Chaining methods and their application to genomic data

DaSciM seminar

Ekaterina Antonenko

Supervisor: Jesse Read

November 26, 2021

Laboratoire d’informatique, École Polytechnique

DigitalentLab, MIA, Moteurs d’Intelligence Artificielle

Table of contents:

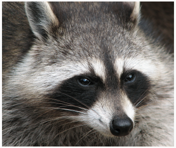
Introduction: Multi-output prediction

Chaining methods

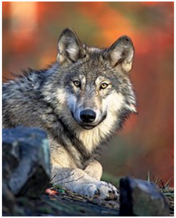
Imputation of missing values in genomic data

Introduction: Multi-output prediction

Machine learning



Raccoon? YES



Raccoon? NO

Machine learning

Raccoon? YES

Raccoon? NO

We want to find the **best** model *f*:

$$X \xrightarrow{f} y$$

$$f : f(X) = \hat{y},$$

such that the loss function $L(\hat{y}, y)$ is minimal.

Examples of loss functions:

- **Regression:** MSE, MAE
- **Classification:** 0/1 loss

Multi-output machine learning

Raccoon? YES

Multi-output machine learning

Raccoon? YES

Wolf? NO

Beaver? NO

Has stripes? YES

Has fur? YES

Multi-output machine learning

Raccoon? **YES**	$f_1(X) = y_1$
Wolf? **NO**	$f_2(X) = y_2$
Beaver? **NO**	$f_3(X) = y_3$
Has stripes? **YES**	$f_4(X) = y_4$
Has fur? **YES**	$f_5(X) = y_5$

Multi-output machine learning

Raccoon? YES

Wolf? NO

Beaver? NO

Has stripes? YES

Has fur? YES

$$f(X) = \mathbf{y}$$

$$\mathbf{y} = (y_1, y_2, y_3, y_4, y_5)$$

Multi-output machine learning

Raccoon? YES
Wolf? NO
Beaver? NO
Has stripes? YES
Has fur? YES

$f(X) = \mathbf{y}$

$\mathbf{y} = (y_1, y_2, y_3, y_4, y_5)$

Idea: to model these labels together in order to get better prediction performance

Chaining methods

Definition of a multi-output problem

Given:

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

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

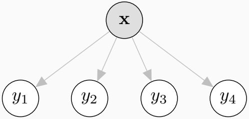
Goal:

Model which outputs predictions $\hat{\boldsymbol{y}}^i = [\hat{y}_1^i, ..., \hat{y}_L^i]$ having $\mathcal{D}$ observed.

	Raccoon?	Wolf?	Beaver?	Has stripes?	Has fur?
x_1	1	0	0	1	1
x_2	1	0	0	0	1
x_3	0	0	1	0	1
x_4	0	1	0	0	1
x_5	0	0	0	1	0
x_6	?	?	?	?	?

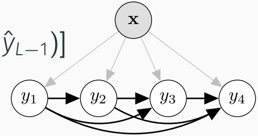
Some approaches to multi-output problems

- **Independent models** (= *binary relevance* for classification):



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

- **Fully-cascaded chain**:



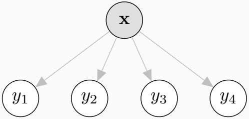
$$\hat{y} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(x), h_2(x, \hat{y}_1), ..., h_L(x, \hat{y}_1, ..., \hat{y}_{L-1})]$$

$h_1, h_2, ..., h_L$ = *Base Estimators* (i.e. any single-output models)

Some approaches to multi-output problems

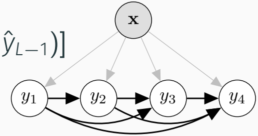
- **Independent models** (= *binary relevance* for classification):

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



- **Fully-cascaded chain**:

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

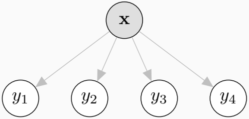


$h_1, h_2, ..., h_L$ = *Base Estimators* (i.e. any single-output models)

	Raccoon?	Wolf?	Beaver?	Has stripes?	Has fur?
x_1	?	?	?	?	?
x_2	?	?	?	?	?
x_3	?	?	?	?	?
x_4	?	?	?	?	?
x_5	?	?	?	?	?

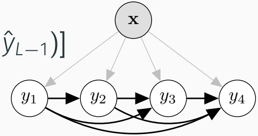
Some approaches to multi-output problems

- **Independent models** (= *binary relevance* for classification):



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

- **Fully-cascaded chain**:



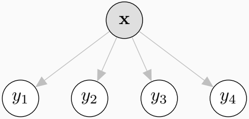
$$\hat{y} = [\hat{y}_1, \ldots, \hat{y}_L] = [h_1(\mathbf{x}), h_2(\mathbf{x}, \hat{y}_1), \ldots, h_L(\mathbf{x}, \hat{y}_1, \ldots, \hat{y}_{L-1})]$$

$h_1, h_2, \ldots, h_L$ = *Base Estimators* (i.e. any single-output models)

	Raccoon?	Wolf?	Beaver?	Has stripes?	Has fur?
x_1	1	?	?	?	?
x_2	1	?	?	?	?
x_3	0	?	?	?	?
x_4	0	?	?	?	?
x_5	0	?	?	?	?

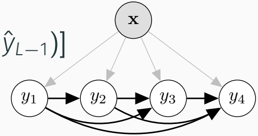
Some approaches to multi-output problems

- **Independent models** (= *binary relevance* for classification):



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

- **Fully-cascaded chain**:



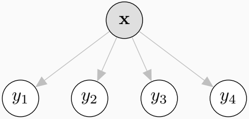
$$\hat{y} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(\mathbf{x}), h_2(\mathbf{x}, \hat{y}_1), ..., h_L(\mathbf{x}, \hat{y}_1, ..., \hat{y}_{L-1})]$$

$h_1, h_2, ..., h_L$ = *Base Estimators* (i.e. any single-output models)

	Raccoon?	Wolf?	Beaver?	Has stripes?	Has fur?
x_1	1	0	?	?	?
x_2	1	0	?	?	?
x_3	0	0	?	?	?
x_4	0	1	?	?	?
x_5	0	0	?	?	?

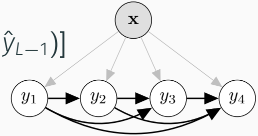
Some approaches to multi-output problems

- **Independent models** (= *binary relevance* for classification):



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

- **Fully-cascaded chain**:



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

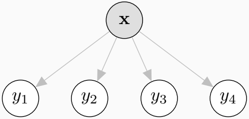
$h_1, h_2, ..., h_L$ = *Base Estimators* (i.e. any single-output models)

	Raccoon?	Wolf?	Beaver?	Has stripes?	Has fur?
x_1	1	0	0	?	?
x_2	1	0	0	?	?
x_3	0	0	1	?	?
x_4	0	1	0	?	?
x_5	0	0	0	?	?

Some approaches to multi-output problems

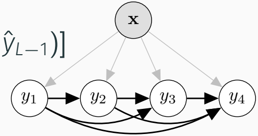
- **Independent models** (= *binary relevance* for classification):

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



- **Fully-cascaded chain**:

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

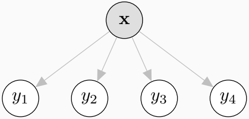


$h_1, h_2, ..., h_L$ = *Base Estimators* (i.e. any single-output models)

	Raccoon?	Wolf?	Beaver?	Has stripes?	Has fur?
x_1	1	0	0	1	?
x_2	1	0	0	0	?
x_3	0	0	1	0	?
x_4	0	1	0	0	?
x_5	0	0	0	1	?

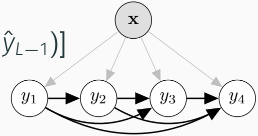
Some approaches to multi-output problems

- **Independent models** (= *binary relevance* for classification):



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

- **Fully-cascaded chain**:



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

$h_1, h_2, ..., h_L$ = *Base Estimators* (i.e. any single-output models)

	Raccoon?	Wolf?	Beaver?	Has stripes?	Has fur?
x_1	1	0	0	1	1
x_2	1	0	0	0	1
x_3	0	0	1	0	1
x_4	0	1	0	0	1
x_5	0	0	0	1	0

Does the chaining approach work?

Classification

Classifier Chains have proved to be flexible and effective and have achieved state-of-the-art empirical performance

- *Classifier Chains: A Review and Perspectives*, Read et al., 2021

Regression

Regressor Chains show relatively few advantages compared to individual regression models. State-of-the-art methods:

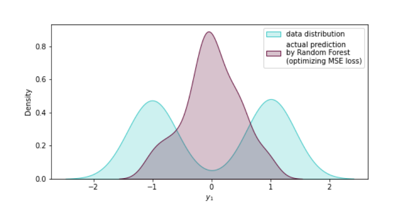
- Multi-output Decision Trees (DT)
- Multi-output Random Forests (RF)
- Independent Regressors (IR)

Regressor chains: why don't they work?

1. **Inadequate choice of the loss function to optimize**

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

   - Example: multi-modal distribution $\implies$ standard models may be inappropriate.



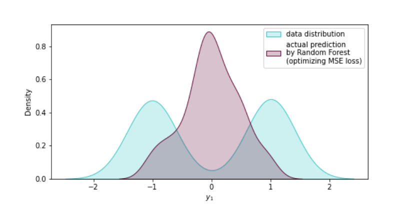
   - Optimizing MSE does not help to exploit the dependencies between the targets.

Regressor chains: why don’t they work?

1. **Inadequate choice of the loss function to optimize**

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

   - Example: multi-modal distribution $\implies$ standard models may be inappropriate.



   - Optimizing MSE does not help to exploit the dependencies between the targets.

2. **Insufficient depth of the model**
   - Only one round of prediction
   - Fixed cascaded order

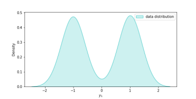
Our improvements for Regressor Chains

1. Multi-Modal Ensembles of Regressor Chains (mmERC) =

   = Ensembles of Regressor Chains +

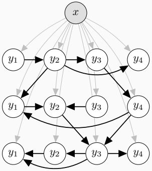
   + Mechanism 1 (BaseEstimator level) +

   + Mechanism 2 (Ensemble level)



   E. Antonenko, J. Read, *Multi-Modal Ensembles of Regressor Chains for Multi-Output Prediction*, submitted to IDA-2022 conference (Rennes, France).

2. Layered Regressor Chains (LRC)



Multi-Modal Ensembles of Regressor Chains (mmERC)

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

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

Goal = problem: optimize UCF.

Multi-Modal Ensembles of Regressor Chains (mmERC)

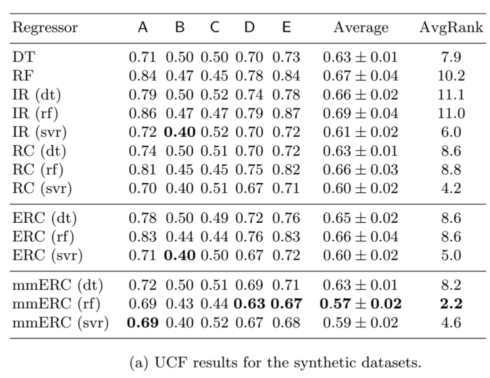
Uniform Cost Function (UCF) is an analogue of 0/1 loss for regression.

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

Goal = problem: optimize UCF.

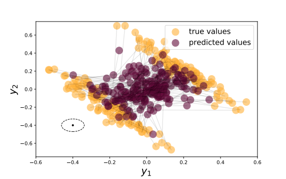
	ERC	mmERC
BaseEstimator level	single round of training	train on all dataset, choose portion of data giving best predictions, retrain on this part
Ensemble level	mean for all predictions	choose the biggest cluster of predictions, take mean for this cluster only

mmERC: results on a synthetic dataset

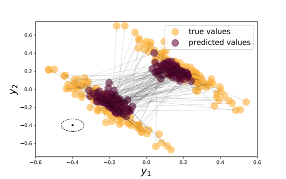


Regressor	A	B	C	D	E	Average	AvgRank
DT	0.71	0.50	0.50	0.70	0.73	0.63 ± 0.01	7.9
RF	0.84	0.47	0.45	0.78	0.84	0.67 ± 0.04	10.2
IR (dt)	0.79	0.50	0.52	0.74	0.78	0.66 ± 0.02	11.1
IR (rf)	0.86	0.47	0.47	0.79	0.87	0.69 ± 0.04	11.0
IR (svr)	0.72	**0.40**	0.52	0.70	0.72	0.61 ± 0.02	6.0
RC (dt)	0.74	0.50	0.51	0.70	0.72	0.63 ± 0.01	8.6
RC (rf)	0.81	0.45	0.45	0.75	0.82	0.66 ± 0.03	8.8
RC (svr)	0.70	0.40	0.51	0.67	0.71	0.60 ± 0.02	4.2
ERC (dt)	0.78	0.50	0.49	0.72	0.76	0.65 ± 0.02	8.6
ERC (rf)	0.83	0.44	0.44	0.76	0.83	0.66 ± 0.04	8.6
ERC (svr)	0.71	**0.40**	0.50	0.67	0.72	0.60 ± 0.02	5.0
mmERC (dt)	0.72	0.50	0.51	0.69	0.71	0.63 ± 0.01	8.2
mmERC (rf)	0.69	0.43	0.44	**0.63**	**0.67**	**0.57 ± 0.02**	**2.2**
mmERC (svr)	**0.69**	0.40	0.52	0.67	0.68	0.59 ± 0.02	4.6

(a) UCF results for the synthetic datasets.

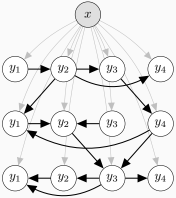


Random Forest



mmERC (RF)

Layered Regressor Chains (LRC)



Example of a single chain in ensemble:
$L = 4$ targets,
$K = 3$ layers,
$p = 2$ inter-layer connections

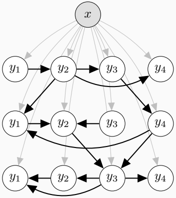
Single chain:

- Generate a random DAG in each of K layers
- Add p inter-layer connections for each two neighbour layers

Ensemble:

- Train n random layered chains
- Extract predictions from the last layer
- For each target, take mean of all predictions

Layered Regressor Chains (LRC)



Example of a single chain in ensemble:
$L = 4$ targets,
$K = 3$ layers,
$p = 2$ inter-layer connections

Single chain:

- Generate a random DAG in each of K layers
- Add p inter-layer connections for each two neighbour layers

Ensemble:

- Train n random layered chains
- Extract predictions from the last layer
- For each target, take mean of all predictions

Comparing to NNs:

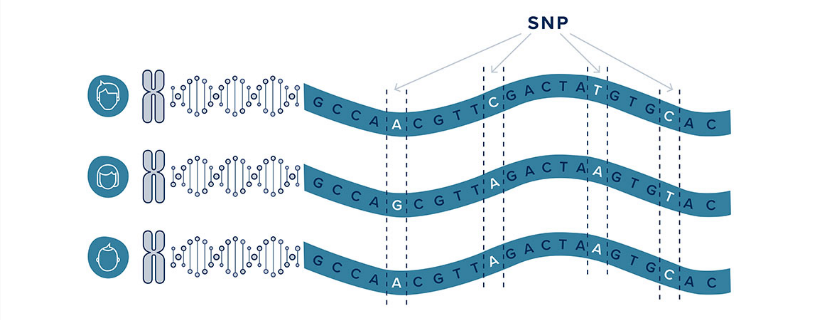
- No back-propagation $\implies$ any BaseEstimator
- Less connections $\implies$ lower complexity
- Work better for small datasets
- Need to train using labels from training data on each layer

mmERC + LRC: results under UCF

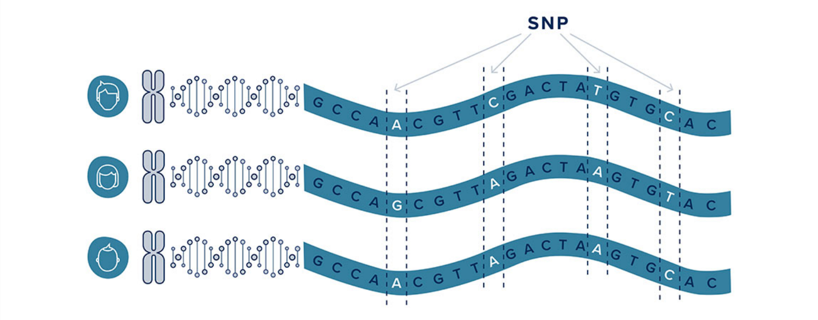
Regressor	andro	atp1d	atp7d	jura	oes97	osales	rf1	slump	scpf	AvgRank
DT	0.72	0.33	0.34	0.48	0.88	**0.94**	0.01	0.66	0.18	7.39
RF	0.71	0.23	0.35	0.41	0.80	0.94	0.02	0.54	0.16	5.94
IR (dt)	0.70	0.32	0.39	0.46	0.91	0.97	0.03	0.53	0.18	8.28
IR (rf)	0.57	0.20	0.33	0.41	**0.78**	0.98	0.03	0.44	0.17	4.61
IR (svr)	0.64	0.70	0.86	0.60	0.93	1.00	0.10	0.46	0.23	10.67
RC (dt)	0.69	0.32	0.40	0.49	0.91	0.97	0.02	0.43	0.17	6.94
RC (rf)	0.66	0.22	0.36	**0.37**	**0.78**	0.99	0.02	0.38	0.18	5.11
RC (svr)	0.96	0.48	0.71	0.55	0.98	0.98	0.82	0.67	0.22	11.83
LRC (dt)	0.57	0.22	0.30	0.43	0.86	**0.94**	0.03	0.50	0.17	5.11
LRC (rf)	0.55	**0.19**	0.25	0.37	**0.78**	0.96	0.01	**0.36**	0.18	**2.56**
LRC (svr)	0.89	0.74	0.81	0.61	0.95	1.00	0.30	0.46	0.23	11.78
LRC + mmERC (dt)	**0.47**	0.25	**0.25**	0.39	0.89	0.98	**0.01**	0.49	**0.16**	4.11
LRC + mmERC (rf)	0.72	0.23	0.43	0.40	0.87	0.99	0.02	0.41	0.20	7.00
LRC + mmERC (svr)	0.94	0.98	0.96	0.67	1.00	1.00	0.57	0.71	0.25	13.67

Imputation of missing values in genomic data

Single Nucleotide Polymorphisms (SNP)


SNP
G C C A A C G T T C G A C T A T G T G C A C
G C C A G C G T T A G A C T A A G T G T A C
G C C A A C G T T A G A C T A A G T G C A C

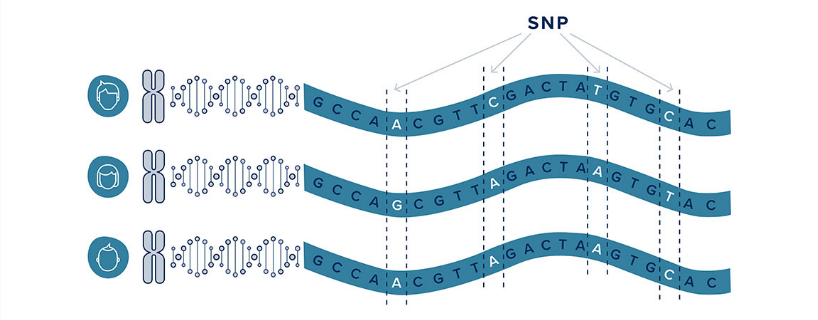
Single Nucleotide Polymorphisms (SNP)



A = prevalent variant (wild-type), a = rare variant (mutant)

$$AA = 0 \qquad Aa = 1 \qquad aa = 2$$

Single Nucleotide Polymorphisms (SNP)


SNP
G C C A A C G T T C G A C T A T G T G C A C
G C C A G C G T T A G A C T A A G T G T A C
G C C A A C G T T A G A C T A A G T G C A C

A = prevalent variant (wild-type), a = rare variant (mutant)

$$AA = 0 \qquad\qquad Aa = 1 \qquad\qquad aa = 2$$

$$\left.\begin{array}{l}\textbf{Features: } M = 10^5 - 10^7\\ \textbf{Samples: } N = 10^3 - 10^5\end{array}\right\} \text{“Fat data” X} \in \{0, 1, 2\}^{N\times M}$$

Genome-Wide Association Studies (GWAS)

What for:

- predicting phenotypes (i.e. diseases/traits)
- prioritizing features

Genome-Wide Association Studies (GWAS)

What for:

- predicting phenotypes (i.e. diseases/traits)
- prioritizing features

State-of-the-art:

Perform a statistical test of association between each feature and the phenotype

Genome-Wide Association Studies (GWAS)

What for:

- predicting phenotypes (i.e. diseases/traits)
- prioritizing features

State-of-the-art:

Perform a statistical test of association between each feature and the phenotype

Limitations:

- lack of statistical power
- dependencies between targets are not taken into consideration

Genome-Wide Association Studies (GWAS)

What for:

- predicting phenotypes (i.e. diseases/traits)
- prioritizing features

State-of-the-art:

Perform a statistical test of association between each feature and the phenotype

Limitations:

- lack of statistical power
- dependencies between targets are not taken into consideration

Overcoming limitations:

Machine learning methods (e.g. linear models on graph networks / deep NNs)

Missing values

Problem:

Missing values (up to $\sim 30 - 40\%$)

Missing values

Problem:

Missing values (up to $\sim 30 - 40\%$)

Imputation of missing values:

- can add more variants to a genetic region and increase the chances of identifying a causal variant
- facilitates the combination of results in meta-analysis when a number of studies is combined
- increases the accuracy in detecting an association signal

Imputation methods for SNP datasets

Reference-based

(fastPHASE (Scheet and Stephens, 2006), IMPUTE4 (Bycroft et al., 2017), BEAGLE (Browning et al., 2018), MACH (Li et al., 2010), etc.)

- Short chromosome segments can be inherited from a distant common ancestor
- In presence of reference panel of high quality: state-of-the-art. The accuracy is mainly determined by quality of the reference panel, and concordance of ethnicity between the data and the reference panel

Imputation methods for SNP datasets

Reference-based

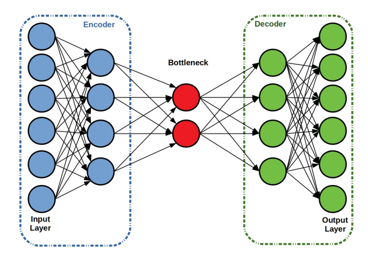
(fastPHASE (Scheet and Stephens, 2006), IMPUTE4 (Bycroft et al., 2017), BEAGLE (Browning et al., 2018), MACH (Li et al., 2010), etc.)

- Short chromosome segments can be inherited from a distant common ancestor
- In presence of reference panel of high quality: state-of-the-art. The accuracy is mainly determined by quality of the reference panel, and concordance of ethnicity between the data and the reference panel

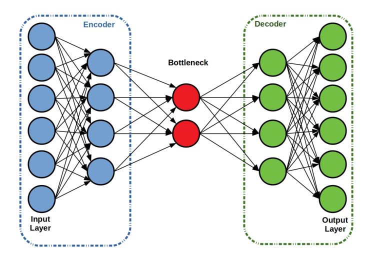
Reference-free

- Replacement with mean, median, or mode statistics
- Nearest Neighbors, Random Forests, Logistic Regression
- Autoencoders (Chen and Shi, 2019)

Autoencoders

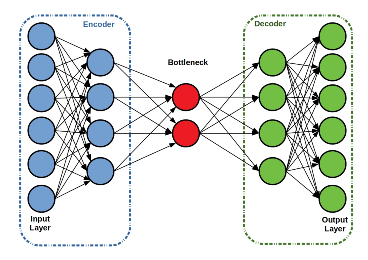


Autoencoders



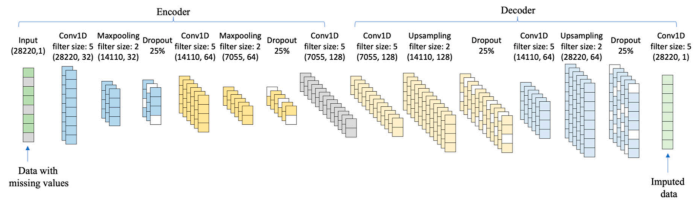
Denoising autoencoders: minimizing $L(X, g(f(\widetilde{X})))$

Autoencoders



Denoising autoencoders: minimizing $L(X, g(f(\widetilde{X})))$

Sparse Convolutional Denoising Autoencoders (Chen and Shi, 2019):


Encoder
Decoder
Input (28220,1)
Conv1D filter size: 5 (28220, 32)
Maxpooling filter size: 2 (14110, 32)
Dropout 25%
Conv1D filter size: 5 (14110, 64)
Maxpooling filter size: 2 (7055, 64)
Dropout 25%
Conv1D filter size: 5 (7055, 128)
Conv1D filter size: 5 (7055, 128)
Upsampling filter size: 2 (14110, 128)
Dropout 25%
Conv1D filter size: 5 (14110, 64)
Upsampling filter size: 2 (28220, 64)
Dropout 25%
Conv1D filter size: 5 (28220, 1)
Data with missing values
Imputed data

Chains for SNP missing values imputation

- Deep learning methods *vs.* fat data: **?**
- Imputing missing values with a mode is known to be more effective than taking a mean
- Chaining approach can be useful in predicting missing values

Chains for SNP missing values imputation

- Deep learning methods *vs.* fat data: **?**
- Imputing missing values with a mode is known to be more effective than taking a mean
- Chaining approach can be useful in predicting missing values

X_1	X_2
🟩	🟩
🟩	?
?	🟩
🟩	?
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	?

Chains for SNP missing values imputation

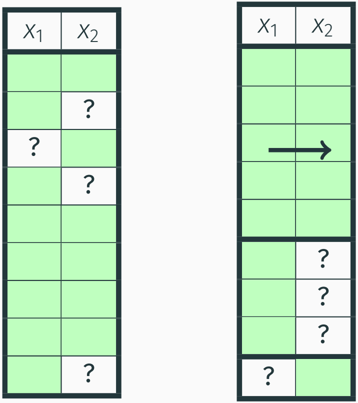
- Deep learning methods *vs.* fat data: **?**
- Imputing missing values with a mode is known to be more effective than taking a mean
- Chaining approach can be useful in predicting missing values

X_1	X_2
🟩	🟩
🟩	?
?	🟩
🟩	?
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	?

X_1	X_2
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	?
🟩	?
🟩	?
?	🟩

Chains for SNP missing values imputation

- Deep learning methods *vs.* fat data: **?**
- Imputing missing values with a mode is known to be more effective than taking a mean
- Chaining approach can be useful in predicting missing values

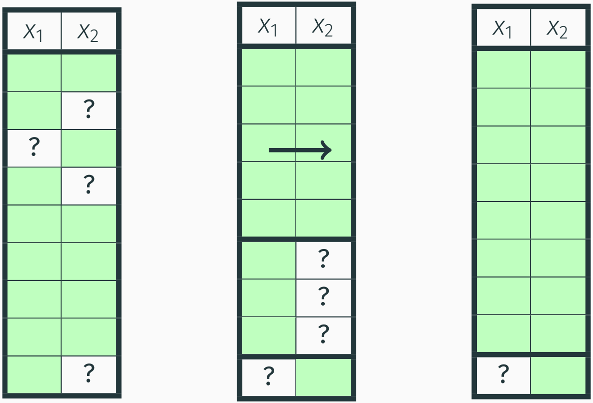


X_1	X_2
🟩	🟩
🟩	?
?	🟩
🟩	?
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	?

X_1	X_2
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	?
🟩	?
🟩	?
?	🟩

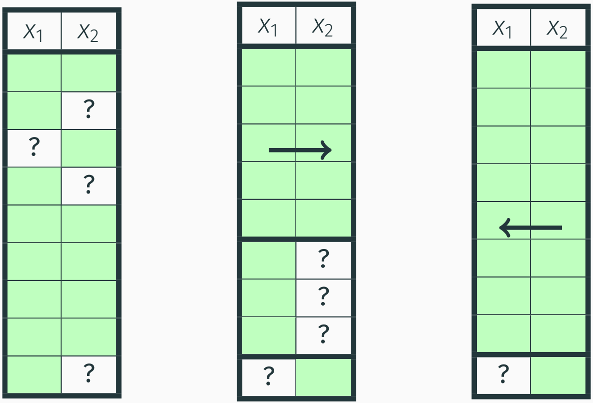
Chains for SNP missing values imputation

- Deep learning methods *vs.* fat data: **?**
- Imputing missing values with a mode is known to be more effective than taking a mean
- Chaining approach can be useful in predicting missing values



Chains for SNP missing values imputation

- Deep learning methods *vs.* fat data: **?**
- Imputing missing values with a mode is known to be more effective than taking a mean
- Chaining approach can be useful in predicting missing values



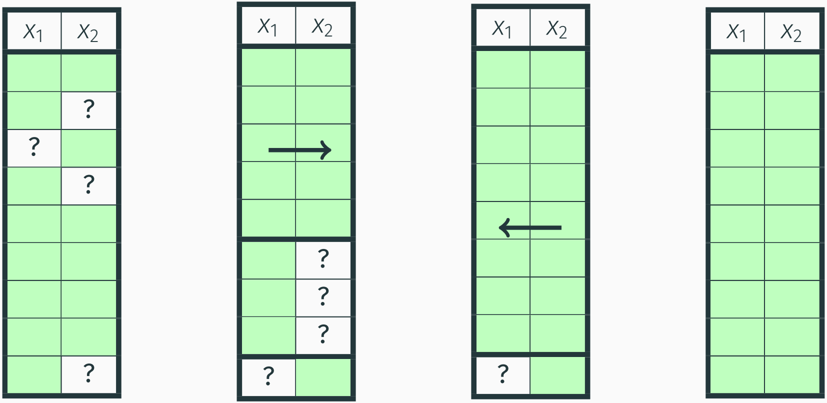
X_1	X_2
🟩	🟩
🟩	?
?	🟩
🟩	?
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	?

X_1	X_2
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	?
🟩	?
🟩	?
?	🟩

X_1	X_2
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
🟩	🟩
?	🟩

Chains for SNP missing values imputation

- Deep learning methods *vs.* fat data: **?**
- Imputing missing values with a mode is known to be more effective than taking a mean
- Chaining approach can be useful in predicting missing values



Preliminary results

Autoencoders idea for non-NN multi-target methods: $\mathsf{f}(\widetilde{\mathsf{X}}) \rightarrow \mathsf{X}$

Preliminary results

Autoencoders idea for non-NN multi-target methods: $\mathsf{f}(\widetilde{\mathsf{X}}) \to \mathsf{X}$

Mushroom dataset

Not SNP, but only categorical features (22 features, 8124 samples)

	Mode	DT	RF	IR(dt)	CC(dt)	MLPc
+ imputed	0.598	0.781	0.753	0.764	0.789	0.739
- changed	0.000	0.071	0.078	0.0002	0.001	0.085

Real SNP dataset (Blueberry)

Slice of data (100 features, 1000 samples)

	Mode	DT	RF	IR(dt)	CC(dt)	MLPc
+ imputed	0.769	0.734	0.796	0.773	0.787	0.821
- changed	0.000	0.229	0.184	0.002	0.004	0.135

Thank you!