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>
<thead>
<tr>
<th>Table of contents:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction: Multi-output prediction</td>
</tr>
<tr>
<td>Chaining methods</td>
</tr>
<tr>
<td>Imputation of missing values in genomic data</td>
</tr>
</tbody>
</table>
Introduction: Multi-output prediction
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

\[ f_1(X) = y_1 \]

Wolf? NO

\[ f_2(X) = y_2 \]

Beaver? NO

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
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

\[ f(X) = y \]

\[ y = (y_1, y_2, y_3, y_4, y_5) \]
Multi-output machine learning

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

\[ f(X) = y \]

\[ y = (y_1, y_2, y_3, y_4, y_5) \]

Idea: to model these labels together in order to get better prediction performance
Chaining methods
Definition of a multi-output problem

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

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

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

<table>
<thead>
<tr>
<th></th>
<th>Raccoon?</th>
<th>Wolf?</th>
<th>Beaver?</th>
<th>Has stripes?</th>
<th>Has fur?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$x_2$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_5$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$x_6$</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
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**:

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

  \( h_1, h_2, ..., h_L = \text{Base Estimators} \) (i.e. any single-output models)
Some approaches to multi-output problems

- **Independent models** (\(=\) binary relevance for classification):
  \[
  \hat{y} = [\hat{y}_1, ..., \hat{y}_L] = [h_1(x), ..., h_L(x)]
  \]

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

\(h_1, h_2, ..., h_L = \text{Base Estimators} \) (i.e. any single-output models)

<table>
<thead>
<tr>
<th>(x)</th>
<th>Racoon?</th>
<th>Wolf?</th>
<th>Beaver?</th>
<th>Has stripes?</th>
<th>Has fur?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_1)</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>(x_2)</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>(x_3)</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>(x_4)</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>(x_5)</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
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**:

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

  $h_1, h_2, ..., h_L = \text{Base Estimators}$ (i.e. any single-output models)

<table>
<thead>
<tr>
<th></th>
<th>Raccoon?</th>
<th>Wolf?</th>
<th>Beaver?</th>
<th>Has stripes?</th>
<th>Has fur?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>1</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$x_2$</td>
<td>1</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$x_5$</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
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**:

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

  \[ h_1, h_2, ..., h_L \text{ = Base Estimators (i.e. any single-output models)} \]

---

<table>
<thead>
<tr>
<th></th>
<th>Raccoon?</th>
<th>Wolf?</th>
<th>Beaver?</th>
<th>Has stripes?</th>
<th>Has fur?</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>1</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>1</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>0</td>
<td>1</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>( x_5 )</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Some approaches to multi-output problems

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

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

- **Fully-cascaded chain**:

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

\( h_1, h_2, \ldots, h_L = \text{Base Estimators} \) (i.e. any single-output models)

<table>
<thead>
<tr>
<th>( x )</th>
<th>Raccoon?</th>
<th>Wolf?</th>
<th>Beaver?</th>
<th>Has stripes?</th>
<th>Has fur?</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>( x_5 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
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):

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

- **Fully-cascaded chain**:

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

\(h_1, h_2, ..., h_L = \text{Base Estimators} \) (i.e. any single-output models)
### 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 $\text{MSE} = \frac{1}{N} \sum_{j=1}^{N} (y_j - \hat{y}_j)^2$.
   - Example: multi-modal distribution $\Rightarrow$ standard models may be inappropriate.

   ![Graph showing multi-modal distribution](image)

   - 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 \( \text{MSE} = \frac{1}{N} \sum_{j=1}^{N} (y_j - \hat{y}_j)^2 \).
   
   - Example: multi-modal distribution \( \implies \) standard models may be inappropriate.

   ![Illustration of multi-modal distributions](image.png)

   - 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)


2. Layered Regressor Chains (LRC)
Uniform Cost Function (UCF) is an analogue of 0/1 loss for regression.

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

Goal = problem: optimize UCF.
Multi-Modal Ensembles of Regressor Chains (mmERC)

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

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

Goal = problem: optimize UCF.

<table>
<thead>
<tr>
<th></th>
<th>ERC</th>
<th>mmERC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BaseEstimator level</strong></td>
<td>single round of training</td>
<td>train on all dataset, choose portion of data giving best predictions, retrain on this part</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Ensemble level</strong></td>
<td>mean for all predictions</td>
<td>choose the biggest cluster of predictions, take mean for this cluster only</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**mmERC: results on a synthetic dataset**

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

(a) UCF results for the synthetic datasets.
Layered Regressor Chains (LRC)

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

Example of a single chain in ensemble:
$L = 4$ targets,
$K = 3$ layers,
$p = 2$ inter-layer connections
Layered Regressor Chains (LRC)

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

<table>
<thead>
<tr>
<th>Regressor</th>
<th>andro</th>
<th>atp1d</th>
<th>atp7d</th>
<th>jura</th>
<th>ores97</th>
<th>oresales</th>
<th>rf1</th>
<th>slump</th>
<th>Scpf</th>
<th>AvgRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>0.72</td>
<td>0.33</td>
<td>0.34</td>
<td>0.48</td>
<td>0.88</td>
<td><strong>0.94</strong></td>
<td>0.01</td>
<td>0.66</td>
<td>0.18</td>
<td>7.39</td>
</tr>
<tr>
<td>RF</td>
<td>0.71</td>
<td>0.23</td>
<td>0.35</td>
<td>0.41</td>
<td>0.80</td>
<td>0.94</td>
<td>0.02</td>
<td>0.54</td>
<td>0.16</td>
<td>5.94</td>
</tr>
<tr>
<td>IR (dt)</td>
<td>0.70</td>
<td>0.32</td>
<td>0.39</td>
<td>0.46</td>
<td>0.91</td>
<td><strong>0.78</strong></td>
<td>0.98</td>
<td>0.03</td>
<td>0.44</td>
<td>8.28</td>
</tr>
<tr>
<td>IR (rf)</td>
<td>0.57</td>
<td>0.20</td>
<td>0.33</td>
<td>0.41</td>
<td>0.78</td>
<td>0.98</td>
<td>0.03</td>
<td>0.44</td>
<td>0.17</td>
<td>4.61</td>
</tr>
<tr>
<td>IR (svr)</td>
<td>0.64</td>
<td>0.70</td>
<td>0.86</td>
<td>0.60</td>
<td>0.93</td>
<td>1.00</td>
<td>0.10</td>
<td>0.46</td>
<td>0.23</td>
<td>10.67</td>
</tr>
<tr>
<td>RC (dt)</td>
<td>0.69</td>
<td>0.32</td>
<td>0.40</td>
<td>0.49</td>
<td>0.91</td>
<td>0.97</td>
<td>0.02</td>
<td>0.43</td>
<td>0.17</td>
<td>6.94</td>
</tr>
<tr>
<td>RC (rf)</td>
<td>0.66</td>
<td>0.22</td>
<td>0.36</td>
<td><strong>0.37</strong></td>
<td>0.78</td>
<td>0.99</td>
<td>0.02</td>
<td>0.38</td>
<td>0.18</td>
<td>5.11</td>
</tr>
<tr>
<td>RC (svr)</td>
<td>0.96</td>
<td>0.48</td>
<td>0.71</td>
<td>0.55</td>
<td>0.98</td>
<td>0.98</td>
<td>0.82</td>
<td>0.67</td>
<td>0.22</td>
<td>11.83</td>
</tr>
<tr>
<td>LRC (dt)</td>
<td>0.57</td>
<td>0.22</td>
<td>0.30</td>
<td>0.43</td>
<td>0.86</td>
<td><strong>0.94</strong></td>
<td>0.03</td>
<td>0.50</td>
<td>0.17</td>
<td>5.11</td>
</tr>
<tr>
<td>LRC (rf)</td>
<td>0.55</td>
<td><strong>0.19</strong></td>
<td>0.25</td>
<td>0.37</td>
<td><strong>0.78</strong></td>
<td>0.96</td>
<td>0.01</td>
<td><strong>0.36</strong></td>
<td>0.18</td>
<td><strong>2.56</strong></td>
</tr>
<tr>
<td>LRC (svr)</td>
<td>0.89</td>
<td>0.74</td>
<td>0.81</td>
<td>0.61</td>
<td>0.95</td>
<td>1.00</td>
<td>0.30</td>
<td>0.46</td>
<td>0.23</td>
<td>11.78</td>
</tr>
<tr>
<td>LRC + mmERC (dt)</td>
<td><strong>0.47</strong></td>
<td>0.25</td>
<td><strong>0.25</strong></td>
<td>0.39</td>
<td>0.89</td>
<td>0.98</td>
<td>0.01</td>
<td>0.49</td>
<td>0.16</td>
<td>4.11</td>
</tr>
<tr>
<td>LRC + mmERC (rf)</td>
<td>0.72</td>
<td>0.23</td>
<td>0.43</td>
<td>0.40</td>
<td>0.87</td>
<td>0.99</td>
<td>0.02</td>
<td>0.41</td>
<td>0.20</td>
<td>7.00</td>
</tr>
<tr>
<td>LRC + mmERC (svr)</td>
<td>0.94</td>
<td>0.98</td>
<td>0.96</td>
<td>0.67</td>
<td>1.00</td>
<td>1.00</td>
<td>0.57</td>
<td>0.71</td>
<td>0.25</td>
<td>13.67</td>
</tr>
</tbody>
</table>
Imputation of missing values in genomic data
Single Nucleotide Polymorphisms (SNP)

Features:

- $M = 10^{5-10^7}$

Samples:

- $N = 10^{3-10^5}$

"Fat data" $X \in \{0, 1, 2\}^{N \times M}$
Single Nucleotide Polymorphisms (SNP)

$A = \text{prevalent variant (wild-type)}$, $a = \text{rare variant (mutant)}$

$AA = 0 \quad Aa = 1 \quad aa = 2$
Single Nucleotide Polymorphisms (SNP)

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

\[ AA = 0 \quad Aa = 1 \quad aa = 2 \]

Features: \( M = 10^5 - 10^7 \)  
Samples: \( N = 10^3 - 10^5 \)

“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

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)
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)
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)
Problem:

Missing values (up to ~ 30 – 40%)
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

Denoising autoencoders:

\[ \text{minimizing} \quad L(X, g(f(eX))) \]

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

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

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

Sparse Convolutional Denoising Autoencoders (Chen and Shi, 2019):
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
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
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
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
Autoencoders idea for non-NN multi-target methods: $f(\tilde{X}) \rightarrow X$
Autoencoders idea for non-NN multi-target methods: \( f(\tilde{X}) \rightarrow X \)

**Mushroom dataset**

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

<table>
<thead>
<tr>
<th>Mode</th>
<th>DT</th>
<th>RF</th>
<th>IR(dt)</th>
<th>CC(dt)</th>
<th>MLPc</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ imputed</td>
<td>0.598</td>
<td>0.781</td>
<td>0.753</td>
<td>0.764</td>
<td>0.789</td>
</tr>
<tr>
<td>- changed</td>
<td>0.000</td>
<td>0.071</td>
<td>0.078</td>
<td>0.0002</td>
<td>0.001</td>
</tr>
</tbody>
</table>

**Real SNP dataset (Blueberry)**

Slice of data (100 features, 1000 samples)

<table>
<thead>
<tr>
<th>Mode</th>
<th>DT</th>
<th>RF</th>
<th>IR(dt)</th>
<th>CC(dt)</th>
<th>MLPc</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ imputed</td>
<td>0.769</td>
<td>0.734</td>
<td>0.796</td>
<td>0.773</td>
<td>0.787</td>
</tr>
<tr>
<td>- changed</td>
<td>0.000</td>
<td>0.229</td>
<td>0.184</td>
<td>0.002</td>
<td>0.004</td>
</tr>
</tbody>
</table>
Thank you!