Machine learning-based tools to model and to remove the off-target effect

Article *in* Pattern Analysis and Applications · April 2015

DOI: 10.1007/s10044-015-0469-z

CITATION
1

READS
164

4 authors, including:

Riwal Lefort
Idiap Research Institute
29 PUBLICATIONS **202** CITATIONS
SEE PROFILE

Ludovico Fusco
GlaxoSmithKline
42 PUBLICATIONS **356** CITATIONS
SEE PROFILE

Francois Fleuret
University of Geneva
178 PUBLICATIONS **8,451** CITATIONS
SEE PROFILE

Pattern Anal Applic
DOI 10.1007/s10044-015-0469-z

THEORETICAL ADVANCES

Machine learning-based tools to model and to remove the off-target effect

Riwal Lefort[1] · Ludovico Fusco[2] · Olivier Pertz[2] · François Fleuret[1,3]

Received: 1 July 2014 / Accepted: 15 March 2015

Abstract A RNA interference, also called a gene knockdown, is a biological technique which consists of inhibiting a targeted gene in a cell. By doing so, one can identify statistical dependencies between a gene and a cell phenotype. However, during such a gene inhibition process, additional genes may also be modified. This is called the "off-target effect". The consequence is that there are some additional phenotype perturbations which are "off-target". In this paper, we study new machine learning tools that both model the cell phenotypes and remove the "off-target effect". We propose two new automatic methods to remove the "off-target" components from a data sample. The first method is based on vector quantization (VQ). The second method we propose relies on a classification forest. Both methods rely on analyzing the homogeneity of several repetitions of a gene knockdown. The baseline we consider is a Gaussian mixture model whose parameters are learned under constraints with a standard Expectation–Maximization algorithm. We evaluate these methods on a real data set, a semi-synthetic data set, and a synthetic toy data set. The real data set and the semi-synthetic data set are composed of cell growth dynamic quantities measured in time laps movies. The main result is that we obtain the best recognition performance with the probabilistic version of the VQ-based method.

Keywords Vector quantization · Random forest · Gaussian mixture model · Bioinformatics · Off-target effect

✉ Riwal Lefort
riwal.lefort@unicaen.fr

Ludovico Fusco
ludovico.fusco@unibas.ch

Olivier Pertz
olivier.pertz@unibas.ch

François Fleuret
francois.fleuret@idiap.ch

[1] Idiap Research Institute, Centre du Parc, Rue Marconi 19, PO Box 592, 1920 Martigny, Switzerland

[2] Department of Biomedicine, University of Basel, Mattenstrasse 28, 4058 Basel, Switzerland

[3] Ecole Polytechnique Fédérale de Lausanne (EPFL), Route Cantonale, 1015 Lausanne, Switzerland

1 Introduction

1.1 The off-target effect

The analysis of gene functions is a major challenge classically carried out using RNA interference: one can infer the role of a gene using a RNA molecule that knockdowns its expression, and examines the resulting phenotypical perturbation. For instance, if a cell is smaller after a gene inhibition, one can conclude that the considered gene has an important role regarding the said size. This technique is used for understanding phenomena such as endocytosis [6] and cell migration [2].

However, the gene inhibition technique with RNA remains often imprecise, and may cause the inhibition of additional and non-targeted genes. This phenomenon, which is called the "off-target effect" [12], may induce some additional changes in the phenotype. For instance, a cell may be smaller after the gene inhibition, but it may also have a greater mobility during its growth, due to a hidden gene inhibition.

In this article, we propose novel machine learning-based tools to automatically remove the "off-target" components from a data set of phenotypic measurements.

Published online: 01 April 2015

1.2 State of the art

The usual protocol used to study the relation between gene expression and phenotype is as follows [6, 8]: The phenotype obtained with a given RNA molecule is first modeled by a vector each component of which is associated to a particular feature of the cell (related to its morphological, dynamic, or contextual properties) and depends on the similarity of that feature between normal cells and perturbed cells.

The measure of similarity can be either a standard statistical measure such as the Kolmogorov–Smirnov (KS) test [30], the z test [25], the t test [25], the χ^2 test [6], or it can be a machine learning-based technique such as the neural networks [2], support vector machine (SVM) [31], and random forest [28].

Then, the methodology to deal with the "off-target effects" is applied to the phenotypic vectors coming from each RNA molecule.

A classical method is based on RNA selection. It consists of using multiple RNA molecules targeting the same gene, and keeping only the common phenotypic perturbations. Echeverri et al. [8] stated that if at least 2 out of 3 used RNA molecules produce the same phenotype changes, we can consider it as "on-target". In addition, they recommend to repeat the experiment to validate the results.

An alternative method consists of making an average gene profile. In that case, using again several RNA molecules targeting the same gene, one averages the multiple resulting vector of phenotypic measurements. Collinet et al. [6] compute the most probable phenotype using a Bayesian model.

1.3 Our contributions

We first propose in Sect. 4 a model based on a vector quantization (VQ) [14, 23]. Using the K-means algorithm, we find clusters in the feature space, and use a hard classification rule to determine whether samples in a cluster are "on-target" components, or "off-target" components (Sect. 4.1). In Sect. 4.2, we propose a soft version of this classification rule using a probabilistic K-means. Note that, choosing the optimal number of clusters is not straightforward. In the context of unsupervised learning, some methods assess the optimal number of clusters [19, 20]. These methods start with an empty set of clusters which is filled dynamically using online learning. In unsupervised learning, the objective is to model the sample distribution. By contrast, we tackle the special case of supervised clustering. We do not want to model the sample distribution, we rather want a model of the class specificities and the class mixtures. In this context, assessing the number of clusters is different from unsupervised techniques [19, 20]. We address the supervised selection of the cluster number in Sect. 4.3.

Our second contribution, presented in Sect. 5, is based on the analysis of the Random Forest outputs [4, 5]. We start from the idea that the "RNA entropy" of an on-target component—that is the entropy of the distribution of the RNA used to produce the phenotypes in that cluster—is high, since multiple RNA molecules are represented. Conversely, an "off-target" component resulting from a single RNA molecule leads to a small RNA entropy. To compute this entropy, each sample is associated to a probability vector that is build from tree-based classification steps.

In addition, we propose to use the two methods above as a mean to initialize the expectation–maximization (EM) procedure. Collinet et al. [6] proposed a Gaussian mixture model (GMM) to characterize the "on-target" component, but the parameters of the model are trained from an EM algorithm which is very sensitive to its initialization. We show that our approach drastically improves the recognition performance. The GMM being our baseline, this is the first methodological approach we present, see Sect. 3.

1.4 Data sets

In Sect. 6.1, to interpret the results, we use a synthetic 2D data set.

Both Sects. 6.2 and 6.3 consider an application in the field of neuroscience. We propose to characterize cells' development dynamics in time laps movies [17]. In the field of neurobiology, it has been observed connections between the static morphological properties of a cell and its genotype [27]. More recent works in oncology have shown that studying the cell growth dynamics provides important information about the cell genotype [11, 24]. Following that line of thinking, biologists want to quantify how much neurites' morphodynamic depends on the genotype characteristics.

For instance, Fig. 1a shows one video of neurons for which the gene RhoA has been inhibited, which leads to longer neurites. Figure 1b shows one video of neurons for which the gene Map2K7 has been inhibited which leads to shorter neurites and faster protrusion and retraction process.

The characterization of such aspects of the neurite phenotypes requires to deal with the "off-target effect", which is our core objective.

2 Basic idea

Before we deal with the methodological and experimental aspects, we present the concepts and the basic idea of the process and introduce notation.

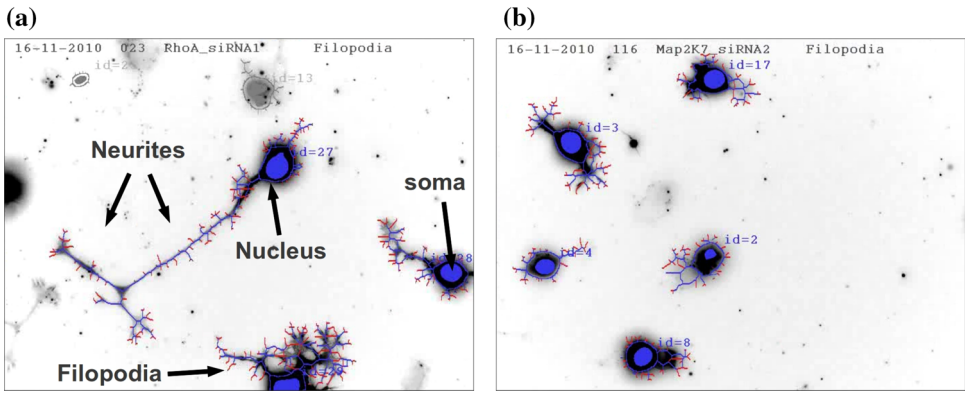


Fig. 1 **a** A video that contains some cells for which the gene RhoA is inhibited. This leads to bigger neurites. **b** A video that contains some cells for which the gene Map2K7 is inhibited. This leads to shorter neurites

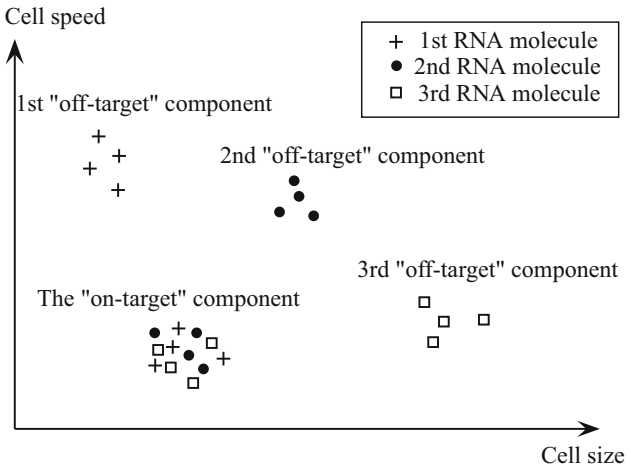


Fig. 2 In this schematic example, we consider that a cell is described by two features ($F = 2$): its speed and its size. In the figure, a *cell* is represented by a vector in the two-dimensional feature space. There are three RNA molecules ($M = 3$) that are represented by *crosses*, *dots*, and *squares*, respectively. The "off-target" components are represented by the clusters that contain vectors from single RNA molecules. The "on-target" component is the cluster that contains vectors from every RNA molecules

We consider that a cell is represented by a set of F features: the soma area, the instantaneous soma speed, the number of neurites, the neurite total length, the instantaneous neurite activity (protruding or retracting), etc. Let $X \in \mathbb{R}^F$ be the feature vector associated to a cell.

We consider M RNA molecules that inhibit the same gene, to which correspond M cell classes. We consider $M + 1$ distributions for X, one corresponding to the "on-target" component, and the M others to the "off-target" components.

This is illustrated in Fig. 2. We consider a synthetic example with three RNA molecules ($M = 3$), and where each cell is represented by two features ($F = 2$): its speed and its size. Each cell is represented by a vector in the two-dimensional feature space. These vectors are depicted by crosses, dots and squares, depending on the RNA molecule which is associated with each cell. We identify four clusters ($M + 1 = 4$). The first cluster is the "on-target" component which is composed of every RNA molecules. The three other clusters are the "off-target" components that are composed of a single RNA molecule each.

The methods we propose aim at characterizing the statistical distribution of each cluster. Then, we can remove the "off-target" components and we are able to extract the phenotypic gene profile. This approach assumes two important properties: (1) the "off-target" components and the "on-target" component are additive in the feature space, and (2) at least one RNA molecule must be absent in each of the "off-target" components.

3 Gaussian mixture model (GMM)

To understand the baseline [6], we introduce the GMM and the EM training procedure. For a random initialization, the method is called "Random+GMM", else, using an other initialization that we call "*init*", the method is designated as "*init*+GMM".

3.1 State of the art

Gaussian mixture models (GMM) are mostly used to model a statistical distribution of vectors. The distribution g of the random variable $X \in \mathbb{R}^F$ has the following form:

$$g(x) = \sum_{c=1}^{C} \rho_c \mathcal{N}(x \mid \mu_c, \Sigma_c) \quad (1)$$

where C denotes the number of components in the mixture, $\mathcal{N}(X \mid \mu_c, \Sigma_c)$ denotes the multivariate normal distribution with mean μ_c and covariance matrix Σ_c, and ρ_c denotes the component prior.

Dempster et al. [7] proposed to assess the parameters $\Theta = \{\rho_c, \mu_c, \Sigma_c\}_{c=1}^{C}$ with an EM algorithm [7]. If we consider a diagonal covariance matrix, K-means [10] is also suitable.

The model can be extended to a classification task where each class is independently modeled using equation (1). Let M be the number of classes, and g_t be the class t distribution. The distribution f of a random variable $X \in \mathbb{R}^F$ has the following expression:

$$f(x) = \sum_{t=1}^{M} \pi_t g_t(x) \tag{2}$$

In supervised learning, the parameters $\{\pi_t, \Theta_t\}_{t=1}^{M}$ are learned independently, class by class, using the usual EM procedure [7], where Θ_t are the parameters of g_t. Note that, the training data set $\{X_n, Y_n\}_{n=1}^{N}$ is composed of the vectors $X_n \in \mathbb{R}^F$ and the labels $Y_n \in \{1, \ldots, M\}$.

In weakly supervised learning, the label Y_n is a hidden variable. The training data set is $\{X_n, \pi_n\}_{n=1}^{N}$, where $\pi_n = \{\pi_{n,t}\}_{t=1}^{M}$ is a vector that indicates the prior probability that sample X_n belong to each class:

$$\pi_{n,t} = p(Y_n = t) \tag{3}$$

In the case of equi-probable prior $\pi_{n,t} \in \{0, 1/M\}$, Bishop and Ulusoy [3] proposed an EM approach to train the parameters $\{\pi_t, \Theta_t\}_{t=1}^{M}$. This method has been extended to any prior value $\pi_{n,t} \in [0, 1]$ in [16].

At iteration (i), given a training data set $\{X_n, \pi_n\}_{n=1}^{N}$ and the current values of the parameters for class t

$$\Theta_t^{(i)} = \left\{\rho_{t,c}^{(i)}, \mu_{t,c}^{(i)}, \Sigma_{t,c}^{(i)}\right\}_{c=1}^{C},$$

the EM procedure consists of: In the first step, which is called the "expectation" step, we compute for each sample both the class posterior:

$$\gamma_{n,t} = \frac{\pi_{n,t} g_t^{(i)}(X_n)}{f^{(i)}(X_n)} \tag{4}$$

and the component posterior:

$$\alpha_{n,t,c} = \frac{\rho_{t,c} \mathcal{N}(X_n \mid \mu_c^{(i)}, \Sigma_c^{(i)})}{g^{(i)}(X_n)} \tag{5}$$

In the second step, which is called the "maximization" step, we update the new set of parameters $\Theta_t^{(i+1)}$:

$$\rho_{t,c}^{(i+1)} = \frac{\sum_{n=1}^{N} \gamma_{n,t} \alpha_{n,t,c}}{\sum_{n=1}^{N} \gamma_{n,t}} \tag{6}$$

$$\mu_{t,c}^{(i+1)} = \frac{\sum_{n=1}^{N} \gamma_{n,t} \alpha_{n,t,c} X_n}{\sum_{n=1}^{N} \gamma_{n,t} \alpha_{n,t,c}} \tag{7}$$

$$\Sigma_{t,c}^{(i+1)} = \frac{\sum_{n=1}^{N} \gamma_{n,t} \alpha_{n,t,c} (X_n - \mu_{t,c}^{(i+1)})(X_n - \mu_{t,c}^{(i+1)})^T}{\sum_{n=1}^{N} \gamma_{n,t} \alpha_{n,t,c}} \tag{8}$$

These two steps are iterated until convergence. We consider that the method converges when the likelihood is stabilized. The number of components C per mixture is assessed by maximizing the likelihood, or using the Mean-shift algorithm [32]. In this paper, we maximize the likelihood to learn the parameters.

3.2 Modeling the RNA perturbations with GMM

Following Sect. 2, we consider cells represented by vectors $\{X_n^m\}_{n=1}^{N_m}$, where $X_n^m \in \mathbb{R}^F$ represents a cell which is obtained from the RNA molecule $m \in \{1, \ldots, M\}$.

Ideally, if there was no "off-target effect", these vectors would all follow the same distribution g_0 that we call the "on-target" component. However, in practice, there is an "off-target" component following the distribution g_m. In this situation, a vector X_n^m is generated by either g_0 or g_m, and its distribution is a mixture of the "on-target" component g_0 and the specific "off-target" component g_m. Let f_m be that mixture distribution. Then, the relation between $\{f_m\}_{m=1}^{M}$ and $\{g_t\}_{t=0}^{M}$ is as follows:

$$f_m = \pi_0^m g_0 + (1 - \pi_0^m) g_m, \tag{9}$$

where $\pi_0^m \in [0, 1]$ is the prior probability that a vector is generated by the "on-target" component. By considering all the RNA molecules, the distribution takes the following expression:

$$f = \sum_{m=0}^{M} (1 - \pi_0^m) g_m \tag{10}$$

where $1 - \pi_0^0 = \sum_{m=1}^{M} \pi_0^m$. The model (10) being the same as the model (2), we can consider the EM algorithm [16] to train the parameters $\{\Theta_t\}_{t=0}^{M}$.

In this application, we can also include some specific constraints. Let Y_n^m be a random variable on $\{0, \ldots, M\}$ which specifies which component generated X_n^m. If $Y_n^m = 0$, sample X_n^m is generated by the "on-target" component g_0, and $\forall t > 0$ if $Y_n^m = t$, sample X_n^m is generated by the "off-target" component g_t. Since we know which RNA molecule is associated to each cell, we have the following rules:

$$\begin{cases} p(Y_n^m = t) = 0 & \text{if} \quad t > 0, t \neq m \\ p(Y_n^m = 0) = \pi_0^m \\ p(Y_n^m = m) = 1 - \pi_0^m \end{cases} \tag{11}$$

In other words, a cell obtained from a RNA molecule m cannot be generated from an "off-target" component of an other RNA molecule $t \neq m$. It is generated by either the "on-target" component or the "off-target" component of the considered RNA molecule m. Then, from the constraints (11), we define the new posteriors at the current iteration (i):

$$\begin{cases} \gamma_{n,t}^m = 0 & \text{if} \quad t \neq m \\ \gamma_{n,t}^m = \dfrac{\pi_0^m g_0^{(i)}(X_n^m)}{f_m^{(i)}(X_n^m)} & \text{if} \quad t = 0 \\ \gamma_{n,t}^m = \dfrac{(1-\pi_0^m) g_m^{(i)}(X_n^m)}{f_m^{(i)}(X_n^m)} & \text{if} \quad t = m \end{cases} \tag{12}$$

This method allows to characterize each component of the distribution of the random variable X_n^m. We can now classify the vector X_n^m as being "on-target" or "off-target". This classification step is achieved using the Bayes rule:

$$\begin{cases} p(Y_n^m = 0 \mid X_n^m, \Theta_0, \ldots, \Theta_M) = \dfrac{\pi_0^m g_0(X_n^m)}{f_m(X_n^m)} \\ p(Y_n^m = m \mid X_n^m, \Theta_0, \ldots, \Theta_M) = \dfrac{(1-\pi_0^m) g_m(X_n^m)}{f_m(X_n^m)} \end{cases} \tag{13}$$

3.3 Estimation of the parameters

The first parameter to set is the number of iterations of the EM algorithm. We stop the iterations when the likelihood $\mathcal{L}\left(\{X_n^m\}, \{\Theta_t^{(i)}\}\right)$ at iteration i is very close to the likelihood at iteration $i+1$. In other words, we stop the iteration whether:

$$\left|\mathcal{L}\left(\{X_n^m\}, \{\Theta_t^{(i+1)}\}\right) - \mathcal{L}\left(\{X_n^m\}, \{\Theta_t^{(i)}\}\right)\right| < \epsilon \tag{14}$$

In practice, using our data set, we noticed that the classification performances do not change after 50 iterations. So, the number of iterations is set to 50. But, if a user does not know a data set, we recommend to train ϵ by grid search.

The second parameter to set is C, i.e., the targeted number of components per class [see Eq. (1)]. The parameter is optimized using a grid search. We choose the optimal value $C \in \{2, 4, 16, 32, 64, 128\}$ that maximizes the likelihood:

$$\hat{C} = \arg\max_C \mathcal{L}\left(\{X_n^m\}, \{\Theta_t\}, C\right) \tag{15}$$

Note that, for our classification task, we do not need a fine evaluation of the number of Gaussian components per class. This value can even be over-estimated to guarantee a proper assessment of the "off-target effect". The cell classification being more important than a good model distribution, we set a rough grid for C.

There is no prior data. Hence, at the initialization step, it is reasonable to give the same weight to both the "off-target effect" and the "on-target effect". As a consequence, each effect has the same probability to appear. So, the priors $\{\pi_0^t\}_{t=0}^M$ are set to 0.5 at the beginning of the EM algorithm.

4 Vector quantization

We investigate in this section an alternative approach to the discovery of the "off-target" effect, which bypass the modeling of the respective distributions, and directly attempts at classifying each sample as either "off-target" or "on-target".

4.1 A hard classification rule (hard-VQ)

This method relies on the analysis of the presence or the absence of the RNA molecules in each region of the feature space. These regions are build using a VQ algorithm. If all the RNA molecules are present in a region of the feature space, we assume that these cells are generated from the "on-target" component. If a RNA molecule is not present in a region of the feature space, we further assume that this region is "off-target". This is illustrated in Fig. 2: the "off-target" components contain only crosses, dots, or squares, while the "on-target" component contains all of the crosses, the dots, and the squares. We call this method "hard-VQ".

Formally, let $\{X_n^m\}_{n=1}^{N_m}$ be a representation of the cells in the feature space, where the vector $X_n^m \in \mathbb{R}^F$ represents a cell which is obtained from the RNA molecule indexed by $m \in \{1, \ldots, M\}$. The first step of the method consists of dividing the feature space into K clusters. This is achieved using the K-means algorithm [10] from all the available vectors $\{\{X_n^m\}_{n=1}^{N_m}\}_{m=1}^M$. Let

$$I_n^m \in \{1, \ldots, K\}$$

be the index of the cluster which is associated to the sample X_n^m, and let

$$H_k = \{H_k^m\}_{m=1}^M$$

be the histogram that of the vector distribution of the M RNA molecules in cluster $k \in \{1, \ldots, K\}$. Each component H_k^m of the histogram H_k takes the following formal expression:

$$H_k^m = \frac{\sum_{n=1}^{N_m} \mathbb{1}_{\{I_n^m = k\}}}{\sum_{j=1}^{M} \sum_{n=1}^{N_j} \mathbb{1}_{\{I_n^j = k\}}} \tag{16}$$

where $\mathbb{1}_{\{I_n^m = k\}} = 1$ if $I_n^m = k$, and $\mathbb{1}_{\{I_n^m = k\}} = 0$ otherwise.

The second step of the method consists of classifying each cluster into either "off-target" or "on-target". To this end, we analyze the histograms $\{H_k\}_{k=1}^K$. Let $d(k) \in \{0, 1\}$ be the classification result, i.e., the decision, where $d(k) = 1$ if the kth cluster is considered as "on-target", and $d(k) = 0$, if the kth cluster is considered as "off-target". The classification rule is defined as follows:

$$d(k) = \mathbb{1}_{\{\forall m,\, H_k^m > 0\}} \tag{17}$$

In other words, if all the RNA molecules are present in the kth cluster, all the components of the vector H_k are different from zero, then, this cluster is considered as an "on-target" component. If there is at least one RNA molecule which is absent in the kth cluster, there is at least one component of the vector H_k which equals to zero, then we consider this cluster as an "off-target" component.

Finally, we define the RNA class posterior $p(Y_n^m \mid X_n^m, H_1, \ldots, H_K)$, where Y_n^m is defined in Sect. 3 from the decision rule (17), with

$$\begin{cases} p\left(Y_n^m = t \mid X_n^m, H_1, \ldots, H_K\right) = 0 \quad \text{if } \mathrm{t} > 0, \mathrm{t} \neq \mathrm{m}, \\ p\left(Y_n^m = 0 \mid X_n^m, H_1, \ldots, H_K\right) = \mathbb{1}_{\{d(I_n^m)=1\}}, \\ p\left(Y_n^m = m \mid X_n^m, H_1, \ldots, H_K\right) = \mathbb{1}_{\{d(I_n^m)=0\}}. \end{cases} \tag{18}$$

In other words, if the vector X_n^m is in a cluster classified as "on-target", we set $Y_n^m = 0$, and if the vector X_n^m is in a cluster classified as "off-target", we set $Y_n^m = m$.

Note that, in this paper, the K-means procedure is initialized using the K-means++ algorithm [1].

4.2 A soft version of the classification rule (soft-VQ)

We propose to extend the hard classification rule (18) to a soft version, using a probabilistic version of the K-means. This method is called "soft-VQ". The idea is to generate several clusterings and to merge them together. By doing this, we aim to reduce the influence of the K-means initialization, and then, we expect to improve the recognition performance. The main reason of that improvement comes from the reduction of the noise brought by the vectors that are close to the cluster boundaries.

The algorithm is based on bootstrapping. We want to combine B classifiers. At step $b \in \{1, \ldots, B\}$, to build a classifier, we apply the K-means algorithm on a random sub-population of the training data. This sub-population is composed of vectors which are sampled at random among $\{\{X_n^m\}_{n=1}^{N_m}\}_{m=1}^{M}$. The proportion of sampled training data is denoted by α. Using the classification rule (17), we classify the clusters at step b into "on-target" component or "off-target" component. Following the "Bagging" idea [4], the final decision to classify X_n^m as being either an "on-target" component or an "off-target" component is taken from a vote.

Formally, let $d^b(k) \in \{0, 1\}$ be the decision taken by the classifier at step b, where $d^b(k) = 1$ if the kth cluster is considered as "on-target", and $d^b(k) = 0$, if the kth cluster is considered as "off-target". Then, the posterior takes the formal expression:

$$\begin{cases} p\left(Y_n^m = 0 \mid X_n^m, H_1, \ldots, H_K\right) = \dfrac{1}{B}\displaystyle\sum_{b=1}^{B} \mathbb{1}_{\{d^b(I_n^m)=1\}} \\ p\left(Y_n^m = m \mid X_n^m, H_1, \ldots, H_K\right) = \dfrac{1}{B}\displaystyle\sum_{b=1}^{B} \mathbb{1}_{\{d^b(I_n^m)=0\}} \end{cases} \tag{19}$$

Note that, in this paper, we set the number of K-means to $B = 100$ and the proportion of random training data to $\alpha = 0.5$.

4.3 Estimation of K

Setting K is very important and not straightforward. If K is too high, all the clusters may contain only one type of RNA molecule. In that case, all the samples may be considered to be "off-target". On the other hand, if K is too small, we can expect that all the RNA molecules are present in each cluster. In that case, all the cells will be considered to be "on-target". To address this issue, we propose an optimization criterion which forces the vector distributions of the "on-target" components to be similar and the vector distributions of the "off-target" components to be different.

Let $\{X_n, Y_n\}_{n=1}^{N}$ be the training data set, where $X_n \in \mathbb{R}^F$ is a cell and $Y_n \in \{1, \ldots, M\}$ indicates the index of the RNA molecule. From either the posterior (18) or the posterior (19), we derive the decision rule $d_n(K)$ which is similar to the decision (17). If X_n is considered to be on-target, $d_n(K) = 0$. Else, if X_n is considered to be "off-target", $d_n(K) = 1$. We want a value of K that both minimizes the distance between the "on-target" distributions and maximizes the distance between the "off-target" distributions.

To achieve this, the optimal value of K is defined as:

$$\hat{K} = \arg\max_K [\Delta_1(K) - \Delta_0(K)] \tag{20}$$

where

$$\Delta_t(K) = \sum_{m_1 \neq m_2} \delta\left[\mu_t^{m_1}(K), \mu_t^{m_2}(K)\right]$$

is an average distance and $\mu_t^m(K) \in [0, 1]^{K'}$ denotes a model of the distribution $\{X_n \mid Y_n = m, d_n(K) = t\}$. The distribution function $\mu_t^m(K)$ is computed using the bag-of-word model [18]. In other word, $\mu_t^m(K)$ is a histogram that counts the number of samples in $\{X_n \mid Y_n = m, d_n(K) = t\}$ that are in each of K' clusters. The K' clusters are computed using a K-means. The Bhattacharyya distance is used to compute the distance:

$$\delta\left[\mu_t^{m_1}(K), \mu_t^{m_2}(K)\right] = 1 - \sum_{k=1}^{K'} \sqrt{\mu_{t,k}^{m_1}(K)\mu_{t,k}^{m_2}(K)} \tag{21}$$

For a given set of values $K \in \{K_1, \ldots, K_Q\}$, we noticed that the value ranges of $\Delta_1(K)$ and $\Delta_0(K)$ are different. To have the same dynamics for both distance measures, the measures are normalized as follows:

$$\Delta_t(K) \leftarrow \frac{\Delta_t(K) - \min\{\Delta_t(K)\}_{K=K_1}^{K_Q}}{\max\{\Delta_t(K)\}_{K=K_1}^{K_Q}} \tag{22}$$

This normalization forces the distance measures in the range from 0 to 1.

In this paper, the optimization criterion (20) is solved using a grid search where $K \in \{2^q\}_{q=2}^{12}$, and the number of clusters for the bag-of-words model is set to $K' = 1024$ [see Eq. (21)].

5 Random forest

Our last method is based on bagging and Random Forests [4, 9, 15]. We train a bag of decision trees to predict the RNA molecule associated to a certain phenotypical vector, and we estimate how the said molecule can be predicted from the phenotype using the posterior entropy. We will refer to this method as "forest".

The idea is that if a cell is "on-target", it is difficult to know from which RNA molecule it is from, since all the RNA molecules are present in an "on-target" component. Thus, if we try to classify the cells among the RNA molecules, the classification error rates of the "on-target" cells will be high. The RNA entropy which characterizes the RNA probability will also be high.

On the other hand, given a set of cells that are "off-target", it is reasonable to suppose that the cell classification error will be low since an "off-target" component contains less RNA molecules than an "on-target" component. Thus, in the case of an "off-target" component, the RNA entropy is low.

From these properties, we can derive an entropy-based index to characterize the level of RNA uncertainty and to know whether a cell is either "on-target" or "off-target". In the next part of this section, we formulate the problem in a formal way.

Let B be the number of trees in the forest. To train a tree f_b of the forest, the data set $\{X_n, Y_n\}_{n=1}^N$ is split randomly into a train set and a test set, where $X_n \in \mathbb{R}$ is a cell and $Y_n \in \{1, \ldots, M\}$ is the index of the RNA molecule. Then, for a given test sample X, we can assign a class $f_b(X) \in \{1, \ldots, M\}$. From the forest, we compute the class probabilities $\gamma_n = \{\gamma_{n,1}, \ldots, \gamma_{n,M}\}$, where $\gamma_{n,m} \in [0, 1]$ and $\sum_{m=1}^M \gamma_{n,m} = 1$. Formally, $\gamma_{n,m}$ is related to the occurrence of each class:

$$\gamma_{n,m} = \frac{1}{B} |\{b \text{ s.t. } f_b(X_n) = m\}| \tag{23}$$

From the probability vector γ_n, we propose an entropy-based rule to decide whether X_n is "on-target" or "off-target". The entropy h_n allows characterizing the homogeneity of the vector γ_n:

$$h_n = -\sum_{m=1}^{M} \gamma_{n,m} \log \gamma_{n,m}$$

We propose the following rule:

$$\begin{cases} \text{if } h_n > \dfrac{1}{N}\displaystyle\sum_{n'=1}^{N} h_{n'}, & X_n \text{ is ``on-target''} \\ \text{if } h_n < \dfrac{1}{N}\displaystyle\sum_{n'=1}^{N} h_{n'}, & X_n \text{ is ``off-target''} \end{cases} \tag{24}$$

More intuitively, if h_n is low, X_n is always classified in the same class, which means that it is from an "off-target" component. While if h_n is high, X_n is classified among each of the RNA molecule, which means that it is from a "on-target" component. In addition, because we know that an "on-target" component is composed of all the classes, we consider the following constraint that if:

$$\prod_{m=1}^{M} \gamma_{n,m} = 0 \tag{25}$$

the vector X_n is considered as "off-target".

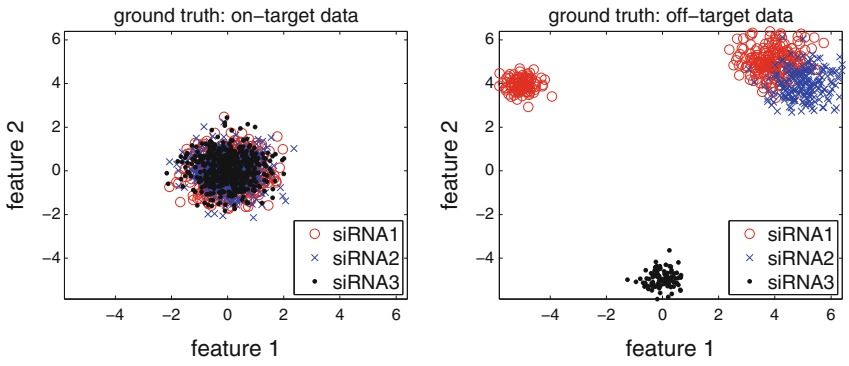
In this paper, the trees are trained using the CART algorithm [5] and we set the number of trees in the forest to $B = 100$. In addition, the proportion between the random training data and the random test data is set to $\alpha = 0.5$.

6 Experiments

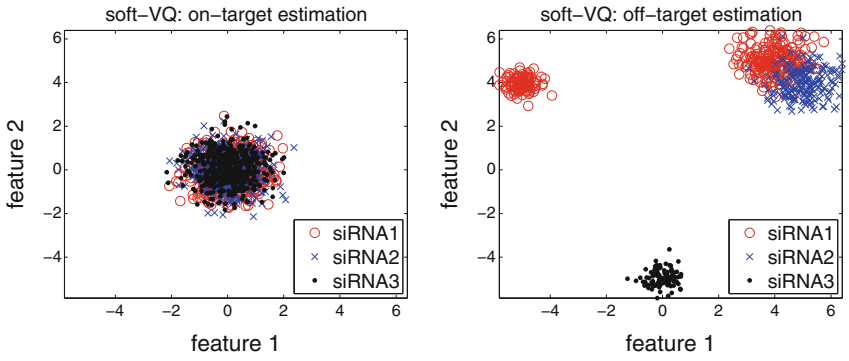
6.1 Toy examples

In this section, a subjective analysis is suggested to show that the proposed methods are able to distinguish the "off-target" components from the "on-target" component. In this way, we generate a random data set that is visually easy to interpret in a 2D feature space. Both the "on-target" component and the "off-target" components follow Gaussian distributions, such that the means and the variances are chosen arbitrarily.

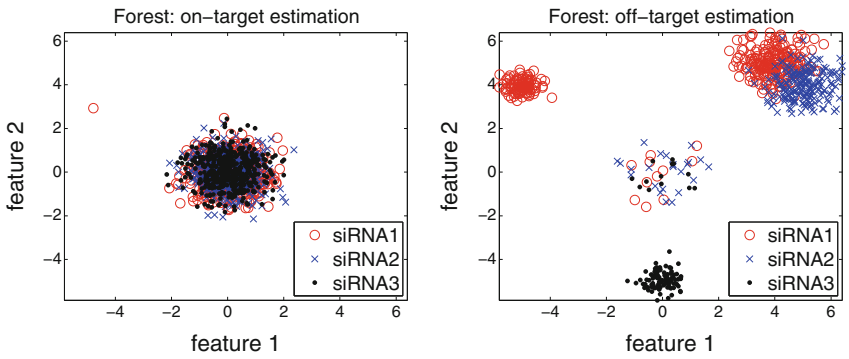
The results are given in Fig. 3. Figure 3a shows the raw data, i.e., the data as they are generated. Figure 3b shows the classification results for soft-VQ (Sect. 4). Figure 3c shows the classification results for the random forest (Sect. 5). Figure 3d shows the classification results for the GMM (Sect. 3).



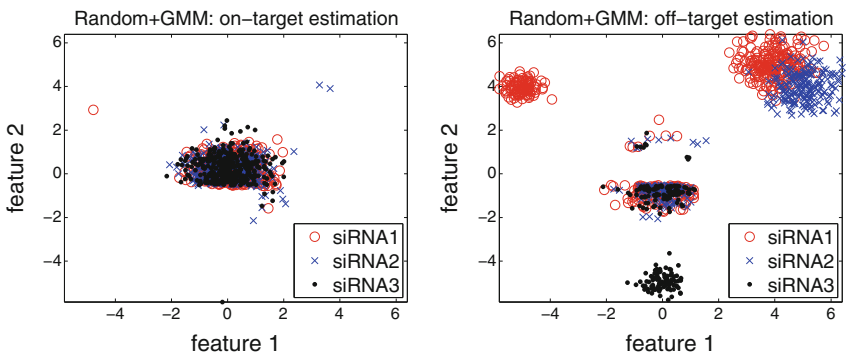
(a) Ground truth



(b) Estimation using soft-VQ



(c) Estimation using the forest



(d) Estimation using Random+GMM

Fig. 3 Subjective analysis from a toy synthetic data set. **a** The ground truth. **b** Estimation using a soft vector quantization (soft-VQ). **c** Estimation using a random forest (Forest). **d** Estimation using a Gaussian mixture model (Random+GMM)

The similarity between the estimated components in Fig. 3b and the ground truth in Fig. 3a shows that the method soft-VQ is able in that case to recognize the "off-target" components. The forest and the GMM [6] filtered a lots of the "off-target" vectors as well, but are less precise.

6.2 Semi-synthetic data set

The subjective analysis in Sect. 6.1 is used to illustrate the issue and the kinds of results we obtain. But, a robust quantitative analysis must be carried out to compare the methods using real data. We use cells in time laps movies (Fig. 1).

Initially, each cell is represented by 240 features. Some features characterize both the soma and nucleus shapes in each frame: Position, area, eccentricity, major axis length, minor axis length, perimeter, and circularity. We also characterize the distribution of these values in a video using the quantiles $\{0, 0.25, 0.5, 0.75, 1\}$. In addition, we compute the dynamical version of these static measures by computing the Euclidean distance between the feature values in two consecutive frames. For instance, the instantaneous speed is the dynamical version of the position. To be more exhaustive, we add both the instantaneous acceleration, and the distance traveled, and we compute their distributions in the video using the quantiles.

Then, for all the static features, we compute the frequency of the protrusion–retraction process, i.e., the number of times a soma is growing or retracting. Other features characterize a neurite in each frame: number of branches, number of leaves, distribution of the branch lengths (using the quantiles), the extreme length (the longest path from the root to a leaf), the total cable length, and the neurite complexity that equals the extreme length divided by the number of branches. Again, to have a neuron-based measure, we characterize their distribution using the quantiles. We also characterize the neurite feature distribution in a video using the quantiles. In addition, the instantaneous number of neurites is computed as well as its distribution in the video. Then, we compute the frequency of the protrusion–retraction process, i.e., the number of times a neurite is growing or retracting.

Finally, some features characterize the change between two images in the video. They are based on the entropy or they are pixel based. Their distribution is computed using the quantiles.

To reduce the complexity, we select the 20 best features using the KS test [30], which leads to a vector of 20 components for each cell.

Because of many possible cell phenotypes, it is not straightforward to have an annotated data set. That is why we built a semi-synthetic data set from a set of 17 RNA molecules. To do this, we first evaluate the phenotypic distance using the classification accuracy as measure of the distance. The classification accuracy is computed from 100 cross validation steps using a Support Vector Machine with a Gaussian kernel [29]. Second, using the distance matrix that is given in Fig. 4, we choose the $M + 1$ RNA molecules that are the most further apart from each other. Then,

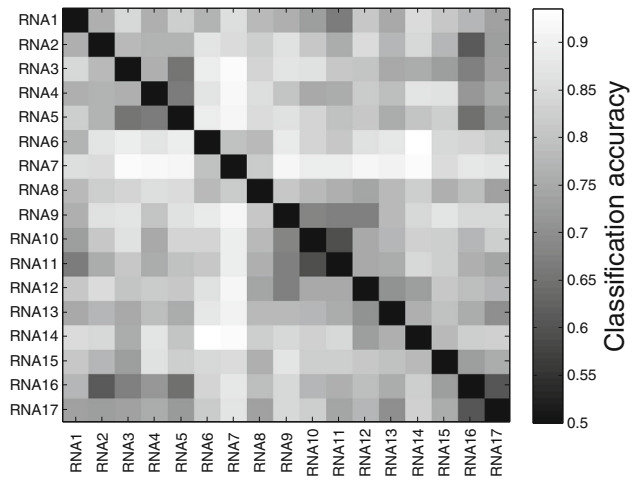


Fig. 4 Measure of the phenotypic distance between each RNA molecule. The higher the classification accuracy the higher the distance

Table 1 Recognition performances as a function of the number of virtual RNA molecules

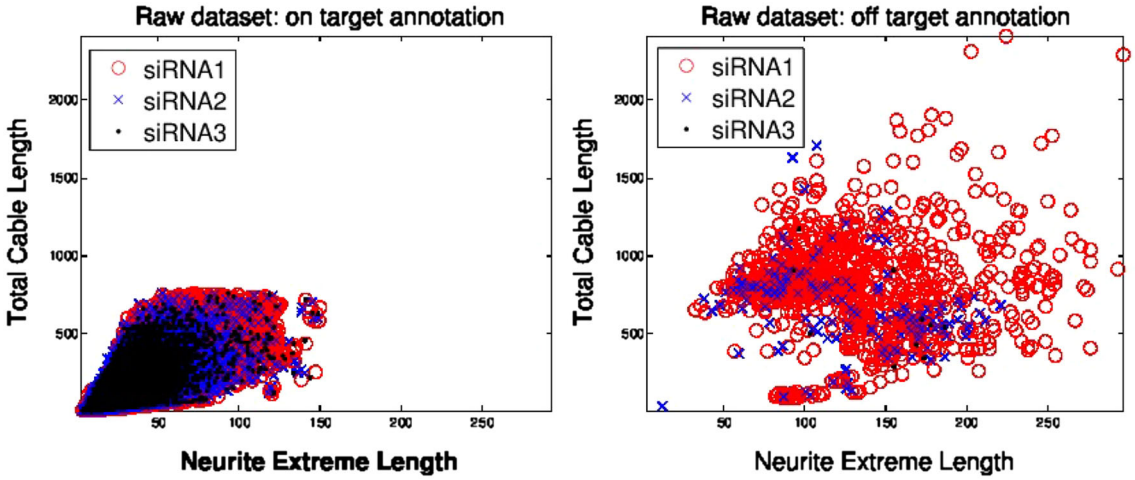
Number of RNA molecules	$M = 2$	$M = 3$	$M = 4$	$M = 5$
Hard-VQ	0.70 ± 0.13	0.65 ± 0.17	0.62 ± 0.17	0.61 ± 0.17
Hard-VQ+GMM	0.71 ± 0.11	0.70 ± 0.09	0.68 ± 0.09	0.64 ± 0.10
Soft-VQ	0.73 ± 0.11	0.70 ± 0.21	0.63 ± 0.18	0.63 ± 0.17
Soft-VQ+GMM	0.73 ± 0.12	**0.73** ± 0.10	**0.69** ± 0.10	**0.66** ± 0.09
Forest	**0.77** ± 0.06	**0.73** ± 0.05	**0.69** ± 0.06	0.64 ± 0.09
Forest+GMM	0.73 ± 0.08	0.68 ± 0.06	0.65 ± 0.05	0.61 ± 0.04
Random	0.37 ± 0.06	0.33 ± 0.04	0.31 ± 0.03	0.30 ± 0.03
Random+GMM	0.51 ± 0.05	0.50 ± 0.04	0.50 ± 0.04	0.50 ± 0.03

The reported value represents the mean correct classification rate

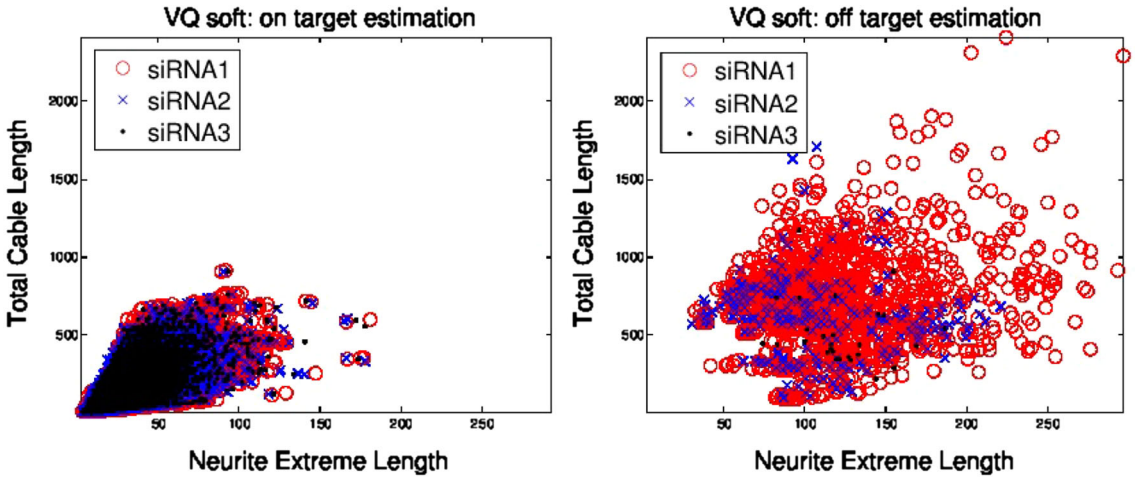
one of the RNA molecule is chosen at random to be the "on-target" component, and the remaining RNA molecules are associated to each of the M RNA "off-target" components. By doing this, we build a data set which ensures both the "on-target" and the "off-target" components to have different distributions, while making these distributions realistic. To statistically characterize the recognition performance and to identify the best method, this random procedure is repeated 100 times. At the end, we compute a mean correct classification accuracy, that is the average between the correct classification rate of the "on-target" class and the correct classification rate of the "off-target" class. The standard deviation is also computed to indicate the unstable methods.

All the results are given in Table 1. For each method, we plot the classification accuracy as a function of M, the number of RNA molecules. We not only compare the GMM (Sect. 3), the hard-VQ (Sect. 4.1), the soft-VQ (Sect. 4.2), and the forest (Sect. 5), we also initialize the GMM using hard-VQ, soft-VQ, and the forest. The method "Random" is a random classification that provides a lower bound. The best performances are given in bold.

From these results, we can make several observations. First of all, soft-VQ and the forest outperform the other methods. Second, in comparison to the GMM [6], we drastically improve the recognition performance using the proposed methods or by initializing the GMM with the proposed methods. This improvement goes from about 10 % to about 25 % in the best case. Also, the probabilistic K-means we proposed (soft-VQ) outperforms the hard decision rule (hard-VQ) with a range from 1 to 5 %. In addition, these results point out that the EM algorithm



(a) Ground truth



(b) Estimation using soft-VQ

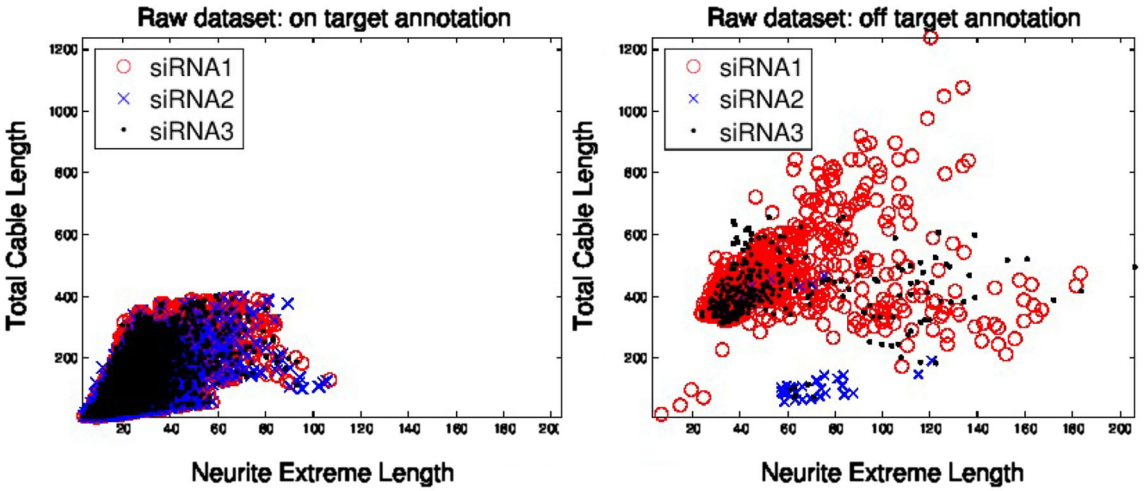
Fig. 5 Classification results for the gene HGS: **a** the raw data set, **b** estimation using soft vector quantization (soft-VQ)

needs a proper initialization. For instance, there is a 23 % improvement by initializing the GMM with the soft-VQ ($M = 2$ and $M = 3$). To be done with the observations, we notice that the VQ-based methods have a standard deviation which is about 15, while it is about 5 for the forest and the GMM. Thus, in comparison to soft-VQ, soft-VQ+GMM allows both improving the recognition performance and reducing the instability.

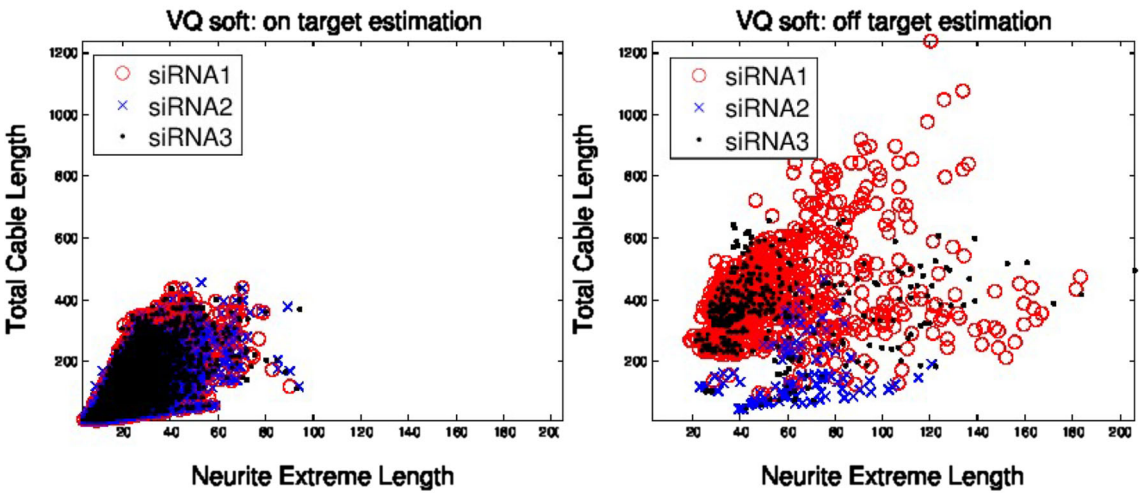
6.3 Real data set

In collaboration with biologists, we manually annotated a data set. We want to characterize the neurite outgrowth of the cells in time laps movies (Fig. 1). A neurite is represented by three features: the total cable length of the neurite, the extreme length of the neurite, and the number of branches in the neurite. We have two sets of movies: cells for which the gene HGS is knockdown and cells for which the gene TRAF2 is knockdown. For each of these two set of movies, there are three sub-set of movies, each sub-set being related to a RNA molecule.

To annotate the data, we use the following protocol. For each possible pair of features, we plot the data in the 2D feature space, and then, we manually designate the "off-target" vectors. The annotation result is shown in Fig. 5a for the gene HGS and in Fig. 6a for the gene TRAF2. The three RNA molecules are, respectively, represented by red circles, blue crosses, and black dots. As waited, the "on-target" component contains cells from the three RNA molecules, while the "off-target" component contains cells from either one or two RNA molecules. Figs. 5b and 6b show the classification results using the soft-VQ model.



(a) Ground truth



(b) Estimation using soft-VQ

Fig. 6 Classification results for the gene TRAF2: **a** the raw data set, **b** estimation using soft vector quantization (soft-VQ)

This subjective analysis demonstrates that we properly identify the different components.

To quantify the error rate, we provide the classification accuracy as well. The recognition performances are given in Table 2 for each of the methods. The best recognition performances are given in bold. As previously, our proposed methods allow improving drastically the classification accuracy in comparison to the GMM baseline [6] in a range from 8 to 26 %. In addition, our proposed methods are better for the GMM initialization than a random GMM initialization (about 10 %). In comparison to Sect. 6.2, the VQ-based methods clearly outperform the forest. This is due to the default values of the forest parameters (number of trees and training proportion) that are not fitted to this new data set. The advantage of the VQ-based methods is that we estimate the optimal number of clusters (Sect. 4.3).

Note that, these results mainly depend on the quality of the data annotation. The fact that we obtain more than 90 % of classification accuracy proves that the annotation procedure has been well designed.

In Table 3, we report an analysis of the main parameter sensitivity. The four main parameters are the bootstrapping parameters (B and α) that are used for both Forest and soft-VQ. We characterize their sensitivity by computing the average standard deviation of the classification accuracy. If the standard deviation is high, the dispersion of the classification accuracy is high. Regarding the parameter B, we note that the accuracy dispersion is lower if $B \in \{100, 200, 300, 400, 500\}$. This means that it is equivalent to use $B = 100$ and $B = 500$. In addition, we note that the classification performance increases from $B = 1$ to $B = 100$ and becomes steady from $B = 100$. For this reason, $B = 100$. Regarding the parameter α, we note that the dispersion always increases when we remove the extreme values of *alpha* (either $\alpha = 0.1$ or $\alpha = 0.9$). As a consequence, setting $\alpha = 0.5$ is a good compromise that ensures a better choice than random.

Table 2 Classification accuracy for the genes TRAF2 and HGS

Gene names	HGS	TRAF2
Hard-VQ	0.94	0.91
Hard-VQ+GMM	0.80	0.84
Soft-VQ	**0.96**	**0.94**
Soft-VQ+GMM	0.83	0.81
Forest	0.76	0.80
Forest+GMM	0.78	0.83
Random	0.25	0.25
Random+GMM	0.70	0.76

The reported value represents the mean correct classification rate

Table 3 In this table, we report an analysis of the sensitivity of the two main parameters B and α that define the bootstrapping environment of both Forest and soft-VQ

parameters	Default	Method	HGS (%)	TRAF2 (%)
$B \in B_1$, $\alpha = \alpha_1$	$B = 100$	Forest	3.9	3.6
$B \in B_2$, $\alpha = \alpha_1$	$B = 100$	Forest	1.4	1.6
$B \in B_1$, $\alpha = \alpha_1$	$B = 100$	Soft-VQ	1.3	1.2
$B \in B_2$, $\alpha = \alpha_1$	$B = 100$	Soft-VQ	0.5	0.2
$\alpha \in \alpha_1$, $B = B_1$	$\alpha = 0.5$	Forest	2.9	3.6
$\alpha \in \alpha_2$, $B = B_1$	$\alpha = 0.5$	Forest	2.2	2.1
$\alpha \in \alpha_3$, $B = B_1$	$\alpha = 0.5$	Forest	1.7	3.6
$\alpha \in \alpha_4$, $B = B_1$	$\alpha = 0.5$	Forest	2.7	2.9
$\alpha \in \alpha_1$, $B = B_1$	$\alpha = 0.5$	Soft-VQ	7.3	8.6
$\alpha \in \alpha_2$, $B = B_1$	$\alpha = 0.5$	Soft-VQ	3.7	1.3
$\alpha \in \alpha_3$, $B = B_1$	$\alpha = 0.5$	Soft-VQ	7.5	10.8
$\alpha \in \alpha_4$, $B = B_1$	$\alpha = 0.5$	Soft-VQ	1.7	0.6

We report the average standard deviation of the classification accuracy. To study the sensitivity of the parameter B, we compute the standard deviation of the classification accuracy for each value of α given a value of B, then we average the standard deviation. To study the sensitivity of the parameter α, we compute the standard deviation of the classification accuracy for each value of B given a value of α, then we average the standard deviation. Let $B_1 = \{10, 50, 75, 100, 200, 300, 400, 500\}$, $B_2 = \{100, 200, 300, 400, 500\}$, $\alpha_1 = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, $\alpha_2 = \{0.3, 0.5, 0.7\}$, $\alpha_3 = \{0.1, 0.3, 0.5\}$, and $\alpha_4 = \{0.5, 0.7, 0.9\}$

6.4 Discussion

The results in Tables 1 and 2 reveal that the forest-based method is less robust. Actually, in Table 1 the forest gives similar to the soft-VQ, while in Table 2 the VQ-based method largely outperforms the forest in a range from 14 to 20 %. The explanation of this phenomenon is the fact that we set the number of trees in the forest to $B = 100$, the data proportion to train a tree at $\alpha = 0.5$. These values are well set in Table 1, but this is not true in Table 2. To improve the chosen value, we could train the optimal value in a grid search, as it is done for the GMM in Sect. 3.3.

We now discuss the fact that the standard deviation of the VQ-based methods is much higher than the forest and the random classification. The VQ methods are trained in an unsupervised context. The consequence being that cluster definition only depends on the random centroid initialization. Hence, the cell classification may strongly vary between two VQ realizations. On the other hand, both GMM and Forest are trained in a supervised manner. In such a case, clusters are not only driven by centroid initialization, but also by the class distributions. This constraint forces the clusters to converge to steadier results, where clusters are centered around the class mixtures or the class specificities. The consequence is a reduction of both the randomness of events and the standard deviation. Note that, the random classification is based on a random class

attribution to each cell. Hence, it is not surprising to have similar error rates between several consecutive experiments.

Now, we address the convergence of EM. In Table 1, VQ+GMM outperforms VQ, but Forest outperforms Forest+GMM. On the contrary, in Table 2, VQ outperforms VQ+GMM while Forest+GMM outperforms Forest. The theory behind this results is that, using EM, the likelihood does not necessarily converge towards a global maximum. Instead, any local extrema such as local maximum, local minimum or saddle point, may be reached. The consequence is that the likelihood may slightly decrease after an EM procedure, especially if the initialization is already interesting. In their works, McLachlan and Krishnan give such examples [21].

7 Conclusion

We have proposed several implementations of a common idea to detect automatically the "off-target" population in a set of samples. One is based on VQ, and the other on a forest of decision trees. Both of these methods are able to cope with large data sets in high-dimension space.

A necessary condition common to all our methods is to have at least one phenotype which is the same for every RNA molecules. In other words, we force a cell to be considered as "off-target" from the moment a RNA molecule is not represented. However, these condition can be relaxed to extend the approach to more general situations. For instance, when lots of different RNA molecules are used to knockdown a gene, we may be interested in considering a percentage of RNA molecules having similar phenotypic components instead of the majority. We could stipulate that, if 80 % of the RNA molecules produce the same phenotype, this phenotype is "on-target". We want to point out that the rules (17) and (25) can easily be changed to use this soft criterion.

In the experimental sections, we compared our proposed models to a baseline based on a GMM [6]. In addition, we proposed to initialize the GMM training step in a proper way, i.e., using our proposed methods. These experiments have been conducted from a real data set which is made of cells in time laps movies, as well as a semi-synthetic data set and a toy synthetic data set.

Our main conclusion is that the VQ-based method drastically outperforms the baseline [6]. The forest also provides good recognition performance, but the evaluation of the parameters is not so easy such that we had bad results on the real data set. An other argument that goes in favor of the VQ-based model, is that this model is very straightforward and simple to understand for non-computer scientists such as biologists. The forest approach may still exhibit interesting properties in different contexts, for instance if the feature space lacks a metric structure.

Acknowledgments This work was supported by the Swiss National Science Foundation under Sinergia grant 127456 "Understanding Brain morphogenesis", and from a Human Frontier Science Program grant.

References

1. Arthur D, Vassilvitskii S (2007) k-means++: the advantages of careful seeding. In: Proceedings of the ACM-SIAM symposium on discrete algorithms, p 1027–1035
2. Bakal C (2007) Quantitative morphological signatures define local signaling networks regulating cell morphology. Science 316:1753–1756
3. Bishop CM, Ulusoy I (2005) Generative versus discriminative methods for object recognition. Conf Comput Vis Pattern Recogn 2:258–265
4. Breiman L (2001) Random forest. Mach Learn 45:5–32
5. Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees. Wadsworth & Brooks/Cole Advanced Books & Software, Monterey
6. Collinet C et al (2010) Systems survey of endocytosis by multiparametric image analysis. Nature 464:243–249
7. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the em algorithm. J R Stat Soc Ser B Methodol 39(1):1–38
8. Echeverri CJ et al (2006) Minimizing the risk of reporting false positives in large-scale RNAi screens. Nat Methods 3(10):777–779
9. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139
10. Hartigan JA (1975) Clustering algorithms. Wiley, New York
11. Held M et al (2010) CellCognition: time-resolved phenotype annotation in high-throughput live cell imaging. Nat Methods 7:747–754
12. Jackson AL, Linsley PS (2010) Recognizing and avoiding siRNA off-target effects for target identification and therapeutic application. Nat Rev Drug Discov 9:57–67
13. Kullback S (1987) Letter to the editor: the Kullback–Leibler distance. Am Stat 41(4):340–341
14. Laptev I, Marszalek M, Schmid C, Rozenfeld B (2008) Learning realistic human actions from movies. In: International conference on computer vision and pattern recognition
15. Lefort R, Fablet R, Boucher J-M (2010) Weakly supervised classification of objects in images using soft random forests. In: European conference on computer vision
16. Lefort R, Fablet R, Boucher JM (2011) Object recognition using proportion-based prior information: application to fisheries acoustics. Pattern Recogn Lett 32(2):153–158
17. Lefort R, Fleuret F (2013) treeKL: A distance between high dimension empirical distributions. Pattern Recogn Lett 34(2):140–145
18. Lowe D (1999) Object recognition with informative features and linear classification. In: International conference on computer vision and pattern recognition
19. Lughofer E (2008) Extensions of vector quantization for incremental clustering. Pattern Recogn 41(3):995–1011
20. Lughofer E (2013) eVQ-AM: an extended dynamic version of evolving vector quantization. In: IEEE conference on evolving and adaptive intelligent systems, p 40–47

21. McLachlan GJ, Krishnan T (2008) The EM algorithm and extensions, 2nd edn. Wiley, New York
22. Mahalanobis PC (1936) On the generalised distance in statistics. Proc Natl Inst Sci India 2(1):49–55
23. Moosman F, Nowak E, Jurie F (2008) Randomized clustering forests for image classification. IEEE Trans Pattern Anal Mach Intell 30(9):1632–1646
24. Neumann B et al (2010) Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. Nature 464:721–72
25. Orvedahl A et al (2011) Image-based genome-wide siRNA screen identifies selective autophagy factors. Nature 480:113–117
26. Parzen E (1962) On estimation of a probability density function and mode. Ann Math Stat 33:1065–1076
27. Pertz O et al (2008) Spatial mapping of the neurite and soma proteomes reveals a functional Cdc42/Rac regulatory network. Natl Acad Sci USA 105:1931–1936
28. Salma J et al (2012) Computational analysis and predictive modeling of small molecule modulators of microRNA. J Cheminform 4(1):16. doi:10.1186/1758-2946-4-16
29. Schölkopf B, Smola AJ (2002) Learning with kernels: support vector machines, regularization, optimization and beyond. MIT Press, Cambridge
30. Yan J et al (2013) Transcription factor binding in human cells occurs in dense clusters formed around cohesin anchor sites. Cell 154(4):801–813
31. Yin Z et al (2013) A screen for morphological complexity identifies regulators of switch-like transitions between discrete-cell shape. Nat Cell Biol 15(7):860–871
32. Yizong C (1995) Mean shift, mode seeking, and clustering. IEEE Trans Pattern Anal Mach Intell 17(8):790–799