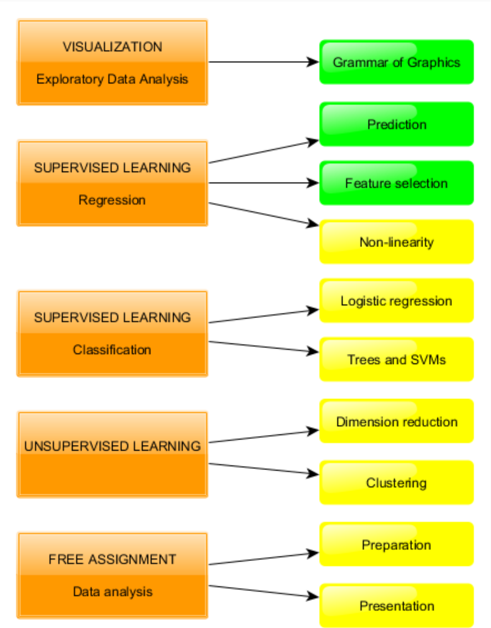
Supervised Learning

Feature selection

Maarten Cruyff

Program


VISUALIZATION
Exploratory Data Analysis
Grammar of Graphics
SUPERVISED LEARNING
Regression
Prediction
Feature selection
Non-linearity
SUPERVISED LEARNING
Classification
Logistic regression
Trees and SVMs
UNSUPERVISED LEARNING
Dimension reduction
Clustering
FREE ASSIGNMENT
Data analysis
Preparation
Presentation

Content

1. Filter methods
2. Wrapper methods
3. Embedded methods
4. Dimension reduction

What's feature selection?

Bias-variance trade-off

- more model complexity reduces bias (good)
- less model complexity reduces variance (good)

Previous example with single feature and polynomial models.

But what if we have lots of features?

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p + \epsilon$$

Flexible model

- Like a kid in candy store with credit card
- It does not stop buying coefficients until the store is sold out!

How do we educate this child?

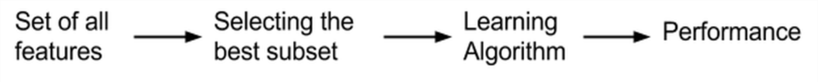
Main classes

1. **Filter methods** (subset selection)
    - select features and train the model
2. **Wrapper methods** (subset selection)
    - train the model on various subsets of features
3. **Embedded methods** (shrinkage, regularization)
    - train the model with penalties on magnitude of coefficients
4. **Dimension reduction** (unsupervised)
    - reduce coefficients by merging similar candies into a single one

Filter methods

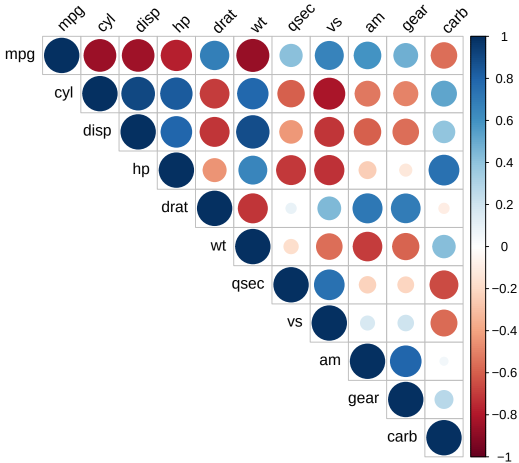
Preselection of features

- you can only buy coefficients you like best
- e.g. predictors with minimum correlation $|r_{y,x}|$



Correlation feature selection

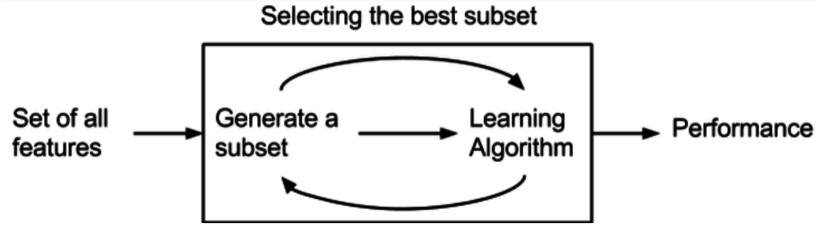
Select features x_j for which $|r_{mpg,x_j}| > r_{min}$



Wrapper methods

Selection best predictors

- you can only buy a limited number of coefficients
- e.g. the best k predictors



Best subset selection

Algorithm

1. For $k = 1, 2, \ldots, p$, fit all $\binom{p}{k}$ models
2. Use cross-validation to determine the MSE
3. Select the model with the smallest cross-validated MSE

Pros and cons

- Finds the best model
- Need to fit 2^p, which for $p = 20$ is little over one million

Alternatives: forward stepwise selection

Algorithm forward

1. Start with the null model and add predictor one-at-a-time
2. At each step, select the predictor yields the best fit
3. Select the model with the smallest cross-validated MSE

Pros and cons

- No guarantee to find the best model
- Fit only $1 + p(p+1)/2$ models, which for $p = 20$ is 211

Alternatives

Backward stepwise selection

- as forward, but the other way around
- starts with full model, and remove variable one-at-a-time

Hybrid stepwise selection

- combination of forward and backward,
- at each step predictors can be entered or removed

Stepwise in R

Backward

```
main_lm <- lm(outcome ~ ., data)
step(main_lm, scope = outcome ~ 1, direction="both")
```

Forward

```
null_lm <- lm(outcome ~ ., data)
step(null_lm, scope = outcome ~ ., direction="both")
```

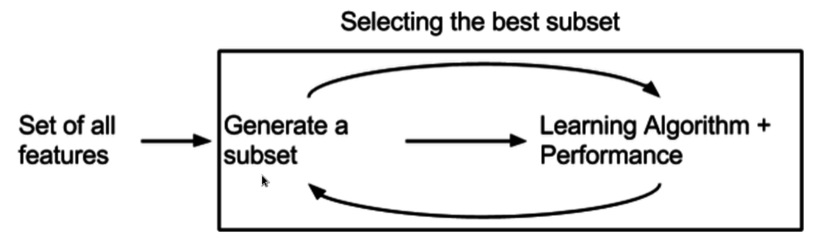
Hybrid

```
step(null_lm, scope = outcome ~ ., direction="both")
```

Embedded methods

Budgets

- You can but anything, but on a limited budget s
- Sum of absolute/squared coefficients cannot exceed s



Regularization, shrinkage, penalization

Algorithm

1. Fit the model with all p **standardized** predictors
   - ensures that size coefficient is independent of scale predictor
2. Constrain the sum of absolute/squared coefficients to budget s
3. Determine with cross-validation the optimal value for s

Lasso penalty: $\lambda \sum_{j=1}^{p} |\beta_j| < s$

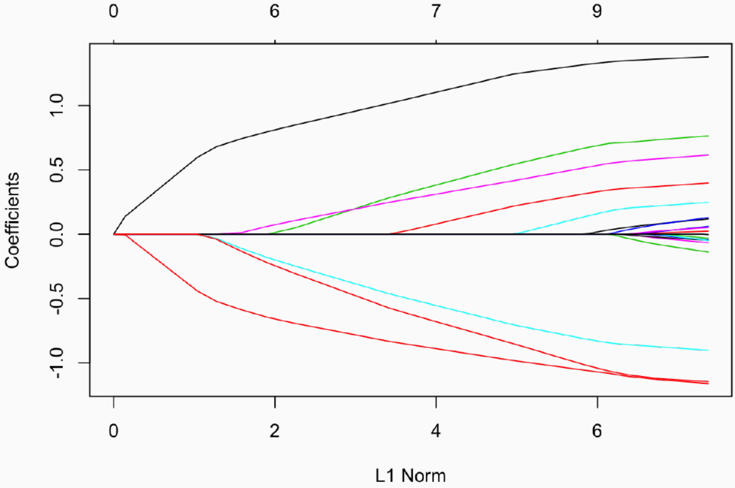
- shrinks coefficients exactly to zero

Ridge penalty: $\lambda \sum_{j=1}^{p} \beta_j^2 < s$

- shrinks coefficients, but not to zero

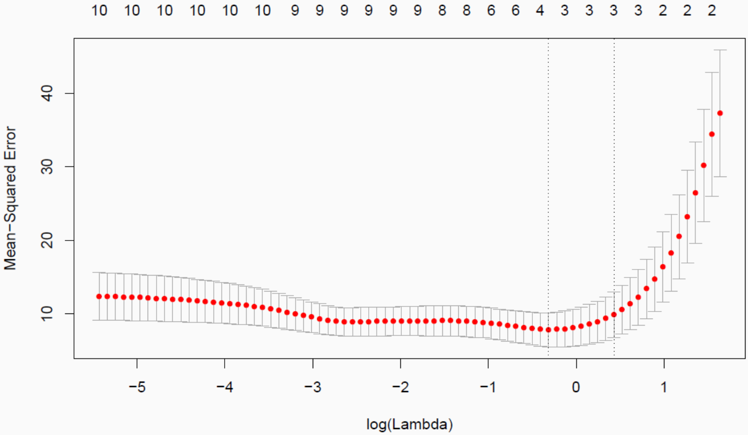
Lasso shrinkage

- As the budget (L1 norm) decreases, coefficients go to zero



Cross-validated λ

- $\log(\lambda_{min}) \approx -0.3$ (4 coefficients)
- $\log(\lambda_{optim}) \approx \log(\lambda_{min}) + 1SD \approx 0.4$ (3 coefficients)



Regularization in R

Package glmnet

```
train_ridge <- glmnet(x = as.matrix(Train[, -nr],
                      y = Train[, <nr>],          # nr = colnr outcome
                      alpha = 0)                  # alpha = 1 for lasso

plot(train_ridge, label=TRUE)                     # shrinkage plot

cv_ridge <- cv.glmnet(x, y,                       # cross validation
                      alpha=0,
                      type.measure = "mse")

plot(cv_ridge)                                     # CV plot

test_ridge <- predict(cv_ridge,
                      newx = as.matrix(Test[, -nr]))
```

Regularization with caret

```
tr <- train(x = scale(Train[, -nr]), y = Train[, nr],
      method    = "glmnet",
      metric    = "RMSE",
      tuneGrid  = expand.grid(alpha  = c(0, 1),
                              lambda = seq(1e-5, 10, length
      trControl = trainControl(method = "cv",
                               number = 5))

ggplot(tr)  # CV plot
```

Dimension reduction

PCA

Principal Components Analysis

- summarize information in p predictors in $q << p$ principal components
- principal components are orthogonal (uncorrelated)
- use PC's for prediction

More detailed discussion of PCA in unsupervied learning