End-of-course internship

Cell cycle analysis by live imaging

ENSAE Tutor : M. Pontil.
Internship supervisor : T. Walter

Peter Naylor

June 8th to October 31 2015

Etude du cycle cellulaire par l'imagerie.

Note de synthèse

J'ai étudié l'apprentissage statistique et les algorithmes d'apprentissage automatiques pendant ma dernière année à l'ENSAE ParisTech. Je souhaitais appliquer mes connaissances dans le domaine académique et plus précisément en imagerie médicale. Thomas Walter, mon maître de stage, m'a donné cette opportunité avec ce stage de recherche au centre de biologie computationnelle (cbio). J'ai aussi pu travailler avec une thésarde du laboratoire. La problématique de mon stage était la suivante: Est-il possible de prédire les différentes phases du cycle cellulaire, qui sont *M, G1, S* et *G2*, à l'aide d'une histone fluorescente `H2B`? Ce marqueur est a priori seulement informatif pour les sous phases de la mitose.

Les données brutes sont des films acquis par des techniques de micro-scopie sur des cellules cancéreuses dans leur environnement. Le cycle cellulaire est composé des différentes phases de la mitose et des trois phases de l'interphase, *G1, S* et *G2*. Le but de ce stage est de différencier les différentes phases de l'interphase, en d'autres mots, est-il possible de trouver des règles de décision liées seulement aux variables prises sur l'image pour différencier les sous phases de l'interphase. Les variables peuvent être de natures très différentes, elles peuvent concerner la taille, l'intensité ou bien d'autres mesures telles que les variables de haralic. Nous avons 239 variables et 1272 individus labélisés, (un individu représente une cellule sur une des images). Nous extrayons les variables et les données à l'aide du logiciel *CellCognition*. Ce logiciel co-developpé par Thomas Walter, permet la segmentation des cellules sur l'image, et à partir de ces segmentations, il en extrait un vecteur de 239 variables. Les biologistes ont contribué à ce projet en annotant les phases des données d'apprentissage. Alice Schoenauer Sebag, une thésarde du laboratoire, a regroupé tous les individus, qui caractérisent une cellule à un instant donné, qui sont issus de la même trajectoire. Alice a pu assembler ces trajectoires par étude de leur mobilité.

Le modèle final peut être décomposé en trois sous modèles. Le premier est un modèle de prédiction binaire où l'enjeu est de prédire si la cellule est en phase de mitose ou non. La mitose est connue pour être facilement reconnaissable avec l'histone `H2B`. Les données pour le premier sous modèle sont les données brutes d'extraction de *CellCognition*. On avait besoin de ce modèle car nous avions initialement peu d'individus labélisés en mitose. Le deuxième classificateur n'essaie de prédire que les phases de l'interphase, c'est à dire *G1*, *S* et *G2*. Pour améliorer les performances pour ce modèle, nous procédons à une normalisation des données pour que celles-ci représentent plutôt leur évolution par rapport à leur donnée initiale. En d'autres termes, nous prenons en compte l'évolution des différentes variables vis-à-vis de leur première apparition après la mitose. Le dernier modèle est un modèle à correction d'erreurs qui va prendre en compte plus fortement l'aspect temporel entre les images. Notre modèle atteint un taux de bonnes prédictions de 86% en moyenne. On remarque cependant que notre modèle peine à différencier les phases *S* et *G2*. Cela peut paraître normal si on regarde les images brutes, car il est très difficile de différencier à l'oeil nu ces deux états avec l'histone `H2B` seulement.

Cet apprentissage avait pour but de prédire les phases des cellules dans la base de données MitoCheck, qui est un projet Européen sur des expériences "less-of-function" analysées par technique d'imagerie. En d'autres mots, nous avons plus de 200 000 vidéos d'imagerie cellulaire et pour chacun de ces films une certaine protéine a été fortement réduite. Cette procédure nous permet d'inférer la

fonction de cette protéine dans le cycle cellulaire. Si nous pouvions correctement pister et classifier les cellules, on pourrait inférer le rôle des protéines dans l'interphase. Cette base de données possède l'histone fluorescent `H2B` qui est très informative pour les phases de la mitose mais ne l'est pas a priori pour les phases de l'interphase. En l'absence de vérité de terrain, nous ne pouvons quantifier les performances de notre modèle. On essaie tout de même de les quantifier en analysant la taille des différentes phases cellulaires. Pour chaque trajectoire, on essaie de prédire à tout instant la phase de la cellule. La plupart des trajectoires sont inutilisables comme le classifieur n'a prédit qu'une classe. Cependant, les 10% des trajectoires validées ont un temps de phase *G1* proche des temps donnés dans la littérature biologique. De plus, pour améliorer le pouvoir de prédiction sur la base MitoCheck nous allons utiliser des techniques de transfert d'apprentissage via la pondération des individus. Le transfert d'apprentissage est une branche de l'apprentissage statistique qui relâche une des hypothèse clés : la distribution de la base d'apprentissage diffère de la base que l'on souhaite prédire. L'idée est de faire une sélection dans la base d'apprentissage des individus qui représentent bien la base que l'on souhaite prédire. Les individus qui représentent la base de prédiction seront sur-pondérés alors que ceux n'apparaissant pas dans la base de prédiction seront sous-pondérés. Deux approches de la même méthode furent essayées, l'une où nous choisissons qui on sur-pondère, dans ce cas là, la variété d'individus dans la base d'apprentissage est faible. Dans la deuxième, nous choisissons qui sous-pondérer. Cette approche a l'avantage de faire augmenter le nombre de trajectoires sur lesquelles on peut mesurer la taille de phase *G1*, 70 trajectoires ce sont rajoutées comparé à la méthode sans poids. En particulier, la base d'apprentissage est plus variée que dans l'autre cas de figure.

Notre modèle semble approprié pour séparer les phases *G1* et *S* mais peine à différencier les autres phases. Il est basé sur des forts a priori qui ne rendent pas le modèle flexible. En effet, le modèle à correction d'erreurs peut forcer des transitions de phases qui n'existent pas si par exemple une des cellules s'arrêtait subitement dans la phase *S*. Ces arrêts dans le cycle cellulaire sont très intéressants car ce sont eux qui permettraient d'inférer des informations sur les rôles des protéines. Avoir des vérités de terrain est vital pour pouvoir mieux quantifier les pouvoirs prédictifs de notre modèle, l'EMBL, Heidelberg est en train de collecter ces informations. Dans la continuité de ce travail, il serait intéressant de mieux regarder les différences sur les distributions entre les différentes bases de données. Il serait aussi intéressant d'appliquer des méthodes de transfert d'apprentissage via la transformation de variables et non par la pondération des individus comme il se peut que celle-ci ne soit pas adaptée. Mieux comprendre les différences sur les distributions entre les différentes bases permettrait de savoir si la méthode de transfert d'apprentissage via la pondération des individus est adaptée ou non dans notre cas.

Cell cycle analysis by live imaging

Summary

I have studied machine learning algorithms during my final year in ENSAE and wished to apply them to research in biology. Thomas Walter kindly gave me this opportunity by offering this research internship at the centre of computational biology (cbio). Working with another PhD student I was assigned a question, is it possible to predict cell cycle phases based only on a certain fluorescent marker, this marker however is meant to be uninformative for this type of prediction by biologists.

The raw data are films acquired by microscopy and contain cell cycle footage. The cycle goes though different mitotic phases and 3 non-mitotic phases, *G1, S* and *G2*. Our goal is be able separate each phase, in other words different we wish to find a set of rules based on image features that can differentiate two phases from each other. These features can be of very different nature, shape, intensity and some more difficult measures such as haralic features. We have 239 features and a set of 1272 labelled instances. The features and labels were generated by annotating and feature extraction program called *CellCognition*. *CellCognition* is software capable of segmenting cells on a picture; from this segmentation we extract 239 features. Biologist contributed by labelling some the segmented cells. This tool was co-developed by Thomas Walter. Alice Schoenauer Sebag, a PhD student at cbio, had clustered many of the instances by trajectories. Before this clustering, features of a certain cell were extracted into a data set with no relationship between them. Alice created them by tracking cells and studying cell mobility through the entire film.

The final predictive model can be decomposed in three separate entities. The first based on the raw extraction of features will predict if the cell is or is not in a mitotic phase. We needed this "pre-classifier" by lack of mitotic samples in our training set. It is known that this classifier is efficient to detect mitotic phases; the first prediction will take over the second if it predicts mitosis. The second classifier focuses on the interesting phases, *G1, S* and *G2*. To improve prediction rate, we try to erase all cell-specific attributes, such as initial size, by normalizing the data so that it represents the evolution compared to its initial state. This second classifier is followed by a correcting model that will take into account time dependencies created by Alice's cell tracking. This model yielded 86% accuracy rate. We notice that the two first phases seem differentiable whereas our classifier seems to have difficulties in finding differences between *S* and *G2*. But it can seem normal if you have a look at the raw imaging, it is hard to distinguish the different phases with your bare eyes.

This training had as an aim the prediction of the MitoCheck data set, a European project of less-of-function experiences of cell live imaging. In other words, we have 200 000 videos of cell imaging and for each film, a certain protein is strongly reduce. This procedure enables us to infer protein function within the cell cycle. If we would be able to track, and correctly predict the phases of the

cells for the non-mitotic phase, we could maybe infer information about the role of each protein. This data set has a `H2B` marker that is usually used for mitotic phase tracking, it is not meant to be informative for the non-mitotic phases. The only issue is that we have no ground truth about the MitoCheck data set. We try to assess the quality of our predictors by studying the lengths of the different non-mitotic phases. For most of the trajectories we only predict one class *G1*. However, the 10% that have a standard predictive cell cycle have literature-like values for cell cycle phase *G1* only. Moreover, we use transfer learning technics in order to improve accepted trajectory rate. Transfer learning is a branch of machine learning that relaxes one of the key assumptions; the distribution of the training set may differ from the distribution of the predictive set. The main idea is to do a selection of the training samples in order to only train on the instances that represent in some way the predictive set. The result of the transfer learning is a vector of weight corresponding to each different sample, we weight them correspondingly to their importance with respect to the predictive set. Two approaches were tested, both deriving from the same technic: kernel mean matching. In the first, we start with an empty training set and select the most important samples. In the second, we start with the whole training set and down weight the least important variables. The second improves considerably the number of accepted trajectories. The down weight might have had some positive effect whereas the first selected too few instances for training resulting in training over less than 50 different instances. The transfer learning increased the number of accepted trajectories by 70.

Our classifier seems to be well adapted for the prediction of *G1* but possesses strong priors, based on biological facts. The classifier would be better suited for our purpose if we could relax the priors on the phase transitions. This would prevent the hidden Markov model to force a transition phase if a cell had an unexpected stop in phase *S*, moreover, cells stopping in phases due to the lack of a protein is what we are interested in. Having some ground truth might be vital in order to assess how good our predictions are, it will be provided by EMBL, Heidelberg. Future work on this project implies checking for differences between the distributions of each feature in both respective sets, it may be difficult with the high number of features and may need some dimensionality reduction. We applied instance selection but it might be more interesting to apply feature transformation which is another type of transfer learning.

Table des matières

1 Acknowledgement

I would like to thank the whole of the Centre of Computational Biology (cbio) for welcoming me in their laboratory. Especially, I would like to thank Alice Schoenauer Sebag and Thomas Walter who helped me with my work throughout the internship. I would also like to give Thomas Walter a very big thank you for giving me this wonderful experience of working on very interesting and technical subjects.

2 Introduction

This project is situated in the field of functional genomics and bioimage informatics. Genomics is the discipline which studies the composition, regulation and evolution of genomes. Functional genomics in particular aims at unravelling the function that is encoded in parts of the genome : while many genomes have been sequences in the last decades, what the concrete role of many gene products is. The functional impact of genes can be studied with imaging approaches. Bioimage informatics is a branch of bioinformatics that aims at answering biological questions from image data as primary source of information. In this project, I use bioimage informatics methods in order to contribute to the field of functional genomics by identifying genes involved in the cell cycle.

The incentive for this internship arose from a Swedish laboratory, SciLifeLab, which had a particular interest in protein function within the cell cycle. SciLifeLab currently build an atlas resource on subcellular protein localization, i.e. they perform a series of experiments where they fluorescently label one protein at the time. By doing this they collect for each protein its localization pattern inside the cell. Such localization patterns can in principle be informative about protein function. Another and more direct way of probing the function is to perform a loss-of-function experiment. In the latter, we perform an experiment where a certain protein is strongly reduced within the cell. By observing the phenotype resulting from this reduction in protein level, one might infer the biological process this protein is required for. Among the investigated proteins, SciLifeLab has identified a subset of proteins, for which the localization pattern suggest a role in the cell cycle. In order to confirm this hypothesis, they could either perform a large number of loss-of-function experiments in which cell cycle defects can be probed, or they could compare their proteins to existing data informative about the cell cycle.

Such a data set does indeed exist. In [1], the authors performed a genome-wide RNA interference (RNAi) screen by live cell imaging. In this screen, they used HeLa cells stably expressing `H2B-GFP`, a marker which is informative about cell division. Consequently, this experiment has allowed them to identify many genes required for cell division, but it has also been shown that this data set of 200,000 videos can be informative on other fundamental processes. However, it remains unclear whether this data set can be used to analyse the cell cycle. The issue at hand is to determine whether or not the non-mitotic phases can be differentiated only with the `H2B` marker and if so, perform it to the whole MitoCheck database.

This report will be divided in 4 parts. The first part will explain the context and how the raw data is collected. The second will discuss the dataset and will formalize the issue at hand. I will then discuss the methods used and I will continue with the results. I will finally explain how I applied my work to the MitoCheck dataset and the state my work is currently in.

3 Cell cycle and the raw data

3.1 Cell cycle

The study of the cell cycle is important, because it will determine how quickly an organism can multiply. This rate will determine at what speed an organism will be able to replace damaged or dead cells. It is also important in cancer research : one of the main features of cancerous cells is uncontrolled proliferation and thus failures in cell cycle regulation. Furthermore, during the cell cycle, the genome of each individual cell is copied, and consequently, failures in the cell cycle regulation can lead to errors and ultimately to `genomic instability`, which means that the genome of cancer cells is heavily altered and the usual regulation mechanisms controlling cellular behaviour, such as growth, reproduction, mobility, speed, when the cell is not functional anymore. Understanding how the cell cycle is regulated therefore also provides us with information on how a cell can become cancerous.

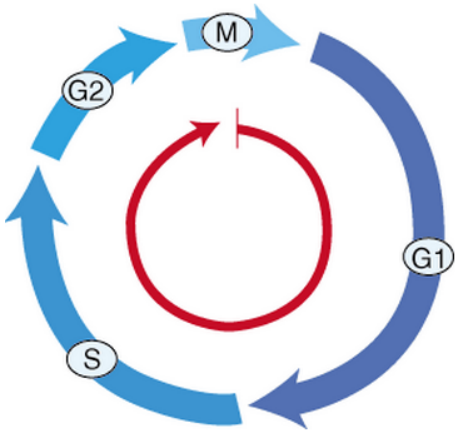


FIGURE 1 – Different phases in the cell cycle

The cell cycle is divided into four phases : *G1*, *S*, *G2*, *M* (see figure 1). The *G* in *G1* and *G2* stands for gap. During phases *G1* and *G2*, the cell prepares for the following step. The step *S* is when DNA is replicated and *M* is for mitosis, the phase where the cell completes cell division. The non-mitotic phases together (*G1*, *S*, *G2*) are called *interphase*. In this phase, the nuclear envelope is intact and the chromosomes (DNA molecules) are in their usual uncondensed state.

The reference data set this study will concentrate on was designed in order to study mitosis. The different mitotic phases were automatically identified in this data set by analysis of the image data. My work has been focused on non-mitotic phases. Concretely, I investigated the question of whether it is possible to distinguish the phases *G1*, *S* and *G2* (collectively *interphase*). As said, these phases are non-mitotic and so the cell trajectories that we will study have no nuclei division.

As we can see in figure 1, the mitotic phase only represents a small lapse of time in the whole cycle whereas the phase *G1* dominates the cycle. For human cells, interphase is 18 to 20 hours long and mitosis is about 2 hours, the length of these phases can vary from cell to cell. It goes without saying that these times are given on average. Also, the length of the cycle depends on the cell line.

3.2 Collecting the raw data

3.2.1 Recall of cell within structure

As stated in figure 2, gene expression results in the creation of a protein. Gene expression is the translation and transcription of a small part of the DNA, this part of the DNA is referred to as a gene. The first step, transcription, is a copy of the DNA that is being expressed to a RNA messenger. This messenger is then sent out of the nuclei to be translated into a protein.

As mentioned in the introduction, SciLifeLab and we use two complementary technics. The first, which is the work of SciLifeLab, is to hypothesis protein function based on its localization in the cell at a certain time of the cell cycle. The second, which was my focus during this internship, was to analyse loss-of-function experiments for a certain protein. Because of the numerous experiments realized, one cannot do this work manually. Our data set for instance is composed of 200,000 videos showing cells (60 on average) for 48 hours.

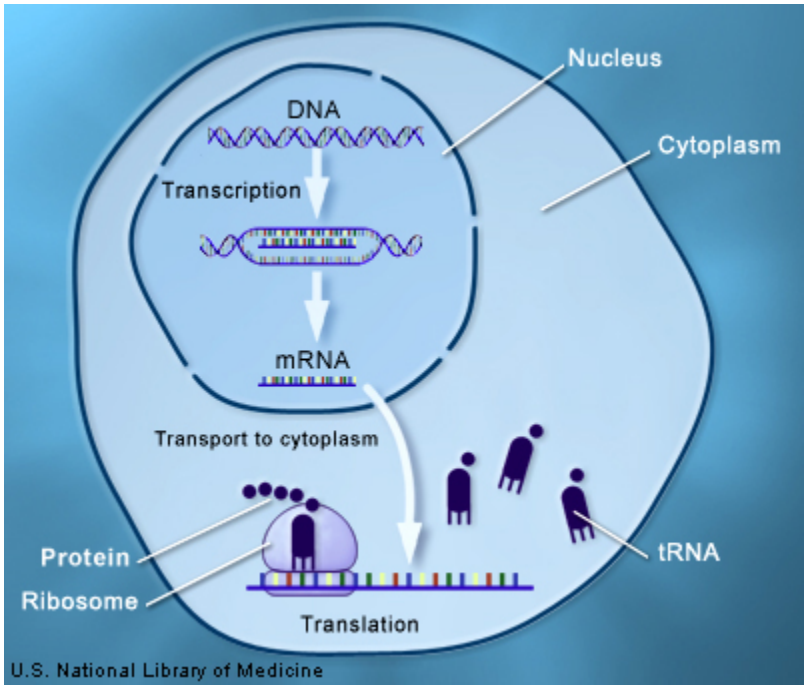


FIGURE 2 – From DNA to protein

3.2.2 Loss-of-function experiments

The rationale of loss-of-function experiments is to reduce the level of one single protein and to observe how this loss of one protein impacts the phenotype of the cell (with phenotype we mean an observable state of the cell). Such experiments can be performed in high throughput, i.e. we can do many experiments in parallel on automized systems. If in addition, we use microscopy allowing detailed observation at the single cell level, we talk about *High Content Screening (HCS)*,

meaning that we are acquiring data in a systematic way on automized systems, but still with a lot of information content. Thanks to RNA inference technology, we are able to suppress the RNA messenger that is expressed by the gene, deleting the later stops the generation of the protein. The implication of a loss-of-function experiment is that the protein's concentration is low but not non-existent. This concentration can sometimes be high at the beginning of the live imaging experiment and the cell cycle could still be temporary functional.

3.3 Raw data : Live cell imaging

The raw data are live imaging, and are recordings of loss-of-function experiments. The data is recorded by biologists and many experiments are performed simultaneously. Each experiment corresponds to one single well, and one plate typically contains 96 or 384 single experiments, as we can see in figure 3.

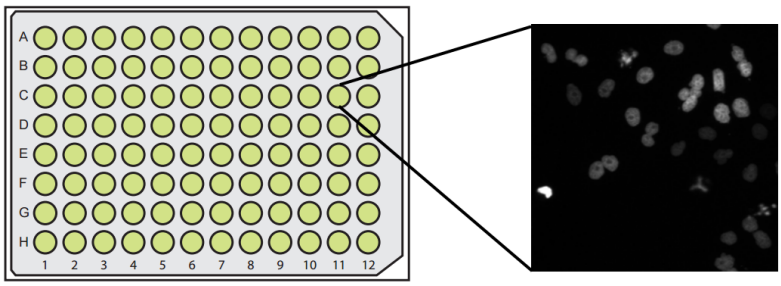


FIGURE 3 – A 96 well plate for High Content Screening

3.3.1 MitoCheck

The data on which we wish to classify cells is a published genome-wide dataset of time-resolved records of cellular phenotype responses to loss-of-function experiments. Nearly all proteins of HeLa cells were targeted, the dataset is available on their website [1]. Each plate contained 8 negative controls (scrambled : not targeting any gene) and 12 positive controls. In total, the data set contains data for 17 816 protein coding genes in 144 909 quality controlled time-lapse experiments.

HeLa cells are epithelial cancer cells which were derived from the adenocarcinoma of Henrietta Lacks in 1951. They are preferred by biologists as they are easy to grow, transfect and, because they are widely used, more comparable then other cells from one experiment to the other. They used widefield epifluorescence microscopy, 10x dry objective (0.645 μm / pixel), 1344x1024 pixel, 96 frames x 30 min = 48 hours.
The HeLa cells were image 18h after the transfection for 48h with a time-lapse of 30 min (Plan10x, NA 0.4, Olympus). Imaging chambers were sealed during imaging.

1. `www.MitoCheck.fr`

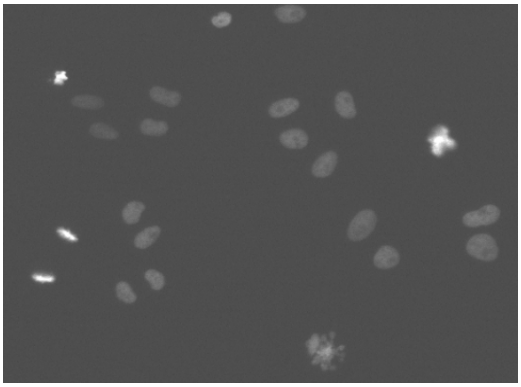


FIGURE 4 – Emission of the H2B marker

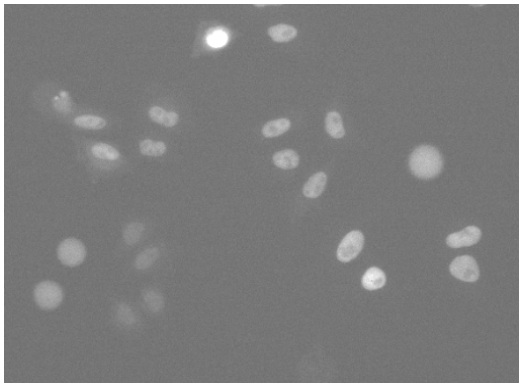


FIGURE 5 – Emission of the PCNA marker

Data acquired by Michael Olma, plate LT0001_01, well position 0015, frame number 60

3.3.2 PCNA data set

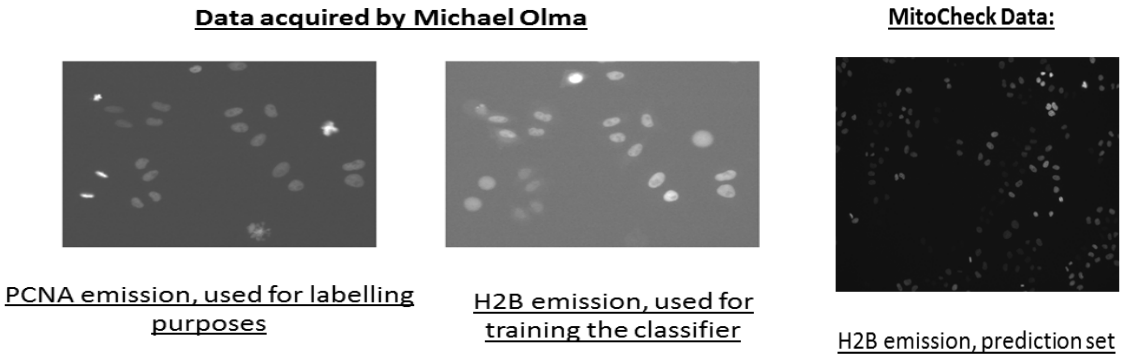
The training data I have been working on was acquired by Michael Olma (Peter group, ETH Zurich) at a Molecular Devices ImageXpress Micro with the PlateScanPackage. They used widefield epifluorescence microscopy, 10x dry objective (0.645 μm / pixel), 1392x1040 pixel, 482 frames x 5.9 min = 47.4 hours. The HeLa cells were imaged for 47.4h with a time-lapse of 5.9 minutes (Plan10x, NA 0.5, Nikon). Cells were maintaint at 37°in humidified atmosphere of 5% CO2 during imaging. This raw data has been partially labelled by a biologist for plate 1, well 0015. The data can be found on the CellCognition website [3].

3.4 Fluroescent markers

The first marker is H2B-mCherry [2], is commonly used to help detect the different stages of mitosis (interphase, prophase, prometaphase, etc...). H2B is one of the 5 main histone proteins involved in the structure of chromatin. The cell line is genetically modified such that the gene also expresses a fluorescent tag that will stick to the chromatin. This tag will emit in a certain spectrum. As the histones are bound to DNA, this marker highlights DNA. While it is not possible to resolve single chromosomes with the used imaging technique, we do observe the cell nuclei with a decent resolution, which allows us to distinguish the different mitotic phases. This can be seen from 4 : for instance, the two cells at the bottom left are bright, elongated and relatively small. This cell has just divided and the chromosomes are still condensed (this is called anaphase) which explains these distinctive features. Classification of cells in the mitosis phases has already been done and yielded a 99.4% accuracy per cell, over 4000 labelled cells, see [1] for more details.

The second marker is PCNA [3]-eGFP [4], this marker wraps itself around the DNA and highlights DNA concentration, which we can notice for the cell at the top of figure 5. This marker helps us differentiate the different non-mitotic phases by emitting during the S phase. See figure 24 in annexe for both emissions.

2. "mCherry" is the spectrum of emission, red here, but the picture was set in black and white to improve contrast.
3. Proliferating cell nuclear antigen
4. "eGFP" is the same as for "mCherry", except that the colour is green.



Supervised learning scheme with two datasets.

3.5 Supervised learning

The classification problem at hand is a supervised learning classification. We wish to use the data given by live-imaging marked with `H2B` in order to predict the non-mitotic phases. The `PCNA` marker allows us, or to be precise, biologists to label the data, because differentiating the non-mitotic phases manually can be a difficult and time-consuming task without the `PCNA` marker. The `PCNA` marker informs us on the non-mitotic phases. The dataset provided by Michael Olma provided us with labelled data thanks to the `PCNA` marker. It also emit in a different spectrum, in the same spectrum as the emission of the `H2B` emission. This second spectrum emission allows us to train our model on these labelled data. We wish to then predict the MitoCheck dataset that have an `H2B` marker.

4 Transforming the data : *CellCognition*

Most of the code for transforming the data has already been implemented. I used a fast and cross-platform image analysis framework, *CellCognition* [3], which is a python based program which when given live imaging, will do a nuclei segmentation and feature extraction. This tool co-developed by Thomas Walter is very useful to annotate cells and to extract features from live imaging. In figure 6, we see how *CellCognition* proceeds to extract features from nuclei. The image on the outer right of figure 6 is an example of an expected outcome of a classification. As this is not the main part of the internship, I will only give a brief description of how the data is transformed from live imaging to single cell features.

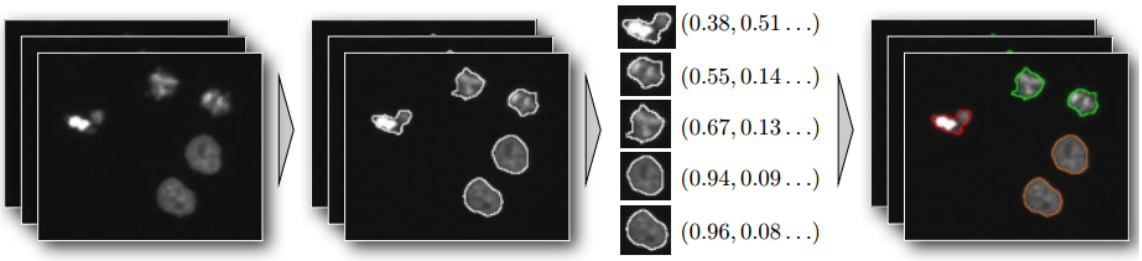


FIGURE 6 – *CellCognition* steps for feature extraction

4.1 Nuclei segmentation

As the pictures have black backgrounds and that the nuclei are bright, the segmentation is in most cases straightforward ; they "cut" around the possible object by levelling it with the background image. The only difficulty appears when two nuclei are in close proximity or overlap. In this scenario they use the watershed algorithm to separate "touching" nuclei. In the next figures, we will describe the watershed technique. In figure 7, we see two overlapping cells. We then assign to each pixel inside the shape its distance to the closest background pixel. This distance map (inverted) is illustrated in 8 along the red line in figure 7. We then simulate a flooding starting at the local minima by successively assigning pixels to the two minima. All pixels where the two "lakes" meet are part of the watershed line and indicate the location of where to cut (see figure 9). Here it will be at point Z_0. The watershed line will be the set of cutting points for the segmentation between the two overlapping cells.

4.2 Extracting the features

From this segmentation, we collect for each individual cell a number of features that can be chosen thanks to the CellCognition interface. We collected a total of 237 features. These features are of very different nature, some are related to the shape. One will take the ratio between his own area and the area of his convex hull. Another feature will count the number of separated compartments (connected components of the set difference between convex hull and initial shape). As an example, we can look at the odd shape in figure 10 where the initial shape is in black and

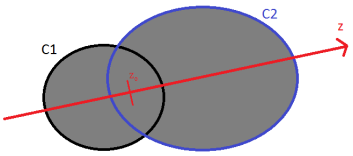


FIGURE 7 – Sketch of two overlapping cells

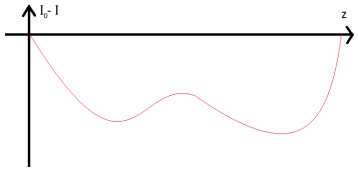


FIGURE 8 – Intensity curves minus the background

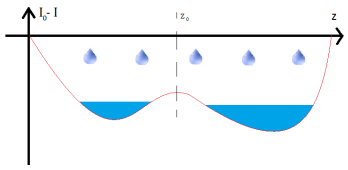


FIGURE 9 – Watershed method for segmentation

the red parts are added in order to create the convex hull. We count 2 separate compartments and the ratio of area will be equal to $\frac{conv(X)}{X} = \frac{C_1+A_1+A_2}{C_1}$, where C_1, A_1, A_2 are the area of their compartments and X is the object. Of course, we expect that many of these variables will be correlated. There are 7 different families of features :

1. Basic features like size, perimeter, mean grey level, etc.
2. Haralick features (characterizing the texture of objects).
3. Levelset features (statistical geometric features).
4. Moment based features.
5. Convex hull features.
6. Granulometry features.
7. Dynamic features.

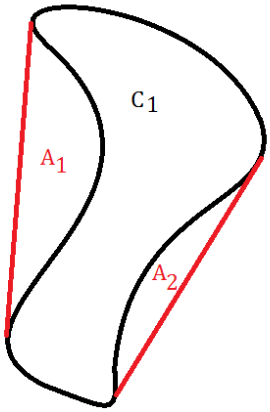


FIGURE 10 – Example of two features.

More information about these features can be found in *Technical Report : Feature extraction* [4]. This reference may be considered "invalid" because it is not a real article, it is a technical report that is confidential. I have joined this article in the annexe if it may interest the reader.

5 Methods

5.1 Nested cross validation

When the number of labelled data is scarce we often choose to do cross validation. Which means that we split the data set into several training sets and testing sets in order to tune each parameter of a specific model (random forest, support vector machine, ...). However, a simple cross validation gives an upper bias estimate of the accuracy of the model. The bias arises because we find the best tuning to predict the test set, so the accuracy for the test set is maximized. To assess without bias the accuracy of a model we have to do a nested cross validation. A nested cross validation is in fact two intertwined simple cross validations. In the "outer" cross validation we split the data set in two, one for choosing/tuning the model and the other for testing. To choose/tune the model, we do an "inner" cross validation on this training set. For a clear representation of a 5 fold nested cross validation scheme, see figure 11.

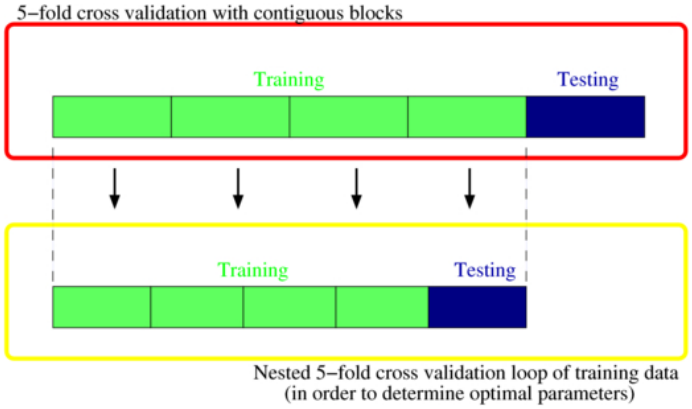


FIGURE 11 – Nested Cross validation

5.2 Classification task

We have partially labelled data from the plate *LT0001_01* and well number *0015* which had in total 1272 annotated cells in phases *G1*, *S*, *G2* and *M*. At first, *S* was divided into three subgroups *earlyS*, *midS* and *lateS*, but I have only worked with *S* because I was only interested in separating the phase *S* from *G1* and *G2*. In these 1272 annotated cells we were able to extract 237 features that are related to size, ratio, etc. For each cell on each image, the cell has an identification number that is associated to a line in our database. Our first task was to classify each cell into one specific group, *G1, S, G2* and *M*. We tried several models such as random forest and support vector machine.

5.3 Time dependencies correction

Taking into account time dependencies and cell evolution should improve the prediction rates. This aspect is crucial, it has yet to be taken into account and could correct the predicitons. Moreover, Alice Schoenauer Sebag, a PhD student at cbio, who's work focused on cell tracking and analyse

of cell mobility put at my disposal tracking results. These results were stored thanks to a python module, named `Pickle`. The tracking allows me to keep track of cells throughout the live imaging and allowed me to take into account time-dependencies.

5.4 Filtering the trajectories

The data, originating from the PhD students work, was the tracking of all of the recorded trajectories. The tracking is not absolute ; it may fail if a cell exits the filmed area or when two or more cells start overlapping. So if we wish to study cell cycle length distribution we have to filter out complete trajectories from the others. A complete trajectory is a trajectory originating from a mitosis and finishing by a mitosis. In order to filter out these trajectories, they developed a scoring approach for each track based on the classifier used to predict the state of the cell in the mitosis stage, see [1]. This classifier is a tuned support vector machine trained on unnormalized data. This scoring approach checks the beginning and the final state and the related siblings of these cells (parents and brother if we are looking at score 1 and children if we are looking at score 2). If at least one of these siblings is in the right state according to a transition then it will be validated. Both scores have to be validated in order to accept the trajectory. In order to take more trajectories into account, we relaxed the filtering by only restricting ourselves to the first score. All final trajectories were derived from a mitosis. In a more formal setting, let $M_{-1} = \{prometaphase,\ metaphase,\ metaphase\ alignment\ problem\}$ and $M_{+1} = \{anaphase\}$. If we denote by τ the track under investigation, T_0 the first moment after the split and $Mother_\tau$ the state of the mother of track τ then $score_{1,\tau} = \mathbf{1}_{\{Mother_\tau \in M_{-1}\}} + \mathbf{1}_{\{\tau_{T_0} \in M_{+1}\}}$. We accepted a trajectory τ if $score_{\tau,1} \geq 1$.

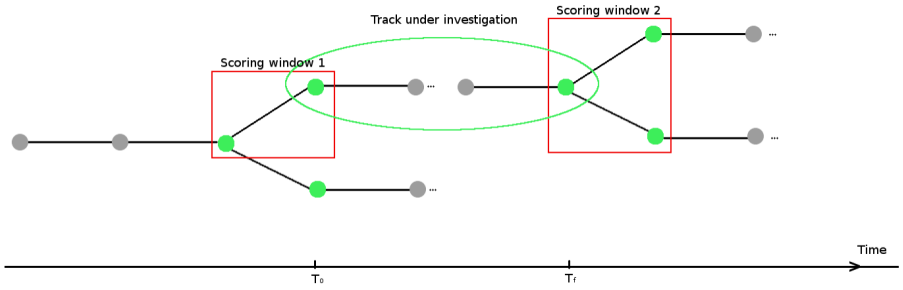


FIGURE 12 – Filtering with two scores
Courtesy from Alice Schoenauer Sebag.

5.4.1 Normalizing the data

Each cell is unique and has its own size and intensity, for example, a cell has its own cell-specific effects. Let's consider a cell at a given time t, let's also suppose that this cell originated from a mitosis. Instead of classifying the cells with respect to their original features we chose to classify the cells with respect to an increase or a decrease of their features compared to the first moment after mitosis. For more clarity please look at figure 13. Using a similar method to panel data and time series, the key is to eliminate the cell-specific effects. Each cell trajectory is considered as a time

series which starts with a mitosis. A random cell on a random frame with a feature vector l will belong to a specific trajectory i and therefore there will exist t such that $X_t^i = l$. Let's denote the first moment after mitosis as $t = 0$. The new features for this cell are $v_t^i = X_t^i / X_0^i$. Of course, not all cells originate from a mitosis, including cells that initially start in phase *G1, S* or *G2*, or cells that move into the filmed area. Because of the restrictions of starting after a mitosis, we significantly reduced the number of annotated cells to 508 labelled cells (or 472 if we do not take account mitosis labels) from the 1162 (or 1272 if we do not take account mitosis labels).

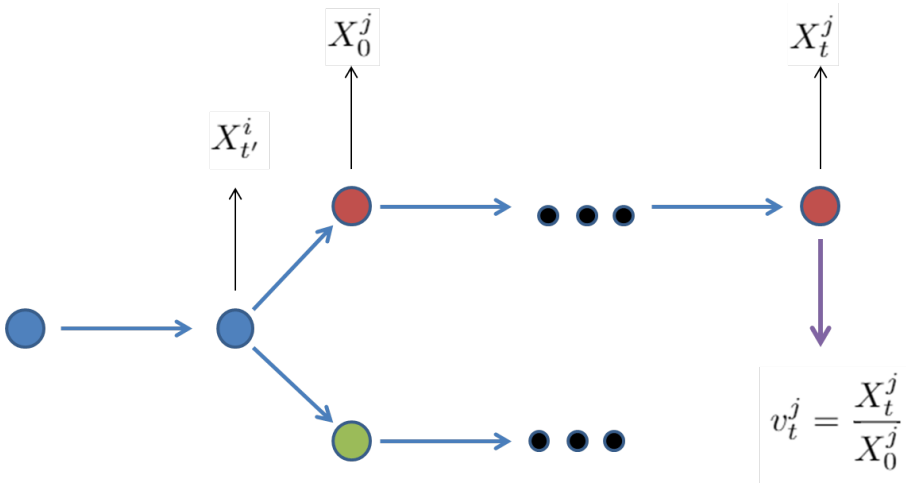


FIGURE 13 – Normalisation of the data

In practice for many i's and t's, the feature vector X_t^i sometimes has a lot of zeros. This can be a problem when it is the first feature vector of a particular trajectory, because all the elements of the trajectory will have infinite values making a lot of the information within this trajectory unusable. To avoid this problem, we added 1 to all individuals for the problematic features. 36 features where modified. We choose this modification because we wish to use a random forest classifier. The random forest algorithm looks at a certain number of features and splits them according to their distribution, if all individuals follow the same modification the classification results should not change.

5.4.2 Hidden Markov Chain

To take more strongly into account time dependencies, we set up an error correction model based on hidden Markov chains. If we consider the trajectory of a single cell, then we know that this cell will go through the non-mitotic phases in the order described in figure 1. We can therefore set a strong prior on the state transition matrix, see figure 14. Therefore, given a certain trajectory of observations given by the previous classification, we wish to find the most likely state path with respect to the initial prediction, see figure 15. We will correct the previous predictions by a standard cell cycle path, as described in figure 1. To set up a hidden Markov model we have to give the possible observation values, hidden states, transition and emission probabilities. For the first case, possible observation values are the predicted states by the first classification and the hidden states are the actual states of the cells. The emission distribution is a multinomial distribution that is given by the confusion matrix of the classification. As said above, the transition distribution is

also a multinomial distribution given by a strong prior and was trained by the baumWelch algorithm that allows you to estimate different parameters of the hidden markov model. We did not allow transitions between states that are not normally connected as we can see in figure 14. We trained the model with the baumWelch algorithm on all of our partially labelled and full trajectories with strong restrictions. We only allowed transitions given by the biological process. The emission probabilities matrix is exactly the confusion matrix given by the classifier ; the confusion matrix informs us of who is well classified and who is misclassified. The parameters for the Markov chain are given in the annexe. The transition state matrix informs us on the temporal aspect and we may have to slightly modify these transition probabilities if temporal aspect change. We also had to change the emission probability matrix, indeed, the probability of emitting S while being in $G2$ was a lot higher then emitting $G2$ while being in $G2$, to counter this we simply put a max on that probability : we set it to 0.5.
We also set up a hidden Markov model with continuous emissions to take more information into account ; the reason for this approach will be given later. The observations in the hidden Markov model would be the probabilities given by the classification, the random vector p conditionnaly on the hidden state would follow a multivariate truncated normal distribution, if p is of size d, $p_t|X_t = i \sim \mathcal{N}_{[0,1]^d}(\mu_i, \Sigma_i)$. More details can be found in the next part because this was an issue I did not succeed in overcoming.

We reach an expected accuracy of 85% ; if we consider S and $G2$ as a whole we reach 93% of correct classification. We can therefore extract valuable information about phase $G1$ but not for phase S which is hardly separated from phase $G2$ with our current classification model.

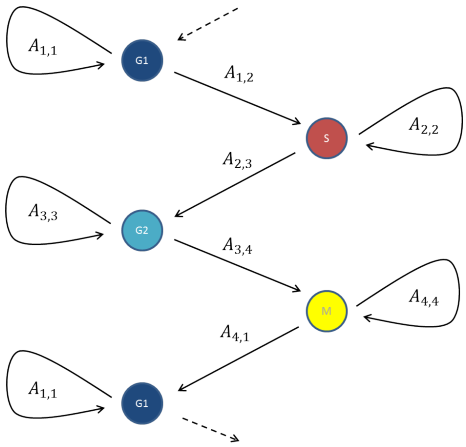


FIGURE 14 – Possible state transitions
We stop transition skips and returns

5.4.3 Issues related to continuous Hidden Markov Chain

To take more information into account, as we can see with figure 26 in annexe, it would be nice to take into account the increase of the probability of phase $G2$ throughout the trajectory. To modelize this process, the emission probabilities were given by the probabilities of each state given by the classifier. For instance, the emission distribution conditionnaly to being in state i is

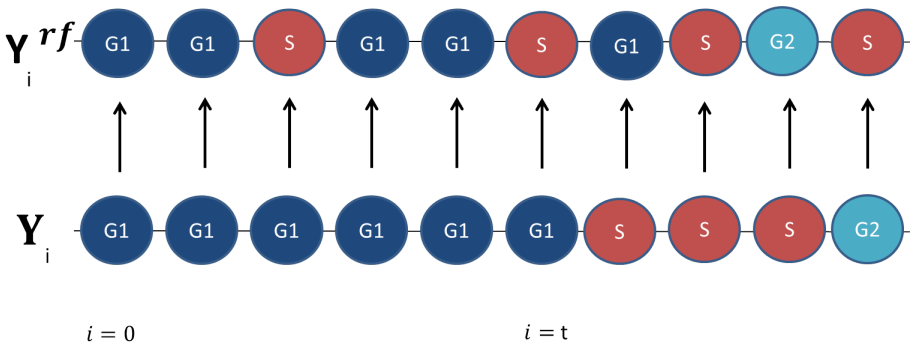


FIGURE 15 – Correcting a trajectory with the hidden markov chain
Y_i^{rf} *is the prediction of the cell at a certain frame* i, Y_i *in its true state*

a truncated multivariate gaussian with mean μ and covariance matrix Σ. For row j (or state j), $mu[j]$ will be equal to the mean of the probability of predicting state j when the true state is i. The matrix Σ will be equal to the covariance matrix of each probability conditionnaly of being in state i. The implementation in R of the continious emission of hidden markov models doesn't finish. On this (or these, because it will happen several times) iteration, the program checks the probability of the given vector of state probabilities and this probability is 0, thus stoping the program. Because of the high dimensiality the probability of occurence of this given vector is 0.

6 Model selection

We are going to test several models in order to deal with the problem. If we do not take into account phase *M*, we have 1160 individuals for the unnormalized data and 472 for the normalized data[5]. If *M* is considered we have 1272 individuals for the unnormalized data and 508 for the normalized data. We tuned each model by simple cross validation and the models accuracy was calculated through nested cross validation.
For the classification accuracy, all valid individuals with respect to the model were taken into account. For example, if the model does not take into account *M* and the data is unnormalized, accuracy is calculated over 1160 individuals. Once the hidden Markov model is added accuracy is only assessed on individuals originating by a mitosis. For example, if we choose a model not taking into account *M* and the data is unnormalized, accuracy is assessed over 460 individuals.
Furthermore, two types of models were tested and tuned, SVM (Support Vector Machine) and RF (random forest) and also three types of outcomes were tested :

— As we are interested in correctly finding the separation *G1/S* and *S/G2*, four of the models are dichotomic, S against the rest. *G1* and *G2* can be separated with respect to time. *M* is not taken into account.

— A model where the response variable can be *G1*, *S* or *G2*.

— A model where the response variable can be *G1*, *S*, *G2* or *M*.

6.1 Results

The results are exposed in figure 16. We notice that phases *S* and *G2* are the least separable. This is why accuracy is also estimated when both classes are regrouped, allowing us to assess how well we can separate *G1* and *S*. We can first establish that the normalization process is indeed very useful as all the models with the normalized data have a higher accuracy. The model yielding the highest accuracy is the Support Vector Machine, with only three responses and normalized data. However we choose to keep the random forest, with all four responses and normalized data as the difference in accuracy is not significant and that *M* is taken into account. Also shown in the annexe is the evolution of each class probability along the first trajectory, which has 185 frames. The raw data for the evolution of each class probability can be found in annexe figure 25, 26 and 26. As predicted, because of the `H2B` marker, the mitosis phases is enhanced, we can easily distinguish between a non-mitotic phase and a mitotic one.

6.2 Conclusion

From this analysis, our model is the normalized data with the hidden markov model, this model yielded an expected accuracy of more than 85%. We can easily distinguish between phase *G1* and the others with an accuracy of 93%.

However, we realized, once we tried predicting the MitoCheck data set, that our classifier predicted several mitotic events in the middle of our trajectories, which is very unlikely. The poor mitotic predicition can be explained by the lack of training mitotic samples. We only have 37 mitotic events

5. i.e Only 472 of the 1160 labelled individuals belong to one of the 235 trajectories that start by a mitosis.

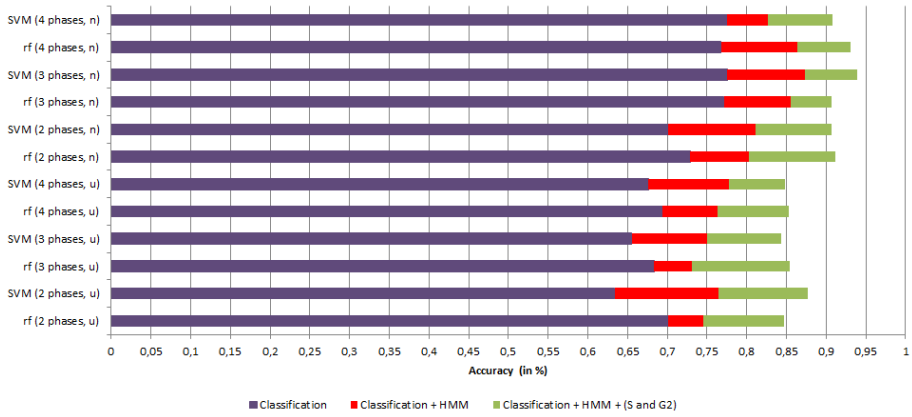


FIGURE 16 – Accuracy of each model

in the PCNA data set. In order to counter this problem we used two classifiers. The first is a binary classifier that predicts if the cell is or is not in a mitotic state. The second is a three class classifier between phases *G1*, *S* and *G2*. This second classifier is nearly as good as our four class classifier that we wished to choose originally. The first classifier is based on the same classifier in [1], it has a very good classification rate for mitotic events and will have priority over the second classifier for the state prediction. No matter what the prediction is in the second classifier, if the first predicts a mitosis then the state of the cell will be *M*. This classifier yields a good accuracy because the raw data has the H2B marker which is used to highlight mitotic events. Our final classifier can be seen in figure 17. We hope to globaly improve prediction rate with a better mitotic classifier as the hidden markov model may help make the transitions between *S* and *M*.

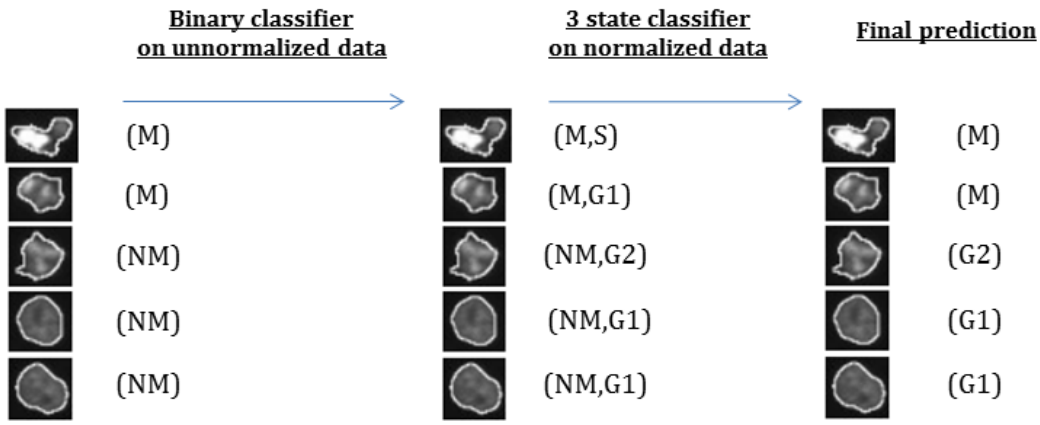


FIGURE 17 – Final classification model

7 Applying it to MitoCheck database

I wish to remind the readers that the final goal of my internship is to know whether or not we can recognize the non-mitotic phase with a marker usually used for the mitotic phases. This would be used in particular on the MitoCheck database in order to combine results from a loss-of-function experiment and protein localization. The MitoCheck data, as described in section 3, is slightly different to the data with the `PCNA` markers, so we wish to try and adapt our predictors in order to improve our prediction over the MitoCheck database. These techniques are widely known as transfer learning or domain adaptation. An additional difficulty is that we have no labelled data for the MitoCheck data base. In order to assess the quality of our classifier we will take our analysis one step further by statistically analysing the lengths of each phase and by comparing them to the literature.
The data we are using are negative controls from plate number 4 and wells numbers 15, 26, 63, 74, 304, 315 and 352. The 2380 collected trajectories verify the first score and these 2380 trajectories represent 81 151 instances.

7.1 With no alteration

For the first analysis, we will use the trained classifier as it has already been trained on the `PCNA` data set. We then correct it. For the MitoCheck data base, as the frame rate is different, the transition matrix for the hidden Markov chain is different. The MitoCheck database has a rate of one frame each 30 minutes over a period of 48 hours and the `PCNA` data base has a rate of one frame every 5.9 minuntes. Roughly, there are 5 times more frames in the latter data base. We have to modify the transition state matrix. The probability that a transition will occur in the MitoCheck database is equal to the same probability that a transition occurs on 5 frames in the `PCNA` data set. For $\forall i \in \{1,2,3,4\}$, representing the state of the cell we have, lets denote by $TP[i,j]$ the value of state transition probability matrix from state i to state j in the `PCNA` data set :

$$\begin{aligned}\mathbb{P}(i \overset{MitoCheck}{\longrightarrow} (i+1)) &= \mathbb{P}(i \overset{PCNA}{\longrightarrow} (i+1) \overset{PCNA}{\longrightarrow} (i+1) \overset{PCNA}{\longrightarrow} (i+1) \overset{PCNA}{\longrightarrow} (i+1)) \\ &+ \mathbb{P}(i \overset{PCNA}{\longrightarrow} i \overset{PCNA}{\longrightarrow} (i+1) \overset{PCNA}{\longrightarrow} (i+1) \overset{PCNA}{\longrightarrow} (i+1)) + ... \\ &= TP[i,i]^3 TP[i,i+1] + TP[i,i]^2 TP[i,i+1] TP[i+1,i+1] \\ &+ TP[i,i] TP[i,i+1] TP[i+1,i+1]^2 + TP[i,i+1] TP[i+1,i+1]^3\end{aligned}$$

We decompose these probabilities by using Markov chain properties.
We have to keep in mind that the lengths of each phase will strongly depend on the type of the cell, the cell line and the environment in which these are in. Therefore there is no guarantee that the lengths of each phases should match. However, we wish to analyse which method accepts the most trajectories. An accepted trajectory for a certain phase will have a beginning and an ending. Trajectories where our classifier predicts an ending in S will only have a possible *G1* phase length.

7.1.1 On the `PCNA` data set

To have benchmark-like results, we decide to first check our classification on the data collected with the `PCNA` marker, we recall that we have 234 trajectories that verify the first score. We get the following table of results :

Length of :	Mean	Standard deviation	Number of trajectories
G1	6.52	3.11	159
S	8.14	3.21	107
G2	1.70	2.00	100
Cell Cycle	17.65	2.2	106

TABLE 1 – Basic characteristics of phases length on the `PCNA` data set.

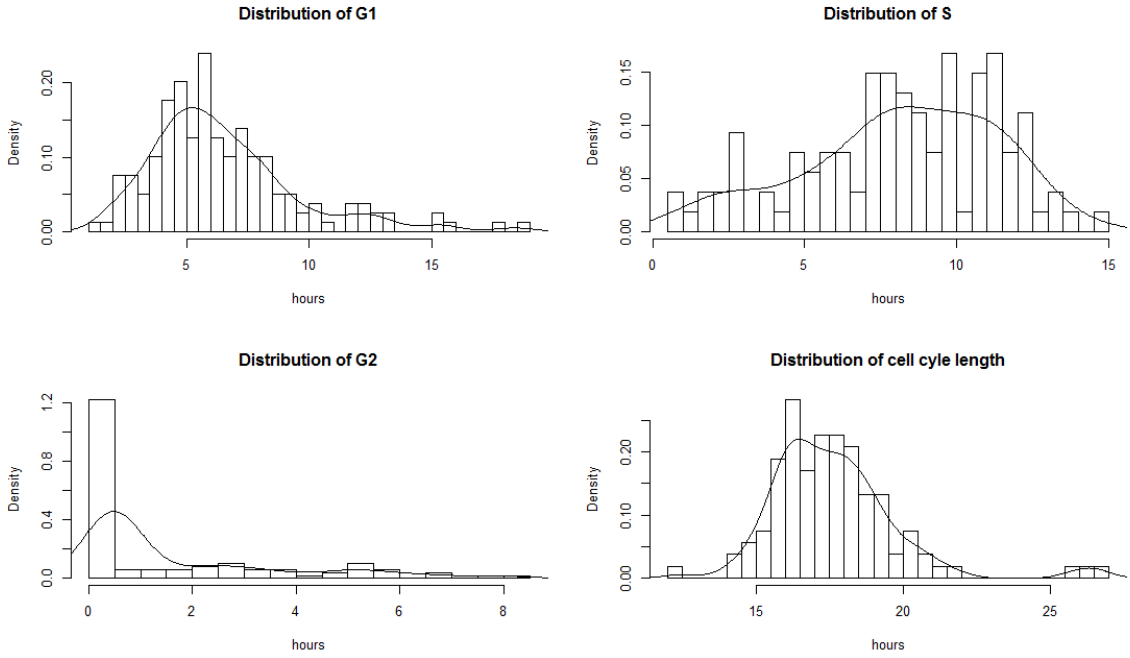


FIGURE 18 – Distribution of phases length on the `PCNA` data set.

We can also plot the distributions of each of these results, see figure 18. What we first notice is that the length of phase S is superior to the length of *G1*. Out of the 234 trajectories that fulfil score 1, our classifier only predicts 106 full cycles. As expected, *G2*'s length is very badly estimated, and we also notice it with the distribution estimations. However the other distributions are neat even with the counter intuitive means of lengths of the two first phases.

7.1.2 On the MitoCheck data set without reweighting

We can also plot the distributions of each of these results, see figure 19 . Out of the 2408 trajectories that fulfil score 1, our classifier only predicts 29 full cycles. What we first notice, contrary to the `PCNA` data set, is that the length of phase *G1* is superior to the length of S and this phase length is superior to the final phase *G2*. The estimation of *G1*'s length is close to what we would expect, the values for the lengths of S and *G2* are not too close to literature.

Length of :	Mean	Standard deviation	Number of trajectories
G1	11.76	6.27	277
S	5.81	3.64	194
G2	5.19	4.19	29
Cell Cycle	23.00	3.94	29

TABLE 2 – Basic characteristics of phases length on the MitoCheck data set.

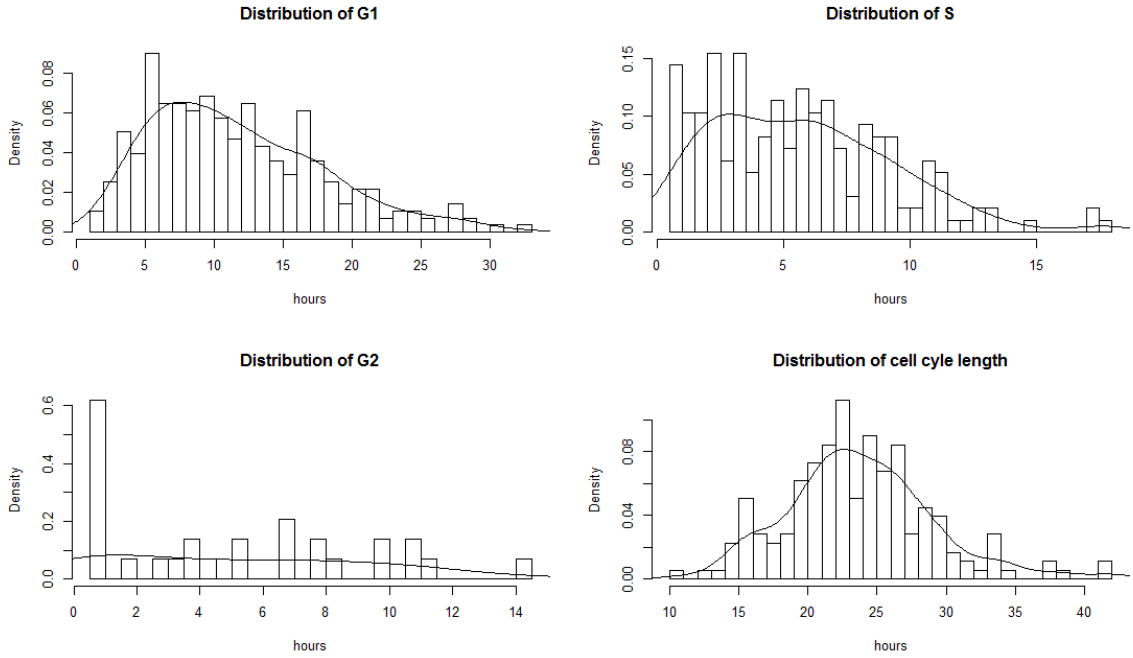


FIGURE 19 – Distribution of phases length on the MitoCheck data set.

7.2 Domain Adaptation with instances reweighting

One of the key assumptions in machine learning and data mining algorithms is that the training and future data must be in the same feature space and must have the same distribution. However, in many real world applications these assumptions are not fully validated. It seems natural that our training set was taken under some circumstances and that our test set was taken under others. It is usually not necessary to adjust the algorithms in order to take into account these differences but in some particular cases it can increase the prediction rates. Transfer learning or domain adaptation is the study of how we should adapt the learning method or the feature space in order to take into account these differences. In our case, we wish to improve the previous results by transfer learning and in particular by instance knowledge transfer. This means that we are going to alter the learning algorithm in order to learn with better suited training instances. These better suited training instances are chosen if they represent the future dataset. From a Bayesian perspective, we make the hypothesis that our training set (respectively the future data) depends on a hidden

variable λ (resp. θ), let's denote by p_λ (resp. p_θ) the probability distribution of the training set (resp. of the future data). We then weight each instance x_i of the training set by $\frac{p_\theta(x_i)}{p_\lambda(x_i)}$. Many other transfer learning via instances use Bayesian frameworks, in most they try to find relations between the hidden variables λ and θ. From a frequentist point of view, we make the assumption that the distribution of labels given the data is the same in both data sets. Domain adaptation can also be referred to as transfer learning with instances and by covariate shift.

7.2.1 Method 1

The first method is straightforward. Estimate the training distribution, estimate the future data distribution and then take the ratio of the estimated densities on each instance, see figure 20. The denominator won't be zero because the training instance is from the training distribution, so $p_\theta(x_i) \neq 0$. We usually adopt kernel method density estimation. This method doesn't work in practice when the number of features is large. Some may find that even 6 dimensions is too big for kernel density estimation. Because of the curse of dimensionality, increasing the number of dimensions will decrease precision of the estimation of both distribution and will drastically increase computation time. Furthermore, because we have to take the ratio of both estimated densities, small errors in estimating Pr can lead to a large coefficient which will then lead to large differences. The training will be based on wrong weights and the final predictions will be poor. This method wasn't possible, even by reducing the number of variables to 6. The selection of these 6 features was done with respect to the sensitivity of each feature to the output. The computation time to estimate the distribution of the future data on these 6 features was too great. I recall that we have over 80 000 instances and 6 dimensions. A stronger analysis of this situation can be interesting as these 6 features may not be the most characteristic for the data set as they may be strongly correlated. It may be efficient to use a principal component analysis in order to reduce de number of dimensionality and to make them uncorrelated, which would work nicely for the density estimations.

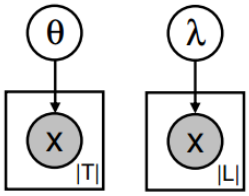


FIGURE 20 – Learning under covariate shift by estimating training and test densities separately (image taken from [6])

7.2.2 Method 2

A more reasonable estimation of the weight is possible by not having to estimate the full data distributions. To do so, Huang et al., see [5] suggest turning this problem into an optimization problem.
Formally speaking, we have training samples $Z = \{(x_1, y_1), ..., (x_m, y_m)\} \subseteq \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the feature space and $\mathcal{Y}$ is the space of labels. This training sample was drawn from a probability

distribution $\Pr(x, y)$ and the same with the future data named Z' that were drawn from Pr'(x, y), the labels in Z' are unknown to us. The key assumption is that given the data, the distribution of labels is the same, we have $\Pr(y|x) = \Pr'(y|x)$.
The task at hand is to minimize the expected risk of the future data set with respect to a certain loss function $l : \mathcal{X} \times \mathcal{Y} :\to \mathbb{R}^+$:

$$R[\Pr', l(x, y)] = \mathbb{E}_{(x,y)\sim\Pr'}[l(x, y)]$$

However, if $\Pr = \Pr'$ we could train on the labelled data and any machine learning algorithm would minimize this expected risk. We have the following equalities :

$$\begin{aligned} R[\Pr', l(x, y)] &= \mathbb{E}_{(x,y)\sim\Pr'}[l(x, y)] \\ &= \mathbb{E}_{(x,y)\sim\Pr}[\frac{\Pr'(x, y)}{\Pr(x, y)} l(x, y)] \text{ (Same technics as in importance sampling)} \\ &= \mathbb{E}_{(x,y)\sim\Pr}[\beta(x, y)l(x, y)] \\ &= R[\Pr, \beta(x, y)l(x, y)] \end{aligned}$$

Where $\beta(x, y) = \frac{\Pr'(x,y)}{\Pr(x,y)} = \frac{\Pr'(x)\Pr'(y|x)}{\Pr(x)\Pr(y|x)} = \frac{\Pr'(x)}{\Pr(x)}$ is the key assumption in this setting. Like in importance sampling, we need the support of $\Pr'$ to be contained in the support of Pr, if not we cannot make the ratio. The equality over both expected risks shows that we can minimize the expected risk of the future data by minimizing the expected risk of the training data weighted by β.
We will present the main theorems and optimization problem to solve this, but for more details please refer to [5]. Let's denote by $\Phi : \mathcal{X} \to \mathcal{F}$ to be a map into a feature space $\mathcal{F}$ and denote by $\mu : \mathcal{P} \to \mathcal{F}$ the expectation operator, $\mu(\Pr) = \mathbb{E}_{x\sim\Pr(x)}[\phi(x)]$. As in many kernel settings we make the assumption that $\mathcal{F}$ is a reproductive kernel Hilbert space. Every function of this particular space is invariant by dot product with a kernel. We can infer a suitable β by solving this optimization problem :

$$\underset{\beta}{\text{minimize}} \; ||\mu(\Pr') - \mathbb{E}_{x\sim\Pr}[\beta(x)\phi(x)]|| \text{ subject to } \beta(x) \geq 0 \text{ and } \mathbb{E}_{x\sim\Pr(x)}[\beta(x)] = 1.$$

As $\mu(\Pr')$ and Pr are unknown and that we only have two samples of finite size, one for training and the other for prediction we have to ensure that the empirical estimates of these quantities converge with a reasonable standard error. We have the following results on the convergences :
If $\beta(x)$ is bounded by B and if $\beta(x)$ for $x \sim \Pr$ has finite mean and non-zero variance then $\frac{1}{m}\sum_i \beta(x_i)$ converges in distribution to a gaussian distribution with mean $\int \beta(x)d\Pr(x)$ and standard error $\frac{B}{2\sqrt{m}}$.
The convergences, like with the central limit theorem, is in $O(1/\sqrt{m})$. The second results demonstrates the deviation between the empirical means of $\Pr(x)$ and $\beta(x)\Pr(x)$ in feature space given $\beta(x)$:
If we have $\{x_1, ...x_m\}$ from Z and $\{x'_1, ...x'_{m'}\}$ from Z' and the same conditions as before on $\beta(x)$, and if the feature map is bounded by R, i.e. for all $x \in \mathcal{X}$, $||\phi(x)|| \leq R$ then with probability $1 - \delta$, given β, we have :

$$||\frac{1}{m}\sum_{i=1}^{m}\beta(x_i)\phi(x_i) - \frac{1}{m'}\sum_{i=1}^{m'}\phi(x'_i)|| \leq (1 + \sqrt{-2\log(\delta/2)})R\sqrt{B^2/m + 1/m'}$$

This second result gives us an upper bound on the difference of the empirical means if β is optimal, with respect towards the population. It is no guarantee that the empirical means should match and the convergence is in $O(\sqrt{1/m + 1/(m'B^2)})$.
We wish to minimize the discrepancy between means subject to constraints $\beta_i \in [0; B]$ and $|\frac{1}{m}\sum \beta_i - 1| \leq \epsilon$. Minimizing the discrepancy to minimize the gap between $\Pr'$ and $\beta\Pr$, whereas the second constraints ensure that $\beta\Pr$ is close to a probability distribution, see the first result on the convergences. One may check that we have the following equality :

$$||\frac{1}{m}\sum_{i=1}^{m}\beta(x_i)\phi(x_i) - \frac{1}{m'}\sum_{i=1}^{m'}\phi(x_i')|| = \frac{1}{m^2}\beta^T K\beta - \frac{2}{m^2}\kappa^T\beta + \text{const}$$

Where K is a $m \times m$ matrix and for all i, j we have $K_{ij} = k(x_i, x_j)$ and κ is a m sized vector where for all i, $\kappa_i = \frac{m}{m'}\sum_{j=1}^{m'} k(x_i, x_j')$. The optimization problem can therefore be rewritten as :

$$\underset{\beta}{\text{minimize}}\frac{1}{2}\beta^T K\beta - \kappa^T\beta \text{ subject to } \beta_i \in [0; B] \text{ and } |\sum_{i=1}^{m}\beta_i - m| \leq m\epsilon$$

.

7.2.3 Weighted Random Forest

Both of these methods rely on a weighted classification ; however in most packages and in most computing languages weighted classification algorithms are not yet implemented. Because of the simplicity of tuning of the random forest and of its efficiency we decided to only explain how we weight the instances within a random forest. We are first going to explain random forests as they were first implemented by Breiman in 2001. If we wish to construct N_trees on a data set of size N, we have the following algorithm :

Result: A regression or classification model.
Input: $N\_trees,\ m,\ n_{min}$
for $b = 1, ..., N\_trees$ **do**
(a) Draw a boostrap sample X^* of size N from the training data.
(b) Grow a random-forest tree T_b to the bootsrapped data, by recursively repating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
(i) Select m variables at random from the p variables.
(ii) Pick the best variables/split-point among the m.
(iii) Split the node into two daughter nodes.
end
Output: Ensemble of trees $\{T_b\}_1^{N\_trees}$.

Algorithm 1: Random Forest for Regression or Classification.

To make a prediction at a new point x :
Regression : $\hat{f}_{\text{rf}}^{N\_tree}(x) = \frac{1}{N\_tree}\sum_{b=1}^{N\_tree} T_b(x)$

Classifications : Let $\hat{C}_b(x)$ be the class prediction of the b-t random-forest tree. Then $\hat{C}_{\text{rf}}^{N-tree}(x) = majority\ vote\ \{\hat{C}_b(x)\}_1^{N-tree}$.
The weighting procedure we apply in our context is based on the training of each individual tree. Therefore we wish to give more times (with respect to its weight) samples when we train an individual tree. As they are implemented in R or Python, the classifier does a uniform bootstrap on the whole training set. Therefore, making a weighted random forest implies modifying the bootstrap procedure in order to sample with weights. However, it is not the most practical way to alter these procedures, especially in R. It would be easier with Python as we can easily access the source code within the package. In order to not alter the standard procedure we use a trick : as the bootstrap method will resample from the training set at each iteration, we will repeat the instances in the original training set according to their weight. In other words, we create a training set of size 1000 (more than twice the size of the original data set), where sample i is repeated $\hat{\beta}_i \frac{1000}{\sum_j \hat{\beta}_j}$.

7.3 Discussion about the domain adaptation technics

7.3.1 Tuning

Huang et al. in figure [5] discuss how to choose each parameter. There are 3 parameters to choose : the choice of the kernel and the associated hyper-parameters, the upper-bound B of the weight coefficients and the ϵ a "closeness" parameter. Let's comment each of these tuning parameters.
The kernel choice will have impacts on the kernel matrix K and the vector κ. K can be seen a weight matrix in the reproductive kernel Hilbert space. If we choose a 0-mean gaussian kernel which has a hyper parameter σ, the impact in variation of σ can be seen as how many neighbours do we wish to associate with a certain instance. Choosing a small σ will result in a stronger discrimination of each instance, or in other words creating smaller "neighbourhoods" for each instance. However if the neighbourhood is too small, everybody may have the same weights. Choosing a bigger σ will results in bigger "neighbourhoods". We expect, if the training data and the future data feature space are close, that the choice of σ will have little impact in the weighting scheme, whereas on the contrary, a stronger discrimination may put forward better suited instances. If no instances are better suited a smaller σ will result in all samples having the same weight, because no samples will get "close enough". We try with $\sigma = 1$ and $\sigma = 10$.
The upper-bound B on the coefficients forces values to stay reasonable, as the average of β is close to 1, the optimization problem may want to be very discriminative by setting most coefficients to 0 and by setting only a few to high values. In Huang et al., they suggest that $B = 1000$ is a suitable value. In our case, having $B = 1000$ will result in a standard error on the estimation of β of $\frac{B}{2\sqrt{m}} = \frac{1000}{2\sqrt{472}} \approx 23$. Reducing B in our case will strongly improve the validation of the second constraint, that is that βPr must be close to probability distribution. We try for several values of $B = 2,\ 10,\ 50$ and 1000.
The final tuning parameter is ϵ. ϵ is involved in the second optimization constraint on the precision of the average of β to 1. He is closely related to B and will have the same impact. In Huang et al., they suggest to choose $\epsilon = \frac{\sqrt{m}-1}{\sqrt{m}} = \frac{\sqrt{472}-1}{\sqrt{472}} = \approx 0.95$. We try 3 values of ϵ, $\epsilon = 0.1,\ 0.5$ and 0.9.

7.3.2 Results

The estimations of each phase length are exposed in table 3. As expected the results for the lengths of phases S ans *G2* are lot poorer than for *G1*. Out of the 2380 trajectories, we accept a lot

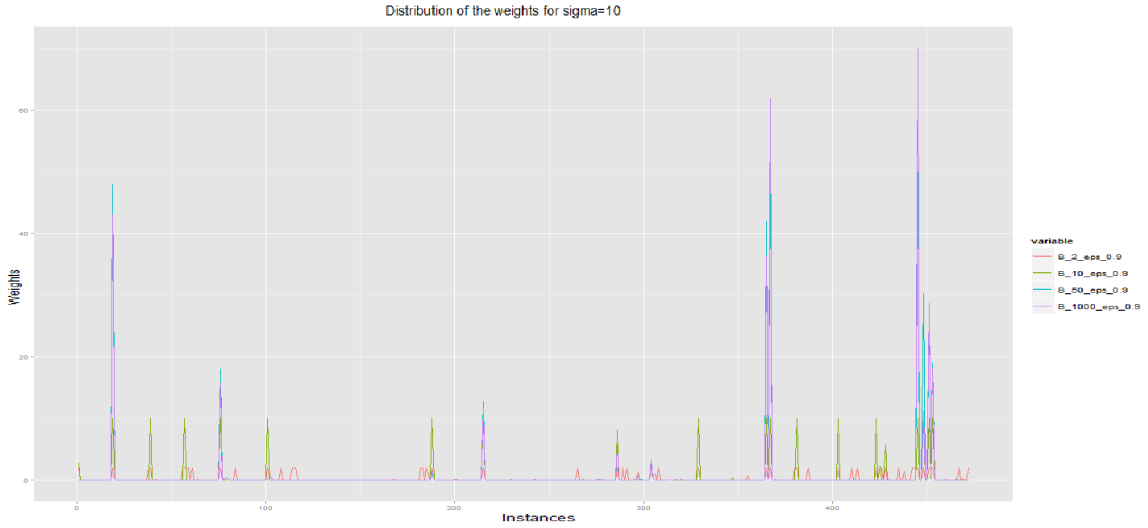


FIGURE 21 – How the weighting scheme varies when B decreases ($\sigma = 10$).

more trajectories for phase *G1* and S for $\sigma = 1$ then for $\sigma = 10$. It is not apparent for $\sigma = 1$ and we will explain why after but the number of trajectories accepted increases with the decrease of B and with the decrease of ϵ. This can be due to the fact that when B or ϵ is large, our weight vector β is sparse and has only a few very large weights. For some values of B and ϵ the model only trains on no more than 20 different instances. In figure 21 and 22 we see how the values of B and ϵ change the weights. We would like to favour a model which will be able to train on a sufficient amount of instances. In figure 21, for $B = 1000$, $\epsilon = 0.9$ and $\sigma = 10$ we have no more than 20 instances with weights higher than 1. Let's discuss $\sigma = 1$, if we would plot the same graphs, we would be confused by the result at first. For a fixed ϵ, the graphs are identical as we can see in figure 23. When $\sigma = 1$, most of the weights seem to take the value of the given ϵ and the scheme down weights not well suited instances. $\epsilon = 0.1$ yields the most variations in $\hat{\beta}$ and figure 23 also shows why the results are similar for $\sigma = 1$. If we try to compare the weighting for $\sigma = 1$ and $\sigma = 10$, we notice that the same instances are chosen or kept (for $\sigma = 1$ these are the "flat regions" whereas for $\sigma = 10$, this corresponds to the "spiky areas").

To conclude we would choose the weighting scheme that keeps the most instances while selecting still selecting those that are better suited. In this case, $\sigma = 10$ chooses too few instances. For $\sigma = 1$ the choice of B does not matter much, so we would choose $\epsilon = 0.9$ to increase the variation in the weighting scheme. We would choose, $\sigma = 1$, $B = 10$ and $\epsilon = 0.9$ as parameters for our transfer learning.

To be clear on the choice of ϵ, if $\sigma = 1$ we start with everyone, so we wish to increase variability in order to have a better selection, to increase variability we increase ϵ. If $\sigma = 10$, we start with nearly no one, so we wish to decrease ϵ to decrease the weights of the selected few and to increase the weights of those not selected.

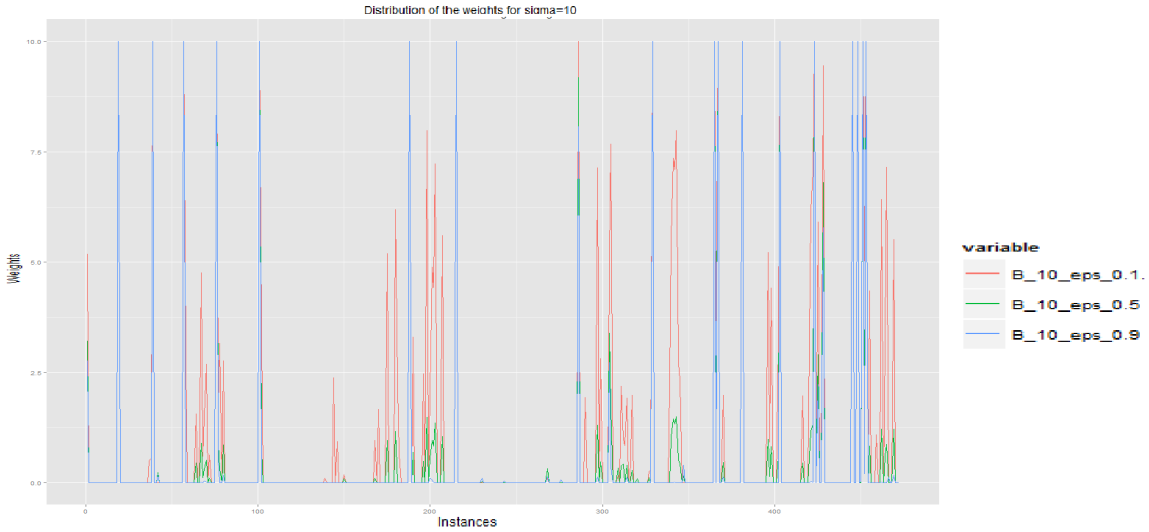


FIGURE 22 – How the weighting scheme varies when ϵ decreases ($\sigma = 10$).

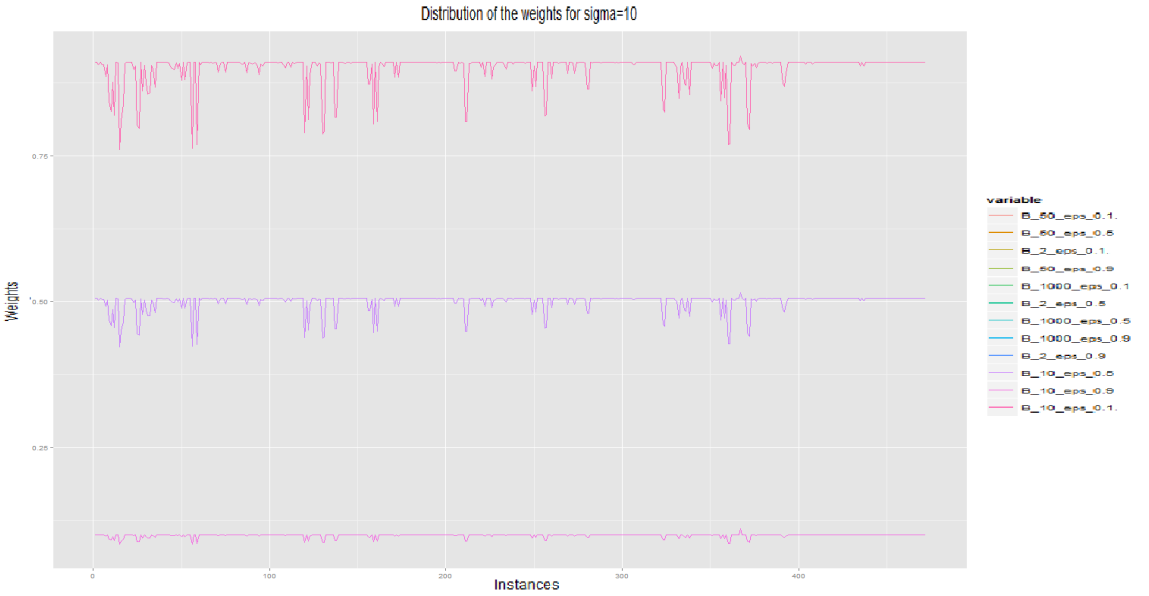


FIGURE 23 – How the weighting scheme varies when ϵ and B vary ($\sigma = 1$).

Paramètre			G1			S			G2			CellCycle		
σ	B	ϵ	Mean	Sd	n	Mean	Sd	n	Mean	Sd	n	Mean	Sd	n
$\sigma = 1$	2	0.1	11.9	6.1	322	5.0	3.6	237	5.8	5.2	107	23.2	3.7	79
		0.5	11.9	6.1	329	5.2	3.7	241	5.6	5.1	106	22.8	3.9	76
		0.9	12.2	6.1	297	4.9	3.5	236	5.4	5.2	104	22.9	3.8	74
	10	0.1	11.8	6.3	302	5.1	3.4	234	5.8	5.0	99	23.1	3.9	72
		0.5	12.2	6.4	327	5.8	3.8	234	5.0	5.0	103	22.8	3.9	75
		0.9	12	6.5	343	5.8	3.9	252	5.4	5.1	108	23	3.8	78
	50	0.1	12.1	6.1	274	4.2	3.2	239	6.1	5.1	104	22.9	3.8	69
		0.5	11.9	6.3	313	5.6	3.8	233	5.6	5.1	102	23.0	3.9	73
		0.9	11.6	6.1	333	5.8	3.7	238	5.2	5.0	107	23.0	3.8	77
	1000	0.1	12.3	6.2	292	4.7	3.3	226	5.6	5.2	100	23.0	3.9	71
		0.5	12.7	6.4	299	4.2	3.5	239	6.2	5.2	101	23.0	3.8	73
		0.9	12.0	6.5	335	5.9	3.6	241	5.0	4.9	112	23.1	3.8	80
$\sigma = 10$	2	0.1	11.5	6.4	260	2.7	2.8	200	8.0	5.6	138	22.9	3.8	92
		0.5	10.6	6.4	245	2.3	2.8	120	8.7	6.1	139	22.9	3.9	84
		0.9	10.5	6.6	221	3.5	4.1	163	8.6	5.9	119	22.9	4	70
	10	0.1	10.8	5.9	171	1.8	2.8	112	6.9	5.7	122	22.6	3.8	60
		0.5	10.4	6.0	202	1.6	2.4	127	8.2	5.9	130	22.6	3.8	60
		0.9	10.7	6.4	96	1.0	0.7	67	5.6	5.3	91	21.9	4.2	36
	50	0.1	11.2	6.3	165	2.5	3.5	127	6.6	5.7	108	22.4	3.9	63
		0.5	15.1	8.8	52	1.2	0.8	61	2.9	4.0	74	21.7	3.9	63
		0.9	14.5	8.4	55	1.2	0.8	53	2.1	2.7	81	20.1	4.9	15
	1000	0.1	11.6	6.1	191	2.7	3.5	162	6.5	5.6	107	22.5	3.8	65
		0.5	13.8	7.6	64	1.1	0.6	40	3.5	4.2	66	20.4	4.7	18
		0.9	12.2	6.9	89	1.4	0.9	74	3.9	4.5	95	21.1	4.8	23

TABLE 3 – Different estimations of phase length on the MitoCheck dataset.

8 Conclusion

I studied whether or not it was possible to detect the non-mitotic cell cycle phases with the H2B marker alone. This fluorescent marker is informative about mitotic events, so it is not guaranteed that we could detect the different non-mitotic phases with the H2B marker alone. On the contrary to the PCNA marker, which is informative about the non-mitotic phases *G1*, *S* and *G2*. The PCNA marker enables us to use a labelled MitoCheck-like data set for training. This training allowed us to infer predictive models for the lengths of the different phases *G1*, *S* and *G2*. But with no ground truth on the future data set it may seem difficult to assess how good our classifiers perform.

The data set is composed of trajectories from different wells. The data was transformed only for the second classifier. A normalization process strongly improved the prediction rate as it takes into account time dependencies and cell evolution through time. Our final classifier combines 2 classification models and a correcting model. The first classifier is binary and checks for mitotic or not events. The first classifier will take priority over the 3 phase random forest classifier, which is the second classifier. The error correcting model is set with a strong prior based on biology beliefs and takes time dependencies more strongly into account. We achieve 86% accuracy on the labelled training set. We can only efficiently detect the transition between phase *G1* and *S*. It can seem fairly natural as even with our own eye it is hard to distinguish the two events.

The classifier was then used on data from a published genome-wide screen, generated by the FP6-financed European Union project, MitoCheck. The MitoCheck data set is a collection of loss-of-function protein experiments where the H2B marker is the only used fluorescent marker. Our classifier was built with this in mind. The raw data of the MitoCheck project is different to the PCNA data set as we have more difficulties in estimating each phase, our classifier validates very few trajectories. This is our only criteria in addition to biological literature. To increase its performance, we use transfer instance learning on the second classifier. These methods are used when there are differences in the training and the prediction set. The one used is based on altering the training set in order to take only the best suited instances. However it is a hard task to validate the performance of our classifiers on the MitoCheck dataset without ground truth.

The next step of this work is to compare it with actual ground truth on the MitoCheck data set which will become available thanks to EMBL, Heidelberg. It would be nice to take time to check the distributions of the most paramount features for classification on both data set (the PCNA data set and the MitoCheck data set) and check that they follow similar distribution. This would allow to assess if transfer learning via instance selection is better suited than transfer learning via feature transformation.

<We may want to relax our classifier, with the hidden Markov model perhaps, in order to track cells which may have become cancerous and therefore do not have standard behaviour.>

If the classifier is good enough, this would enable protein function inference through loss-of-function experiments that is the MitoCheck project.

9 Annexe : Hidden Markov Model

9.1 Emission Matrix :

$$\textbf{EmissionMatrix} = \begin{array}{c} \\ G1 \\ S \\ G2 \\ M \end{array} \begin{array}{c} \begin{array}{cccc} G1 & S & G2 & M \end{array} \\ \begin{pmatrix} .824 & .156 & .005 & .015 \\ .106 & .871 & .023 & . \\ . & .500 & .470 & .030 \\ .033 & .033 & .033 & .901 \end{pmatrix} \end{array}$$

0.023 is equal to the probability of being in the actual state "S" but the classifier says that he is "G2".

9.2 Transition Matrix for the `PCNA` data set :

$$\textbf{TransitionMatrixPCNA} = \begin{array}{c} \\ G1 \\ S \\ G2 \\ M \end{array} \begin{array}{c} \begin{array}{cccc} G1 & S & G2 & M \end{array} \\ \begin{pmatrix} .982 & .018 & . & . \\ . & .921 & .079 & . \\ . & . & .972 & .028 \\ .275 & . & . & .725 \end{pmatrix} \end{array}$$

0.079 is the probability of transiting from state "S" to state "G2".

9.3 Transition Matrix for the `MitoCheck` data set :

$$\textbf{TransitionMatrixMitoCheck} = \begin{array}{c} \\ G1 \\ S \\ G2 \\ M \end{array} \begin{array}{c} \begin{array}{cccc} G1 & S & G2 & M \end{array} \\ \begin{pmatrix} .938 & .062 & . & . \\ . & .796 & .204 & . \\ . & . & .929 & .071 \\ .700 & . & . & .300 \end{pmatrix} \end{array}$$

0.204 is the probability of transiting from state "S" to state "G2".

9.4 Starting probability

$$\textbf{StartingProbabilities} = \begin{array}{c} G1 \\ S \\ G2 \\ M \end{array} \begin{pmatrix} .72 \\ . \\ . \\ .28 \end{pmatrix}$$

10 Annexe : Image

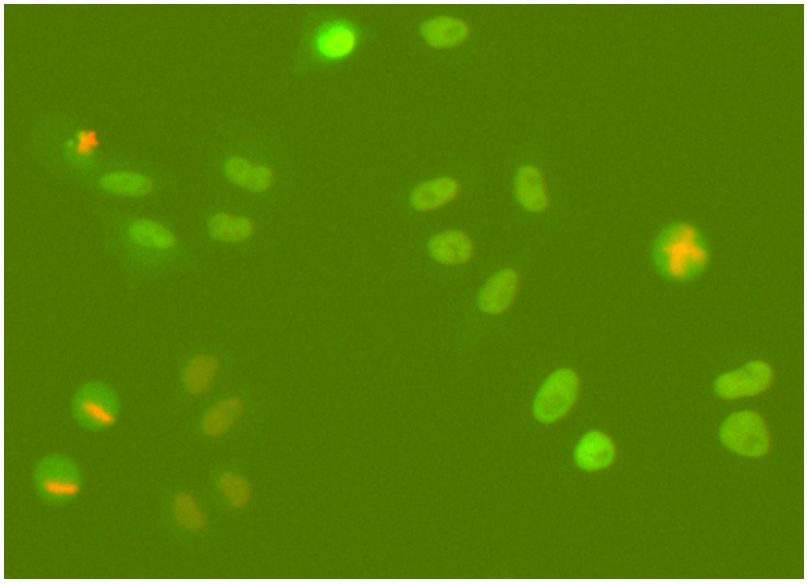


FIGURE 24 – Raw image with emission from H2B-mcherry (red) and from PCNA-EGFP (green)
Data acquired by Michael Olma (Peter group, ETH Zurich) at a Molecular Devices ImageXpress Micro with the PlateScanPackage. They used widefield epifluorescence microscopy, 10× dry objective (0.645 µm / pixel), 1392×1040 pixel, 482 frames × 5.9 min = 47.4 hours. plate LT0001_01, well position 0015, frame number 60

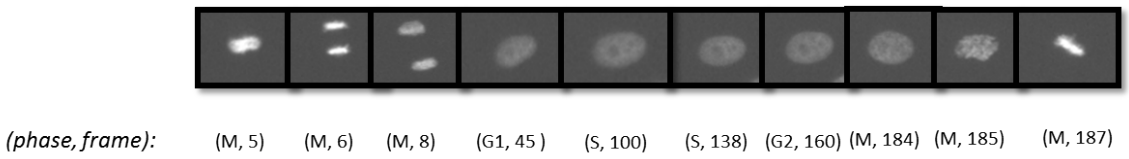


FIGURE 25 – Raw data for trajectory 0.
Data acquired by Michael Olma (Peter group, ETH Zurich) at a Molecular Devices ImageXpress Micro with the PlateScanPackage. They used widefield epifluorescence microscopy, 10× dry objective (0.645 µm / pixel), 1392×1040 pixel, 482 frames × 5.9 min = 47.4 hours. plate LT0001_01, well position 0015, frame number 60

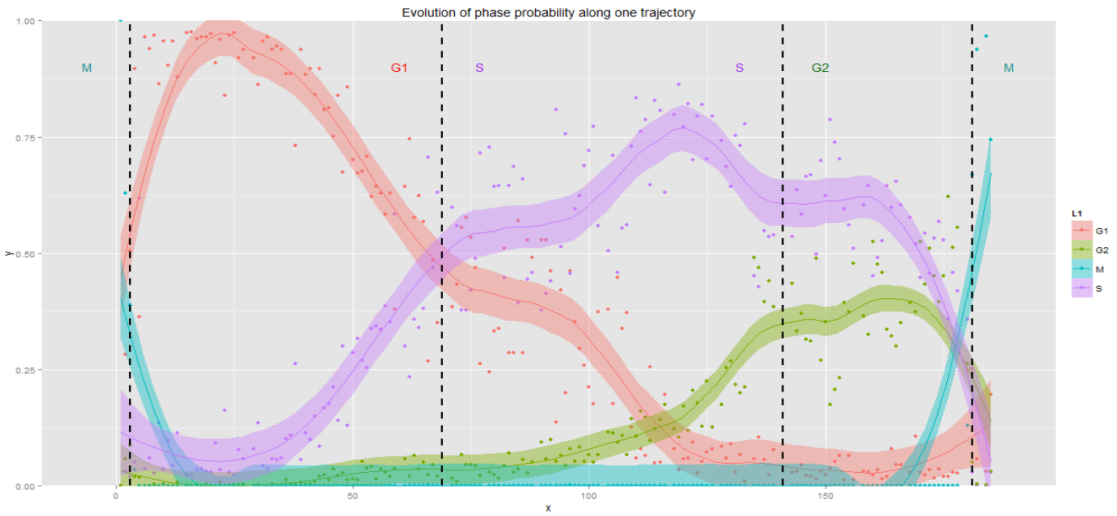


FIGURE 26 – Class probability evolution given by random forest with all four responses and the data is normalized

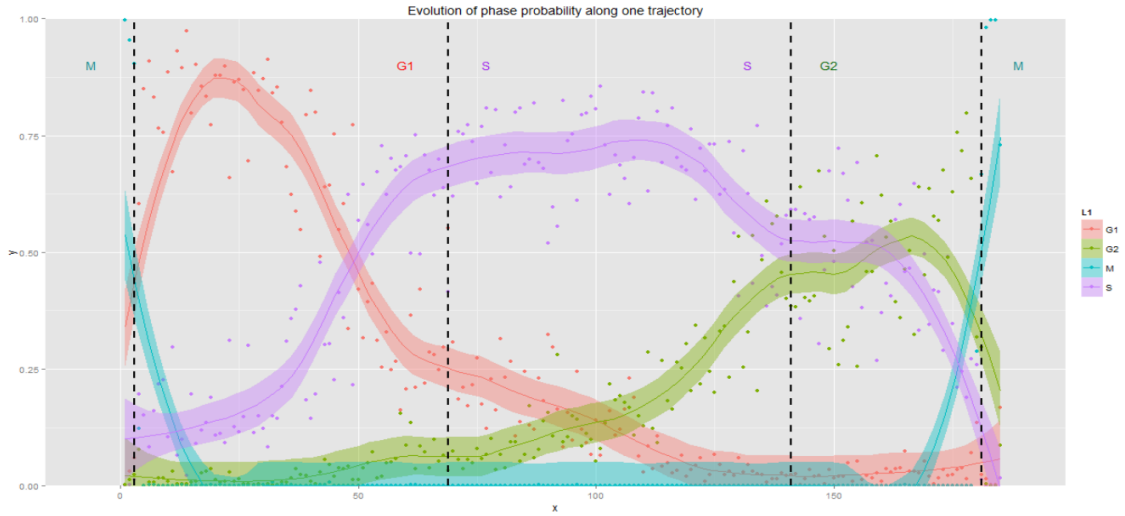


FIGURE 27 – Class probability evolution given by random forest with all four responses and the data is unnormalized

Références

[1] Held M., Schmitz M.H., Fischer B., Walter T., Neumann B., Olma M.H., Peter M., Ellenberg J. & Gerlich, D.W. : *CellCognition : time resolved phenotype annotation in high throughput live cell imaging.* **Nature Methods.** August 2010.

[2] Presentation of the MitoCheck project, `http://www.mitocheck.org/`.

[3] The CellCognition software and presentation, `http://www.cellcognition.org/`.

[4] *Technical Report : Feature extraction*, Walter T. January 2007. The paper was added to the annexe, in the following pages.

[5] Huang, Jiayuan, et al. *Correcting sample selection bias by unlabeled data.* Advances in neural information processing systems. 2006.

[6] Bickel, Steffen, Michael Brückner, and Tobias Scheffer. "Discriminative learning for differing training and test distributions." Proceedings of the 24th international conference on Machine learning. ACM, 2007.

Technical Report — Confidential

Technical Report: Feature extraction

Thomas Walter

January 14, 2007

The purpose of this document is to give the list of features which are extracted in order to allow a classification of cell nuclei into phenotypic classes, in the framework of the mitocheck primary screen.

This report contains a simple list of features, with little explanation and no efficiency evaluation.

1 Feature types

The extraction of features is a crucial part of the pattern recognition pipeline. Generally speaking, the less care is taken while choosing the features, the more complicated become the next steps of the pattern recognition pipeline. In our approach, we have used the rather general features defined in [2] and added some other general and some dedicated features, which are more problem dependent, as they include some a priori knowledge and which have been designed in order to allow one to distinguish important phenotypes.

1.1 Feature groups

The features we have used can be divided roughly in the following groups according to the algorithmic method to obtain them:

1. Basic features like size, perimeter, mean grey level, etc.
2. Haralick features.
3. Levelset features.
4. Moment based features.
5. Convex hull features.
6. Granulometry features.

7. Dynamic features.

Alternatively, features can be divided into shape descriptors and grey level descriptors. Grey level descriptors are often identified with texture features, which is not exact. Texture features are supposed to describe substructures and repeated subpatterns of the object.

The grey level (mostly texture) features we have used are:

- Basic grey level features.
- Haralick features.
- Levelset features.
- Granulometry features (based on image volume, applied on the original image).
- Dynamics of the original image.

Our shape descriptors are:

- Basic shape features.
- Moment based features.
- Convex hull features.
- Granulometry features (based on image area, applied on the image mask).
- Dynamics applied on the distance map of the mask.

The basic features, as well as the Haralick features and the levelset features have been published in [2] and used in [5]. The moment based features are basically taken from [6], granulometries have been described in [7] and applied to classification of cell nuclei in haematology in [4]. The convex hull has been used by many authors in order to characterize shape, but some of these features have not yet been used previously to our knowledge. Dynamics are normally used as markers for interesting image structures; in the context of phenotypic classification they have not yet been applied.

1.2 Requirements

A priori knowledge about the object Features should be meaningful, i.e. they should describe the objects according to what we consider to be important or interesting. Many learning methods are quite sensible to meaningless features ("noise").

Invariance If the position and the orientation of the objects is not an important aspect — like it is in our case —, the features must be invariant with respect to translation and rotation. If their size is not important, they must also be scale invariant.

Many features are inherently invariant to rotation and translation. For some features, particular care must be taken (e.g. moments).

Independence Ideally, features should be decorrelated, even if there are classification methods which can deal with strongly correlated features. In practice, this is hard to guarantee: feature reduction methods must often be applied (feature selection or feature combination (e.g. PCA)). Sometimes, there is a mathematical correlation (i.e. the same thing is measured). This can eventually be avoided by a careful choice of features. But sometimes, the correlation lies in the very nature of the objects.

2 Basic features

The basic features characterizing the shape of the objects are:

- `roisize`: the number of pixels N of object pixels.
- `perimeter`: the object perimeter P.
- `dist_max`: the maximal distance Δ_{max} between center of gravity and border pixels.
- `dist_min`: the minimal distance Δ_{min} between center of gravity and border pixels.
- `dist_ratio`: the ratio of minimal to maximal distance between center of gravity and border pixels (which takes 1 as a maximal value for a perfectly round object and 0 for an object with the center of gravity lying on its border (due to a strong concavity).
- `circularity`: is a rough estimation of the "roundness" of the object: $circ = \frac{P}{2\sqrt{\pi N}}$. It should be one for a perfect circle, becoming larger with decreasing similarity to a circle.
- `irregularity`: is a more robust way of calculating circularity, as it does not rely on the perimeter, but on Δ_{max}: $irr = \frac{1+\sqrt{\pi}\Delta_{max}}{\sqrt{N}} - 1$. The measure tends towards 0 for a perfect circle.
- `irregularity2`: is the same as irregularity, but Δ_{max} is replaced by the average difference $\overline{\Delta}$ between center of gravity and the border pixels.
- `n_wdist`: average distance to the center of gravity: $\frac{1}{N}\sum|x-\overline{x}|$.

The basic features characterizing grey level properties of the objects are:

- `min`: the minimum value.
- `max`: the maximum value.
- `n_avg`: the average grey value.
- `n_stddev`: the standard deviation.
- `n_wavg`: the average, weighted by the distance between pixel and center of gravity: $\frac{1}{N}\sum|x-\overline{x}|f(x)$.

- `n_wiavg`: the average, weighted by the inverse distance between pixel and center of gravity: $\frac{1}{N}\sum\frac{f(x)}{|x-\bar{x}|+1}$.
- `n2_avg, n2_stddev, n2_wavg, n2_wavg`: the respective features without normalization.

3 Haralick features

The Haralick features aim at characterizing the texture of objects by means of joint distribution of pixel value combinations.

In histograms, one tries to analyze the frequency of certain grey level values in an image. The inconvenience of this representation is that the spatial distribution of these values is completely lost. One method to address this inconvenience is to record combinations of pixel values at a certain distance. This can be done by the cooccurrence matrix depending on distance d and angle Φ, which takes as element $c_{i,j}^{d,\phi}$ the number of pixel pairs x, y with distance d at angle ϕ, fulfilling $f(x) = i$ and $f(y) = j$.

In order to obtain rotational invariance, the mean of the cooccurrence matrices is calculated for four different angles ($\phi_i = 0^0, 45^0, 90^0, 145^0$). Let $P_{i,j} = \frac{c_{i,j}}{N}$ be the cooccurrence probability for values i and j (for a given distance d in all four angles) and let $\mu = \sum_j j \sum_i P(i,j)$ the mean grey level value and $\sigma^2 = \sum_j (j-\mu)^2 \sum_i P(i,j)$ its variance. Then, the following features can be calculated from the averaged cooccurrence matrix:

- `h2<i>_ASM`: Angular Second Moment (Energy), i.e. $\sum P_{i,j}^2$.
- `h2<i>_IDM`: Inverse Difference Moment (Homogeneity), i.e. $\sum\frac{P_{i,j}}{1+(i-j)^2}$.
- `h2<i>_CON`: Contrast (Inertia), i.e. $\sum P_{i,j}(i-j)^2$
- `h2<i>_VAR`: Variance, i.e. $\sum(i-\mu)^2 P(i,j)$
- `h2<i>_PRO`: Prominence, i.e. $\sum(i+j-2\mu)^4 P(i,j)$
- `h2<i>_SHA`: Shade, i.e. $\sum(i+j-2\mu)^3 P(i,j)$
- `h2<i>_COR`: Correlation, i.e. $-\sum\frac{(i-\mu)(j-\mu)}{\sigma^2}P(i,j)$
- `h2<i>_SAV`: Sum average, i.e. the average of partial sums, subject to the condition $|i+j| = k$: $\sum_k k \sum_{|i+j|=k} P(i,j)$
- `h2<i>_DAV`: Difference average, i.e. $\sum_k k \sum_{|i-j|=k} P(i,j)$
- `h2<i>_SVA`: Sum variance, i.e. $\sum_k (k-SAV)^2 \sum_{|i-j|=k} P(i,j)$
- `h2<i>_ENT`: Entropy, i.e. $-\sum P_{i,j} \log P_{i,j}$
- `h2<i>_SET`: Sum entropy, i.e. $\sum_k \sum_{|i-j|=k} P(i,j) \log \sum_{|i-j|=k} P(i,j)$

- `h2<i>_COV`: Coefficient of variation, i.e. $\frac{\sigma^2}{\mu}$
- `h2<i>_average`: Average μ
- `h2<i>_variance`: Variance σ^2

The Haralick features have been calculated for the distances $d = 1, 2, 4, 8$. We have to note that with increasing distance, the invariance of the features is no longer guaranteed.

For a discussion of the meaning of these features, please refer to [2].

4 Levelset features

Levelset features (or statistical geometric features) are shape features for different levelsets (i.e. results of different thresholds) and, as such, texture features.

For each of the following features, a distribution of values is calculated according to the set of thresholds. For each of these distributions, the maximal feature value, the average feature value, the sample mean and the sample standard deviation are calculated as statistics (`max_value, avg_value, sample_mean, sample_sd`). Furthermore, all features are calculated on the foreground (0) and on the background (1).

- `ls<0,1>_IRGL_<statistic>`: Irregularity
- `ls<0,1>_NCA_<statistic>`: Normalized Number of Connected Regions (number of connected components divided by the area of the object).
- `ls<0,1>_DISP_<statistic>`: Average Clump Displacement (estimation of the average distance between center of gravity of the connected component and the center of gravity of the cell).
- `ls<0,1>_INTERIA_<statistic>`: Average Clump Interia (like average clump displacement, but weighted by the area of each connected component).
- `ls<0,1>_CAREA_<statistic>`: Average Clump Area (average area of the connected components).
- `ls<0,1>_TAREA_<statistic>`: Total Clump Area.

5 Moment based features

Moments and derived features have been initially defined to characterize distributions of values (like histograms), but they can also be used as shape or texture descriptors. Discrete moments m_{pq} are defined as:

$$m_{pq} = \sum x^p y^q f(x, y) \tag{1}$$

with $f(x, y) \in \{0, 1\}$ for shape descriptors and $f(x, y)$ the grey level value for grey level descriptors. In pattern recognition tasks where objects are to be characterized

independently from their position and orientation, translation and rotation invariance are required. This can be achieved by several means. One possibility is to use moment invariants [3] as features, i.e. to define polynomial combinations of moments which are invariant with respect to an affine transformation. A second possibility is to find the principal axis of the pattern and to calculate the moments with respect to this axis and the axis perpendicular to it. This corresponds to a rotation of the object in such a way that its principal axis coincides with the x-axis. These moments are called standard moments [6].

Furthermore, we can see from equation 1, that the moments can be written as $m_{pq} =< x^p y^q, f(x,y) >$ with $< \cdot, \cdot >$ the scalar product. This means that m_{pq} is nothing else than the projection of f on the monomial $x^p y^q$. As the $x^p y^q$ are not orthogonal, the decomposition is suboptimal in terms of redundancy. Therefore, the monomials can be replaced by orthogonal polynomials, like Zernike or Legendre polynomials. A comparison of the resulting methods can be found in [8]. In [6], the authors claim that standard moments give similar performance in pattern recognition tasks as Zernike moments. We have not followed the approach of Zernike or Legendre moments.

- `moment_I1`: algebraic invariant I_1
- `moment_I2`: algebraic invariant I_2
- `moment_I3`: algebraic invariant I_3
- `moment_I4`: algebraic invariant I_4
- `moment_I5`: algebraic invariant I_5
- `moment_I6`: algebraic invariant I_6
- `moment_I7`: algebraic invariant I_7
- `eccentricity`: like circularity, but under the hypothesis of an ellipse (how much does the object look like an ellipse?)
- `gyration_radius`: the radius of a circle centered in the origin with the same second order moments as the object.
- `gyration_ratio`: is the ratio of the gyration radius to the maximal distance Δ_{max}.
- `ellip_major_axis`: major axis of the ellipse having the same second order moments as the object.
- `ellip_minor_axis`: minor axis of the ellipse having the same second order moments as the object.
- `ellip_axis_ratio`: ratio of minor axis to major axis.
- `princ_gyration_x`: distance between the principal axis and a line with the same second order moment as the object.

- `princ_gyration_y`: distance between the axis perpendicular to the principal one and a line with the same second order moment as the object.
- `princ_gyration_ratio`: ratio of the two features above.
- `skewness_x`: projection skewness (normalized third order moment) in the direction of the principal axis.
- `skewness_y`: projection skewness (normalized third order moment) in the direction perpendicular to the principal axis.

6 Convex hull features

The convex hull of a set X (binary image) is the smallest convex set containing X. Important features can be derived from the convex hull and from the set difference $D = Conv(X) \setminus X$. We have defined the following features:

- `convexhull_cc`: number of connected components of D.
- `convexhull_thresh_cc`: the number of large connected components (larger than a threshold) of D.
- `convexhull_area_ratio`: the ratio of the surface of the object to the surface of its convex hull. This feature equals 1 for convex sets, if not, it is smaller.
- `convexhull_rugosity`: the ratio of the object perimeter to the cell perimeter. For convex objects, this feature equals 1, if the object is not convex, it is smaller than 1.
- `convexhull_max_val_<0, 1, 2>`: the 3 maximal areas of the connected components of D. Actually, these values give a hint of how symmetric the concavities are.
- `convexhull_acd`: average clump displacement (average difference between the center of gravity of the cell and the centers of the connected components of D, weighted by their area).
- `convexhull_mean_area`: mean area of the connected components of D.
- `convexhull_variance_area`: area variance of the connected components of D.

To study the maximal areas of the connected components of D is useful in order to find binuclear cell nuclei for instance. A perfect binuclear cell should have two concavities of more or less the same area. A trinuclear cell will have three concavities of similar size, if the three nuclei are positioned on the corners of a triangle. If three nuclei are positioned in a row, the number of concavities is either higher or the concavities have no symmetric size.

The size distribution of concavities is certainly a powerful feature. But we cannot conclude directly the number of possibly involved cell nuclei. The reason is that a "bended" row of several nuclei results on one side in one large concavity and on the other side in several small concavities. In order to deal with this problem, it would be good, to be able to control the degree or size of concavity we are interested in. This can be done by the use of granulometries.

7 Granulometry features

Granulometry allows one to study the size distribution of objects (or structures) in an image. For this, the image is successively simplified by operators which remove all (bright or dark) structures up to a certain size. A record is kept of how much is removed from the image with each filtering step, leading to a distribution of measurements m_i, which can be seen as size dependent texture or shape descriptors.

More mathematically speaking, each anti-extensive (extensive), increasing and absorptive operator ψ with $\psi_0 f = f$ can be used for the definition of a granulometry [7]. These properties make sure that the family of operators behaves like a set of sieves: with increasing size, the sieves remove larger grains of the substance to be sieved. In practice, morphological openings (γ: anti-extensive) and closings (ϕ: extensive) are mostly used for this purpose. In figure 1, the successive application of closings with increasing size are shown in the first row: the concavities are successively filled. In the second row, the successive application of openings with increasing size is shown: small details are successively removed.

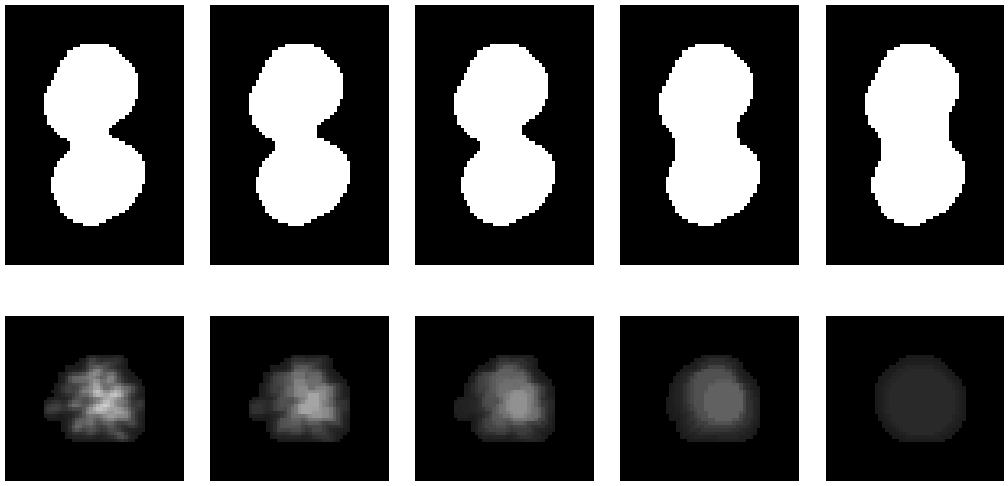


Figure 1: First row: binary granulometry by morphological closing. Second row: grey-scale granulometry by morphological opening.

Let γ_{sB} be a morphological opening with the structuring element B of size s. Then the family $\Gamma = \{\gamma_{sB}\}$ defines a series of operators which can be successively applied to the image f. For each step, we record a measure (e.g. the sum of grey level values) of what has been removed from the image. Hence, we obtain a list of measures $m_i = m(s_i)$ with $s_i \in \{s_1, s_2, s_3 \ldots\}$. We have used the volume and the area, where the image volume $Vol(\cdot)$ means the sum of grey levels and the area $Area(\cdot)$ means the number of non zero pixels.

$$\begin{aligned} v_i^{\phi} &= \frac{Vol(\phi_{s_{i+1}} f - \phi_{s_i} f)}{Vol(f)} \\ a_i^{\phi} &= \frac{Area(\phi_{s_{i+1}} f - \phi_{s_i} f)}{Area(f)} \\ v_i^{\gamma} &= \frac{Vol(\gamma_{s_i} f - \gamma_{s_{i+1}} f)}{Vol(f)} \\ a_i^{\gamma} &= \frac{Area(\gamma_{s_i} f - \gamma_{s_{i+1}} f)}{Area(f)} \end{aligned} \tag{2}$$

The distribution values `granu_<close,open>_<area,vol>_<1,2,3,5,7>` are taken as features.

The features v_i^{ϕ} and v_i^{γ} describe the texture in terms of size of dark and bright substructures. If one of these values is particularly high, there were many substructures of this size in the object. We not that the last image in the second row of figure 1 looks like an interphase. We can state, that the operators have removed and recorded the difference between prometaphase and interphase. However, there may be a problem with the last structuring element: if the cell is too small, the last opening will eventually remove the whole cell. In this case, we would record an abnormally high value.

The features a_i^{γ} and a_i^{ϕ} describe the irregularity of the shape. Whereas a_i^{ϕ} characterize the concavities (more precisely than the convex hull features, see the first row of figure 1), the a_i^{γ} describe prominent spikes.

These features are inherently invariant to rotation, translation, and, due to the division by $Vol(f)$ and $Area(f)$, to scaling.

8 Dynamics

Morphological dynamics [1] aim at finding markers for important structures in an arbitrary image. This is crucial for the application of many segmentation methods, e.g. the watershed transformation.

The dynamic of a minimum is defined as minimal height which has to be climbed in order to reach a lower minimum (see figure 2). For the lowest minimum, the dynamic is not defined; a reasonable choice is the maximal dynamic range of the image, i.e. $\max f(x) - \min f(x)$. The dynamic can be interpreted as a local contrast measure: it is the "depth of the lake" assigned to a minimum.

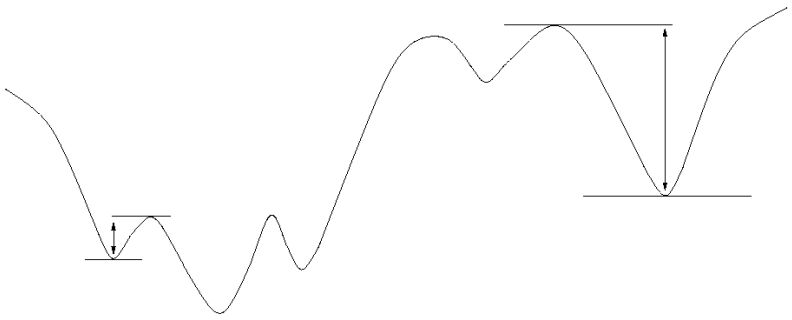


Figure 2: The definition of dynamics of minima.

If there are several maxima (or minima) with sufficiently high dynamic, this means that there are several prominent, high contrasted bright (or dark) substructures in the cell nucleus. This can be quantified by the following measures:

- `dyn_<maxima, minima>_<i>`: i-th highest dynamic.
- `dyn_<maxima, minima>_mean`: mean of dynamics.
- `dyn_<maxima, minima>_number_high`: number of significant maxima or minima
- `dyn_<maxima, minima>_number`: number of maxima or minima.

In this sense, the dynamics define a texture descriptor which does not, like the level set features, depend on a particular choice of a threshold set. On the other hand, they measure only the depth (or height) of a minimum (or maximum), but not the geometry at different levels.

With the same algorithm, it is also possible to define a shape feature with high potential. The idea is the following: let D_X be the distance map of set X (binary image), i.e. $D_X(x)$ is the distance of pixel $x \in X$ to the nearest pixel $y \notin X$. If the object corresponds to an ellipse for instance, we can expect one prominent maximum in the distance function. If it could be decomposed into two overlapping ellipses, where each of these ellipses are well recognizable, on would expect two prominent maxima. Actually, if the basic shapes are sufficiently prominent, the number of prominent maxima should be the same as the number of the basic shapes. We therefore add the following features:

- `dyn_distance_radius_<0,1,2,3>`: i-th highest dynamic of the distance function corresponding to the
- `dyn_<maxima, minima>_mean`: mean of dynamics.
- `dyn_<maxima, minima>_number_high`: number of significant maxima or minima
- `dyn_<maxima, minima>_number`: number of maxima or minima.

References

[1] Michel Grimaud, *La géodésie numérique en morphologie mathématique. application à la détection automatique de microcalcifications en mammographie numérique*, phd thesis, Center of Mathematical Morphology, Paris School of Mines, December 1991. 9

[2] Michael Held, *Automatic identification of mitotic rnai phenotypes in digital images of live human cells*, Diplomarbeit, Technische Universität Dresden, Department of computer science - artificial intelligence institute, April 2005. 1, 2, 5

[3] Ming-Kuei Hu, *Visual pattern recognition by moment invariants*, IRE Transactions on Information Theory **IT** (1962), no. 8, 179–187. 6

[4] Jesus Angulo Lopez, *Morphologie mathématique et indexation d'images couleur. application à la microscopie en biomédecine*, phd thesis, Center of Mathematical Morphology, Paris School of Mines, December 2003. 2

[5] Beate Neumann, Michael Held, Urban Liebel, Holger Erfle, Phill Rogers, Rainer Pepperkok, and Jan Ellenberg, *High-throughput rnai screening by time-lapse imaging of live human cells*, Nature Methods (2006), no. 3, 385–390. 2

[6] Richard J. Prokop and Anthony P. Reeves, *A survey of moment-based techniques for unoccluded object representation and recognition*, CVGIP: Graphical Models and Image Processing **54** (1992), no. 5, 438–460. 2, 6

[7] Jean Serra, *Image analysis and mathematical morphology*, Academic Press, Inc., Orlando, FL, USA, 1983. 2, 8

[8] Cho-Huak Teh and Roland T. Chin, *On image analysis by the methods of moments*, IEEE Trans. Pattern Anal. Mach. Intell. **10** (1988), no. 4, 496–513. 6