We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter 1

A Proposal for a Machine Learning Classifier for Viral Infection in Living Cells Based on Mitochondrial Distribution

Juan Carlos Cardona-Gomez, Leandro Fabio Ariza-Jimenez and Juan Carlos Gallego-Gomez

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/61293

Abstract

The study of viral infections using live cell imaging (LCI) is an important area with multiple opportunities for new developments in computational cell biology. Here, this point is illustrated by the analysis of the sub-cellular distribution of mitochondrium in cell cultures infected by Dengue virus (DENV) and in uninfected cell cultures (Mock-infections). Several videos were recorded from the overnight experiments performed in a confocal microscopy of spinning disk. The density distribution of mitochondrium around the nuclei as a function of time and space $\rho(r,\ \theta,\ t)$ was numerically modeled as a smooth interpolation function from the image data and used in further analysis. A graphical study shows that the behavior of the mitochondrial density is substantially different when the infection is present. The DENV-infected cells show a more diffuse distribution and a stronger angular variation on it. This behavior can be quantified by using some usual image processing descriptors called *entropy* and *uniformity*. Interestingly, the marked difference found in the mitochondria density distribution for mock and for infected cell is present in every frame and not an evidence of time dependence was found, which indicate that from the start of the infections the cells are showing an altered subcellular pattern in mitochondrium distribution. Ulteriorly, it would be important to study by analysis of time series for clearing if there is some tendency or approximate cycles. Those findings are suggesting that using the image descriptors *entropy* and *uniformity* it is possible to create a machine learning classifier that could recognize if a single selected cell in a culture has been infected or not.

Keywords: Computational Cell Biology, Dengue Virus, Mitochondria, Machine Learning

1. Introduction

At the latter years of the past century, cell biology experienced a fast growth, thanks to the convergence of several techniques, which have substantially improved the confocal microscopy field. Now, the observation in real time of the structural and functional unit of life is possible. The ultrarefrigerated CCD-cameras with electromultipliers; the implementation of confocality based on disk spinning without the necessity of high-energy lasers (which could damage the living cells in a few seconds); the increasing capacity of computational processors; and the ability of the genetic engineering for coding fluorescent proteins mutants [1, 2], offering the possibility of a color palette that was previously unthinkable for the cell molecular biologists [3, 4]. All together with the ability to generate cells with fluorescent compartments, opened the doors to maybe the most remarkable and important scientific and technological development for a new era in cell biology named Live Cell Imaging. Before the 1990'-s this kind of research was known by the unpopular name "time-lapse video microscopy", as it is detailed in a protocol book widely known at that time written by A.J. Lacey [5].

At the beginning of the new millennium, the necessity of introducing new and improved mathematical and computational tools was made evident. This was because the amount of data produced in a single experiment could overload the capacity of personal computers and the conventional software was not loaded with the required algorithms to process such data. Then a strategic alliance with researchers on the areas of artificial intelligence, applied mathematics, and physics was apparent. These new cooperations make perfect sense due that even from the beginning of life science studies, it was clear that the dynamical rules involved were complex, non-linear, and possibly not even deterministic but probabilistic. The virologists, for example, have discovered that the infection rate is governed by a non-linear pattern and the cellular physiology of several processes turn out to be more complicated than it was expected. In consequence, the mathematical modeling became the main strategy in the journey for knowing and understanding the cell biology. The amount of the data available nowadays could not be analyzed by conventional human heuristics. Fortunately, the computational biology field and its tools offer the required resolution and robustness in diverse problems. It goes even further, because the computational algorithms work evenly in any case. When dealing with complex biological problems, to have a working computational model will get us closer to the reality and help us avoiding the human bias present in heuristic approaches. It is in complex problems where the convenience of using powerful statistical tools to build models became apparent. The main strategy here is to try to "learn" the model directly from the experimental or observational data.

The term "machine learning" (ML) refers to a branch of the artificial intelligence field, that concerns to the study and construction of algorithms with the ability to learn from the existing data. In such algorithms, a set of parameters is fitted to provide the best input-output relationship between the information available. When talking about a computational code that implements the techniques, algorithms, or principles found in machine learning theory, it is usually called a machine learning program. The literature on this topic is quite large; however, some very popular books are those by Duda et al. [6], Webb [7], and Bishop [8].

A commonly accepted definition of the process of "learning" is due to Tom M. Mitchell [9]: *a machine learns to perform a task T if its performance as measured by $\mathcal{P}$ increases with the experience E*. The experience *E* is the feedback the machine received to validate its output. The ML set of techniques have a broad range of applications in several fields of knowledge, including the building of autonomous robots [10], the astrophysical data mining [11], the study of dynamical systems and complex networks without the explicit knowledge of the dynamical equations [12], the patterns and shape recognition [6, 8] in images as used by the face recognition programs in social networks web pages, the hand writing OCR (optical character recognition), and of course, in medicine (automatic diagnosis based on symptoms) and biology (gene sequencer, classification of cellular morphology, etc.).

Particularly, the shape recognition capabilities have important applications to live cell image processing. For example, in 2006, Neumann and collaborators use live cell imaging to study the RNAi screening [13]. They developed a ML that recognize the morphologies present in the cells images and associate them (classify) with the corresponding phenomenology: interphase, mitosis, apoptosis and binucleated cells morphologies were studied through a multi-class classifier using support vector machine (SVM). They reported to obtain up to 97% accuracy from the SVM in comparison with "manual" classification through the observation by some very well-trained biologists.

Due to the huge amount of data provided by a live cell image (LCI) measure, it becomes unpractical to relay only on the lecture and interpretation by a well-trained researcher. It is also possible to have different interpretations coming from different scientists when analyzing the same image. Then, the ML ability to recognize and characterize particular morphologies present in an image is very useful to avoid the slow and tedious process of visual discrimination. Also, it can avoid some human bias by following well-defined rules. However, to fully train the machine, it can be necessary to have a large number of image samples from the phenotype under study. It means, to have enough sample cells expressing the phenotype and some other cells to use as a control group. Sometimes, that condition is not fulfilled. In order to asset this kind of problem, Thouis R. Jones and collaborators implement a ML with interactive feedback to characterize diverse and complex morphological phenotypes [14]. They use the criteria of well-trained researchers as a feedback in the learning stage of the machine, and provide the code [15] for the world to use under a free license.

Several generic implementations of ML techniques have been developed and presented as toolbox in scientific software. However, it is pretty common to find the particular phenomenon under study to be better fitted by some unique implementation developed explicitly to deal with it. This can be a consequence of the particularities of the problem or sometimes this is just due to the lack of proper documentation on the available tools.

This chapter is organized as follows: first, a brief description of some common methods used to build a ML are provided, followed by a description of the performed experiment and computational analysis to obtain the information from the graphical data. Finally, the results and a proposal to create a ML to characterize viral infection are presented.

2. Machine learning concepts overview

From the mathematical point of view, a ML can have one of two primary objectives: regression and classification. When the machine is used to compute the best response to a given situation among a continuous range of possible answers, it is called a regression problem. And when the machine is due to choose among a discrete set of possibilities, it is called a classification problem.

The shape recognition and feature extraction from images is a classification problem, where the duty of the machine is to find the class which has the highest probability to contain the current input value. In this context, a class is defined based on a set of measurable attributes found in an image; it can be geometrical attributes (length, shape, eccentricity, size), pixel intensity, etc. In general, the input for a ML program is a set of measurable variables or attributes, which are set in vectors. It is common in ML literature to call these attributes *features* and the vectors *feature vectors*, so these names will be used in such framework in the rest of this chapter. Each input in a feature vector represents an attribute and each vector represents a state of the system.

Before the machine is ready to be used as a classifier or predictor, it needs to be trained among some data. Here, "training" refers to the process of parameters optimization, where the machine is optimized to get the best result against the training set. This process is not perfect, and some human criteria need to be implemented. If the model has not enough freedom to fit to the training set, it gets under fitted and do not reproduce the characteristics of the system under study. On the other hand, having too much freedom in the ML leads to a model that fits pretty well in the training set, but is unable to predict accurately the outcome for a feature vector outside of the training set. This is called bias. To avoid bias, it is customary to split the available data in two sets, the training set and the testing set. A trained machine is challenged with the testing set, and the accepted ML model is the one that has the best results against it.

There are two main paradigms for the training of a classifier, the supervised and the unsupervised learning. In supervised learning, each sample in the training set is consisting on a features vector and a class flag, i.e., the classes to which each sample in training set belongs are known a priori. After the learning process, a computational model that can predict the right class flag for most of the training set is obtained, and hopefully, it would predict the correct class for a new sample with high accuracy. Also, the classifier must return information about how confident its prediction is, i.e., a value of dude must be reported.

When no predefined classification is available, a ML algorithm can be used to search for common patterns or similarities into the training set, which do not contain class information yet. The machine would cluster samples with similar feature vectors to define a class, and then, it will use the found class to characterize new input data. To do that, it is necessary to define some measure of similarity (Euclidean distance in features space, for example) that can be used to group the input vectors into clusters. The objectives of this kind of ML are first to cluster the data from the training set into classes, and then, set a classifier to characterize new inputs.

http://dx.doi.org/10.5772/61293

ML are suitable to treat complex problems in which the explicit mathematical form describing the interactions occurring in the process are not known, i.e., the dynamical equation ruling the systems are unknown. Being so, the computations involved in a model built with ML are not deterministic but probabilistic, based on the information gathered by direct measures. The more data are available to train the machine, the more accurate the prediction will become.

2.1. Supervised learning

The objective of the classifier is to draw a frontier that splits the feature space into k disjoint subsets [16] called the border line or border hyperplane. Hopefully, each subset will contain the feature subspace associated with one single class.

2.1.1. Hypothesis function

Let the features space be called $\mathcal{X}$, where a feature vector is x, and let the whole set of classes be $\mathcal{Y}$, with an individual class denoted y. Develop a ML model consists in building a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ that is a good predictor of the corresponding y to a given input x. In supervised learning, a computational model function with adaptable parameters θ, $h_\theta(x)$must be build; this function is usually called "hypothesis". Also, a cost function $J(\theta)$ must be defined to provide an idea of how accurate the hypothesis is when predicting the output values or classes for the whole training set. A common choice is the least square cost function:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2, \tag{1}$$

where m is the number of samples or elements in the training set, and the notation $x^{(i)}$ and $y^{(i)}$ denotes the feature vector of the i-th sample and the class value for it. The optimization of the model (learning process) is then achieved by minimizing the cost function. Once the ML model is set up, the predictions are made with the optimized hypothesis as $y = h(x)$. Depending on if $h(x)$ is a continuous function or a discrete one, the machine will be doing regression or classification respectively.

Another approach comes from a probabilistic interpretation of the hypothesis function. Suppose that $y^{(i)} = h\left(x^{(i)}\right) + \in^{(i)}$, where $\in^{(i)}$ is a random error which takes care of unmodeled effects and possibly random noise. And assume that $\in^{(i)}$ are IID (independent and identically distributed), and follows a Normal distribution (Gaussian distribution), of mean zero and some variance σ^2, $P(\in^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{-\left(\in^{(i)}\right)^2}{2\sigma^2}\right)$, then the probability of $y^{(i)}$ conditioned to $x^{(i)}$ and parametrized by θ is:

$$P\left(y^{(i)} \mid x^{(i)}; \theta\right) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{-\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2}{2\sigma^2}\right). \tag{2}$$

As any set $(x^{(i)}, y^{(i)})$ is independent from the others, the probability of the whole set $P(\mathcal{Y} \mid \mathcal{X}; \theta)$ is the product of all the individual probabilities. When $P(\mathcal{Y} \mid \mathcal{X}; \theta)$ is taken as a function of the parameters θ, it is called the likelihood function:

$$\ell(\theta) = \prod_{i=1}^{m} P\left(y^{(i)} \mid x^{(i)}; \theta\right). \tag{3}$$

The principle of maximum likelihood establishes that the best model representation of the data is given by the set of parameters θ that provides the maximum probability. Then, maximizing the likelihood function for the whole training set (or any monotonically increasing function of it) is equivalent to minimize the cost function Eq. (1).

2.1.2. Logistic regression

Suppose the problem at hand is to determine if the measure of some experiment belong to one out of two possible outputs (like, for example, to determine if a tumor is benign or malign). A class flag 0 or 1 must be associated for each output. In this case, a common approach is to propose a logistic function (also known as sigmoid function) as a classifier, it is called a *logistic regression*:

$$g\left(\theta^T x\right) = \mathrm{sig}\left(\theta^T x\right) = \frac{1}{1 + \exp\left(-\theta^T x\right)}, \tag{4}$$

$$h_\theta\left(x\right) = g\left(\theta^T x\right) \tag{5}$$

The sigmoid function sig(z) has asymptotic values of 1 when $z \to \infty$ and 0 when $z \to -\infty$. So the conditional probability for the feature vector x to belong to each one of the two available classes is written as:

$$\begin{aligned} P\left(y = 1 \mid x; \theta\right) &= h_\theta\left(x\right), \\ P\left(y = 0 \mid x; \theta\right) &= 1 - h_\theta\left(x\right). \end{aligned} \tag{6}$$

Which can be summarized in a single probability density function (PDF)

$$P\left(y \mid x; \theta\right) = \left(h_\theta\left(x\right)\right)^y \left(1 - h_\theta\left(x\right)\right)^{1-y}. \tag{7}$$

Once the PDF is set, the process of learning consist in maximizing the likelihood of such PDF to the training data set. By computational simplicity, it is convenient to maximize instead some

monotonically increasing function of the likelihood. It is common to work with the logarithm of the likelihood (log-likelihood function). When using the logistic regression, this hypothesis function would not return the prediction of an output class, but the probability for the sample feature vector belongs to a given class.

If it is needed to get a class value as an output, it can be done by setting $z = \text{sig}(\theta^T x)$ instead of Eq. (4) and defining

$$g\left(\theta^T x\right) = \begin{cases} 1 & if\ z > 0.5, \\ 0 & otherwise, \end{cases} \tag{8}$$

then minimizing the cost function Eq. (1). This last strategy is known as the *perceptron learning algorithm*. The classifier in this example can be extended to k classes by a simple one vs all algorithm. It is, defining a logistic regression to compute the probability for any of the k classes, $h_{\theta_1,1}(x)$, $h_{\theta_2,2}(x)$, $h_{\theta_{k-1},k-1}(x)$,..., $h_{\theta_k,k}(x)$, and returning the class which has the highest probability. Note that for each class, a parameters vector θ_i must be optimized.

2.1.3. Non-linear classifiers

One limitation of the techniques summarized so far is that they provide a linear model for the classifier, i.e., the decision border is represented by a straight line or hyperplane. This can work perfectly fine if the data are linear separable, or if a linear border line provides enough accuracy in the final prediction. What happens if the feature space requires a more complex non-linear decision border? i.e., if the decision border is given considerably better by an hyper surface? One possible way to create non-linear models is to use *neural networks* (NN). A neuron is a computational unit, i.e., a piece of code that performs a single task or function, usually called the activation function. This method was developed as a gross mimic of a biological neuron, where each computational neuron has a set of wires connecting it with its input and a set of wires that are used to communicate its output to the next set of neurons. Each wire between two neurons has associated a parameter, sometimes called "weight", which is adjusted in the leaning process. So the computational model is created as an array of neurons, configured in layers that can be fully or partially connected. This kind of computational structure allows the creation of pretty complex nonlinear functions just by the selection of the network wires. After optimization, the NN computes a continuous function ready to be used for regression. For classification, a helper selection function can be implemented. A simple and yet powerful arrangement is the feedforward structure (see Figure 1), in which the neurons are arranged in a network where each layer receives its inputs directly from the layer before, and provides its outputs only to the layer after it, i.e., where the i-th layer receives the information from the (i - 1)-th layer, and sends its output to be the inputs to the (i + 1)-th layer. A common neural network used for classifiers is a three layer fully connected network, i.e., each neuron receives information for any neuron in the layer before. The first layer, called input layer, has a computational unit for each attribute in the feature vector. This unit sends its associated value to any neuron in the second or hidden layer, where each neuron computes a sigmoid function

$h_{\theta_i}(x)=g\left(\theta_i^T x\right)$ as in Eq. (4). θ_i is the parameters vector, containing the weights for each input wire to neuron i. Each neuron in the hidden layer sends its output to any neuron in the third or output layer. And finally, the neurons in the output layer computes another logistic function $h\left(\theta_j^{'T} z\right)$, where z is the vector of outputs from the second layer and $\theta_j^{'}$ is the vector of weights for each input wire to the output neuron j. $h\left(\theta_j^{'T} z\right)$ corresponds to the probability for the input feature vector to belong to the class j. Then, a selector function chooses the class with the highest probability to be assigned as the final output of the classifier.

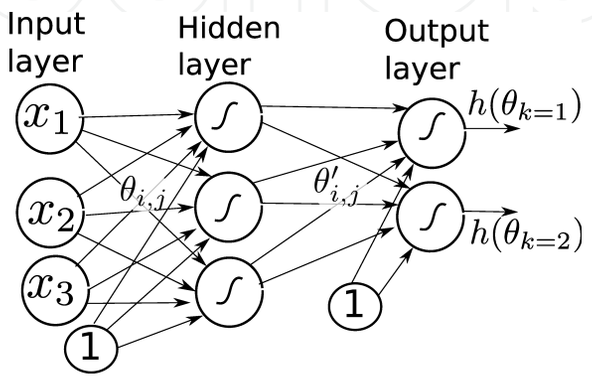

Figure 1. A schematic figure of a feed-forward three layers neural network. Here, an input vector of three features is classified in one out of two known classes. The hidden and output layers compute logistic functions in each neuron, represented by the s-like curve. $\theta_{i,j}$ is the weight of the i-th wire to the neuron j.

Any neuron in the second and third layer has an associated vector of parameters that need to be trained. The training of a NN is a difficult task, where the weights connecting each pair of neuron must be learned for all neuron in any layer. For the output layer, the cost function can be computed taking into account the expected values in the training set. But for the inner layers, no expected value is known. As a consequence of this, the cost function associated to a NN is in general a non-convex function. This has strong repercussions in the optimization problem. Due to the existence of several local minimum, the convergence to a global minimum is not guaranteed.

The variational parameters have several ways to be changed that will provide approximately the same level of correction from one iteration to the next. One strategy to find the "right direction" to move the network is to take minimal changes, i.e., from all the possible variation of parameters providing the same level of correction, the network is changed in the way that the set of parameters defers the less from its previous state. This is done by applying a generalization of the gradient descent method to deal with multilayer networks, called the back propagation algorithm, the complete description of which will be found elsewhere [17].

2.1.4. Support vector machine

Another method to create a nonlinear classifier is the *support vector machine* SVM. To illustrate the idea behind SVM, consider a two class problem. The main objective is to draw the decision

http://dx.doi.org/10.5772/61293

border between two classes that provides the best separation of sub sets among all the possible border lines (see Figure 2).

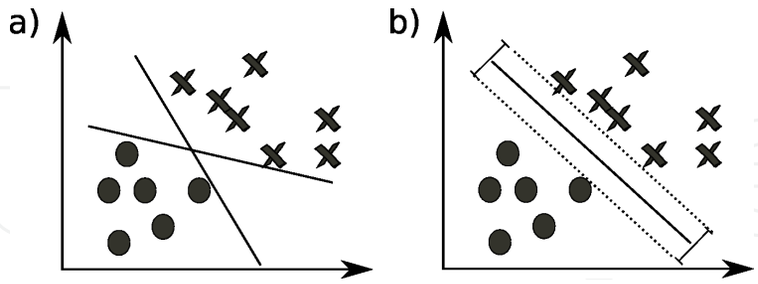

Figure 2. Among all the possible border lines, SVM chooses the one that provides the maximum margin between the classes. Here, two classes, marked with a circle and an x in a two-dimensional feature space are shown. a) Some possible non-optimal border lines. b) The right, the maximum margin border. Margins are shown in dots.

SVM finds the border line that has the largest margin or distance from the closest sample of each class. The equation of the decision border is then the equation of the hyperplane:

$$W^T x + b = 0 \tag{9}$$

W is a vector perpendicular to the hyperplane. x is a vector in feature space and b is a bias term. In this part, it is convenient to split the optimization parameters θ in W and b, to explicitly take the interception term apart from others. Now propose the hypothesis function:

$$h(x) = g\left(W^T x + b\right); \tag{10}$$

$$g(z) = \begin{cases} 1 if\, z \geq 1, \\ -1 if\, z \leq 1, \end{cases} \tag{11}$$

and the optimization problem is to find the optimal values for W and b that maximize the margin size. Starting from the Lagrangian

$$\mathcal{L}(W,b,\alpha) = \frac{1}{2} W^T W - \sum_{i=1}^{m} \alpha_i \left[y^{(i)} \left(W^T x^{(i)} + b \right) - 1 \right], \tag{12}$$

which includes the Lagrange multipliers α_i to hold the restrictions imposed by $g(z)$ in Eq. (11). The optimality conditions for the objective function are found by differentiating Eq. (12) against W and b and setting it equal to zero. It can be found that $W = \sum \alpha_i y^{(i)} x^{(i)}$ and

$\sum_{i}^{m} \alpha_i y^{(i)}=0$. Also, the multipliers vanish for all the feature vectors outside the margin lines, the remaining vectors are called support vectors and give rise to the method's name. After some algebra, the minimization problems turn out to be the maximization of the objective function [17]

$$\mathcal{J}(\alpha)=\sum_{i}^{m}\alpha_i-\frac{1}{2}\sum_{i}^{m}\sum_{j}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}K\left(x^{(i)},x^{(j)}\right), \tag{13}$$

under the constrains $\alpha_i \geq 0$ and $\sum_{i}^{m} \alpha_i y^{(i)}=0$. Here, the kernel function is just the inner product $K(x^{(i)}, x^{(i)})=(x^{(i)})^T x^{(i)}$ that corresponds to a linear classifier, i.e., when the two classes can be separated by a straight line. In the case that the data are not linear separable, SVM can become non-linear by simply replacing the kernel by a non-linear one. If the kernel represents the inner product of two vectors in the feature space, what does it mean a non-linear kernel function? When changing the kernel, a representation of the feature space in a higher dimensional space is obtained, related to the original feature space by some nonlinear transformation than is not necessary to know. The only thing required is the form of the inner product in the new coordinates expressed in terms of the original ones. This "kernel trick" allows the computation of pretty complex decision borders. However, not any function of two features constitutes a valid kernel. To solve this situation, a special case of Mercer's theorem [18] guarantees the validity of a kernel function, as far as the kernel matrix $k_{i,j}=K(x^{(i)}, x^{(j)})$ is a symmetric positive semi-definitive matrix.

In real application, it is common to find that the training set at hand is not separable, i.e., it is not possible to find a border hyper surface that splits the feature space without misclassification of some training samples. And forcing the model to fit any training vector will produce high bias. Then it is a good practice to implement *regularization*. This is done by introducing the regularization term $C\sum_{i}^{n} \xi([\theta]_i)$ in the cost to minimize. Here $[\theta]_i$ is the i-th component of the parameter vector, ξ is a penalty function which accomplish $\xi(\theta)\geq 0$ and C is the regularization parameter set by the user. C allows to control how much bias is acceptable for the final model.

2.1.5. Confusion matrix

To assess the accuracy of the ML model a *confusion matrix* for our classifier can be build. The confusion matrix is an evaluation of how many feature vectors in the training and/or testing set are misclassified. The matrix is built by contrasting the predicted class flag with the real one for each feature vector. For example, consider a two class problem (like the benign or malign tumor problem) that when trained against a set of 100 feature vectors report the classification presented in table 1:

Predicted / Real	Benign	Malign
Benign	45	12
Malign	15	25

Table 1. Example of a confusion matrix for a two classes clasiffier based on imaginary benign or malign tumor data.

In the example, 45 benign tumors have been classified as benign by the ML, and 15 have been misclassified as malign tumors. The error estimate for a ML code is computed as the average of miscalculated classes over the total of samples. It is the sum over all the off diagonal elements over the total number of cases. In the example, the error range of 30%. And the accuracy, defined as $1 - err$ is 70%.

2.2. Unsupervised learning

In this case, the first objective of the ML is to find some similarities among the data, which can be used to divide it into clusters. Each cluster will then define a class, and new inputs to the machine (outside the training set) will be classified following the clustering of the training space.

2.2.1. Clustering

So far, the feature vector is represented as a set of numerical values in real space. From the mathematical point of view, each input "box" inside a vector is called a dimension and the value in the box is a coordinate. The length of the vector is the number of dimensions of the containing space. A space is a collection of vectors that follow a set of rules (an algebra). If a distance measure for any two points in the space (features vectors) can be build, then the data can be clustered [19]. A common choice is the Euclidean distance, defined as the square root of the sums of the squares of the differences between the coordinates of the two vectors in each dimension, i.e., if **A**, **B** are vectors in $\mathbb{R}^n$, the Euclidean distance between them is given by:

$$\|A-B\| = \sqrt{\sum_{i=1}^{n}\left(x_{A,i}-x_{B,i}\right)^2} \tag{14}$$

2.2.2. K-means

A common clustering algorithm known as K-means is as follows: chooses a number of clusters k to be found, and initialize the clusters centroids μ_1, μ_2,...,μ_k randomly. Then assign each one of the training feature vectors to a cluster by relating it to the closest centroid (the one which has the minimum distance to the sample). When all vectors in the training set are labeled, recompute the position of the centroids as the average of the feature vectors inside the cluster, for each cluster. If there are centroids without any feature vector assigned, it can be dismissed or repositioned randomly. Now iterate again relabeling the training set and recomputing the centroids positions until convergence is achieved.

2.2.3. Hierarchical clustering

It is a quite expensive algorithm to obtain clusters which is based on finding a partition hierarchy among the data. It can be started by making each feature vector a cluster with one single member. Then, the distance between any pair of vectors is computed. If the distance is lesser than some selection parameter, the clusters are mixed to form one. In the new distribution of clusters, each cluster is represented by its centroid, and iterates the process until some convergence criteria is achieved. For example, a predefined number of clusters is reached. Due to the computational cost involved in hierarchical clustering, it is not recommended for problems with large training set.

2.2.4. The CURE algorithm

This is a large-scale-clustering algorithm. When centroids are used to define clusters, it is expected that any cluster would have a regular shape in features space, and the space is expected to be Euclidean. The clustering using representatives (CURE) algorithm is a little more general, due that it can handle irregular shaped clusters. This method defines a cluster in terms of a set of representative members of the cluster. These representatives must be chosen in a way that they are as far as possible from each other. Then, the representatives are points on the "surface" of the cluster. This kind of construction allows any shape for the cluster, including rings. To apply the CURE algorithm, first an initial clustering must be done, then the representatives are chosen for each cluster, and finally, two clusters are united if they have a pair of representatives that are close enough following some user-definedcriteria.

3. LCI experiments and data extraction

3.1. Experiment description

3.1.1. Materials and methods

3.1.1.1. Cell lines expressing fluorescent mithondria (Vero-Mito)

The Vero epithelial cells (ATCC) were maintained under standard culture conditions as described in other works from this lab [20-21], and in another chapter of this book [22]. The temperature was set at 37°C in a humidified atmosphere of 95% air and 5% carbon dioxide. The monoclonal cell line over-expressing the plasmid pmKate2-Mito (Evrogen®) were obtained with a cell sorter (Moflo XDP, Beckman Coulter®), with ulterior antibiotic selection (Kanamycin) of transfectants during 21 days in accordance to the experimental procedures described in detail in [23].

3.1.1.2. Virus preparation, titration and infection protocols

The strain New Guinea of Dengue Virus Serotype 2 (DENV-2) was grown and maintained in insect cells C6/36 HT under the standard practices as described in [20, 21], and this book [22].

Briefly the DENV were amplified at a very low MOI (multiplicity of infection) to avoid genetic drift and apparition of DIs (defective interfering particles), which could be altering the whole data concerning the real synchronized infections [24]. Viral titers were detected by plaque assay, using a Vero cell monolayer culture under 1% methylcellulose overlay medium as it was reported by [20, 21]. The viral infections were done by the same way of our previously reported works [20, 21], with the difference that for live cell imaging the cells were seed and registered in 35-mm glass bottom dishes (MatTek Corporation) with 0.7 mm in thickness of the glass coverslides, which is adequate in refraction index for this kind of inverted confocal microscope for registering living cells. The negative controls of infections were named mock infections, as it had been standardized traditionally for the virology community [24].

3.1.1.3. Live cell imaging

The Vero-Mito (3x105 cells) cell line was seed in Petri dishes adequate for living cells with bottom with coverslide of 0.17 mm, and previous to the register the normal culture medium used was changed by a DMEM without red phenol for avoiding the autofluorescence of this pH-indicator chemical. The videos of living cells over expressing fluorescent mitochondria (+/- infections) were obtained with a confocal microscopy based on disk spinning Unit (Olympus®IX-81 DSU), coupled to incubator and mixing gases Tokai-Hit Co® systems, which regulate the micro environment of cell culture with temperature and carbon dioxide in all system. The mock infections and infections of the overnight micrographs were captured in an OrcaR_2CCD (Hamamatsu®) ultra-refrigerated camera with electro multiplier, coupled to the illumination systems with Arc burners of 150 W constituted by mercury-xenon or xenon lamps (Olympus®-MT10 Illumination System). The photonic signals emitted by the biological specimen were transduced to electromagnetic waves for the CCD (charge coupled device), transmitted by a light fiber 2 m of single quartz to the Workstation Xcellence-Pro (Olympus®) for image processing, which also include the application of deconvolution tools for improving the signal/noise ratio of images.

3.2. Image segmentation and extraction of information

A total of nine videos generated by LCI where studied. Four of them correspond to mock cells (uninfected) and five to infected cells. Each video has 36 frames taken each 20 min for a period of 12 h. The videos are recorded in color at 1024x1344ppi resolution. As the color has no relevant information, they have been converted into gray scale. The last 34 vertical lines of each frame were dismissed in order to get rid of the microscope watermark. Then the resolution has been decreased to 495 x 672. The cells in this study have been selected under the following criteria:

- The whole cell is present in any frame of the video and the nucleus was clearly distinguishable on any frame.
- There were not other cells too close, so the mitochondrial distribution does not seem to be overlapped with those from neighbor cells.

Under this procedure, 11 cells were selected, 4 mock and 7 infected. Any selected cells were modeled as having an elliptical nuclei, by manually choosing four points on the nuclei

borderline and applying the Hough transform [25] (see Figure 3 and 4). The nucleus and the cell are proven to be approximately aligned [26] so the nuclear envelope is approximated by another ellipsis centered in the nuclei, with the same inclination and the axis twice as long. This rudimentary model provides us with the necessary segmentation to perform the cell tracking by simply creating a mask over the region where the ellipsis is located for any frame in the video. The images are stored as intensity matrices where each entry is a pixel. Masks are stored as matrices of the same dimension, whose entries take values of zero or one. If the pixel belongs to the segmented region the associated mask value is one. Those matrices are stored in binary format to be processed in a Python script that takes advantage of numpy and Scipy libraries for further analysis.

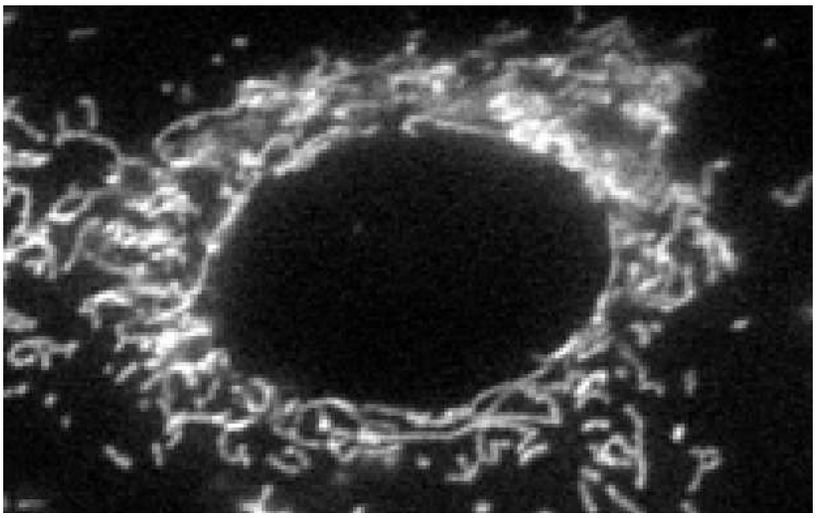

Figure 3. An infected cell appearing in the original image present on video.

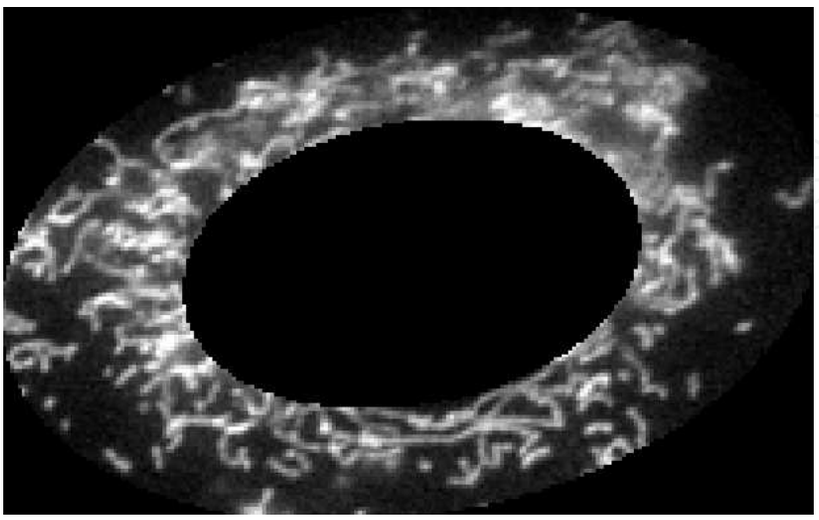

Figure 4. The segmented region used to track and study the cell. Nucleus is modeled as having an elliptic shape. The exterior membrane is modeled as a concentric ellipsis with the major semi axis twice as long as the nucleus.

http://dx.doi.org/10.5772/61293

3.3. Information processing

In the gray-scale video, the mitochondrial distribution is shown as bright points, been more brilliant those places where the density of mitochondrium is higher. Then, the density distribution of mitochondrium can be estimated as proportional to the intensity distribution $\rho(x,\ y,\ t) = \alpha I(x,\ y,\ t)$. In each frame of each video, a discrete set of pixel intensity values is recorded. This means that the intensity distribution in a discrete grid of point $(X,\ Y)$ has been measured, where $x \in X$ and $y \in Y$ are the sets of pixels coordinates.

The shape of the density distribution of mitochondrium is the same shape of the intensity distribution. Those functions differ only by a constant of proportionality that became irrelevant when the density function is normalized. So further in this reading both functions would be referred indistinguishable as $\rho(x,\ y,\ t)$.

The continuous density function can be approximated from the set of pixel intensity measurements by some interpolation method. In this work, a two-dimensional interpolation in terms of bivariate splines has been used on each segmented frame. This procedure allows us to extract important information about the mitochondrial behavior. Each frame is taken after a fixed period of time of 20 mins, so they form a time series of the density distribution function $\rho(x,\ y,\ t)$ through the whole experiment.

4. Results and discussion

In cell biology studies live cell imaging is a newcomer; however, the innovation in computational biology tools is been forced by the convergence of distinct research programs. Being so, LCI is no longer only a "technique" but a new exploratory science [27], that brings the possibility of encompassing cross-disciplines. In this sense, the subcellular patterns of distinct cellular organelles and macromolecular structures within the cell are important for dynamical studies, which will be useful in predictive medicine [28].

The mitochondrial morphology is a remarkable area for biomedical research since more than a decade [29], because these cellular organelles change under physiological and pathological conditions, like metabolism, thermogenesis, homeostasis of calcium and several kinds of cell death [30, 31]. But there is lacking information about the subcellular distribution of mitochondrium after a cell injury like the viral infections, and this quantitative information is key for tracking some cellular events of virus cycle that have been covered to the computational cell biology exploration.

New developments have been focused in the high-resolution microscopy images of the fine morphology of mitochondria [32]. Having in mind the improved time resolution, this information is decisive for understanding of the dynamics and functioning of these cellular organelles at high-throughput screenings [33].

But here, the work was mainly directed to study and characterize the subcellular distribution of mitochondria with and without Dengue virus infections on epithelial cells that are constitutively expressing these organelles in red fluorescence.

Recently, it had demonstrated that not only shape, number and size of the organelles are important for the cellular function, but also their subcellular distribution, which is the consequence of the intracellular transport [34]. Since Dengue virus like many other members of the most diverse viral families are using the cytoskeleton [22], here we have tried to follow indirectly the infection process using the alterations of subcellular distributions of mitochondria.

Both mock and DENGV infected cells have been prepared by a standardized procedure that provides approximately the same initial state among all cells of each type even when different experiments are considered. So at frame 0 each of the studied cells provides a possible initial state. Each of these states must adjust to the density distribution of mitochondrium. Assuming that any possible initial state is equally probable, then our approximation to the density function at time zero $\rho(r, \theta, t = 0)$ must come from the average over all the known (measured) possible states. Each frame has the same time spacing for all the videos (20 minutes), so the same analysis is valid for the i−th frame and the mitochondria density distribution at time t would be the average over all the known states for all cells of the same kind.

The distribution as a function of two variables given by the pixel position related to the center of the cell is $\rho(x, y)$. However, before taking the average, it has to be taken into account that a single cell can have any random orientation. The elliptical shape of the nucleus makes the distribution not symmetrical. This means that all function must be rotated to have the same orientation before been able to average over them.

A Python script was written to automatically determine the major semi axis of each cell in each frame by measuring the maximum distance between two points into the segmented region. The center of the ellipsis is found as the average of all the coordinates in the image. To get $\rho(r, \theta, t)$ for any cell referred to the same polar axis, and so be able to compute the density average, the polar axis is set equal to the major semi axis in each of the cells analyzed. In polar coordinates $\rho(r, \theta)$ will give us the density of mitochondrium at a distance r from the nuclei center and at an angle θ from the major semi axis.

The average density of mitochondrium distribution $\rho(r, \theta, t)$ is shown for three frames (t = 0, 320, 720 minutes) in Figures 5, 6, and 7. Those are the initial state, one intermediate state and the final state of the study. The vertical axis is the level of intensity (proportional to the mitochondrium density) and the horizontal axis is the distance to the nuclei center in pixels. In each subplot the projection of $\rho(r, \theta, t)$ for a given angle in radians is presented.

It can be seen that mock cells present prominent peaks for some radial positions, i.e., the distance between the peak and the closest local minimum is large compared with the background density. Also, the background density is low. This suggests that mock cells have the mitochondrium distributed in clusters around the nuclei. At variance, the infected cells average mitochondrium density distribution presents a higher background intensity compared with the maximum of the distribution. The peaks are less defined than those for mock cells, which means that the mitochondrium clusters are less defined or inexistent. And the mitochondrium tends to fill all the space available. Is also noticeable that the infected cells

http://dx.doi.org/10.5772/61293

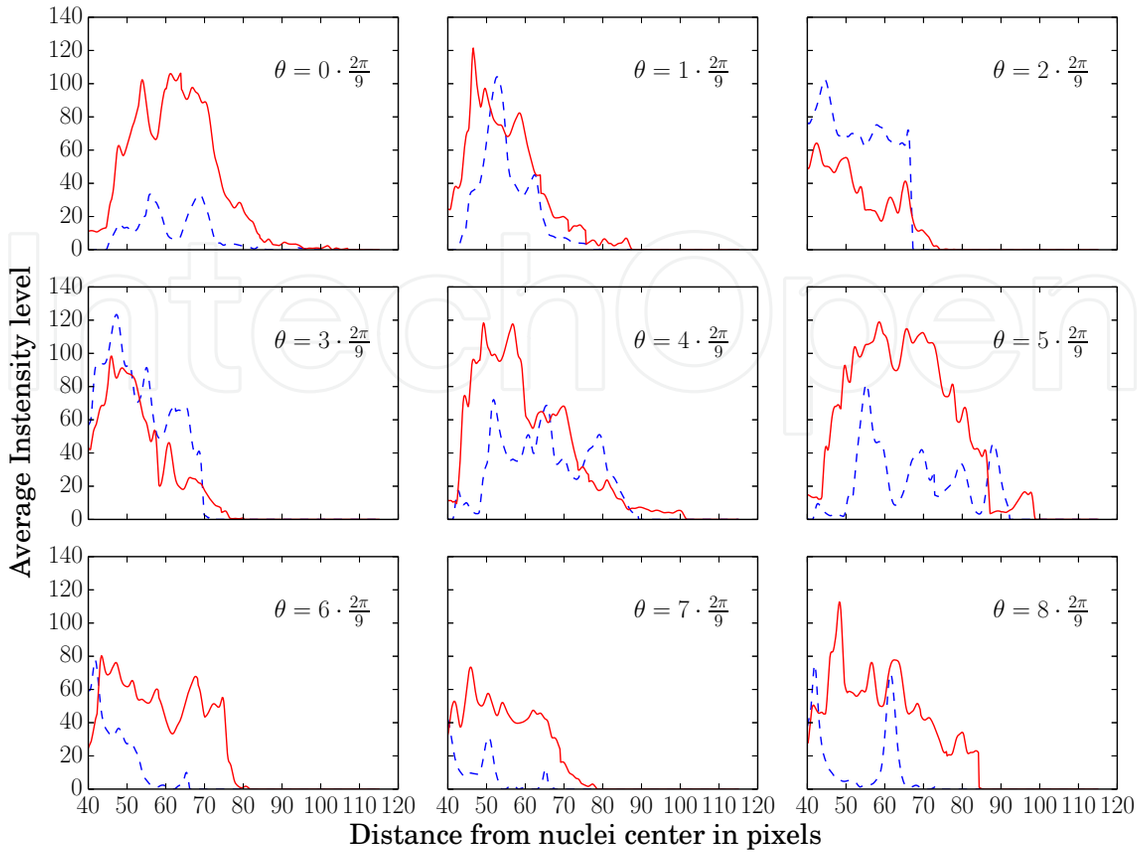

Figure 5. Average density of mitochondria $\rho(r, \theta, t)$, for $t = 0$ min. Mock cell mitochondrial density average is in blue (color online) dashed line and infected mitochondrial density is in red continuous line.

show more local maximum, which suggest that the distribution is somehow disorganized (more random).

These findings imply that a general structural change in mitochondrium distribution is caused by viral infection and it can be evidenced directly by examination of a cell's picture. The clustered behavior presented by mitochondrium on mock cells implies that they are grouped when normal function of the cell is in process. This is an organized distribution. On the other hand, the lack of clusters in infected cells shows that when infected the mitochondrium distribution became erratic, maybe random, which will be associated with a lack of organization.

A possible way to detect the presence of a viral infection will be to measure the level of randomness present in mitochondrium distribution. Remembering that $\rho(r, \theta, t)$ is experimentally measured through the pixel intensity in each video frame, it is found that the randomness in mitochondrium distribution will be the same than the randomness in pixel intensity distribution of the segmented image.

An image on gray scale is described digitally in terms of intensity values ranging from 0 (black) to 255 (white), a total of 256 possible shades of gray, each one of those possibilities is known as a level of intensity. A common descriptor used to classify a picture is the *entropy*

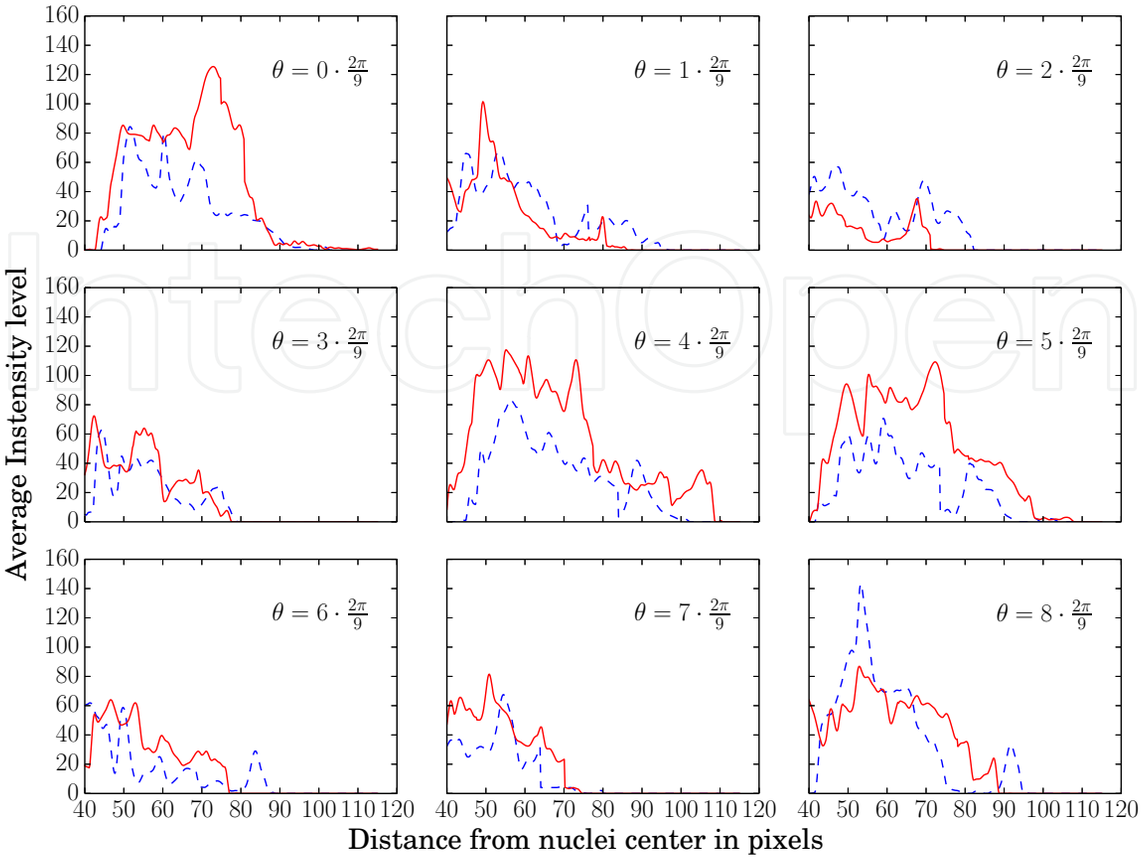

Figure 6. Average density of mitochondria $\rho(r, \theta, t)$, for $t = 320$ min. Mock cell mitochondrial density average is in blue (color online) dashed line and infected mitochondrial density is in red continuous line.

$$E = -\sum_{i=0}^{255} P(i) \log_2 \left(P(i) \right). \tag{15}$$

This is a measure of how "random" the levels of intensity are distributed on a gray scale picture. $P(i)$is the probability of finding the i−th level of intensity inside the image. As mock cells shows more order than infected cells in $\rho(r, \theta, t)$, it can be expected that for the entropy in mock cell's image to be lesser than the entropy in the image of an infected one.

Another commonly used descriptor for images is *uniformity*, defined as:

$$u = \sum_{i=0}^{255} P(i)^2, \tag{16}$$

which is a measure of how much the levels of intensity change through the image. This descriptor is maximum if all the image presents one single level, and decreases with the level changes. By carefully looking at Figures 5, 6, and 7 the reader will note that infected cells shows more oscillations in the density of mitochondrium (local peaks). In terms of pixel intensity, it

http://dx.doi.org/10.5772/61293

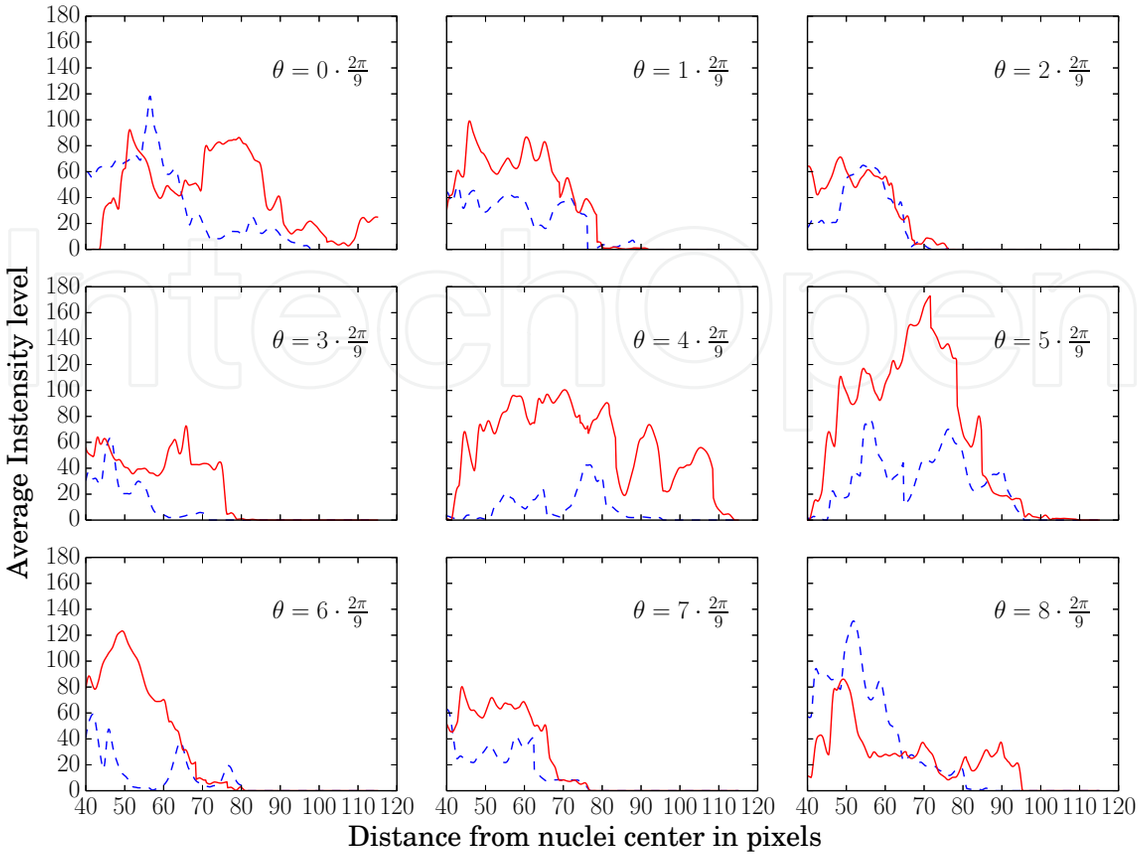

Figure 7. Average density of mitochondria $\rho(r, \theta, t)$, for t = 720 min. Mock cell mitochondrial density average is in blue (color online) dashed line and infected mitochondrial density is in red continuous line.

means that the intensity is changing more frequently, so the tone in the pictures is less uniform. Then it can be expected from the uniformity descriptor on the picture of an infected cell to be low.

In Figure 8, a plot on the uniformity vs entropy parameters space shows the computed values for those image descriptors for all studied cell in all frames. It can be seen that the mock and infected cells occupy mainly different regions on parameter space. So, these descriptors constitute a promising candidate to be a feature vector (or a part of it) in a machine learning code designed to classify infected cells.

5. Conclusion

A detailed analysis of the mitochondrium distribution around the nuclei for seven infected cells and four mock cells in nine videos has been performed. The study shows that mock cells clusters its mitochondrium and present an organized distribution in space. The organized character of the mitochondrium density distribution is maintained through time. At variance, infected cells loose these organized characteristics and the distribution of mitochondrium become erratic. This suggests that the mitochondrium are clustered when the healthy cell is

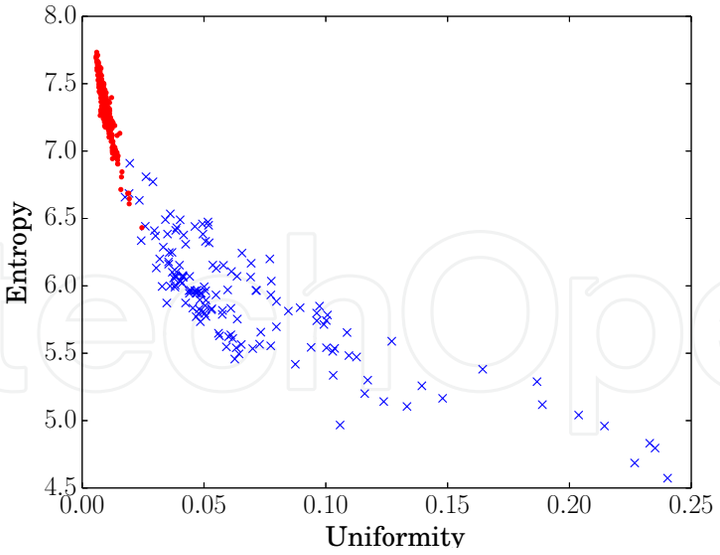

Figure 8. (Color online) Parameter space graph for Mock cell (blue x) and infected cells (red dots). Each marker represents the value for a single cell in a single time instant.

performing its natural process. But when a DENV infection is affecting the cell, those natural process are interrupted and it is reflected in the way how mitochondrium behaves. From this analysis, two image attributes are found to be suitable to be used as *features* in a ML classifier between infected and mock cells. These features are simple common image processing descriptors: *entropy* and *uniformity*, whose computation is easy and fast. Entropy is related with the randomness presented in the gray tones of the image and uniformity is related with the prevalence of a single gray tone. Both image attributes in a LCI photograph taken over a cell culture prepared with coloured cells are directly related with the mitochondrium density distribution behavior $\rho(r, \theta, t)$. The presence of clusters in mock cells and the softer $\rho(r, \theta, t)$ behavior are translated to higher values of uniformity and lower values of entropy image descriptors than those present in infected cells images.

Acknowledgements

This research was supported by COLCIENCIAS grant 111554531592 from the Colombian government. JCGG was the recipient of a Full-Time Professor Program (Exclusive Dedication) for the Medicine Faculty at University of Antioquia for 2014–2015.

Author details

Juan Carlos Cardona-Gomez[1*], Leandro Fabio Ariza-Jimenez[2] and
Juan Carlos Gallego-Gomez[2*]

*Address all correspondence to: juanc.gallegomez@gmail.com; jccg77@gmail.com

http://dx.doi.org/10.5772/61293

1 Optical Spectrometry Group, Faculty of Basic Sciences, Universidad del Atlántico, Barranquilla, Colombia

2 Translational and Molecular Medicine Group, Medellín Medical Research Institute, Faculty of Medicine, Universidad de Antioquia, Medellín, Colombia

References

[1] Brandenburg B., Zhuang X. Virus trafficking – learning from single-virus tracking. Nat Rev Microbiol. 2007;5 (3):197-208. DOI: 10.1038/nrmicro1615

[2] Lippincott-Schwartz J., Patterson G. H. Development and use of fluorescent protein markers in living cells. Science. 2003;300 (5616):87-91. DOI: 10.1126/science.1082520

[3] Frigault M.M., Lacoste J., Swift J.L., Brown C.M. Live-cell microscopy – tips and tools. J Cell Sci. 2009;122 (6):753-767. DOI: 10.1242/jcs.033837

[4] Wyckoff J., Gligorijevic B., Entenberg D., Segall J., Condeelis, J. High-resolution multiphoton imaging of tumors in vivo. In: Spector D.L., Goldman R.D., ed. Live Cell Imaging: A Laboratory Manual. 2nd edn. New York: CSHL Press, Cold Spring Harbor; 2010. p. 441-461.

[5] Lacey A.J., ed. Light microscopy in biology: A Practical Approach. 1st ed. IRL Press; 1989. 348 p.

[6] Duda R.O, Hart P.E., Stork D.G. Pattern Classification. 2nd ed. Wiley-Interscience; 2000. 680 p.

[7] Webb A.R. Statistical Pattern Recognition. 2nd ed. Wiley; 2003. 514 p.

[8] Bishop C. Pattern Recognition and Machine Learning. 1st ed. New York: Springer; 2007. 738 p.

[9] Mitchell T.M. Machine Learning. 1st ed. New York: McGraw-Hill; 1997. 414 p.

[10] Dorigo M., Schnepf U. Genetics-based machine learning and behavior-based robotics: a new synthesis. IEEE Trans Syst Man Cybernetics 1993;23 (1):141 154. DOI: 10.1109/21.214773

[11] Ball N.M., Brunner R.J. Data mining and machine learning in astronomy. Int J Mod Phys D. 2010;19 (07):1049-1106. DOI: 10.1142/S0218271810017160

[12] Clark M. Application of machine learning principles to modeling of nonlinear dynamical systems. Proc Arkansas Acad Sci. 1994;48:36-40.

[13] Neumann B., Held M., Liebel U., Erfle H., Rogers P., Pepperkok R., et al. High-throughput RNai screening by time-lapse imaging of live human cells. Nat Meth. 2006;3 (5):385-390. DOI: 10.1038/nmeth876

[14] Jones T. R., Carpenter A. E., Lamprecht M. R., Moffat J., Silver S. J., Grenier J. K, et al. Scoring diverse cellular morphologies in image-based screens with iterative feedback and machine learning. Proc Nat Acad Sci. 2009;106 (6):1826-1831. DOI: 10.1073/pnas.0808843106

[15] Carpenter A E, Jones T.R., Lamprecht M.R., Clarke C., Kang I.H., Friman O., et al. CellProfiler: image analysis software for identifying and quantifying cell phenotypes. Genome Biol. 2006;7 (10):R100. DOI: 10.1186/gb-2006-7-10-r100

[16] Tarca A L, Carey V.J., Chen X.-w., Romero R., Drăghici S.. Machine learning and its applications to biology. PLoS Comput Biology. 2007;3 (36):e116. DOI: 10.1371/journal.pcbi.0030116

[17] Haykin S. Neural networks: a comprehensive foundation. 2nd ed. Upper Saddle River, New Jersey: Prentice-Hall; 1999. 842 p.

[18] Mercer J. Functions of positive and negative type, and their connection with the theory of integral equations. Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character. 1909;209:415-446.

[19] Rajaraman A, Ullman J.D. Mining of Massive Datasets. 1st ed. New York: Cambridge University Press; 2011. 326 p.

[20] Martínez-Gutierrez M., Castellanos J.E, Gallego-Gómez J.C. Statins reduce dengue virus production via decreased virion assembly. Intervirology. 2011;54 (4):202-216. DOI: 10.1159/000321892

[21] Padilla-S L., Rodríguez A., Gonzales M.M., Gallego-G, J C., Castaño-O, J.C. Inhibitory effects of curcumin on dengue virus type 2-infected cells in vitro. Arch Virol. 2014;159 (3):573-579. DOI: 10.1007/s00705-013-1849-6

[22] Orozco-García E., Trujillo-Correa A., Gallego-Gómez J. Cell biology of virus infection: The role of cytoskeletal dynamics integrity in the effectiveness of dengue virus infection. In: Najman S, ed. Cell Biology. 1st ed. InTech; 2015.

[23] Acevedo-Ospina H. Biología celular de la infección por virus dengue: Relación entre las mitocondrias y los virus [thesis]. Medellín, Colombia: Institute of Biology, University of Antioquia; 2014.

[24] Mahy B.W.J. The Dictionary of Virology. 4th ed. Academic Press; 2008. 520 p.

[25] Duda R.O., Hart P.E. Use of the Hough transformation to detect lines and curves in pictures. Communi ACM. 1972;15 (1):11-15. DOI: 10.1145/361237.361242

[26] Zhao T., Murphy, R.F. Automated learning of generative models for subcellular location: building blocks for systems biology. Cytometry Part A. 2007;71A (12):978--990. DOI: 10.1002/cyto.a.20487

[27] Wang Y.-l. Hahn K.M., Murphy R.F., Horwitz A.F. From imaging to understanding: frontiers in live cell imaging, Bethesda, MD, April 19–21, 2006. J Cell Biol. 2006;174 (4):481-484. DOI: 10.1083/jcb.200607097

[28] Tárnok A., Mittag A., Lenz, D. Clinical cytomics. In: Imaging, Manipulation, and Analysis of Biomolecules, Cells, and Tissues IV; February 21, 2006; SPIE Proceedings; 2006. p. 60880K-60880K-12. DOI: 10.1117/12.645024

[29] Kuznetsov A.V., Usson Y., Leverve X., Margreiter R. Subcellular heterogeneity of mitochondrial function and dysfunction: Evidence obtained by confocal imaging. Mole Cellular Biochemi. 2004;256-257 (1-2):359-365. DOI: 10.1023/B:MCBI.0000009881.01943.68

[30] Leonard A P, Cameron R.B., Speiser J L., Wolf B.J., Peterson Y.K., Schnellmann R.G., et al. Quantitative analysis of mitochondrial morphology and membrane potential in living cells using high-content imaging, machine learning, and morphological binning. Biochimi et Biophys Acta (BBA) - Mole Cell Res. 2015;1853 (2):348-360. DOI: 10.1016/j.bbamcr.2014.11.002

[31] Zhan M., Brooks C., Liu F., Sun L., Dong Z. Mitochondrial dynamics: regulatory mechanisms and emerging role in renal pathophysiology. Kidney Int. 2013;83 (4): 568-581. DOI: 10.1038/ki.2012.441

[32] Lihavainen E., Mäkelä J., Spelbrink J N, Ribeiro A S. Mytoe: automatic analysis of mitochondrial dynamics. Bioinformatics. 2012;28 (7):1050-1051. DOI: 10.1093/bioinformatics/bts073

[33] Reis Y., Bernardo-Faura M., Richter D., Wolf T., Brors B., Hamacher-Brady A., et al. Multi-parametric analysis and modeling of relationships between mitochondrial morphology and apoptosis. PLoS ONE. 2012;7 (1):e28694. DOI: 10.1371/journal.pone.0028694

[34] van Zutphen T., van der Klei I.J. Quantitative analysis of organelle abundance, morphology and dynamics. Curr Opin Biotechnol. 2011;22 (1):127-132. DOI: 10.1016/j.copbio.2010.10.015

IntechOpen