



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

- Dimensionality reduction
 - Feature extraction
 - Feature selection
 - Regularized classifiers

Overview

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

Overview

- **Feature selection**
 - Criteria
 - Search algorithms
 - n-best selection
 - Forward selection
 - Backward selection
 - ...

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?

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 (features) are useful and which are not (reduce redundancy)
3. **Visualisation**

Curse of dimensionality

- **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)

Curse of dimensionality

- **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

* measurements

Curse of dimensionality

- **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!

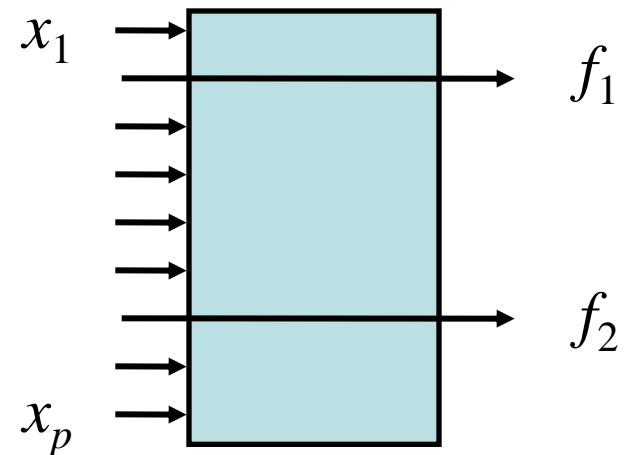
* measurements

Curse of dimensionality

- **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, $n = 38$

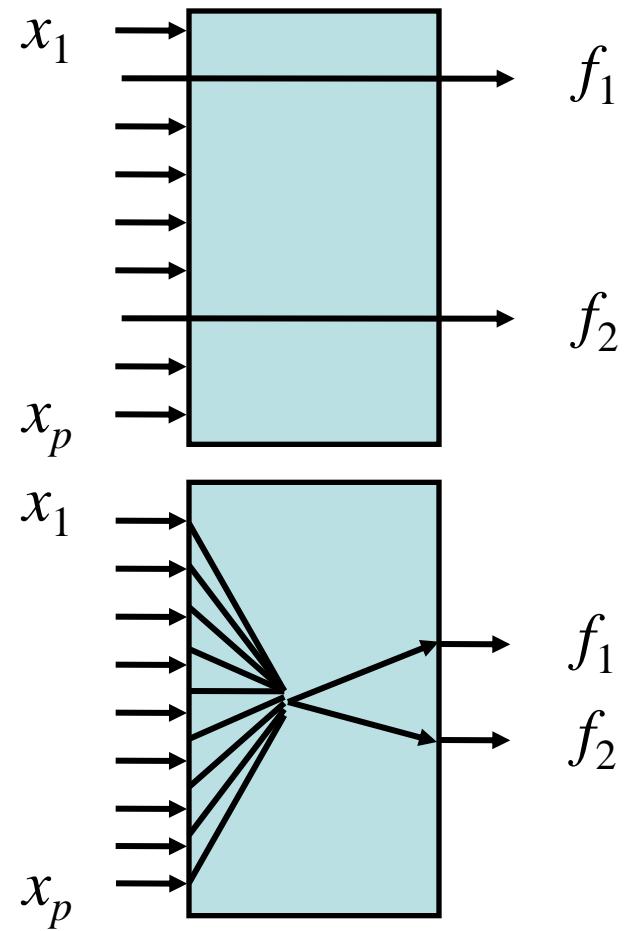
Feature selection vs. extraction

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



Feature selection vs. extraction

- **Feature selection:**
select d out of p features
- **Feature extraction:**
map p features
to d features
(e.g. PCA)



Feature selection v extraction (2)

	Advantage	Disadvantage
Selection	cut in features easy interpretation	expensive often approximate

Feature selection v extraction (2)

	Advantage	Disadvantage
Selection	cut in features	expensive
	easy interpretation	often approximate
Extraction	cheap	need all features
	linear and nonlinear	criterion sub-optimal
	not axis aligned	

Feature extraction (2)

- **Linear, unsupervised (= no class labels):**
 - Principal Component Analysis (PCA)
- **Linear, supervised (= use class labels):**
 - Linear Discriminant Analysis (LDA)

Principal component analysis

(Unsupervised feature extraction)

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

Principal component analysis

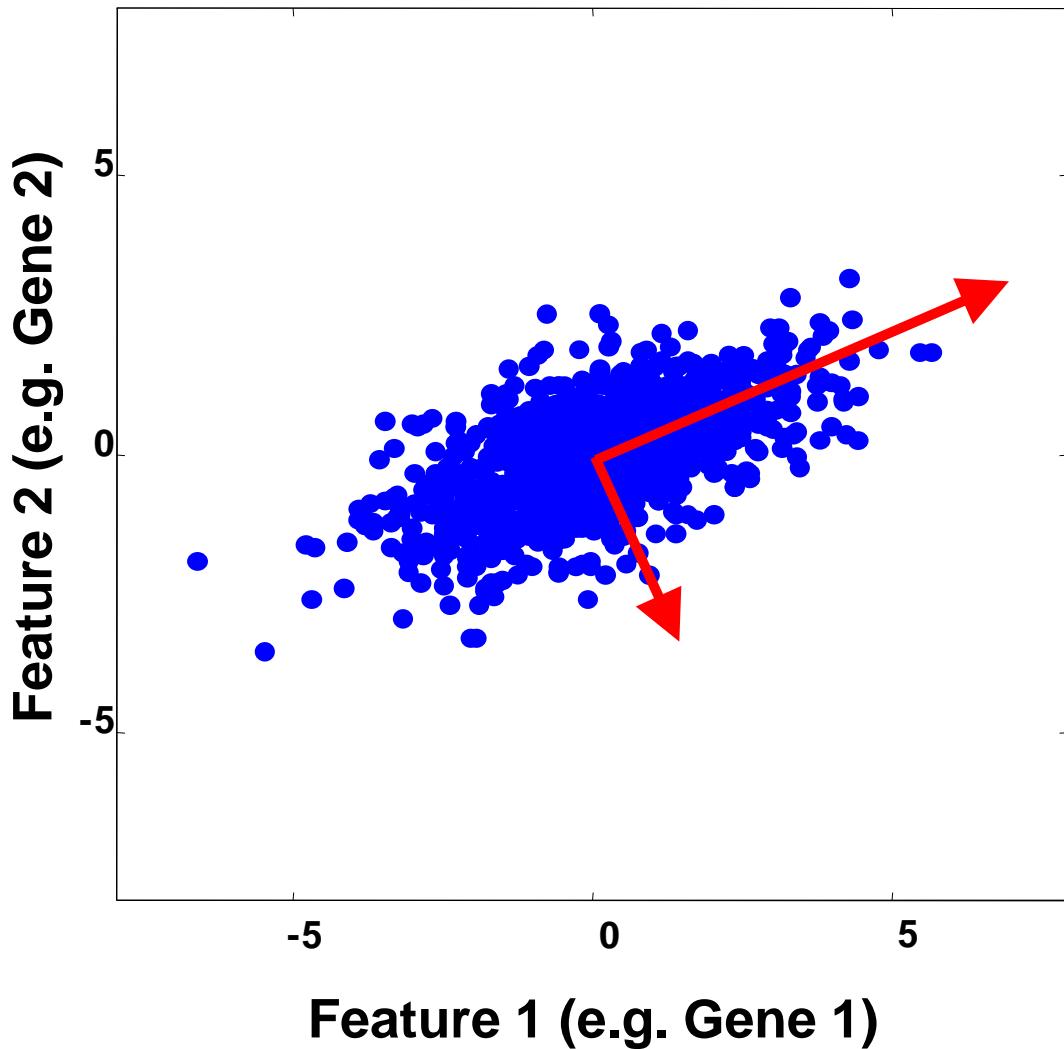
(Unsupervised feature extraction)

- **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 minimise squared *reconstruction error*

Principal component analysis (Unsupervised feature extraction)



Steps:

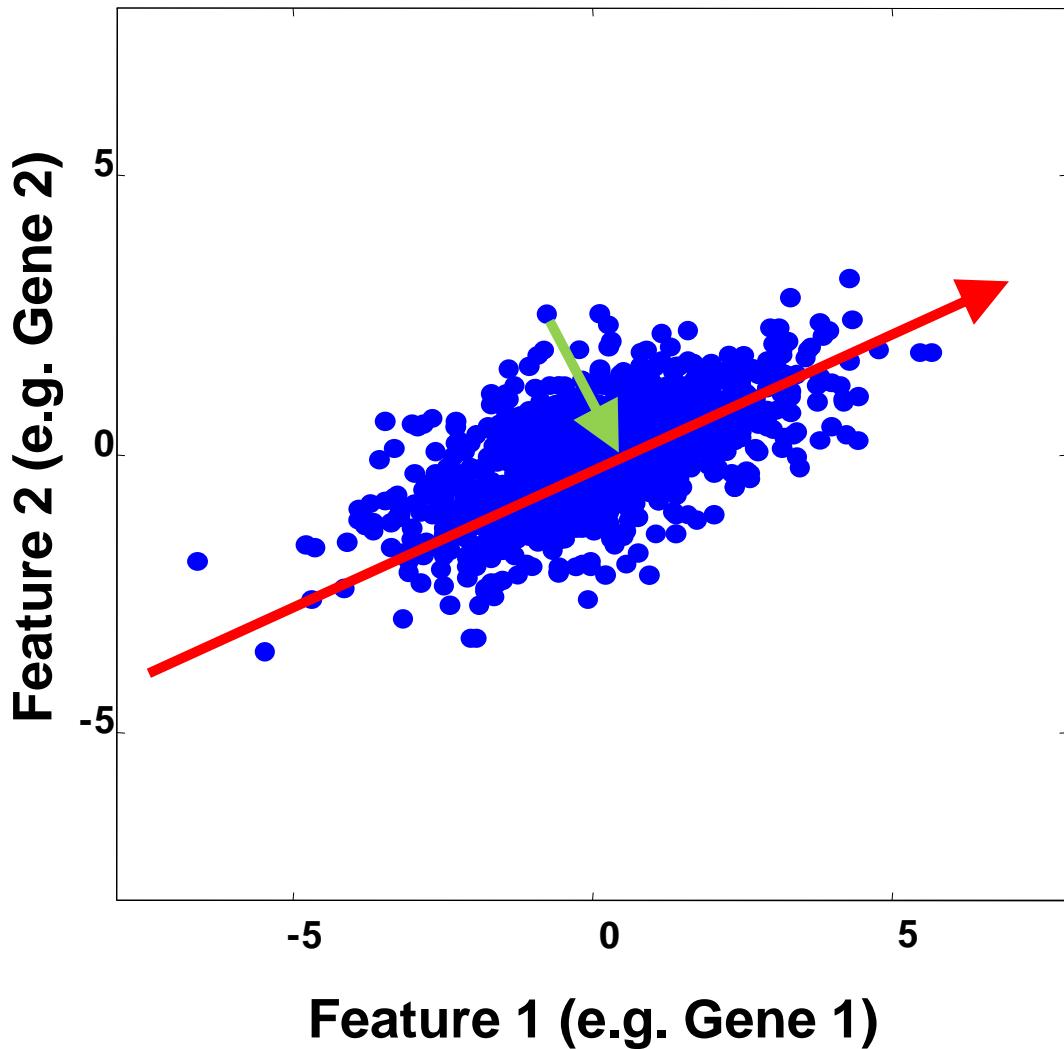
1. Center data
2. Compute covariance, C
3. Perform PCA on C

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,

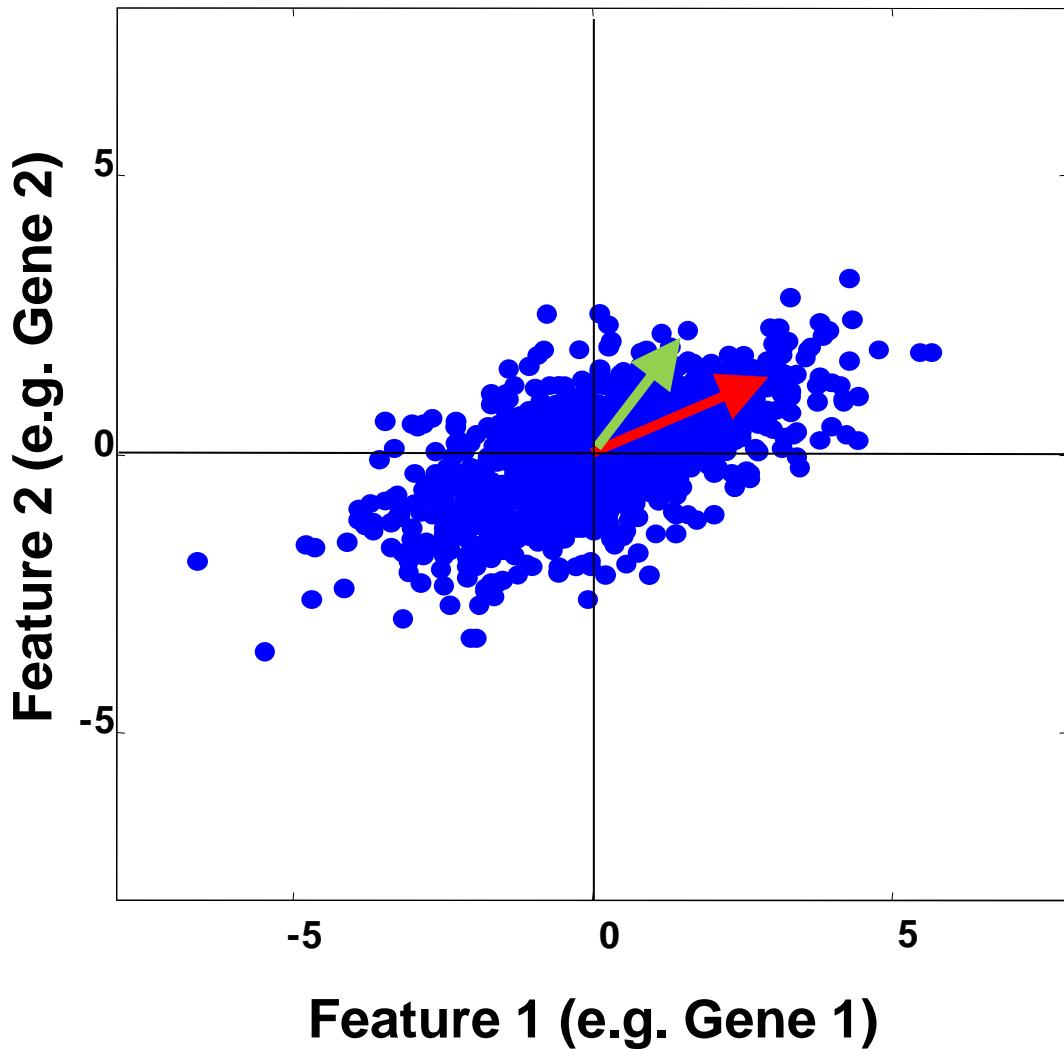
$$\sum_{i=1}^d \lambda_i$$

- or the ratio of retained variance,

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

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

Principal component analysis (Unsupervised feature extraction)



Steps:

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

Output:

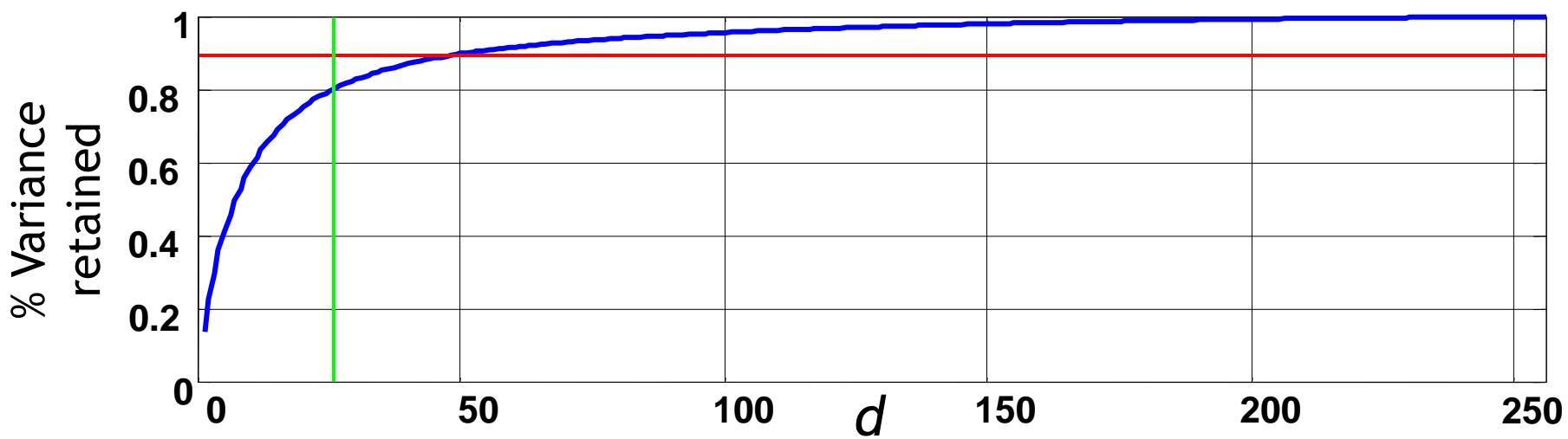
1. Eigenvectors: $\mathbf{e}_1, \mathbf{e}_2$
2. Eigenvalues: λ_1, λ_2

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 tips

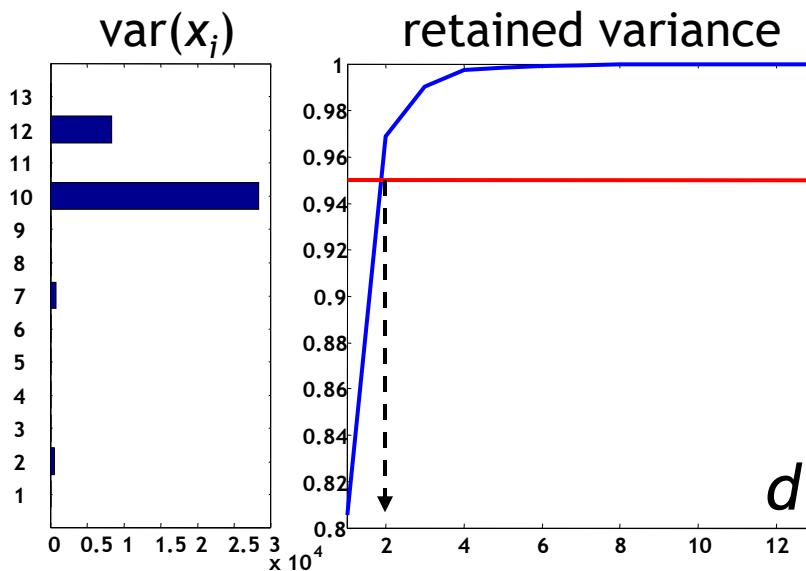
- Ensure data is centered (mean of each feature is zero):
 $x' \leftarrow (x - \mu)$

PCA tips

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

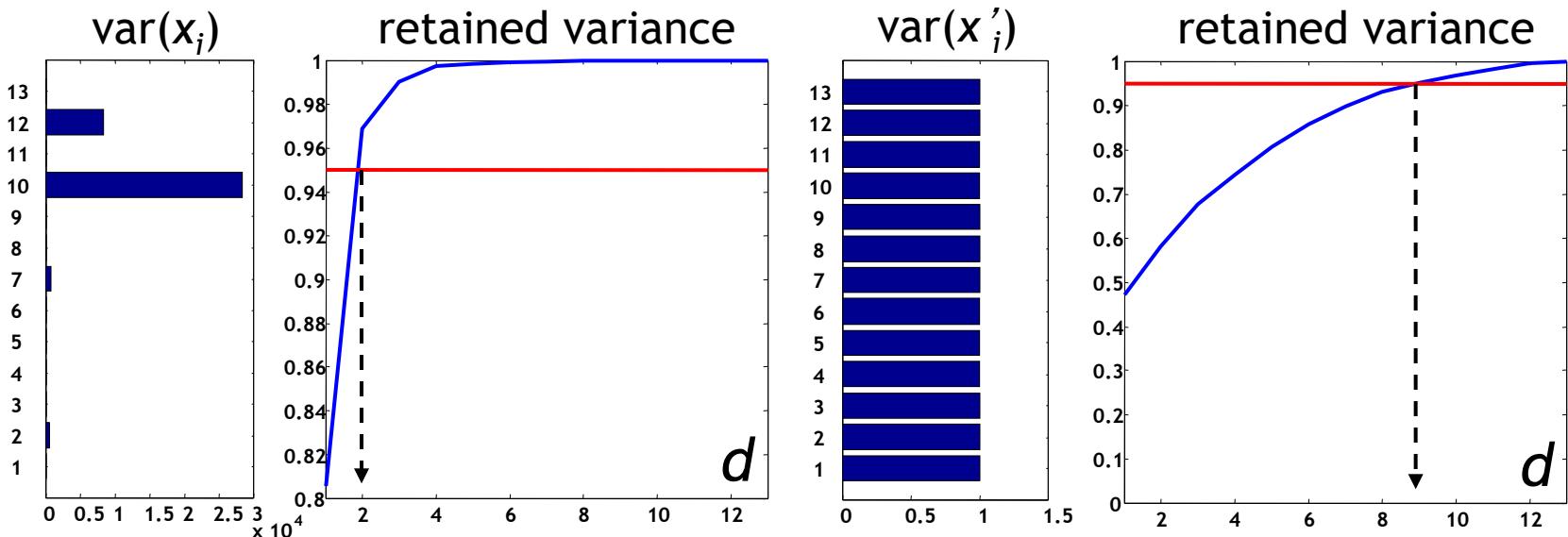
PCA tips

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



PCA tips

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

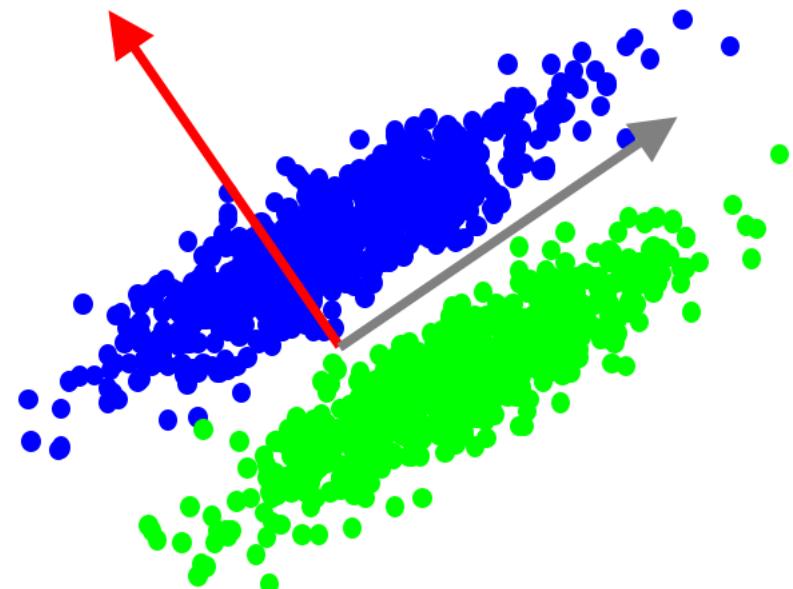


PCA conclusions

- PCA:
 - Is **global** and **linear**
 - **Criterion:** maximizes the retained variance
 - Is **unsupervised** (but we can do PCA on each class)
 - Needs a **lot of data** to estimate the **covariance matrix** well.

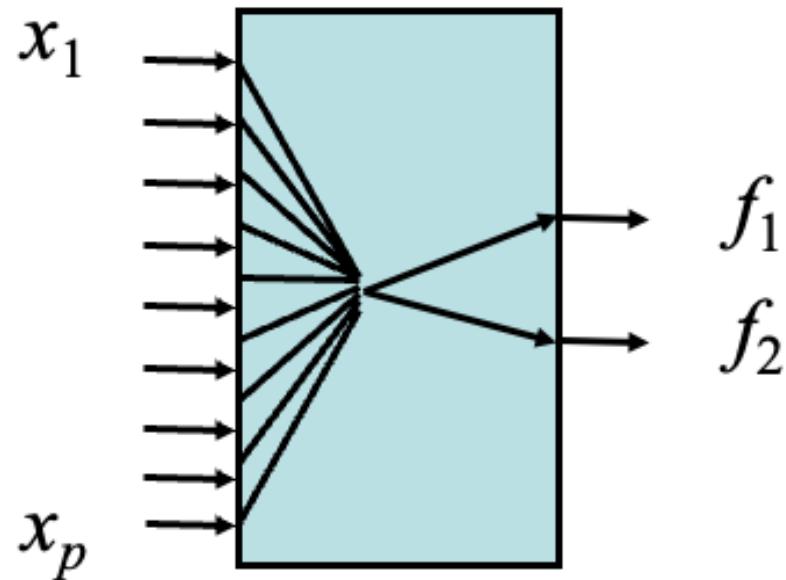
PCA conclusions

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



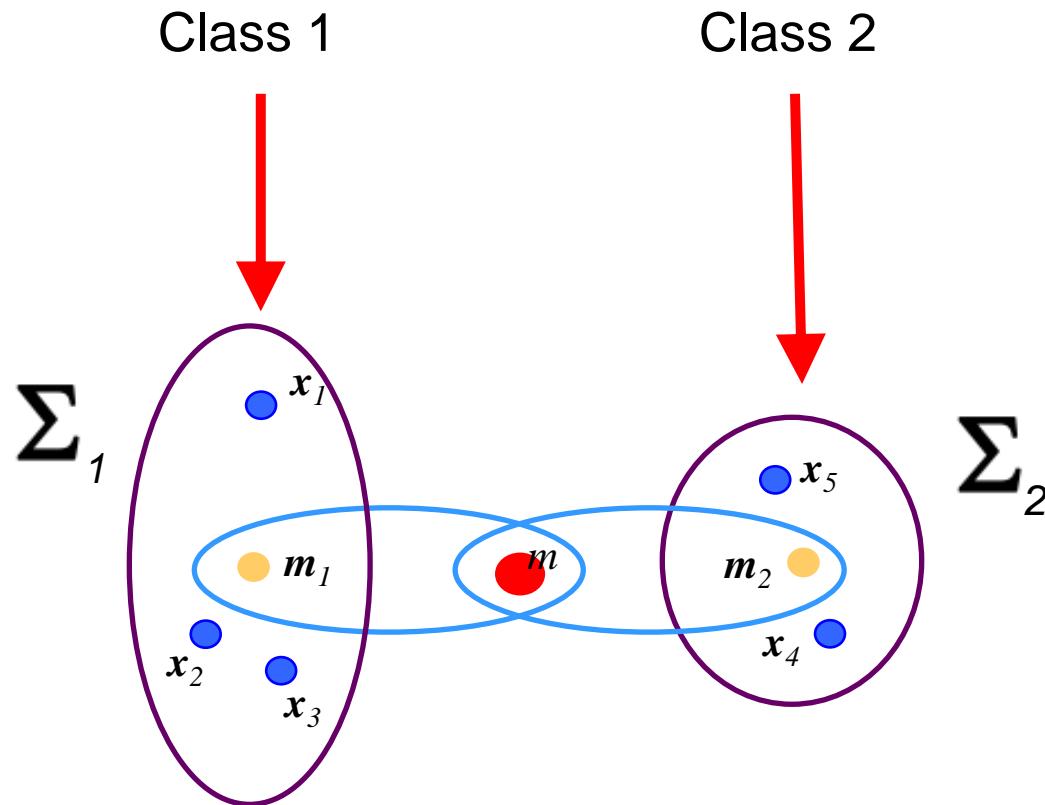
Supervised, linear feature extraction

- **Extraction:** mapping of features to new (sub)space (figure)
- Class label is given, hence **supervised** extraction
- **Criterion:** Reduce dimensionality and maximize class separation
- Examples: Fisher mapping; Linear Discriminant Analysis



Supervised feature extraction > Criteria

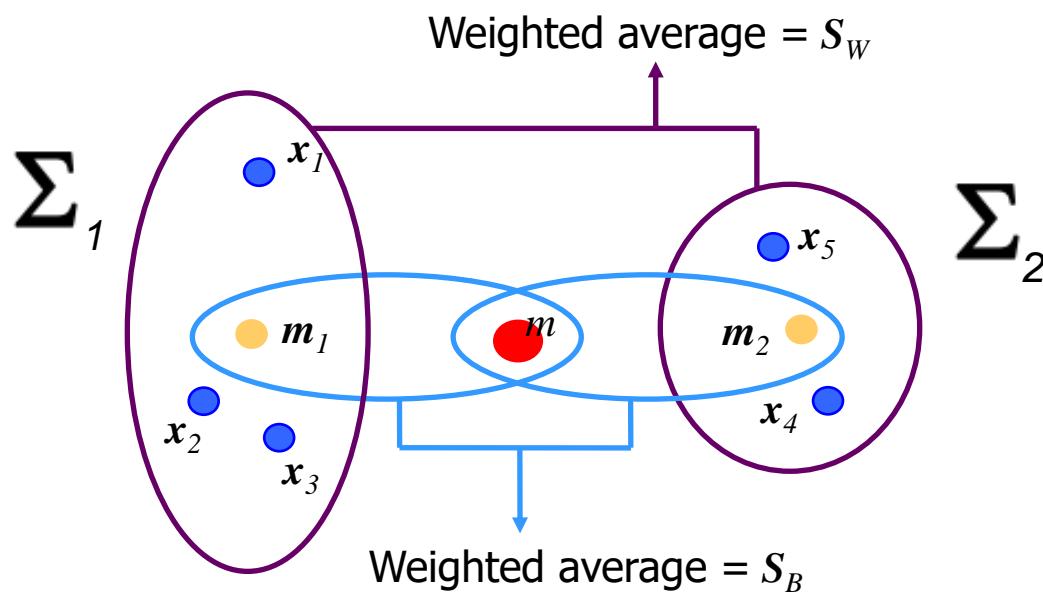
(supervised = we know the class labels)



Supervised feature extraction > Criteria

Within-class and between-class scatter matrices:

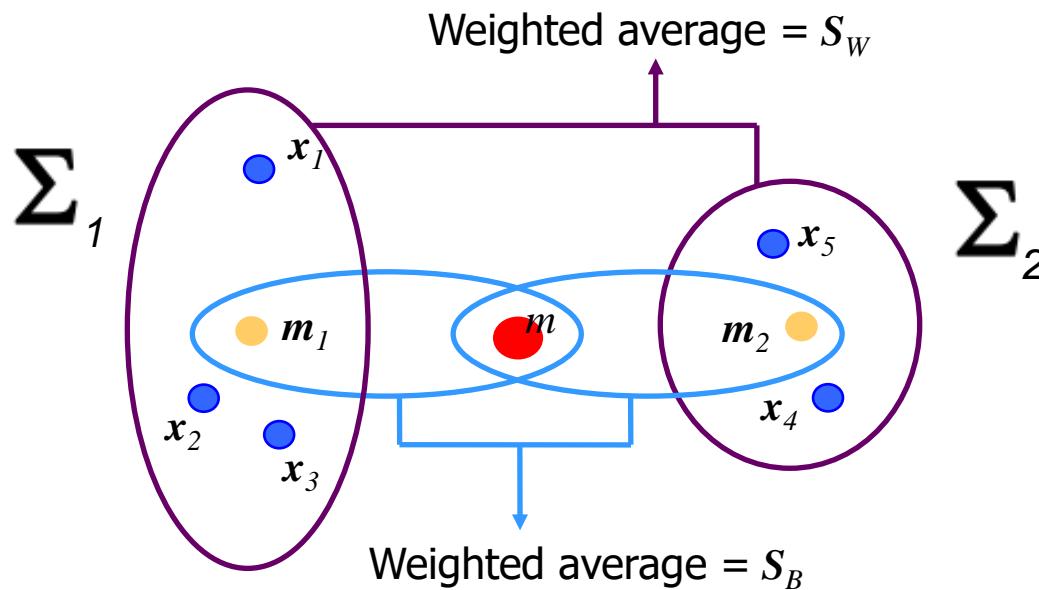
- Within-class: $S_w = \sum_{i=1}^C \frac{n_i}{n} \Sigma_i$



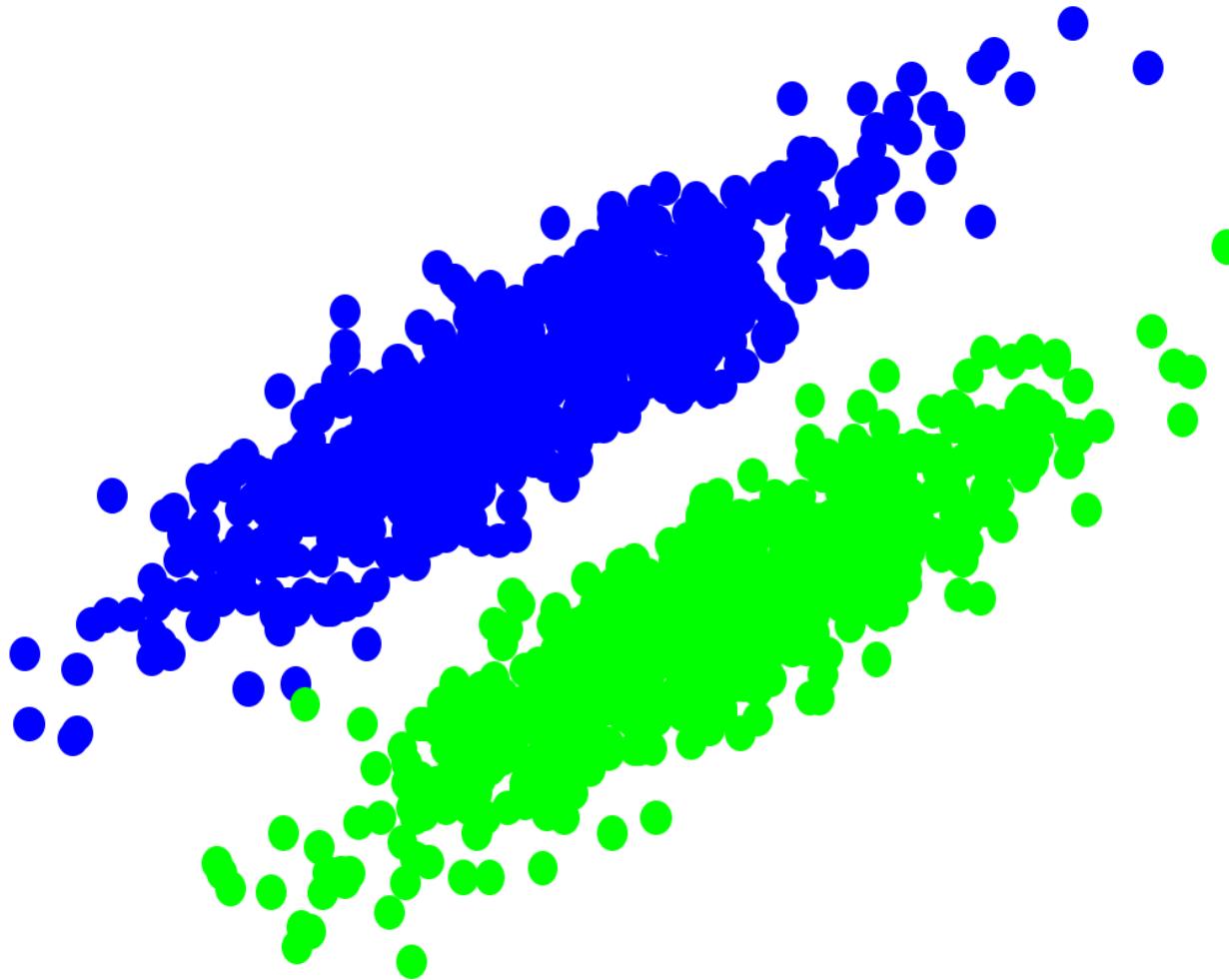
Supervised feature extraction > Criteria

Within-class and between-class scatter matrices:

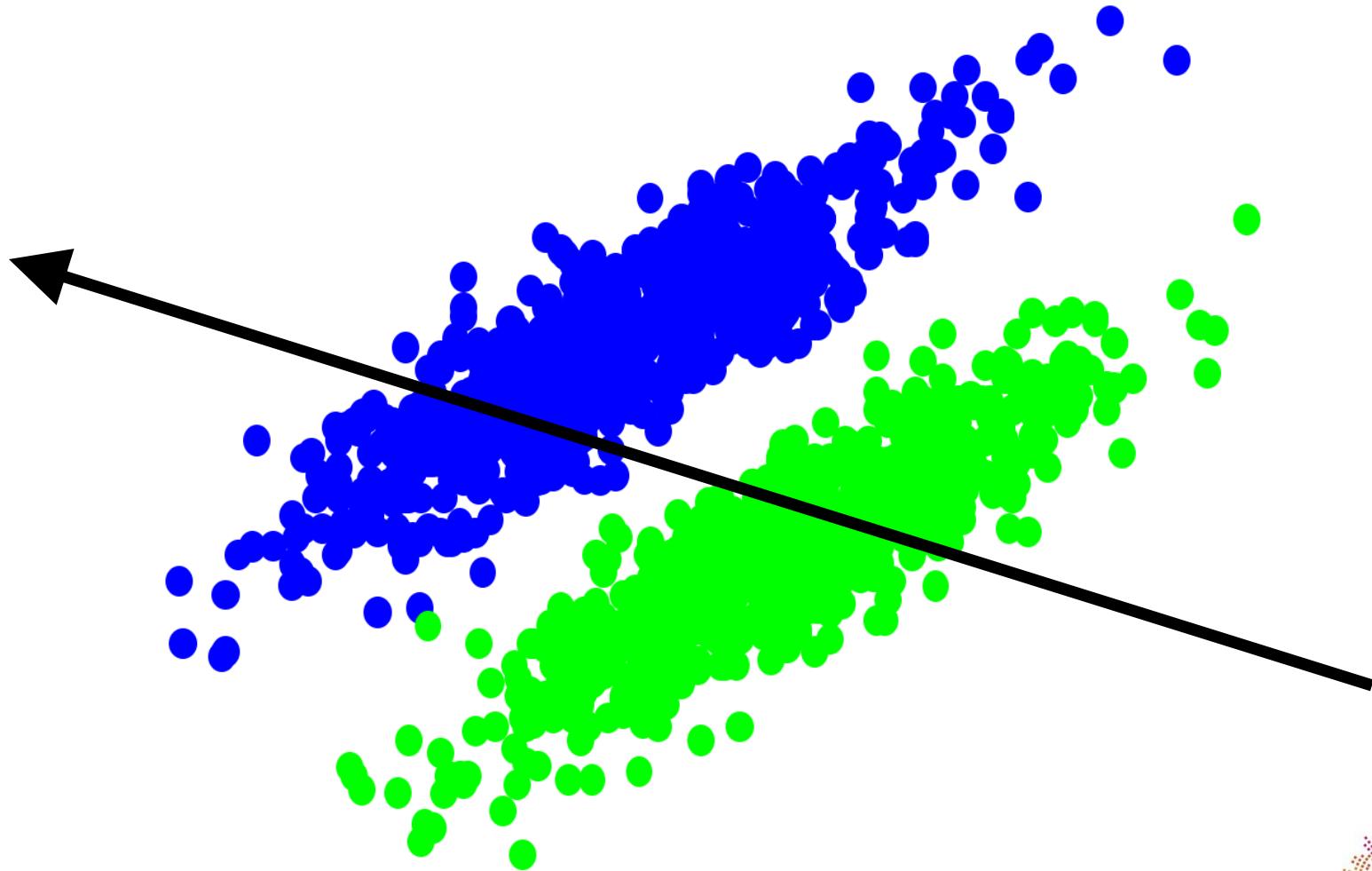
- Within-class: $S_w = \sum_{i=1}^C \frac{n_i}{n} \Sigma_i$
- Between-class: $S_B = \sum_{i=1}^C \frac{n_i}{n} (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$



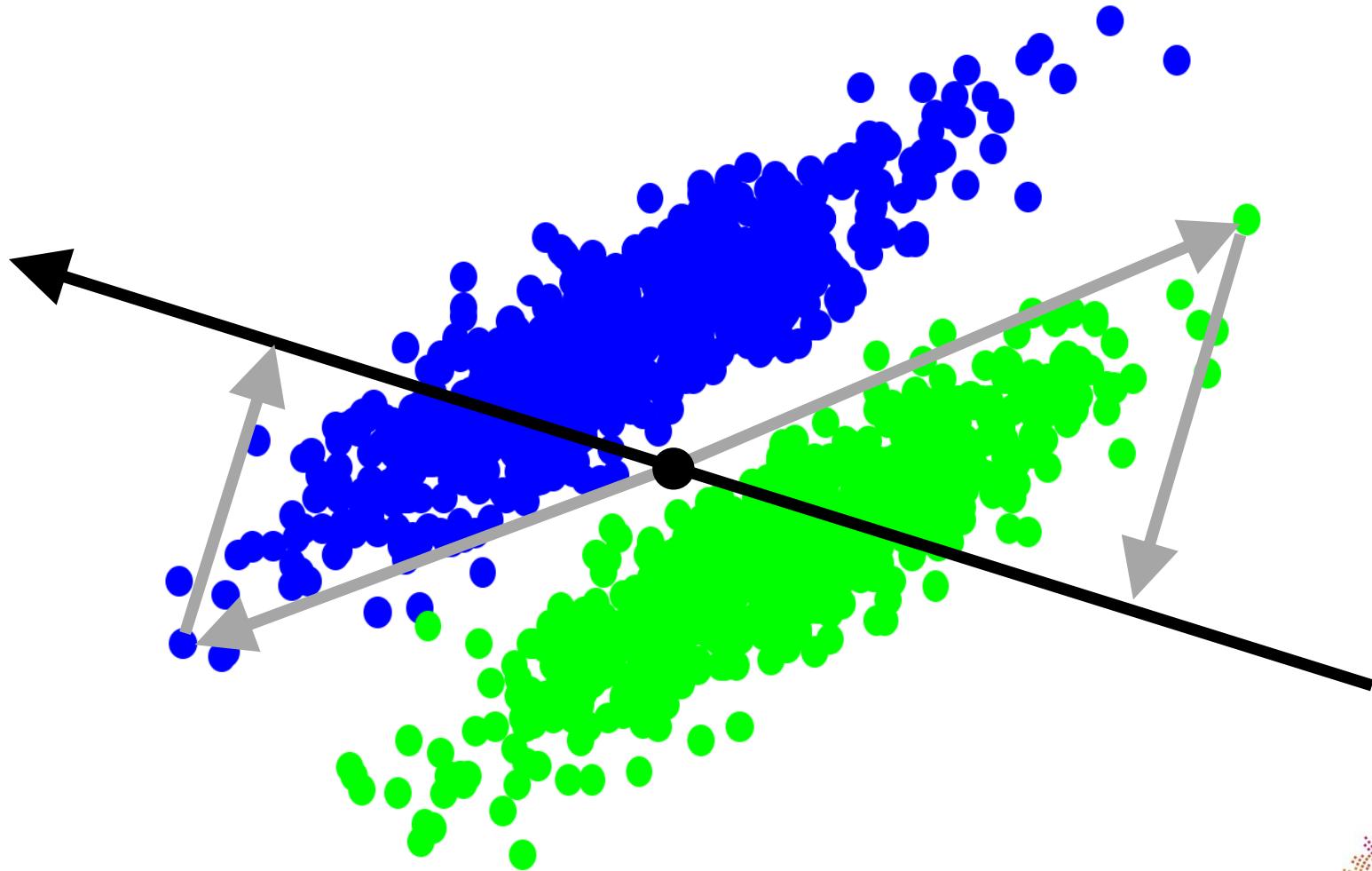
Fisher mapping: finding the direction (subspace) to project onto for the best class separation



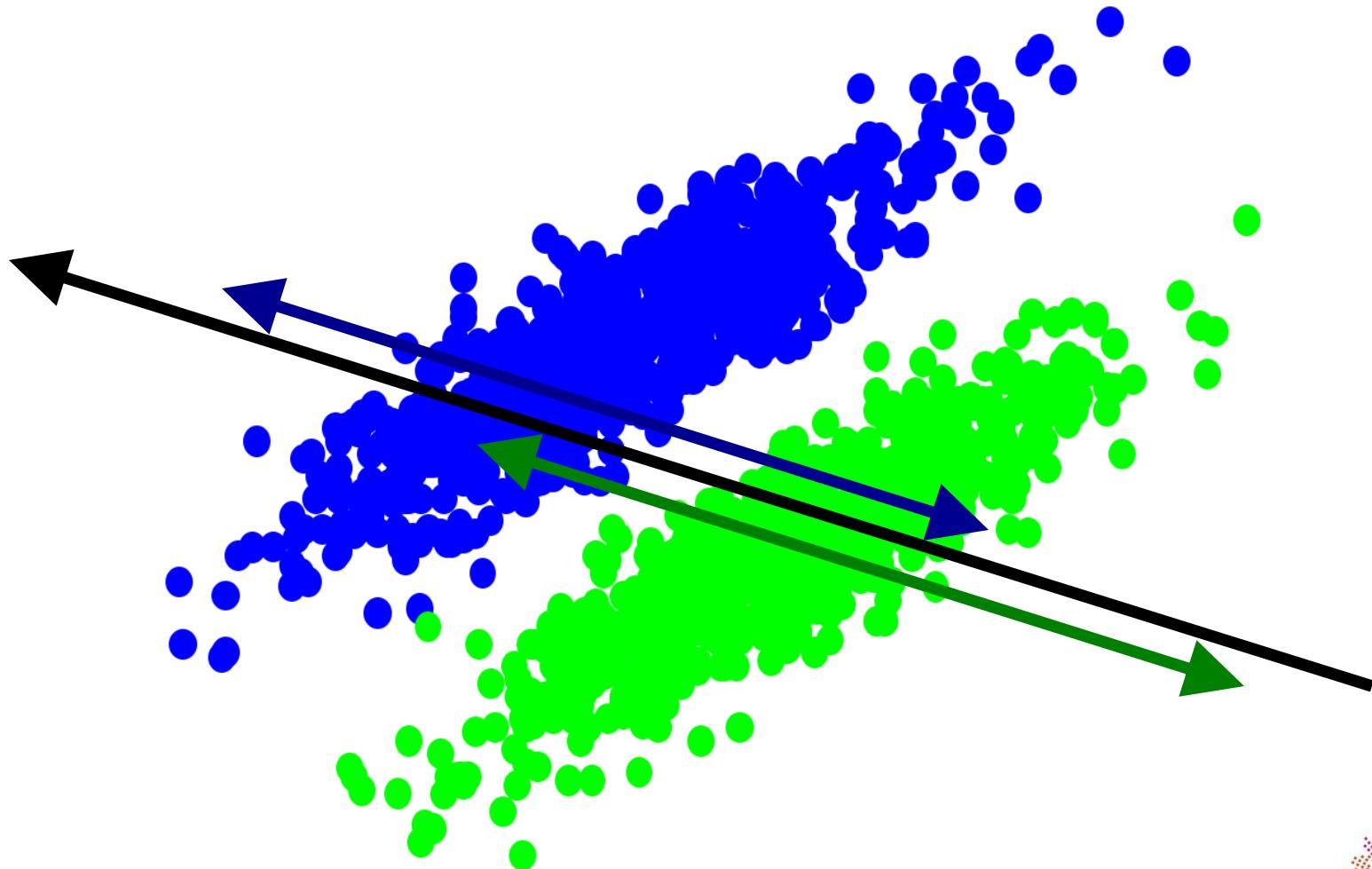
Fisher mapping: defining the Fisher criterion



Fisher mapping: defining the Fisher criterion

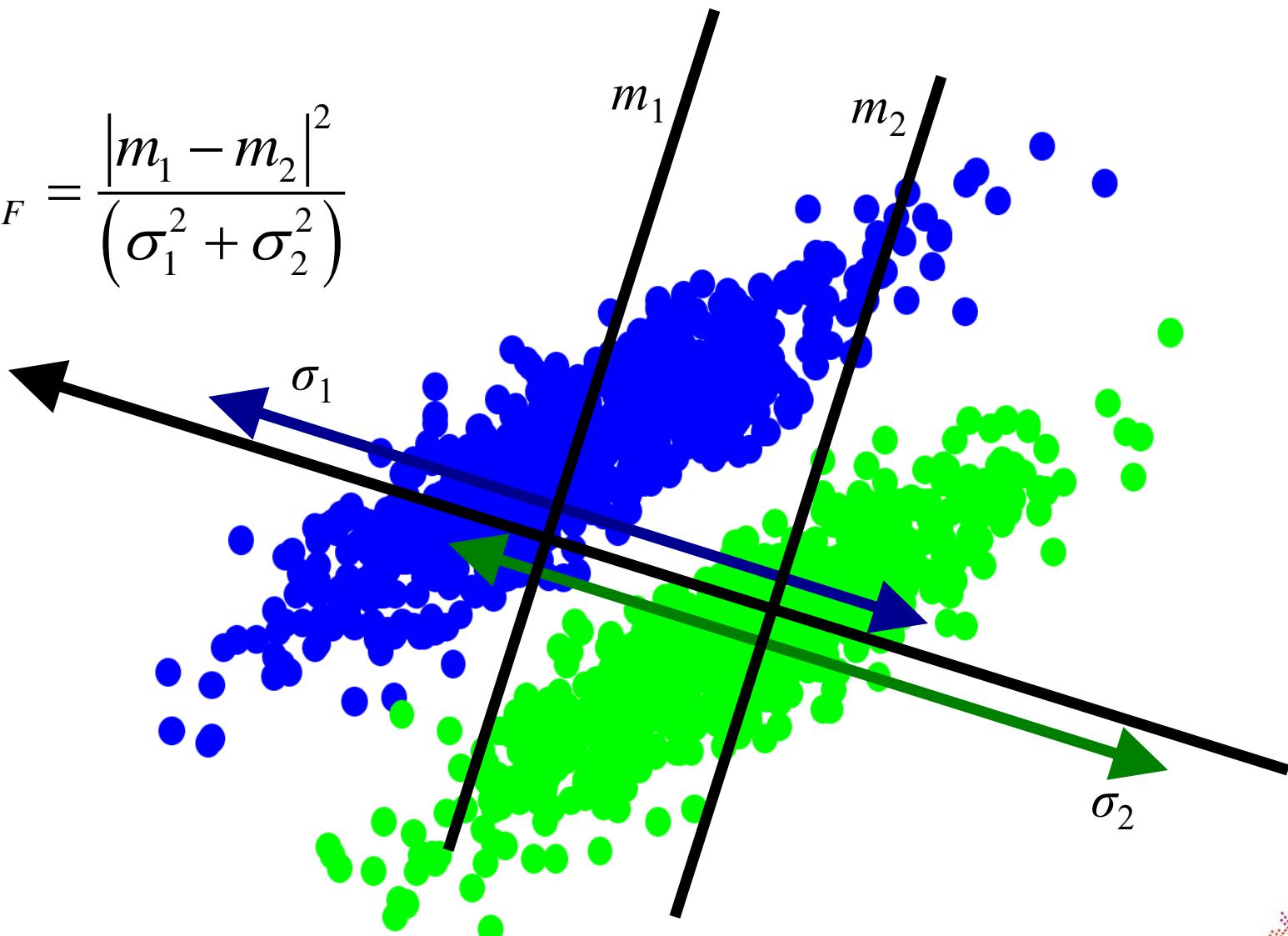


Fisher mapping: defining the Fisher criterion

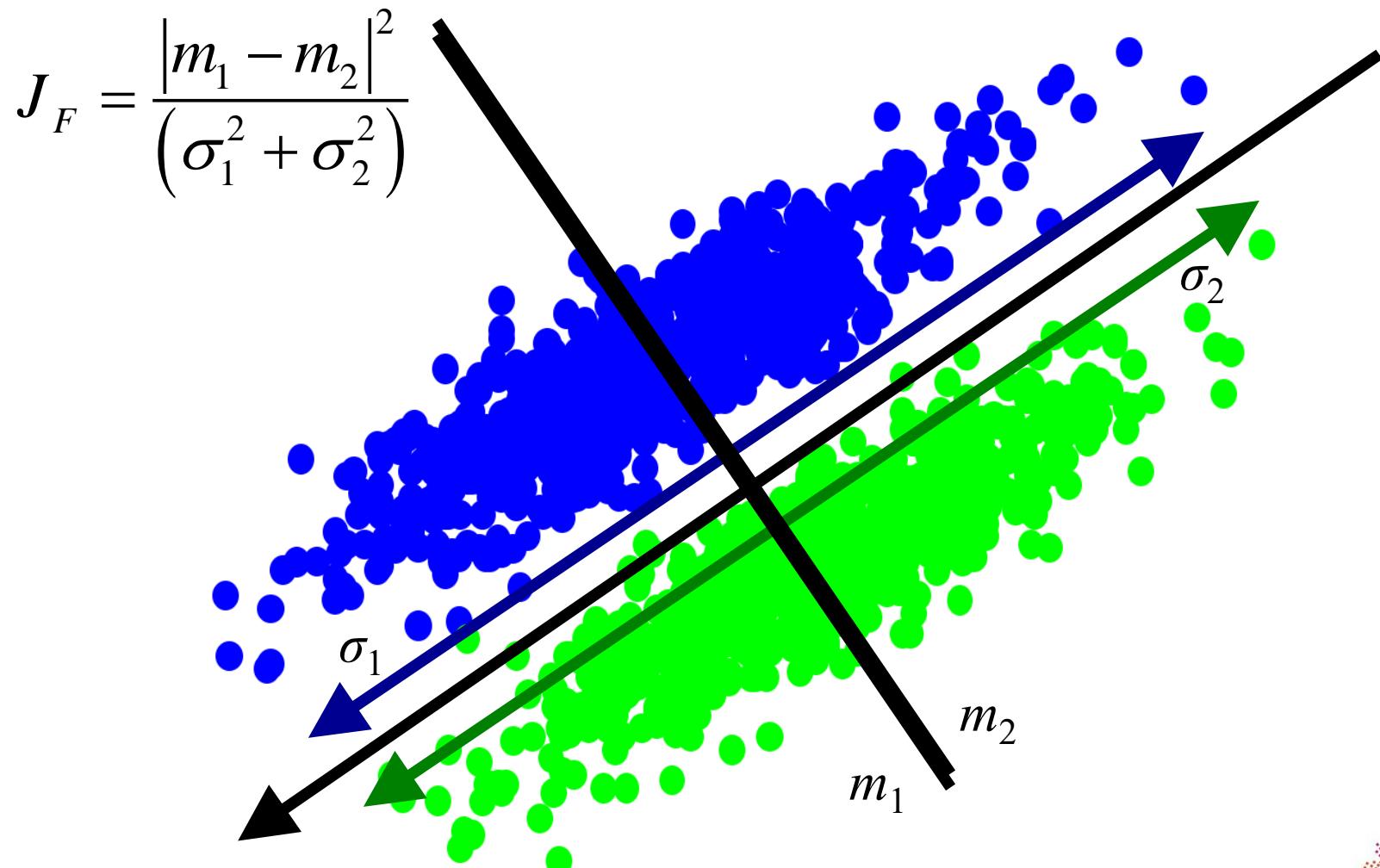


Fisher mapping (Fisher criterion)

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

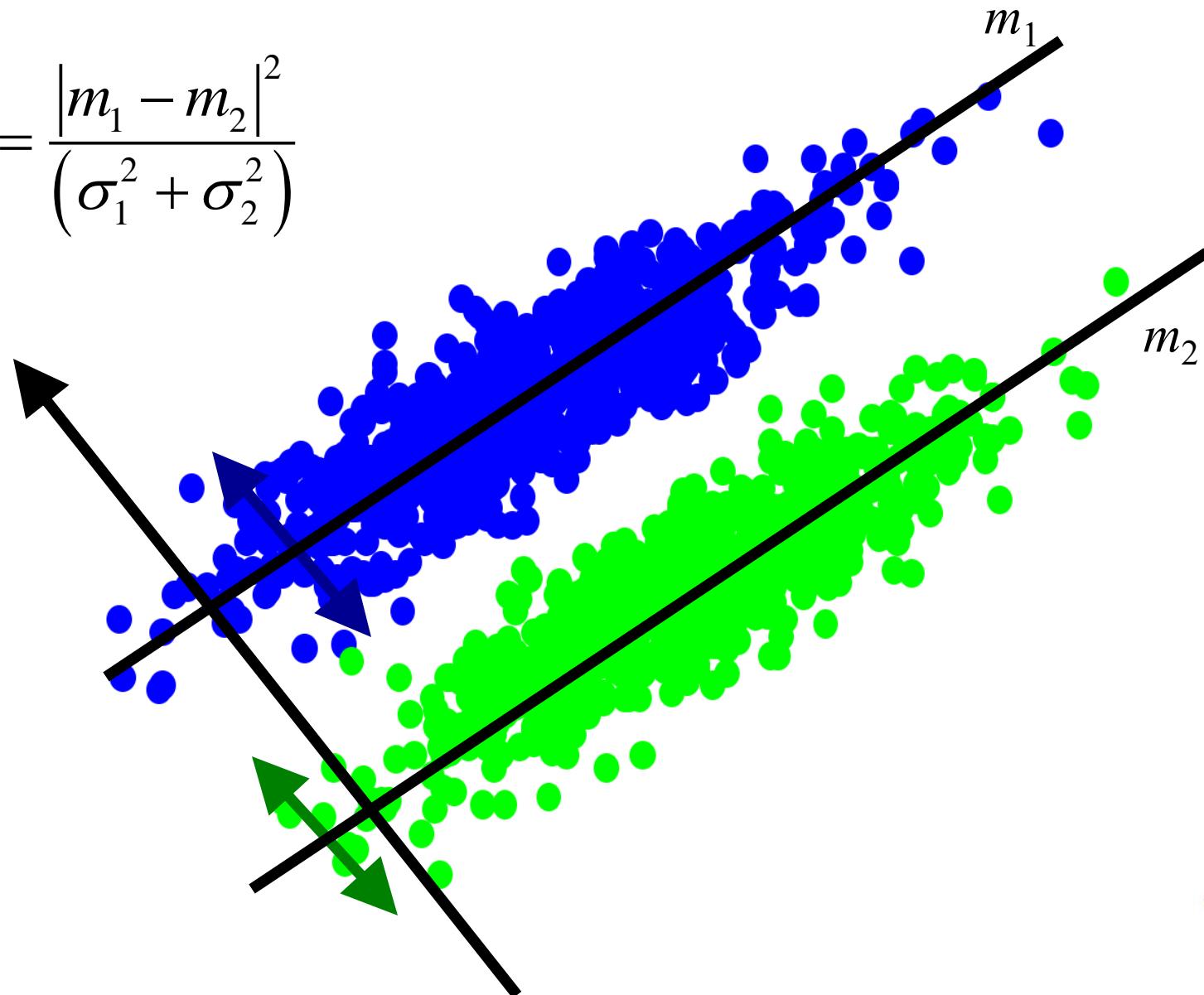


Fisher mapping: defining the Fisher criterion



Fisher mapping: defining the Fisher criterion

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

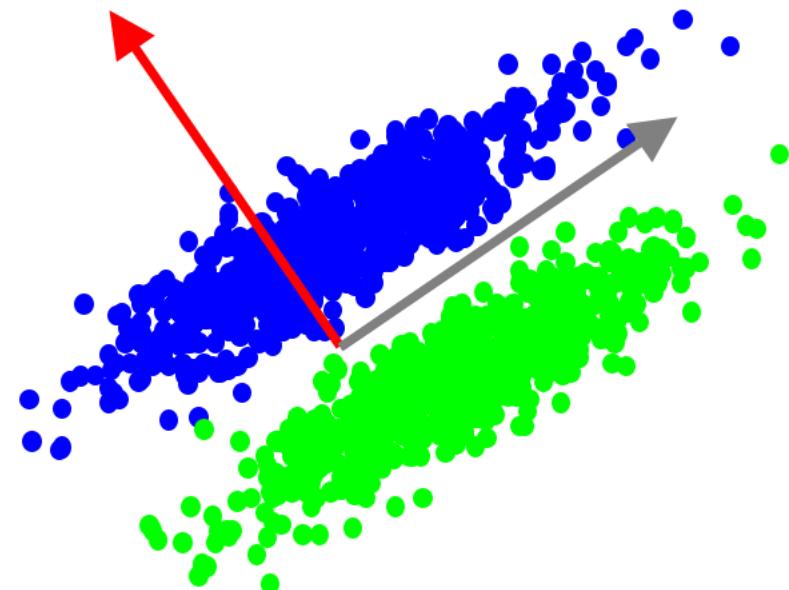


Fisher mapping

- Find basis vector a_1 for $\{x\}$ such that in the projections, the classes are maximally separated
- Choose a_1 to maximise Fisher criterion:

$$J_F(a_1) = \frac{a_1^T S_B a_1}{a_1^T S_W a_1}$$

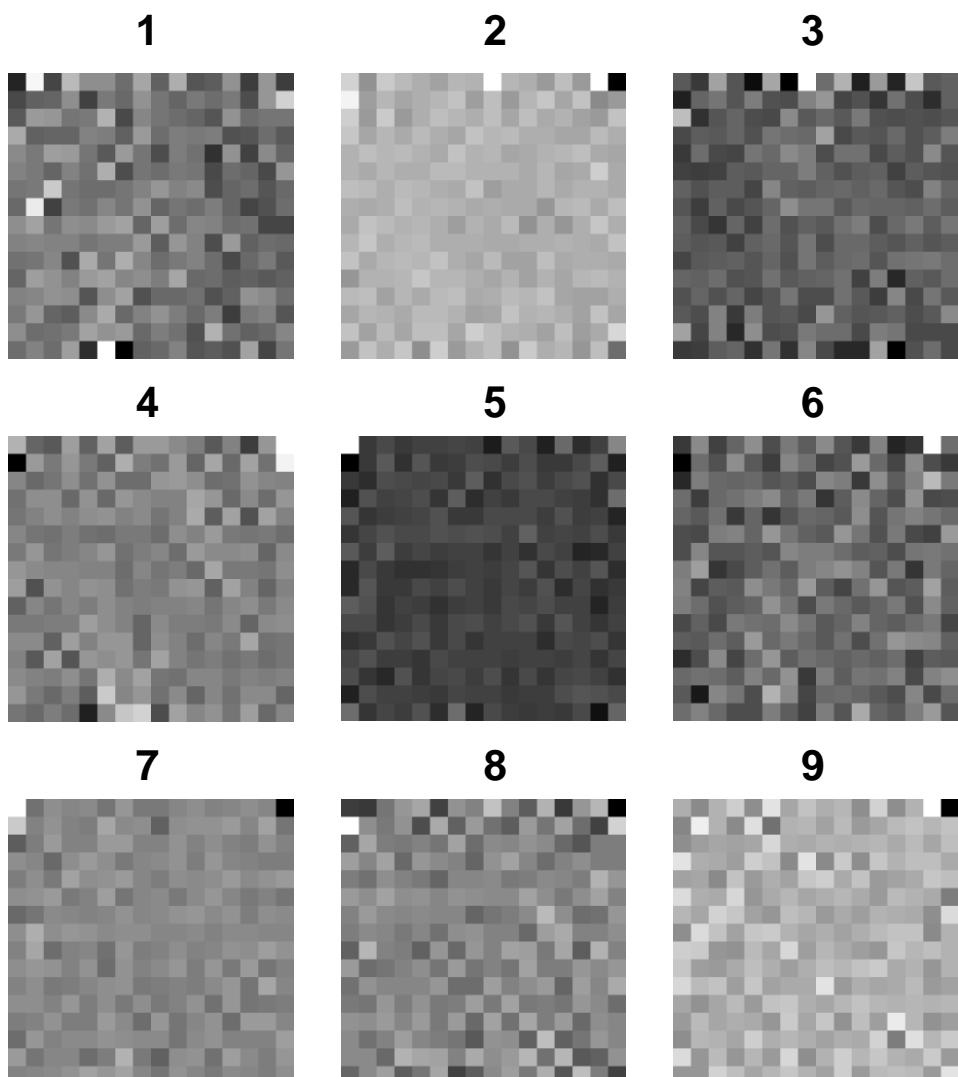
- Maximize between class variance
Minimize within class variance



- Solution:
 - eigen-analysis on $S_W^{-1}S_B$
 - select $c-1$ (# classes – 1) dimensions for final classifier

Fisher mapping (3)

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



Fisher mapping (4)

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

Fisher mapping (4)

- To avoid fitting noise, can do PCA first
- If system is underdetermined ($n \leq p$), first doing PCA is required, otherwise matrix inversion results in singularity
- But then we might be destroying the class separation as PCA is *unsupervised*

Summary

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

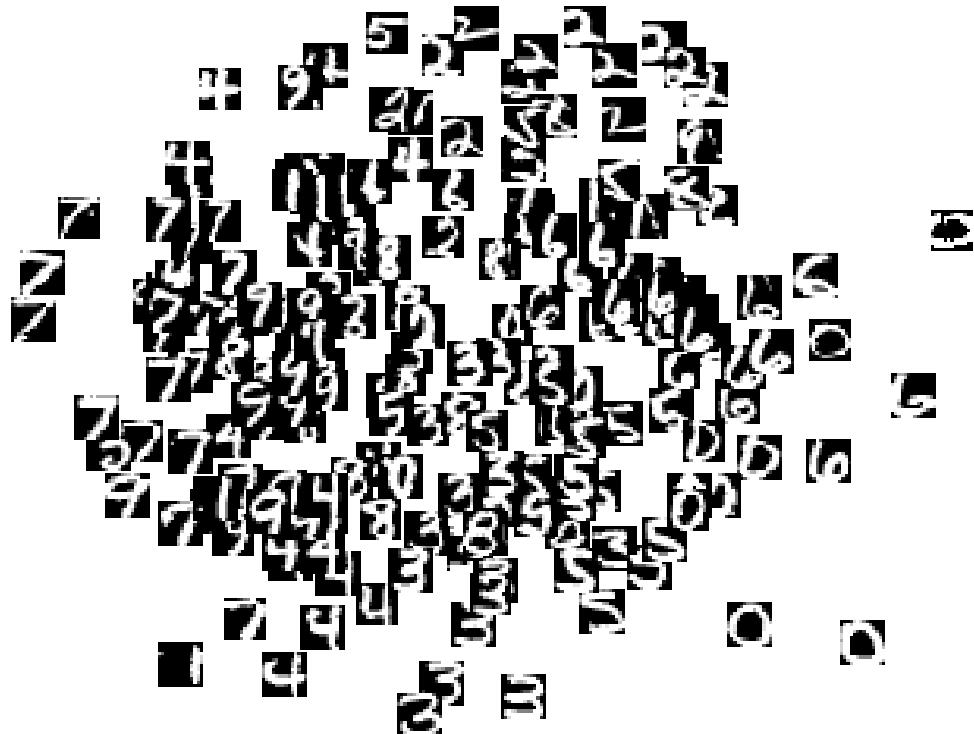
Nonlinear, unsupervised feature extraction

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

Nonlinear feature extraction (3)

Example: embedding

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

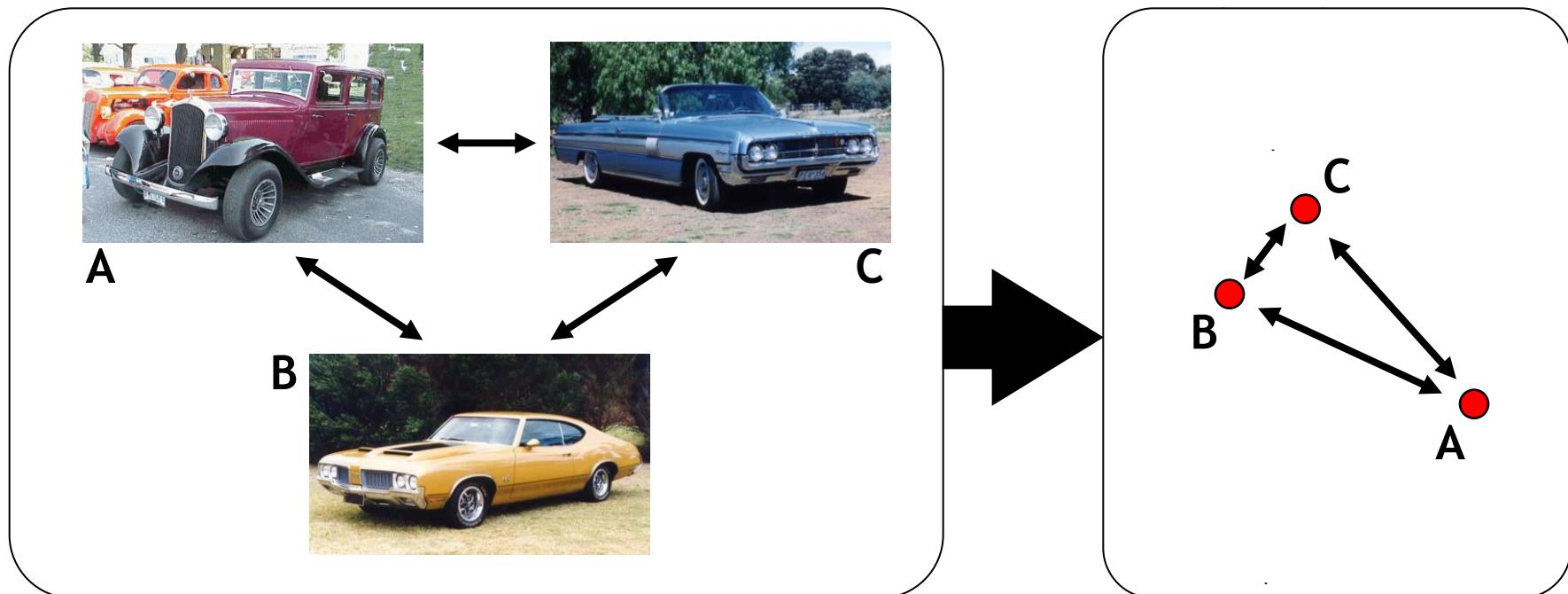


Multidimensional scaling (MDS)

- 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 (2)

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

MDS (2)

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

MDS (2)

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

MDS (2)

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

MDS (2)

- Advantages of using distances:
 - do not necessarily need original data
 - allows inclusion of **knowledge on objects** (e.g. characteristics of amino acids when comparing proteins)
 - allows inclusion of **knowledge of relations** (e.g. invariances) in distance measure (e.g. Pearson correlation being shift and scale invariant 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 (5)

- d_{ij} : distance $\|x_i - x_j\|$ in original space ($? - \text{dimensional}$)
- δ_{ij} : distance $\|y_i - y_j\|$ in new space $(d - \text{dimensional})$

MDS: Non-linear mappings (5)

- d_{ij} : distance $\|x_i - x_j\|$ in original space ($? - \text{dimensional}$)
- δ_{ij} : distance $\|y_i - y_j\|$ in new space ($d - \text{dimensional}$)

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

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

Sammon mapping: $q = -1$

MDS: Non-linear mappings (6)

- **Procedure:**
 - Initialize positions of samples in lower dimensional space (y_i)

MDS: Non-linear mappings (6)

- **Procedure:**
 - Initialize positions of samples in lower dimensional space (y_i)
 - Compute stress

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

MDS: Non-linear mappings (6)

- **Procedure:**
 - Initialize positions of samples in lower dimensional space (y_i)
 - Compute stress

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

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

MDS: Non-linear mappings (6)

- **Procedure:**
 - Initialize positions of samples in lower dimensional space (y_i)
 - Compute stress

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

- 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 (6)

- **Procedure:**

- Initialize positions of samples in lower dimensional space (y_i)
- Compute stress

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

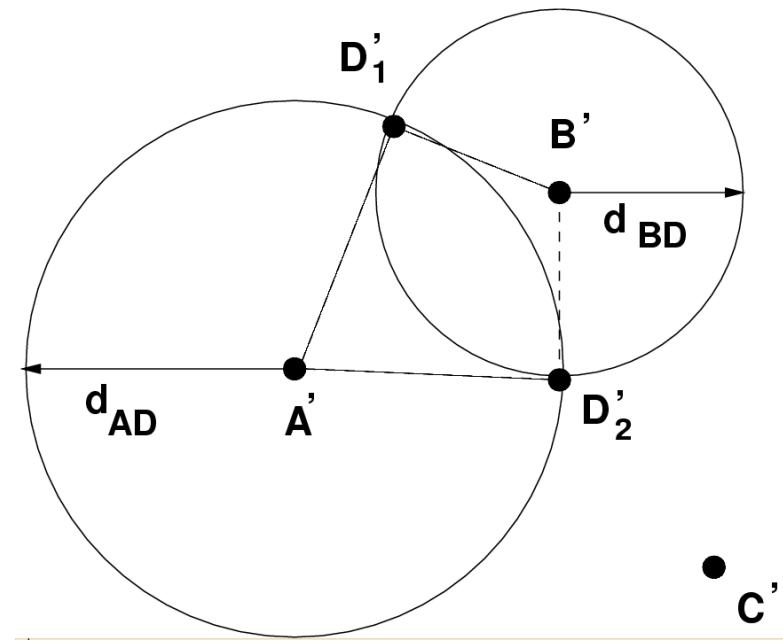
- Compute derivative of the stress with respect to positions of samples in new space
- Adapt the positions of samples in lower dimensional space

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

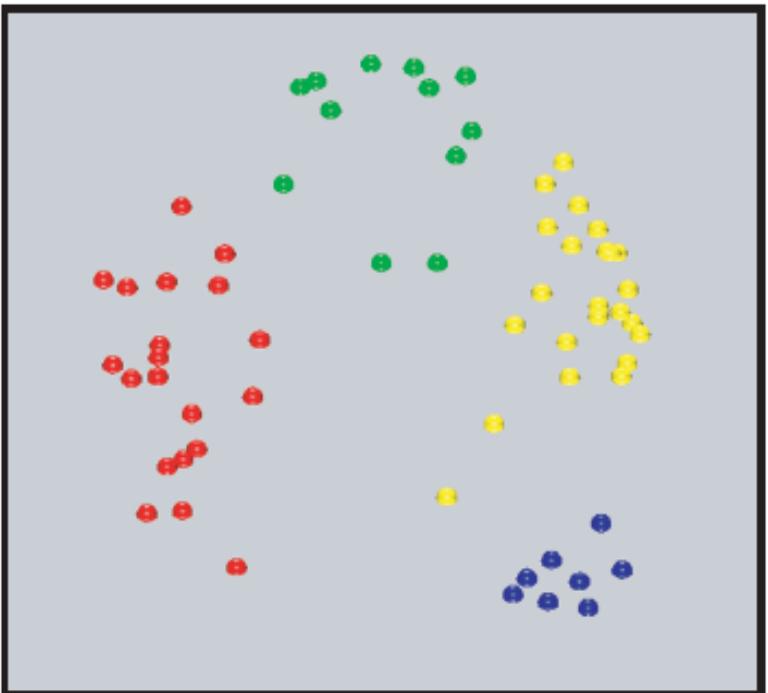
- Repeat till convergence (positions of samples do not change)

Embedding new points

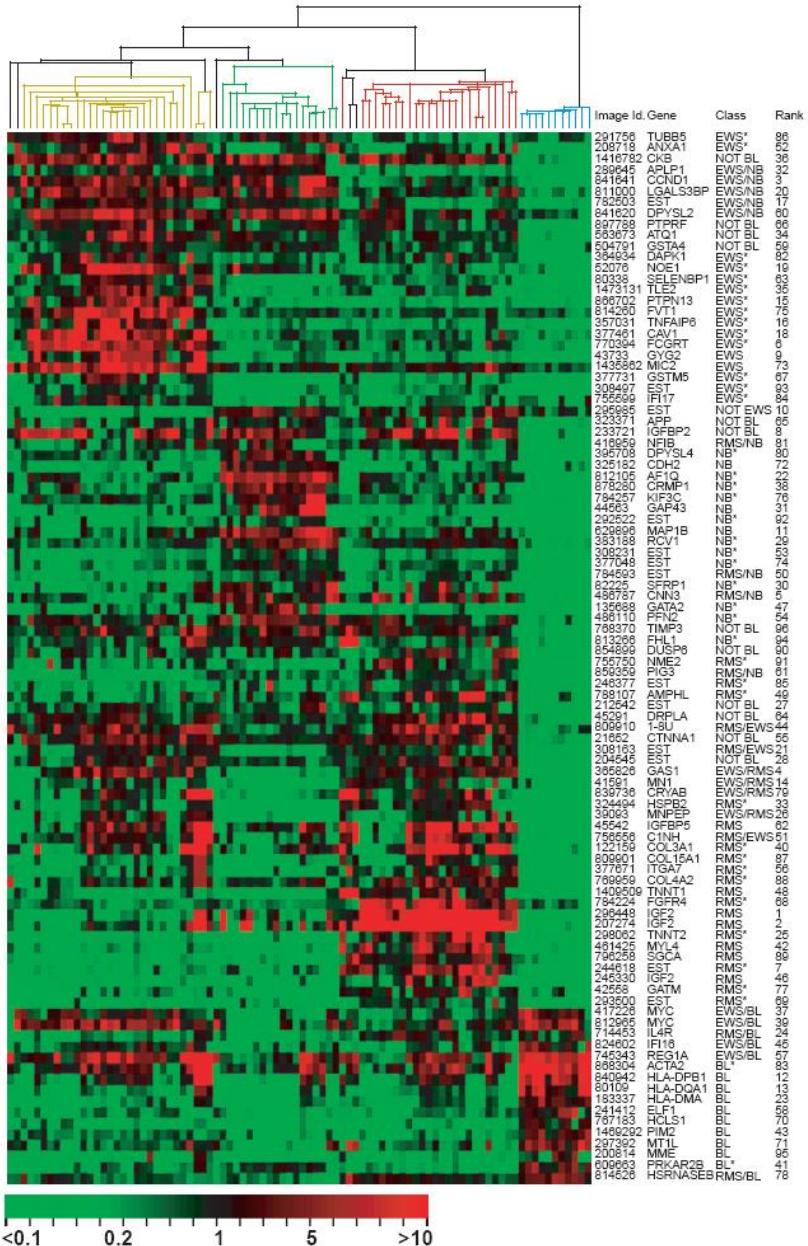
- Problematic: re-run entire algorithm...
- Sub-optimal solution: triangulation
 - Embed new point **D**
 - **D** has **A** and **B** as neighbors in original space
 - Preserve distance to two embedded neighbours **A'**, **B'** exactly
 - Use **C'** to decide which of the two candidates **D₁', D₂'** to use



MDS example



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

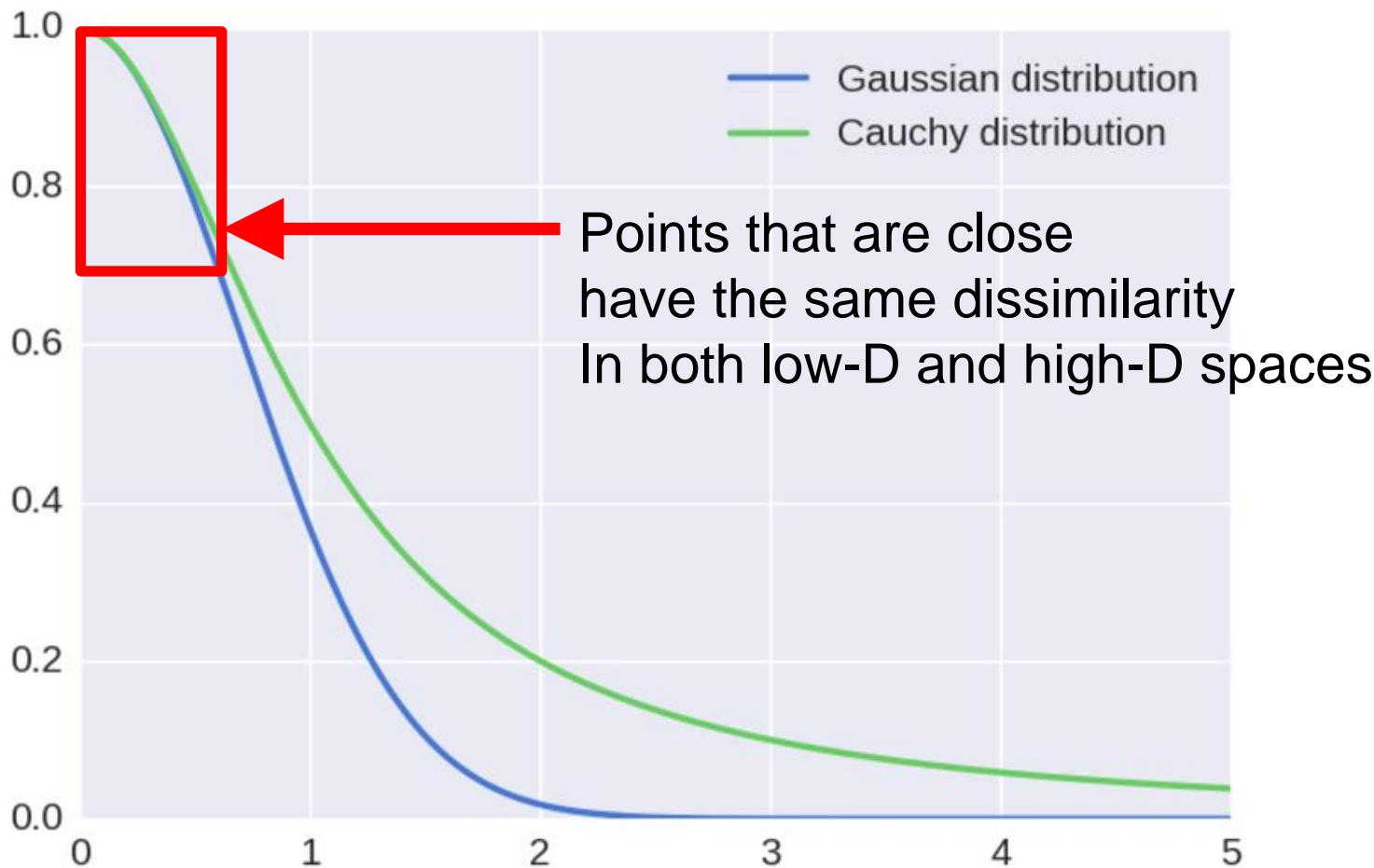


Khan et al, Nature Medicine, 2001

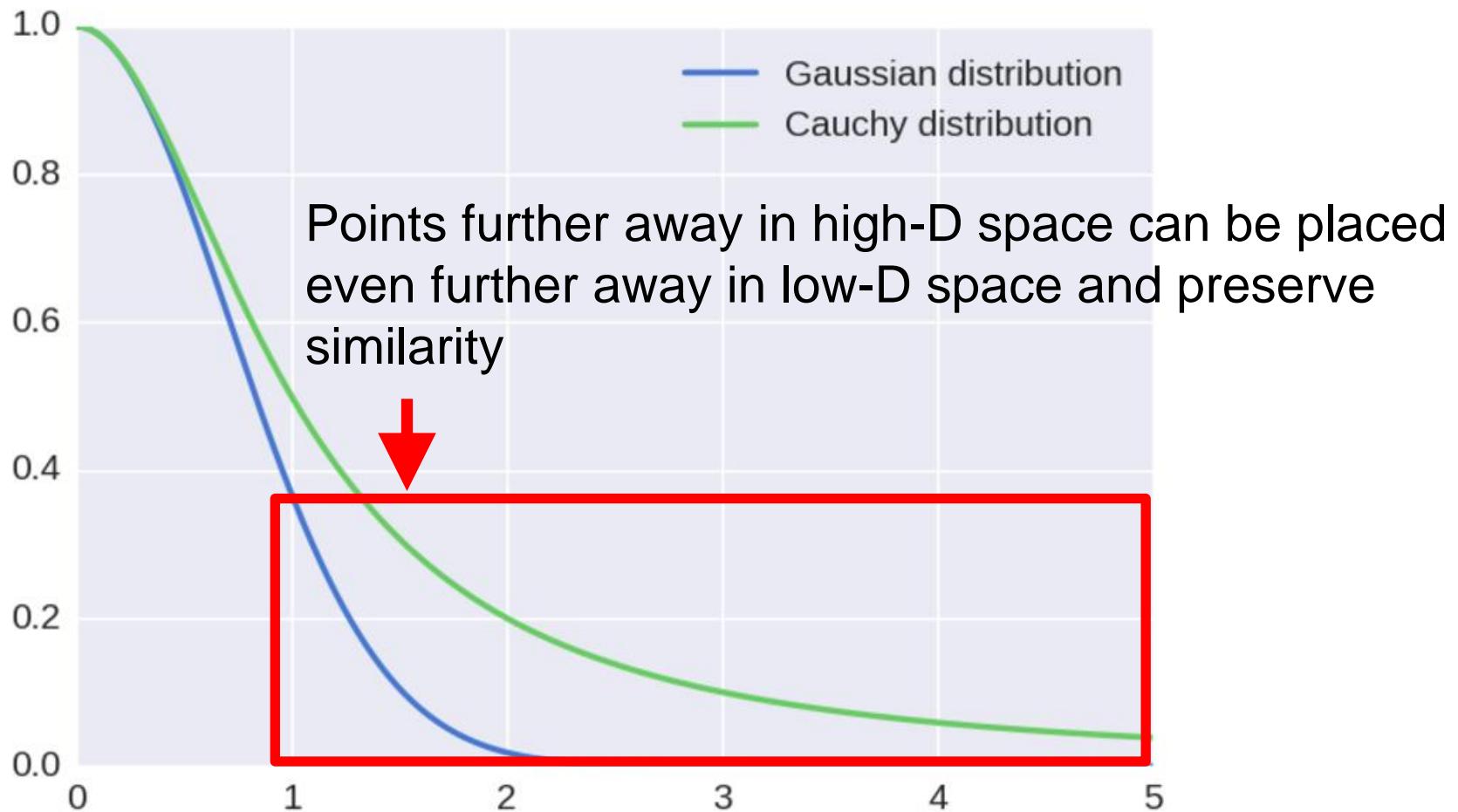
t-SNE (t-distributed stochastic neighbor embedding) (van der Maaten et al, 2008)

- In the input (high-D) space, X: compute dissimilarities between all pairs of points using a gaussian dissimilarity measure, p_{ij}
- In the output (low-D) space, Y: compute dissimilarities between all pairs of points using a t-distribution (with 1 d.o.f. (Cauchy)) dissimilarity measure, q_{ij}
- Minimize the Kullback-Leibler distance between these two distributions

t-SNE: Cauchy and Gaussian distribution



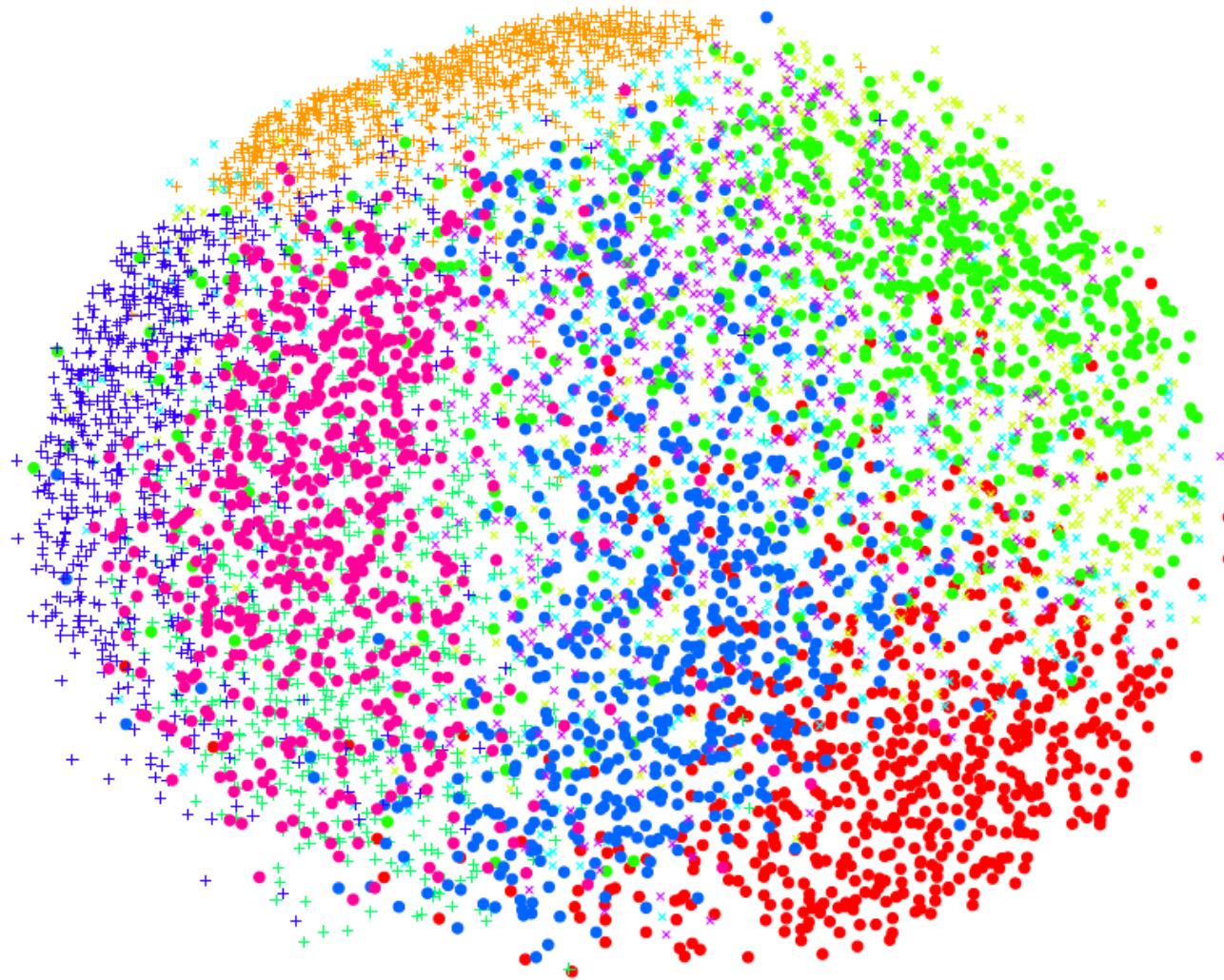
t-SNE: Cauchy and Gaussian distribution



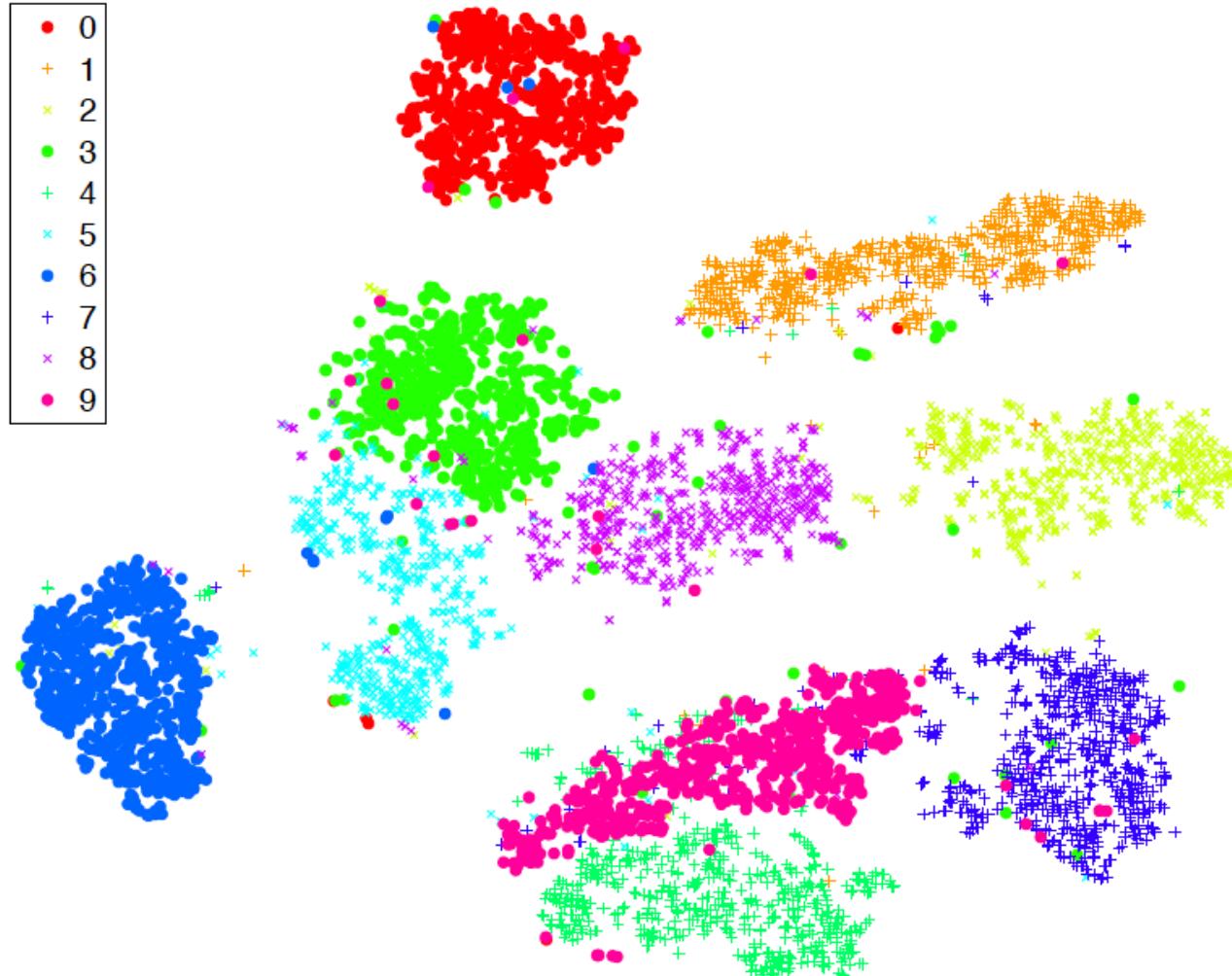
t-SNE (t-distributed stochastic neighbor embedding) (van der Maaten et al, 2008)

- In the input (high-D) space, X: compute dissimilarities between all pairs of points using a gaussian dissimilarity measure, p_{ij}
- In the output (low-D) space, Y: compute dissimilarities between all pairs of points using a t-distribution (with 1 d.o.f. (Cauchy)) dissimilarity measure, q_{ij}
- Minimize the Kullback-Leibler distance between these two distributions (P and Q)
- t-SNE faithfully retains small distances

t-SNE: Sammon map of digit data ($q = -1$; de-emphasis of large distances)



t-SNE: t-SNE map of digit data



UMAP

(Uniform Manifold Approximation and Projection)

- As t-SNE constructs a high-D graph representation of the data then optimizes a low-D graph to be as structurally similar

UMAP

(Uniform Manifold Approximation and Projection)

- As t-SNE constructs a high-D graph representation of the data then optimizes a low-D graph to be as structurally similar
- High-D graph:
 - "fuzzy simplicial complex"
 - weighted graph: edge weights representing the likelihood that two points are connected.
 - Connects points within a certain radius
 - Radius includes nth neighbor
 - "Fuzzy" likelihood of connection decreases with radius

UMAP

(Uniform Manifold Approximation and Projection)

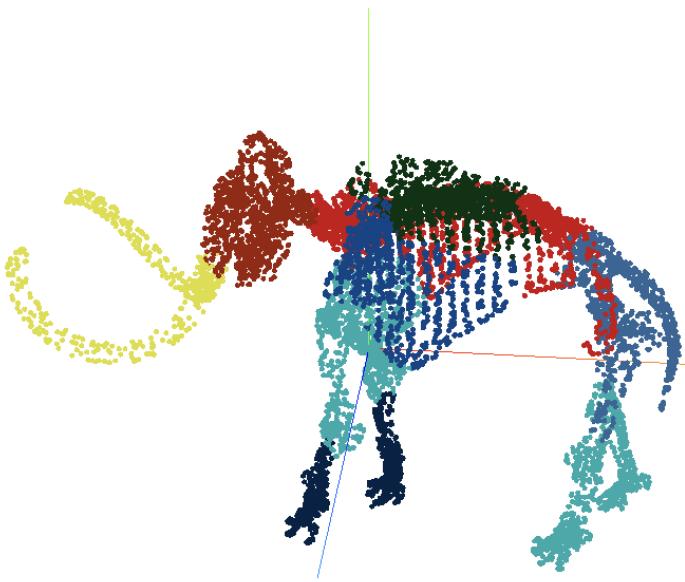
- As t-SNE constructs a high-D graph representation of the data then optimizes a low-D graph to be as structurally similar
- High-D graph:
 - "fuzzy simplicial complex"
 - weighted graph: edge weights representing the likelihood that two points are connected.
 - Connects points within a certain radius
 - Radius includes nth neighbor
 - "Fuzzy" likelihood of connection decreases with radius
- Low-D graph:
 - optimizes layout of a low-D graph to be as similar as possible
 - This process is essentially the same as in t-SNE
 - Using a few clever tricks for speed

UMAP Parameters

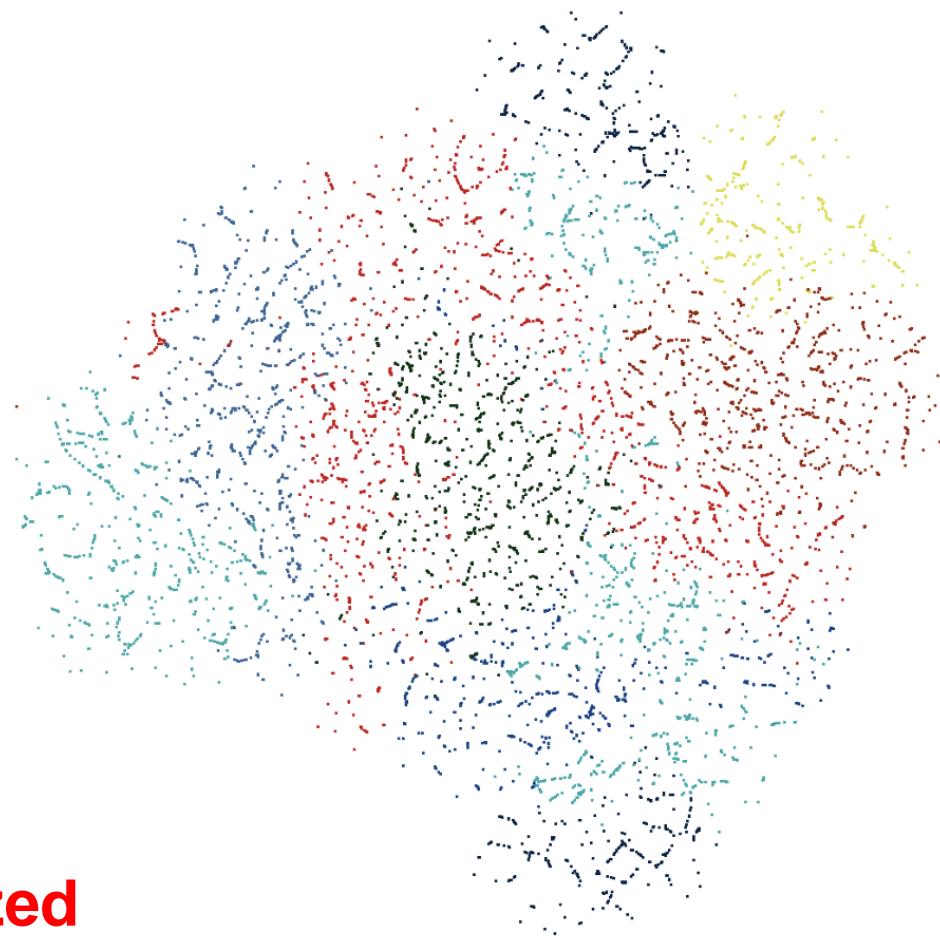
- **n_neighbors**
 - # approximate nearest neighbors used to construct the initial high-D graph.
 - controls how UMAP balances local versus global structure
- **min_dist:**
 - minimum distance between points in low-D space.
 - controls how tightly UMAP clumps points together
 - low values = tightly packed embeddings.

UMAP Parameters

Original 3D Data



2D UMAP Projection



Local structure emphasized
Tightly packed low-D

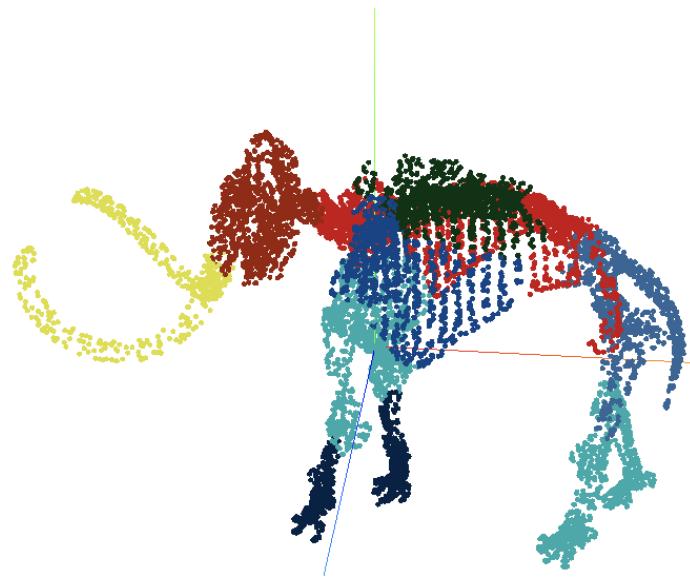
3D

n_neighbors: 3

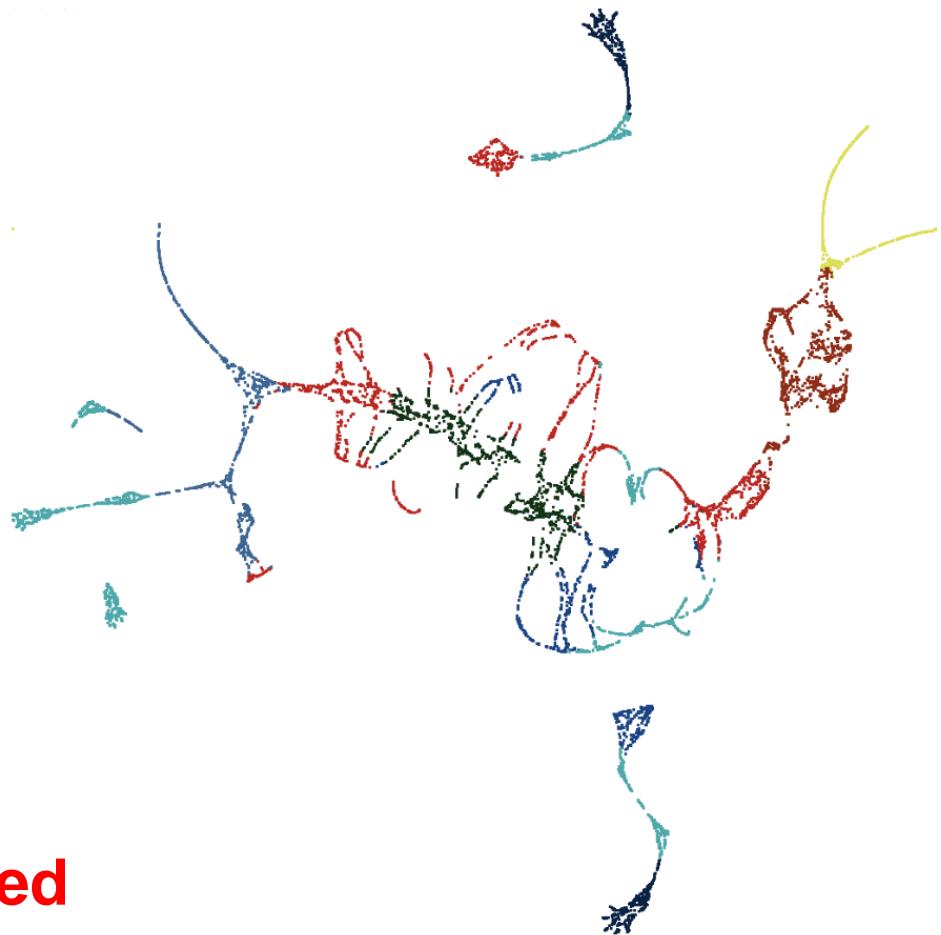
min_dist: 0.0

UMAP Parameters

Original 3D Data



2D UMAP Projection



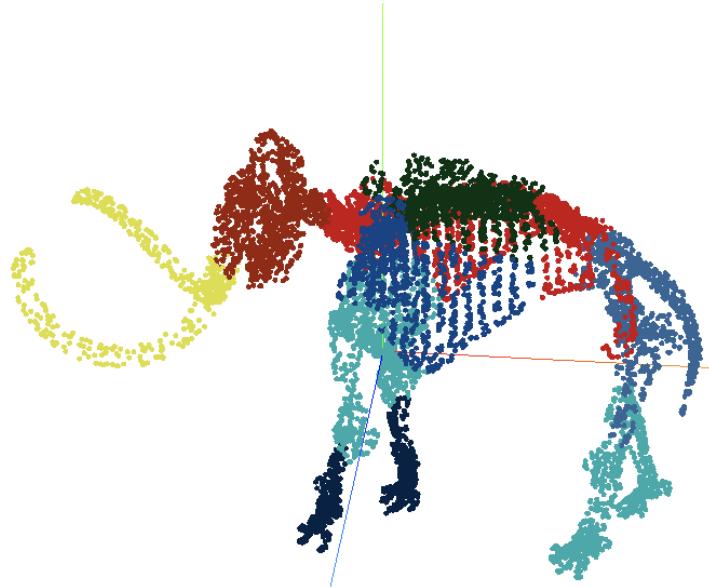
**Global structure emphasized
Tightly packed low-D**

3D

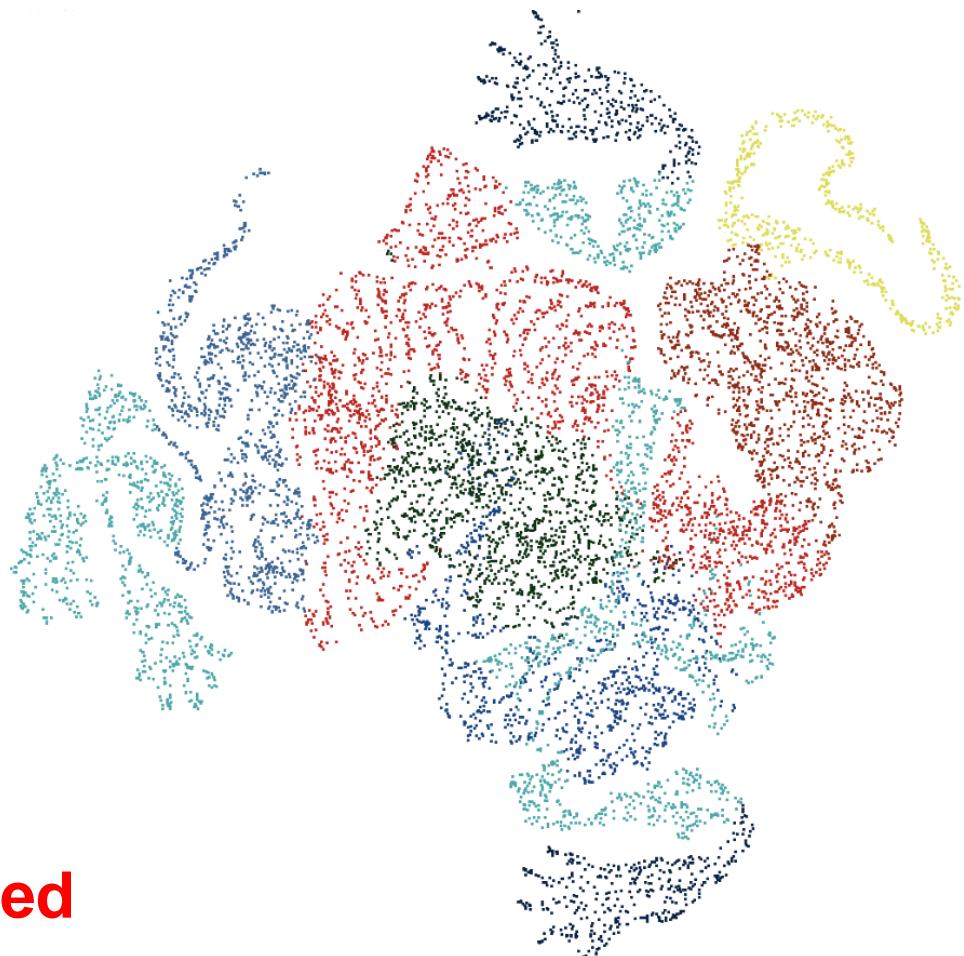
n_neighbors: 200
min_dist: 0.0

UMAP Parameters

Original 3D Data



2D UMAP Projection



Global structure emphasized
Loosely packed low-D

(3D)

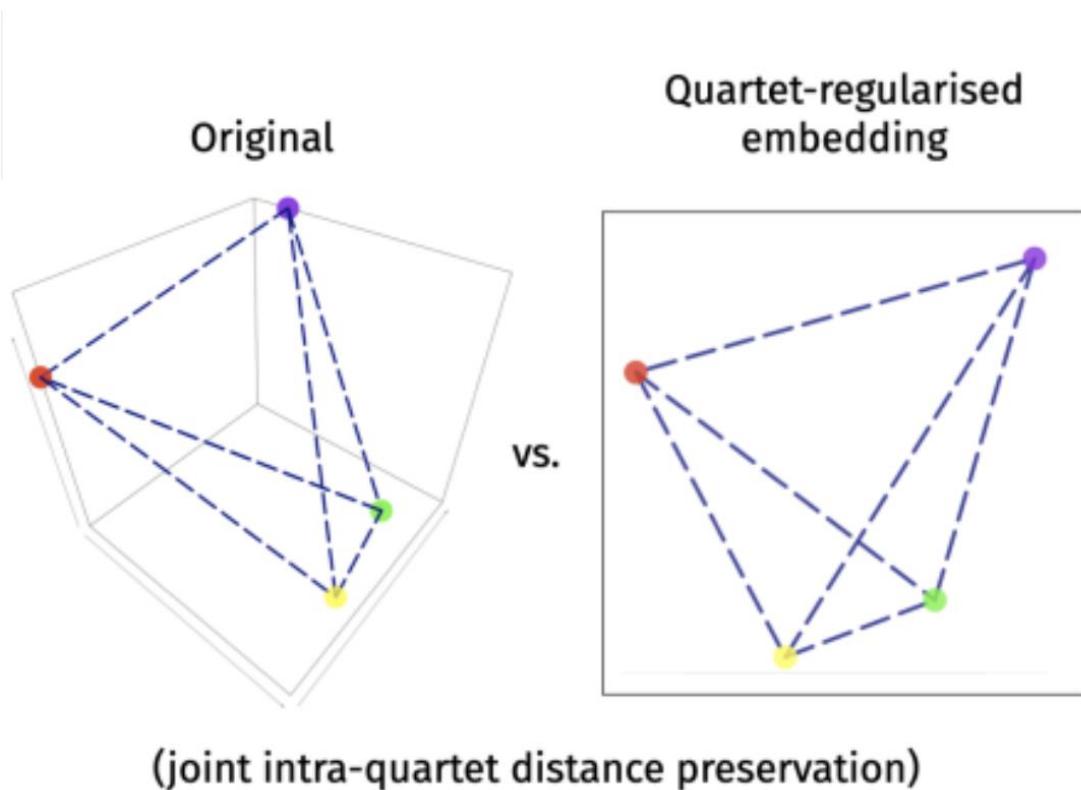
n_neighbors: 200

min_dist: 0.99



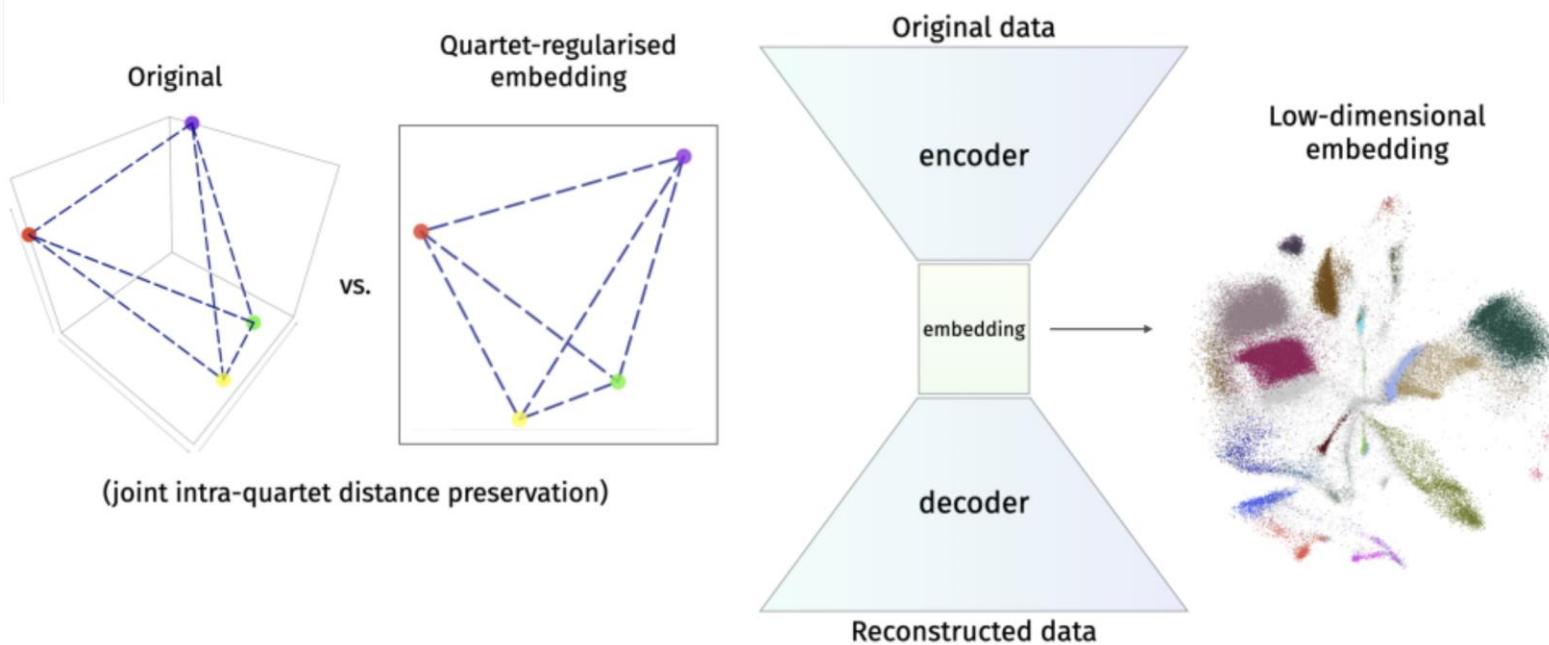
ViVAE (Novak et al. bioRxiv, 2024)

- Imposes a structure-preserving (Quartet loss):
 - Preserve relative distances within quartets (groups of 4) of points that are randomly (repeatedly) drawn



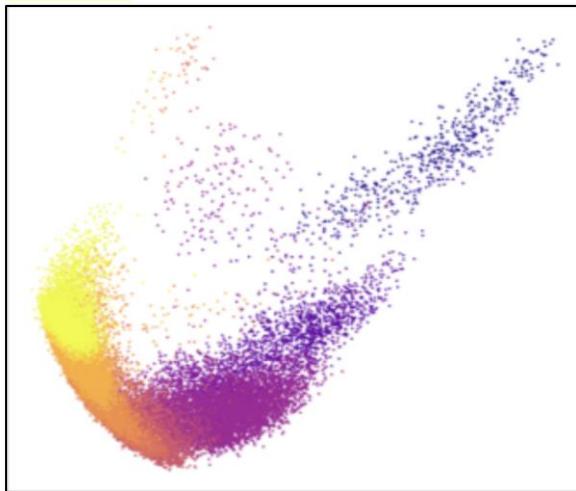
ViVAE (Novak et al. bioRxiv, 2024)

- Uses a variational autoencoder (VAE), to optimise both local and global distances between points.
- VAE is trained to optimise all the intra-quartet distances jointly at each iteration.



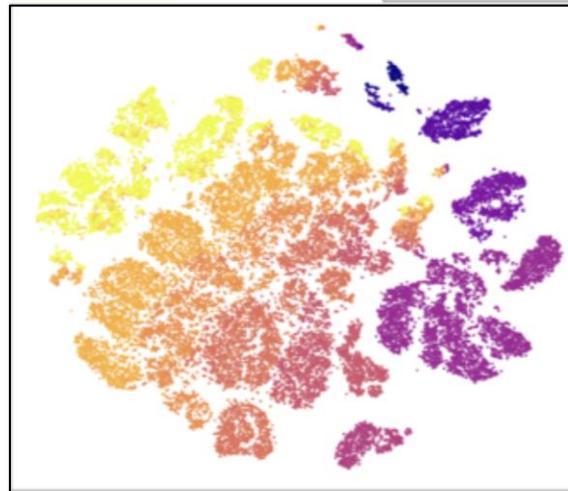
Comparison of approaches (zebrafish embryo)

PCA



t-SNE

(best S_L)

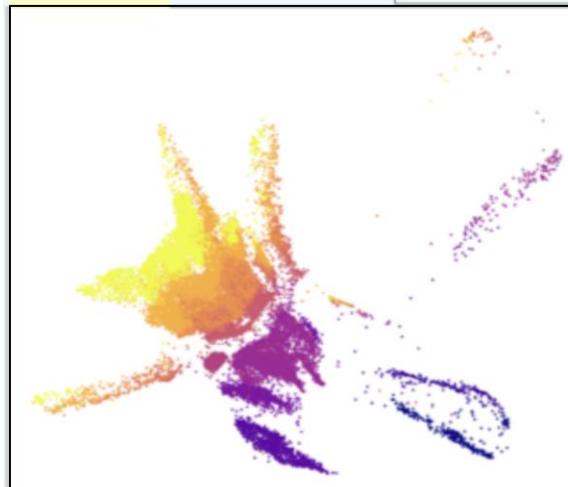


UMAP



VivAE

(best F)



Early

- 03.3-HIGH
- 03.8-OBLONG
- 04.3-DOME
- 04.8-30%
- 05.3-50%
- 06.0-SHIELD
- 07.0-60%
- 08.0-75%
- 09.0-90%
- 10.0-BUD
- 11.0-3-Somite
- 12.0-6-Somite

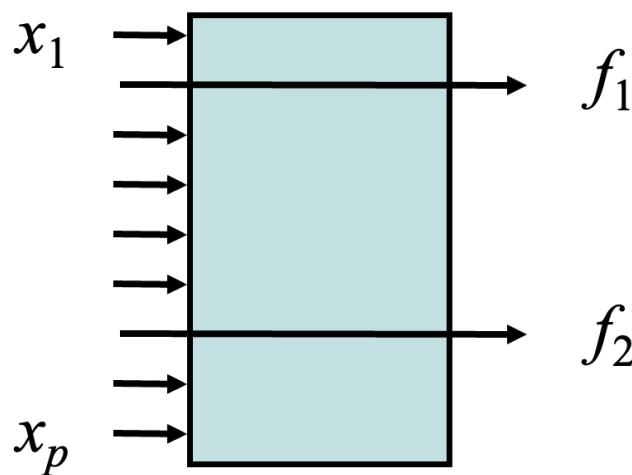
Late

MDS conclusions

- Experts or measurements provide distances
- Optimise a *stress-function* (MDS) or KL distance (t-SNE)
- Important:
 - *the distance measure used*: is it representative?
 - *Parameter choices* can influence outcome heavily.
- Remaining problem: embedding new data points
- t-SNE (and now UMAP, ViVAE) are modern techniques to perform representation of data in high-D space in 2D
- Use multiple methods (or multiple parameter settings) to prevent over-interpretation

Supervised Feature selection

- For supervised 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



Feature selection > Criteria > Wrapper

1. **Wrapper:** exact performance measure

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

Feature Selection > Criteria > Wrapper

1. **Wrapper**: direct 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!

Feature Selection > Criteria

1. **Wrapper**: direct 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!

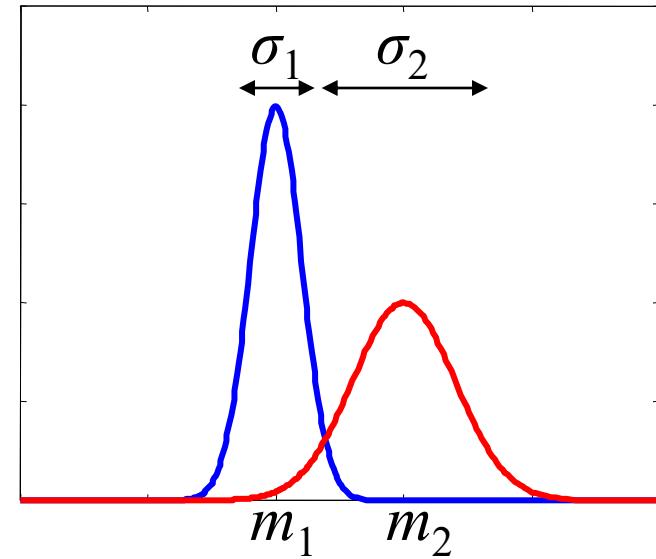
2. **Filter**: approximate performance predictors:

- 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.

Feature Selection > Criteria > Filter

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

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



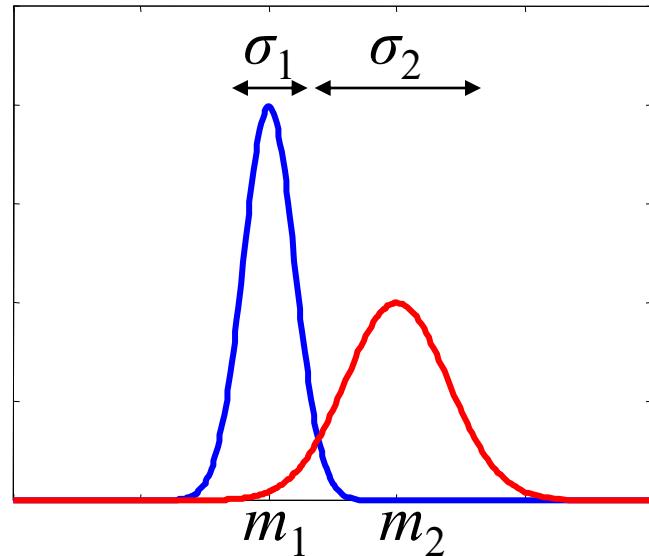
Feature Selection > Criteria > Filter

- 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

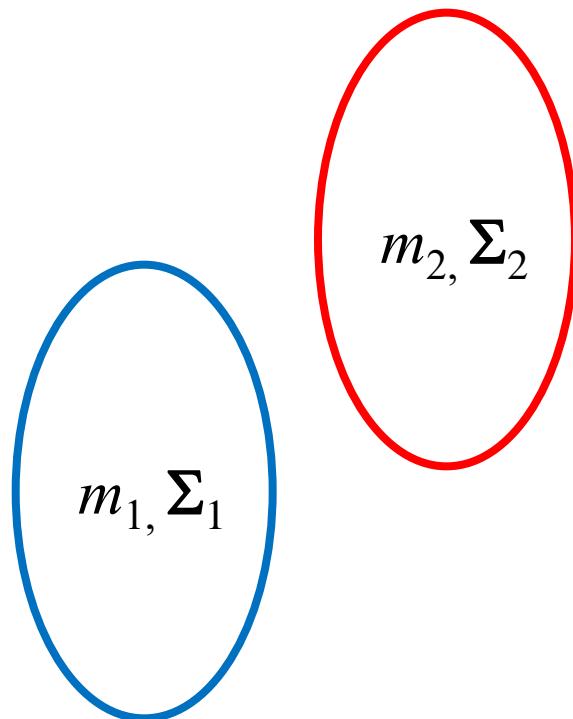


Feature Selection > Criteria > Filter

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

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

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



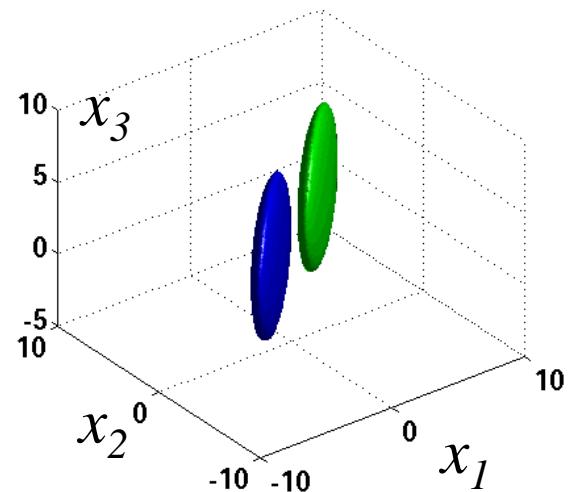
Feature Selection > Search algorithms

- **Feature selection:** select a subset of d out of p features which optimises the criterion
- Brute force solution: consider 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

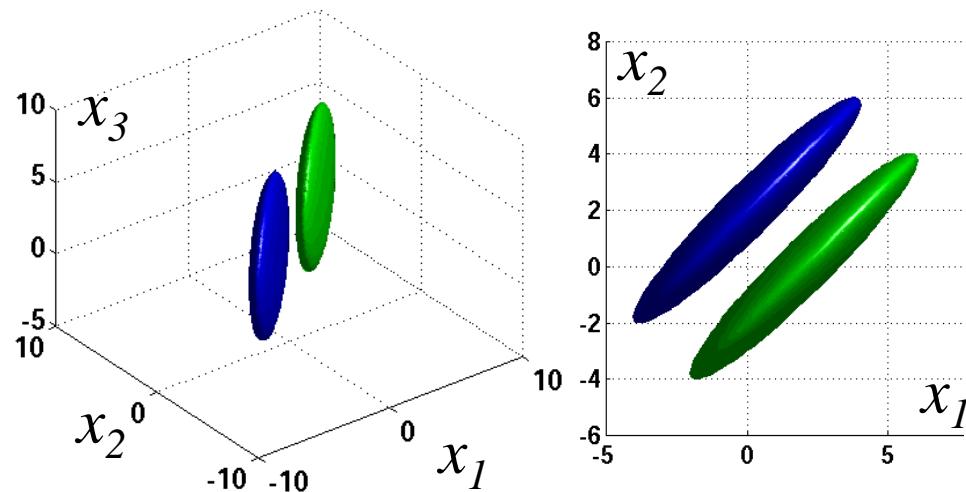
Feature Selection > Search algorithms

- Sub-optimal algorithms:
 - Simplest: d best (individually evaluated)
but these are not necessarily the best d ! (“dB not Bd”)
 - Demonstration: two Gaussians;
select 2 features out of 3 for classification

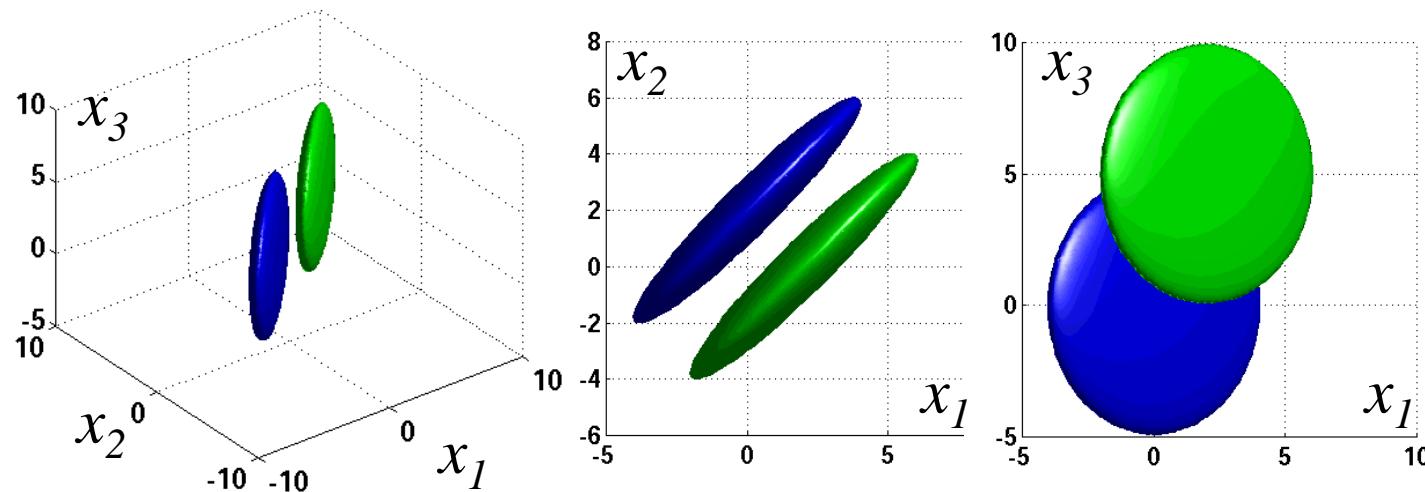
Feature Selection > Search algo's (2B not B2)



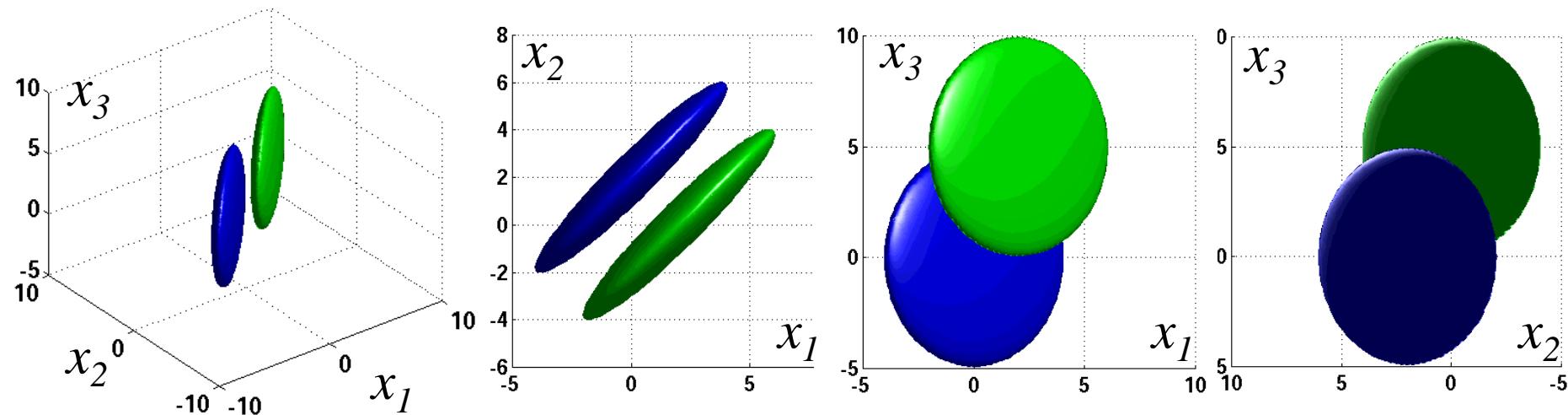
Feature Selection > Search algo's (2B not B2)



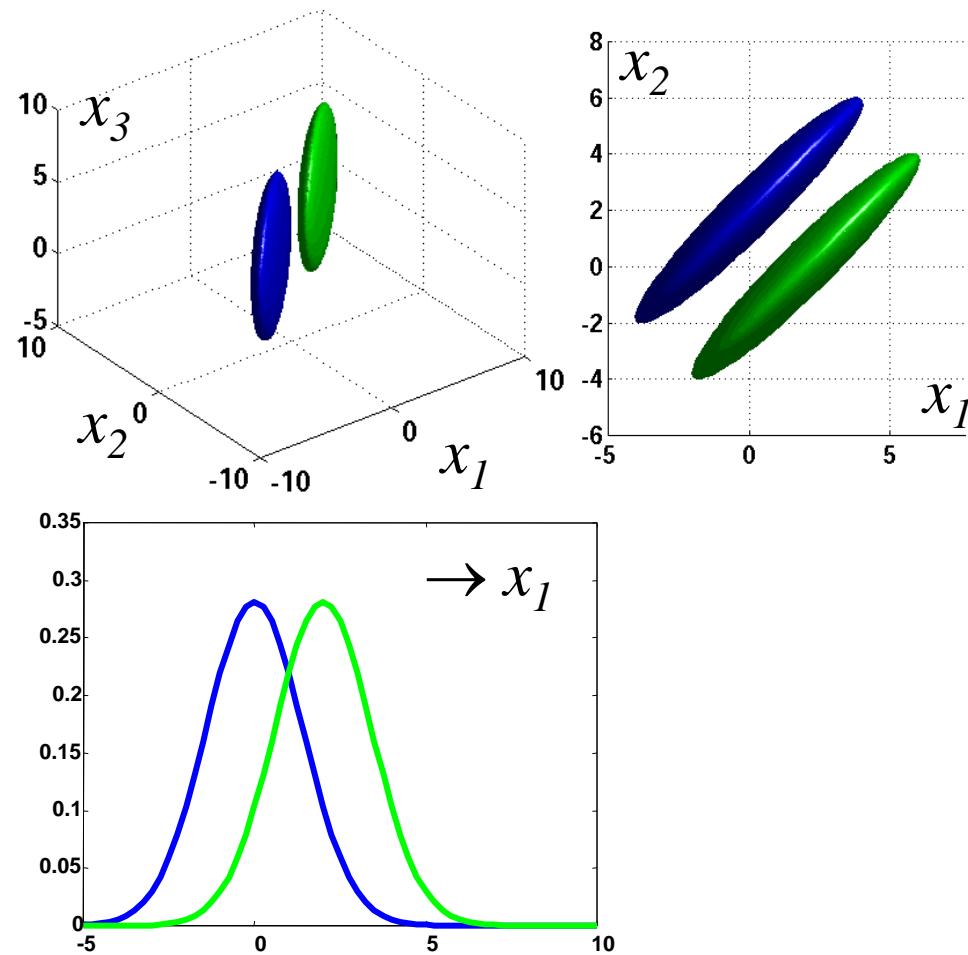
Feature Selection > Search algo's (2B not B2)



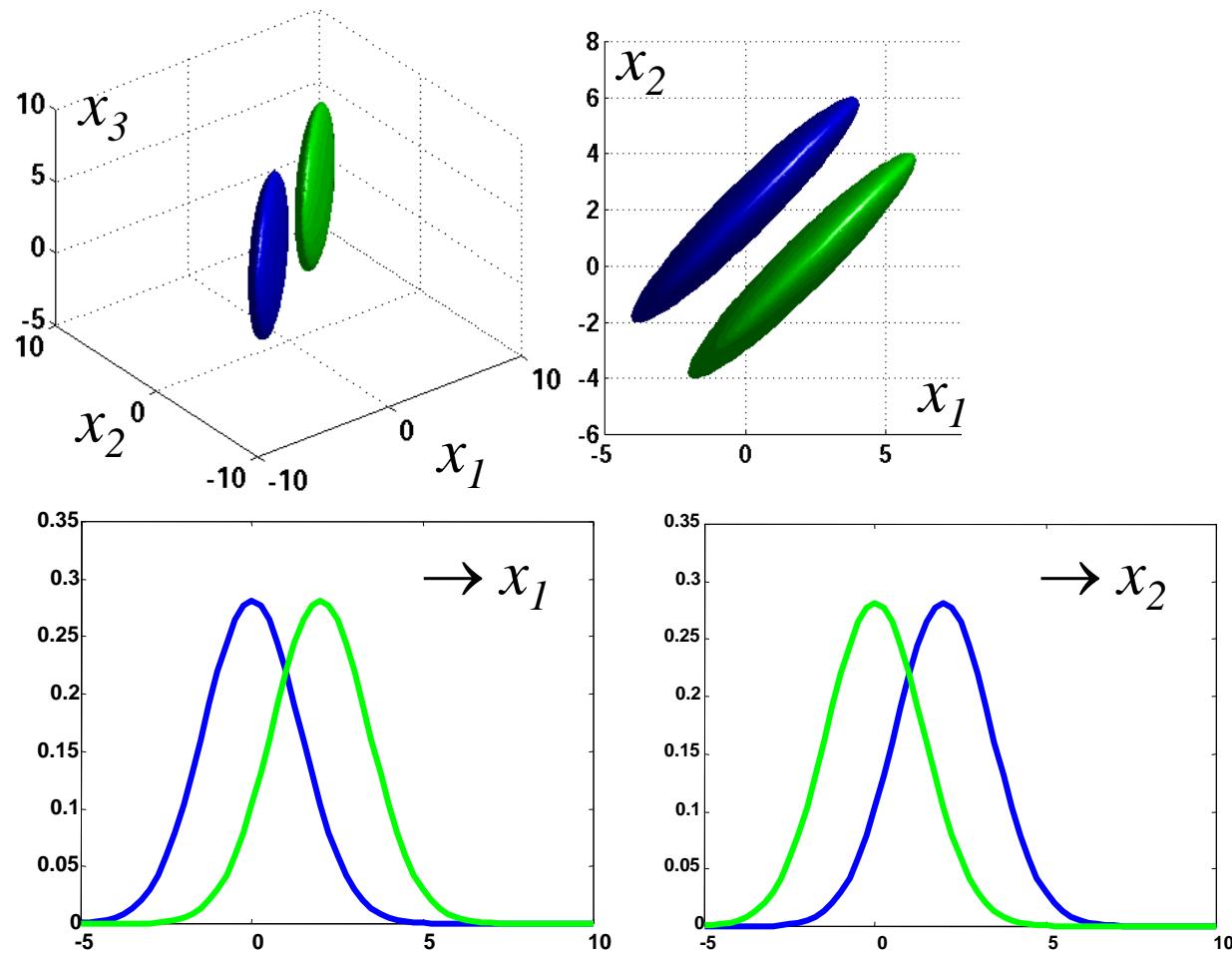
Feature Selection > Search algo's (2B not B2)



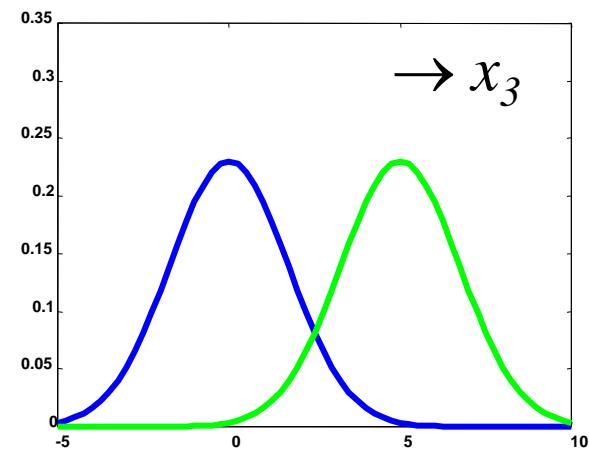
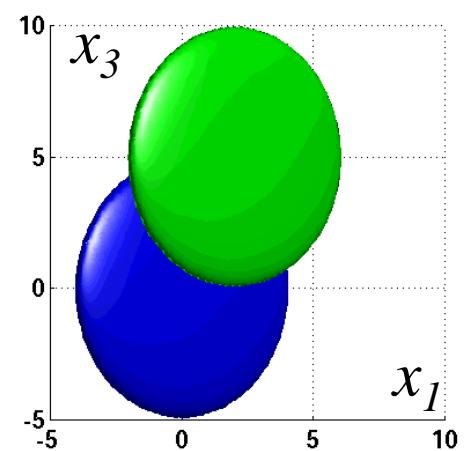
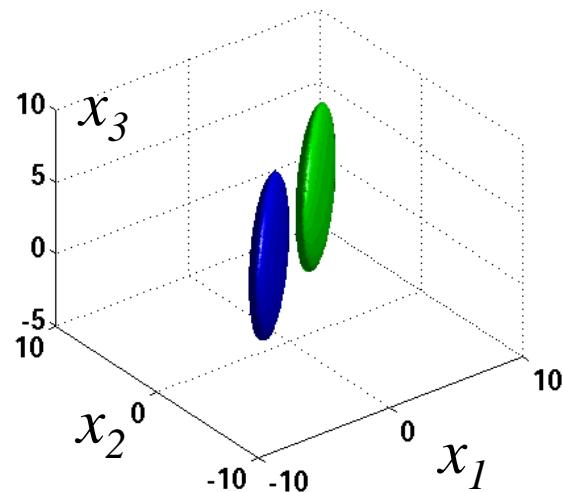
Feature Selection > Search algo's (2B not B2)



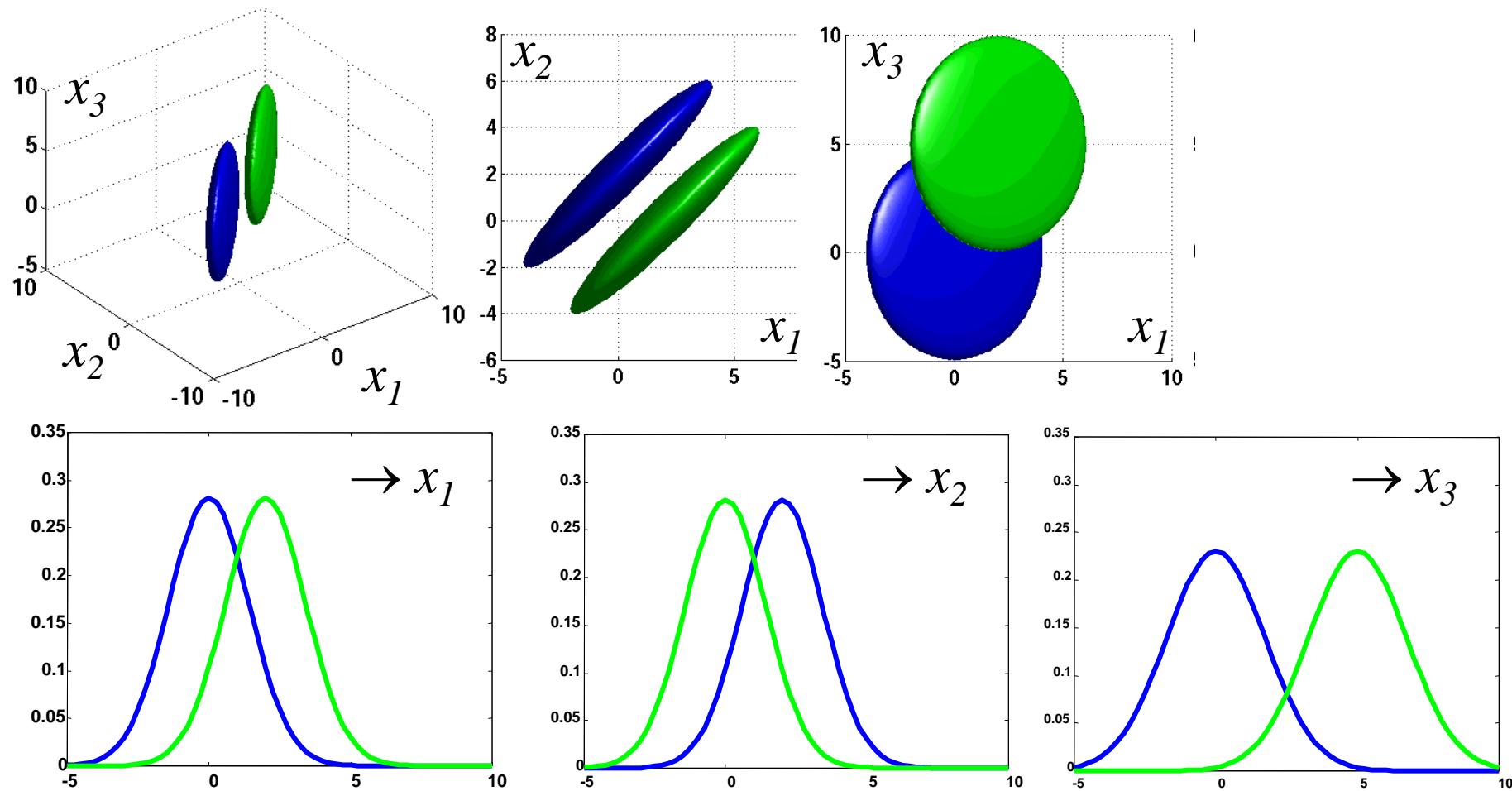
Feature Selection > Search algo's (2B not B2)



Feature Selection > Search algo's (2B not B2)



Feature Selection > Search algo's (2B not B2)



Feature Selection > Search algorithms

- 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

Feature Selection > Search algorithms

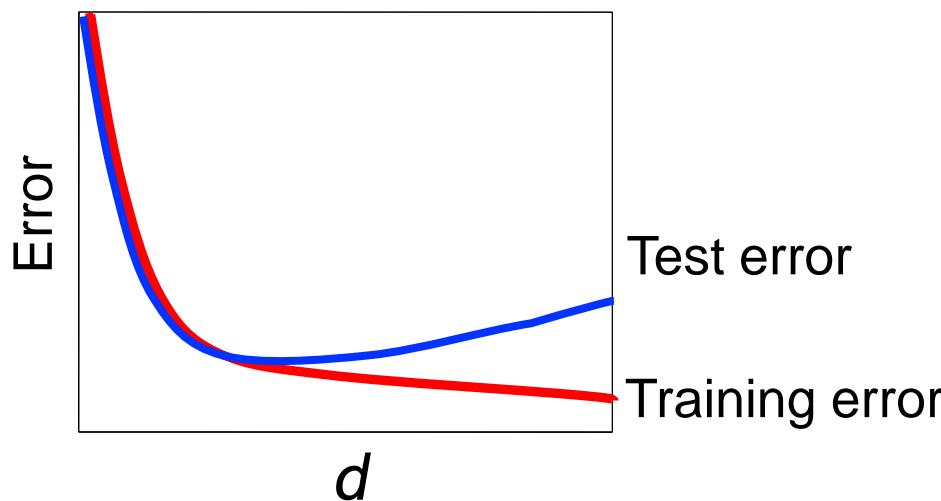
- Other sub-optimal algorithms:
 - Forward selection (for when d is low)
 - start with empty set
 - keep adding one feature at a time
so that the entire subset so far performs best
 - Backward selection (for when d is high)
 - start with entire set
 - keep removing one feature at a time
so that the entire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

Feature Selection > Search algorithms

- 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

Feature Selection > Search algorithms > Stopping

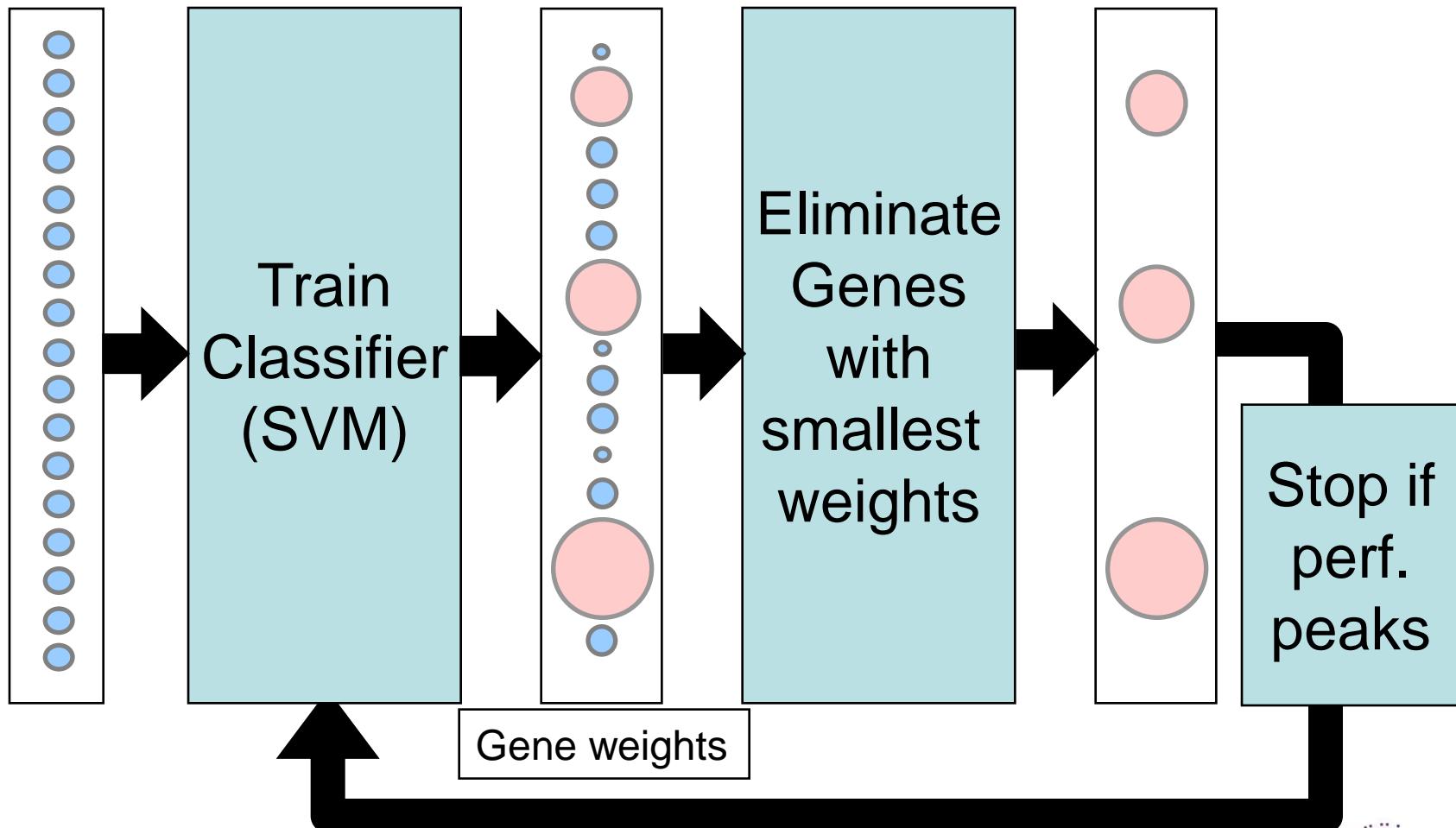
- When should we stop?
 - Due to estimation problems (e.g. covariance matrix), we may be overtraining on training set
 - This is revealed by increasing error on the test set



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

Example: Recursive feature elimination (RFE)

Wrapper, Backward search

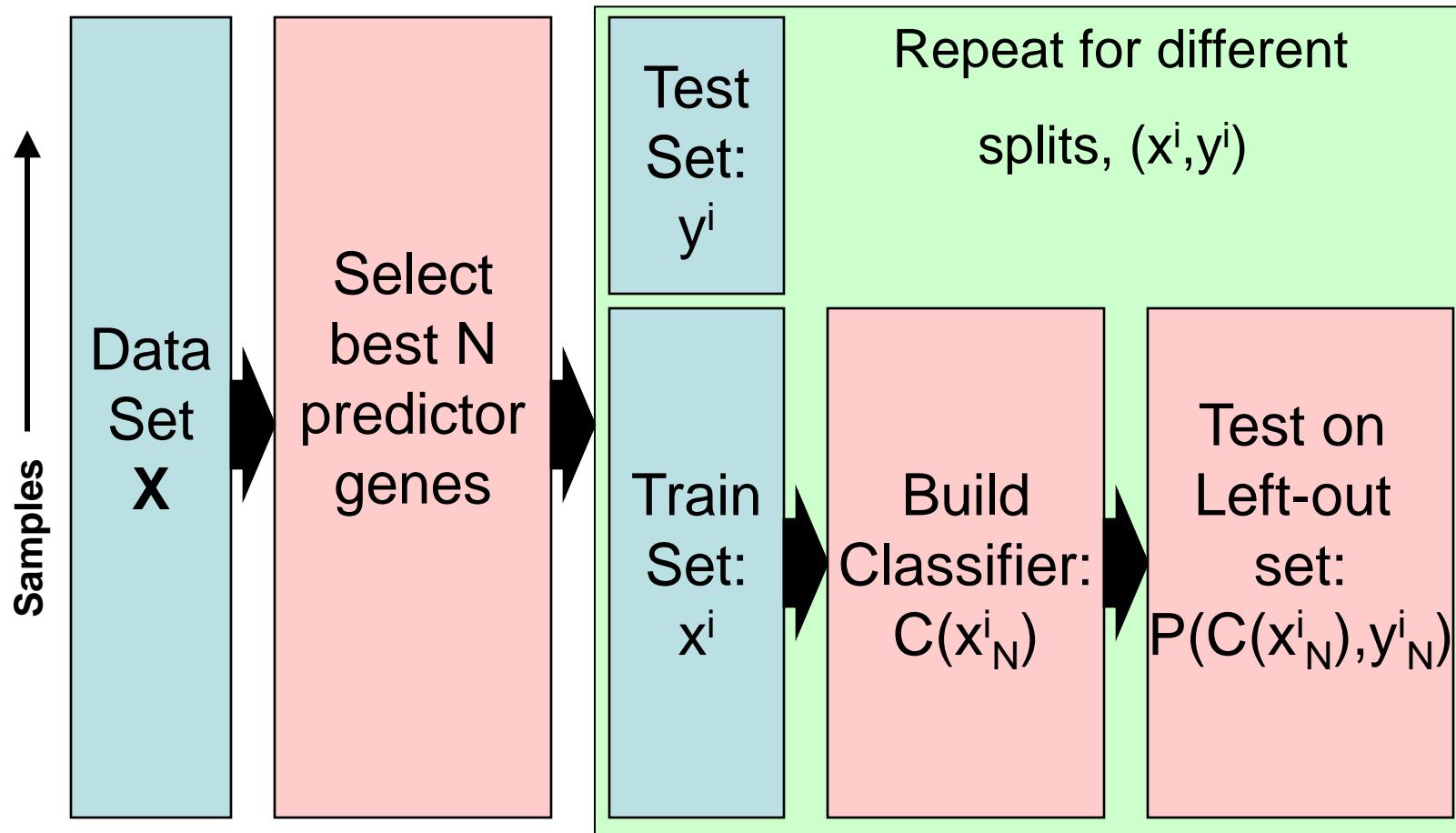


What can go wrong?

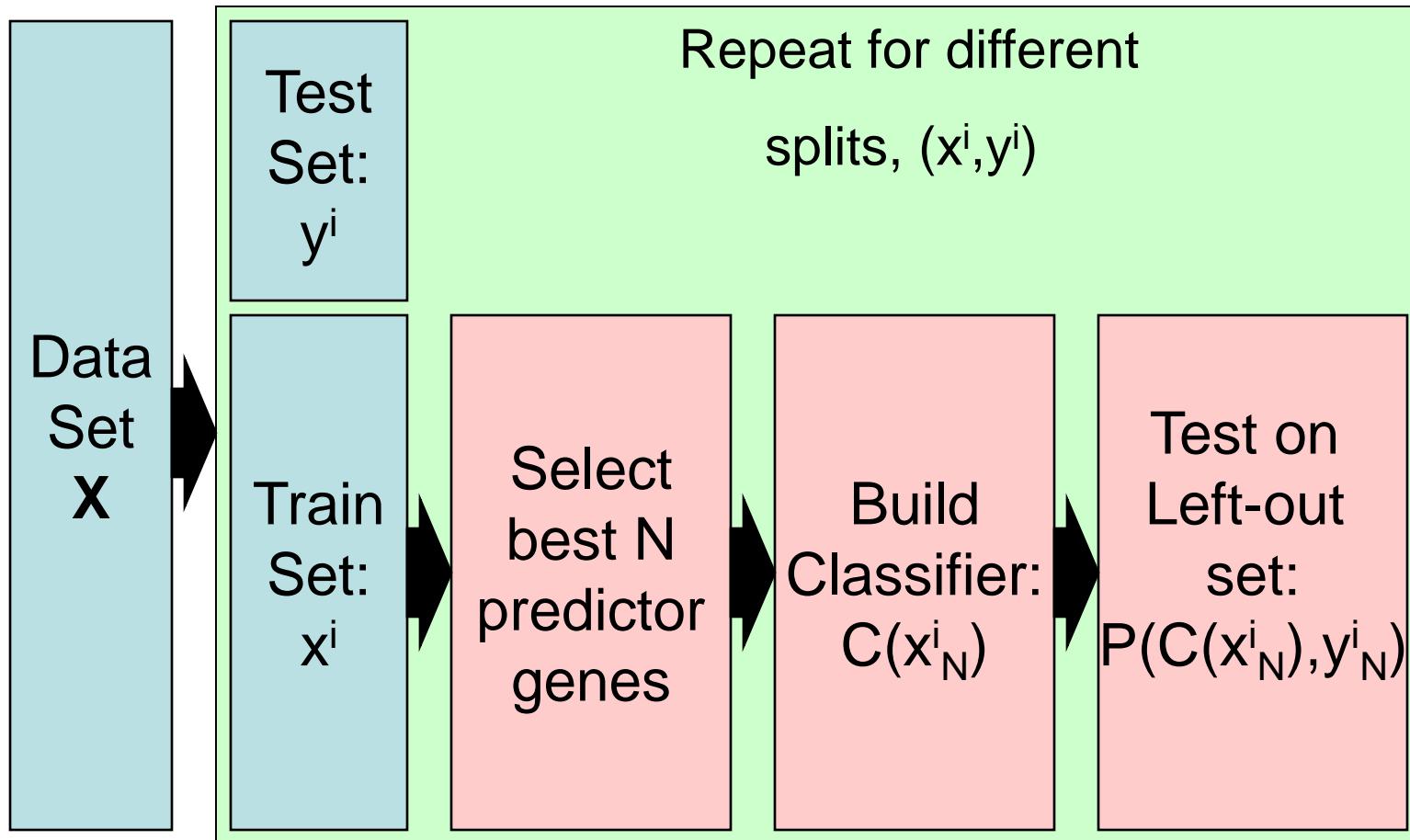
Selection bias...

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

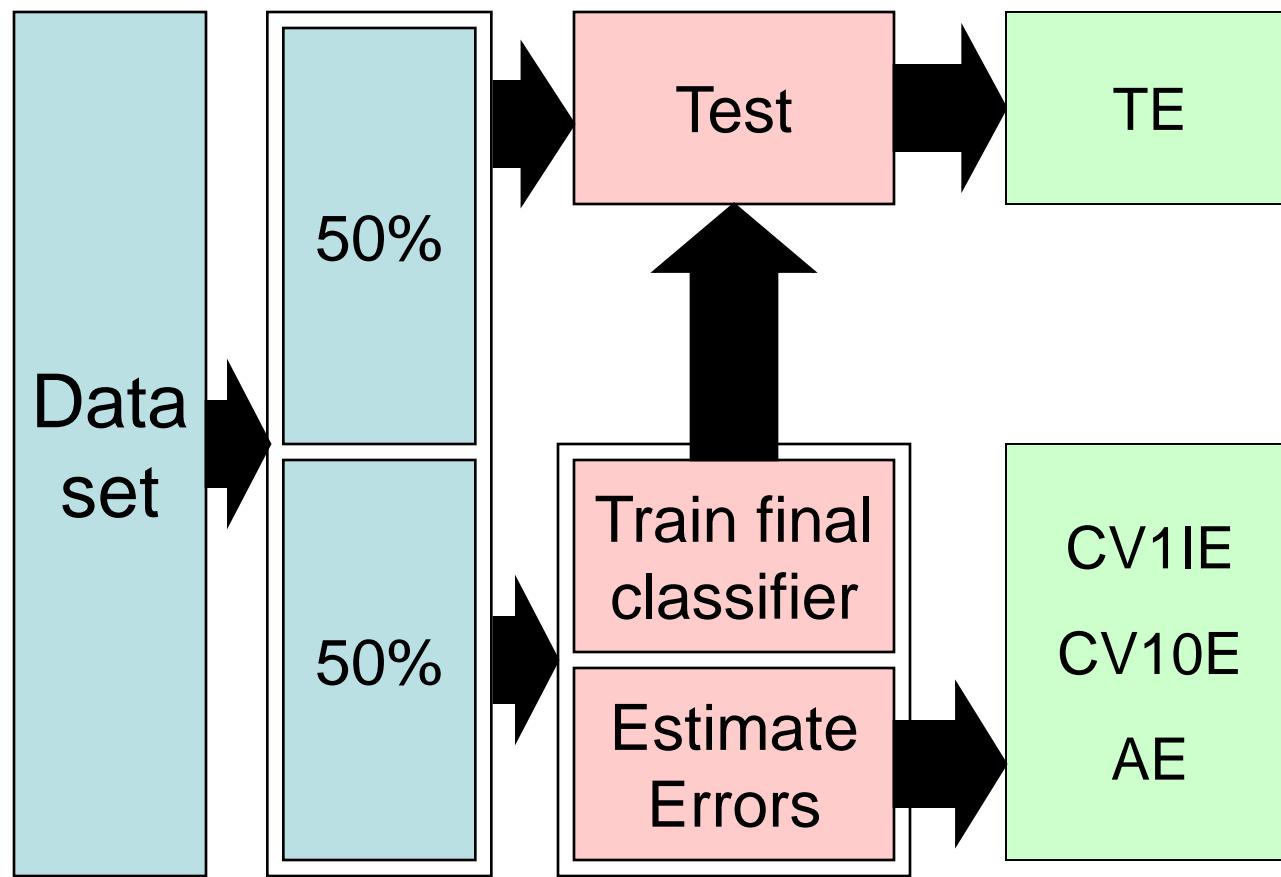
Biased selection



Unbiased selection

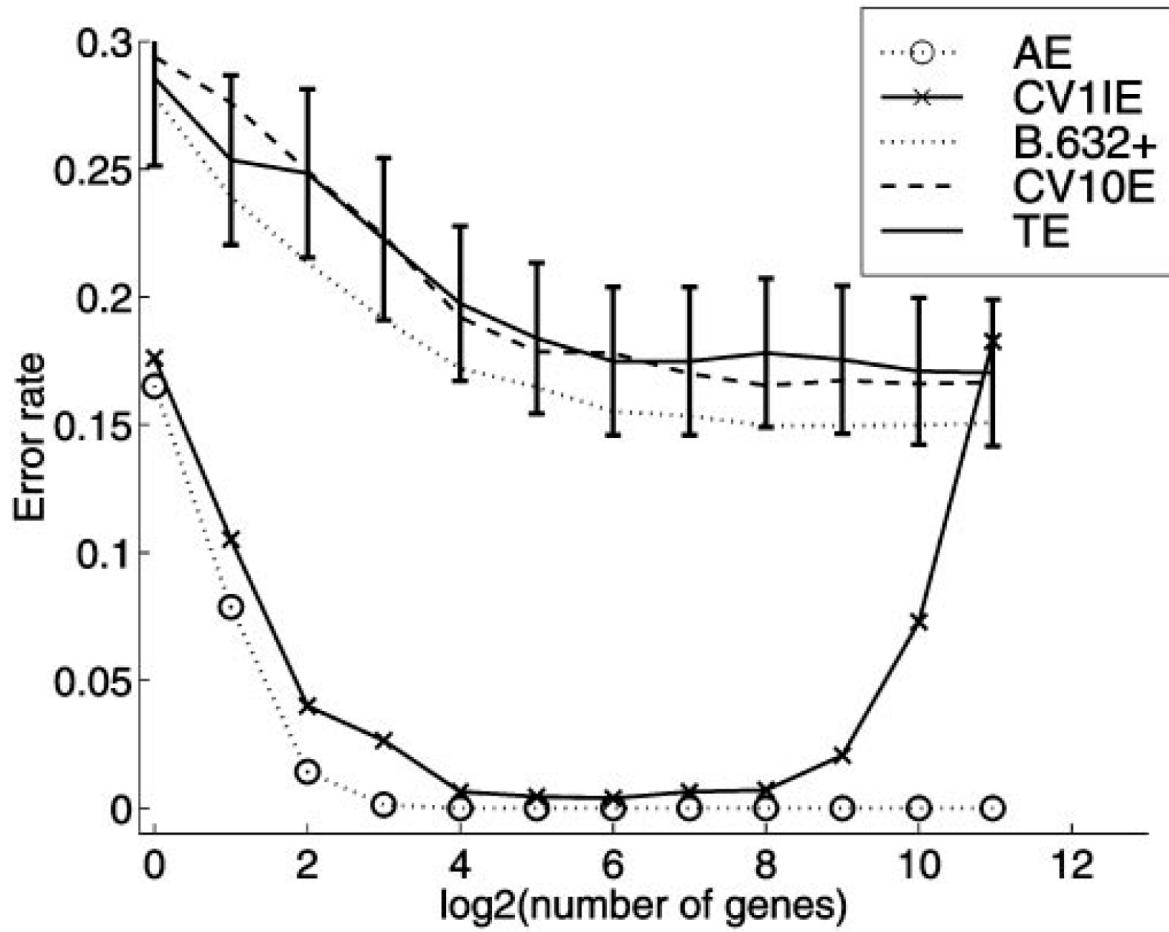


Ambroise & McLachlan experiments



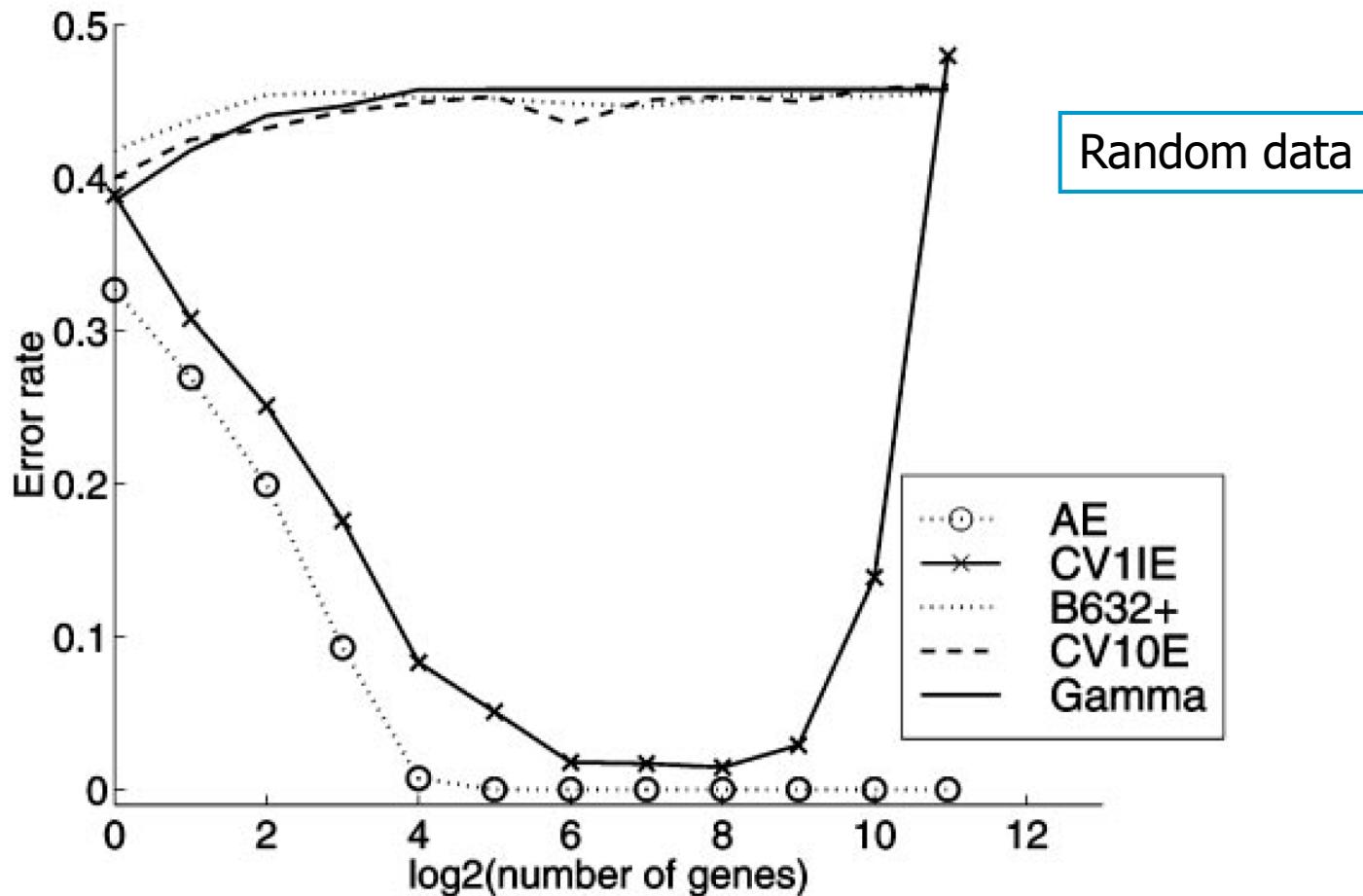
Ambroise & McLachlan experiments

Colon vs. normal data



Ambroise & 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 ' $p >> n$ ' problems, e.g. RNAseq data
- 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: control contribution of genes to classifier based on individual quality and control degree of contribution with cross-validated classification error
- Regularization 2: combines accuracy (error) and penalty on large weights (= simple models) in one criterion.

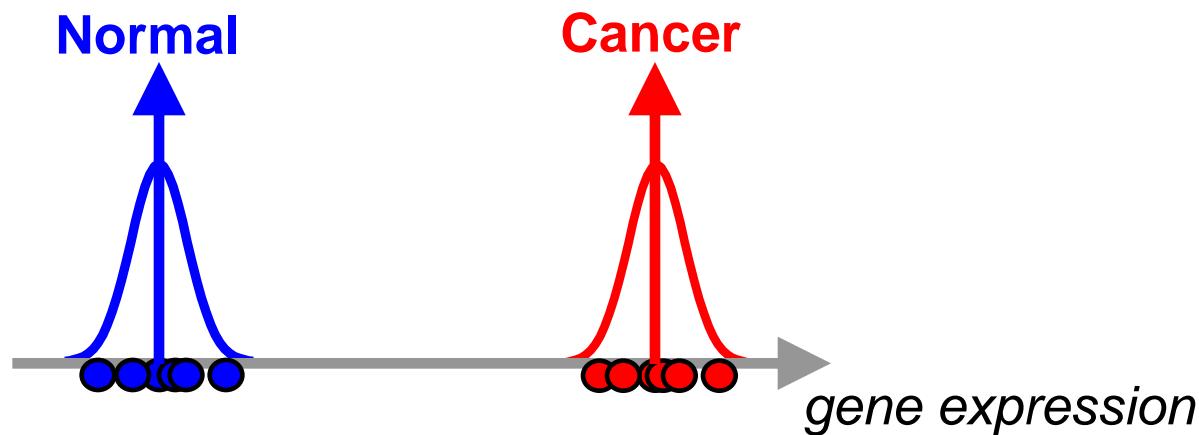
Shrunken centroids

- 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.

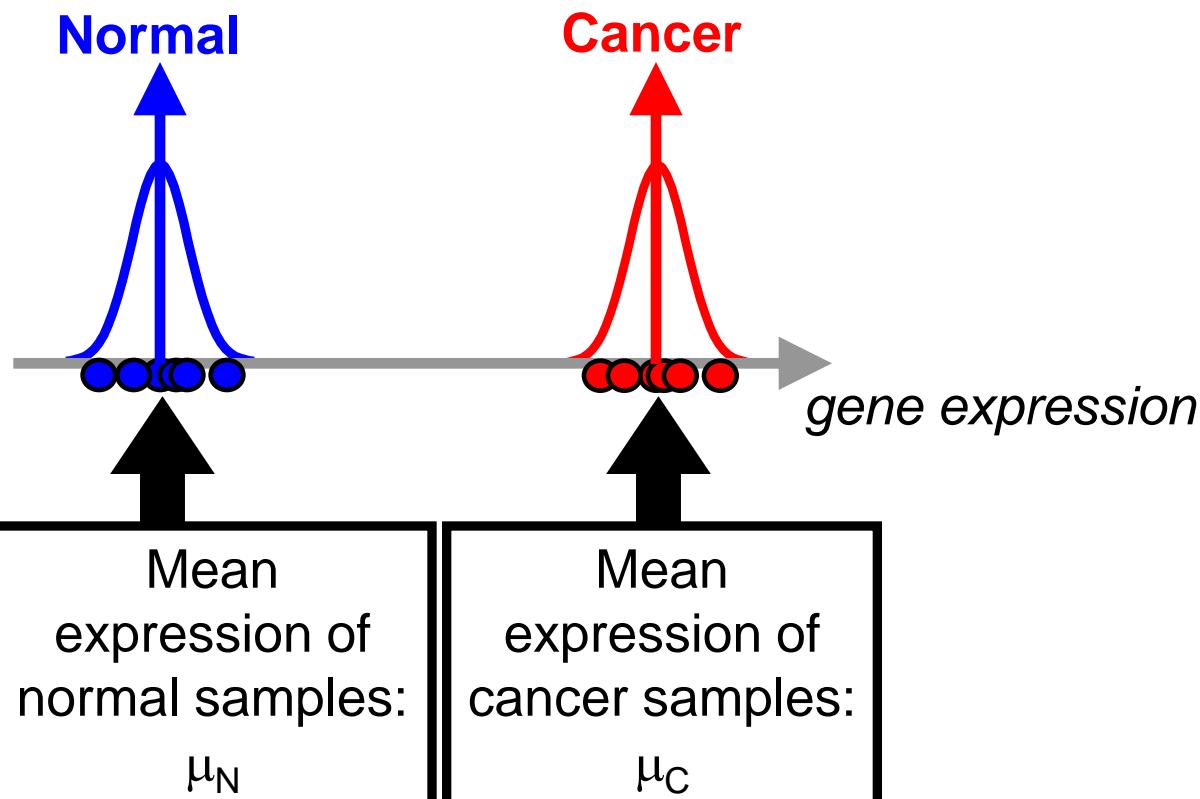
Shrunken centroids (1)

Step 1: Compute class centroids per gene



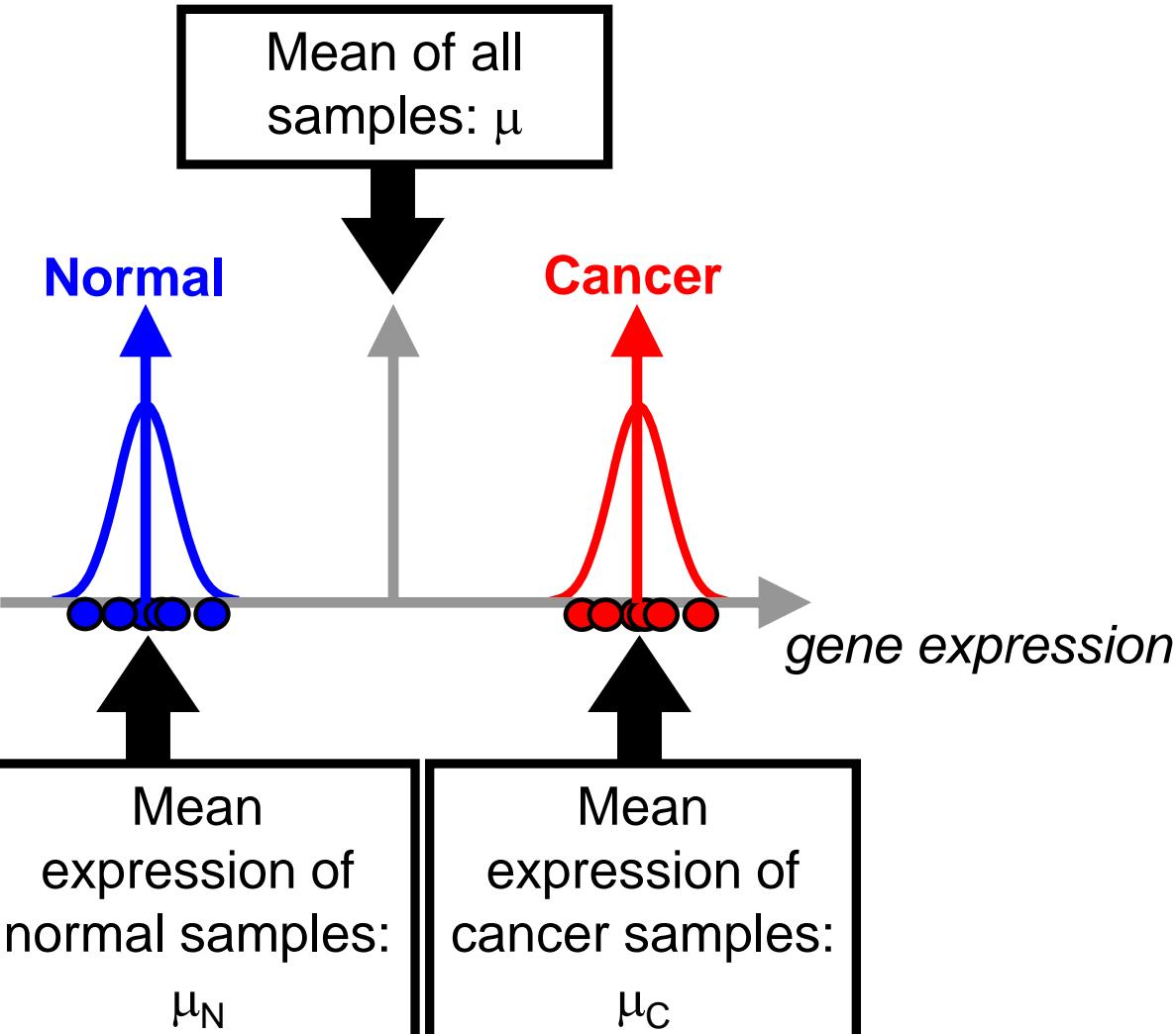
Shrunken centroids (2)

Step 1: Compute class centroids per gene



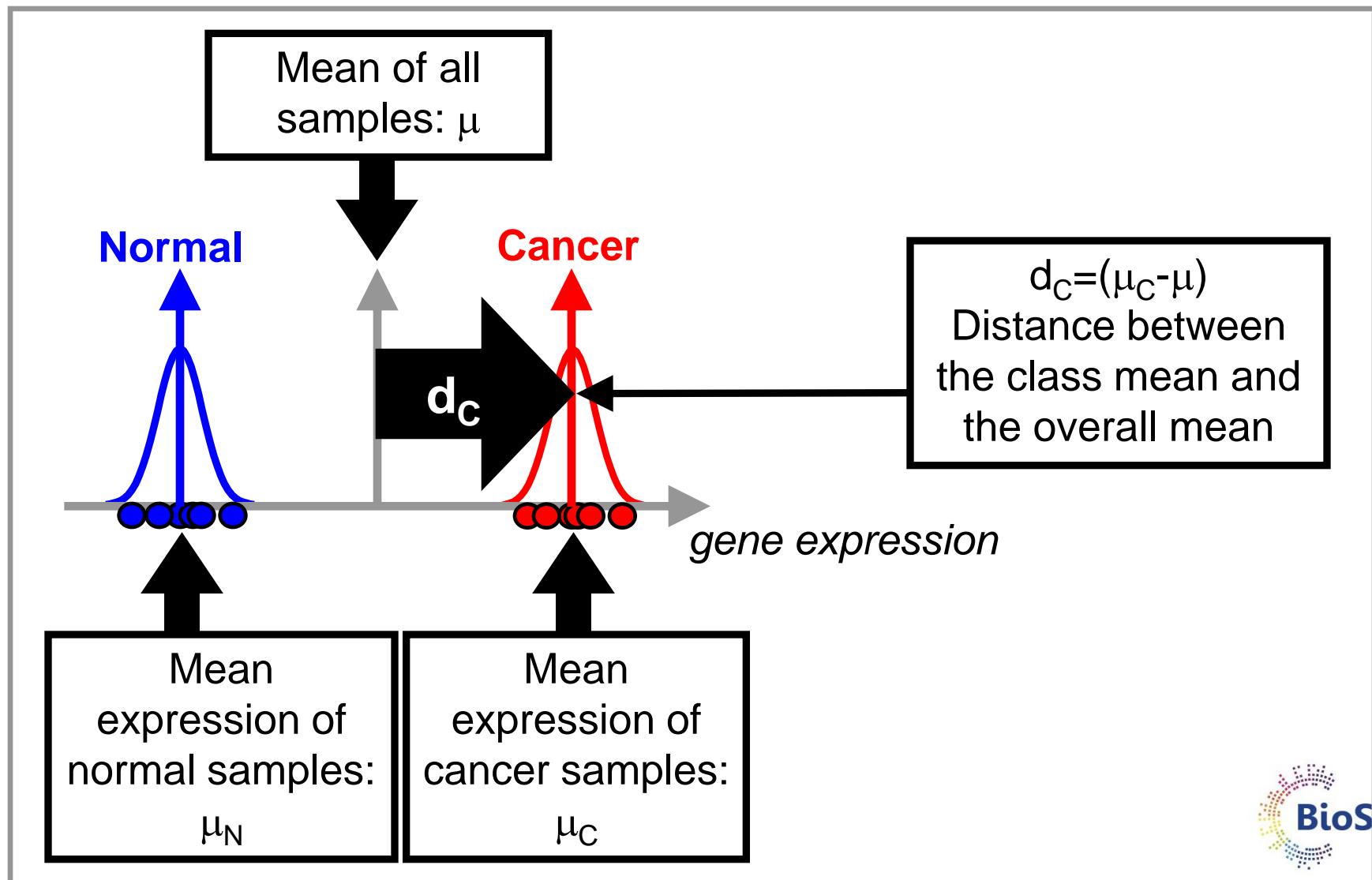
Shrunken centroids (3)

Step 2: Compute overall centroids per gene



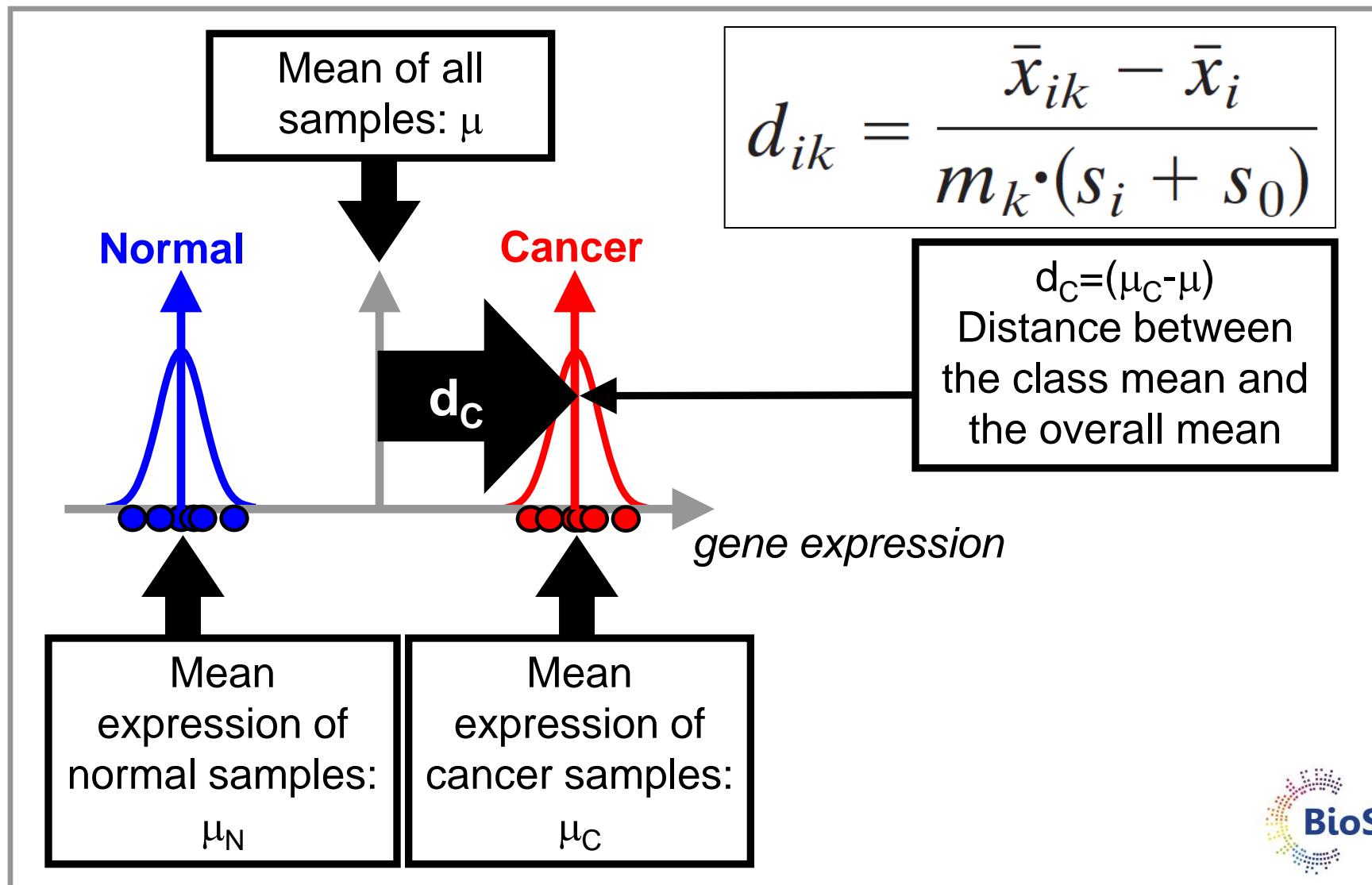
Shrunken centroids (4)

Step 3: Compute d per gene



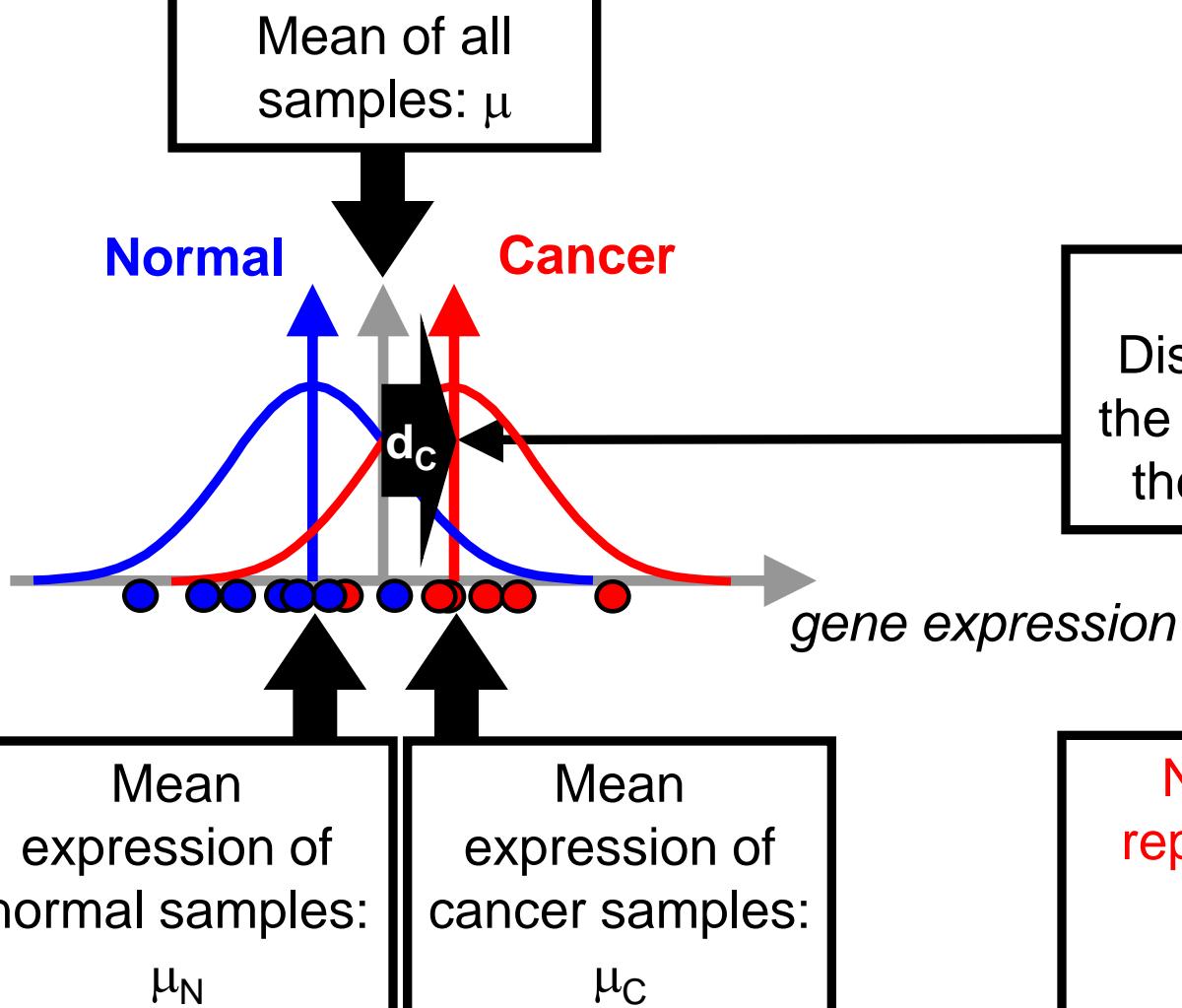
Shrunken centroids (4)

Step 3: Compute d per gene



Shrunken centroids (5)

Step 3: Compute d per gene

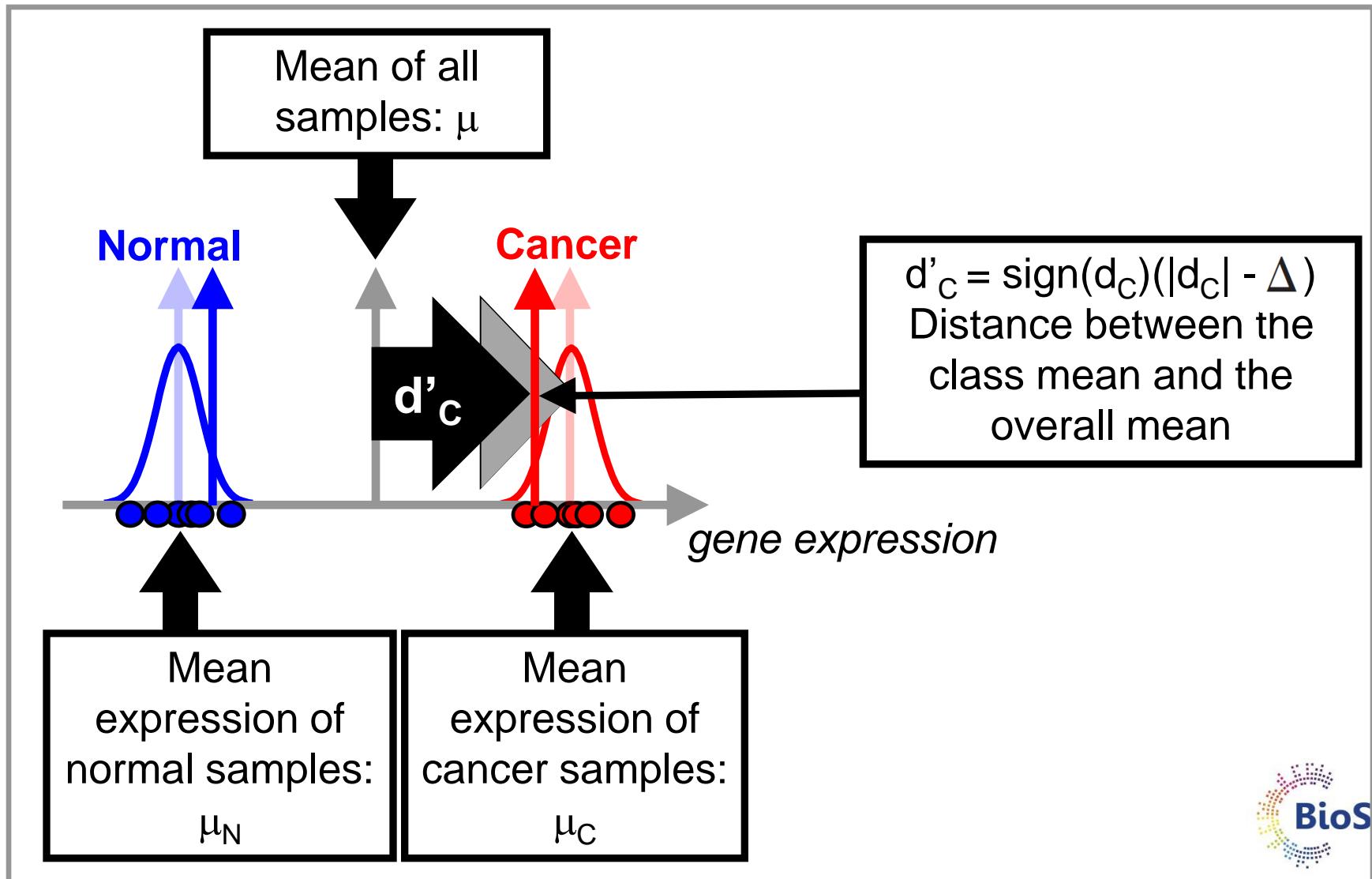


$d_C = (\mu_C - \mu)$
Distance between
the class mean and
the overall mean

Note: for poor
reporters, SNR is
small AND
D is small

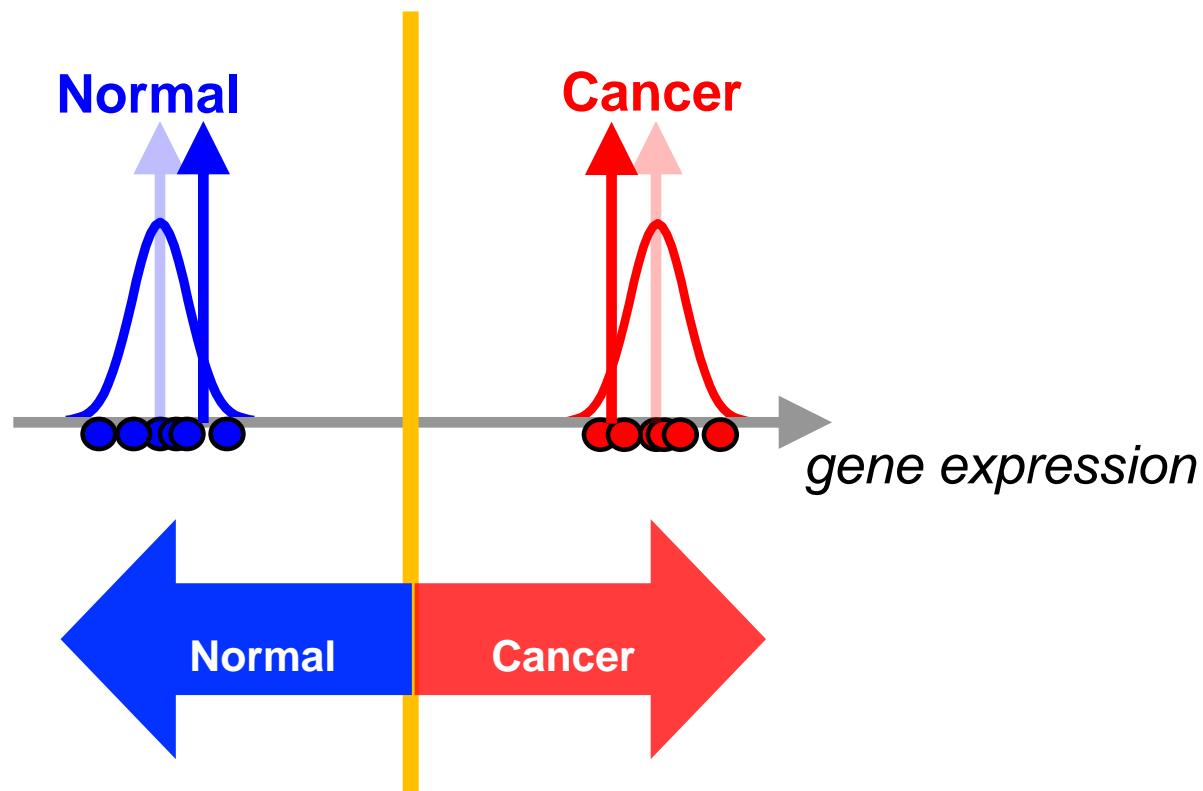
Shrunken centroids (6)

Step 4: Shrink the centroids



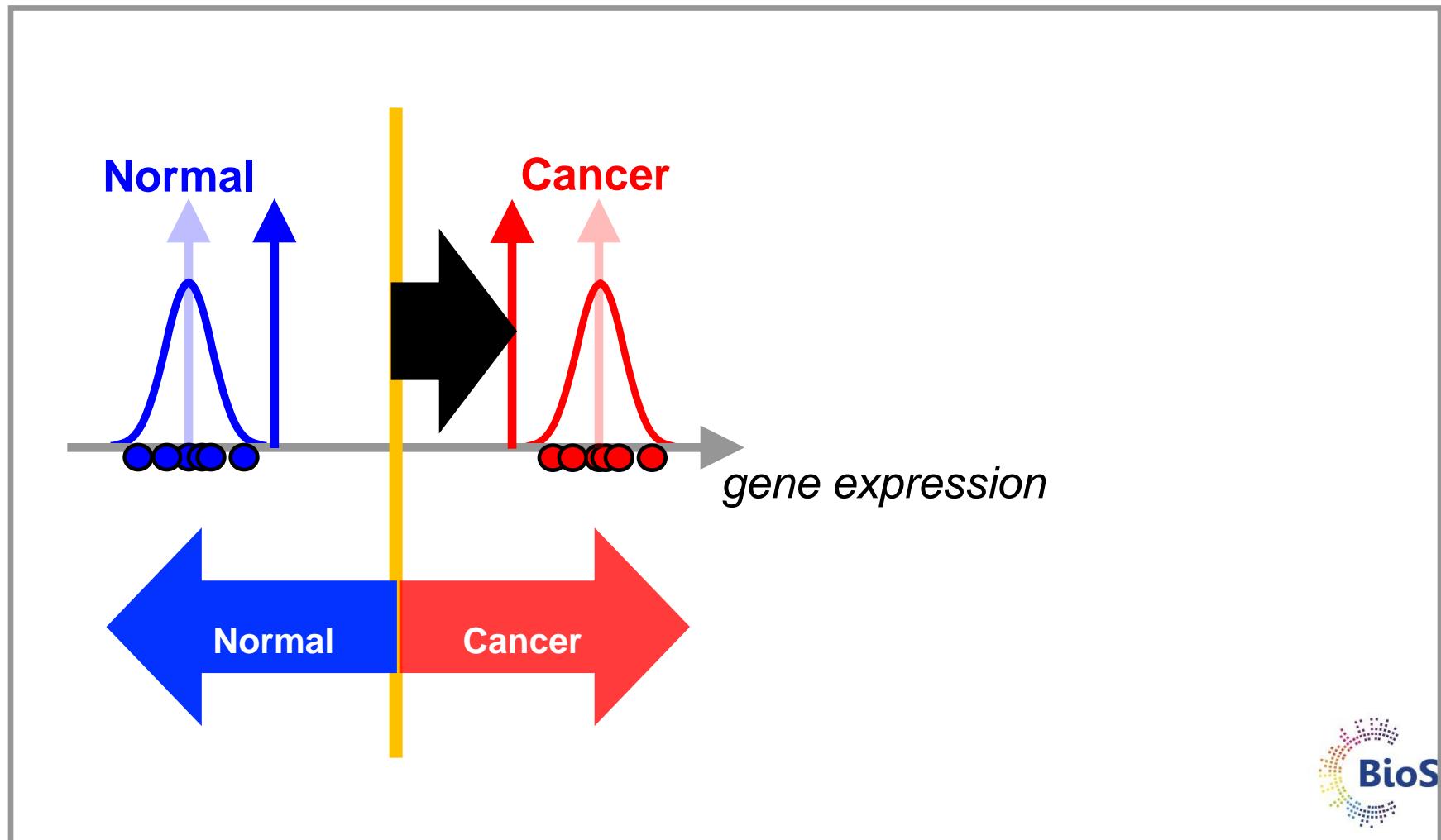
Shrunken centroids (7)

Step 5: Classify with shrunken centroids / perf.



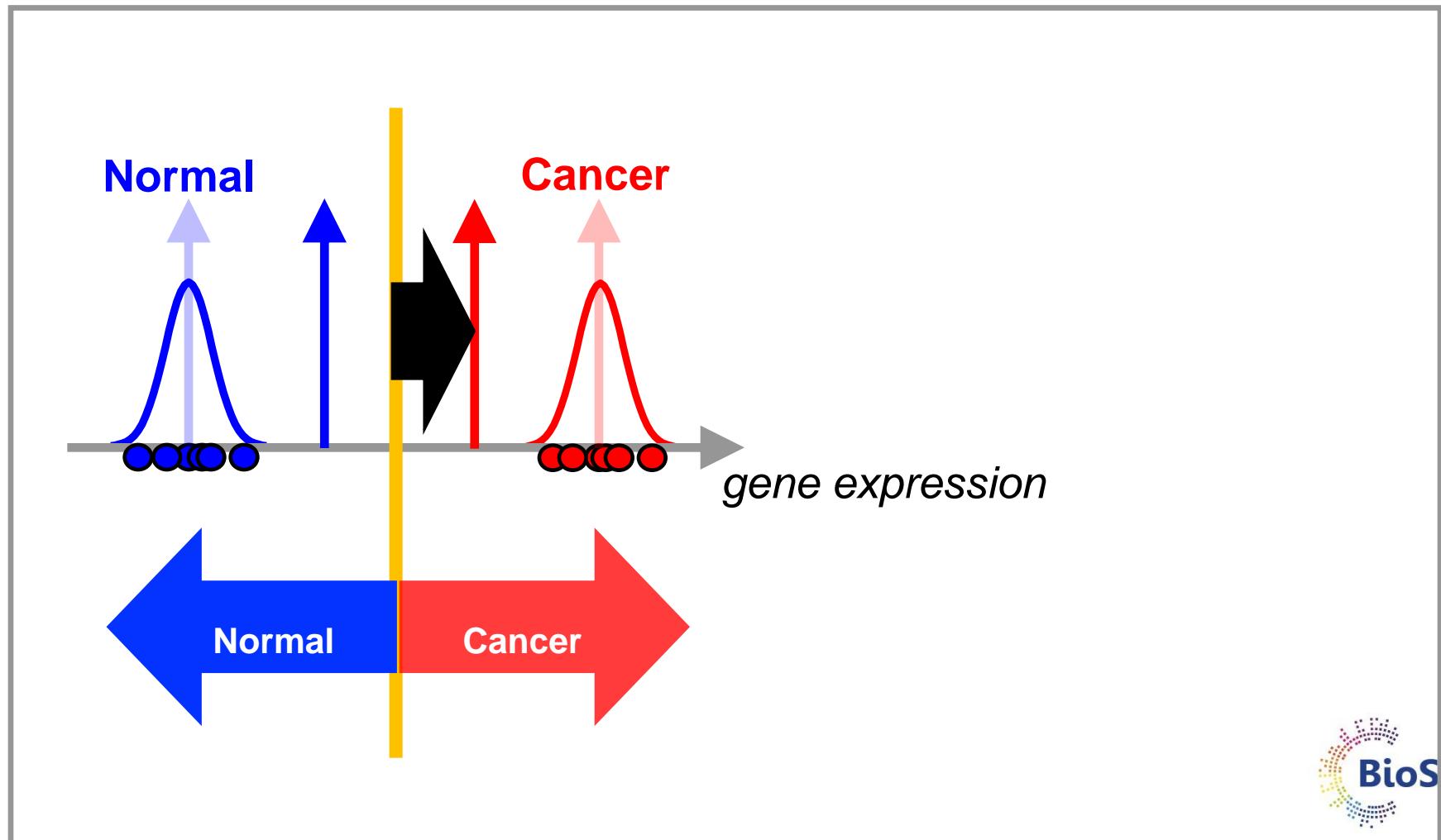
Shrunken centroids (8)

Repeat shrinking and evaluation of classifier till all genes shrunk away



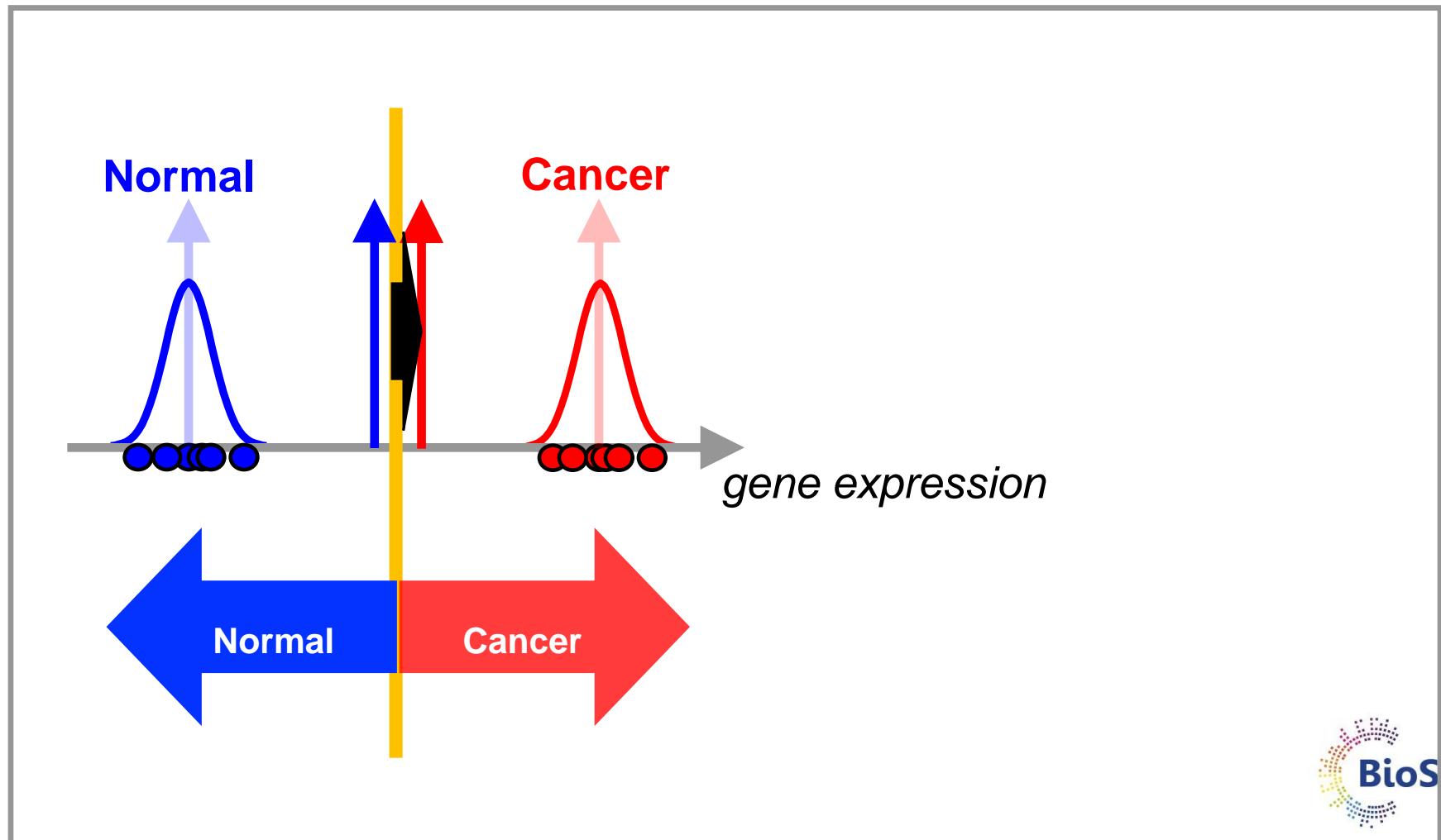
Shrunken centroids (8)

Repeat shrinking and evaluation of classifier till all genes shrunk away



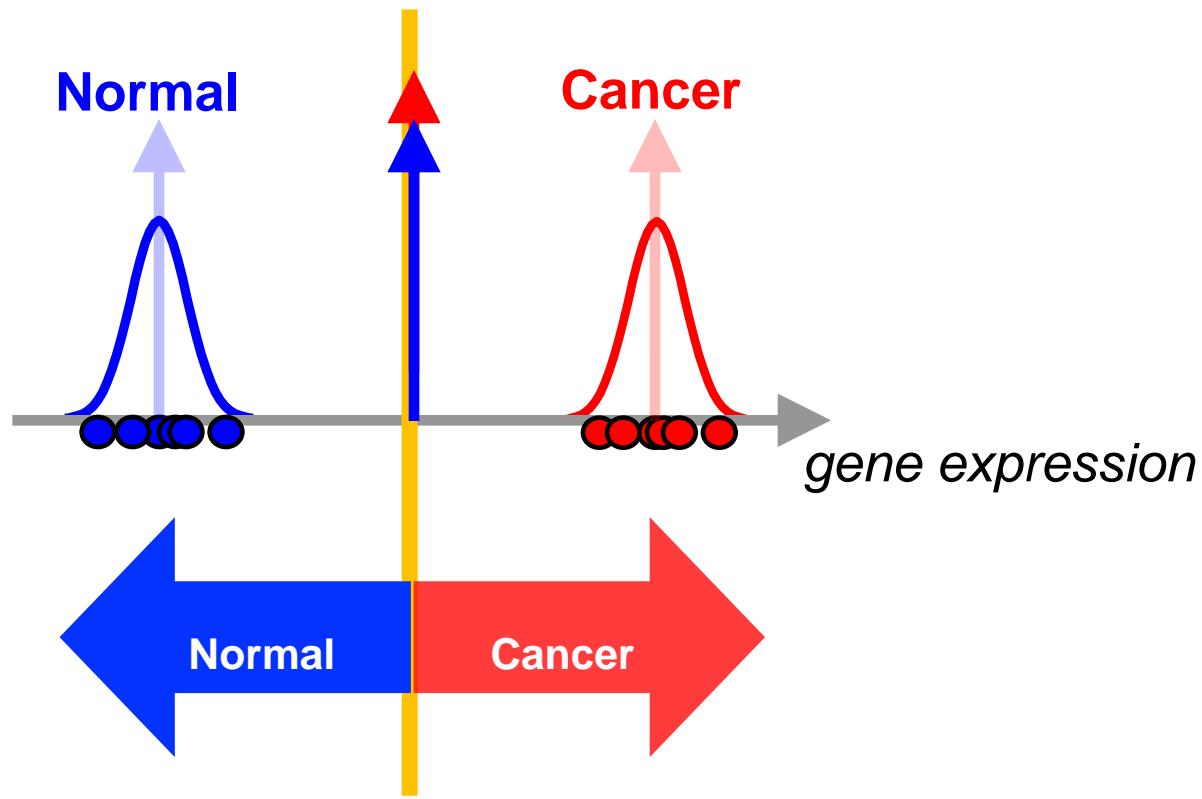
Shrunken centroids (8)

Repeat shrinking and evaluation of classifier till all genes shrunk away

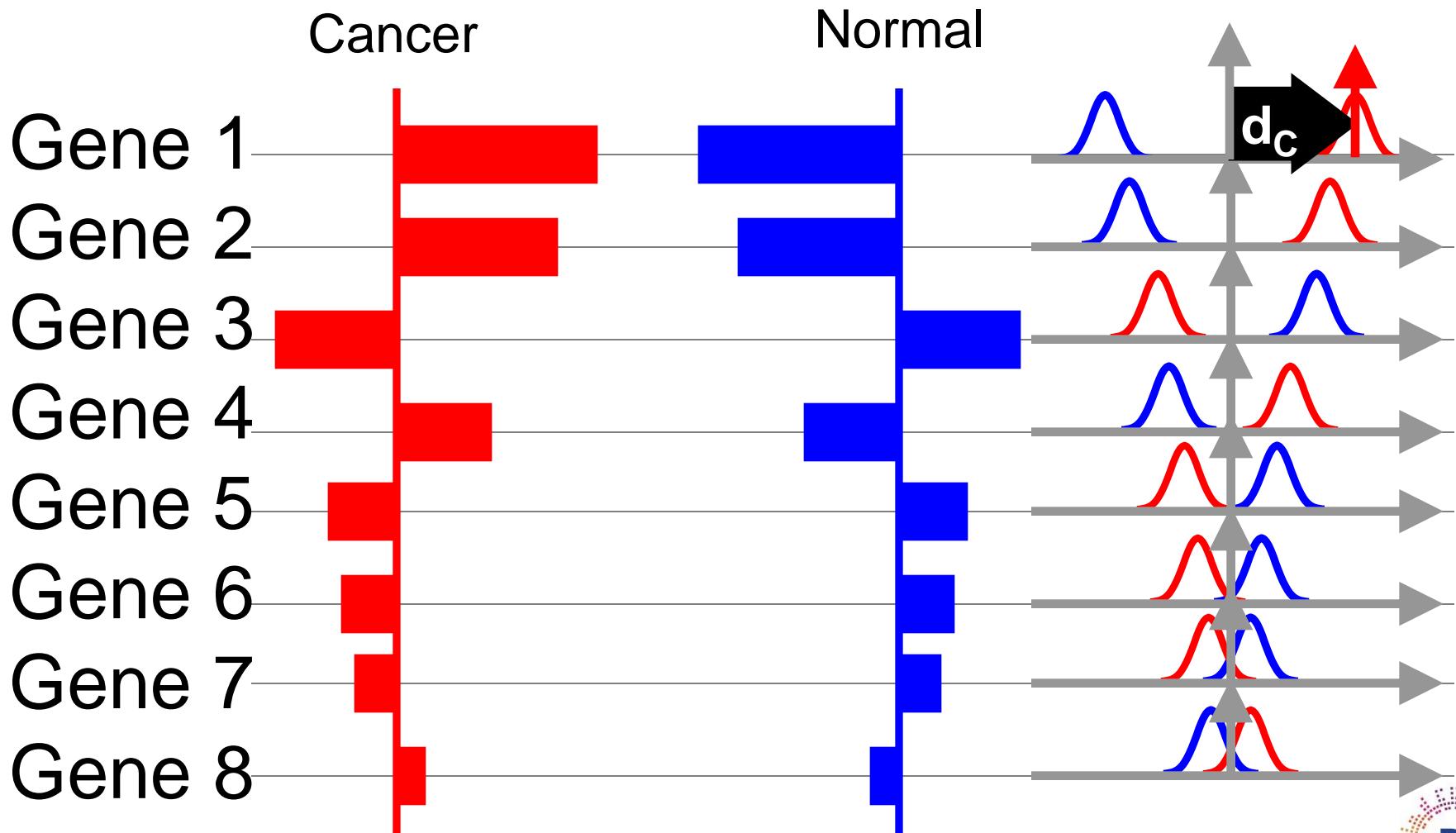


Shrunken centroids (8)

Repeat shrinking and evaluation of classifier till all genes shrunk away

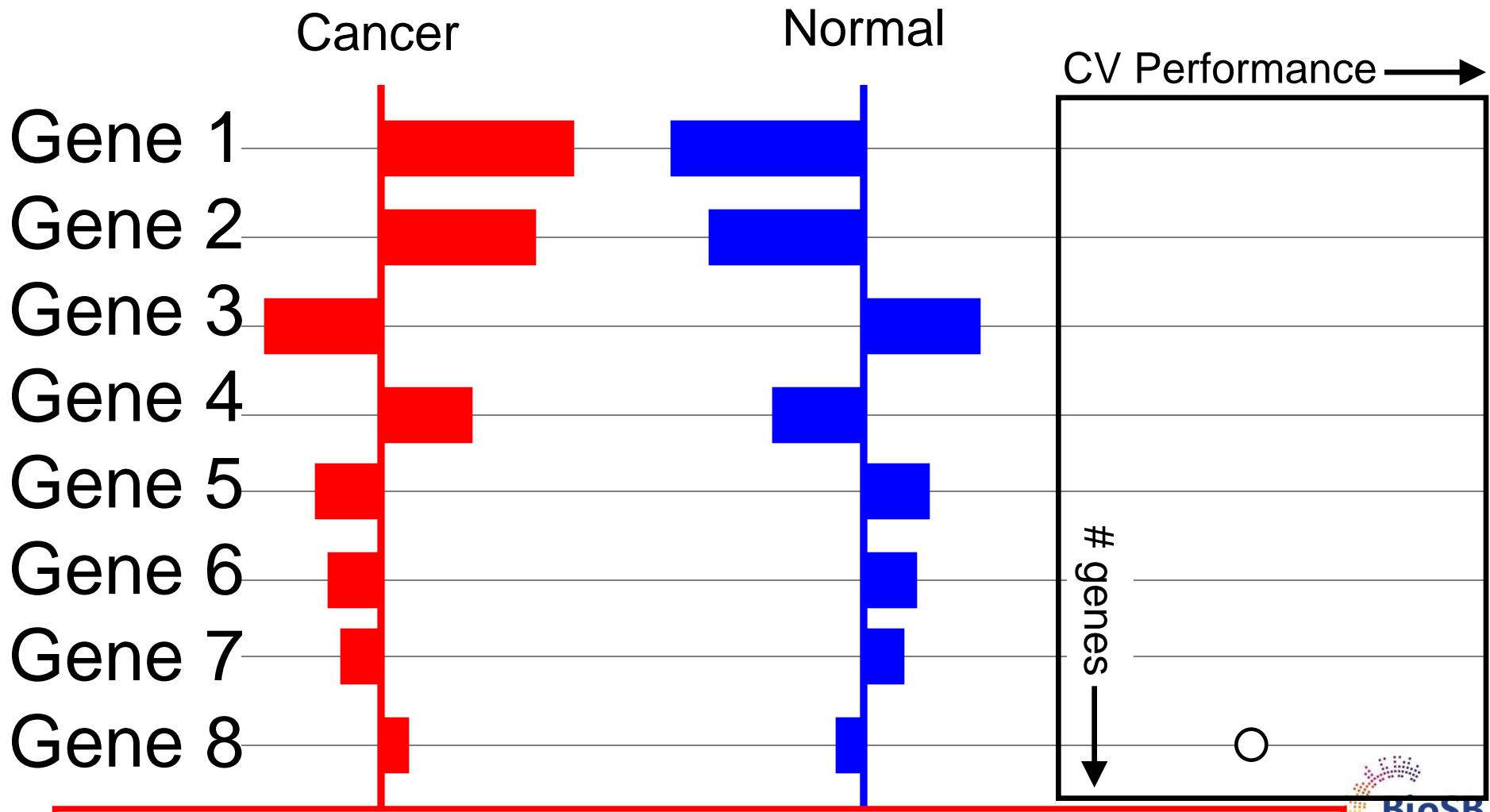


Shrunken centroids: selecting the genes



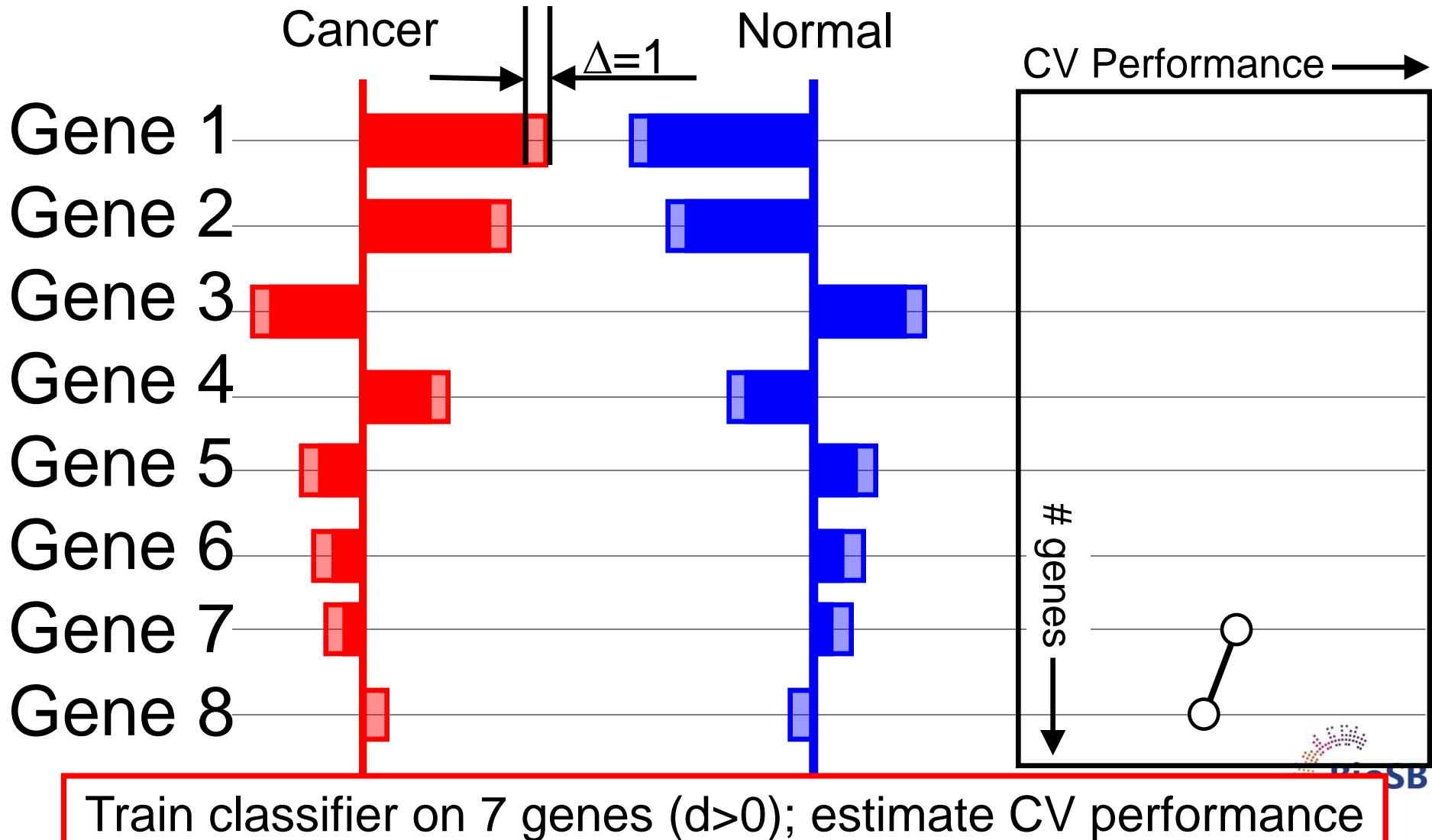
Genes sorted based on D-measure: best to worse

Shrunken centroids: selecting the genes

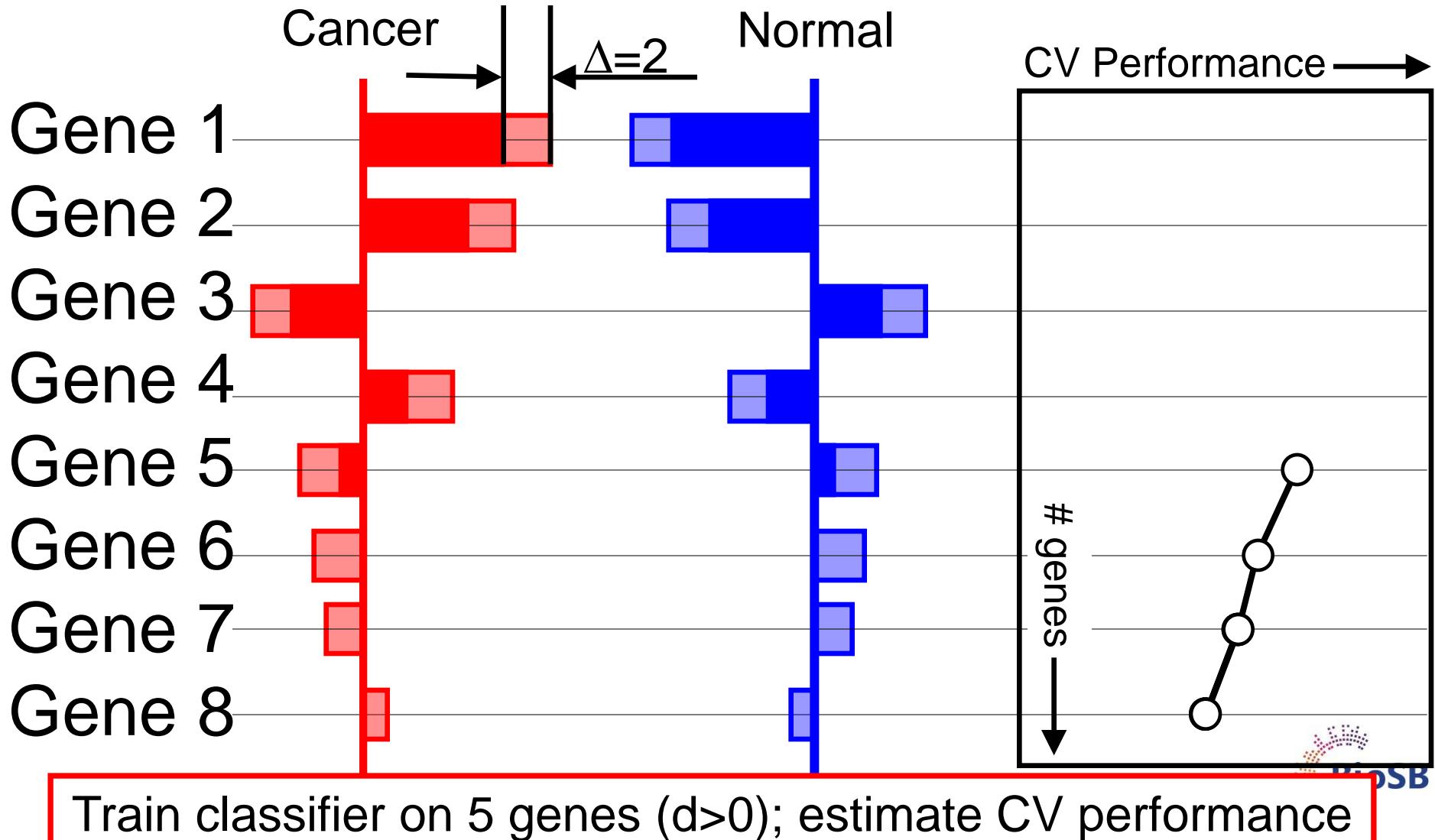


Train classifier on all 8 genes; estimate CV performance

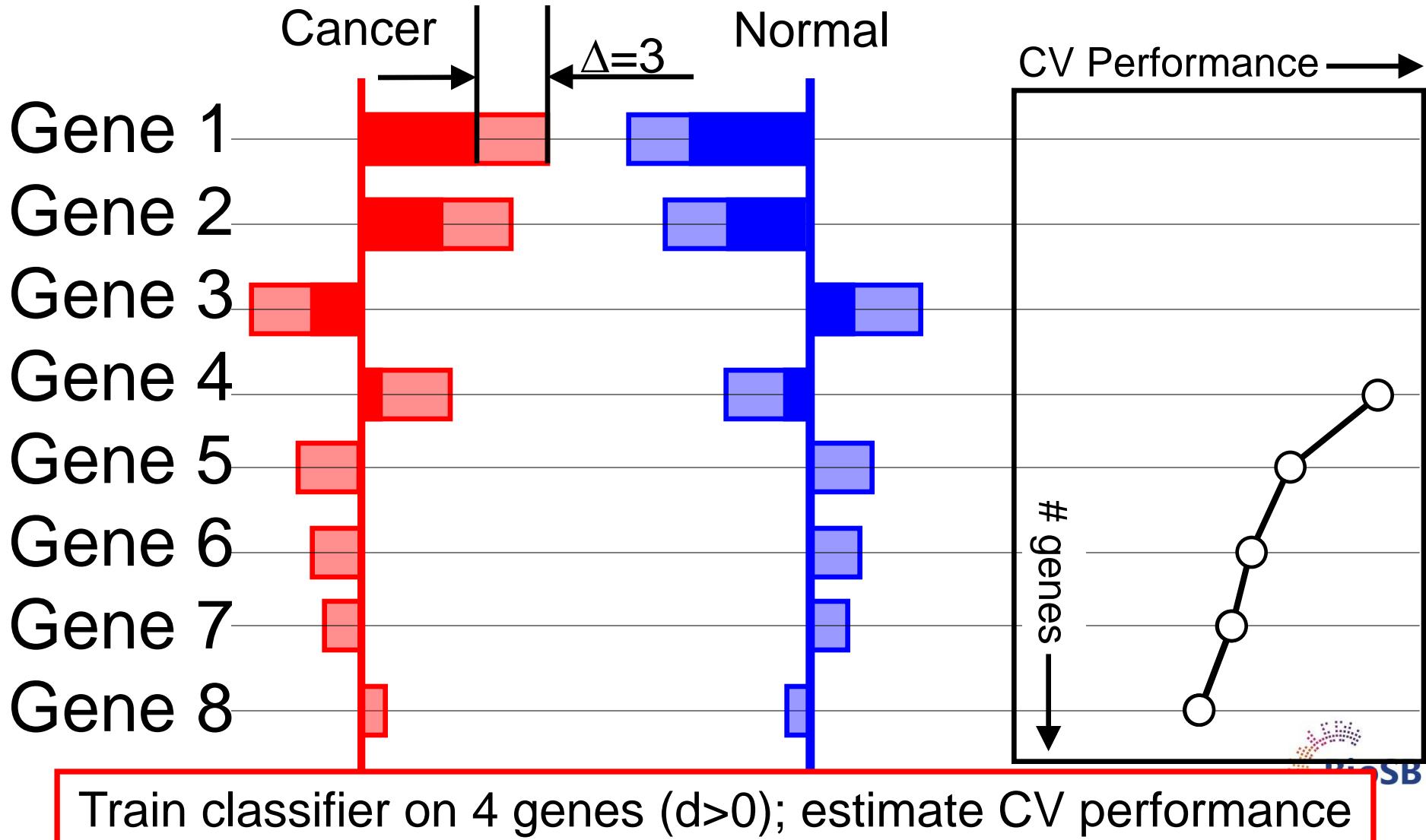
Shrink all d by $\Delta=1$: reduce length by 1



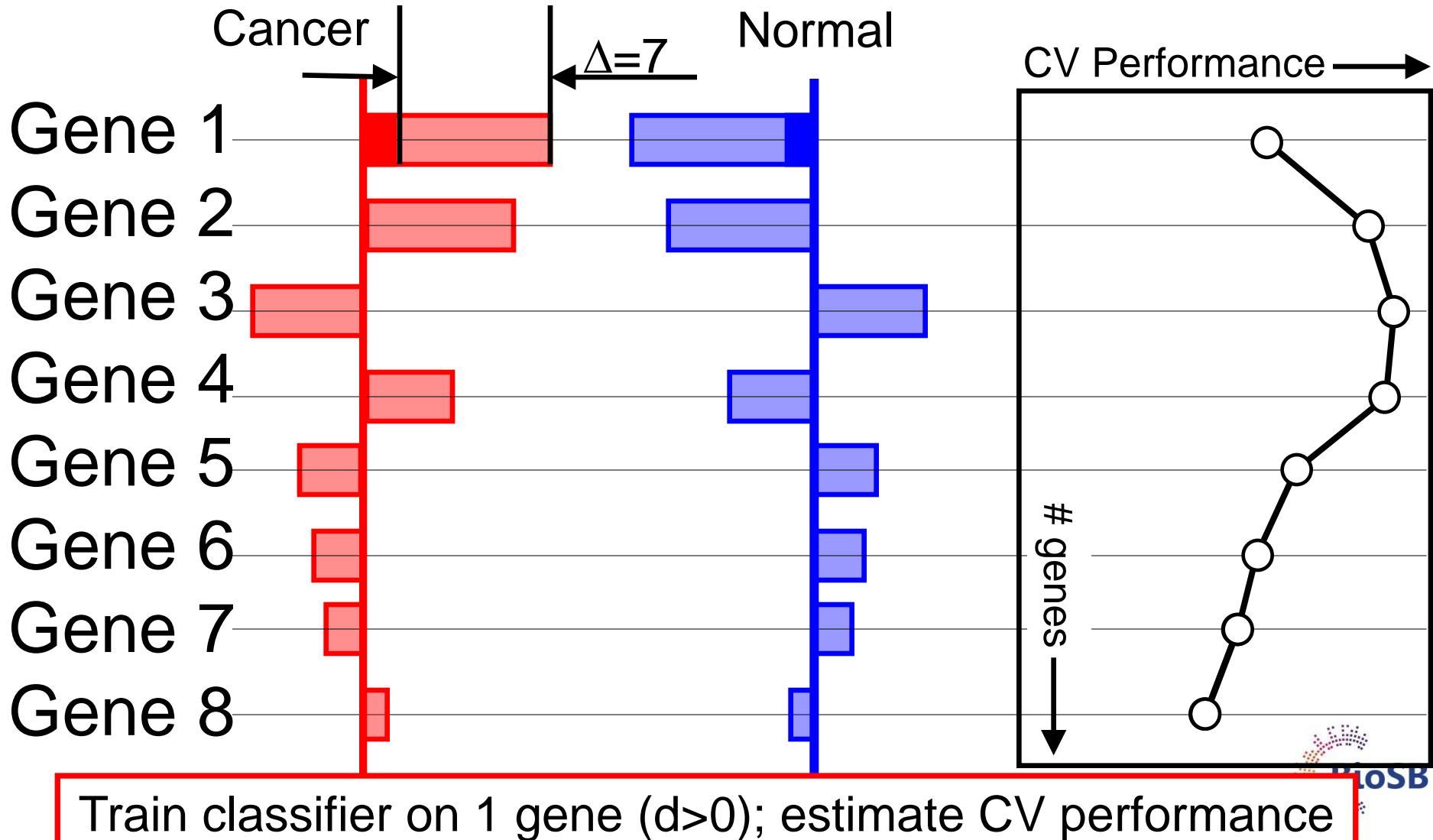
Shrink all d by $\Delta=2$: reduce length by 2



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



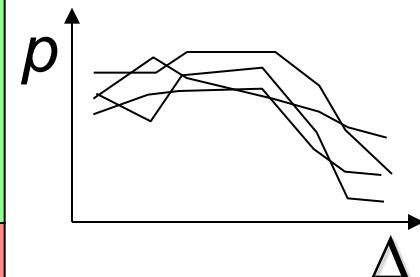
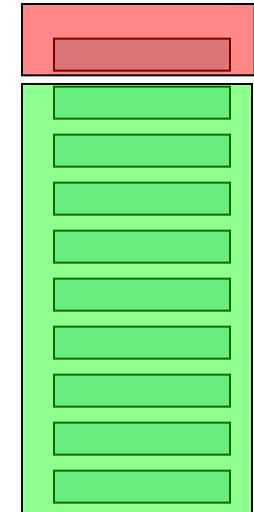
Shrink all d by $\Delta=7$: reduce length by 7



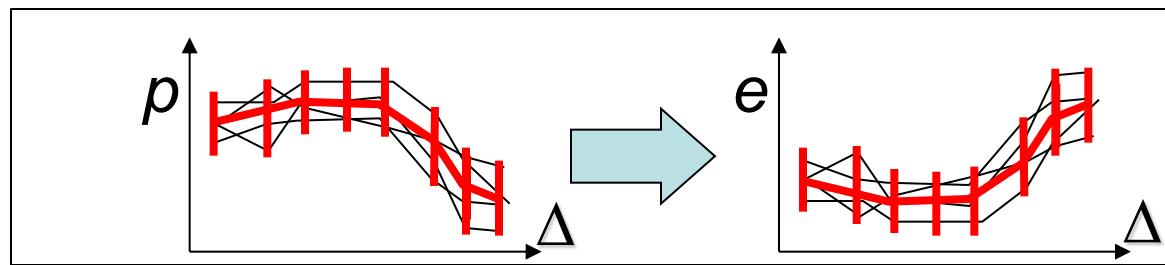
Determining the optimal Δ

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 'shrunken centroids' on training set
 - iii. Test the resulting classifier on the test set (x_i)

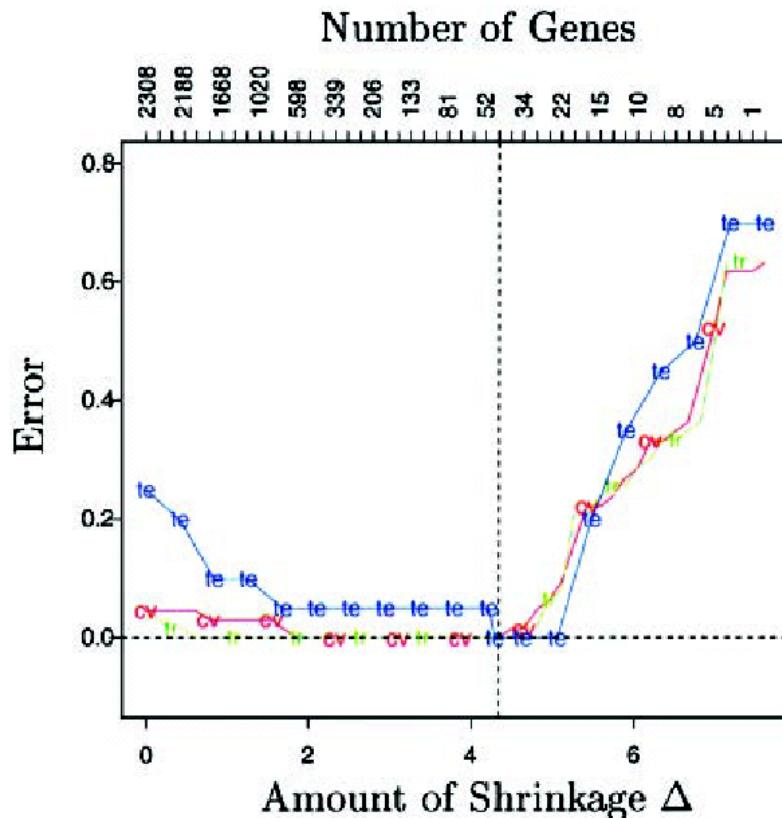


3. Result: 10 Curves of performance vs. Δ
4. Average all 10 curves and compute std. dev. at each Δ
5. Pick the Δ where the performance is maximal (error min.)



PAM

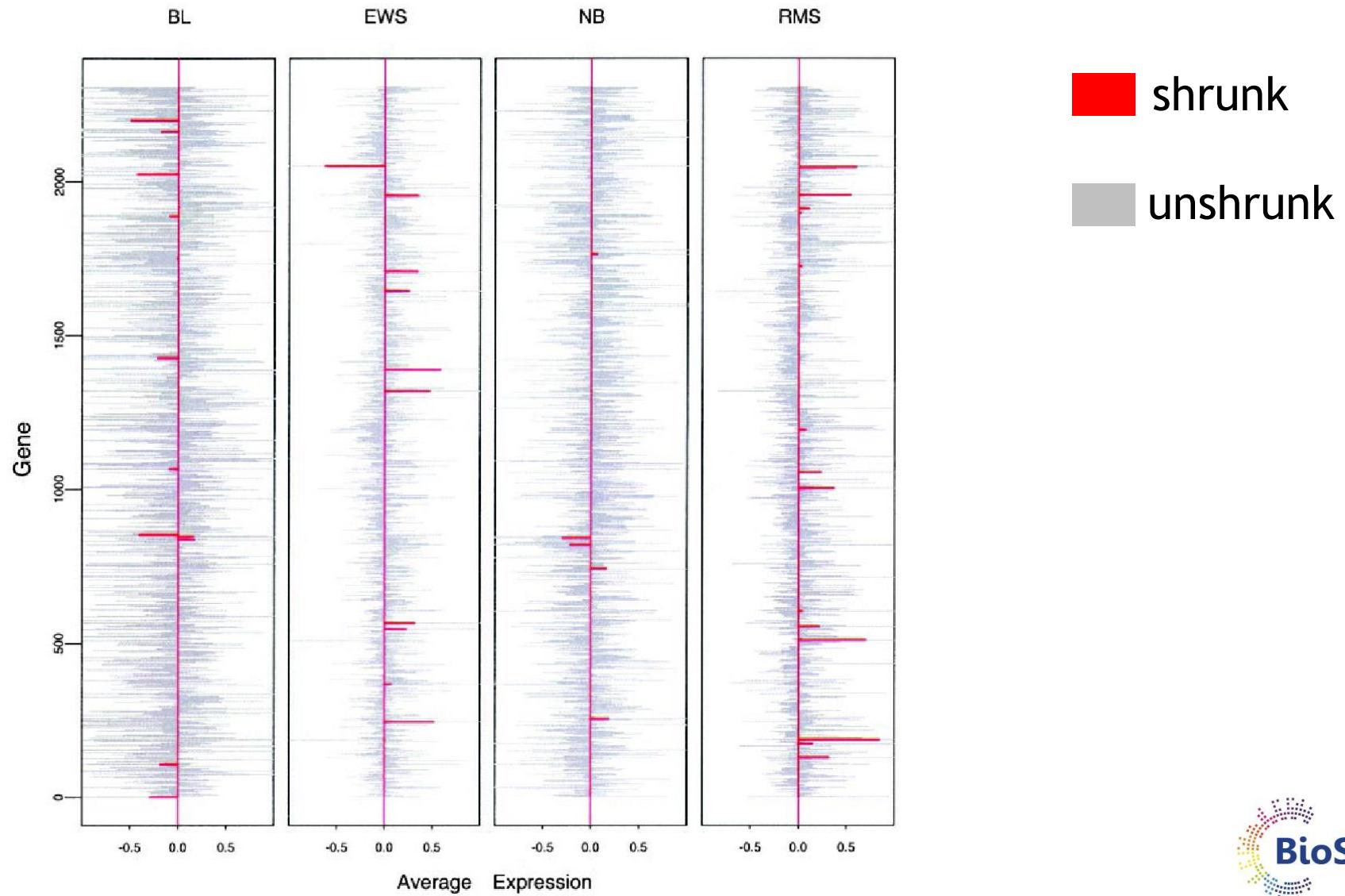
- For the Khan datat set*: 4 classes of **small round blue cell tumors** (SRBCT): BL, EWS, NB, RMS
- At optimal Δ : 43 genes *not* shrunk away



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

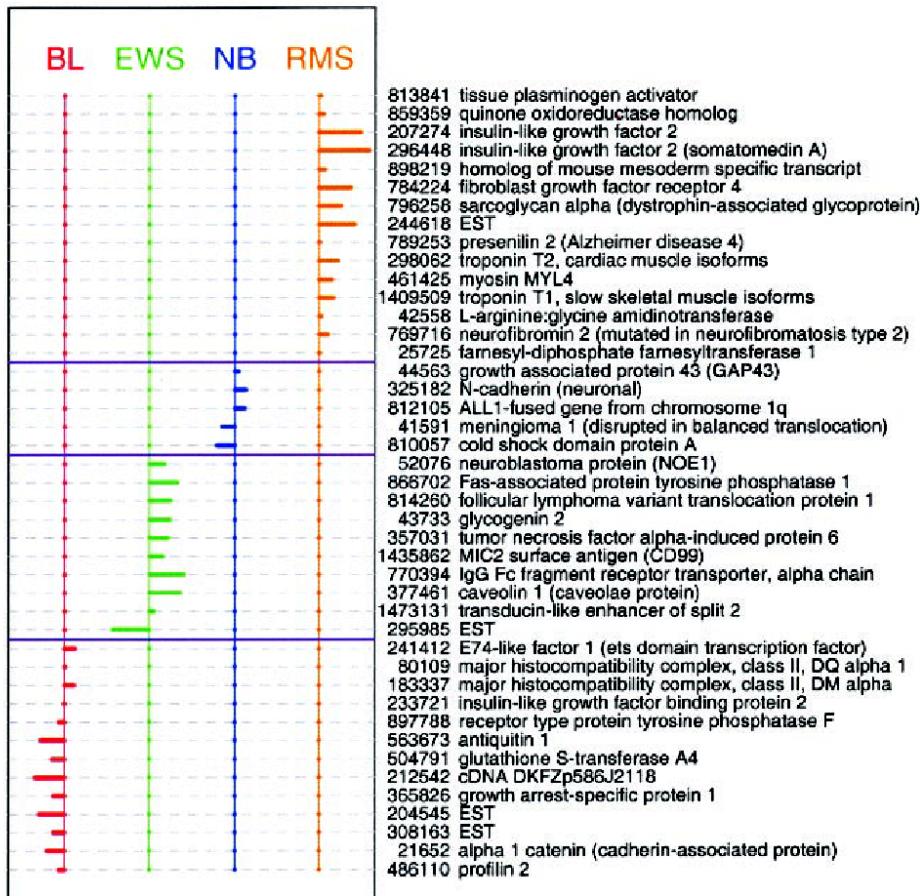
*R. Tibshirani *et al.* (2002) PNAS 99(10):6567-6572, 2002.

PAM (2)



PAM (3)

At optimal Δ : 43 genes *not* shrunk away

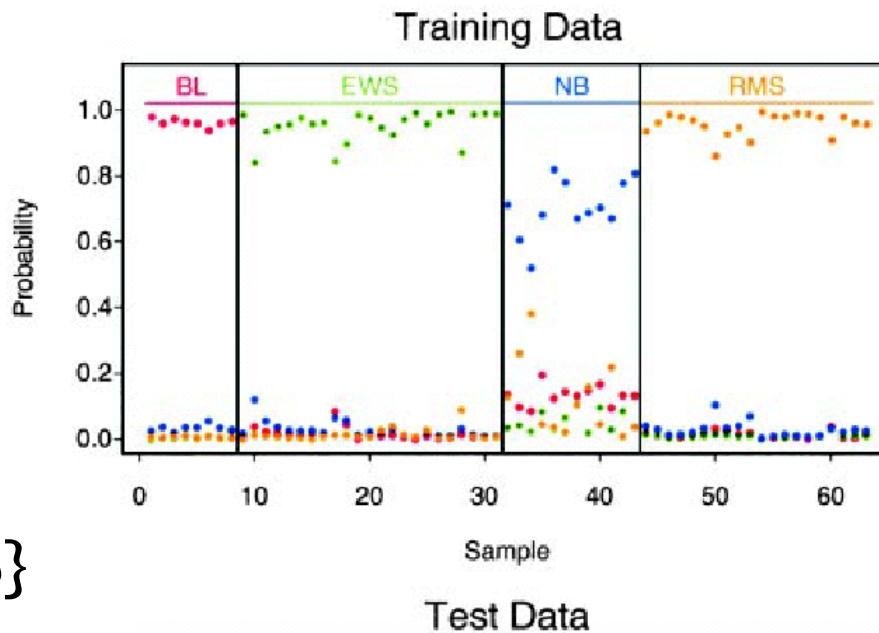


- Neuroblastoma (NB)
- Rhabdomyosarcoma (RMS)
- Burkitt lymphoma (BL)
- Ewing family of tumors (EWS),

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

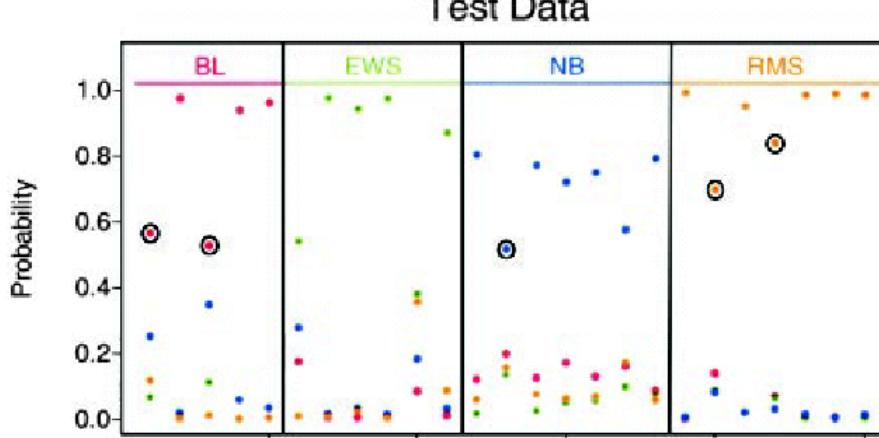
Scoring samples by posterior prob's

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



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

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



Shrinkage

- PAM: controls contribution of genes to classifier based on individual quality (d-measure) and controls degree of contribution with cross-validated classification error
- Other approach: regularisation, combine error and penalty for number of genes explicitly

Regularization

- Regularization 1: control contribution of genes to classifier based on individual quality and control degree of contribution with cross-validated classification error
- Regularization 2: combines accuracy (error) and penalty on large weights (= simple models) in one criterion.

Shrinkage (2)

- Model: $y = \beta_0 + \sum_{i=1}^p \beta_i x_i + \varepsilon$
- Penalised (aka regularised) least squares:
 - Ridge regression:

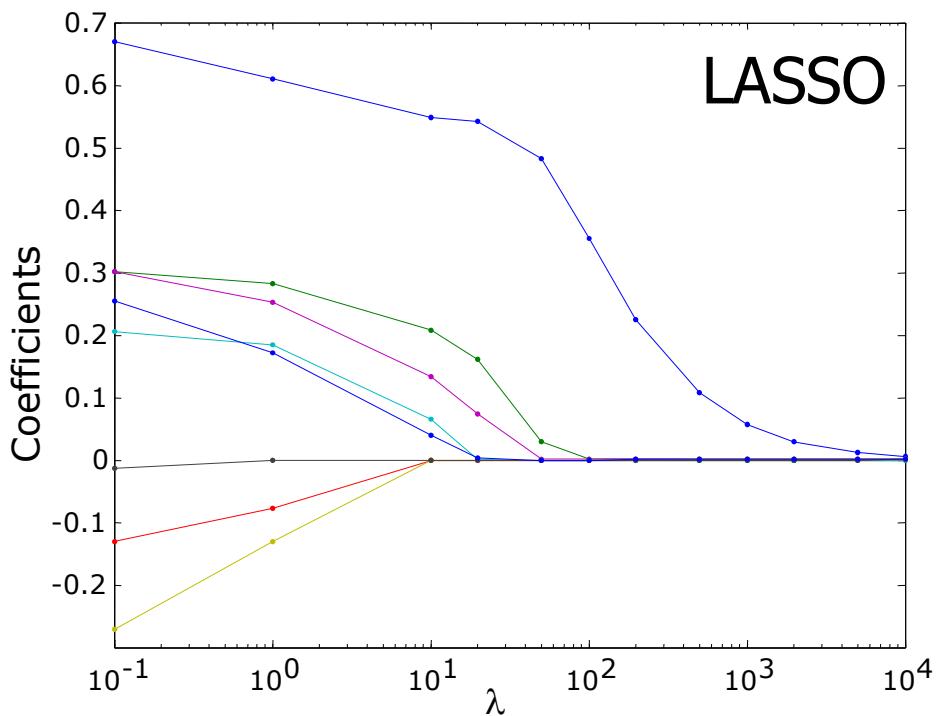
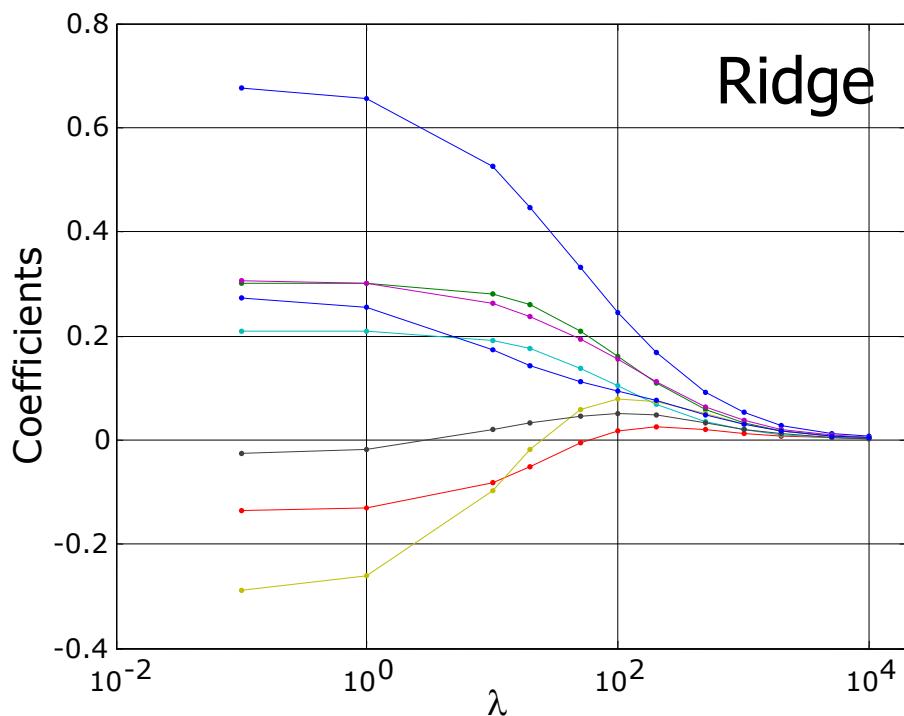
$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta}} \left[\sum_{j=1}^n \left(y_j - \beta_0 - \sum_{i=1}^p \beta_i x_{j,i} \right)^2 + \lambda \sum_{i=1}^p \beta_i^2 \right]$$

- 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!)



Final summary

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

Practical session: Feature selection

- All datasets are called a
- fsel creates a mapping w which can be applied to a data set.
- Example, select d = 5 features from a:
 - `w = fsel(a,'individual','NN',5)`
 - `b = a*w` will give you a dataset b with 5 features
 - To return a list of ranked features, call it like this
 - `[w,list] = fsel(a);`
 - Then create a dataset b with the best d features like this:
 - `b = a*w(:,1:d);`

Practical session: feature selection

- `[W,LIST] = fsel (DATA, ALGORITHM, CRITERION, P)`
- Defaults: `ALGORITHM = 'individual'`, `CRITERION = 'NN'`
- Smarter ALGORITHMs are:
 - forward selection: `'forward'`
 - backward selection: `'backward'`
 - **(Ignore rest)**
- CRITERION: (only use these)
 - `'maha-s'`: sum of estimated Mahalanobis distances.
 - `'NN'` : 1-NN leave-one-out classification performance
 - **(Ignore rest)**

Exercises

- Exercise 3.1
 - Only use the `iris` dataset (not biomed)
- Exercise 3.3
 - Script on next page
- Exercise 3.7d onwards: 2 modes of calling PCA
 - load `housing`
 - `[W,FRAC] = pca(a,1); % get a mapping W`
 - `W.data.rot(:,1);`
 - `figure(1); plot(W.data.rot(:,1));`
 - `v = pca(a,0); % get the variance retained v`
 - `figure(2);plot(v);`
 - `figure(3); plot(var(a));`