Multi-output machine learning with applications to genomics

Ekaterina Antonenko
May 26, 2023

Laboratoire d’informatique, École Polytechnique, IP Paris
Table of contents

About me

Introduction: Multi-output prediction

Regression: some problems

Classification: missing value imputation
About me
About me

Education

since 2020  PhD candidate: École Polytechnique
Data Science and Mining (DaSciM) team, Laboratoire d’informatique (LIX), CIFRE with Digitalent, Scientific advisor: Jesse Read

2012–2014  2-year program: Moscow Bioinformatics School

2009–2014  Diploma in Mathematics: Moscow State University

Employment

since 2020  École Polytechnique: Researcher, Teaching assistant (Machine learning bachelor course)

2017–2019  Math instructor, New York, USA

2013–2016  Analyst, Moscow, Russia
About me

Keywords

Multi-output prediction; Missing value imputation; Explainability and interpretability; Tree methods; Probabilistic inference.

Papers

pre-print E. Antonenko, R. Beigaitė, M. Mechenich, J. Read and I. Žliobaitė, Backward inference in probabilistic Regressor Chains with distributional constraints.

pre-print E. Antonenko, A. Carreño, J. Read, Autoreplicative Random Forests for missing value imputation.

pre-print M. Konnova, E. Antonenko, J. Read, Missing value imputation for genomics data using a Sequence Based Generative Adversarial Network (SBGAN).

ECML 2022 E. Antonenko, J. Read, Chains of Autoreplicative Random Forests for missing value imputation in high-dimensional datasets, [Best paper award].

IDA 2022 E. Antonenko, J. Read, Multi-modal ensembles of regressor chains for multi-output prediction, Advances in Intelligent Data Analysis XXI - 21st International Symposium.

Introduction: Multi-output prediction
Definition of a multi-output problem

**Given:** Dataset $D = \{(x^i, y^i)\}_{i=1}^{N}$ of $N$ samples:

- features $x^i = [x_1^i, ..., x_M^i]$
- outputs $y^i = [y_1^i, ..., y_L^i]$

**Goal:** Model $f(X) = y$ which outputs predictions $\hat{y}^i = [\hat{y}_1^i, ..., \hat{y}_L^i]$ having $D$ observed.
Definition of a multi-output problem

**Given:** Dataset $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^N$ of $N$ samples:

- features $x^i = [x^i_1, ..., x^i_M]$
- outputs $y^i = [y^i_1, ..., y^i_L]$

**Goal:** Model $f(X) = y$ which outputs predictions $\hat{y}^i = [\hat{y}^i_1, ..., \hat{y}^i_L]$ having $\mathcal{D}$ observed.

**Example:**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Age</th>
<th>Height</th>
<th>Body length</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>12,44</td>
<td>127,4</td>
<td>151</td>
<td>294,5</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>9,44</td>
<td>137,6</td>
<td>156</td>
<td>328</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>10,44</td>
<td>128,6</td>
<td>157</td>
<td>377</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>6,13</td>
<td>125,6</td>
<td>150</td>
<td>305,5</td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>6,15</td>
<td>139</td>
<td>156</td>
<td>325</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Definition of a multi-output problem

Given: Dataset $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^N$ of $N$ samples:

- features $x^i = [x_{i1}, ..., x_{iM}]$
- outputs $y^i = [y_{i1}, ..., y_{iL}]$

Goal: Model $f(X) = y$ which outputs predictions $\hat{y}^i = [\hat{y}_{i1}, ..., \hat{y}_{iL}]$ having $\mathcal{D}$ observed.

Example:

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>Age</th>
<th>Height</th>
<th>Body length</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>...</td>
<td>12,44</td>
<td>127,4</td>
<td>151</td>
<td>294,5</td>
</tr>
<tr>
<td>B</td>
<td>...</td>
<td>9,44</td>
<td>137,6</td>
<td>156</td>
<td>328</td>
</tr>
<tr>
<td>C</td>
<td>...</td>
<td>10,44</td>
<td>128,6</td>
<td>157</td>
<td>377</td>
</tr>
<tr>
<td>D</td>
<td>...</td>
<td>6,13</td>
<td>125,6</td>
<td>150</td>
<td>305,5</td>
</tr>
<tr>
<td>E</td>
<td>...</td>
<td>6,15</td>
<td>139</td>
<td>156</td>
<td>325</td>
</tr>
<tr>
<td>F</td>
<td>...</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Idea: to model these labels together in order to get better prediction performance
Some approaches to multi-output problems

- Independent models

\[ \hat{y} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(x), ..., h_L(x)] \]
Some approaches to multi-output problems

- **Independent models**

  \[ \hat{y} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(x), ..., h_L(x)] \]

- **Fully-cascaded chain**

  \[ \hat{y} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(x), h_2(x, \hat{y}_1), ..., h_L(x, \hat{y}_1, ..., \hat{y}_{L-1})] \]

  \( h_1, ..., h_L = \text{any single-output models} \)
Some approaches to multi-output problems

- **Independent models**

\[ \hat{y} = [\hat{y}_1, \ldots, \hat{y}_L] = [h_1(x), \ldots, h_L(x)] \]

- **Fully-cascaded chain**

\[ \hat{y} = [\hat{y}_1, \ldots, \hat{y}_L] = [h_1(x), h_2(x, \hat{y}_1), \ldots, h_L(x, \hat{y}_1, \ldots, \hat{y}_{L-1})] \]

\[ h_1, \ldots, h_L = \text{any single-output models} \]

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y_1 )</th>
<th>( y_2 )</th>
<th>( y_3 )</th>
<th>( y_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>( \ldots )</td>
<td>12,32</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Some approaches to multi-output problems

- **Independent models**

\[
\hat{y} = [\hat{y}_1, \ldots, \hat{y}_L] = [h_1(x), \ldots, h_L(x)]
\]

- **Fully-cascaded chain**

\[
\hat{y} = [\hat{y}_1, \ldots, \hat{y}_L] = [h_1(x), h_2(x, \hat{y}_1), \ldots, h_L(x, \hat{y}_1, \ldots, \hat{y}_{L-1})]
\]

\(h_1, \ldots, h_L = \text{any single-output models}\)

- **Multi-output Decision Trees and Random Forests**

\[
\hat{y} = [\hat{y}_1, \ldots, \hat{y}_L] = [h(x)]
\]

all labels are assigned simultaneously

BUT the metric to optimize is still decomposable
Some approaches to multi-output problems

- **Independent models**

\[ \hat{y} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(x), ..., h_L(x)] \]

- **Fully-cascaded chain**

\[ \hat{y} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(x), h_2(x, \hat{y}_1), ..., h_L(x, \hat{y}_1, ..., \hat{y}_{L-1})] \]

\[ h_1, ..., h_L = \text{any single-output models} \]

- **Multi-output Decision Trees and Random Forests**

\[ \hat{y} = [\hat{y}_1, ..., \hat{y}_L] = [h(x)] \]

all labels are assigned simultaneously

BUT the metric to optimize is still decomposable

<table>
<thead>
<tr>
<th></th>
<th>( y_1 )</th>
<th>( y_2 )</th>
<th>( y_3 )</th>
<th>( y_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F )</td>
<td>( 12,32 )</td>
<td>( 125,1 )</td>
<td>\textbf{143}</td>
<td>?</td>
</tr>
</tbody>
</table>
Some approaches to multi-output problems

- Independent models

\[ \hat{y} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(x), ..., h_L(x)] \]

- Fully-cascaded chain

\[ \hat{y} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(x), h_2(x, \hat{y}_1), ..., h_L(x, \hat{y}_1, ..., \hat{y}_{L-1})] \]

\( h_1, ..., h_L = \) any single-output models

<table>
<thead>
<tr>
<th></th>
<th>( x )</th>
<th>( y_1 )</th>
<th>( y_2 )</th>
<th>( y_3 )</th>
<th>( y_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F )</td>
<td>( \cdot \cdot \cdot )</td>
<td>12,32</td>
<td>125,1</td>
<td>143</td>
<td>289,3</td>
</tr>
</tbody>
</table>

- Multi-output Decision Trees and Random Forests

\[ \hat{y} = [\hat{y}_1, ..., \hat{y}_L] = [h(x)] \]

all labels are assigned simultaneously

BUT the metric to optimize is still decomposable
Regression: some problems
Problem 1. Multi-modal distributions

Multi-output models may not work well for multi-modal distributions.

One possible reason: inadequate choice of the loss function.

Most models optimize $\text{MSE} = \frac{1}{N} \sum_{j=1}^{N} (y_j - \hat{y}_j)^2$.

Optimizing MSE does not help to exploit the dependencies between the targets.
Uniform Cost Function (UCF) is an analogue of 0/1 loss for regression.

$$UCF(\delta) = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} 0 & \text{if } \|y^i - \hat{y}^i\|_2 < \frac{\delta}{2}, \\ 1 & \text{otherwise.} \end{cases}$$

Correntropy is a smooth version of UCF.

**FORMULA**
Uniform Cost Function (UCF) is an analogue of 0/1 loss for regression.

\[
\text{UCF}(\delta) = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} 
0 & \text{if } \|y^i - \hat{y}^i\|_2 < \frac{\delta}{2}, \\
1 & \text{otherwise}.
\end{cases}
\]

Correntropy is a smooth version of UCF.

**Goal = challenge:** optimize UCF or correntropy in Regressor Chains with any base estimator.
Uniform Cost Function (UCF) is an analogue of 0/1 loss for regression.

\[
\text{UCF}(\delta) = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} 
0 & \text{if } \| y^i - \hat{y}^i \|_2 < \frac{\delta}{2}, \\
1 & \text{otherwise.}
\end{cases}
\]

Correntropy is a smooth version of UCF.

Goal = challenge: optimize UCF or correntropy in Regressor Chains with any base estimator.

✓
Problem 2. Backward inference in Regressor Chains
Problem 2. Solution
Classification: missing value
imputation
Missing data

Why is data missing?

- Errors in sensors
- Human factor (reluctance to answer particular questions)
- Combining different studies
- ...

Why to impute the missing data?

- Most off-the-shelf statistical and machine learning methods cannot handle missing values
- Considering only instances with complete information can lead to a loss of necessary information and can yield a very poor or even empty dataset
- Missing data itself might be of interest

Types of missingness

- Missing Completely at Random (MCAR): entirely independently of feature values
- Missing at Random (MAR): depends only on the observed feature values
- Missing Not at Random (MNAR): depends on both the observed and the unobserved feature values
Why is data missing?

- Errors in sensors
- Human factor (reluctance to answer particular questions)
- Combining different studies
- ...

Why to impute the missing data?

- Most off-the-shelf statistical and machine learning methods cannot handle missing values
- Considering only instances with complete information can lead to a loss of necessary information and can yield a very poor or even empty dataset
- Missing data itself might be of interest
**Missing data**

Why is data missing?

- Errors in sensors
- Human factor (reluctance to answer particular questions)
- Combining different studies
- ...

Why to impute the missing data?

- Most off-the-shelf statistical and machine learning methods cannot handle missing values
- Considering only instances with complete information can lead to a loss of necessary information and can yield a very poor or even empty dataset
- Missing data itself might be of interest

Types of missingness

- **Missing Completely at Random (MCAR):** entirely independently of feature values
- **Missing at Random (MAR):** depends only on the observed feature values
- **Missing Not at Random (MNAR):** depends on both the observed and the unobserved feature values
Procedural and Iterative Imputation

Procedural

One procedure:

- Train a model on complete instances
- Use the fitted model to predict on instances containing missing values
- Correct observed values if changed

Iterative

Impute randomly, then repeat:

- Train a model on previous imputation
- Use the fitted model to predict on all instances
- Correct observed values if changed

Examples: mode, kNN, PCA, Autoencoders

+ Autoreplicative Random Forests
Procedural and Iterative Imputation

**Procedural**

One procedure:

- Train a model on **complete** instances
- Use the fitted model to predict on **instances containing missing values**
- Correct observed values if changed

**Examples:** mode, kNN, PCA, Autoencoders
Procedural and Iterative Imputation

**Procedural**
One procedure:
- Train a model on complete instances
- Use the fitted model to predict on instances containing missing values
- Correct observed values if changed

**Examples:** mode, kNN, PCA, Autoencoders

**Iterative**
Impute randomly, then repeat:
- Train a model on previous imputation
- Use the fitted model to predict on all instances
- Correct observed values if changed

**Examples:** MICE, MissForest, PCA, Autoencoders
**Procedural and Iterative Imputation**

### Procedural

One procedure:

- Train a model on **complete** instances
- Use the fitted model to predict on instances **containing missing values**
- Correct observed values if changed

**Examples:** mode, kNN, PCA, Autoencoders

### Iterative

Impute randomly, then repeat:

- Train a model on **previous imputation**
- Use the fitted model to predict on **all** instances
- Correct observed values if changed

**Examples:** MICE, MissForest, PCA, Autoencoders
Procedural and Iterative Imputation

**Procedural**
One procedure:

- Train a model on complete instances
- Use the fitted model to predict on instances containing missing values
- Correct observed values if changed

**Iterative**
Impute randomly, then repeat:

- Train a model on previous imputation
- Use the fitted model to predict on all instances
- Correct observed values if changed

**Examples:** mode, kNN, PCA, Autoencoders

**Examples:** MICE, MissForest, PCA, Autoencoders

+ Autoreplicative Random Forests
Why to use multi-label methods?

- Compared to one-by-one methods: may deeper exploit interdependencies between the targets + less computationally expensive
- Compared to neural networks: fewer parameters (good for low-sampled data)
- No need for hidden layers

Which methods?

- Decision Trees, Random Forests
- Classifier Chains, Multilabel k Nearest Neighbours, Random k-Labelsets, Conditional Dependency Networks, etc.
Why to use multi-label methods?

- Compared to one-by-one methods: may deeper exploit interdependencies between the targets + less computationally expensive
- Compared to neural networks: fewer parameters (good for low-sampled data)
- No need for hidden layers

Which methods?

- Decision Trees, Random Forests
- Classifier Chains, Multilabel k Nearest Neighbours, Random k-Labelsets, Conditional Dependency Networks, etc.

Autoencoders $\Rightarrow$ Autoreplicative Random Forests
Autoencoders $\Rightarrow$ Autoreplicative Random Forests

Why to use multi-label methods?

- compared to one-by-one methods: may deeper exploit interdependencies between the targets + less computationally expensive
- compared to neural networks: fewer parameters (good for low-sampled data)
- no need for hidden layers
Autoencoders $\implies$ Autoreplicative Random Forests

Why to use multi-label methods?

- compared to one-by-one methods: may deeper exploit interdependencies between the targets + less computationally expensive
- compared to neural networks: fewer parameters (good for low-sampled data)
- no need for hidden layers

Which methods?

- Decision Trees, Random Forests, Extra Trees
- Classifier Chains, Multilabel k Nearest Neighbours, Random k-Labelsets, Conditional Dependency Networks, etc.
Results: iterative ARF do converge

Imputation via Iterative Random Forests converges after several iterations
## Accuracy of imputation

<table>
<thead>
<tr>
<th></th>
<th>MVR</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MICE</td>
<td>0.658</td>
<td>0.715</td>
<td>0.741</td>
<td><strong>0.769</strong></td>
<td><strong>0.777</strong></td>
<td>0.884</td>
<td>0.884</td>
<td>0.879</td>
<td>0.867</td>
<td>0.850</td>
<td>0.761</td>
<td>0.768</td>
<td>0.748</td>
<td>0.754</td>
<td>0.735</td>
<td></td>
</tr>
<tr>
<td>itARF</td>
<td>0.730</td>
<td>0.740</td>
<td>0.747</td>
<td>0.734</td>
<td>0.707</td>
<td>0.824</td>
<td>0.850</td>
<td>0.832</td>
<td>0.815</td>
<td>0.789</td>
<td>0.652</td>
<td>0.672</td>
<td>0.645</td>
<td>0.660</td>
<td>0.620</td>
<td></td>
</tr>
<tr>
<td>pARF</td>
<td><strong>0.748</strong></td>
<td><strong>0.774</strong></td>
<td><strong>0.761</strong></td>
<td>0.671</td>
<td>0.478</td>
<td>0.804</td>
<td>0.779</td>
<td>0.600</td>
<td>–</td>
<td>–</td>
<td>0.639</td>
<td>0.696</td>
<td>0.650</td>
<td>0.694</td>
<td>0.635</td>
<td></td>
</tr>
<tr>
<td>itAE</td>
<td>0.608</td>
<td>0.618</td>
<td>0.604</td>
<td>0.584</td>
<td>0.569</td>
<td>0.653</td>
<td>0.607</td>
<td>0.608</td>
<td>0.584</td>
<td>0.590</td>
<td>0.721</td>
<td>0.732</td>
<td>0.692</td>
<td>0.711</td>
<td>0.710</td>
<td></td>
</tr>
<tr>
<td>pAE</td>
<td>0.580</td>
<td>0.494</td>
<td>0.491</td>
<td>0.538</td>
<td>0.428</td>
<td>0.653</td>
<td>0.622</td>
<td>0.594</td>
<td>–</td>
<td>–</td>
<td>0.721</td>
<td>0.718</td>
<td>0.692</td>
<td>0.690</td>
<td>0.497</td>
<td></td>
</tr>
<tr>
<td>itPCA</td>
<td>0.604</td>
<td>0.627</td>
<td>0.622</td>
<td>0.623</td>
<td>0.618</td>
<td>0.667</td>
<td>0.692</td>
<td>0.671</td>
<td>0.646</td>
<td>0.603</td>
<td>0.721</td>
<td>0.740</td>
<td>0.692</td>
<td>0.711</td>
<td>0.710</td>
<td></td>
</tr>
<tr>
<td>pPCA</td>
<td>0.600</td>
<td>0.587</td>
<td>0.578</td>
<td>0.537</td>
<td>0.441</td>
<td>0.655</td>
<td>0.639</td>
<td>0.620</td>
<td>–</td>
<td>–</td>
<td>0.721</td>
<td>0.671</td>
<td>0.688</td>
<td>0.626</td>
<td>0.411</td>
<td></td>
</tr>
</tbody>
</table>

### Mushroom [8,124 x 22]

- Complete cases: 80.1%
- MICE: 0.658
- itARF: 0.730
- pARF: **0.748**
- itAE: 0.608
- pAE: 0.580
- itPCA: 0.604
- pPCA: 0.600

### Soybean [307 x 35]

- Complete cases: 69.7%
- MICE: 0.665
- itARF: 0.714
- pARF: 0.636
- itAE: 0.697
- pAE: 0.638
- itPCA: 0.665
- pPCA: 0.595

### Tumor [339 x 17]

- Complete cases: 83.8%
- MICE: 0.761
- itARF: 0.721
- pARF: 0.639
- itAE: 0.667
- pAE: 0.679
- itPCA: 0.721
- pPCA: 0.600

### Votes [435 x 16]

- Complete cases: 85.3%
- MICE: 0.768
- itARF: 0.719
- pARF: 0.730
- itAE: 0.697
- pAE: 0.638
- itPCA: 0.665
- pPCA: 0.595

### Lymphography [148 x 18]

- Complete cases: 81.8%
- MICE: 0.750
- itARF: 0.714
- pARF: 0.636
- itAE: 0.700
- pAE: 0.679
- itPCA: 0.686
- pPCA: 0.693

### Financial Survey [6,394 x 212]

- Complete cases: 11.8%
- MICE: –
- itARF: –
- pARF: –
- itAE: –
- pAE: –
- itPCA: –
- pPCA: –

### Procedural ARFs: may be powerful when enough complete instances

### MICE: as powerful as computationally expensive

### Iterative ARFs: still powerful + significantly quicker
Time complexity

In theory:

- \( p \) = number of features
- \( n_{\text{iter}} \) = number of iterations

<table>
<thead>
<tr>
<th>Method</th>
<th>itARF</th>
<th>pARF</th>
<th>MICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{O}(n_{\text{iter}} \cdot p) )</td>
<td>( \mathcal{O}(p) )</td>
<td>( \mathcal{O}(n_{\text{iter}} \cdot p^2) )</td>
<td></td>
</tr>
</tbody>
</table>
Time complexity

In theory:

- $p =$ number of features
- $n_{iter} =$ number of iterations

\[
\begin{array}{ccc}
\text{itARF} & \text{pARF} & \text{MICE} \\
\mathcal{O}(n_{iter} \cdot p) & \mathcal{O}(p) & \mathcal{O}(n_{iter} \cdot p^2) \\
\end{array}
\]

In practice:
Usecase: Single Nucleotide Polymorphisms (SNP)

- Categorical: 0 (dominant-dominant), 1 (dominant-mutant), 2 (mutant-mutant)
- High-dimensional ($10^5 - 10^6$) and low-sampled ($10^2 - 10^3$)
- Ordering is important
- Missing values occur due to external mechanisms $\implies$ MCAR
Usecase: Single Nucleotide Polymorphisms (SNP)

- Categorical: 0 (dominant-dominant), 1 (dominant-mutant), 2 (mutant-mutant)
- High-dimensional \((10^5 \rightarrow 10^6)\) and low-sampled \((10^2 \rightarrow 10^3)\)
- Ordering is important
- Missing values occur due to external mechanisms \(\Rightarrow\) MCAR

Methods:

- reference-based (state-of-the-art for human data)
- reference-free (when reference panels are not available)
Chains of Autoreplicative Random Forests

- **Procedural approach:** one window of size $\Delta = \text{complete instances} + \text{instances with missing values}$
- **Chain of windows:** on each step, stacking $\nu$ windows with already imputed values as additional features
- **Ensemble of chains:** one forward chain, one backward chain, several random chains
Gridsearch for parameters $\Delta$ and $\nu$

<table>
<thead>
<tr>
<th>$\Delta$</th>
<th>1%</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nu$</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\Delta$</th>
<th>1%</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nu$</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

Lighter color / higher accuracy

$\Delta$: bigger fraction of missing values $\rightarrow$ smaller size of window $\implies$ can be estimated theoretically, no need for search

$\nu$: may depend on problem
<table>
<thead>
<tr>
<th></th>
<th>Maize [247 x 44,729]</th>
<th>Eucalyptus [970 x 33,398]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChARF</td>
<td>0.952 0.935 0.916 0.882 0.845</td>
<td>0.970 0.950 0.926 0.866 0.810</td>
</tr>
<tr>
<td>kNN (5/10)</td>
<td>0.803 0.802 0.801 0.798 0.794</td>
<td>0.851 0.849 0.847 0.843 0.839</td>
</tr>
<tr>
<td>mode</td>
<td>0.727 0.727 0.726 0.727 0.726</td>
<td>0.725 0.732 0.731 0.730 0.729</td>
</tr>
<tr>
<td>SVD (50/500)</td>
<td>0.647 0.648 0.645 0.643 0.636</td>
<td>0.788 0.788 0.788 0.785 0.780</td>
</tr>
<tr>
<td>MICE</td>
<td>– – – – –</td>
<td>– – – – –</td>
</tr>
</tbody>
</table>

|                  | Colorado Beetle [188 x 34,186] | Arabica Coffee [596 x 4,666] |
| ChARF            | 0.835 0.824 0.818 0.805 0.792 | 0.897 0.886 0.878 0.866 0.854 |
| kNN (50/10)      | 0.765 0.763 0.765 0.765 0.764 | 0.867 0.866 0.866 0.865 0.864 |
| mode             | 0.761 0.760 0.762 0.761 0.761 | 0.807 0.804 0.805 0.805 0.804 |
| SVD (50/100)     | 0.740 0.737 0.737 0.735 0.734 | 0.693 0.694 0.696 0.692 0.690 |
| MICE             | – – – – –           | 0.757 0.741 0.724 0.689 0.664 |

|                  | Wheat [388 x 9,763] | Coffea Canephora [119 x 45,748] |
| ChARF            | 0.821 0.808 0.795 0.777 0.762 | **0.799** 0.781 0.761 0.731 0.717 |
| kNN (10/10)      | **0.823** 0.819 0.818 0.815 0.811 | 0.737 0.739 0.737 **0.734** 0.731 |
| mode             | 0.729 0.727 0.729 0.729 0.727 | 0.691 0.693 0.692 0.692 0.691 |
| SVD (200/50)     | 0.622 0.618 0.609 0.600 0.594 | 0.456 0.453 0.450 0.449 0.450 |
| MICE             | 0.641 0.635 0.621 0.585 0.545 | – – – – –               |

- MICE: run with 10 neighbors for each feature, still worked only for smaller data
- Autoencoders: not taken into comparison (no complete data for training)
- Well-known methods for SNP imputation: \( k \) Nearest Neighbors, Single Value Decomposition
Conclusions

- Unusual and effective usage of multi-label methods, e.g. Random Forests:
  - autoreplication
  - missing value imputation
  - denoising
  - outlier detection
Conclusions

- Unusual and effective usage of multi-label methods, e.g. Random Forests:
  - autoreplication
  - missing value imputation
  - denoising
  - outlier detection

- We show how probabilistic training can be easily added to the model
Conclusions

- Unusual and effective usage of multi-label methods, e.g. Random Forests:
  - autoreplication
  - missing value imputation
  - denoising
  - outlier detection

- We show how probabilistic training can be easily added to the model

- ARF vs MICE: high quality and much faster
Conclusions

- Unusual and effective usage of multi-label methods, e.g. Random Forests:
  - autoreplication
  - missing value imputation
  - denoising
  - outlier detection
- We show how probabilistic training can be easily added to the model
- ARF vs MICE: high quality and much faster
- ARF vs Autoencoders:
  - no need for one-hot encoding $\Rightarrow$ less features.
  - lower time complexity $\Rightarrow$ works for high-dimensional datasets
  - no need for complete data
Questions of interest

• Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?

• For very wide datasets (>10,000 features) multi-target methods are very memory expensive

• Studies for MAR and MNAR scenarios

• Regression (e.g. gene expression)

• How can we avoid overfitting in Iterative RF?
Questions of interest

- Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?
- For very wide datasets (> 10,000 features) multi-target methods are very memory expensive
Questions of interest

- Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?
- For very wide datasets (> 10,000 features) multi-target methods are very memory expensive
- Studies for MAR and MNAR scenarios
Questions of interest

- Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?
- For very wide datasets (> 10,000 features) multi-target methods are very memory expensive
- Studies for MAR and MNAR scenarios
- Regression (e.g. gene expression)
Questions of interest

- Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?
- For very wide datasets (> 10,000 features) multi-target methods are very memory expensive
- Studies for MAR and MNAR scenarios
- Regression (e.g. gene expression)
- How can we avoid overfitting in Iterative RF?
Thank you!