Predicting prokaryotic incubation times from genomic features

Maeva Fincker - mfincker@stanford.edu

Final report

Introduction

We have barely scratched the surface when it comes to microbial diversity. For every microorganism we can culture and study in lab, there exist at least another 50 in the environment that are labelled “uncultivable”. Although most of these uncultivable strains resist growing under laboratory conditions, recent advances in DNA sequencing has given us access to their genome. One of the main questions that bioinformatics in the field of microbiology is trying to answer is the following: how can we infer function from genomic information without access to experimental validation?[1] **This project focuses on the prediction of microbial growth rate (how fast can a microorganism divide) from genomic information**. More specifically, counts of specific genomic features encoding microbial functions are extracted from annotated microbial genomes for which the incubation time in known. These features are then used as input variables for thee different classifiers (softmax classification, SVM and random forest) whose role is to find the most probable label among 6 classes (representing different incubation times).

Previous work

Literature review did not reveal many example of using machine learning tools to predict physiological characteristics at the level of an microorganism from genomic features. Lauro et al. used self-organizing map to cluster microorganisms based on genomic features and relate the cluster to types of metabolism.[2] Roller et al. showed that the number of 16S rRNA operon correlates with growth rate and used it for supervised regression to predict bacterial reproductive strategy. Their results, that growth rate use be dictated by metabolic functions, are the basis for this project. [3]

Datasets & Feature selection

The growth rate of a microorganism describes how long it takes for a cell to divide. When a new organism is isolated, microbiologists report growth rates in various ways, which make direct comparisons difficult, if they even measure such a physiological trait at all. Since mining scientific literature for growth rates values could be a machine learning project on its own that I did not wish to tackle, I decided to use **incubation time** as a proxy for growth rate. Incubation times are reported by strain collections and represent how long approximately one has to wait to observe full growth of a microbe. The BacDive database [4] contains the incubation times of 2312 microorganisms in 6 different bins of time: 1-2 days, 2-3 days, 3-7 days, 8-14 days, 10-14 days and > 14 days. Out of these 2312 organisms, 783 have a complete genome (or ordered scaffold genome assembly) available on NCBI [5] but only 596 of these genomes have been properly annotated. **Therefore, my dataset is composed of 596 genomes and their**

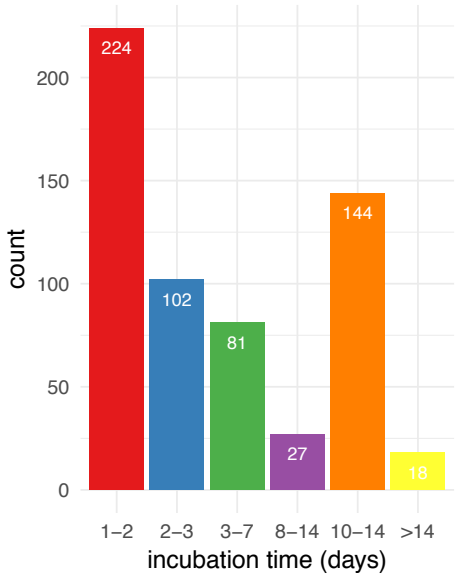

Fig. 1: histogram of the labels in the dataset

associated incubation time. Figure 1 shows the histogram of the labels of my 596 examples. Classes are very unbalanced, with an over representation of class 1-2 and 10-14 days.

Given the hypothesis that incubation time is dictated by metabolic capabilities and functions, I focused on extracting metabolically-relevant features from the genomes: counts of proteins belonging to each pFam families, length of the genome and number of 16S rRNA operons. The Pfam families are clusters of homolog proteins sharing a similar function. Extraction pFam counts yielded a total of 14618 features, which was too large a number of features when I only had access to 596 genomes. Keeping only Pfam features that appear in at least 3 genomes lowered the total number of features to 7549.

Proteins from more than one pFam cluster can be responsible for the same metabolic function, making my dataset highly redundant. In order to further decrease the number of feature, I used a feature selection algorithm called Fast Correlation-Based Filter (fcbf) [6]. This algorithm uses symmetrical uncertainty (SU) of two variables as a measure of their correlation. Given two variables X and Y, SU is defined as two times the ratio of information gain of X given Y, over the sum of their entropy. It first computes the SU between each variable and the label over the dataset. Keeping only features whose SU with respect to the labels are above a threshold d, the algorithm then consider a pair of features and calculates their SU with respect to each other. If the SU between these two features is higher than the SU between either of them and the label (meaning they are more correlated to each other than to the label), the feature with the lowest SU with respect to the label is removed from the dataset. The algorithm loops over pairs of features until all remaining features have a higher SU with respect to the label than to each other. Using a threshold d of 0 (which means keeping all non redundant features that correlates with the labels), the Fast Correlation-Based Filter selected 120 features.

Classifier building and selection were therefore carried out on two datasets of 596 examples each: the full dataset contained 7549 features and the filtered "fcbf dataset"contained 120 features. Both datasets were scaled (s.d. 1), centered (mean 0) and BoxCox transformed to correct for skewness for downstream applications. The datasets were divided between training set (537 examples, 90%) and a test set (59 examples, 10%).

Methods

I built and compared 3 different classifiers: **softmax classification with L1-regularization, one vs. one SVM with RBF kernel, and random forest**.

Softmax classification:

In softmax regression, the hypothesis (for a problem with K classes) takes the following form:

$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \vdots \\ P(y=K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)} \begin{bmatrix} \exp(\theta^{(1)\top} x) \\ \exp(\theta^{(2)\top} x) \\ \vdots \\ \exp(\theta^{(K)\top} x) \end{bmatrix}$$

And the likelihood function to maximize is: $\ell(\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta) = \sum_{i=1}^{m} \log \prod_{l=1}^{k} \left(\frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \right)^{1\{y^{(i)}=l\}}$

There is no close form for the maximum likelihood estimates and I used batch gradient descent to find them. Since I had a large number of features, I also tried adding a L1 regularization term to the likelihood function such that the function to maximize becomes: $l(\theta)^* = l(\theta) + C||\theta||_1$

The L1 regularization term brings down to zero the weights of unimportant variables, thereby effectively selecting features of importance. The model was fitted to both the full and the filtered training set and the best value for C for each dataset was selected via 10-fold cross-validation one the training set. Finally, the models were evaluated on the test set.

SVM classification:

SVMs are linear classifiers that use the hinge loss. They enable the use of kernels to separate data in a high-dimensional space that is not necessarily linearly separable in its own space.

Using the RBF kernel, the loss function to minimize is:

$$J_\lambda(\alpha) = \frac{1}{m}\sum_{i=1}^{m}\left[1 - y^{(i)}K^{(i)^T}\alpha\right]_+ + \frac{\lambda}{2}\alpha^T K\alpha$$

with the matrix $K = [K^{(1)} \ \cdots \ K^{(m)}]$ being defined as $K_{ij} = K(x^{(i)}, x^{(j)})$ and $K(x,z) = \exp\left(-\ \gamma\ \|x-z\|_2^2\right)$.

The 2 model hyperparameters to optimize are therefore λ and γ.

SVMs are inherently 2-classes classifiers. In order to use them for J-class problem, one can either train J one vs. all SVM or J(J-1) one vs. one SVMs. Since my classes were extremely unbalanced, I decided to use one vs. one SVMs as it would decrease the impact of having very large and very small classes. Furthermore, I weighted each example depending on their label: examples from overrepresented classes had a small weight whereas examples from underrepresented classes had a large one (weights: “1-2" = 0.5, “2-3" = 1, “3-7" = 1.2, “8-14" = 3.4, “10-14" = 0.7, ">14" = 5.7). As for softmax regression, the model was fitted using both the full and the filtered dataset. The best values for lambda and gamma were selected with 10-fold cross-validation on the training set and the models with the best parameters were retrained on the full data-set before being evaluated on the test set.

Random forest

Random forest classifiers build a collection of *ntree* decision trees, each tree being computed over a subset of *nex* samples, sampled with replacement from the training set. For each tree, at each decision split, a random subset of *mtry* features is selected and the decision split only decided over this subset. At each split, the best feature and decision threshold to split upon is the feature that minimizes the Gini index.

The Gini index I_G is measure how often an example would be mislabeled under a certain probability (with $f_i = p(y = i)$).

$$I_G(f) = \sum_{i=1}^{J} f_i(1 - f_i)$$

I trained random forest classifiers on both the full and filtered training set and tried to optimize the number *mtry* of features selected at each split. I used the randomForest package in R to fit random forest models.

Because the classes in my dataset were very unbalanced, accuracy was not a good measure of learning. Instead, I used a definition of precision and recall generalized to multi-class to optimize the hyperparameters:

$$P = \frac{\sum_j \frac{\text{number of properly classified examples from class j}}{\text{number of examples classified in class j}}}{J} \qquad R = \frac{\sum_j \frac{\text{number of properly classified examples from class j}}{\text{number of examples that are truly in class j}}}{J}$$

Results

Softmax regression

The softmax regression model was fitted using batch gradient descent with batches of 100 examples for 100 iteration with a learning parameter of 0.05. Better precision and recall were obtained with the filtered dataset, indicating probably overfitting of the model to unimportant features in the full dataset.

The L1 regularization hyper parameter was optimized using 10-fold cross-validation (CV) on the training set. Mean precision and recall of the CV are presented in figure 2. Best performances were achieved with a L1 parameter of 0.6 and evaluation on the filtered test set showed a precision of 0.43 and a recall of 0.44.

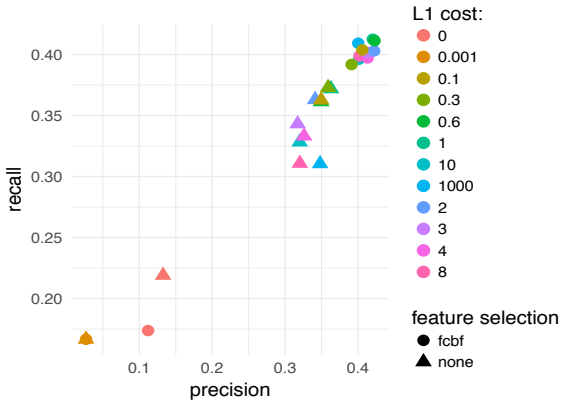

Fig. 2: Softmax classification
10-fold CV of the hyper parameter
over the training set

SVM

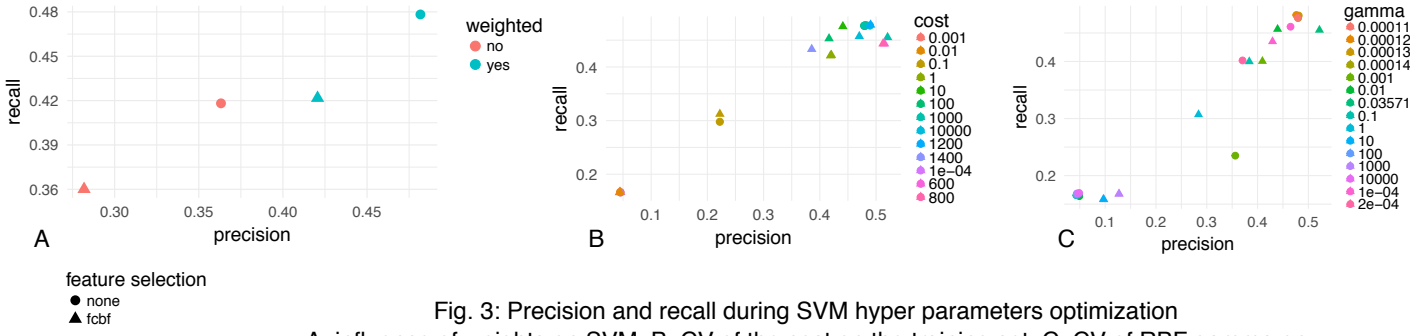

Fig. 3: Precision and recall during SVM hyper parameters optimization
A: influence of weights on SVM, B: CV of the cost on the training set, C: CV of RBF gamma on the training set (values in B and C are mean precision and recall during CV)

A first attempt at implementing one vs. all SVMs for this classification problem did not yield any useful result because of the imbalance of classes. Therefore, a one vs. one SVM classification approach was employed. All model fitting were carried out using batch gradient descent (batches of 100 examples, annealed learning rate of 1/sqrt(t) for 40*m iteration). I first tested the influence of weights on the model using gamma = 1/num features, and cost = 1 (fig. 3). Using weights improved precision and recall by 6-10% on both the filtered and full dataset. The best cost value for the full dataset was 100 and 1200 for the filtered dataset. Optimization of RBF gamma yielded the following values: 0.035 for the filtered dataset, and 0.00013 for the full dataset. Evaluation on the test set showed that the best model had a precision and recall of 0.48 and 0.48 for the full dataset, and of 0.52 and 0.46 for the filtered dataset.

Random Forest

I trained random forest models with 2500 trees and variable number of features subsampled at each split. Optimization of the number of features for split decision yielded different results for the full and filtered dataset: 20 features worked best for the filtered dataset, whereas 5 features were better for the full dataset. The best random forest model yielded a precision of 0.48 and recall of 0.47 with the filtered dataset (fig. 4).

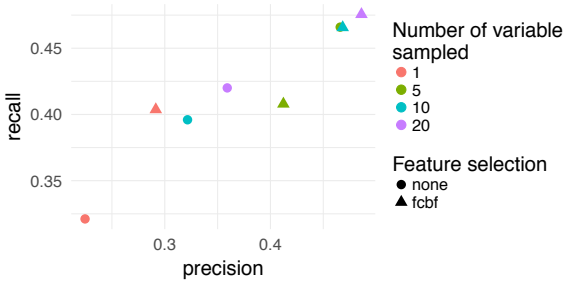

Fig. 4: Random forest model - Precision and recall during optimization of the number of subsampled features

Model comparison and PCA

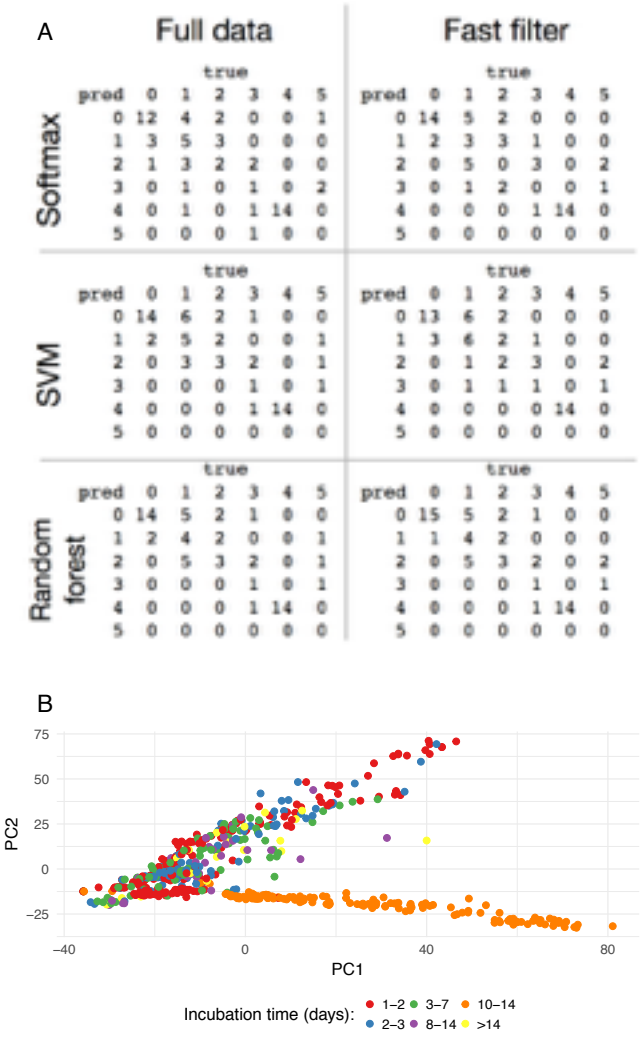

Fig. 5: Confusion matrices from the best models (A), plot of the first two components of PCA over the full dataset (B)

All three models performed better with the filtered dataset but did not have vastly different classification powers, as the comparison of confusion matrix shows in figure X. All classifiers were able to classify fast and slow growing organisms (class 0: 1-2 and class 4: 10-14 days) relatively well but were not able to resolve medium growth-rate (classes 2, 3) or extremely slow organisms (class 5).

PCA of the dataset shows a clear chasm between slow an fast growing organisms (class 0 and class 4). However, Fast growing organisms overlap with medium-fast organisms on the plot, which might explain the difficulty for the classifiers to properly classify them.

Conclusion

This project focused on predicting microbial incubation times from genomic features. All classifiers trained yielded precision and recall values in the 0.4 - 0.5 range, indicating that no model was vastly superior. The classifiers were able to classify well fast and slow growing organisms but not the organisms whose incubation are in between (or extremely long). This lack of predictive power can be explained by the imbalance of classes, the small number of examples compared the number of features or, more likely, a lack of knowledge of the optimum growth condition for the organisms. Under optimal growth condition, it is possible that the organisms in class 2 and 3 would actually grow faster and would fall in class 0.

References:

1. Libbrecht MW, Noble WS. 2015. Machine learning applications in genetics and genomics. Nature Reviews Genetics. 16(6):321–32

2. Lauro FM, McDougald D, Thomas T, Williams TJ, Egan S, et al. 2009. The genomic basis of trophic strategy in marine bacteria. Proc. Natl. Acad. Sci. U.S.A. 106(37):15527–33

3. Roller BRK, Stoddard SF, Schmidt TM. Exploiting rRNA operon copy number to investigate bacterial reproductive strategies. Nature Microbiology. 1:16160EP

4. Söhngen C, Bunk B, Podstawka A, Gleim D. 2013. BacDive—the Bacterial Diversity Metadatabase. Nucleic Acids Research

5. https://www.ncbi.nlm.nih.gov

6. Q. Song, J. Ni and G. Wang. A Fast Clustering-Based Feature Subset Selection Algorithm for High-Dimensional Data. IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 1, pp. 1-14, Jan. 2013.