Modeling Stem Cell Culture Growth Using Pattern Recognition

Sunil Pai
Stanford University
sunilpai@stanford.edu

Introduction

While embryonic stem cell biology has shown significant promise, translating this research to the clinic has been challenging due to concerns regarding tumor cell impurities in stem cell injections. To design a culture for embryonic stem cell therapy (to maintain cells for transplantation), cells are placed on a culture dish and monitored using time-lapse microscopy imaging for abnormal growth conditions and bulk morphology characteristics. To determine optimal initial condition for a stem cell culture to assure maximal purity of desired cells, two models need to be designed:

1. monitor cell growth patterns of embryonic stem cells during differentiation (in space and time)
2. monitor cell differentiation (development of stem cells into specific types of cells) using cell pattern recognition

In this paper, I will describe general support vector machine (SVM) model from which conclusions about both cell growth patterns and cell differentiation rates can be inferred. Furthermore, I will generalize the discrete SVM classifications into smooth 2-dimensional signals (confidence profiles) using a probabilistic (sigmoidal) interpretation of SVM classification. This novel interpretation (a slight twist on Platt scaling [4]) actually improves the performance of support vector machines (in terms of probability-adjusted error analysis), and to my knowledge has not been analyzed previously in the literature for multiclass SVM settings. This approach, however, is not the first to perform quality assurance measurements of stem cell cultures as studies by Maddah et al. [5] have shown the promise of pattern recognition in recognizing stress effects during stem cell differentiation procedures. My project builds on this prior work by demonstrating a probabilistic approach which more accurately models the differentiation process.

Approaches for Cell Phenotype Growth Modeling

The overall approach and impact of this work is outlined in Figure 1.

Inputs and Feature Extraction

For the inputs, I used computer vision and signal processing techniques to extract spatial and temporal signatures in timelapse movies of stem cell cultures. Timelapse phase contrast images were taken throughout the quarter using phase contrast and quantitative phase optical techniques (so that we can compare the pattern recognition performance of the two techniques).
timelapse movies usually show undirected differentiation of H9 human embryonic stem cells (the cells extracted from human embryos) into more specific cell types (like smooth muscle cells and neurons). I used these movies to develop a training bank of cell phenotypes and cell textures (using spatial filters like Haralick and Gabor features to reduce the dimensionality and increase the expressivity of the training set). **Haralick features** use gray-level co-occurrence matrices (or GLCMs) for fixed window sizes to give an indication of texture non-uniformity for a total of 20 features in a rotation-invariant manner. **Gabor wavelets** use radial and angular components to determine statistics (e.g. energy/mean amplitude) at several scales and orientations for a total of 60 features. Therefore, for every training example \(x\), I obtained 80 features (20 Haralick, 60 Gabor). Interestingly, Gabor feature extraction is used often for texture analysis because it mimics the manner in which human eyes perceive textures. [1-3]

**Outputs**

The output for cell pattern recognition will classify cell types by recognizing spatial patterns and how they may change over time. While a spatial classifier has already been accomplished previously, I would now like to improve test accuracy for high-noise texture classification sets (which may, prior to model changes, achieve around 77% accuracy). Free parameters that can change accuracy include the regularization parameter \(\mu\) and the noise parameter \(\lambda\).

**Oracles, Baselines, and Metrics for Cell Differentiation Modeling**

For oracle/baseline, I used a multiclass SVM (but with only a single feature) that I have developed previously. A baseline measure can be implemented which measures the average brightness in texture for a given spatial location and a simple set of training images (which usually correlates to cell type). It is readily apparent that this strategy will fail because brightness varies across images and this is confirmed by a training accuracy of 54%, close to an effectively random baseline training accuracy of 36%. While it is difficult to obtain an oracle for cell differentiation modeling (we have not yet developed a scientific ground truth for each classification), our best attempt at an “oracle” has been the human expert that trains the dataset beforehand. After expert training, we can perform \(k\)-fold cross validation where we select part of our training set to put into the SVM and the rest to use as “test” data.

**Classification Model**

**Motivation and Prior Work**

After evaluating the performance of my machine learning algorithm when I took CS229, I determined that linear SVM (implemented using LIBSVM) outperformed quadratic and Gaussian SVM models as well as linear discriminant analysis (LDA) and therefore focused on linear SVM for this project. This year, in CS221, I decided to develop my own SVM models using convex optimization tools. Convex optimization obtains the multi-class hinge loss SVM classifier and provides better access to classifier boundary parameters and confidence estimation than black box tools that I have used previously. For cell pattern recognition I invoke Haralick and Gabor wavelet features (spatial filters that are commonly used for texture classification) to determine spatial patterns. Comparing \(w\)-SVM and \(\lambda\)-SVM models, I have determined methods with which to improve the accuracy and expressivity of SVM models for signal processing applications such as cell migration.

**The \(w\)-SVM Model**

Previously, I classified the images by analyzing texture “windows” in the images and evaluated which class they belonged to. Using convex optimization toolboxes, I actually implemented
my own SVM model (which I will call \(w\)-SVM) based on the hinge loss model we derived in

class, training the images using a custom-made MATLAB GUI. This quarter, in CS221, I am
developing a different SVM model: one whose classification scheme varies with time using a
weighted loss function.

For feature classification, we use a convex linear mean max loss multi-class SVM classifier model.
The data consists of ordered pairs of features and classifications \((x^{(i)}, y^{(i)}) \in \mathbb{R}^n \times \{1, 2, \ldots, K\}\) for \(i = 1, 2, \ldots, m\), where \(n\) is the number of features, \(K\) is the number of classes, and \(m\) is the
number of samples. Because we are using a linear kernel for the SVM, we use \(K\) affine functions
\(f_k(x) = a_k^\top x + b_k, 1, \ldots, K\), from which we define the vector function \(f = Ax + b\), where the rows
of \(A\) are \(a_k^\top\) and the elements of \(b\) are just \(b_k\). For any given feature \(x\), we guess the classification
label \(\hat{y} = \arg \max_k f_k(x)\). We also assume that \(\sum b_k = 0\). A correct classification requires that
\(f_{y^{(i)}}(x^{(i)}) > \max_{k \neq y^{(i)}} f_k(x^{(i)})\) for all \(i\). This is feasible when \(f_{y^{(i)}}(x^{(i)}) \geq 1 + \max_{k \neq y^{(i)}} f_k(x^{(i)})\).

We define a loss for sample \(i\) as the quantity:
\[
l_i(A, b) = \max(1 + \max_{k \neq y^{(i)}} f_k(x^{(i)}) - f_{y^{(i)}}(x^{(i)}), 0)
\]
(1)

We need to optimize over \(A\) and \(b\) to minimize the average loss over all of the samples. However,
when we are more interested in a particular class (for example, we are more interested in differ-
entiated cells), we amplify the loss for samples of that class in the SVM model. This justifies the
weight vector \(w \in \mathbb{R}^K\) which satisfies the condition \(\sum_{k=1}^{K} w_k = 1\). We get our final loss function
which is a convex function for \(m\) samples, defined as:
\[
L(A, b) = \sum_{i=1}^{m} w_{y^{(i)}} l_i(A, b)
\]
(2)

To regularize this problem, we use the use of the weighted \(\ell_2\)-norms of the column vectors
\[
\|A\|_w = \sum_{k=1}^{K} (1 - w_{y^{(i)}}) \|a_k\|_2.
\]
The final objective function \(J(A, b)\) (to optimize over \(A, b\)) is shown below:

\[
\text{minimize } J(A, b) = L(A, b) + \mu \|A\|_w
\]
subject to \(A^\top b = 0\)

where \(\mu\) is a regularization parameter. The \(w\)-SVM strategy ideally imposes stricter requirements
on losses with higher weights and looser requirements on losses with lower weights and therefore
significantly affects the ultimate choice of \(A\) and \(b\).

The \(\lambda\)-SVM Model

While \(w\)-SVM can bias SVM classification over the course of a time-lapse video, results have
shown that adjusting \(w\) only provides marginal benefits to cross-validation accuracy (this will
be demonstrated in the following sections). A more important consideration when classifying
images that evolve over time from one set of textures to another is to consider the noise between
different texture levels. Importantly, the \(\lambda\)-SVM allows one to observe intermediate patterns
over the course of the movie (a pattern that belongs to no one texture, but rather to all textures
with some set of probabilities).

The \(\lambda\)-SVM classifier is actually identical to the original SVM classifier. However, instead of
strictly classifying test examples, we assign a probability for all possible classifications based
on the confidence we have in such classifications. In the original problem, we had defined the hinge loss function $l_i$, but we can also define the residual $r_i$ which may be positive or negative depending on which side of the margin the sample $x$ is located. Note that parameters $A, b$ and the affine classifier $f$ are defined as in the previous section:

$$r_k(A, b, x) = \max_{k' \neq k} f_{k'}(x) - f_k(x)$$

Based on this residual, we can define a modified confidence estimate for multiclass support vector machines for each class using the sigmoid function $\sigma(x) = (1 + e^{-x})^{-1}$ based on the idea by original SVM inventor John Platt (though his idea was only described for the case of two classes) [4]. The confidence $c_k(x)$ is defined as:

$$c_k(x) = \sigma\left(-\frac{1 + r_k(A, b, x)}{\lambda}\right)$$

where $\lambda$ (the namesake of the model) represents the noise-tuning parameter for the sigmoid function. Figure 2 demonstrates the behavior of the sigmoid function with varying $\lambda$. The parameter $\lambda$ is the noise tuning parameter because we calculate the probabilities of the classifications by normalizing the confidence estimates as follows:

$$p_k(x) = \frac{c_k(x)}{\sum_k c_k(x)} \propto c_k(x)$$

**How does $\lambda$ affect the $p_k$?**

For large $\lambda$, we expect the classifier to assign uniform probabilities for a given sample across all classes. Whereas for the original SVM (which has $\lambda = 0$) we expect the classifier to assign a probability of 1 to predicted label and 0 to the others and this setting has the least amount of noise. This is because the sigmoid function approaches the step function as $\lambda \to 0$. An example of the effect of $\lambda$ is shown in Figure 3.

**How does one calculate the probabilistic error for a test set?**

In order to test the accuracy of a given test set, we need to develop a reasonable measure that approaches the original SVM accuracy when $\lambda \to 0$. To accomplish this task, we can simply define the error $e_k$ for class $k$ as (where $m_k$ is the number of test examples $(x^{(i)}, y^{(i)})$ with $y^{(i)} = k$):

$$e_k = \frac{1}{m_k} \sum_{i, y^{(i)} = k} (1 - p_k(x^{(i)}))$$

In other words, we take the difference between the expected probability from our test set, 1, and predicted probabilities $p_k$ to find the error. This generalizes to the original case where $\lambda = 0$ and $p_k(x^{(i)}) \in \{0, 1\}$ where accuracy becomes simply the number incorrect divided by the number of test examples with classification $k$.

**Results**

While I had already implemented an SVM with high accuracy (about 95% for phase contrast) in CS229, I did not adapt my classification scheme for cell growth/probabilistic modeling until CS221 when I found that noisier datasets could have validation accuracies as low as 75%. I accomplished this feat by performing several experiments with SVM.
Smoothing with Temporal Features

In an effort to smooth classifications temporally, I simply added Haralick and Gabor wavelet features from the previous frame and the future frame to capture short-term time evolution of cell textures. Using this algorithm, obtained a training accuracy of 100% which is better than what I was able to achieve previously, I decided to test overfitting by performing 2-fold cross validation. I obtained a cross-validation of 68.2% for my current implementation and 80% for my old implementation (which might be considered an “oracle” for my current endeavor). Of course, I will need to improve on my cross-validation accuracy of 68.2% and the fact that the original cross validation was much higher (despite the lower training accuracy) suggests that there was indeed overfitting and a better approach is required.

\( w \)-SVM

To test whether \( w \)-SVM had a significant effect on classification accuracy, I designed a simple experiment to test cross validation for different \( w \). I compared the performance of three weighting schemes: uniform weights (original SVM), least frequency, and most frequency. The results for this are shown in Table 1:

<table>
<thead>
<tr>
<th>Method</th>
<th>Validation Accuracy</th>
<th>Training Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>77.43 %</td>
<td>81.97 %</td>
</tr>
<tr>
<td>( w )-SVM*</td>
<td>77.62 %</td>
<td>81.45 %</td>
</tr>
<tr>
<td>( w )-SVM**</td>
<td>77.67 %</td>
<td>82.1 %</td>
</tr>
</tbody>
</table>

Table 1: A comparison of the validation accuracies and training errors for several different weighting schemes (uniform, \( w_k = \nu_k ^* \), \( w_k = 1 - \nu_k ^** \)) where \( \nu_k \) is frequency of class \( k \).

The results indicate that there is no significant improvement in the classification accuracy after changing the weighting scheme from uniform to least or most frequency. However, the results also indicate that my SVM algorithms perform significantly better than the baseline strategy (about 20% more accurate).

\( \lambda \)-SVM

After compiling large training databases for phase contrast and quantitative phase images, we numerically determined the \( \lambda \) at which the 10-fold cross-validation accuracy is minimized. Our experiments show that the noise rating \( \lambda_{\text{min}} \) for phase contrast images (\( \approx 0.5 \)) is less than the noise rating for the quantitative phase images (\( \approx 1 \)) which was to be expected because the distinction between cell textures in phase contrast images were clearer to the naked eye. Furthermore, the key result is a reduction in probabilistic error of 3.6% using \( \lambda \)-SVM compared to the original SVM for quantitative phase images. For most of the investigations in this project, \( \lambda \)-SVM was applied to phase contrast and quantitative phase images with \( \lambda = 1 \). Results for \( \lambda \)-SVM are shown in the graph in Figure 4.

Test Set Results Were Reproducible

For the first time, I have determined that my SVM classification identifies cell phenotype growth patterns consistently for three quantitative phase movies (the magic number for biological publications). This suggests that with further investigations of stem cell cultures in similar settings, one might determine a reproducible population growth model using \( \lambda \)-SVM. The results for reference are shown in Figure 5.
Analysis and Discussion

Learning Curve

Since the learning does not employ the stochastic gradient descent method, but rather convex optimization methods (which give slightly more optimal solutions and converge faster), I artificially generated a learning curve by randomly selecting a range of training sets of different sizes and determining their accuracy on randomly generated test sets. This resulted in the learning curve shown in Figure 6.

Time Complexity

More work is still required to reduce the time required for feature extraction. While training takes around 5 seconds, the classification of an image can take around a minute to complete (because there are thousands of test examples to classify per image), making timelapse classification a prolonged task. Other types of features that are quicker to calculate (for example, the local binary pattern used by Maddah et al. [5]) and similar in accuracy may be a next step to increase the efficiency of data collection.

How w-SVM Failed

A surprising finding was that w-SVM did not provide significant benefits over traditional SVM. In particular, although \( A \) and \( b \) were different, the affine function \( f(x) \) classified all \( x \) similarly to the original SVM. Furthermore, for changes in \( w \) to significantly affect the cross validation accuracy, I had to set the weights to real numbers of the order \( 10^{-7} \) or \( 10^{-8} \).

Why \( \lambda \)-SVM Worked

The comparison in Figure 4 suggests that \( \lambda_{\text{min}} \) can describe the noisiness of data sets. For the quantitative phase time-lapse training data, the different textures were sometimes difficult to distinguish, even with the naked eye. This led me to believe there would be some noise in the training data and I should expect relatively higher \( \lambda_{\text{min}} = 1 \). On the other hand, the textures in the phase contrast images were significantly easier to distinguish, contributing to both the lower cross-validation accuracy and the lower noise rating \( \lambda_{\text{min}} = 0.5 \).

The advantage of using \( \lambda \)-SVM is that it greatly complements the generation of smooth spatial signals and increases the probabilistic accuracy defined in the Classification Model section. In particular, \( \lambda \)-SVM is useful for data sets like mine where one cell phenotype smoothly transitions into another over time. For example, if an embryonic stem cell becomes integrated into a dense colony of cells, it will achieve a state between a single cell texture and a highly compact texture, which would require a probabilistic interpretation.

Biological Interpretation

As can be inferred from Figure 5, there are three distinct cell morphologies that are expressed in the timelapse movie: highly compact cells, single cells, and medium-compact cells. The single cells slowly form into medium compact cells (which is responsible for the slight rise in the magenta curves in the first 1000 minutes), and the medium compact cells then become increasingly compact and slowly join the highly compact cell population (resulting in a simultaneous dip in the magenta curve and rise in the red curve). While this growth pattern makes sense at a qualitative biological level, quantitative analysis of these curves may allow future clinicians to assess the growth quality and general health of their stem cell cultures over time compared to some standard.
Conclusion

In this project, we can report two key findings: (1) $\lambda$-SVM provides a highly accurate, probabilistic interpretation of support vector machine classification and (2) population growth data extracted from $\lambda$-SVM were shown to follow similar trajectories, demonstrating biological relevance of my technique. Ultimately, the hope of this work is to translate these findings into viable stem cell quality assurance software that can track the proliferation, migration, and differentiation of stem cells over extended periods of time. Future applications may include integrating this tool with laser microdissection of differentiated cells from stem cell cultures to pave the way for one of the first FDA-approved stem cell transplantations from embryonic stem cells. Furthermore, the findings I have reported for $\lambda$-SVM may be extended to any application that seeks to assess training and test data noise during classification.

References


Appendices

Figure 1: Pattern recognition of stem cell differentiation. 1) Select region of interest (ROI); 2) Determine Gabor filter bank and graylevel co-occurrence matrix; 3) Get Gabor and Haralick statistics; 4) Determine cell phenotype.
Figure 2: The sigmoid function approximates the confidence that any given training example is correctly classified in an SVM. Adjusting $\lambda$ allows one to adjust confidence for data points that lie close to the boundary (which in this case is at $x = -1$).

Figure 3: (left) Sample classification of a quantitative phase image. **Blue** = background, **Red** = highly compact, **Magenta** = medium compact, **Green** = single cell. (right) Confidence profiles for the classes highly compact, single cell, medium compact (across) and for noise levels $\lambda = 1, 3$ (vertical). These profiles were smoothed using a Gaussian filter.
Figure 4: Determination of noise levels for training sets with varying amounts of noise ratings $\lambda_{\text{min}}$. 

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{determination_of_noise_levels.png}
\caption{Determination of True Classification Noise $\lambda_{\text{min}}$.}
\end{figure}
Figure 5: Texture frequency curves for textures in a quantitative phase time-lapse video (H9 embryonic stem cell differentiation). Signals for each texture significantly change depending on λ. The curves measure the frequency of each texture in the images over time, weighted by the confidence of each texture. This information suggests that pattern recognition is biologically relevant and the results are reproducible for three time-lapse movies.
Figure 6: The learning curve for the SVM classifier shows that the number of training examples is within the region of convergence in the learning curve (top). Also, the gap between the test set error and the training set error is greatly reduced as expected.