, many more (smoothing, blurring, segmentation,…)

FEATURE SELECTION

- Now that we understand the depth and complexity of the problem…
- We are going to look at some of the simplest, most straightforward, and *general* methods (i.e. what can be done aside of using deep domain knowledge!)
  - The problem: there are many variables available for trying to model the outcome
  - We know the models tend to overfit or exhibit other kinds of undesired behavior if "everything is just thrown into the mix"
  - Can we use some rational procedure for selecting an "optimal" set of "best" variables?

OUTLINE

- Importance of physical model for feature selection
- Best subset selection
- Backward/forward/alternating feature selection
- Model independent feature selection
- Shrinkage methods (ridge and lasso)
- Principal components regression
- Wrong and right way to perform cross-validation

BEST SUBSET SELECTION

- The simplest answer to the question "which model is the best one?":
- "Try them all!"
  - Assuming we have a metric for comparing different models, of course, but we do: adjusted model statistics, information criteria, cross-validation
- Algorithm is straightforward:

> • Start with model M_0 with 0 predictors (what is this model?)
> • For k = *1*…*p*: (where *p* is the total number of variables measured in the dataset)
> Fit all models that that contain exactly *k* predictors out of *p*
> Pick the best among these models of size *k*, call it M_k (what does "the best" mean?)
> • Select a single best model from among M_0, …, M_p by using cross-validation

BEST SUBSET SELECTION: CONTINUED

- Exhaustive search for the best combination of predictor variables is very expensive
  - If p predictor variables are available, then:
  - p models with 1 predictor, $p(p-1)/2$ models with 2 predictors, $p(p-1)(p-2)/6$ models with 3 predictors, … , C_p^k models with k predictors
  - Total number of models to consider is 2^p (all combinations from 0 to p predictors).
  - Effective number of "predictors" p is even larger than the number of measured variables when polynomial regression is used
  - Ideally, we would want to characterize each possible model by its cross-validation MSE. In practice, we already compromise by e.g. selecting the model with the best *fit* (or at best *adjusted* R^2) for the given k
  - Many modern datasets have hundreds or thousands of features making best subset selection prohibitively expensive

BEST SUBSET SELECTION: EXAMPLE

- Let us use the Algae dataset again (the code continues after what was shown last week)

```
# keep only the variables we are going to use (some are log-transformed):
> a = a[,c("LAG1","season","size","velocity","C1","C2","LC3","C4","LC5","LC6","LC7","LC8")]
# also, for the sake of a simpler exercise we might consider only continuous predictors:
> ac = a[,c("LAG1", "C1","C2","LC3","C4","LC5","LC6","LC7","LC8")]

> library(leaps) # install the package first if needed

> subset.full = regsubsets(LAG1 ~ . , ac)
> summary(subset.full)
Subset selection object
Call: regsubsets.formula(LAG1 ~ ., ac)
1 subsets of each size up to 8
...
Selection Algorithm: exhaustive
         C1  C2  LC3 C4  LC5 LC6 LC7 LC8
1  ( 1 ) " " " " " " " " " " " " "*" " "
2  ( 1 ) " " " " " " " " " " " " "*" "*"
3  ( 1 ) " " " " " " " " " " "*" "*" "*"
4  ( 1 ) " " " " "*" " " " " "*" "*" "*"
...
```

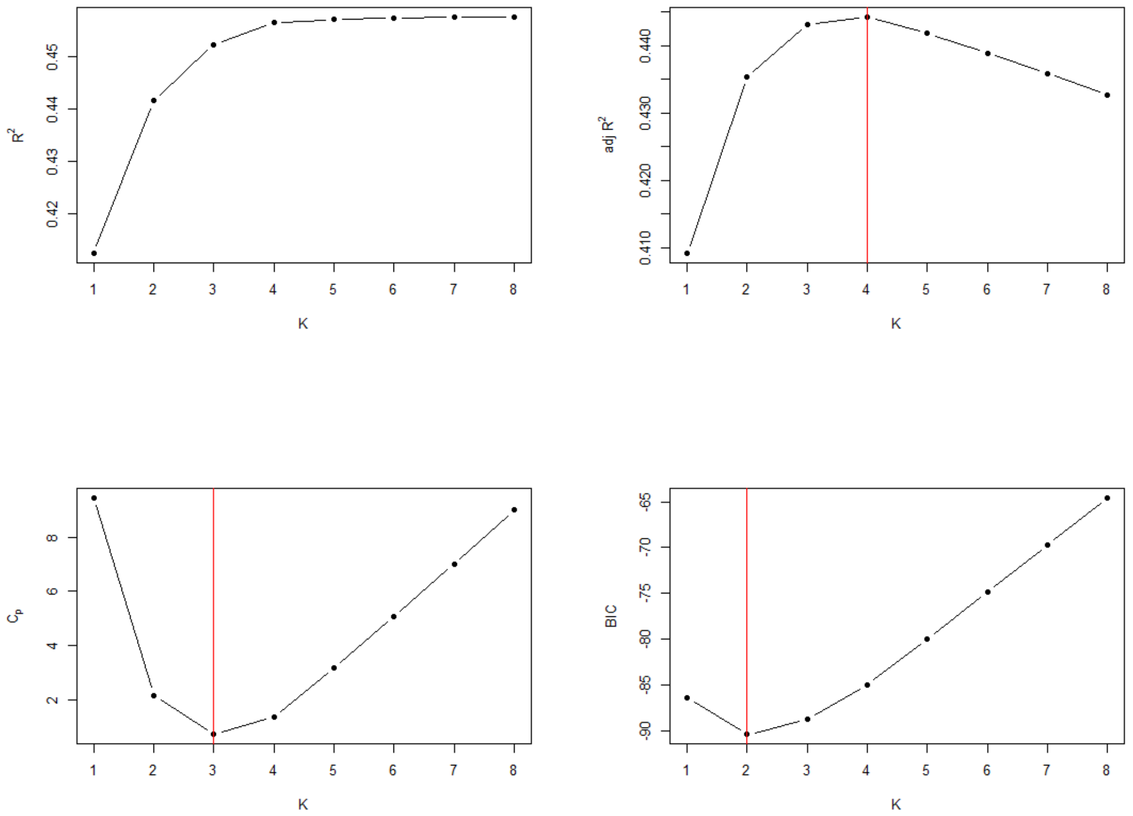
EXAMINING SUBSETS

- Some fit quality statistics are already calculated by `regsubsets()` function:
  - In the example shown below we will extract from the summary and plot R^2, adjusted R^2, C_p and BIC vs the number of variables in the model , K
  - $C_p = \frac{1}{n}(\mathrm{RSS} + 2\ K\ \hat{\sigma}^2)$ is the *estimate* of the MSE we should expect on a *test* set. It is an analytical result that is *asymptotically* correct (as compared to brute force cross-validation)

```
names(summary(subset.full))
[1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
oldpar=par(mfrow=c(2,2))
plot(summary(subset.full)$rsq,pch=19,cex=0.8,type="b",xlab="K",ylab=expression(R^2))
plot(summary(subset.full)$adjr2,pch=19,cex=0.8,type="b",xlab="K",
       ylab=expression(paste("adj ",R^2)))
abline(v=which.max(summary(subset.full)$adjr2),col="red")
plot(summary(subset.full)$cp,pch=19,cex=0.8,type="b",xlab="K",ylab=expression(C[p]))
abline(v=which.min(summary(subset.full)$cp),col="red")
plot(summary(subset.full)$bic,pch=19,cex=0.8,type="b",xlab="K",ylab="BIC")
abline(v=which.min(summary(subset.full)$bic),col="red")
par(oldpar)
```

BEST SUBSET

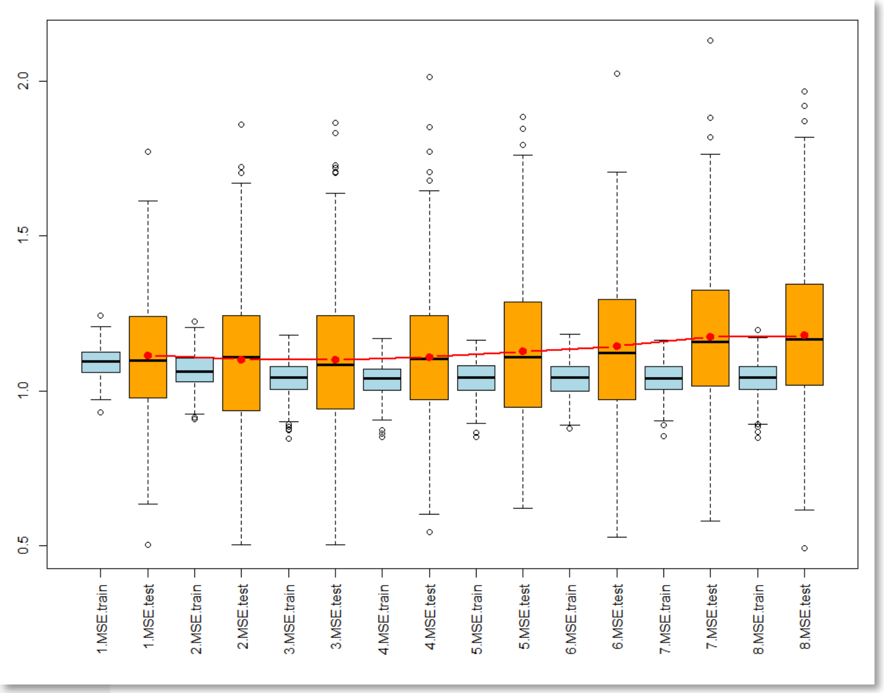
- The code in the previous slide generates the plot shown below
- Different criteria are not perfectly consistent (which is expected) and suggest the optimal model size between K=2 and K=4. The differences between these models do not appear to be striking
- Let us try brute-force cross-validation next



CROSS-VALIDATING VARIABLE SUBSETS

- Use function xval() we discussed last week! Takes a formula and data and does the rest!

```
> s = summary(subset.full)
> s$which
  (Intercept)    C1    C2   LC3    C4   LC5   LC6  LC7   LC8
1        TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
2        TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE  TRUE
…
> mse=list()
> mean.mse=numeric()
> for( i in 1:8 ) {
     l = xval(LAG1 ~., data=ac[,s$which[i,]],prefix=i)
     mean.mse[i] = mean(l[[2]])
     mse=c(mse,l)
  }
> boxplot(mse,col=c('lightblue','orange'),las=3)
> points(2*(1:8),mean.mse,pch=19,cex=1.2,type='b',col="red",lwd=2)
> which.min(mean.mse); min(mean.mse)
[1] 3
[1] 1.099246
```



OUTLINE

- Importance of physical model for feature selection
- Best subset selection
- Backward/forward/alternating feature selection
- Model independent feature selection
- Shrinkage methods (ridge and lasso)
- Principal components regression
- Wrong and right way to perform cross-validation

STEPWISE SELECTION: MOVING FORWARD

- Exhaustive search is powerful but often impractical.
- A heuristic approach: let's add one predictor at a time:

  - Start with model M_0 with 0 predictors
  - For k = $0 \ldots p\text{-}1$: (where p is the total number of variables measured in the dataset)
    Consider all $p\text{-}k$ models in which exactly one yet unused predictor is added to M_k
    Pick the best among these $p\text{-}k$ models, call it M_{k+1}
  - Select a single best model from among $M_0, \ldots, M_p$ by using direct cross-validation or C_p, BIC, adjusted R^2 etc

- Number of models that we need to fit: $p+(p-1)+(p-2)+\ldots+1=p(p+1)/2$
- How do we pick the best model at given k – again, we might use expensive technique like cross-validation but in practice we use something simpler such as (adjusted) R^2

STEPWISE SELECTION: EXAMPLE

- On the algae dataset, a simpler (and faster!) forward selection approach results in exactly the same sequence of models as suggested by the thorough best subset selection we performed earlier!
  - There is no need to reexamine fit statistics or cross-validation MSE, of course – these are the same models
  - Forward selection is not guaranteed of course to always give the same result as best subset selection

```
> subset.fwd=regsubsets(LAG1~.,ac,method = "forward")
> sf = summary(subset.fwd)
> sf
Subset selection object
Call: regsubsets.formula(LAG1 ~ ., ac, method = "forward")
8 Variables  (and intercept)
...
1 subsets of each size up to 8
Selection Algorithm: forward
        C1  C2  LC3 C4  LC5 LC6 LC7 LC8
1  ( 1 ) " " " " " " " " " " " " "*" " "
2  ( 1 ) " " " " " " " " " " " " "*" "*"
...
> all(s$which==sf$which)
[1] TRUE
```

BACKWARD STEPWISE SELECTION

- Backward selection:
  - As the name suggests: same thing but… backwards!

- Start with model M_p with all p predictors included
- For k = p, p-$1, \ldots$:
  Consider all k models in which exactly one predictor is removed from M_k
  Pick the best among these k models, call it M_{k-1}
- Select a single best model from among $M_0, \ldots, M_p$ by using direct cross-validation or C_p, BIC, adjusted R^2 etc

  - Just use `method="backward"` in the call to `regsubsets()`, the rest of the code and the structure of the data will be the same
  - Also gives the same hierarchy of best models on the algae dataset
  - Can be applied only when $p < n$ where n is the number of observations

ALTERNATING STEPS AND OTHER VARIATIONS

- The problem of either forward or backward stepwise selection is that we always follow the same path, which is not guaranteed to be optimal
  - The first feature X1 we add in forward selection is always the one that exhibits the strongest correlation with the outcome
  - What if among the models with two predictors, the best fit is in fact provided by variables X2 and X3? We will never find such a model, because we are only allowed to *add* to what's already selected (X1)
  - Backward selection has similar problems: we remove *one* feature X1 such that the removal leads to the least degradation of the fit. But what if when we are allowed to remove two features at once, the least degradation would occur upon removal of X2, X3? Again, we will never discover that if we are only allowed to remove one variable (X1) in one step, and then another in the next step.
- Alternating steps: keep building the chain of forward selected models, then switch direction to backward and remove one/a few variable(s) that don't affect the fit much (anymore). Switch to forward again…
  - It is also possible of course to start from backward selection and periodically switch to forward in the hope of adding back variables that ceased being "unimportant"
- Ensemble approaches (probabilistic): perform a few stepwise selection runs, in each run select (randomly, with proper weights) one of more likely variables (i.e. large improvement in the fit) but not necessarily the best one

OUTLINE

- Importance of physical model for feature selection
- Best subset selection
- Backward/forward/alternating feature selection
- Model independent feature selection
- Shrinkage methods (ridge and lasso)
- Principal components regression
- Wrong and right way to perform cross-validation

MODEL-INDEPENDENT STEPWISE SELECTION

- Yet another related approach is to look at correlations among the predictors and correlations between predictors and outcomes. (the original paper suggested looking at mutual information, see https://www.researchgate.net/publication/3301850_Using_Mutual_Information_for_Selecting_Features_in_Supervised_Neural_Net_Learning ; the approach shown here is a simplified adaptaion)
- The selection is performed as follows:
  - Calculate all the correlations (once!), C(Y,Xi) and C(Xi,Xj), for all i,j
  - Select the predictor that exhibits the strongest correlation with the outcome (argmax $_i$ C(Y,X$_i$))
  - In each step afterwards, favor features that strongly correlate with the outcome but also penalize correlations with features that are already selected. More formally:

    - Start with empty set S of selected predictors
    - For k = *1…p*:
      Select the predictor $j \notin S$ that maximizes expression $C(Y, X_j) - \alpha \sum_{i\in S} C(X_i, X_j)$
      Add *j* to *S*
    - Select a single best model from among *S[1], S[1:2], S[1:3], …*by using direct cross-validation or C_p, *BIC*, adjusted R^2 etc

WHY MODEL-INDEPENDENT?

- Isn't it always better to select features that provide the largest improvement to the fit of the particular model we are using?
  - It is!
- Fitting multiple models, even in the restricted space of forward selection can be very costly and outright prohibitive
  - linear models are not the only models in the world; neural networks, for instance, are very expensive computationally!
  - Number of available features (predictor variables) can be *very* large. Think about 20,000+ gene expression levels measured in a typical high-throughput genomic experiment; suppose we want to build a model that uses gene expression(s) as variables for predicting cellular response, condition (e.g. stress/no stress), disease etc. Let's assume we want a very simple and "light" model that uses "only" 20 variables (genes). Calculate how many choices of 20 out of 20000 are possible (best subset selection). Calculate how many models would need to be considered (and fitted!) by stepwise forward selection

MODEL-INDEPENDENT SELECTION: IMPLEMENTATION

- The function shown below calculates correlations $C(Y,X_i)$ and $C(X_i, X_j)$ and returns them as two separate elements of a list

```
correlations = function(Y,data) {
    predictors=1:ncol(data)
    if ( length(Y) == 1 ) {
      if ( is.character(Y) ) {
          ycol = match(Y,names(data))
      } else { ycol=Y }
      Y = data[[Y]]
      predictors = predictors[ -ycol ]
    }
    # note that we build a simplified function here that
    # can only take care of continuous variables
    corr.mat=cor( cbind(Y,data[,predictors]), use="pair" )

    return(list(outcome=corr.mat[-1,1],
                predictors=corr.mat[-1,-1]))
}
```

MODEL-INDEPENDENT SELECTION: CONTINUED

- The following function takes pre-computed correlations and performs forward selection of N predictors by maximizing the selection score (as defined earlier) in every step. 'a' is the tuning parameter α of the score function

sum sum sum

```
            C1          C2          LC3        ...
C1    1.0000000 -0.16310523  0.1552826     ...
C2   -0.1631052  1.00000000 -0.3909205
LC3   0.1552826 -0.39092053  1.0000000
C4   -0.1014517  0.06187919  0.4975654
LC5   0.1263912 -0.41797967  0.5868157
...      ...        ...         ...
```

Illustration of term2 calculation. Suppose that C2 and LC5 are already selected

$$C(Y,X_j) - \alpha \sum_{i \in S} C(X_i,X_j)$$

```
forward.select = function(corr, a, N) {
    selected = which.max(abs(corr$outcome))
    if ( N <= 1 ) return(rownames(corr$predictors)[selected])
    term1 = abs(corr$outcome)
    for ( i in 1:(N-1) ) {
        term2 = apply(abs(corr$predictors[selected,,drop=F]),2,sum)
        score = term1 - a*term2
        # find NOT yet selected featire with largest score:
        for ( s in order(score,decreasing=T) ) {
            if ( ! s %in% selected ) {
                selected = c(selected,s)
                break
            }
        }
    }
    return(rownames(corr$predictors)[selected])
}
```

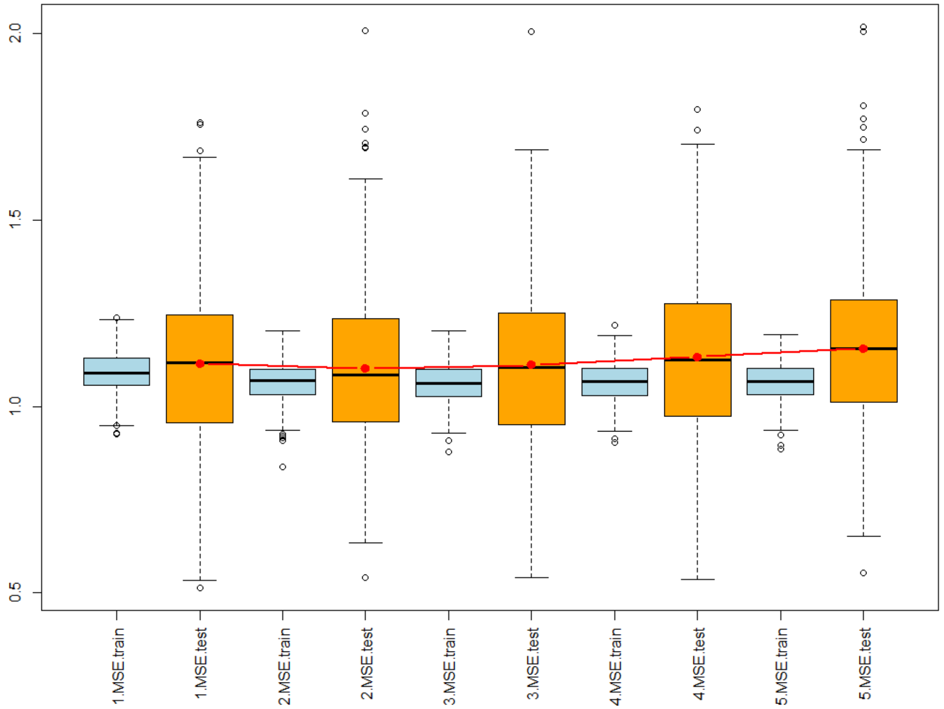
MODEL-INDEPENDENT SELECTION: EXAMPLE

- The code below shows a usage example for our new functions
- Note that the "optimal" model(s) we select are different from what we have seen earlier (expected!)

```
> cor.data = correlations("LAG1",ac)
> cor.data$outcome
        C1         C2        LC3         C4        LC5        LC6        LC7        LC8 
-0.2893086  0.3088848 -0.5776702 -0.3255383 -0.4490481 -0.6531987 -0.6721586 -0.5365388 
> cor.data$predictors[1:3,1:5]
            C1         C2        LC3          C4        LC5
C1   1.0000000 -0.1631052  0.1552826 -0.10145169  0.1263912
C2  -0.1631052  1.0000000 -0.3909205  0.06187919 -0.4179797
LC3  0.1552826 -0.3909205  1.0000000  0.49756542  0.5868157
> features=forward.select(cor.data,a=0.5,N=5)
> features
[1] "LC7" "LC8" "C4"  "C2"  "C1"
> forward.select(cor.data,a=2,N=5)  # ← different predictor cross-correlation penalty!
[1] "LC7" "C1"  "C4"  "C2"  "LC8"
```

CROSS-VALIDATION

- However if we rerun cross-validation code in a way very similar to what we did before (try it as an exercise!), we will see that the models we are now selecting are nearly as good – another confirmation of the fact that we observed before: apparently many models with very different predictors result in nearly the same prediction accuracy in this dataset.



GENERALIZATIONS OF MODEL-INDEPENDENT SELECTION

- Same modifications can be applied to the model-independent selection process:
  - Alternating forward-backward steps (in the backward step, *remove* one of already selected predictors that has the lowest score)
  - Probabilistic path building
- Can be generalized to include categorical variables
  - Need some metric for continuous-categorical and categorical-categorical associations that's on the same scale and comparable to correlation coefficients. Possibilities: run a linear model $Y \sim X_{cat}$, use R^2 (which as we remember is normally a square of the correlation coefficient); use intraclass correlation (ICC), Cramer's V for categorical vs categorical etc.
- Finally, correlation coefficient captures only linear dependence. The original version of the algorithm discussed here was using mutual information (MI) in place of correlation.

OUTLINE

- Importance of physical model for feature selection
- Best subset selection
- Backward/forward/alternating feature selection
- Model independent feature selection
- Shrinkage methods (ridge and lasso)
- Principal components regression
- Wrong and right way to perform cross-validation

SHRINKAGE METHODS

- "Shrinkage" in a sense of shrinking the model coefficients towards zero
- It's the same idea of penalizing complex models that we have seen before
- Use all variables, but the penalty is applied at the outset: instead of RSS the following function is minimized:

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Ridge regression

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j| = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Lasso

WHY DO SHRINKAGE METHODS WORK?

- Shrinkage methods affect the bias-variance trade-off. With many variables included into the model and unrestricted optimization allowed, the model is prone to fitting noise (high variance)
- By restricting the coefficients of the model, we make it harder to fit any random twist and bend in the noise (the model becomes less flexible, lower variance), but the bias somewhat increases (since we effectively *prevent* the model from being the best possible fit to the data). Since with large number of variables included we usually err on the high variance side, the process moves us towards the optimal balance between the variance and the bias
- Note that despite apparent similarity, ridge regression and lasso implement very different restrictions
  - With ridge, all the coefficients of the fitted model are usually non-zero. It is just their magnitude that gets shrunk
  - With lasso, at least some coefficients of the fitted model become 0 (model terms get eliminated), hence lasso (but not ridge) is a form of variable selection approach!

RIDGE: EXAMPLE

- glmnet() can do both ridge (alpha=0) and lasso (alpha=1) regression and it can interpolate between the two
- predict() can be applied to glmnet model to predict either outcomes at new data values (new=) or the values of the model coefficients at new value of lambda (type="coefficients")

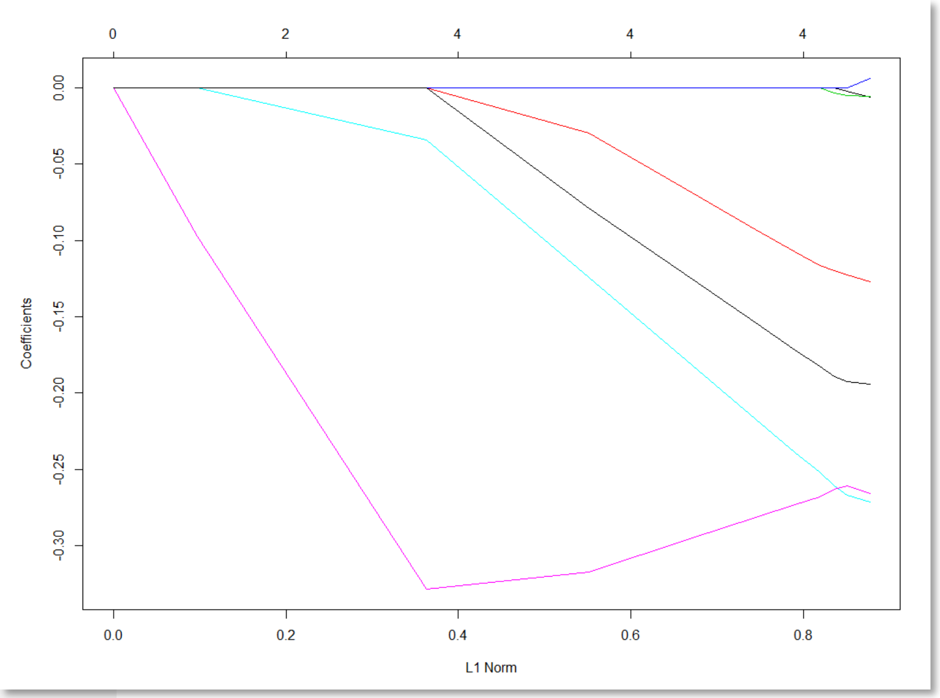
```
> library(glmnet)  # install if needed
> na.rows = apply(ac,1,function(x) { any(is.na(x)) }) # glm does not like NA
> l.grid=10^(seq(2,-2,length=20))
> M.ridge=glmnet(as.matrix(ac[!na.rows,-1]),ac$LAG1[!na.rows],alpha=0,lambda = l.grid)
> coef(M.ridge)[1:3,1:5]
3 x 5 sparse Matrix of class "dgCMatrix"
                     s0          s1          s2          s3          s4
(Intercept)  2.159179299  2.27547176  2.449552562  2.699012327  3.035026008
C1          -0.008286852 -0.01303874 -0.020155391 -0.030354029 -0.044063940
C2           0.002000158  0.00312160  0.004763631  0.007031415  0.009899582
```

Columns correspond to the successive values of the penalty weight λ in the grid used in the function call (l.grid)

LASSO: EXAMPLE

- Same code with alpha=1 can be used to perform Lasso regression fit

```
> l.grid=10^(seq(2,-2,length=20))
> M.lasso=glmnet(as.matrix(ac[!na.rows,-1]),ac$LAG1[!na.rows],
        alpha=1,lambda = l.grid)
> coef(M.lasso)[,10:13]
9 x 4 sparse Matrix of class "dgCMatrix"
                  s9          s10          s11          s12
(Intercept) 1.956194  2.39501358  3.56342053  4.08056030
C1          .         .           .           .
C2          .         .           .           .
LC3         .         .           .          -0.02994176
C4          .         .           .           .
LC5         .         .           .           .
LC6         .         .          -0.03400938 -0.12444456
LC7         .        -0.09729983 -0.32858523 -0.31732755
LC8         .         .           .          -0.07914471
> plot(M.lasso)
```

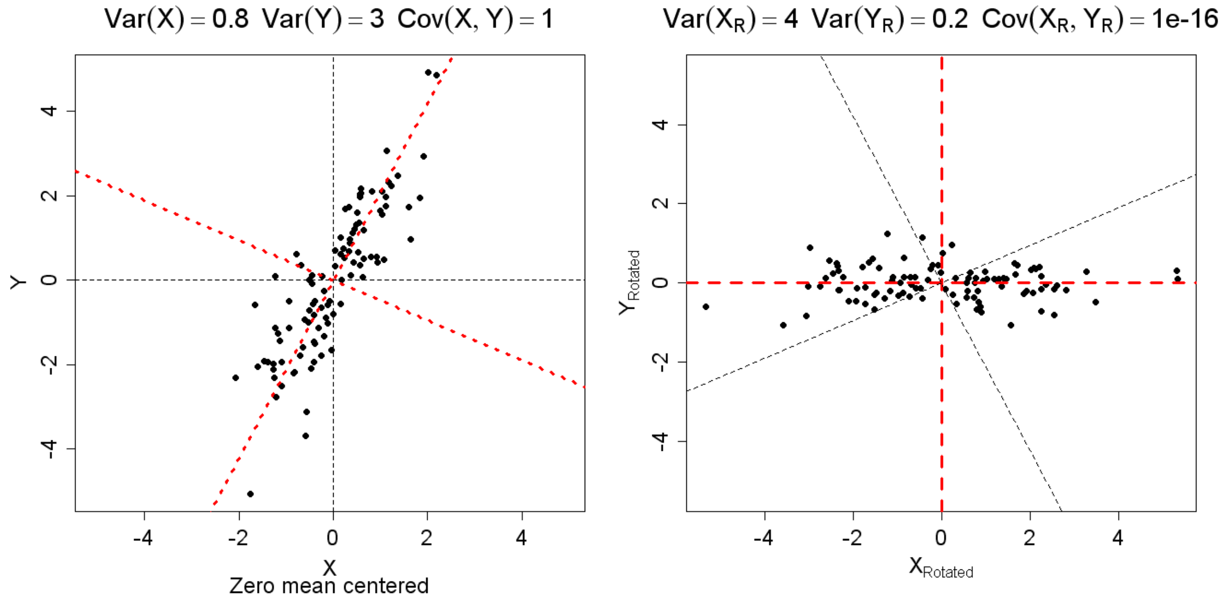


OUTLINE

- Importance of physical model for feature selection
- Best subset selection
- Backward/forward/alternating feature selection
- Model independent feature selection
- Shrinkage methods (ridge and lasso)
- Principal components regression
- Wrong and right way to perform cross-validation

PRINCIPAL COMPONENT ANALYSIS (PCA)

- We will talk about it in more detail when we look unsupervised methods
- The main idea is qualitatively simple: find orthogonal basis, such that the variance of the data is the largest along the first direction (first principal component, or PC1), followed by the variance along PC2, etc
- This is simply a rotation in the multi-dimensional space defined by the predictor variables). This rotation diagonalizes the covariance matrix
- The rationale behind PCA: we *hope* that the direction(s) of the largest variance is where the signal is strongest.
- As PCA is a rotation, the new coordinate vectors (new "variables") are linear transformations of the original variables



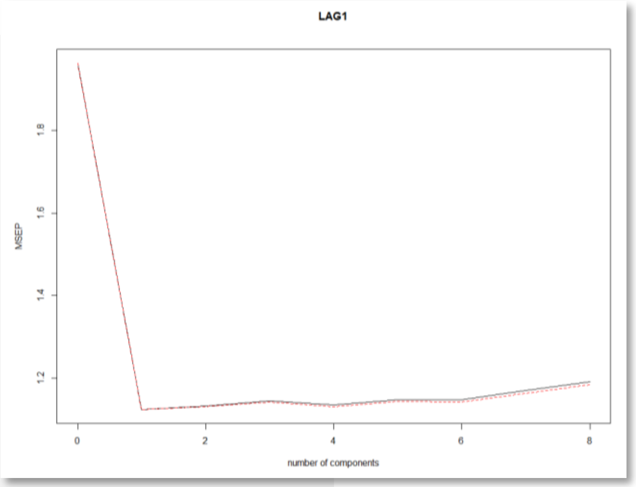
PRINCIPAL COMPONENT REGRESSION

- We can regress on PCA-transformed variables just like on any other combinations of predictor variables

```
> library(pls)
> pcr.fit=pcr(LAG1~.,data=ac,validation="CV",scale=T)
> summary(pcr.fit)
Data:    X dimension: 182 8
     Y dimension: 182 1
Fit method: svdpc
Number of components considered: 8

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
CV           1.401     1.06    1.064    1.070    1.065    1.071    1.071    1.082    1.092
adjCV        1.401     1.06    1.063    1.069    1.063    1.069    1.069    1.079    1.088

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X       50.69    66.97    80.36    86.04    91.29    95.61    99.10   100.00
LAG1    42.69    43.00    43.51    44.19    44.19    45.77    45.77    45.77
> validationplot(pcr.fit,val.type = "MSEP")
```

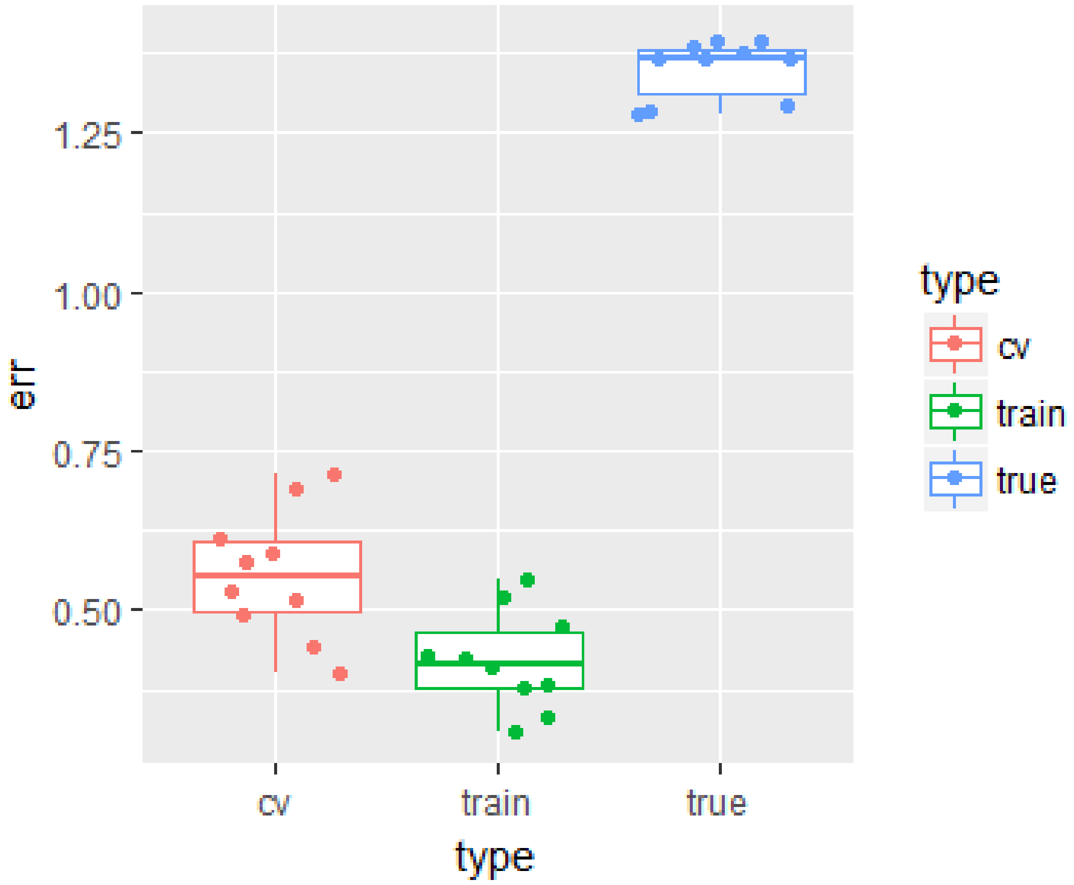


OUTLINE

- Importance of physical model for feature selection
- Best subset selection
- Backward/forward/alternating feature selection
- Model independent feature selection
- Shrinkage methods (ridge and lasso)
- Principal components regression
- Wrong and right way to perform cross-validation

WRONG WAY TO DO CROSS-VALIDATION (AND VARIABLE SELECTION)

```
> library(boot)
> makeNull <- function(nVars=10000,nObs=100) {
   dRet <- matrix(rnorm((nVars+1)*nObs),ncol=nVars+1)
   colnames(dRet) <- c("Y",paste0("X",1:nVars))
   data.frame(dRet)
 }
> nTop <- 10
> dTmp <- makeNull()
> r <- cor(dTmp[,1],dTmp[,-1])
> attrIdx <- c(1,1+order(abs(r),decreasing=TRUE)[1:nTop])
> g <- glm(Y~.,dTmp[,attrIdx],family=gaussian)
> mean((predict(g)-dTmp$Y)^2)
[1] 0.3464968
> cv.glm(dTmp[,attrIdx],g,K=5)$delta[1]
[1] 0.4382631
> dNew <- makeNull(nObs=1000)
> mean((predict(g,newdata=dNew[,attrIdx])-dNew$Y)^2)
[1] 1.22786
>
```



- What's wrong? It's not glm's fault!

CHOOSING THE BEST PREDICTORS, ON THE ENTIRE DATASET, *PRIOR* TO RESAMPLING

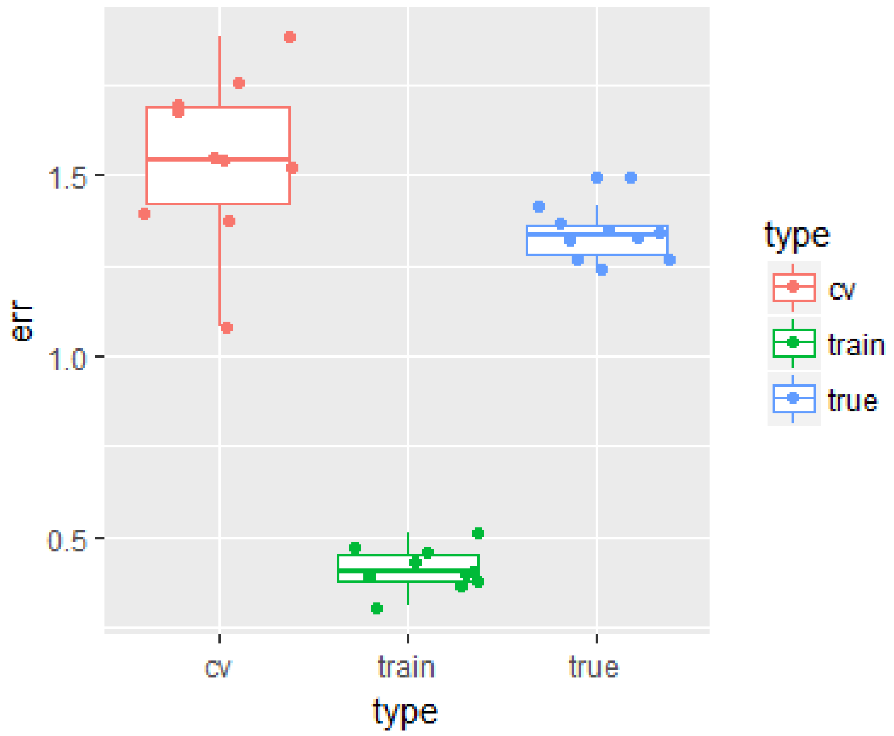
…is very likely to underestimate the error rate on a new dataset from the same distribution

The impact is the worst when you are at your most vulnerable – when the signal is weak or absent and there are many predictors to choose from

See also ESL (not ISLR!) Ch.7.10.2

NOW, SAME PROCESS CORRECTED

```
>   dTmpAll <- makeNull()
>   folds <- sample(rep(1:5,length.out=nrow(dTmpAll)))
>   cvPredObs <- NULL
>   for ( kTmp in unique(folds) ) {
+     dTmp <- dTmpAll[folds!=kTmp,]
+     r <- cor(dTmp[,1],dTmp[,-1])
+     attrIdx <- c(1,1+order(abs(r),decreasing=TRUE)[1:nTop])
+     g <- glm(Y~.,dTmp[,attrIdx],family=gaussian)
+     cvPreds <- predict(g,newdata=dTmpAll[folds==kTmp,])
+     cvPredObs <- c(cvPredObs,(cvPreds-dTmpAll[folds==kTmp,"Y"])^2)
+   }
>   mean(cvPredObs)
[1] 1.400218
>   r <- cor(dTmpAll[,1],dTmpAll[,-1])
>   attrIdx <- c(1,1+order(abs(r),decreasing=TRUE)[1:nTop])
>   g <- glm(Y~.,dTmpAll[,attrIdx],family=gaussian)
>   mean((predict(g)-dTmpAll$Y)^2)
[1] 0.3869772
>   dNew <- makeNull(nObs=1000)
>   mean((predict(g,newdata=dNew[,attrIdx])-dNew$Y)^2)
[1] 1.272707
>
```



- Notice where features are chosen
- CV more indicative of true error

SUMMARY

- Many different methods exist that at least provide some guidance for model selection and/or offer principled procedures. They include:
  - Best subset selection
  - Forward/backward stepwise selection, model based and model-free
  - Shrinkage methods: ridge regression, lasso
- Finding the "right" model is still very difficult
  - True dependence maybe indistinguishable under noise – there's never enough data!
  - Understanding subject domain and physical principles behind the model helps
  - Data preprocessing and curated feature selection
- It matters where feature selection happens with respect to resampling!
  - Always (always!) on training dataset