

Statistical Computing

Michael Mayer

March 2023



Statistical Computing: What will we do?

Chapters

1. R in Action
2. Statistical Inference
3. Linear Models
4. Model Selection and Validation
5. Trees
6. Neural Nets

Remarks

- ▶ Chapters 3 to 6:
Statistical ML in Action
- ▶ Two weeks per chapter
- ▶ Exercises at end of chapter notes

Model Selection and Validation

Two Questions

→ prevent overfitting (more params → more overfit)

- ▶ “How good is our model?”
- ▶ “Which model to choose among alternatives?”

Problem and solution

- ▶ “In-sample” performance is biased
evaluating model with train data
- ▶ Overfitting should not be rewarded
- ▶ Use data splitting to get fair results

Notation

- ▶ Total loss $Q(f, D) = \sum_{(y_i, \mathbf{x}_i) \in D} L(y_i, f(\mathbf{x}_i))$
data (points to D) *loss fct.* (points to L)
- ▶ Average loss $\bar{Q}(f, D) = Q(f, D)/|D|$
Aug Bernoulli deviance (points to \bar{Q})
- ▶ Performance measure or evaluation metric $S(f, D)$ of interest, often $S = \bar{Q}$ or a logistic function
logistic function (points to S)

$$\text{if } L(y_i, f(\mathbf{x}_i)) = \|y_i - f(\mathbf{x}_i)\|^2$$

→ S might be $\sqrt{\bar{Q}}$ to conserve the dimension

Outline

- ▶ Nearest-Neighbor
- ▶ Simple Validation
- ▶ Cross-Validation
- ▶ Test Data and Final Workflow
- ▶ Excursion: SQL and Spark

Excursion: k -Nearest-Neighbor (k -NN)

"Black box model"

- ▶ Alternative to linear model
- ▶ How does it work?
- ▶ Classification and regression
- ▶ Standardization?

Covariates - mean
stdev.

Select covariates

look at objects with similar covariates
take average response variable

maybe weighting the

covariates, measure the distance with euclidean dist.

Example

categories ordinal measuring
clarity is ordered 1
2
3
:
categories are not
→ dummy variables

Simple Validation

- ▶ In-sample, 1-NN would win any comparison!? *→ overfitting*
- ▶ Split data into training and validation sets D_{train} and D_{valid} , e.g., 80%/20%
- ▶ Use performance $S(\hat{f}, D_{\text{valid}})$ on validation set to make decisions (choose models, choose parameters like k)
- ▶ Measure amount of overfitting/optimism by

$$S(\hat{f}, D_{\text{valid}}) - S(\hat{f}, D_{\text{train}})$$

Example

packages
caret
MLR3 (R6)
tidymodels very strange

K-fold Cross-Validation (CV)

Simple validation is neither economic nor robust, except for large data

Algorithm

1. Split the data into K pieces $D = \{D_1, \dots, D_K\}$ called "folds". Typical values for K ?
2. Set aside one of the pieces (D_k) for validation
3. Fit model \hat{f}_k on $D \setminus D_k$
4. Calculate performance $\hat{S}_k = S(\hat{f}_k, D_k)$
5. Repeat Steps 2 – 4 for each k
6. Calculate CV performance $\hat{S}_{CV} = \frac{1}{K} \sum_{k=1}^K \hat{S}_k$

not the K -NN
range 5-10

Remarks

► How to choose and fit best/final model?

► What means «best»?

► Stability of results? calculate standard deviation of \hat{S}_k

► Repeated CV?

Example

repeat CV with different seeds, if small dataset

fit K -NN on the whole dataset for a final model

Hyperparameter Tuning

- ▶ Choosing k in k -NN is example of “hyperparameter tuning”
- ▶ Algorithms with more than 1 hyperparameter? *xg boost*
- ▶ Grid Search CV *grid of hyp. params*
- ▶ Randomized Search CV *if too big grid
→ randomised search*

Test Data and Final Workflow

Problematic consequence of model tuning?

- ▶ **Overfitting** on validation data or on CV!
- ▶ Performance of final model? → **Test data**

Workflow A *simple validation*

1. Split data into train/valid/test, e.g., by ratios 60%/20%/20%
70 20 10
2. Train different models on training data and assess performance on validation data. Choose best model, re-train on (training + validation) data, and call it "final model" (Simplification?) *by not retraining on both*
3. Assess performance of final model on test data

Workflow B *cross validation*

1. Split data into train/test, e.g., by ratios 80%/20%.
2. Evaluate and tune different models by K-fold CV on training data. Choose best model, re-train on full training data
3. Assess performance of final model on test data

Example of Workflow B

we can include the test data to fit the model

When test data not necessary?

Ridge Regression

High dim data, L^1 penalty
(more cols)
some or most of the
 β are zero
→ feature selection

- ▶ Example of **penalized** regression
- ▶ Model equation similar to usual linear regression

$$\mathbb{E}(Y | \mathbf{x}) = f(\mathbf{x}) = \beta_0 + \beta_1 x^{(1)} + \dots + \beta_p x^{(p)}$$

- ▶ But with penalized least-squares objective

$$Q(f, D_{\text{train}}) = \sum_{(y_i, \mathbf{x}_i) \in D_{\text{train}}} (y_i - f(\mathbf{x}_i))^2 + \lambda \sum_{j=1}^p \beta_j^2$$

can be tuned
↓
effect: make
 β closer to zero
 L^2 penalty

- ▶ L^2 penalty pulls coefficients slightly towards 0, fighting overfitting
- ▶ $\lambda_{\text{opt}} \geq 0$ with best (cross-)validation result → use to fit final model
- ▶ Intercept? Standardization? L^1 ? Elastic-net?

Example

normally not
penalised

scaling
happens internally
for most packages

also (β_j)

both, L^1 and L^2 penalty

Using Independent Partitions is Essential

Random splits *generate indep. partitions*

Grouped splits

*rows from different clients
overly optimistic it have leakage*

*→ data from same client
in train and test data*

→ avoid by random split by groupid

Time-Series splits

different logic

Stratified splits

*in binary response:
less 1's than 0's → keep the distribution
in splits:*

Excursion: SQL and Spark

Data science is 80% preparing data, 20% complaining about preparing data.

Typical preprocessing steps?

Good moment to learn

- ▶ data structure *→ what does the rows mean?*
- ▶ meaning of columns
- ▶ sources of bias
 - removed rows
 - missing values
 - outliers

How to do preprocessing?

Data = files on disk or tables in database

- ▶ If small → R/Python / *Raw data*
- ▶ If large? → Database Management System (DBMS) or Spark
- ▶ Communication via SQL

*Preprocess
on DB
→ SQL
Big data stuff
→ Spark*

SQL

Structured Query Language

- ▶ Pronounced?
- ▶ Important in data science
- ▶ In DBMS or R/Python
- ▶ ISO norm ↔ dialects
- ▶ SQL queries

SASS is not ISO normed

Learn SQL with examples

- ▶ Diamonds (from memory)
- ▶ Taxi (from Parquet)

cloud databases uses mostly parquet

DuckDB (since 2018)

- ▶ In-process, open-source DBMS
- ▶ Easy to install in R/Python *in comparison to oracle/msft*
- ▶ No dependencies (Java etc.) *unlike H2O*
- ▶ Fast
- ▶ Out-of-core capabilities → *externalise out of RAM*
runs on same thread as R/python
→ saved in parquet files

Apache Spark

- ▶ Distributed, open-source cluster computing system for big data *multiple computers*
- ▶ Apache project since 2013
- ▶ Heavily used in industry
- ▶ Written in Scala
- ▶ Contains SQL engine
- ▶ Can be used from R/Python

Examples

- ▶ Diamonds (from memory)
- ▶ Taxi (from Parquet)