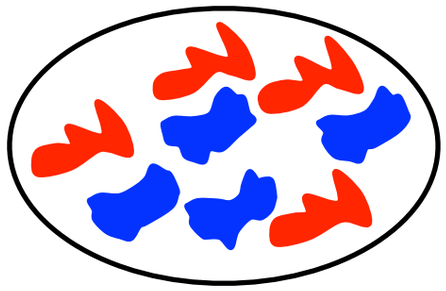
Classifier evaluation



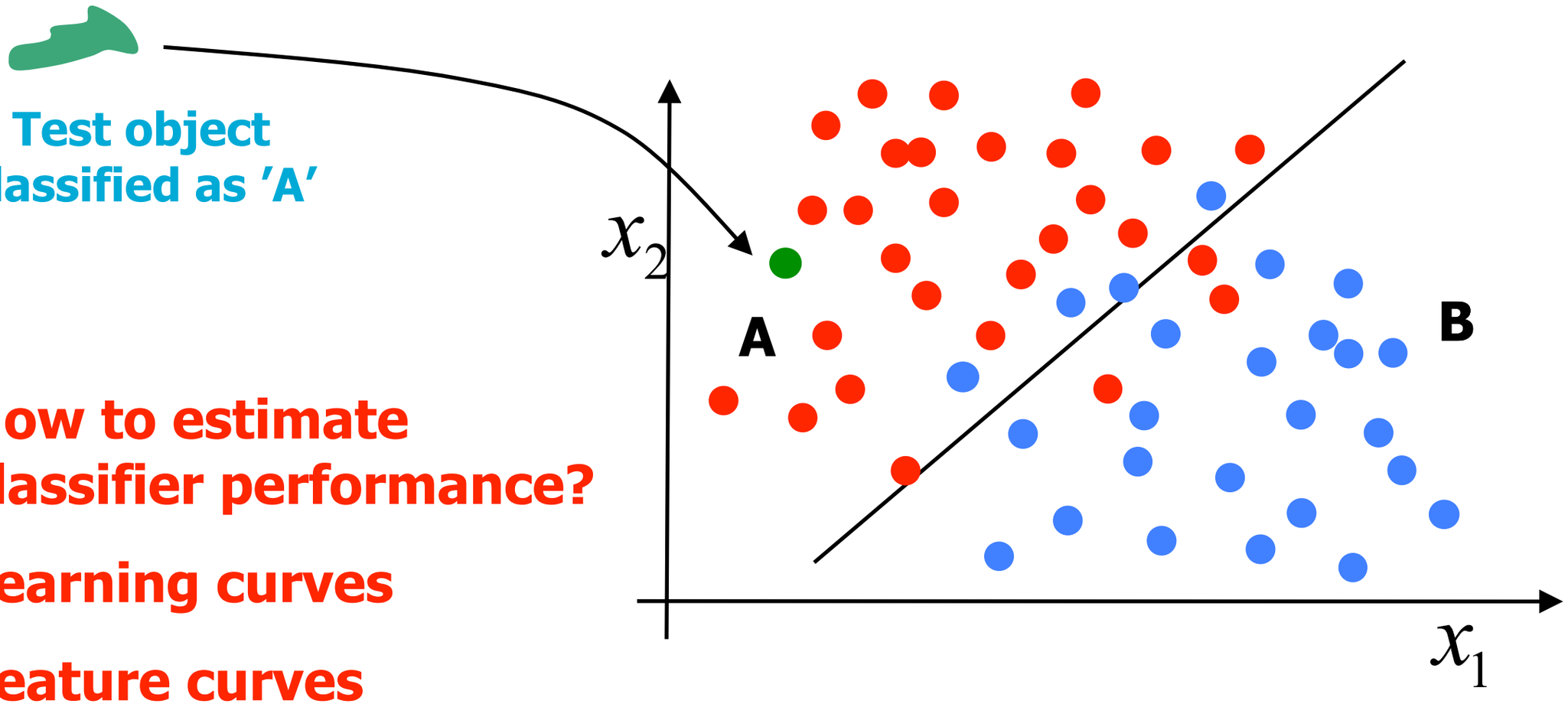
Sensor

Representation

Generalization

Feature Space

Classification



How to estimate classifier performance?

Learning curves

Feature curves

Recap classifiers

$$p(y|\mathbf{x})$$

generative classifiers:

$$\hat{p}(y|\mathbf{x}) = \frac{\hat{p}(\mathbf{x}|y)\hat{p}(y)}{\hat{p}(\mathbf{x})}$$

discriminative classifiers:

$$\hat{p}(y|\mathbf{x}) = f(\mathbf{x};\mathbf{w})$$

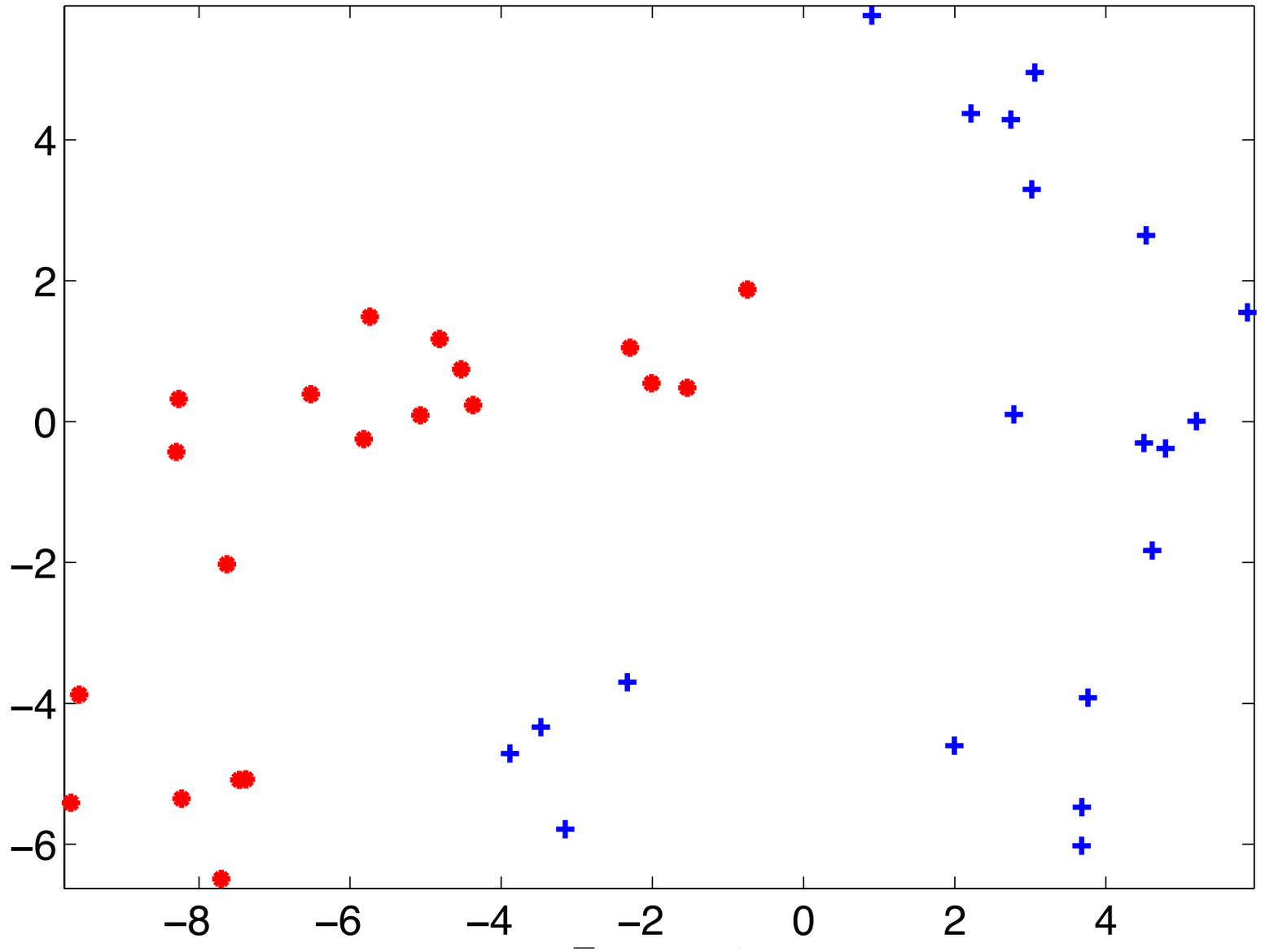
parametric densities:
- QDC
- LDC
- Nearest mean

nonparametric densities:
- k-nearest neighbour
- Parzen classifier
- Naive Bayes

- logistic classifier
- support vector
- decision trees
- perceptron
- neural networks

What classifier to use here?

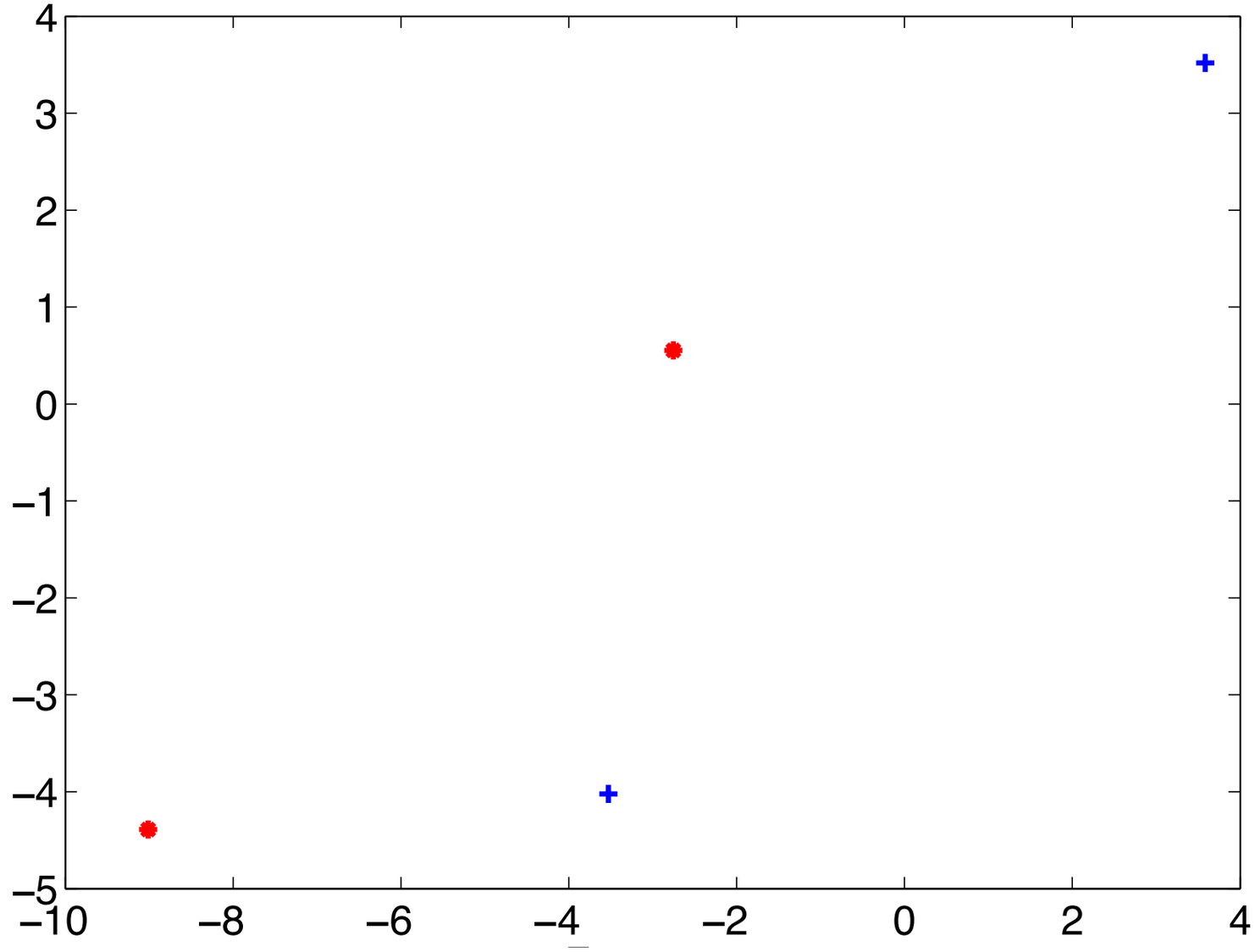
- NMC
- LDC
- QDC
- k-NN
- Parzen classifier
- Naive Bayes
- Logistic
- Support vector
- Decision tree
- Neural network
- …



TU Delft

What classifier to use here?

- NMC
- LDC
- QDC
- k-NN
- Parzen classifier
- Naive Bayes
- Logistic
- Support vector
- Decision tree
- Neural network
- …



TU Delft

What classifier to use here?

- Loaded our famous Iris dataset
- 150 objects in 4D

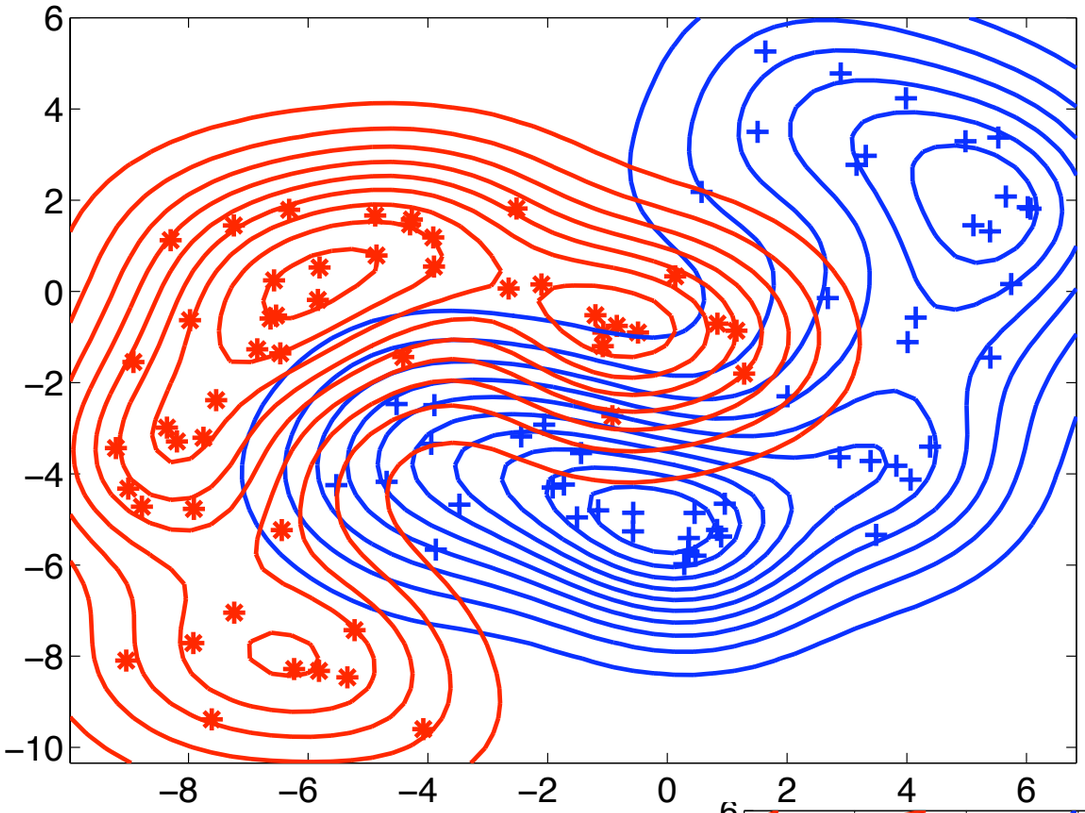
- What now?

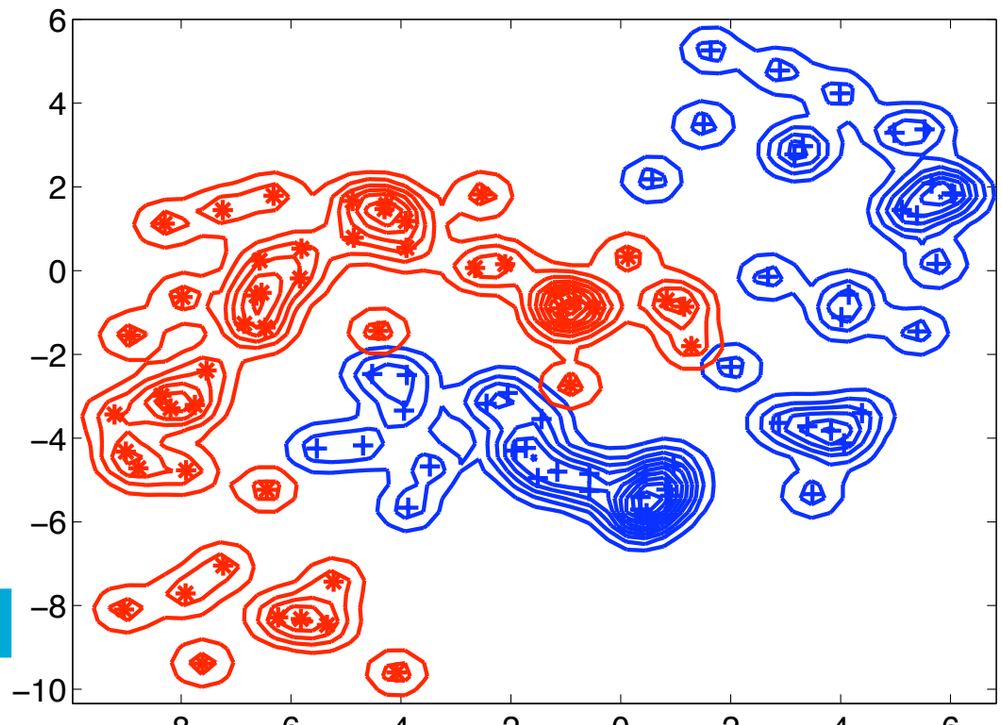
iris =

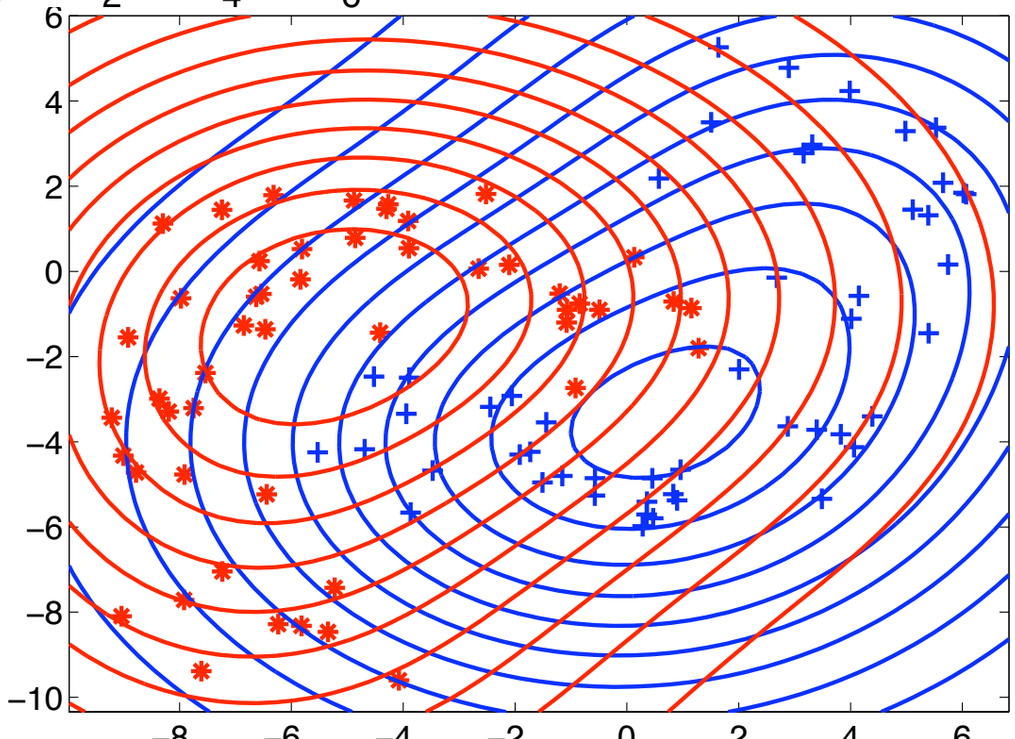
	sepal l.	sepal w.	petal l.	petal
	5.1000	3.5000	1.4000	0.2000
	4.9000	3.0000	1.4000	0.2000
	4.7000	3.2000	1.3000	0.2000
	4.6000	3.1000	1.5000	0.2000
	5.0000	3.6000	1.4000	0.2000
	5.4000	3.9000	1.7000	0.4000
	4.6000	3.4000	1.4000	0.3000
	5.0000	3.4000	1.5000	0.2000
	4.4000	2.9000	1.4000	0.2000
	4.9000	3.1000	1.5000	0.1000
	5.4000	3.7000	1.5000	0.2000
	4.8000	3.4000	1.6000	0.2000
	4.8000	3.0000	1.4000	0.1000
	4.3000	3.0000	1.1000	0.1000
	5.8000	4.0000	1.2000	0.2000
	5.7000	4.4000	1.5000	0.4000
	5.4000	3.9000	1.3000	0.4000

Optimal width for Parzen classifier

- What is the optimal h?







What classifier to use?

- If cannot visualise high-dimensional data:
  how to choose the classifier?
  or:
  how to choose hyperparameters of a classifier?

- Need some objective criterion!
- Typically: classification performance/error

True or false?

- The classification error of the training set is a good measure of the true classification error

TRUE? FALSE?

- A good estimate of the actual, true, error is all we are after

TRUE? FALSE?

What classifier to use?

- If cannot visualise high-dimensional data:
  how to choose the classifier?
  or:
  how to choose hyperparameters of a classifier?

- Need some objective criterion!
- Typically: classification performance/error

- Test it on **independent** data
- For simplicity we assume now that classification error is good enough

Classifier evaluation

- This lecture:
  - Train and test set
  - Bootstrapping, cross-validation, leave-one-out
  - Learning curve
  - Classifier complexity
  - Bias-variance tradeoff
  - Confusion matrices
  - ROC curve

  - Reject curve

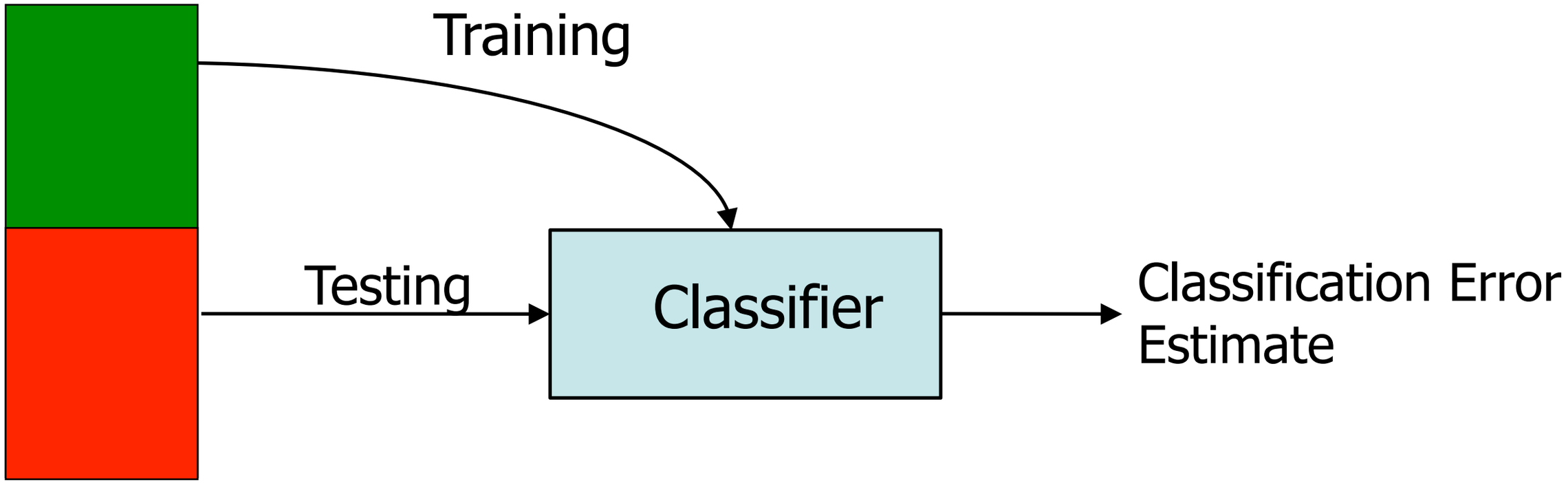
Material/literature

- Chapter 19 "Design and Analysis of Machine Learning experiments" in 'Introduction to Machine Learning' by E. Alpaydin
- Ebook available at library.tudelft.nl

- Some sections are not discussed (19.8 and further)

- 'Reject option' is (unfortunately) not in the book: get it from the slides.

Error Estimation by Test Set



Design Set

Other training set → other classifier
Other test set → other error estimate $\hat{\varepsilon}$

Some Error Estimate Variability

- Error is a sum of Bernoulli random variables:

$$\hat{\varepsilon} = \frac{1}{N}\sum_{i=1}^{N} Z_i$$

where

$$Z_i = \begin{cases} 0 & \text{if } \mathbf{x}_i \text{ is correctly classified} \\ 1 & \text{if } \mathbf{x}_i \text{ is incorrectly classified} \end{cases}$$

- You can show that the variance is:

$$\sigma^2_{\hat{\varepsilon}} = \mathrm{Var}(\hat{\varepsilon} \mid \text{test set size } N) = \frac{\varepsilon(1-\varepsilon)}{N}$$

Some Error Estimate Variability

- Error is a sum of Bernoulli random variables:

$$\hat{\varepsilon} = \frac{1}{N}\sum_{i=1}^{N} Z_i$$

- When: $\sigma^2_{\hat{\varepsilon}} = \mathrm{Var}(\hat{\varepsilon} \mid \text{test set size } N) = \dfrac{\varepsilon(1-\varepsilon)}{N}$

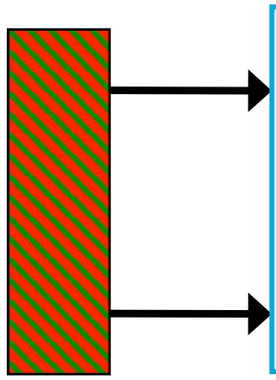
you can compute the standard deviation for different sample sizes and errors:

$$\sigma_{\hat{\varepsilon}} = \sqrt{\frac{\varepsilon(1-\varepsilon)}{N}}$$

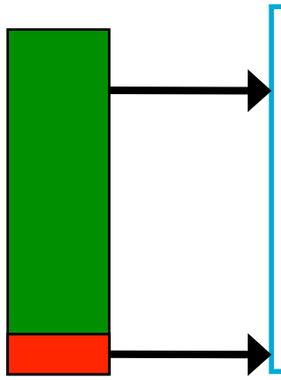
N \ ε	0.01	0.03	0.1
10	0.031	0.054	0.095
100	0.010	0.017	0,030
1000	0.003	0.005	0.0095

Training Set Size vs. Test Set Size

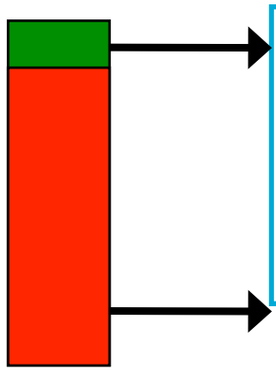
- Large training set -> good classifiers
- Large test set -> reliable, unbiased error estimate
- In practice often just a single design set is given



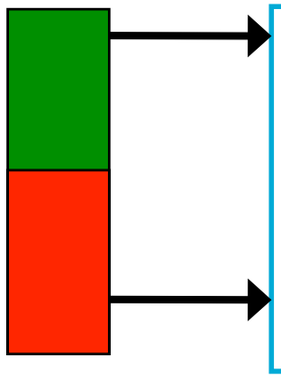
Same set for training and testing may give a good classifier, but will definitely yield an optimistically biased error estimate



A small, independent test set yields an unbiased, but unreliable [large variance] error estimate for a well trained classifier



A large, independent test set yields an unbiased and reliable [small variance] error estimate for a badly trained classifier

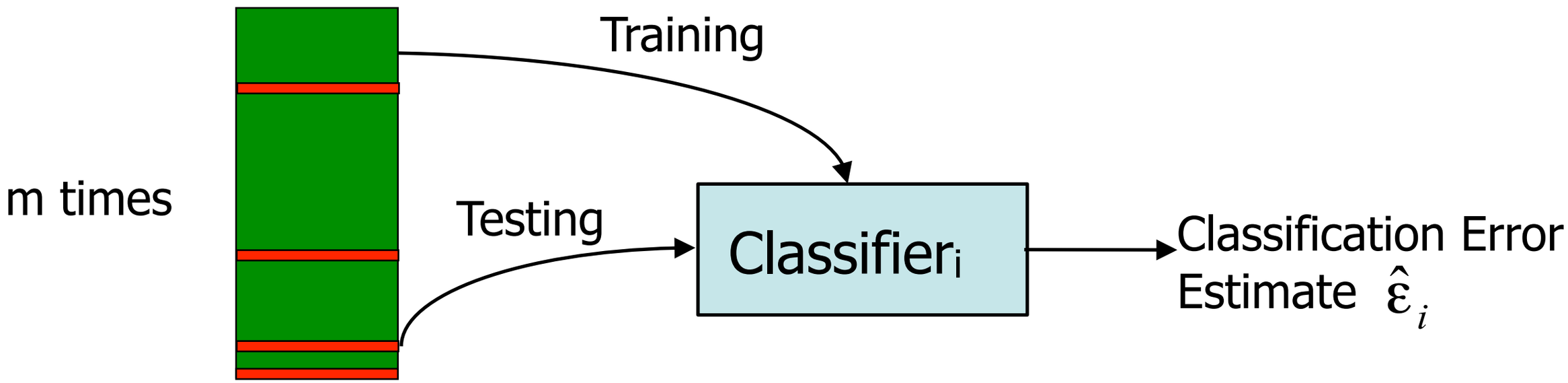


50-50 is an often used standard solution for the sizes for training and testing

But is it really good?

Bootstrapping

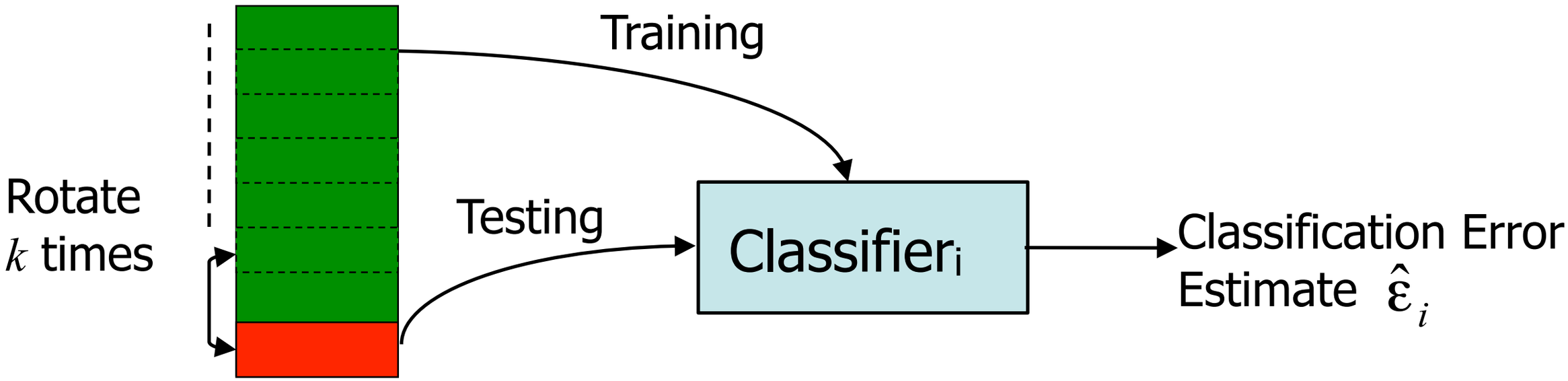
- Randomly draw N objects from N objects (**with** replacement)



$$\hat{\varepsilon} = \frac{1}{m} \sum_i \hat{\varepsilon}_i$$

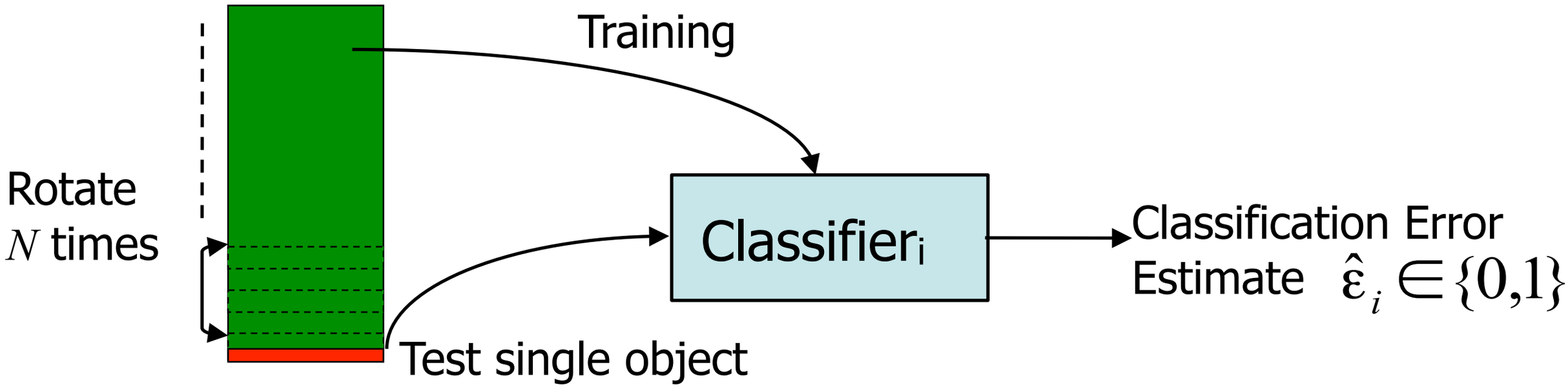
- Left-over objects are used for testing
- Repeat m times

k-fold cross validation



$$\hat{\varepsilon} = \frac{1}{k} \sum_i \hat{\varepsilon}_i$$

Leave-one-out Procedure



$$\hat{\varepsilon} = \frac{1}{N} \sum_i \hat{\varepsilon}_i$$

Optimising 'hyperparameters'

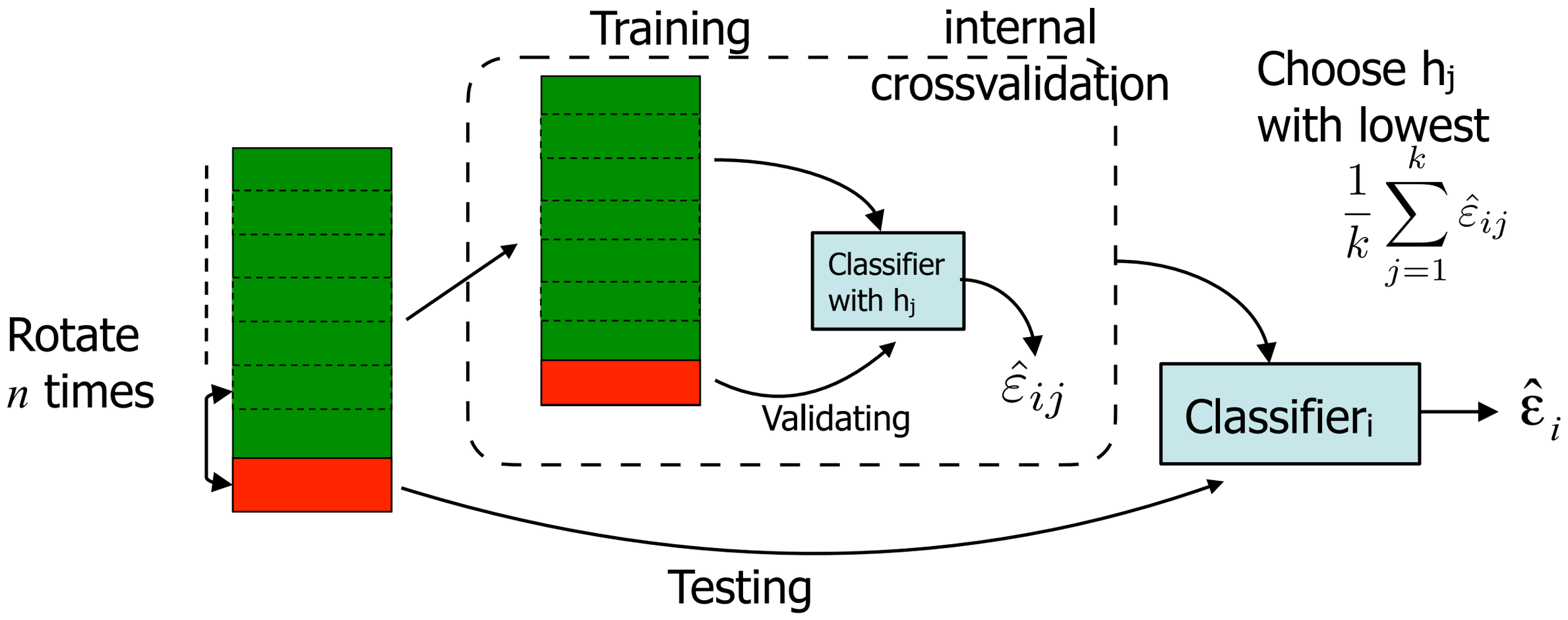
- Machine learning methods often have 'hyperparameters'

- Parzen density estimator: width parameter h
- k-nearest neighbour: number of neighbours k
- Decision trees: pruning method, stopping criterion
- Neural networks: architecture, learning rate, …

- DON'T optimise these numbers by looking at the test set!
  Then you're **CHEATING**!

Double cross-validation

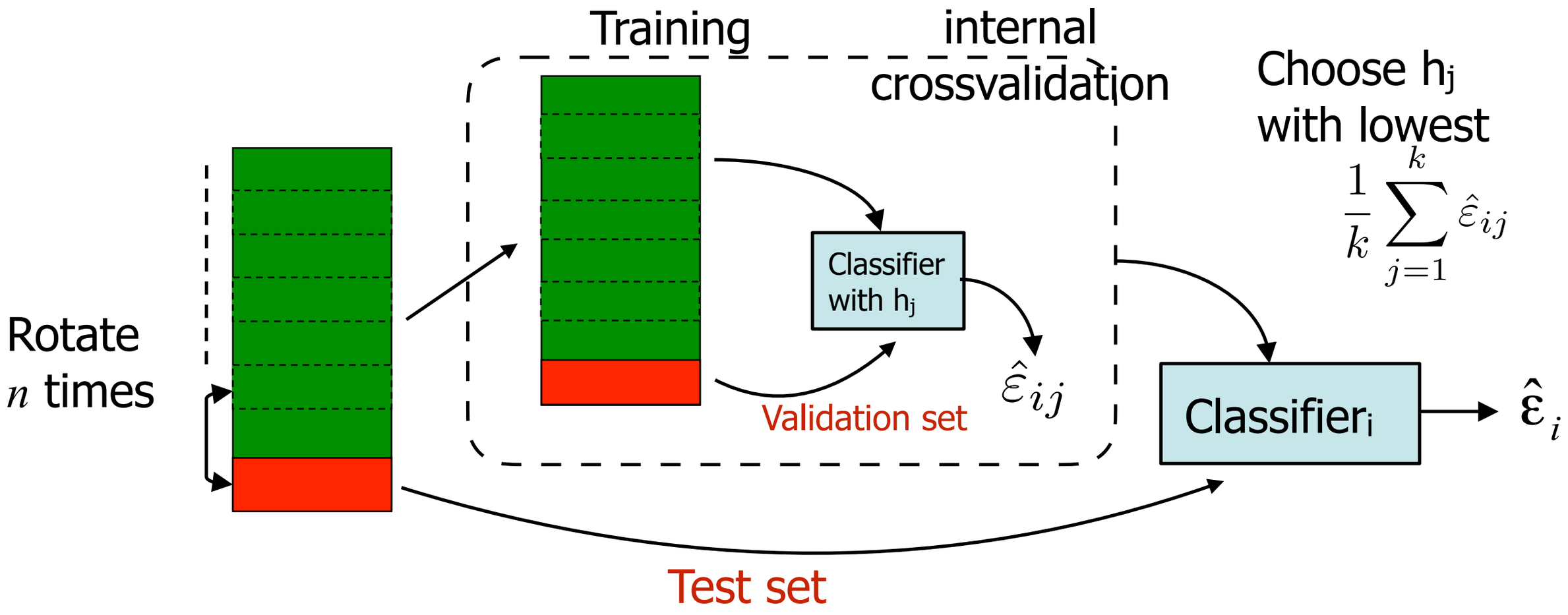
- To optimise hyperparameters, do cross-validation **inside** another cross-validation:



TU Delft

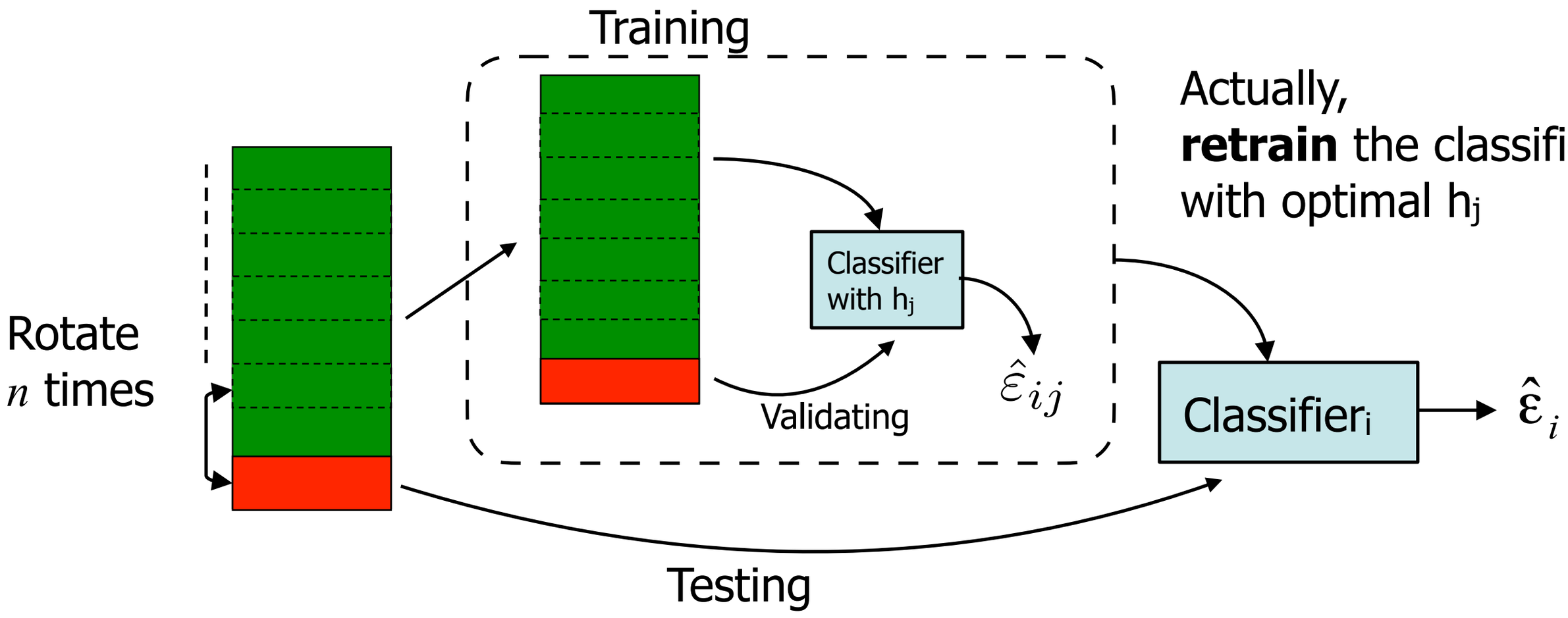
Double cross-validation

- To optimise hyperparameters, do cross-validation **inside** another cross-validation:



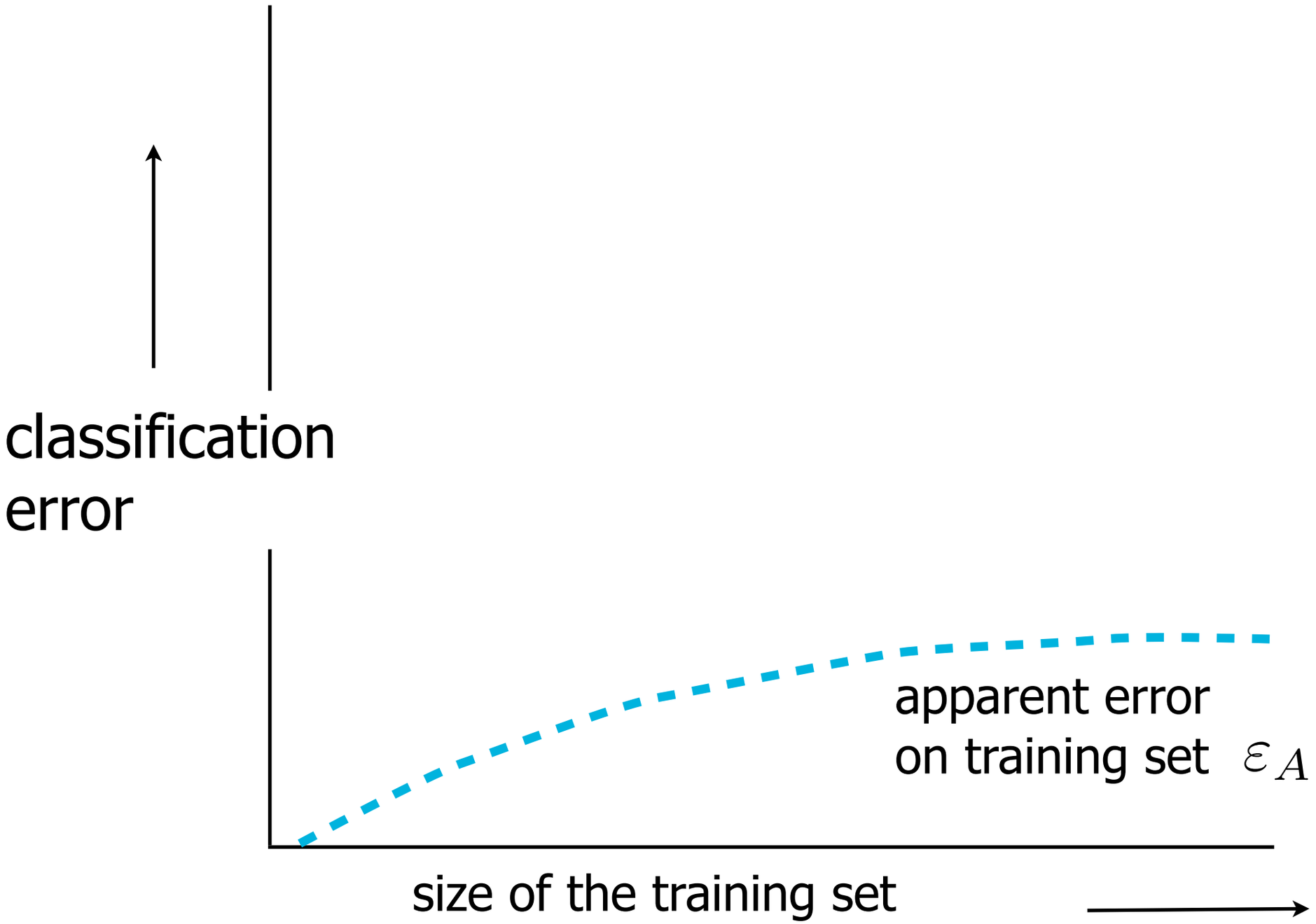
Double cross-validation

- To optimise hyperparameters, do cross-validation **inside** another cross-validation:

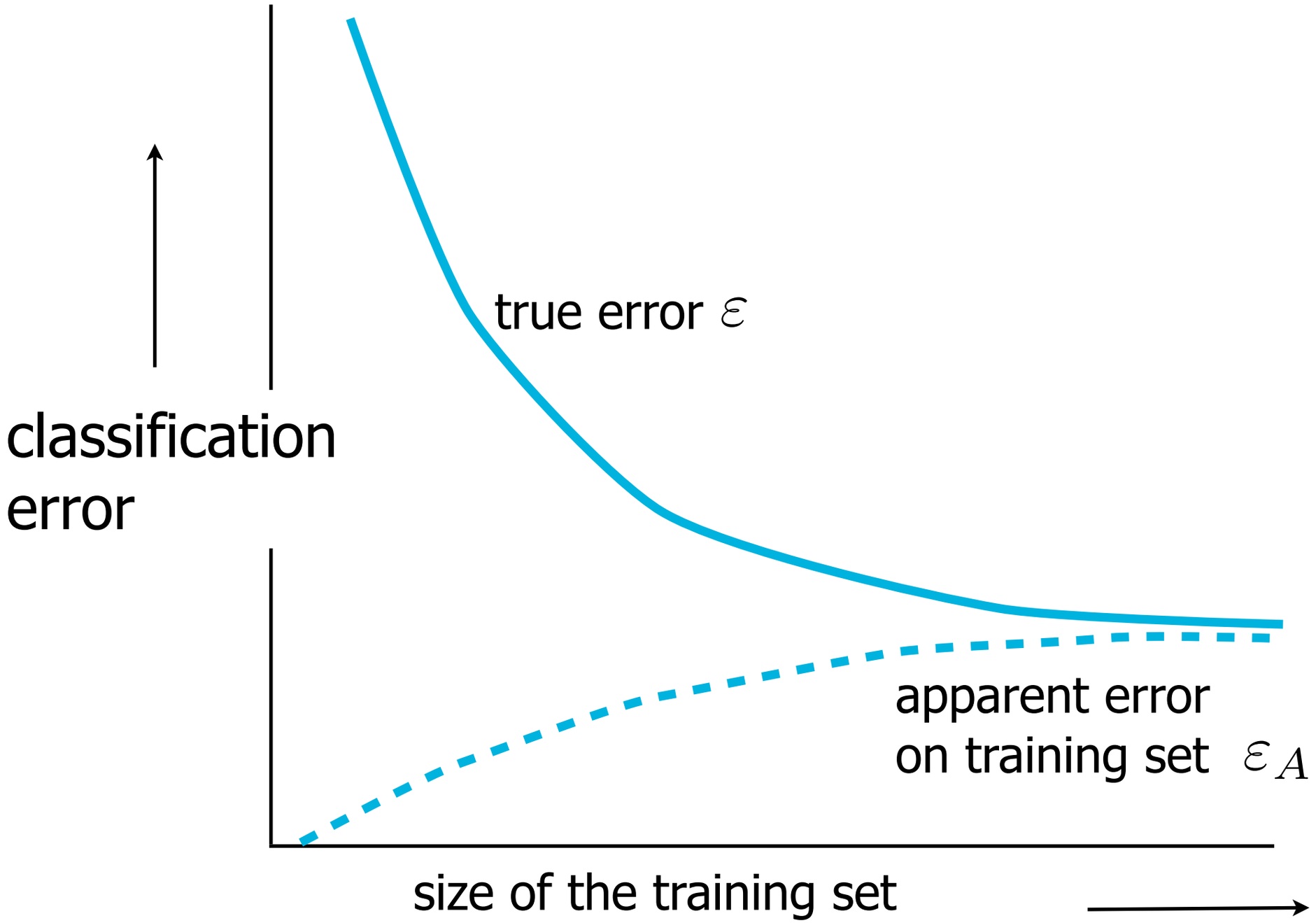


Actually,
retrain the classifier
with optimal h_j

Apparent Classification error

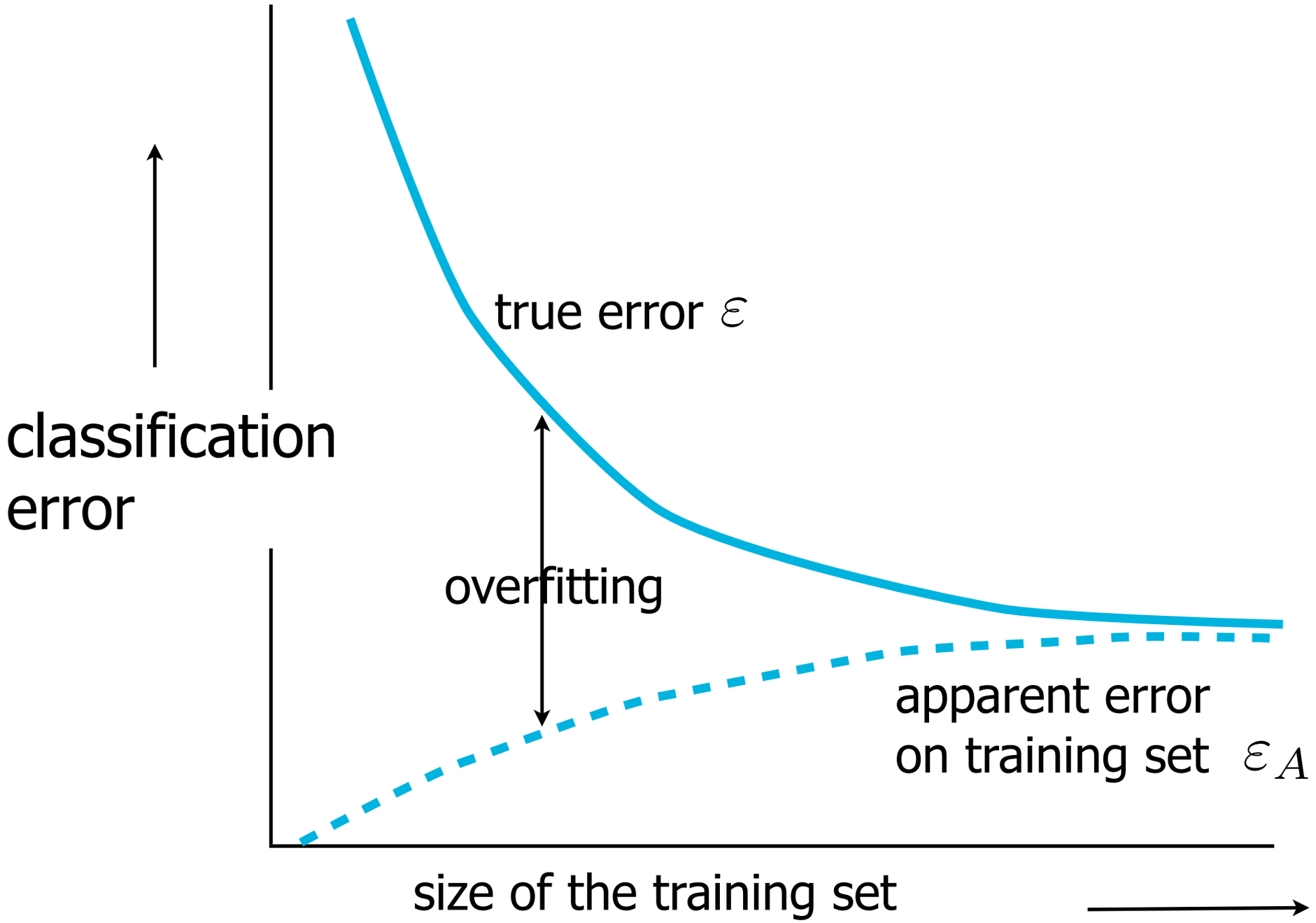


Learning curve

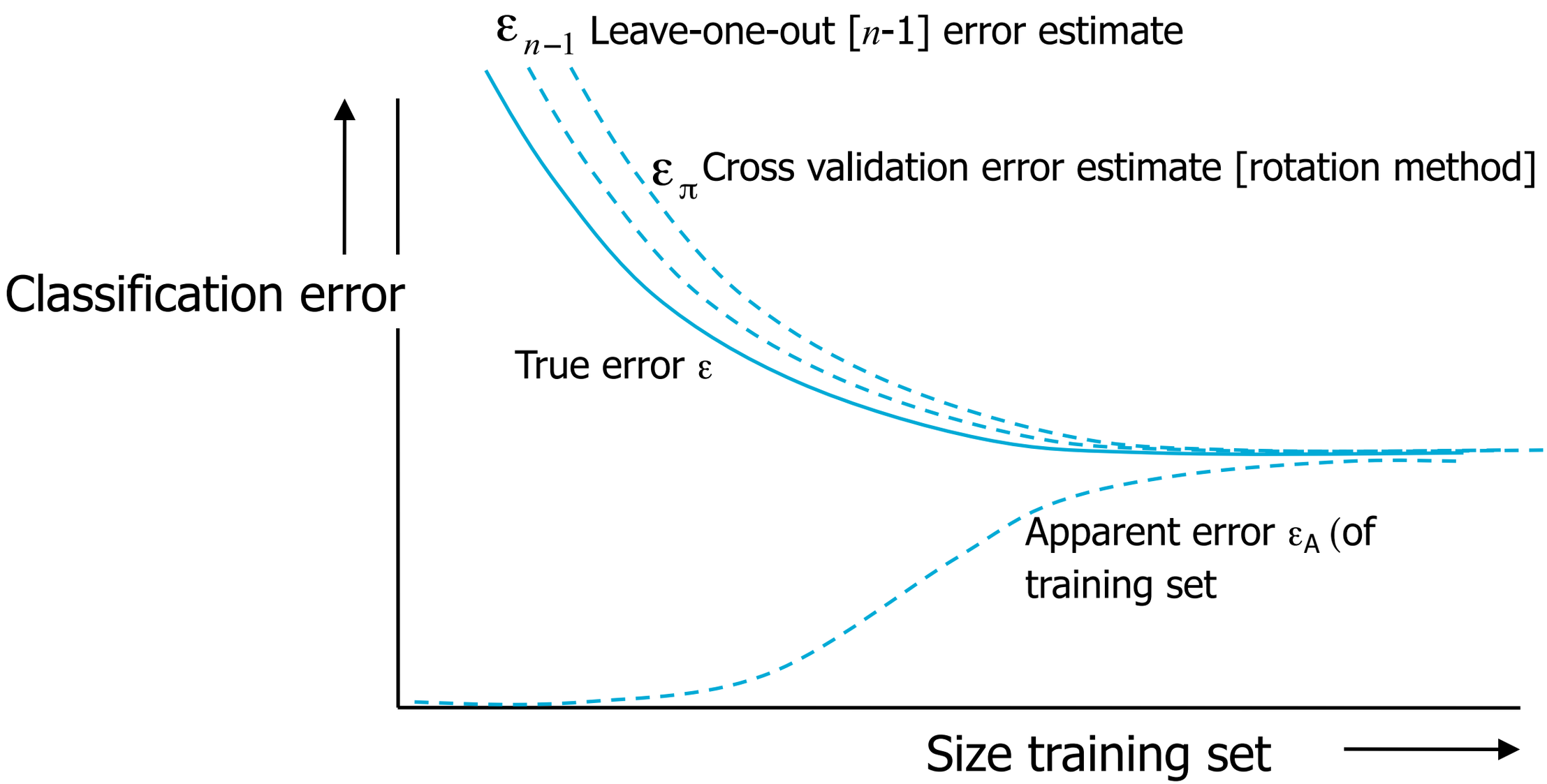


TU Delft

Learning curve



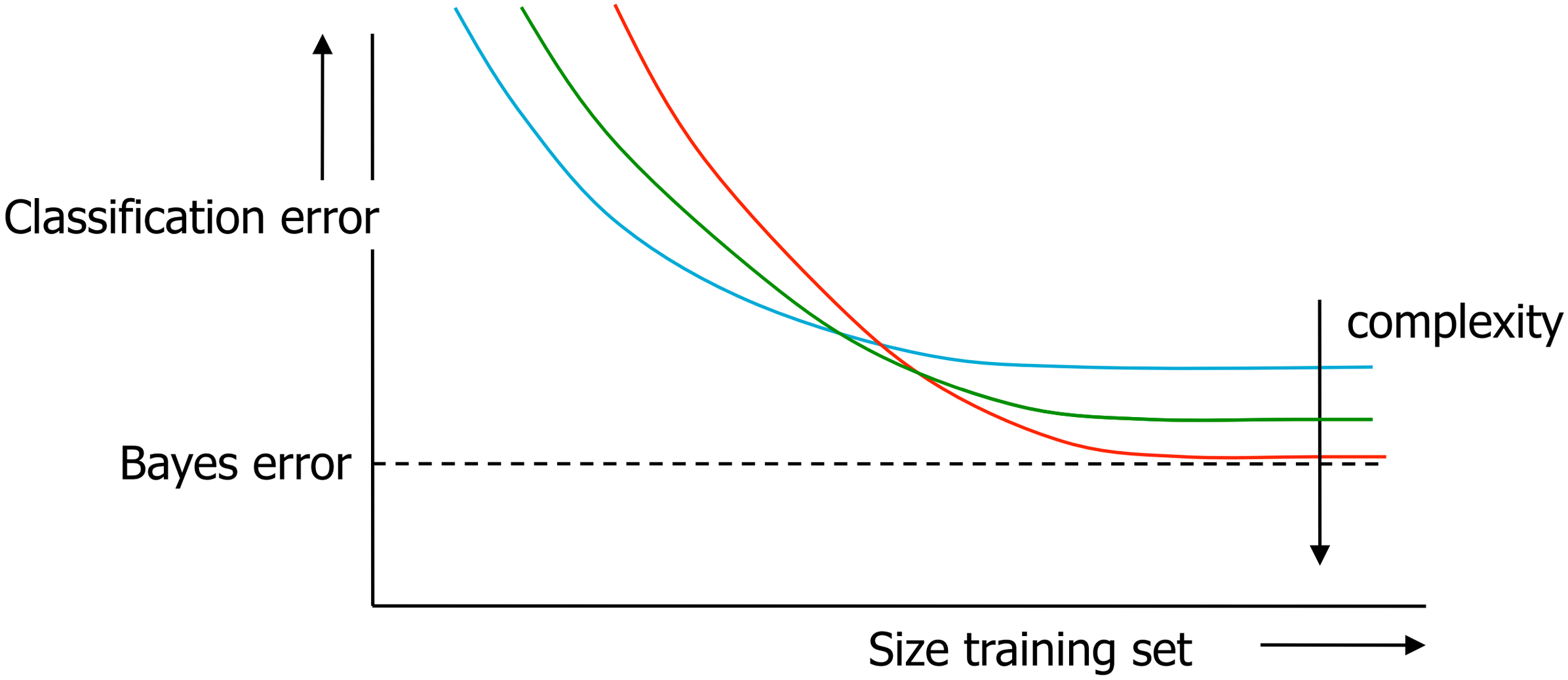
Cross Validation Curves [and Related]



Learning Curves

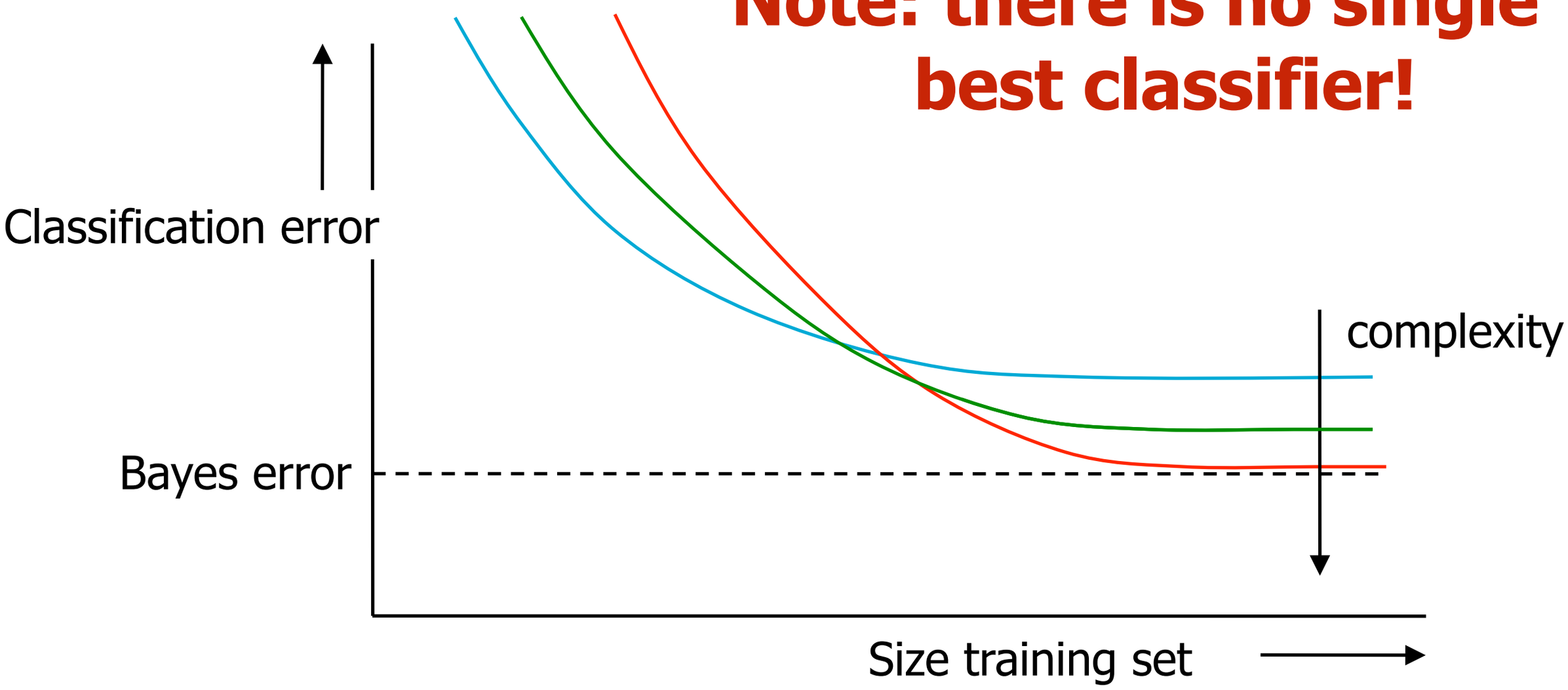
- Curves that plot [estimated] classification errors against the number of samples in training set
  - Usually plot error both on training and on test set
- Gives insight in, e.g.
  - Amount of overtraining
  - Usefulness of additional data
  - How different classifiers compare
  - Stability of training
  - …

Different Classifier Complexity

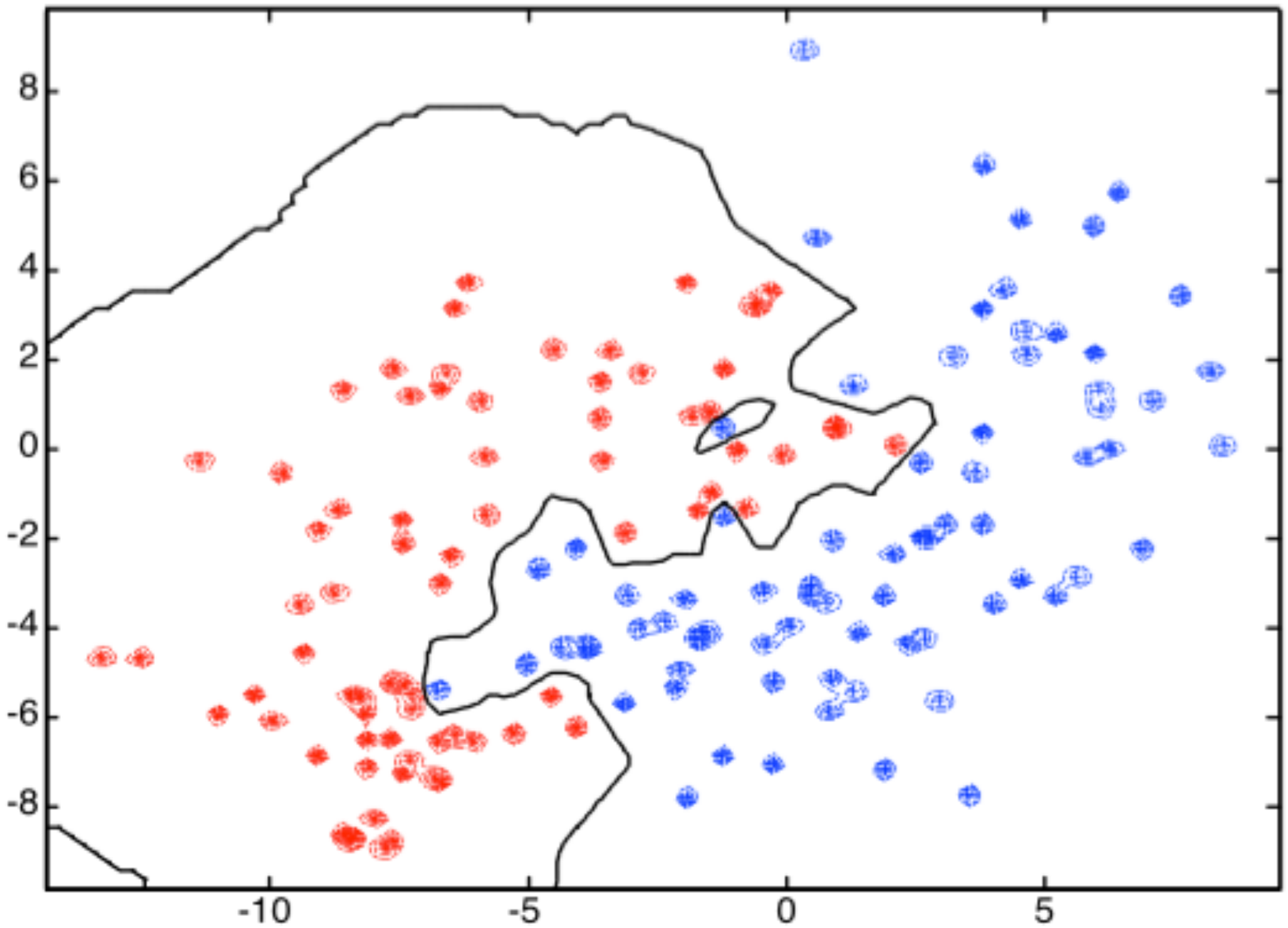


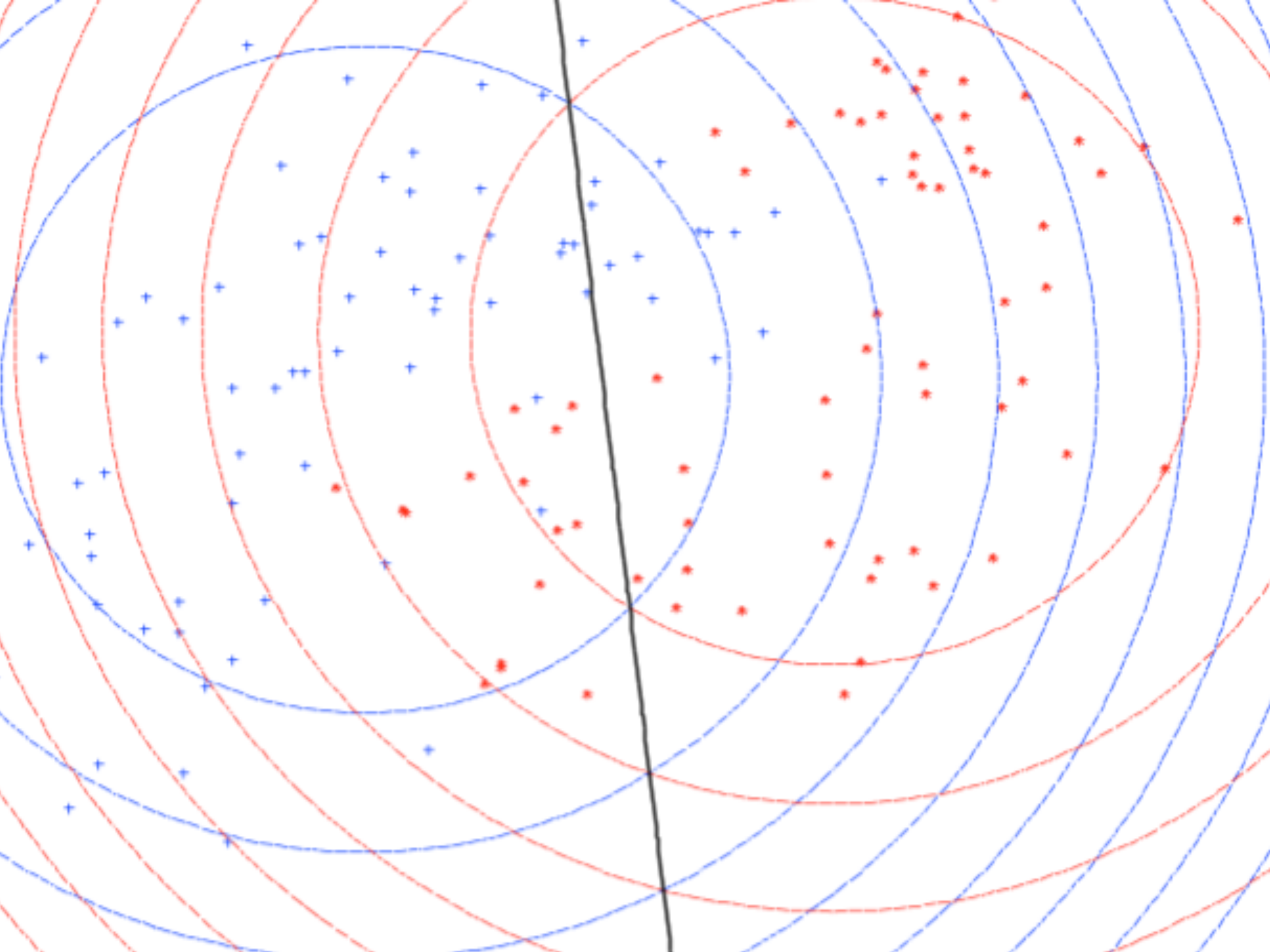
Different Classifier Complexity

Note: there is no single best classifier!

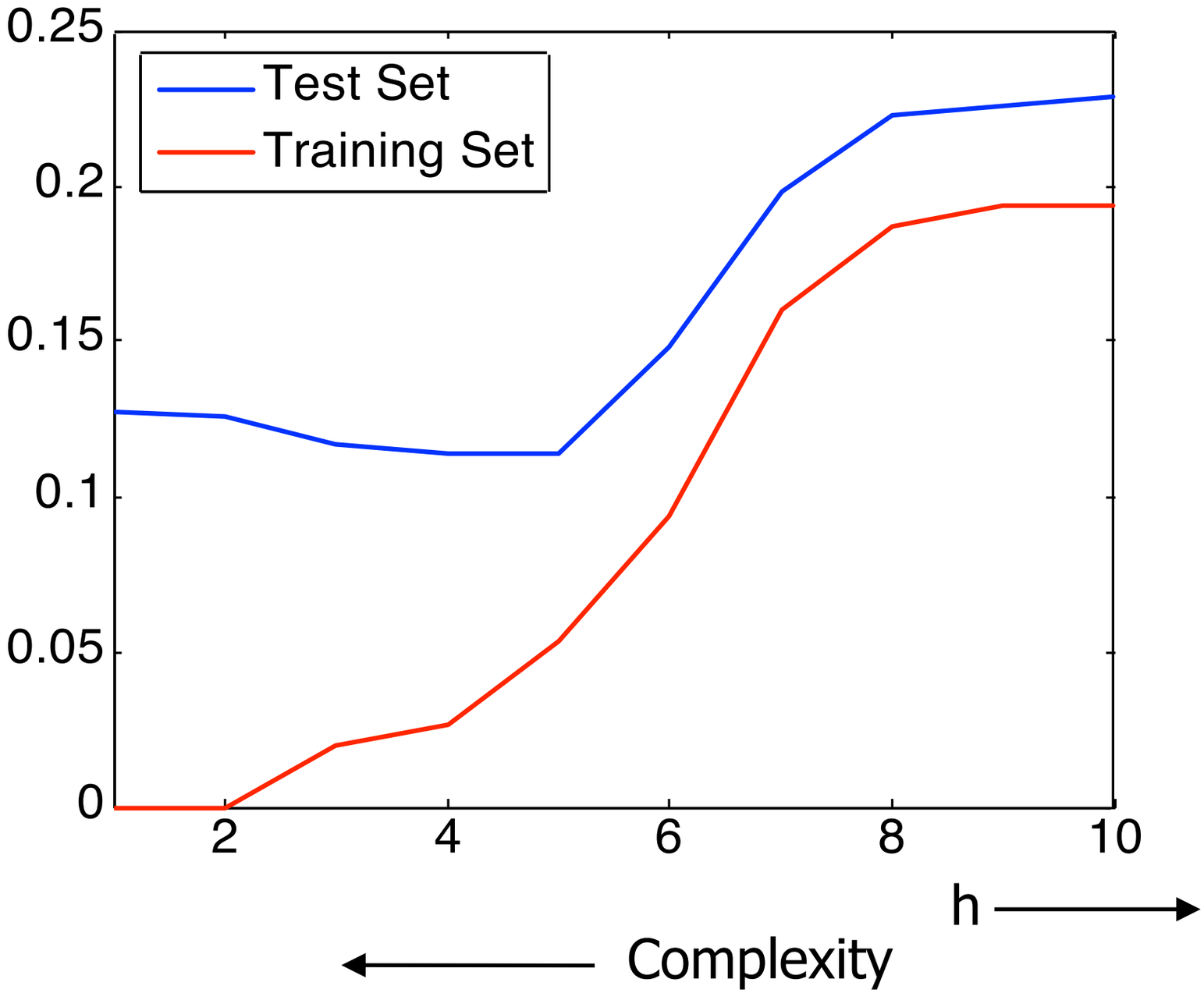


`Parzenc` Complexity Example





`Parzenc` Complexity Example



Some intermediate conclusions…

- Larger training sets yield better classifiers
- Independent test sets needed for unbiased error estimates
- Larger test sets yield more accurate error estimates
- LOO cross validation "optimal", but might be infeasible
- 10-fold crossvalidation is often used
- More complex classifiers need larger training sets
  - Same holds for larger feature set sizes
- Small training sets need simpler classifiers or smaller feature sets

Classifier evaluation

- This lecture:
  - Train and test set
  - Bootstrapping, cross-validation, leave-one-out
  - Learning curve
  - Classifier complexity
  - Bias-variance tradeoff
  - Confusion matrices
  - ROC curve
  - Reject curve

Squared Error

- When we have the error

$$L(\mathbf{w}) = E[\|g(\mathbf{x}) - y\|^2]$$

  we can actually derive something more general...

- In general?!

- Unfortunately, theory in Pattern Recognition is tough… how to deal with **all** possible datasets?

Bias-variance dilemma

- When we are given some data, we may get lucky, or unlucky:
  sometimes we get very a-typical data:-(

- To say something general, we need to **average** over different (training-) sets

$$\mathcal{D} = \{(y_i, \mathbf{x}_i); i = 1, .., N\}$$

- The classifier is now also a function of the training set:

$$g(\mathbf{x}; \mathcal{D})$$

Bias-variance dilemma

- Consider the squared error:

$$E_{\mathcal{D}}\left[(g(\mathbf{x};\mathcal{D}) - E[y|\mathbf{x}])^2\right]$$

- $E[y|\mathbf{x}]$ is the optimal mean-squared regressor (not proven now; see book)
- Now we do a trick:

$$= E_{\mathcal{D}}\left[(g(\mathbf{x};\mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})] + E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})] - E[y|\mathbf{x}])^2\right]$$

Bias-variance dilemma

- Now working out the square:

$$= E_{\mathcal{D}}\left[(g(\mathbf{x};\mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})])^2 \right.$$
$$+ 2(g(\mathbf{x};\mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})])(E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})] - E[y|\mathbf{x}])$$
$$\left. + (E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})] - E[y|\mathbf{x}])^2\right]$$

Bias-variance dilemma

- Now working out the square:

$$= E_{\mathcal{D}}\left[(g(\mathbf{x};\mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})])^2 + 2(g(\mathbf{x};\mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})])(E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})] - E[y|\mathbf{x}]) + (E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})] - E[y|\mathbf{x}])^2\right]$$

$$= E_{\mathcal{D}}\left[(g(\mathbf{x};\mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})])^2\right] + E_{\mathcal{D}}\left[2(g(\mathbf{x};\mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})])(E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})] - E[y|\mathbf{x}])\right] + E_{\mathcal{D}}\left[(E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})] - E[y|\mathbf{x}])^2\right]$$

(variance)

$= 0$

(bias²)

Bias-variance dilemma

(variance)

$$MSE = E_{\mathcal{D}}\left[(g(\mathbf{x};\mathcal{D}) - E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})])^2\right]$$
$$+ E_{\mathcal{D}}\left[(E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})] - E[y|\mathbf{x}])^2\right]$$

(bias^2)

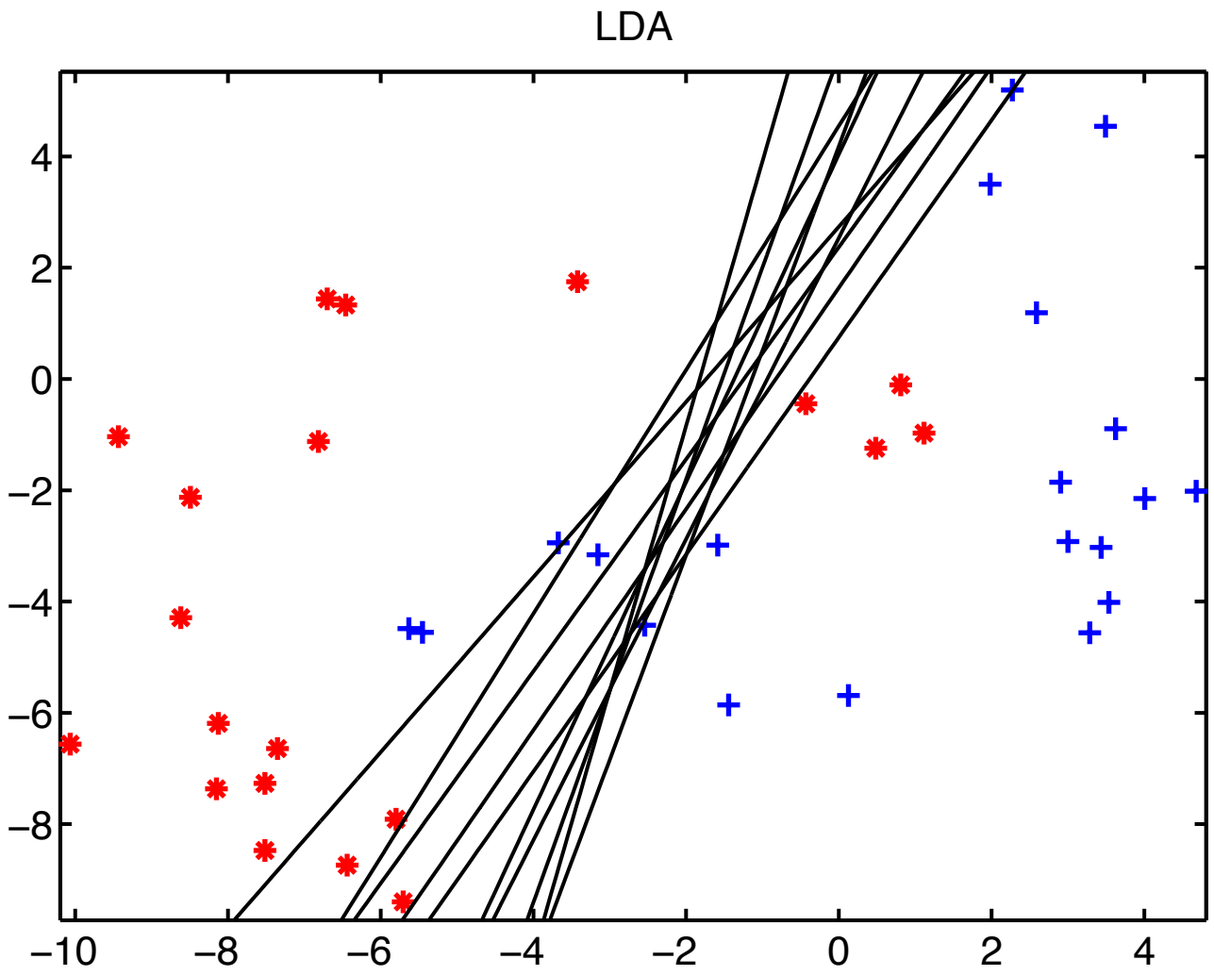
- variance: how much does classifier g vary over different training sets

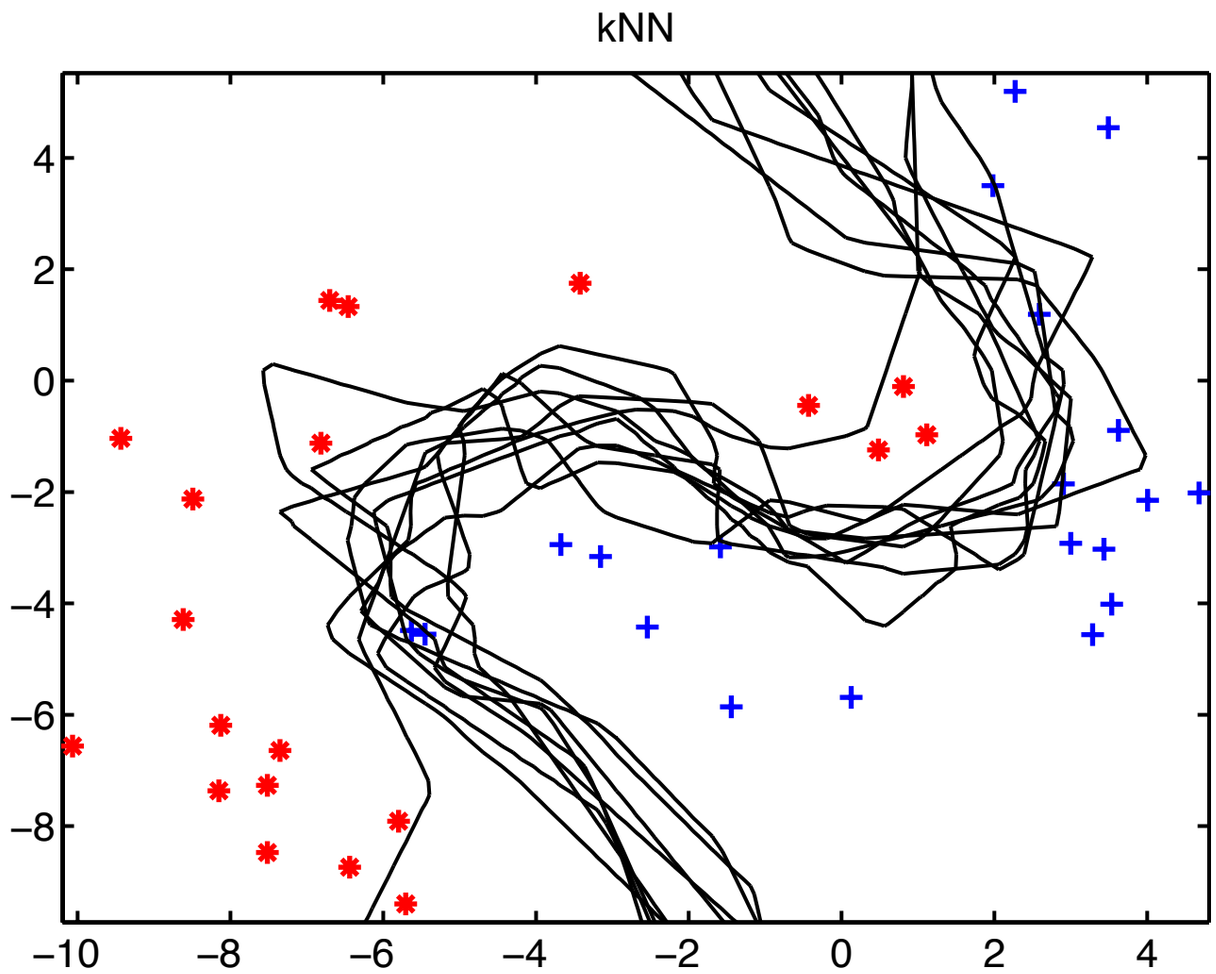
- bias: how much does the average classifier g differ from the true output

Bias-variance dilemma

- Compare LDA with kNN:

low variance, high bias





high variance, low bias

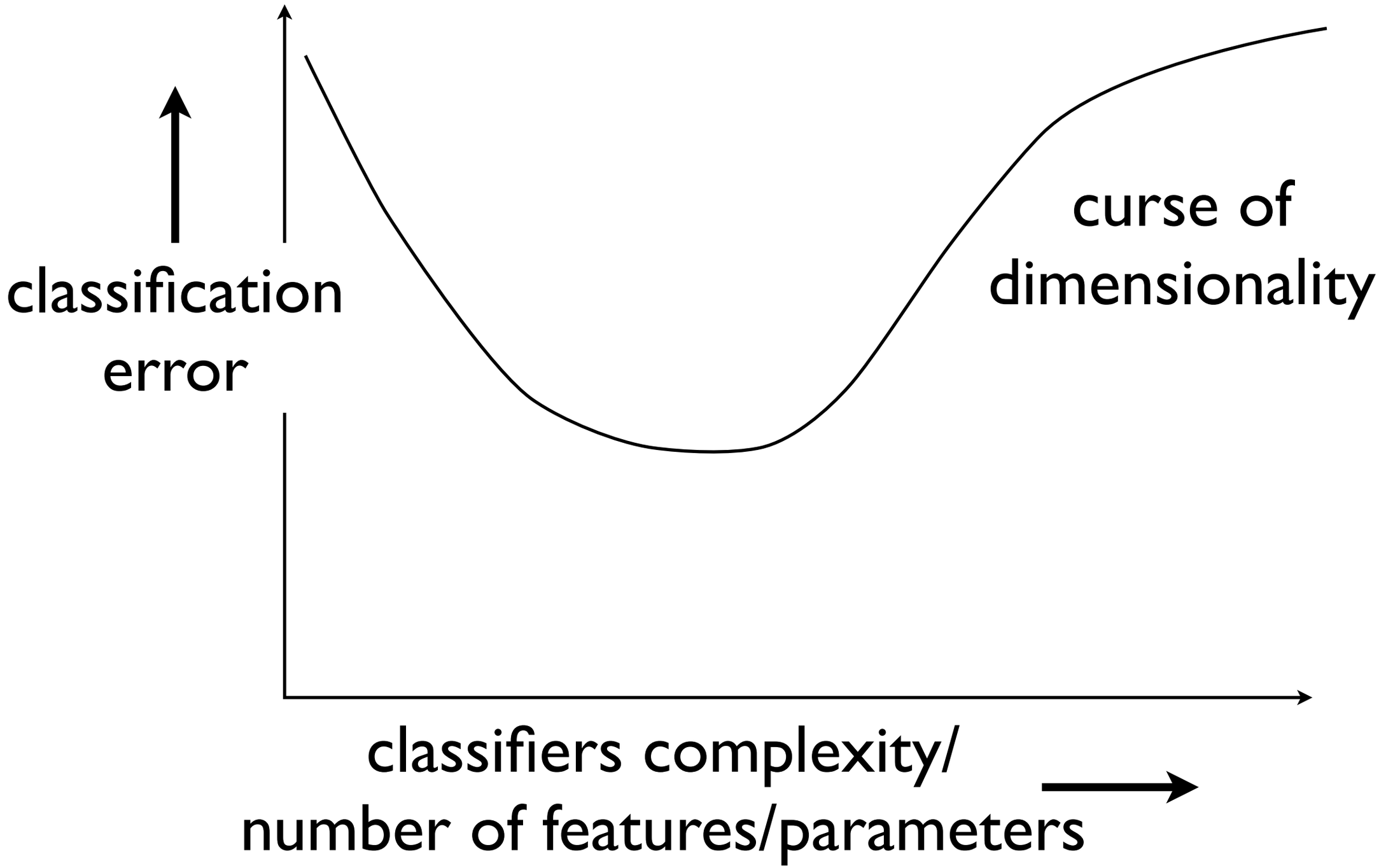
TU Delft

Bias-variance tradeoff

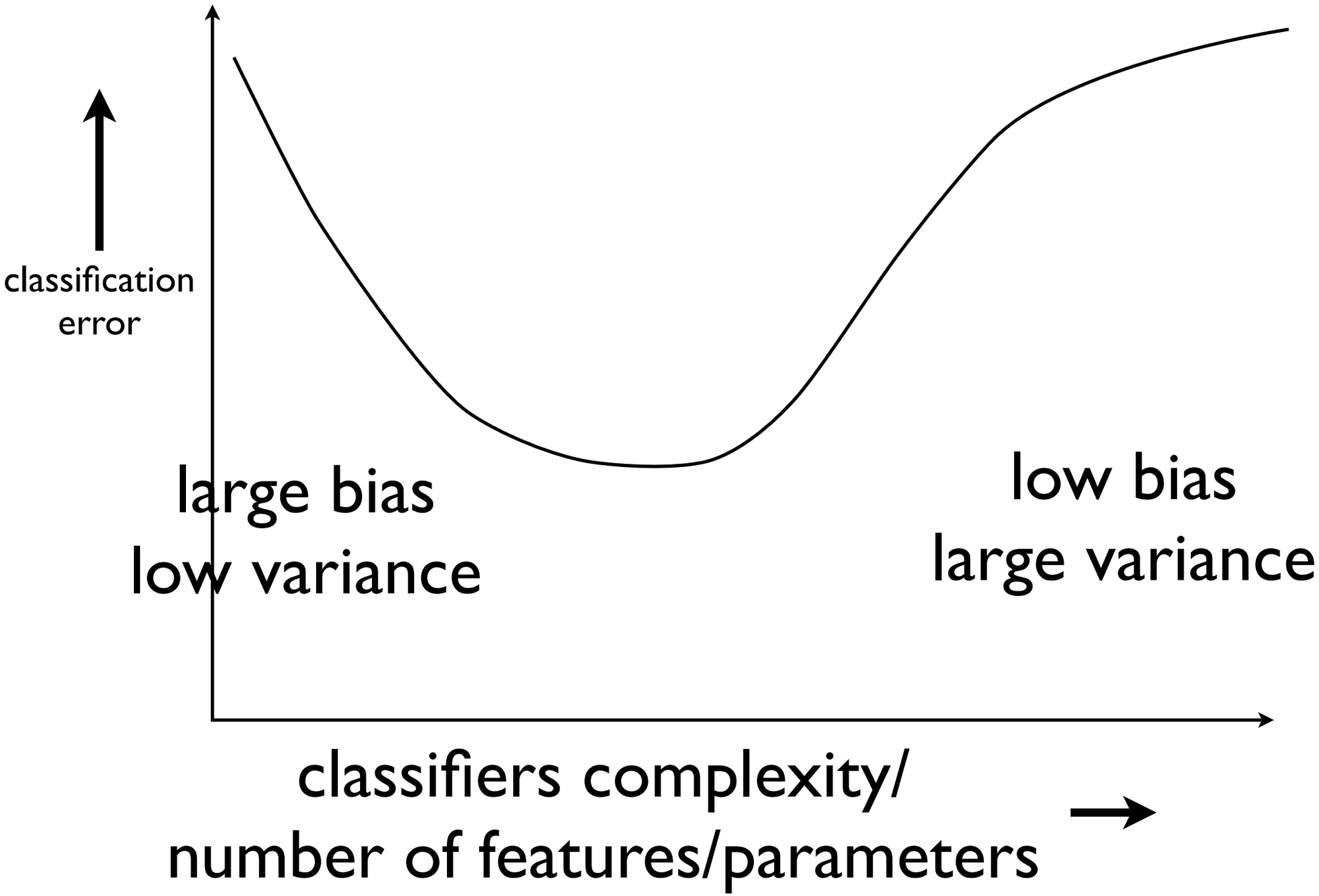
- Originally derived for neural networks and squared error
- It is a very general phenomenon: we encounter it more often in pattern recognition

- More simple classifier is more stable (and need less data to train)
- More complex classifier only works when you have sufficient number of training data

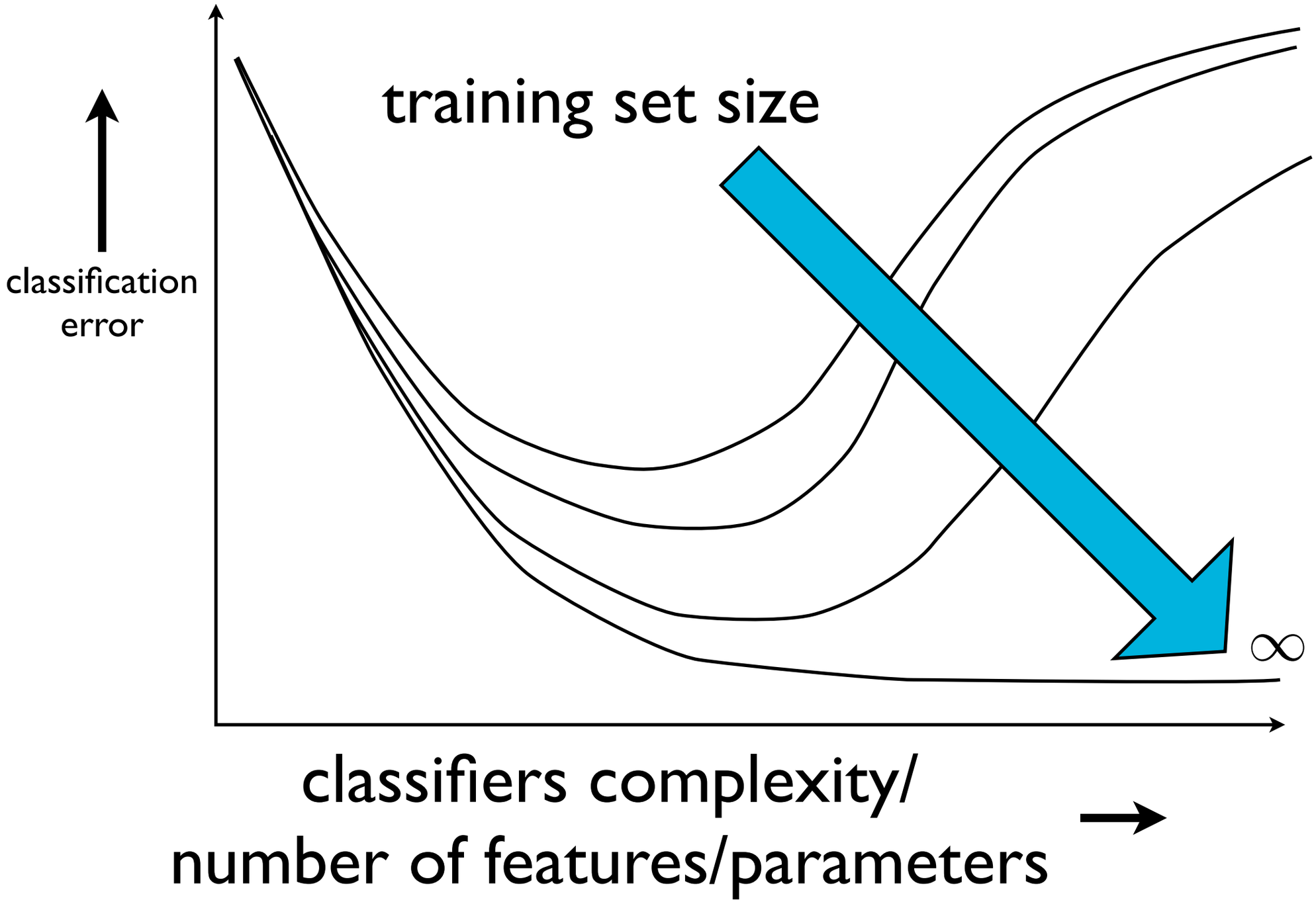
Feature curve



Feature curve



Feature curve



For two-class classification

- There is a fundamental tradeoff between the two errors/performances of the two classes

- Standard classification error: $\varepsilon = \varepsilon_1 p(y_1) + \varepsilon_2 p(y_2)$

- Weighted classification error:

$$\varepsilon = \lambda_{12}\varepsilon_1 p(y_1) + \lambda_{21}\varepsilon_2 p(y_2)$$

- F1-score (harmonic mean):

$$F_1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- ...

Error / Performance Measures

- **Error** : probability of erroneous classifications
- **Performance / accuracy** : 1 – error
- **Sensitivity** of a target class [e.g. diseased patients] : performance for objects from that target class
- **Specificity** : performance for all objects outside target class
- **Precision** of a target class : fraction of correct objects among all objects assigned to that class.
- **Recall** : fraction of correctly classified objects; identical to sensitivity when related to particular class
- **True positive rate** : identical to sensitivity
- **False positive rate** : error for all objects outside target

Confusion Matrices

- Provides counts of class-dependent errors : How many object have been classified as A that should have been classified as B?

- Give a more detailed view than overall error rate

- Can be used to estimate overall cost for particular classifier

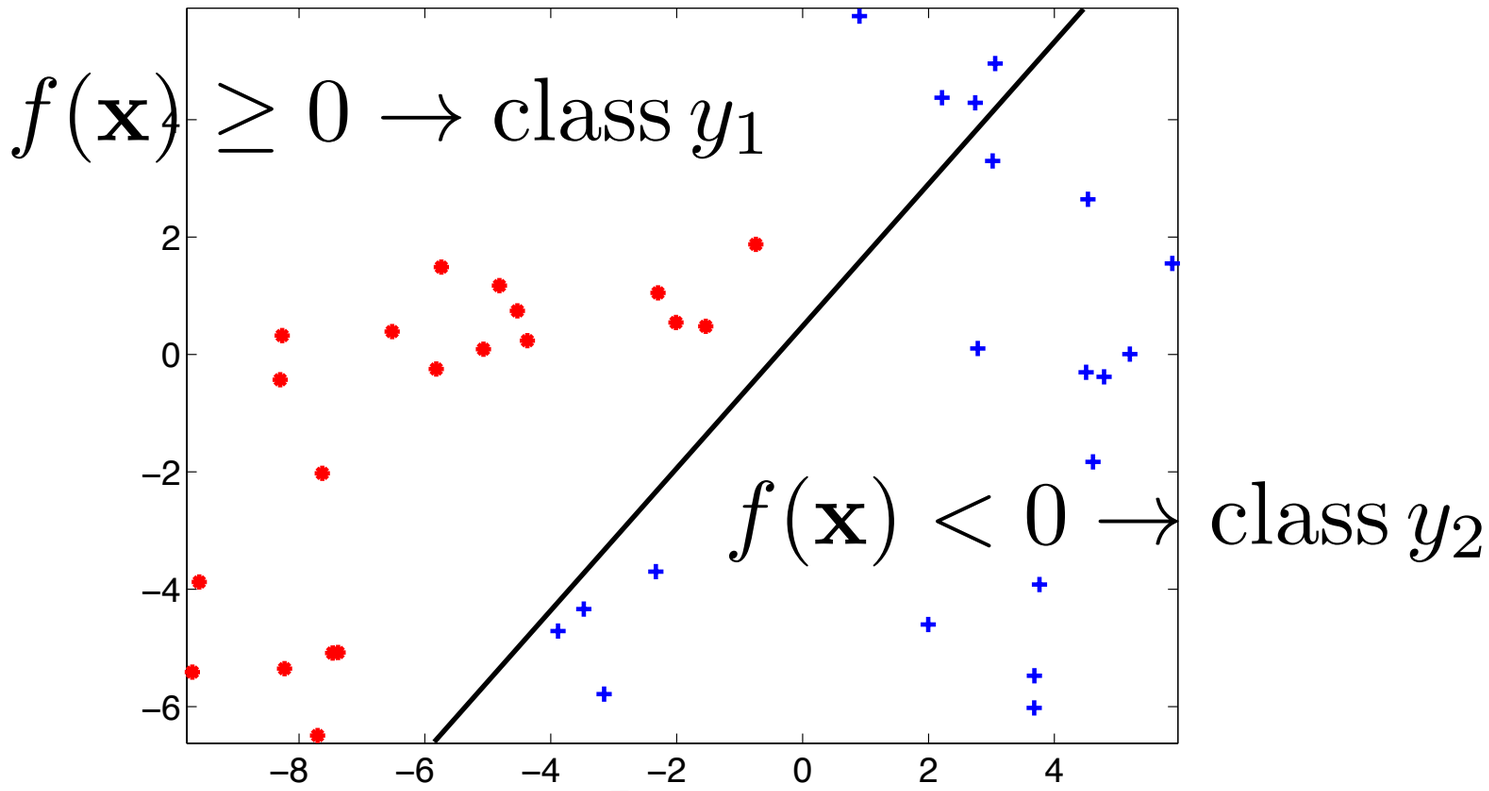
Confusion Matrix (1)

Real labels:

$$\Lambda = \begin{bmatrix} \lambda_1 \\ \dots \\ \lambda_N \end{bmatrix}$$

Predicted labels:

$$L = \begin{bmatrix} l_1 \\ \dots \\ l_N \end{bmatrix}$$



Confusion matrix:

$$C = \begin{bmatrix} c_{11} & \dots & c_{1K} \\ \dots & \dots & \dots \\ c_{K1} & \dots & c_{KK} \end{bmatrix}$$

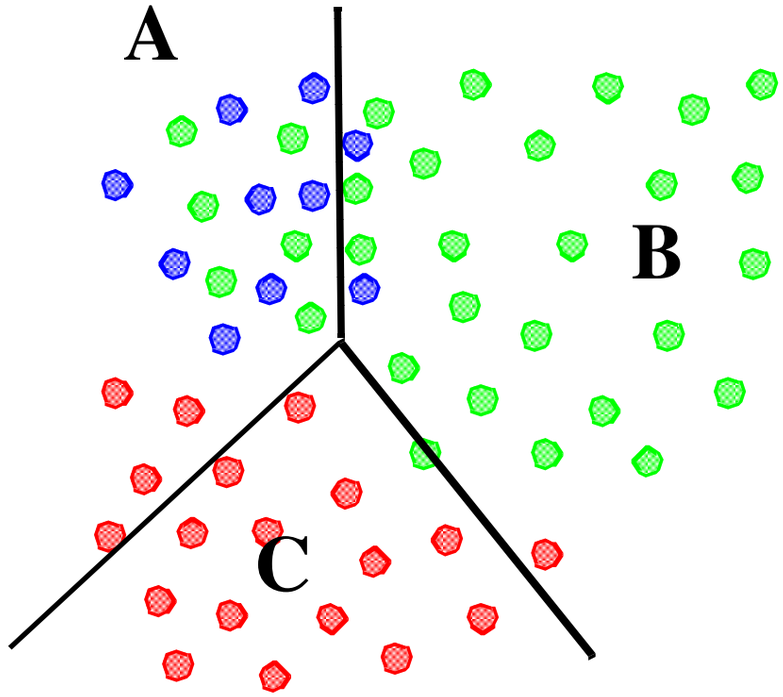
$$c_{ij} = N \cdot P[l_j | \lambda_i]$$

Confusion Matrix (2)

$$N_A = 10,\ N_B = 30,\ N_C = 20$$

$$E = \frac{c_{12} + c_{13} + c_{21} + c_{23} + c_{31} + c_{32}}{N_A + N_B + N_C}$$

$$E = 14/60 = 0.2333$$



$C = \text{confmat}(\Lambda, L)$

Λ real labels

L obtained labels

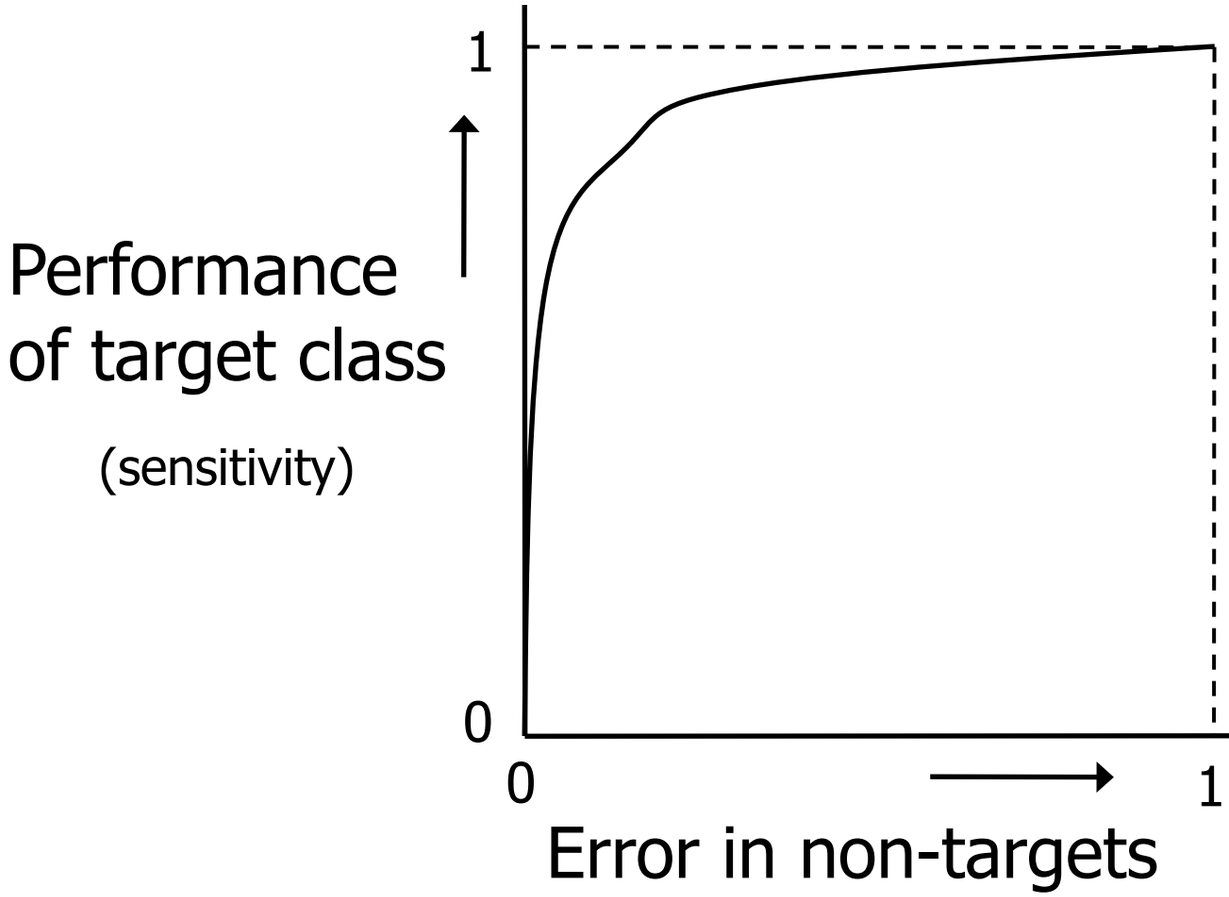
objects from	classified to A	B	C
class A	8	2	0
class B	6	23	1
class C	4	1	15

0.20 error in class A
0.23 error in class B
0.25 error in class C

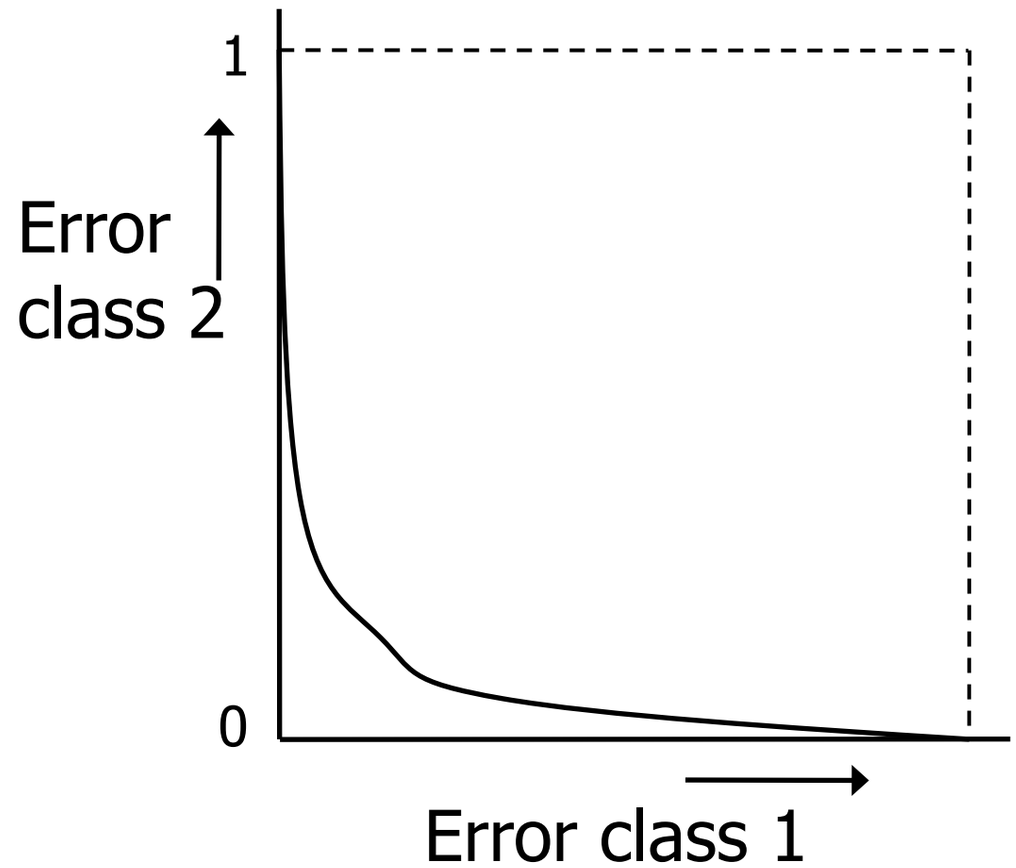
0.228 averaged error

ROC Analysis

ROC: Receiver-Operator Characteristic (from communication theory)



Medical diagnostics
Database retrieval

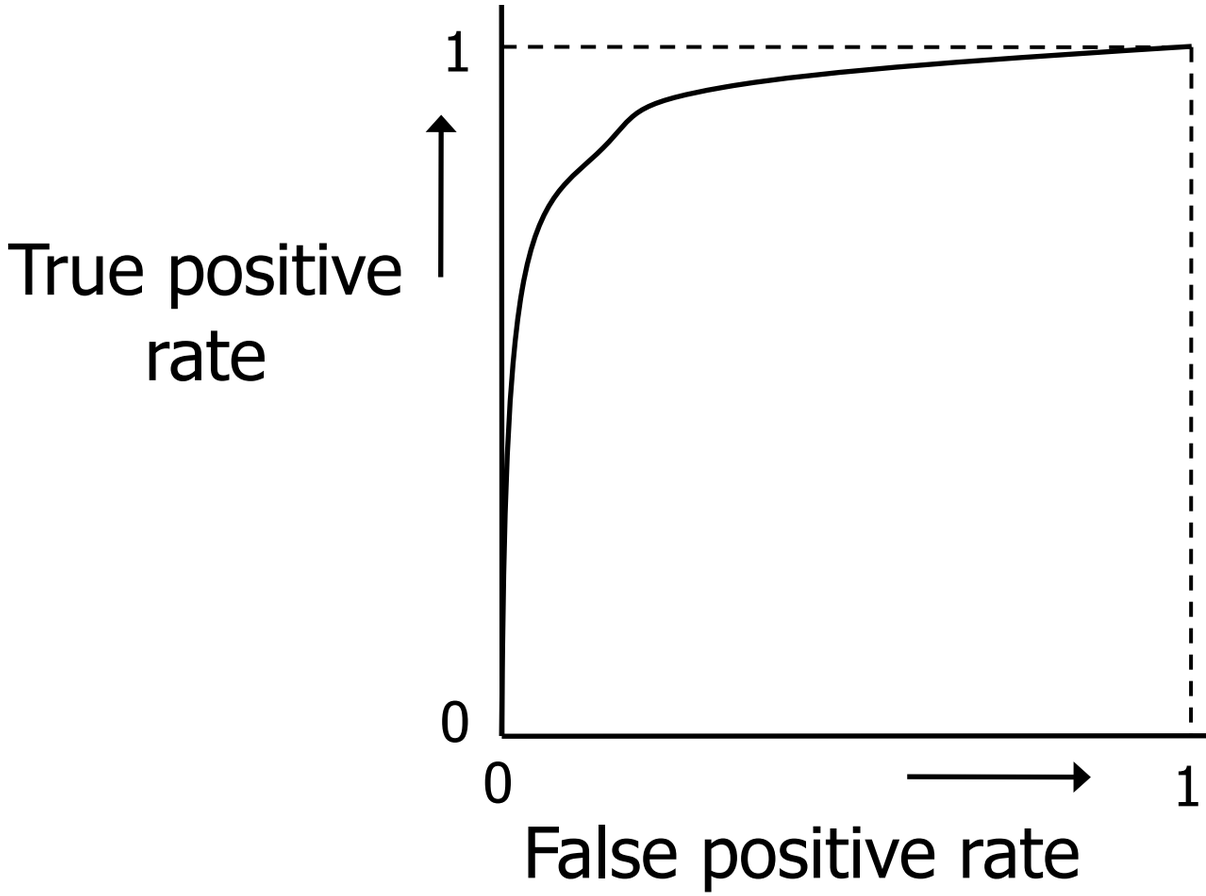


2-class pattern recognition

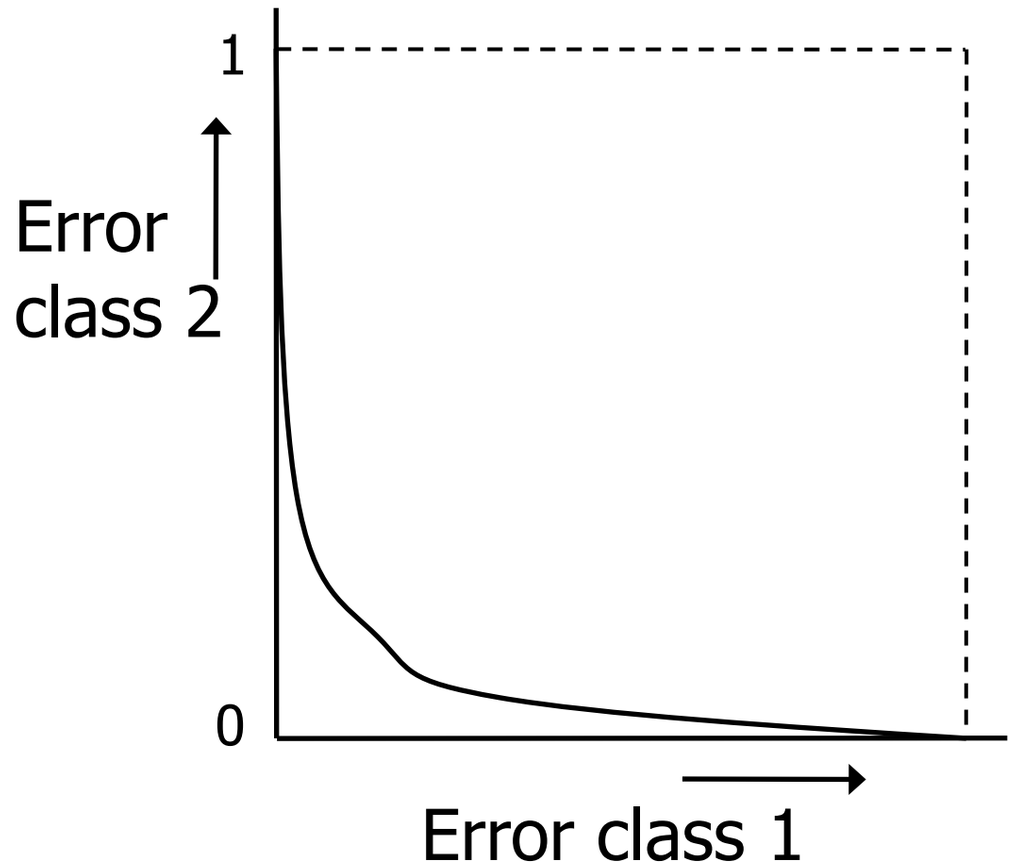
TU Delft

ROC Analysis

ROC: Receiver-Operator Characteristic (from communication theory)



Medical diagnostics
Database retrieval

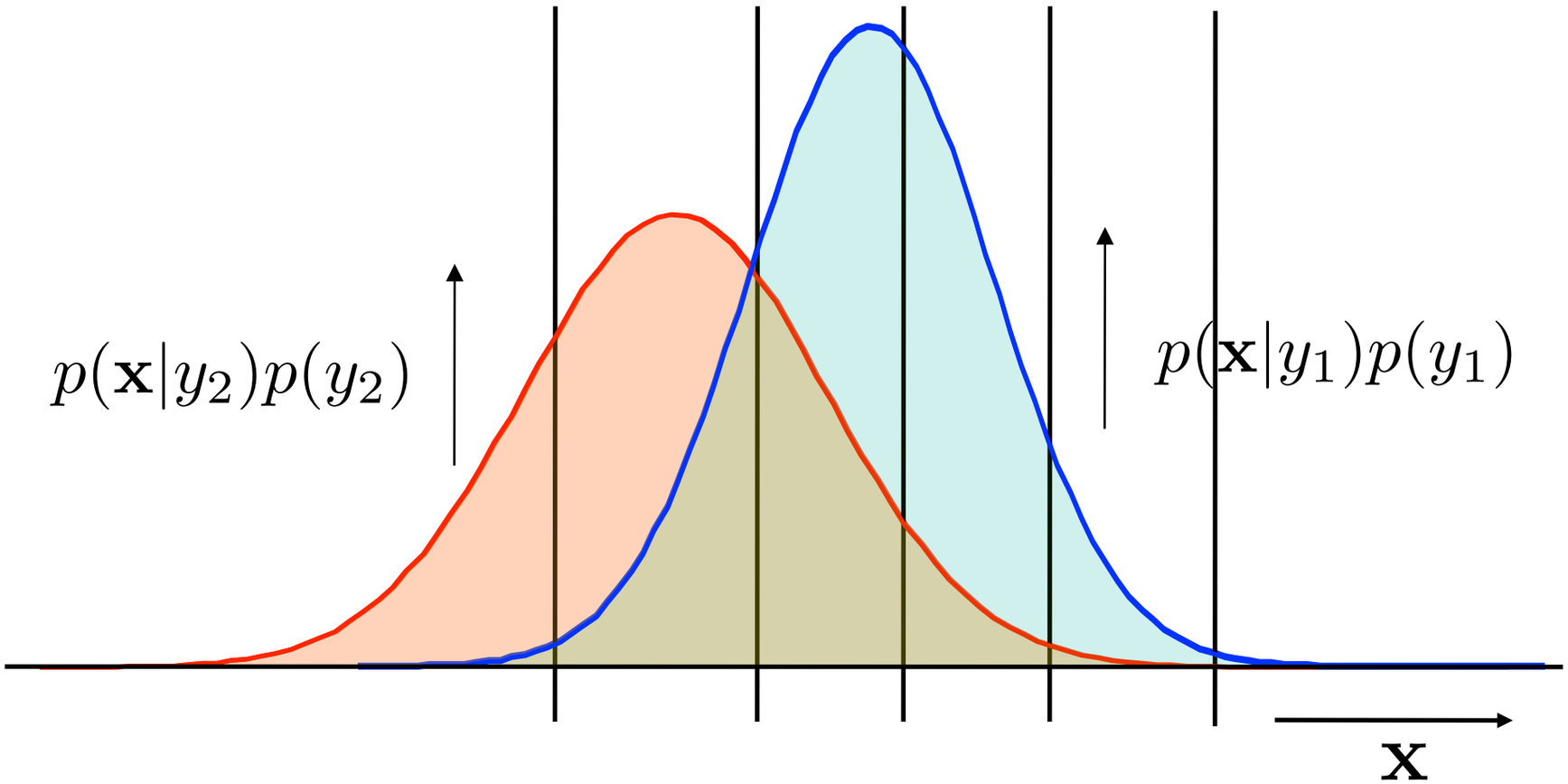


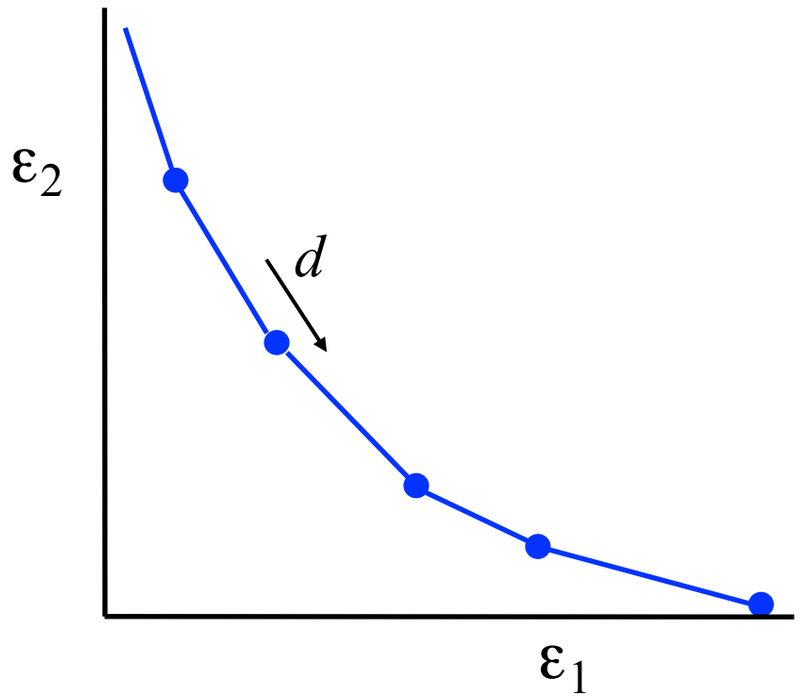
2-class pattern recognition

TU Delft

ROC Curve

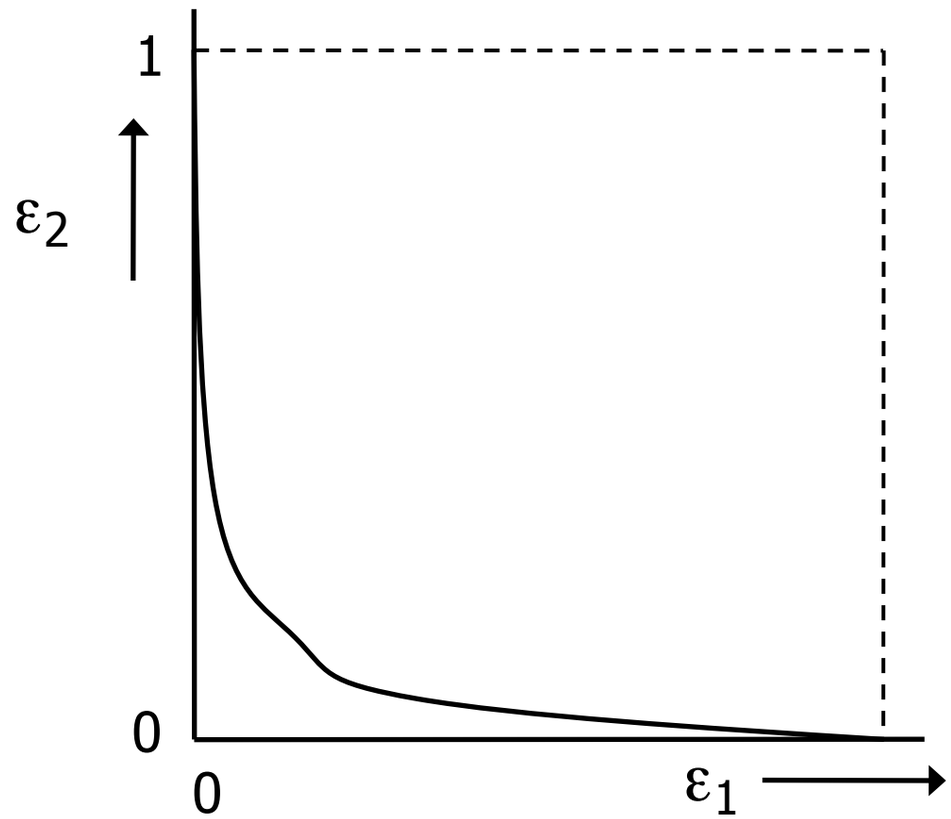
- Curve is obtained by varying classifier threshold d





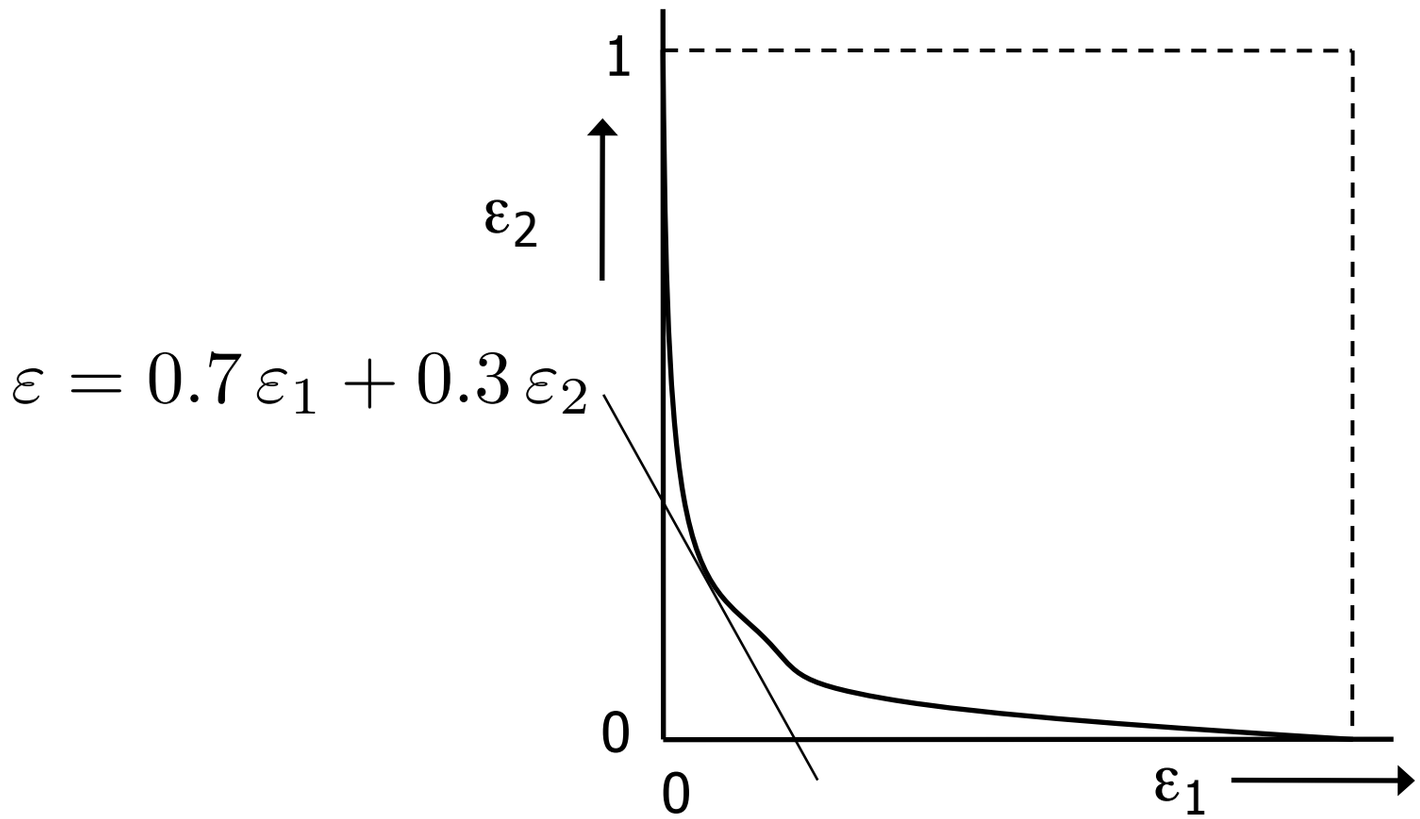
TU Delft

Effect of Changing Priors/Costs



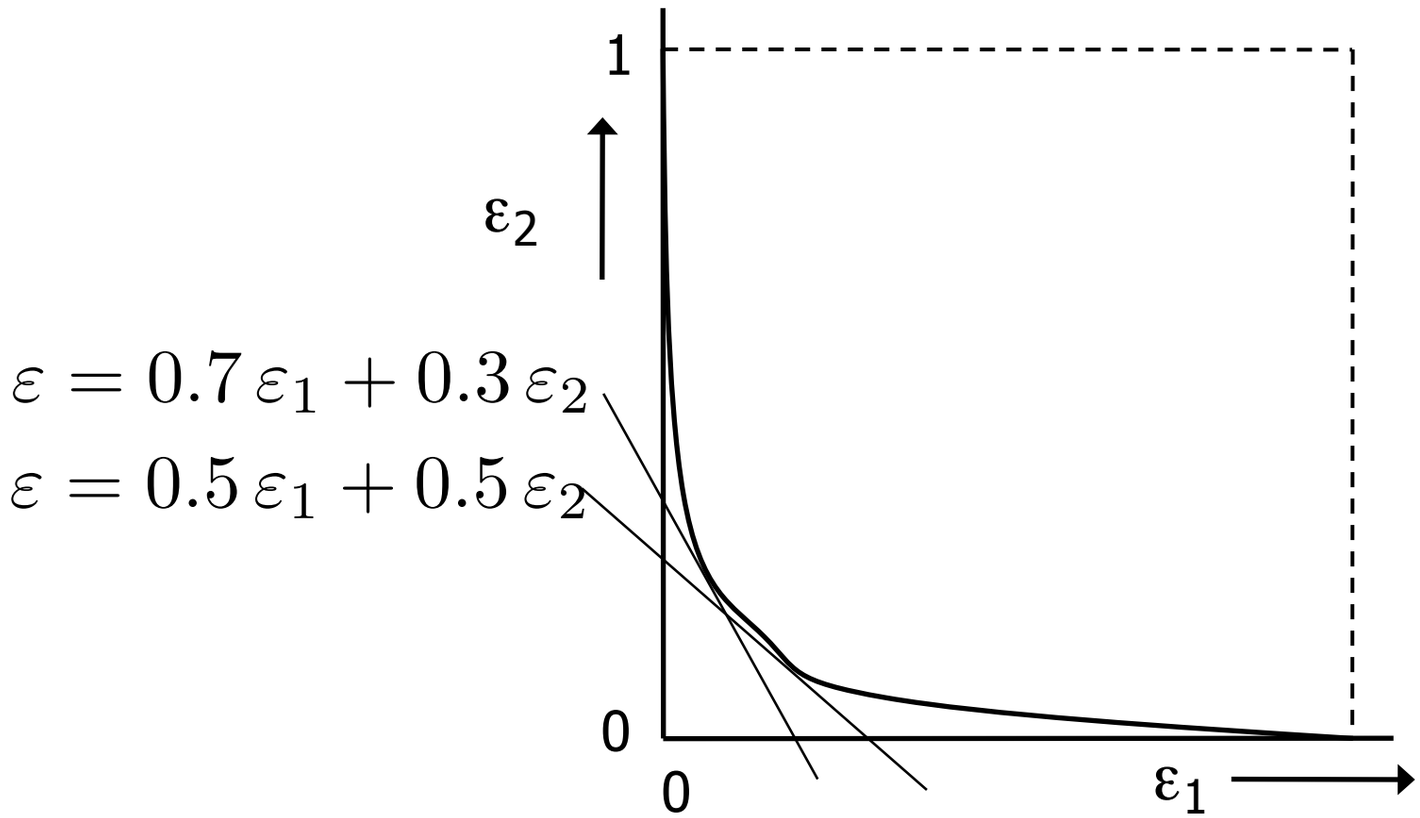
$$\varepsilon = p(y_1)\,\varepsilon_1 + p(y_2)\,\varepsilon_2$$

Effect of Changing Priors/Costs



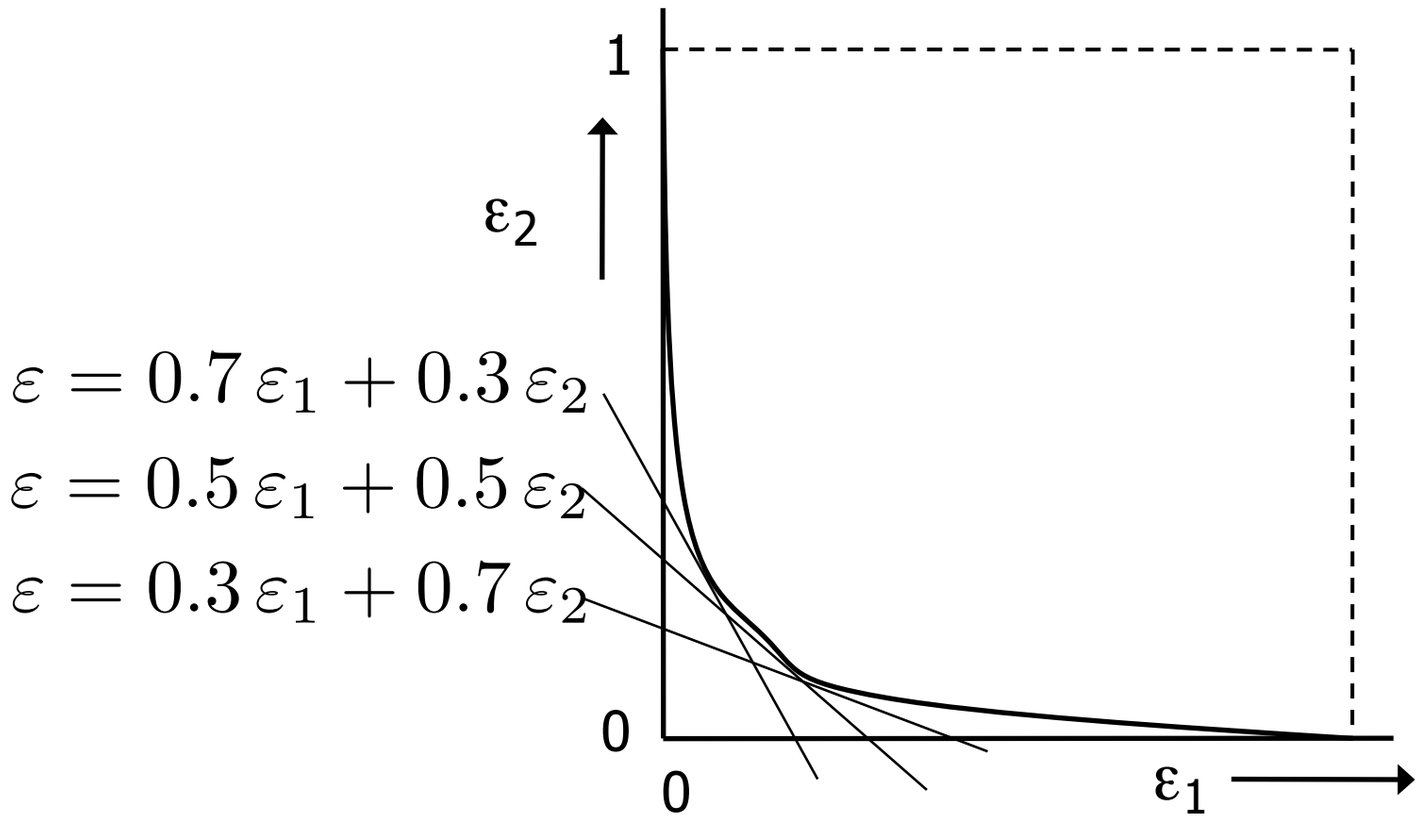
$$\varepsilon = p(y_1)\,\varepsilon_1 + p(y_2)\,\varepsilon_2$$

Effect of Changing Priors/Costs



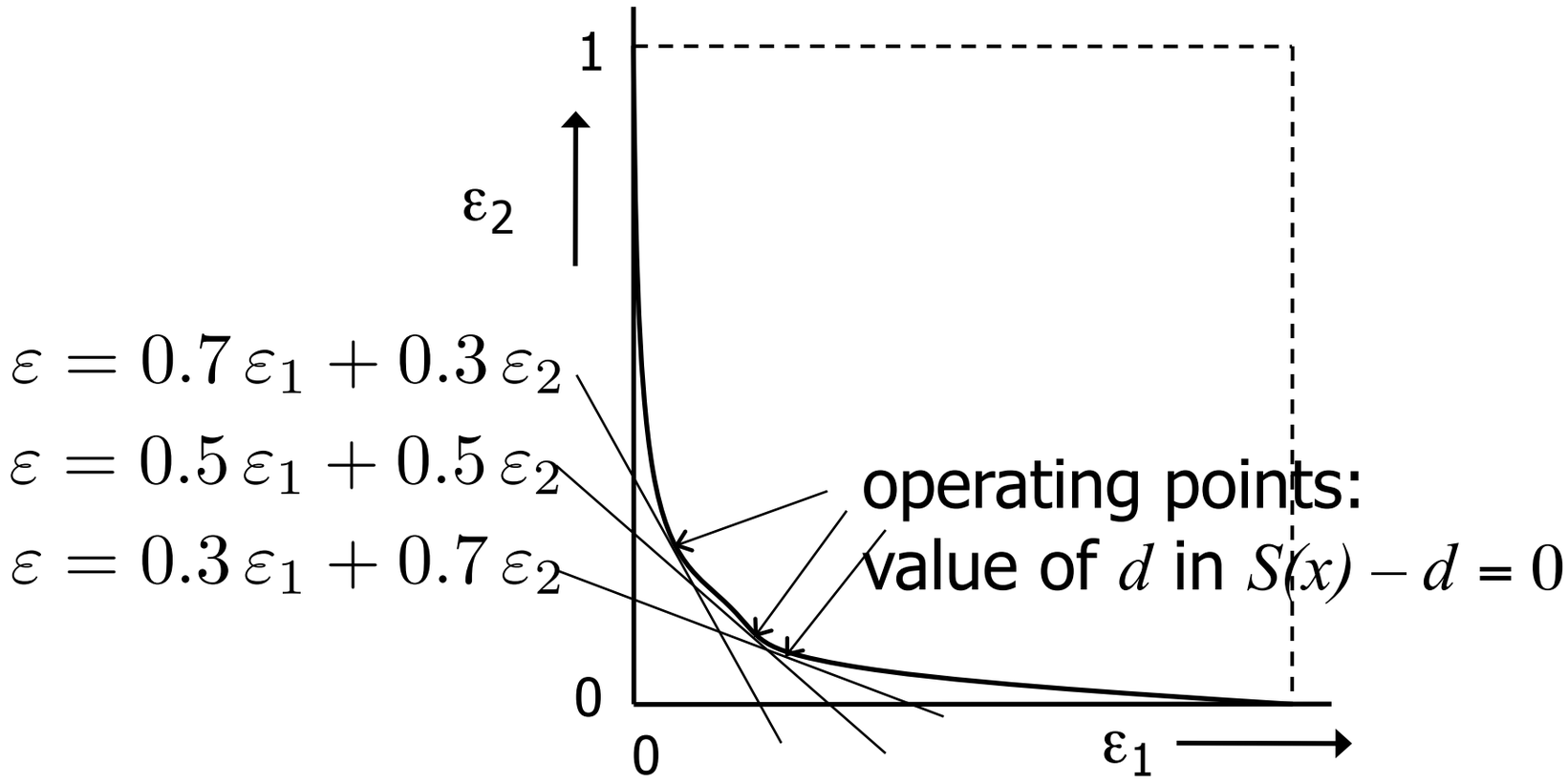
$$\varepsilon = p(y_1)\,\varepsilon_1 + p(y_2)\,\varepsilon_2$$

Effect of Changing Priors/Costs



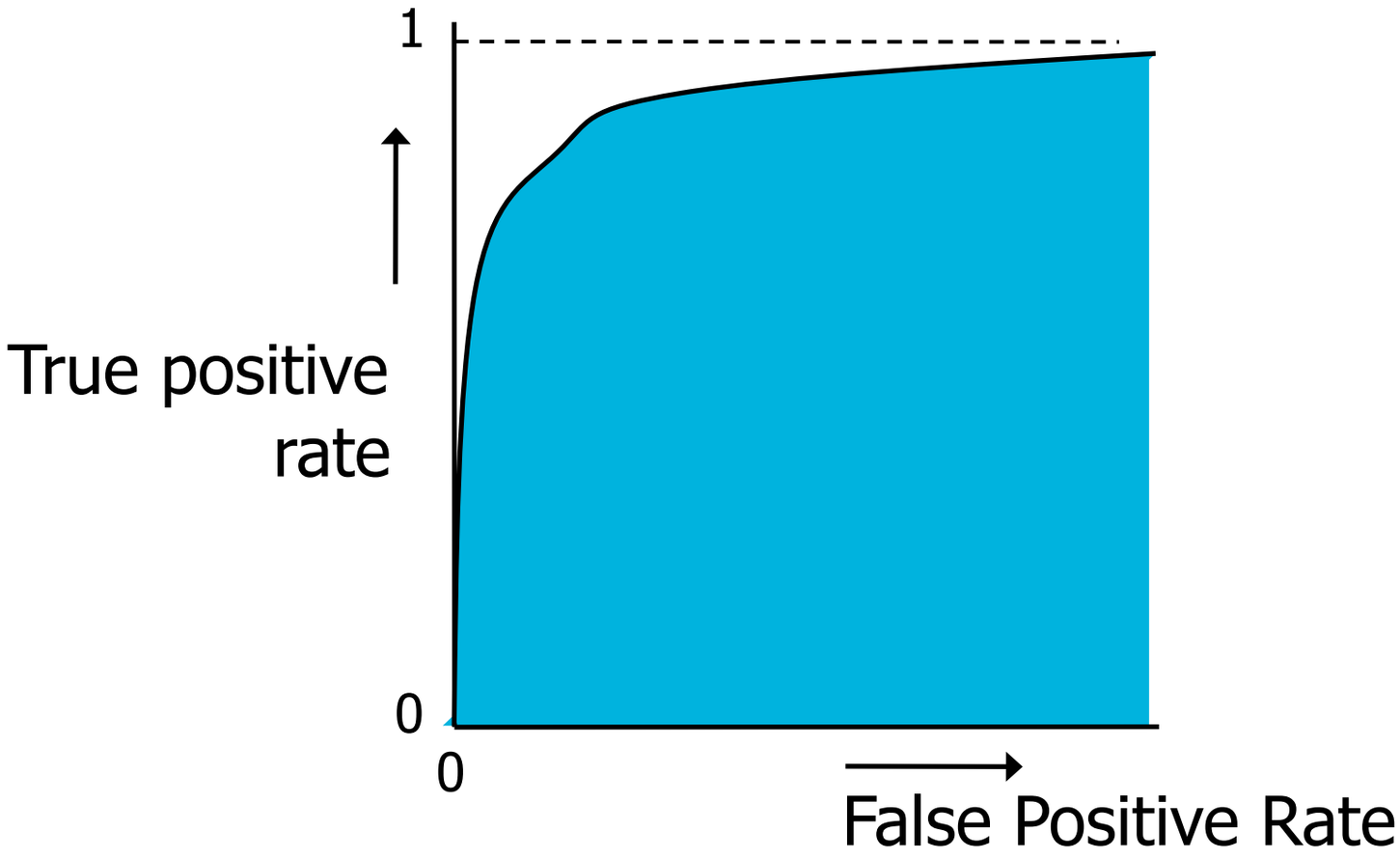
$$\varepsilon = p(y_1)\,\varepsilon_1 + p(y_2)\,\varepsilon_2$$

Effect of Changing Priors/Costs



$$\varepsilon = p(y_1)\,\varepsilon_1 + p(y_2)\,\varepsilon_2$$

Area under the ROC curve: AUC



- Integrate the area: perfect classifier gives AUC=1.0
- Random classifier gives AUC=0.5
- Performance measure that is insensitive to class priors

Example to compute ROC curve

- Assume I trained a classifier, and testing it on a test set:
- Four objects of class 1:

$$\hat{p}(y_1|\mathbf{x}_1) = 0.3$$
$$\hat{p}(y_1|\mathbf{x}_2) = 0.8$$
$$\hat{p}(y_1|\mathbf{x}_3) = 0.9$$
$$\hat{p}(y_1|\mathbf{x}_4) = 0.55$$

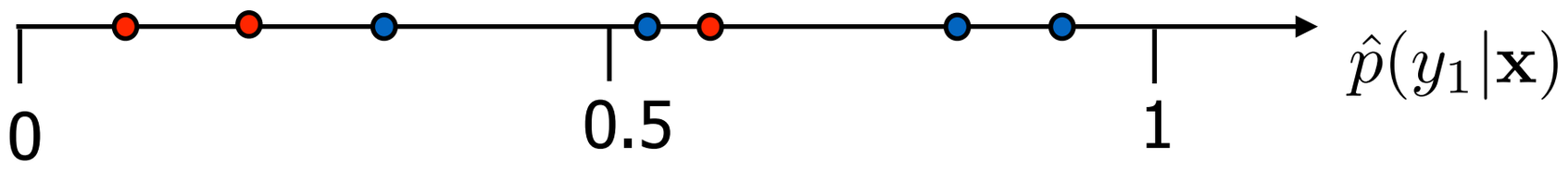
- Three objects of class 2:

$$\hat{p}(y_1|\mathbf{x}_5) = 0.2$$
$$\hat{p}(y_1|\mathbf{x}_6) = 0.1$$
$$\hat{p}(y_1|\mathbf{x}_7) = 0.6$$

- How does the ROC curve look like? ε_2 vs. ε_1

Example to compute ROC curve

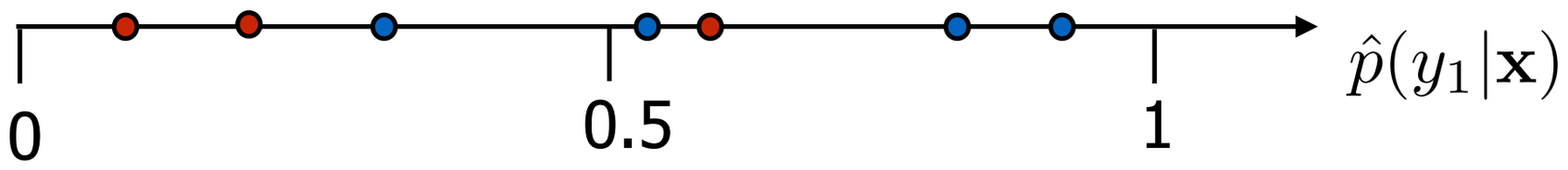
- Find out how obj's are classified for different thresholds



- For example, what are the errors for $d = 0.5$?

Example to compute ROC curve

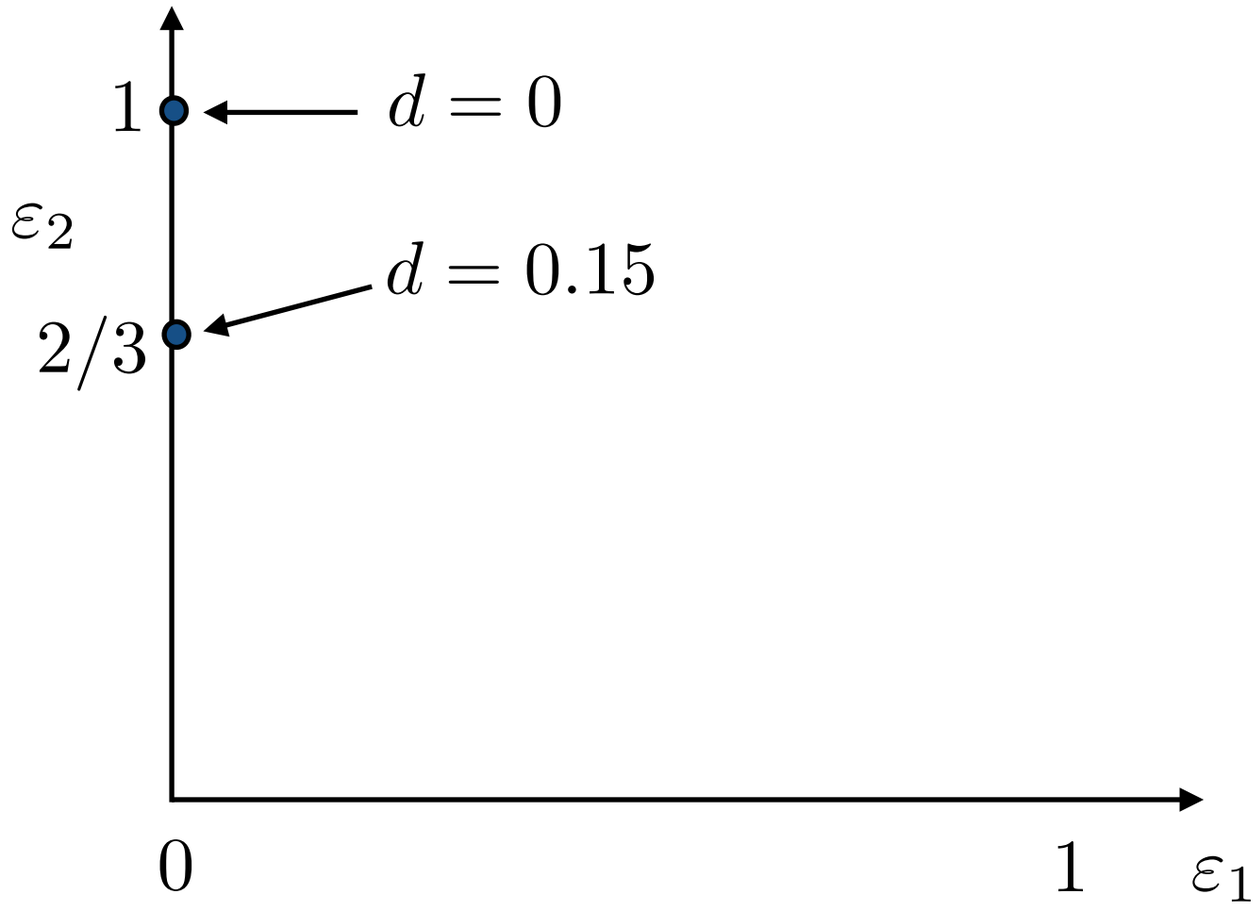
- Find out how obj's are classified for different thresholds



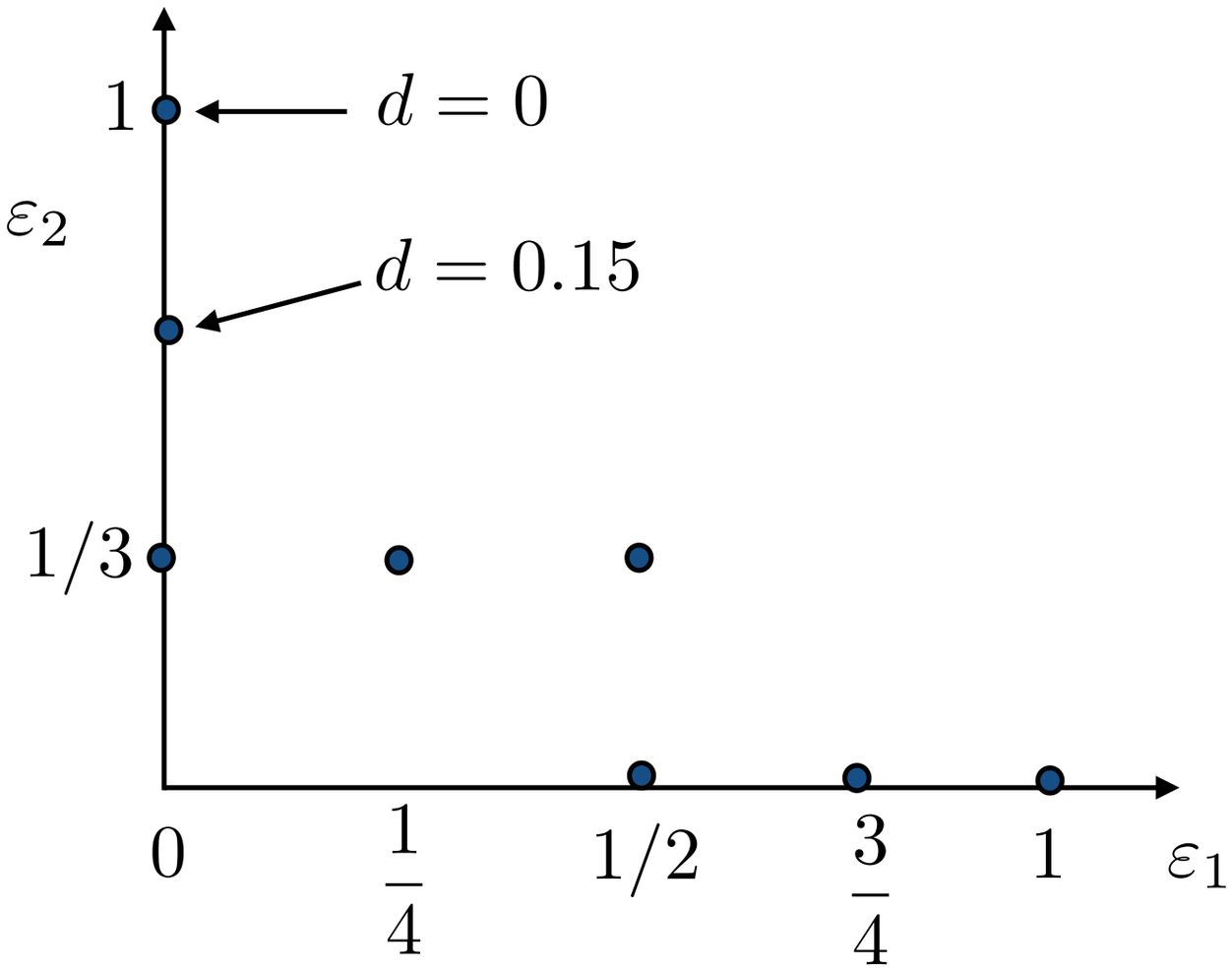
- For example, what are the errors for $d = 0.5$?

$$\varepsilon_1 = 1/4$$
$$\varepsilon_2 = 1/3$$

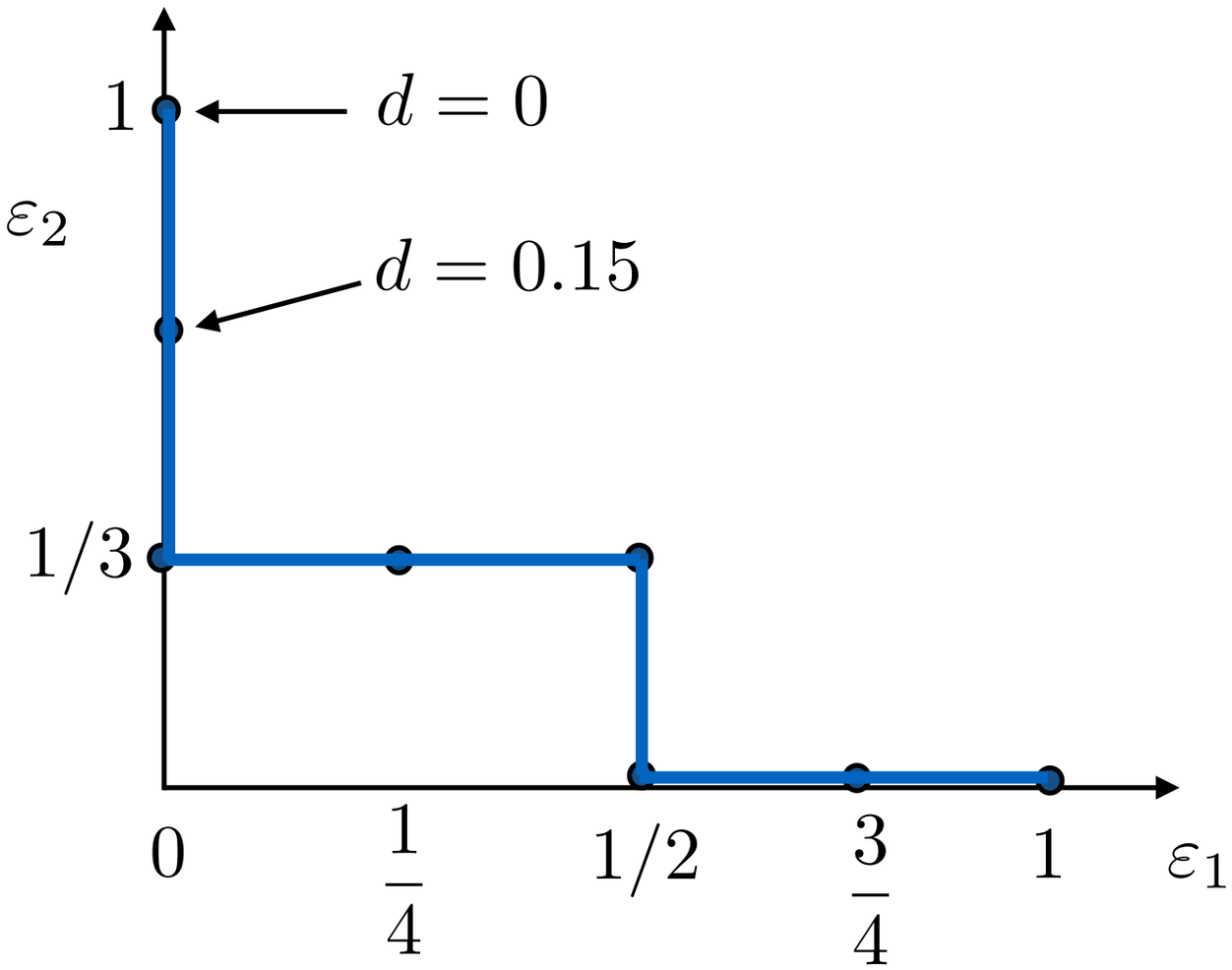
Example to compute ROC curve



Example to compute ROC curve



Example to compute ROC curve



Conclusions

- There is no best classifier
- There are alternative principles to find a good classifier:
  - maximising the likelihood
  - minimising the classification error
  - minimising the mean squared error
  - ...
- There is a fundamental tradeoff between the bias and variance of a classifier (depending on how flexible/ complex a classifier is)
- Finding the correct regulariser is a ‘black art’ of ML