Autoreplicative Random Forests for missing value imputation

Ekaterina Antonenko and Ander Carreño

DaSciM seminar, École Polytechnique, March 10, 2023

Laboratoire d'informatique, École Polytechnique, IP Paris

BCAM – Basque Center for Applied Mathematics

Missing data

Why is data missing?

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

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

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

Imputation methods

- Multiple Imputation by Chained Equations (MICE)
  - first, imputes randomly
  - then iteratively models each feature by all other features
- Autoencoders
  - are neural networks with an output equal to the input
  - model hidden structure
  - are able to “denoise” data
  - require complete data for training
- PCA transformation
  - is essentially similar to Autoencoders with one hidden layer and linear activation function

Imputation methods

- Multiple Imputation by Chained Equations (MICE)
  - first, imputes randomly
  - then iteratively models each feature by all other features
- Autoencoders
  - are neural networks with an output equal to the input
  - model hidden structure
  - are able to "denoise" data
  - require complete data for training
- PCA transformation
  - is essentially similar to Autoencoders with one hidden layer and linear activation function

Our contribution:

- framework unifying the methods above
- new methodology: Autoreplicative Random Forests (ARF)
- code implementation of the framework

Procedural and Iterative Imputation

Framework

Iterative: first, impute randomly; then update iteratively

Procedural: train on complete instances; predict for incomplete instances

	Iterative	**Procedural**
Single-target	MICE	
Multi-target		Autoencoders PCA

Framework

Iterative: first, impute randomly; then update iteratively

Procedural: train on complete instances; predict for incomplete instances

	Iterative	**Procedural**
Single-target	MICE	
Multi-target	Autoencoders PCA	Autoencoders PCA

Framework

Iterative: first, impute randomly; then update iteratively

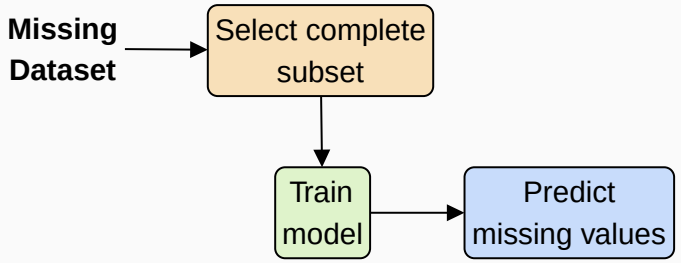
Procedural: train on complete instances; predict for incomplete instances

	Iterative	**Procedural**
Single-target	MICE	
Multi-target	Autoencoders PCA ARF	Autoencoders PCA ARF

Procedural models

One procedure:

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

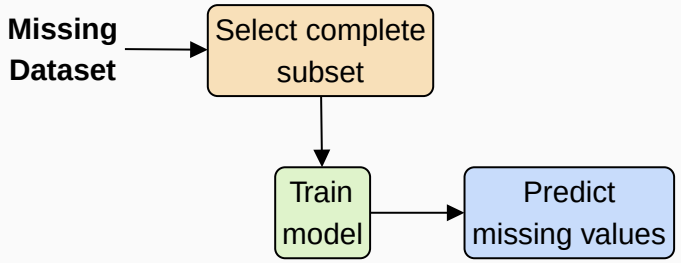


1: **procedure** PROCEDURAL IMPUTATION(X_{na})
2: $X_{train} \leftarrow X_{complete}$ ▷ Select complete cases for training
3: $\tilde{X}_{train} \leftarrow X_{train}$ corrupted with missing values ▷ Uniformly distributed, % of m.v. calculated from $X^{missing}$
4: $X_{test} \leftarrow X^{missing}$
5: Fit model on $(\tilde{X}_{train}, X_{train})$
6: $X_{pred} \leftarrow$ replace m.v. with predictions of fitted model on X_{test}

Procedural models

One procedure:

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



1: **procedure** PROCEDURAL IMPUTATION(X_{na})
2: $X_{train} \leftarrow X_{complete}$ ▷ Select complete cases for training
3: $\tilde{X}_{train} \leftarrow X_{train}$ corrupted with missing values ▷ Uniformly distributed, % of m.v. calculated from $X^{missing}$
4: $X_{test} \leftarrow X^{missing}$
5: Fit model on $(\tilde{X}_{train}, X_{train})$
6: $X_{pred} \leftarrow$ replace m.v. with predictions of fitted model on X_{test}

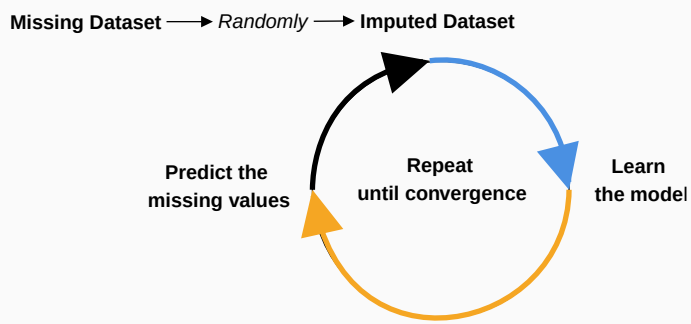
NB: needs enough complete data to train a reliable model

Iterative model

First, impute randomly

Then, iteratively until convergence is reached:

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



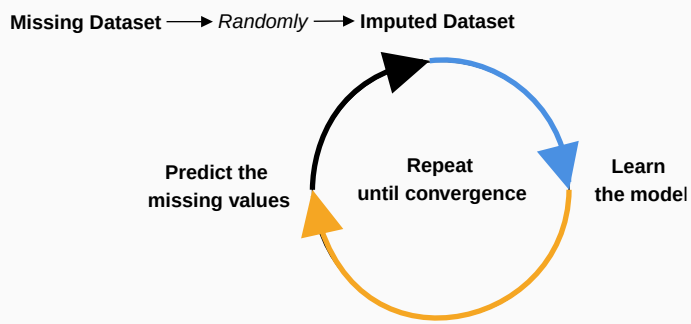
1: **procedure** ITERATIVE IMPUTATION(X_{na}, α)
2: $X^0_{imp} \leftarrow$ random imputation of m.v. in X_{na}
3: **while** $\Delta_{imp} > \alpha$ **do**
4: Fit model on (X_{na}, X^{n-1}_{imp})
5: $X^n_{imp} \leftarrow$ replace m.v. with predictions of fitted model on X_{na}
6: $\Delta_{imp} \leftarrow$ distance [accuracy] between X^n_{imp} and X^{n-1}_{imp}

Iterative model

First, impute randomly

Then, iteratively until convergence is reached:

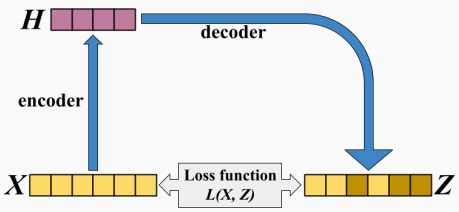
- Train a [multi-target] model on previous imputation
- Use the fitted model to predict on all instances
- Correct observed values if changed



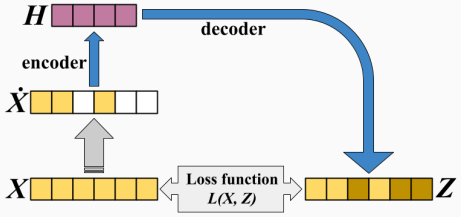
1: **procedure** ITERATIVE IMPUTATION(X_{na}, α)
2: $X^0_{imp} \leftarrow$ random imputation of m.v. in X_{na}
3: **while** $\Delta_{imp} > \alpha$ **do**
4: Fit model on (X_{na}, X^{n-1}_{imp})
5: $X^n_{imp} \leftarrow$ replace m.v. with predictions of fitted model on X_{na}
6: $\Delta_{imp} \leftarrow$ distance [accuracy] between X^n_{imp} and X^{n-1}_{imp}

Also: probabilistic extension (more on that later)

Autoencoders $\Longrightarrow$ Autoreplicative Random Forests

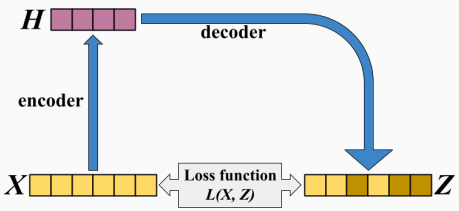


Autoencoder

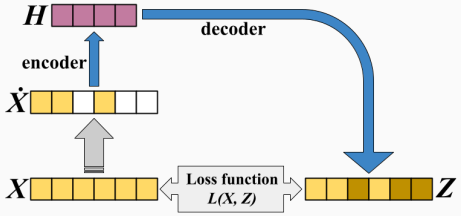


Denoising Autoencoder

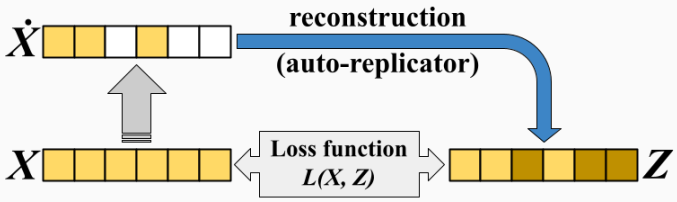
Autoencoders $\Longrightarrow$ Autoreplicative Random Forests



Autoencoder

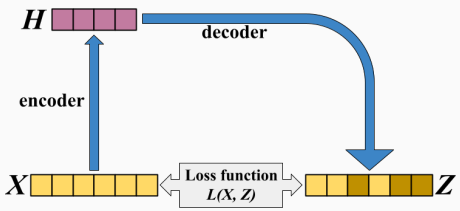


Denoising Autoencoder

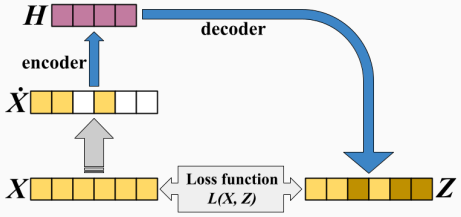


Autoreplicator

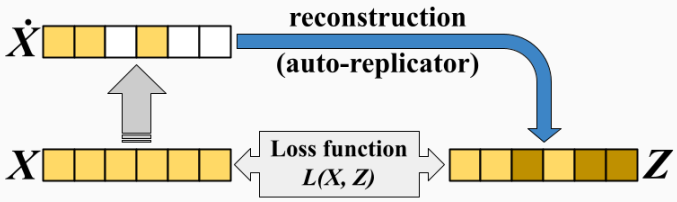
Autoencoders $\Longrightarrow$ Autoreplicative Random Forests



Autoencoder



Denoising Autoencoder

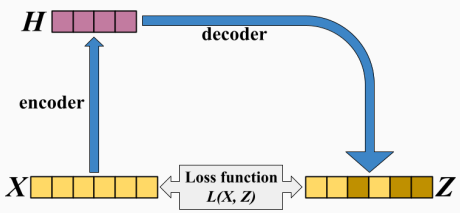


Autoreplicator

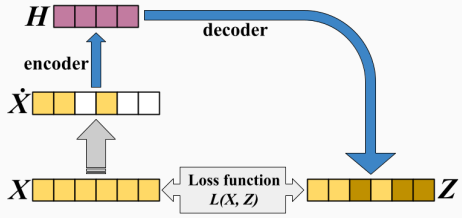
Why to use multi-label methods?

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

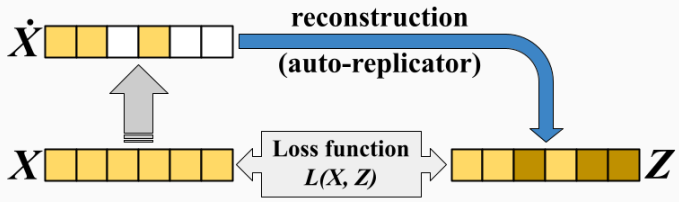
Autoencoders $\Longrightarrow$ Autoreplicative Random Forests



Autoencoder



Denoising Autoencoder



Autoreplicator

Why to use multi-label methods?

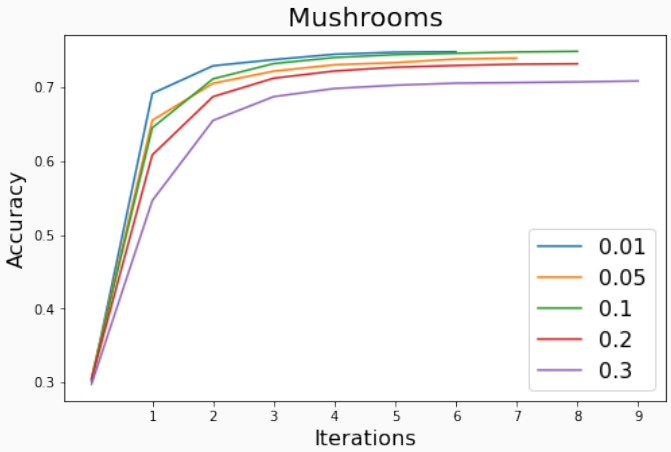
- compared to one-by-one methods: may deeper exploit interdependencies between the targets
- compared to neural networks: fewer parameters (good for low-sampled data)
- no need for hidden layers

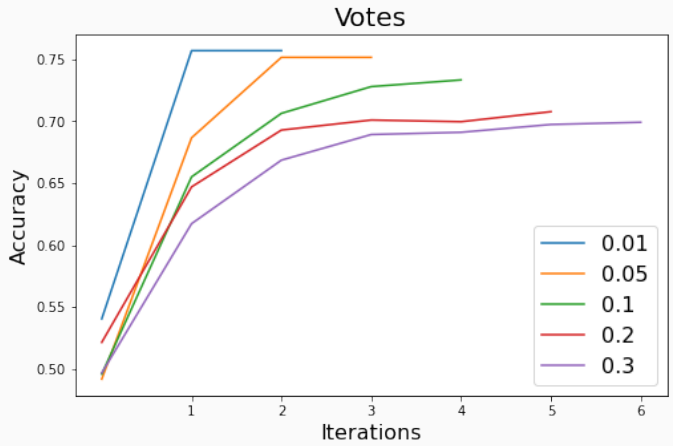
Which methods?

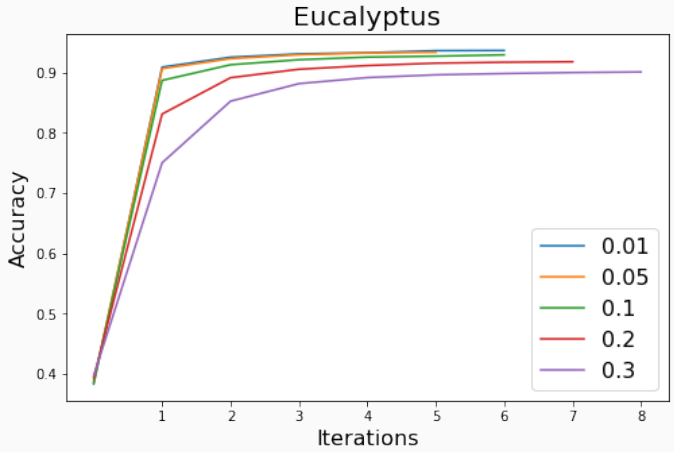
- Decision Trees, Random Forests, Extra Trees
- Classifier Chains, Multilabel k Nearest Neighbours, Random k-Labelsets, Conditional Dependency Networks, etc.

Results: iterative ARF do converge

Imputation via Iterative Random Forests converges after several iterations







Accuracy of imputation

MVR	0.01	0.05	0.1	0.2	0.3	0.01	0.05	0.1	0.2	0.3	0.01	0.05	0.1	0.2	0.3
	Mushroom [8,124 x 22]					Soybean [307 x 35]					Tumor [339 x 17]				
Complete cases	80.1%	32.3%	10.1%	0.7%	0.04%	69.7%	13.7%	1.0%	0%	0%	83.8%	38.9%	15.0%	1.2%	0.3%
MICE	0.658	0.715	0.741	**0.769**	**0.777**	**0.884**	**0.884**	**0.879**	**0.867**	**0.850**	**0.761**	**0.768**	**0.748**	**0.754**	**0.735**
itARF	0.730	0.740	0.747	0.734	0.707	0.824	0.850	0.832	0.815	0.789	0.652	0.672	0.645	0.660	0.620
pARF	**0.748**	**0.774**	**0.761**	0.671	0.478	0.804	0.779	0.600	–	–	0.639	0.696	0.650	0.694	0.635
itAE	0.608	0.618	0.604	0.584	0.569	0.653	0.607	0.608	0.584	0.590	0.721	0.732	0.692	0.711	0.710
pAE	0.580	0.494	0.491	0.538	0.428	0.653	0.622	0.594	–	–	0.721	0.718	0.692	0.690	0.497
itPCA	0.604	0.627	0.622	0.623	0.618	0.667	0.692	0.671	0.646	0.603	0.721	0.740	0.692	0.711	0.710
pPCA	0.600	0.587	0.578	0.537	0.441	0.655	0.639	0.620	–	–	0.721	0.671	0.688	0.626	0.411

	Votes [435 x 16]					Lymphography [148 x 18]					Financial Survey [6,394 x 212]				
Complete cases	85.3%	42.2%	18.5%	1.3%	0.4%	81.8%	40.5%	14.9%	2.7%	0%	11.8%	0%	0%	0%	0%
MICE	**0.768**	**0.795**	**0.771**	**0.768**	**0.782**	**0.750**	**0.679**	**0.665**	**0.648**	**0.651**	–	–	–	–	–
itARF	0.719	0.726	0.728	0.723	0.718	0.714	0.639	0.638	0.628	0.600	**0.684**	**0.677**	**0.676**	**0.667**	**0.661**
pARF	0.730	0.758	0.756	0.522	0.495	0.636	0.647	0.604	0.608	–	0.633	–	–	–	–
itAE	0.697	0.563	0.602	0.578	0.570	0.700	0.474	0.485	0.448	0.487	0.626	0.617	0.616	0.604	0.596
pAE	0.638	0.546	0.600	0.524	0.488	0.679	0.514	0.563	0.611	–	0.313	–	–	–	–
itPCA	0.665	0.583	0.567	0.572	0.570	0.686	0.513	0.477	0.468	0.484	0.653	0.645	0.645	0.634	0.627
pPCA	0.595	0.499	0.567	0.507	0.453	0.693	0.536	0.562	0.502	–	0.299	–	–	–	–

- 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_{iter} = number of iterations

itARF	pARF	MICE
$\mathcal{O}(n_{iter} \cdot p)$	$\mathcal{O}(p)$	$\mathcal{O}(n_{iter} \cdot p^2)$

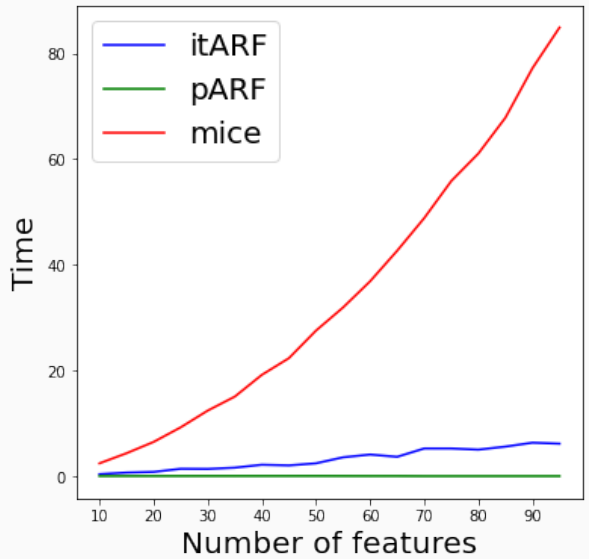
Time complexity

In theory:

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

itARF	pARF	MICE
$\mathcal{O}(n_{iter} \cdot p)$	$\mathcal{O}(p)$	$\mathcal{O}(n_{iter} \cdot p^2)$

In practice:



Probabilistic Autoreplicative Random Forests

Extension of iterative Autoreplicative Random Forests:

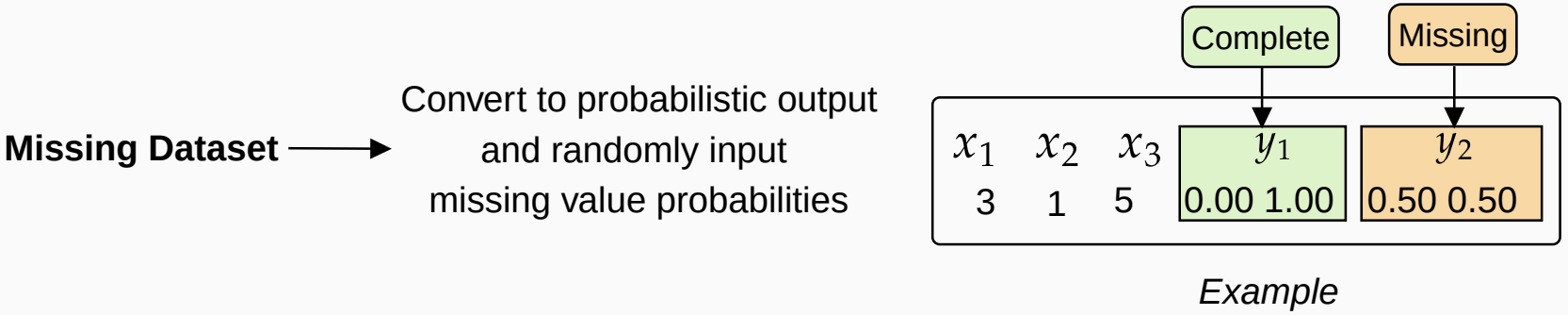
- First, randomly impute *with probabilities*
- In each iteration:
  - M imputations are sampled from the previous distribution
  - M trees of a Random Forest are trained on different imputations
  - The Random Forest produces one probabilistic imputation

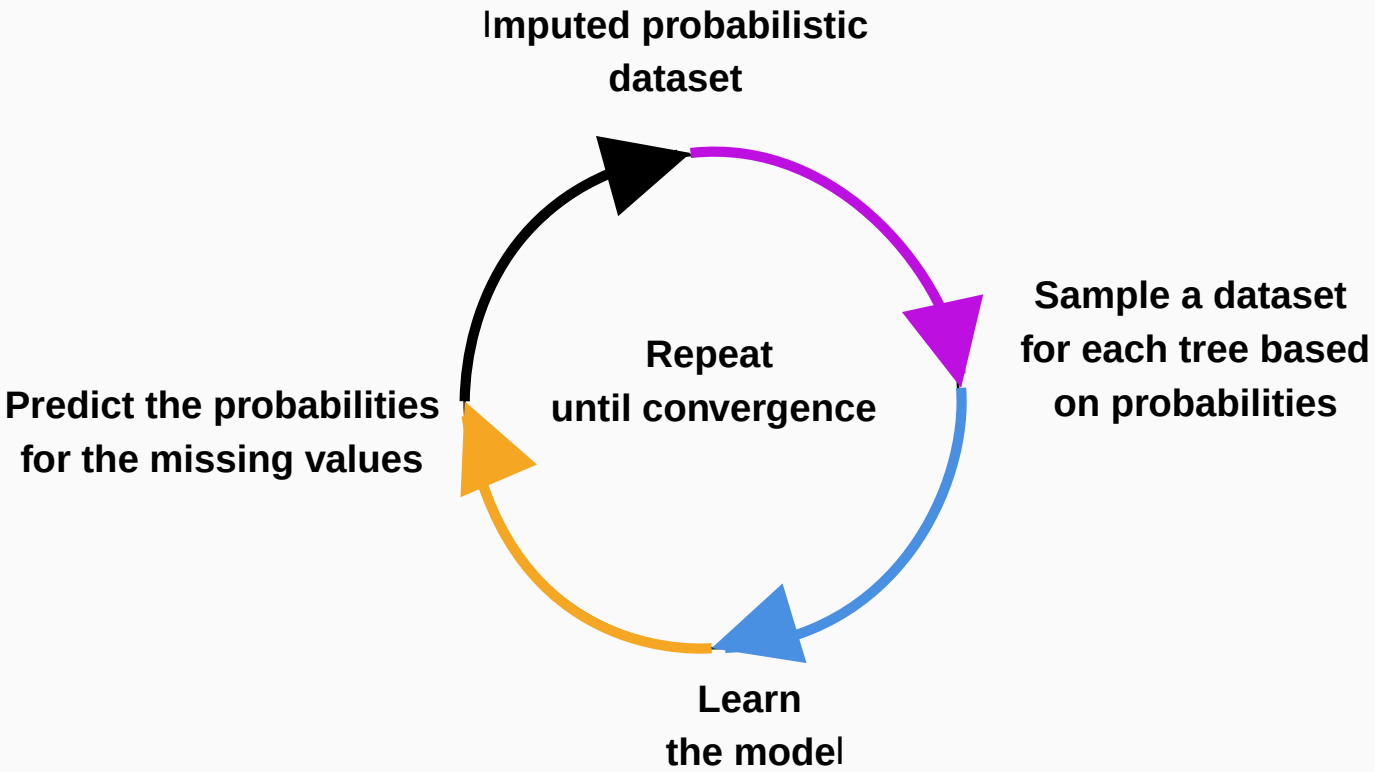
---

procedure PROBABILISTIC ITERATIVE IMPUTATION(X_{na}, α)
  $\mathcal{H}^0 \leftarrow \{h_1^0, h_2^0, \ldots, h_M^0\}$ ▷ Random Forest of M trees
  $\mathbf{p}_{imp}^0 \leftarrow$ random imputation with probabilities from $\{\mathcal{U}_{[0,1]}\}$
  while $\Delta_{imp} > \alpha$ **do**
    $\mathcal{H}^n \leftarrow \{h_1^n, h_2^n, \ldots, h_M^n\}$ ▷ Random Forest of M trees
    for $h_m^n \in \mathcal{H}^n$ **do**
      $X_{imp}^{n,m} \sim \mathbf{p}_{imp}^{n-1}$ ▷ Impute by sampling from distributions
      Fit a tree h_m^n on $(X_{na}, X_{imp}^{n,m})$
    $\mathbf{p}_{imp}^n \leftarrow$ probabilities provided by fitted $\mathcal{H}^n$
    $\Delta_{imp} \leftarrow$ distance between $\mathbf{p}_{imp}^n$ and $\mathbf{p}_{imp}^{n-1}$

Probabilistic Autoreplicative Random Forests

Graphically, we can see it as:





Probabilistic Autoreplicative Random Forests

Similarities with the Expectation Maximization (EM) algorithm.

BUT! EM converges in Likelihood. What about RF?

Which function are we optimizing?

- We say it has converged, when there are no changes in the predicted probabilities.

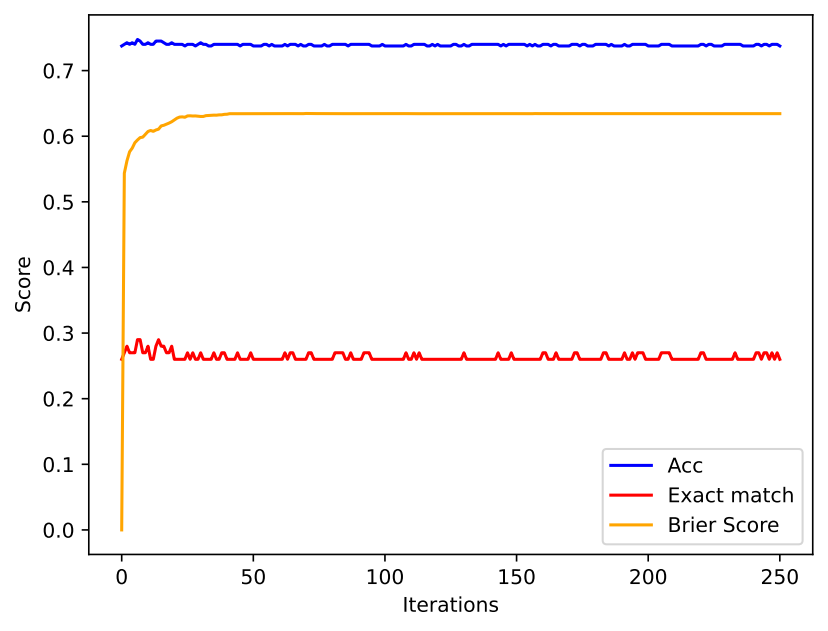
Probabilistic Autoreplicative Random Forests

In practice, we are overfitting the data.

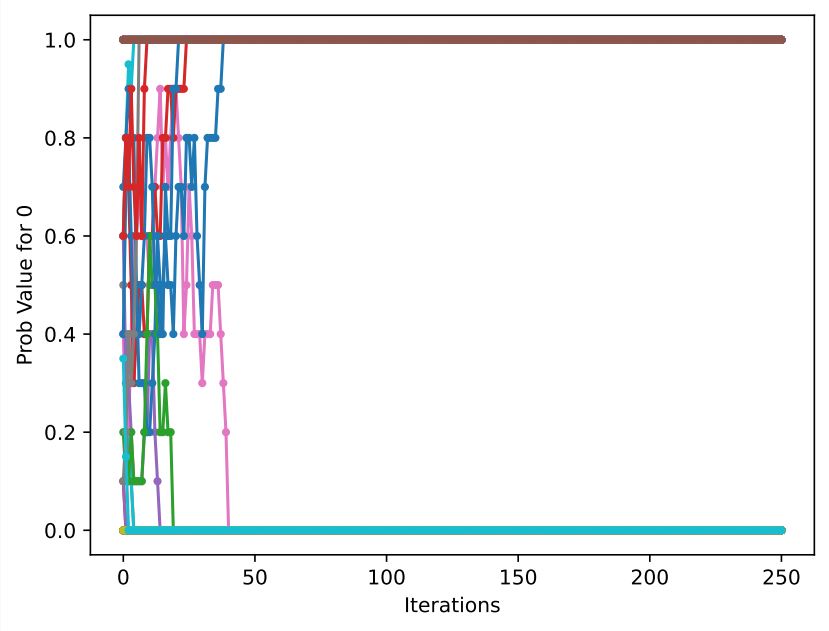
Experimental settings:

- Select the amount of $\mathbf{x}(m_x)$ and $\mathbf{y}(m_y)$ variables and its cardinality.
- Specify the % of missingness.
- Select the amount of instances we want to generate (n).
- Generate a $p(\mathbf{x}, \mathbf{y})$ assigning probabilities to each pair $(\mathbf{x}, \mathbf{y})$ from a normal distribution.
- Obtain the following metrics for each iteration:
  - Accuracy
  - Exact match: $\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(\hat{\mathbf{y}} = \mathbf{y})$
  - Brier Score: $\frac{1}{n}\sum_{i=1}^{n}(p(\mathbf{y}|\mathbf{x}) - \hat{p}(\mathbf{y}|\mathbf{x}))^2$

Probabilistic Autoreplicative Random Forests



(a) Scores obtained at each iteration.



(b) Each individual line represents the $\hat{p}(y_1 = 0|\mathbf{x})$ through the iterations.

Figure 2: Experiments on a synthetic dataset. $n = 100$, $m_x = 4$, $m_y = 4$. The cardinality of the variables is 4. 20% of instances with missing values.

Ideas to improve Probabilistic ARFs

We don't know exactly if we are going to converge to the true posterior distribution $p(\mathbf{y}|\mathbf{x})$
Is there anything related in the literature?

Estimation of Distribution Algorithms (EDAs) [1] were proposed for optimization problems.

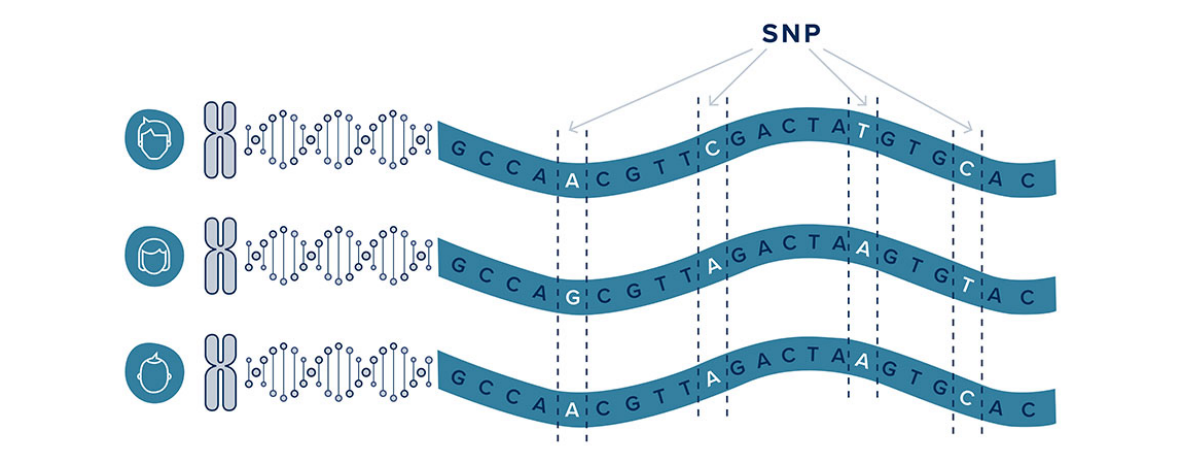
Ingredients:

- Population $\rightarrow$ Instances of our dataset.
- Spinoff generation $\rightarrow$ **How to sample a RF?**
- Fitness function $\rightarrow$ 0-1 loss, accuracy.
- Data selection $\rightarrow$ Based on Accuracy? Minimizing entropy?

[1] Larrañaga, P. and Lozano, J. A. (Eds.). (2001). Estimation of distribution algorithms: A new tool for evolutionary computation (Vol. 2). Springer Science Business Media.

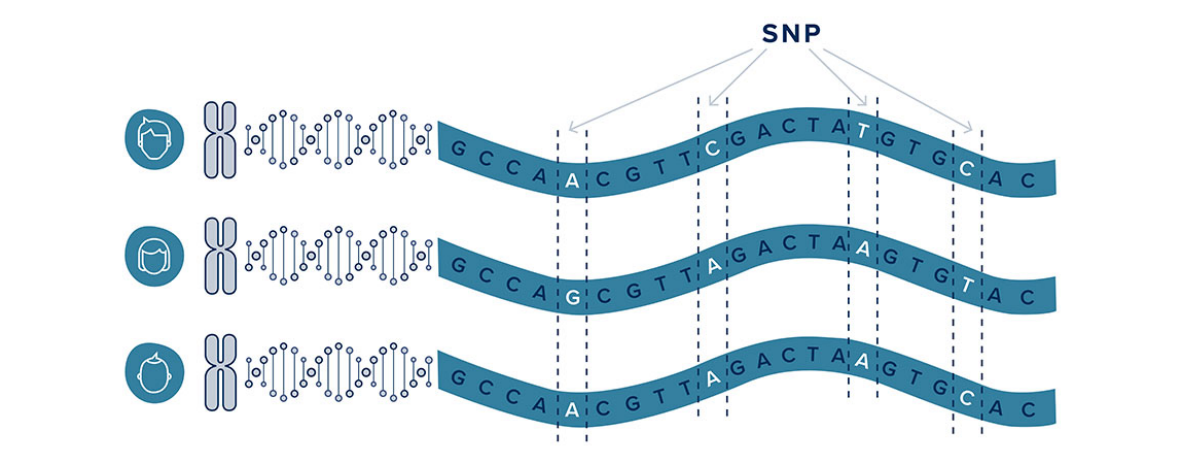
Chains of Autoreplicative Random Forests

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)

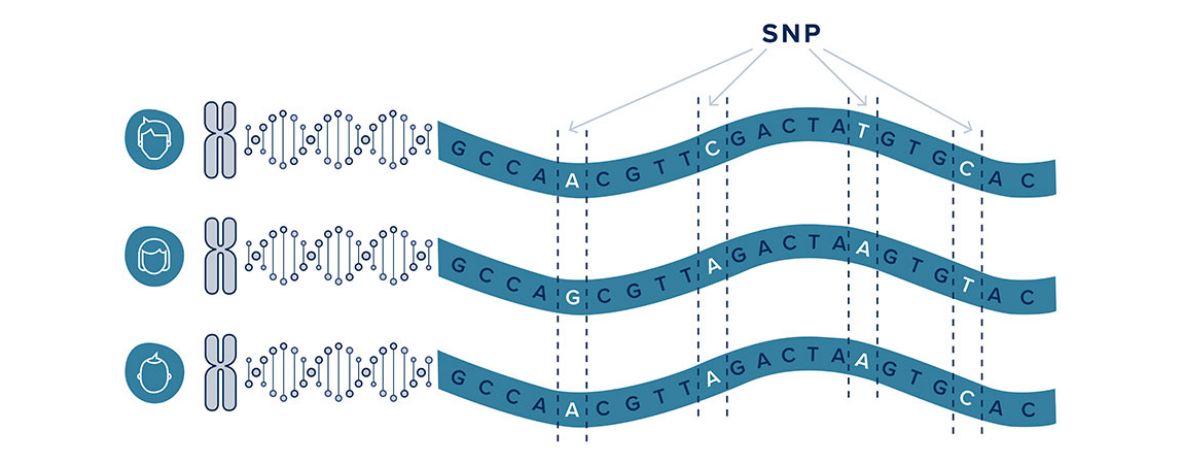


- Categorical: 0 (dominant-dominant), 1 (dominant-mutant), 2 (mutant-mutant)
- High-dimensional ($10^5 - 10^6$) and low-sampled ($10^2 - 10^3$)
- Ordering is important
- Missing values occur due to external mechanisms $\implies$ MCAR

Methods:

- reference-based (state-of-the-art for human data)
- reference-free (when reference panels are not available).

Usecase: Single Nucleotide Polymorphisms (SNP)



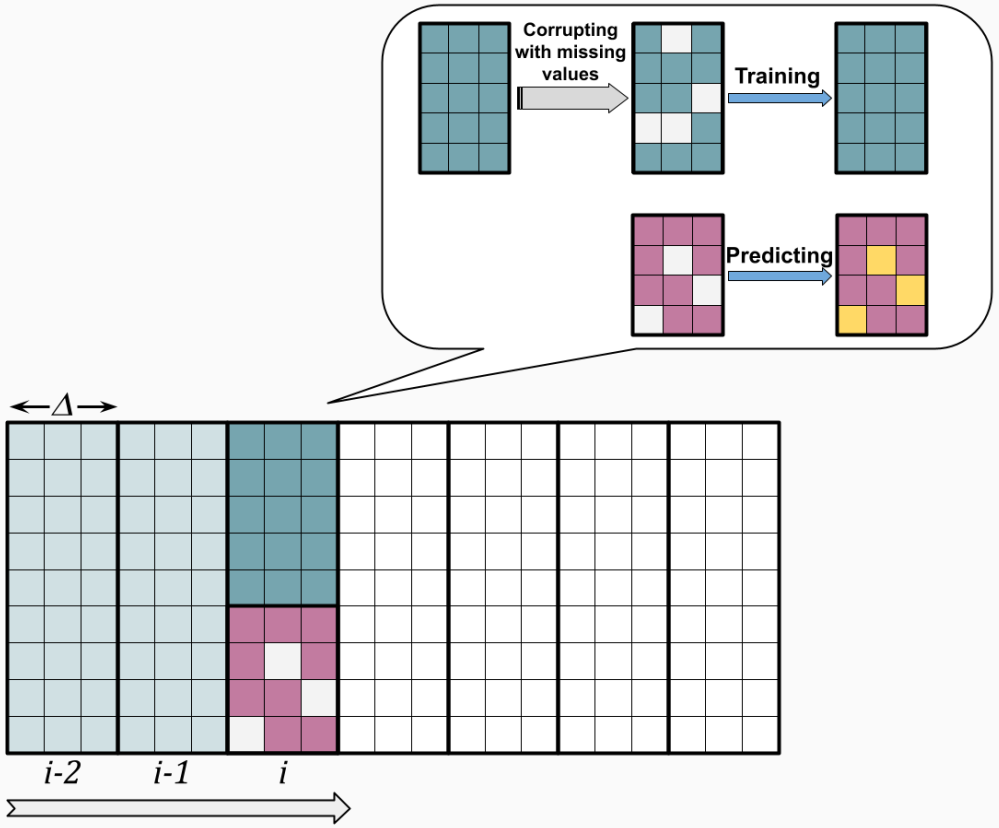
- Categorical: 0 (dominant-dominant), 1 (dominant-mutant), 2 (mutant-mutant)
- High-dimensional ($10^5 - 10^6$) and low-sampled ($10^2 - 10^3$)
- Ordering is important
- Missing values occur due to external mechanisms $\implies$ MCAR

Methods:

- reference-based (state-of-the-art for human data)
- reference-free (when reference panels are not available).

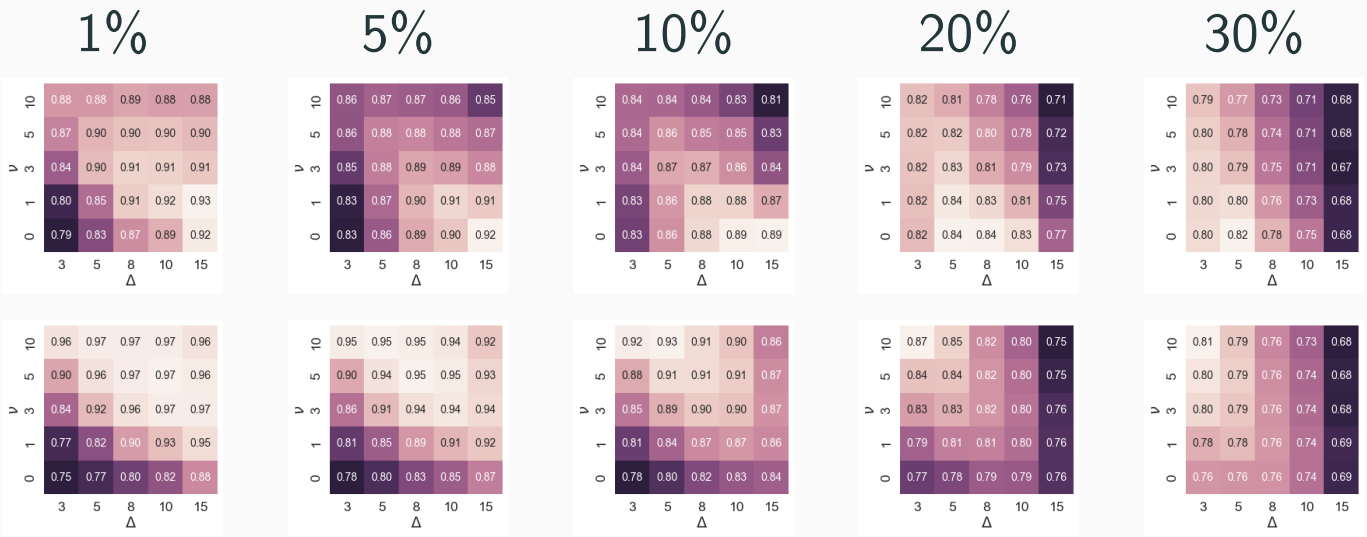
Other possible examples: gene expression arrays, classification problems in astronomy, tool development for finance data, and weather prediction.

Chains of Autoreplicative Random Forests



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

Gridsearch for parameters Δ and ν



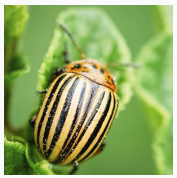
Lighter color / higher accuracy

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

ν: may depend on problem

Accuracy

	0.01	0.05	0.1	0.2	0.3	0.01	0.05	0.1	0.2	0.3
	Maize [247 x 44,729]					Eucalyptus [970 x 33,398]				
ChARF	**0.952**	**0.935**	**0.916**	**0.882**	**0.845**	**0.970**	**0.950**	**0.926**	**0.866**	0.810
kNN (5/10)	0.803	0.802	0.801	0.798	0.794	0.851	0.849	0.847	0.843	**0.839**
mode	0.727	0.727	0.726	0.727	0.726	0.725	0.732	0.731	0.730	0.729
SVD (50/500)	0.647	0.648	0.645	0.643	0.636	0.788	0.788	0.788	0.785	0.780
MICE	–	–	–	–	–	–	–	–	–	–



	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 $\implies$ less features.
  - lower time complexity $\implies$ 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?

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!