Good Cell, Bad Cell: Classification of Segmented Images for Suitable Quantification and Analysis

Derek Macklin, Haisam Islam, Jonathan Lu

December 14, 2012

***Abstract*—While open-source tools exist to automatically segment and track cells in time-lapse microscopy experiments, the resulting output must be inspected to detect and remove spurious artifacts. In addition, data not exhibiting proper experimental controls must be discarded before performing downstream analysis. Currently, the data set is manually examined to accomplish these tasks. Here we present machine learning approaches for classifying the suitability of data traces for downstream processing and analysis. We integrate the most successful approaches into the experimental workflow.**

I. Introduction

Time-lapse live-cell microscopy is a powerful technique to study the temporal dynamics of signaling pathways in single cells. Using fluorescently-labeled proteins, researchers can perform experiments to observe the interactions of signaling components to understand the underlying mechanisms of action. Analyzing data from these experiments requires image processing to quantify the spatio-temporal dynamics of labeled proteins [1]. Recently, sophisticated open-source software packages have been released to help automate biological image processing tasks such as cell segmentation [2] and tracking [3]. While these packages can be combined to create an image processing pipeline, the data traces that are produced must be inspected to determine if they are appropriate for inclusion in downstream analysis. Figure 1 shows an overview of the workflow.

Data traces produced by the image processing pipeline are excluded from further scientific analysis for two reasons. One reason is that the segmentation algorithm occasionally fails to uniquely identify and separate adjacent cells, thereby introducing large artifacts in some of the generated traces. Another reason is that some traces may not exhibit features associated with proper experimental controls. For example, some cells may not contain any fluorescently-labeled protein when the labeling process is not 100% efficient. Presently, cells and their corresponding traces are visually inspected to determine if they should be analyzed further. In this paper we present our efforts to apply supervised learning techniques to automate this classification of "good" and "bad" cells.

II. Methods

A. Data

Three researchers from the Covert Lab at Stanford University have provided data sets containing time courses of multiple features per cell and the associated "good" or "bad" labeling. Each of the three data sets reports features and labelings for approximately 2000-3000 cells. The features for each cell are produced by CellProfiler [2] which reports approximately 40 properties, including: *MeanIntensity*, *StdIntensity*, *MeanIntensityEdge*, *Area*, *Perimeter*, and *Eccentricity*. These features are reported for each cell in the frame, for all frames, thus generating two-dimensional time-series traces representing each cell. When a cell is not in the field of view, its features are all recorded as "NaN" for that frame.

We have chosen not to aggregate the data from the three researchers because each researcher uses different experimental conditions and thus has different criteria for classifying cells as "good" or "bad". We seek to develop and demonstrate an algorithm which works for each researcher when trained on the properties produced by CellProfiler in conjunction with the manually-determined labels.

B. Features: Part I

The provided data sets are not immediately amenable to machine learning algorithms. Each cell is represented by two dimensions of data: (1) the properties reported by CellProfiler, and (2) time. In addition, cells remain in the field of view for differing amounts of time. Thus, many commonly-used techniques which require input vectors to be in the same n-dimensional space are not well-suited

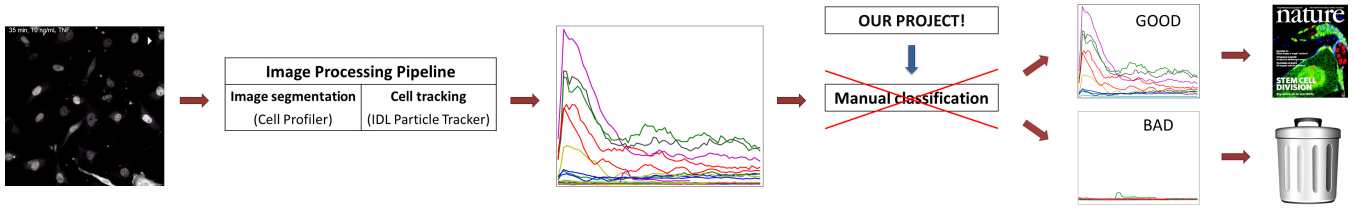


Fig. 1: A schematized (and satirized) overview of the experimental workflow, from data collection to publication. This project seeks to minimize the manual classification of data that must occur prior to the scientific analysis required for publication.

unless missing values are carefully imputed for frames in which a cell is outside the field of view.

Our initial approach is to choose features that are summary statistics of the temporal dynamics of each property reported by CellProfiler—this circumvents the need to impute missing values. We take the minimum, maximum, mean, and standard deviation of each property over all frames in which a cell is in the field of view, thereby representing each cell with one dimension of data. We recognize that this transformation discards many of the temporal features that may be critical for this particular classification problem. Initially, though, we have decided to investigate different classification algorithms with this overview feature set.

Prior to training different algorithms, we examine the pairwise correlations of all the features to remove highly-correlated features. We also remove features which have very low correlation with the observed labeling.

C. Features: Part II

We augment our initial feature vector (described above) with more nuanced temporal information to improve classifier performance. For cells that remain within the field of view for at least 20 frames we compute the Fast Fourier Transform (FFT) on a sliding window of 20 time points and average the successive windows, thereby generating a vector in $\mathbb{R}^{20}$ that represents the frequency content of differing-length traces. We append this to our initial feature vector. For cells that are recorded for less than 20 frames, we append a zero vector.

D. Features: Part III

To further improve performance, we add custom features that incorporate characteristics that each researcher takes into consideration when classifying his or her data set. For example, in Data Sets 2 and 3, researchers look for mean intensity changes in the first 15-20 frames. These intensity changes need not be large in absolute magnitude but must be noticeable compared to an initial baseline. Thus we normalize the *MeanIntensity* time-series property to its starting and maximum value and append features describing the derivative of this normalized trace in the first 15 frames. For Data Set 3 we also append a feature indicating how long a cell remains within the field of view. For Data Set 1 we append features describing the temporal dynamics of *Area* since researchers look at the dynamics of cell area when performing manual classification for this data set.

E. Logistic regression

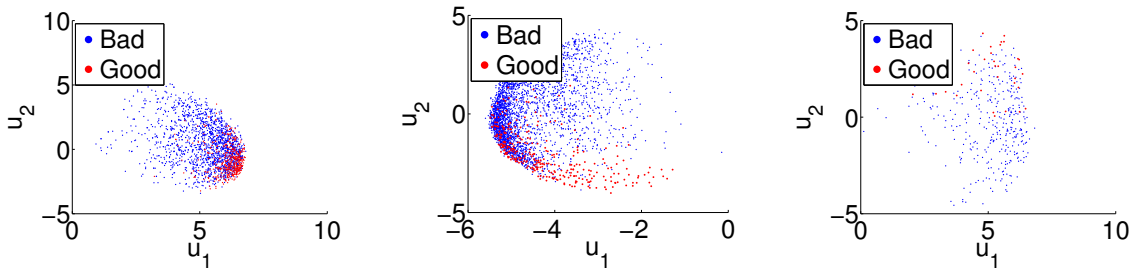
We perform logistic regression on each data set using the `glmfit` and `glmval` functions in MATLAB (The MathWorks, Natick, MA, USA). We have made a slight modification to the `glmfit` function to allow it to iterate for more than the default 100 iterations.

F. Naive Bayes

We implement a Naive Bayes classifier with a multinomial event model. Since our features are reported as continuous values, we discretize these values into a finite number of bins so that we can model $p(x_i|y)$ for each feature x_i as a multinomial distribution. To perform the discretization, we take the maximum value of each feature, the minimum value of each feature, and divide that range into a number of bins. To select the number of bins, we perform cross-validation varying the number of bins from 5 to 150, and select the number which maximizes area under the (receiver operator characteristic) curve.

G. Support vector machine

We train a support vector machine (SVM) using the LIBSVM library [4]. We select a radial basis kernel and select appropriate parameters by performing a grid search in log-space to maximize area under the (receiver operator characteristic) curve obtained by cross-validation.



(a) Data Set 1 (b) Data Set 2 (c) Data Set 3

Fig. 2: Initial Features (c.f. Section II-B) of each data set projected onto the first two principal components.

H. Evaluation of learning algorithms

For each learning algorithm we perform hold-out cross validation, training on 50% of the data and testing on the other 50%. Because the labelings are skewed with 10-20% of cells marked "good" depending on the researcher, we report area under the curve (AUC) rather than accuracy. In general we display and report AUC for precision-recall curves (AUC_{PR}). We also report AUC for receiver operator characteristic curves (AUC_{ROC}) which compare true-positive and false-positive rates. The relationship between precision-recall (PR) and receiver operator characteristic (ROC) curves is discussed in [5] and the authors suggest that PR curves can be more informative for skewed data sets.

To generate PR and ROC curves and compute their respective AUCs, we generally use MATLAB's `perfcurve` function. For SVMs we use slight modifications of LibSVM's `plotroc` function (which calls `perfcurve`).

I. Principal component analysis

In order to visualize our data in two dimensions, we use the `svd` function in MATLAB after standard preprocessing to normalize the mean and variance.

J. Pipeline integration

We integrate the most successful classifiers into the image processing pipeline to augment the manual classification process (the motivation for this project). To achieve this, we save the models and any associated normalization constants so that new data can be appropriately transformed prior to classification.

III. Results

A. Classification results: Part I

AUC_{ROC} and AUC_{PR} values for three different classification algorithms, trained and tested on our first set of selected features (c.f. Section II-B), are reported in Table I.

TABLE I: Classifier performance using initial feature set.

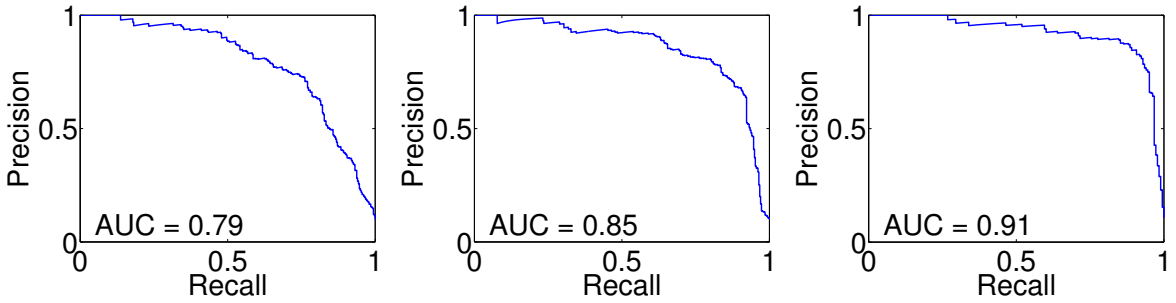
	Data Set 1		Data Set 2		Data Set 3	
	AUC_{ROC}	AUC_{PR}	AUC_{ROC}	AUC_{PR}	AUC_{ROC}	AUC_{PR}
Logistic Regression	0.80	0.48	0.94	0.72	0.61	0.15
Naive Bayes	0.92	0.82	0.93	0.71	0.96	0.58
Support Vector Machine	0.87	0.63	0.95	0.79	0.93	0.54

We see that Logistic Regression does not perform well for Data Sets 1 and 3, while its performance is much more comparable to the Naive Bayes and SVM classifiers for Data Set 2. To investigate this, we observe Figure 2, which shows the initial features for all three data sets projected onto the first two principal components. We see that the data is not linearly separable in low-dimension for any of the data sets, but for Data Set 2, there is a cluster of "good" cells somewhat removed from "bad" cells. This may enable Logistic Regression to better separate data in Data Set 2. However, the best-performing classifier for Data Set 2 is the SVM.

To our surprise, Naive Bayes, when appropriately discretized for each data set, performs better than the SVM for Data Sets 1 and 3.

B. Classification results: Part II

After characterizing the performance of the classification algorithms on the initial feature set, we attempt to make improvements by adding more discriminative features (discussed in Sections II-C and II-D), first by incorporating information describing the temporal dynamics. In particular, we focused on Data Set 2 since the



(a) Initial Features (c.f. Section II-B)

(b) With FFTs (c.f. Section II-C)

(c) Data-specific (c.f. Section II-D)

Fig. 3: Precision-recall curves for the SVM classifiers trained on Data Set 2. Performance, as measured by the area under the curve, increases as the feature set is improved and refined.

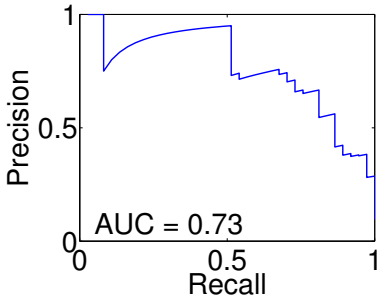


Fig. 4: Precision-recall curve for the SVM classifier trained on Data Set 3 using the full feature set (c.f. Section II-D).

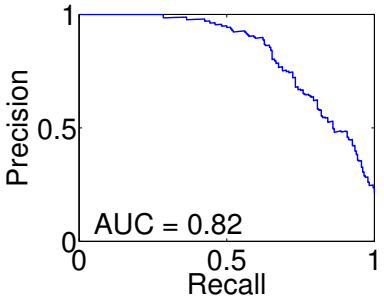


Fig. 5: Precision-recall curve for the Naive Bayes classifier trained on Data Set 1 using the initial feature set (c.f. Section II-B).

researcher who provided it is currently the most invested in the automation of the classification task.

Figure 3b shows the precision-recall (PR) curve of the SVM trained on the feature set described in Section II-C. As a reference, Figure 3a shows the PR curve when the SVM is trained on the initial feature set described in Section II-B. We see that the new feature set which includes a representation of the temporal dynamics improves SVM performance.

We note that that the new feature set improves the SVM performance for Data Sets 1 and 3, but the performance is still less than that of the Naive Bayes classifier trained on the initial feature set (data not shown).

C. Classification results: Part III

We again attempt to further improve classifier performance by incorporating features that reflect the criteria used by researchers when manually labeling their data sets, as described in Section II-D.

Figure 3c shows the PR curve of the SVM trained on the full feature set. We see that the new features have once again improved performance in classifying Data Set 2.

In addition, the full feature set improves SVM performance for Data Set 3 over that of Naive Bayes (trained on any of the feature sets). Figure 4 shows the associated PR curve.

New features did not improve SVM performance on Data Set 1 over that of a Naive Bayes classifier trained on the initial feature set (c.f. Section II-B). The PR curve for the Naive Bayes classifier trained on this data set is shown in Figure 5.

D. Pipeline Integration

The best-performing classifiers are integrated into the experimental workflow at the end of the image processing pipeline. The classifiers make an initial attempt at classifying cells and allow the researchers to decide the final labelings. The researchers use the initial labelings as they rapidly search through their data sets to identify and prioritize which results are worth manually classifying. This was not possible when the overwhelming number of

"bad" time-series traces were not differentiated from—and therefore masked—the "good" time-series traces.

IV. Conclusion

In this work we present our effort to augment and partially automate some of the manual classification tasks involved in processing data from time-lapse microscopy experiments. We have integrated the best-performing classifiers into the experimental workflow, enabling researchers to quickly observe trends in the data so that they can prioritize their analysis tasks. We hope to make further improvements in the future.

Acknowledgements

We thank Jake Hughey, Keara Lane, and Sergi Regot from the Covert Lab for providing their labeled data sets and for offering their feedback as we iterated designs. We thank Professor Andrew Ng and the teaching staff for their guidance this quarter.

References

[1] T. K. Lee and M. W. Covert, "High-throughput, single-cell nf-kb dynamics," *Current Opinion in Genetics and Development*, vol. 20, no. 6, pp. 677 – 683, 2010, genetics of system biology.

[2] L. Kamentsky, T. R. Jones, A. Fraser, M.-A. Bray, D. J. Logan, K. L. Madden, V. Ljosa, C. Rueden, K. W. Eliceiri, and A. E. Carpenter, "Improved structure, function, and compatibility for cellprofiler: modular high-throughput image analysis software," *Bioinformatics*, 2011.

[3] D. Blair and E. Dufresne, "The matlab particle tracking code repository." [Online]. Available: http://physics.georgetown.edu/matlab/

[4] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[5] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 233–240. [Online]. Available: http://doi.acm.org/10.1145/1143844.1143874