BIOINFORMATICS ORIGINAL PAPER Vol. 27 no. 13 2011, pages 1854–1859
doi:10.1093/bioinformatics/btr286

Systems biology Advance Access publication May 9, 2011

Model building and intelligent acquisition with application to protein subcellular location classification

C. Jackson[1,2], E. Glory-Afshar[1,2], R. F. Murphy[1,2,3,4,5] and J. Kovačević[1,2,3,6,*]

[1]Center for Bioimage Informatics, [2]Department of Biomedical Engineering, [3]Lane Center for Computational Biology, Carnegie Mellon University, 5000 Forbes Ave., [4]Department of Biological Sciences, Carnegie Mellon University, 4400 Fifth Ave., [5]Machine Learning Department, Carnegie Mellon University and [6]Department of Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA

Associate Editor: Olga Troyanskaya

ABSTRACT

Motivation: We present a framework and algorithms to intelligently acquire movies of protein subcellular location patterns by learning their models as they are being acquired, and simultaneously determining how many cells to acquire as well as how many frames to acquire per cell. This is motivated by the desire to minimize acquisition time and photobleaching, given the need to build such models for all proteins, in all cell types, under all conditions. Our key innovation is to build models during acquisition rather than as a post-processing step, thus allowing us to intelligently and automatically adapt the acquisition process given the model acquired.

Results: We validate our framework on protein subcellular location classification, and show that the combination of model building and intelligent acquisition results in time and storage savings without loss of classification accuracy, or alternatively, higher classification accuracy for the same total acquisition time.

Availability and implementation: The data and software used for this study will be made available upon publication at http://murphylab.web.cmu.edu/software and http://www.andrew.cmu.edu/user/jelenak/Software.

Contact: jelenak@cmu.edu

Received on September 25, 2010; revised on March 28, 2011; accepted on April 20, 2011

1 INTRODUCTION

The creation of accurate and predictive models for cells and tissues will require detailed information on the subcellular location of all proteins. The field of location proteomics (Murphy, 2005) is concerned with learning this information for entire proteomes and with capturing it in the form of generative models that can be used in cell simulations. Given the scale of the problem, efforts to optimize acquisition of location information are highly desirable (Jackson *et al.*, 2009).

When studying the spatiotemporal behavior of proteins in a single cell using fluorescence microscopy, we typically have to choose a priori the number of frames to acquire. This can be problematic as we do not generally know in advance how many frames will be necessary to obtain the information we seek. We thus risk acquiring either too few or too many frames than needed for our application, thereby wasting time and storage space. We use the term *frame* to represent a time point in the time-lapse imaging of live cells.

Similarly, when learning about a homogeneous population of cells, here called *class* of cells, we may take several cells from this class and acquire a movie of each. However, we do not know in advance how many cell movies we should acquire to gain an understanding of the class nor how many frames to acquire for each cell movie. Once again, we risk acquiring too few cells and frames and not learning what we wish to know, or wasting time by acquiring too many cells and frames. Reducing the area and duration of exposure to the exciting light also protects the sample from photobleaching and phototoxicity in fluorescence microscopy. In this work, we propose intelligent model building and acquisition algorithms to deal with the above problems. These algorithms automatically determine, during acquisition, when to stop acquiring frames from a particular cell, and when to stop acquiring cells from a particular class. As shown in Figure 1, they work by building models during acquisition. This is in contrast to the sequential approach to processing microscopy images, which would view model building as a post-processing step. We apply the algorithms to 3D movies of 12 3T3 cell lines tagged with green fluorescent protein (GFP), with a different protein labeled in each cell line. We consider each cell line to be a different class and determine the parameters of acquisition (how many cell movies we need to learn about each class, and how many frames we need in each cell movie).

We test these algorithms by trying to recognize (classify) the pattern of the labeled proteins and show that:

(1) We can build models both of an individual cell and of a class of cells, which can then be used to correctly classify the subcellular location pattern of a given cell.

(2) When we build models from data that is intelligently acquired, the models achieve a higher classification accuracy on an independent test set than when we build models from the same amount of data acquired with standard acquisition methods (a fixed number of frames per cell and a fixed number of cells per class).

2 METHODS

2.1 Model building and classification

We distinguish between two types of models: a *cell model* is based entirely on a movie of a single cell. A *class model* is based on all cell models from

*To whom correspondence should be addressed.

© The Author 2011. Published by Oxford University Press. All rights reserved. For Permissions, please email: journals.permissions@oup.com

Downloaded from http://bioinformatics.oxfordjournals.org/ at :: on August 30, 2016

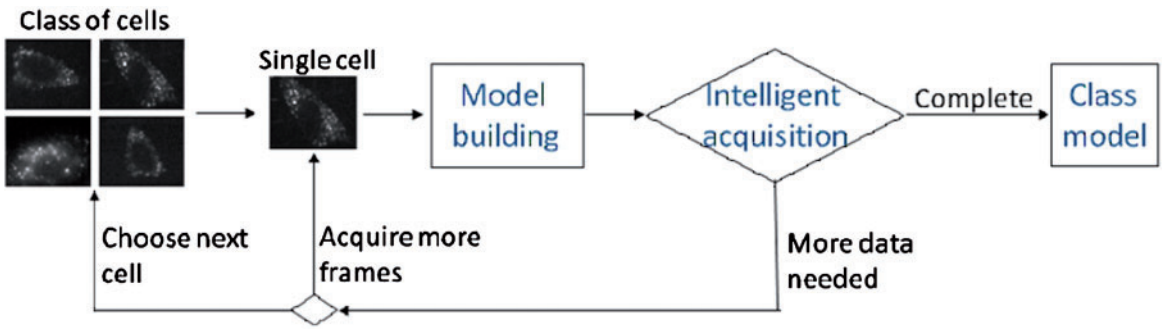

Fig. 1. Framework for intelligent acquisition of a class model.

Table 1. List of static subcellular object features (SOFs)

Name	Description
3D-SOF1.1	Number of voxels in object
3D-SOF1.4	Measure of eccentricity of the object
3D-SOF1.6	A measure of roundness of the object
3D-SOF1.7	The length of the object's skeleton by homotopic thinning
3D-SOF1.8	The ratio of skeleton length to the area of the convex hull of the skeleton
3D-SOF1.9	The fraction of object pixel contained within the skeleton
3D-SOF1.11	The ratio of the number of branch points in skeleton to length of skeleton

The complete 3D-SOF1 set is described in Velliste (2002), based on the 2D-SOF1 set described by Zhao *et al.* (2005). In this study, we use only the features calculated from the protein channel that do not require a nuclear channel to be calculated.

that class. In this application, a class consists of all the cells expressing a specific GFP-tagged protein. Note that while some of these proteins show similar static location patterns, they are largely distinguishable when temporal information is available (Hu *et al.*, 2006, 2010). We now describe how to build cell models and class models, as well as our classification method.

2.1.1 Cell models Our models are based entirely on objects present in the movie. The basis for this approach is our previous work demonstrating that subcellular patterns in static images can be adequately recognized by finding objects by thresholding, describing each object by numerical features, identifying *object types* by clustering and representing an image by the fraction of objects of each type that it contains (Velliste, 2002; Zhao *et al.*, 2005).

The object detection stage aims to find sets of connected pixels in 3D that display a higher intensity level than their local environment. The pixel intensities represent the GFP signal in the protein channel, and the resulting objects represent the pattern displayed by a labeled protein within a cell. Because the background intensity is not uniform and objects may be touching, we use a multi-threshold approach for object detection (Gonzalez and Woods, 2008). First, the background is removed in each z-slice by subtracting the most common pixel intensity below the average intensity. Then, we apply a Gaussian filter (diameter = 5 pixels, SD = 1) to reduce the noise of the image. Finally, we find local thresholds (above a minimum threshold determined empirically), such that objects represent the biggest set of 3D-connected pixels that contain only one maximum intensity.

To assign object types, we compute seven static features based on previous work (Velliste, 2002; Zhao *et al.*, 2005), as shown in Table 1. Similarly to that previous work, we take all objects across all frames and movies, normalize their features to unit standard deviation, and cluster them based on these features using the batch k-means algorithm. An object's type is defined as the number of the cluster into which it falls. We choose the number of types, k, to maximize classification accuracy on the training set using nested cross-validation.

The cell model then consists of three components ($m_\lambda, m_{\lambda,\lambda'}, m_{\lambda,\emptyset}$):

(1) The first component, m_λ, is a *k-by-1* vector representing the proportion of objects of type λ.

(2) The second component, $m_{\lambda,\lambda'}$, is a *k-by-k* matrix representing the proportion of objects of type λ that have a *nearby object* of type λ' in the subsequent frame. We define a *nearby object* as an object whose center is within a distance $d_{\max}$ of the original object. We choose $d_{\max}$ to be 0.5 µm (corresponding to 1 pixel in the z-direction, and 4.5 pixels in the x, y-directions).

(3) The third component, $m_{\lambda,\emptyset}$, is a *k-by-1* vector representing the proportion of objects of type λ that have no nearby objects in the subsequent frame.

When computing these proportions, we take the posterior mean under the assumption of a uniform prior (Bishop, 2006). Hence, to determine m_λ, we count the number of objects of type λ across all frames, storing as N_λ. If N is the total number of objects, we calculate m_λ as:

$$m_\lambda = \frac{N_\lambda + 1}{N + k}$$

Similarly, to determine $m_{\lambda,\lambda'}$, we iterate through all objects of type λ, and look for nearby objects in the subsequent frame of type λ'. If the total number of such events is $N_{\lambda,\lambda'}$, we calculate $m_{\lambda,\lambda'}$ as:

$$m_{\lambda,\lambda'} = \frac{N_{\lambda,\lambda'} + 1}{N_\lambda + k}$$

We derive $m_{\lambda,\emptyset}$ in the same fashion as $m_{\lambda,\lambda'}$, but replacing $N_{\lambda,\lambda'}$ with $N_{\lambda,\emptyset}$, where $N_{\lambda,\emptyset}$ is the number of objects of type λ with no nearby objects in the subsequent frame.

Cell models can be built up while the movie is being acquired, simply by including all objects present up until the current frame. As more frames are acquired, the cell model is refined by adding in the newly available objects.

2.1.2 Class models A class model is simply the collection of cell models for that class. Hence, we can view the class model as a mixture model, where its constituent cell models form the component models of that mixture. As additional cells from the class are acquired, the class model is refined to include the resulting cell models.

2.1.3 Classification To classify a cell model of unknown class, we first measure the likelihood of each of the class's constituent cell models. The likelihood of a constituent cell model m' given the observed cell model

Downloaded from http://bioinformatics.oxfordjournals.org/ at :: on August 30, 2016

m, $L(m'|m)$, is found by looking at the three components of the model individually:

$$L(m'|m) = L_1(m'|m)L_2(m'|m)L_3(m'|m)$$

with:

$$L_1(m'|m) = \prod_{\lambda=1}^{k} {m'}_{\lambda}^{N_\lambda}(1-m'_\lambda)^{(N-N_\lambda)}$$

$$L_2(m'|m) = \prod_{\lambda=1}^{k}\prod_{\lambda'=1}^{k} {m'}_{\lambda,\lambda'}^{N_{\lambda,\lambda'}}(1-m'_{\lambda,\lambda'})^{(N-N_{\lambda,\lambda'})}$$

$$L_3(m'|m) = \prod_{\lambda=1}^{k} {m'}_{\lambda,\emptyset}^{N_{\lambda,\emptyset}}(1-m'_{\lambda,\emptyset})^{(N-N_{\lambda,\emptyset})}$$

Note that the above expressions assume all components of the model are conditionally independent given the underlying cell model m'. This is analogous to the assumption made in a naïve Bayes classifier (Ng and Jordan, 2002), and has been shown to give good classification results even when it does not strictly hold (Zhang, 2004).

We classify by assigning the cell model to the class with the maximum likelihood.

2.2 Intelligent acquisition

We now describe our intelligent acquisition algorithms. We will show it results in a better model for a given amount of data (as verified by classification accuracy). We begin by discussing how many frames to acquire when building a cell model.

2.2.1 Cell models: when to stop acquiring frames for each cell When building a cell model with the end application of classification, the goal is to stop acquiring when the classification decision is unlikely to change. We distinguish between two scenarios:

(1) *Extrinsic scenario*: we have access to the class models during acquisition. This means that we can classify the cell after acquiring each frame, and stop acquiring when this classification result reaches a sufficient confidence.

(2) *Intrinsic scenario*: we do not have access to the class models during acquisition. Hence, we cannot classify during acquisition, and our choice of when to stop acquiring the cell must be based solely on the data from that cell.

As discussed earlier, we classify by choosing the class of maximum likelihood. Equivalently, we can use the log-likelihood of a class, $\ell(c)$, and calculate the standard error of this log-likelihood, $e(c)$.

To demonstrate this, we first rewrite the equations as log-likelihoods, by taking logarithms of both sides:

$$\ell(m'|m) = \ell_1(m'|m) + \ell_2(m'|m) + \ell_3(m'|m)$$

with:

$$\ell_1(m'|m) = \sum_{\lambda=1}^{k}\left[N_\lambda \ln(m'_\lambda) + (N-N_\lambda)\ln(1-m'_\lambda)\right]$$

$$\ell_2(m'|m) = \sum_{\lambda=1}^{k}\sum_{\lambda'=1}^{k}\left[N_{\lambda,\lambda'} \ln(m'_{\lambda,\lambda'}) + (N-N_{\lambda,\lambda'})\ln(1-m'_{\lambda,\lambda'})\right]$$

$$\ell_3(m'|m) = \sum_{\lambda=1}^{k}\left[N_{\lambda,\emptyset} \ln(m'_{\lambda,\emptyset}) + (N-N_{\lambda,\emptyset})\ln(1-m'_{\lambda,\emptyset})\right]$$

A cell model m is built from all of the N individual object observations in that cell. We can see from the above equations that, when we wish to determine $\ell(m'|m)$, the log-likelihood of a cell model m' given the observed cell model m, each individual object observation contributes toward $\ell(m'|m)$. Therefore, we could express $\ell(m'|m)$ as the sum of the log-likelihoods given each individual object observation, $\ell^{(1)}(m'), \ldots, \ell^{(N)}(m')$. We can then express the standard error of $\ell(m'|m)$ as the standard deviation of $\ell^{(1)}(m'), \ldots, \ell^{(N)}(m')$, multiplied by the square root of N, where there are N object observations.

If c_1 is the most likely class and c_2 is the second most likely class, we define the *classification confidence*, C, as:

$$C = \frac{\ell(c_1) - \ell(c_2)}{\sqrt{e(c_1)^2 + e(c_2)^2}}$$

In the extrinsic scenario, we acquire until C exceeds a given *classification confidence threshold*, α. In the intrinsic scenario, we cannot compute the log-likelihood of a class model, because the class models are unavailable. However, we can still compute the log-likelihood of some sample model, including its standard error, providing an indication of how accurately we expect to predict the log-likelihood of the class models when they become available. The natural choice of a sample model is the model of the cell being acquired. We refer to the standard error of the resulting log-likelihood as the *likelihood uncertainty*, and acquire until this likelihood uncertainty falls below the *likelihood uncertainty threshold*.

2.3 Class models: when to stop acquiring frames for each cell

When acquiring a cell to build a *class model*, intelligent acquisition can give even better results. In the extrinsic scenario, we stop acquiring a cell if we recognize that it is similar to a previous cell in that class. This allows us to focus our time and resources on acquiring cells that are different from previously acquired cells of that class, as these provide the most new information.

To assess whether a cell is similar to previous cells in the class, we simply classify it. If it is correctly classified with high confidence (exceeding the classification confidence threshold), we know that this must be the case and we stop acquiring. Moreover, we also stop acquiring when the likelihood uncertainty falls below the likelihood uncertainty threshold ensuring that we do eventually stop acquisition even when the cell is very different from previously acquired cells of that class.

This method implicitly assumes that acquisition alternates between cells of all the classes, such that we can build and refine a class model of every class simultaneously. In the intrinsic scenario, we do not alternate between cells, and thus the classes are not available before starting the acquisition. In this scenario, we can still use the likelihood uncertainty threshold as before, which gives almost as good results for small numbers of cells.

2.4 Class models: how many cells to acquire

Finally, after discussing how many frames to acquire from each cell, we now discuss when to stop acquiring cells altogether. Although all class models will improve when we acquire more cells of that class, we want to identify those for which further acquisition is especially beneficial. We again consider two scenarios:

(1) *Global scenario:* we acquire N_1 cells from every class, but then have time to revisit r classes and acquire an additional N_2 cells. We want to decide which r classes to revisit.

(2) *Local scenario*: we acquire N_1 cells from a class, and then have to decide whether we should acquire an additional N_2 cells. Unlike the global scenario, we do not have access to the other classes, and thus must make this decision locally with the information available at that time.

In the global scenario, we revisit the r classes with the highest *global marginal utility*. We define this as the decrease in classification accuracy on the training set that results from removing a cell from that class. To measure this for a cell C, we begin by testing whether C is classified correctly when all other cells are used for training. Let $\gamma_1 = 1$ when it is classified *incorrectly,* and 0 otherwise. Next, we measure the classification accuracy of all other cells in the same class as C under two scenarios: (i) when training with all cells except the test cell, and (ii) when training with all cells except both the test cell *and* C. Let γ_2 be the mean drop in accuracy from the first scenario to the second. The global marginal utility of cell C is then given by $\gamma_1 + \gamma_2$. The global marginal utility of an entire class is the mean of the utility of its constituent cells.

In the local scenario, for each cell model m in the class, we determine its *closest match*, m', which is the cell model that has the highest likelihood given m. We define the *local marginal utility* of a class as the proportion of

Downloaded from http://bioinformatics.oxfordjournals.org/ at :: on August 30, 2016

Table 2. Overview of the experimental dataset composed of 12 cell lines

Gene	Protein	Location	# cells	# frames
Dia1	Cytochrome b-5 reductase	Cytoplasm	20	19–24
Anxa5	Annexin A5	Nucleus, cytoplasm	18	11–25
Sdpr	Serum deprivation response protein	Vesicles, cytoplasm	23	20–31
Adfp	Adipose differentiation related protein	Vesicles	51	21–27
Timm23	ADP-ATP translocase 23	Mitochondria	40	16–22
Atp5a1	ATP synthase	Mitochondria	20	21
Hspa9a	Mitochondrial stress-70 protein	Mitochondria	24	9–46
Glut1	Glucose transporter 1	Plasma membrane	17	13–82
Cav	Caveolin	Plasma membrane	16	11–49
Tctex1	t-complex testis expressed 1	Cytoskeleton	30	13–36
Actn4	Actinin, alpha 4	Cytoskeleton	29	9–25
Cald1	Caldesmon 1	Cytoskeleton	16	12–34

The number of frames per cell varies depending on the amount of photobleaching incurred during acquisition.

cell models in that class that are chosen at least once as a closest match. This estimates the probability that a new cell will affect the classification result of existing cells. We revisit the classes with high-local marginal utility.

3 RESULTS

3.1 Dataset

We perform our experiments on a collection of 304 3D movies of GFP-tagged proteins in NIH 3T3 cells over 12 different cell lines, with a different protein labeled in each cell line. The lines were generated by CD tagging (Jarvik *et al.*, 2002) and the microscope and image acquisition parameters are as described in Hu *et al.* (2006, 2010). At each time point of the movie, we have a single-channel stack of 15 z-slices, 1280×1024 pixels each. The x,y-resolution is 0.11 microns, and the distance between pixels in the z-direction is 0.5 microns. There is a 45 s interval between frames.

The set of proteins included in this study are located in six major subcellular structures. Table 2 summarizes the tagged gene/protein for each cell line, along with the location in the cell where the protein is expressed, the number of movies for each cell line, and the number of frames in each movie. Note that we aim to distinguish proteins even when they appear in the same subcellular compartment. This is possible because such proteins still exhibit different dynamic behavior, and may also have a different spatial distribution within the subcellular compartment. As a known example, histones, RNA polymerases and nucleoporins are three nuclear proteins with very different distributions, which are respectively involved in the compaction of DNA into chromatin, the replication of DNA (nucleoli) and the nuclear pore complex.

3.2 Classification

We tested our classification method using leave-one-out cross-validation (i.e. when classifying a test cell, we built the class models from all cells except for this test cell), and achieved 84.2% accuracy on our dataset. A previous result on this same 3T3 dataset used only the middle z-slice at each time point, and with a range of morphological, texture and temporal features, achieved an accuracy of 80.9% (Hu *et al.*, 2010). Therefore, in bettering the classification accuracy, we verify that our models are capturing relevant discriminative information present in the movies.

If we consider only the static component of our model—the proportion of objects in each type m_λ—we get 70.4% accuracy. The addition of the dynamic components, $(m_{\lambda,\lambda'}, m_{\lambda,\emptyset})$, completes the model, giving our final result of 84.2%.

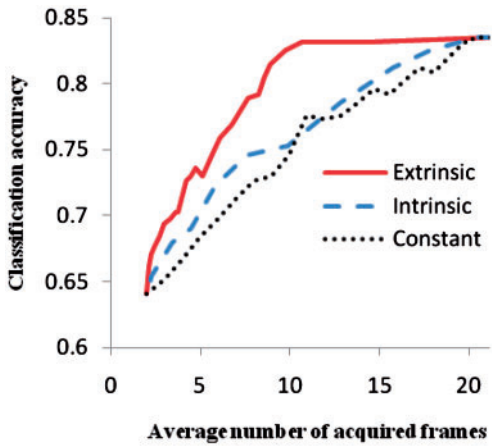

Fig. 2. When to stop acquiring (cell models). The average classification accuracy is shown as a function of the average number of frames acquired with one of three methods to choose when to stop acquiring. The first method (solid line) uses the intelligent algorithm for the extrinsic scenario. The second method (dashed line) uses the intelligent algorithm for the intrinsic scenario. The third method (dotted line) acquires the same number of frames for each movie. The two intelligent methods outperform the standard one, with the best results in the extrinsic scenario.

3.3 Cell models: when to stop acquiring frames for each cell

To test our intelligent stopping algorithms, we take each movie in our dataset and determine how many frames would have been acquired for a given stopping algorithm and threshold. We then measure the average number of frames acquired and the average classification accuracy. For example, in the extrinsic scenario with $\alpha = 0.7$, we acquire 3–35 frames per cell, with an average of 8.0 frames, yielding 80.3% classification accuracy. We compare this to the standard method that acquires exactly eight frames for each cell, yielding only 75.0% classification accuracy. To reach 80% classification accuracy using the standard method, we would need to acquire over 14 frames per cell—almost twice as many as with our intelligent method.

In Figure 2, we compare our intelligent methods for the extrinsic and intrinsic scenarios with the standard method, using a range of stopping thresholds to get the different points on the curves. The intelligent algorithms achieve significantly higher accuracy for the same average number of frames acquired, with the best results for the extrinsic scenario.

3.4 Class models: when to stop acquiring frames for each cell

To test these intelligent acquisition algorithms, we set aside 10 movies as our testing set. We take the remaining 294 training

Downloaded from http://bioinformatics.oxfordjournals.org/ at :: on August 30, 2016

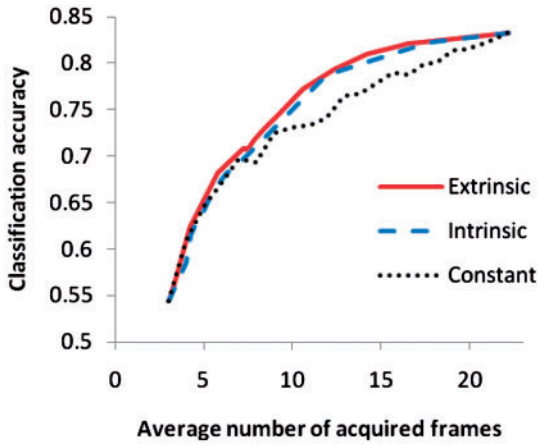

Fig. 3. When to stop acquiring (class models). The average classification accuracy is shown as a function of the average number of frames acquired with one of three methods to choose when to stop acquiring. The first method (solid line) uses the intelligent algorithm for the extrinsic scenario. The second method (dashed line) uses the intelligent algorithm for the intrinsic scenario. The third method (dotted line) acquires the same number of frames for each movie. The two intelligent methods outperform the standard one, with the best results in the extrinsic scenario.

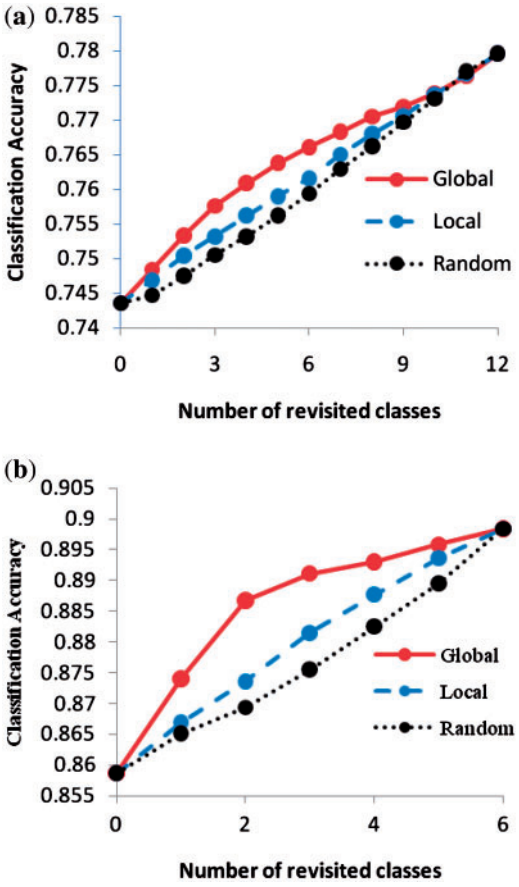

Fig. 4. How many cells to acquire. Initially, we acquire 10 movies from each class. We then acquire five more movies (**a**) or 10 more movies (**b**) from a selected set of classes. The acquired movies are used to train a classifier that is then evaluated on held-out testing data. The solid line shows results when revisit the classes with the high-global marginal utility. The dashed line shows when we revisit the classes with the high-local marginal utility. The dotted line shows the accuracy when we randomly choose which classes to revisit. We can see that both of the intelligent methods perform better than choosing the classes randomly, with the best results attained using the global marginal utility.

movies and determine which frames would have been acquired for each intelligent stopping method and threshold. We then use these to build a class model for each of the 12 classes, and attempt to classify the testing set. We use every frame for movies in the testing set. We ran 10 000 trials, randomizing the testing and training sets, including their order, in each trial. In Figure 3, we show the resulting classification accuracy against the average number of frames acquired per cell with (i) the intelligent acquisition algorithm for the extrinsic scenario, varying the classification confidence threshold to get the different points on the curve, (ii) the intelligent acquisition algorithm for the intrinsic scenario, varying the likelihood uncertainty threshold to get the different points on the curve and (iii) a standard acquisition algorithm that acquires a fixed number of frames per cell. Once again, we see that the intelligent algorithms achieve significantly higher accuracy than the standard method for the same average number of frames acquired; the results for the extrinsic scenario are best. In this scenario, the intelligent algorithm requires only 10 frames per cell to reach a classification accuracy of 80%, whereas the standard algorithm requires 18 frames. We expect that the difference between the two intelligent methods will increase as the number of cells per class increases.

3.5 Class models: how many cells to acquire

To test our methods for how many cells to acquire, we randomly choose one movie (cell) per class to serve as the testing set. From the remaining movies, we randomly choose 10 movies per class as the training set. We choose a set of r classes to revisit and add five more movies from each of these classes to the training set. Finally, we classify the testing set, and record the accuracy.

We vary the number of classes to revisit, r, from 0 (none) to 12 (all). We test three methods for choosing which r classes to revisit: those of high-global marginal utility, those of high-local marginal utility and randomly chosen. The results are averaged over 100 000 trials and shown in Figure 4a, which displays classification accuracy as a function of r. We can see that intelligently choosing which classes to revisit gives higher classification accuracy, with the best results for the global scenario.

Figure 4b repeats the above experiment, but acquires 10 additional movies from the revisited classes (instead of 5). Because we have fewer than 20 movies available for some classes, we do this for only six classes. The results are similar to those of Figure 4a, but the increase in accuracy with intelligent acquisition is greater.

3.6 Computation time

The time taken to build a model is about 1–3 s per frame (Intel Core Duo 2.2 GHz processor, 1.96 GB of memory), depending on the number of objects; the time taken for each of the intelligent acquisition algorithms is negligible. This is relatively fast in comparison with the acquisition time of 45 s per frame. Furthermore, for the algorithms that determine when to stop acquiring frames, we

Downloaded from http://bioinformatics.oxfordjournals.org/ at :: on August 30, 2016

can immediately begin acquiring the subsequent frame even while we are evaluating whether we should stop acquiring or not; thus, this computational penalty only applies on the one frame at which we choose to stop acquiring. For the algorithm that determines how many cells to acquire, the added computation time for building the model of a cell is negligible in comparison with the time it takes to acquire that cell.

4 DISCUSSION

We have demonstrated that intelligently choosing when to stop acquiring frames and cells leads to an increased accuracy for a given amount of acquired data, or equivalently, a reduced acquisition time and resources for a given accuracy. The intelligent acquisition algorithms described here are not closely tied to the model building procedure used, and thus have broad applicability in other modeling scenarios.

In addition, we have presented a model-building technique based solely on the locations and types of objects present within a cell, and shown that the resulting models can classify with a higher accuracy than previous results using all of the image data.

Automated microscopy is increasingly used both for basic research in cell and systems biology and for drug screening and development (Taylor *et al.*, 2006). Approaches such as those we have described here can be directly incorporated into automated microscopes, such as high-content screening systems, which are typically designed to include decision making during acquisition. Alternatively, they can also be relatively easily added to conventional microscopes. In this case, the main challenge is to create a control loop between the model-building software (MBS) and the microscope control software (MCS), in particular to give the MBS the ability to control rudimentary aspects of microscope acquisition. The two critical requirements are for the MBS to be able to retrieve each cell image after it is acquired, and for the MBS to be able to either initiate acquisition of the next frame or stop acquisition of the next frame (assuming that continued acquisition of a large number of frames is the default). These are surprisingly difficult to achieve with the MCS of commercial microscopes as typically configured, because some microscopes wait until all (or a certain number) of frames have been acquired before writing them to disk, and because the MCS often cannot itself be controlled other than via a graphical user interface. At least three solutions present themselves. The first is to configure the microscope with optional 'macro' languages provided by the manufacturer that can incorporate external software into the control loop. The second is to use third-party MCS, such as the open-source MicroManager (Edelstein *et al.*, 2010), which give nearly complete microscope control. This is an excellent solution, with the main disadvantages being that manufacturer support may be lacking in the case of hardware problems and that the performance (latency, acquisition speed) may not be as good as the software that has been optimized for the manufacturer's hardware. The last is to use software that simulates interaction with the graphical user interface to perform basic control. This solution is the lowest cost and has the minimal impact, but the external control software may need to be extensively modified to work with new versions of the manufacturer software.

Once the control loop is established, implementation of the approaches described here is straightforward. As discussed above, the time required for computing models is small relative to acquisition time. The one potential exception is the time required for defining the object types, which is not included in the model-building time (the object types are assumed to be constant during the model building). To learn object types we currently use k-means clustering for many different values of k, and the time required can be many minutes. In the extrinsic scenario, this is not a problem since classes do not change. In the intrinsic scenario, the object types may need to be relearned when a new class is observed. Possible solutions include reducing the range of k over which the search is done, using more highly optimized clustering code, and/or using 'online' or incremental clustering approaches.

For the future, we plan to extend the methods described here to allow models to be built from image series collected with spatial and temporal resolution that may vary under computer control.

ACKNOWLEDGEMENTS

We thank Yanhua Hu, Jesus Carmona and Theodore Scott Nowicki for acquiring the images used in this study.

Funding: National Science Foundation (grant EF-0331657, in part); National Institutes of Health (grants GM075205 and U54 RR022241, in part); the PA State Tobacco Settlement, Kamlet-Smith Bioinformatics Grant (in part).

Conflict of Interest: none declared.

REFERENCES

Bishop,C.M. (2006) Probability distributions. In Jordan,M. *et al*. (eds) *Pattern Recognition and Machine Learning*. Information Science and Statistics, 1st edn. Springer, New York, NY, USA, pp. 76–78.

Edelstein,A. *et al.* (2010) Computer control of microscopes using μManager. *Curr. Protoc. Mol. Biol.*, **Chapter 14**, Unit 14.20**.**

Hu,Y. *et al.* (2006) Application of temporal texture features to automated analysis of protein subcellular locations in time series fluorescence microscope images. In *Proceedings of the 2006 IEEE International Symposium on Biomedical Imaging*, IEEE, Arlington, VA, USA, pp. 1028–1031.

Hu,Y. *et al.* (2010) Automated analysis of protein subcellular locations in time series images. *Bioinformatics*, **26,** 1630–1636.

Jackson,C. *et al.* (2009) Intelligent acquisition and learning of fluorescence microscope data models. *IEEE Trans. Image Proc.*, **18**, 2071–2084.

Jarvik,J.W. *et al.* (2005) In vivo functional proteomics: mammalian genome annotation using CD-tagging. *BioTechniques*, **33**, 852–867.

Murphy,R.F. (2005) Location proteomics: a systems approach to subcellular location. *Biochem. Soc. Trans.*, **33**, 535–538.

Ng,A.Y. and Jordan,M.I. (2002) On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. *Adv. Neural Inform. Process. Syst.*, **2**, 841–848.

Gonzalez,R.C. and Woods,R.E. (2008) *Digital Image Processing*, 3rd edn. Prentice Hall, Upper Saddle River, NJ, pp. 752–763.

Taylor,D.L. *et al.* (2006) *High Content Screening: A Powerful Approach to Systems Cell Biology and Drug Discovery*. Springer, New York.

Velliste,M. (2002) Image interpretation methods for a systematics of protein subcellular location. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

Zhang,H. (2004) The optimality of naïve Bayes. In *Seventeenth Florida Artificial Intelligence Research Society Conference*. The AAAI Press, Florida, pp. 562–567.

Zhao,T. *et al.* (2005) Object type recognition for automated analysis of protein subcellular location. *IEEE Trans. Image Process.*, **14**, 1351–1359.

Downloaded from http://bioinformatics.oxfordjournals.org/ at :: on August 30, 2016