



Machine Learning for Bioinformatics & Systems Biology

3. Feature selection and extraction

Lodewyk Wessels *The Netherlands Cancer Institute*

Marcel Reinders *Delft University of Technology*

Perry Moerland *Amsterdam UMC,
University of Amsterdam*

Some material courtesy of Robert Duin and David Tax

Overview

- Feature extraction
- Feature selection
- Regularized classifiers

Overview

- **Feature extraction**
 - Linear:
 - PCA
 - Fisher
 - Non-linear
 - MDS (Multi-dimensional scaling)

Overview

- **Feature selection**
 - Criteria
 - search algorithms
 - Forward selection
 - Backward selection
 - Branch & Bound search

Overview

- **Regularized classifiers**
 - PAM (Prediction Analysis of Micro-arrays = shrunken centroids)
 - Ridge regression
 - LASSO (Least Absolute Shrinkage and Selection Operator)

Dimensionality reduction

Aim of Feature Extraction and Selection: reduce dimensionality

Dimensionality reduction

Aim of Feature Extraction and Selection: reduce dimensionality

Why is reducing dimensionality useful?

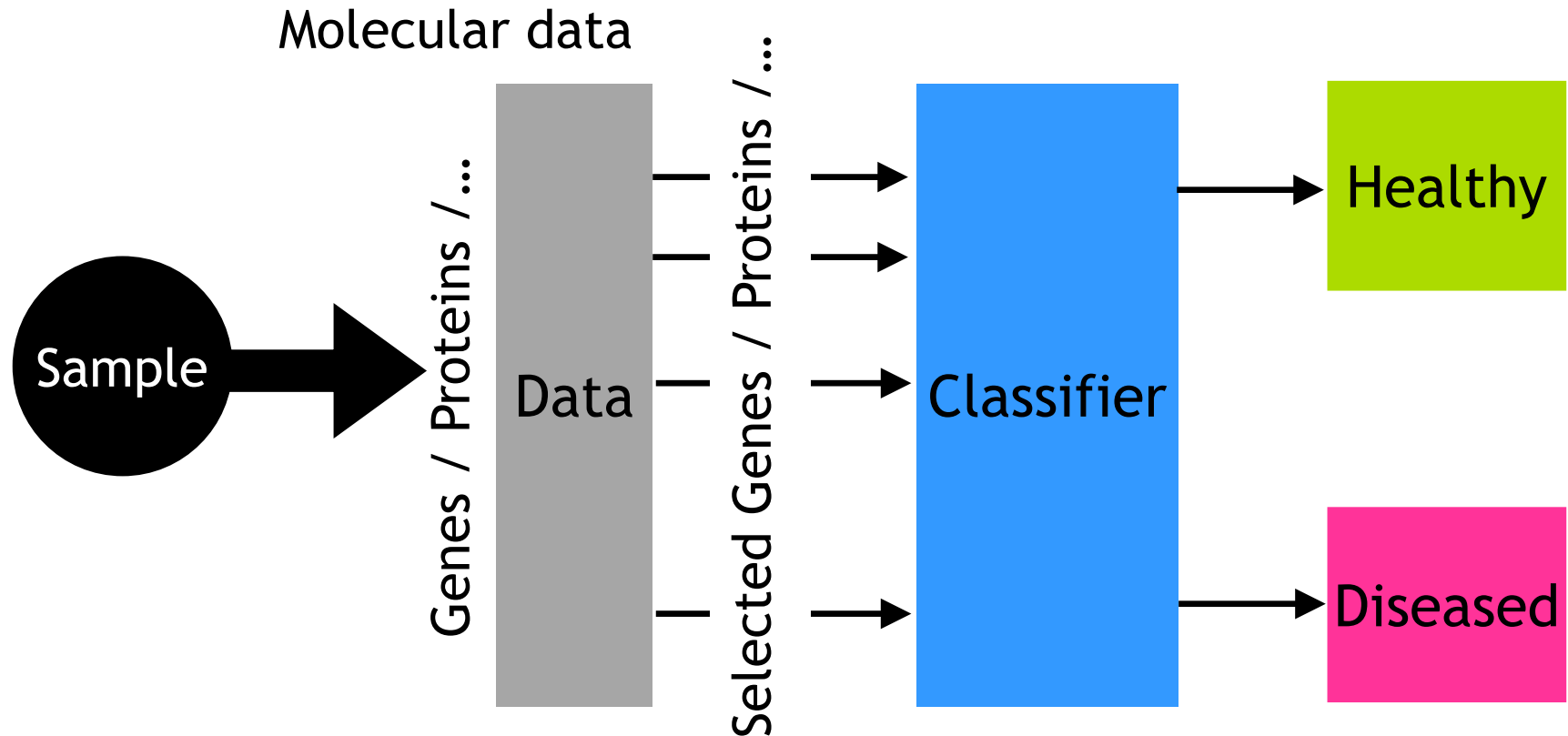
1. **Fewer parameters:** faster, easier to estimate – possibly better performance
2. **Explain** which measurements (genes) are useful and which are not (reduce redundancy)
3. **Visualisation**

Dimensionality reduction

Aim of Feature Extraction and Selection: reduce dimensionality

Why is reducing dimensionality useful?

Example: molecular diagnostic classifiers



Molecular data (e.g. RNAseq data)

- **Curse of dimensionality (# features / # samples):**
 - for **fixed** sample size
 - and **increasing** number of features (number of parameters)
 - performance **decreases**
 - (There are fewer samples per parameter, i.e. worse estimates)

Molecular data (e.g. RNAseq data)

- **Curse of dimensionality (# features / # samples):**
 - for **fixed** sample size
 - and **increasing** number of features (number of parameters)
 - performance **decreases**
 - (There are fewer samples per parameter, i.e. worse estimates)
- **Traditional assumption in pattern recognition:**
 - need 5-10 times as many samples as there are parameters
 - with regularization we can do with fewer

Molecular data (e.g. RNAseq data)

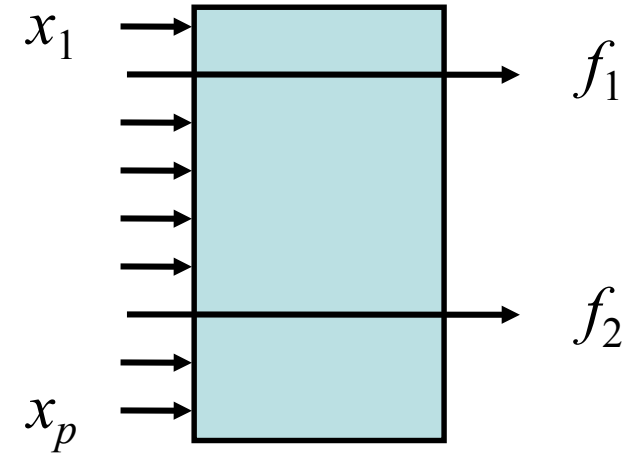
- **Curse of dimensionality (# features / # samples):**
 - for **fixed** sample size
 - and **increasing** number of features (number of parameters)
 - performance **decreases**
 - (There are fewer samples per parameter, i.e. worse estimates)
- **Traditional assumption in pattern recognition:**
 - need 5-10 times as many samples as there are parameters
 - with regularization we can do with fewer
- **But genomic data (e.g. RNAseq) is extreme:**
 - 100-1000 times *fewer* samples than parameters!

Molecular data (e.g. RNAseq data)

- **Curse of dimensionality (# features / # samples):**
 - for **fixed** sample size
 - and **increasing** number of features (number of parameters)
 - performance **decreases**
 - (There are fewer samples per parameter, i.e. worse estimates)
- **Traditional assumption in pattern recognition:**
 - need 5-10 times as many samples as there are parameters
 - with regularization we can do with fewer
- **But genomic data (e.g. RNAseq) is extreme:**
 - 100-1000 times *fewer* samples than parameters!
- **For example: nearest mean classifier on Golub data**
 - $p = 3051, k = 2 \rightarrow$ number of parameters = 6102
 - Number of samples, $n = 38$

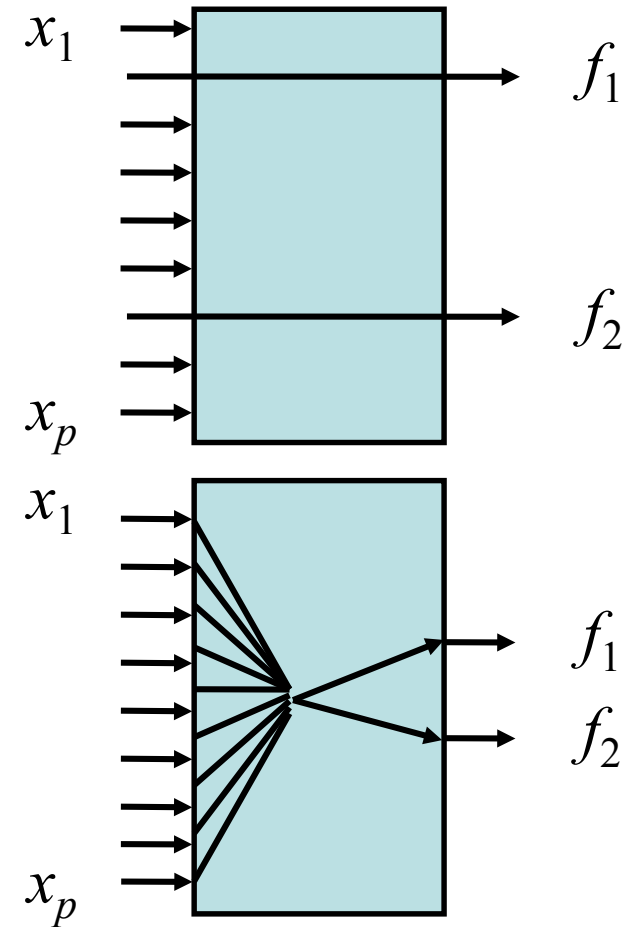
Feature selection vs. extraction

- **Feature selection:**
select d out of p measurements



Feature selection vs. extraction

- **Feature selection:**
select d out of p measurements
- **Feature extraction:**
map p measurements
to d measurements
(e.g. PCA, CCA, LDA, MDS)



Feature selection v extraction (2)

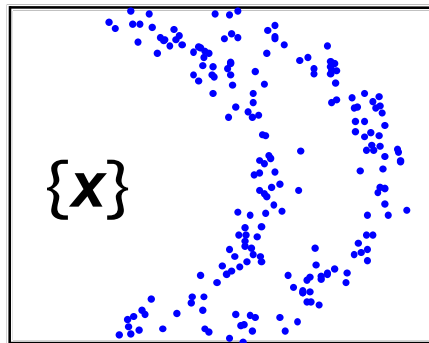
	Advantage	Disadvantage
Selection	cut in measurements easy interpretation	expensive often approximate

Feature selection v extraction (2)

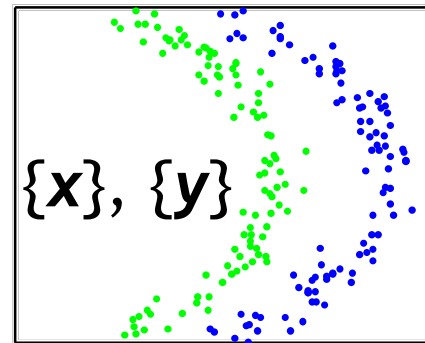
	Advantage	Disadvantage
Selection	cut in measurements easy interpretation	expensive often approximate
Extraction	cheap can be nonlinear not axis aligned	need all measurements criterion sub-optimal

Feature extraction (1)

Main types: supervised/unsupervised



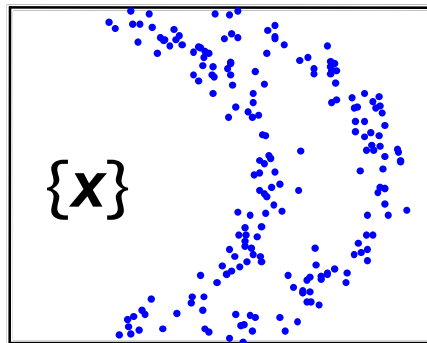
Unsupervised



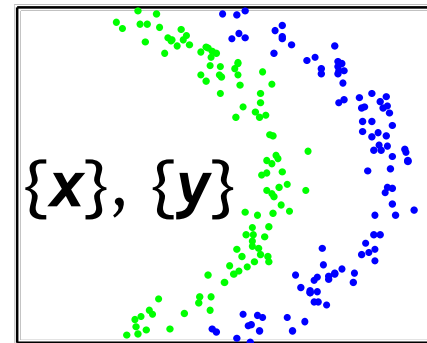
Supervised

Feature extraction (1)

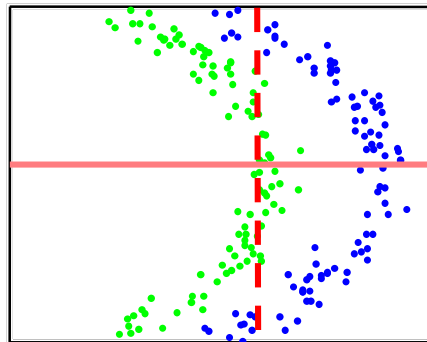
Main types: Linear / Nonlinear



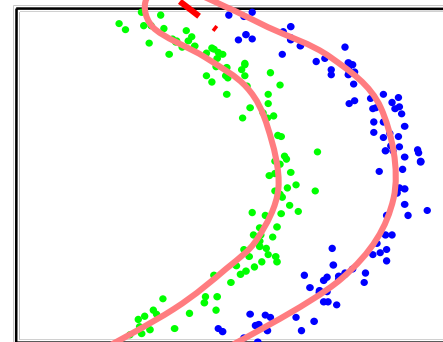
Unsupervised



Supervised



Linear



Nonlinear

Feature extraction (2)

- **Linear, unsupervised:**
 - Principal Component Analysis (PCA)
- **Linear, supervised:**
 - Linear Discriminant Analysis (LDA)

Principal component analysis

(Unsupervised feature extraction)

- **Principal component analysis (PCA, 1901):**
Goal: find directions in data...

Principal component analysis

(Unsupervised feature extraction)

- **Principal component analysis (PCA, 1901):**
Goal: find directions in data...
 - which retain as much *variation* as possible

Principal component analysis

(Unsupervised feature extraction)

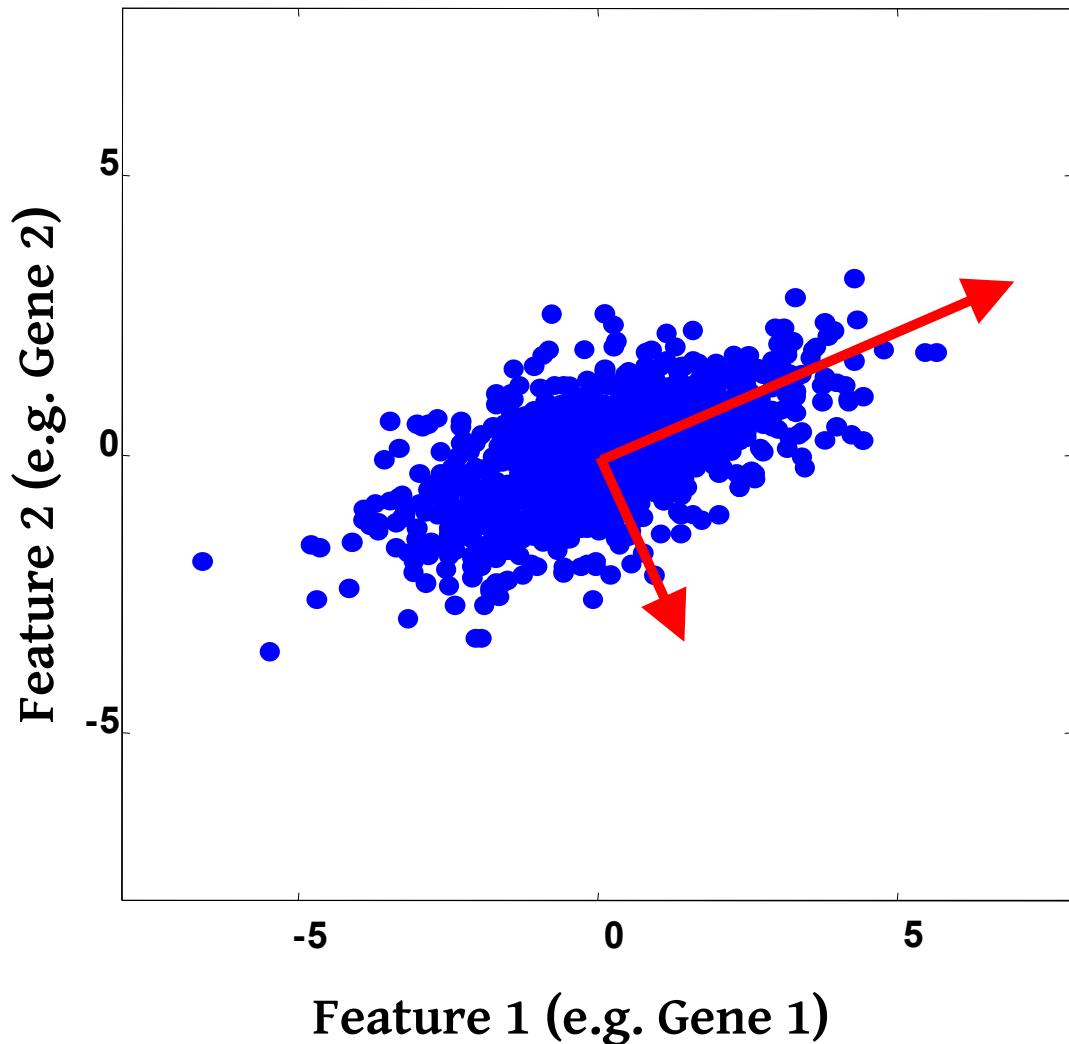
- **Principal component analysis (PCA, 1901):**
Goal: find directions in data...
 - which retain as much *variation* as possible
 - which make projected data *uncorrelated*

Principal component analysis

(Unsupervised feature extraction)

- **Principal component analysis (PCA, 1901):**
Goal: find directions in data...
 - which retain as much *variation* as possible
 - which make projected data *uncorrelated*
 - which minimise squared *reconstruction error*

Principal component analysis (Unsupervised feature extraction)



Steps:

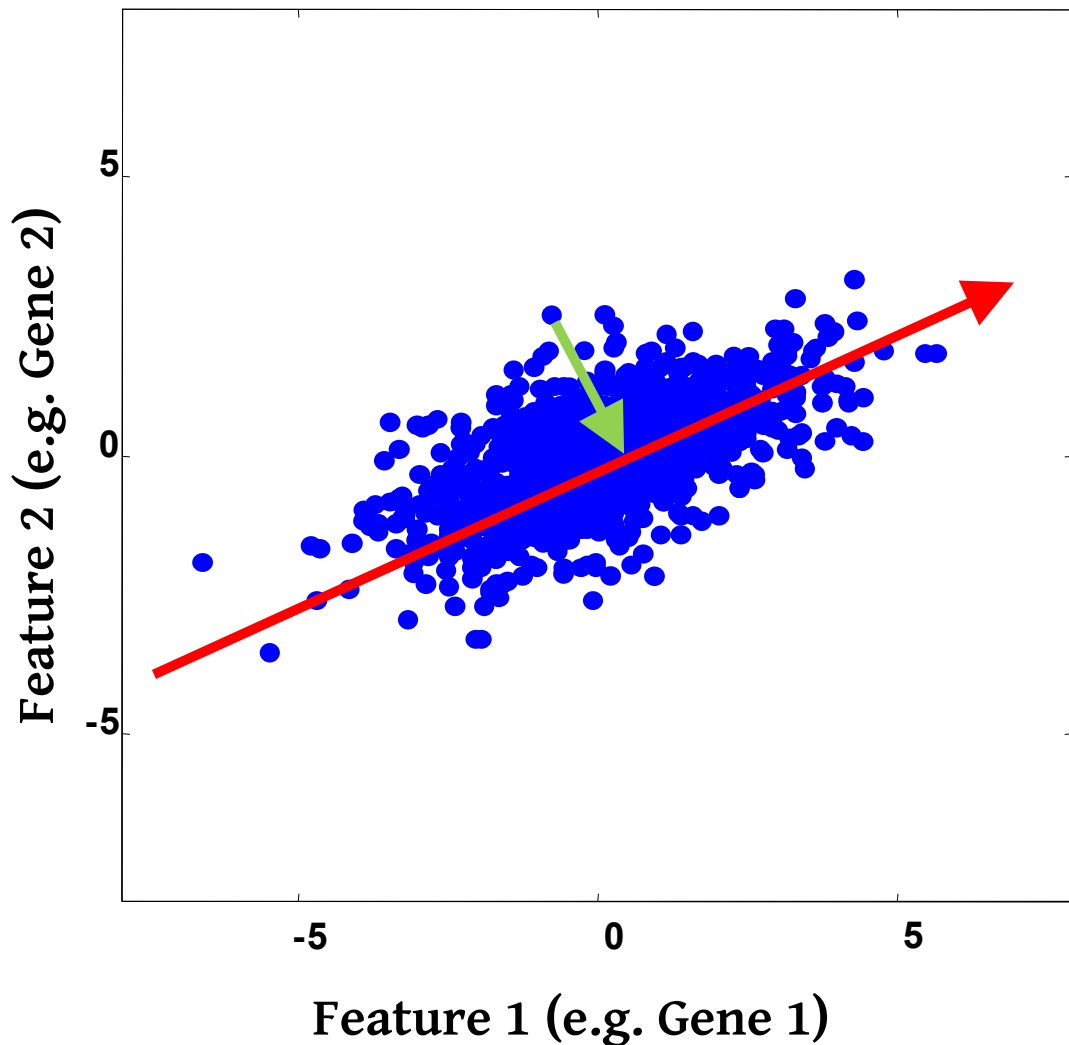
1. Center data
2. Compute covariance, C
3. Perform PCA on C

Output:

1. Eigenvectors: $\mathbf{e}_1, \mathbf{e}_2$
2. Eigenvalues: λ_1, λ_2

Reducing dimensions:
Choosing 'd'

Principal component analysis (Unsupervised feature extraction)



Steps:

1. Center data
2. Compute covariance, C
3. Perform PCA on C

Output:

1. Eigenvectors: $\mathbf{e}_1, \mathbf{e}_2$
2. Eigenvalues: λ_1, λ_2

Reducing dimensions:

1. Choosing $d = 1$
2. *Project data on \mathbf{e}_1*

Choosing reduced dimensionality

- To choose d inspect the retained variance,

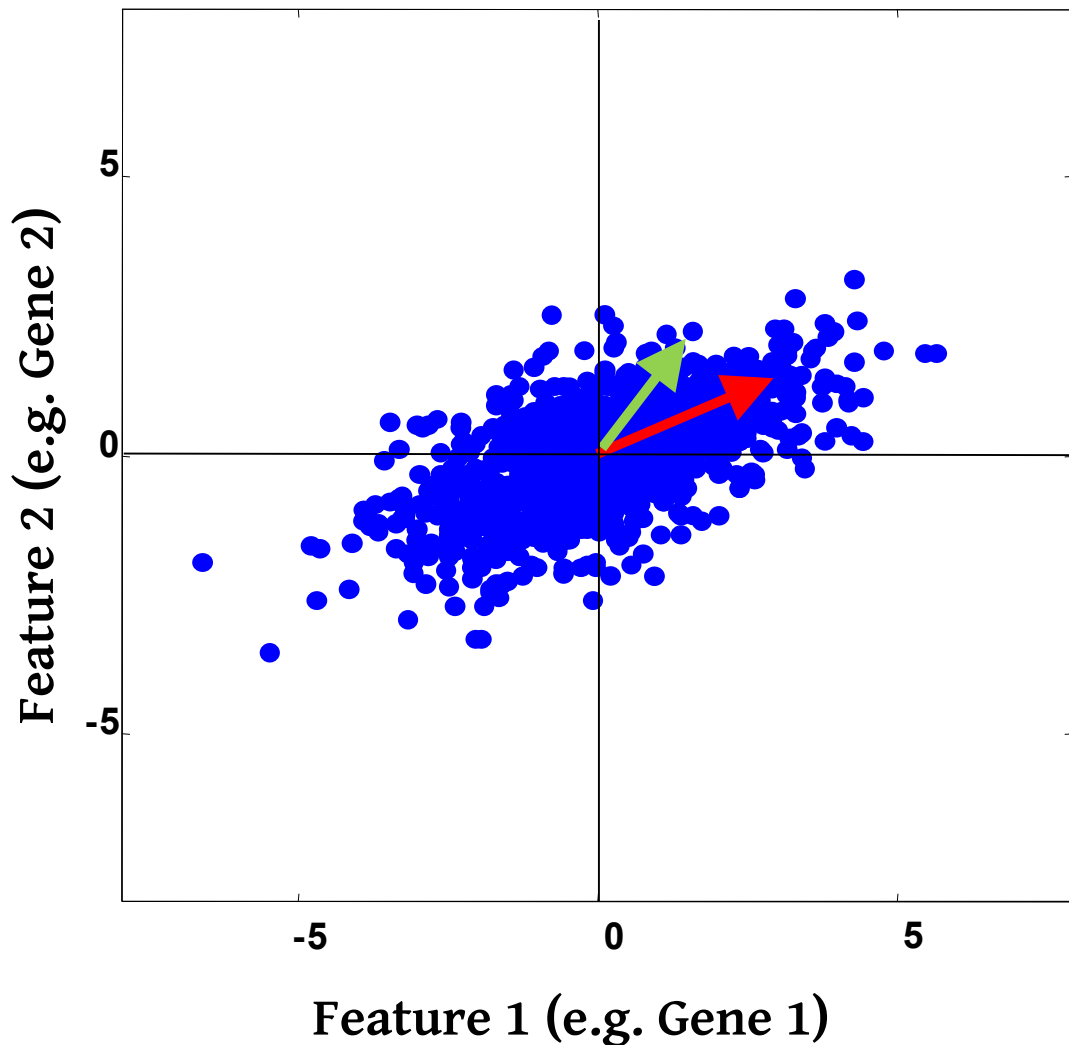
$$\sum_{i=1}^d \lambda_i$$

- or the ratio of retained variance,

$$\sum_{i=1}^d \lambda_i / \sum_{j=1}^p \lambda_j$$

- Rule of thumb: Select d for which 80-90% variance is retained
- Reduced dimensionality data set
 - $[\mathbf{x}_1^T; \mathbf{x}_2^T; \dots; \mathbf{x}_2^T][\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_d]$

Principal component analysis (Unsupervised feature extraction)



Steps:

1. Center data
2. Compute covariance, C
3. Perform PCA on C

Output:

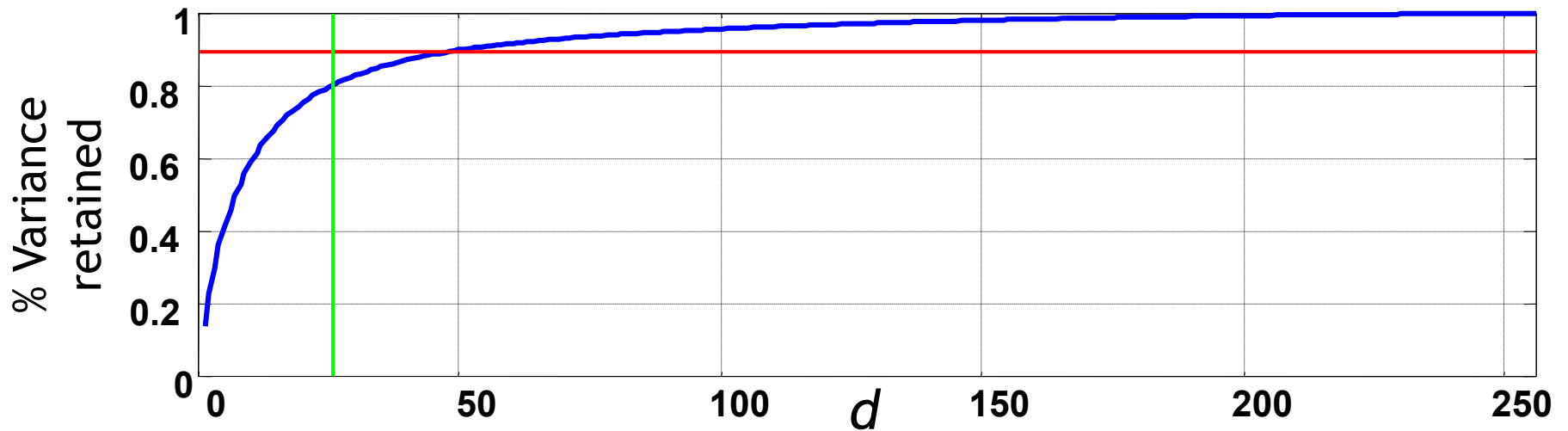
1. Eigenvectors: $\mathbf{e}_1, \mathbf{e}_2$
2. Eigenvalues: λ_1, λ_2

Reducing dimensions:

1. Choosing $d = 1$
2. *Project data on \mathbf{e}_1*

PCA example

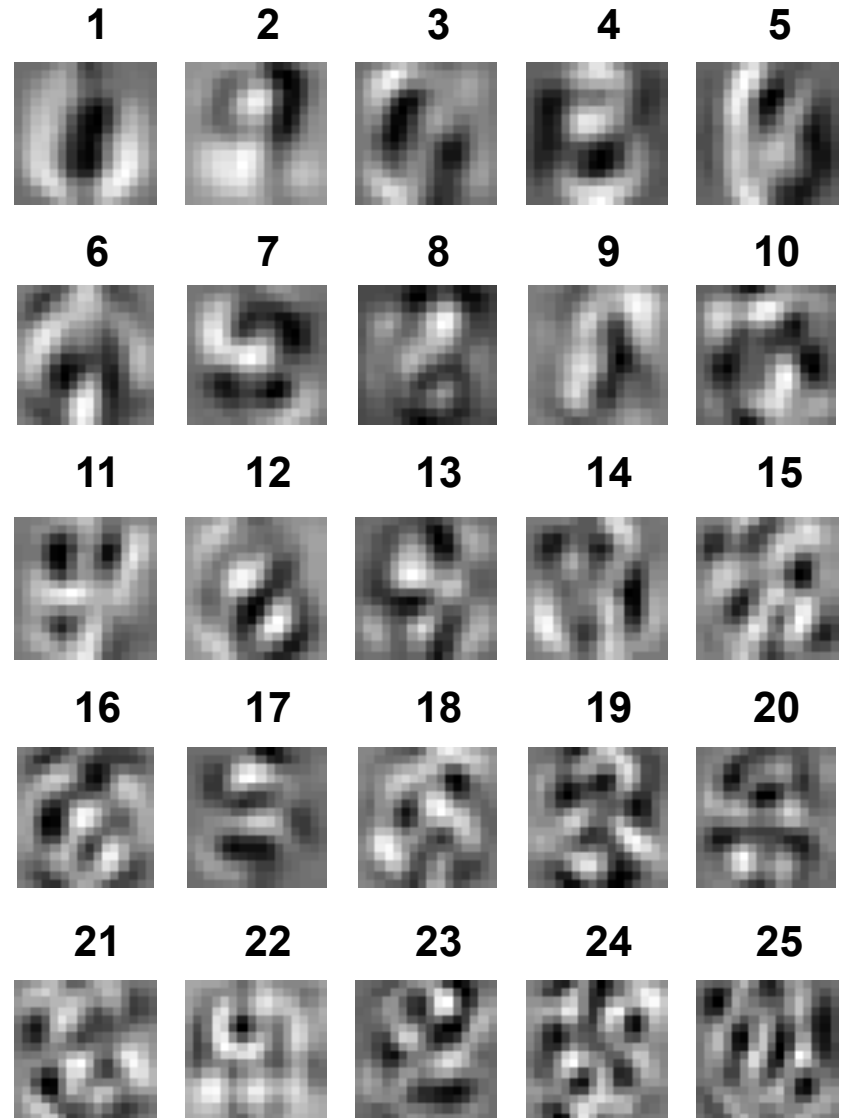
- *e.g.* NIST digits: 2000 samples, $p = 256$ (16 X 16)



PCA example (2)

- For image data, principal components can also be interpreted...

most often occurring variations between digits



PCA tips

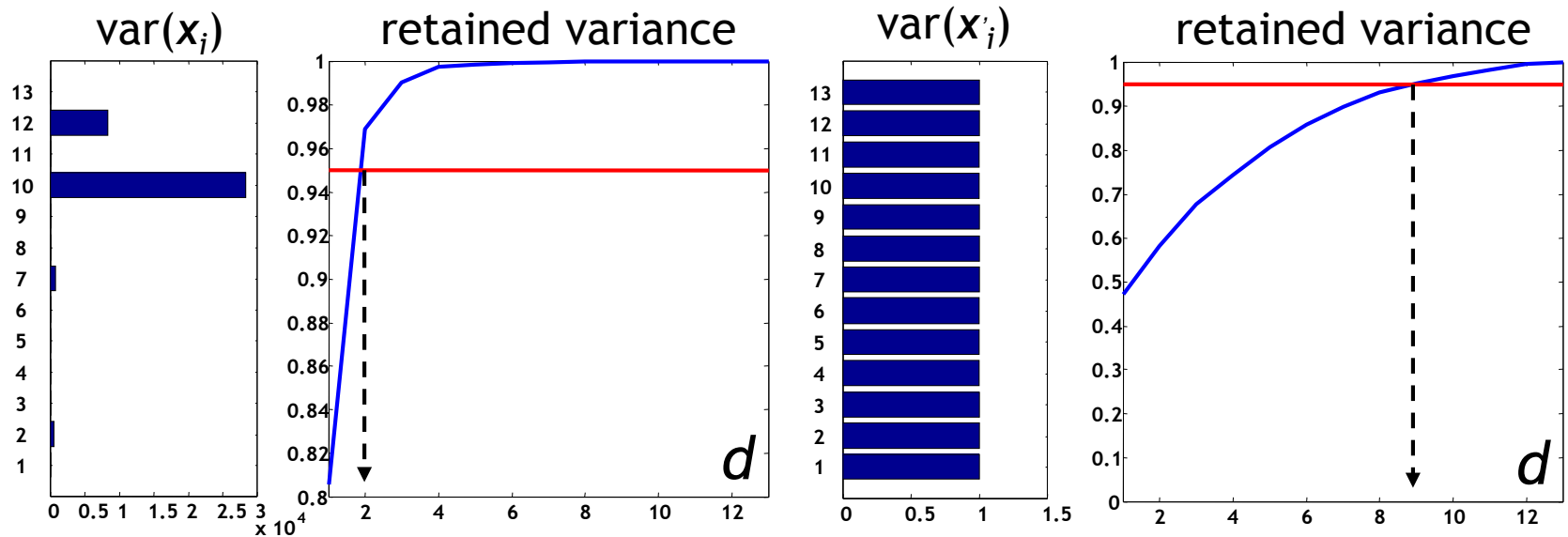
- Ensure data is centered (mean of each feature is zero):
 $x' \leftarrow (x - \mu)$

PCA tips

- Ensure data is centered (mean of each feature is zero):
 $\mathbf{x}' \leftarrow (\mathbf{x} - \boldsymbol{\mu})$
- PCA is sensitive to scaling
 - length in cm has a much larger variance than length in m
 - best to standardise: $\mathbf{x}' \leftarrow (\mathbf{x} - \boldsymbol{\mu}) / \boldsymbol{\sigma}$

PCA tips

- Ensure data is centered (mean of each feature is zero):
 $\mathbf{x}' \leftarrow (\mathbf{x} - \boldsymbol{\mu})$
- PCA is sensitive to scaling
 - length in cm has a much larger variance than length in m
 - best to standardise: $\mathbf{x}' \leftarrow (\mathbf{x} - \boldsymbol{\mu}) / \sigma$

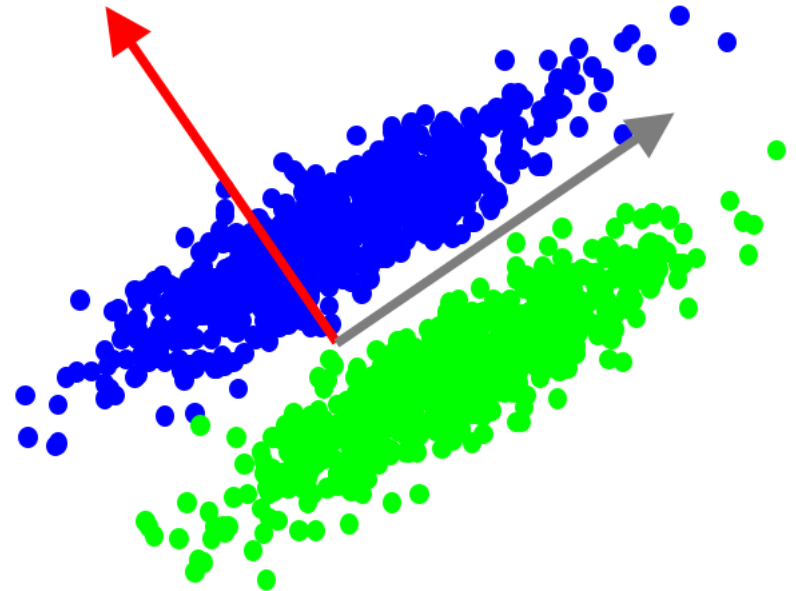


PCA conclusions

- PCA:
 - Is **global** and **linear**
 - Is **unsupervised** (but we can do PCA on each class)
 - Needs a **lot of data** to estimate Σ well.

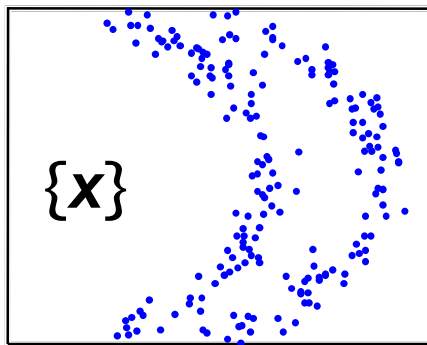
PCA conclusions

- PCA:
 - Is **global** and **linear**
 - Is **unsupervised** (but we can do PCA on each class)
 - Needs a **lot of data** to estimate Σ well.
- Danger:
 - Criterion is not necessarily related to the goal;
 - Might discard important directions

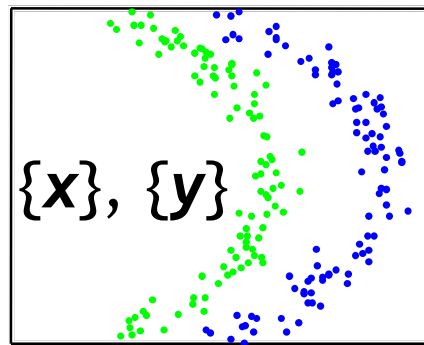


Supervised, linear feature extraction

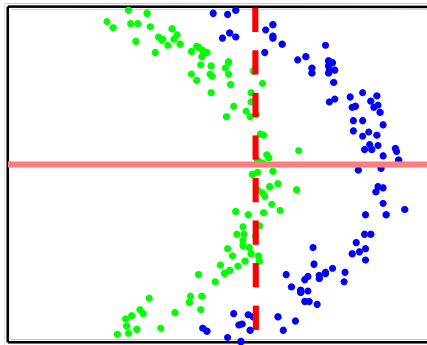
- If class label ω (or y) is given, supervised extraction
- Examples: Fisher mapping; Linear Discriminant Analysis (Day 2)



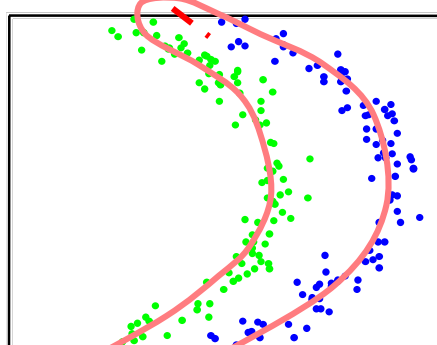
Unsupervised



Supervised

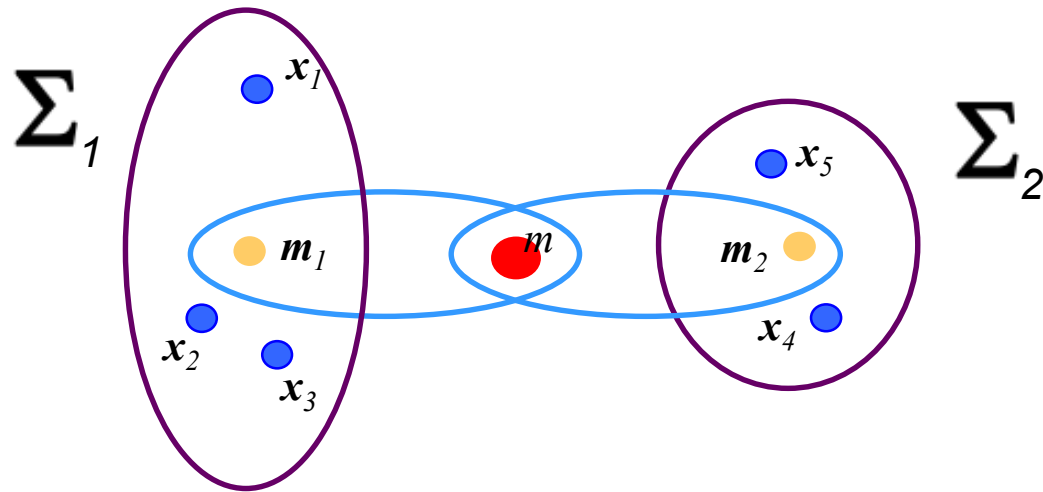


Linear



Nonlinear

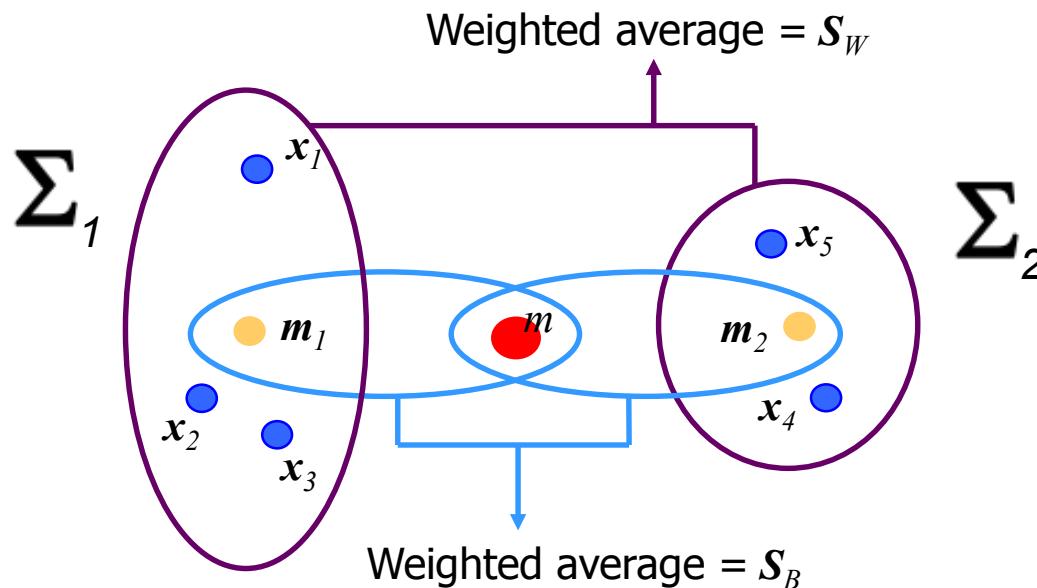
Supervised feature extraction



Supervised feature extraction

Within-class and between-class scatter matrices:

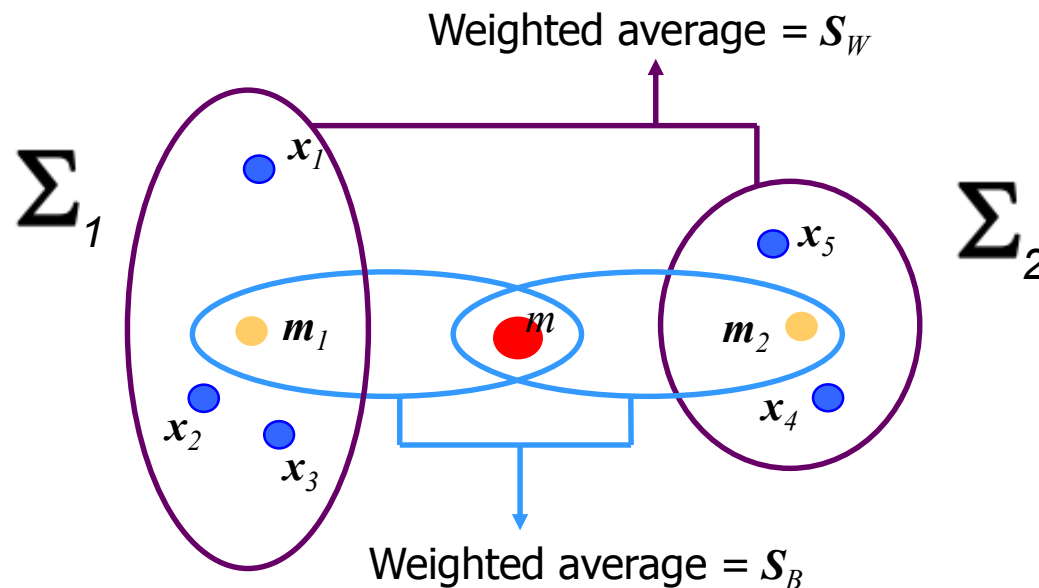
- Within-class:
$$S_w = \sum_{i=1}^C \frac{n_i}{n} \Sigma_i$$



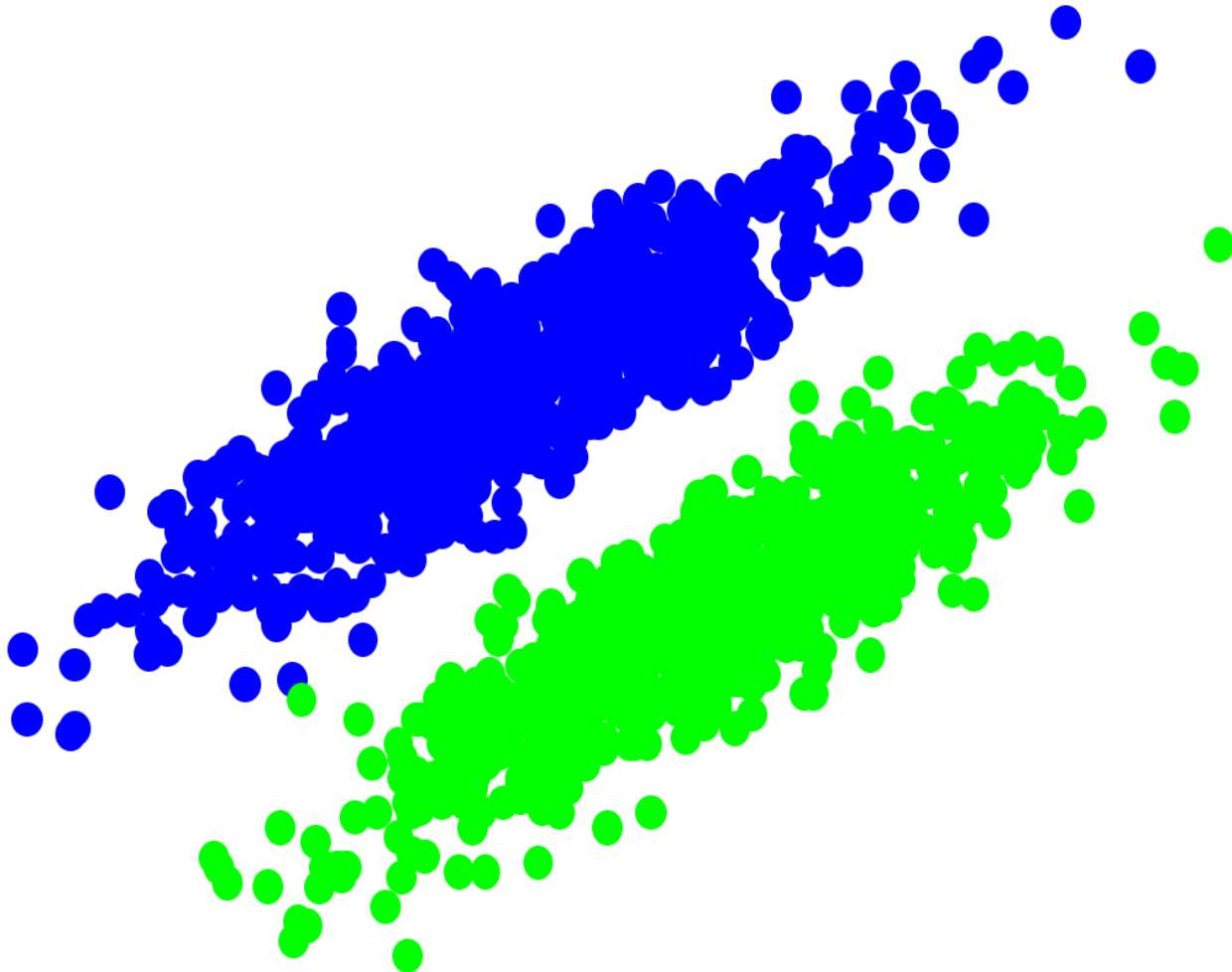
Supervised feature extraction

Within-class and between-class scatter matrices:

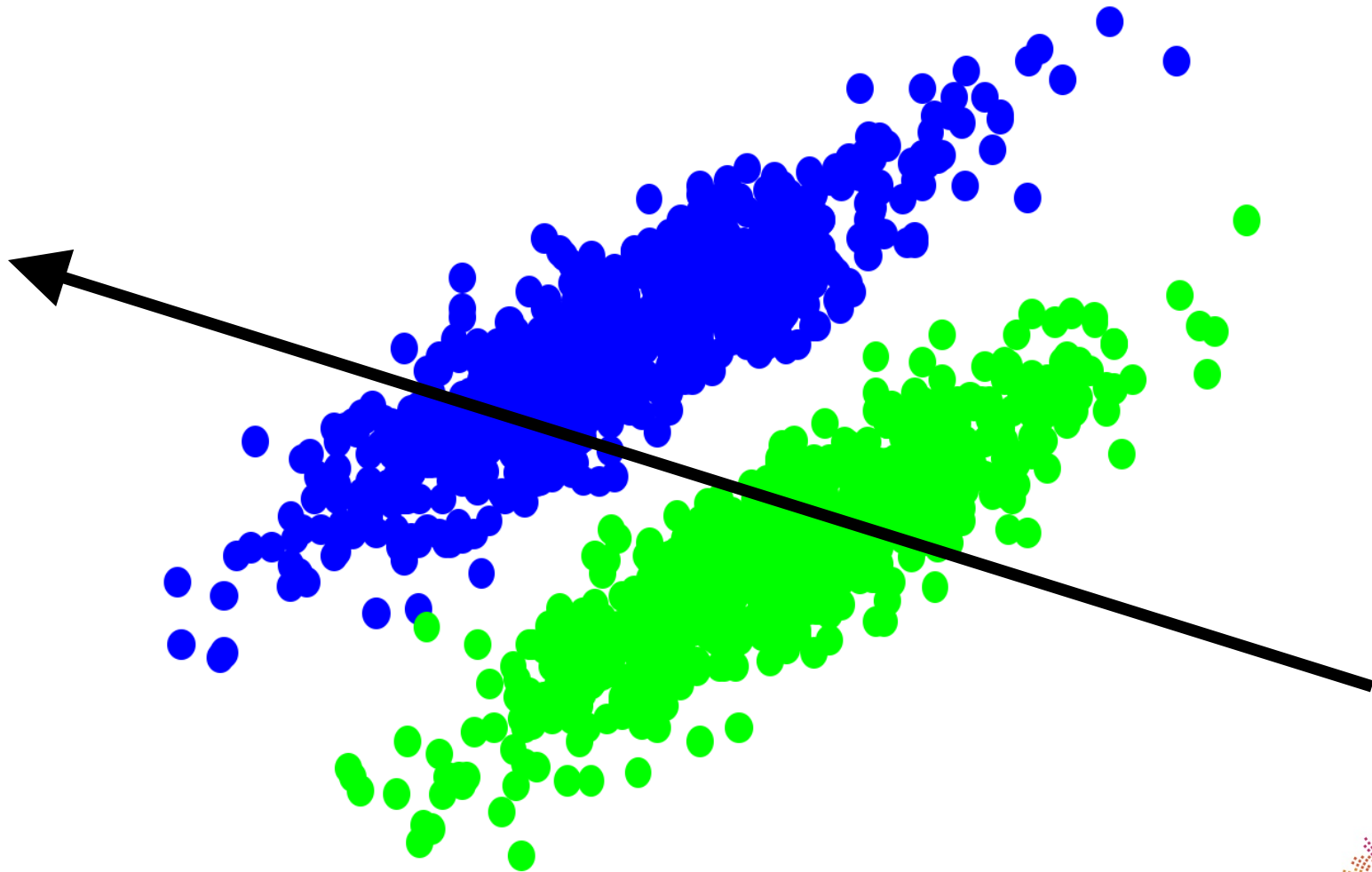
- Within-class:
$$\mathbf{S}_w = \sum_{i=1}^C \frac{n_i}{n} \mathbf{\Sigma}_i$$
- Between-class:
$$\mathbf{S}_B = \sum_{i=1}^C \frac{n_i}{n} (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$$



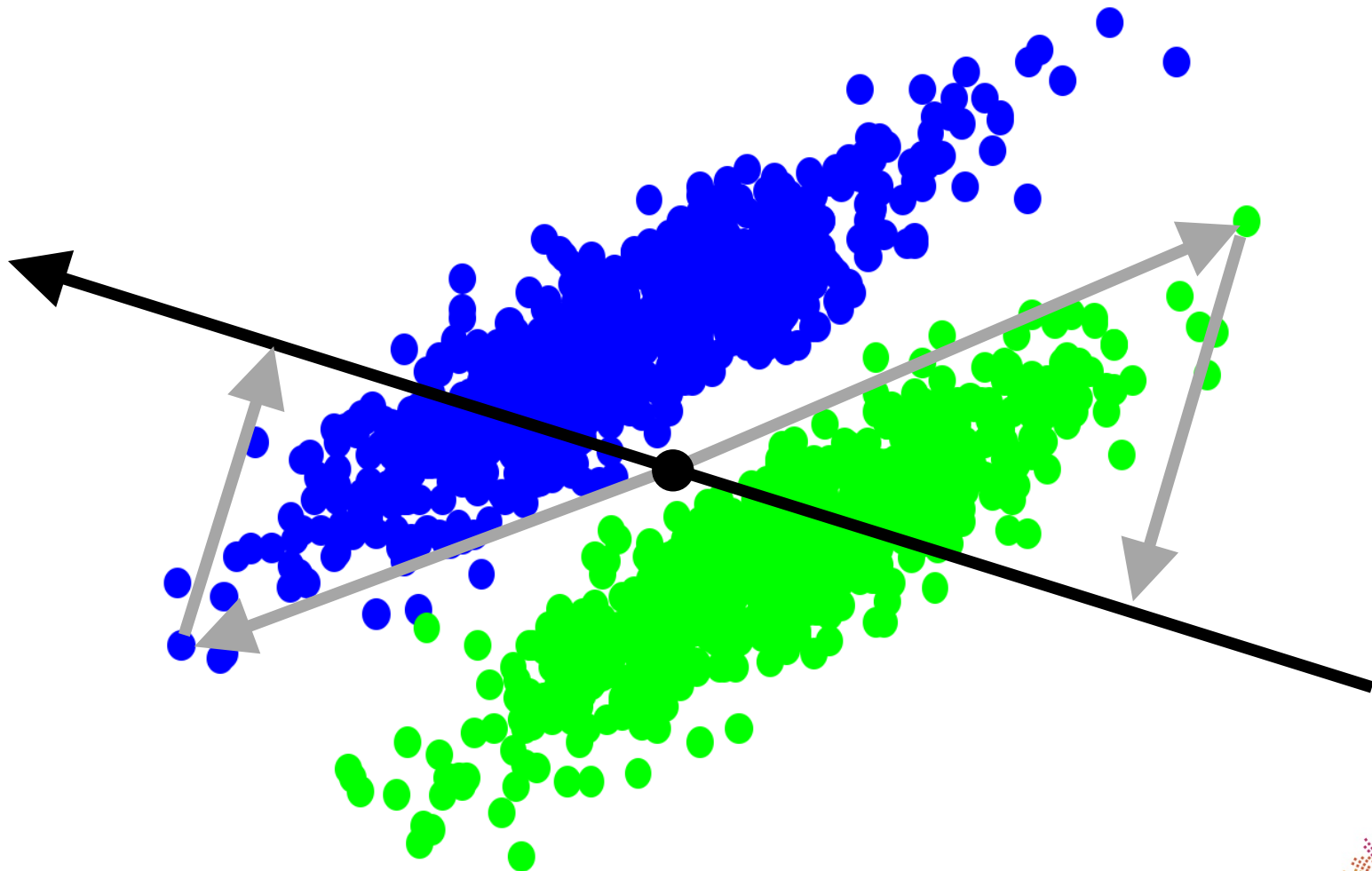
Fisher mapping: finding the direction (subspace) to project onto for the best class separation



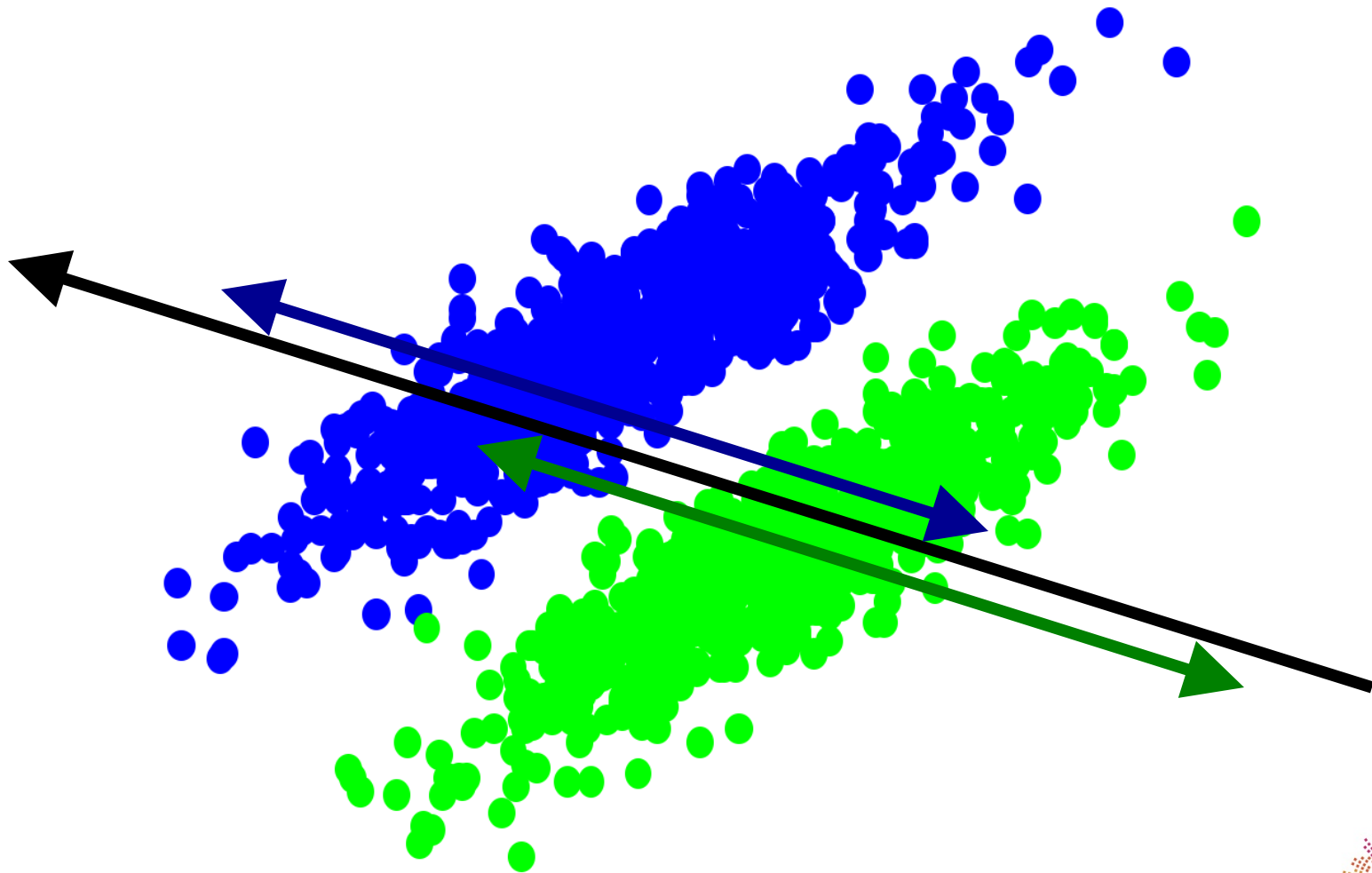
Fisher mapping: defining the Fisher criterion



Fisher mapping: defining the Fisher criterion

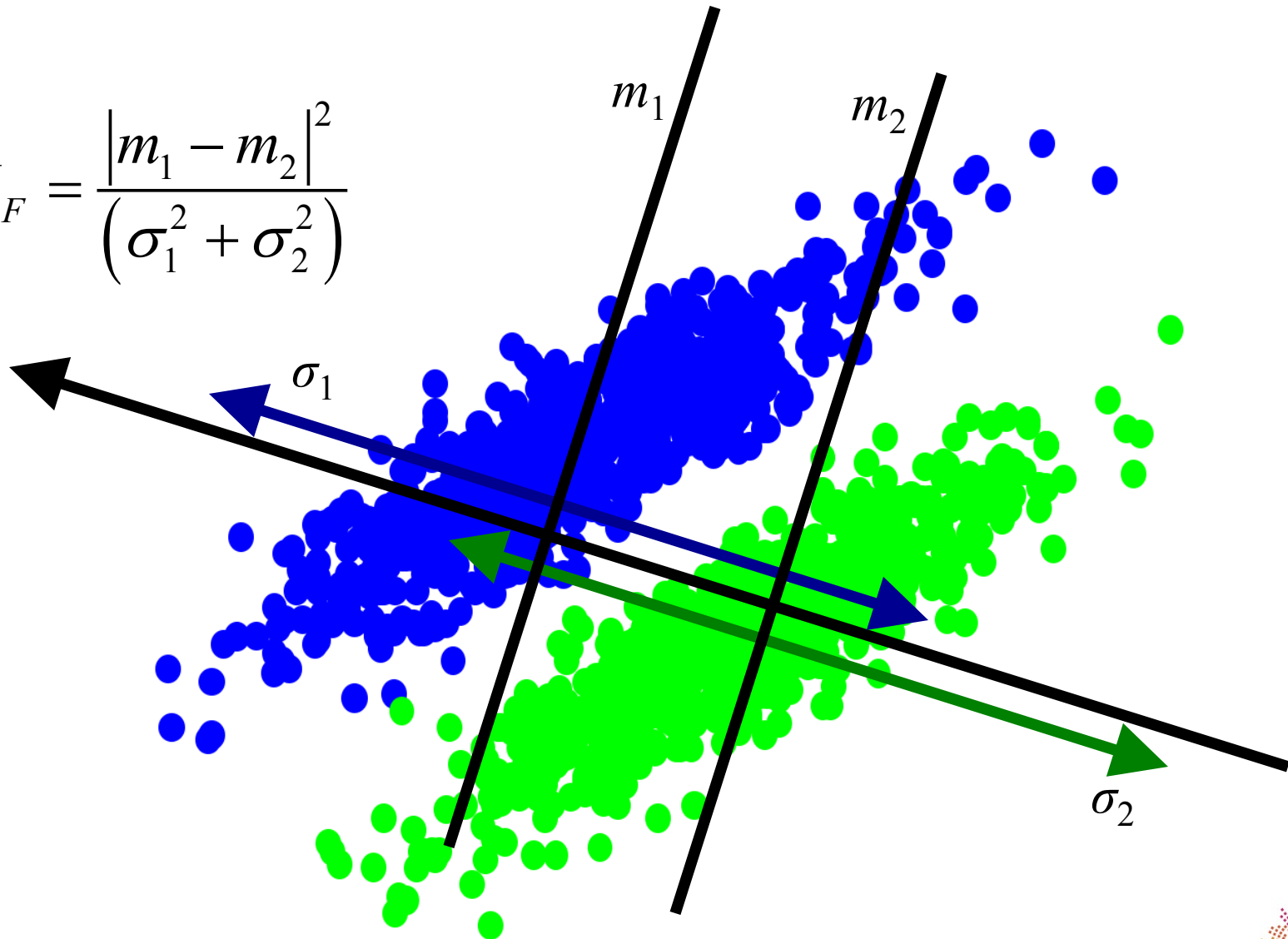


Fisher mapping: defining the Fisher criterion



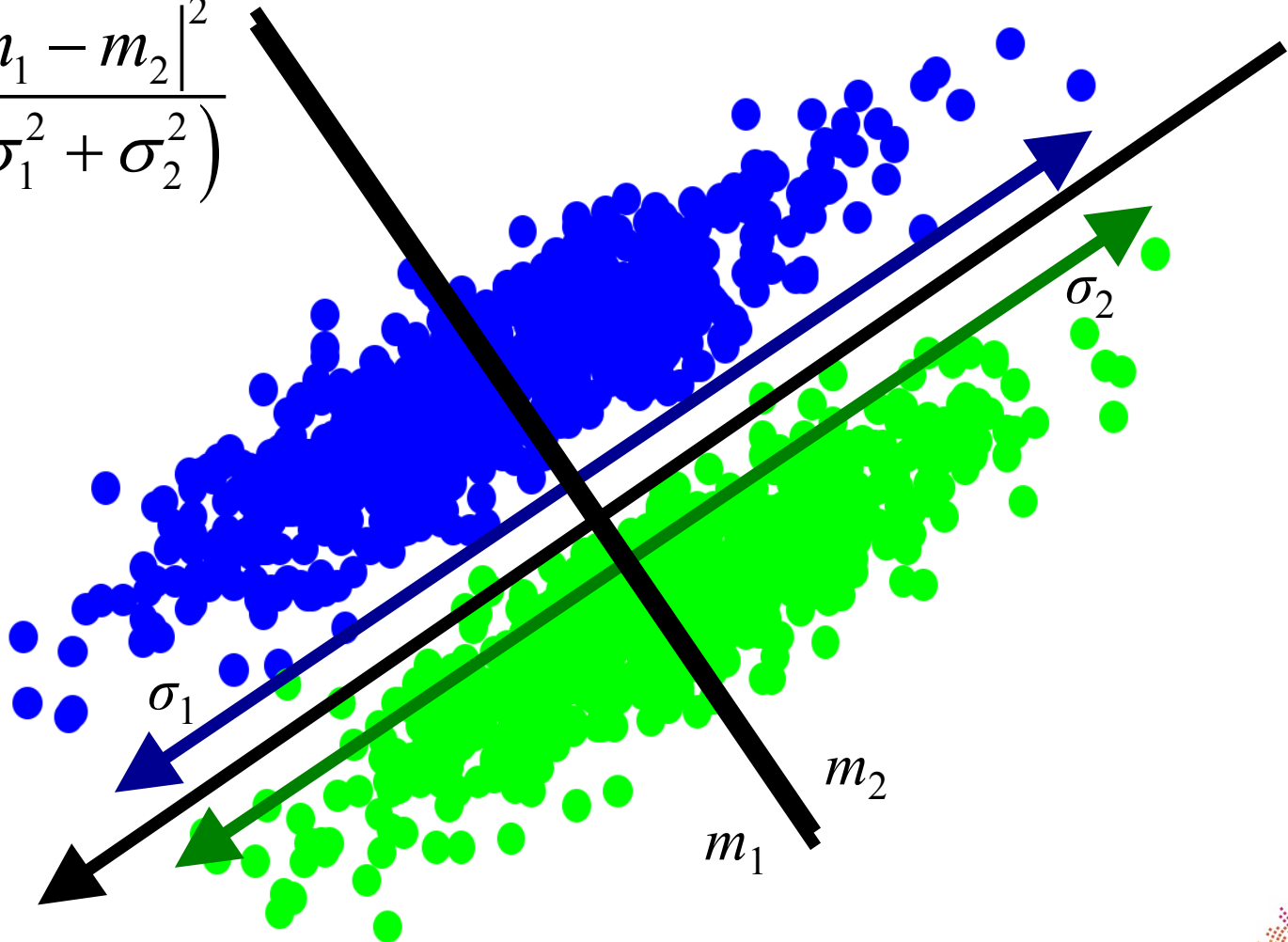
Fisher mapping (Fisher criterion)

$$J_F = \frac{|m_1 - m_2|^2}{(\sigma_1^2 + \sigma_2^2)}$$



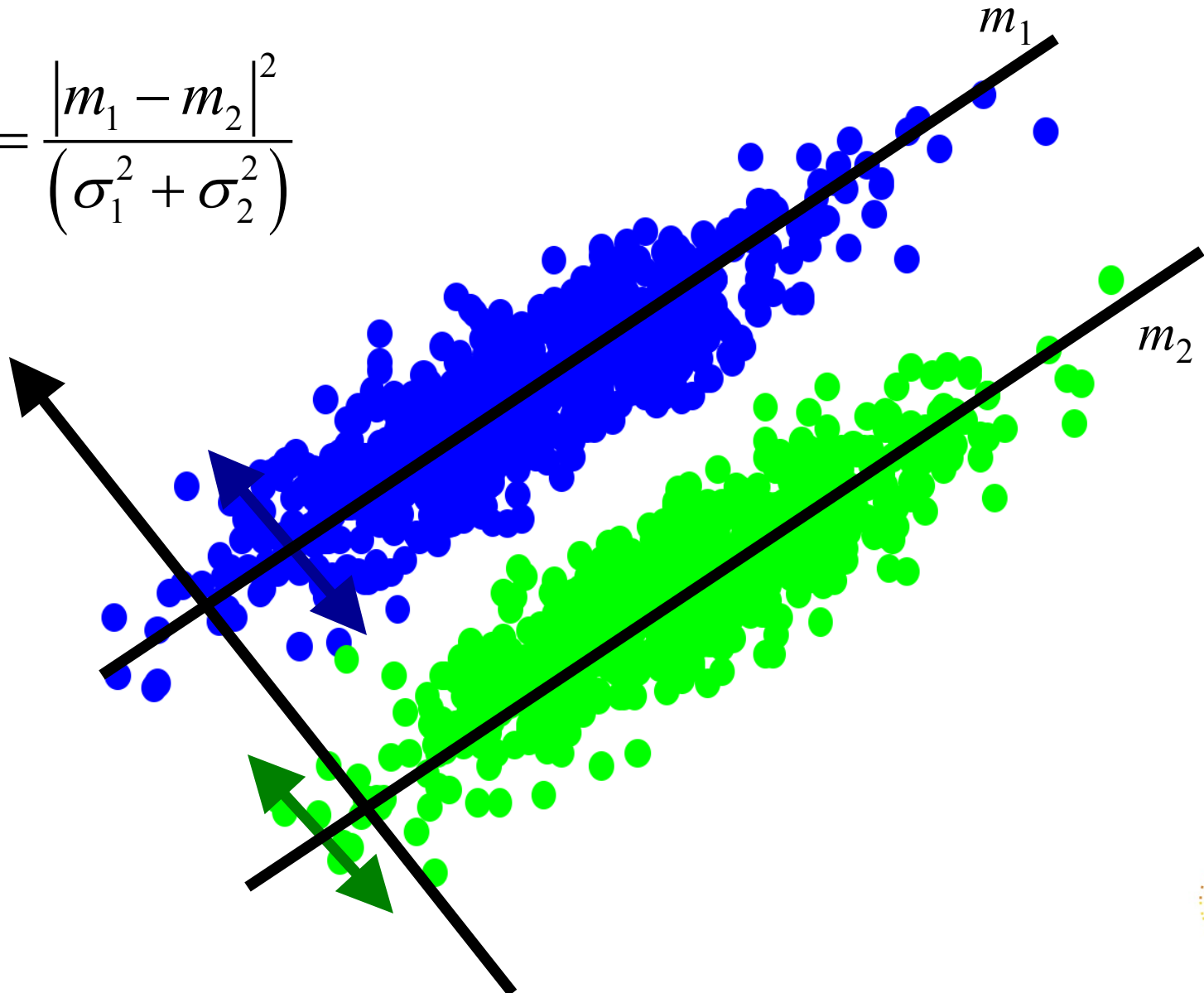
Fisher mapping: defining the Fisher criterion

$$J_F = \frac{|m_1 - m_2|^2}{(\sigma_1^2 + \sigma_2^2)}$$



Fisher mapping: defining the Fisher criterion

$$J_F = \frac{|m_1 - m_2|^2}{(\sigma_1^2 + \sigma_2^2)}$$



Fisher mapping

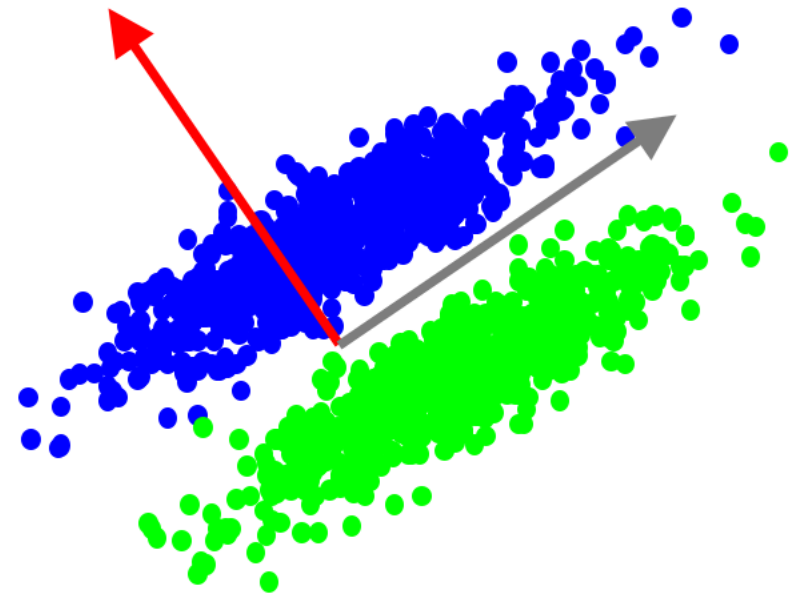
- Find basis vector \mathbf{a}_1 for $\{\mathbf{x}\}$ such that in the projections, the classes are maximally separated
- Choose \mathbf{a}_1 to maximise Fisher criterion:

$$J_F(\mathbf{a}_1) = \frac{\mathbf{a}_1^T \mathbf{S}_B \mathbf{a}_1}{\mathbf{a}_1^T \mathbf{S}_W \mathbf{a}_1}$$

- Maximize between class variance
Minimize within class variance

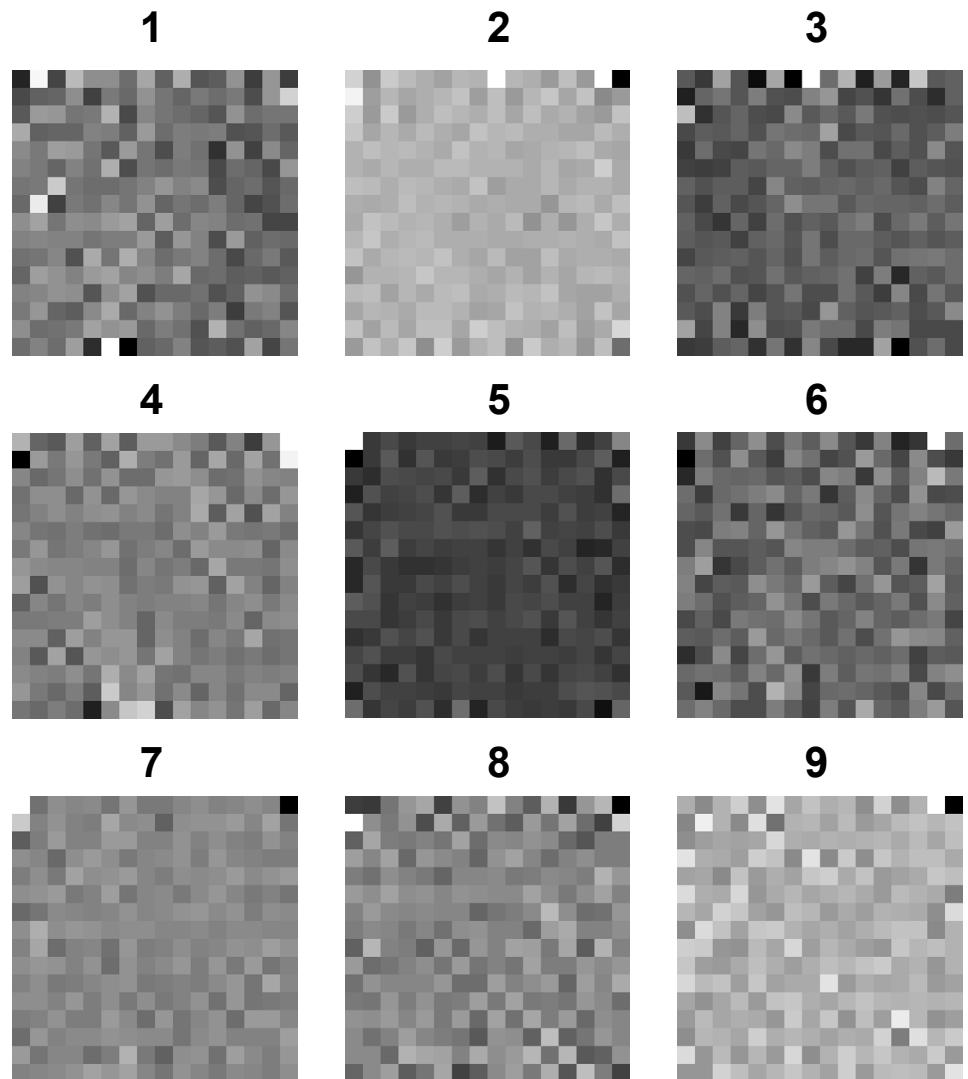
- Solution:

- eigen-analysis on $\mathbf{S}_W^{-1} \mathbf{S}_B$
- select $c-1$ (# classes - 1) dimensions for final classifier
(allows linear separation of classes)



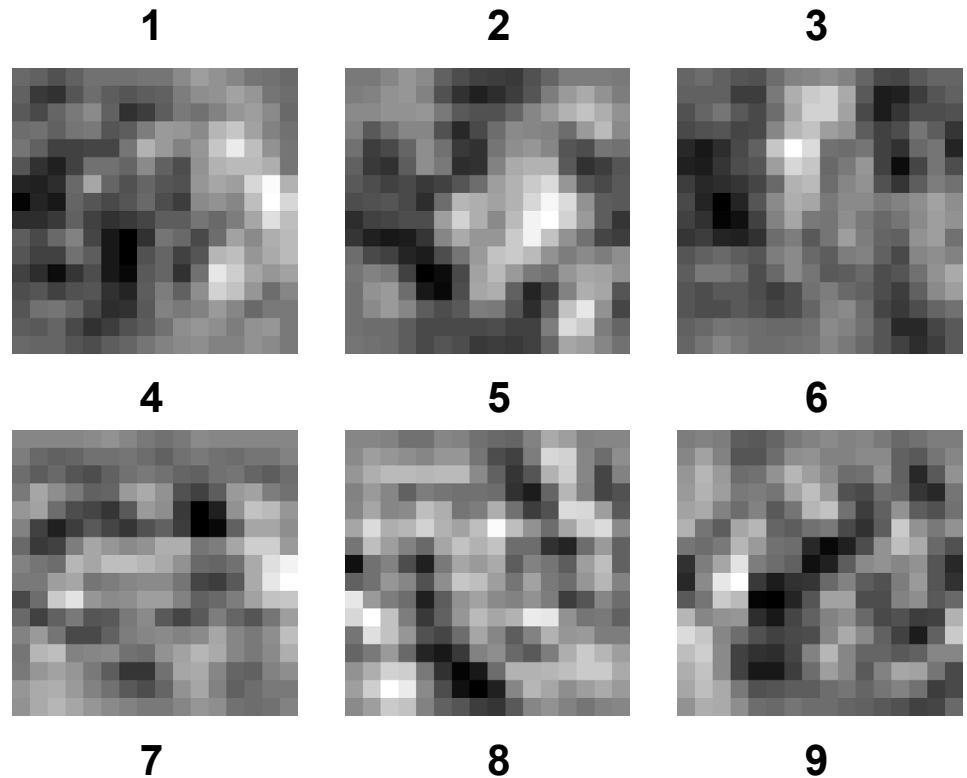
Fisher mapping (2)

- Map down to a maximum of $c - 1$ dimensions
- Example: NIST digits



Fisher mapping (3)

- To avoid fitting noise, can do PCA first
- If system underdetermined ($n \leq p$), first doing PCA is required, otherwise matrix inversion results in singularity

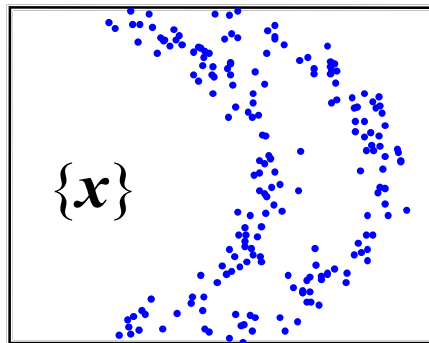


Summary

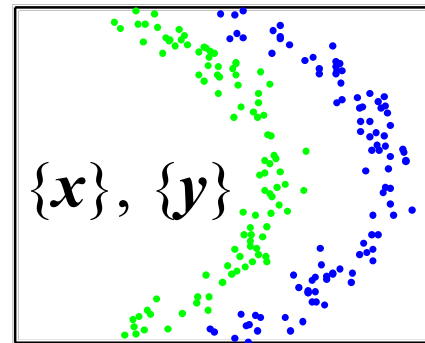
- Discussed:
 - Linear feature extraction
 - Unsupervised: Principal Component Analysis (PCA)
 - Supervised: Fisher mapping

Feature extraction

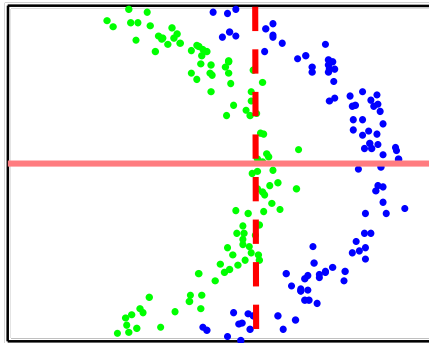
- Main types:



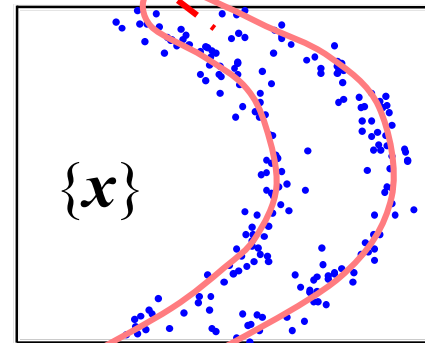
Unsupervised



Supervised



Linear



Nonlinear, unsupervised

Nonlinear, unsupervised feature extraction

- **Multidimensional scaling (MDS):**
 - Nonlinear:
 - Sammon mapping
 - t-SNE

Nonlinear feature extraction

Example: embedding

- Find new representation such that distances between samples are preserved as well as possible

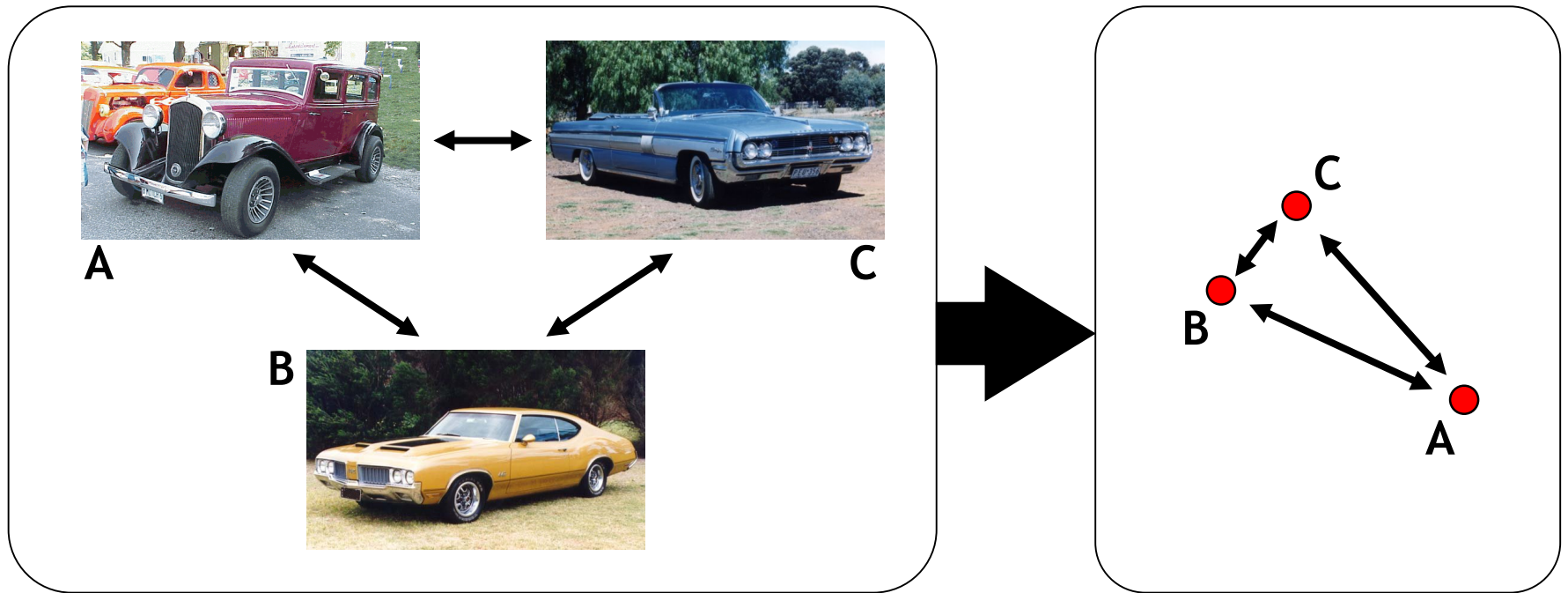


Multidimensional scaling (MDS)

- Criterion: preserve all inter-sample distances
- Needed: $n \times n$ distance matrix between all samples
- Map samples to a new (lower dimensional) space

Multidimensional scaling (MDS)

- Criterion: preserve all inter-sample distances
- Needed: $n \times n$ distance matrix between all points
- Map samples to a new (lower dimensional) space



MDS

- Advantages of using distances:
 - do not necessarily need original data

MDS

- Advantages of using distances:
 - do not necessarily need original data
 - allows for inclusion of **knowledge on objects** (e.g. characteristics of amino acids when comparing proteins)

MDS

- Advantages of using distances:
 - do not necessarily need original data
 - allows for inclusion of **knowledge on objects** (e.g. characteristics of amino acids when comparing proteins)
 - allows for inclusion of **knowledge of relations** (e.g. invariances) in distance measure (e.g. Pearson correlation being shift and scale independent when comparing expression profiles)

MDS

- Advantages of using distances:
 - do not necessarily need original data
 - allows for inclusion of **knowledge on objects** (e.g. characteristics of amino acids when comparing proteins)
 - allows for inclusion of **knowledge of relations** (e.g. invariances) in distance measure (e.g. Pearson correlation being shift and scale independent when comparing expression profiles)
 - easy to introduce nonlinearity

MDS

- Advantages of using distances:
 - do not necessarily need original data
 - allows for inclusion of **knowledge on objects** (e.g. characteristics of amino acids when comparing proteins)
 - allows for inclusion of **knowledge of relations** (e.g. invariances) in distance measure (e.g. Pearson correlation being shift and scale independent when comparing expression profiles)
 - easy to introduce nonlinearity
- Algorithms should find:
 - new, low-dimensional coordinates for each object
 - the number of dimensions to embed the data in

MDS: Non-linear mappings

- d_{ij} : distance $\| \mathbf{x}_i - \mathbf{x}_j \|$ in original space (? - dimensional)
- δ_{ij} : distance $\| \mathbf{y}_i - \mathbf{y}_j \|$ in new space (d - dimensional)

MDS: Non-linear mappings

- d_{ij} : distance $\| \mathbf{x}_i - \mathbf{x}_j \|$ in original space (? - dimensional)
- δ_{ij} : distance $\| \mathbf{y}_i - \mathbf{y}_j \|$ in new space (d - dimensional)

$$\text{Stress}(\mathbf{y}) = \frac{1}{\sum_i \sum_{j>i} d_{ij}^{(q+2)}} \sum_i \sum_{j>i} d_{ij}^q (\delta_{ij} - d_{ij})^2$$

- weight factor $q = \dots, -2, -1, 0, 1, 2, \dots$
 - $q > 0$: emphasise large distances
 - $q < 0$: de-emphasise large distances (smallest more important)

Sammon mapping: $q = -1$

MDS: Non-linear mappings (2)

- **Procedure:**
 - Initialize positions of samples in lower dimensional space (\mathbf{y}_i)

MDS: Non-linear mappings (2)

- **Procedure:**
 - Initialize positions of samples in lower dimensional space (\mathbf{y}_i)
 - Compute stress

$$\text{Stress}(\mathbf{y}) = \frac{1}{\sum_i \sum_{j>i} d_{ij}^{(q+2)}} \sum_i \sum_{j>i} d_{ij}^q (\delta_{ij} - d_{ij})^2$$

MDS: Non-linear mappings (2)

- **Procedure:**

- Initialize positions of samples in lower dimensional space (\mathbf{y}_i)
- Compute stress

$$\text{Stress}(\mathbf{y}) = \frac{1}{\sum_i \sum_{j>i} d_{ij}^{(q+2)}} \sum_i \sum_{j>i} d_{ij}^q (\delta_{ij} - d_{ij})^2$$

- Compute derivative of the stress with respect to positions of samples in new space

MDS: Non-linear mappings (2)

- **Procedure:**

- Initialize positions of samples in lower dimensional space (\mathbf{y}_i)
- Compute stress

$$Stress(\mathbf{y}) = \frac{1}{\sum_i \sum_{j>i} d_{ij}^{(q+2)}} \sum_i \sum_{j>i} d_{ij}^q (\delta_{ij} - d_{ij})^2$$

- Compute derivative of the stress with respect to positions of samples in new space
- Adapt the positions of samples in lower dimensional space

$$\mathbf{y}' = \mathbf{y} - \alpha \frac{\partial Stress(\mathbf{y})}{\partial \mathbf{y}}$$

MDS: Non-linear mappings (2)

- **Procedure:**

- Initialize positions of samples in lower dimensional space (\mathbf{y}_i)
- Compute stress

$$Stress(\mathbf{y}) = \frac{1}{\sum_i \sum_{j>i} d_{ij}^{(q+2)}} \sum_i \sum_{j>i} d_{ij}^q (\delta_{ij} - d_{ij})^2$$

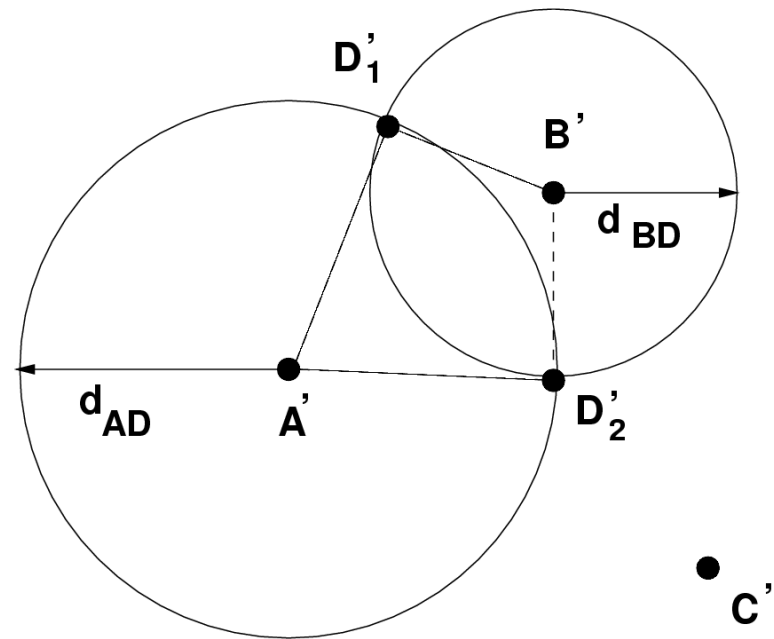
- Compute derivative of the stress with respect to positions of samples in new space
- Adapt the positions of samples in lower dimensional space

$$\mathbf{y}' = \mathbf{y} - \alpha \frac{\partial Stress(\mathbf{y})}{\partial \mathbf{y}}$$

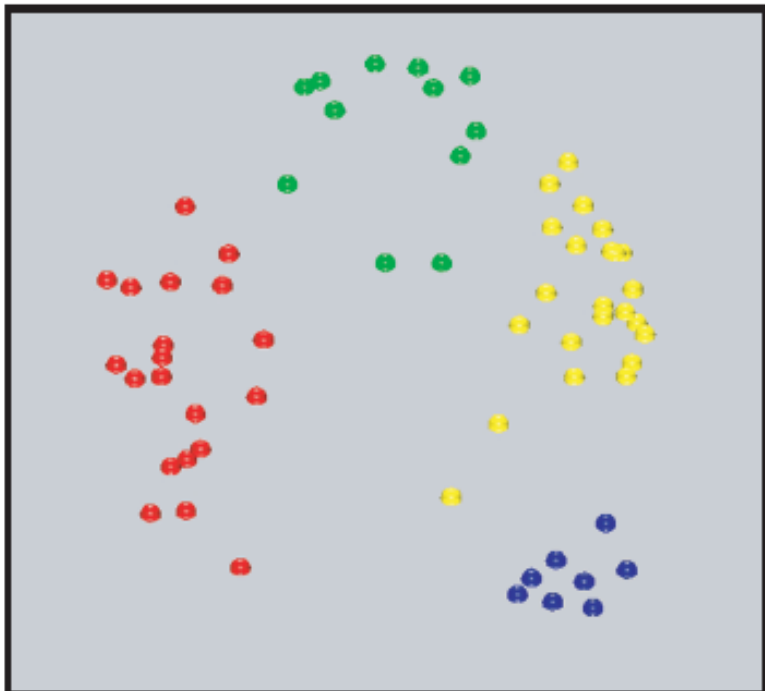
- Repeat till convergence (positions of samples do not change)

Embedding new points

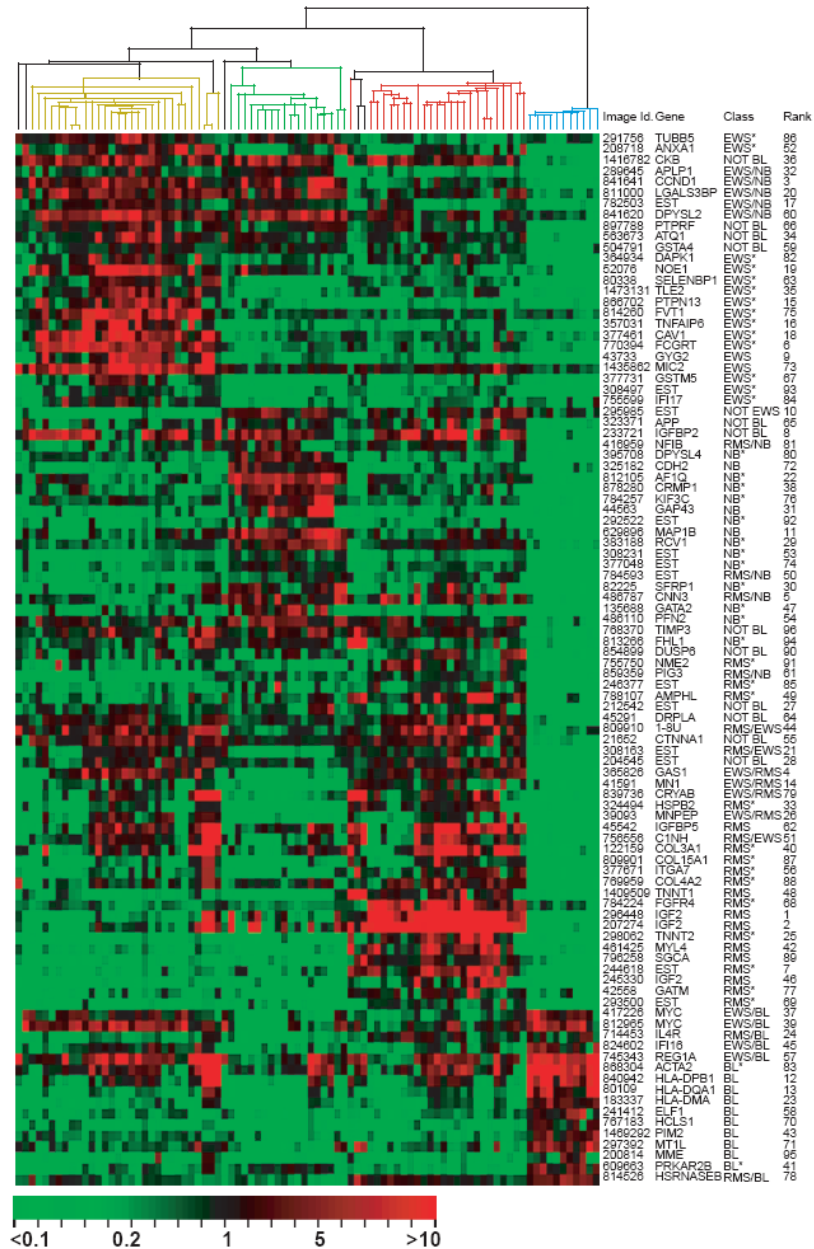
- Problematic: re-run entire algorithm...
- Sub-optimal solution: triangulation
 - Embed new point **D**
 - **D** has **A** and **B** as neighbors in original space
 - Preserve distance to two embedded neighbours **A'**, **B'** exactly
 - Use **C'** to decide which of the two candidates **D'₁'**, **D'₂'** to use



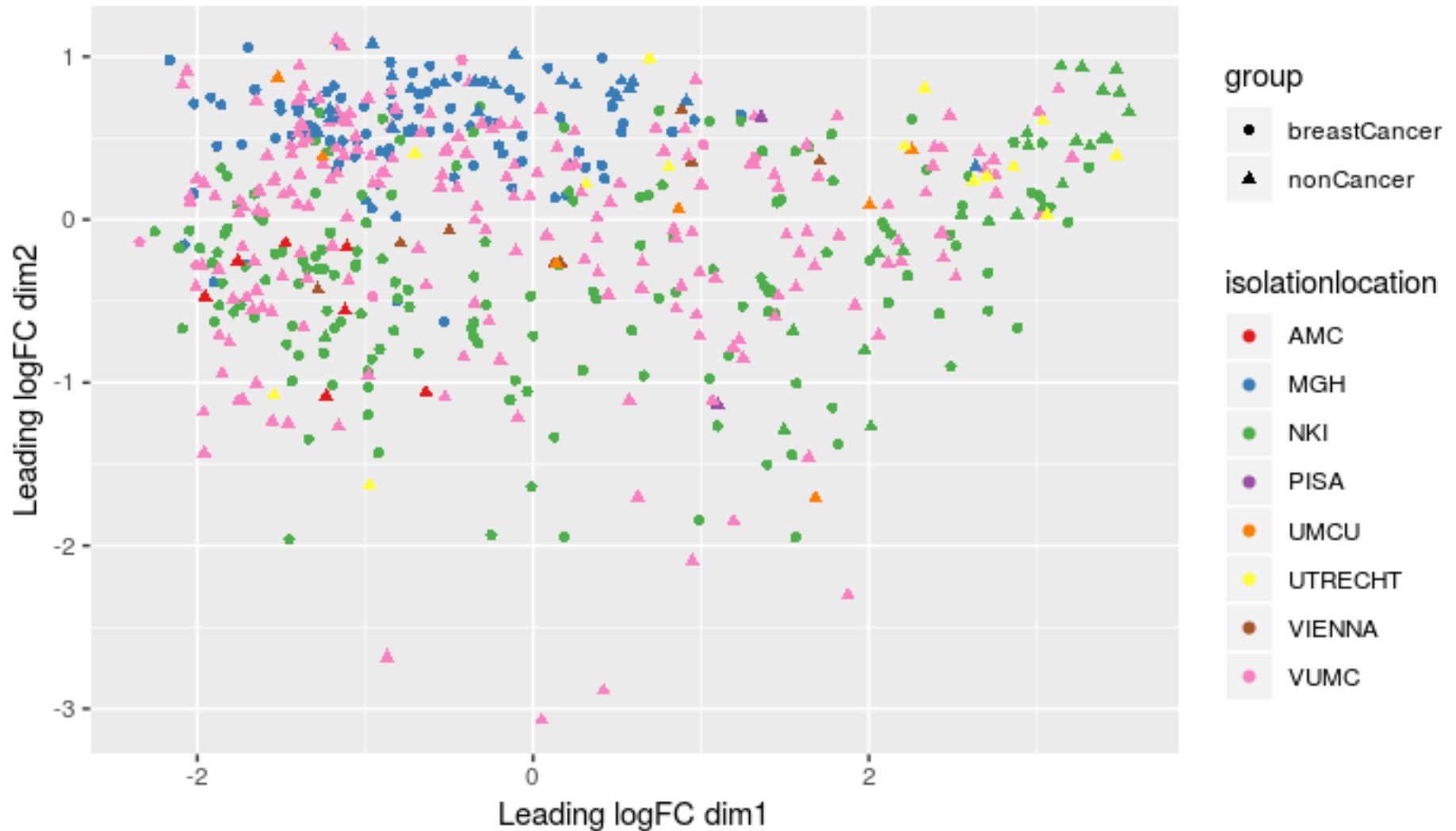
MDS example



- Neuroblastoma (NB)
- Rhabdomyosarcoma (RMS)
- Burkitt lymphoma (BL)
- Ewing family of tumors (EWS),



MDS example: detecting batch effects



t-SNE (t-distributed stochastic neighbor embedding) (van der Maaten et al, 2008)

- **Definitions:**

- A **data point** is a point x_i in the original data space R^D
- A **map point** is a point y_i in the map space R^2
- This space will contain our final representation of the dataset
- Every map point represents one of the data points.

t-SNE (2)

- **How do we choose the positions of the map points?**
 - We want to conserve the structure of the data
 - data points close \implies corresponding map points close

t-SNE (2)

- **How do we choose the positions of the map points?**

- We want to conserve the structure of the data
- data points close \implies corresponding map points close

- **Similarity of data points**

- Let $|x_i - x_j|$ be the Euclidean distance between two data points
- The conditional similarity between the two data points is:

$$p_{j|i} = \frac{\exp(-|x_i - x_j|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-|x_i - x_k|^2/2\sigma_i^2)}$$

t-SNE (2)

- How do we choose the positions of the map points?
 - We want to conserve the structure of the data
 - data points close \implies corresponding map points close

- Similarity of data points

- Let $|x_i - x_j|$ be the Euclidean distance between two data points
- The conditional similarity between the two data points is:

$$p_{j|i} = \frac{\exp(-|x_i - x_j|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-|x_i - x_k|^2/2\sigma_i^2)}$$

Convert a *dissimilarity*
(Euclidean distance)
to a *similarity*

- This represents the likelihood of x_j given x_i
- Assuming a Gaussian distribution around x_i with variance σ_i^2

t-SNE (3)

- The conditional similarity between the two data points is:

$$p_{j|i} = \frac{\exp(-|x_i - x_j|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-|x_i - x_k|^2 / 2\sigma_i^2)}$$

- σ_i^2 is different for every point
 - * dense areas are given a smaller variance
 - * sparse areas are given a larger variance
- Similarity is symmetrized form:

$$p_{ij} = (p_{j|i} + p_{i|j}) / 2N$$

t-SNE (4)

- **Similarity of map points:**

- Let $|y_i - y_j|$ be the Euclidean distance between the map points

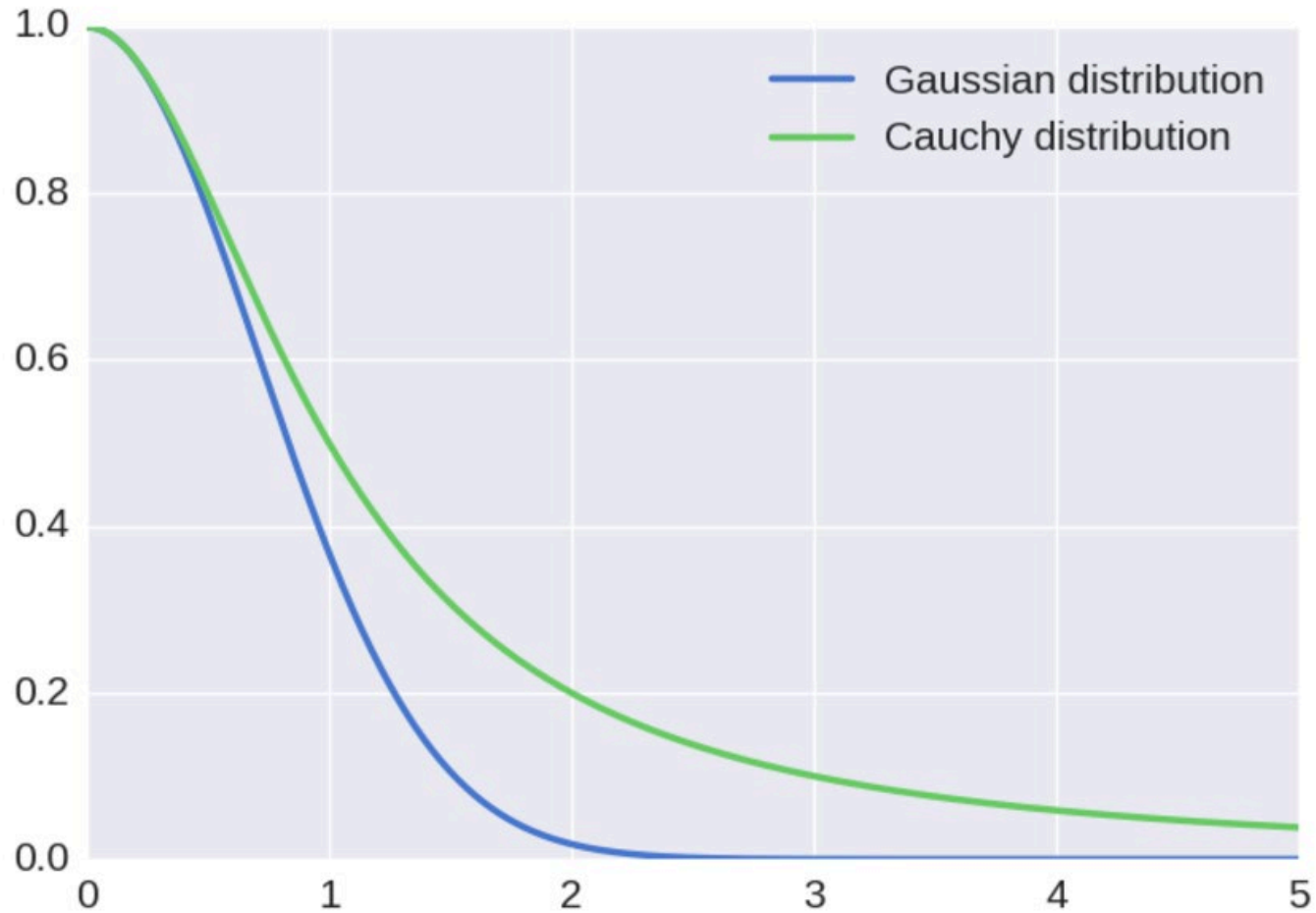
$$q_{ij} = \frac{f(|y_i - y_j|)}{\sum_{k \neq i} f(|y_i - y_k|)}$$

with

$$f(z) = 1/(1 + z^2)$$

- Same as for the data points, but different distribution:
- t-Student with one degree of freedom, or Cauchy distribution

t-SNE (5): Cauchy and Gaussian distribution



t-SNE (6)

- **How do we choose the positions of the map points?**
 - Data similarity matrix (p_{ij}) is fixed
 - Map similarity matrix (q_{ij}) depends on the map points
 - Get two matrices as similar as possible
 - Idea: similar data points yield similar map points

***Distribution of data and
map similarities is the
same***

t-SNE (7)

- **Algorithm**

- Minimizing the Kullback-Leibler divergence between p_{ij} and q_{ij} :

$$KL(P, Q) = \sum_{i,j} p_{ij} \log(p_{ij}/q_{ij})$$

- $KL(P, Q)$ measures the distance between our two distributions

t-SNE (7)

- **Algorithm**

- Minimizing the Kullback-Leibler divergence between p_{ij} and q_{ij} :

$$KL(P, Q) = \sum_{i,j} p_{ij} \log(p_{ij}/q_{ij})$$

- $KL(P, Q)$ measures the distance between our two distributions
- To minimize this score, we perform a gradient descent
- The gradient can be computed analytically:

$$\frac{\partial KL(P, Q)}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij}) f(|y_i - y_j|)$$

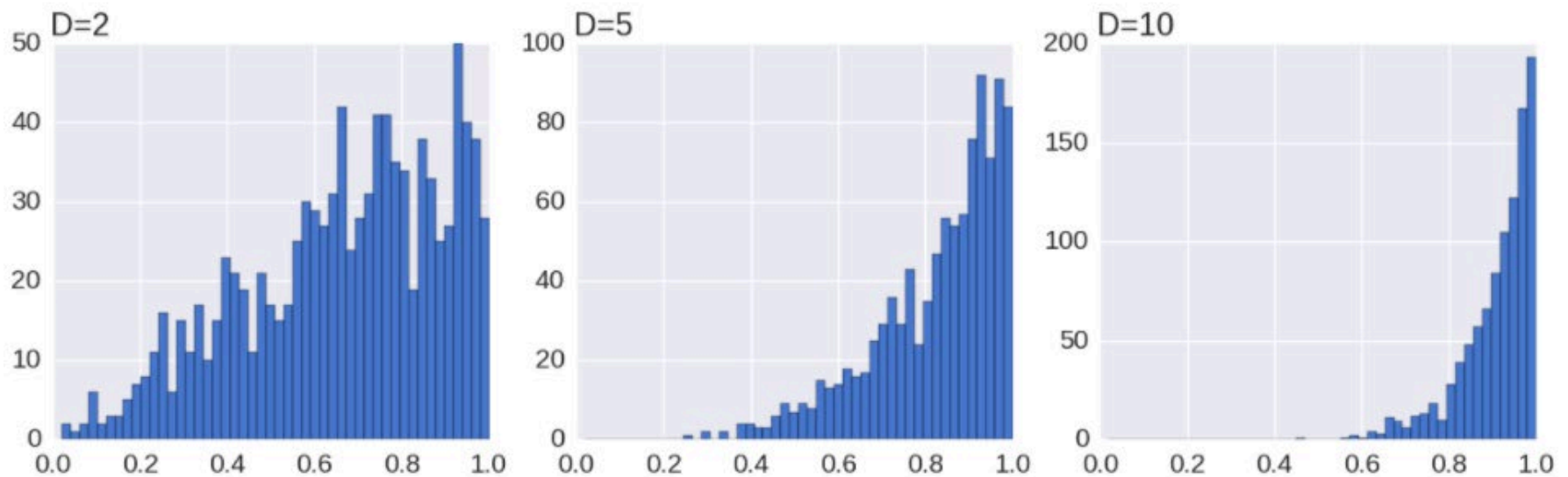
where $f(z) = z/(1 + z^2)$

- The gradient expresses the sum of all spring forces applied to map point i .

t-SNE (8)

- Why the t-Student distribution?

- For large N , random points are close to the surface, not in center



Distance from origin (radius, r)

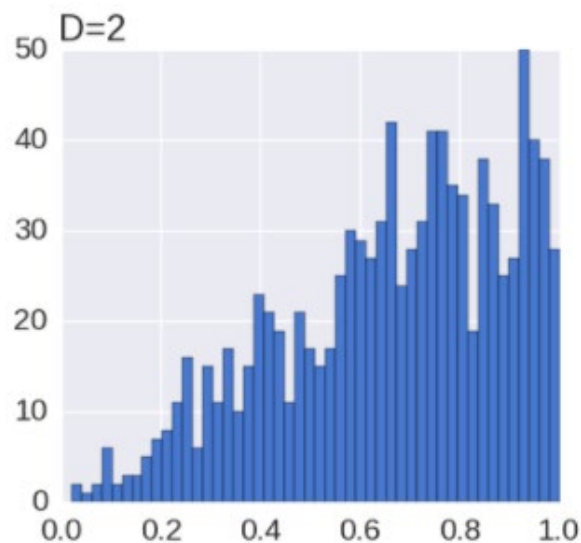
t-SNE (9)

- **Why the t-Student distribution?**
 - We go from high dimensionality to low dimensionality
 - If we use the same distribution in data and map space
 - There is an imbalance in the distribution of the distances of a point's neighbors.
 - As distances are so different between a high-D and low-D space
 - Yet, t-SNE tries to reproduce the same distances in the two spaces
 - Result: excess of 'attraction forces' (gradient) that move map space points
 - t-SNE compensates this by the heavier tail of the t-distribution in low-D
 - Larger distances in low-D get more weight so match high-D

t-SNE (9)

- Why the t-Student distribution?

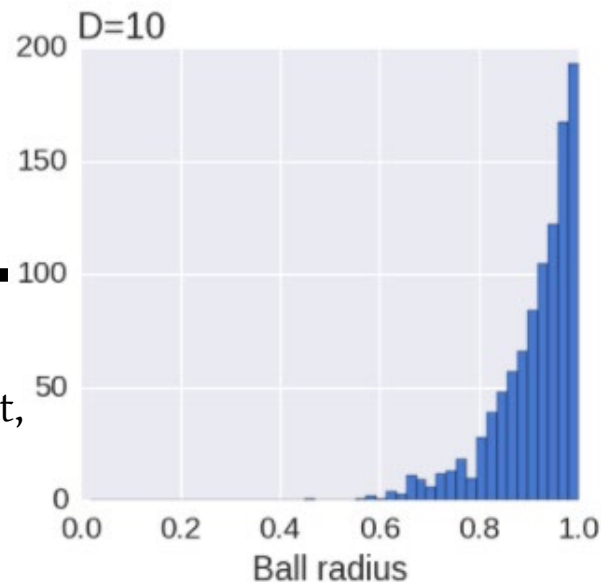
- We go from high dimensionality to low dimensionality
- If we use the same distribution in data and map space
- There is an imbalance in the distribution of the distances of a point's neighbors.
- As distances are so different between a high-D and low-D space



Uniformly (randomly)
Sampled in both
2D and 10D



But
distributions different,
Cauchy enlarges
larger distances



t-SNE (10)

- **Setting σ_i**
 - Unlikely that a single value of σ_i is optimal

t-SNE (10)

- **Setting σ_i**
 - Unlikely that a single value of σ_i is optimal
 - Density of datapoints varies
 - * In dense regions a smaller value
 - * In sparse regions a larger value

t-SNE (10)

- **Setting σ_i**
 - Unlikely that a single value of σ_i is optimal
 - Density of datapoints varies
 - * In dense regions a smaller value
 - * In sparse regions a larger value
 - σ_i induces a probability distribution, P_i , across all datapoints

t-SNE (10)

- **Setting σ_i**

- Unlikely that a single value of σ_i is optimal
- Density of datapoints varies
 - * In dense regions a smaller value
 - * In sparse regions a larger value
- σ_i induces a probability distribution, P_i , across all datapoints
- This distribution's entropy increases as σ_i increases

If *sigma* is small, all
pairwise similarities = 0
= low entropy

t-SNE (10)

- **Setting σ_i**

- Unlikely that a single value of σ_i is optimal
- Density of datapoints varies
 - * In dense regions a smaller value
 - * In sparse regions a larger value
- σ_i induces a probability distribution, P_i , across all datapoints
- This distribution's entropy increases as σ_i increases
- t-SNE searches for the σ_i such that P_i has the specified *perplexity* (J):

If *sigma* is small, all pairwise similarities = 0 = low entropy

$$J(P_i) = 2^{H(P_i)}$$

Perplexity proportional to entropy

- where $H(P_i)$ is the Shannon entropy of P_i in bits:

$$H(P_i) = - \sum_j p_{j|i} \log_2(p_{j|i})$$

t-SNE (10)

- **Setting σ_i**

- Unlikely that a single value of σ_i is optimal
- Density of datapoints varies
 - * In dense regions a smaller value
 - * In sparse regions a larger value
- σ_i induces a probability distribution, P_i , across all datapoints
- This distribution's entropy increases as σ_i increases
- t-SNE searches for the σ_i such that P_i has the specified *perplexity* (J):

If *sigma* is small, all pairwise similarities = 0 = low entropy

$$J(P_i) = 2^{H(P_i)}$$

Perplexity proportional to entropy

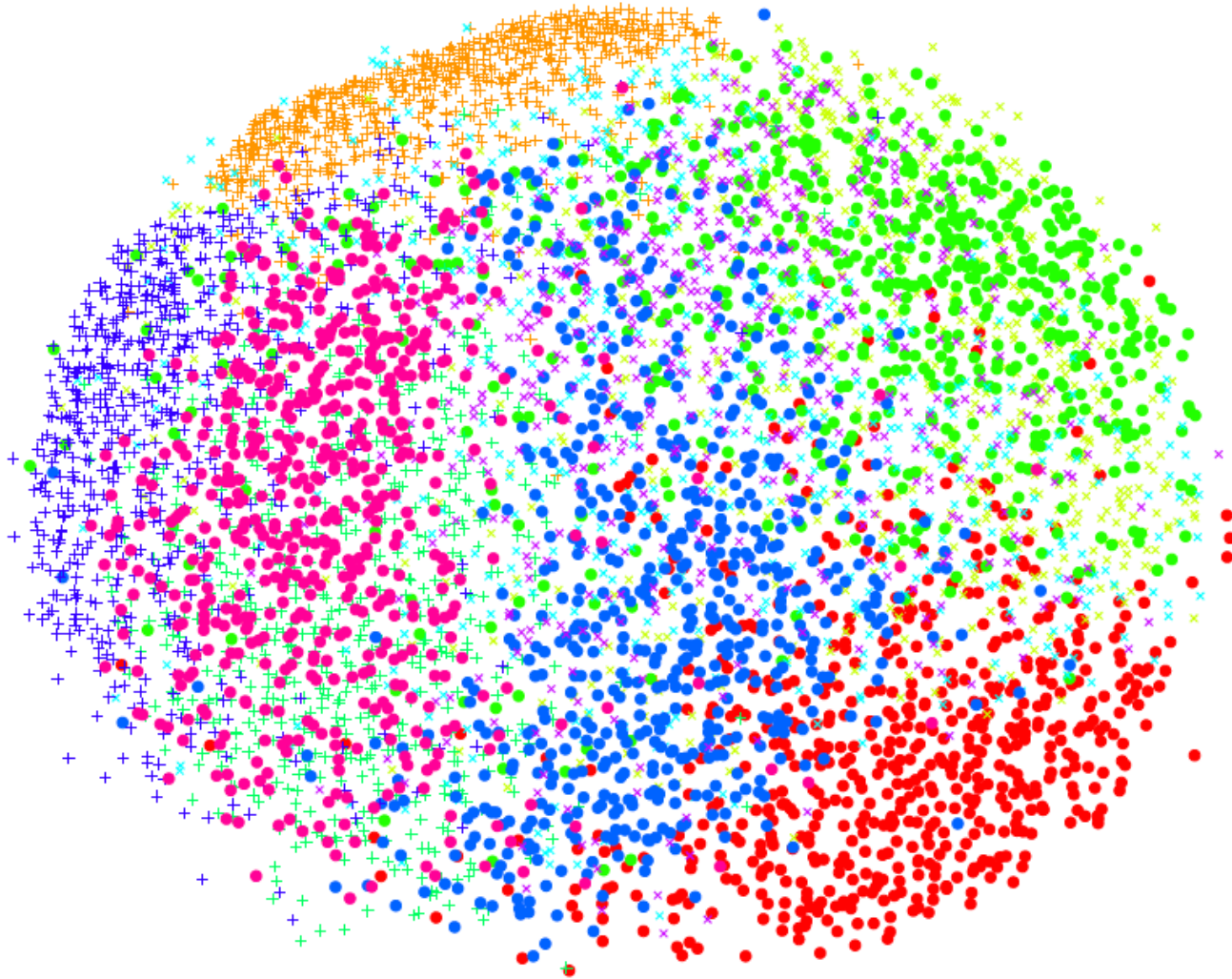
- where $H(P_i)$ is the Shannon entropy of P_i in bits:

$$H(P_i) = - \sum_j p_{j|i} \log_2(p_{j|i})$$

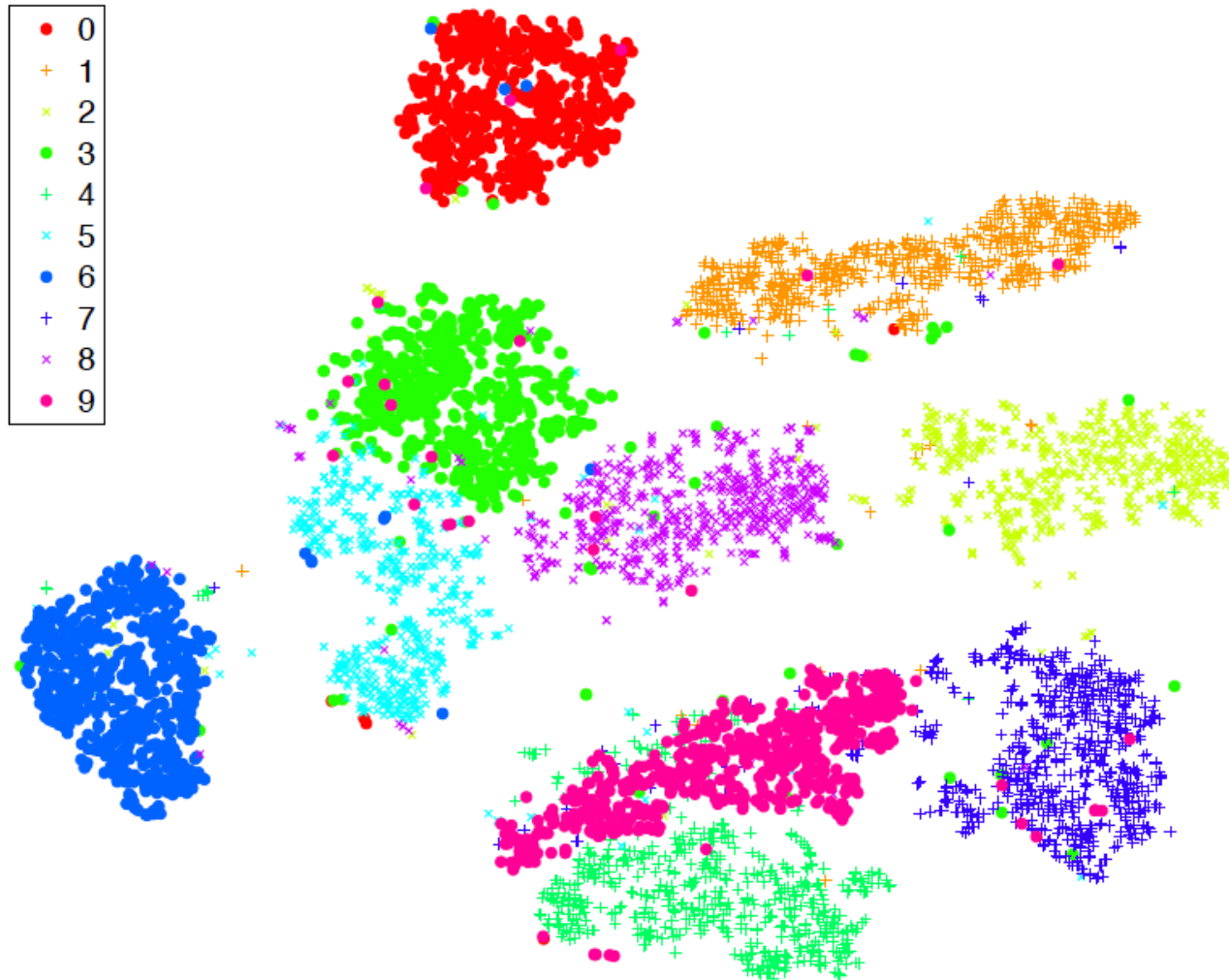
- Perplexity is a smooth measure of the effective number of neighbors
- t-SNE is not sensitive to this value: set between 5 and 50



t-SNE (11): Sammon map of digit data



t-SNE (12): t-SNE map of digit data



MDS and t-SNE conclusions

- Experts or measurements give distances
- Optimise a *stress-function* (MDS) or KL distance (t-SNE)
- Important:
 - *the distance measure used*: is it representative?
 - *the weighting of distances (q)*: can influence outcome heavily.
 - t-SNE run with defaults is quite reliable
- Largest risk: seeing structure in the data that is not really there
- Remaining problem: embedding new data points
- t-SNE (and now UMAP) are modern techniques to perform representation of data in high-D space in 2D

Feature selection

- For feature selection, we need:
 - A **criterion** function
e.g. error, class overlap, information loss
 - A **search algorithm**
e.g. pick the best single feature at each time

Criteria

1. **Wrapper**: exact performance measure

- base performance estimate on classifier;
- estimate performance using cross-validation:
- very expensive!

Criteria

1. **Wrapper**: exact performance measure

- base performance estimate on classifier;
- estimate performance using cross-validation:
- very expensive!

Note:

we should never use the training set to calculate performance; this will give a biased estimate!

Criteria

1. **Wrapper**: exact performance measure

- base performance estimate on classifier;
- estimate performance using cross-validation:
- very expensive!

Note:

we should never use the training set to calculate performance; this will give a biased estimate!

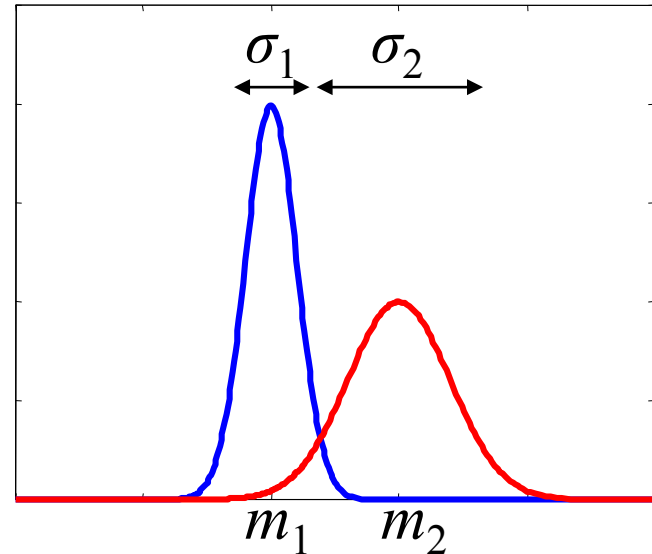
2. **Filter**: approximate performance predictors:

- calculate the performance of an easy-to-use/'cheap' model
- indication of how well a more powerful model may perform
- is much faster to compute.

Criteria (2)

- Example
 - Simple measure of the ‘separability’ of classes given a feature
 - 1D case: Signal-to-Noise Ratio (SNR) or Fisher criterion:

$$J_F = \frac{|m_1 - m_2|^2}{(\sigma_1^2 + \sigma_2^2)}$$



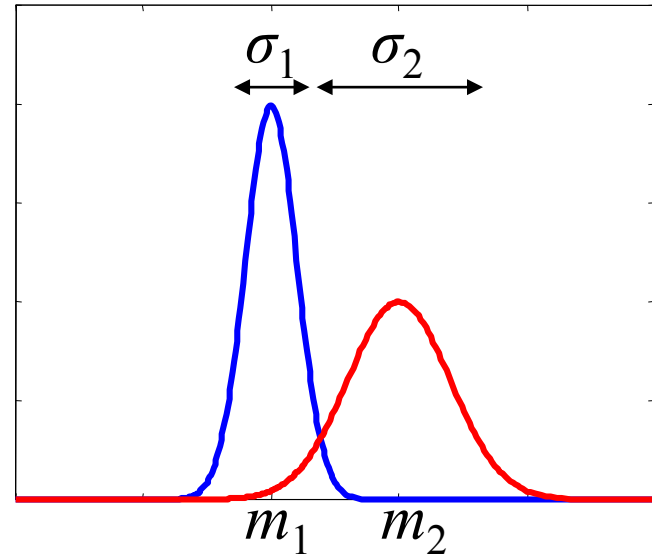
Criteria (3)

- Example

- Simple measure of the ‘separability’ of classes given a feature
- 1D case: Signal-to-Noise Ratio (SNR) or Fisher criterion:

$$J_F = \frac{|m_1 - m_2|^2}{(\sigma_1^2 + \sigma_2^2)}$$

- If J_F is large: good separability
- If J_F is small: poor separability

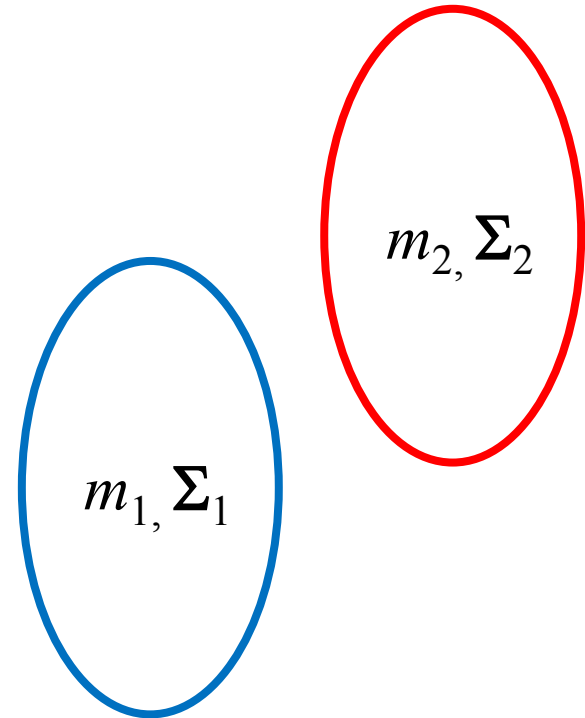


Criteria (4)

- The multi-variate equivalent of Fisher criterion is the
- Mahalanobis distance:
 - assumes
 - Gaussian distributions with
 - **equal** covariance matrix Σ :

$$D_M = (m_1 - m_2)^T \Sigma^{-1} (m_1 - m_2)$$

$$\Sigma = \sum_{i=1}^C \frac{n_i}{n} \Sigma_i$$



Criteria (5)

- Recall

$$\mathbf{S}_w = \sum_{i=1}^C \frac{n_i}{n} \mathbf{\Sigma}_i$$

$$\mathbf{S}_B = \sum_{i=1}^C \frac{n_i}{n} (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$$

- Scatter-based classification performance indicators:

$$J_1 = \text{trace} (\mathbf{S}_w + \mathbf{S}_B) = \text{trace} (\mathbf{\Sigma})$$

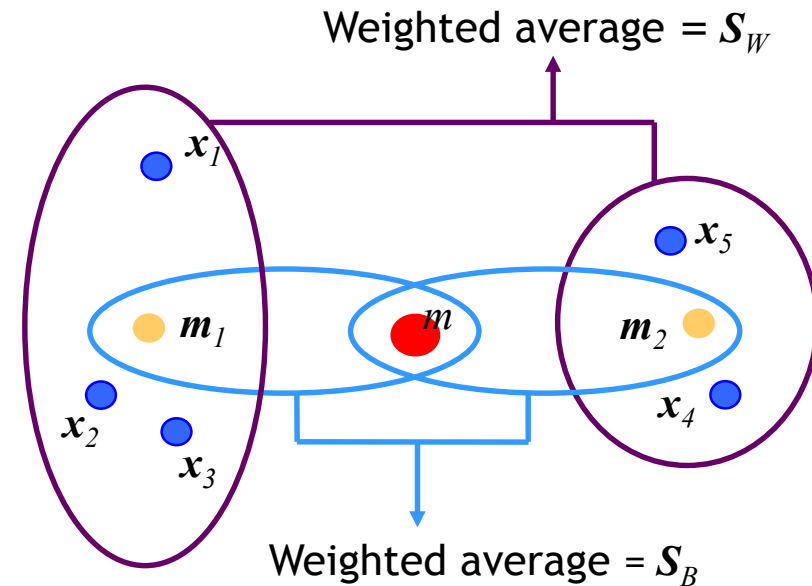
$$J_2 = \text{trace} (\mathbf{S}_B / \mathbf{S}_w)$$

$$J_3 = \det (\mathbf{\Sigma}) / \det (\mathbf{S}_w)$$

$$J_3 = \text{trace} (\mathbf{S}_w) / \text{trace} (\mathbf{S}_B)$$

(trace = sum of diagonal elements)

- These are all just approximations!



Search algorithms

- **Feature selection:** select a subset of d out of p features which optimises the criterion
- Simplest solution: look at all possible subsets

- Problem: there are $\binom{p}{d} = \frac{p!}{(p-d)!d!}$ subsets

- *e.g.* $p = 50$ features,

$d = 2$: 1225 subsets

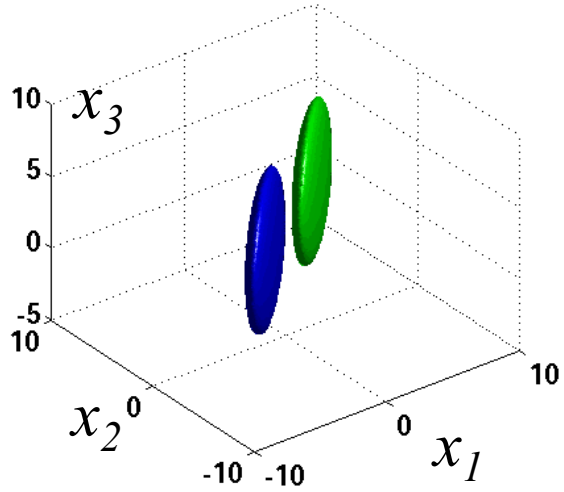
$d = 5$: 2.1×10^6 subsets

$d = 25$: 1.3×10^{14} subsets

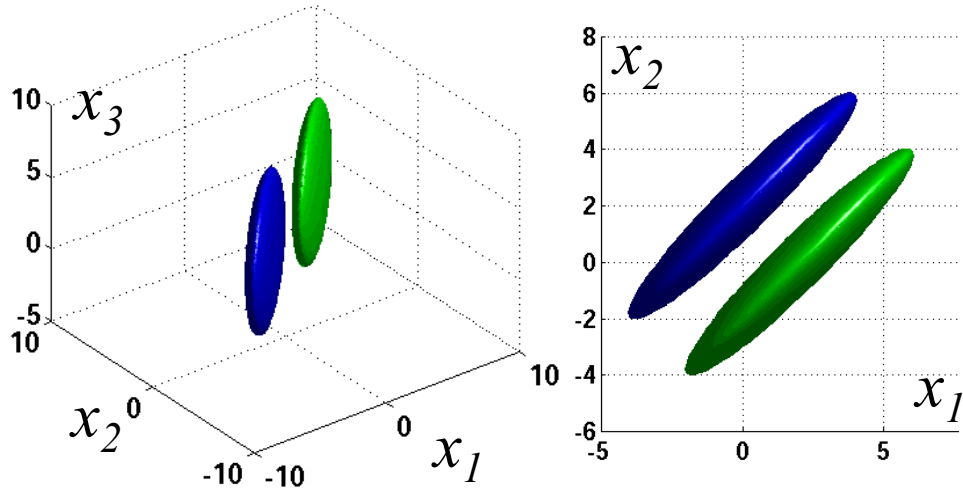
Search algorithms (2)

- Sub-optimal algorithms: select or deselect one feature (or a few features) at a time
- Simplest: best individual d
but these are not necessarily the best d !
- Demonstration: two Gaussians;
select 2 features out of 3 for classification

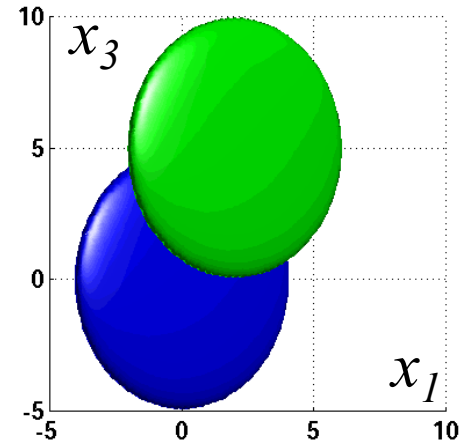
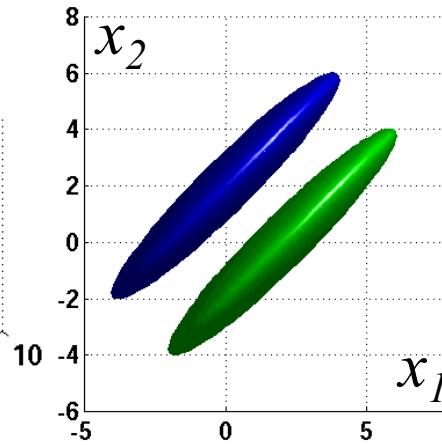
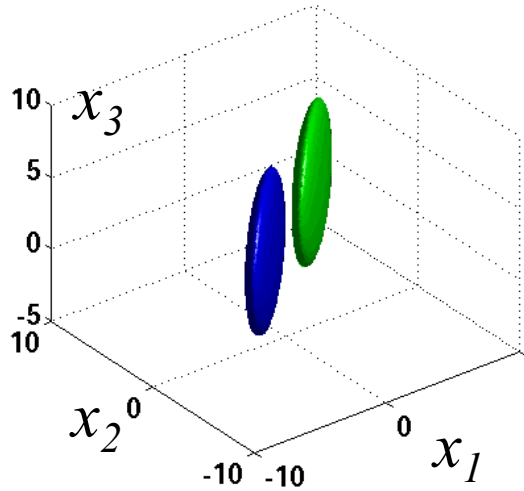
Search algorithms (3)



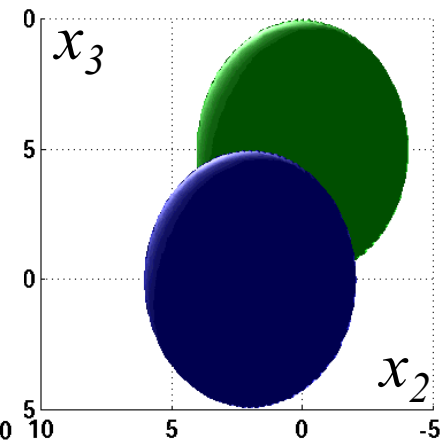
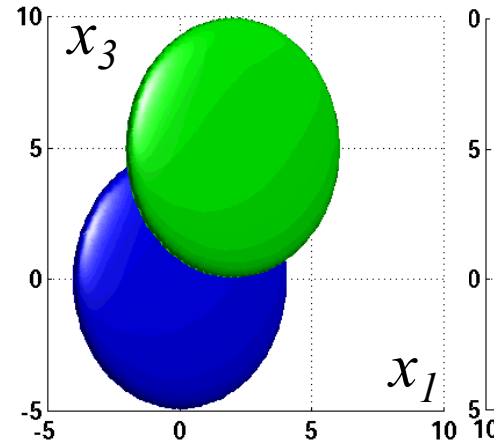
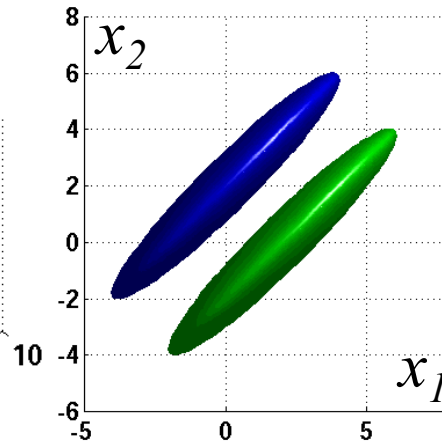
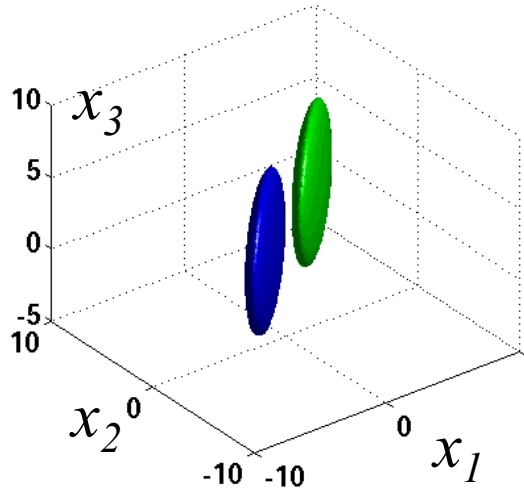
Search algorithms (3)



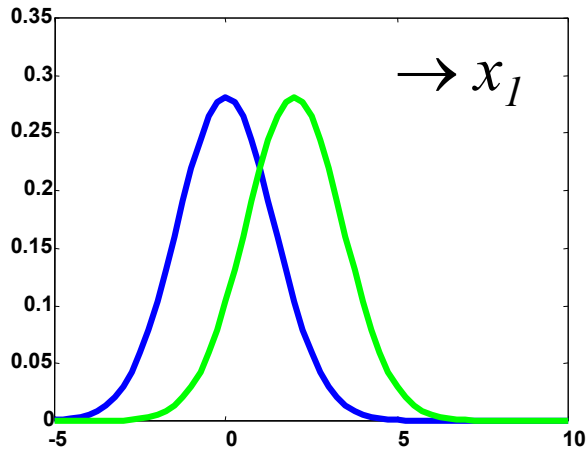
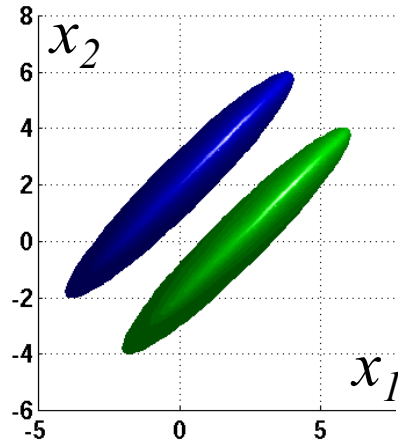
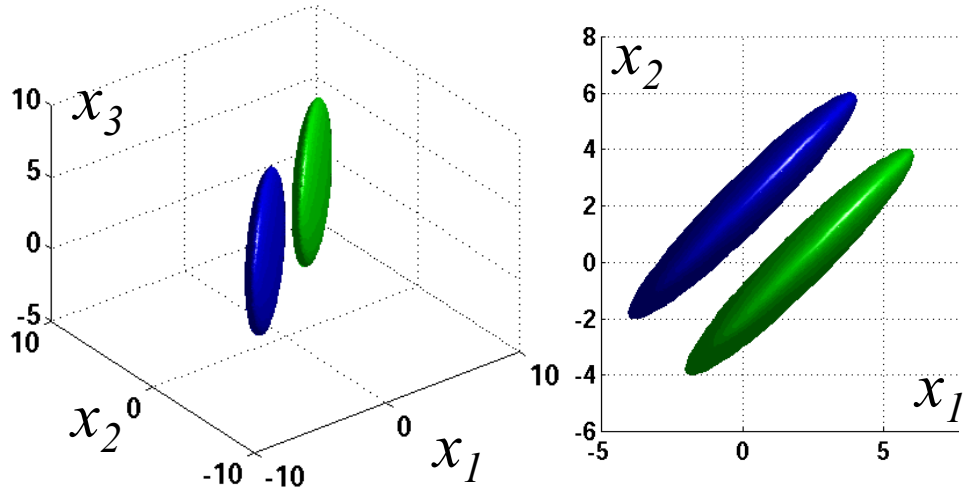
Search algorithms (3)



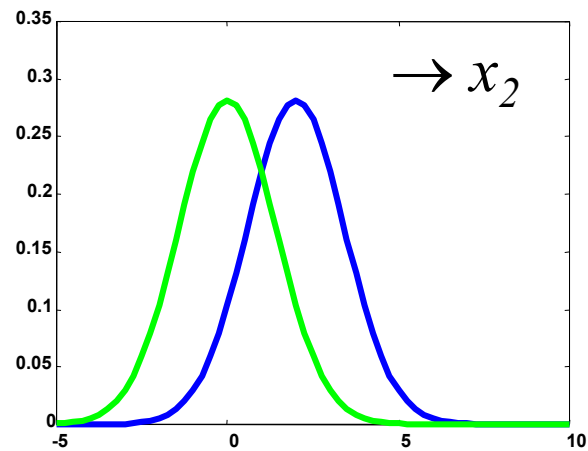
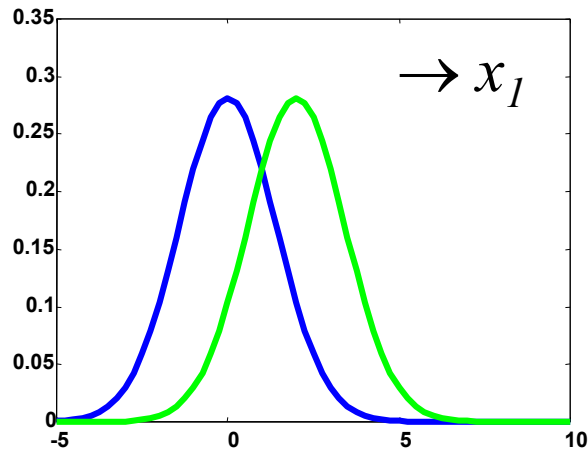
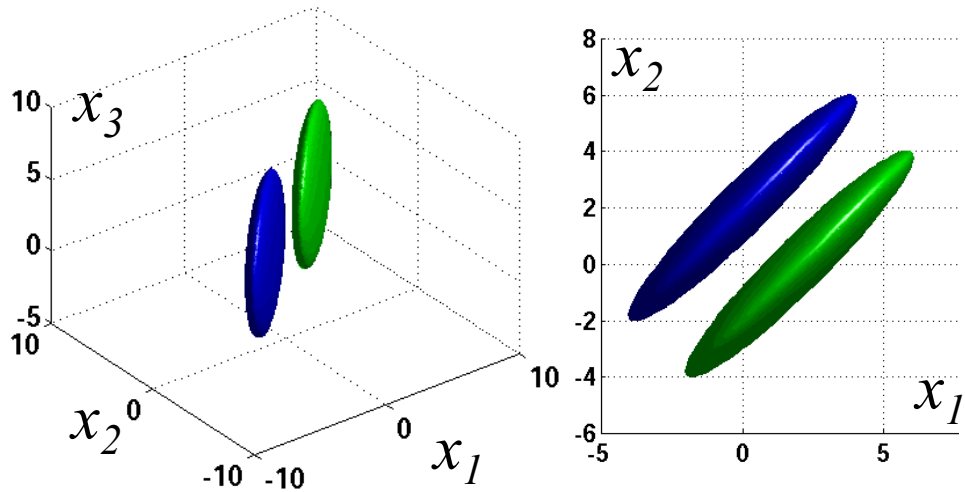
Search algorithms (3)



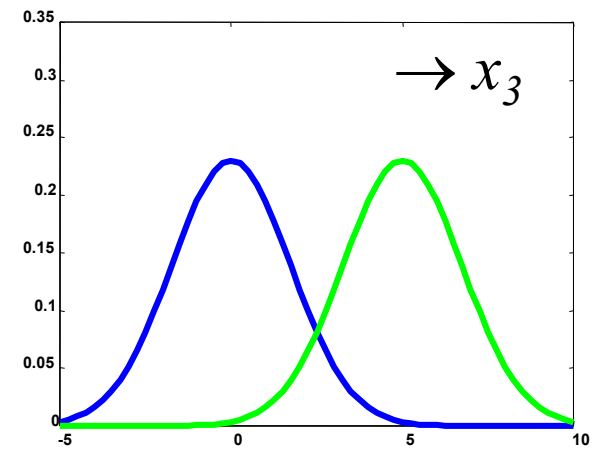
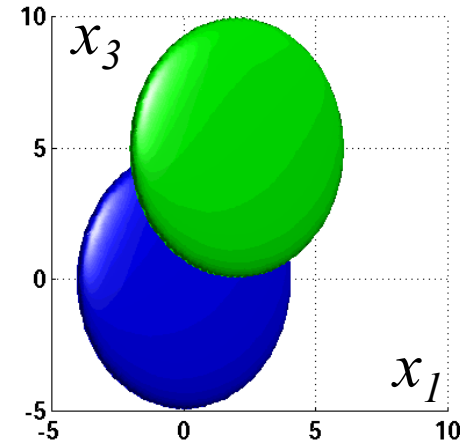
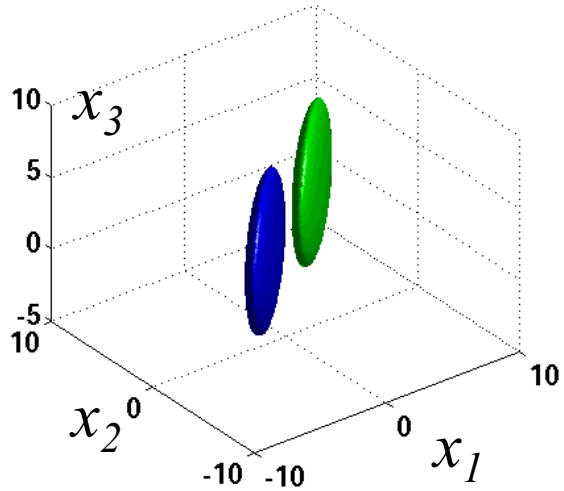
Search algorithms (3)



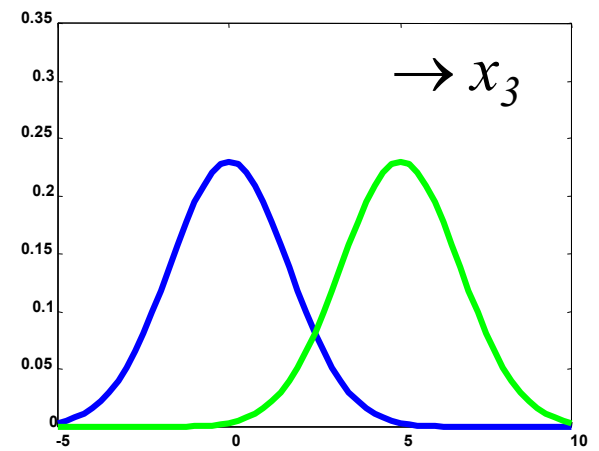
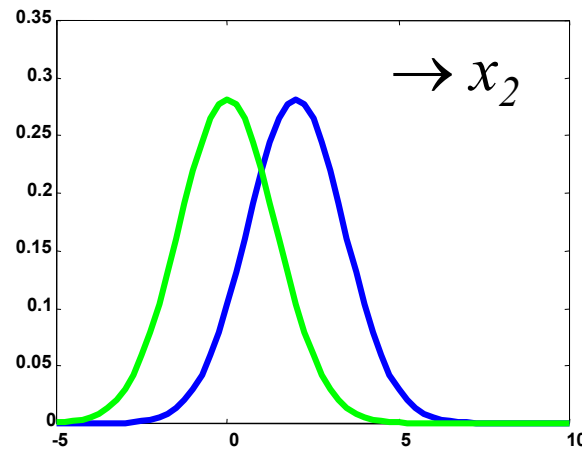
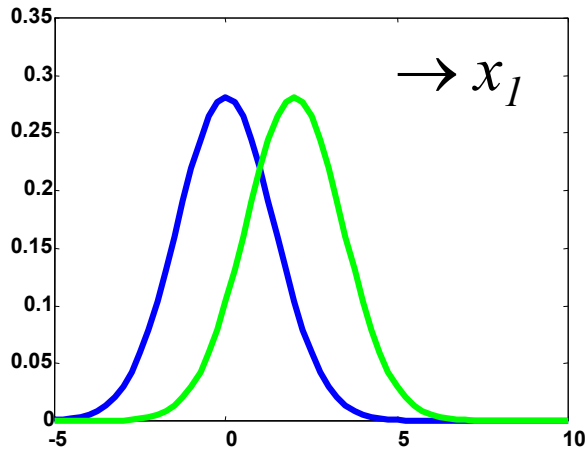
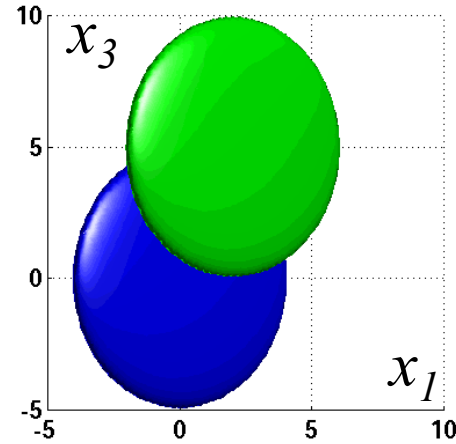
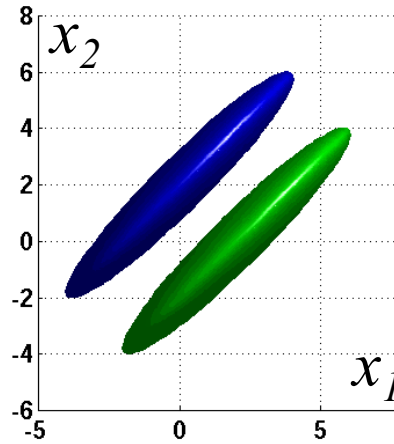
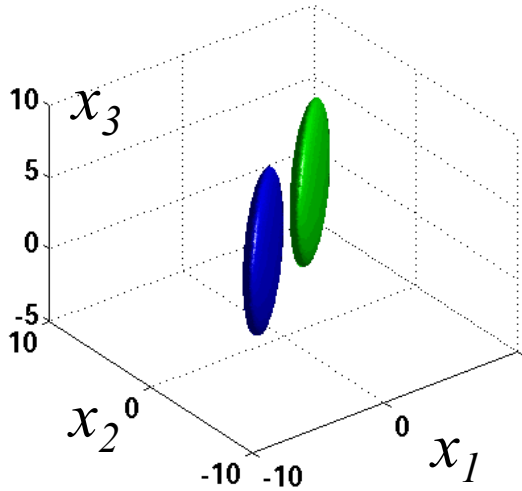
Search algorithms (3)



Search algorithms (3)



Search algorithms (3)



Search algorithms (4)

- Other sub-optimal algorithms:
 - Forward selection (for when d is low)
 - start with empty set
 - keep adding one feature at a time so that the entire subset so far performs best

Search algorithms (4)

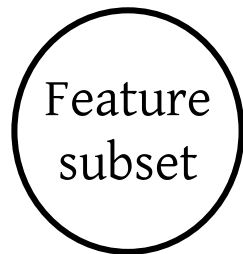
- Other sub-optimal algorithms:
 - Forward selection (for when d is low)
 - start with empty set
 - keep adding one feature at a time
so that the entire subset so far performs best
 - Backward selection (for when d is high)
 - start with entire set
 - keep removing one feature at a time
so that the entire subset so far performs best

Search algorithms (4)

- Other sub-optimal algorithms:
 - Forward selection (for when d is low)
 - start with empty set
 - keep adding one feature at a time
so that the entire subset so far performs best
 - Backward selection (for when d is high)
 - start with entire set
 - keep removing one feature at a time
so that the entire subset so far performs best
 - Plus- l -takeaway- r (may be slightly better)
 - start with empty set (if $l > r$) or entire set (if $l < r$)
 - keep adding best l and removing worst r

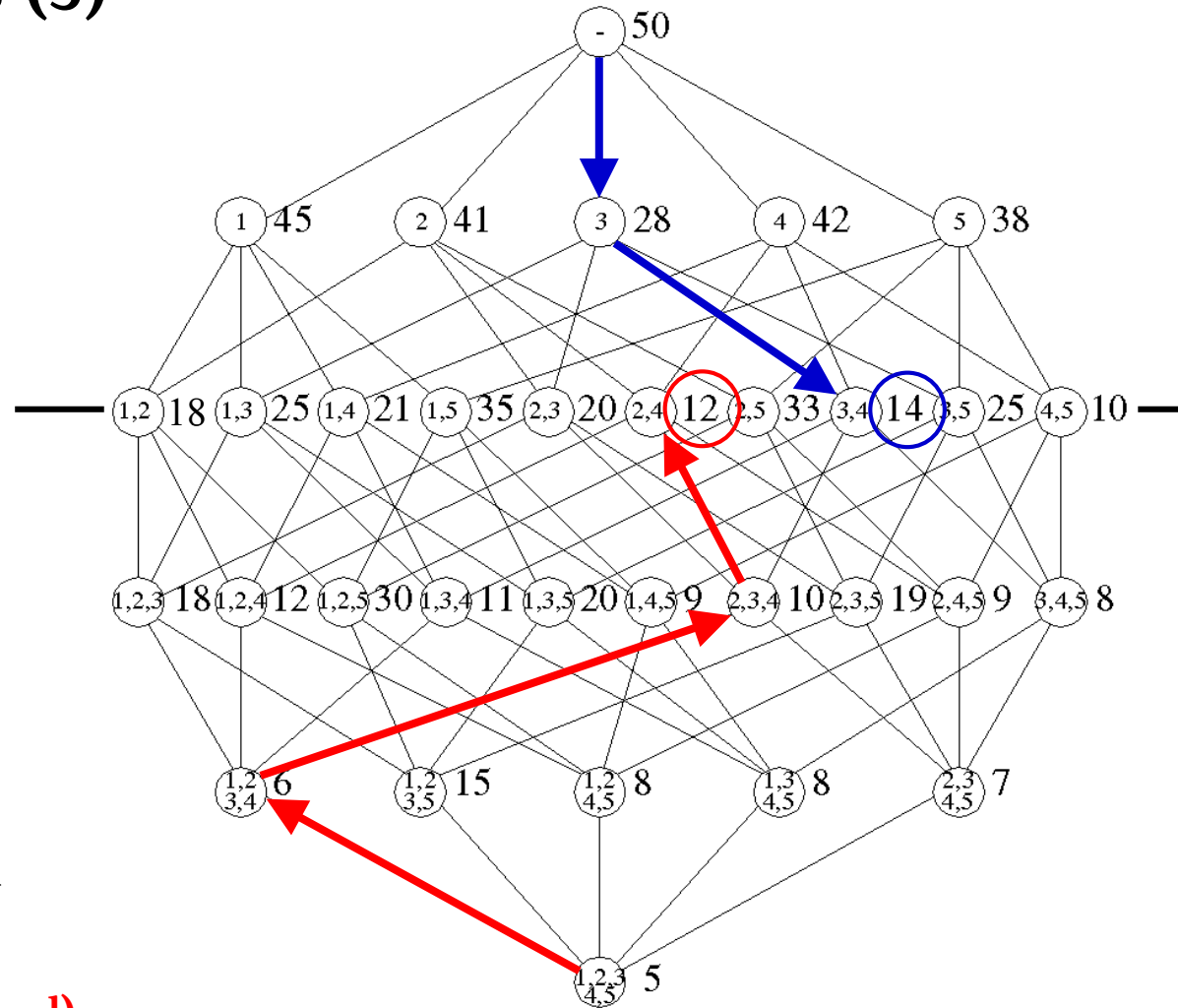
Search algorithms (5)

- Select $d = 2$ out of $p = 5$ features
- Sub-optimality illustrated:
 - forward
 - backward



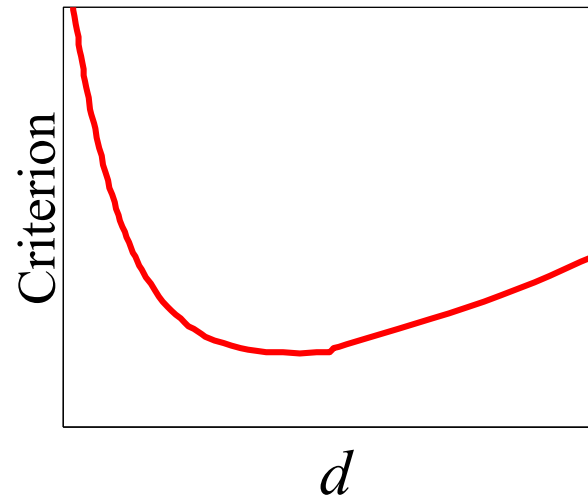
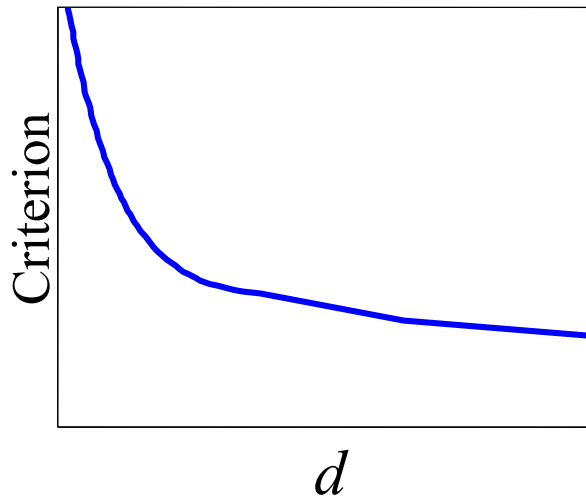
Criterion value

(Low is good)



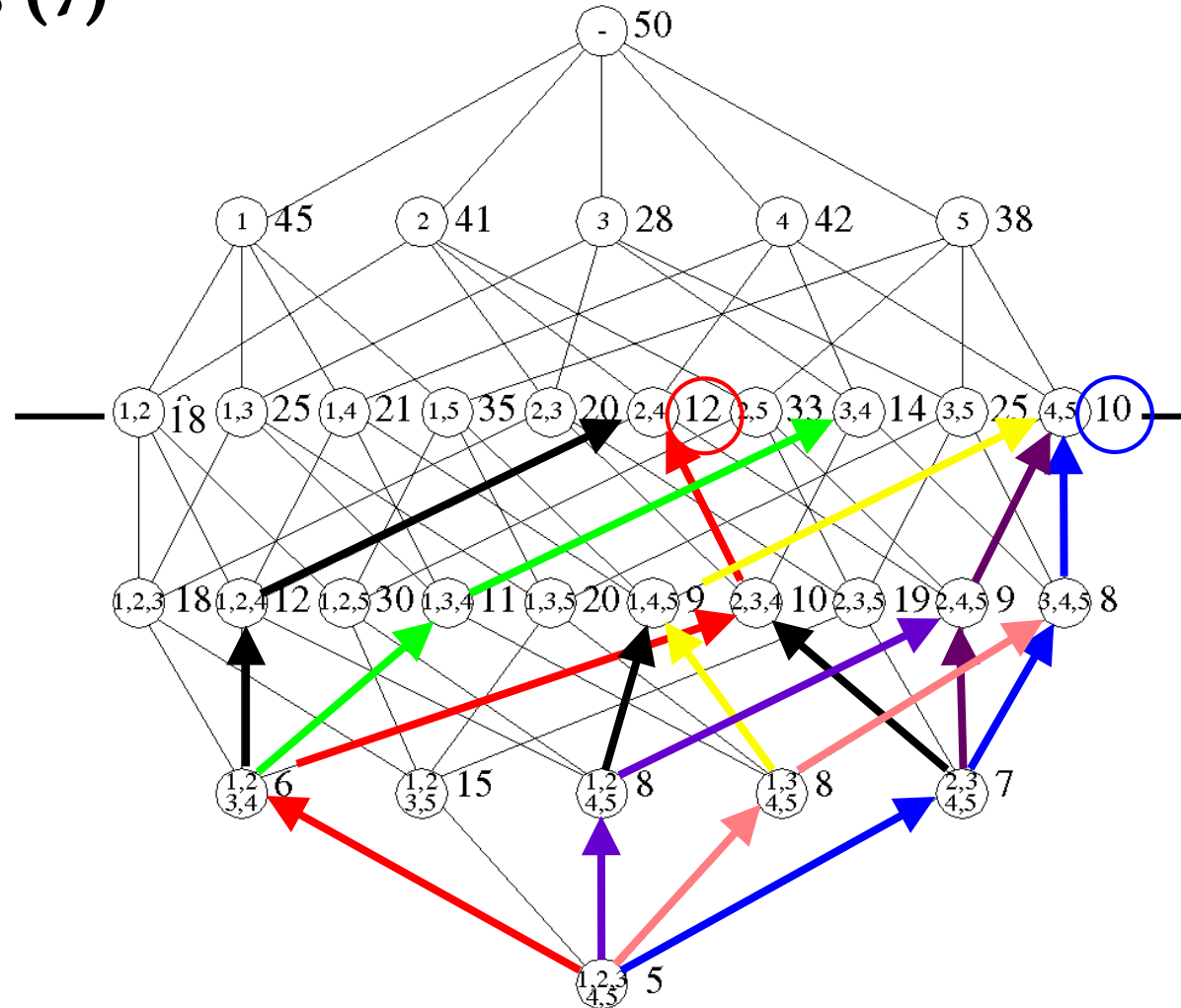
Search algorithms (6)

- Branch & bound: backtracking
- Optimal when criterion is monotonic in the number of features d



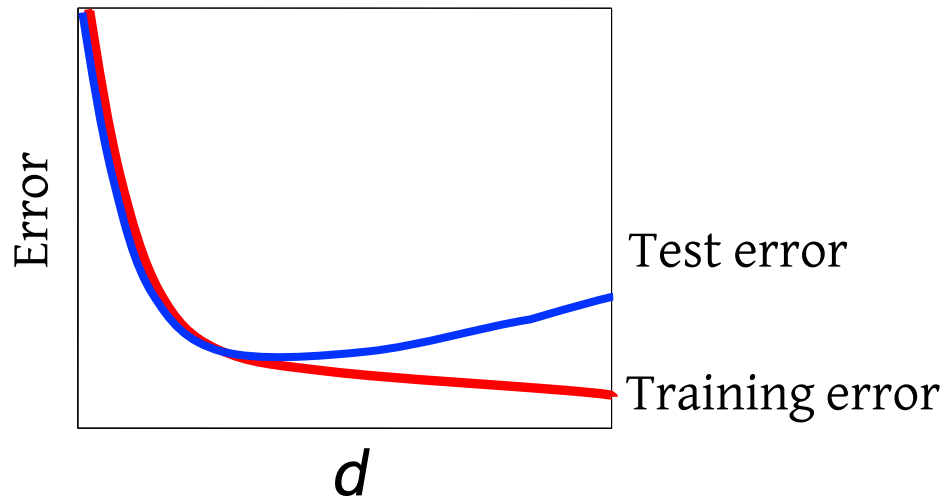
Search algorithms (7)

- Branch & bound
 - Use backward search to find preset number of features
 - Set *bound*, backtrack (*branch*) and use backward search again, considering just sets with criterion values better than the bound



Search algorithms (8)

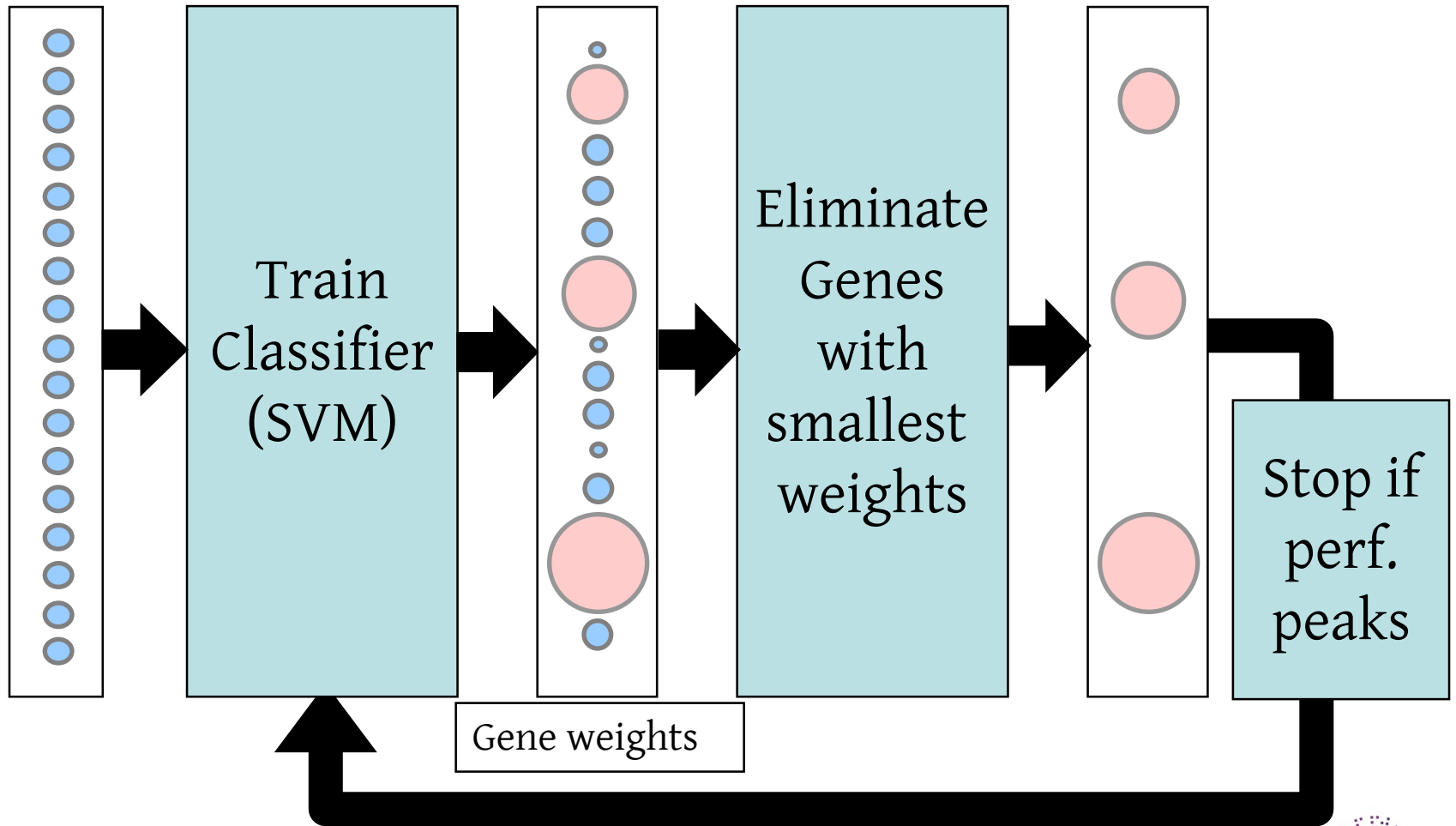
- When should we stop?
 - Due to estimation problems (e.g. covariance matrix), we may be overtraining on training set
 - This is revealed by increasing error on the test set



- Otherwise (with very large sample sizes), we will have to specify a desired number of measurements

Example: Recursive feature elimination (RFE)

Wrapper, Backward search

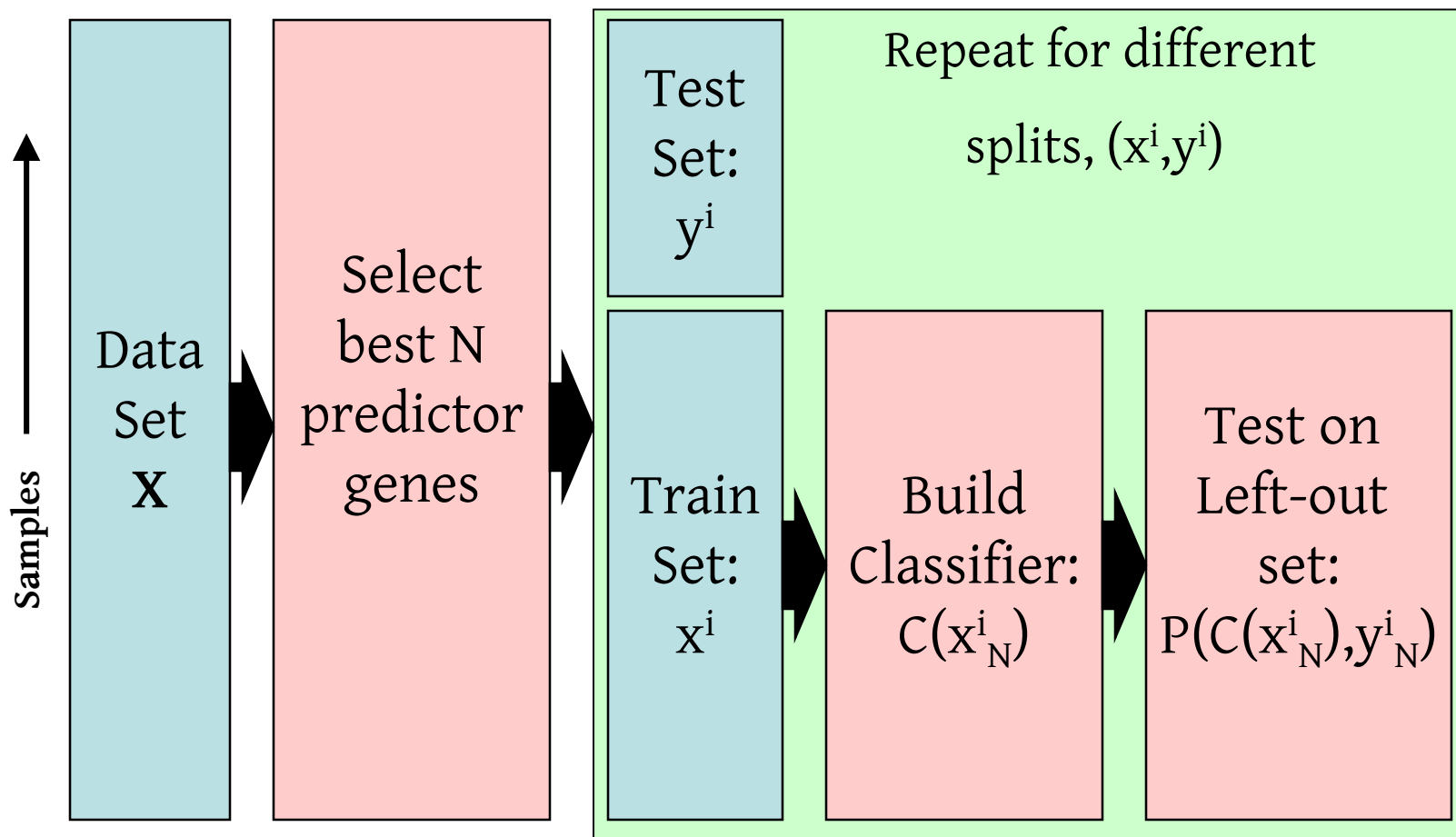


What can go wrong?

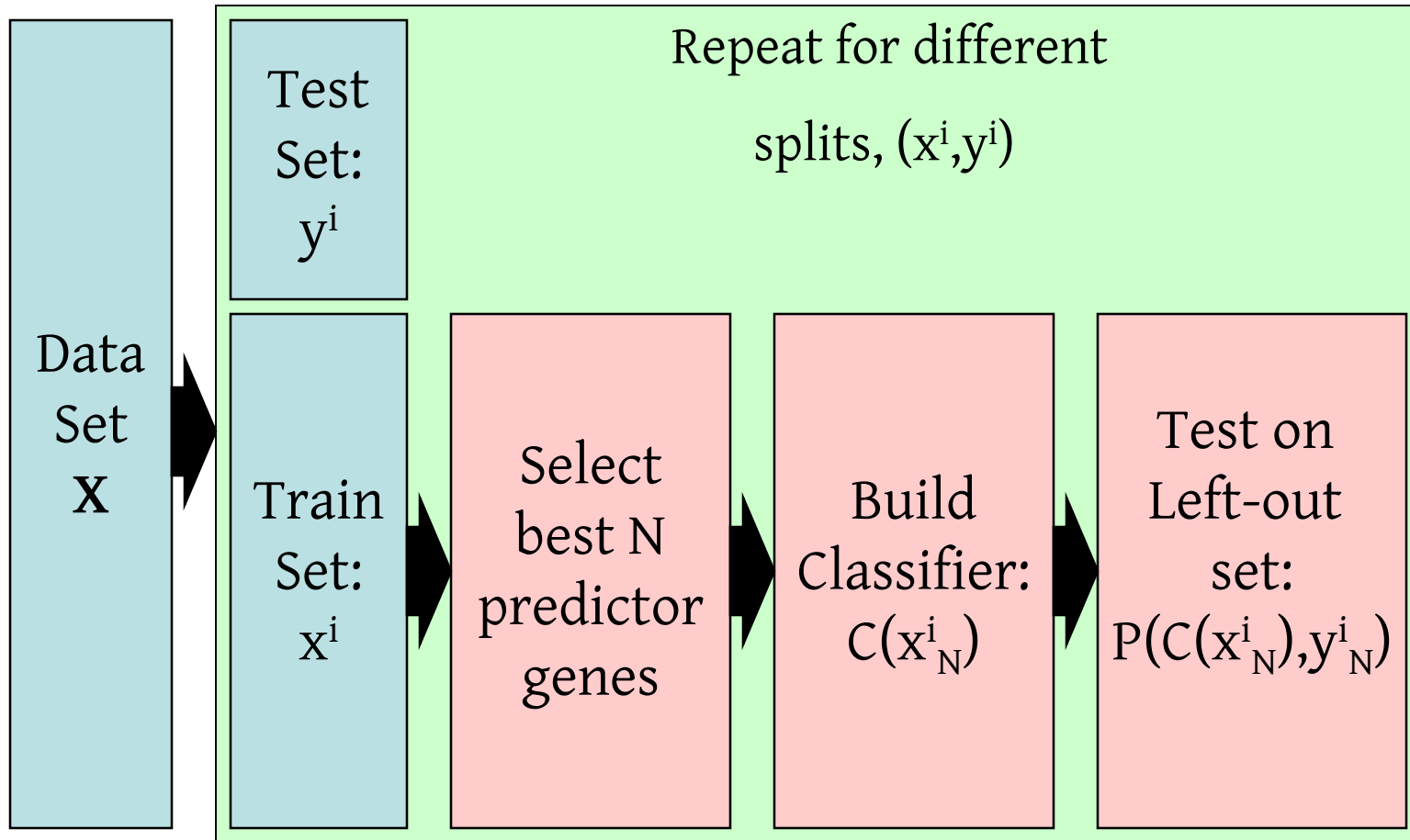
Selection bias...

- Guyon et al. (2002). Machine Learning **46**, 389 – 422.
- Ambroise and McLachlan (2002). PNAS **99**, 6562-6566.

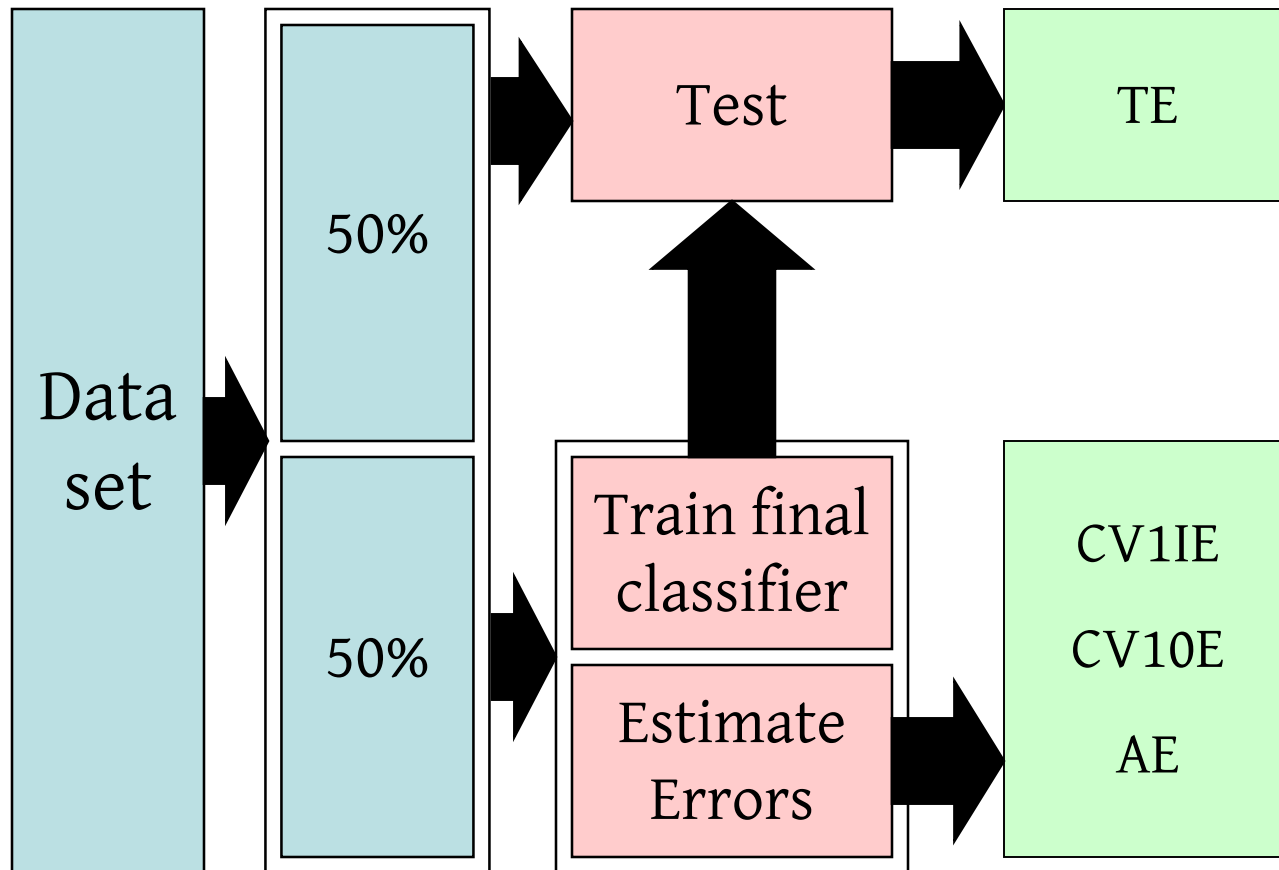
Biased selection



Unbiased selection

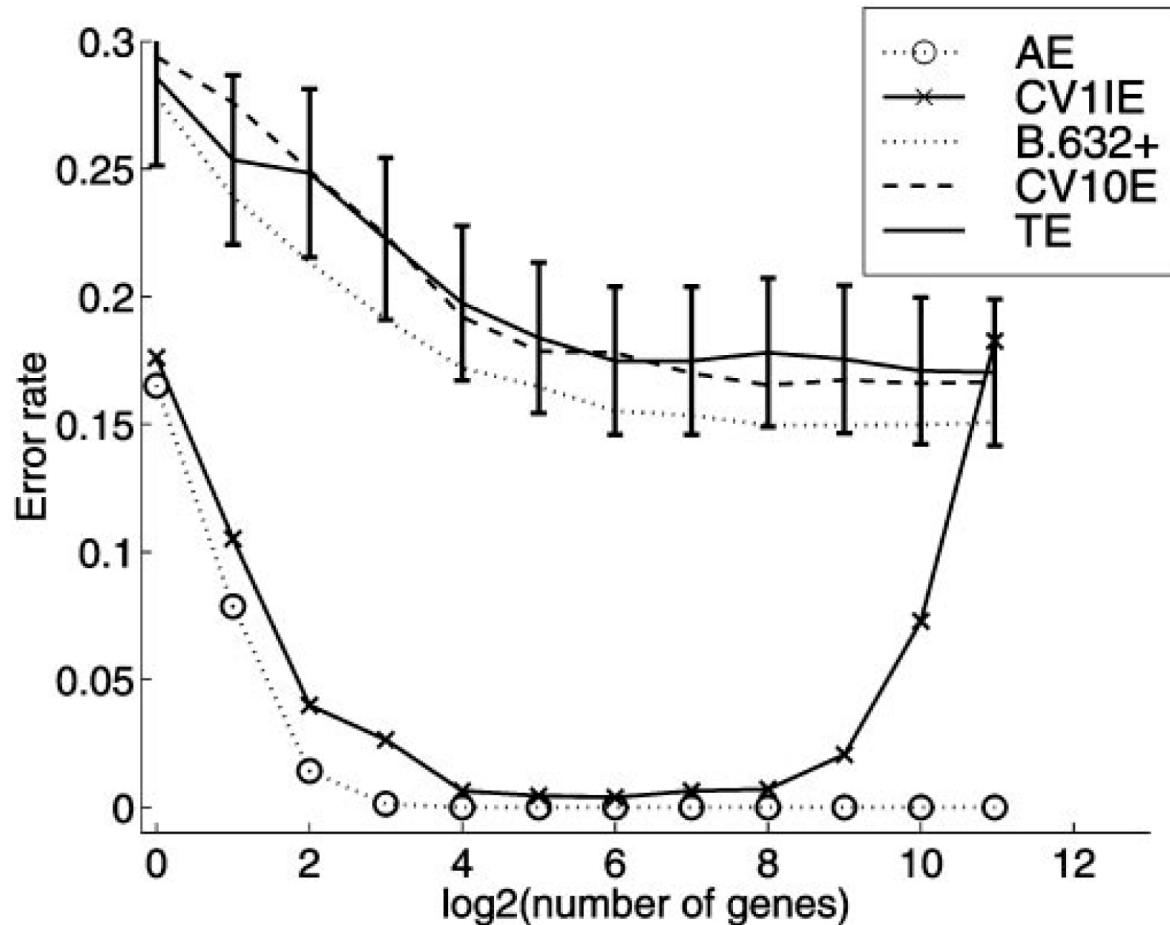


Ambroise & McLachlan experiments



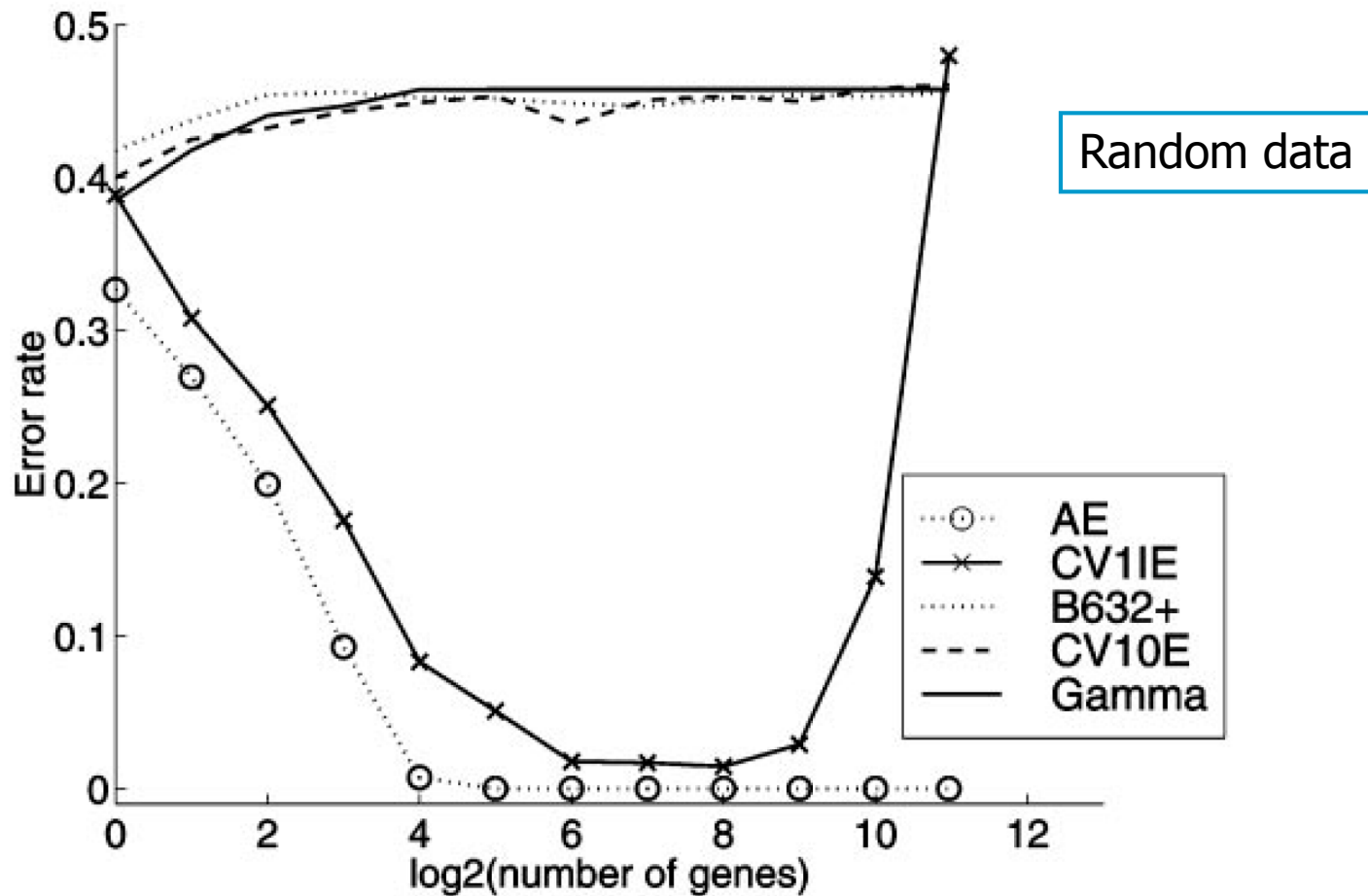
Ambrose & McLachlan experiments

Colon vs. normal data



Ambrose & McLachlan experiments

Random data



Cross-validation

- Remember:

Note:

we should never use the training set to calculate performance; this will give a biased estimate!

- for small sample size: use cross-validation
- Cross-validation should be applied to *every choice* made, including:
 - the number of features to use
 - the features to use
 - the type of classifier to use
 - ...

Feature selection: summary

- Feature selection can improve performance and help interpretation
- Requirements: a criterion and a search algorithm
- Methodology (cross-validation) is very important, especially for RNAseq data ('p >> n')
- There seems to be some evidence that the simplest methods (individual selection) work best

Shrinkage

- Feature selection: selects a subset of features (1/0)
- Feature extraction: combinations of features are constructed based on variance and accuracy arguments
- Regularization 1: 'shave off' genes based on individual quality and control degree of 'shaving' with error
- Regularization 2: combines accuracy (error) and penalty on large weights (= simple models) in one criterion.

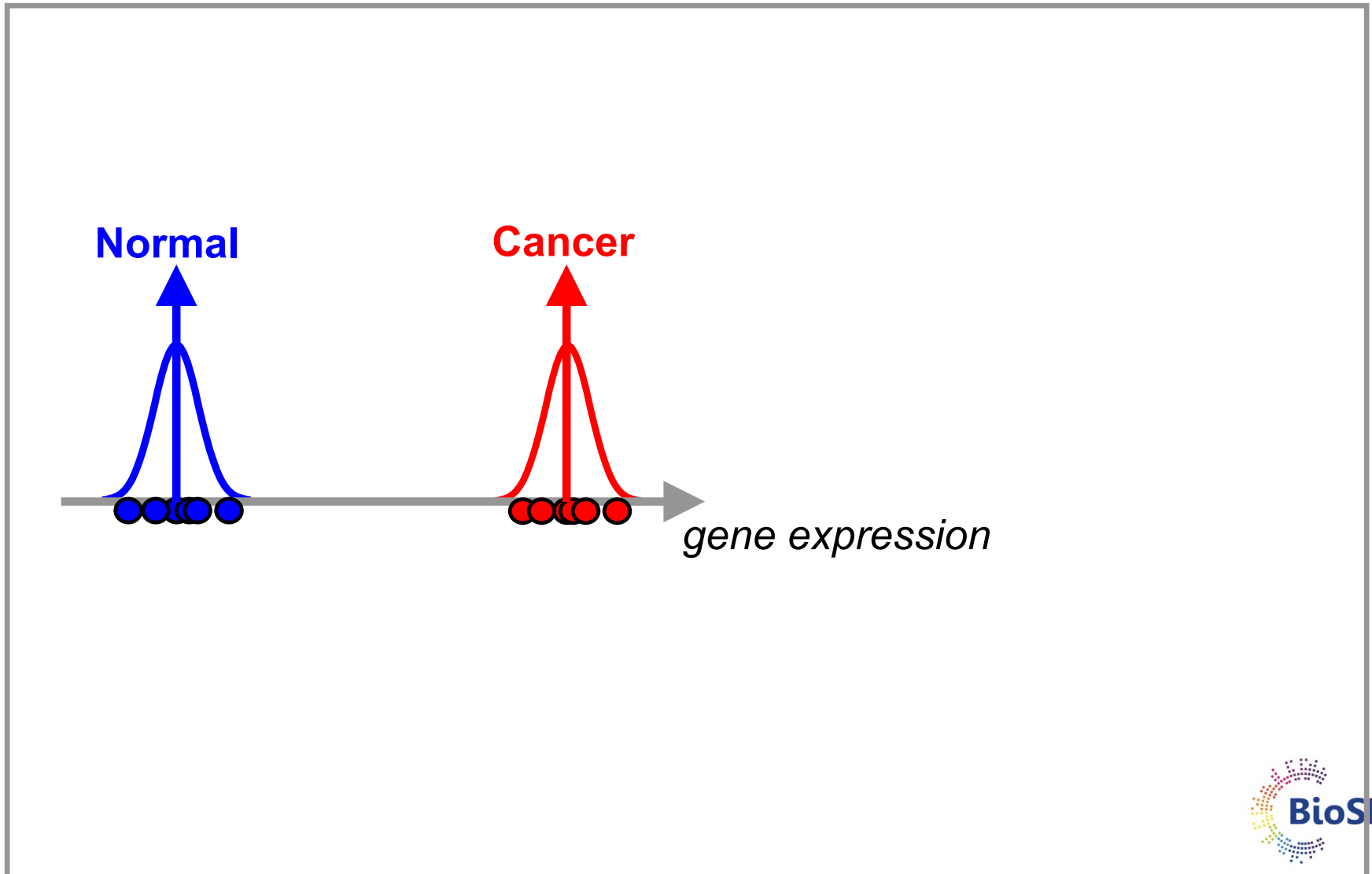
Shrunken centroids

- Same principle as forward filtering
- Genes are evaluated *individually*
- BUT, do not start with the best and keep adding;
- RATHER, start removing worst genes from the back
- In PAM* genes can participate ‘partially’, in forward filtering a gene is either 100% in or out.

* PAM: Prediction analysis of micro-arrays; R. Tibshirani, T. Hastie, B. Narasimhan and G. Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. PNAS 99(10):6567-6572, 2002.

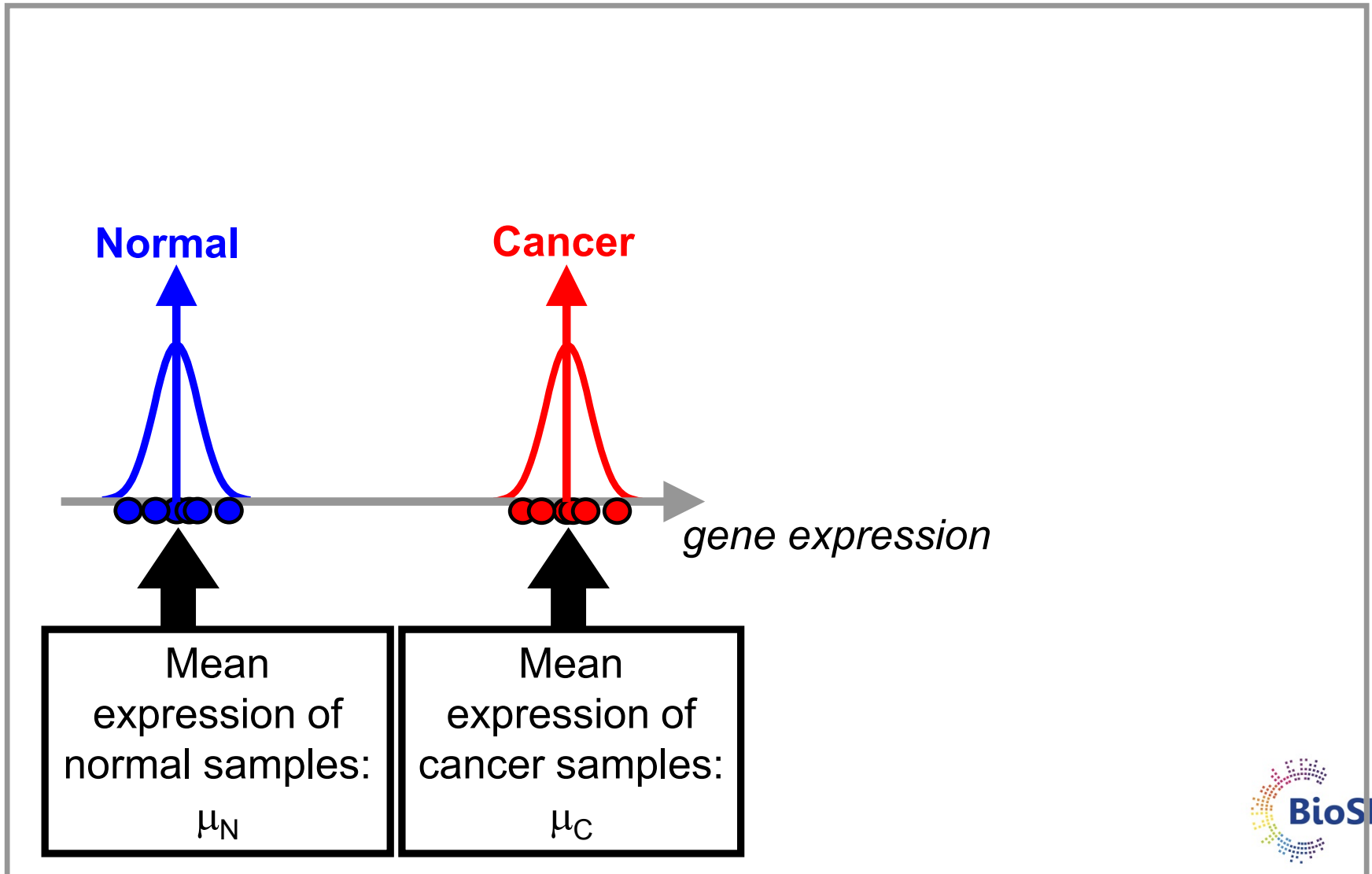
Shrunken centroids (1)

Step 1: Compute class centroids per gene



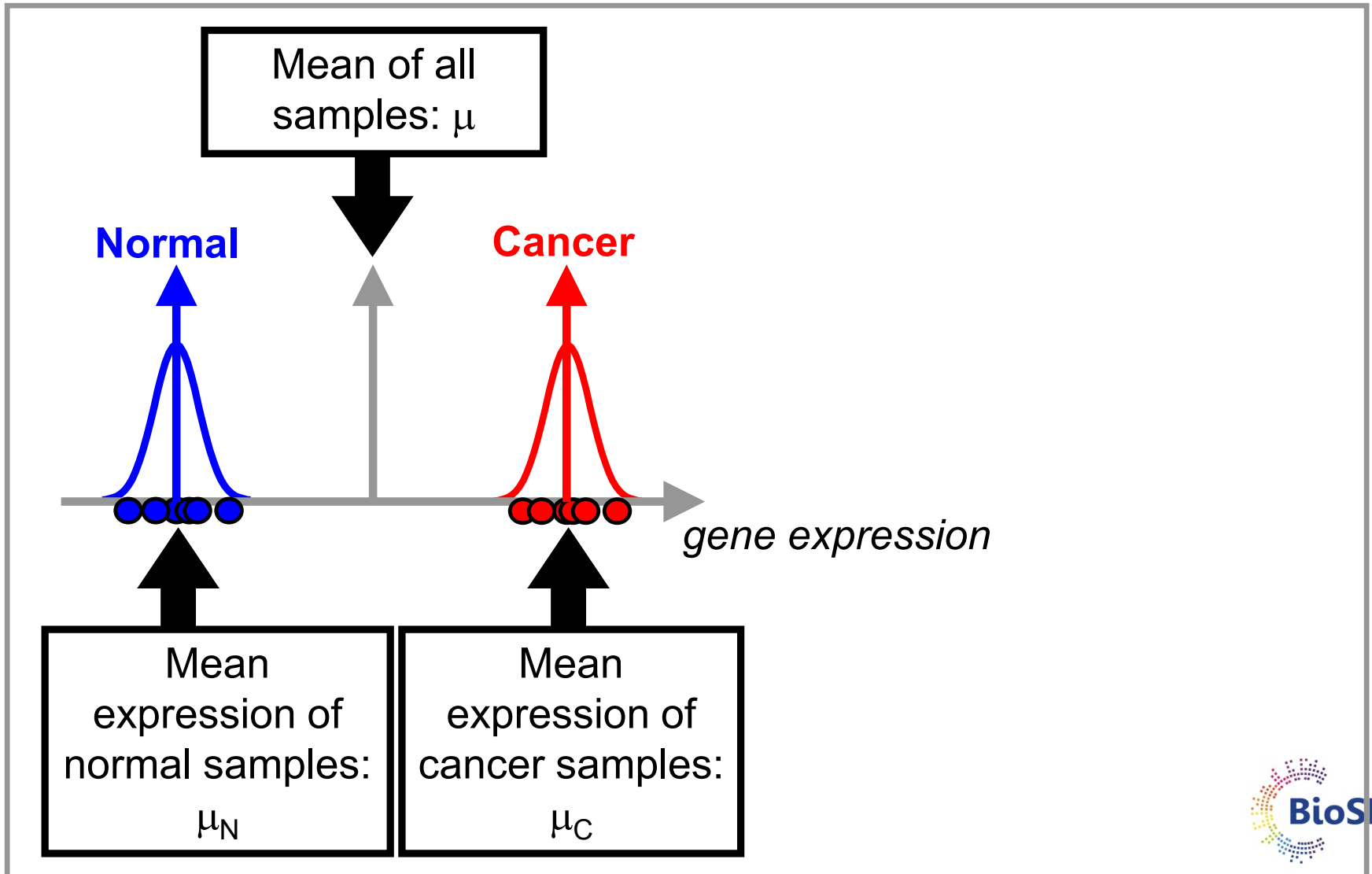
Shrunken centroids (2)

Step 1: Compute class centroids per gene



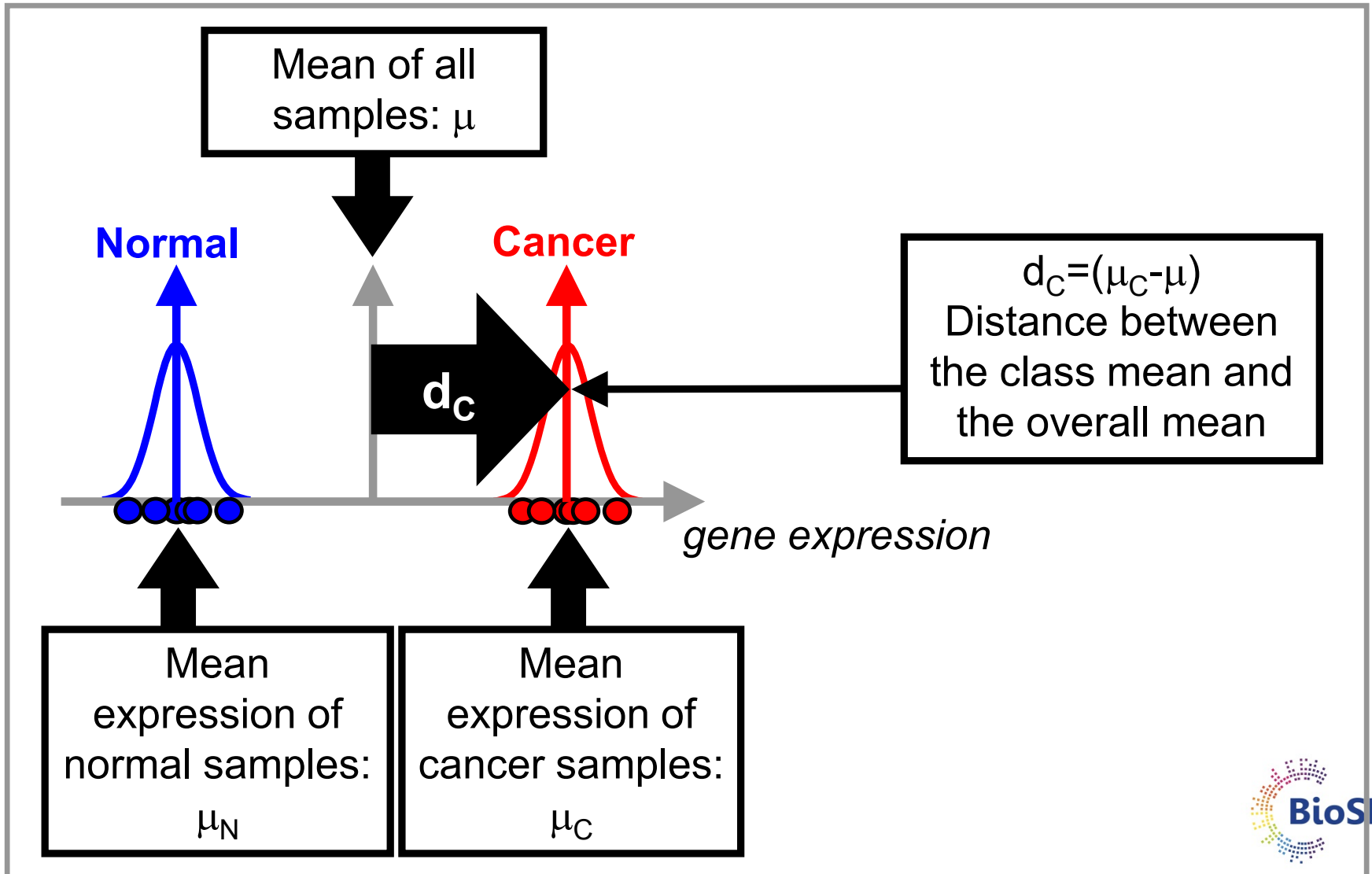
Shrunken centroids (3)

Step 2: Compute overall centroids per gene



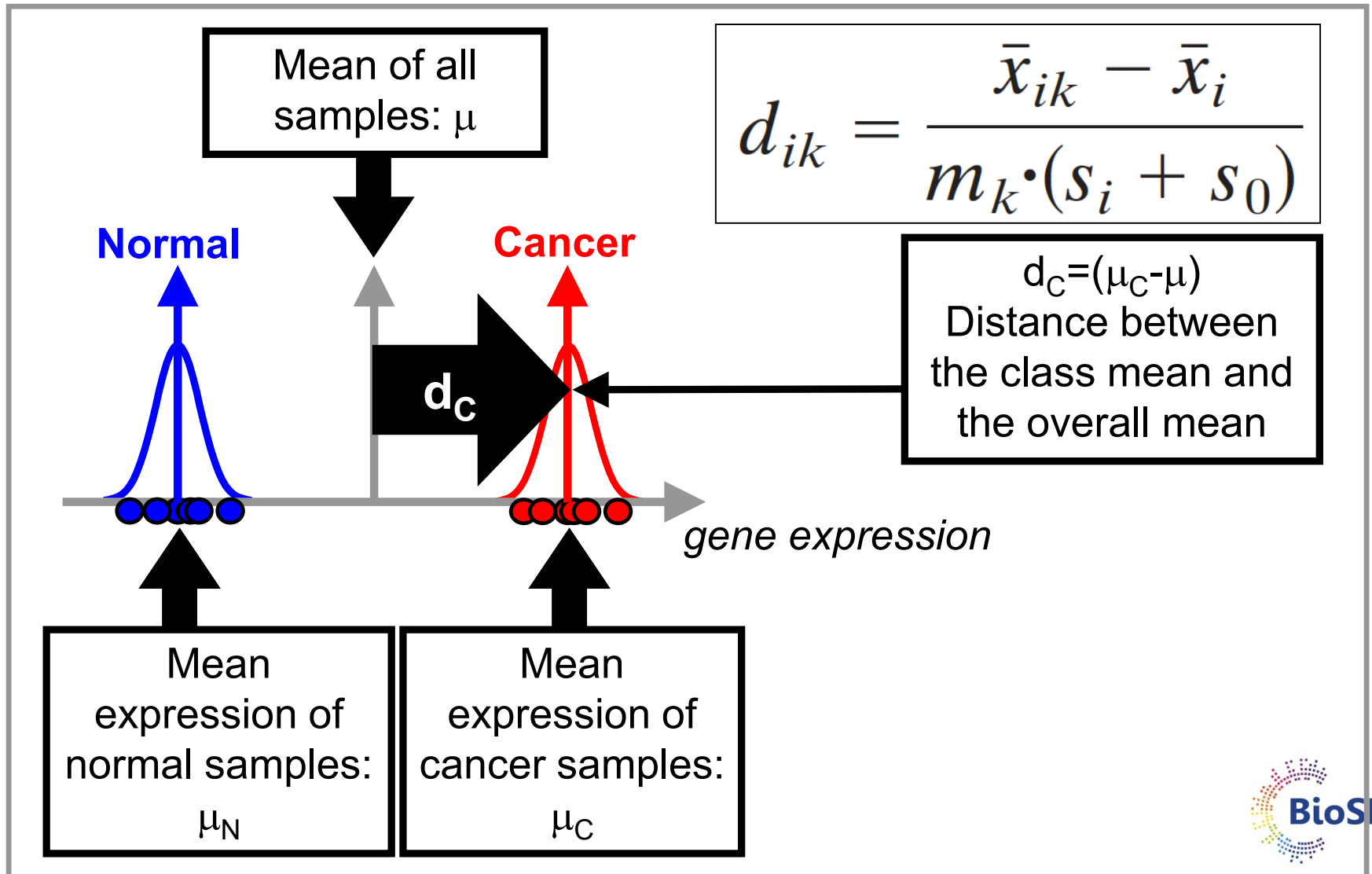
Shrunken centroids (4)

Step 3: Compute d per gene



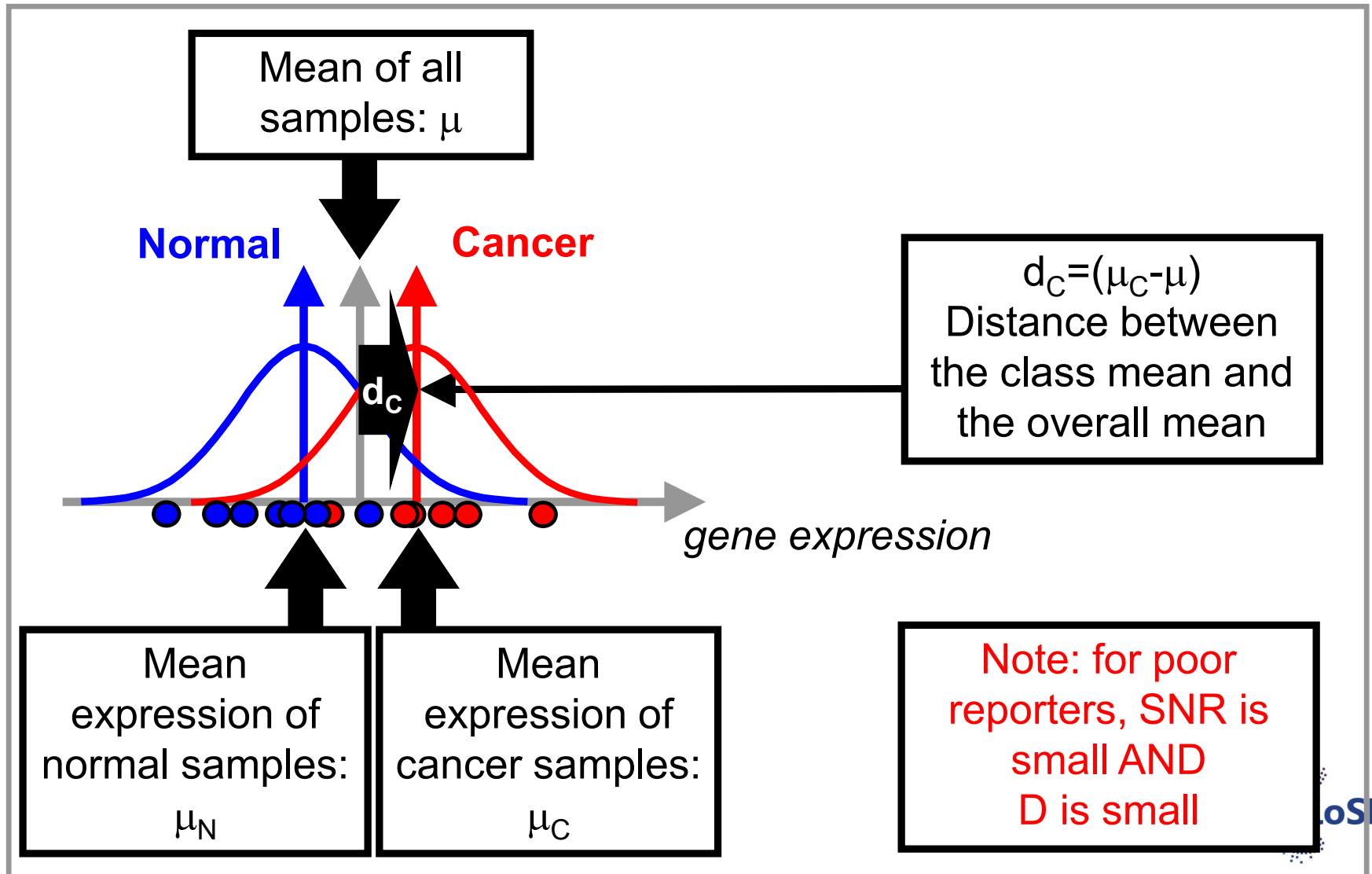
Shrunken centroids (4)

Step 3: Compute d per gene



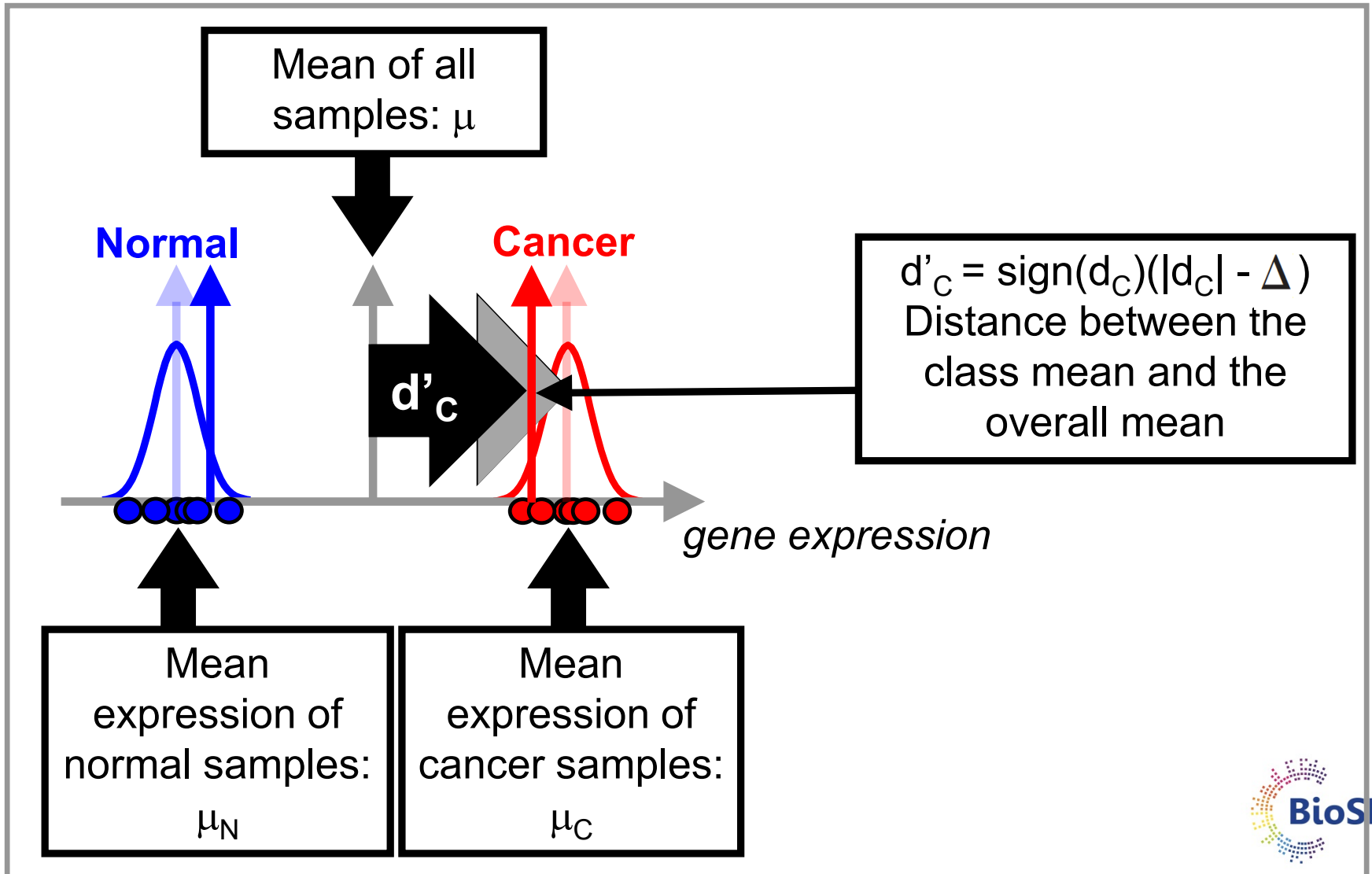
Shrunken centroids (5)

Step 3: Compute d per gene



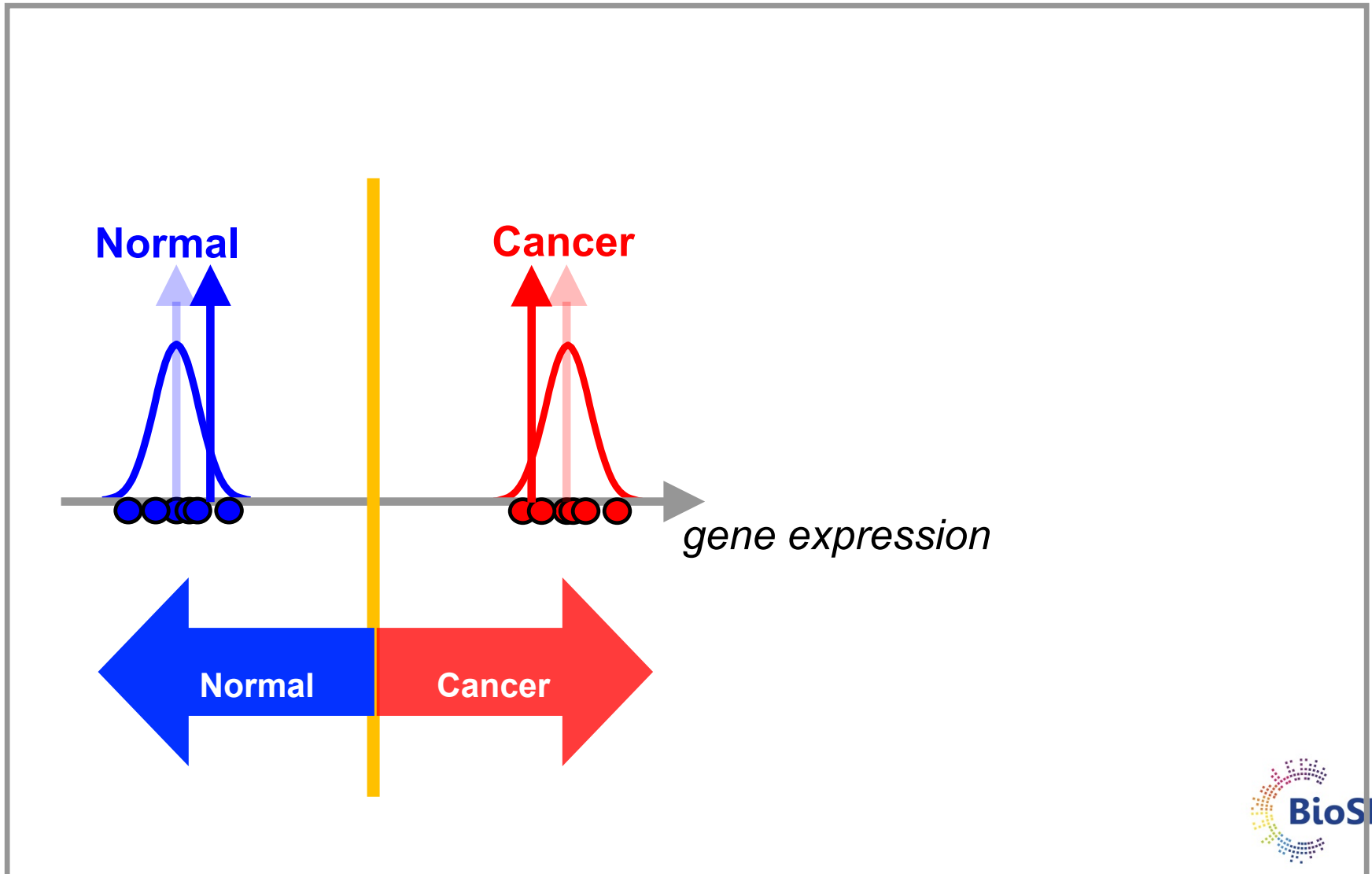
Shrunken centroids (6)

Step 4: Shrink the centroids



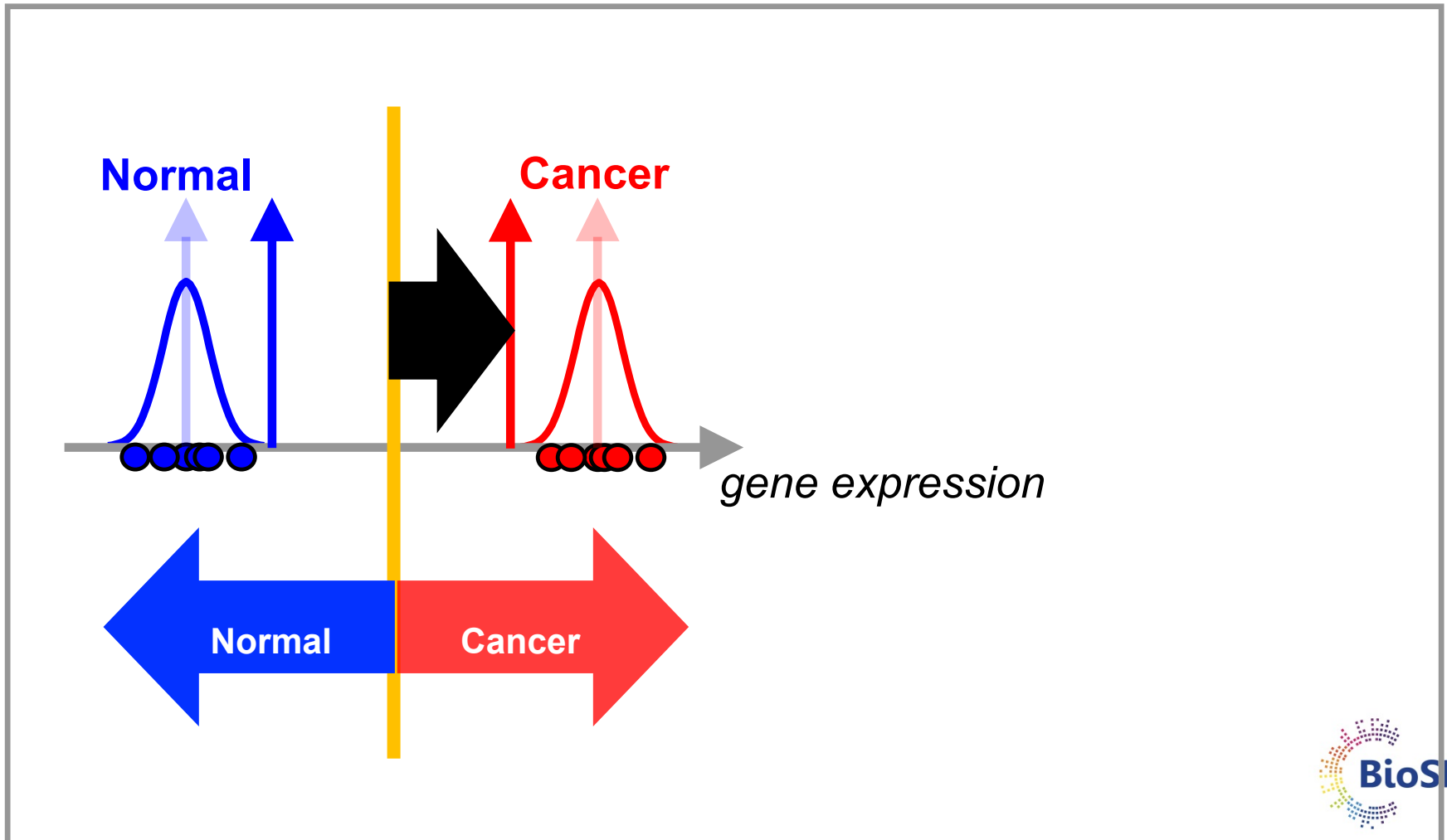
Shrunken centroids (7)

Step 5: Classify with shrunken centroids / perf.



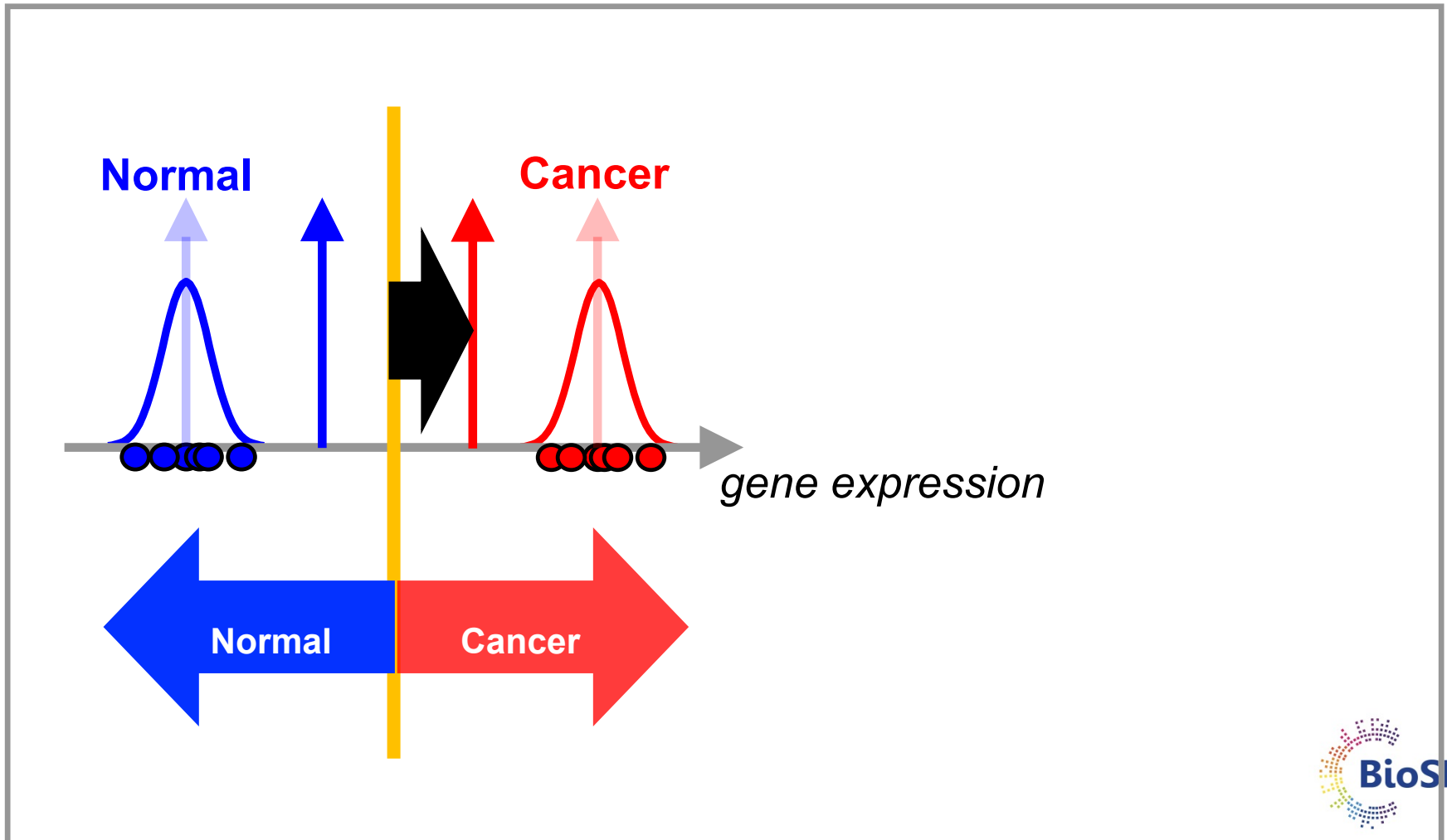
Shrunken centroids (8)

Repeat shrinking and evaluation of classifier till all genes shrunk away



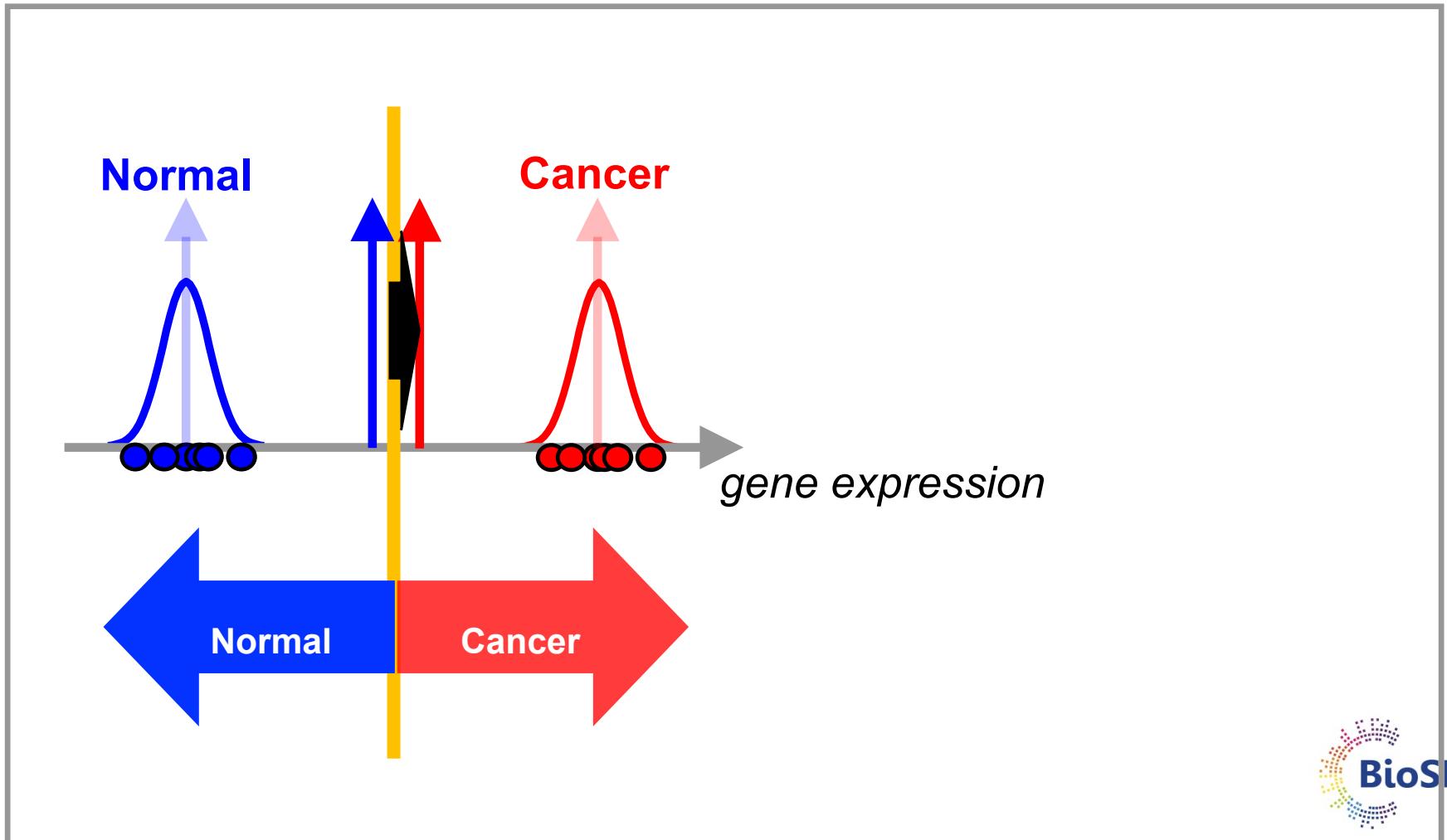
Shrunken centroids (8)

Repeat shrinking and evaluation of classifier till all genes shrunk away



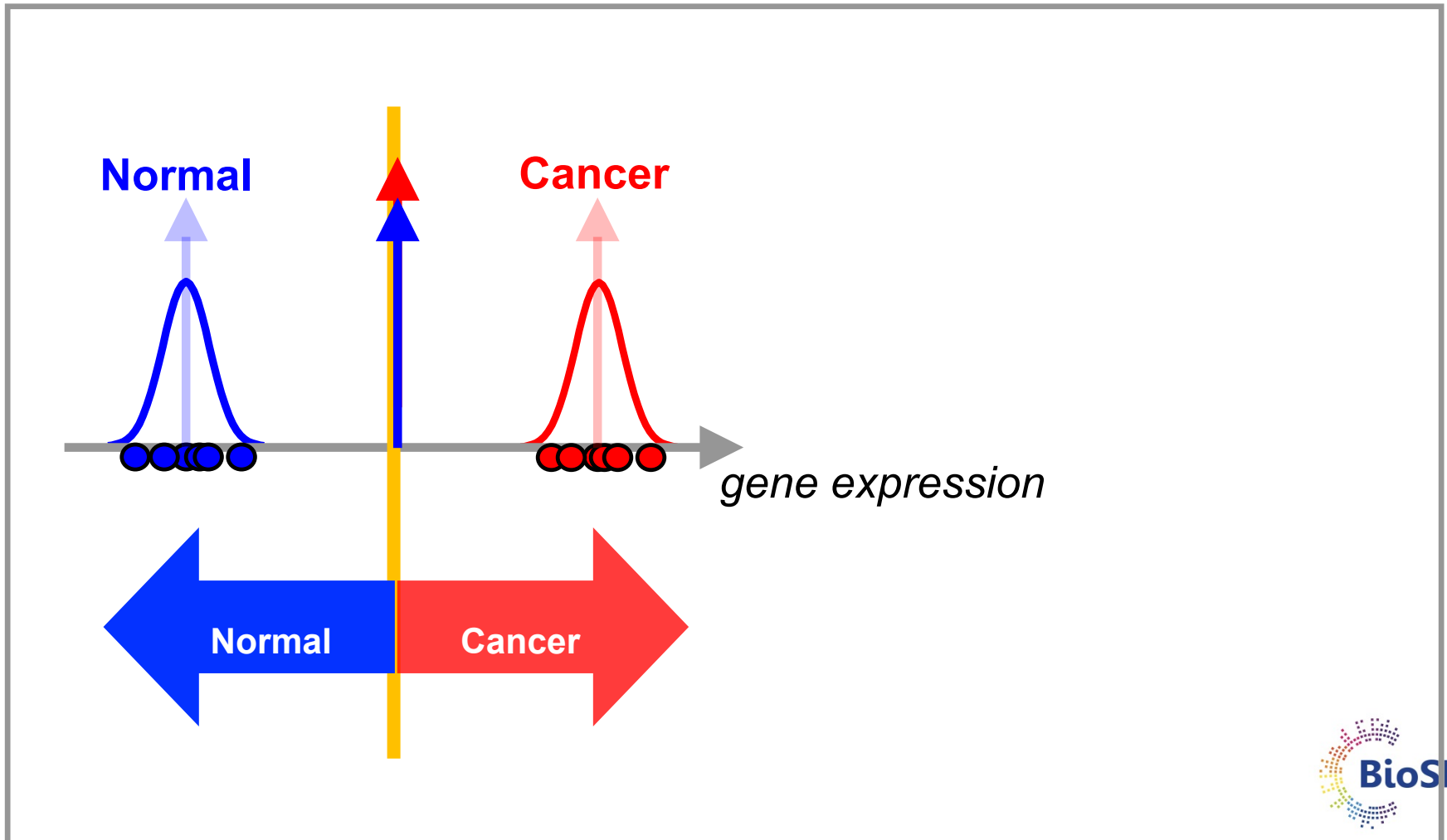
Shrunken centroids (8)

Repeat shrinking and evaluation of classifier till all genes shrunk away

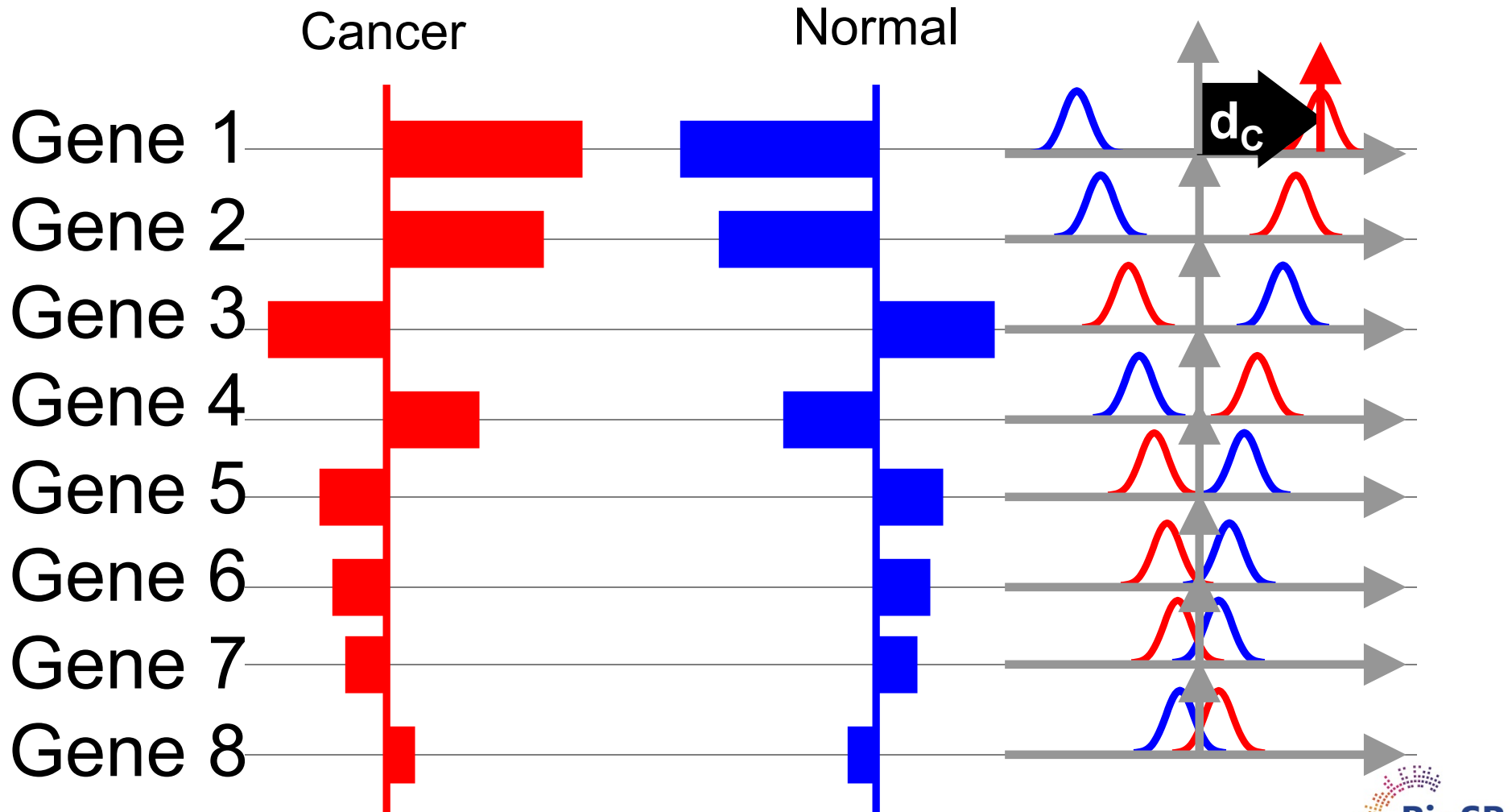


Shrunken centroids (8)

Repeat shrinking and evaluation of classifier till all genes shrunk away

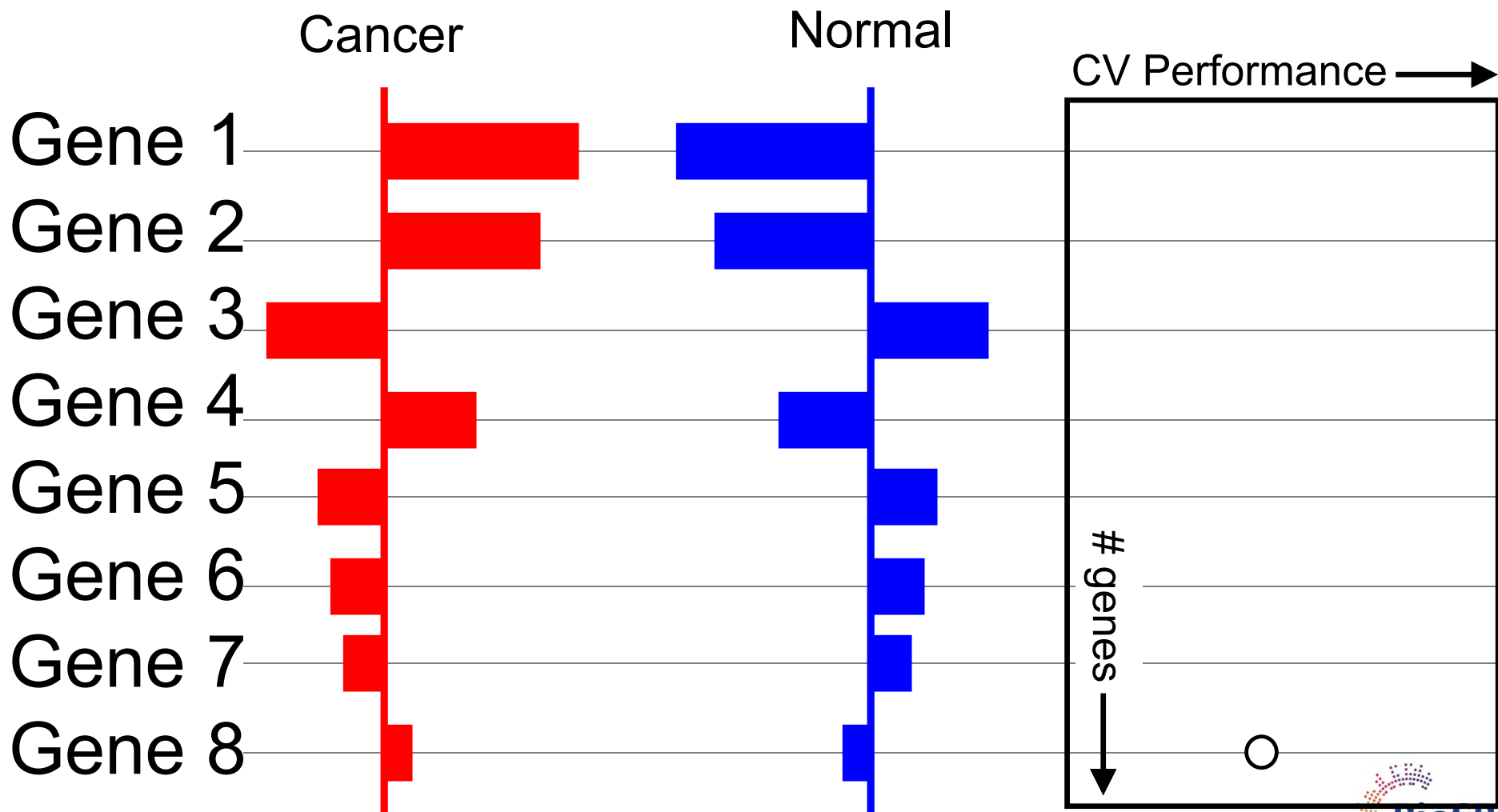


Shrunken centroids: selecting the genes



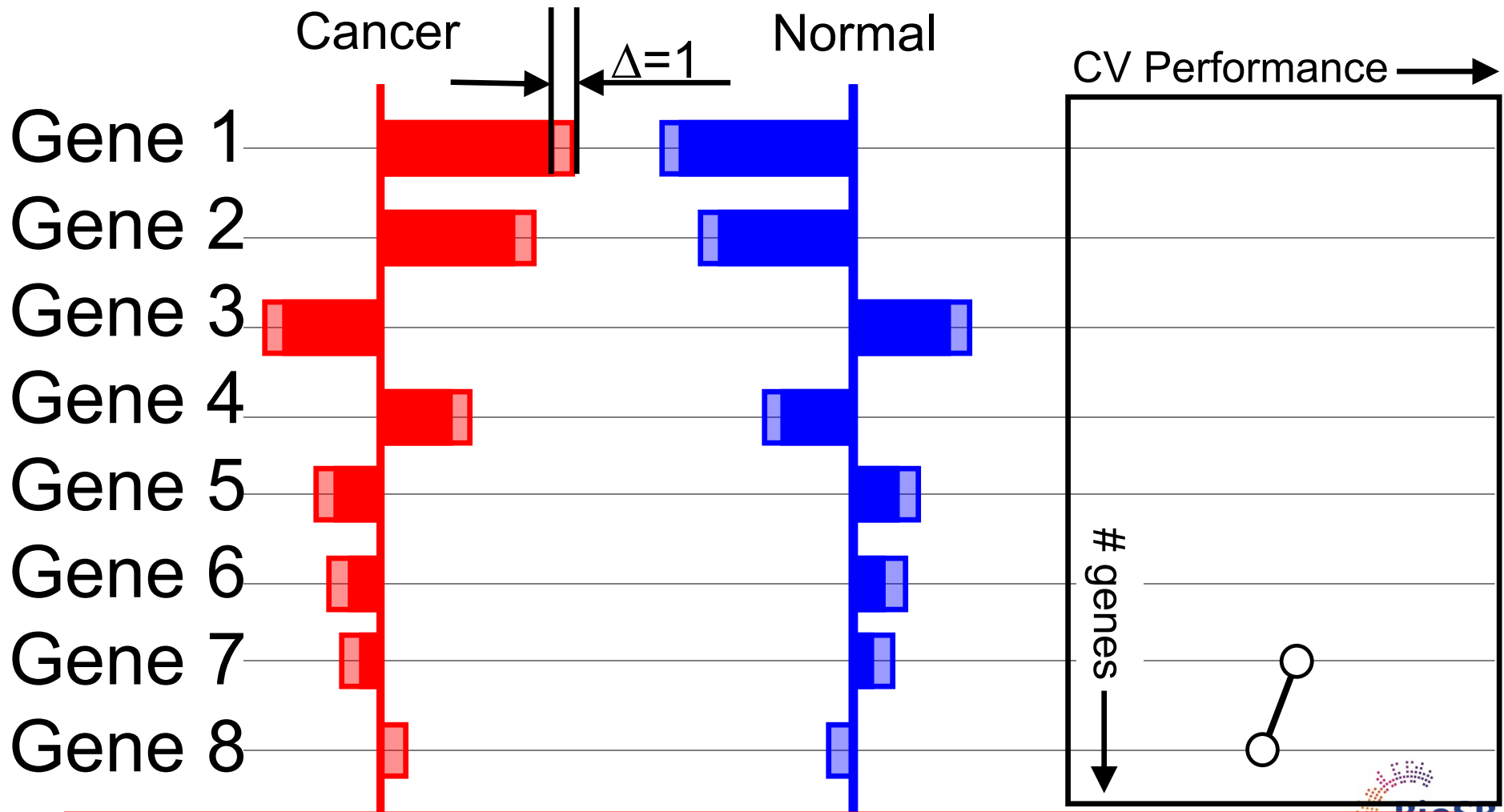
Genes sorted based on D-measure: best to worse

Shrunk centroids: selecting the genes



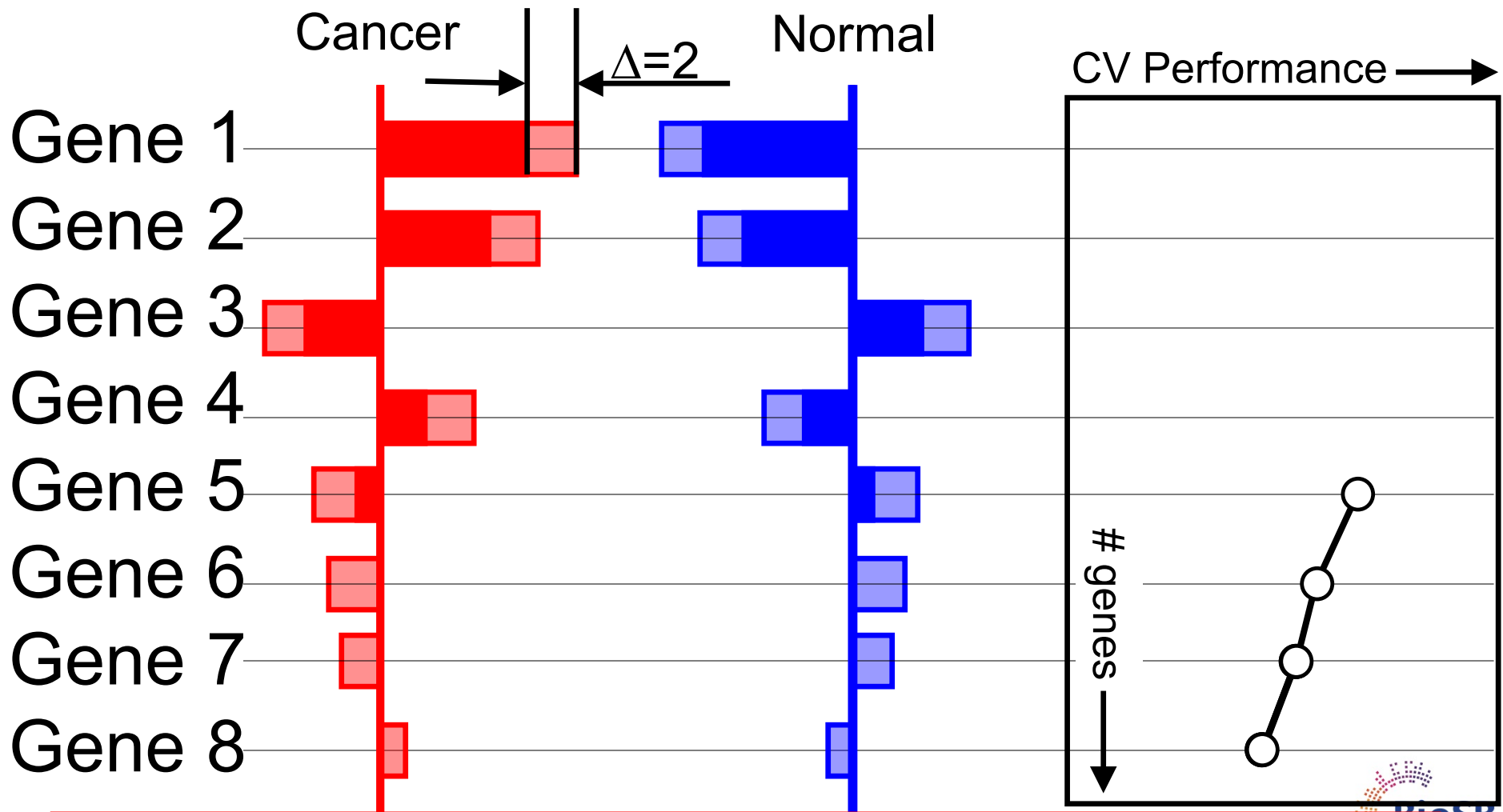
Train classifier on all 8 genes; estimate CV performance

Shrink all d by $\Delta=1$: reduce length by 1



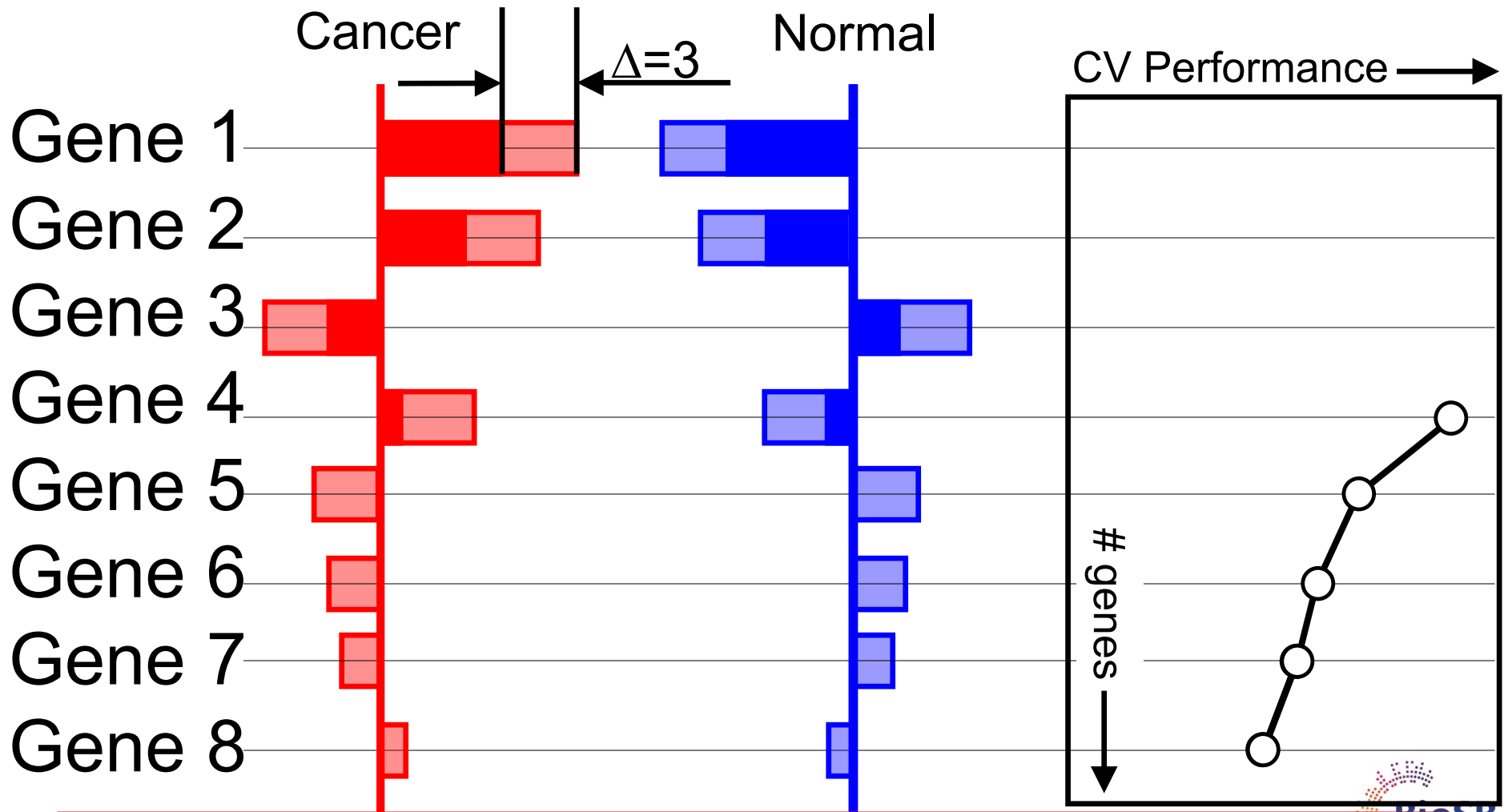
Train classifier on 7 genes ($d>0$); estimate CV performance

Shrink all d by $\Delta=2$: reduce length by 2



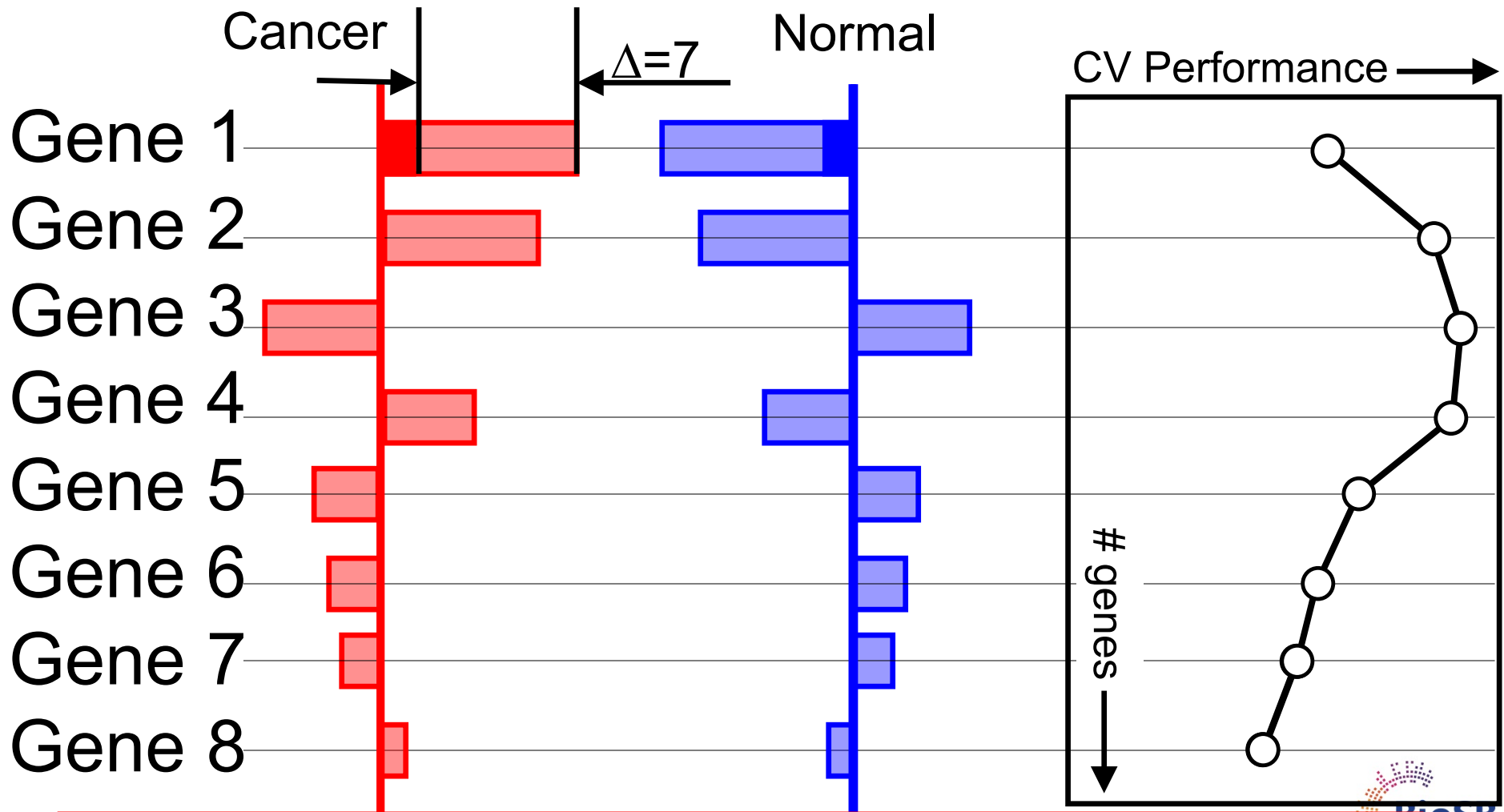
Train classifier on 5 genes ($d>0$); estimate CV performance

Shrink all d by $\Delta=3$: reduce length by 3



Train classifier on 4 genes ($d>0$); estimate CV performance

Shrink all d by $\Delta=7$: reduce length by 7



Train classifier on 1 gene ($d>0$); estimate CV performance

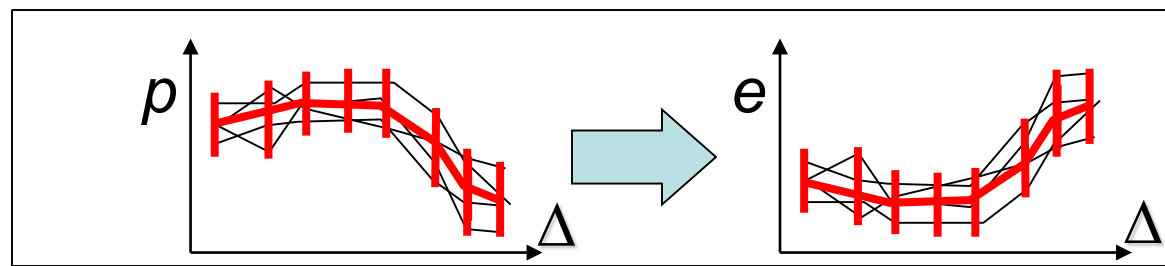
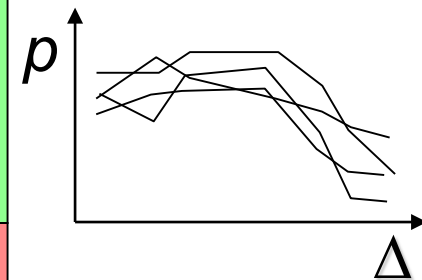
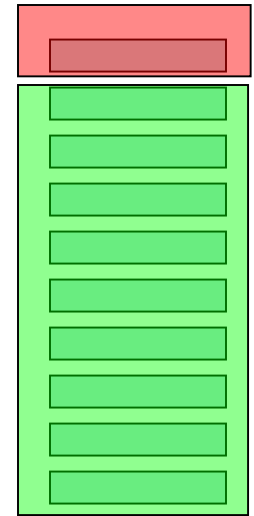
Determining the optimal Δ

1. Split the data (X) in 10 equal parts (x_1, \dots, x_{10})
2. For each of the 10 folds ($i=1, 2, \dots, 10$)
3. On the training set ($X \setminus x_i$)

1. Compute the class and overall centroids
2. For a range of Δ ($\Delta = [0, 0.5, \dots, 7]$)
 - i. Shrink d for all genes
 - ii. Compute 'shrunk centroids' on training set
 - iii. Test the resulting classifier on the test set (x_i)

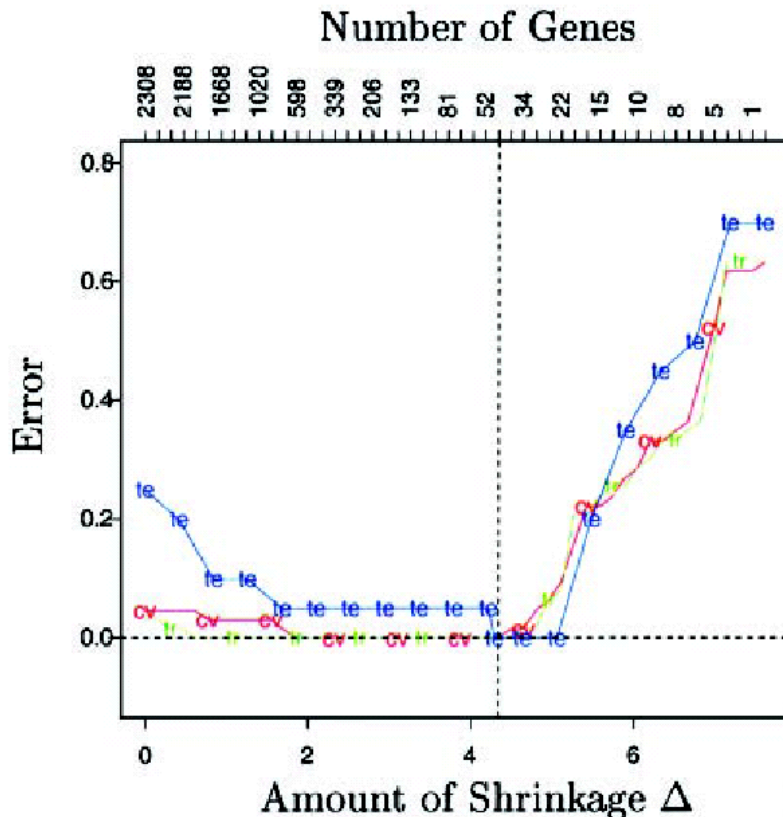
3. Result: 10 Curves of performance vs. Δ

4. Average all 10 curves and compute std. dev. at each Δ
5. Pick the Δ where the performance is maximal (error min.)



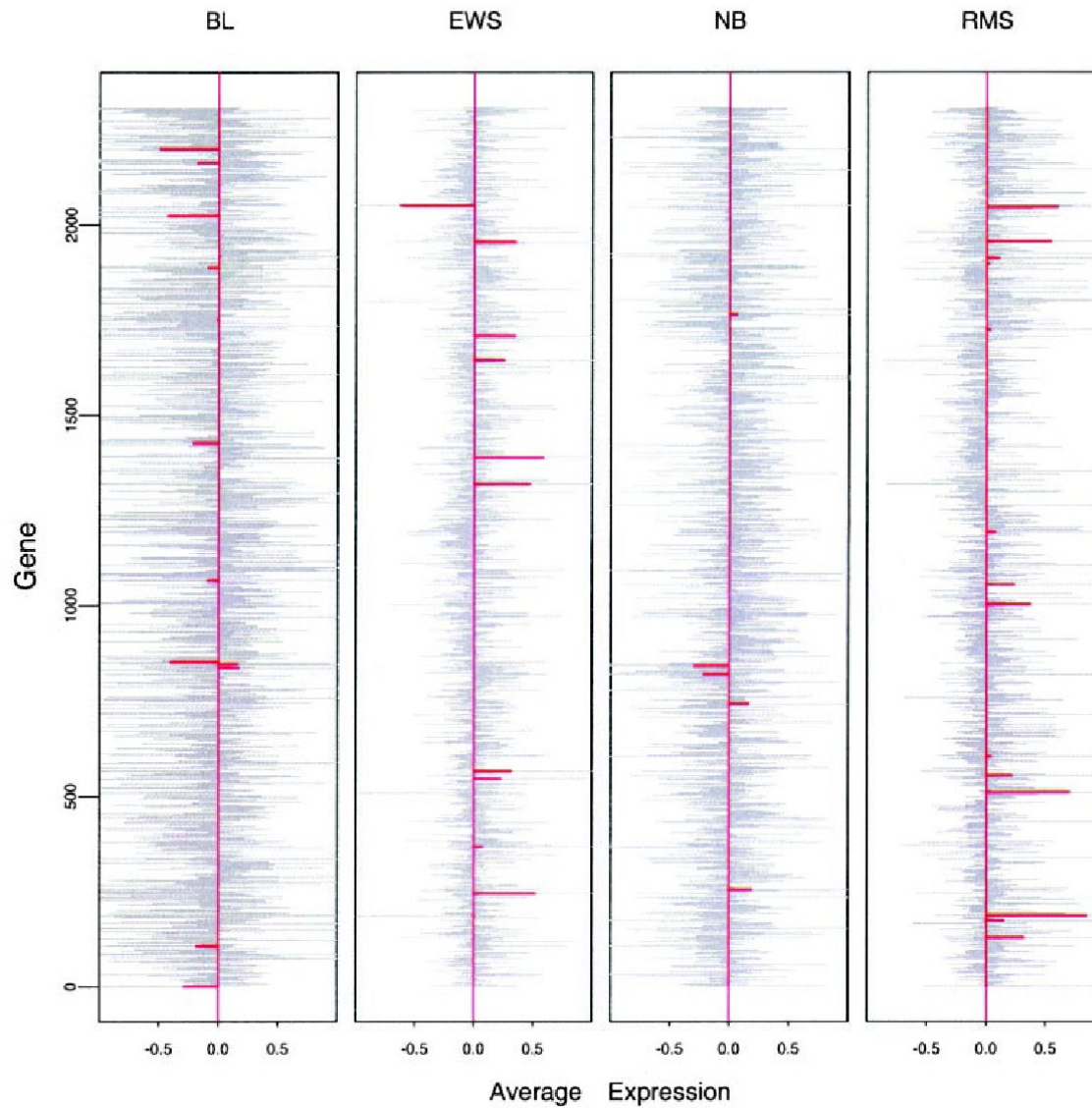
PAM



- For the Khan dataset; 4 classes: BL, EWS, NB, RMS
- At optimal Δ : 43 genes *not* shrunk away



Neuroblastoma (NB)
Rhabdomyosarcoma (RMS)
Burkitt lymphoma (BL)
Ewing family of tumors (EWS),

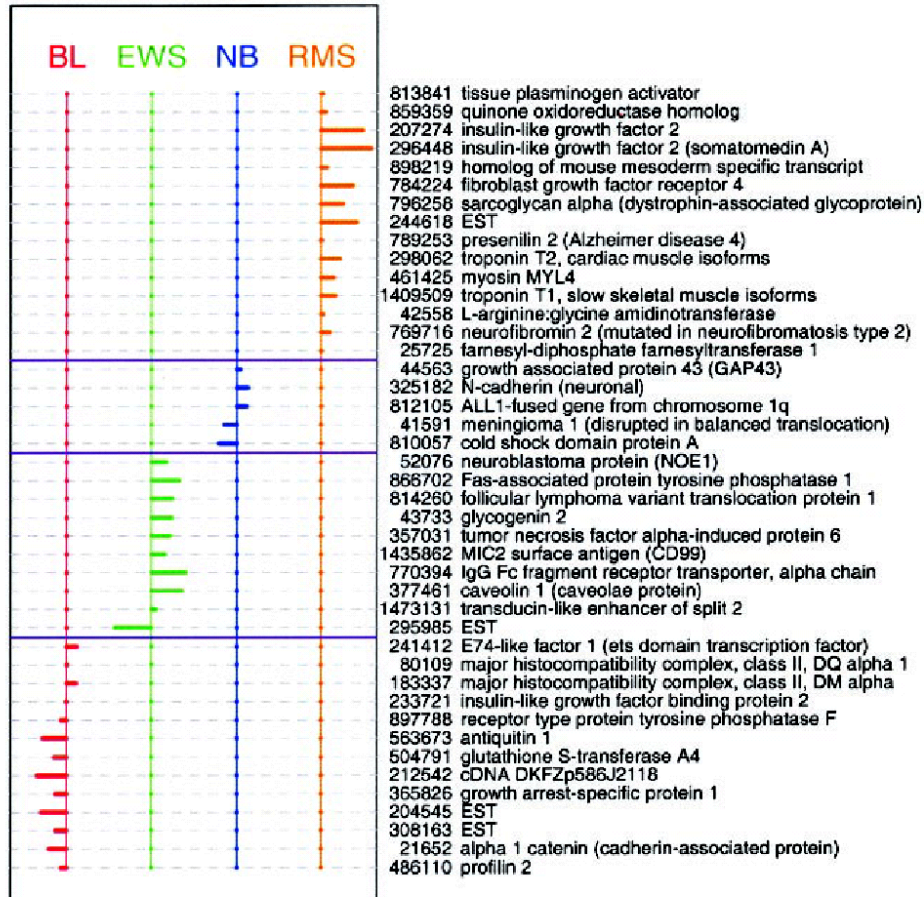
PAM (2)



 shrunk
 unshrunk

PAM (3)

At optimal Δ : 43 genes *not* shrunk away



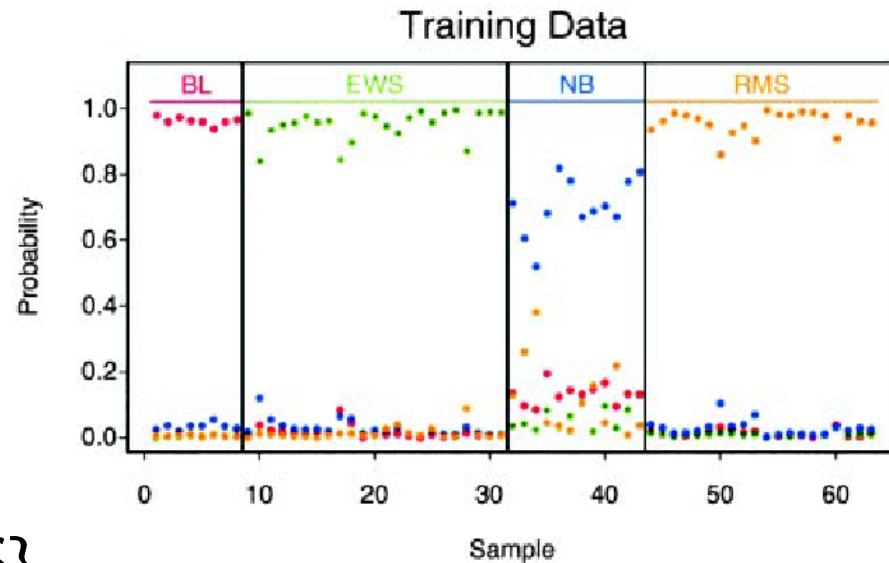
- Neuroblastoma (NB)
- Rhabdomyosarcoma (RMS)
- Burkitt lymphoma (BL)
- Ewing family of tumors (EWS),

R. Tibshirani *et al.* (2002) PNAS 99(10):6567-6572, 2002.

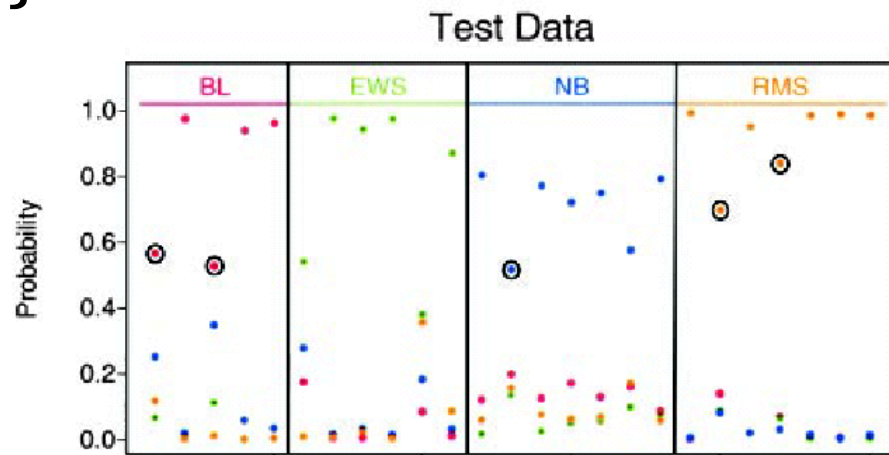
Scoring samples by posterior prob's

$$\hat{p}(k|x^*)$$

$$k = \{BL, EWS, NB, RMS\}$$



$$\hat{p}(k|x^*)$$



Shrinkage

- PAM determines a weight for every gene based on the predictive capacity of the gene (type of t-statistic)
- This weight determines role of the gene in a DLDA classifier (weights can be zero, i.e. no participation)
- Weights of all genes are shrunk by the same amount (Δ)
- PAM computes effect of shrinkage on error rate, and chooses shrinkage (=number of genes) with lowest error
- PAM: implicit simultaneous error and complexity minimisation
- Other approach: regularisation, combine error and penalty for number of genes explicitly

Shrinkage (2)

- Model: $y = \beta_0 + \sum_{i=1}^p \beta_i x_i + \varepsilon$

- Penalised (aka regularised) least squares:

- Ridge regression:

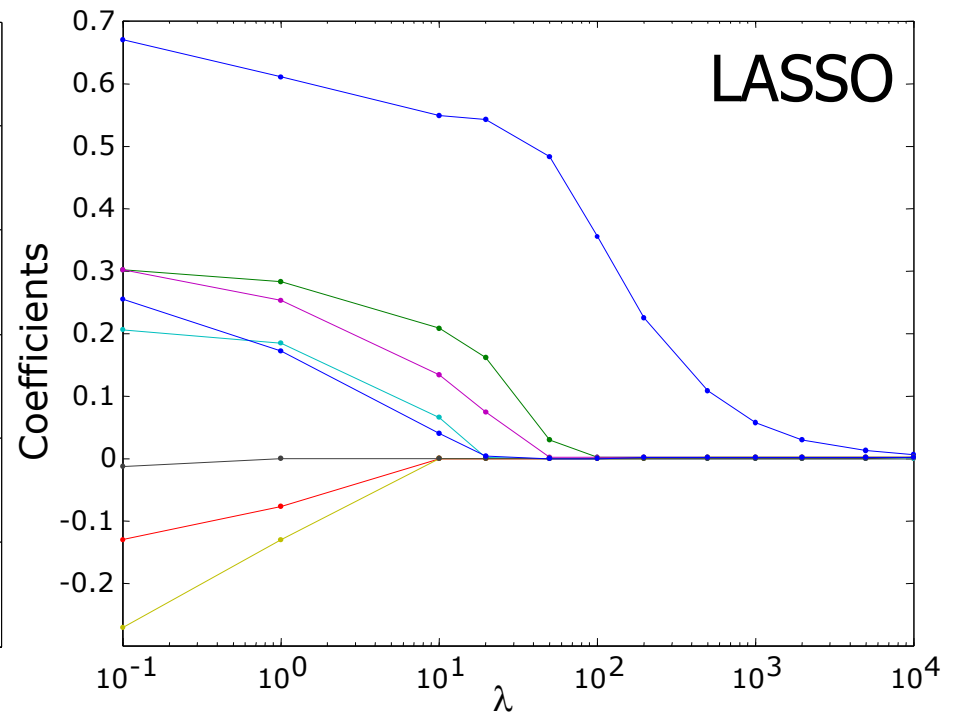
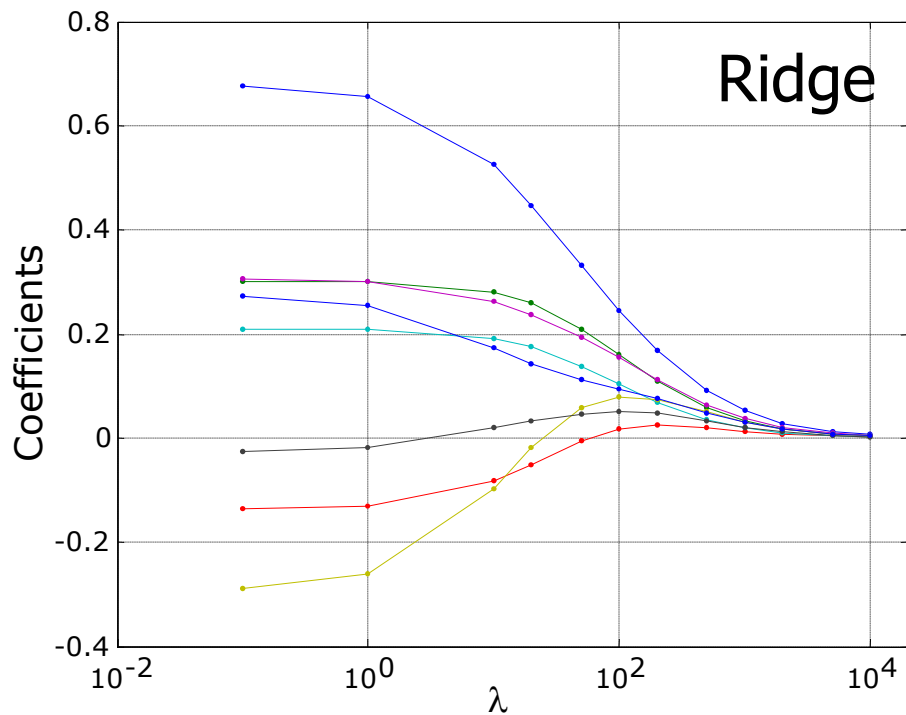
$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta}} \left[\sum_{j=1}^n \left(y_j - \beta_0 - \sum_{i=1}^p \beta_i x_{j,i} \right)^2 + \lambda \sum_{i=1}^p \beta_i^2 \right]$$

- LASSO: minimise

$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta}} \left[\sum_{j=1}^n \left(y_j - \beta_0 - \sum_{i=1}^p \beta_i x_{j,i} \right)^2 + \lambda \sum_{i=1}^p |\beta_i| \right]$$

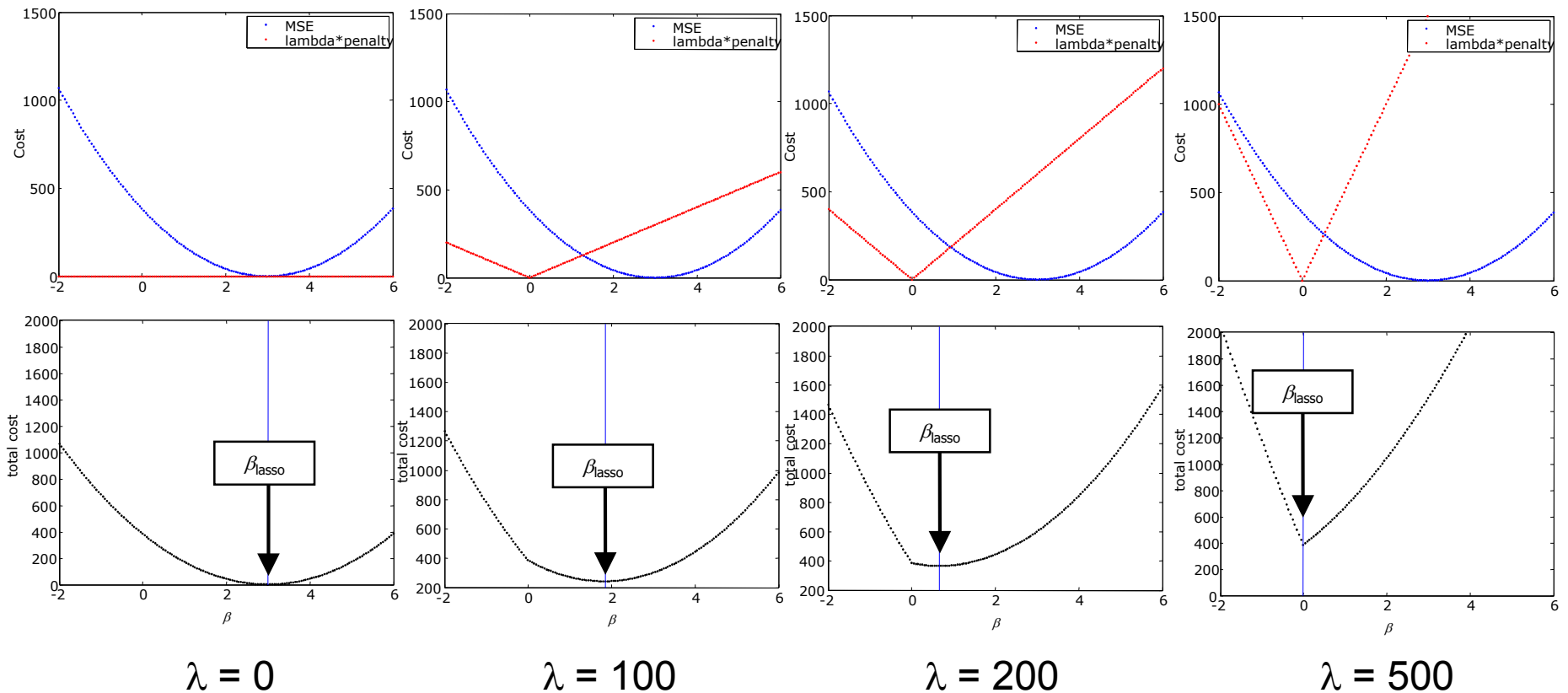
LASSO

- Difference seems small, but effect of LASSO is that genes are no longer used (like in PAM!)



LASSO (2)

- Example: true function $y = 3x$

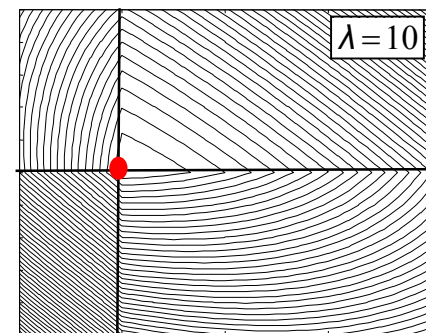
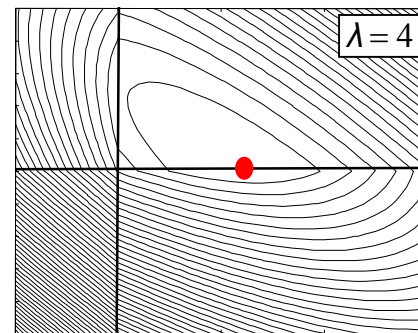
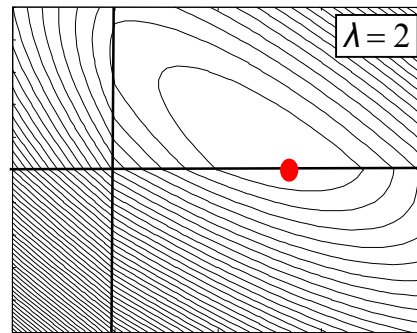
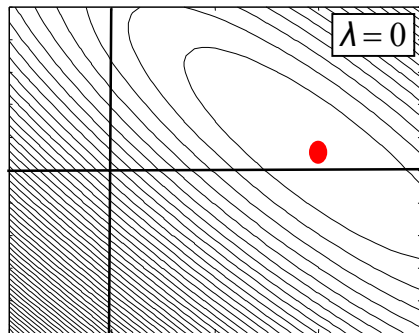
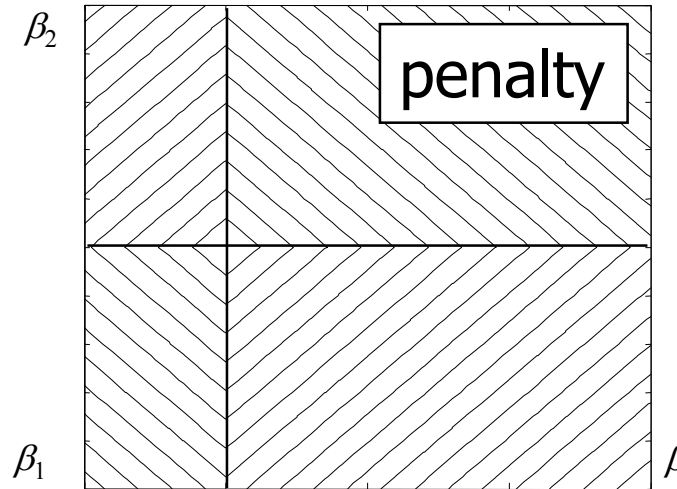
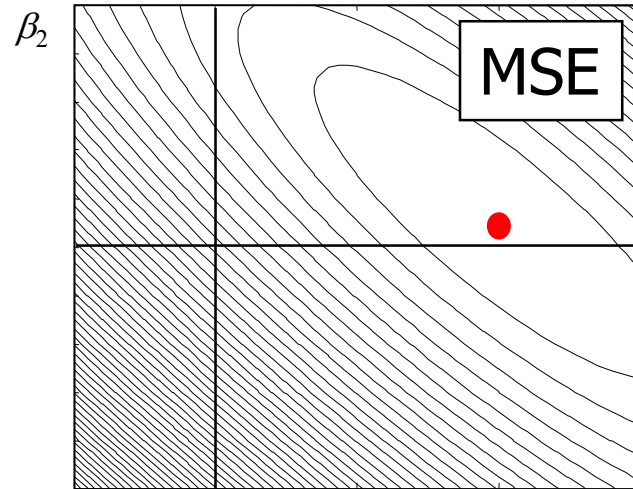


LASSO (3)

Example:

$$y = 0.1x_1 + 0.01x_2$$

● Optimum



β_1 →

Final summary

- Feature extraction:
 - Linear:
 - PCA,
 - Fisher
 - Non-linear
 - MDS
- Feature selection:
 - Criteria
 - search algorithms
 - forward,
 - backward,
 - branch & bound.
- Sparse classifiers:
 - Ridge,
 - LASSO