Introduction to Statistical Learning and Applications
CC3: Model Selection, Cross-Validation and Principal Component Regression

Razan MHANNA
STATIFY team, Inria centre at the University Grenoble Alpes
Team 5, Grenoble Institute of Neurosciences GIN

5 March 2024

Contents

Current Section

Least Squares

- We distinguish between the residuals $r_i = y_i - \hat{a} + \hat{b}x_i$ and the errors $\varepsilon_i = y_i - a + bx_i$. The model is written in terms of the errors, but it is the residuals that we can work with: we cannot calculate the errors as to do so would require knowing a and b.
- The residual sum of squares is SSR= $\mathscr{P}_{i=1}^{n} y_i - \hat{a} + \hat{b}x_i{}^2$
- The $\hat{a}, \hat{b}$ that minimizes RSS is called the least squares, and can be written in matrix notation as, $\hat{\beta} = X^T X^{-1} X^T y$,
- In the case of regression with just one predictor, we can write the solution as, $\hat{b} = \frac{\mathscr{P}_{i=1}^{n} x_i - \bar{x} y_i - \bar{y}}{\mathscr{P}_{i=1}^{n} x_i - \bar{x}^2}$ and $\hat{a} = \bar{y} - \hat{b}\bar{x}$.
- The standard estimate of the residual standard deviation is $\hat{\sigma} = \frac{RSS}{n-p-1}$.

Current Section

Outline

Outline

Current Section

Linear regression issues

- Sensitivity to outliers,
- Multicollinearity leads of high variance of the estimator,
- Prone to overfit if there is a lot of variables,
- Hard to interpret if the number of predictors is large(needs a smaller subset that exhibits strongest effects).

- *Predictive Accuracy*
- *Model Interpretability by removing irrelevent features(by setting the corresponding coefficient estimates to zero), we can obtain a model that is more easily interpreted.*

Model selection

Three classes of methods:

- Subset Selection by taking a subset of the p predictors, then fit a model using least squarres on the reduced set of variables.
- Shrinkage: All p predictors are involved, but the estimated coefficients are shrinken towards zero relative to the least squares estimates. The shrinkage (also known as regularization) has the effect of reducing variance and can also perform variable selection.
- Dimension Reduction: Project the p predictots into a M-dimensional subspace, where M < p. This is achieved by computing M different linear combinations, or projections, of the variables. Then these M projections are used as predictors to fit a linear regression model by least squarres.

Concerns about model selection

- Given several candidate models, we can use **cross validation** to compare their predictive performance. But does it always make sense to choose the model that optimizes estimated predicted performance?
  1. If the only goal is prediction on data similar to what was observed, it can be better to average across models rather than to choose just one.
  2. If the goal is prediction on a new set of data, we should reweight the predictive errors accordingly to account for differences between sample and population,
  3. Rather than choosing among or averaging over an existing set of models, a better choice can be to perform continuous model expansion, constructing a larger model that encompasses the earlier models as special cases.

Overfitting and Unseen Data

If we keep adding more predictors to our model, the residuals will continue to decrease, but this will not actually mean that our model is better$\longrightarrow$ Instead, what we will be doing is over-fitting to the data.
That is, our model will really just be "memorizing" the data itself rather than learning a model.
The true test of model quality is how well it does at predicting for data that we didn't see.
That is, if we fit our model on data X_i, Y_i for $i = 1, 2, \ldots, n$, how well does our model do on a previously unseen data point X_{n+1}, Y_{n+1}?
Specifically, in the case of regression, we want our model to minimize

$$E\,\hat{Y}_{n+1} - Y_{n+1}{}^2,$$

where $\hat{Y}_{n+1}$ is our model's prediction based on coefficients estimated from our n training observations.

Cross-validation

What Does Cross-validation Mean?

- *Cross-validation is a statistical approach for determining how well the results of a statistical investigation generalize to a different data set.*
- *Cross-validation is commonly employed in situations where the goal is prediction and the accuracy of a predictive model's performance must be estimated.*

[1] Jun Shao.
Linear model selection by cross-validation.
Journal of the American Statistical Association, *88:486–494, 1993.*

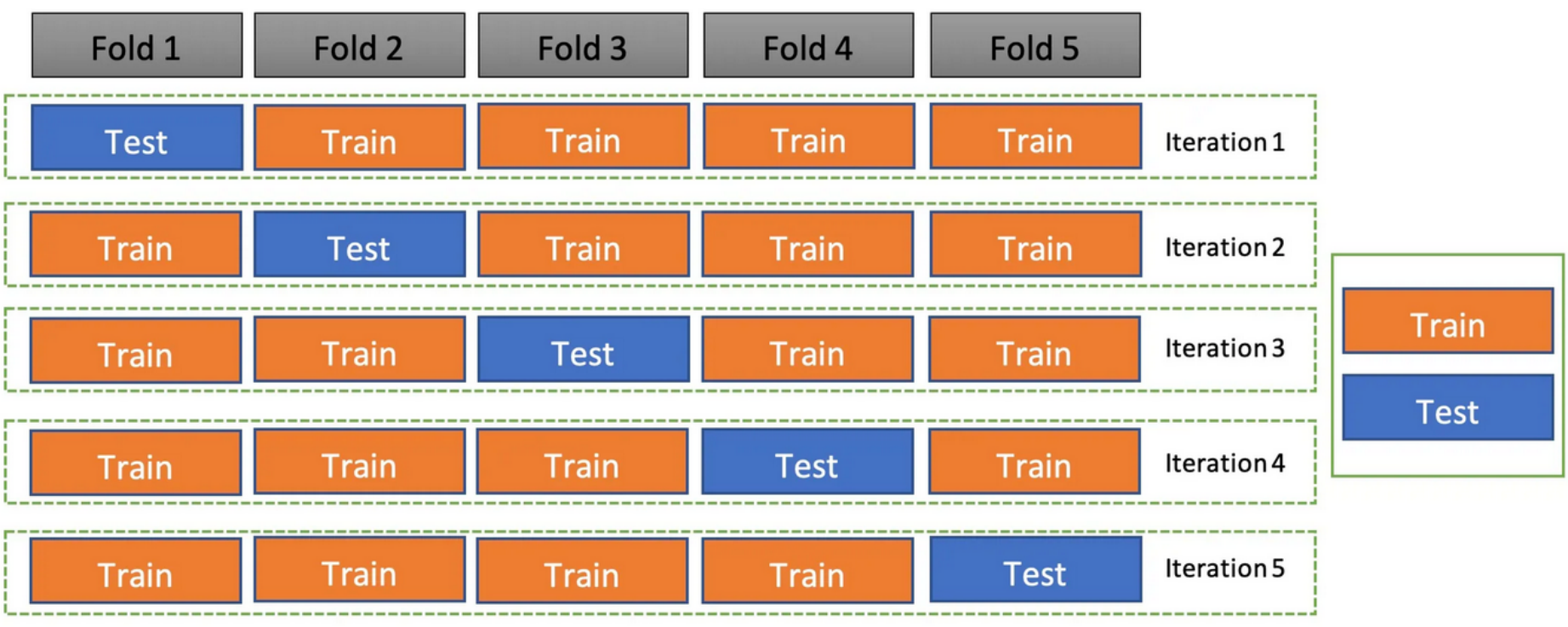
Cross-Validation

- Cross Validation is a model validation technique for accessing how the result of a statistical analysis will generate to an independent dataset,
- So we can say cross validation is used for:
  1. Finding or estimating expected error,
  2. Helps in selecting the best fit model,
  3. Avoiding Over-fitting.

Current Section

Outline

Definition


Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Test
Train
Train
Train
Train
Iteration 1
Train
Test
Train
Train
Train
Iteration 2
Train
Train
Test
Train
Train
Iteration 3
Train
Train
Train
Test
Train
Iteration 4
Train
Train
Train
Train
Test
Iteration 5
Train
Test

Advantages and Limitations

Advanatges of k-fold Cross-Validation:

- Low bias and variance
- Better estimation of model performance
- Maximizes data utilization
- Robustness to randomness in data splitting
- Reduces overfitting
- Allows for hyperparameter tuning
- Provides insights into model behavior
- Useful for small datasets
- Suitable for imbalanced datasets

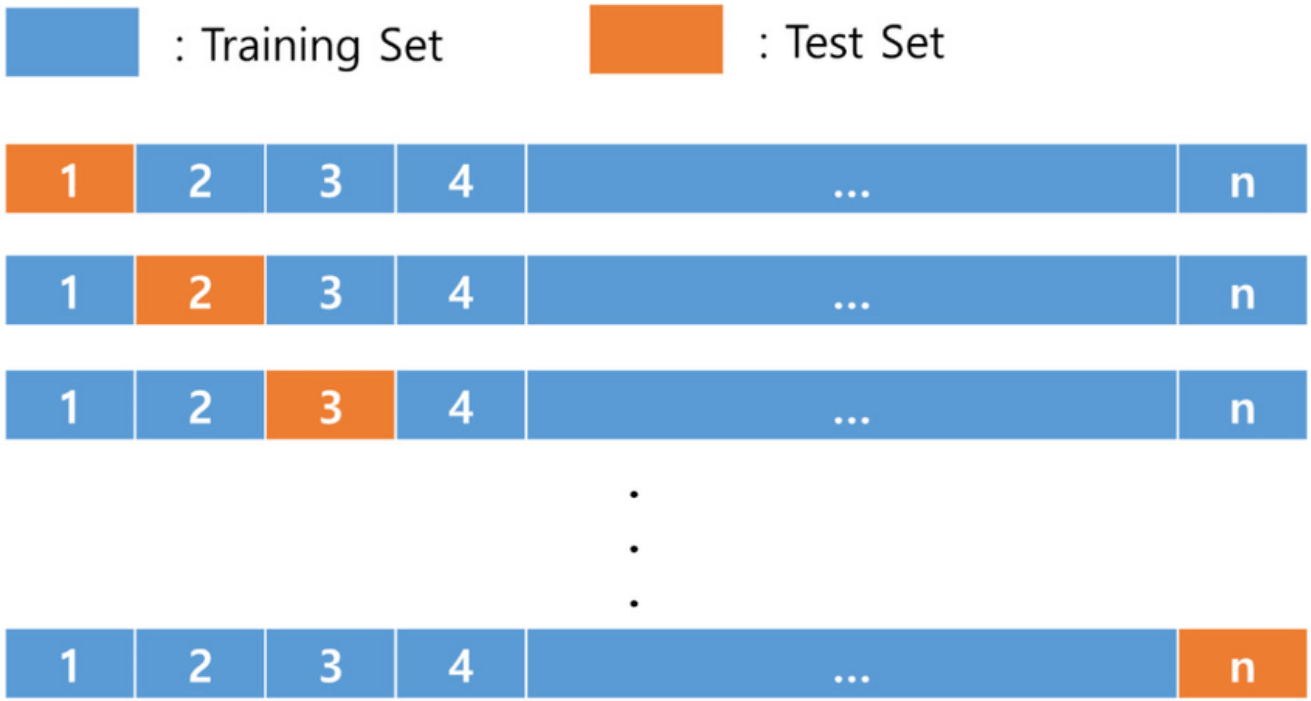
Limitations of k-fold Cross-Validation

- Computational cost
- May not be suitable for very large datasets -consuming for complex models information leakage
- Not suitable for time-series data in results
- Dependent on the choice of k

Outline

Leave-One-Out Cross-Validation (LOOCV)

- LOOCV is a cross-validation technique where a single data point is used as the validation set, and the rest of the data is used for training.
- This process is repeated for each data point, ensuring that each observation is used once as the validation data.



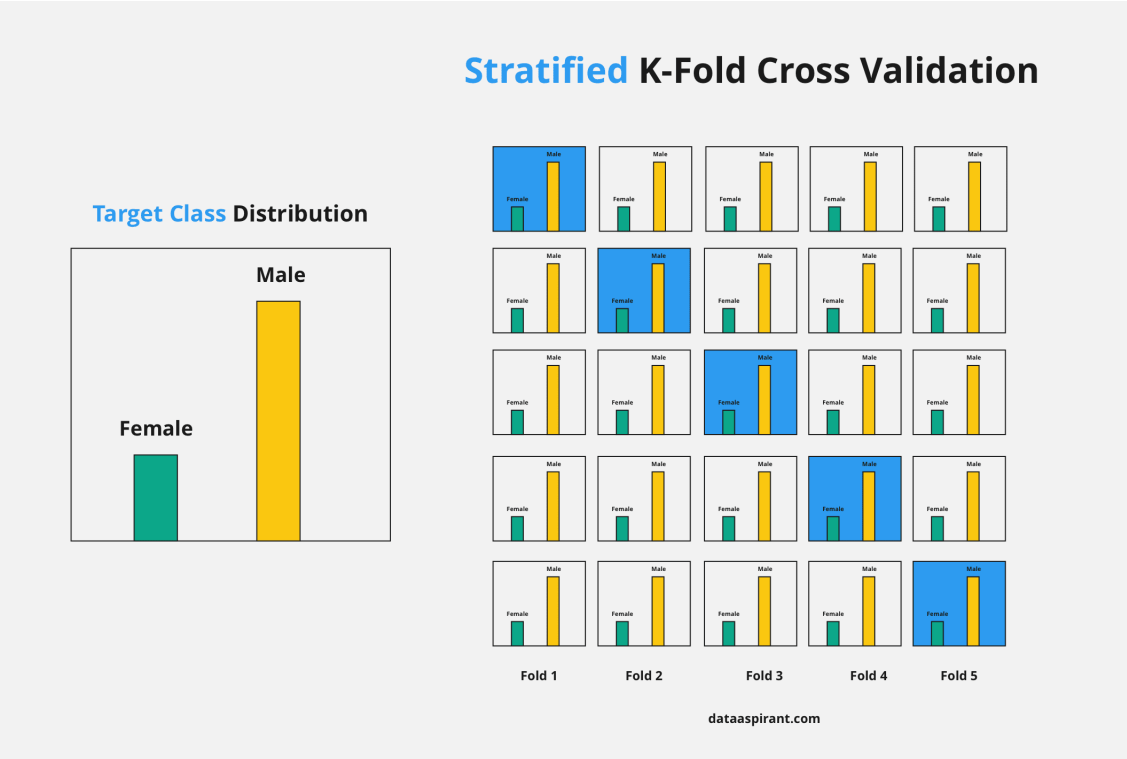
Advantages and Limitations

Advantages	Limitations
—Utilizes all available data for training and validation. —Provides an unbiased estimate of model performance. —Low bias as each model is trained on almost all data points.	—Computationally expensive for large datasets. —High variance in model performance estimation. —Not suitable for large sample sizes due to limited improvement over other methods.

Outline

Definitions

Stratified cross-validation is a variation of k-fold cross-validation where each fold or partition of the data preserves the proportion of samples from each class or category. This ensures that the distribution of classes is consistent across all folds, making it particularly useful for classification tasks with imbalanced class distributions.



Advantages

- Preserves Class Distribution where each fold contains a representative sample of all classes,
- By preserving the class distribution in each fold, stratified cross-validation provides a more accurate estimate of model performance, leading to better generalization to unseen data.
- Helps to reduce bias in model evaluation,
- Improved Model Evaluation: Provides a more reliable assessment of model performance,

Disadvantages

- Increased Computational Cost compared to simple random sampling, as it involves the extra step of stratifying the data based on class labels.
- Complex Implementation, particularly when dealing with large and multi-class datasets.
- Dependency on Class Labels: Relies on accurate class labels for stratification, which may be challenging to obtain in certain real-world datasets or may introduce biases if the labels are noisy or inaccurate.
- In some cases, stratified cross-validation may lead to overfitting, particularly if the class distribution in the dataset is highly skewed or if the number of samples per class is very small.
- While highly beneficial for classification tasks, stratified cross-validation is not directly applicable to regression problems, where the target variable is continuous rather than categorical.

Current Section

Outline

Definitions

- Principal Component Regression (PCR) is a regression technique that serves the same goal as standard linear regression - model the relationship between a target variable and the predictor variables,
- The difference is that PCR uses the principal components (PCs) as the predictor variables for regression analysis instead of the original features,
- PCR works in three steps:
  1. Apply PCA to generate principal components from the predictor variables, with the number of principal components matching the number of original features p
  2. Keep the first q principal components that explain most of the variance (where $q < p$), where q is determined by cross-validation
  3. Fit a linear regression model (using ordinary least squares) on these q principal components
- The idea is that the smaller number of principal components represents most of the variability in the data and (presumptively) the relationship with the target variable.

Outline

Key notes

Outline

Problems with PCA

- PCA assumes approximate normality of the input space distribution
- PCA may still be able to produce a âgoodâ low dimensional projection of the data even if the data isnât normally distributed
- PCA may "fail" if the data lies on a âcomplicatedâ manifold
- PCA assumes that the input data is real and continuous.
- Extensions to consider:
  - Collins et al, "A generalization of principal components analysis to the exponential family",
  - Hyv arinen, A. and Oja, E., "Independent component analysis: algorithms and applications",
  - ISOMAP, LLE, Maximum variance unfolding, etc.

Outline

PCR in R programming language[1]

Step 1: Load Necessary Packages

The easiest way to perform principal components regression in R is by using functions from the pls package.

install pls package (if not already installed)
install.packages("pls")
load pls package
library(pls)

Step 2: Fit PCR Mode

Using mtcars dataset: For this example weâll fit a principal components regression (PCR) model using *hp* as the response variable and the following variables as the predictor variables: *mpg, disp, drat, wt and qsec.*

[1] Full Tutorial here.

PCR in R language

```
#view first six rows of mtcars dataset
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
#make this example reproducible
set.seed(1)

#fit PCR model
model <- pcr(hp~mpg+disp+drat+wt+qsec, data=mtcars, scale=TRUE, validation="CV")
```

PCR in R

1. scale=TRUE: This tells R that each of the predictor variables should be scaled to have a mean of 0 and a standard deviation of 1. This ensures that no predictor variable is overly influential in the model if it happens to be measured in different units.
2. validation=âCVâ: This tells R to use k-fold cross-validation to evaluate the performance of the model. Note that this uses k=10 folds by default. Also note that you can specify âLOOCVâ instead to perform leave-one-out cross-validation.

Step 3: Choose the Number of Principal Components

Once weâve fit the model, we need to determine the number of principal components worth keeping.

The way to do so is by looking at the test root mean squared error (test RMSE) calculated by the k-fold cross-validation.

PCR in R

```
#view summary of model fitting
summary(model)

Data:   X dimension: 32 5
        Y dimension: 32 1
Fit method: svdpc
Number of components considered: 5

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps
CV           69.66    44.56    35.64    35.83    36.23    36.67
adjCV        69.66    44.44    35.27    35.43    35.80    36.20

TRAINING: % variance explained
    1 comps  2 comps  3 comps  4 comps  5 comps
X     69.83    89.35    95.88    98.96   100.00
hp    62.38    81.31    81.96    81.98    82.03
```

There are two tables of interest in the output:

PCR in R

1. VALIDATION: RMSEP
   This table tells us the test RMSE calculated by the k-fold cross validation. We can see the following:
   - If we only use the intercept term in the model, the test RMSE is 69.66.
   - If we add in the first principal component, the test RMSE drops to 44.56.
   - If we add in the second principal component, the test RMSE drops to 35.64.

We can see that adding additional principal components actually leads to an increase in test RMSE. Thus, it appears that it would be optimal to only use two principal components in the final model.

2.TRAINING:% variance explained

This table tells us the percentage of the variance in the response variable explained by the principal components. We can see the following:

- By using just the first principal component, we can explain 69.83% of the variation in the response variable. By adding in the second principal component, we can explain 89.35% of the variation in the response variable.

Note that weâll always be able to explain more variance by using more principal components, but we can see that adding in more than two principal components doesnât actually increase the percentage of explained variance by much.

PCR in R

We can also visualize the test RMSE (along with the test MSE and R-squared) based on the number of principal components by using the validationplot() function.

```
#visualize cross-validation plots
validationplot(model)
validationplot(model, val.type="MSEP")
validationplot(model, val.type="R2")
```

Step 4: Use the Final Model to Make Predictions
We can use the final PCR model with two principal components to make predictions on new observations.

PCR in R

The following code shows how to split the original dataset into a training and testing set and use the PCR model with two principal components to make predictions on the testing set.

```
#define training and testing sets
train <- mtcars[1:25, c("hp", "mpg", "disp", "drat", "wt", "qsec")]
y_test <- mtcars[26:nrow(mtcars), c("hp")]
test <- mtcars[26:nrow(mtcars), c("mpg", "disp", "drat", "wt", "qsec")]

#use model to make predictions on a test set
model <- pcr(hp~mpg+disp+drat+wt+qsec, data=train, scale=TRUE, validation="CV")
pcr_pred <- predict(model, test, ncomp=2)

#calculate RMSE
sqrt(mean((pcr_pred - y_test)^2))

[1] 56.86549
```