MA8701 Advanced methods in statistical inference and learning

L5: Trees, bagging and random forest

Mette Langaas IMF/NTNU

07 February, 2021

Missing covariates

(ELS 9.6, and van Buuren: “Flexible imputation of missing data”)
When performing data analysis we often encounter data sets where some observations have missing values. It is important to understand the underlying mechanism for the observations to be missing, so that we may treat the missing data appropriately.
We first look at the general definitions of missing variants, and then move to how missing data can be handled for trees.

Notation

- $\mathbf{y}$: response vector (no missing values)
- $\mathbf{X}$: the full covariate matrix
- $\mathbf{Z} = (\mathbf{X}, \mathbf{y})$: full responses and covariates
- $\mathbf{X}_{\text{obs}}$: the observed part of the covariate matrix
- $\mathbf{Z}_{\text{obs}} = (\mathbf{X}_{\text{obs}}, \mathbf{y})$
- $\mathbf{R}$: indicator matrix (0/1) for missingness in $\mathbf{X}$, the observability of $\mathbf{X}$.
- θ: some parameter in the distribution of $\mathbf{R}$.

The missing data mechanism is characterized by the conditional distribution of $\mathbf{R}$

$$P(\mathbf{R} \mid \mathbf{Z}, \theta)$$

Missing completely at random (MCAR)

$$P(\mathbf{R} \mid \mathbf{Z}, \theta) = P(\mathbf{R} \mid \theta)$$

- All observations have the same probability of being missing, and
- the missing mechanism is not related to the data.

If observations with MCAR data are removed that should not bias the analyses (but the sample size will of cause be smaller).

Examples:

- measure weight, and the scales run out of battery
- similar mechanism to taking a random sample

Missing at random (MAR)

$$P(\mathbf{R} \mid \mathbf{Z}, \theta) = P(\mathbf{R} \mid \mathbf{Z}_{\mathsf{o}bs}, \theta)$$

- All observations in a group defined by the observed data have the same probability of being missing.
- Remark: not dependent on what could have been observed.

Example:

- measure weight, and the scales have different missing proportions when being on a hard or soft surface

Most methods for handling missing data require the data to be MAR.

Missing not at random (MNAR)

We have MNAR if we dont have MAR or MCAR.
Then the missing mechanicm could depend on what we could have meaured.
Examples:

- the scales give more often missing values for heavier objects than for lighter objects
- a patient is too sick to perform some procedure that would show a high value of a measurement

General solutions to missing covariates

These solutions require that the missingness is MCAR.

Complete case analysis: discard all observations containing missing values. Wasteful.

Let each variable have a probability for missing values of 0.05, then for 20 variables the probability of an observation to be complete is $(1-0.05)^2 0 = 0.36$, for 50 variables 0.08. Not many observations left with complete case analysis.

Pairwise deletion: for example when calculating a correlation matrix only complete pairs may enter in the calculation. Not so relevant for regression and classification.

LOCF: Last observation carried forward. Time series etc. Not recommended, unless there is a reason to believe that nothing has changed.

Mean imputation: Replace the missing value with the mean of the covariate over all samples. Will decrease the variability in the data. "Common solution" within machine learning, but not so common in statistics(?).

Multiple imputation: Devise a method to construct the distribution of each covariate (that can be missing) based on other covariates (often a regression method). Sample multiple observation for each missing value, and get m complete dataset. Analyse all m dataset and weigh the results together. R: package `mice`.

Use a method that handles missing data: such as trees!

Handling missing covariates in trees

Instead of removing observation with missing values, or performing single or multiple imputation, there are two popular solutions to the problem for trees:

Make a “missing category”

- If you believe that missing covariates behave in a particular way (differently from the non-missing values), we may construct a new category for that variable.

Use surrogate splits

The best split at a node is called the *primary split*.

An observation with missing value for variable x_1 is dropped down the tree, and arrive at a split made on x_1.

A "fake" tree is built to predict the split, and the observation follows the predicted direction in the tree. This means that the correlation between covariates are exploited - and the higher the correlation between the primary and predicted primary split - the better.

This is called a *surrogate split*.

If the observation is missing the surrogate variable, there is also a back-up surrogate variable that can be used (found in a similar fashion.)

If the surrogate variable is not giving more information than following the majority of the observations at the primary split, it will not be regarded as a surrogate variable.

The R package `rpart` vignette page 18 gives the following example:

- Assume that the split (age <40, age 40) has been chosen.
- Surrogate variables are found by *re-applying the partitioning algorithm* (without recursion=only one split?) to predict the two categories age <40 vs. age 40 using the other covariates.
- Using “number of misclassified”/“number of observations” as the criterion: the optimal split point is found for each covariate.
- A competitor is the majority rule - that is, go in the direction of the split where the majority of the training data goes. This is given misclassification error min(p, $1 - $p) where p = (# in A with age $<$ 40) / nA.
- A ranking of the surrogate variables is done based on the misclassification error for each surrogate variable, and variables performing better than the majority rule is kept.

- Surrogate variables are found by *re-applying the partitioning algorithm* (without recursion=only one split?) to predict the two categories age <40 vs. age 40 using the other covariates.
- Using "number of misclassified"/"number of observations" as the criterion: the optimal split point is found for each covariate.
- A competitor is the majority rule - that is, go in the direction of the split where the majority of the training data goes. This is given misclassification error min(p, 1 − p) where p = (# in A with age < 40) / nA.
- A ranking of the surrogate variables is done based on the misclassification error for each surrogate variable, and variables performing better than the majority rule is kept.

Example

Look at the Boston default tree with `tree` and `rpart` to see how the two handles missing values.

```
## [1] "tree package"

##      crim zn indus chas   nox    rm  age  dis rad tax ptratio black lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.09   1 296    15.3 396.9    NA   24

##       1
## 19.8223

## [1] "rpart package"

##        1
## 27.82308

## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 354 32270.0 22.95
##    2) rm < 6.945 296 10830.0 19.82
##      4) lstat < 14.405 177  3681.0 23.17
##        8) rm < 6.543 138  1690.0 21.86 *
##        9) rm > 6.543 39   908.2 27.82 *
##      5) lstat > 14.405 119  2215.0 14.84
##       10) crim < 5.76921 63   749.9 17.33 *
##       11) crim > 5.76921 56   636.1 12.04 *
##    3) rm > 6.945 58  3754.0 38.92
##      6) rm < 7.445 33   749.7 33.13 *
##      7) rm > 7.445 25   438.0 46.56 *
```

Other issues

(ELS 9.2.4)

- Categorical predictors: For a predictor with q levels (may be unordered) the number of possible partitions into two groups is large. A trick is used in the processing, where first dummy variable coding is performed then sorted by increasingly popular categories into a ordered categorical variable. Proofs exists that this gives optimal splits for cross-entropy, Gini, squared loss (see ELS page 310 for references).
- Categorical predictors with many levels may have a advantage for the splits, because there are so many possible splits that often one is very good. This may lead to overfitting if q is large.
- For multiclass problems loss matrices may be included easily in the Gini loss.

Other issues

(ELS 9.2.4)

- Categorical predictors: For a predictor with q levels (may be unordered) the number of possible partitions into two groups is large. A trick is used in the processing, where first dummy variable coding is performed then sorted by increasingly popular categories into a ordered categorical variable. Proofs exists that this gives optimal splits for cross-entropy, Gini, squared loss (see ELS page 310 for references).
- Categorical predictors with many levels may have a advantage for the splits, because there are so many possible splits that often one is very good. This may lead to overfitting if q is large.
- For multiclass problems loss matrices may be included easily in the Gini loss.

- Binary splits: multiway splits into more than two groups is possible, but may fragment the data very quickly (too quickly). Multiway splits is achived by a series of binary splits. Thus, we stay with binary splits.
- Due to the binary splits it may be hard to model an additive structure.
- Linear combination splits: is possible by including also finding linear weight parameters for the splits. This may improve predictive power, but hurt interpretability.
- There exists other tree-building procedures than CART. One such is C5.0 by Quinlan, see ELS page 312 for reference.
- For regression trees the regression surface will be non-smooth, which may degrade performance. For classification trees where there response is a classification (and thus not smooth) this is not a large problem.

Questions

Choose your favourites below and discuss in group. (maybe use the Boston and Pima indian analysis to look for examples)

1) Check out the partitioning of the predictor space to recursive binary three - or opposite (given as a problem earlier in this note).
2) Why do we say that trees can automatically handle (and find?) non-linearities? Give example.
3) Same, but now interactions in data. Give example.
4) What rule do you think is used for the prediction with missing value for the `tree` and `rpart` example above?
5) Make list of pros and cons for trees for regression and classification.
6) For Project 1 - do you have missing data in the data you have chosen? If yes, do you think the missing data are missing at random? How did you handle the missing data?
7) Discuss the bias-variance tradeoff of a regression tree when increasing/decreasing the number of terminal nodes, i.e What happens to the bias? What happens to the variance of a prediction if we reduce the tree size?

Some answers

5)

Advantages (+) of using trees

- Trees automatically select variables
- Tree-growing algorithms scale well to large n, growing a tree greedily
- Trees can handle mixed features (continuouos, categorical) seamlessly, and can deal with missing data
- Small trees are easy to interpret and explain to people
- Some believe that decision trees mirror human decision making
- Trees can be displayed graphically
- Trees model non-linear effects
- Trees model interactions between covariates
- Trees handle missing data in a smart way!
- Outliers and irrelevant inputs will not affect the tree.

There is no need to specify the functional form of the regression curve or classification border - this is found by the tree automatically.

Disadvantages (-) of using trees

- Large trees are not easy to interpret
- Trees do not generally have good prediction performance (high variance)
- Trees are not very robust, a small change in the data may cause a large change in the final estimated tree
- Trees do not produce a smooth regression surface.

7)

As the tree size increase the bias will decrease, and the variance will increase. This is the same as any other method when we increase the model complexity.

What is next?

- **Bagging**: grow many trees (from bootstrapped data) and average - to get rid of the non-robustness and high variance by averaging
- **Random forest**: inject more randomness by just allowing a random selection of predictors to be used for the splits at each node.
- The Out-of-Bag available validation set and
- importance plot