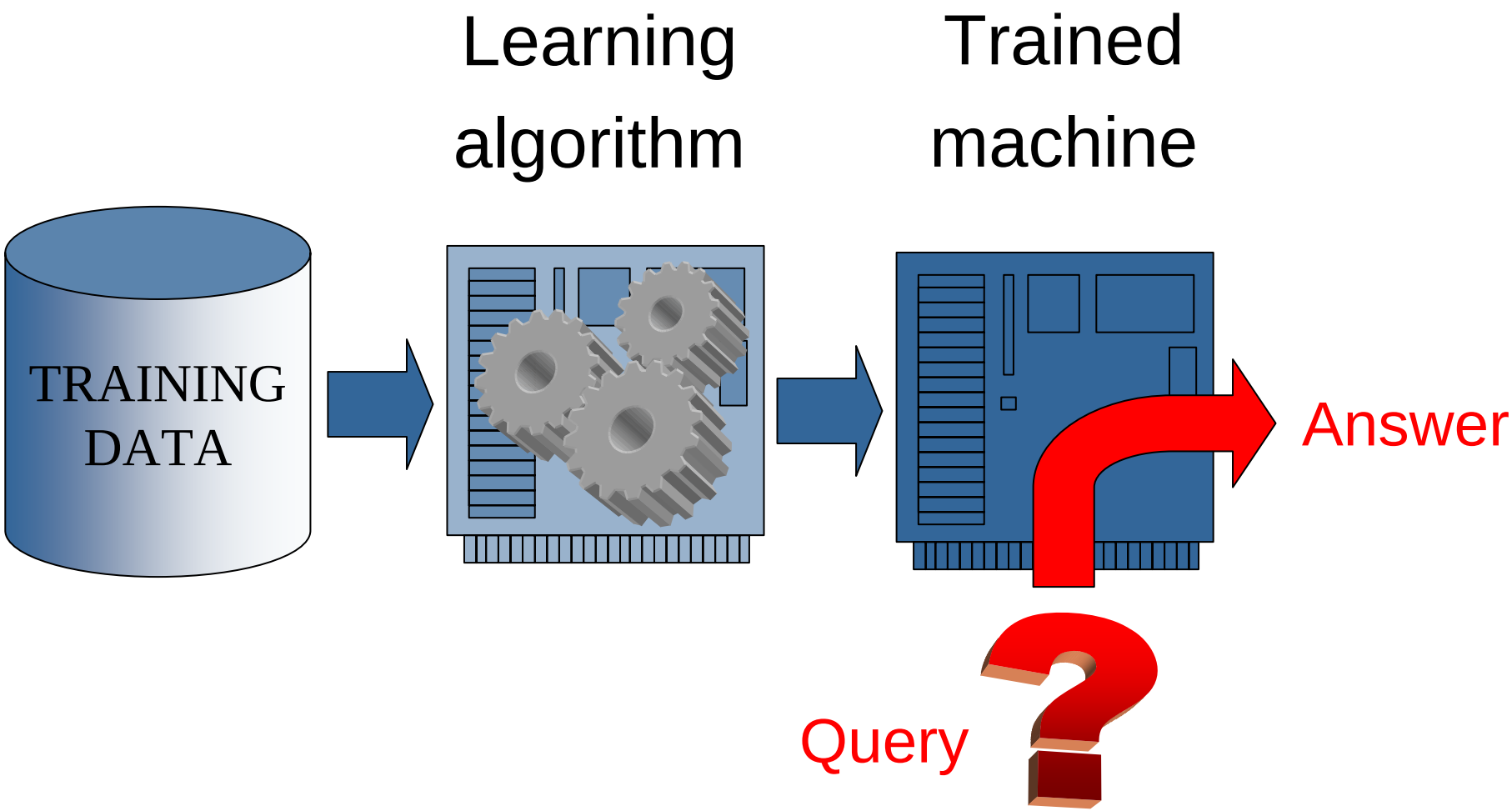
Introduction to Machine Learning

Isabelle Guyon

isabelle@clopinet.com

What is Machine Learning?



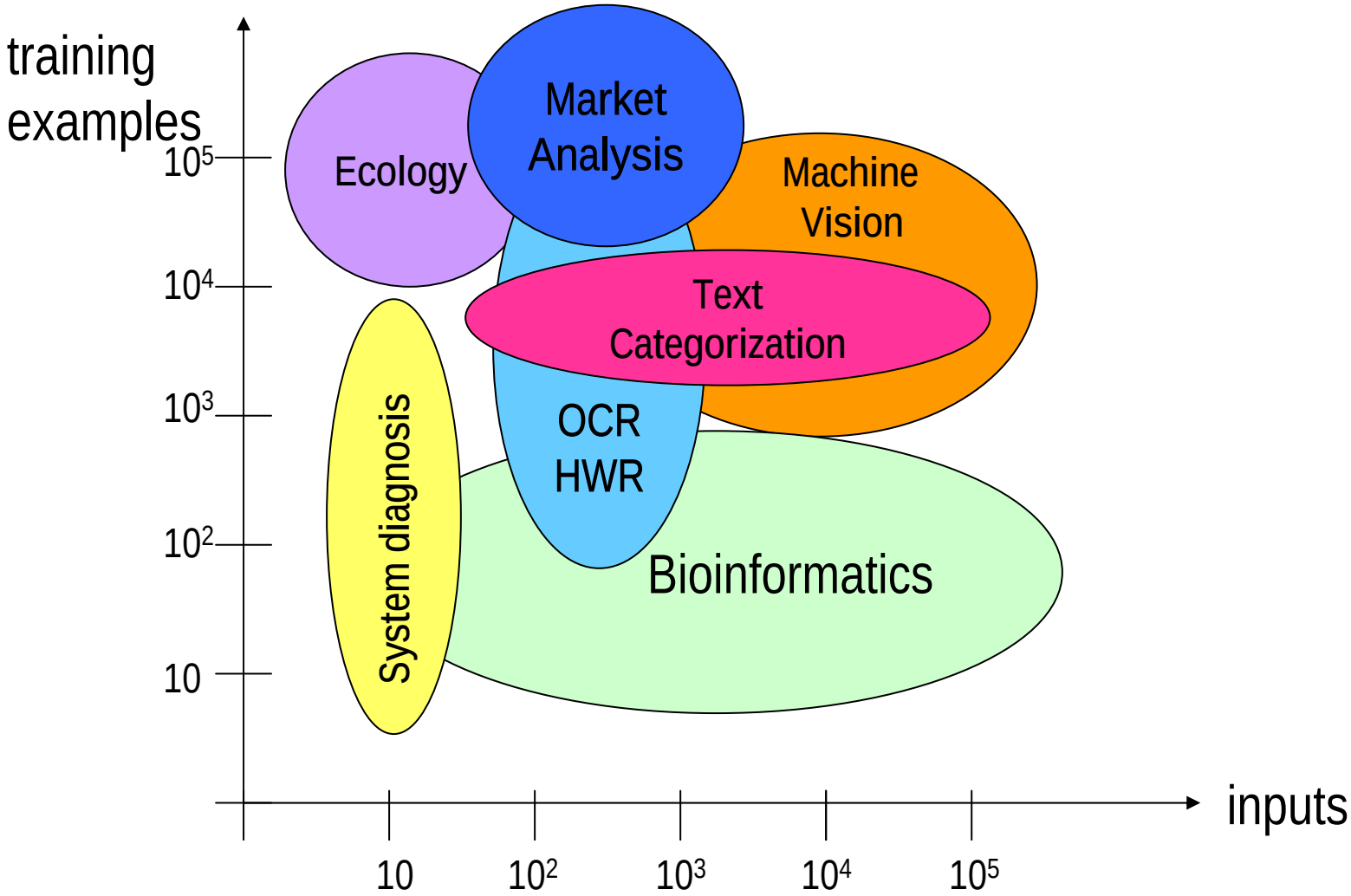
What for?

- Classification
- Time series prediction
- Regression
- Clustering

Some Learning Machines

- Linear models
- Kernel methods
- Neural networks
- Decision trees

Applications



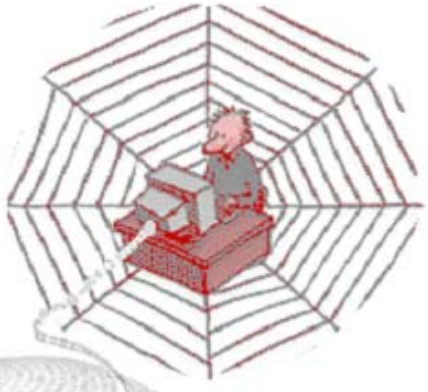
Banking / Telecom / Retail

- **Identify:**
  - **Prospective customers**
  - **Dissatisfied customers**
  - **Good customers**
  - **Bad payers**
- **Obtain:**
  - **More effective advertising**
  - **Less credit risk**
  - **Fewer fraud**
  - **Decreased churn rate**

Biomedical / Biometrics

- **Medicine:**
  - **Screening**
  - **Diagnosis and prognosis**
  - **Drug discovery**
- **Security:**
  - **Face recognition**
  - **Signature / fingerprint / iris verification**
  - **DNA fingerprinting**

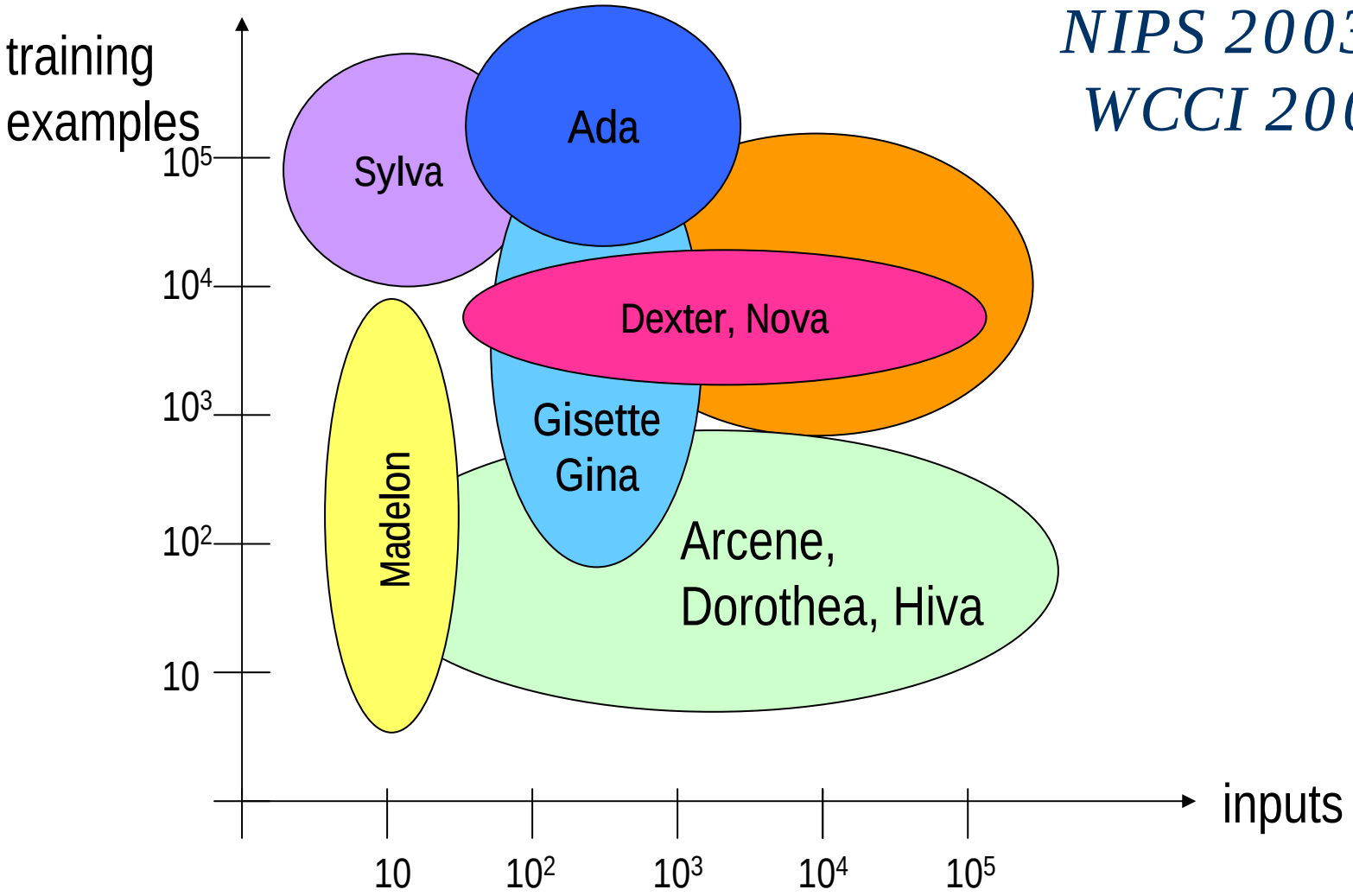
Computer / Internet



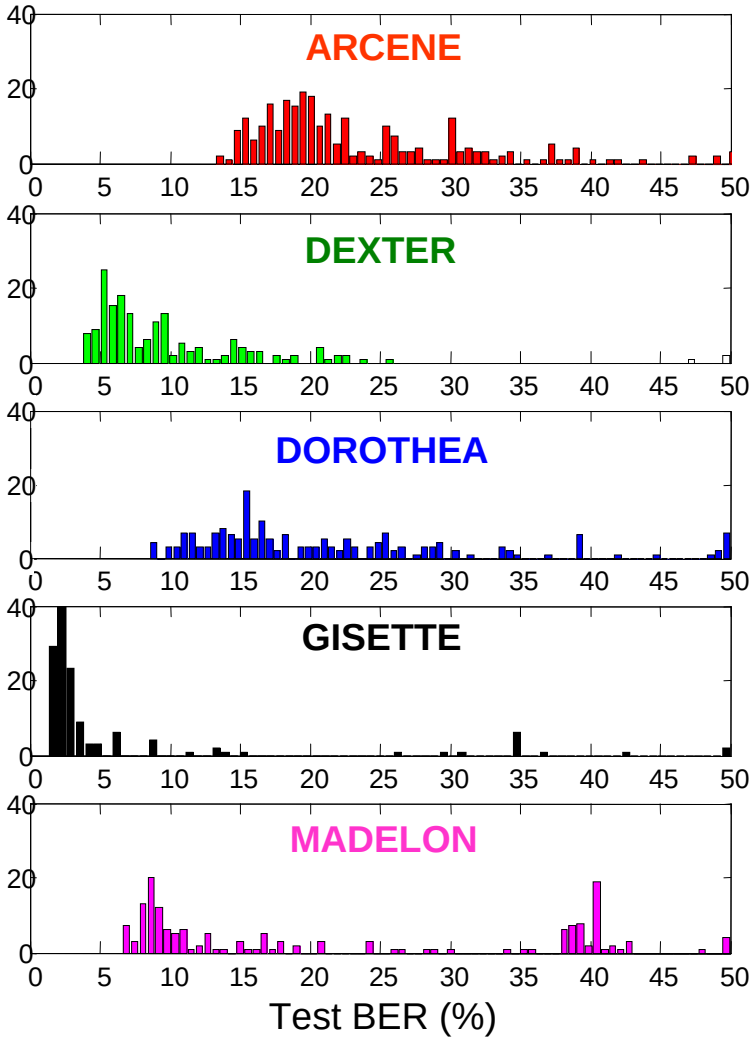
- **Computer interfaces:**
  - **Troubleshooting wizards**
  - **Handwriting and speech**
  - **Brain waves**
- **Internet**
  - **Hit ranking**
  - **Spam filtering**
  - **Text categorization**
  - **Text translation**
  - **Recommendation**

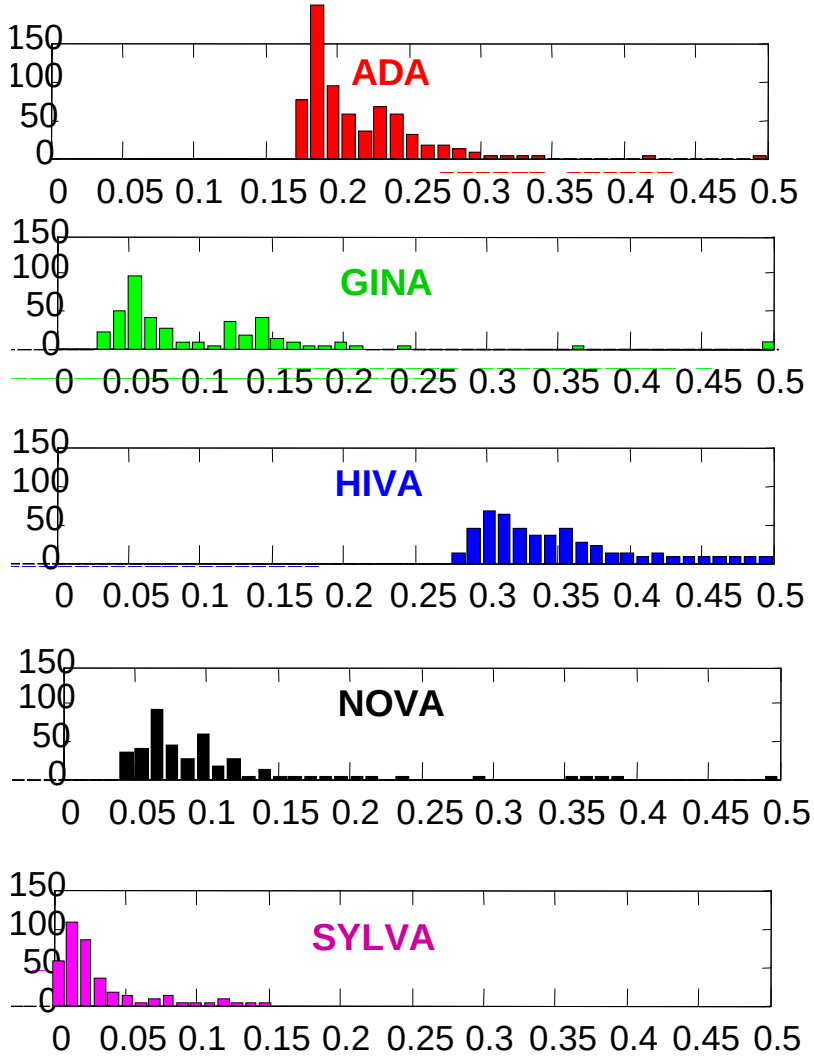
Challenges

NIPS 2003 & WCCI 2006

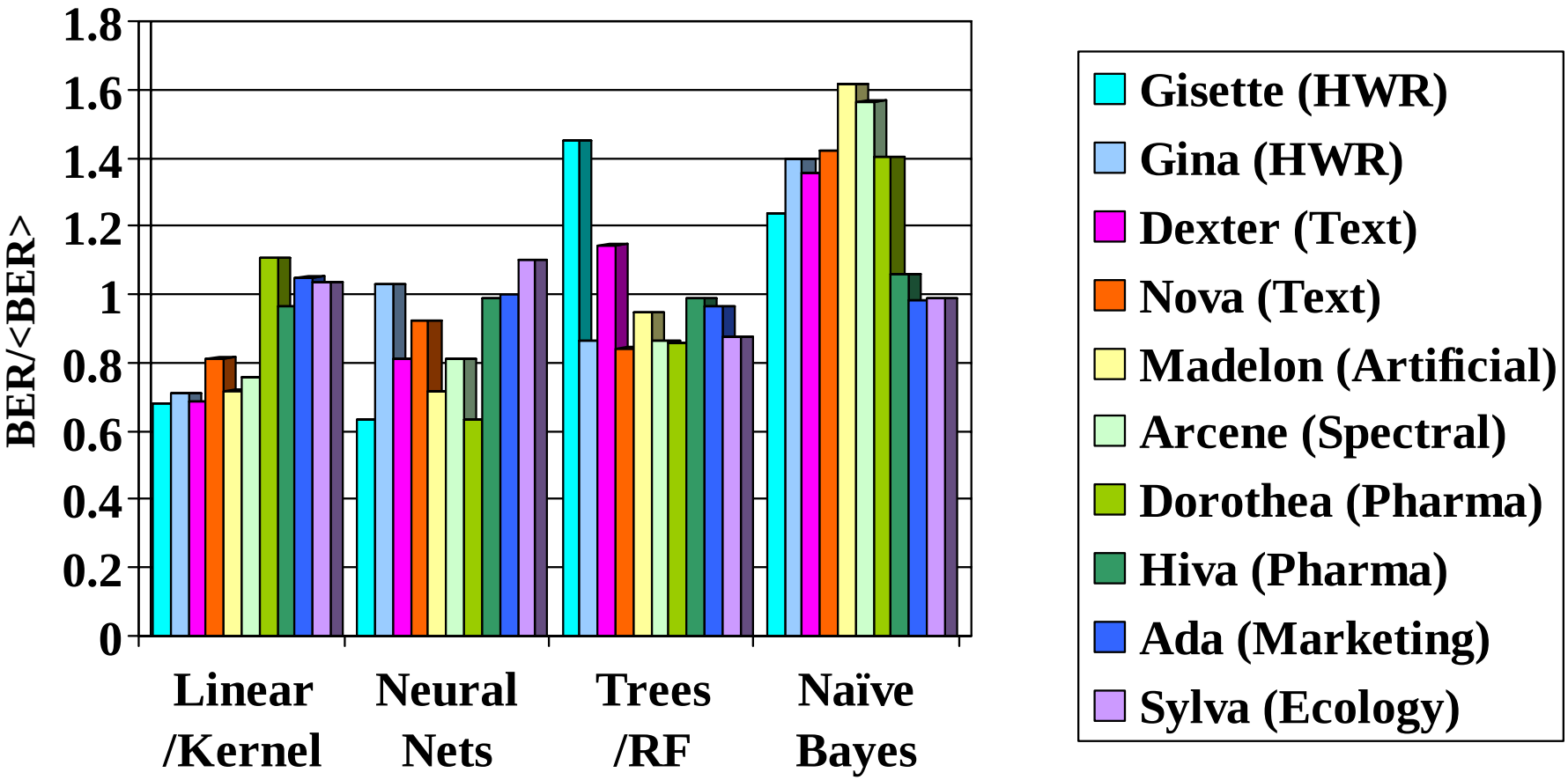


Ten Classification Tasks

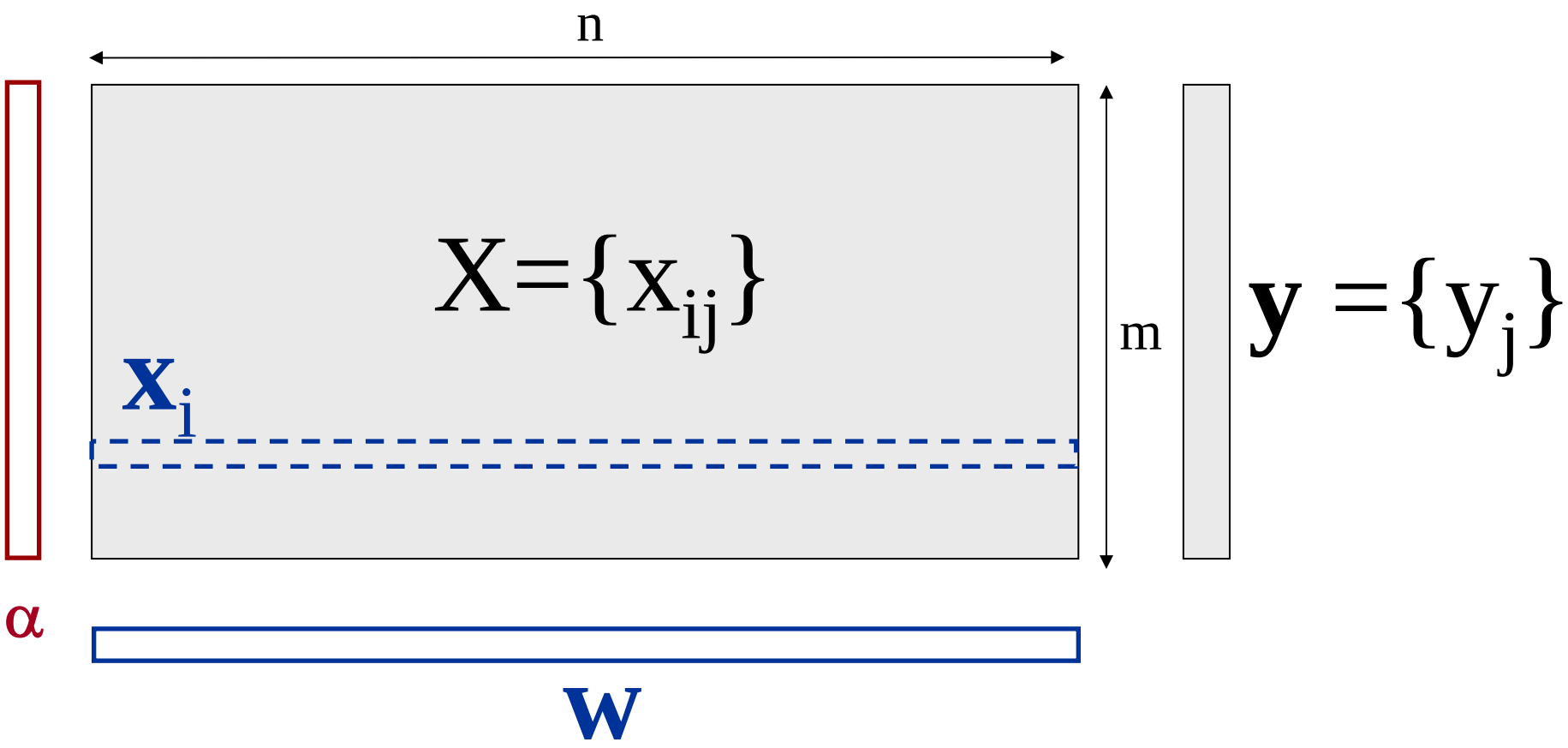




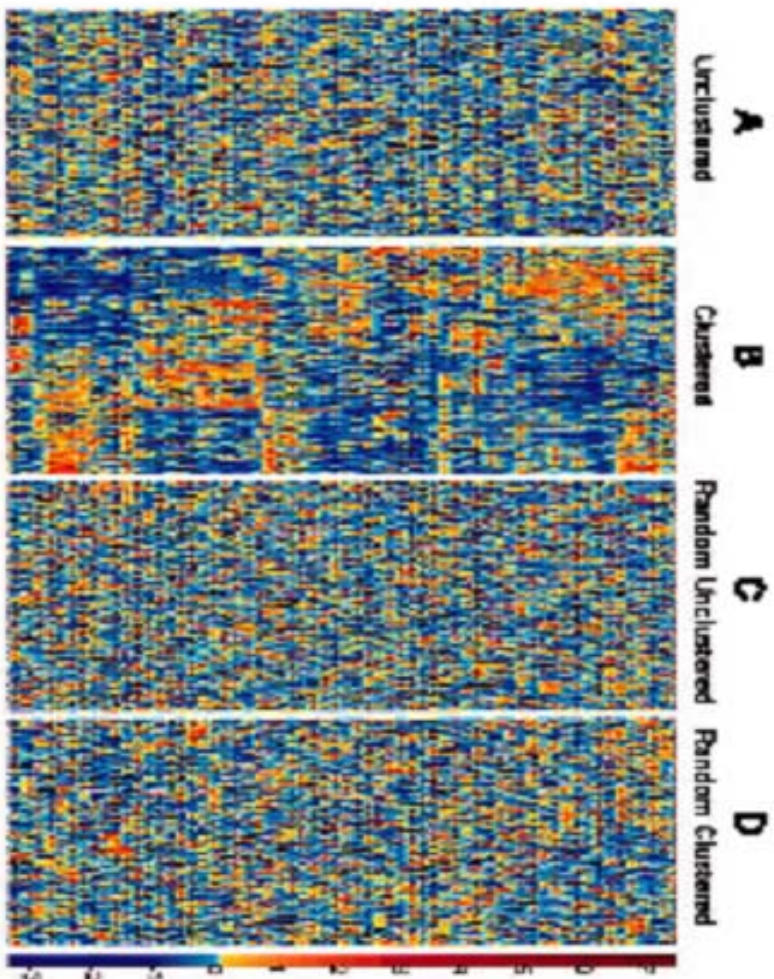
Challenge Winning Methods



Conventions



Learning problem



Colon cancer, Alon et al 1999

Data matrix: X

m lines = patterns (data points, examples): samples, patients, documents, images, …

n columns = features: (attributes, input variables): genes, proteins, words, pixels, …

Unsupervised learning

Is there structure in data?

Supervised learning

Predict an outcome **y**.

Linear Models

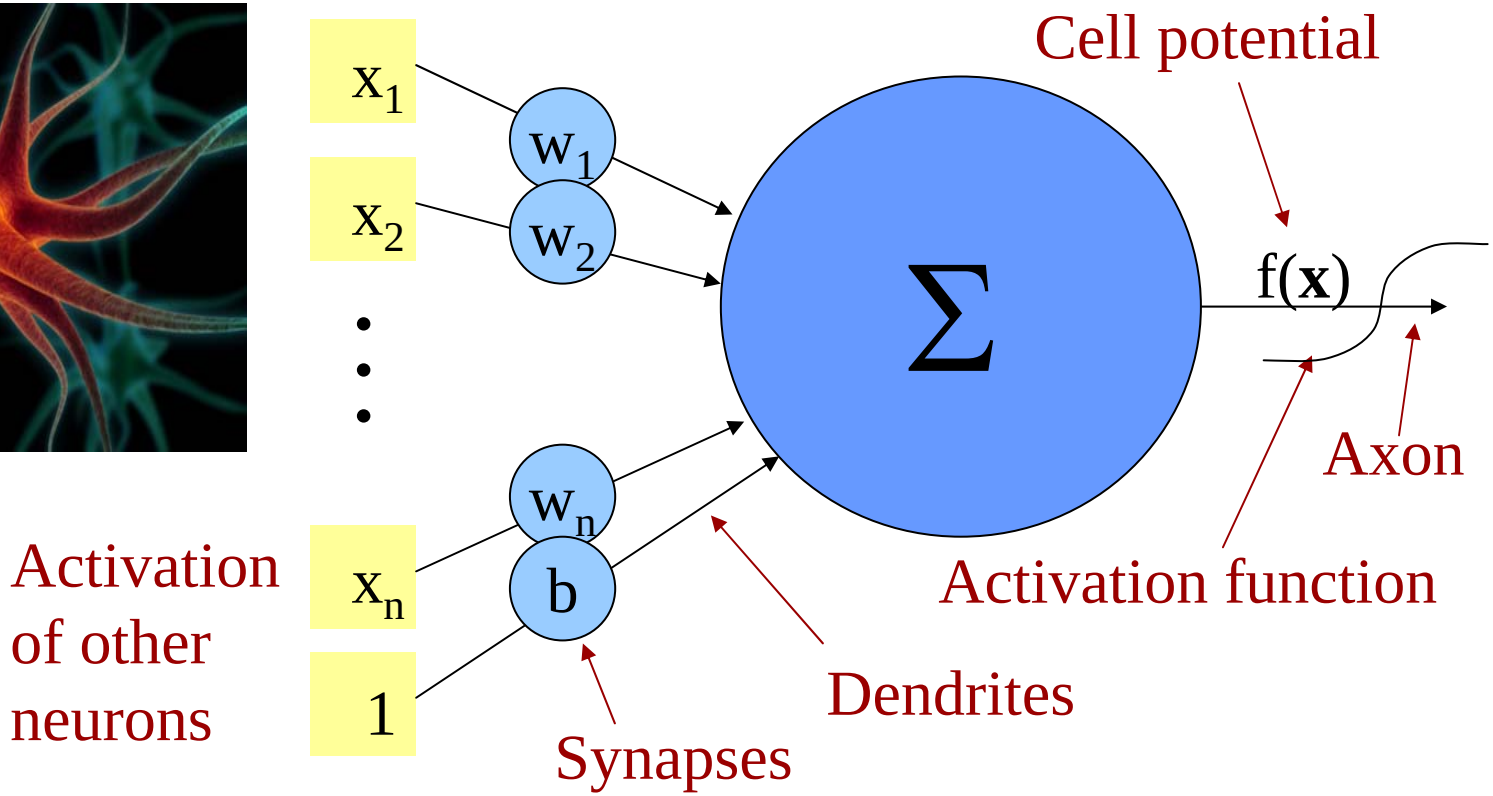
- $f(\mathbf{x}) = \mathbf{w} \bullet \mathbf{x} + b = \sum_{j=1:n} w_j x_j + b$

  Linearity in the parameters, NOT in the input components.

- $f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x}) + b = \sum_j w_j \phi_j(\mathbf{x}) + b$ *(Perceptron)*

- $f(\mathbf{x}) = \sum_{i=1:m} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b$ *(Kernel method)*

Artificial Neurons

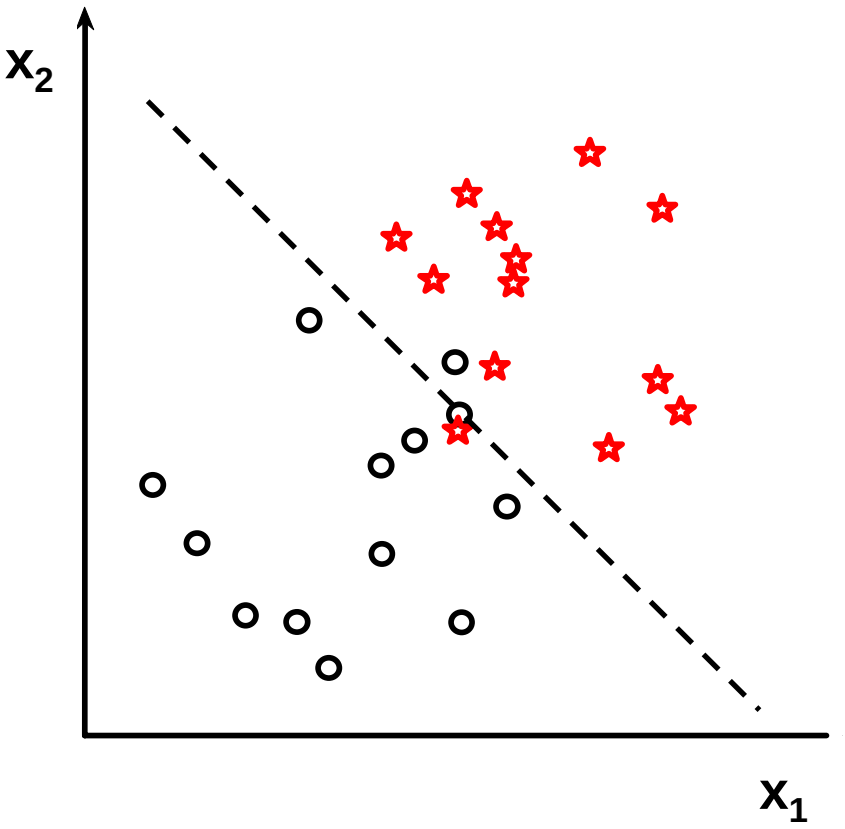


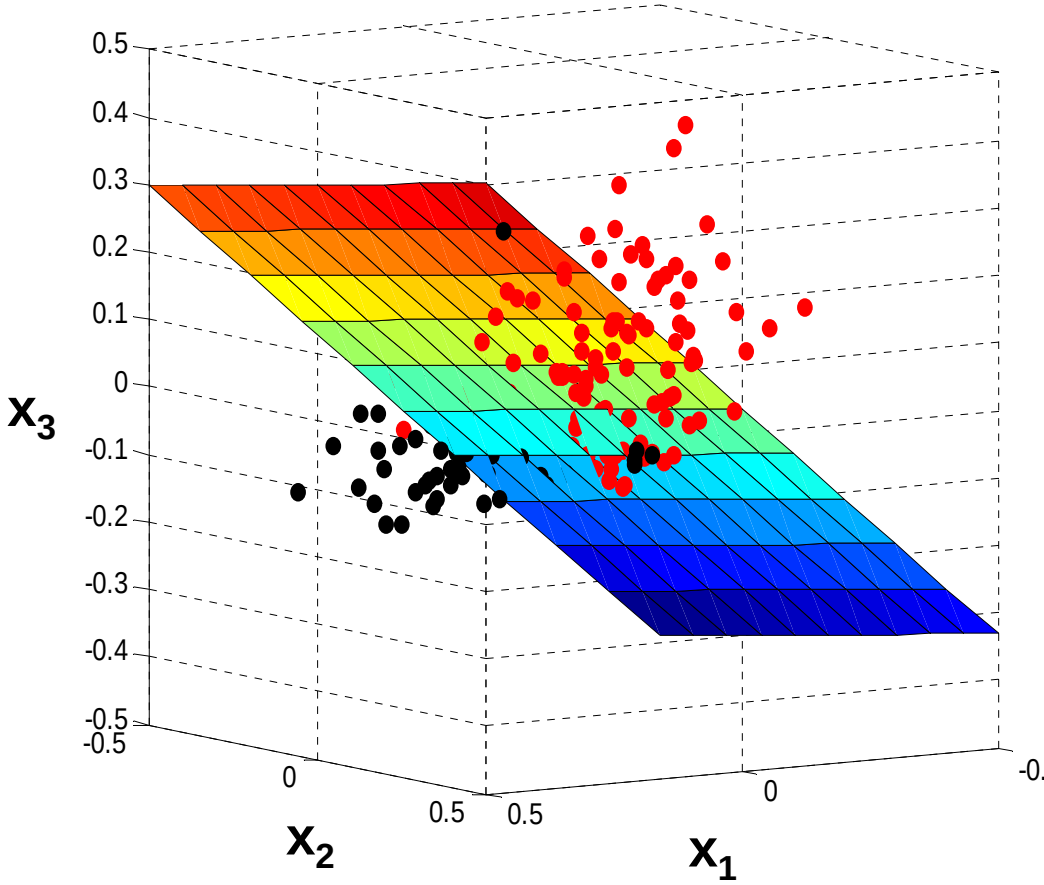
McCulloch and Pitts, 1943

$$f(\mathbf{x}) = \mathbf{w} \bullet \mathbf{x} + b$$

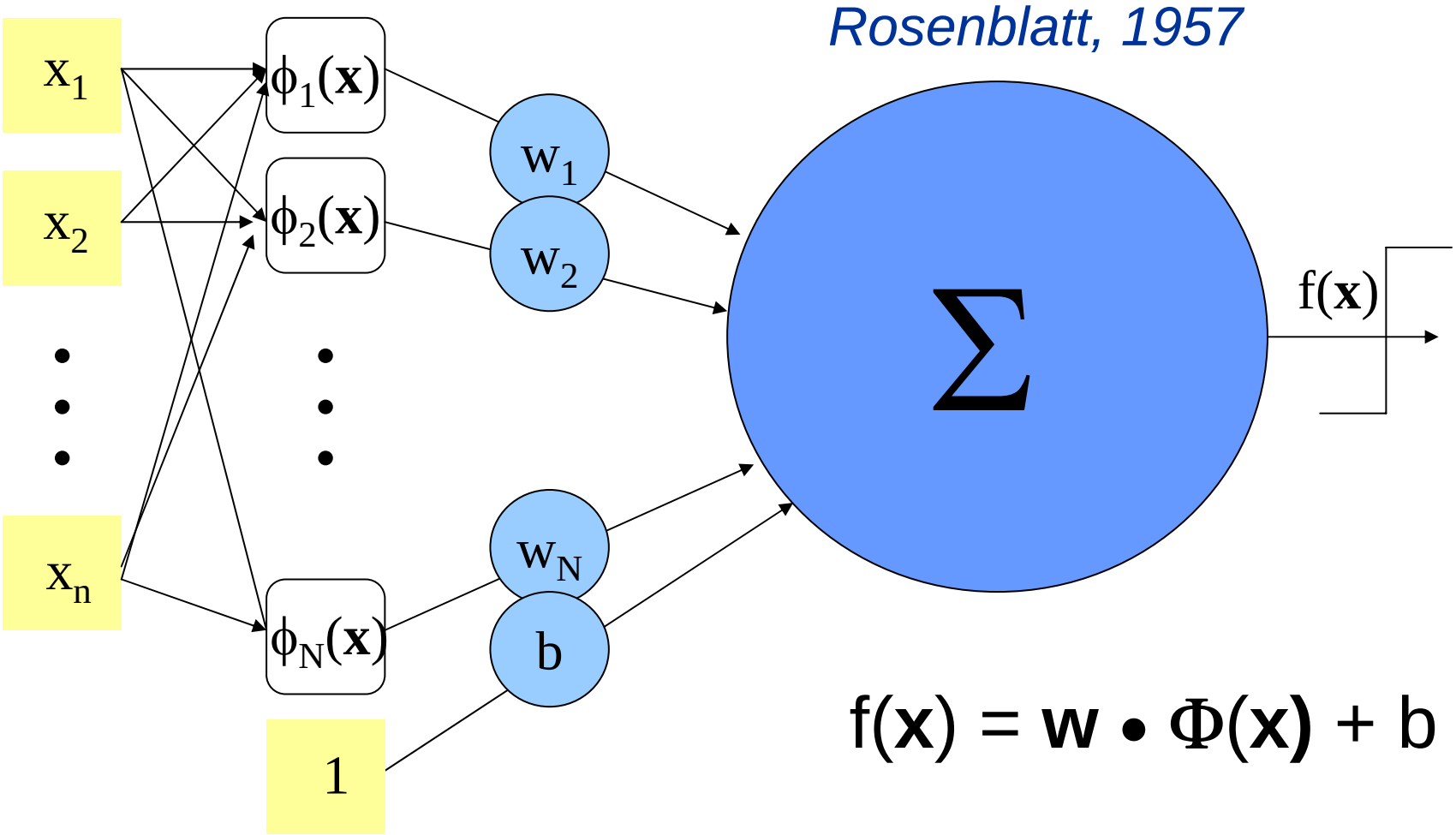
Linear Decision Boundary

hyperplane

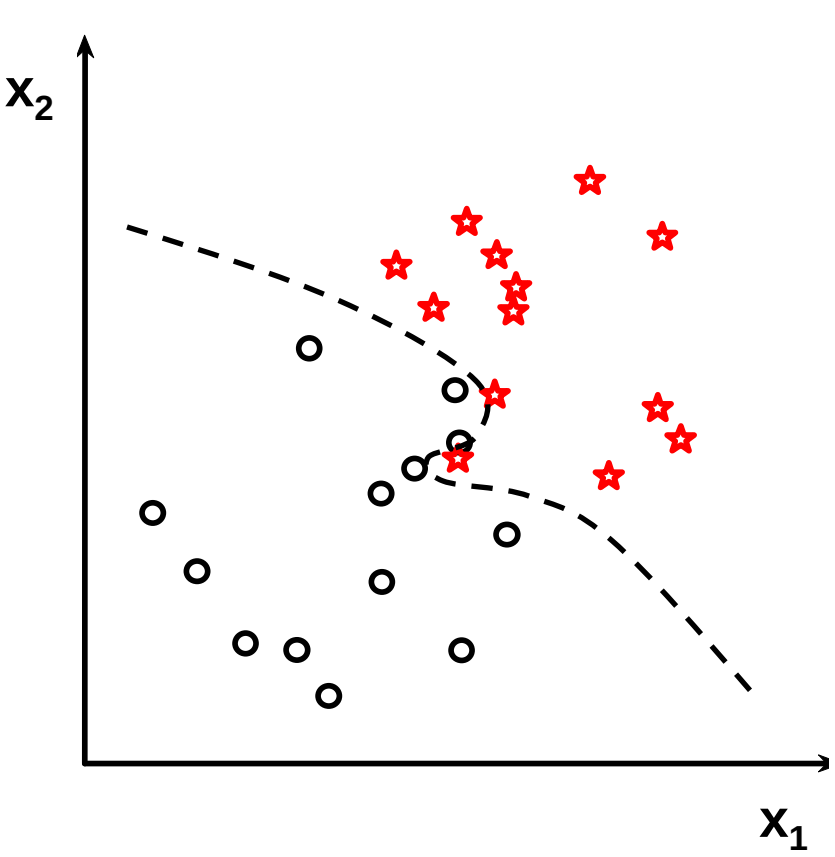


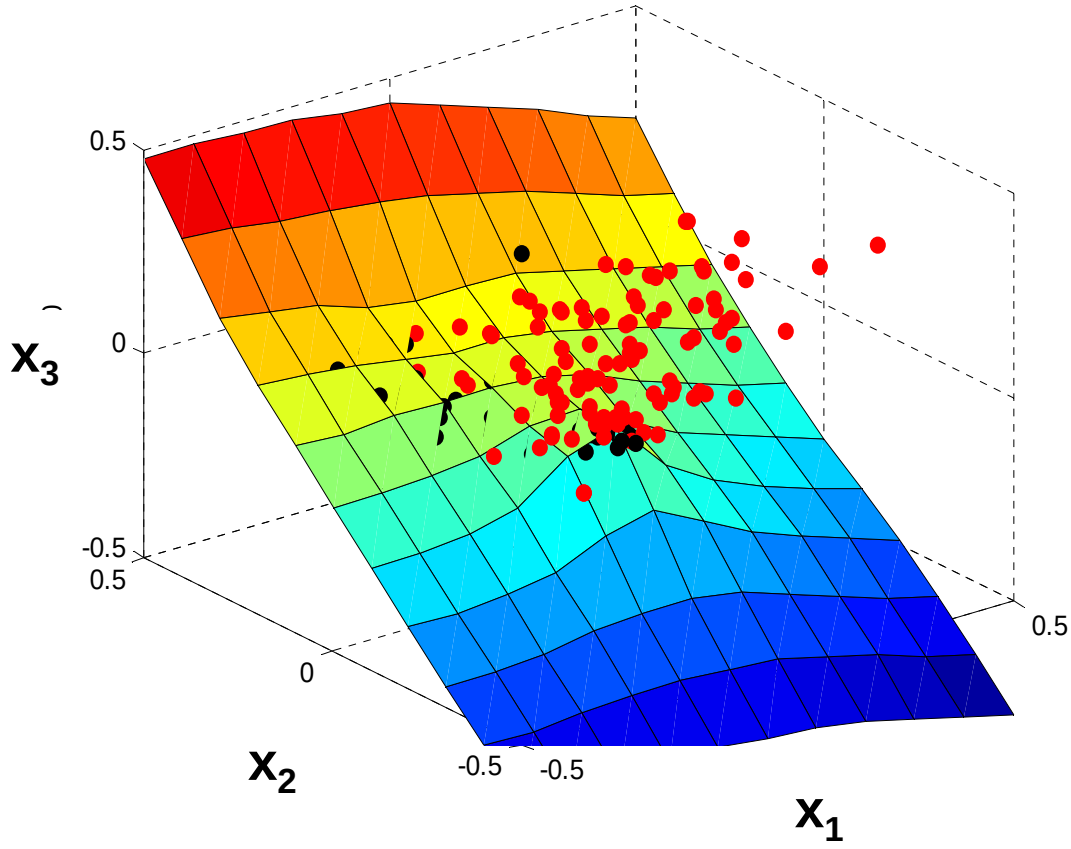


Perceptron



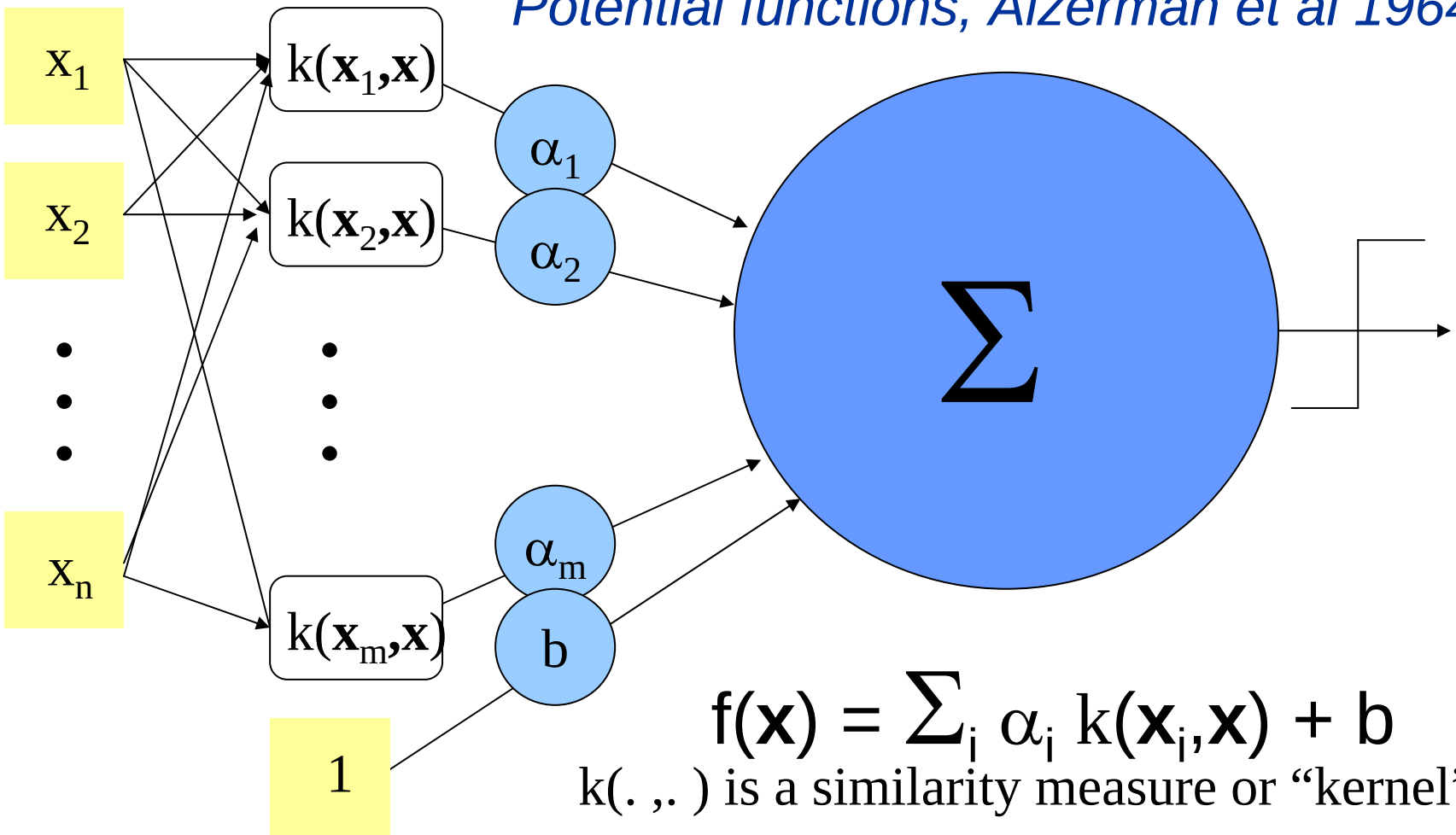
NL Decision Boundary





Kernel Method

Potential functions, Aizerman et al 1964

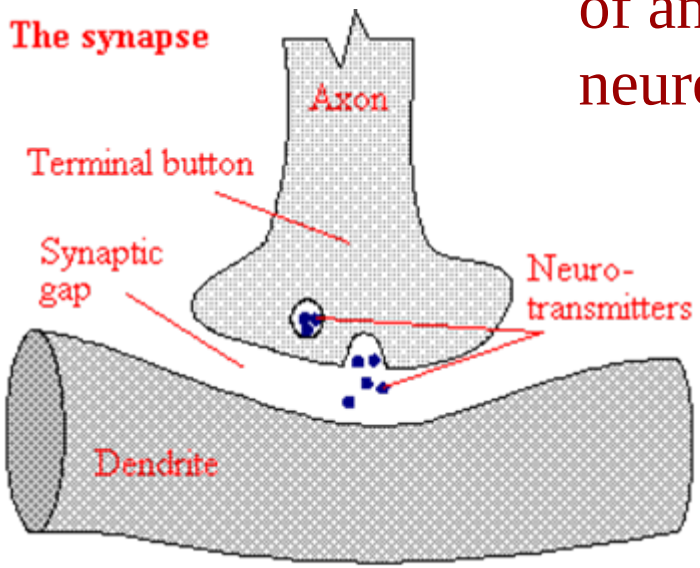


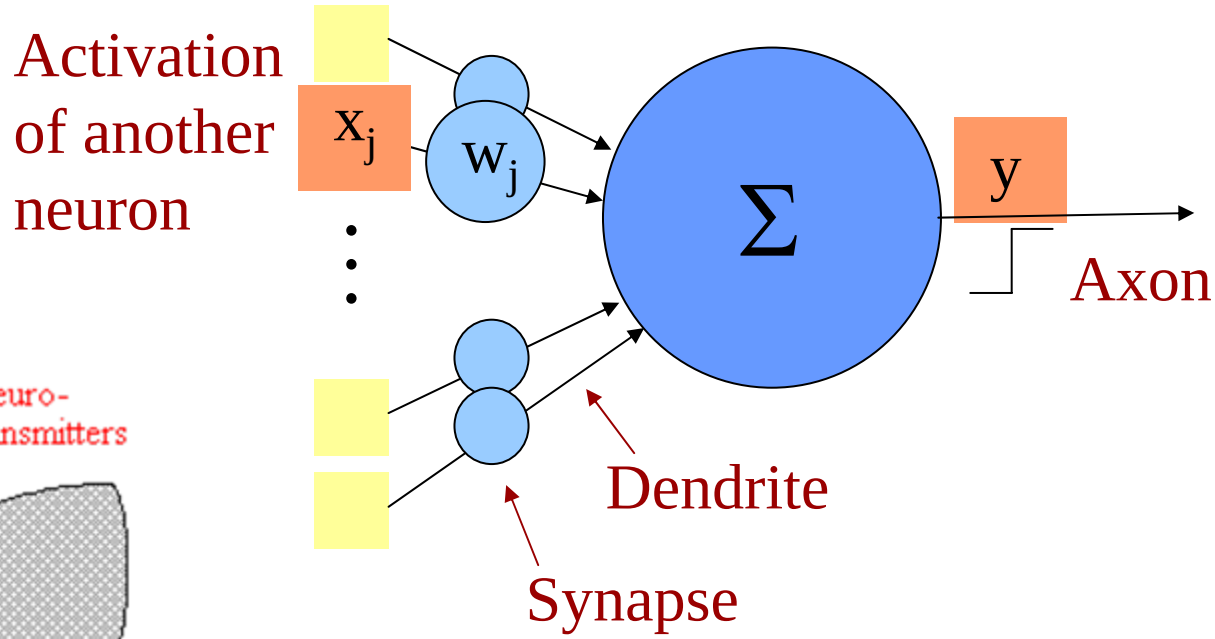
$$f(\mathbf{x}) = \sum_i \alpha_i \, k(\mathbf{x}_i,\mathbf{x}) + b$$

k(. ,.) is a similarity measure or “kernel”.

Hebb's Rule

$$w_j \leftarrow w_j + y_i x_{ij}$$





Link to "Naïve Bayes"

Kernel "Trick" (for Hebb's rule)

- Hebb's rule for the Perceptron:

$$\mathbf{w} = \sum_i y_i \Phi(\mathbf{x}_i)$$

$$f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x}) = \sum_i y_i \Phi(\mathbf{x}_i) \bullet \Phi(\mathbf{x})$$

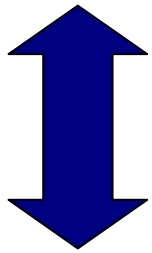
- Define a dot product:

$$k(\mathbf{x}_i,\mathbf{x}) = \Phi(\mathbf{x}_i) \bullet \Phi(\mathbf{x})$$

$$f(\mathbf{x}) = \sum_i y_i k(\mathbf{x}_i,\mathbf{x})$$

Kernel "Trick" (general)

- $f(\mathbf{x}) = \sum_i \alpha_i \, k(\mathbf{x}_i, \mathbf{x})$
- $k(\mathbf{x}_i, \mathbf{x}) = \Phi(\mathbf{x}_i) \bullet \Phi(\mathbf{x})$



Dual forms

- $f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x})$
- $\mathbf{w} = \sum_i \alpha_i \, \Phi(\mathbf{x}_i)$

What is a Kernel?

A kernel is:

- a **similarity measure**
- a **dot product** in *some* feature space: $k(\mathbf{s}, \mathbf{t}) = \Phi(\mathbf{s}) \bullet \Phi(\mathbf{t})$

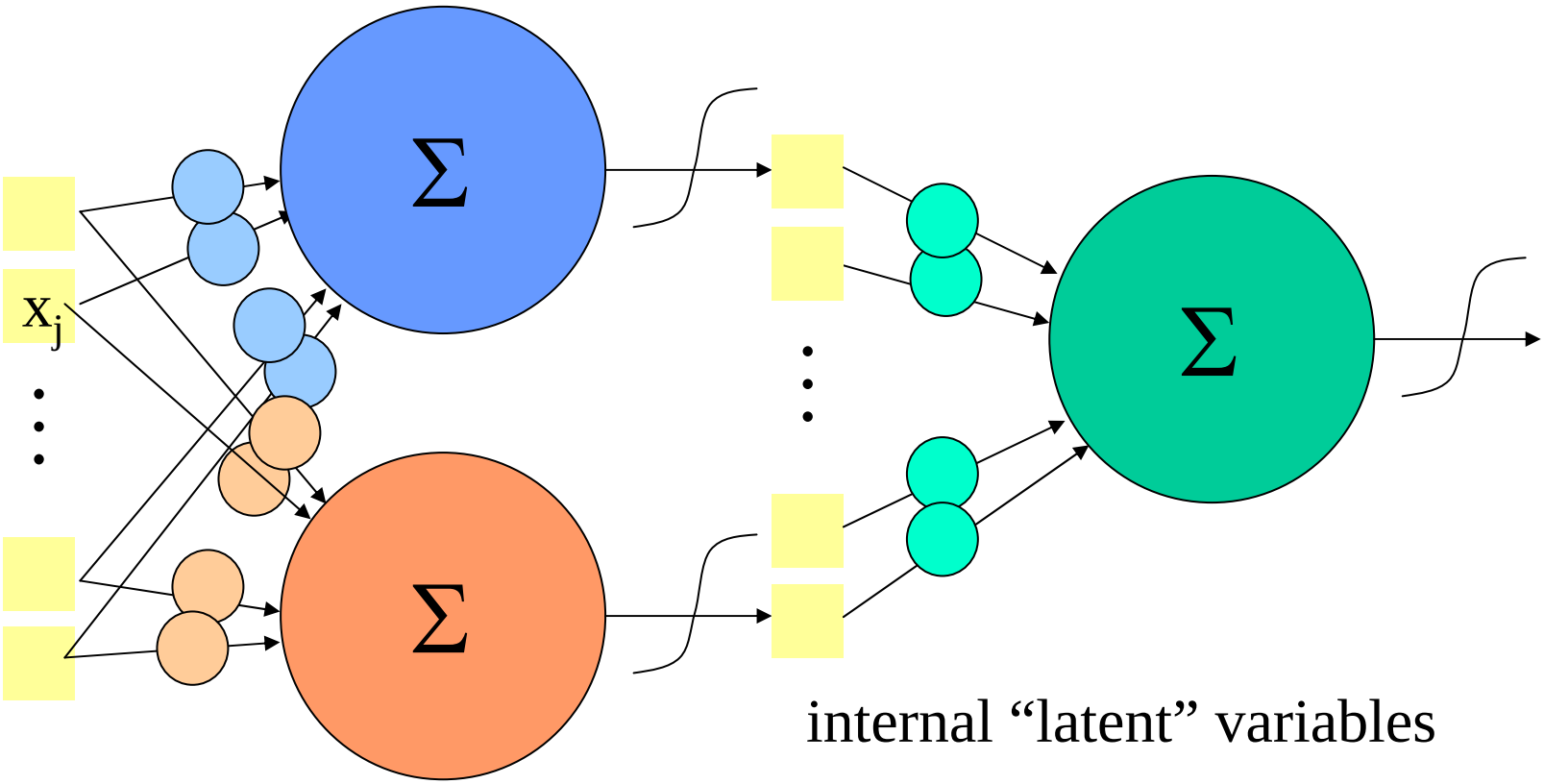
But we do not need to know the Φ representation.

Examples:

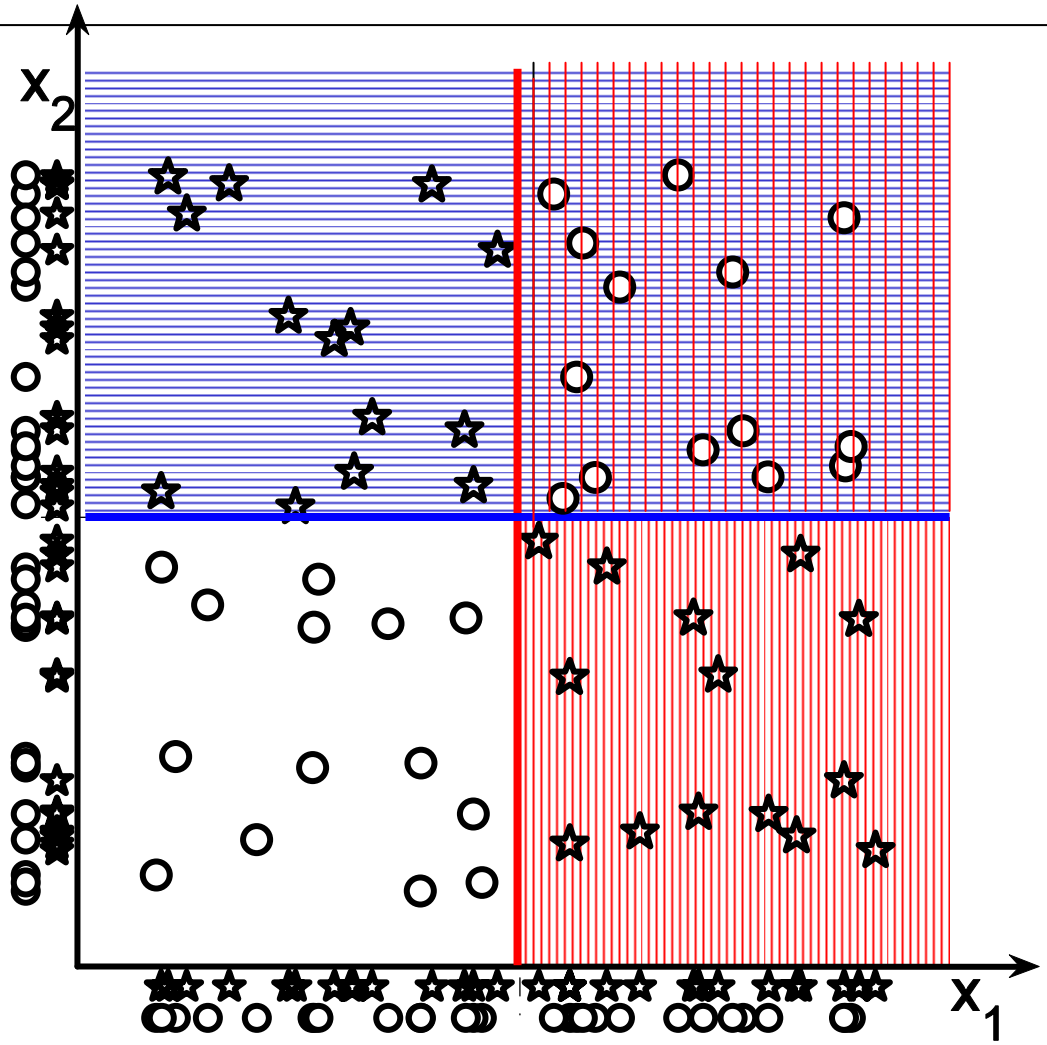
- $k(\mathbf{s}, \mathbf{t}) = \exp(-||\mathbf{s}-\mathbf{t}||^2/\sigma^2)$ *Gaussian kernel*
- $k(\mathbf{s}, \mathbf{t}) = (\mathbf{s} \bullet \mathbf{t})^q$ *Polynomial kernel*

Multi-Layer Perceptron

Back-propagation, Rumelhart et al, 1986

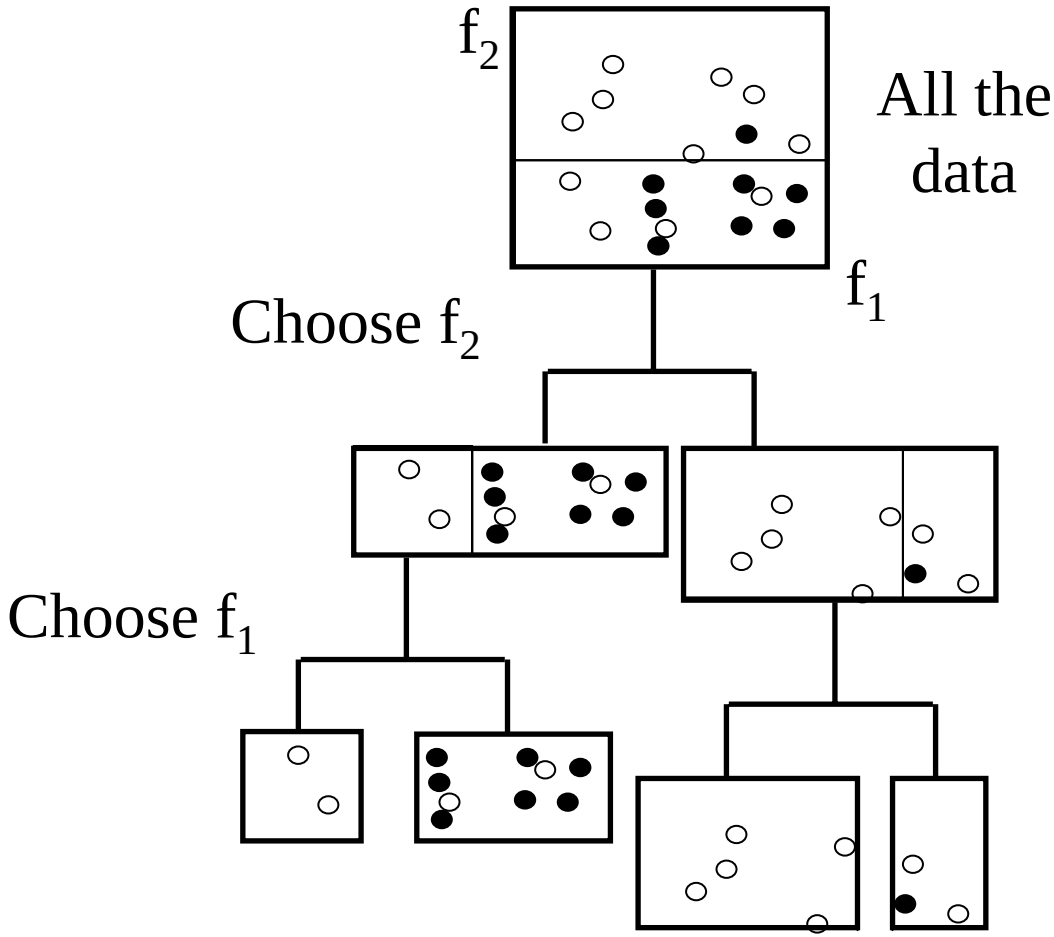


Chessboard Problem



Tree Classifiers

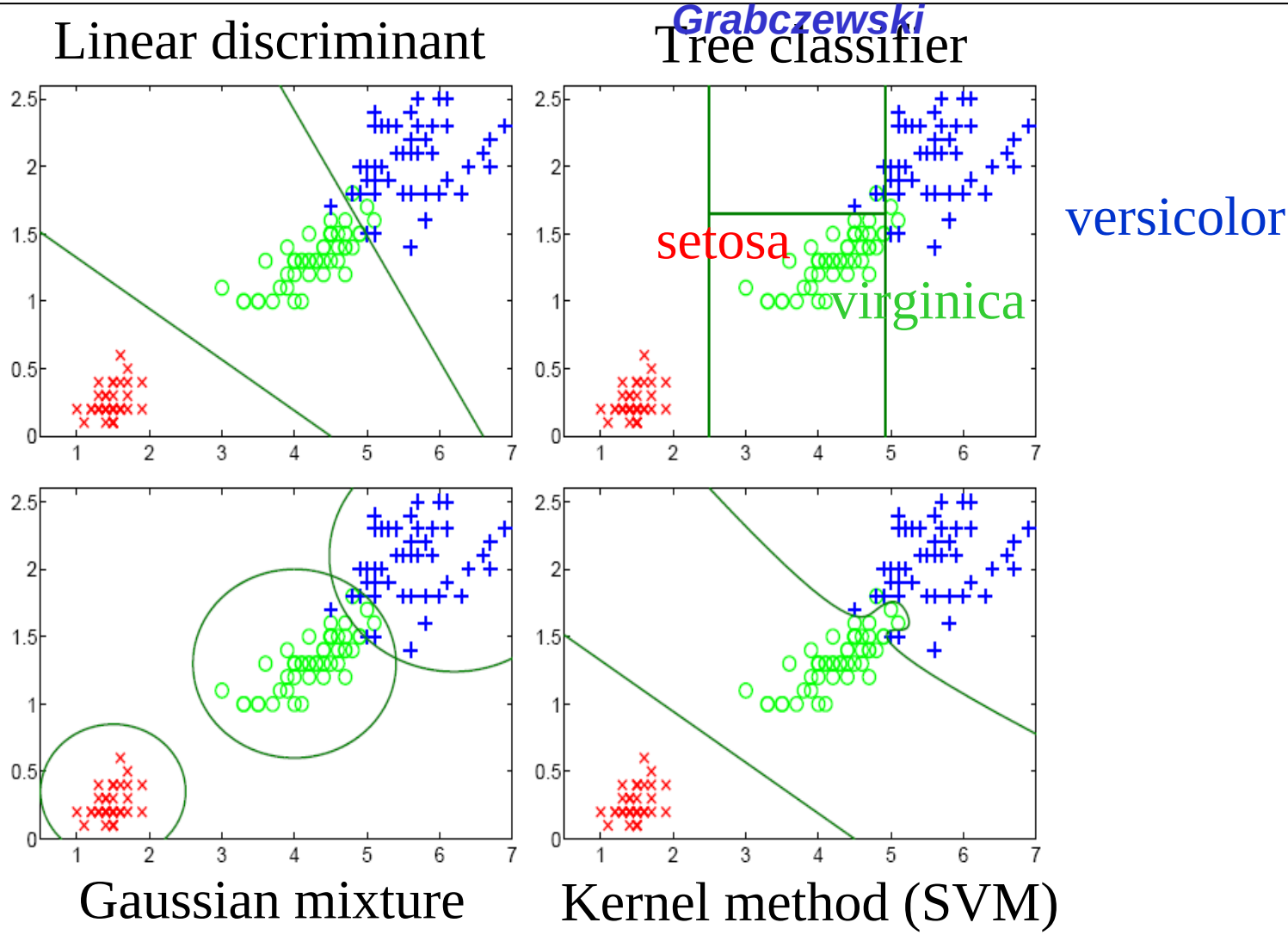
CART (Breiman, 1984) or *C4.5 (Quinlan, 1993)*



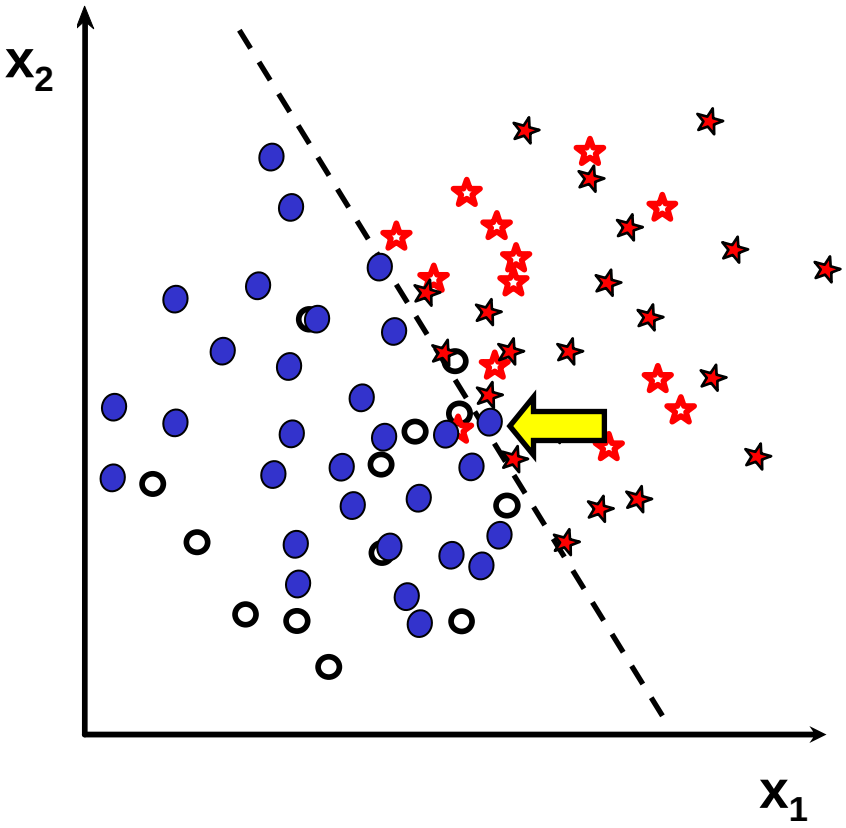
At each step, choose the feature that "reduces entropy" most. Work towards "node purity".

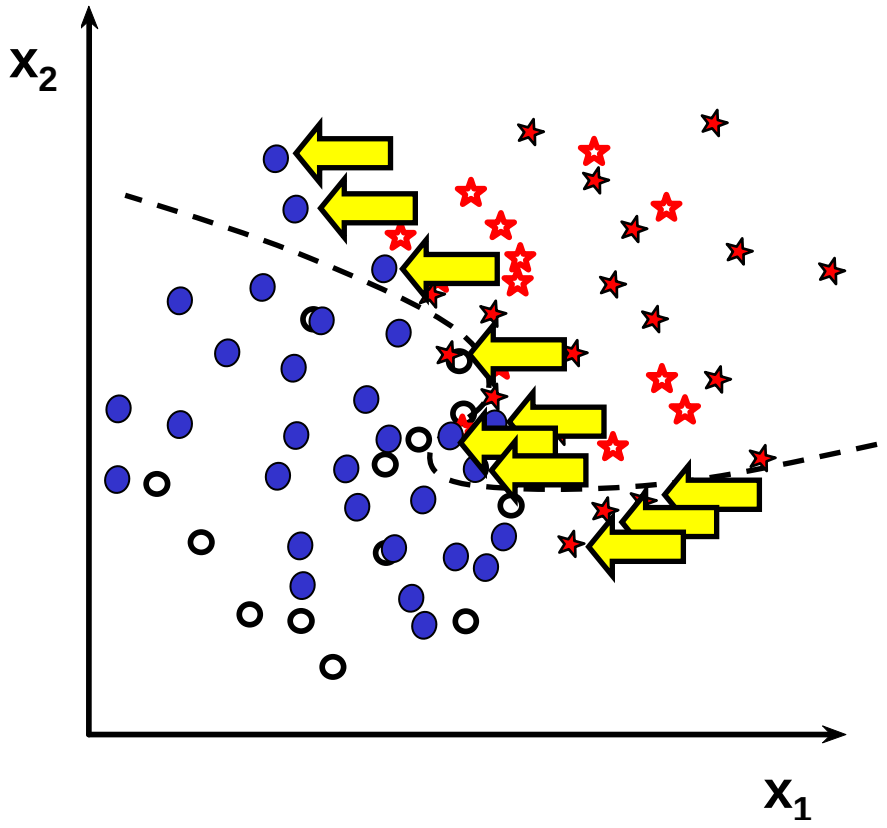
Iris Data (Fisher, 1936)

Figure from Norbert Jankowski and Krzysztof Grabczewski

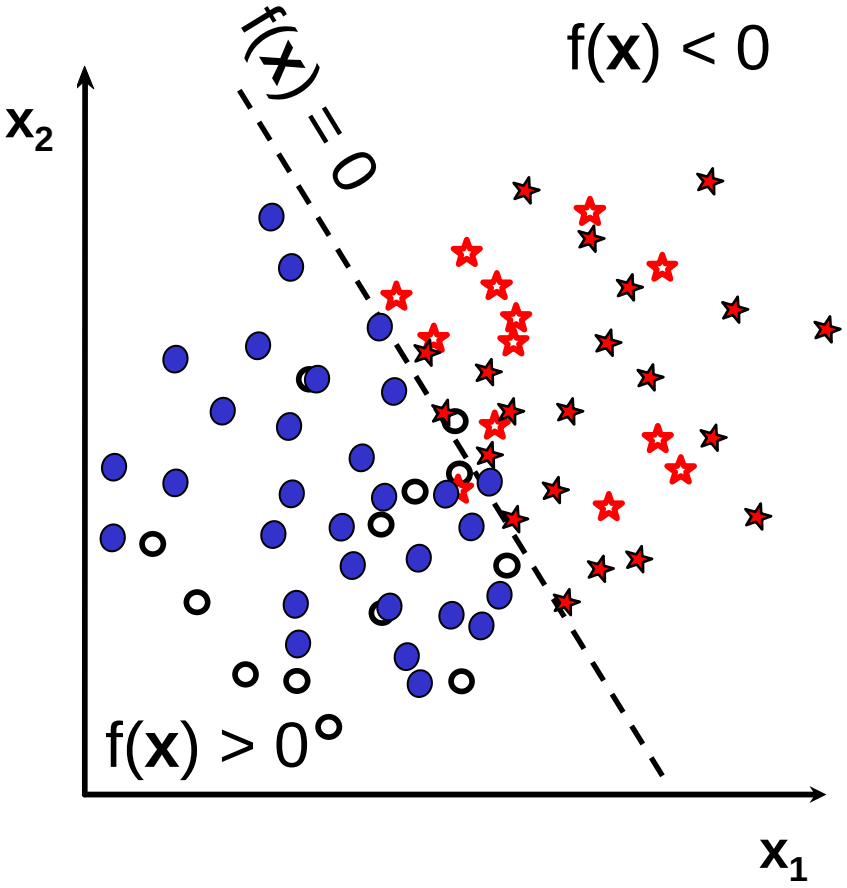


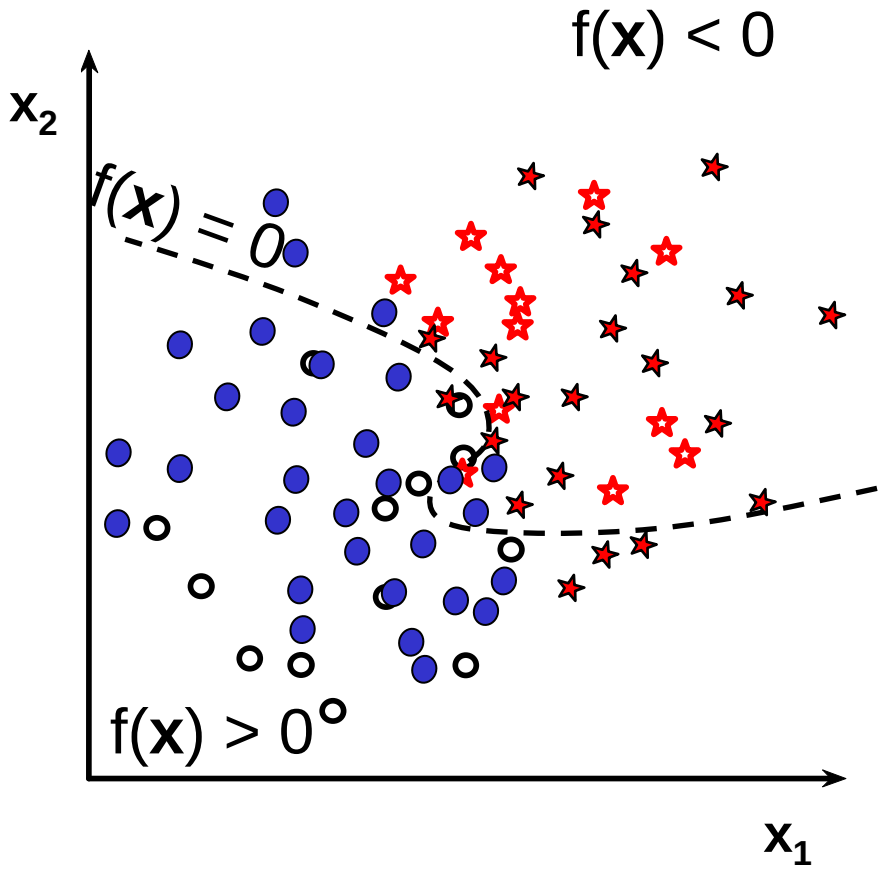
Fit / Robustness Tradeoff



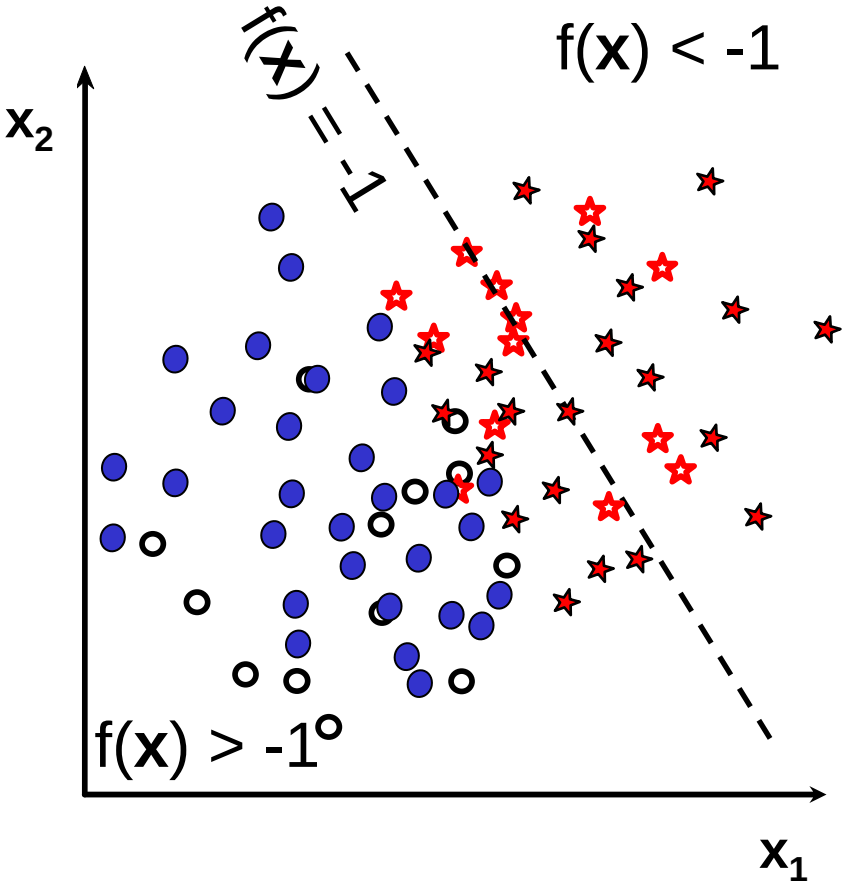


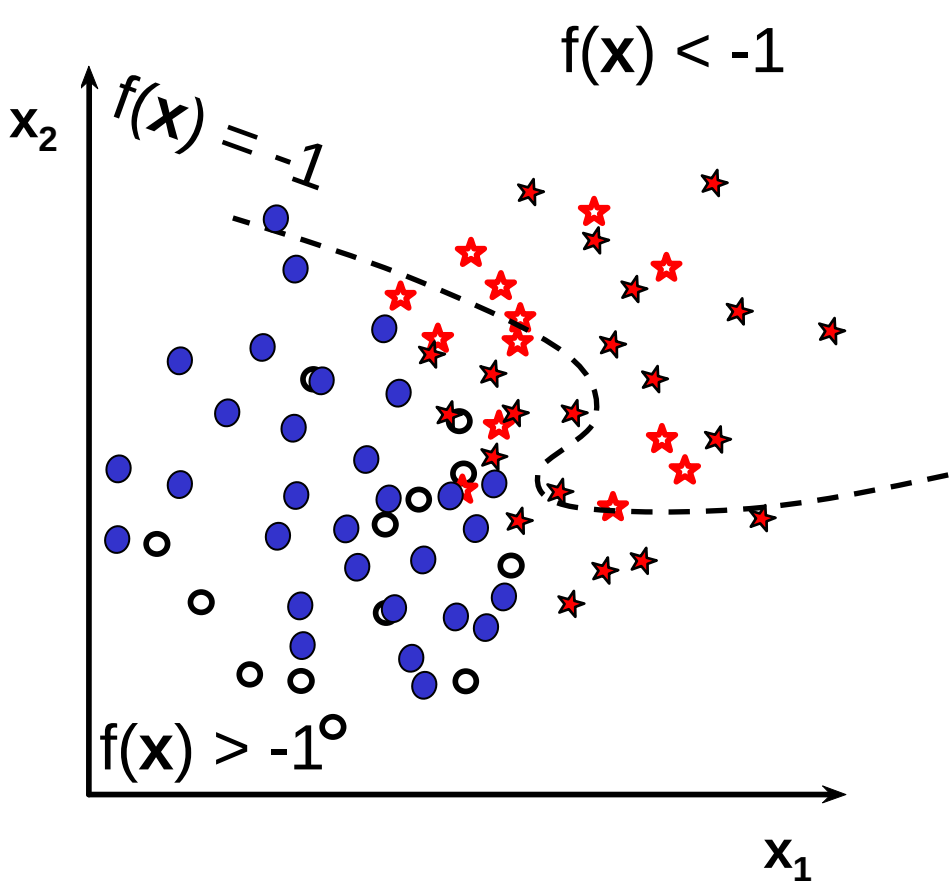
Performance evaluation



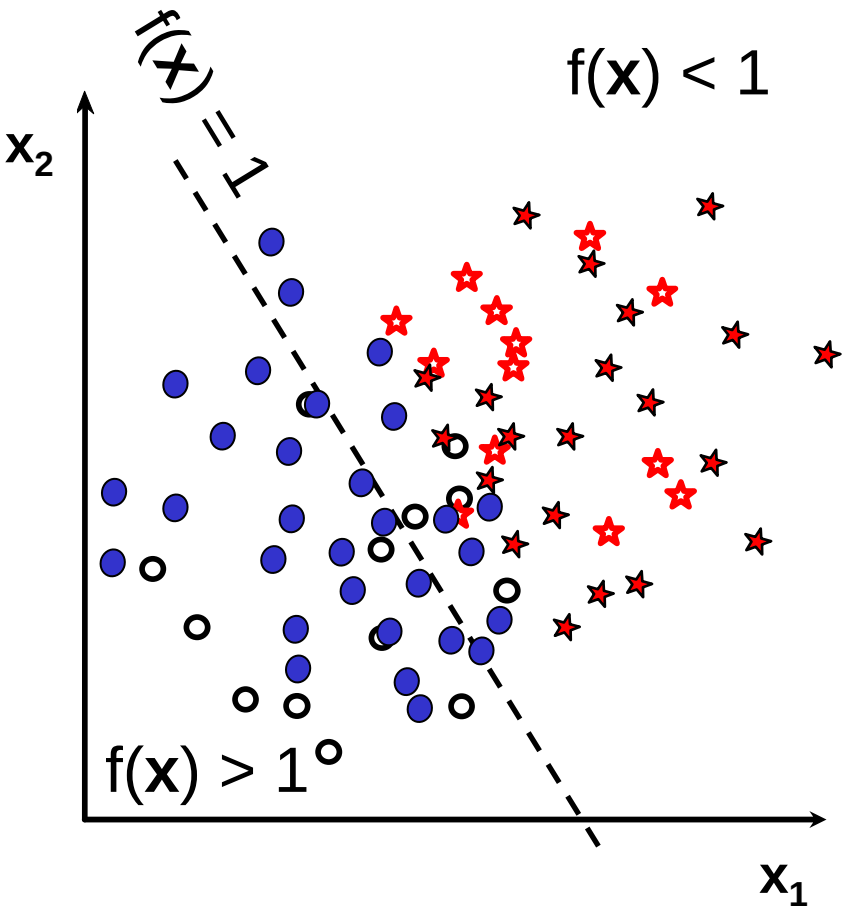


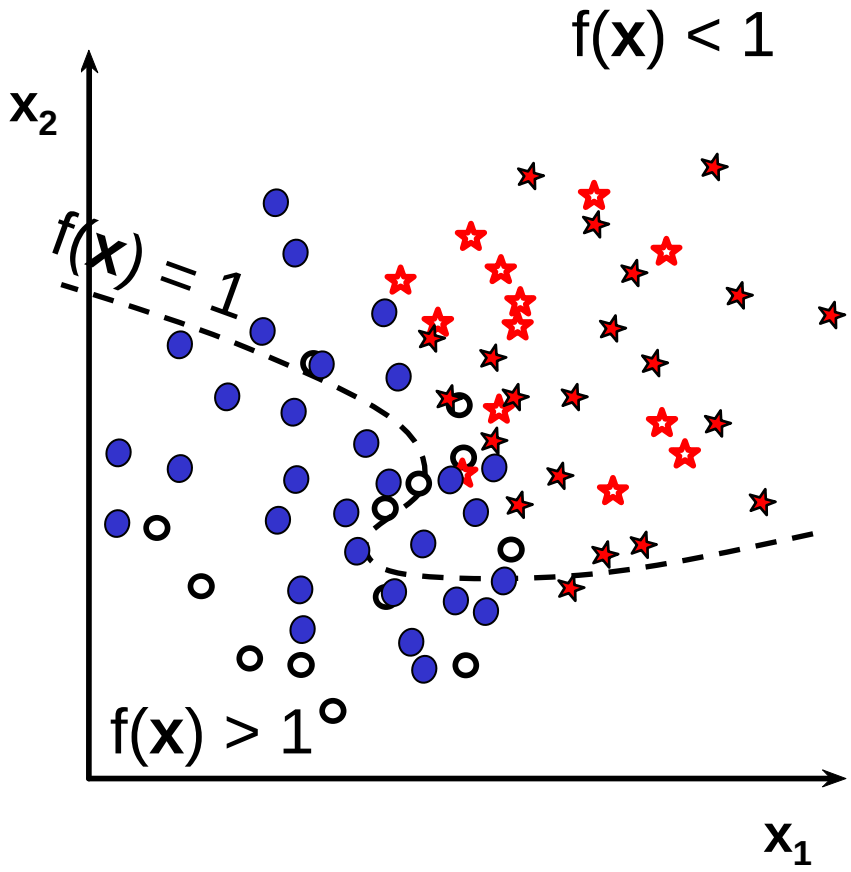
Performance evaluation





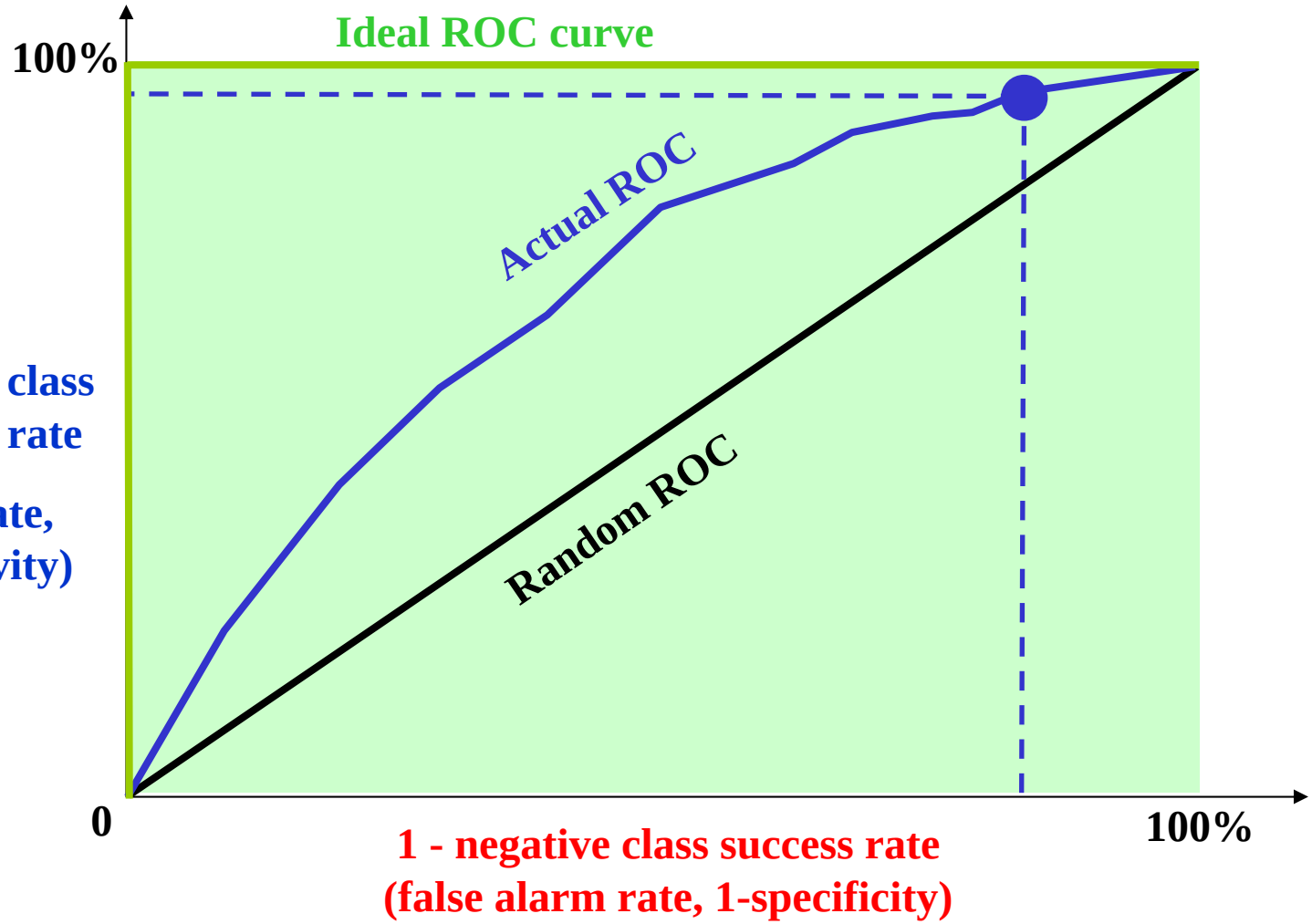
Performance evaluation





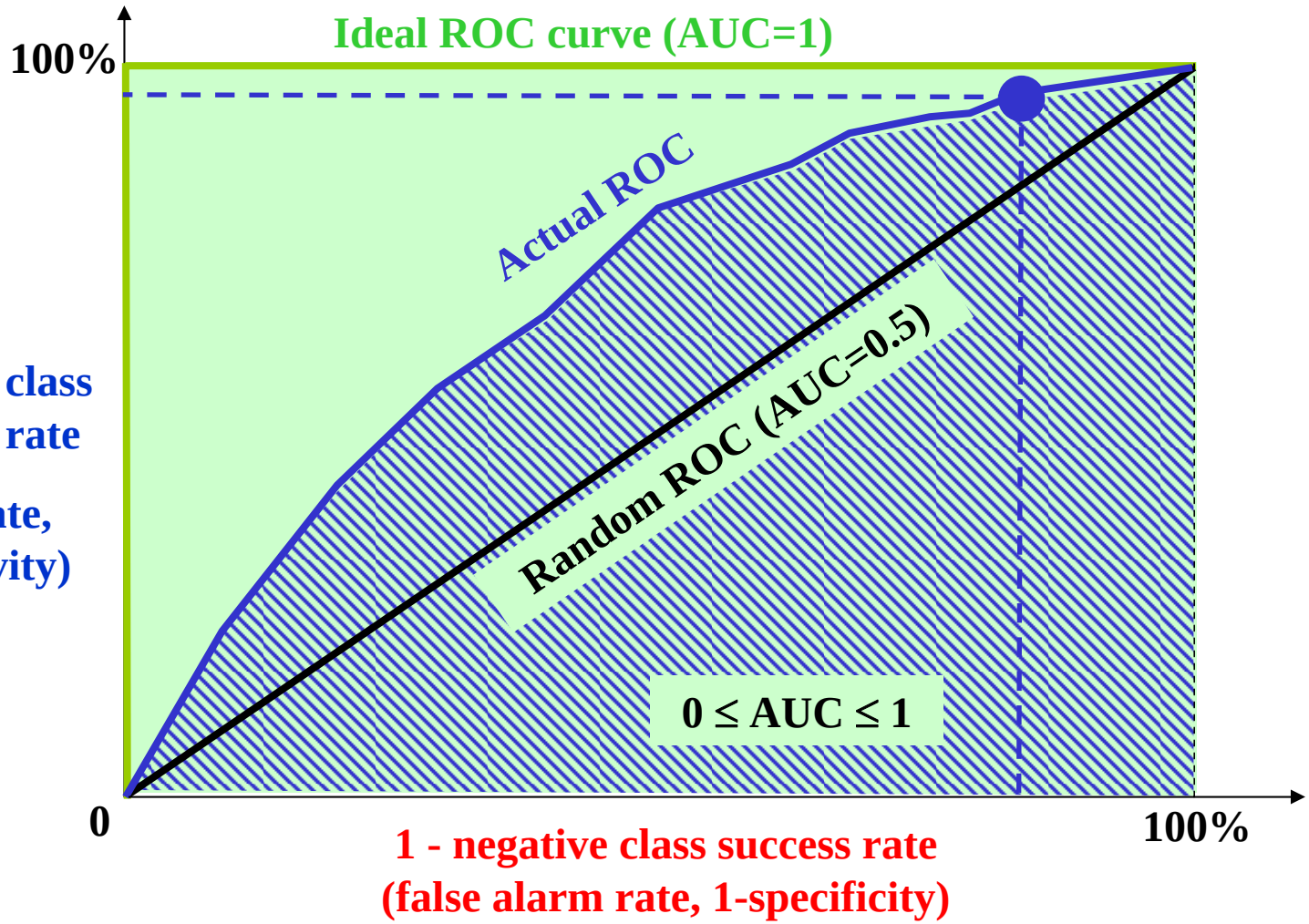
ROC Curve

For a given threshold on f(x), you get a point on the ROC curve.



ROC Curve

For a given threshold on f(x), you get a point on the ROC curve.



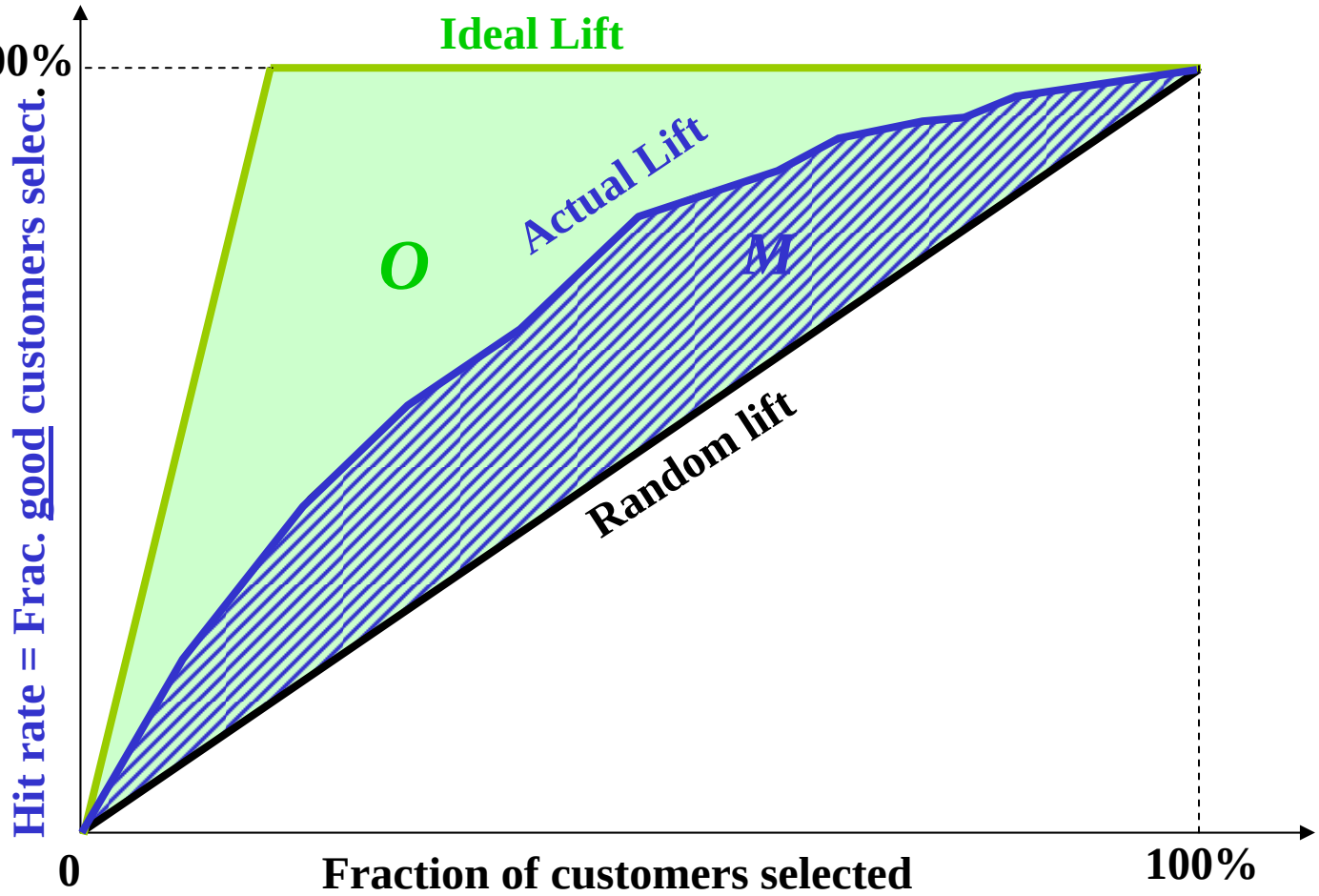
Lift Curve

Customers ranked according to f(x); selection of the top ranking customers.

$$Gini = M/O$$

Gini=2 AUC-1

0 ≤ Gini ≤ 1



Performance Assessment

Cost matrix

		Predictions: F(x)			
		Class -1	Class +1	Total	Class +1 / Total
Truth: y	Class -1	tn	fp	neg=tn+fp	False alarm = fp/neg
	Class +1	fn	tp	pos=fn+tp	Hit rate = tp/pos
	Total	rej=tn+fn	sel=fp+tp	m=tn+fp+fn+tp	Frac. selected = sel/m
	Class+1 /Total		Precision = tp/sel	False alarm rate = type I errate = 1-specificity Hit rate = 1-type II errate = sensitivity = recall = test power	

Compare F(x) = sign(f(x)) to the target y, and report:

- **Error rate = (fn + fp)/m**
- **{Hit rate , False alarm rate} or {Hit rate , Precision} or {Hit rate , Frac.selected}**
- **Balanced error rate (BER) = (fn/pos + fp/neg)/2 = 1 – (sensitivity+specificity)/2**
- **F measure = 2 precision.recall/(precision+recall)**

Vary the decision threshold θ in F(x) = sign(f(x)+θ), and plot:

- **ROC curve: Hit rate *vs.* False alarm rate**
- **Lift curve: Hit rate *vs.* Fraction selected**
- **Precision/recall curve: Hit rate *vs.* Precision**

What is a Risk Functional?

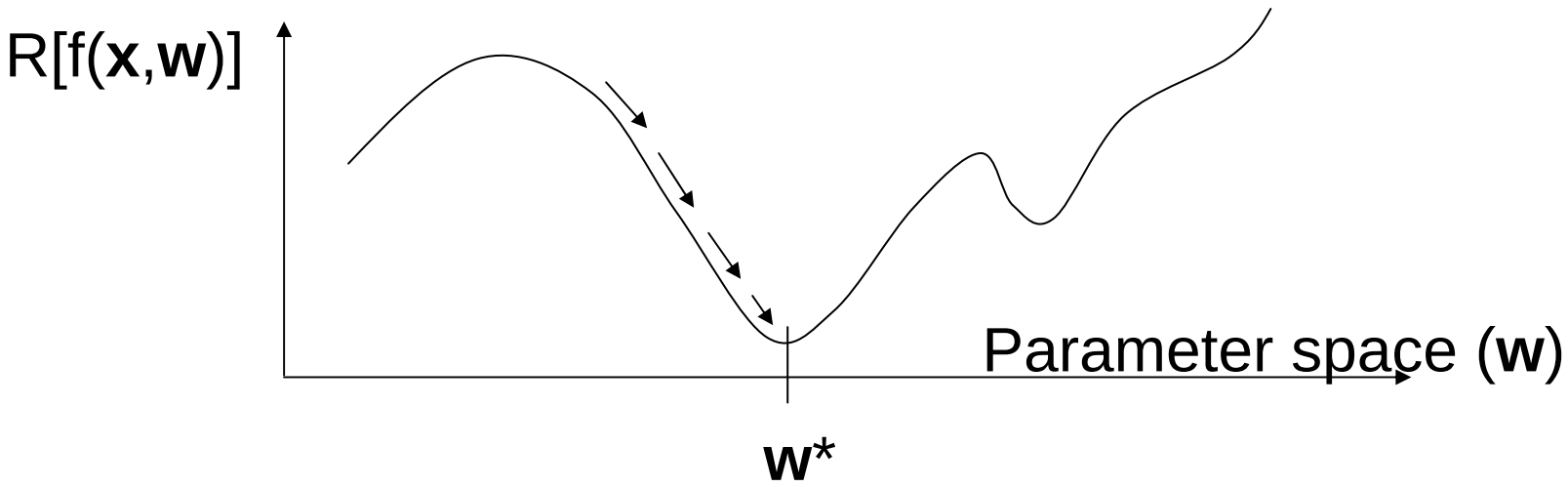
A function of the parameters of the learning machine, assessing how much it is expected to fail on a given task.

Examples:

- **Classification:**
  - **Error rate:** $(1/m)\ \Sigma_{i=1:m}\ \mathbf{1}(F(\mathbf{x}_i)\neq y_i)$
  - **1- AUC** (Gini Index = 2 AUC-1)
- **Regression:**
  - **Mean square error:** $(1/m)\ \Sigma_{i=1:m}(f(\mathbf{x}_i)-y_i)^2$

How to train?

- Define a risk functional R[f(**x**,**w**)]
- Optimize it w.r.t. **w** (gradient descent, mathematical programming, simulated annealing, genetic algorithms, etc.)



(… to be continued in the next lecture)

How to Train?

- Define a risk functional R[f(**x**,**w**)]
- Find a method to optimize it, typically "gradient descent"

$$w_j \leftarrow w_j - \eta \, \partial R / \partial w_j$$

or any optimization method (mathematical programming, simulated annealing, genetic algorithms, etc.)

(… to be continued in the next lecture)

Summary

- With linear threshold units (“neurons”) we can build:
  - Linear discriminant (including Naïve Bayes)
  - Kernel methods
  - Neural networks
  - Decision trees
- The architectural hyper-parameters may include:
  - The choice of basis functions ϕ (features)
  - The kernel
  - The number of units
- Learning means fitting:
  - Parameters (weights)
  - Hyper-parameters
  - Be aware of the fit *vs.* robustness tradeoff

Want to Learn More?

- **Pattern Classification,** *R. Duda, P. Hart, and D. Stork.* Standard pattern recognition textbook. Limited to classification problems. Matlab code. http://rii.ricoh.com/~stork/DHS.html
- **The Elements of statistical Learning: Data Mining, Inference, and Prediction.** *T. Hastie, R. Tibshirani, J. Friedman,* Standard statistics textbook. Includes all the standard machine learning methods for classification, regression, clustering. R code. http://www-stat-class.stanford.edu/~tibs/ElemStatLearn/
- **Linear Discriminants and Support Vector Machines,** *I. Guyon and D. Stork,* In Smola et al Eds. Advances in Large Margin Classiers. Pages 147--169, MIT Press, 2000. http://clopinet.com/isabelle/Papers/guyon_stork_nips98.ps.gz
- **Feature Extraction: Foundations and Applications.** *I. Guyon et al, Eds.* Book for practitioners with datasets of NIPS 2003 challenge, tutorials, best performing methods, Matlab code, teaching material. *http://clopinet.com/fextract-book*