Computer vision for computational phenotyping

Thomas Walter

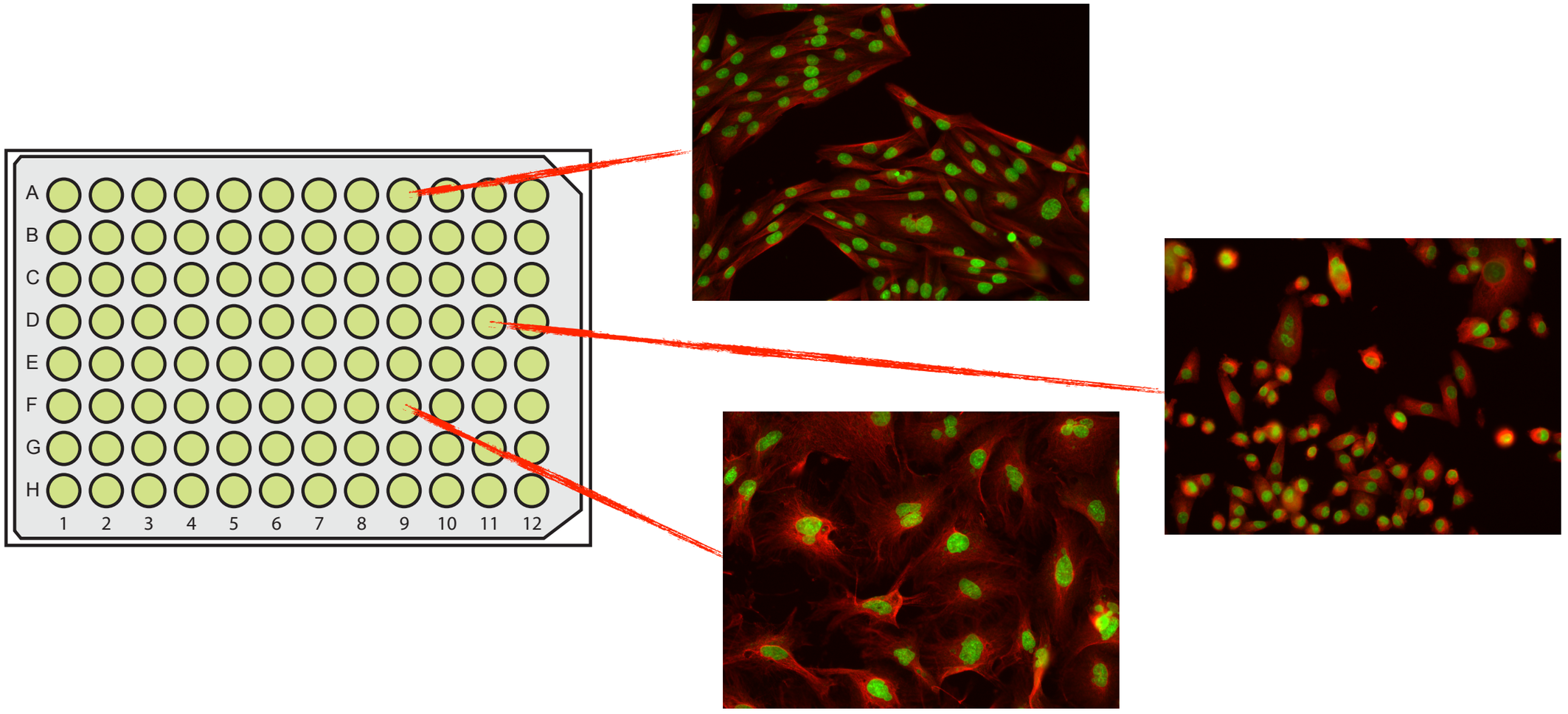
Overview

- Motivation: High Content Screening (HCS) and computational phenotyping

- The problem: classification of cells
  - Segmentation
  - Features for texture and shape description
  - Classification

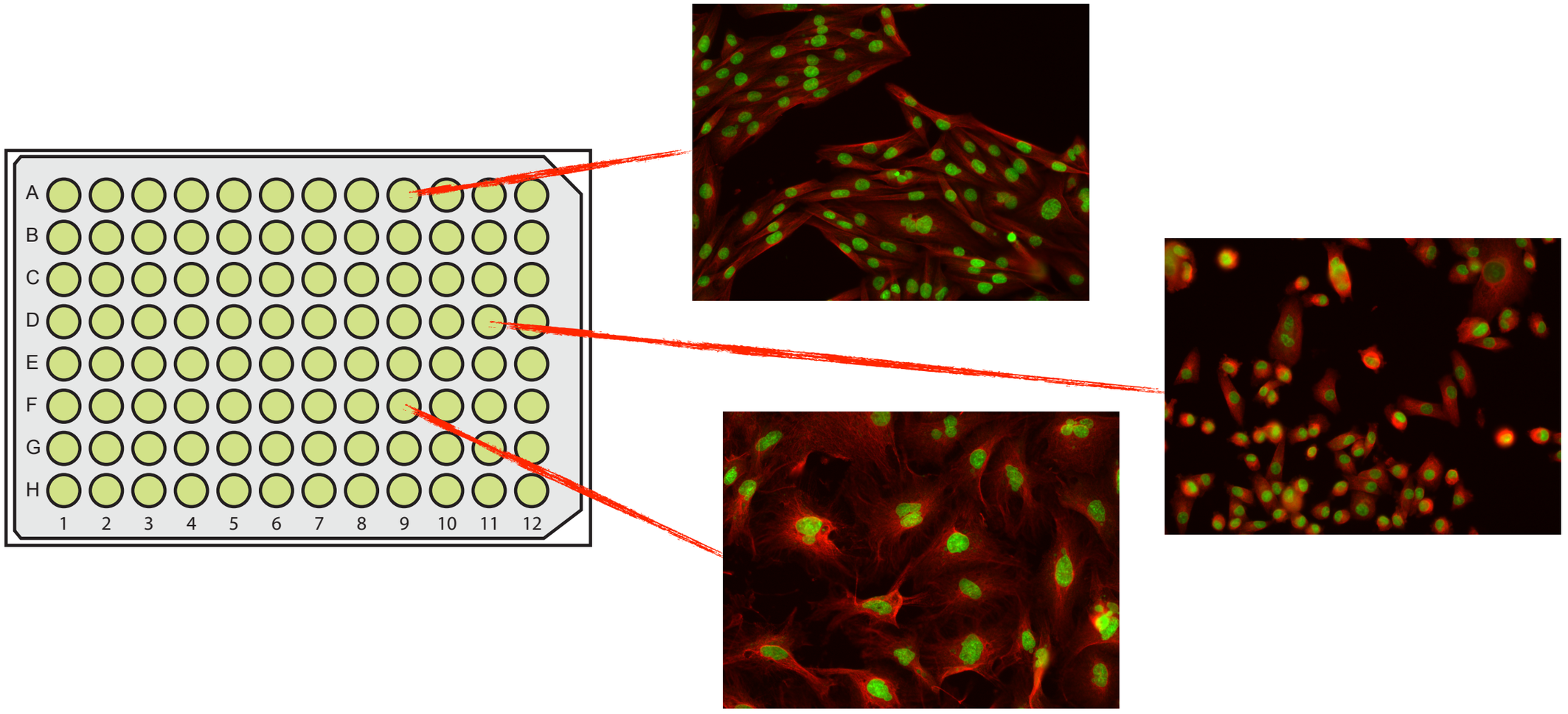
- Applications

Motivation: HCS and computational phenotyping



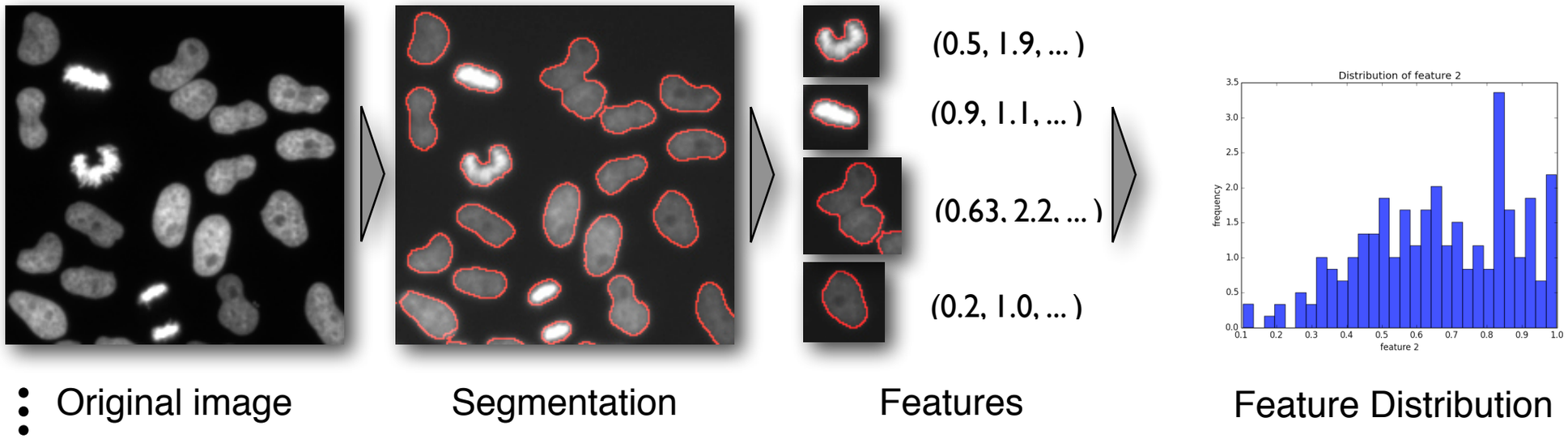
- High Throughput Screening (HTS): performing a large number of experiments
- High Content Screening (HCS): each experiment has a large information content (i.e. we can distinguish individual cells and make measurements on each of them)

Motivation: HCS and computational phenotyping



- High automation: liquid handler, spotting robots, automatic microscopes to perform a large number of experiments (> 10.000).
- Applications:
  - Drug screening: measure the effect of drugs on cells / organisms.
  - Genetic screening: infer the function of a protein from the measured effect of its absence.

Computational analysis of population phenotypes



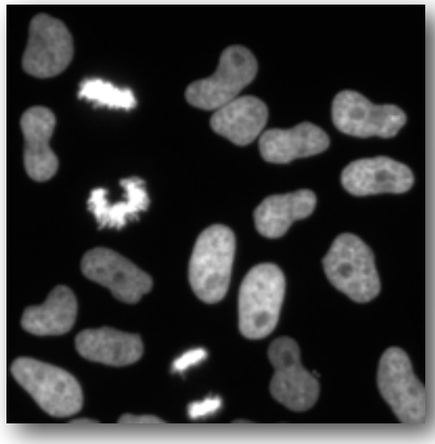
Original image | Segmentation | Features | Feature Distribution

Phenotype description by a single feature distribution

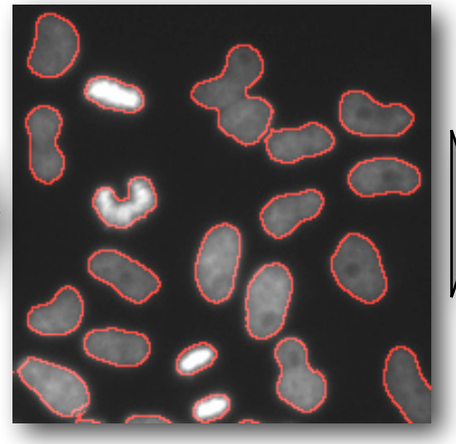
The ideal situation arises, if we can directly measure the relevant biological feature.

Examples: particle speed, elongation of cells, number of spots, …

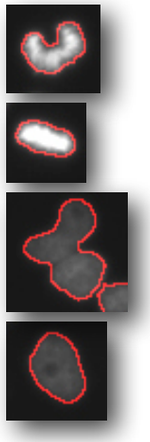
Computational analysis of population phenotypes



Original image



Segmentation



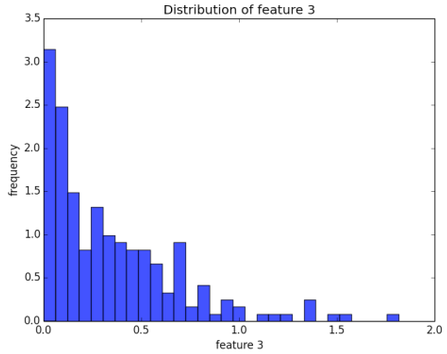
(0.5, 1.9, ...)

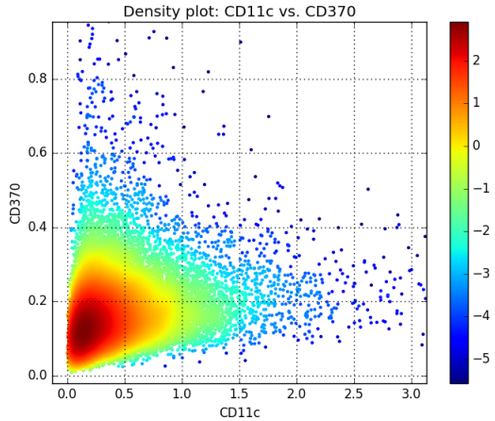
(0.9, 1.1, ...)

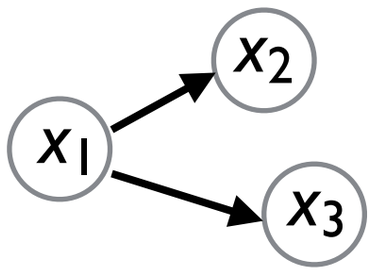
(0.63, 2.2, ...)

(0.2, 1.0, ...)

Features



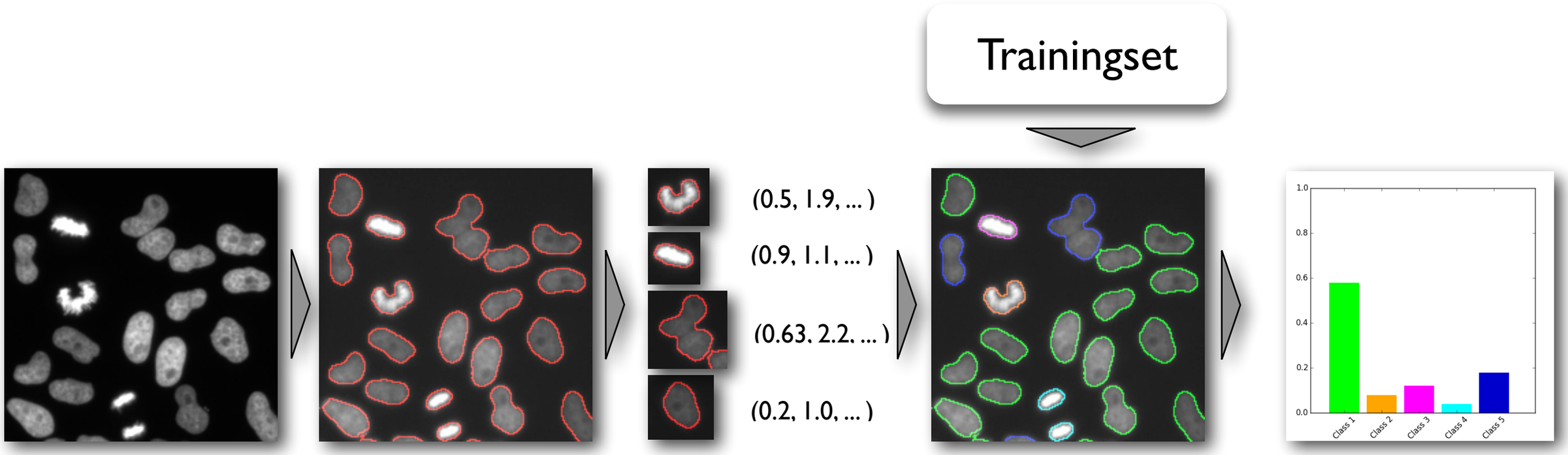




Phenotype description by feature distributions

Sometimes, we are interested in more than one interpretable feature. The challenge is then to analyze the distributions of these features in a univariate or multivariate setting.

Computational analysis of population phenotypes

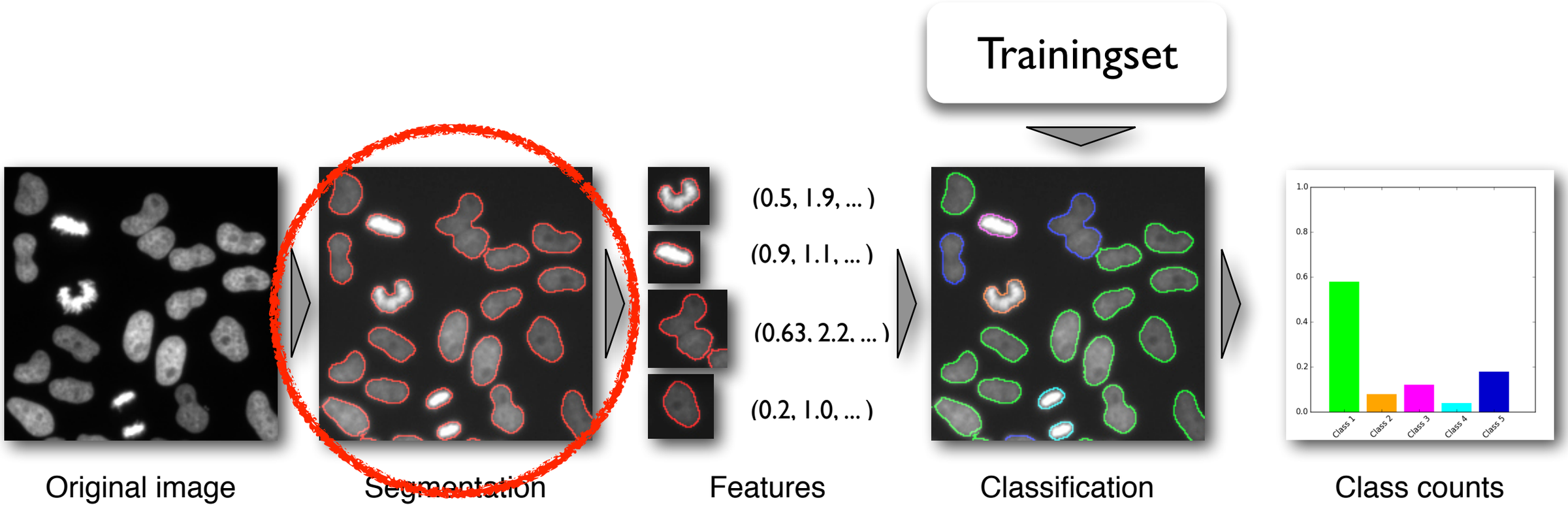


Original image | Segmentation | Features | Classification | Class counts

Phenotype description by single cell classification

Sometimes, the biological information cannot be easily represented by a single or multiple interpretable features. In this case we can use many uninterpretable features to derive biologically meaningful classes.

Computational analysis of population phenotypes



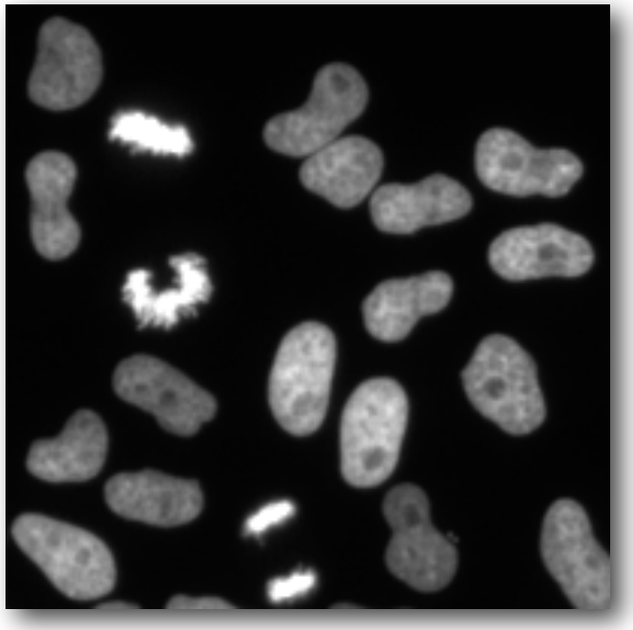
Original image | Segmentation | Features | Classification | Class counts

Phenotype description by single cell classification

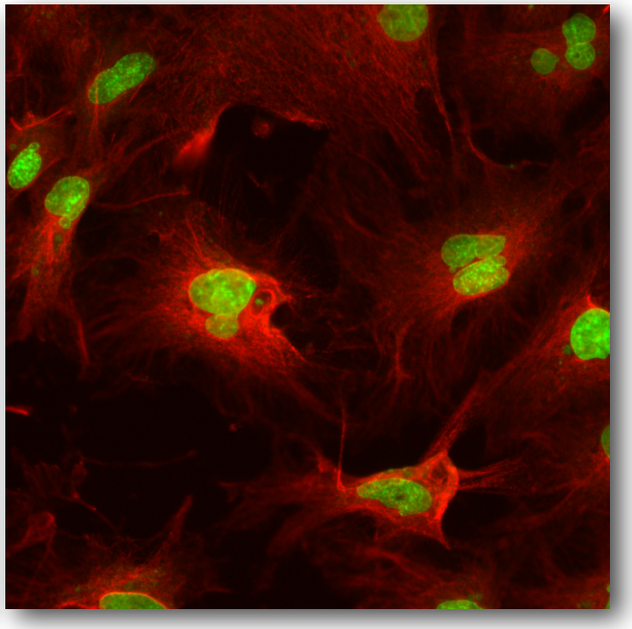
Sometimes, the biological information cannot be easily represented by a single or multiple interpretable features. In this case we can use many uninterpretable features to derive biologically meaningful classes.

Segmentation: nuclei

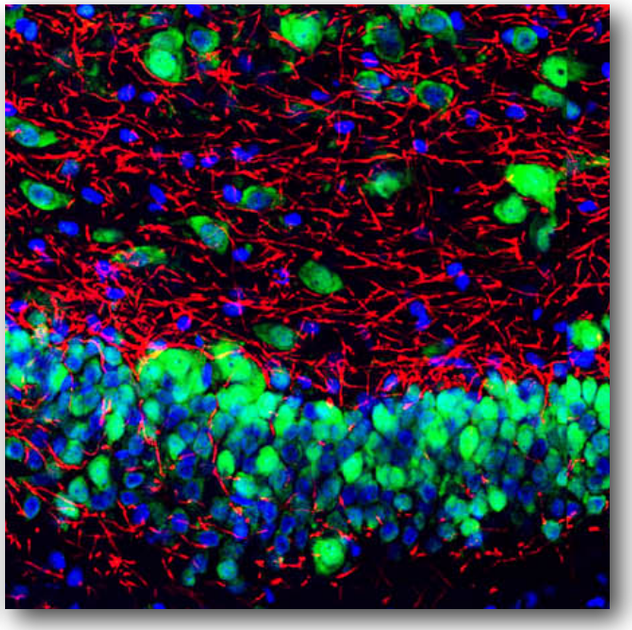
- The most important (most frequent) segmentation task in bioimaging is the segmentation of nuclei:
  - Nuclei are the place where DNA is stored.
  - The nuclear shape is indicative for many cell types and cellular states.
  - Nuclei can be stained efficiently and have normally a simple shape. They are thus easier to delineate than most other cellular organelles.
  - Nuclei are the most popular reference point used in cellular imaging.



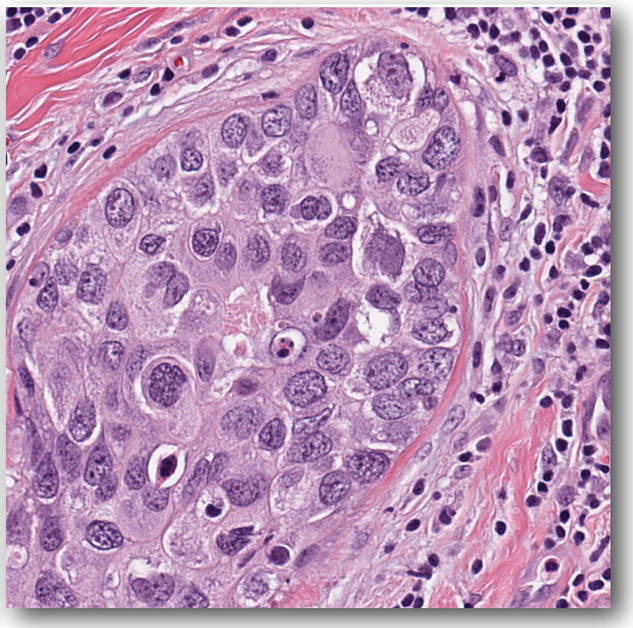
Cultured cells
10x wide field
fluorescence microscopy



Cultured cells
Green: nuclei,
Red: microtubules



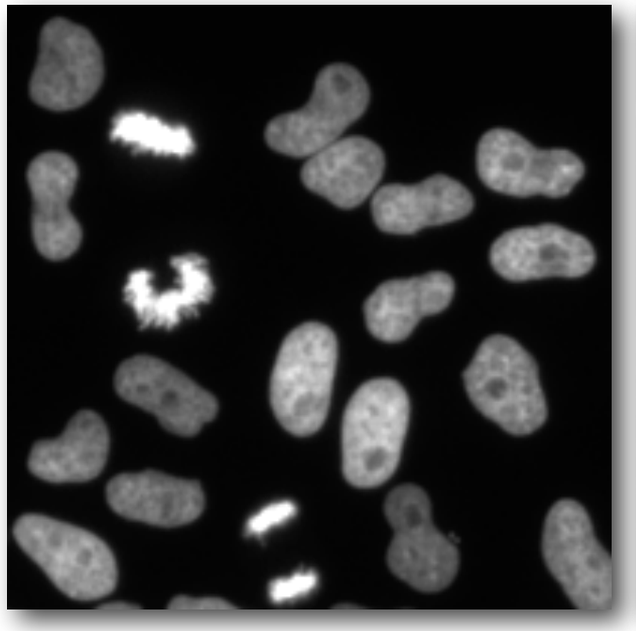
Nuclei in the brain
(DAPI staining)



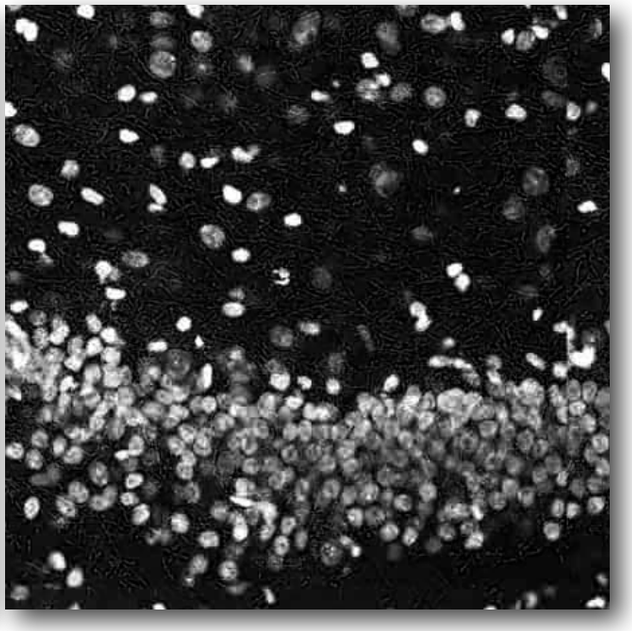
Nuclei in the breast with non-fluorescent staining.

Segmentation: nuclei

- The most important (most frequent) segmentation task in bioimaging is the segmentation of nuclei
- In fluorescence microscopy, pixel segmentation can be achieved "easily" by traditional techniques:
  - Prefiltering including noise suppression and background correction
  - Thresholding or other classical techniques
  - Splitting of "touching objects"
- Nuclei segmentation for non-fluorescent imaging can be very challenging.
- Splitting approaches remain important, in particular for dense clusters.

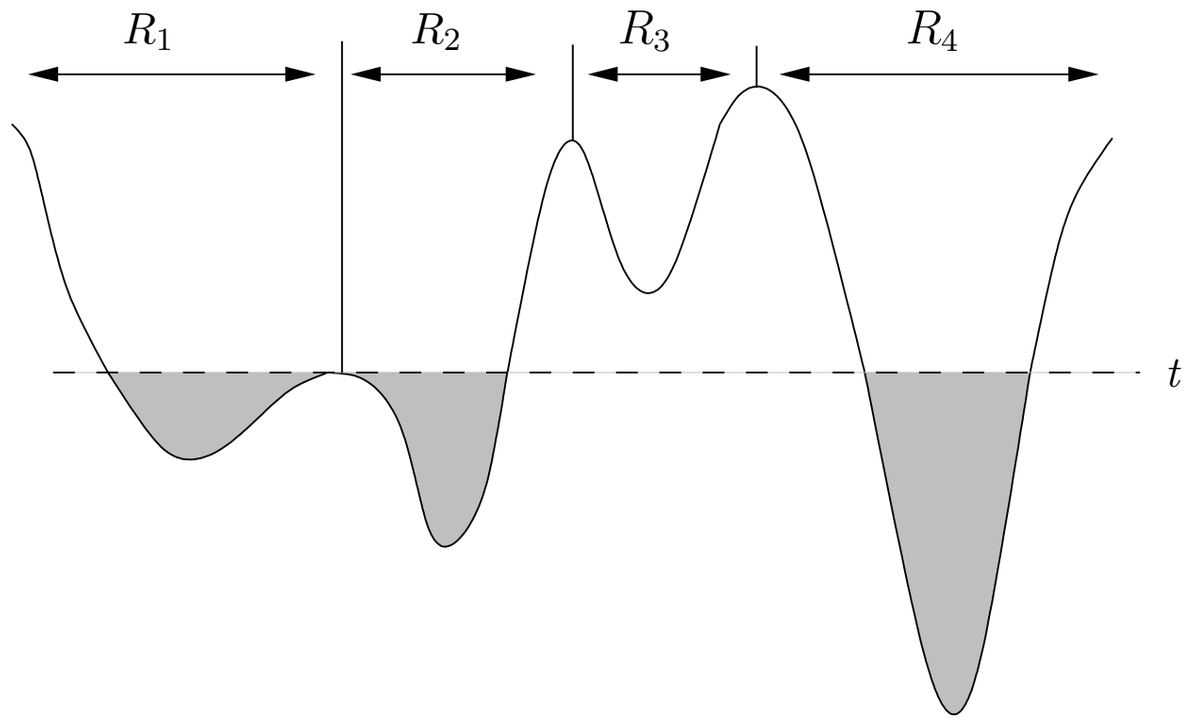


Cultured cells, 10x wide field fluorescence microscopy



Crowded Nuclei in the brain (DAPI staining)

Splitting: the Watershed transform (informal)

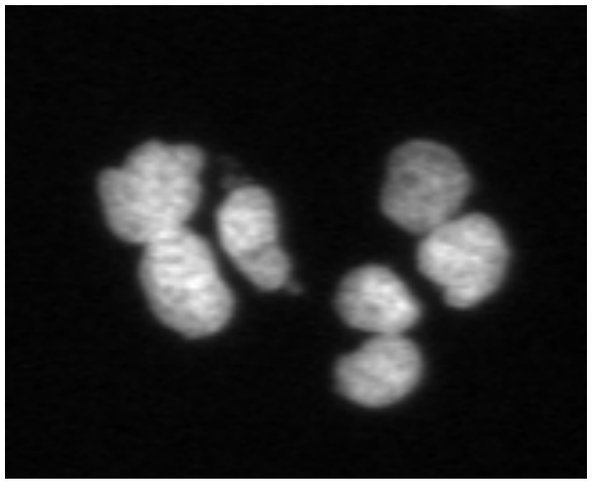


The image is interpreted as a topographic surface, which is "flooded":

- The flooding starts at the minimum value f_{min} by building the first catchment basins R_i as connected components of all pixels with $f(x) = f_{min}$.
- To build the catchment basins at level $t > f_{min}$ from the catchment basins at level $t - 1$, all local minima with level t are added as new catchment basins and all pixels x with $f(x) = t$ not belonging to local minima are either added to the only adjacent catchment basin, or - in case there are several adjacent catchment basins - labeled as part of the watershed line.

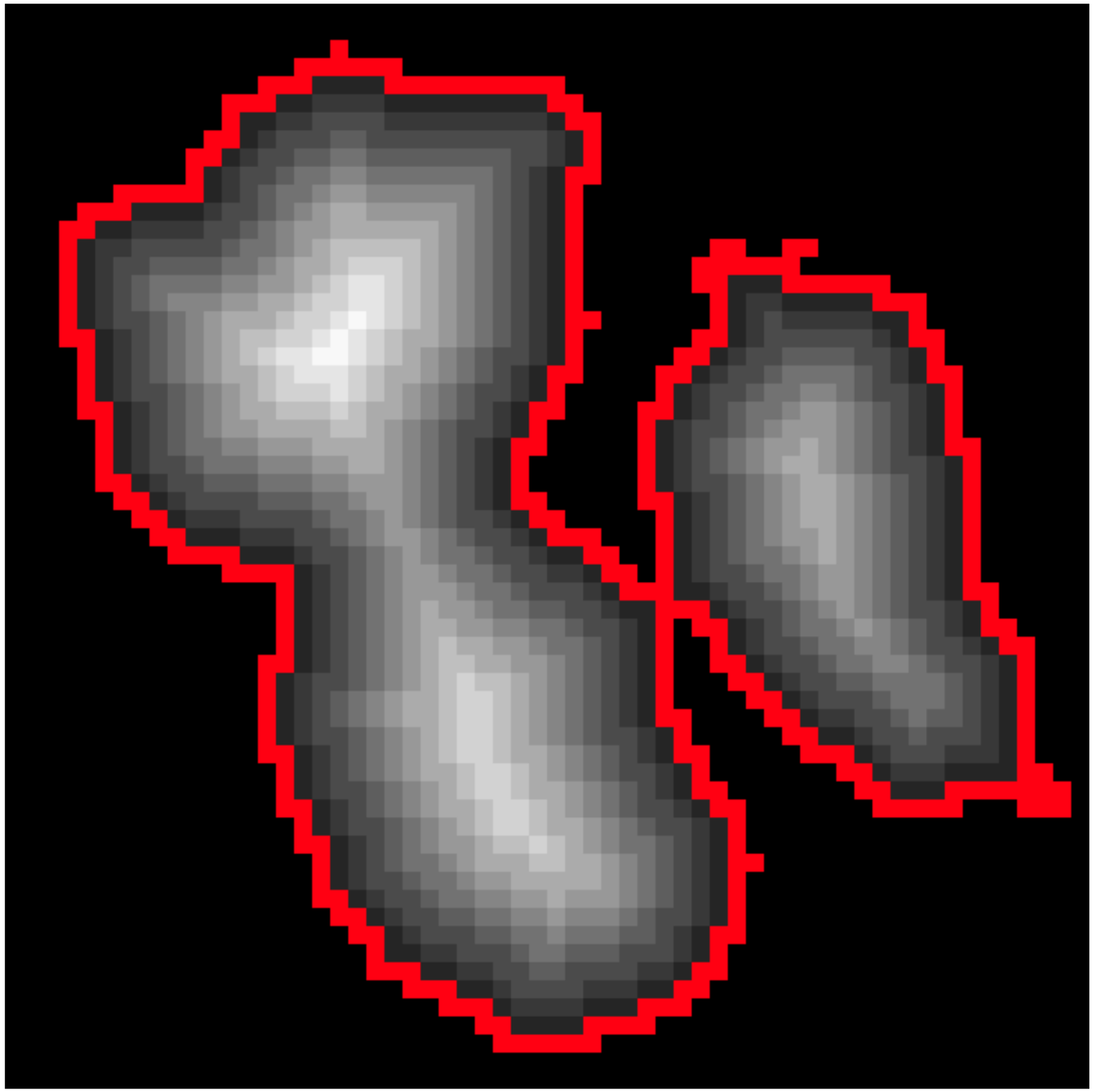
Splitting: distance map and watershed

Original



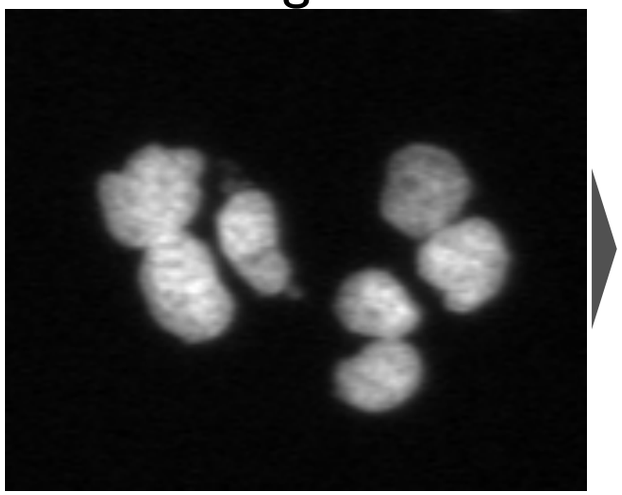
Segmentation

Distance map



Splitting: distance map and watershed

Original



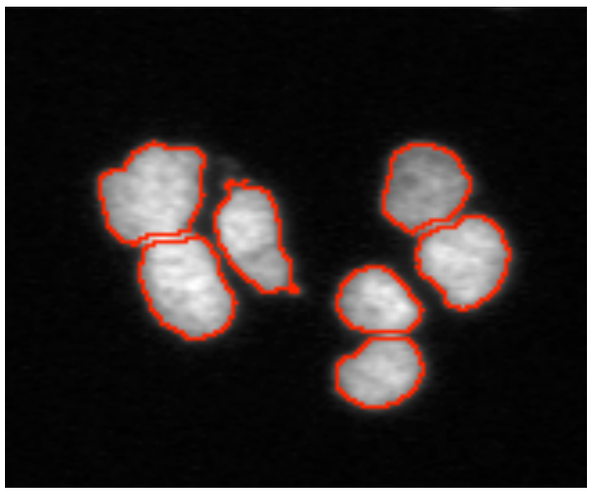
Segmentation

Distance map (inv)

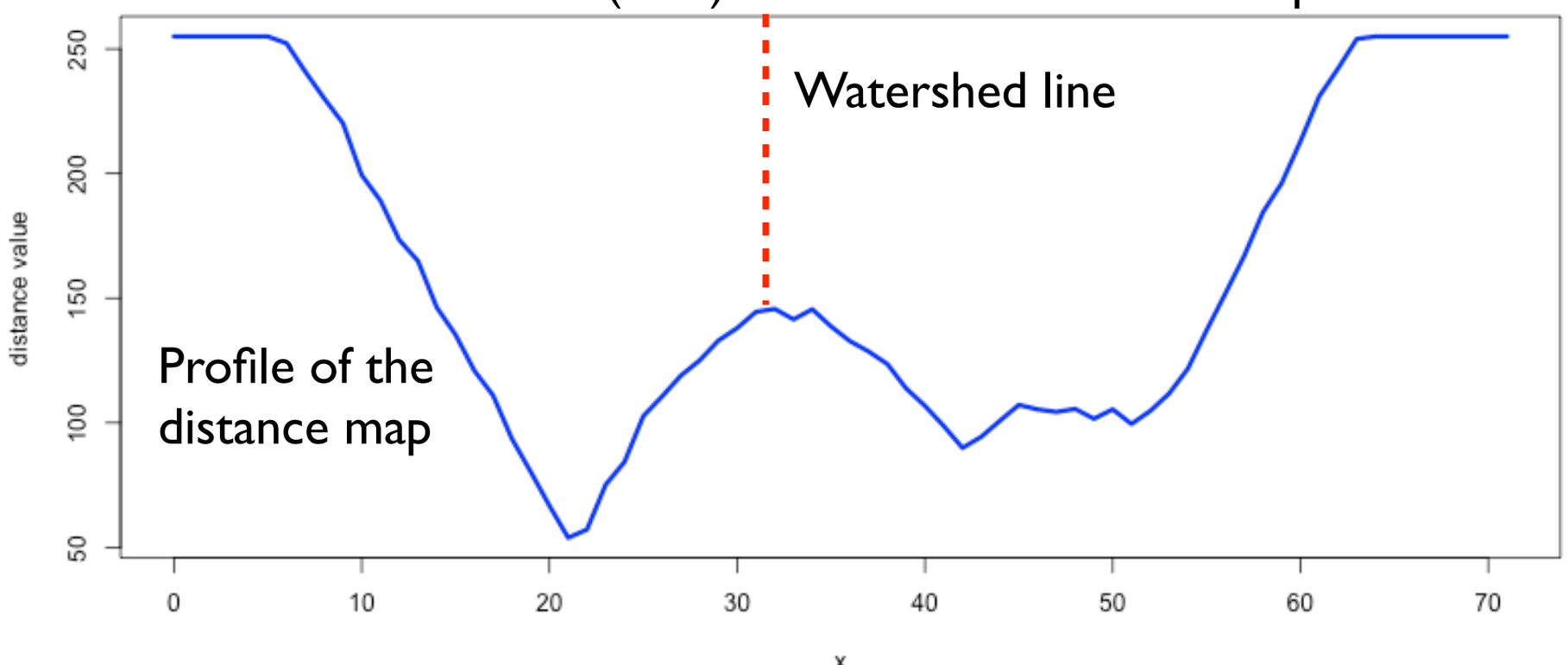
Split objects (WS)

Plot along the profile

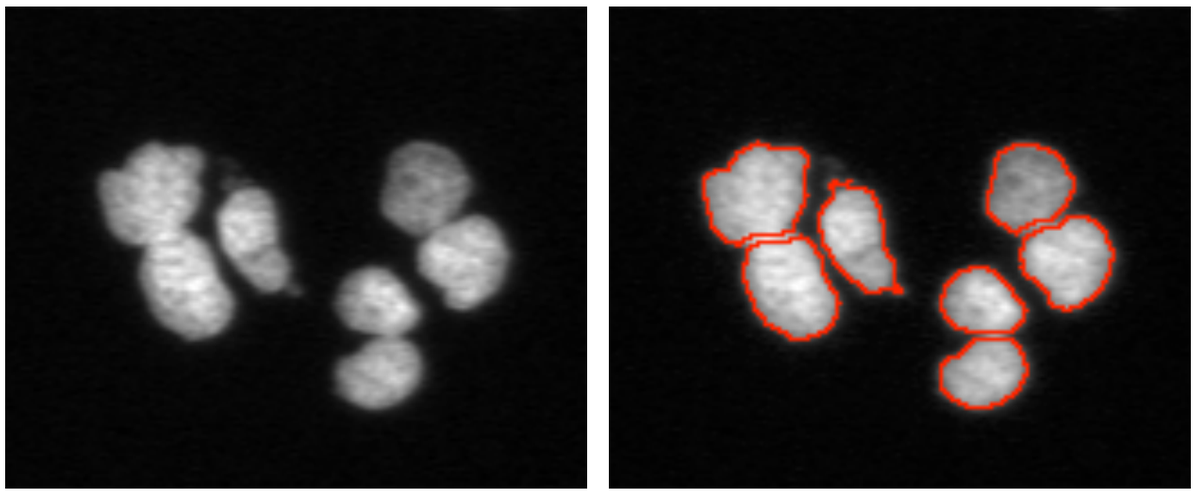
Result



Profile (blue) of the inverse distance map



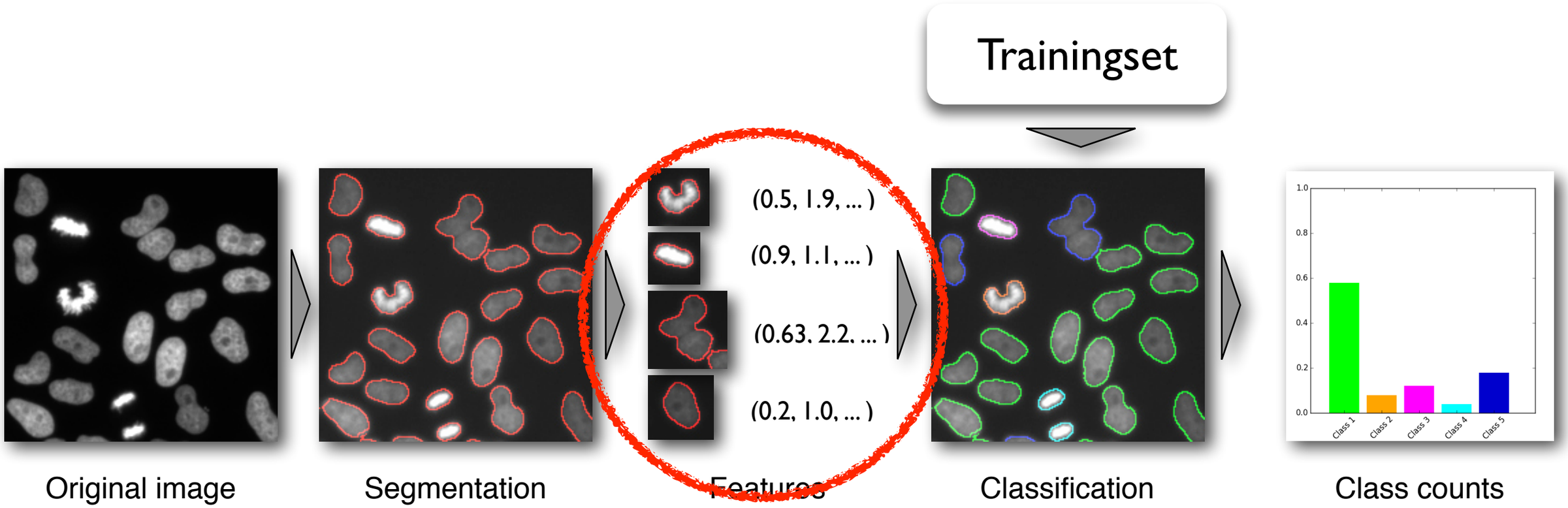
Splitting: distance map and watershed



Original *Result after split*

- Nearly all nuclei segmentation algorithms (including those based on deep learning) have some splitting algorithm associated to it, often based on the watershed transform.
- The watershed strategy makes an assumption on the convexity of the shape: there are as many splits as maxima in the distance map.
- Nuclei segmentation remains a hot topic in bioimage analysis: https://www.kaggle.com/c/data-science-bowl-2018

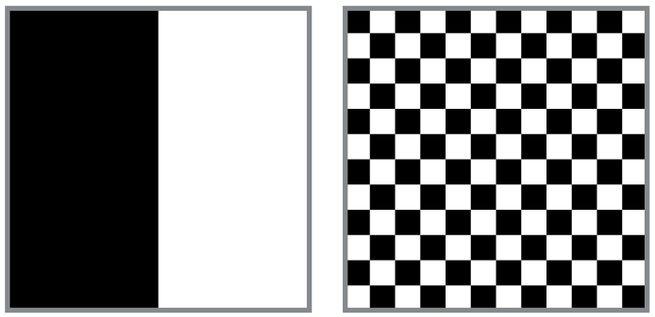
Computational analysis of population phenotypes



Original image | Segmentation | Features | Classification | Class counts

Features for computational phenotyping

- Basic shape features: size, perimeter, circularity, moment invariants, …
- Basic intensity features: average grey level, standard deviation, …
- Second order statistics: features calculated on the co-occurrence matrix



These two images have exactly the same mean, standard deviation and higher order moments

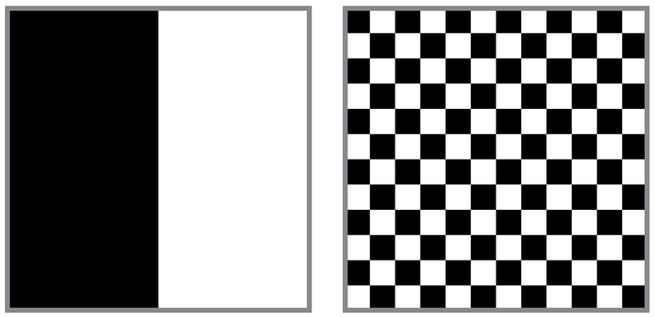
$$f = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 2 & 2 & 2 & 1 & 0 \\ 2 & 2 & 1 & 1 & 0 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \end{bmatrix}$$

1. Define an orientation Δ. ↔
2. Count the cooccurrences of values *i* and *j*.

$$C(\Delta) = \begin{bmatrix} \cdots & 7 & \cdots \\ 7 & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}$$

Features for computational phenotyping

- Basic shape features: size, perimeter, circularity, moment invariants, …
- Basic intensity features: average grey level, standard deviation, …
- Second order statistics: features calculated on the co-occurrence matrix



These two images have exactly the same mean, standard deviation and higher order moments

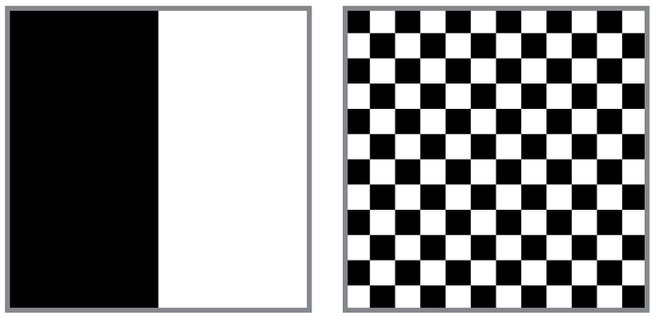
$$f = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 2 & 2 & 2 & 1 & 0 \\ 2 & 2 & 1 & 1 & 0 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \end{bmatrix}$$

1. Define an orientation Δ. ↔
2. Count the cooccurrences of values *i* and *j*.
3. Do this for all value pairs (i, j).

$$C(\Delta) = \begin{bmatrix} 6 & 7 & 2 \\ 7 & 10 & 3 \\ 2 & 3 & 10 \end{bmatrix}$$

Features for computational phenotyping

- Basic shape features: size, perimeter, circularity, moment invariants, …
- Basic intensity features: average grey level, standard deviation, …
- Second order statistics: features calculated on the co-occurrence matrix



These two images have exactly the same mean, standard deviation and higher order moments

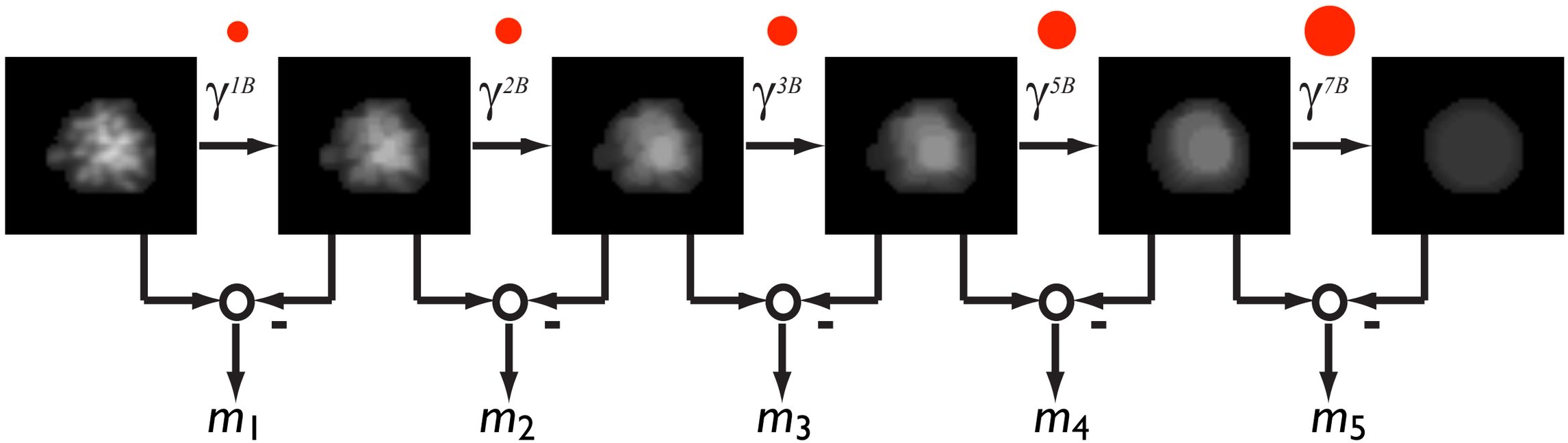
$$f = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 2 & 2 & 2 & 1 & 0 \\ 2 & 2 & 1 & 1 & 0 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \end{bmatrix}$$

$$C(\Delta) = \begin{bmatrix} 6 & 7 & 2 \\ 7 & 10 & 3 \\ 2 & 3 & 10 \end{bmatrix}$$

- From the co-occurrence matrix, we can now calculate descriptors (Haralick features).
- These features are typically averaged over several rotation angles.

Features for computational phenotyping

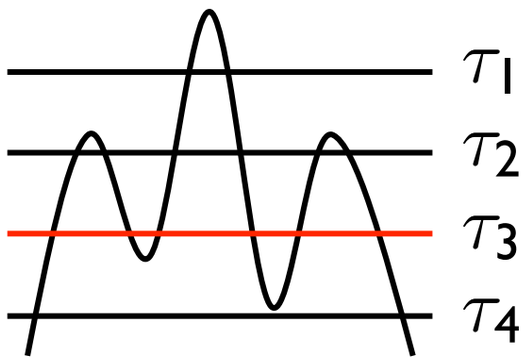
- Basic shape features: size, perimeter, circularity, moment invariants, …
- Basic intensity features: average grey level, standard deviation, …
- Second order statistics: features calculated on the co-occurrence matrix
- Morphological granulometries:

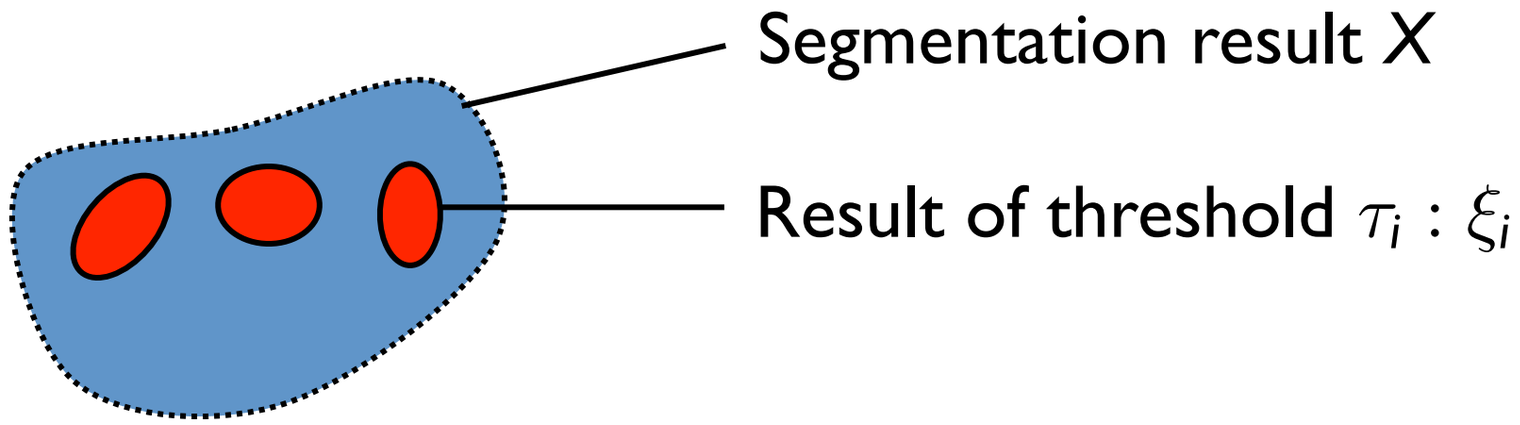


The image is successively simplified (here: morphological openings) and we measure how much of the signal has been removed.

Features for computational phenotyping

- Basic shape features: size, perimeter, circularity, moment invariants, …
- Basic intensity features: average grey level, standard deviation, …
- Second order statistics: features calculated on the co-occurrence matrix
- Morphological granulometries.
- Statistical geometric features:





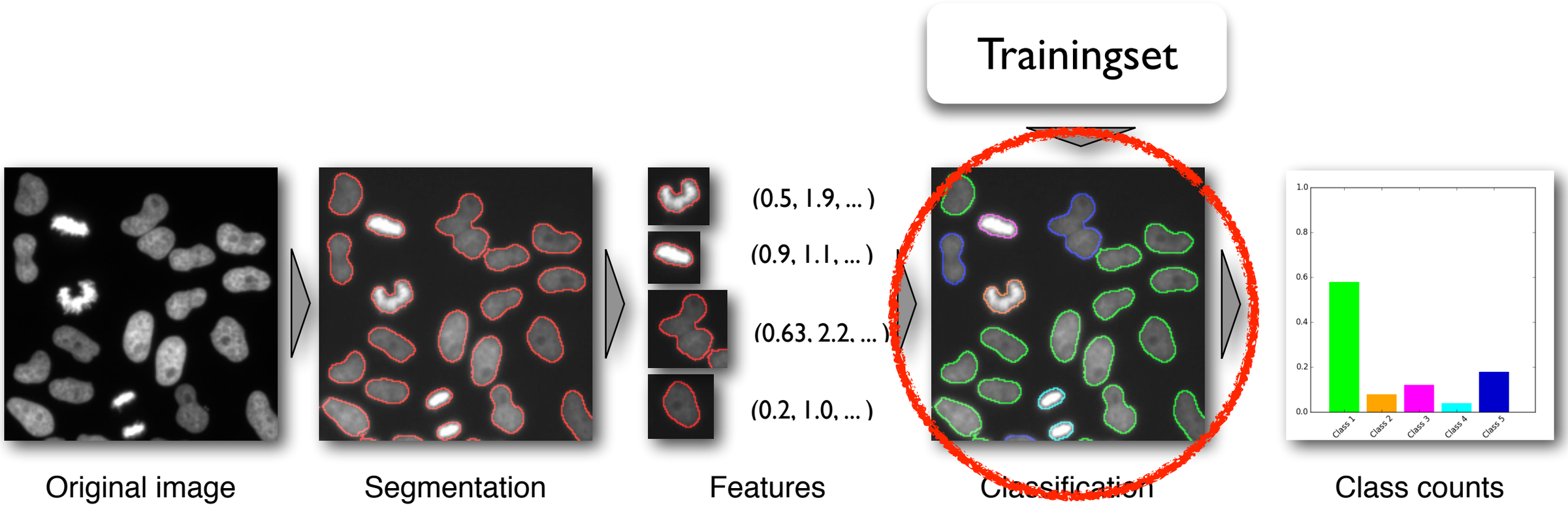
For each threshold, shape descriptors of the binary image are calculated (such as the number of connected components, the average distance to the center, …)

Features for computational phenotyping

- Basic shape features: size, perimeter, circularity, moment invariants, …
- Basic intensity features: average grey level, standard deviation, …
- Second order statistics: features calculated on the co-occurrence matrix
- Morphological granulometries.
- Statistical geometric features.
- Zernike moments
- Wavelet features
- Convex hull features
- …

With this, we can map each cell to a P-dimensional feature vector $x \in \mathbb{R}^P$

Computational analysis of population phenotypes



Original image | Segmentation | Features | Classification | Class counts

Supervised Learning

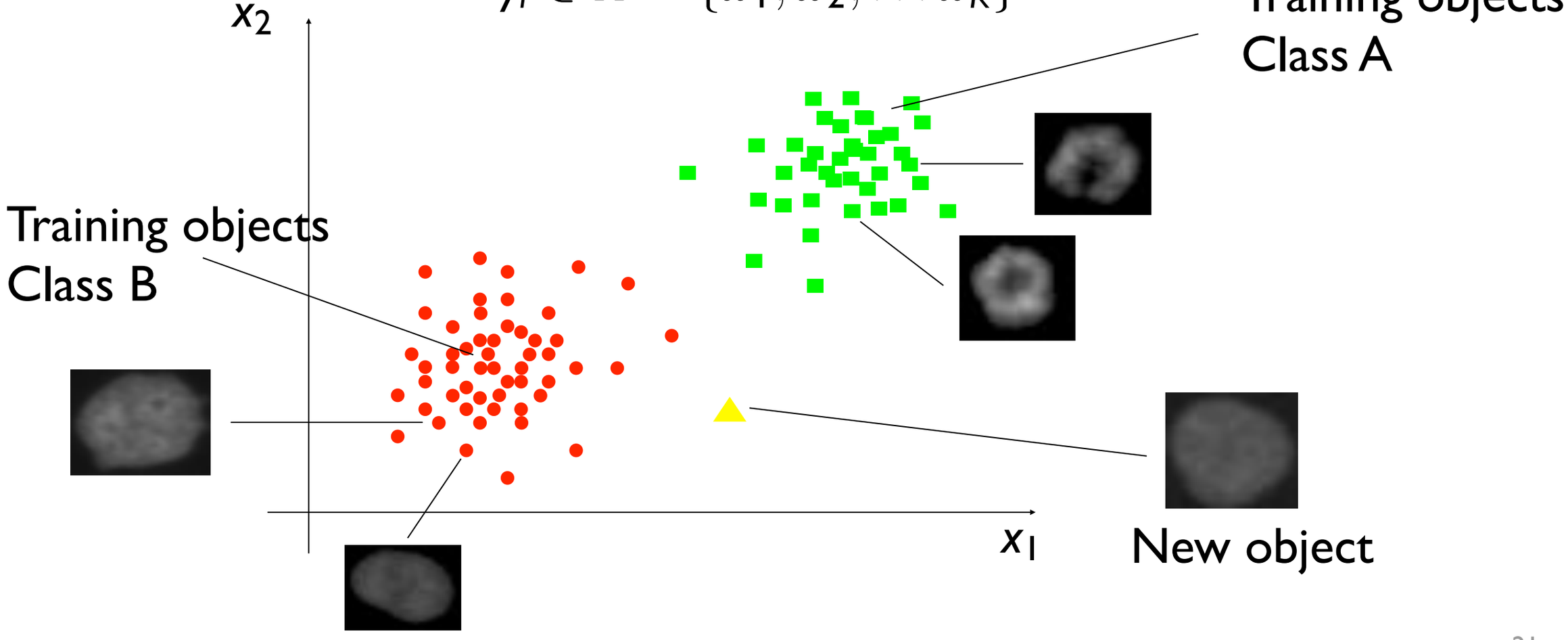
Classification: Learn a rule from the training set capable of assigning to each newly presented object one of the defined class labels.

Training set

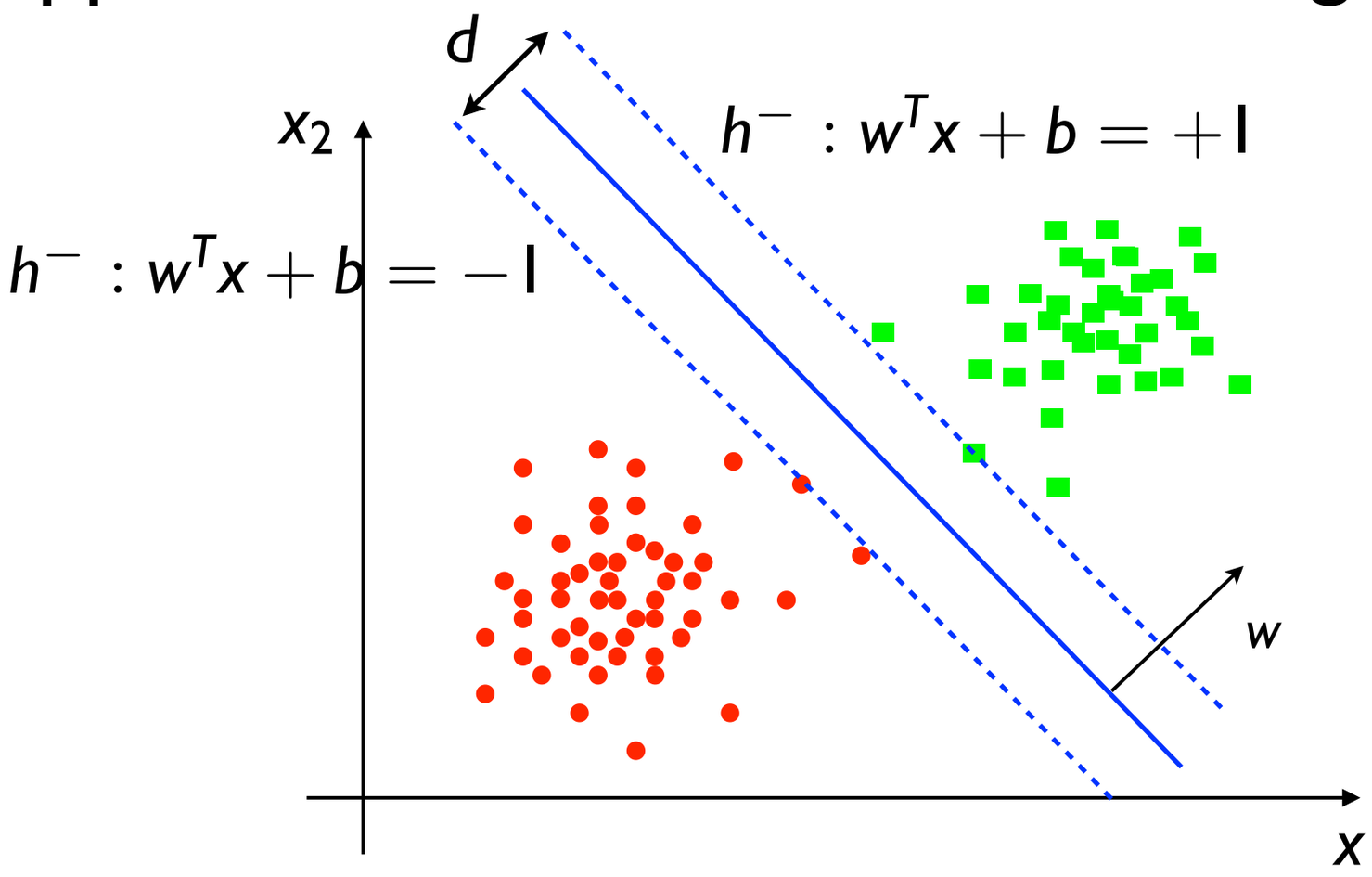
$$T = \{(x_i, y_i) \mid i = 1 \ldots N\}$$

$$x_i \in \mathbb{R}^P,\ i = 1 \ldots N$$

$$y_i \in \Omega = \{\omega_1, \omega_2, \ldots \omega_K\}$$



Support vector machines: maximizing the margin



w: normal vector

d: margin

$x_i \in \mathbb{R}^P$: feature vector

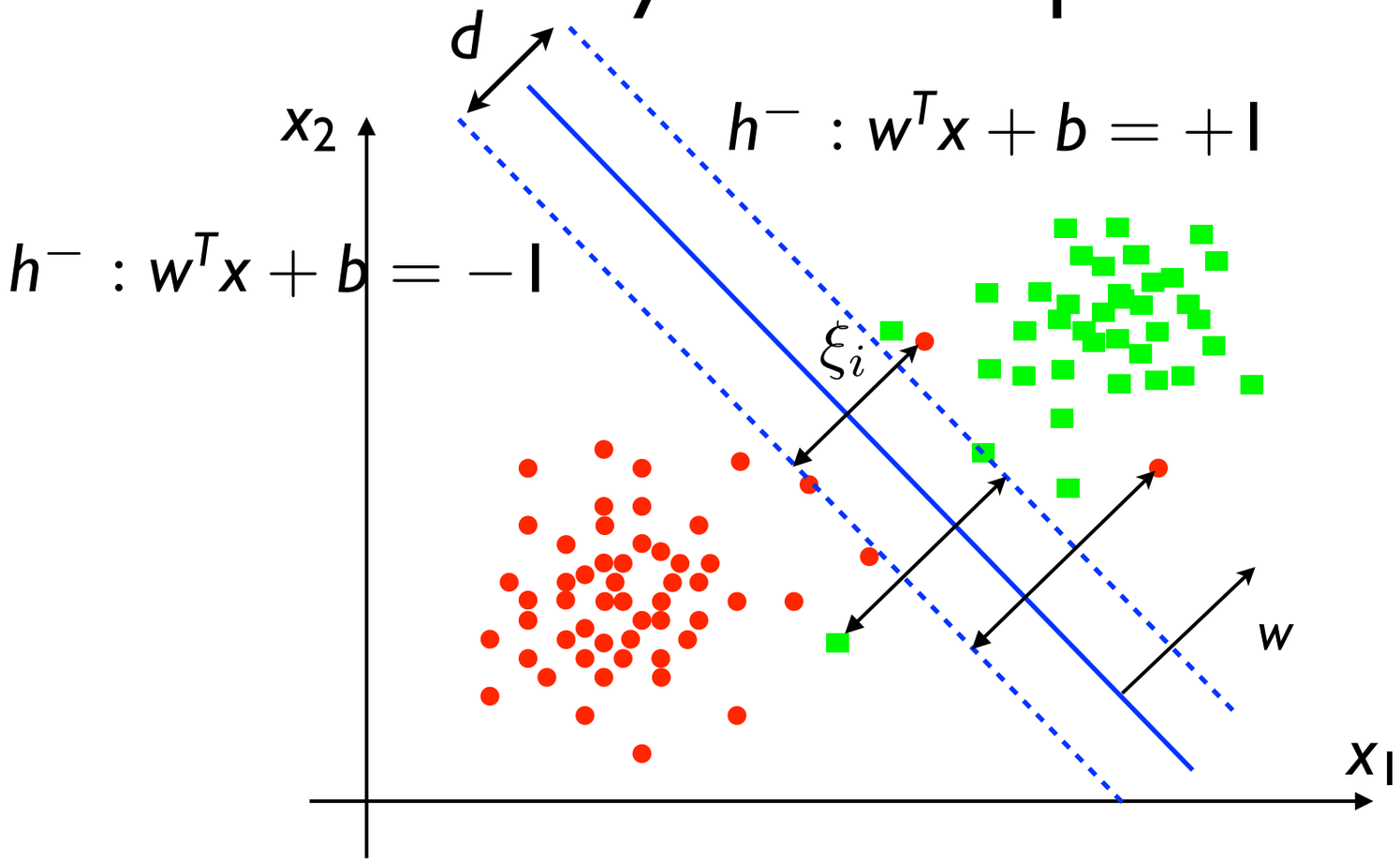
$y_i\{-1, +1\}$: output label

Maximizing the margin d is equivalent to minimizing $\|w\|^2$. Therefore, we can write the problem as a constraint optimization problem:

$$\begin{aligned} \text{minimize} \quad & \|w\|^2 \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1 \quad i = 1, \ldots, N \end{aligned}$$

which is a convex optimization problem with N linear inequality constraints and which therefore can be solved with the Langrangian formalism.

Classification by convex optimization: SVM



In the more realistic case where classes are not linearly separable, Support Vector Machines solve the following constrained optimization problem:

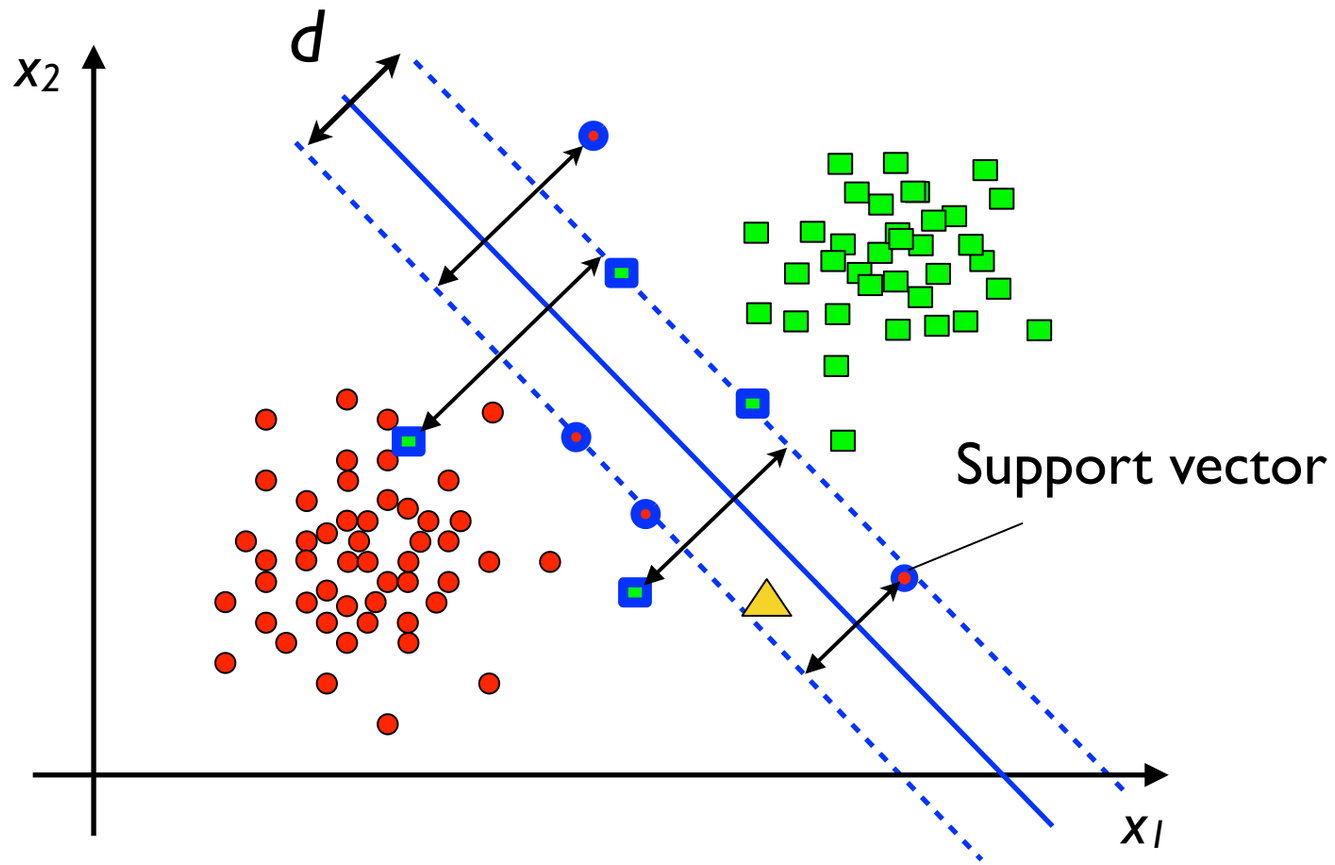
$$\begin{aligned} \min_{w,\xi} \quad & \|w\|^2 + C\sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i \quad i = 1, \ldots, N \\ & \xi_i \geq 0 \quad i = 1, \ldots, N \end{aligned}$$

We note that we now minimize a term of the form $\mathcal{R}(w) + C \cdot Err$, i.e. we minimize a weighted sum of a data term (error) and a regularization term.

SVM: only scalar products

- Only a few data points matter for the decision boundary: the **support vectors**.
- For prediction, it is sufficient to compare the point to these support vectors. Indeed, it is possible to show that f can be written as:

$$f(x) = \sum_{i \in SV} \alpha_i x_i^T x + b$$



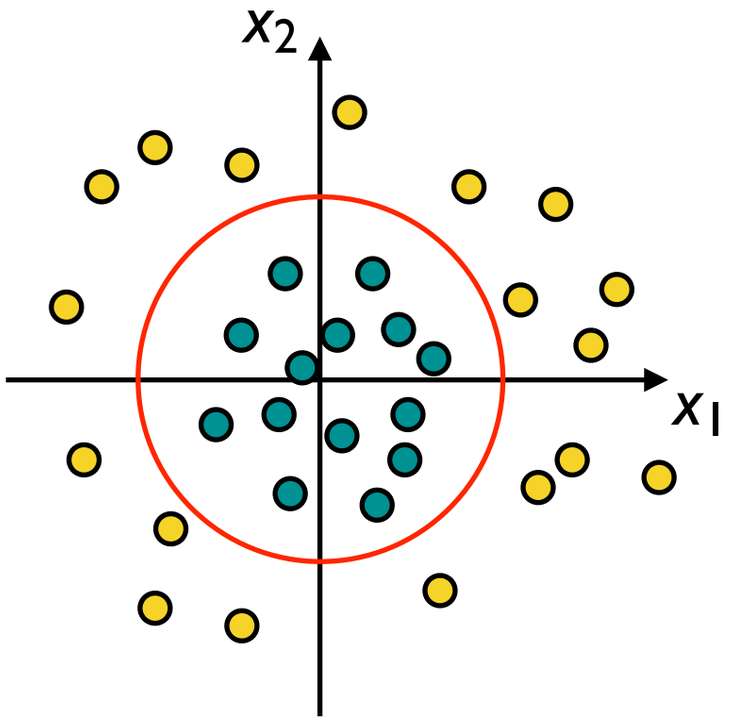
SVM: non-linear decision boundaries.

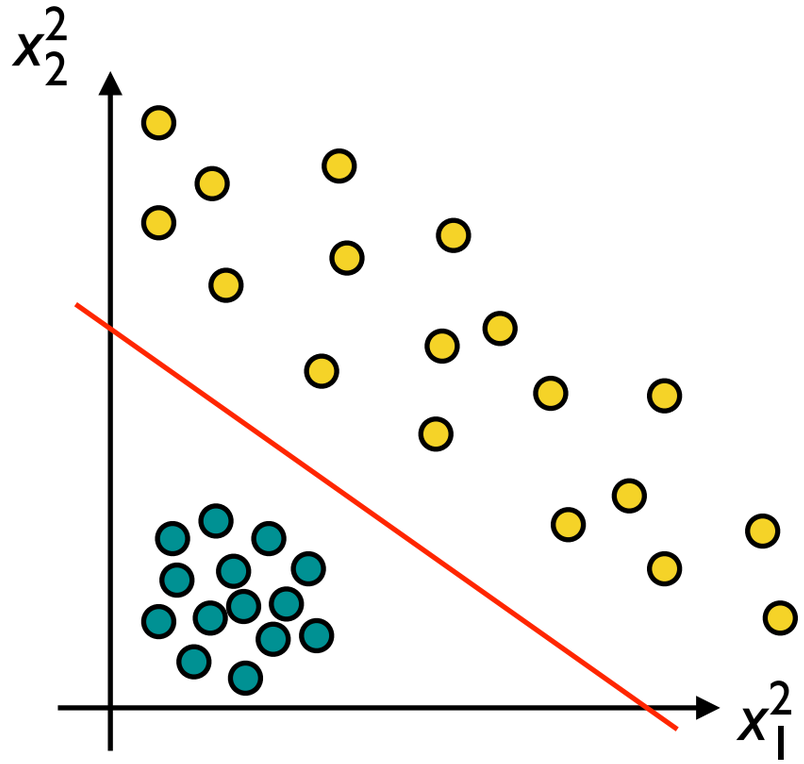
- Extension to non-linear decision boundaries by feature mapping:

$$f(x) = \sum_{i \in SV} \alpha_i \phi(x_i)^T \phi(x) + b = \sum_{i \in SV} \alpha_i K(x_i, x) + b$$

Kernel

- Example:

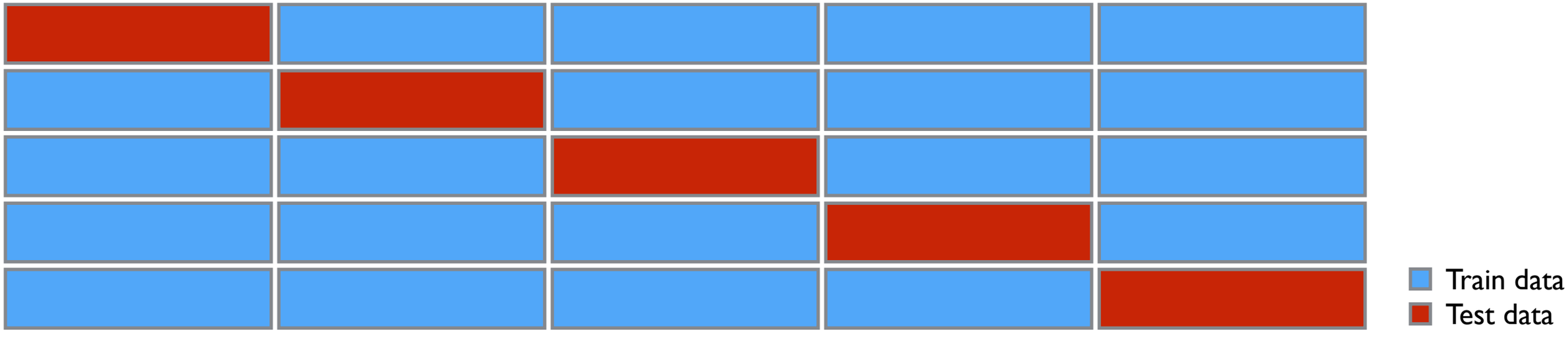




$$x = (x_1, x_2)^T \rightarrow \phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T$$

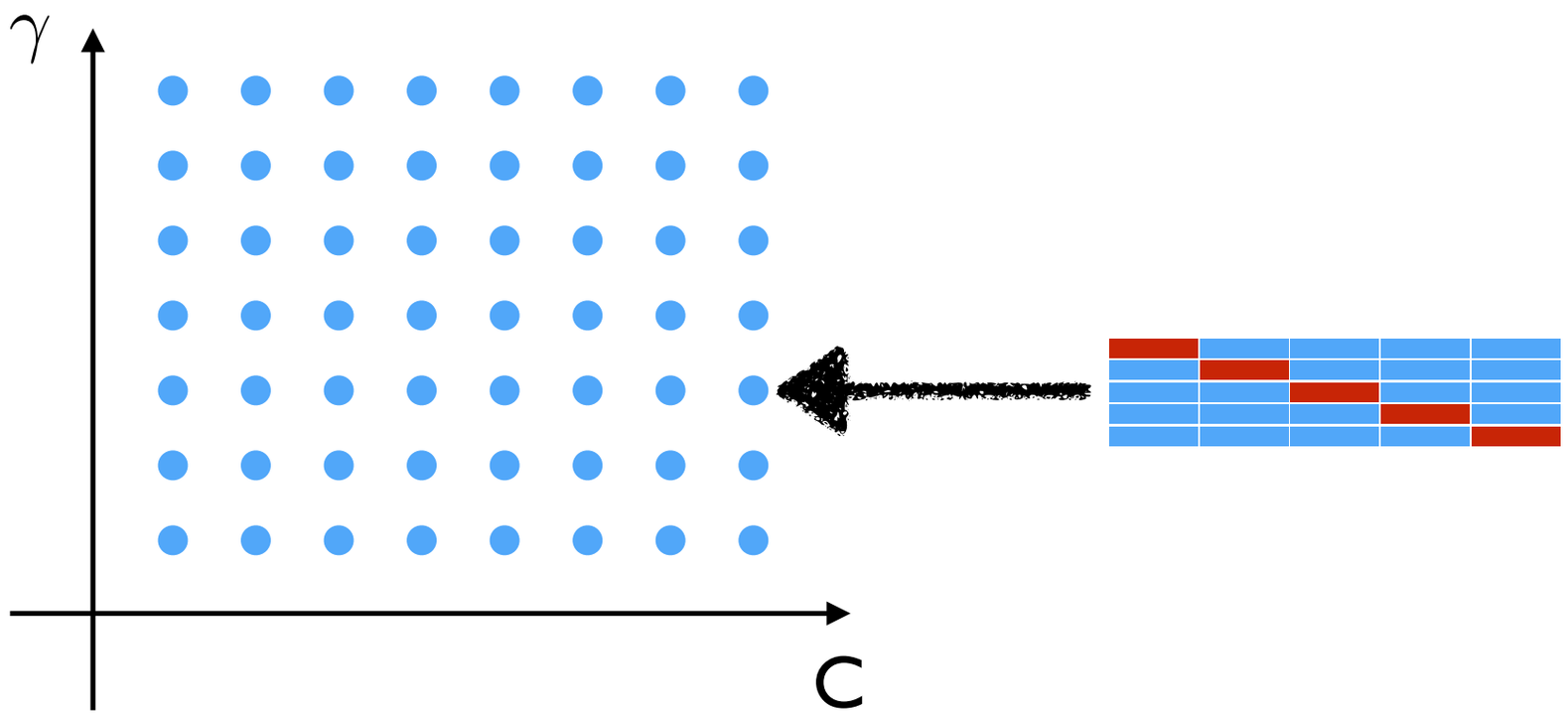
$$K(x, x') = (x^T x')^2$$

Support vector machines: parameter tuning



- The training set is divided into K equal parts (here K=5).
- For a given C the classifier is trained on the training data ((K-1)/K of the data points).
- The accuracy is determined on the test data (1/K of the data points).
- This is done on all folds in turn, and the average accuracy over all folds is assigned to C.
- The parameter C with the best performance is chosen.
- The final classifier is obtained by training on all the data with the parameter C.

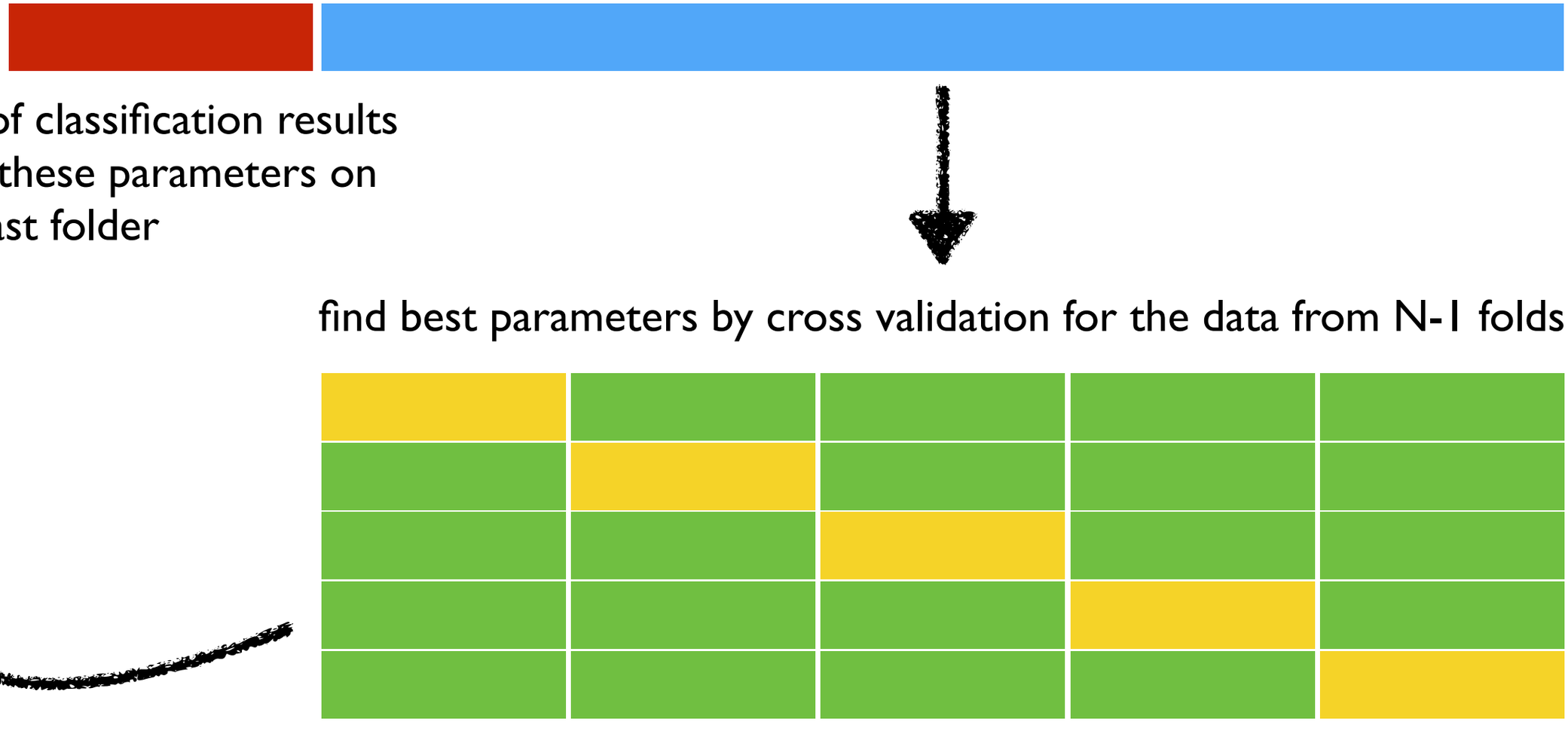
Support vector machines: grid search



- If there are more parameters, combinations of parameters are tested in the same way.
- The best combination of parameters is kept and the final classifier is trained on all data points.
- This is time consuming; we cannot afford to follow this procedure for many parameters.
- We often follow a refinement strategy.

Support vector machines: performance assessment

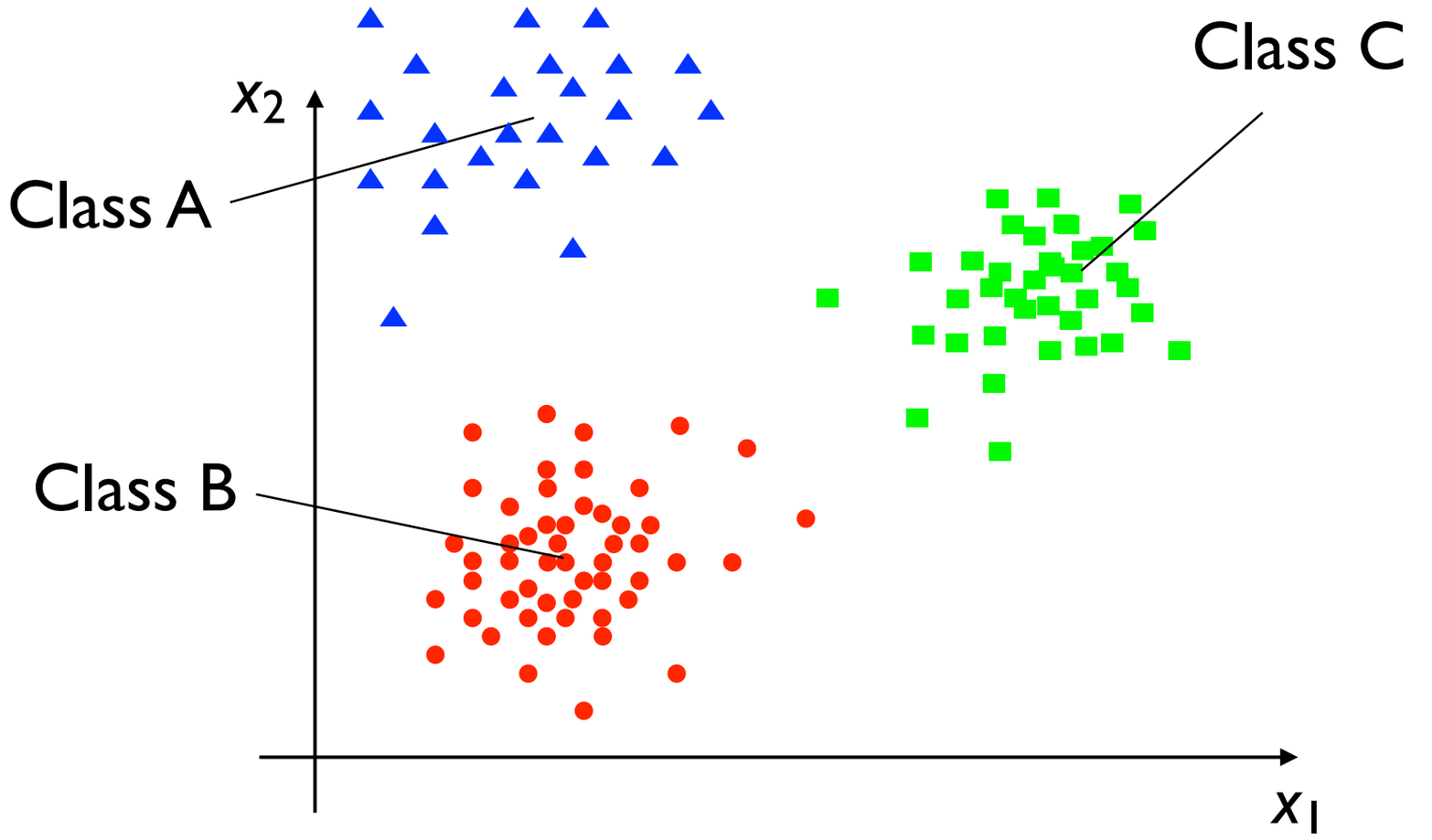
- However, in this way we cannot assess performance by another round of cross validation: all samples have contributed to the choice of the best parameters. If we reuse them for performance assessment, we will overestimate performance.
- Solution: nested cross validation:



Support vector machines: performance assessment

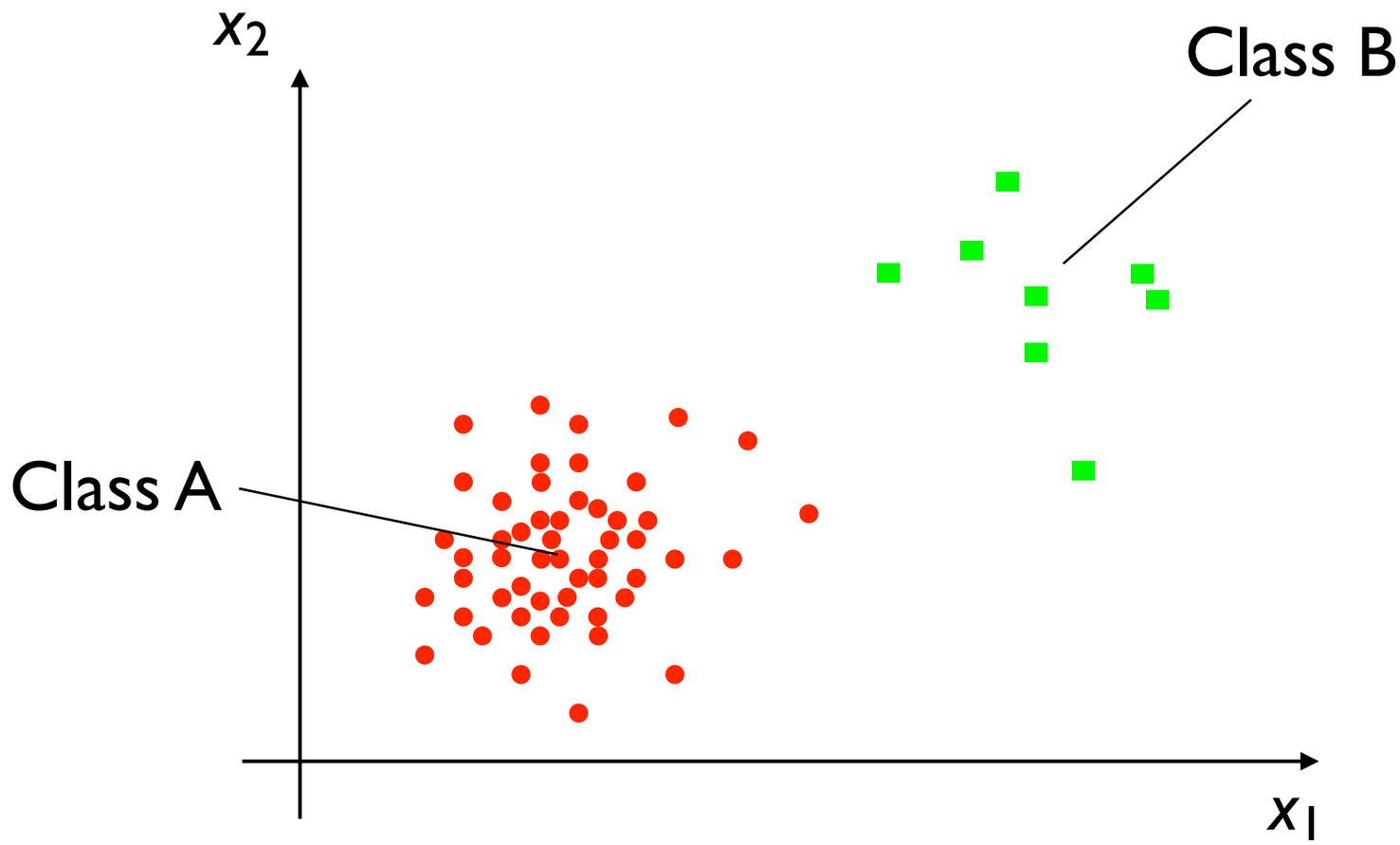
- However, in this way we cannot assess performance by another round of cross validation: all samples have contributed to the choice of the best parameters. If we reuse them for performance assessment, we will overestimate performance.
- Solution: nested cross validation.
- How important the overestimation really is will depend on the size of the dataset, the difficulty of the classification task and the stability of the model.
- Importantly, if the question is only to find the best model (i.e. to find the parameters that perform best), then a simple cross validation is sufficient. Nested cross validation only becomes necessary, when we want to obtain a faithful performance estimation.

Multi-class problems (K classes)



- one vs. one:
  - train a classifier on each pair of classes: K(K-1)/2 binary classifiers
  - The decision is taken by majority vote.
  - In case of ties (or alternatively to the majority votes), the distances to the hyperplane or derived probabilities can be used.
- one vs. all:
  - train a classifier for each class against all other classes together (K binary classifiers).
  - The decision can be reached by taking the maximum distance to the hyperplane.

A remark on unbalanced training data



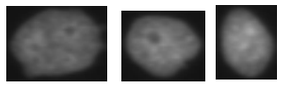
- It can happen, that in the training set, there are much less annotations for one class than for the other class.
- To handle this case, the penalty term C can be weighted in order to give more weight to errors in underrepresented classes. Alternatively, the overrepresented class can be subsampled.

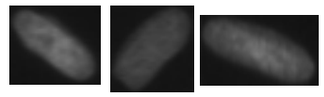
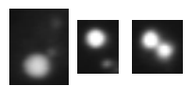
SVM

- SVM work well in high dimensions, even in the case of P > N.
- The optimization is convex (global optimum).
- Training can be slow for large N (but there are ways to avoid this).
- Parameters need to be tuned with cross validation.
- Prediction is computationally very efficient.
- The kernel trick allows to apply SVM on more complicated data structures (other than feature vectors).

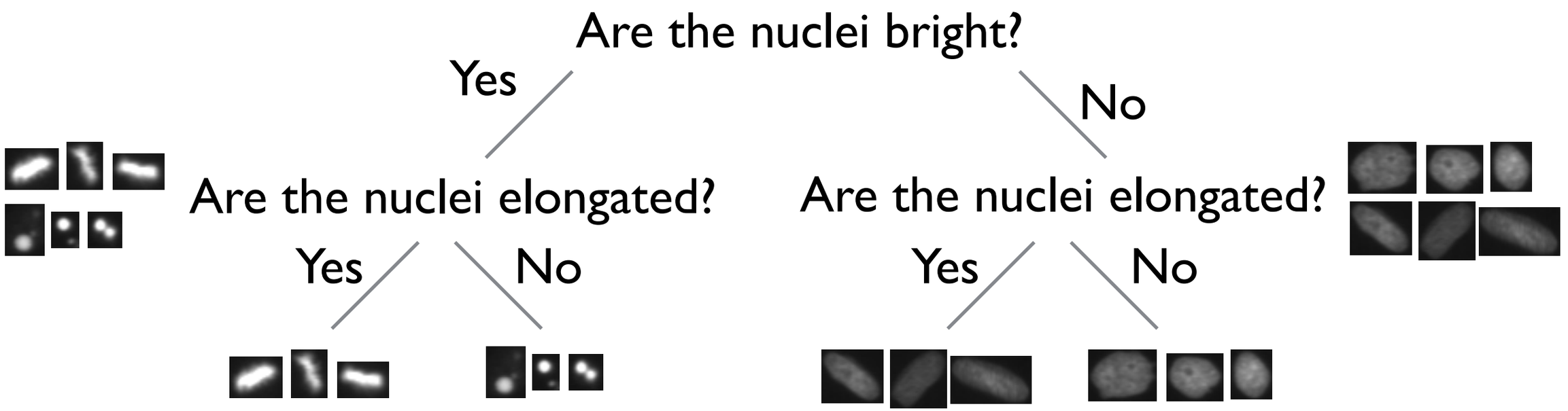
Random forests: decision trees

- Decision trees provide an intuitive classification procedure, by subsequently applying simple rules.
- For illustration, we take the following classification problem:

Interphase: 
Metaphase: 

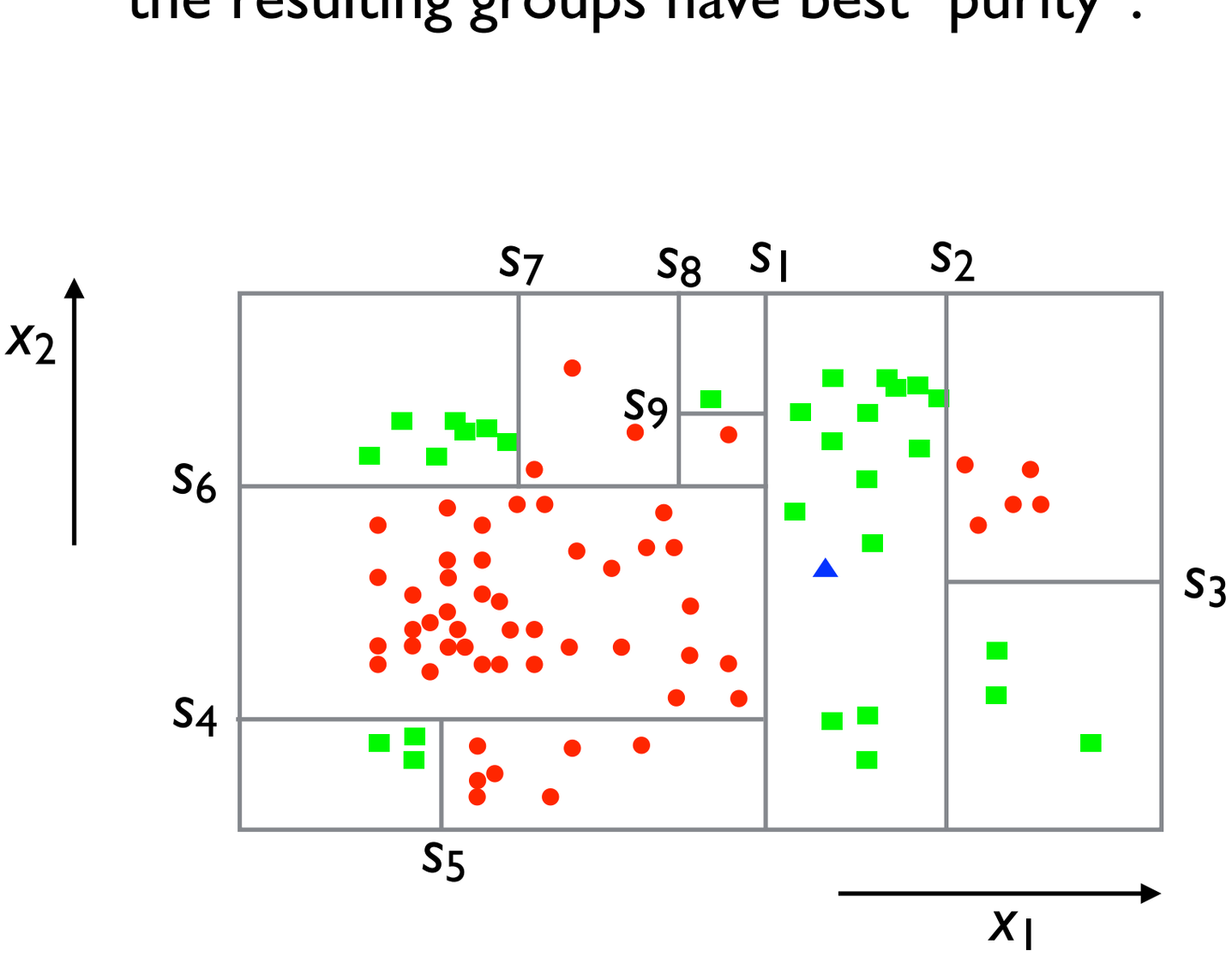
Elongated: 
Apoptosis: 

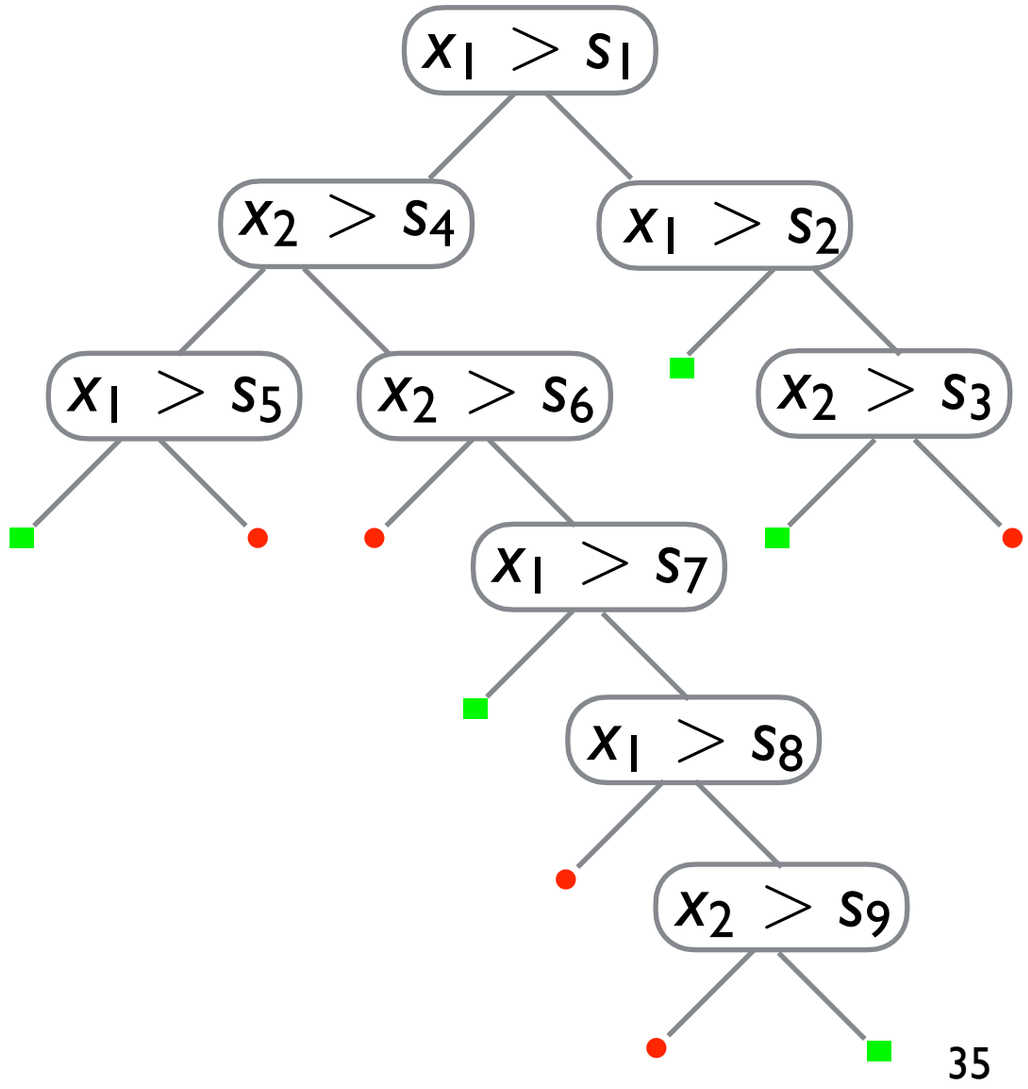
- And we now classify them according to the following scheme:



Random forests: decision trees

- Such decision trees can be built from data in an optimal way, by repeated division of the feature space.
- At every step, we split the data set into two, according to one feature.
- The feature and the threshold are chosen automatically in such a way that the resulting groups have best "purity".





Random forests: decision trees

- Such decision trees can be built from data in an optimal way, by repeated division of the feature space.
- At every step, we split the data set into two, according to one feature.
- The feature and the threshold are chosen automatically in such a way that the resulting groups have best "purity".
- This can be achieved by minimizing the GINI impurity. For *K* classes, GINI impurity is defined as:

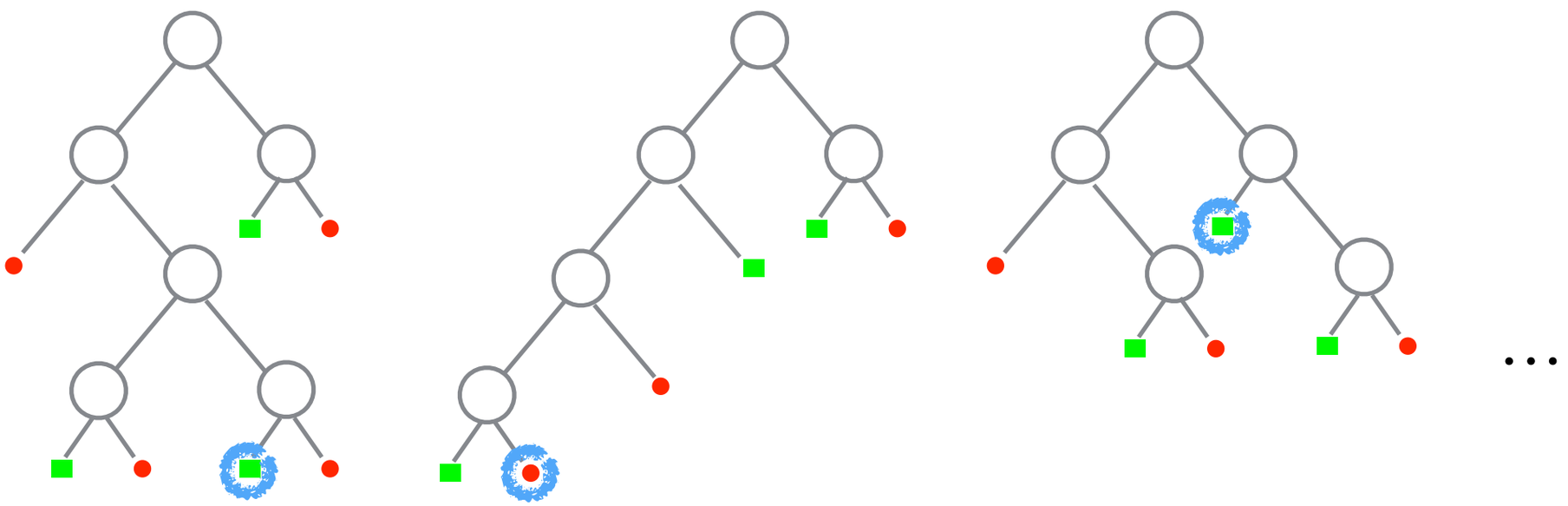
$$GI(R_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

$$GI(s) = \sum_{R_m} \frac{|R_m|}{N} GI(R_m)$$

where R_m are the sets resulting from the split s and $\hat{p}_{mk}$ is the probability of a sample in R_m to belong to class k.

Random forests

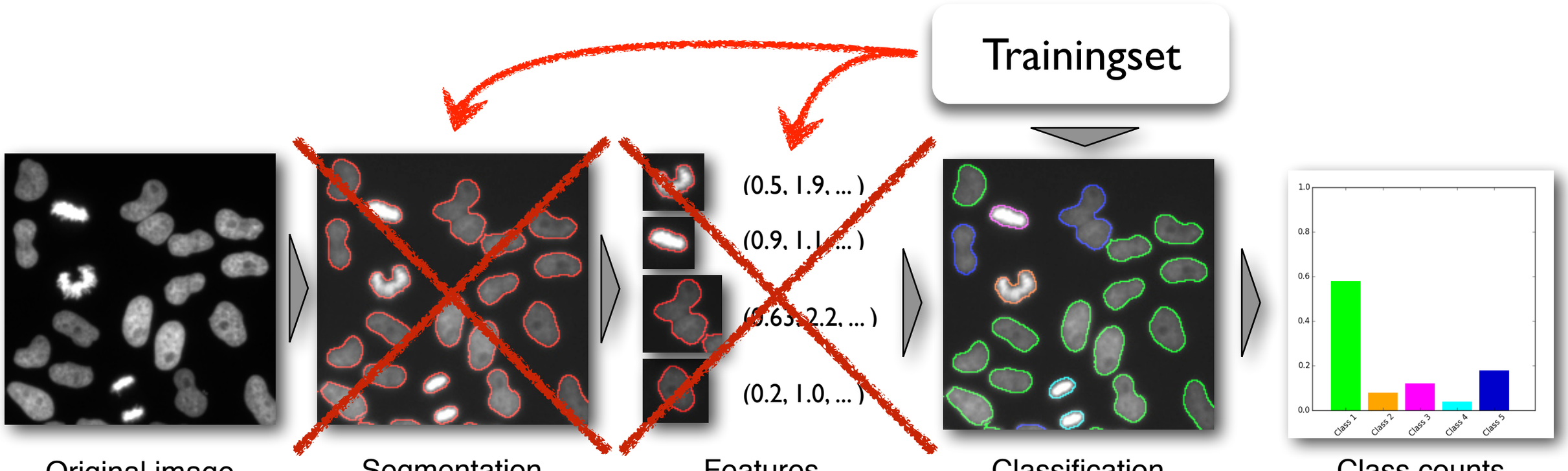
- Decision trees can approximate very complicated decision boundaries, but they tend to "fit to much" to the data (overfitting).
- Random forests: set of decision trees, each learned on a different (**randomly drawn**) portion of the data and with different (**randomly selected**) features.
- Each tree will give a classification result.
- The final result is obtained by a majority vote.



Random forests

- Decision trees can approximate very complicated decision boundaries, but they tend to "fit to much" to the data (overfitting).
- Random forests: set of decision trees, each learned on a different (**randomly drawn**) portion of the data and with different (**randomly selected**) features.
- Each tree will give a classification result.
- The final result is obtained by a majority vote.
- Typically, we build many trees (500-1000) to build this majority vote. Each one of these trees is a "weak classifier".
- Together, they usually give very good result **without parameter tuning**.
- Random forests work well in high dimensions, even in the case of P > N.

Deep Learning



Original image | Segmentation | Features | Classification | Class counts

- Neural networks have gained a lot of interest in the recent years - under a new name: **deep learning**.
- The rationale of deep learning is that unlike in traditional methods, we do not only optimize the label assignment, but also the representation of the data (**no hand-crafted features**).
- Following the end-to-end paradigm, we might would not go for single cell classification, but classify the effect on the entire population.

Deep Learning for Bioimaging

- Deep learning often outperforms competing methods, in particular for image classification, given that **much annotated data** is at hand.
- In some cases, deep learning does outperform humans.
- The question today: how to generate large amounts of annotated data?
  - simulation
  - crowd sourcing
  - online games
  - intelligent experiment design (external ground truth by experiments).
- Another strategy: adapt existing networks to new problems.
- The situation is strikingly different in medical imaging (MI):
  - In MI, there are strict protocols for image acquisition, and a relatively structured way of annotating them. This makes the generation of large and coherent data sets much easier.
  - In Bioimaging, protocols change rapidly, images are very different from each other and

Supervised vs. unsupervised learning

Unsupervised learning

- Learn classes from the data.
- Advantages:
  - unbiased
  - find "new" morphologies
- Disadvantages:
  - computational complexity
  - biological sense not guaranteed

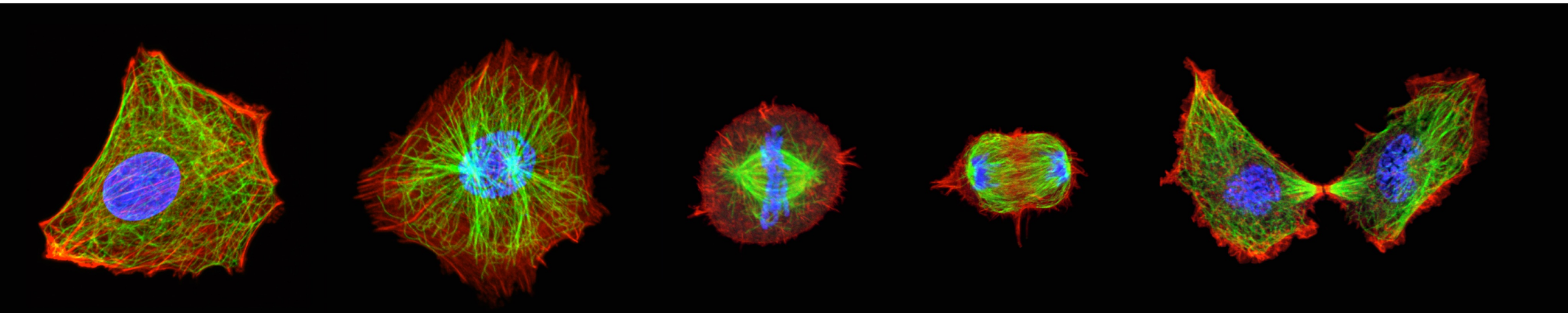
Supervised Learning

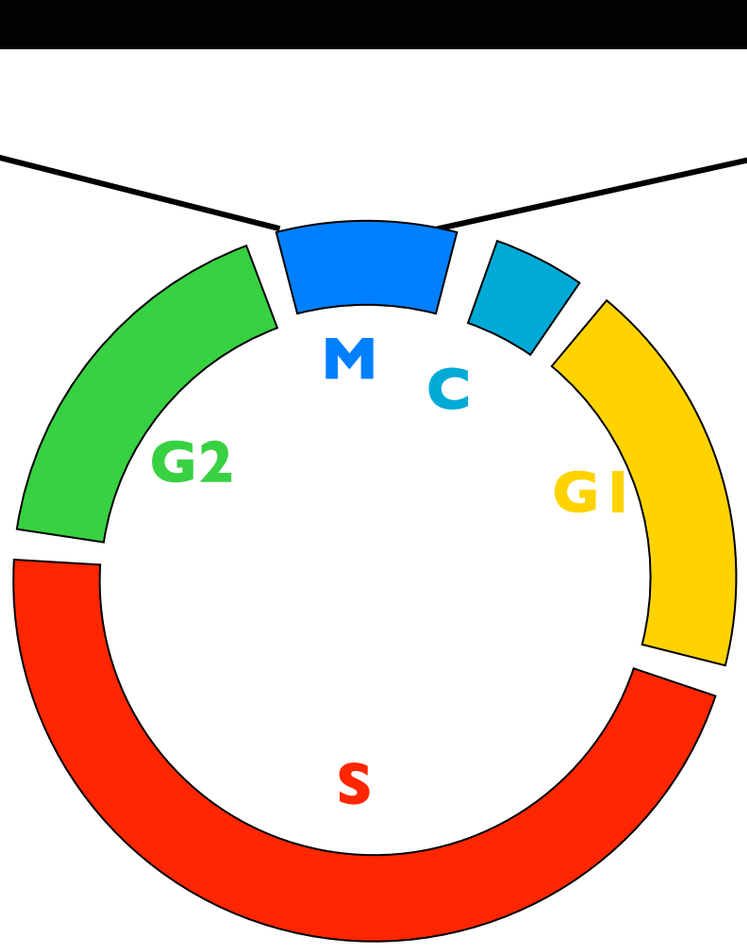
- Classes are defined from known phenotypes.
- Advantages:
  - biologically meaningful classes
  - most natural approach
- Disadvantages:
  - biased approach
  - tedious redefinition and reprocessing when new classes are found

Application of computational phenotyping to High Content Screening (HCS)

Morphological Changes during Mitosis

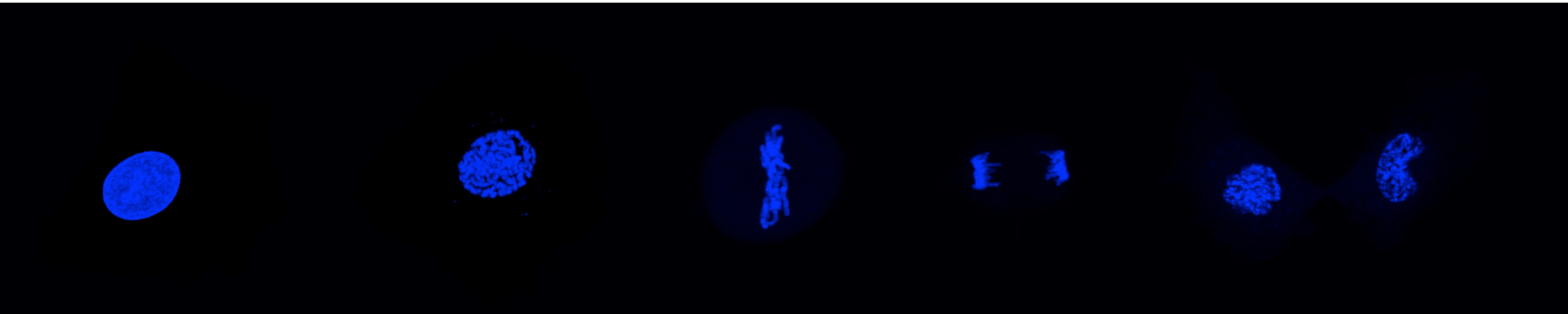
Interphase Prophase Metaphase Anaphase Telophase

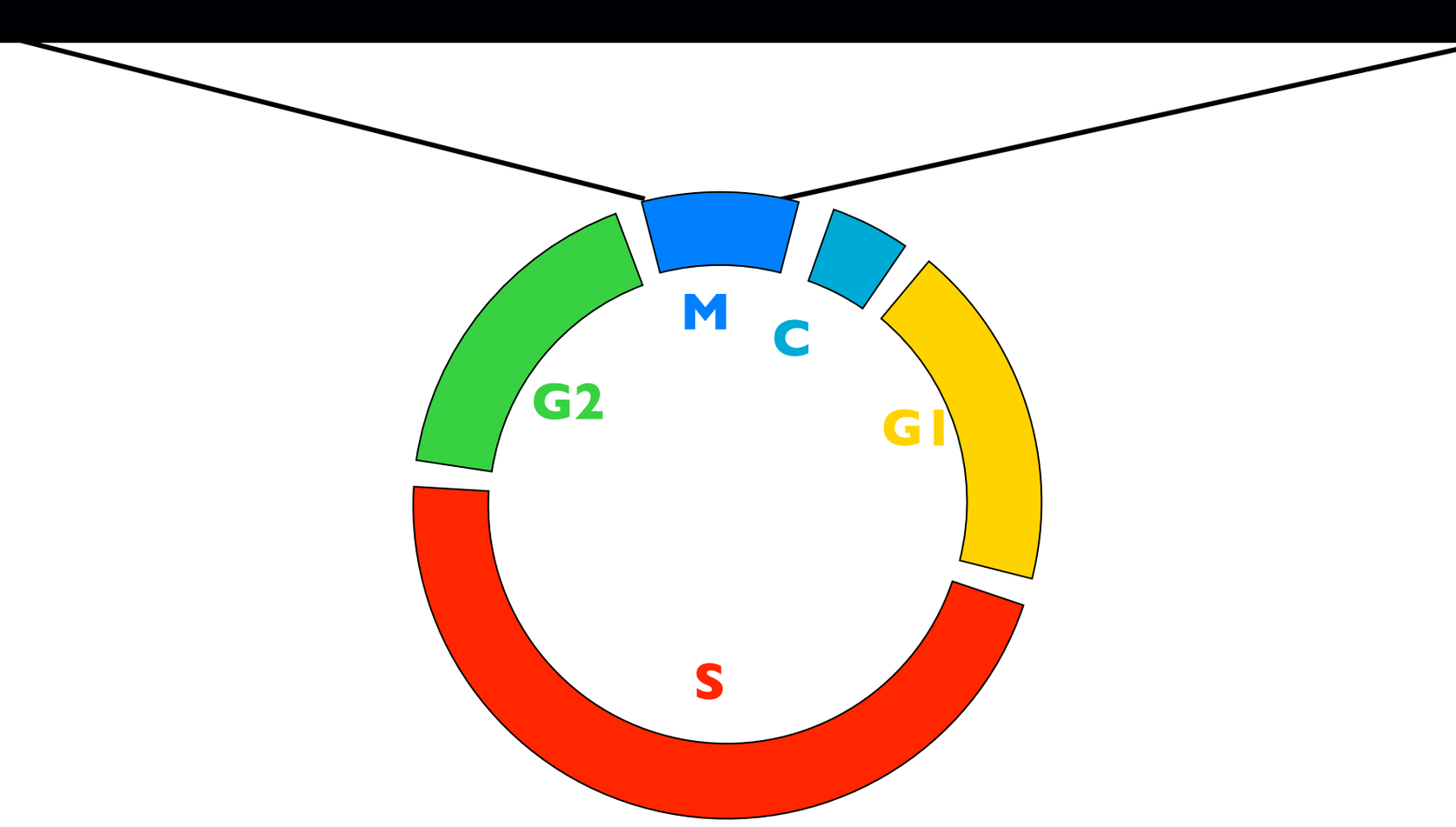




Morphological Changes during Mitosis

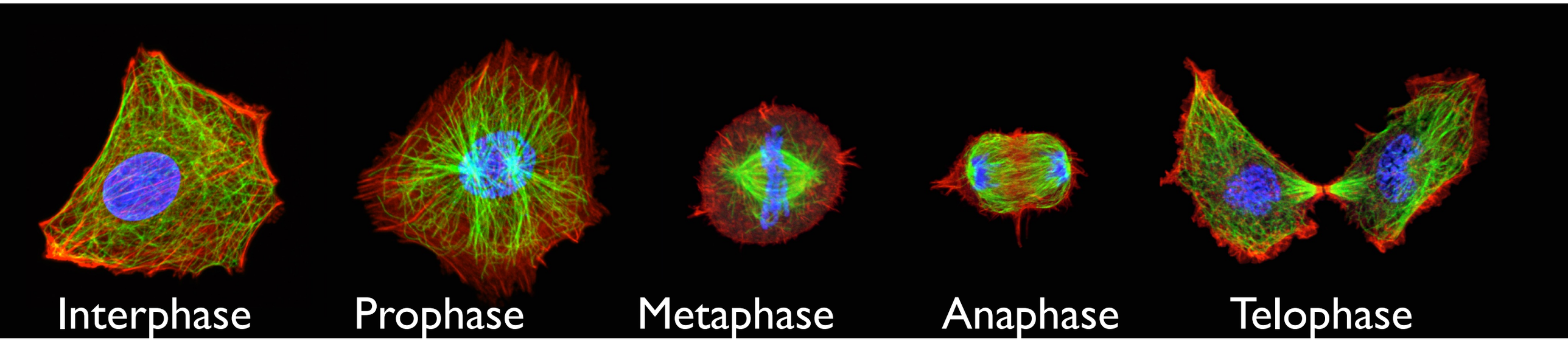
Interphase Prophase Metaphase Anaphase Telophase



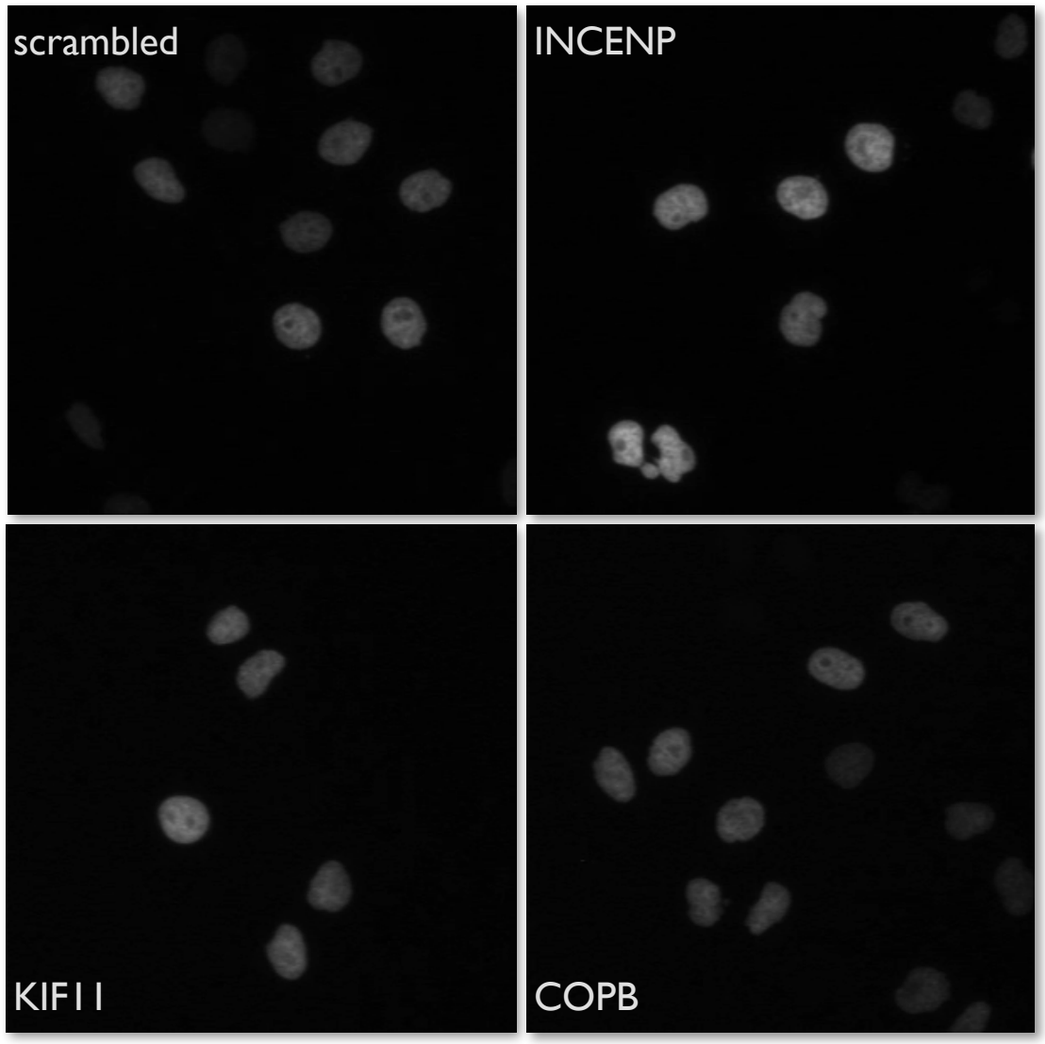


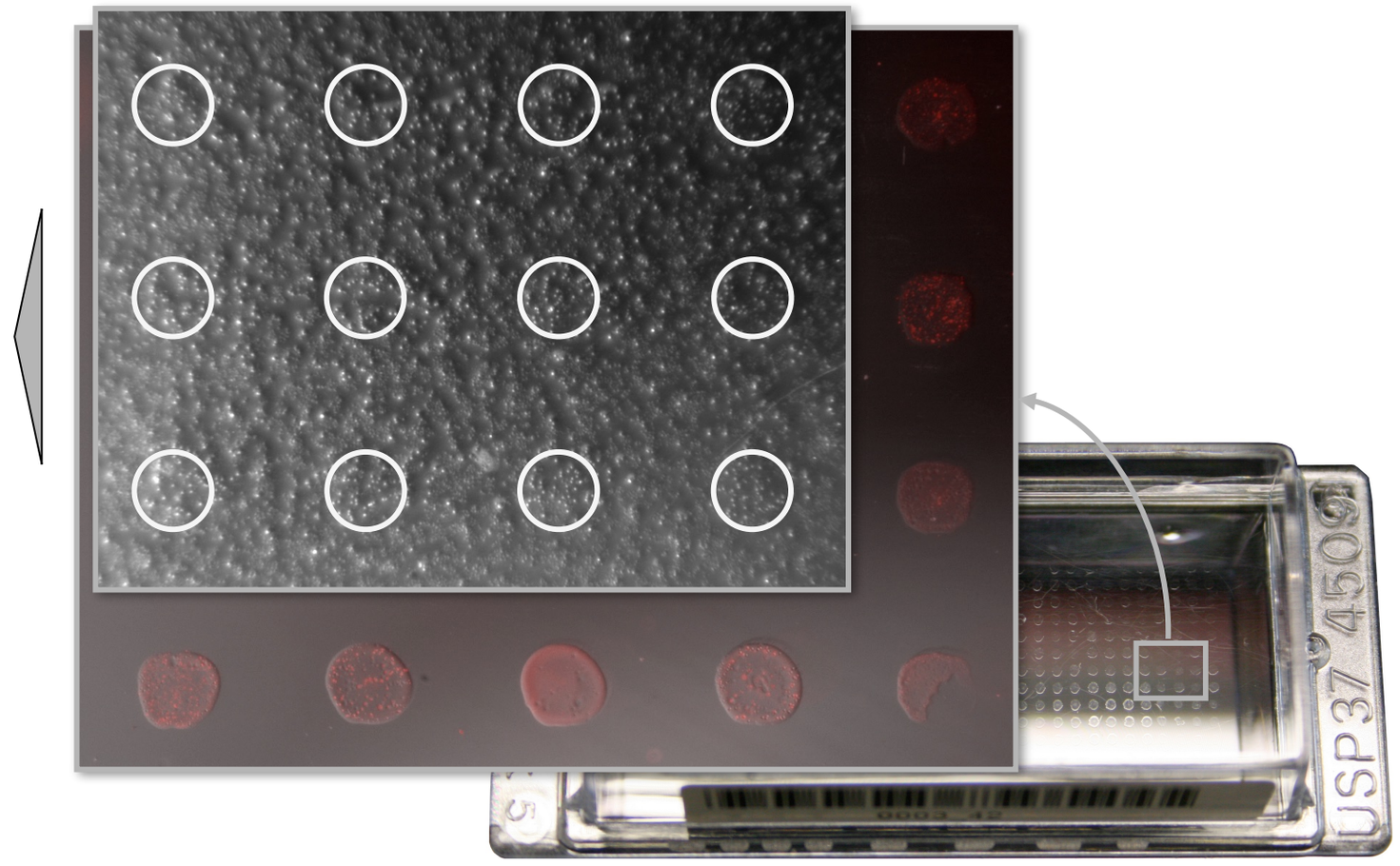
The molecular basis of cellular functions

- While we know the sequences of most genes in many organisms, we still have a very incomplete picture of the function of these genes.
- Functional genomics aims at understanding which gene is important for which biological process and by what mechanism.
- One approach to study this is called "loss-of-function": we remove the gene product from the system and we observe what happens.
- Here we wanted to identify the human genes required for cell division.



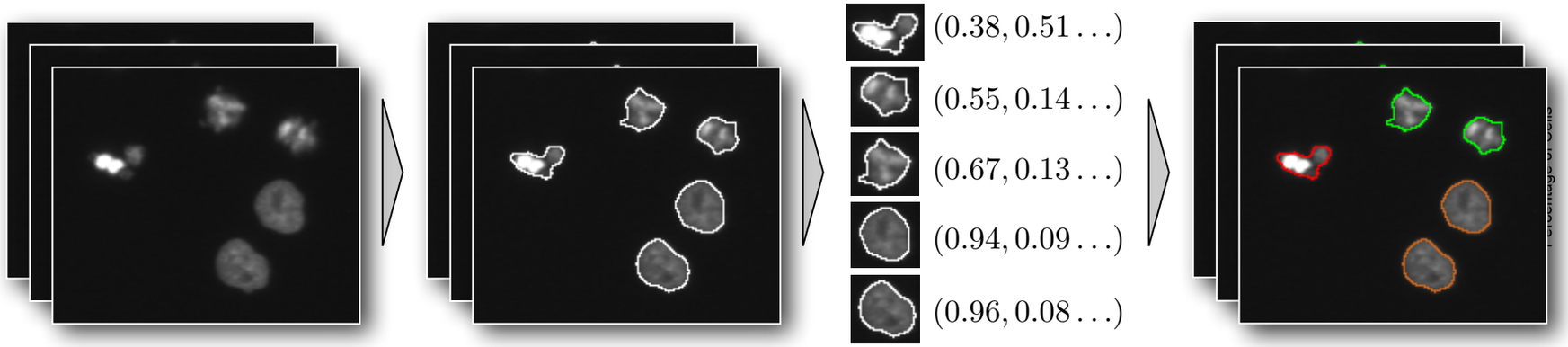
A genome wide RNAi screen for the identification of mitotic genes



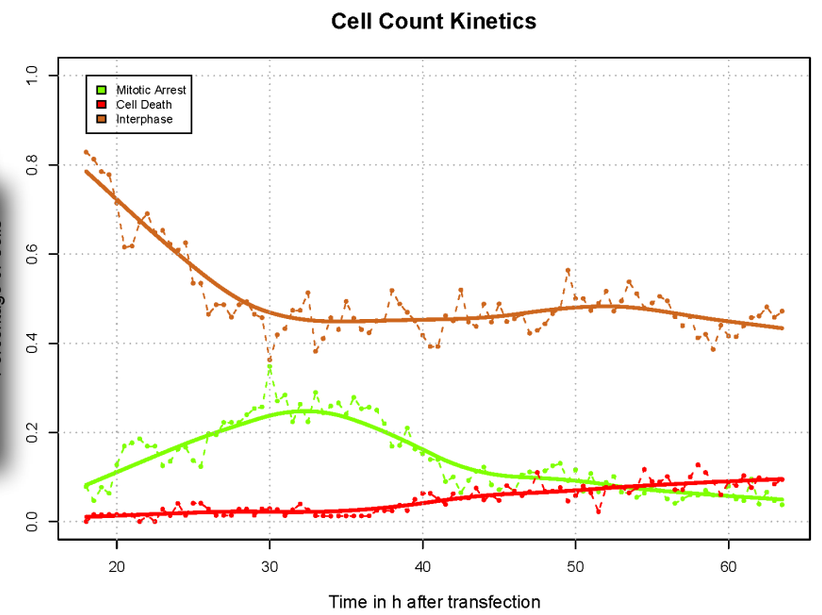


- Imaging: 48 hours, time-lapse = 30 min
- data for ~21 000 protein coding genes (2 siRNAs, 3 replicates)
- ~200 000 movies (after QC)
- storage (raw image data): ~24 TB (after QC)

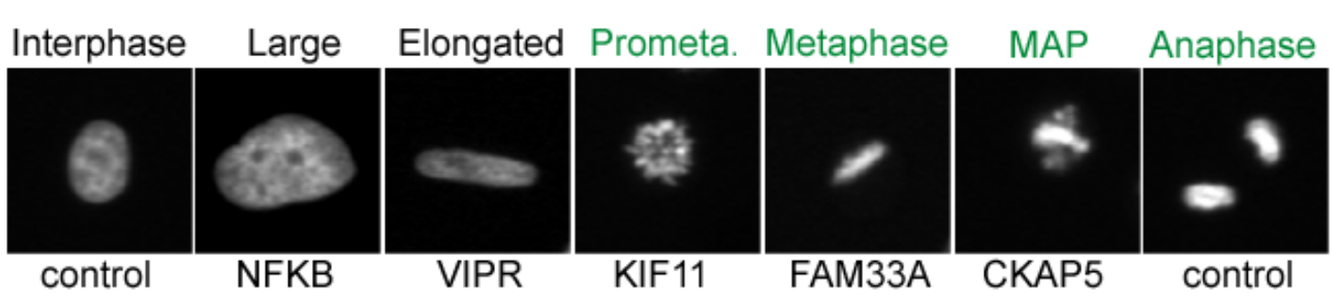
Analysis of high-throughput time-lapse microscopy data



$(0.38, 0.51\ldots)$
$(0.55, 0.14\ldots)$
$(0.67, 0.13\ldots)$
$(0.94, 0.09\ldots)$
$(0.96, 0.08\ldots)$

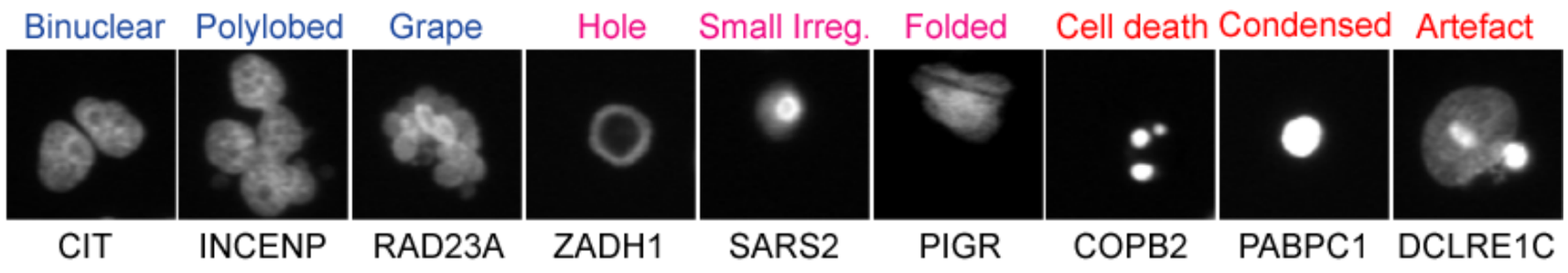


Nuclear morphologies



Mitotic consequences

Dynamic changes



1247 mitotic candidate genes

572 validated mitotic hits

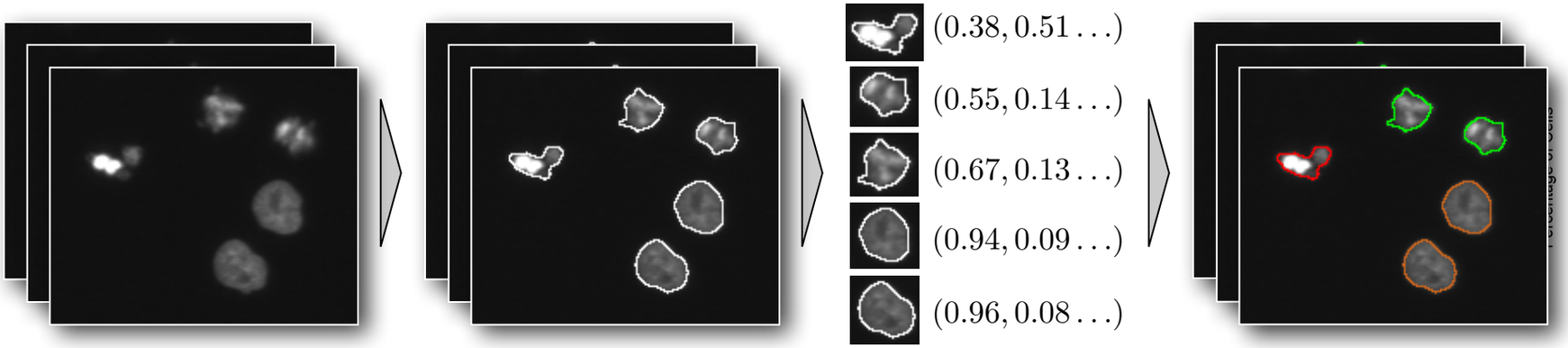
Neumann, Walter et al., Nature. 2010 Apr 1;464(7289).

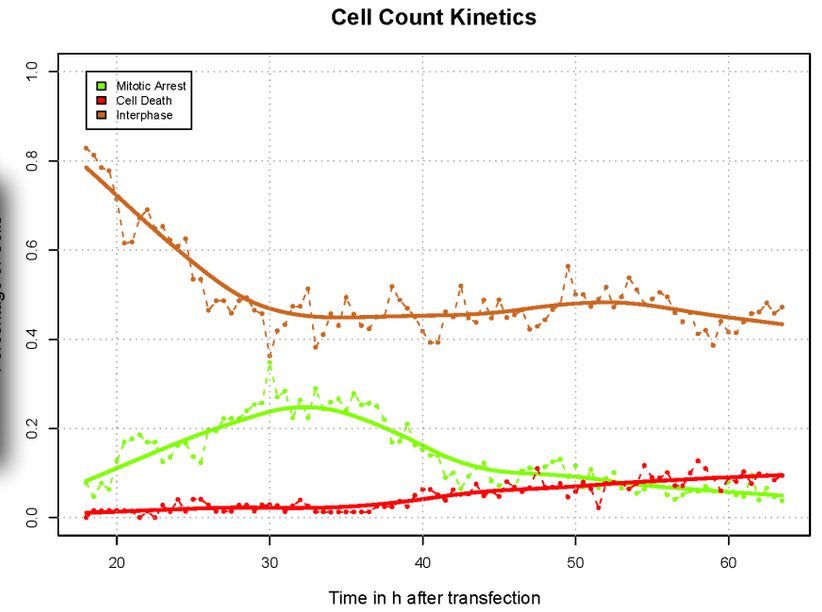
Walter, Held et al., J Struct Biol. 2010 Apr;170(1).

Pau, Walter, et al. BMC Bioinf. 2013 Oct 16; 14(308).

Schoenauer Sebag, et al. Bioinf. 2015 Jun; 31(12).

Analysis of high-throughput time-lapse microscopy data

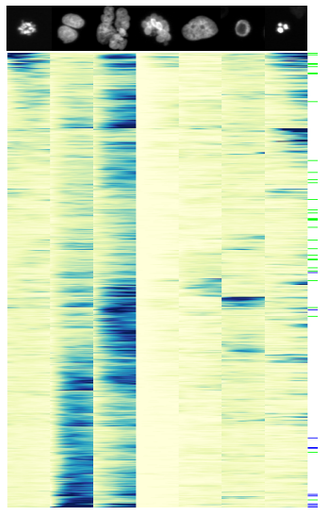




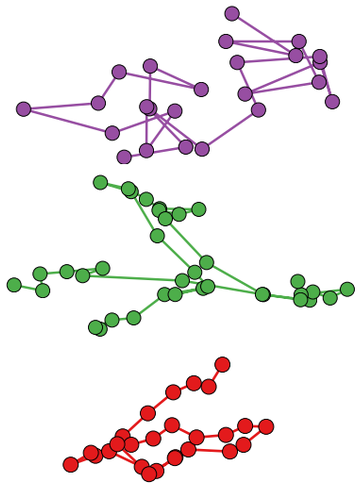
1247 mitotic candidate genes

572 validated mitotic hits

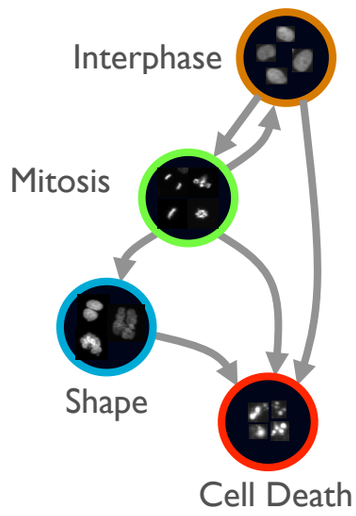
Time series clustering in order to identify genes with similar phenotypes

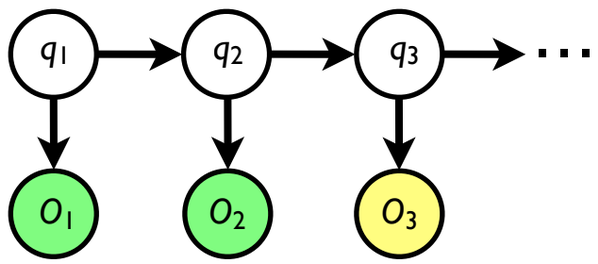


Trajectory analysis in order to find genes with similar movement



ODE modeling in order to find genes with similar phenotypic transitions





HMM for error correction and fate analysis.

Neumann, Walter et al., Nature. 2010 Apr 1;464(7289).

Walter, Held et al., J Struct Biol. 2010 Apr;170(1).

Pau, Walter, et al. BMC Bioinf. 2013 Oct 16; 14(308).

Schoenauer Sebag, et al. Bioinf. 2015 Jun; 31(12).

Conclusion

- Computer Vision for Cell Biology: classification of cellular morphologies
  - Segmentation
  - Feature extraction
  - Classification

- Applications in HCS:
  - Screening as a tool in functional genomics.
  - Examples for large scale screens.
  - Phenotypic distances and phenotypic clustering.

Bibliography

(1) Hastie, T., Tibshirani, R., & Friedman, J. (2009). *Statistical Learning: Data Mining, Inference and Prediction*.

(2) Jones, T. R., Carpenter, A., & Golland, P. (2005). Voronoi-Based Segmentation of Cells on Image Manifolds. *Computer Vision for Biomedical Image Applications, LNCS*, *3765*.

(3) Lamprecht, M., Sabatini, D., & Carpenter, A. (2007). CellProfiler™: free, versatile software for automated biological image analysis. *BioTechniques*, *42*(1), 71–75.

(4) Neumann, B., Walter, T., et al. (2010). Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. *Nature*, *464*(7289), 721–7.

(5) Rajaram, S., Pavie, B., Wu, L. F., & Altschuler, S. J. (2012). PhenoRipper: software for rapidly profiling microscopy images. *Nature Methods*, *9*(7), 635–637.

(6) Reeve, R. J., & Prokop, A. P. (1992). A Survey of Moment-Based Techniques For Unoccluded Object Representation and Recognition. CVGIP: Graphical Models and Image Processing, 438–460.

(7) Roerdink, J. B. T. M., & Meijster, A. (2001). The Watershed Transform: Definitions , Algorithms and Parallelization Strategies. *Fundamenta Informaticae*, *41*, 1–40.

(8) Soille, P. (2003). *Morphological Image Analysis: Principles and Applications* (2nd ed.). Springer-Verlag New York, Inc.

(9) Terjung, S., Walter, T. et al. (2010). High-throughput microscopy using live mammalian cells. In *Live Cell Imaging: A Laboratory Manual*. Cold Spring Harbor Laboratory Press, Cold Spring Harbor.

(10) Wählby, C., et al. (2002). Algorithms for cytoplasm segmentation of fluorescence labelled cells. *Analytical cellular pathology: the journal of the European Society for Analytical Cellular Pathology*, *24*(2-3), 101–11.

(11) Walker, R. F., & Jackway, P. T. (1996). Statistical Geometric Features - Extensions for Cytological Texture Analysis. *ICPR - International Conference on Pattern Regognition*.

(12) Walter, T., Held, M., et al. (2010). Automatic identification and clustering of chromosome phenotypes in a genome wide RNAi screen by time-lapse imaging. *Journal of structural biology*, *170*(1), 1–9.

Annex

Unsupervised learning in a nutshell

Clustering and HCS

- First level: unsupervised learning can be used in order to identify morphological (or phenotypic) classes from large scale image data.

- Second level: unsupervised learning is often used to describe similarities between genes or drug treatment: if a phenotype is showing high similarity, we can assume that the genes have similar function or the drug has similar mechanism of action.

Hierarchical clustering in a nutshell

- Agglomerative: bottom-up approach
  - start with single data points as separate clusters
  - join the two closest clusters
  - calculate the single cluster's new representation
  - iterate this until there is nothing more to merge

- Divisive: top-down approach
  - start with one single cluster
  - divide the cluster into two according to some heuristic
  - iterate this until there is nothing more to merge
  - Here we deal only with agglomerative hierarchical clustering methods.

Hierarchical clustering in a nutshell

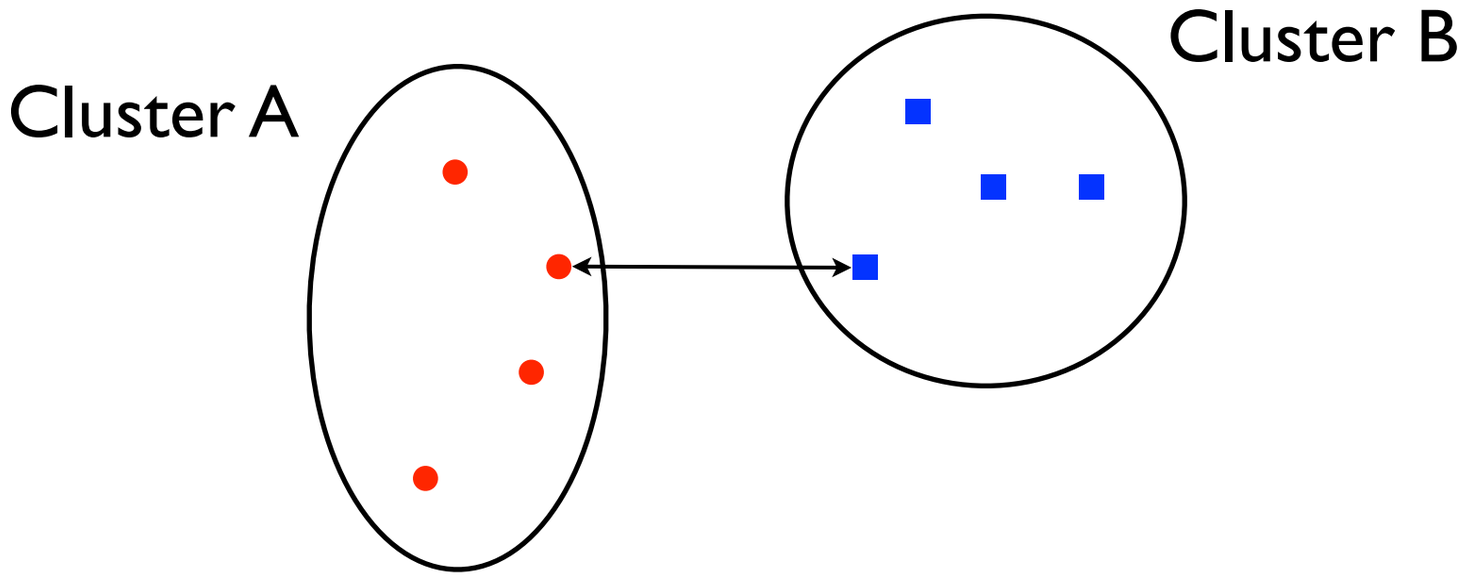
- Hierarchical clustering is widely used in biology:
  - No prior information, except for the choice of the method (e.g. no number of clusters has to be given).
  - Nested structure allows to study different levels of organization in the data.
  - Results can be visualized in dendrograms and heatmaps.

- Two elements: distance between points and distance between clusters

Hierarchical clustering in a nutshell: distance metrics

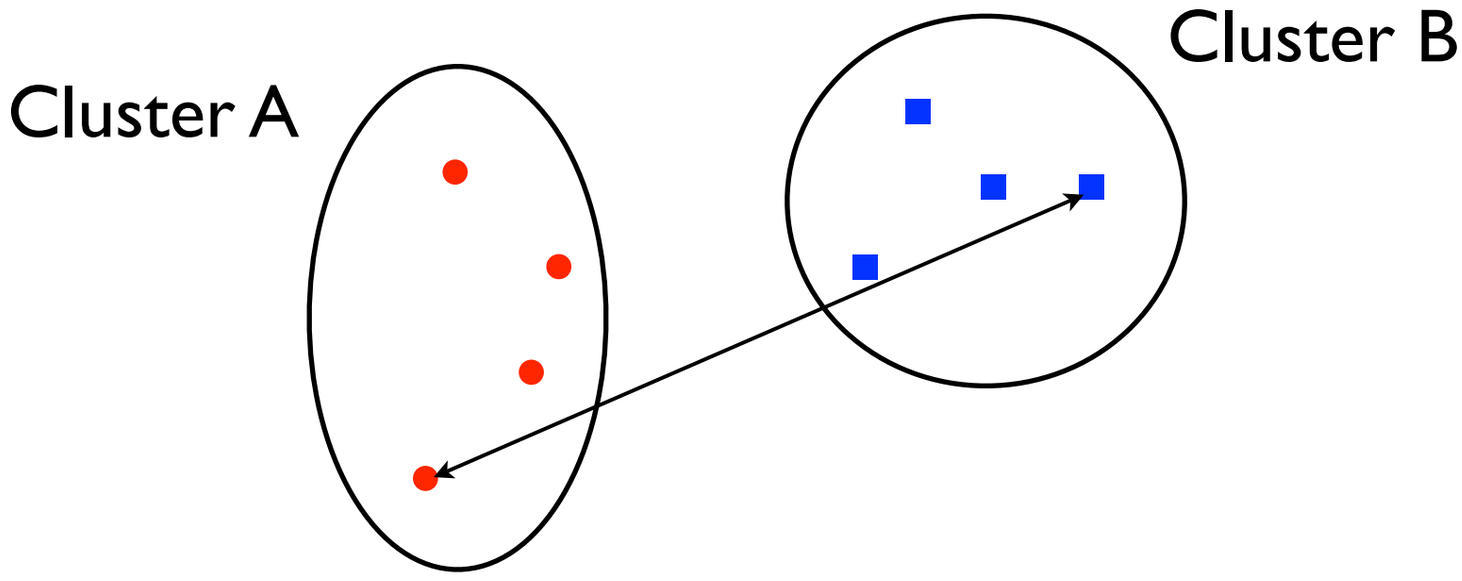
- Euclidean distance: $d_{i,j}^2 = (x_i - x_j)^T(x_i - x_j) = \sum_k (x_{i,k} - x_{j,k})^2$
- Manhattan distance: $d_{i,j} = \sum_k |x_{i,k} - x_{j,k}|$
- Generalized Euclidean distance: (e.g. Mahalanobis distance): $d_{i,j}^2 = (x_i - x_j)^T A (x_i - x_j)$
- Correlation based distance: $d_{i,j} = \frac{1}{2} - \frac{1}{2}\rho(x_i, x_j)$
- Hamming distance (string vectors): $d_{i,j} = |\{k \mid x_{i,k} \neq x_{j,k}\}|$
- User defined distances

Hierarchical clustering in a nutshell: agglomeration functions



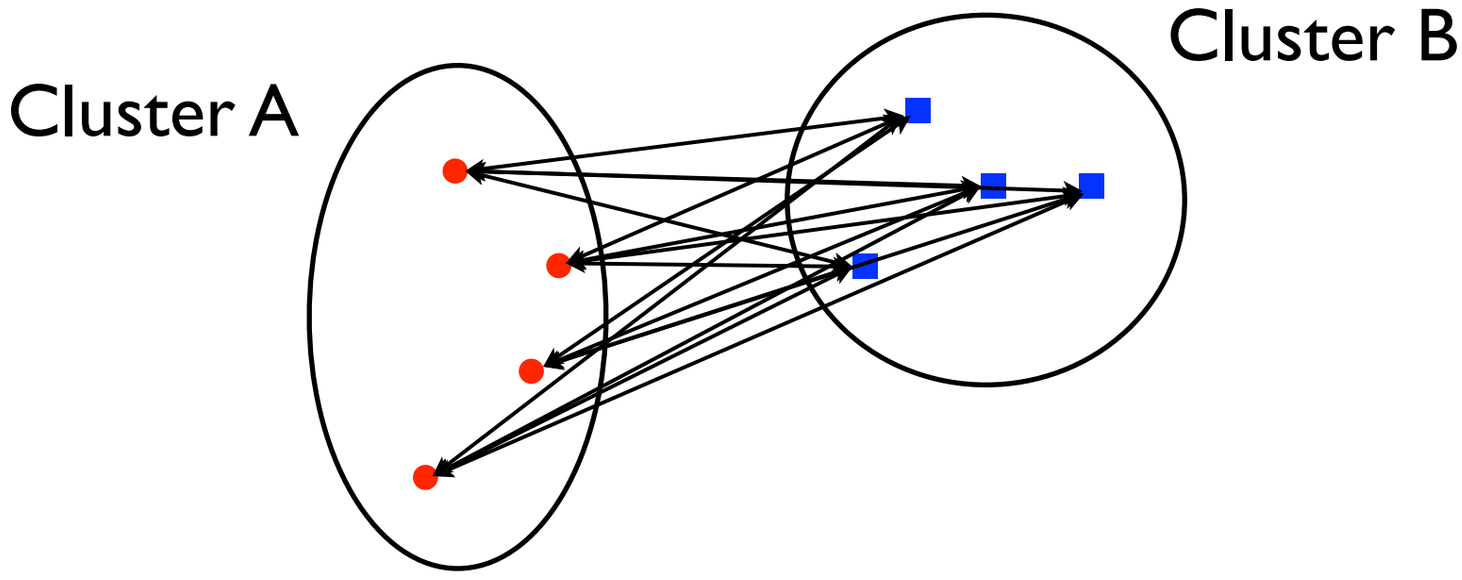
- Single linkage: $d_{A,B} = \min_{(x_i, x_j) \in A \times B} d_{i,j}$

Hierarchical clustering in a nutshell: agglomeration functions



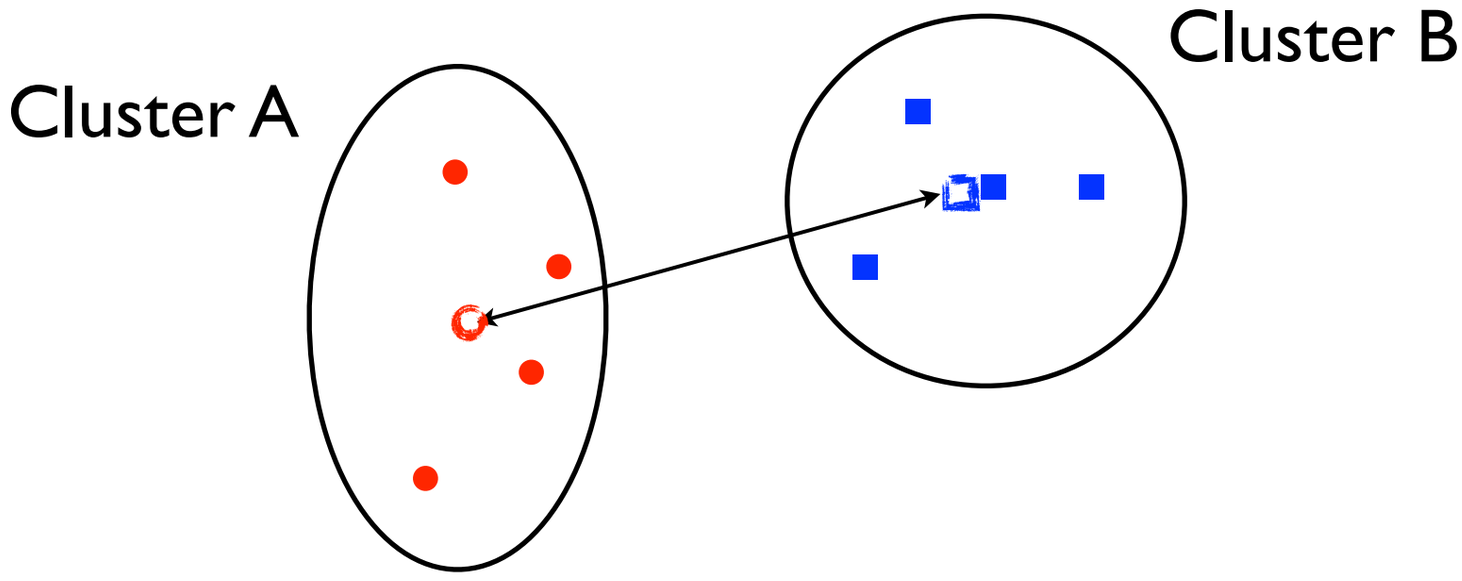
- Single linkage: $d_{A,B} = \min_{(x_i,x_j)\in A\times B} d_{i,j}$
- Complete linkage: $d_{A,B} = \max_{(x_i,x_j)\in A\times B} d_{i,j}$

Hierarchical clustering in a nutshell: agglomeration functions



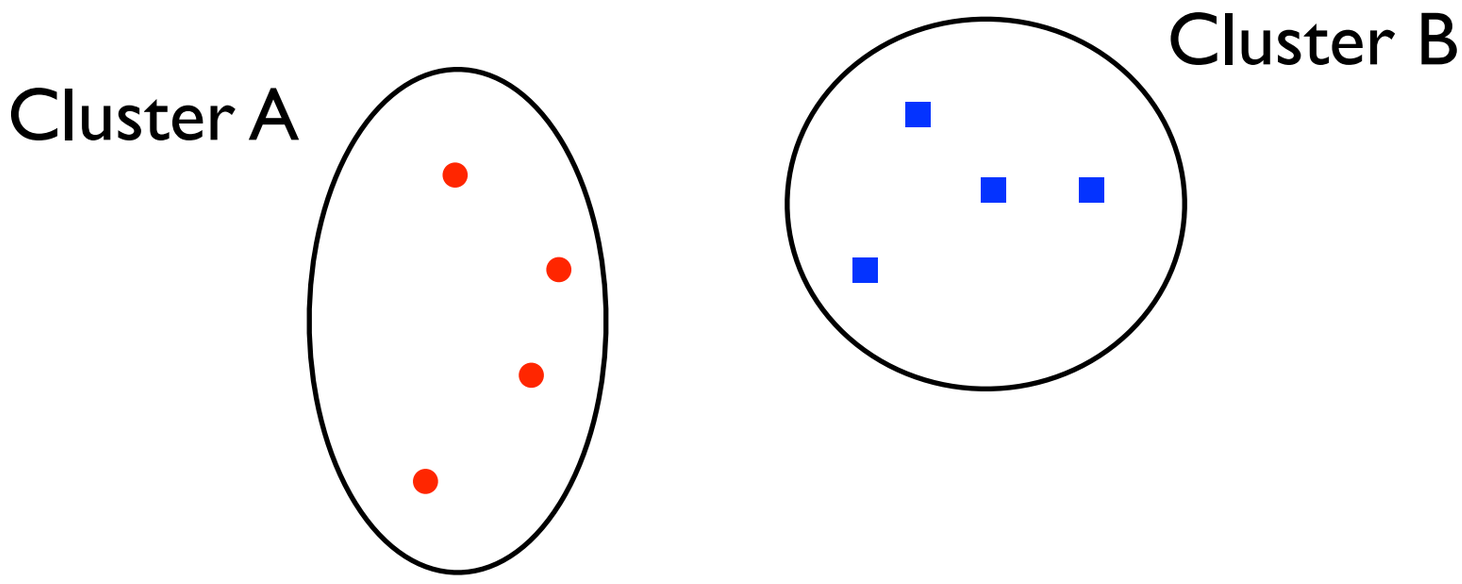
- Single linkage: $d_{A,B} = \min_{(x_i,x_j)\in A\times B} d_{i,j}$
- Complete linkage: $d_{A,B} = \max_{(x_i,x_j)\in A\times B} d_{i,j}$
- Average linkage: $d_{A,B} = \frac{1}{N_A N_B} \sum_{(x_i,x_j)\in A\times B} d_{i,j}$

Hierarchical clustering in a nutshell: agglomeration functions



- Single linkage: $d_{A,B} = \min_{(x_i,x_j)\in A\times B} d_{i,j}$
- Complete linkage: $d_{A,B} = \max_{(x_i,x_j)\in A\times B} d_{i,j}$
- Average linkage: $d_{A,B} = \frac{1}{N_A N_B} \sum_{(x_i,x_j)\in A\times B} d_{i,j}$
- Centroid: $d_{A,B} = d(\mu_A, \mu_B)$ with μ_A, μ_B the average of x in A and B respectively.

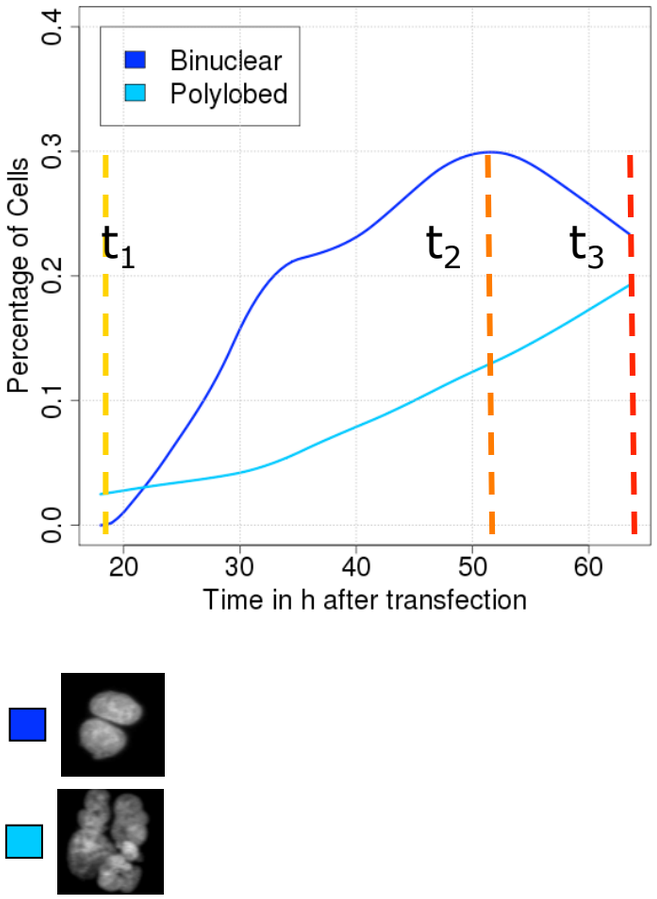
Hierarchical clustering in a nutshell: agglomeration functions

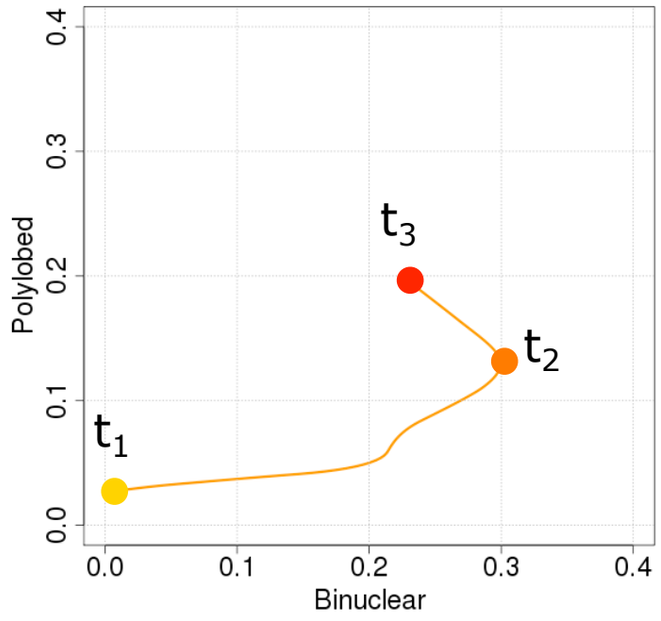


- Single linkage: $d_{A,B} = \min_{(x_i,x_j)\in A\times B} d_{i,j}$
- Complete linkage: $d_{A,B} = \max_{(x_i,x_j)\in A\times B} d_{i,j}$
- Average linkage: $d_{A,B} = \frac{1}{N_A N_B} \sum_{(x_i,x_j)\in A\times B} d_{i,j}$
- Centroid: $d_{A,B} = d(\mu_A, \mu_B)$ with μ_A, μ_B the average of x in A and B respectively.
- Ward: $d_{A,B} = \frac{N_A N_B}{N_A+N_B} \|\mu_A - \mu_B\|^2$ with μ_A, μ_B the average of x in A and B respectively. The ward method chooses the cluster fusion producing the smallest increase in variance.

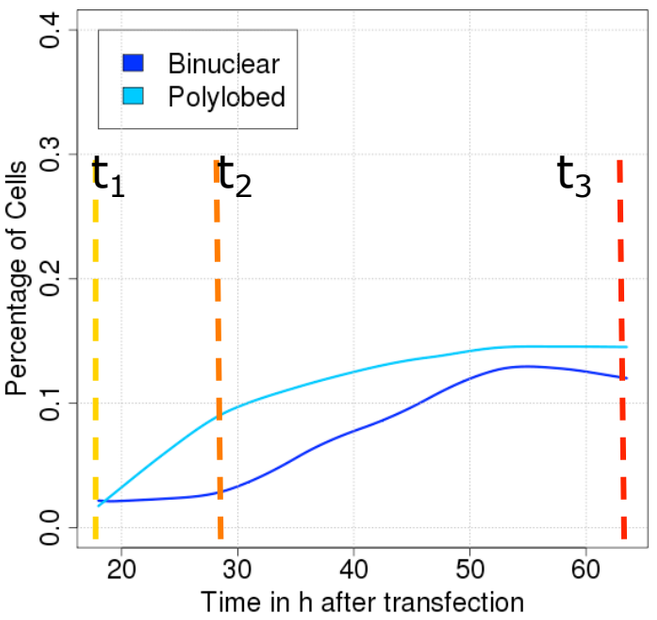
Applications (phenotypic clustering)

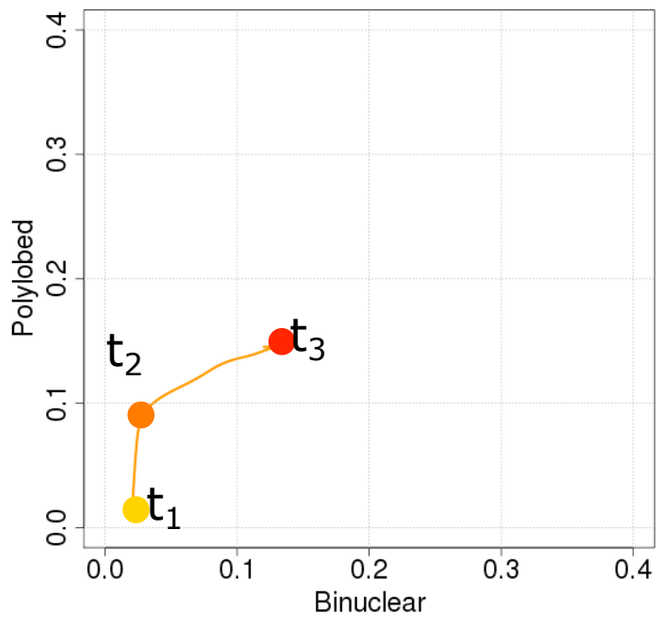
Phenotypic Clustering of the mitotic hits

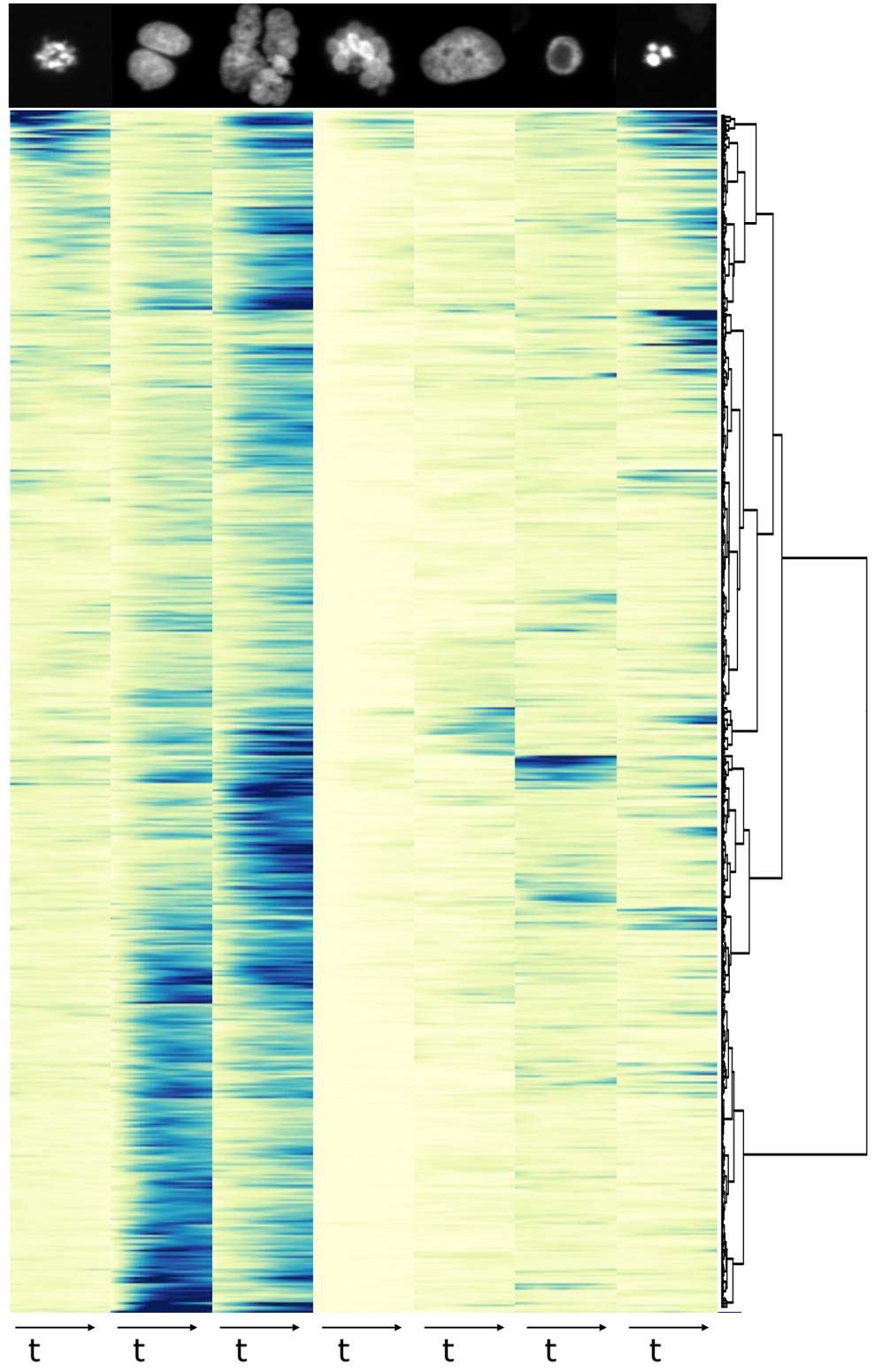




Trajectory distance measure

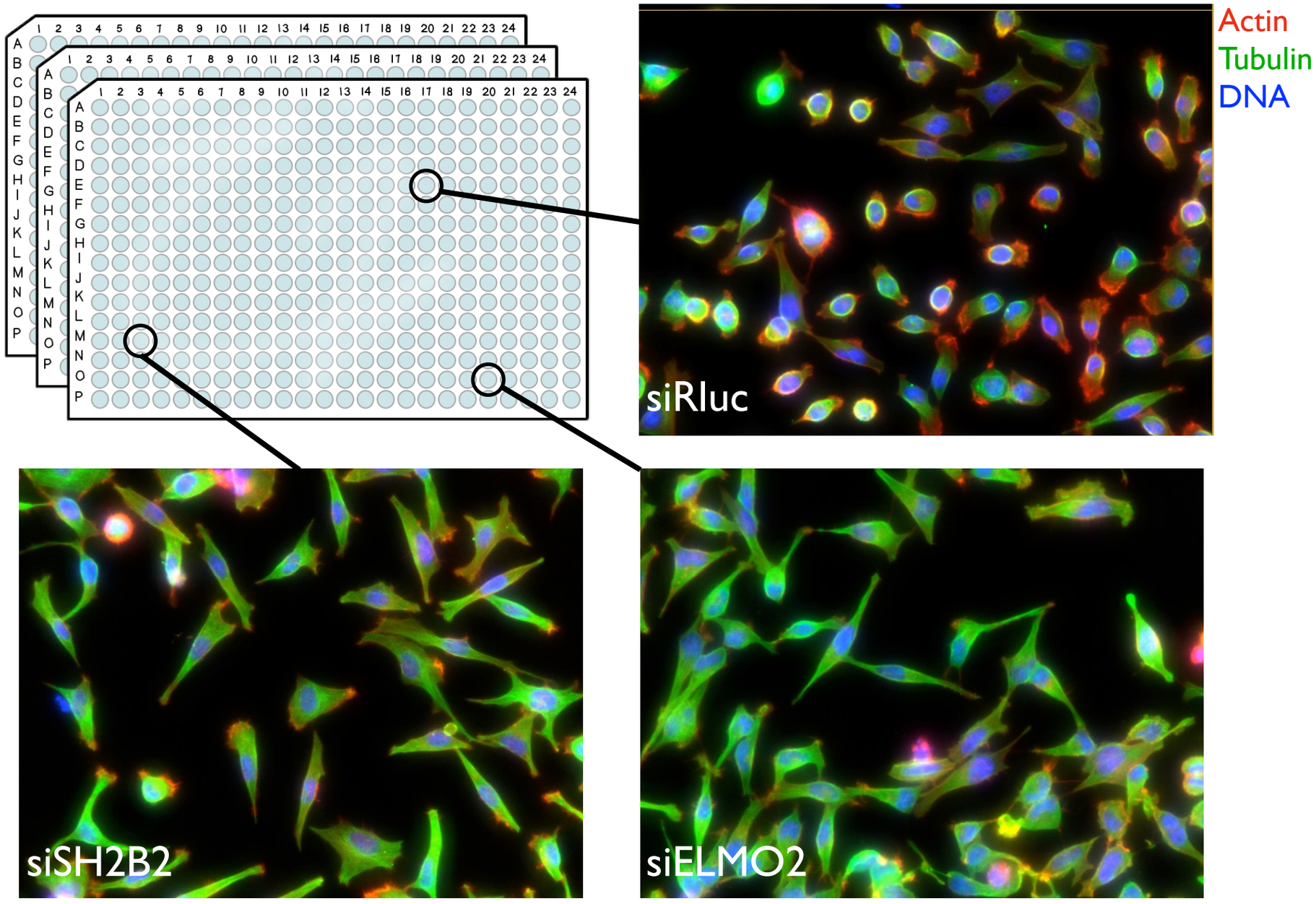






More on phenotypic distances

Genome-wide Screen in HeLa cells with 3 markers (endpoint assay).

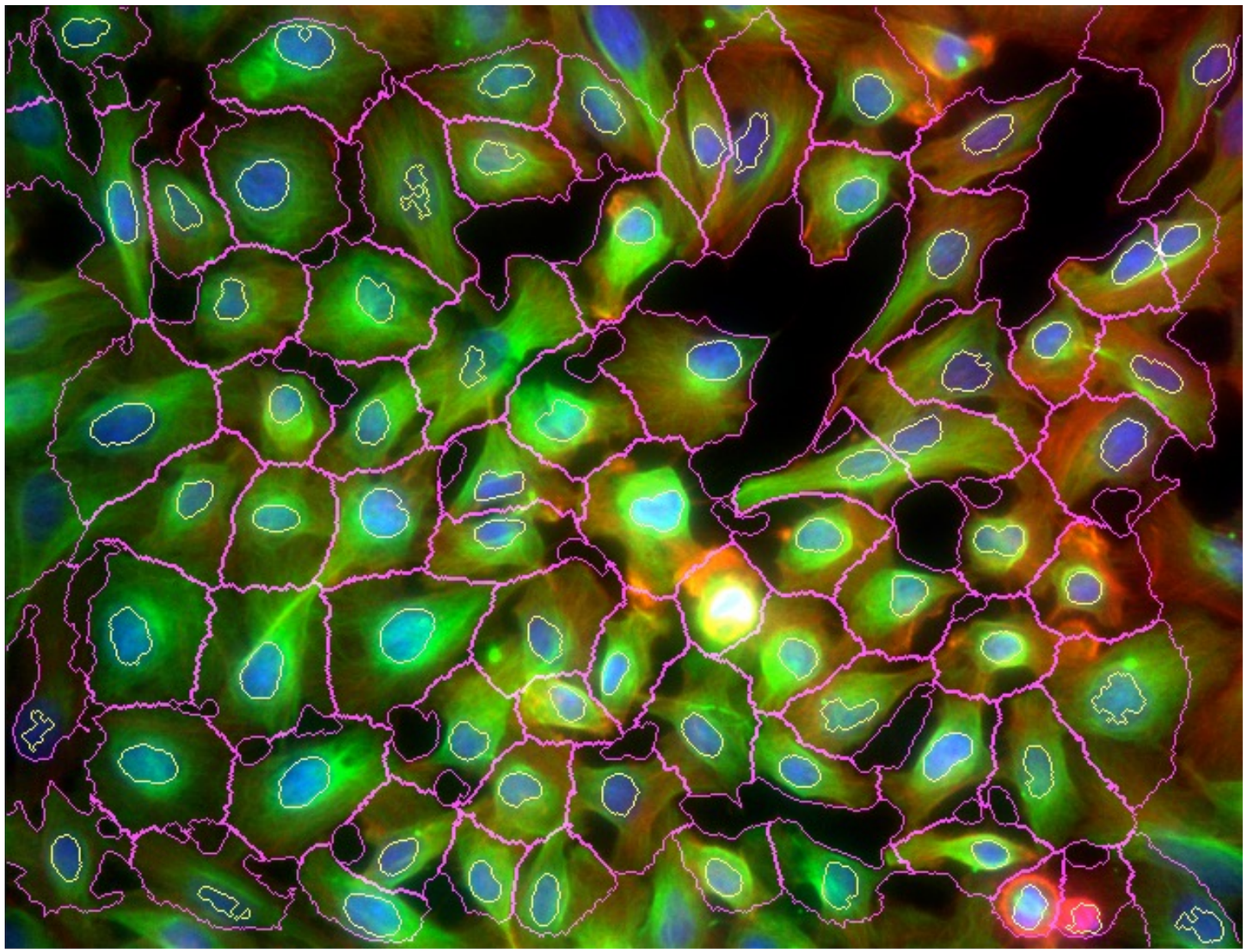


Fuchs, F., Pau, et al. (2010). Clustering phenotype populations by genome-wide RNAi and multiparametric imaging. *Molecular systems biology*, *6*(370), 370.

Slide courtesy by Gregoire Pau

Phenotypic profiles

- Average score in various features.
- Percentage of cells in the different morphological classes
- They form together a vector that describes the phenotype of the cellular population.



n	289
ext	34.33118
ecc	0.472934
Next	2857.356
Nint	485.2710
a2i	0.828876
Next2	0.098647
AF %	0.049594
BC %	0.081746
C %	0.158817
M %	0.179339
LA %	0.009249
P %	0.219697

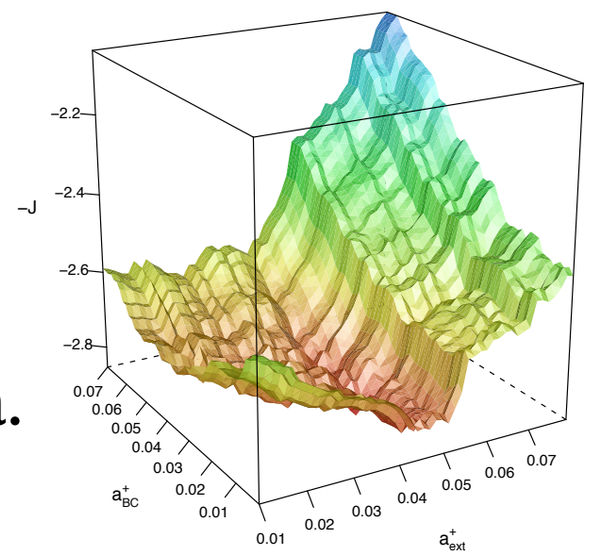
Phenotypic distance learning

An alternative is to learn the distance metric from examples: if we know a priori that certain phenotypes are close, others are far from each other, we might learn parameters of a function which guarantee this relationship. Concretely, we can write the parametrized distance d_θ of x, y:

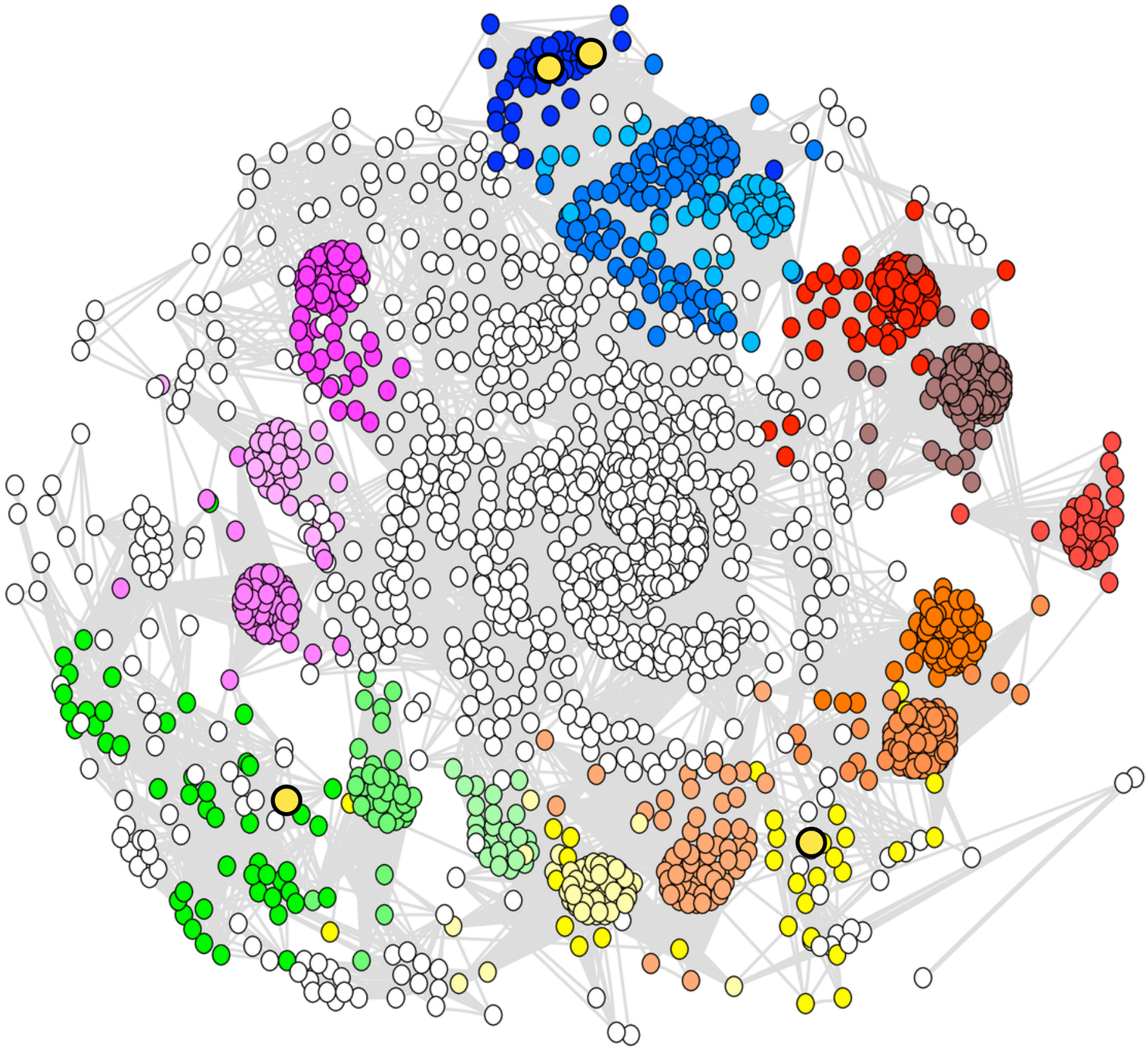
$$d_\theta(x, y) = |f_\theta(x) - f_\theta(y)|$$

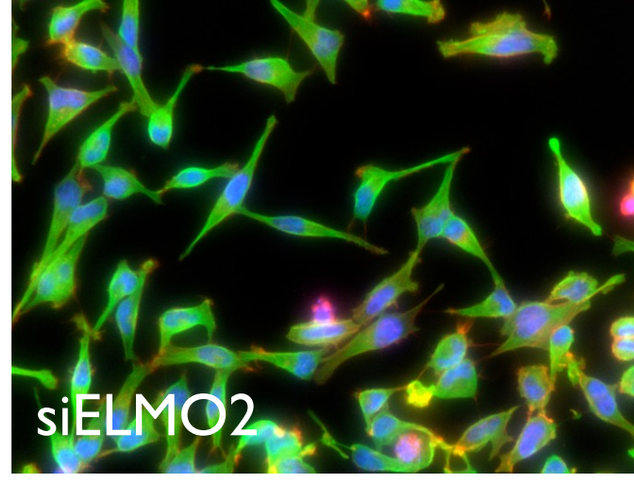
where $f_\theta(\cdot)$ indicates a sigmoidal function with parameters θ. Given a set S of related proteins (for which we assume to observe similar phenotypes) and a set R of unrelated proteins (random sets), we can maximize:

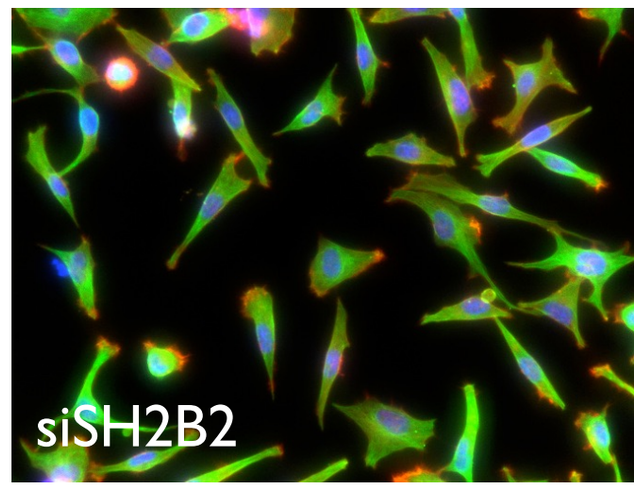
$$J = \frac{\#\{(x, y) \in S \text{ and } d_\theta(x, y) < t\}}{\#\{(x, y) \in R \text{ and } d_\theta(x, y) < t\}}$$

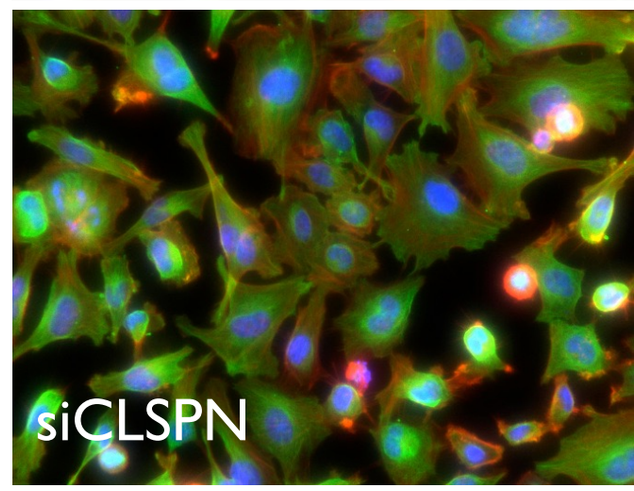


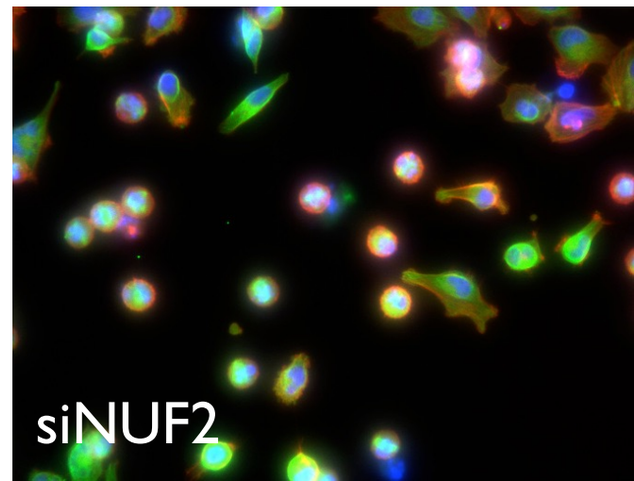
This is a non-convex objective function, and has local maxima.




siELMO2


siSH2B2


siCLSPN


siNUF2