Epitope-mapping of Anti-HLA cl.I and II Antibodies in Highly Sensitized Patients by Machine Learning

March 2, 2018

1 Problem Statement

Organ donors can be matched by comparing their reactivity to certain HLA alleles. Given a reactivity per allele, which contains an amino acid sequence, we wish to find subsequences of these that contain the most information for different levels of reactivity. In essence, we wish to compress amino acid sequences as much as possible while still containing a significant amount of the information needed to classify each allele into the right reactivity category, i.e. find the amino acids that contain this information. The idea is to see if there are any recurring patterns associated with different reactivity levels.

2 Data and Method

Each data set represents one patient/individual contains a number of data points, each representing an HLA-allele and a reactivity value for the corresponding bead. In total, there are 98 data sets (28 from Aarhus, 20 from Helsinki, 25 from Oslo, and 25 from Uppsala), each with 97 data points. Each HLA-alleles is made up of a sequence of amino acids of which the first 274 amino acids constitute the data points used in this work. In addition to actual amino acids, their type (acidic, basic, etc.) is also included. Thus, each data point has 548 features. Reactivity values are classified using the following thresholds (can be configured when running the program):

- Level 0: below 1000
- Level 1: 1000-2999
- Level 2: 3000-5999
- Level 3: 6000 and above

In order to extract from the amino acid sequences the smallest possible number of amino acids that contain the information needed to correctly classify each allele, a machine learning model is trained to classify the alleles given the entire 274-length sequences. Many common machine learning models such as neural networks are black box models, i.e. they provide no details about *how* they make their decisions. In order to extract the most important amino acids, a more transparent machine learning was needed, and so the number of feasible algorithms was limited. Tree-based algorithms seemed like the obvious choice because of their transparency, and so a selection of different types of model were used: Adaptive Boosting (AdaBoost), Extremely Randomised Trees (Extra-Trees), and Random Forests. Each model is comprised of an ensemble of weaker learners and base their final decision on either the majority or a weighted average of the weak learners' decisions. While Random Forests and Extra Trees always use decision trees, AdaBoost can – in theory – use any type of learner, but they use most commonly use decision trees, and this has also been the case in this project. Also, simple Decision Trees were experimented with.

Data was split into a training data set (50 samples) and a test data set (47 samples).

Once trained, the most valuable features (amino acid positions) in the sequences were extracted. This was a trade-off between quality of classification versus explainability of results. Keeping 90 percent of the variance seemed like a sweet-spot. This allowed models to significantly reduce the number of amino acids needed to classify while still performing well in terms of classification.

3 Results

Of the four different algorithms, AdaBoost seems by far the best at finding a small number of important features compared to the three other models, although its precision is not nearly as high as that of the other models; both Extra Trees and Decision Trees were able to score a precision of 1.0 (100%), but the Extra Trees yielded very large sequences. The Random Forests yield good precision but also at the cost of large sequences. AdaBoost and Decision Trees seem like the model of choice with one doing very well in terms of finding a small number of important features while still having a respectable precision, and Decision Trees yield very good precision but does perform as well in terms of finding a small number of important features.

Below is a table showing the some results found with each of the different models. It is important to note that, given the number possible of parameter configurations, there is likely to be a better configuration.

Model	Estimators	Test Precision	Features
AdaBoost	1	0.804	**1.00**
AdaBoost	3	0.823	2.60
AdaBoost	6	0.815	3.66
AdaBoost	10	0.822	4.36
AdaBoost	25	0.846	5.11
AdaBoost	50	0.835	5.33
Extra Trees	1	**1.00**	11.03
Extra Trees	3	**1.00**	20.80
Extra Trees	6	**1.00**	27.16
Extra Trees	10	**1.00**	30.85
Random Forest	1	0.897	7.91
Random Forest	3	0.942	16.09
Random Forest	6	0.970	22.44
Random Forest	10	0.987	28.65
Decision Tree	-	**1.0**	7.30

For example, the second row shows that an AdaBoost classifier with 3 estimators was able to classify with 82% precision using 2.75 features (on average) across the 98 data sets.

In general, execution is very fast and should run with no problem on any modern machine.

4 Usage

The program is a Python script and can be run in the terminal or command prompt using

```
python3 epitopes.py —–data <path/to/datafiles>
```

with the following optional parameters:

- –**hla** (path to file containing HLA alleles (assumed by default to be in the working directory with name 'LuminexDefinedClassIalleles.fasta')
- –**model** (which model to use – default is AdaBoost with 50 learners (this is the default in the machine learning library used))
- –**numest** (the number of weak learners of the model (Decision Trees ignore this parameter))
- –**threshold** (which thresholds to use for reactivity classification – default configuration is the one described in 2)

Output Program output shows which features are the most important ones and prints, for each record in a data set, the most important amino acids for that specific record, ordered by importance. Output printed in the terminal or command prompt and is written to several text-files (one for each data set). These output files are saved to a 'Output' directory, created in the working directory.

Since each of the aforementioned machine learning models uses an ensemble of different learners, it is difficult to visiualise the patterns found by the models.

Thus, a simple decision tree learner can be trained to learn the patterns of the compressed data, making it easier to visualise. Trees are saved to a 'Trees' directory, created in the working directory.

Figure 1 show an example of a (very simple) tree. Decision Trees are based on following a path from the root to a leaf node based on inequalities at each split. In this example, we look at the amino acid at index 177 (0-indexed) and

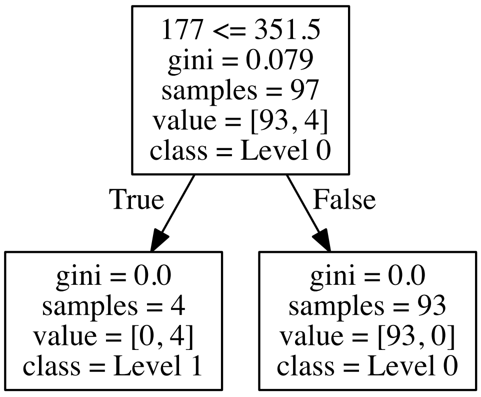

Figure 1: Example tree illustrating the patterns found.

ask if its value is below 351.5. If true, the reactivity value is at level 1 (see Section 2), otherwise it is level 0 – there is a total of 97 data points divided into the two classes with 93 at level 0 and 4 at level 1. The value 351.5 might not make much sense at first glance – all the methods work with numerical values, so each amino acid is given a value according to the following:

- A = 0, G = 1, I = 2, L = 3, P = 4, V = 5 (aliphatic)
- F = 100, W = 101, Y = 102 (aromatic)
- D = 200, E = 201 (acidic)
- R = 300, H = 301, K = 302 (basic)
- S = 400, T = 401 (hydroxylic)
- C = 500, M = 501 (sulfur-containing)
- N = 600, Q = 601 (amidic)