Variable Selection

Mladen Kolar (mkolar@chicagobooth.edu)

Ockham's Razor

We have focused on training different types of machine learning models and evaluating their fit.

- typically, we used all available variables...

Now, we are going to look for principled ways to choose variable that enter our models.

The underlying principle is Ockham's Razor: "simpler is better"

- a philosophy that transcends all scientific disciplines.

We have seen that overly complicated models lead to bad forecasts.

- "Over-fit" $\Longrightarrow$ lower generalizability

We are going to try to find simple models with simple well-defined procedures.

- Carrying out those procedures may involve a lot of (computer) work.

Why Should We Perform Variable Selection

Prediction Accuracy

- Trade off some bias to control variance.
- Less predictors in the model $\rightarrow$ higher bias, lower variance.

Model Interpretability

- By removing irrelevant features we can obtain a model that is more easily interpreted.

Linear models

Example: Default prediction

To start off as simply as possible, we will first consider the case where we have a binary `y` and one numeric `x`.
The Default data (from the ISLR book):

- `y`: whether or not a customer defaults on their credit card
- `x`: The average balance that the customer has remaining on their credit card after making their monthly payment.
- 10,000 observations, 333 defaults (.0333 default rate).

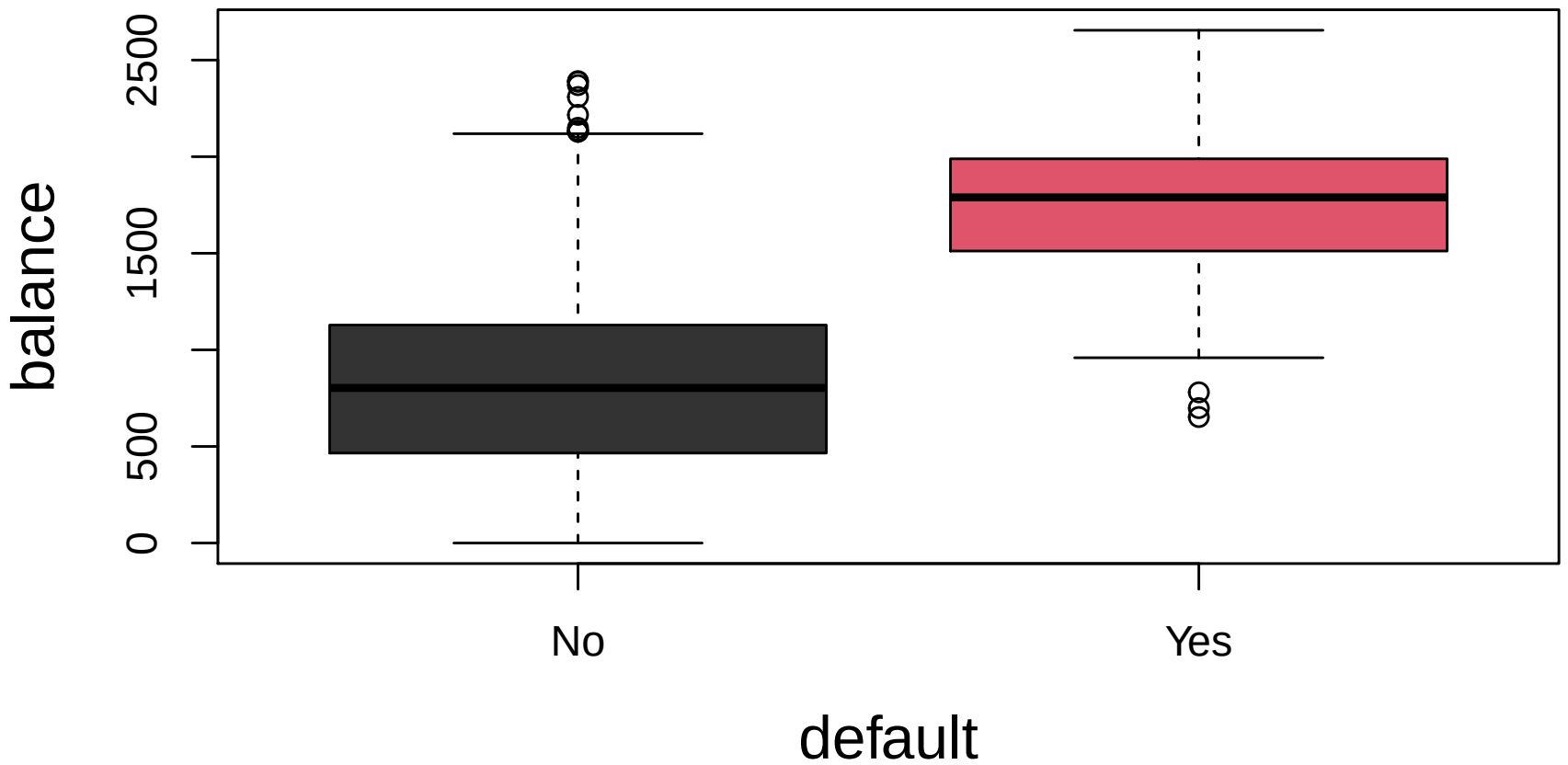
Example: Default prediction

```
library(ISLR)
head(Default)
```

```
##   default student   balance    income
## 1      No      No   729.526 44361.63
## 2      No     Yes   817.180 12106.13
## 3      No      No 1073.549 31767.14
## 4      No      No   529.251 35704.49
## 5      No      No   785.656 38463.50
## 6      No     Yes   919.589  7491.56
```

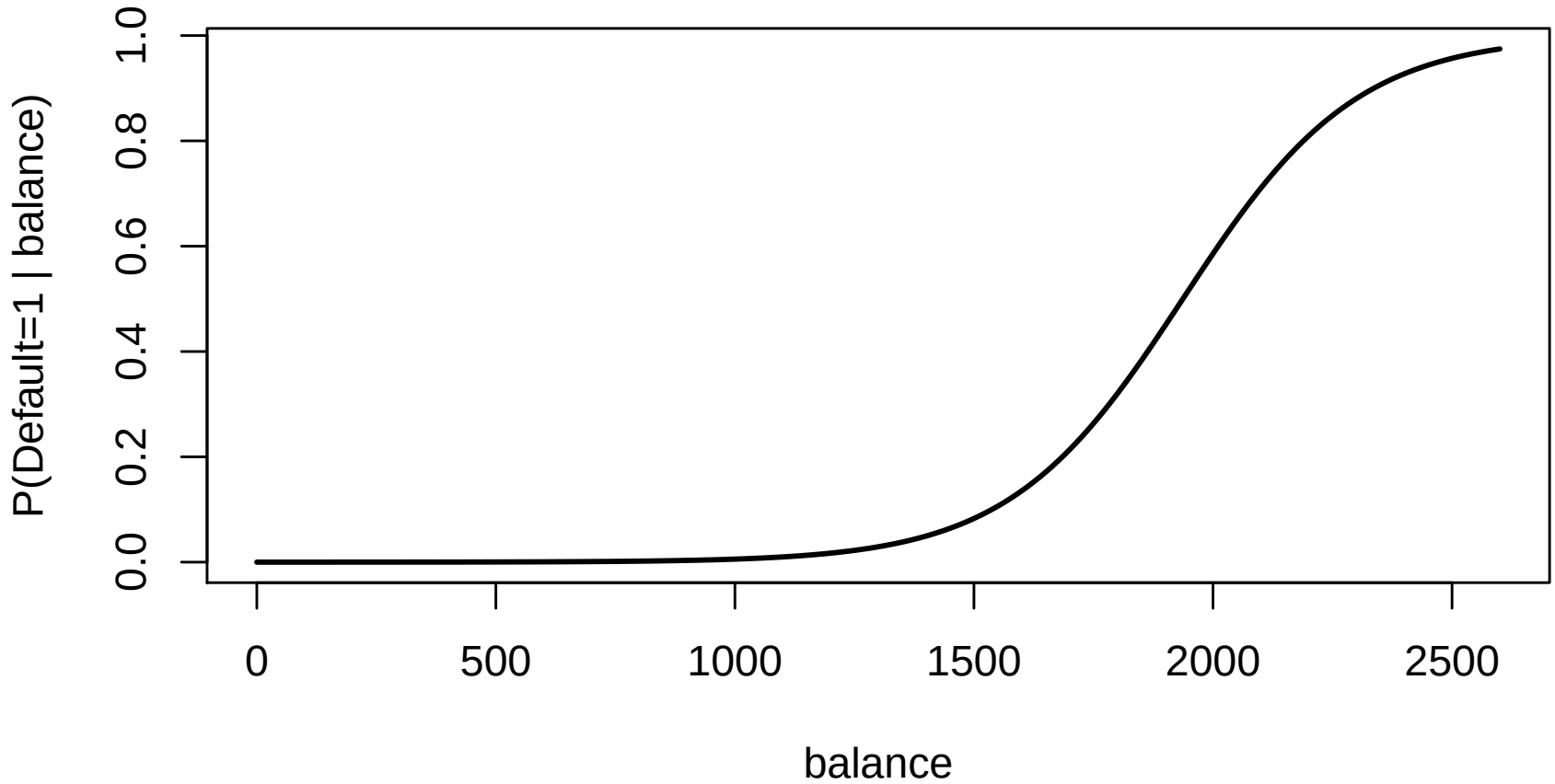
Example: Default prediction

```
plot(balance ~ default, data=Default, col=c(grey(.2),2:6),cex.lab=1.4)
```



Example: Default prediction

```
out = glm(default ~ balance, data=Default, family = "binomial")
xx.grid = seq(0,2600,length.out=1000)
phat = predict(out, data.frame(balance=xx.grid), type="response")
plot(xx.grid, phat, xlab = "balance", ylab="P(Default=1 | balance)", type="l", lwd=2)
```

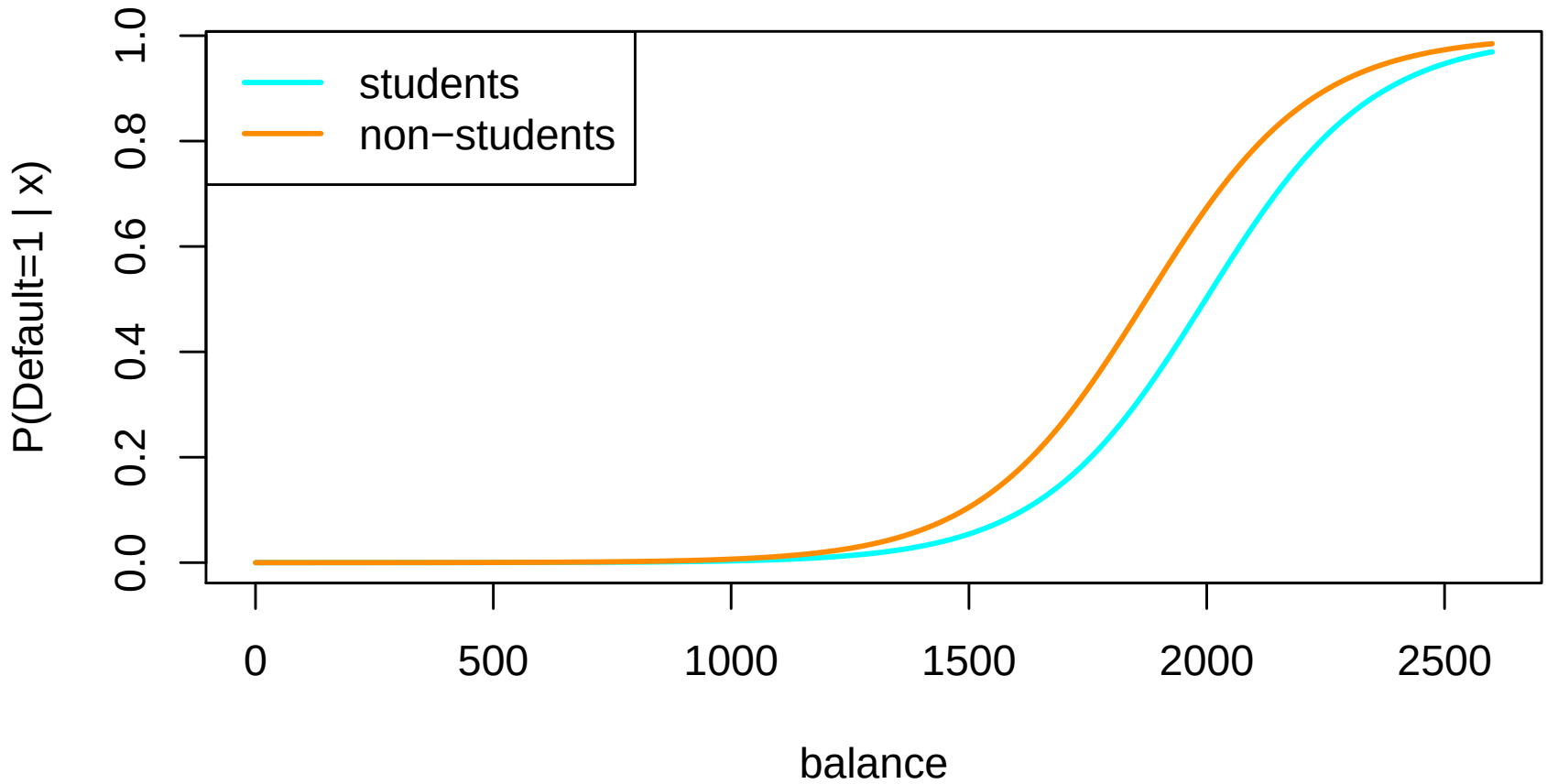



```
coef(out)
```

```
##  (Intercept)      balance
## -10.65133061   0.00549892
```

Example: Default prediction

```
out = glm(default ~ balance + student, data=Default, family = "binomial")
phat_stud = predict(out, data.frame(balance=xx.grid, student=rep("Yes", length(xx.grid))), type="response")
phat_nstud = predict(out, data.frame(balance=xx.grid, student=rep("No", length(xx.grid))), type="response")
plot(xx.grid, phat_stud, xlab = "balance", ylab="P(Default=1 | x)", type="l", lwd=2, col="cyan")
lines(xx.grid, phat_nstud, xlab = "balance", ylab="P(Default=1 | x)", type="l", lwd=2, col="darkorange")
legend("topleft", legend = c("students", "non-students"), col=c("cyan", "darkorange"), lwd=2)
```




```
coef(out)
```

```
## (Intercept)     balance  studentYes
## -10.7494959   0.0057381  -0.7148776
```

Example: Default prediction

The logistic regression output using all three x's in the data set.

```
summary( glm( default ~ balance + student + income, data=Default, family = "binomial") )
```

```
##
## Call:
## glm(formula = default ~ balance + student + income, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.469  -0.142  -0.056  -0.020   3.738
##
## Coefficients:
##              Estimate Std. Error z value
## (Intercept) -1.09e+01   4.92e-01  -22.08
## balance      5.74e-03   2.32e-04   24.74
## studentYes  -6.47e-01   2.36e-01   -2.74
## income       3.03e-06   8.20e-06    0.37
##             Pr(>|z|)
## (Intercept)   <2e-16 ***
## balance       <2e-16 ***
## studentYes    0.0062 **
## income        0.7115
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1571.5  on 9996  degrees of freedom
## AIC: 1580
##
## Number of Fisher Scoring iterations: 8
```

AIC and BIC in Linear/Logistic Regression

Recall the deviance

$$\text{deviance} = -2\sum_{i=1}^{n} \log(P(Y = y_i \mid X = x_i; \hat{\beta}))$$

The better the fit of the model, the smaller the deviance. If we add more variables to the model, we can make the deviance smaller.

- overfitting

Rather than minimizing deviance, we minimize

$$\text{deviance} + \text{penalty}(\#\text{variables})$$

- $\text{AIC} = \text{deviance} + 2 \cdot (\#\text{variables} + 1)$
- $\text{BIC} = \text{deviance} + \log(n) \cdot (\#\text{variables} + 1)$

BIC has a bigger penalty, so it suggest smaller models than AIC.

Example: AIC and BIC for Default prediction

```
out1 = glm(default ~ balance, data=Default, family = "binomial")
c(deviance(out1), AIC(out1), BIC(out1))
```

```
## [1] 1596.45 1600.45 1614.87
```

```
out2 = glm(default ~ balance + student + income, data=Default, family = "binomial")
c(deviance(out2), AIC(out2), BIC(out2))
```

```
## [1] 1571.54 1579.54 1608.39
```

```
out3 = glm(default ~ balance + student, data=Default, family = "binomial")
c(deviance(out3), AIC(out3), BIC(out3))
```

```
## [1] 1571.68 1577.68 1599.31
```

```
out4 = glm(default ~ student, data=Default, family = "binomial")
c(deviance(out4), AIC(out4), BIC(out4))
```

```
## [1] 2908.68 2912.68 2927.10
```

pick balance+student

Model Probabilities

One (very!) nice thing about the BIC is that you can interpret it in terms of **model probabilities**.

Given a list of possible models $\{M_1, M_2, \ldots, M_3\}$, the probability that model i is correct is

$$P(M_i) = P(M_i) \approx \frac{e^{-\frac{1}{2}\mathrm{BIC}(M_i)}}{\sum_{r=1}^{R} e^{-\frac{1}{2}\mathrm{BIC}(M_r)}} = \frac{e^{-\frac{1}{2}[\mathrm{BIC}(M_i)-\mathrm{BIC}_{\min}]}}{\sum_{r=1}^{R} e^{-\frac{1}{2}[\mathrm{BIC}(M_r)-\mathrm{BIC}_{\min}]}}$$

- $\mathrm{BIC}_{\min}$ is subtracted for numerical stability.

Which models to consider?

Subset Selection

You need to narrow down your options before comparing models.

- What if there are too many possibilities!

1. Identify a subset of the p predictors that we believe to be related to the response.
2. Fit a model using least squares on the reduced set of variables.

Selection method	**Which models to compare?**
Best subset	???
Forward stepwise	???
Backward stepwise	???

Best Subset Selection

Idea: Compare all models for each possible combination of the p predictors.

Algorithm:

1. Let $\mathcal{M}_0$ denote the *null model*, which contains no predictors.
2. For $k = 1, \ldots, p$:
   - 2.1 Fit all $\binom{p}{k}$ models that contain exactly k predictors.
   - 2.2 Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validation score, AIC, BIC, etc.

Best Subset Selection

Selection method	**Which models to compare?**
Best subset	Every possible combination of p predictors
Forward stepwise	???
Backward stepwise	???

Simple and conceptually appealing approach.

Suffers from computational limitations. Cannot be applied when p is large.

- For p predictors, there are 2^p models to compare.

Even when we can fit 2^p models, it is not clear that we should.

- With that many models, one by chance might look good.

We need computationally efficient alternatives to best subset selection.

- **Stepwise** methods explore a far more restricted set of models.

Forward Stepwise Selection

Forward stepwise selection begins with a model containing no predictors, and then adds predictors to the models, one at a time, until all of the predictors are in the model.

At each step, the variable that gives the greatest additional improvement to the fit is added to the model.

Forward Stepwise Selection

1. Let $\mathcal{M}_0$ denote the null model, which contains no predictors.

2. For $k = 0, \ldots, p-1$:
   - 2.1 Consider all $p - k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
   - 2.2 Choose the **best** among these $p - k$ models, and call it $\mathcal{M}_{k+1}$.

3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross validation, AIC, BIC, etc.

Forward Stepwise Selection

Selection method	**Which models to compare?**
Best subset	Every possible combination of p predictors
Forward stepwise	Series of models with one predictor added each time
Backward stepwise	???

Computational advantage over best subset selection.

- Need to fit $p + (p-1) + \cdots + 1 = \frac{p(p-1)}{2}$ models, compared to 2^p for best subset selection.

It is not guaranteed to find the best possible model out of all 2^p models.

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

1. Let $\mathcal{M}_p$ denote the full model, which contains all p predictors.

2. For $k = p, p-1, \ldots, 1$:
   - 2.1 Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   - 2.2 Choose the best among these k models, and call it $\mathcal{M}_{k-1}$.

3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross validation, AIC, BIC, etc.

Backward Stepwise Selection

Selection method	**Which models to compare?**
Best subset	Every possible combination of p predictors
Forward stepwise	Series of models with one predictor added each time
Backward stepwise	Series of models with one predictor removed each time

Computational advantage over best subset selection.

- Need to fit $p + (p-1) + \cdots + 1 = \frac{p(p-1)}{2}$ models.

It is not guaranteed to find the best possible model.

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.

Choosing the Optimal Model

REMEMBER: We cannot use the training error to select the model

- The model containing all predictors will always have the smallest RSS and the largest R^2.

We can indirectly estimate test error by making an adjustment to the training error to account for the bias due to overfitting.

- This is possible in linear/logistic regression models...

We can directly estimate the test error, using either a validation-set approach or a cross-validation approach.

Cross-Validation: Right and Wrong

Consider a simple classifier applied to some two-class data:

1. Starting with 5000 predictors and 50 samples, find the 100 predictors having the largest correlation with the class labels.
2. We then apply a classifier such as logistic regression, using only these 100 predictors.

How do we estimate the test set performance of this classifier?

Cross-Validation: Right and Wrong

Can we apply cross-validation in step 2, forgetting about step 1?

This would ignore the fact that in Step 1, the procedure has already seen the labels of the training data, and made use of them. This is a form of training and must be included in the validation process.

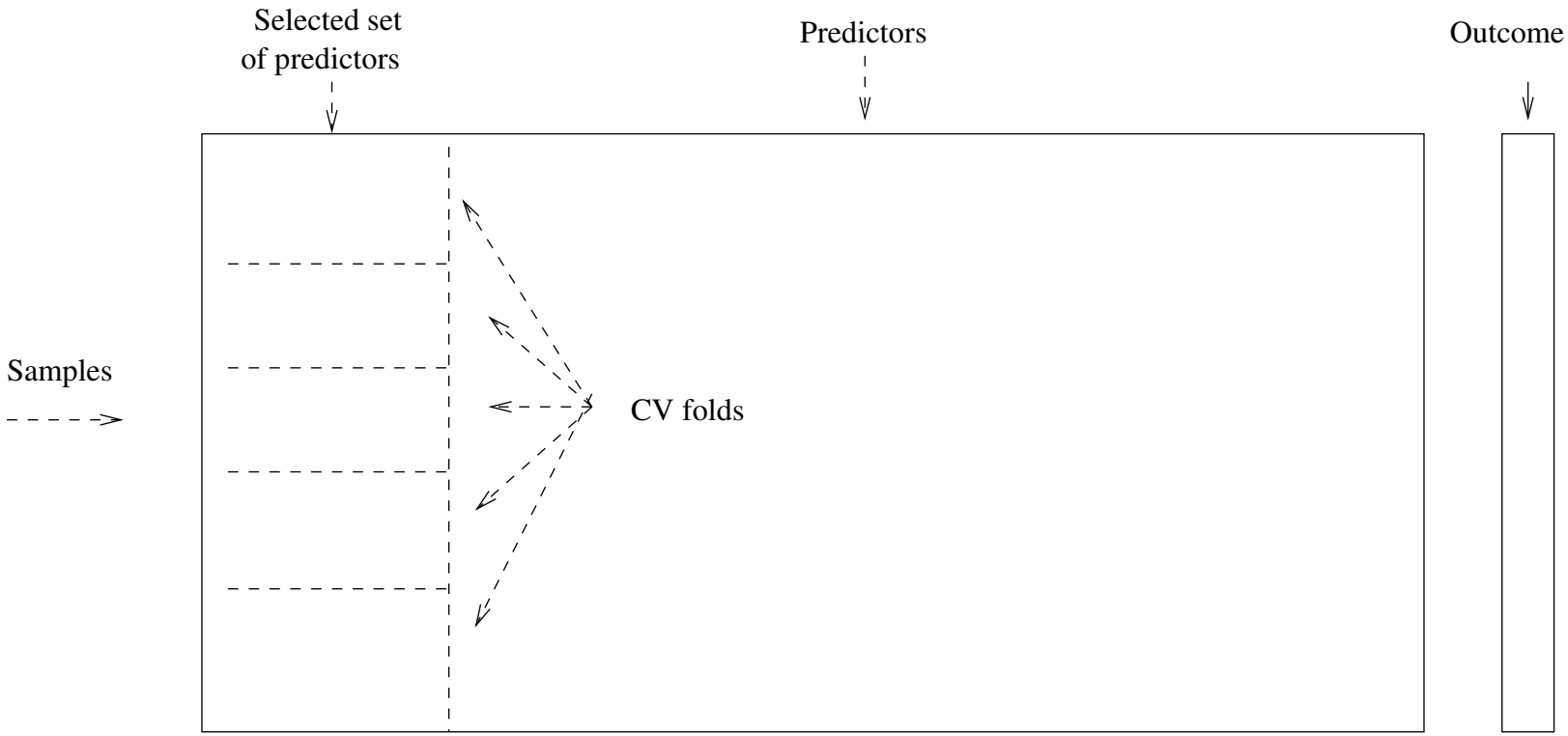
It is easy to simulate realistic data with the class labels independent of the outcome, so that true test error $= 50\%$, but the CV error estimate that ignores Step 1 is zero!

Cross-Validation: Right and Wrong

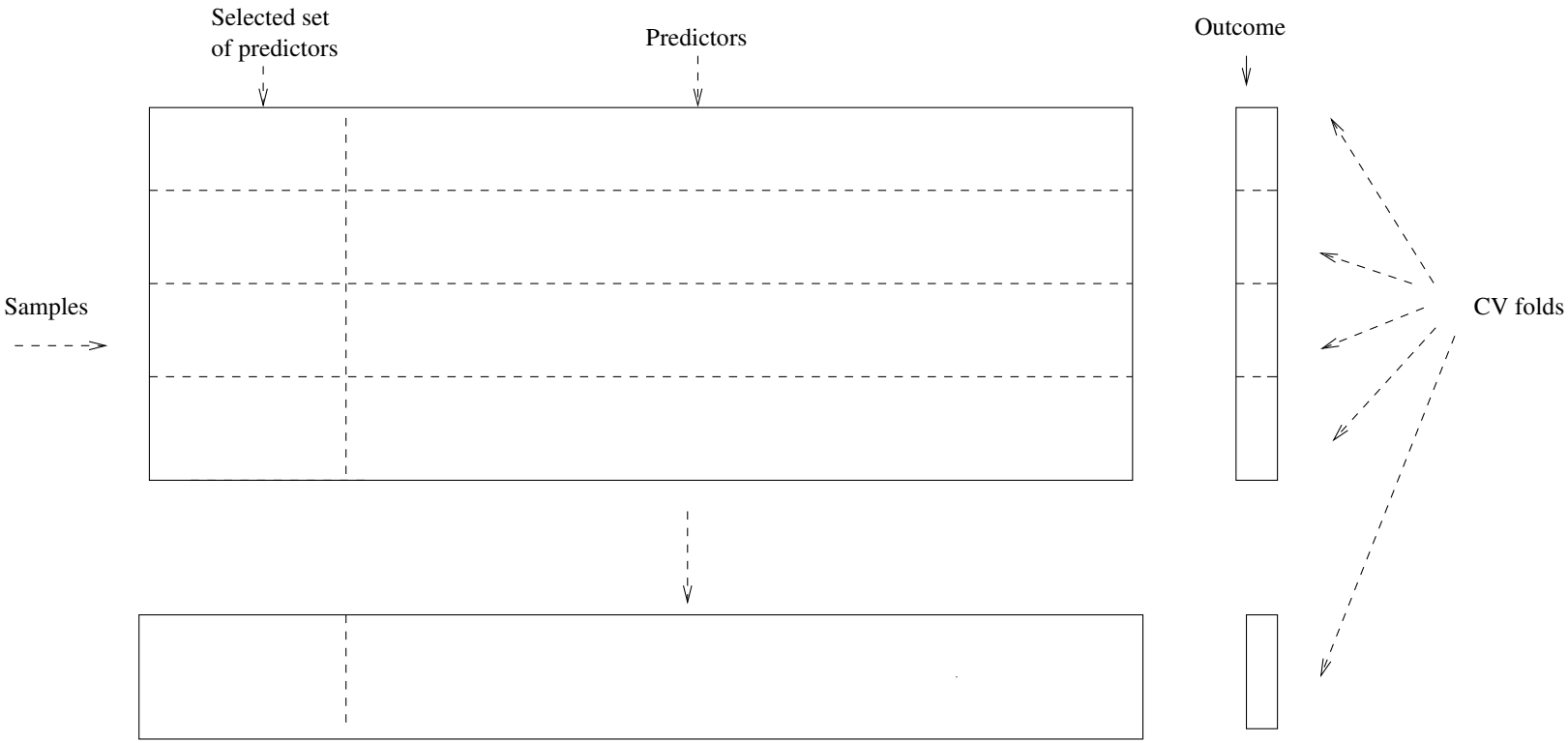
Wrong: Apply cross-validation in step 2.

Right: Apply cross-validation to steps 1 and 2.

Wrong Way


Selected set
of predictors
Predictors
Outcome
Samples
CV folds

Right Way


Selected set
of predictors
Predictors
Outcome
Samples
CV folds

Shrinkage Methods

Shrinkage Methods

A modern alternative to subset selection

We can fit a model containing all p predictors using a technique that constrains or regularizes the coefficient estimates, or equivalently, that shrinks the coefficient estimates towards zero.

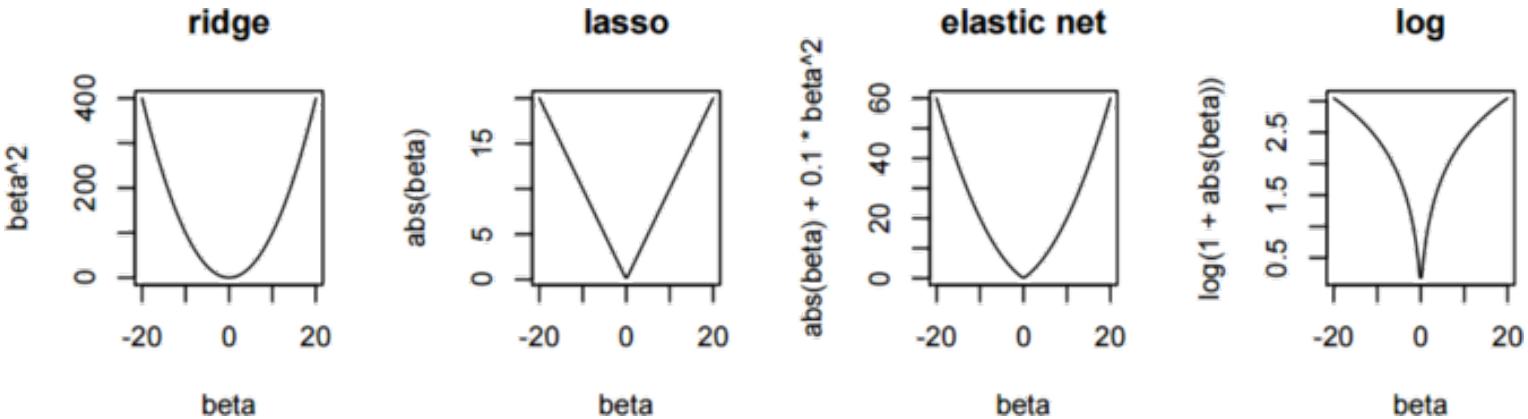
Main idea behind: By shrinking the coefficient, compromise some bias, to trade for less variance. Get better prediction performance (small test error).

Regularization for Linear/Logistic Regression

Regularized objective

$$\hat{\beta} \leftarrow \min_{\beta} \left\{ \underbrace{\text{deviance}(\beta)}_{-\frac{2}{n}\sum_{i=1}^{n} \log P(Y=y_i|X=x_i;\beta)} + \lambda \sum_{j=1}^{p} \text{pen}(\beta_j) \right\}$$

- pen is often the absolute value (L1) or square (L2),
- the lasso and ridge solutions, colloquially.



Regularization for Linear/Logistic Regression

Lasso penalty

$$\hat{\beta} \leftarrow \min_{\beta} \left\{ \text{deviance}(\beta) + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

Ridge penalty

$$\hat{\beta} \leftarrow \min_{\beta} \left\{ \text{deviance}(\beta) + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

```
library(glmnet)
```

- by default implements the lasso penalty
- Just like `glm`, it can handle Binomial, Poisson, Gaussian families and more.

Regularization for Linear/Logistic Regression

Standard least squares 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 $1/c$.

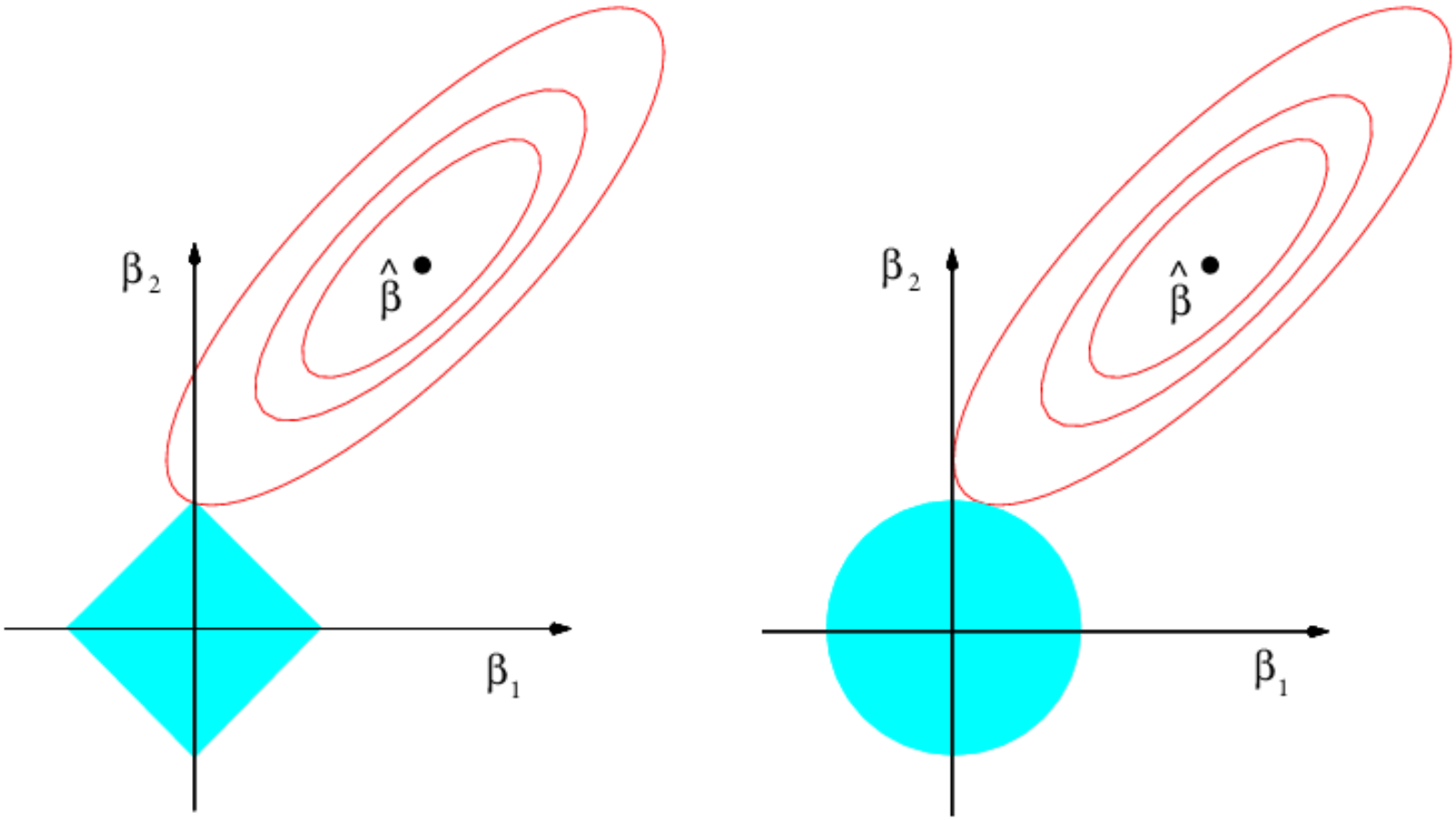
The lasso and ridge regression coefficient estimates can change substantially when multiplying a predictor by a constant.

- first standardize the input variables

Both the lasso and ridge regression shrink the coefficient estimates towards zero.

- The Lasso performs variable selection

Lasso vs Ridge



- Red contour: RSS given by (β_1, β_2)
- Blue area: constrained sets $|\beta_1| + |\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$.