Machine Learning II

Prof. Dr. Tim Downie

Lecture 1 – 6th October 2023

Dealing with Missing Data

Berliner Hochschule für Technik

Dealing with Missing Data

Topics in Machine Learning I

Unsupervised Learning

- K-Means
- K-Medoids
- Hierarchical Clustering
- Soft Clustering: EM Algorithm

Supervised Learning

- Regression Problems
  - Linear and multiple regression, ridge regression, lasso regression.
- Classification Problems
  - Bayes classifier (theoretical), logistic regression.
- Tree Methods Regression & Classification
- Ensemble Methods (applied to tree models)

Other Bias-Variance Trade Off, Cross-Validation, Bootstrapping

Machine Learning II

This course builds on the ideas learnt in ML I and deals exclusively with *supervised* learning.

The main topics are

- Dealing with missing values
- Non-linear regression methods incl. spline smoothing and GAMs
- Linear and quadratic discriminant ananlysis
- Support Vector Machines
- Projection Pursuit Regression
- Artificial Neural Networks
- Simple Bayesian methods: naive Bayes classifier, conditional independence, hidden Markov Models.

A provisional week by week semester plan is given in Moodle.
As in ML I, the methods will be applied using the software R.

Assessment

To pass the course you need to gain at least 45 out of 100 course marks.

The course is examined by a 90 minute written exam worth 65%. Everything covered in the course including computational aspects are examinable unless explicitly stated by the lecturer.

Aspects covered in ML1, that are directly relevant to ML 2, such as MSE, are also examinable.

In addition there will be a major project worth 35% of the Marks. The project is not compulsory, but in practice passing the course will be difficult without it.

Provisional dates for the exams are:

- 1st exam: Friday 19th January 2023.
- 2nd exam: Monday 25th March 2023.

The exams will be held in-person on the BHT campus.

Information on all exam timetabling can be found at:
https://polli.bht-berlin.de/exams/M-DS

Project

- You will work in small groups analysing a medium sized data set and comparing two supervised learning regression methods.
- A regression method means that the outcome variable is continuous or at least numeric.
- Your choice of 2 methods can include:
  - Any method learnt in *Machine Learning II*
  - Some of the methods learnt in *Machine Learning I*.
- Full details about which methods are allowed will be handed out late November.

- Finding a suitable data set is a part of the project. You should check with the lecturer that the data are appropriate before analysing the data.
- The provisional time plan for the project is:
  - 24th November – Project details given out.
  - 8th December or before – details of your proposed data set and the students in your group to be submitted to the lecturer.
  - Tuesday 9th January 2024 – Deadline for submitting your work via Moodle.

Text Books

As in Machine Learning I, you will be using James et al. as the main text book, which is available to download for free. For the more advanced subjects, you will use Hastie et al., which also covers more technical details of the subjects covered in James.

James, Witten, Hastie and Tibshirani *An Introduction to Statistical Learning with Applications in R*. Second edition, Springer-Verlag (2021). `https://www.statlearning.com/` and click on *download the second edition*.

Hastie, Tibshirani and Friedman *The Elements of Statistical Learning (2nd edition).* Springer-Verlag (2009). `https://web.stanford.edu/~hastie/ElemStatLearn/`

Baumer, Kaplan and Horton *Modern Data Science with R.* Chapman and Hall (2017).

W.N. Venables and B.D. Ripley *Modern Applied Statistics with* S (fourth edition, Springer (2002)).

Chantal D. Larose, Daniel T. Larose (2019) *Data Science Using Python and R*, Wiley.

Dealing with missing data

Introduction

In ML I Workshop 7 you used a data set called `Diabetes` to develop a logistic regression classifier.

The data set comes form the National Health and Nutrition Examination surveys (NHANES) in the USA. The full data contains 10 000 elements but there are a lot of missing vales.

We will now consider a version of the same data set containing 13 variables called `Diabetes2`.

Of these 13 variables, only three variables contain complete data.
One variable has 3508 missing values.

```
> Diabetes2[100:102,]
     YN Gender   Race1   BMI Age Pulse BPSysAve BPDiaAve HealthGen
100  No female Mexican 18.62  12    74      120       55      Good
101 Yes   male   White 34.04  80    60      125       56      Good
102 Yes female   Black 28.60  66   110      136       70      <NA>

    DaysPhysHlthBad DaysMentHlthBad LittleInterest Depressed
100               5               0           <NA>      <NA>
101               0               0           None      None
102              NA              NA           <NA>      <NA>
```

There is an R-Function called `na.omit()`, which returns the input vector/data frame excluding any element containing a missing value. Many data analysis programs do this internally to strip any missing values.

This is the quickest and easiest way to deal with missing values and is what we have implicitly used until now.

`na.omit()` on the `Diabetes2` data results in a data set with 6492 elements, we are throwing away a third of our data.

If we were to choose more than 13 variables, then we would have to throw out even more elements.

A quick and easy method is usually a poor method.

Aim of imputing missing values

Imputation is the name for any method which estimates missing data to fill-in the non-missing data.

The aim of imputing missing values is **not** to exactly predict the value that is missing.

The aim is to complete the data so that the entire data can be used, therefore increasing the model accuracy by reducing the variance.

The imputation process should not bias the estimates and predictions of a machine learning method.

Types of missing data

Missing completely at random (MCAR) The probability that a missing value occurs[1] is constant. The occurrence is independent from any other variable in the data

Missing at random (MAR) The probability is not constant, but can be fully explained using variables in the data set.

Missing not at random (MNAR) The probability that a missing value occurs depends on non observed variables.

Example of MAR data. A hospital records blood pressure (BP) for all it's private patients, but only on 60% of patients with state insurance. The probability of missingness in BP is not constant for all patients but probability of missingness is constant once the type of patient insurance is known.

[1] "missing value occurs" is often called "missingness"

Example of MNAR data: In a particular clinical study, patients who were given a new treatment were more likely to get headaches and drop out of the study. If the side effect information is not recorded, then the data collected at the end of the study is missing not at random (MNAR).

Example of MNAR data: In a particular clinical study, patients who were given a new treatment were more likely to get headaches and drop out of the study. If the side effect information is not recorded, then the data collected at the end of the study is missing not at random (MNAR).

There is a special case of missing not at random which can be a particular problem. When the missing status of a variable depends on the value of that variable itself. E.g. Obese people are less likely to report their weight.

For MCAR or MAR data then omitting the data should not bias the results, but we will lose prediction power. The variance of our estimates will increase, and so will the mean squared error (MSE).

It is usually difficult to asses whether the missing data is MCAR, MCAR or MNAR.

Univariate imputation methods: Fixed methods

Mean replacement For each variable independently, calculate the mean of the non-missing values and set the missing values equal to that mean.

E.g. `pulse` rate has 1437 missing values, and the mean pulse rate of the known values is 73.56.
Mean replacement will replace the missing values with 73.56.

This method is not good because the standard deviation of the imputed variable will decrease. The correlation with other variables will also decrease.

E.g. The standard deviation of the non-missing values for `pulse` is 12.15 but after mean replacement the standard deviation drops to 11.25

Missing as a category

For a **categorical variable** add a new level called missing. E.g. for `gender` define three levels: female, male and missing. This is a reasonable a method.

For a **numeric variable** replace the missing values with the mean or median value, and create a new variable called missing. This is not a good method unless the number of missing values is small. It can lead to biassed coefficients for the other variables.

Using logic rules

Sometimes a value can be imputed exactly or with high accuracy.

E.g. if the a person's `weight` and `BMI` is known, but `height` is missing, then it can be calculated from the definition of body mass index.

E.g. If a variable is number of `pregnancies` and a positive number is given then a missing value for `gender` can be imputed as female.

E.g. Only 1% of breast cancer cases are in males. This means that if there is an indication of current or previous breast cancer, it is acceptable to impute that the `gender` is female.

Univariate imputation methods: Random methods

Mean/Variance Simulation

Suppose the variable x_1 has missing values.
Compute the mean $\overline{x}_1$ and the standard deviation s_{x_1} of the known values.
Replace the missing values with simulations from a normal $N(\overline{x}_1, s_{x_1}^2)$ distribution.

This is a simple method and is effective provided x_1 is roughly normal distributed. If the distribution is very non-normal, then the variable after imputation will become closer to normally distributed.

The disadvantage with this and all univariate methods, is that it does not consider any co-dependencies with other variables.

Direct Random Sampling

Sample the non missing values with replacement, to fill in the missing values.

```
> tt
 [1]  3  1  0  2  0  2 NA  2 NA  1  5 NA  0  3  4 NA  0  4  4 NA
> tt[!is.na(tt)]
 [1] 3 1 0 2 0 2 2 1 5 0 3 4 0 4 4
> tt[is.na(tt)]<-sample(tt[!is.na(tt)],5,replace=TRUE)
> tt
 [1] 3  1  0  2  0  2  2  2  1  1  5  2  0  3  4  3  0  4  4  2
```

This approach is good when the missingness is independent of all other variables. Compared to the *mean/variance simulation* method, direct random sampling will preserve the distributional properties of the variable, and can be used for all data types.

It is often used as a first step in multivariate methods.

Multivariate imputation for one variable

With most data there is codependency (correlation) between some variables, which can and should be incorporated into our imputation method.

E.g. If we know a person's `height`, `age` and `gender`, then we can impute the `weight` much more accurately than just sampling the non-missing values of `weight`.

We will assume from here on that the data consists entirely of continuous variables. Allowing other types of variables is more complex, but follows similar methods.

Multivariate Regression

We can take the elements with complete data and fit a regression model to predict the missing values for another variable.

The type of regression can be any regression method but is often multiple linear regression.

E.g. Assume we have complete data (n=10 000) for `age`, `race`, `gender` and `BMI`. The variable `Pulse` has 1437 missing values.

We can fit a linear regression model

```
Pulse ~ BMI + age + race + gender
```

using the 8563 elements with complete data.

The predicted missing values should then be **simulated with an error term**. If $\widehat{y}_i$ is the predicted value from the regression model, then the imputed value should be $\widetilde{y} = \widehat{y}_i + \epsilon_i$, where ϵ_i is a random normal simulation with the model residual variance.

As with *Mean/Variance Simulation*, simulating the error term avoids a reduction in the variance of that variable. Remember, our aim is not to get the best prediction for the missing value but to preserve the overall properties of the complete data set.

The regression method is usually used for continuous variables.
With binary variables logistic regression can be used.
For a factor variable with more than two levels use a tree model or other type of simple classifier.

When more than one variable has missing values

The *multivariate regression method* easily adapts itself to imputing multiple variables, by looping over the the variables with missing values.

Suppose $x_1, \dots, x_k$ are variables with some missing values and $x_{k+1}, \dots, x_p$ are complete.

- Fit the regression $x_1 \sim x_{k+1} + x_{k+2} + \dots + x_p$, and impute all the missing values in x_1 including a simulated error term.
- Then fit the regression $x_2 \sim x_1 + x_{k+1} + x_{k+2} + \dots + x_p$, and impute all the missing values in x_2 with simulated error term.
- Continue until all variables $x_1, \dots, x_k$ have been completed.

A problem with this method is that if x_1 is correlated with x_2 then this is not considered when imputing x_1, but is considered when imputing x_2

So in practice the following method is used.

- First, use a simple univariate method such as *direct random sampling* to complete each variable. Keep a record of the “positions” of the original missing values.
- Fit the regression $x_1 \sim x_2 + x_3 + \ldots + x_p$, using all elements for which x_1 was *not* imputed.
- Re-impute all the originally missing values in x_1 including a simulated error term.
- Then fit the regression $x_2 \sim x_1 + x_3 + x_4 + \ldots + x_p$, using all elements for which x_2 was *not* imputed and re-impute.
- Repeat for $x_3, \ldots, x_k$.
- Because the x_1 imputations were based on some *direct random sampled* values of $x_2 \ldots, x_k$, it is best to repeat this loop a couple more times.

This method is a variant of the so called EM-Algorithm e.g. which was covered in soft clustering (ML I, Lecture 4).

Imputing missing values using Gibbs sampling

To paraphrase the Mean/Variance Simulation (Part 1)
Replace the missing values of x_1 with simulations from a normal $N(\widehat{\mu}_1, \widehat{\sigma}_1^2)$ distribution.

...

The disadvantage with this and all univariate methods, is that it does not consider the co-dependencies other variables.

Gibbs sampling (GS) is a method which simulates the missing data by estimating $\widehat{\mu}_1, \widehat{\sigma}_1, \widehat{\mu}_2, \widehat{\sigma}_2, \ldots \widehat{\mu}_p, \widehat{\sigma}_p$ simultaneously.

In fact, in each GS iteration you get a slightly different value for each of these parameter estimates which reflects that the estimates themselves have some variability.

What is Gibbs Sampling?

Later in this course you will learn about **Bayesian estimation** and conditional independence.

There is a wide class of simulation algorithms called **Markov Chain Monte Carlo (MCMC)** simulation which uses Bayesian estimation. These methods allow us to simulate parameter values that come from the so called **posterior distribution** of these parameters.

Simulating from the posterior distribution using MCMC is a powerful tool in Bayesian estimation.

Gibbs sampling is one of the most common MCMC algorithms. Each time a new parameter value is simulated it depends on the values of the known data, the values of the other parameters, and the other imputed values.

More details are in the further reading section for this week. We will only consider Gibbs sampling as a method of imputing missing values using the R-package `mice`.

Imputation using Gibbs Sampling in R: Quick example

R-package `mice`: Multivariate Imputation by Chained Equations, uses Gibbs sampling.

This package easily allows for multiple realisations of the complete data set. The data analysis (if using R) can then be repeated on each of the realisations (using one command) and an aggregate model obtained.

Data: `airquality` data set in R.
"Daily air quality measurements in New York, May to September 1973."

Variables: `Ozone`, `Solar.R` (Solar radiation in Langleys), `Wind` and `Temp`. The data frame `airquality` also contains date data which we will ignore.

Inspect the pattern of missing values

```
> library(mice)
> md.pattern(data)
    Wind Temp Solar.R Ozone
111    1    1       1     1  0
35     1    1       1     0  1
5      1    1       0     1  1
2      1    1       0     0  2
       0    0       7    37 44
```

`md.pattern()` outputs the table and the graphic.

`Wind` and `Temp` are complete.

`Solar.R` has 7 missing values.

`Ozone` has 37 missing values.

Two rows have both `Solar.R` and `Ozone` missing.

The function `mice` runs the Gibbs sampler.

`m=5`: obtain 5 different imputations (realisations)

`meth=norm` : the method used to obtain the conditional distributions and updates, `norm` means “Bayesian linear regression”

```
> tempData <- mice(data,m=5,maxit=50,meth='norm',seed=500)
 iter imp variable
  1    1  Ozone  Solar.R
... snip ...
```

These are the imputed values for `Ozone`. The 5 columns are the replications, and the row numbers are given.

```
> tempData$imp$Ozone
      1   2   3   4  5
5    14  18  19  19 18
10   41  45  44  32 22
25    6  19  18  19 18
... snip ...
115  21  22  32  14 13
119  77  76  50  85 66
150   9  24  23  41 13
```

To obtain a full data set with the 1st imputation.

Check using `md.pattern()`

```
> completedData <- complete(tempData,1)
> md.pattern(completedData)
 /\     /\
{  `---'  }
{  O   O  }
==>  V <==  No need for mice. This data set is completely observed.
 \  \|/  /
  `-----'

    Ozone Solar.R Wind Temp
153     1       1    1    1 0
        0       0    0    0 0
```

Suppose the original plan was to fit a multiple linear regression model to Temperature using the variables `Ozone`, `Solar.R` and `Wind`.

Now that we have complete data we can fit this regression model, but we have 5 realisations. Rather than choose one of the 5, we can fit the model **to each of the realisations** and “pool” (average) the results. This should be done using the function `pool()` in the `mice` package.

```
> modelFit1 <- with(tempData,lm(Temp~ Ozone+Solar.R+Wind))
> summary(pool(modelFit1))
               estimate   std.error statistic        df      p.value
(Intercept) 71.927280378 2.782518407 25.849705 107.83634 0.000000e+00
Ozone        0.173190306 0.024925656  6.948275  56.09270 2.933880e-10
Solar.R      0.009809583 0.007712214  1.271954  35.91124 2.061256e-01
Wind        -0.305981221 0.206219420 -1.483765  97.51888 1.407882e-01
```

Further reading

A couple of articles on the subject of imputing missing values are available online:

Ashcroft, Module 6 Missing Data.
Essentials of Data Analytics and Machine Learning Link
Tiny URL: `https://tinyurl.com/y3fzhhao`

Gelman and Hill, Chapter 25 Missing-data imputation.
Data Analysis Using Regression and Multilevel/Hierarchical Models. Link
`http://www.stat.columbia.edu/~gelman/arm/missing.pdf`

At the end of these notes there is some background reading to the theory of MCMC methods.

Workshop

- Take a 15 minute break.
- In Workshop 1
  - You will investigate the tropical atmosphere ocean data, which has a reasonable amount of missing values. High correlation between variables means that univariate imputation is insufficient.
  - You will complete both the tropical atmosphere ocean dataset and the diabetes data set using Gibbs Sampling with the `mice` package.
  - There is a template file with R-Code in Moodle.

Background theory

The details here are not examinable, unless they come up later in the course.

The Bayesian approach In statistics there are two main approaches to estimating a parameter, frequentist and Bayesian.

In a very simple case we have a sample $x_1, \dots, x_n$ and we estimate the mean of the underlying population (μ).

The frequentist approach says: we estimate μ using the sample mean, so

$$\widehat{\mu} = \overline{x} = \frac{\sum x_i}{n}.$$

The Bayesian approach says: we have a very rough initial idea about μ, this is called our *prior knowledge*.

We assign a distribution to this prior knowledge.
Our *prior parameter* μ_0 is assigned a *prior distribution* $\mu_0 \sim \pi_0$.

Once we have obtained the data $x_1, \ldots, x_n$ we can update our information about μ to give us a **posterior distribution** $\mu_1 \sim \pi_1$ using Bayes theorem.

This posterior distribution is more accurate than the prior because it makes use of the data.

Often the posterior μ_1 is written $\mu|\underline{x}$ “mu given the data $\underline{x}$”.

Bayes Theorem: from ML I Lecture 7

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Let A be the parameter μ and B be the data $x_1, \ldots, x_n$

Very roughly $P(A)$ is the prior distribution π_0 and $P(A|B)$ is the posterior distribution π_1 given the data.

MCMC

An explicit formula for the posterior distribution is only realistic under very specific circumstances.

Very often an algorithm is used to simulate parameter values coming from the posterior distribution.

Suppose we now have many parameters expressed as a vector $\boldsymbol{\theta}$. The posterior distribution is $\pi_1(\boldsymbol{\theta}|x_1, \ldots, x_p)$.
If we can simulate a vector $\boldsymbol{\theta}$ which comes from the posterior distribution, then we can use many simulated values to make inferences.

E.g. we simulate 1000 values of $\boldsymbol{\theta}$ which includes 1000 simulations of $\theta_1 = \mu_1$, we choose the mean of these 1000 values to be our estimate $\widehat{\mu}$.

There is a wide class of such simulation algorithms called **Markov Chain Monte Carlo (MCMC)** simulation. Simulating from the posterior distribution using MCMC is a powerful tool in Bayesian statistics.

The two most common MCMC algorithms are **Gibbs sampling** and the **Metropolis-Hastings Algorithm**.

The Metropolis-Hastings (MH) Algorithm is very adaptable and easy to code. The disadvantage is that it uses a rejection algorithm, and so typically twice as many iterations are needed compared to Gibbs sampling.

Gibbs sampling is easier to understand at the theoretical level and is quicker than MH, but more work is needed to define the Gibbs sampler probabilities and can be more sensitive to model assumptions.

All MCMC methods apply an iterative procedure. Arbitrary starting values for $\boldsymbol{\theta}$ are assigned, and new values of $\boldsymbol{\theta}$ are iteratively simulated using the “Markov chain”.

After a **burn-in** period the Markov chain converges and the values of $\boldsymbol{\theta}$ after each full iteration are simulations from the posterior distribution $\pi(\boldsymbol{\theta}|x_1, \ldots, x_p)$.

Gibbs sampling

If we know the conditional distribution of θ_1 given everything else, e.g. $\theta_1|\theta_2, \ldots, \theta_L, x_1, \ldots, x_p$, then we can simulate a value for θ_1 assuming the other values are known.

We update θ_1 with this simulated value using the *current* values of $\theta_2, \ldots, \theta_L$.

Then move on to θ_2:

If we know the conditional distribution of θ_2 given everything else, e.g. $\theta_2|\theta_1, \theta_3, \ldots, \theta_L, x_1, \ldots, x_p$, then we can simulate a value for θ_1 assuming the other values are known.

We update θ_2 with this simulated value using the *current* values of $\theta_1, \theta_3, \ldots, \theta_L$.

etc.

Once we have completed updated the value of θ_L, we have finished one iteration, and have a new value for $\boldsymbol{\theta}$.

Statistical theory shows that, after the burn in period, each updated $\boldsymbol{\theta}$ is a simulation from the posterior distribution $\pi(\boldsymbol{\theta}|x_1, \ldots, x_p)$.

Gibbs sampling to impute missing values

In the above algorithm the data $x_1, \dots, x_p$ are known and fixed.

If x_1 contains missing values then we can also update x_1 given everything else.

If x_1 has, say, 10 values to be imputed, then we simulate 10 values from the conditional distribution $x_1|\theta_1, \dots, \theta_L, x_2, \dots, x_p$
Continue updating all variables with missing data.
$x_2|\theta_1, \dots, \theta_L, x_1, x_3 \dots, x_p$ etc.

A full iteration now consists of updating all the θs and all the missing values.

Reminder: the aim is not exact predictions of the missing values, so once the burn in period is complete we only require one complete simulation.

The `mice` algorithm runs the algorithm a few more iterations, so we can obtain several different fully imputed data sets.