Semi-supervised learning for phenotypic profiling of high-content screens (DRAFT)

Roger Bermudez-Chacon
Supervisor: Peter Horvath

ETH Zurich

July 31, 2013

Abstract

Semi-supervised machine learning techniques are particularly useful in experiments where data annotation and classification is time- and resource-consuming or error-prone. In biological experiments this is often the case. Here, we apply a graph-based machine learning method to classify cells in different stages of infection with the Semliki Forest Virus (SFV), which features have been extracted from image analysis of fluorescence microscopy results, obtained in turn from a genome-wide high-content screening experiment. The aim of this project is to investigate whether and to which extent intelligent control experiment design combined with semi-supervised learning can reach the accuracy of a human annotator and/or in certain cases substitute it.

Introduction

Recent advancements in high-throughput microscopy and data analysis made possible to perform large scale biological experiments and automatically evaluate them. For the detection of sub-cellular changes caused by different perturbations in the cell (RNAi or drugs), often supervised machine learning (SML) is used. Reliable training of an SML method, however, requires significant effort from a field expert.

As an alternative, semi-supervised machine learning (SSL) methods make use of information intrinsically found in the entire data, both annotated and unannotated, thus allowing to make use of a larger amount of information by exploiting, alongside with the annotated data, the relative distribution of unannotated data on the feature space [1]. This paradigm, under a few assumptions[1], has proven valuable in exploring and classifying biological data in fields as diverse as drug-protein interactions [2], gene expression [3], and medical diagnosis [4].

use footnote with assumptions or just reference to publication?

Materials and Methods

High-content screening

A human genome-wide siRNA library was used to produce human cell cultures with knocked-out genes, stored in a collection of 55 384-well plates spanning the entire genome. These cell cultures were exposed to a genetically engineered fluorescent SFV strand, and the corresponding green fluorescent protein production on all of the cultures was tracked over time. K repetitions of this experiment were carried out.

expand this introduction?

The protein expression was stopped at different time points after culture infection with SFV, for all siRNA-mediated phenotypes, and microscopic pictures of the sample were obtained under a light microscope. Samples with no exposure to SFV were also analyzed as a control experiment, in the exact same manner as for the infected samples.

confirm this is correct/well-written, explain this in more detail?

ask how many

Image adquisition and analysis

For every sample at each infection stage, 9 tiled images were captured via a light microscope, by composing the green fluorescent signal of the produced protein, and a blue-colored image of the nuclei. All images were subsequently processed with an automatic random forest-based segmentation tool to identify individual cells from the images.

With the mapping between microscopic pictures and individual image segments representing cells, features for each cell were extracted with CellProfiler[5]. A total of 93 features were retrieved and used in this experiment, corresponding to color intensity, area, shape, and texture descriptors. (For the complete list of features extracted, see Appendix A)

according to G. Balistreri. Confirm. Expand on this?

Unannotated data

The above process was performed automatically on the $55 \cdot 384 \cdot 9 \cdot K$ images retrieved from the siRNA-mediated phenotypes. All 93 features[2] were extracted as floating point values, and stored in text files, one line per cell. Unlabeled information for M cells was collected by this process.

replace with actual number

figure out this number

Annotated data

From the genome-wide information, a small subset of the data was manually annotated by an expert on SFV infection, by visually identifying cell phenotypes directly from the segmented microscopic images and cross-checking with the time annotation on the respective source plate, and classifying them into the different stages of infection. This manual point-and-click process yielded 3098 annotated cells.[3]

software reference for this?

Control data

As control cultures, 8 plates treated with a control siRNA that has no effect on the expressed phenotype of the cells were used. Control plates not exposed to SFV, as well as infected plates analyzed at 4, 5, 6, and 7 hours after infection, were used to retrieve control information during the time course. The exact same image analysis and feature extraction procedure described for the unannotated cultures was performed on these plates.

Semi-supervised learning implementation

A graph-based label propagation (label spreading[6]) approach was followed. In this kind of approach, an undirected graph is built using the data points (cells) as vertices, and edges are created for all pairs of vertices that satisify a neighboring condition, with weights proportional to some measurement of association between the pair of vertices. This degree of association is often assumed as related to the distance between the data points in the n-dimensional feature space, in a linear, exponential, or Gaussian fashion, among others, and can be either limited in space (k-nearest neighbors, cutoff distance, ...), or consist of a complete graph that considers all possible pairwise relationships.

In the original formulation, labels are associated to the vertices corresponding to annotated data, and neutral labels likewise to the unannotated data; then, in an iterative fashion, the labeled vertices propagate along the edges to their neighbors' labels, and compete against the propagation of other labeled vertices with a strength proportional to their relatedness (edge weight W).

In the present implementation, prior knowledge of the nature of the data was incorporated as an additional level of *soft labeling*, to exploit the fact that, for data points taken from the experimental control, the cells (vertices) can be tracked back to their experimental conditions and time course, which have a direct influence on what specific phenotypes (labels) are more likely to occur.

Development and runtime environment

To read and analyze the data, a script was coded in Python 2.7.4. Extensive use of the open source libraries `numpy` and `scipy` were used for matrix and numerical manipulation, as well as `matplotlib` for data visualization.

[1] Smoothness, Cluster, and Manifold assumptions, see[1] p. 4-6

[2] In total, 95 features were extracted from the images. Spatial coordinates, however, were regarded as of little, if any value for data analysis in this work.

[3] A small number of imaging artifacts were also identified manually. However, accounting for such information was out of the scope of this work.

Many options for this script are customizable via command line parameters. Appendix B includes a description of all the possible parameters and a quick user guide.

Feature selection

The values for all the features from the annotated cells were analyzed with Weka[7]. The InfoGain attribute evaluator was used to determine the information gain ratio for each feature, and features with a score of above F were chosen as the selected dimensions to represent the data for further analysis.

run weka again

Due to the heterogeneity on the range of values between the selected features, spanning several orders of magnitude, normalization of the data was required. The relative InfoGain score among the group of selected features was used as a weight for feature normalization, so that when pairwise distance calculations were performed over the reduced g-dimensional space, each feature would reflect its relative importance.

how many?

The selected features were: feature1, feature2, feature3, feature4.

how many features selected?

make table

Feature	Relative score
feature1	0.32
feature2	0.33
feature3	0.11

Data pre-processing

Text files in both `arff` and `txt` formats containing feature information for labeled, soft-labeled, and unlabeled data were read into a feature matrix M (n data points $\times m$ features).

fill this table with actual values

The `arff` files containing *labeled data* were read and loaded into the feature matrix, by filtering the read fields to include only the features obtained in the feature selection phase. The possible classes or labels are loaded from the `arff` file by parsing the line starting with `@attribute class` (this prefix can be overriden via the `--label-line-prefix` parameter in the command line). As a parameter to the program, a list of ignored labels can be also passed with the option `--ignored-labels`. Data points annotated with any of these label identifiers will be left out of the feature matrix. No further sampling was performed over the labeled data, i.e. all remaining (non-ignored) data points were kept.

The `txt` files containing *soft-labeled data* were read a similar way, except there was no need for parsing any formatting of the files. Each line in these `txt` files corresponds to a cell, and contains the values for the features, space-separated, in the same ordering as the labeled files. To assign actual soft labels, the relative file location in the file system was used as follows: the user indicates a root directory with all the soft-labeled data, and the files are expected in differents directories within, which are internally mapped (via a python dictionary) to the actual labels.

The `txt` files containing *unlabeled data* were read exactly as described above for the soft-labeled data. A default neutral label was assigned to all entries read from this files.

Due to the massive amount of information, sampling parameters over the soft-labeled and unlabeled data were implemented. The command-line parameter `--num-samples N` controls how many data points to use from both soft-labeled and unlabeled data together ($N/2$ each). An additional flag parameter `--class-sampling` indicates that the script must sample the soft-labeled data uniformly over classes, to avoid sampling bias due to large differences between the number of data points on each class.

As an outcome of this pre-processing step, the feature matrix containing the values of the selected features for the labeled, soft-labeled, and unlabeled cells (after sampling, when specified) was returned, along with the initial label matrix.

explain how to construct this

Data normalization

Pellentesque interdum sapien sed nulla. Proin tincidunt. Aliquam volutpat est vel massa. Sed dolor lacus, imperdiet non, ornare non, commodo eu, neque. Integer pretium semper justo. Proin risus. Nullam id quam. Nam neque. Duis vitae wisi ullamcorper diam congue ultricies. Quisque ligula. Mauris vehicula.

Graph construction

The graph was internally represented by its weight matrix $\mathbf{W}$ ($\mathbf{W}_{ij} > 0$ if there exists an edge between the vertices x_i and x_j, zero otherwise), plus a $n \times m$ (n cells, m possible labels or classes) label matrix Y , with valid values

ranging from 0 to 1. A value of $Y_{i,j} = 1$ represents complete confidence that the i-th cell in the data set, belongs to the j-th phenotypic class of cells. Likewise, a value of 0 indicates absolute disbelief that a cell corresponds to a class, and values of 0.5 indicate complete uncertainty about class membership.

Label propagation

Phasellus fringilla, metus id feugiat consectetuer, lacus wisi ultrices tellus, quis lobortis nibh lorem quis tortor. Donec egestas ornare nulla. Mauris mi tellus, porta faucibus, dictum vel, nonummy in, est. Aliquam erat volutpat. In tellus magna, porttitor lacinia, molestie vitae, pellentesque eu, justo. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Sed orci nibh, scelerisque sit amet, suscipit sed, placerat vel, diam. Vestibulum nonummy vulputate orci. Donec et velit ac arcu interdum semper. Morbi pede orci, cursus ac, elementum non, vehicula ut, lacus. Cras volutpat. Nam vel wisi quis libero venenatis placerat. Aenean sed odio. Quisque posuere purus ac orci. Vivamus odio. Vivamus varius, nulla sit amet semper viverra, odio mauris consequat lacus, at vestibulum neque arcu eu tortor. Donec iaculis tincidunt tellus. Aliquam erat volutpat. Curabitur magna lorem, dignissim volutpat, viverra et, adipiscing nec, dolor. Praesent lacus mauris, dapibus vitae, sollicitudin sit amet, nonummy eget, ligula.

Results

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

Discussion

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

Conclusions

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Bibliography

[1] O. Chapelle, B. Schölkopf, A. Zien, *et al.*, *Semi-supervised learning*, vol. 2. MIT press Cambridge, 2006.

[2] Z. Xia, L.-Y. Wu, X. Zhou, and S. Wong, "Semi-supervised drug-protein interaction prediction from heterogeneous biological spaces," *BMC Systems Biology*, vol. 4, no. Suppl 2, pp. 1–16, 2010.

[3] I. Costa, R. Krause, L. Opitz, and A. Schliep, "Semi-supervised learning for the identification of syn-expressed genes from fused microarray and in situ image data," *BMC Bioinformatics*, vol. 8, no. Suppl 10, p. S3, 2007.

[4] E. Bair and R. Tibshirani, "Semi-supervised methods to predict patient survival from gene expression data," *PLoS biology*, vol. 2, no. 4, p. e108, 2004.

[5] A. E. Carpenter, T. R. Jones, M. R. Lamprecht, C. Clarke, I. H. Kang, O. Friman, D. A. Guertin, J. H. Chang, R. A. Lindquist, J. Moffat, *et al.*, "Cellprofiler: image analysis software for identifying and quantifying cell phenotypes," *Genome biology*, vol. 7, no. 10, p. R100, 2006.

[6] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in neural information processing systems*, vol. 16, no. 16, pp. 321–328, 2004.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[8] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2001.

[9] R. M. Haralick, K. Shanmugam, and I. H. Dinstein, "Textural features for image classification," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 6, pp. 610–621, 1973.

Appendix A

Features analyzed

[Table with cell/nuclei intensity, shape and Haralick[9] texture features...] format this

1:2	nuclei location
3	green mean intensity nuclei
4	green std intensity nucleii
5	green mean intensity cells
6	green std intensity cells
7	blue mean intensity nuclei
8	blue std intensity nuclei
9	blue mean intensity cells
10	blue std intensity cells
11:20	AreaShape
21:35	nuclei texture green
36:50	nuclei texture blue
51:65	cell texture green
66:80	nuclei texture green
81:95	cell texture green

Appendix B

Script parameters and help

up to date?

```
$ python hcs.py -h
usage: hcs.py [-h] [-t] [-l LABELED_FILE [LABELED_FILE ...]]
              [-u UNLABELED_FILE [UNLABELED_FILE ...]] [-s SOFT_LABELED_PATH]
              [-L NUM_LABELED_POINTS] [-n NUM_SAMPLES] [-c]
              [--max-iterations MAX_ITERATIONS] [-d WIDTH]
              [-nf {exp,knn3,knn4,knn5,knn6}]
              [-dm {euclidean,cityblock,cosine,sqeuclidean,hamming,chebyshev}]
              [-f FEATURE_INDEX [FEATURE_INDEX ...]] [-q]

Label propagation

optional arguments:
  -h, --help            show this help message and exit
  -t, --test            Performs a test run.
  -l LABELED_FILE [LABELED_FILE ...], --labeled LABELED_FILE [LABELED_FILE ...]
                        Labeled files.
  -u UNLABELED_FILE [UNLABELED_FILE ...], --unlabeled UNLABELED_FILE [UNLABELED_FILE ...]
                        Unlabeled files.
  -s SOFT_LABELED_PATH, --soft-labeled SOFT_LABELED_PATH
                        Path to soft labeled files. One directory per label
                        expected.
  -L NUM_LABELED_POINTS, --num-labeled NUM_LABELED_POINTS
                        Number of labeled data points to use. Default: use all
                        available
  -n NUM_SAMPLES, --num-samples NUM_SAMPLES
                        Number of samples. Default: 3000
  -c, --class-sampling  Distributes the number of samples given by
                        [NUM_SAMPLES] uniformly over all soft classes
  --max-iterations MAX_ITERATIONS
                        Maximum number of iterations. Default: 1000
  -d WIDTH, --display-columns WIDTH
                        Max width used for matrix display on console
  -nf {exp,knn3,knn4,knn5,knn6}, --neighborhood-function {exp,knn3,knn4,knn5,knn6}
                        Neighborhood function to use. Default: exp
  -dm {...}, --distance-metric {euclidean,cityblock,cosine,sqeuclidean,hamming,chebyshev}
                        Metric for calculating pairwise distances. Default:
                        euclidean
  -f FEATURE_INDEX [FEATURE_INDEX ...], --features FEATURE_INDEX [FEATURE_INDEX ...]
                        Selected feature indices (as given by the labeled
                        data).
  -q, --quiet           Displays progress and messages.
```