Parameters tuning boosts hyperSMURF predictions of rare deleterious non-coding genetic variants

A. Petrini, M. Schubach, M. Re, M. Frasca, M. Mesiti, G. Grossi, T. Castrignanò, P. Robinson, G. Valentini
The detection of *deleterious* genetic variants in human genome: a key problem in Personalized and Precision Medicine

*HyperSMURF*, an imbalance-aware ML method for detecting pathogenic variants in non-coding genome

Tuning its learning parameters can boost *hyperSMURF* predictions

An ongoing *HPC massively parallel implementation* of the method to automatically fit different genomic problems characterized by imbalanced big data
Issues:

- How to find pathogenic variants in the sea of background (neutral) genetic variation in human genome?
- A huge imbalance between deleterious (positive examples) and neutral (negative examples) variants (e.g. 1/36000 ratio in Mendelian diseases, Smedley et al., 2016)
- Which features should be used to train learning machines for the prediction of pathogenic variants?

Classical ML algorithms fail: 
they are biased toward the majority class

Parameters tuning boosts hyperSMURF predictions
Prediction of deleterious variants: state of the art

State-of-the-art ML methods for the prediction of deleterious variants

- CADD (Kircher, et al. 2014)
- GWAVA (Ritchie et al. 2014)
- DeepSEA (Zhou & Troyanskaya, 2015)
- FATHMM-MKL (Shibab et al. 2015)
- Eigen (Ionita-Laza et al. 2016)

Quite surprisingly none of the above methods (apart from GWAVA) use imbalance-aware learning strategies.
A Whole-Genome Analysis Framework for Effective Identification of Pathogenic Regulatory Variants in Mendelian Disease

Damian Smedley,1,2,15 Max Schubach,3,15 Julius O.B. Jacobsen,4,15 Sebastian Köhler,3 Tomasz Zemojtel,3,5 Malte Spielmann,3,6 Marten Jäger,3,7 Harry Hochheiser,8 Nicole L. Washington,9 Julie A. McMurry,10 Melissa A. Haendel,10 Christopher J. Mungall,9 Suzanna E. Lewis,9 Tudor Groza,11,12 Giorgio Valentini,13 and Peter N. Robinson3,6,7,14,16,*

• REMM (Regulatory Mendelian Mutation Score) a first version of hyperSMURF is part of the Genomiser tool for the identification of pathogenic regulatory variants in Mendelian disease (Smedley et al. AJHG, 2016)
HyperSMURF: a flexible tool for the prediction of deleterious variants

Parameters tuning boosts hyperSMURF predictions

HyperSMURF - Hyper-ensemble SMote Undersampled Random Forest: a novel multi-parametric version of the method able to fit different problems in the context of the prediction of deleterious variants
A ML approach to deleterious variants detection
Hyper-ensemble of Smote Undersampled Random Forests
\textit{(HyperSMURF)}

- Balancing training data through differential sampling:
  - Oversampling of the minority class
  - Partitioning and undersampling of the majority class

- Data coverage improvement and variance reduction through ensembling techniques

- Enhancing accuracy and diversity of the base learners through Hyper-ensembling

Parameters tuning boosts hyperSMURF predictions
HyperSMURF: Hyper-ensemble of SMote Undersampled Random Forests

- Data partitioning
- Oversampling of positives and undersampling of negatives
- RF learning
- hyperensemble combination

Parameters tuning boosts hyperSMURF predictions
SMOTE:
Synthetic Minority Oversampling Technique (Hall et al. 2002)

SMOTE allows to generate new synthetic samples (the red point) close to the “true” positives.
Pseudocode of the HyperSMURF algorithm

Input:
- $\mathcal{P}$: set of positive examples (Deleterious variants)
- $\mathcal{N}$: set of negative examples (Non-deleterious variants)
- $n$: number of partitions
- $k$: number of nearest neighbors for SMOTE oversampling
- $f$: oversampling factor

begin algorithm

01: (i) Initialization and partitioning of $\mathcal{N}$:
02: \[ n_{ex} := (f + 1)|\mathcal{P}| \]
03: \[ \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_n\} := \text{Do.partition}(\mathcal{N}, n) \]
04: \[ i := 1 \]
05: while ($i \leq n$) do

06: (ii) SMOTE oversampling:
07: \[ \mathcal{P}_S := \text{SMOTE}(\mathcal{P}, k, f) \]
08: (iii) Undersampling of non-deleterious variants:
09: \[ \mathcal{N}' := \text{Undersample}(\mathcal{N}_i, n_{ex}) \]
10: (iv) Training set assembly:
11: \[ \mathcal{T} := \mathcal{P} \cup \mathcal{P}_S \cup \mathcal{N}' \]
12: (v) Random Forest training:
13: \[ M_i := \text{RF}(\mathcal{T}) \]
14: \[ i := i + 1 \]
15: end while

end algorithm

Output:
- $M = \{M_1, M_2, \ldots, M_n\}$: a set of RF models

Output on a test variant $x$:
- $H_{score}(x) := \frac{1}{n} \sum_{i=1}^{n} P(x \text{ is positive } | M_i)$
Genomic experiments

1) *Mendelian diseases*: 406 SNV mutations manually curated (positive examples)
14M neutral variants (negatives)

2) *Complex diseases*: 2115 regulatory GWAS hits from the GWAS catalog (National Human Genome Research Institute)
1.4M neutral variants (negatives)

Parameters tuning boosts hyperSMURF predictions
Experimental set-up

Genomic attributes

1) Mendelian data: 26 genomic attributes downloaded from public data bases (UCSC, Stanford, NCBI and others):

- Conservation scores
- Transcriptional features
- Regulation features
- Overlapping CNVs
- GC content
- Epigenomic features

2) GWAS data: 1842 genomic attributes directly extracted from DNA sequence through deep convolutional networks (Zhou & Troyanskaya, 2015)

- DNAse features
- Transcription factor features
- Histone features
- Conservation scores

Parameters tuning boosts hyperSMURF predictions
HyperSMURF is very competitive with state-of-the-art methods:

AUPRC comparative results with state-of-the-art methods (Schubach et al. 2017)

Mendelian diseases

- hyperSMURF (0.427)
- Eigen–PC (0.044)
- CADD (0.093)
- GWAVA (0.156)
- Eigen (0.013)
- DeepSEA (0.052)

Complex diseases

- hyperSMURF (0.635)
- Eigen–PC (0.004)
- CADD (0.037)
- GWAVA (0.402)
- Eigen (0.004)
- DeepSEA (0.239)

10-fold “cytoband-aware” cross-validation: precision/recall curves

Parameters tuning boosts hyperSMURF predictions
Results

Compared precision, recall and F-score (complex diseases)

Parameters tuning boosts hyperSMURF predictions
HyperSMURF is effective with imbalanced data

AUPRC results of HyperSMURF and CADD at different imbalance levels

Parameters tuning boosts hyperSMURF predictions
Fitting different prediction problems requires proper tuning of the learning parameters.
HyperSMURF strength is evident with imbalanced data. Parameters tuning boosts hyperSMURF predictions.

Learning parameters strongly affect HyperSMURF performances

High impact of the hyperSMURF learning parameters on:

- Coverage of the data
- Balancing between deleterious and neutral variants
- Informativeness of the positive (deleterious) examples
- Effectiveness of the representation of the learning space
- Runtime and learning process
- Accuracy and diversity of the base learners

Results highly depend on the correct selection of the parameters for the specific problem under study.
An experimental study of the impact of learning parameters for the prediction of non-coding deleterious variants in Mendelian diseases

- Same data used in *Schubach et al.*, 2017:
  - 406 SNV mutations manually curated (positives)
  - 14M neutral variants (negatives)
  - 26 genomic features indicators of variant functionality (e.g. GC-content, conservation, histone modifications, DNase I accessibility, overlap with TFB sites and enhancers, overlapping CNVs)

- Hold-out setting for performance evaluation and internal cytoband-aware cross-validation (Smedley et al. 2016) for parameter tuning.

- 100 hyperSMURF models trained considering different combinations of $n$, $f$ and $u$ parameters

- Results obtained using a serial implementation and an arrays of jobs on the CINECA Marconi HPC cluster.
Cross-validation results on the training set across the 100 models

Best model: $n=300$, $f=1$, $u=10$

<table>
<thead>
<tr>
<th></th>
<th>AU PRC</th>
<th>AU ROC$_{50}$</th>
<th>AU ROC$_{100}$</th>
<th>AU ROC$_{500}$</th>
<th>AU ROC$_{1000}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>hyperSMURF default par.</td>
<td>0.3568</td>
<td>0.8600</td>
<td>0.9300</td>
<td>0.9091</td>
<td>0.8868</td>
</tr>
<tr>
<td>hyperSMURF best par.</td>
<td>0.4156</td>
<td>0.9220</td>
<td>0.9610</td>
<td>0.9407</td>
<td>0.9460</td>
</tr>
</tbody>
</table>

Results on an independent test set.
Default parameters: $n=100$, $f=2$, $u=3$ (Schubach et al., 2017)
• Results show that parameter tuning can boost hyperSMURF results

• Drawbacks: training and testing require from 2 to about 20 hours of computation for each model using Intel Xeon processors E5, 2.30 GHz and 128 GB RAM

• The situation can be even worse if we use e.g. thousands of features extracted from DNA with deep convolutional networks (Zhou and Troyanskaya, 2015)

• A serial implementation, even with a cluster and arrays of jobs is not enough
Par-hyperSMURF: HPC version of hyperSMURF through a mixed MPI/OpenMP parallel implementation

A very flexible HPC architecture by which we can apply hyperSMURF not only to the prediction of pathogenic variants, but more in general to genomic problems characterized by big data and very small a priori available knowledge.
Conclusions

- Data imbalance in genome-wide studies motivates *hyperSMURF*

- Drawbacks of *hyperSMURF*: many learning parameters that significantly affect prediction performance

- Parameter tuning can significantly boost *hyperSMURF* results

*Par-hyperSMURF* - ongoing HPC parallel version of *hyperSMURF*:

- Automatic tuning of learning parameters
- Application of *Par-hyperSMURF* to:
  - Whole genome ranking and detection of mutations in genetic diseases
  - Ranking and detection of cancer driver mutations
  - Personalized Medicine problems characterized by small a priori available knowledge and big data
References:


Thank you for your attention!

http://anacletolab.di.unimi.it