Learning from data

Support vector machines and neural networks

Similarities and differences

Vojislav Kecman, The University of Auckland, Auckland, NZ

Slides accompanying The MIT Press‘ book: Learning and Soft Computing

The slides shown here are developed around the basic notion of

Learning from Experimental Data (samples, observations, records, measurements, patterns, examples) by SVMs and NNs

as presented in the book

LEARNING AND SOFT COMPUTING

Support Vector Machines, Neural Networks and Fuzzy Logic Models

Author:Vojislav KECMAN

The MIT Press, Cambridge, MA, 2001
ISBN 0-262-11255-8
608 pp., 268 illustrations, 47 examples, 155 problems

They are intended to support both the instructors in the development and delivery of course content and the learners in acquiring the ideas and techniques presented in the book in a more pleasant way than just reading.

T O P I C S in INTRODUCTARY PART

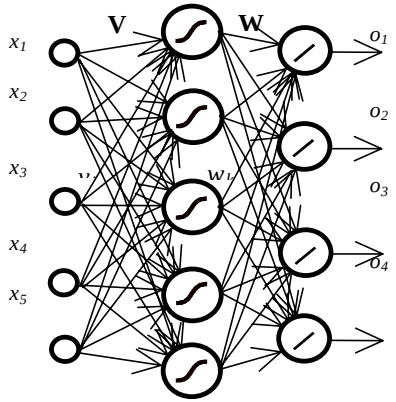
- SVMs & NNs as the BLACK box modeling, and
- FLMs as the WHITE box modeling
- Motivation and basics. Graphic presentation of any approximation or classification scheme leads to the so-called NNs and/or SVMs
- Curve and surface fittings, multivariate function approximation, nonlinear optimization and NN

NN, SVM AND FL MODELING

NNs & SVMs

Black-Box

No previous knowledge, but there are measurements, observations, records, data.



Behind NN stands the idea of **learning** from the data.

FL Models

White-Box

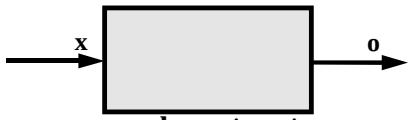
Structured knowledge (*experience, expertise or heuristics*). **IF - THEN** rules are the most typical examples of the structured knowledge.

Example: Controlling the distance between two cars.

R1: IF the speed is *low* AND the distance is *small*
THEN the force on brake should be *small*
R2: IF the speed is *medium*
AND the distance is *small*
THEN the force on brake should be *big*
R3: IF the speed is *high*
AND the distance is *small*
THEN the force on brake should be *very big*

Behind FLM stands the idea of **embedding human knowledge** into workable algorithms.

At many instances we do have both ***some knowledge*** and ***some data.***



This is the most common **gray box** situation covered by the paradigm of **Neuro-Fuzzy *or* Fuzzy-Neuro** models.

If we do not have any prior knowledge AND we do not have any measurements *(by all accounts very hopeless situation indeed)* it may be hard to expect or believe that the problem at hand may be approached and solved easily. This is a **no-color box** situation.

Classical approximation techniques in NN graphical appearance

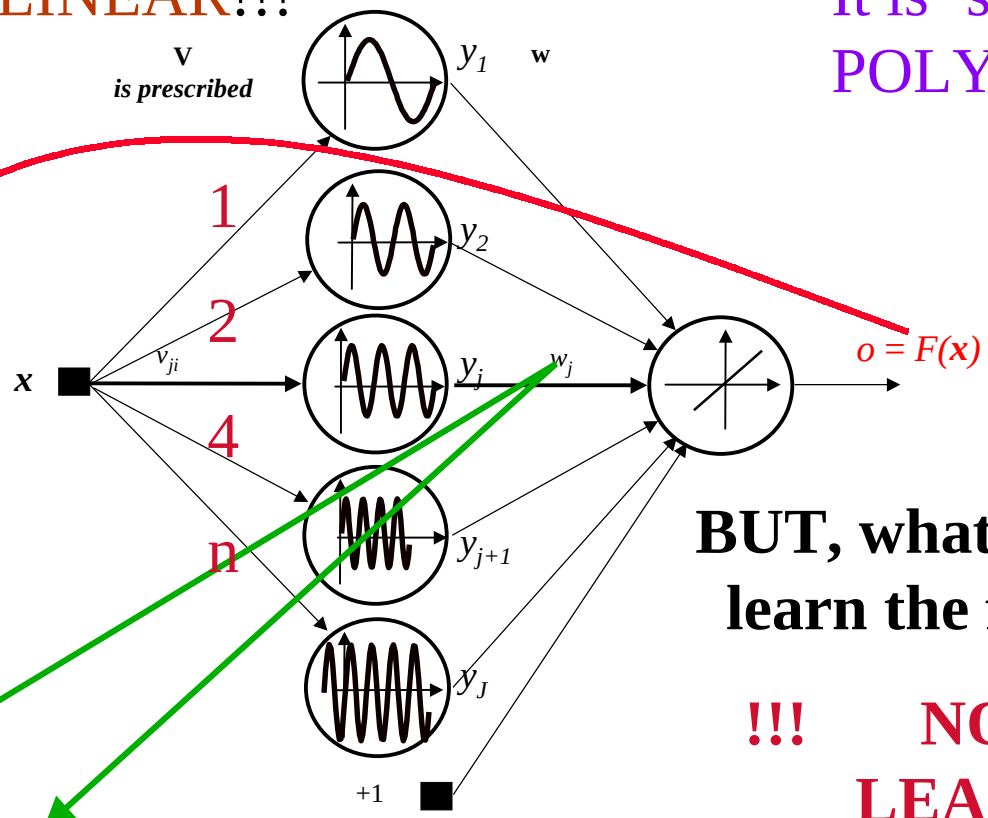
FOURIER SERIES

AMPLITUDES and PHASES of sine (cosine) waves are ?, but frequencies are known because

Mr Joseph Fourier has selected frequencies for us -> they are INTEGER multiplies of some pre-selected base frequency.

And the problem is LINEAR!!!

It is ‘same’ with POLYNOMIALS

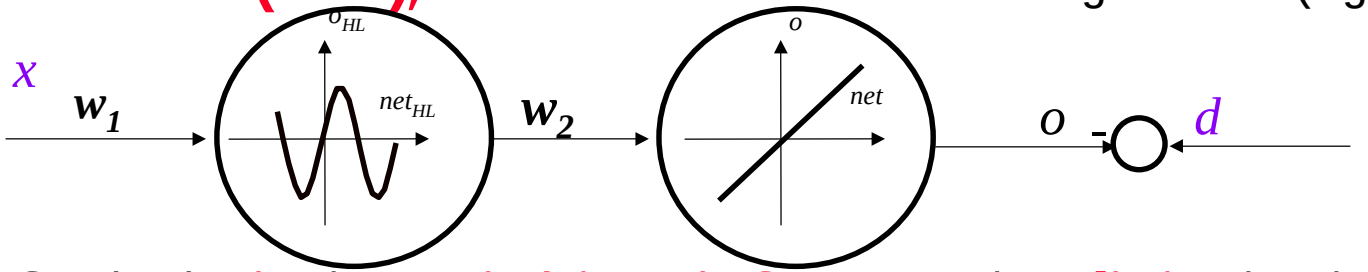


$$F(x) = \sum_{k=1}^{N} a_k \sin(kx),\ or\ b_k \cos(kx),\ or\ both$$

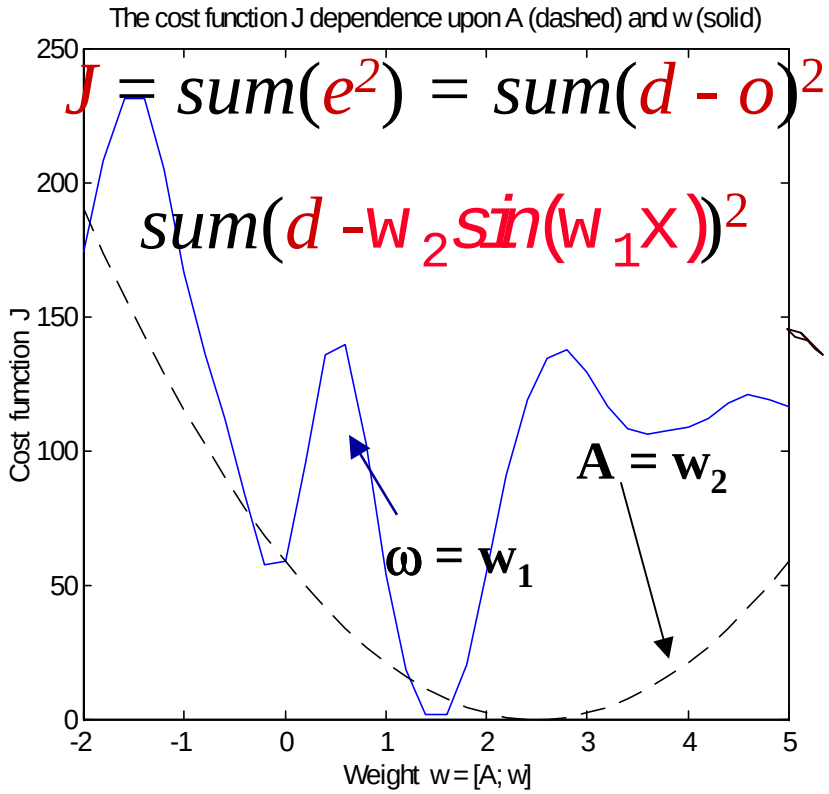
BUT, what if we want to learn the frequencies?

!!! NONLINEAR LEARNING PROBLEM !!!

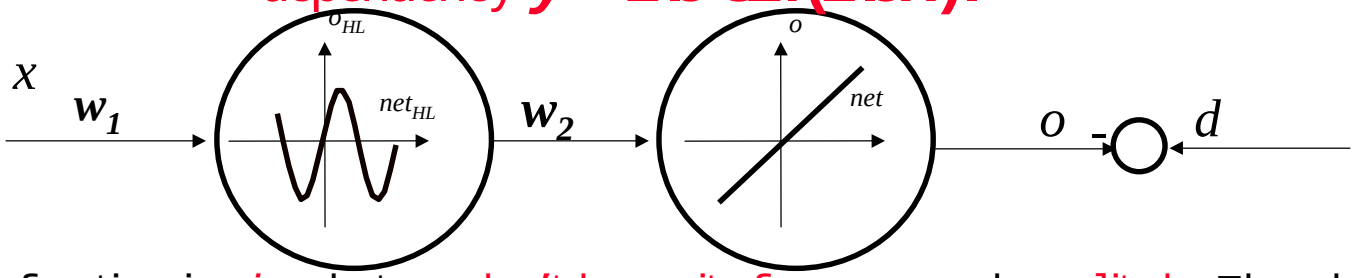
Now, we want to find Fourier 'series' model $y = w_2 \sin(w_1 x)$ of the underlying dependency $y = 2.5 \sin(1.5x)$, known to us but not to the learning machine (algorithm).



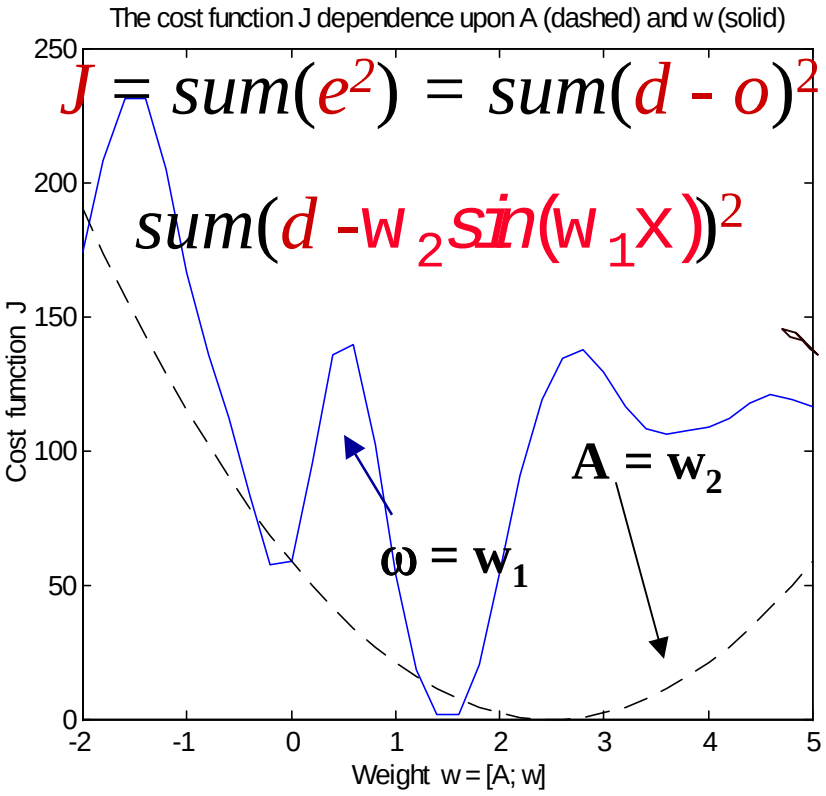
We know that the function is *sinus* but we don't know its frequency and amplitude. Thus, by using the training data set {**x**, *d*}, we want to model this system with the NN model consisting of a single neuron in HL (having *sinus* as an activation function) as given above.

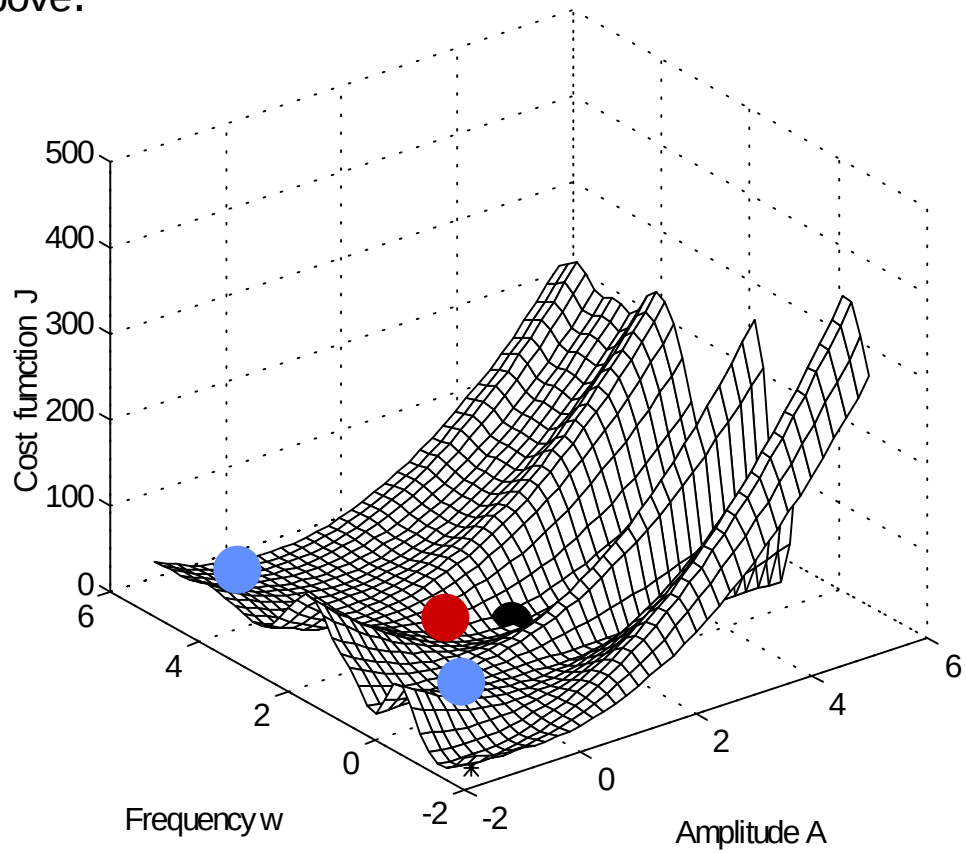


Now, we want to find Fourier 'series' model $y = w_2 \sin(w_1 x)$ of the underlying dependency $y = 2.5 \sin(1.5x)$.



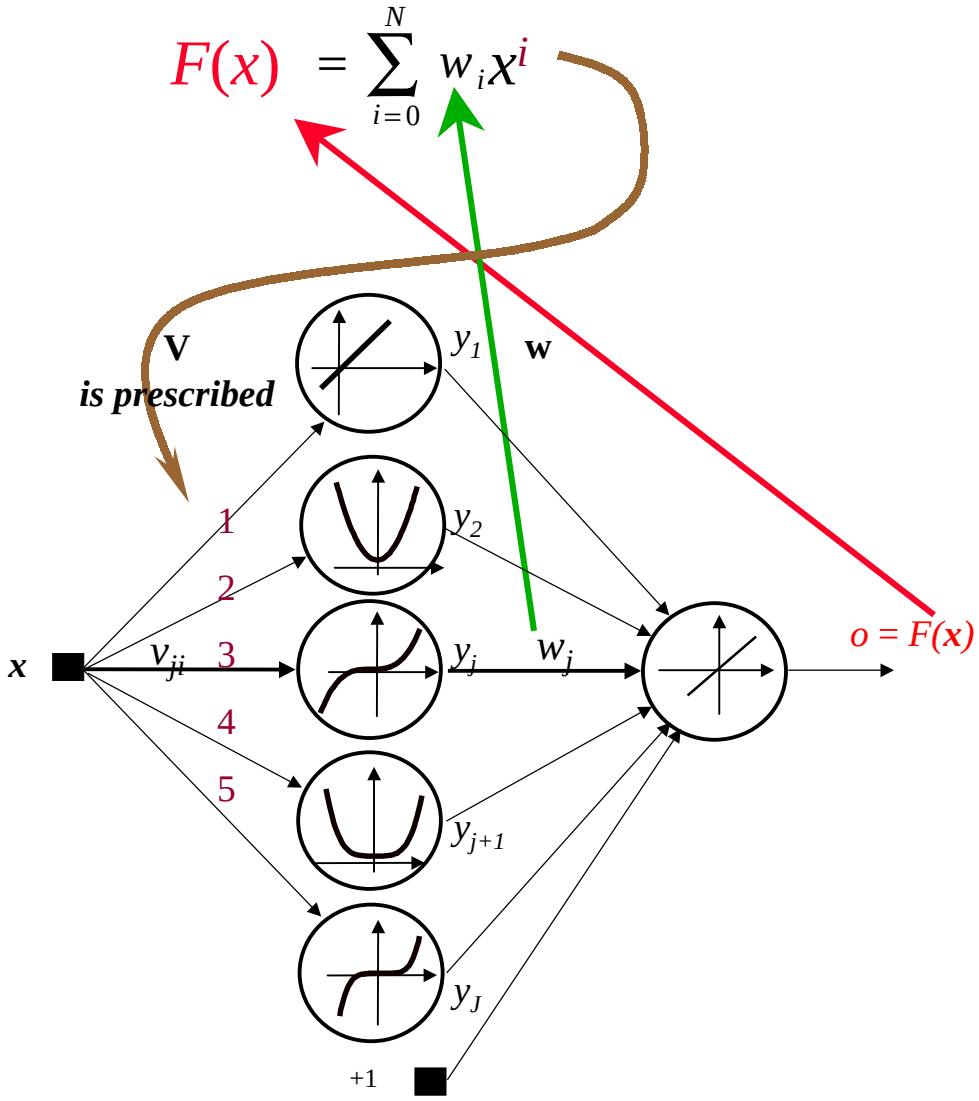
We know that the function is *sinus* but we don't know its frequency and amplitude. Thus, by using the training data set {x, d}, we want to model this system with the NN model consisting of a single neuron in HL (having *sinus* as an activation function) as given above.





Another classical approximation scheme is a

POLYNOMIAL SERIES

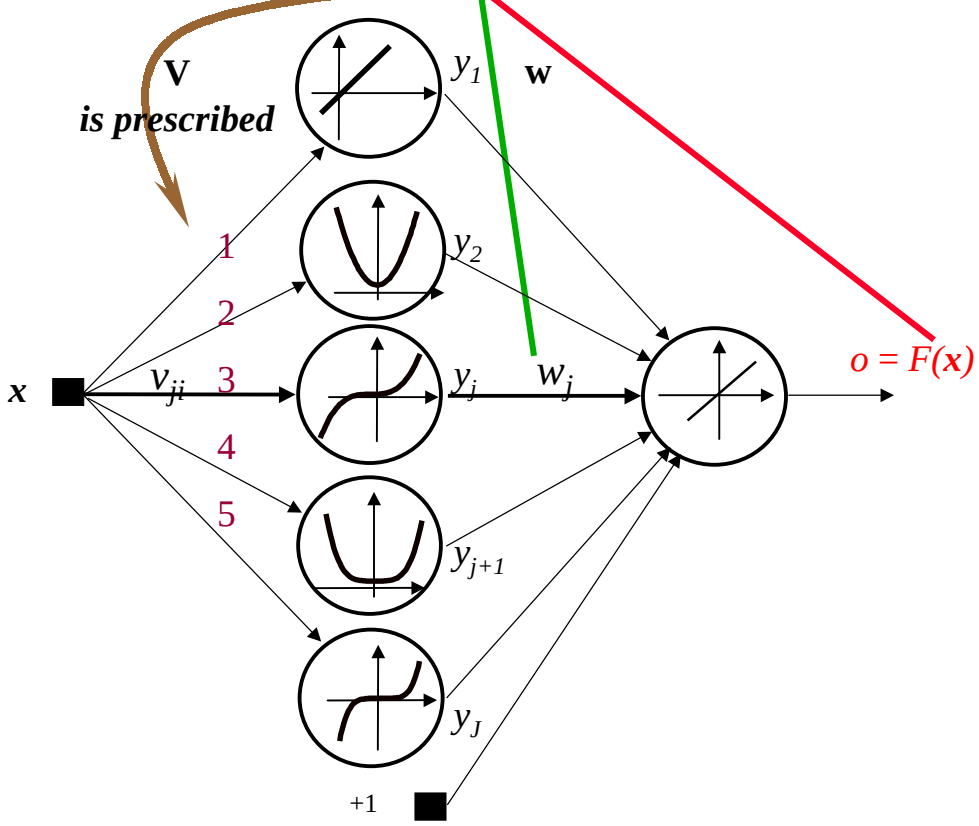


Another classical approximation scheme is a

POLYNOMIAL SERIES

$$F(x) = \sum_{i=0}^{N} w_i x^i$$

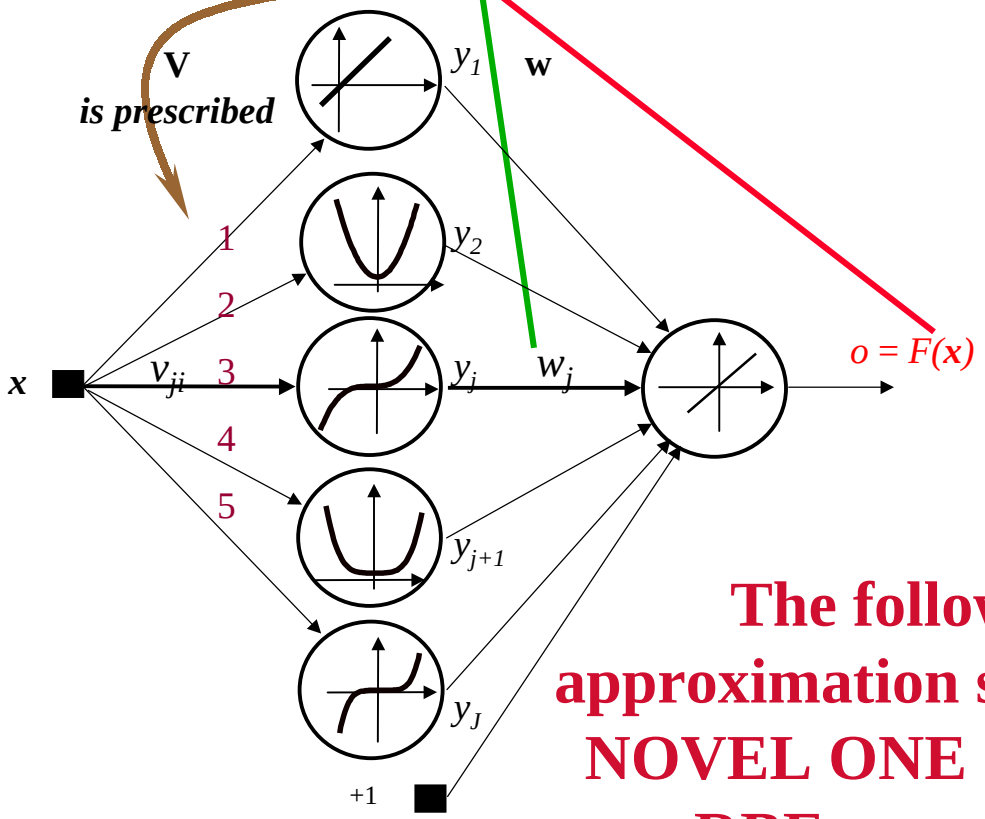
With a prescribed (integer) exponents this is again LINEAR APPROXIMATION SCHEME. Linear in terms of parameters to learn and not in terms of the resulting approximation function. This one is NL function for $i > 1$.



Another classical approximation scheme is a

POLYNOMIAL SERIES

$$F(x) = \sum_{i=0}^{N} w_i x^i$$



With a prescribed (integer) exponents this is again LINEAR APPROXIMATION SCHEME. Linear in terms of parameters to learn and not in terms of the resulting approximation function. This one is NL function for $i > 1$.

The following approximation scheme is a NOVEL ONE called NN or RBF network here.

Approximation of some

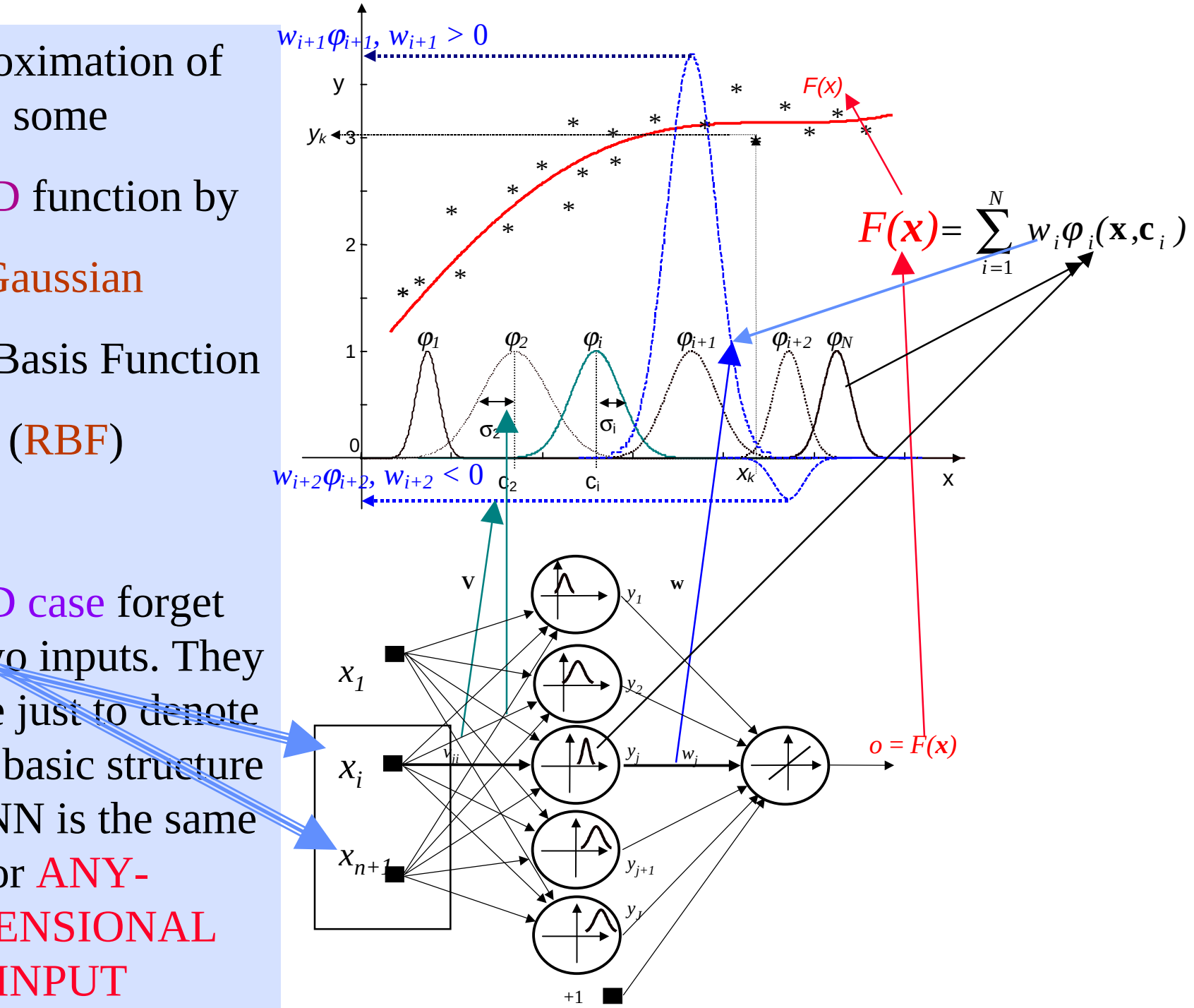
NL 1D function by

Gaussian

Radial Basis Function

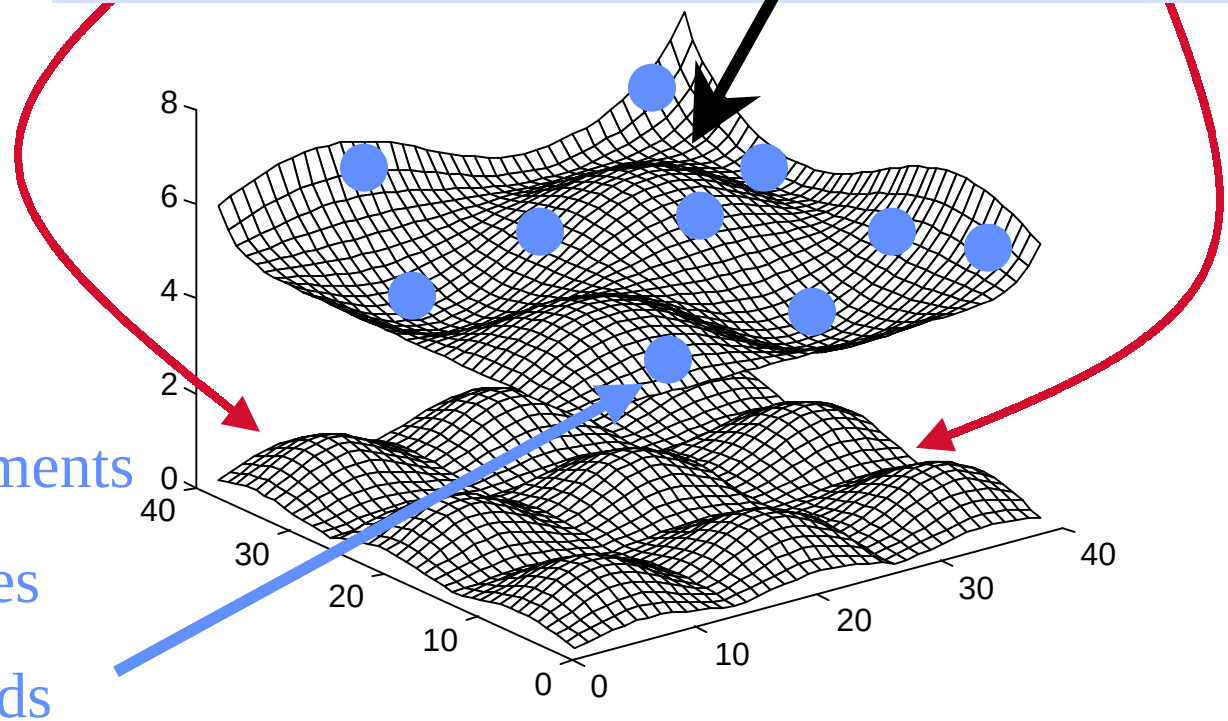
(RBF)

In 1-D case forget these two inputs. They are here just to denote that the basic structure of the NN is the same for ANY-DIMENSIONAL INPUT



Approximation of some NL 2D function by

Gaussian Radial Basis Function (RBF)



Measurements

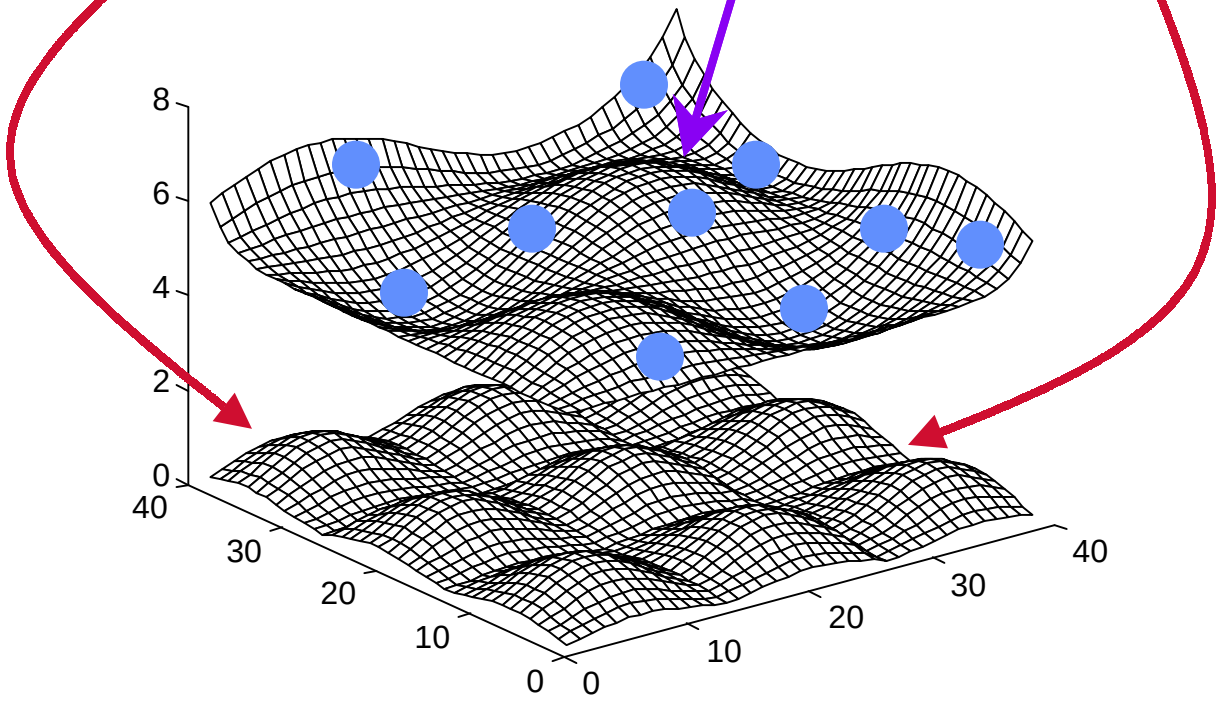
Images

Records

Observations

Data

Approximation of some NL 2D function by Gaussian Radial Basis Functions (RBF)



For FIXED Gaussian RBFs a LEARNING FROM DATA is LINEAR PROBLEM. If the Centers and Covariance matrices are the subjects of learning, problem becomes NONLINEAR.

The learning machine that uses data to find the APPROXIMATING FUNCTION (in regression problems) or the SEPARATION BOUNDARY (in classification, pattern recognition problems), is the same in high-dimensional situations.

It is either the so-called **SVM** or the **NN**.

WHAT are DIFFERENCES and SIMILARITIES?

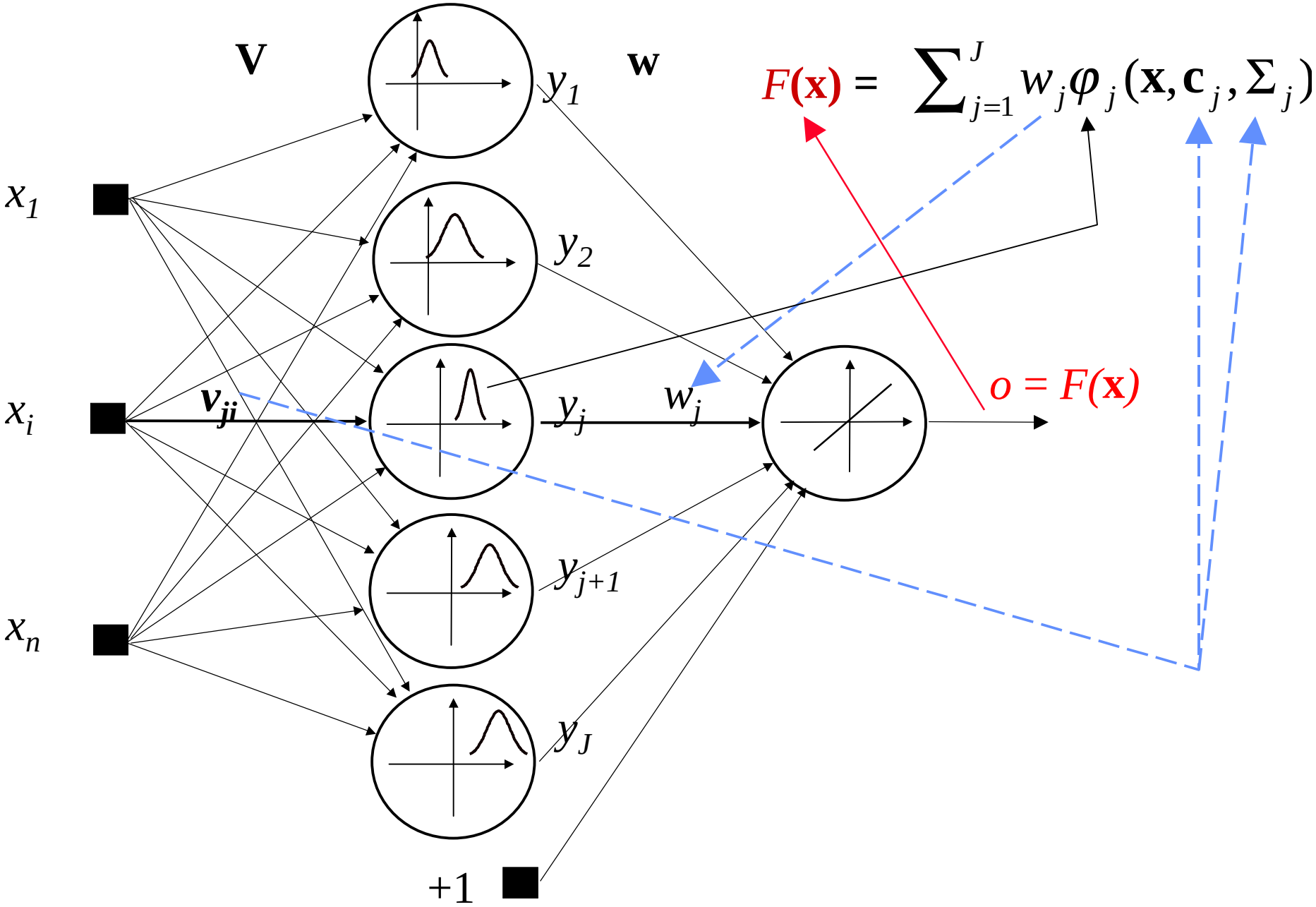
The learning machine that uses data to find the APPROXIMATING FUNCTION (in regression problems) or the SEPARATION BOUNDARY (in classification, pattern recognition problems), is the same in high-dimensional situations.

It is either the so-called **SVM** or the **NN**.

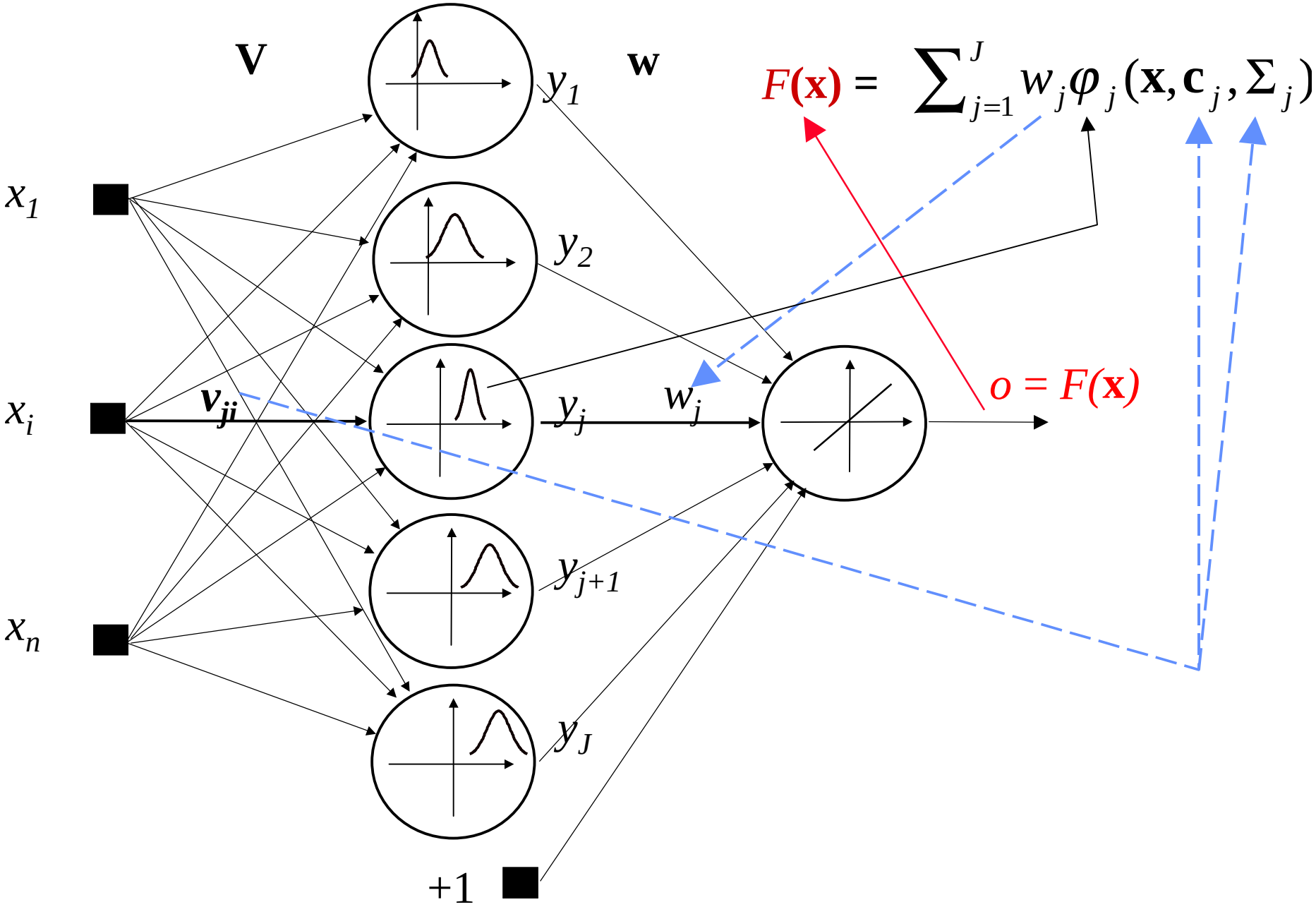
WHAT are DIFFERENCES and SIMILARITIES?

WHATCH CAREFULLY NOW !!!

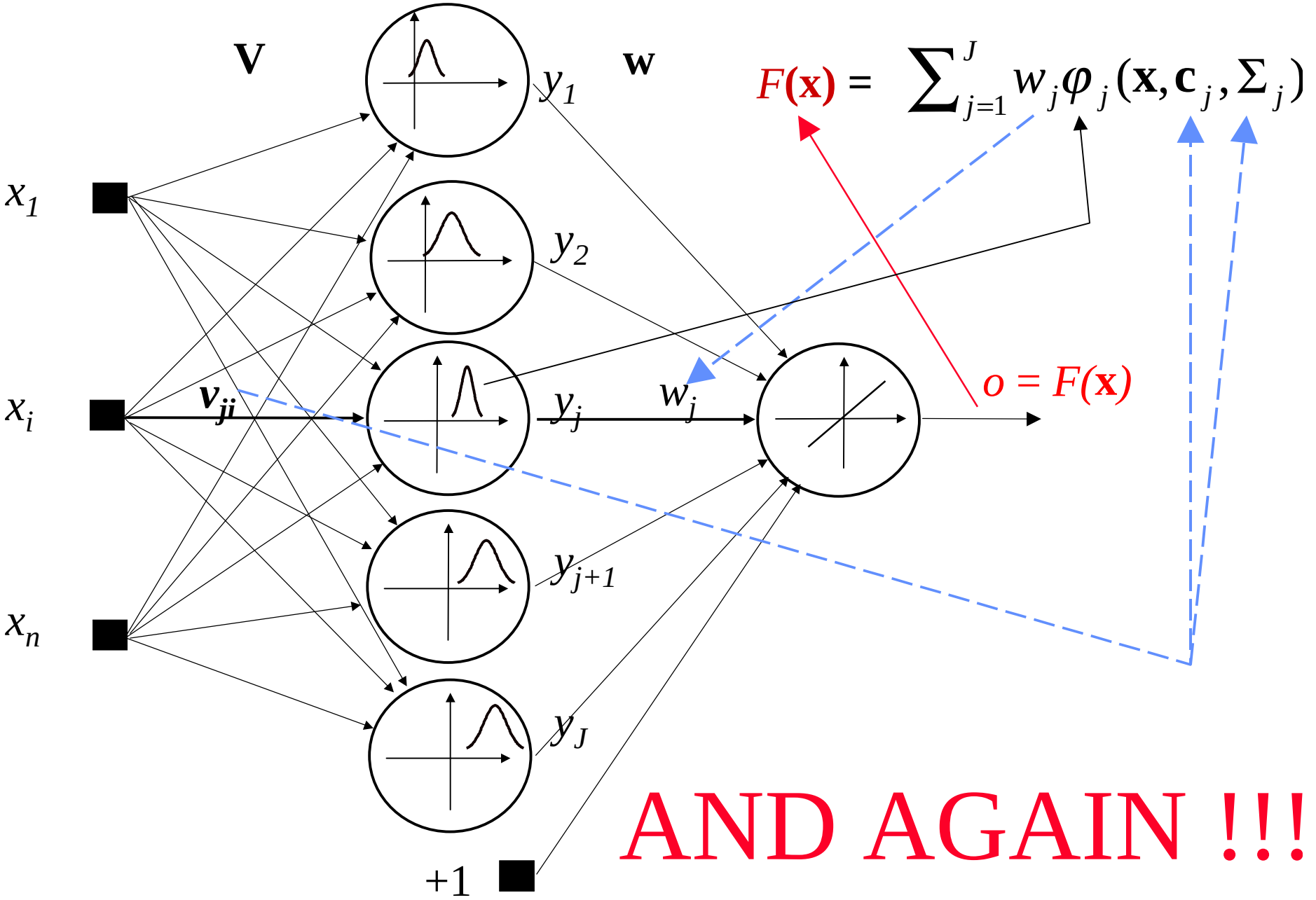
This is a Neural Network,



and, this is a Support Vector Machine.

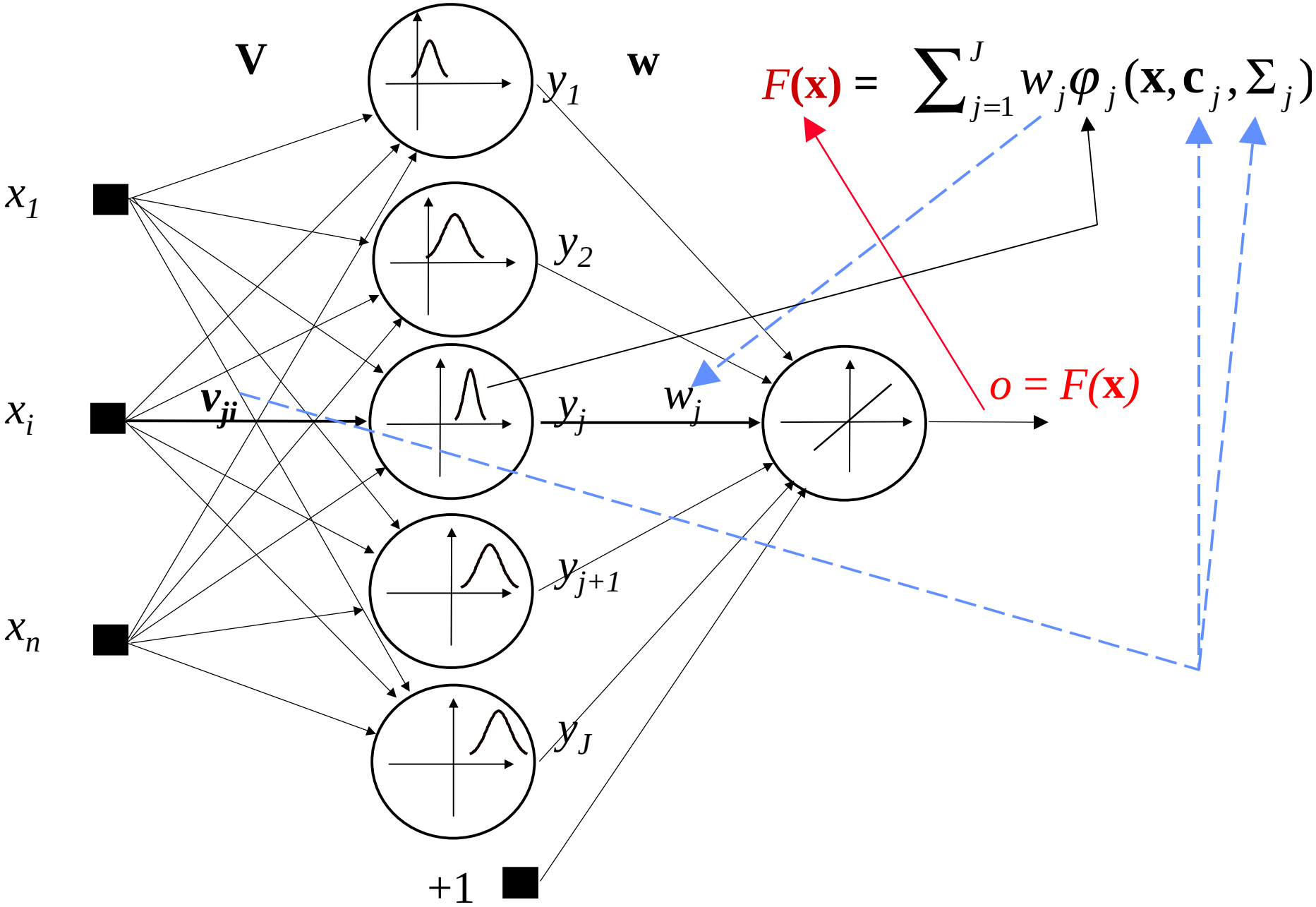


and, this is a Support Vector Machine.

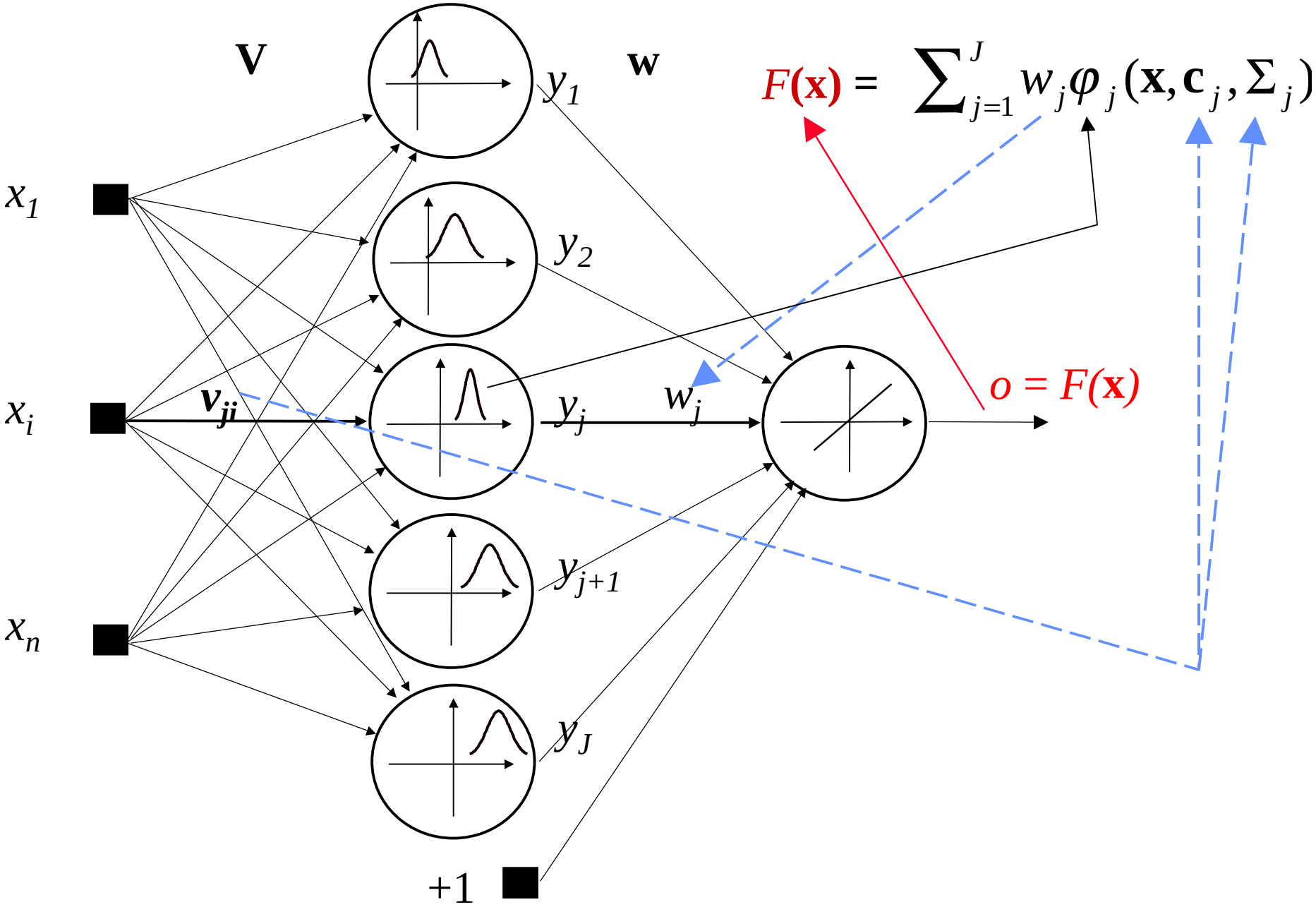


AND AGAIN !!!

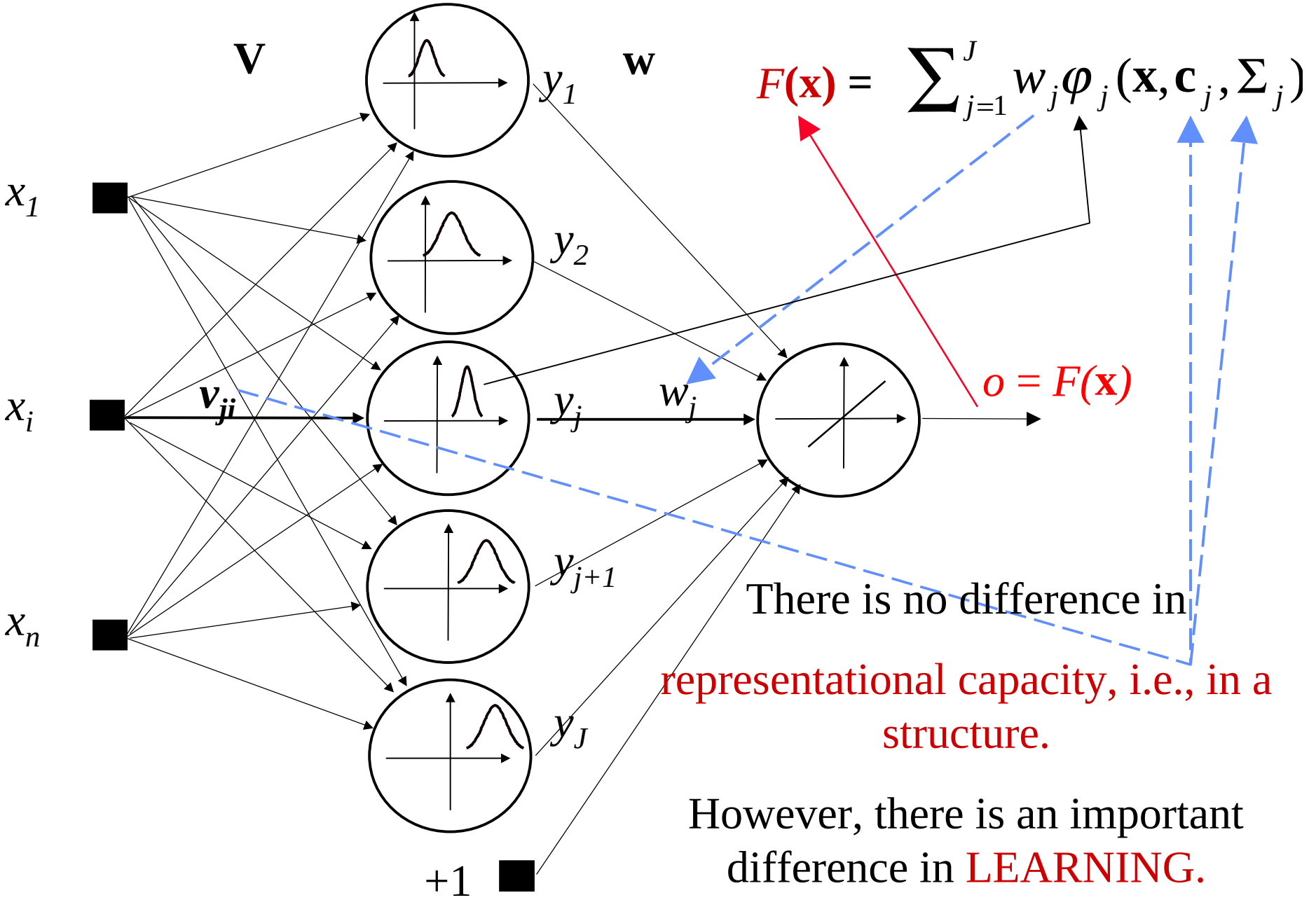
This is a Neural Network,



and, this is a Support Vector Machine.



and, this is a Support Vector Machine.



There is no difference in

representational capacity, i.e., in a structure.

However, there is an important difference in LEARNING.

Therefore,

let's talk about basics of

the learning from data first.

Note that you may find different names for the L from D:

identification, estimation, regression, classification, pattern recognition, function approximation, curve or surface fitting etc.

All these tasks used to be solved previously.

Thus, THERE IS THE QUESTION:

Is there anything new in respect to the classical statistical inference?

The classical regression and (Bayesian) classification statistical techniques are based on the very strict assumption that probability distribution models or probability-density functions are known.

Classical statistical inference is based on the following three fundamental assumptions:

The classical regression and (Bayesian) classification statistical techniques are based on the very strict assumption that probability distribution models or probability-density functions are known.

Classical statistical inference is based on the following three fundamental assumptions:

***Data can be modeled by a set of linear in parameter functions; this is a foundation of a parametric paradigm in learning from experimental data.**

The classical regression and (Bayesian) classification statistical techniques are based on the very strict assumption that probability distribution models or probability-density functions are known.

Classical statistical inference is based on the following three fundamental assumptions:

***Data can be modeled by a set of linear in parameter functions; this is a foundation of a parametric paradigm in learning from experimental data.**

***In the most of real-life problems, a stochastic component of data is the normal probability distribution law, i.e., the underlying joint probability distribution is Gaussian.**

The classical regression and (Bayesian) classification statistical techniques are based on the very strict assumption that probability distribution models or probability-density functions are known.

Classical statistical inference is based on the following three fundamental assumptions:

*Data can be modeled by a set of linear in parameter functions; this is a foundation of a parametric paradigm in learning from experimental data.

*In the most of real-life problems, a stochastic component of data is the normal probability distribution law, i.e., the underlying joint probability distribution is Gaussian.

*Due to the second assumption, the induction paradigm for parameter estimation is the maximum likelihood method that is reduced to the minimization of the sum-of-errors-squares cost function in most engineering applications.

All three assumptions on which the classical statistical paradigm relied, turned out to be inappropriate for many contemporary real-life problems (Vapnik, 1998) due to the facts that:

All three assumptions on which the classical statistical paradigm relied, turned out to be inappropriate for many contemporary real-life problems (Vapnik, 1998) due to the facts that:

***modern problems are high-dimensional, and if the underlying mapping is not very smooth the linear paradigm needs an exponentially increasing number of terms with an increasing dimensionality of the input space *X*, i.e., with an increase in the number of independent variables. This is known as 'the curse of dimensionality',**

All three assumptions on which the classical statistical paradigm relied, turned out to be inappropriate for many contemporary real-life problems (Vapnik, 1998) due to the facts that:

***modern problems are high-dimensional, and if the underlying mapping is not very smooth the linear paradigm needs an exponentially increasing number of terms with an increasing dimensionality of the input space *X*, i.e., with an increase in the number of independent variables. This is known as 'the curse of dimensionality',**

***the underlying real-life data generation laws may typically be very far from the normal distribution and a model-builder must consider this difference in order to construct an effective learning algorithm,**

All three assumptions on which the classical statistical paradigm relied, turned out to be inappropriate for many contemporary real-life problems (Vapnik, 1998) due to the facts that:

***modern problems are high-dimensional, and if the underlying mapping is not very smooth the linear paradigm needs an exponentially increasing number of terms with an increasing dimensionality of the input space *X*, i.e., with an increase in the number of independent variables. This is known as 'the curse of dimensionality',**

***the underlying real-life data generation laws may typically be very far from the normal distribution and a model-builder must consider this difference in order to construct an effective learning algorithm,**

***from the first two objections it follows that the maximum likelihood estimator (and consequently the sum-of-error-squares cost function) should be replaced by a new induction paradigm that is uniformly better, in order to model non-Gaussian distributions.**

There is a real life fact

the probability-density functions are TOTALLY unknown,

and there is the question

HOW TO PERFORM a *distribution-free*

***REGRESSION* or *CLASSIFICATION* ?**

There is a real life fact

the probability-density functions are TOTALLY unknown,

and there is the question

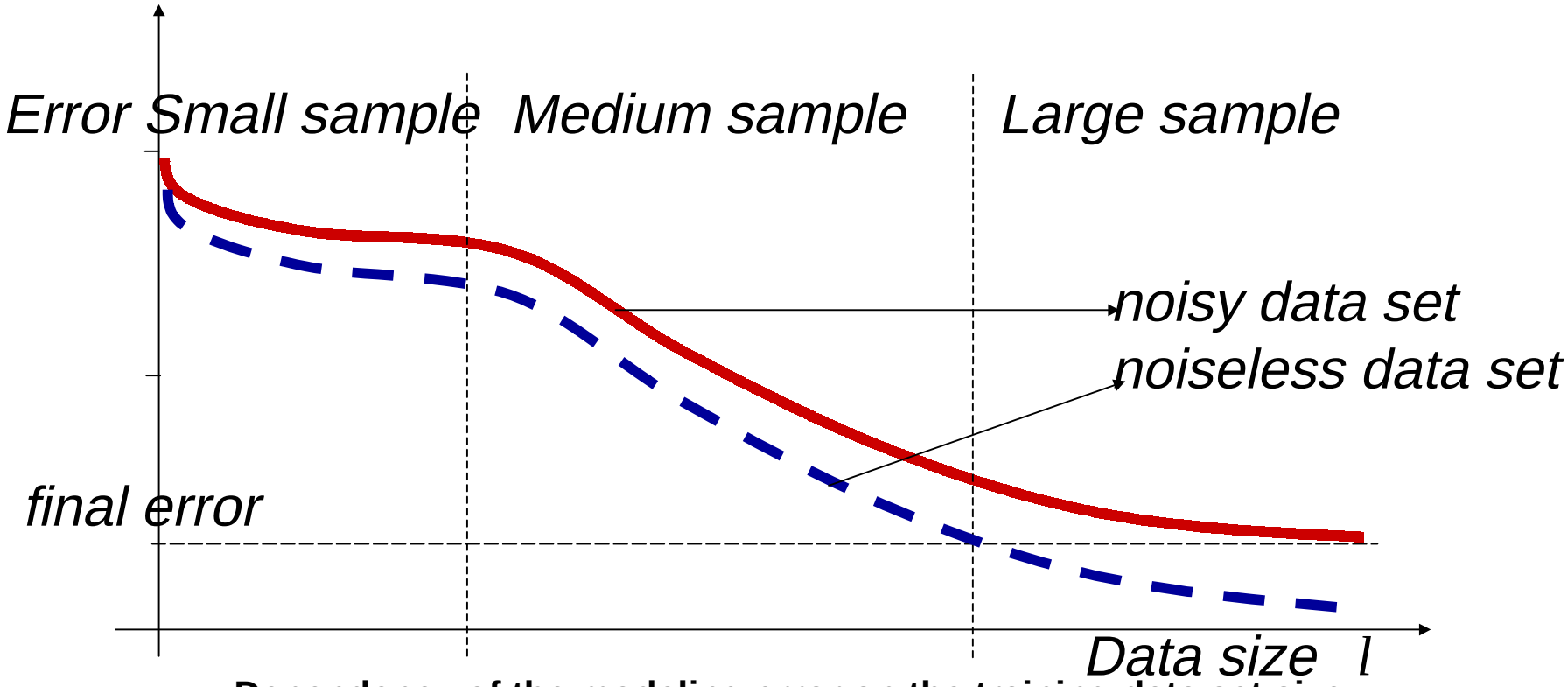
HOW TO PERFORM a *distribution-free*

***REGRESSION* or *CLASSIFICATION* ?**

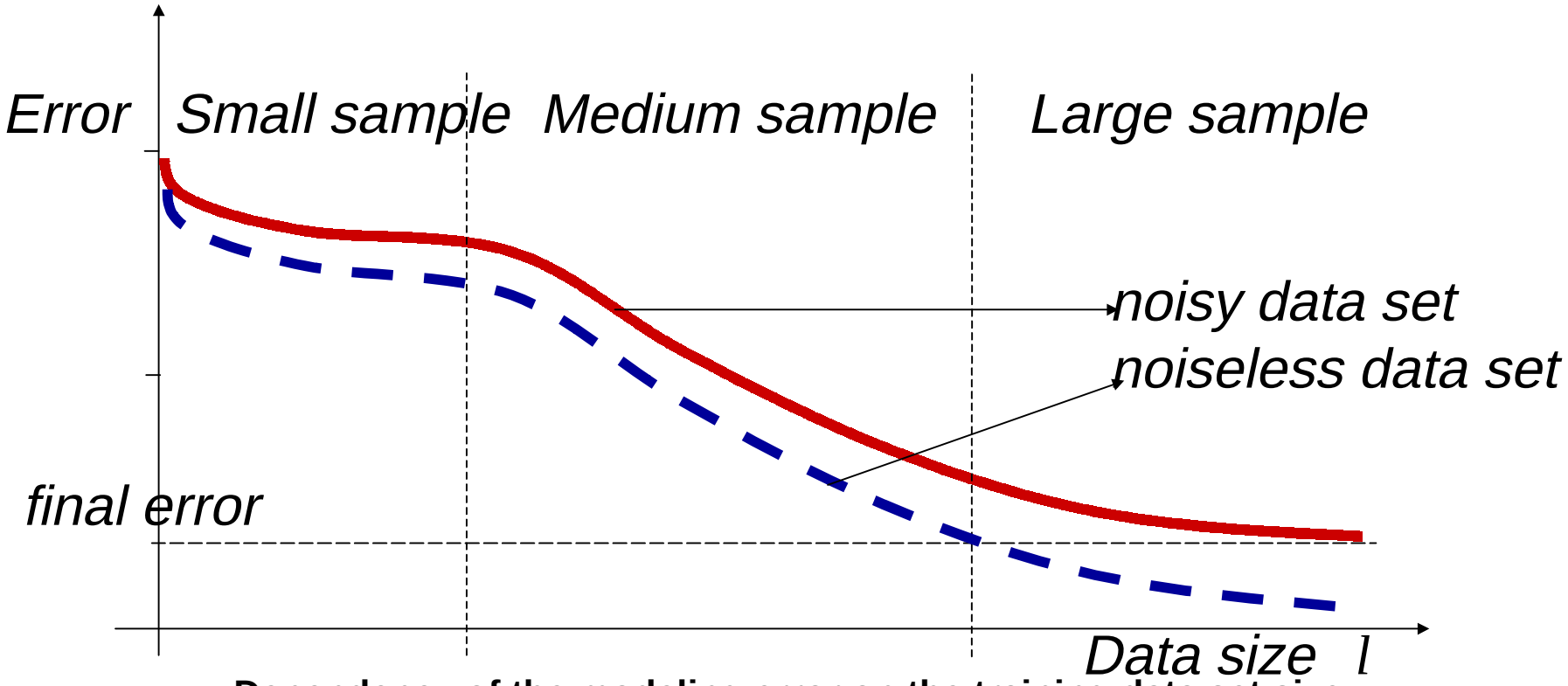
Mostly, all we have are recorded **EXPERIMENTAL DATA** (training patterns, samples, observations, records, examples) that are usually **high-dimensional** and **scarce** in present day applications.

High-dimensional spaces seem to be **terrifyingly empty** and our learning algorithms (i.e., machines) should be able to operate in such spaces and to **learn from sparse data**.

There is an old saying that *redundancy provides knowledge*. Stated simpler – we expect that the more data pairs we have the better results will be. These essentials are depicted in the next figure.



Dependency of the modeling error on the training data set size.



Dependency of the modeling error on the training data set size.

Glivenko-Cantelli-Kolmogorov results.

Glivenko-Cantelli theorem states that:

Distribution function $P_{emp}(x) \rightarrow P(x)$

as the number of data $l \rightarrow \infty$.

However, for both regression and classification we need probability density functions $p(x)$, i.e., $p(x|\omega)$ and not a distribution P(x).

Therefore, there is a question:

whether $p_{emp}(x) \to p(x)$
as the number of data $l \to \infty$.

Answer is both not straight and not guaranteed,

despite the fact that

$$\int p(x)dx = P(x).$$

(Analogy with a classic problem Ax = y, x = A^{-1}y)!

Therefore, there is a question:

whether $p_{emp}(x) \rightarrow p(x)$
as the number of data $l \rightarrow \infty$.

Answer is both not straight and not guaranteed,

despite the fact that

$$\int p(x)dx = P(x).$$

(Analogy with a classic problem Ax = y, x = A^{-1}y)!

What is needed here is the theory of
UNIFORM CONVERGENCE of the set of functions implemented by a model, i.e., learning machine
(Vapnik, Chervonenkis, 1960-70ties).

We, will skip a discussion about this part here.

Skip it, but not forget it.

Here, we discuss **nonlinear** **and** **'nonparametric'** models exemplified by NNs and SVMs.

Nonlinear means two things:

1) our model class will be not restricted to linear input-output maps and,

2) the dependence of the cost function that measures the goodness of our model, will be **nonlinear** in respect to the unknown parameters.

In passing it may be noted that **the second nonlinearity is the part of modeling that causes most of the problems**.

‘Nonparametric’ does not mean that our models do not have parameters at all. On contrary,

their learning (meaning selection, identification, estimation, fitting or tuning) is the crucial issue here.

‘Nonparametric’ does not mean that our models do not have parameters at all. On contrary,

their learning (meaning selection, identification, estimation, fitting or tuning) is the crucial issue here.

However, unlike in classical statistical inference, now **they are not predefined** but rather their number depends on the training data used.

In other words, **parameters** that define the capacity of the model **are data driven** in such a way as to match the model capacity to the data complexity. This is a basic paradigm of the structural risk minimization (SRM) introduced by Vapnik and Chervonenkis and their coworkers.

The main characteristics of all modern problems is the mapping between the high-dimensional spaces.

Let’s see the following pattern recognition (classification) example!

Gender recognition problem: These two faces are female or male?

F or
M?

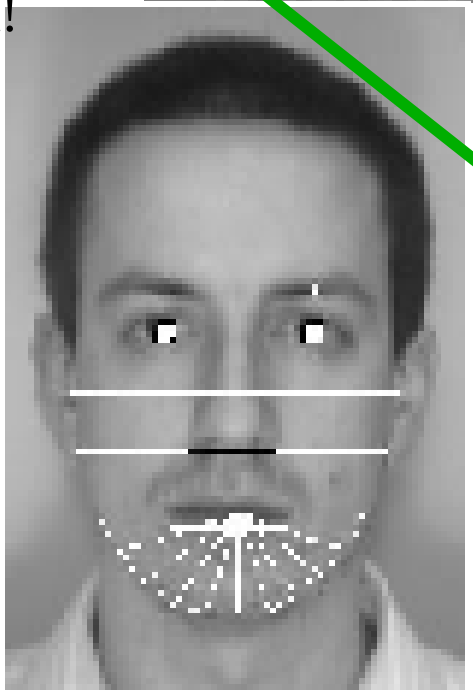
M or
F?

Gender recognition problem: These two faces are female or male?

F or
M?

M or
F?

There must be something in the geometry of our faces. Here, 18 input variables, features, were chosen!



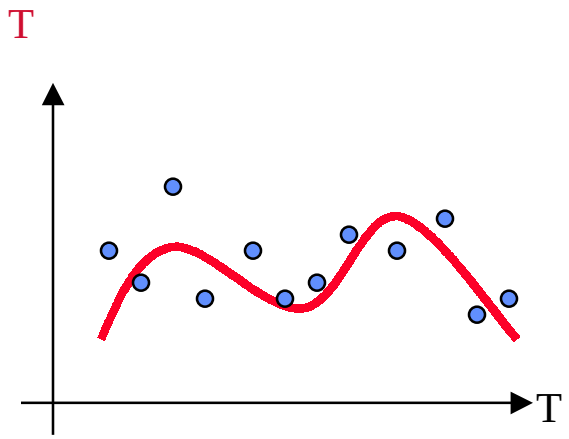
Nr	Feature
1	pupil to nose vertical distance
2	pupil to mouth vertical distance
3	pupil to chin vertical distance
4	nose width
5	mouth width
6	zygomatic breadth
7	bigonial breadth
8-13	chin radii
14	mouth height
15	upper lip thickness
16	lower lip thickness
17	pupil to eyebrow separation
18	eyebrow thickness

Problem from

Brunelli & Poggio, 1993.

Just a few words about the data set size and dimensionality. Approximation and classification are 'same' for any dimensionality of the input space. Nothing (!) but size changes. But the change is DRASTIC. High dimensionality means both an EXPLOSION in a number OF PARAMETERS to learn and a SPARSE training data set.

High dimensional spaces seem to be terrifyingly empty.

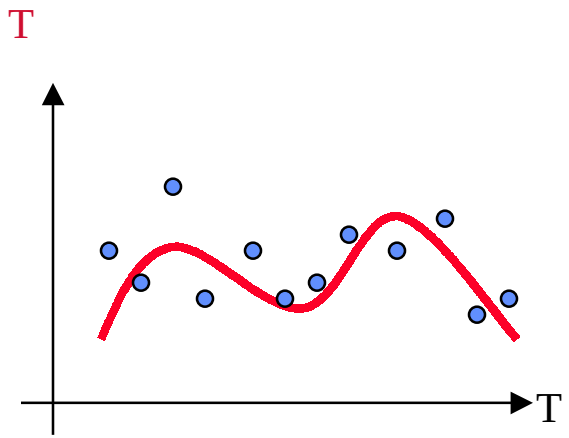


$\Re \rightarrow \Re$

N data

Just a few words about the data set size and dimensionality. Approximation and classification are 'same' for any dimensionality of the input space. Nothing (!) but size changes. But the change is DRASTIC. High dimensionality means both an EXPLOSION in a number OF PARAMETERS to learn and a SPARSE training data set.

High dimensional spaces seem to be terrifyingly empty.

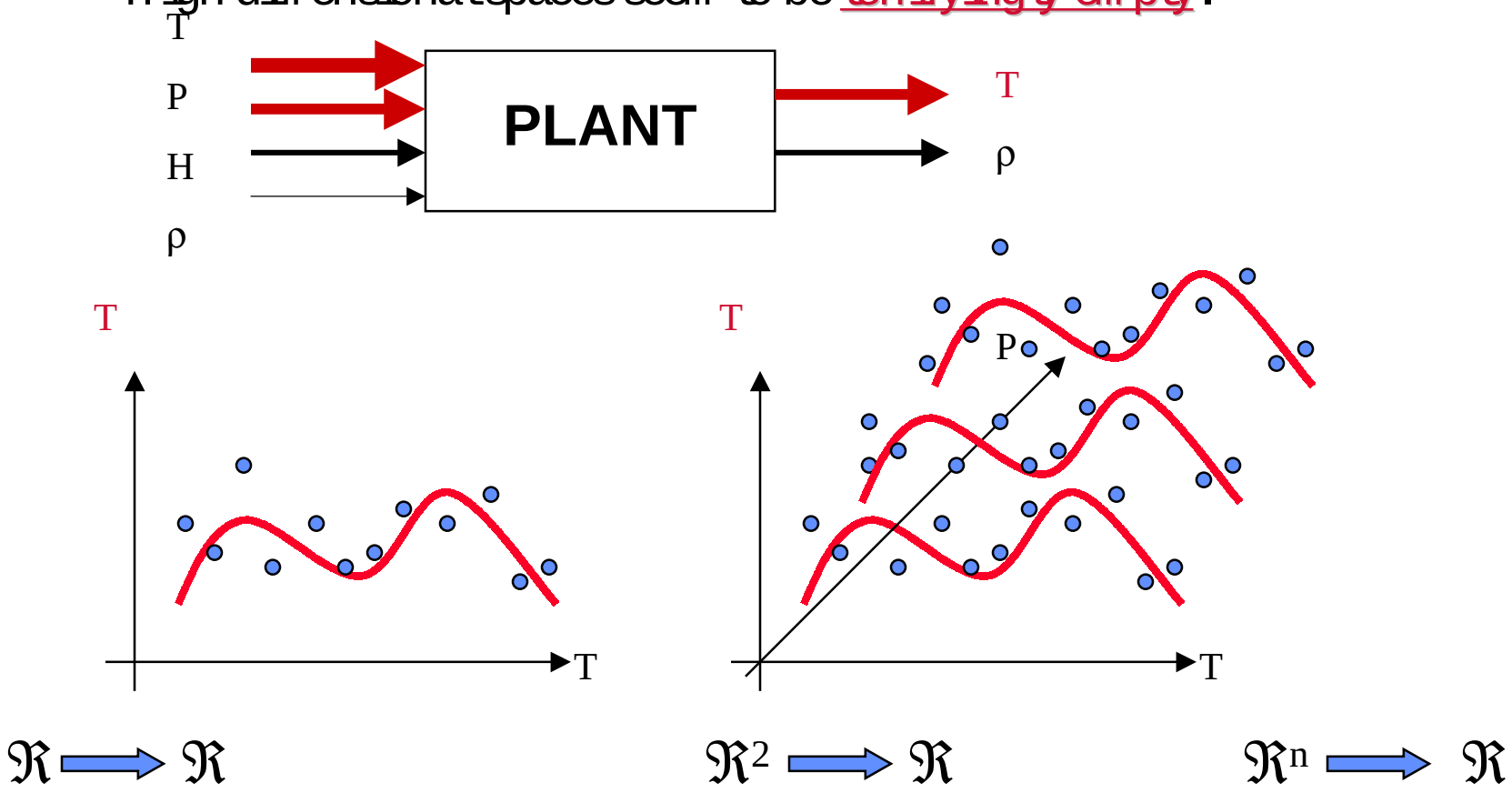


$\Re \Rightarrow \Re$ **However, for 2 inputs (T and P)**

N data **do we need 2N or N^2 data?**

Just a few words about the data set size and dimensionality. Approximation and classification are 'same' for any dimensionality of the input space. Nothing (!) but size changes. But the change is DRASTIC. High dimensionality means both an EXPLOSION in a number OF PARAMETERS to learn and a SPARSE training data set.

High dimensional spaces seem to be terrifyingly empty.



High dimensional spaces are terrifyingly empty.

Many different remedies proposed to resolve this

CURSE of DIMENSIONALITY.

One of proposals to resolve it is to utilize the BLESSING of SMOOTHNES
⟹ REGULARIZATION THEORY ⟹ RBF which contain classical splines.

CURSE of DIMENSIONALITY and SPARSITY OF DATA.

The newest promising tool FOR WORKING UNDER THESE CONSTRAINTS are the SUPPORT VECTOR MACHINES based on the STATISTICAL LEARNING THEORY (VLADIMIR VAPNIK and ALEKSEICHERVONENKIS).

WHAT IS THE contemporary BASIC LEARNING PROBLEM ???

LEARN THE DEPENDENCY (FUNCTION, MAPPING) from SPARSE DATA, under NOISE, in HIGH DIMENSIONAL SPACE!

Recall - the redundancy provides the knowledge!

A lot of data - 'easy' problem.

CURSE of DIMENSIONALITY and SPARSITY OF DATA.

The newest promising tool FOR WORKING UNDER THESE CONSTRAINTS are the SUPPORT VECTOR MACHINES based on the STATISTICAL LEARNING THEORY (VLADIMIR VAPNIK and ALEKSEICHERVONENKIS).

WHAT IS THE contemporary BASIC LEARNING PROBLEM ???

LEARN THE DEPENDENCY (FUNCTION, MAPPING) from SPARSE DATA, under NOISE, in HIGH DIMENSIONAL SPACE!

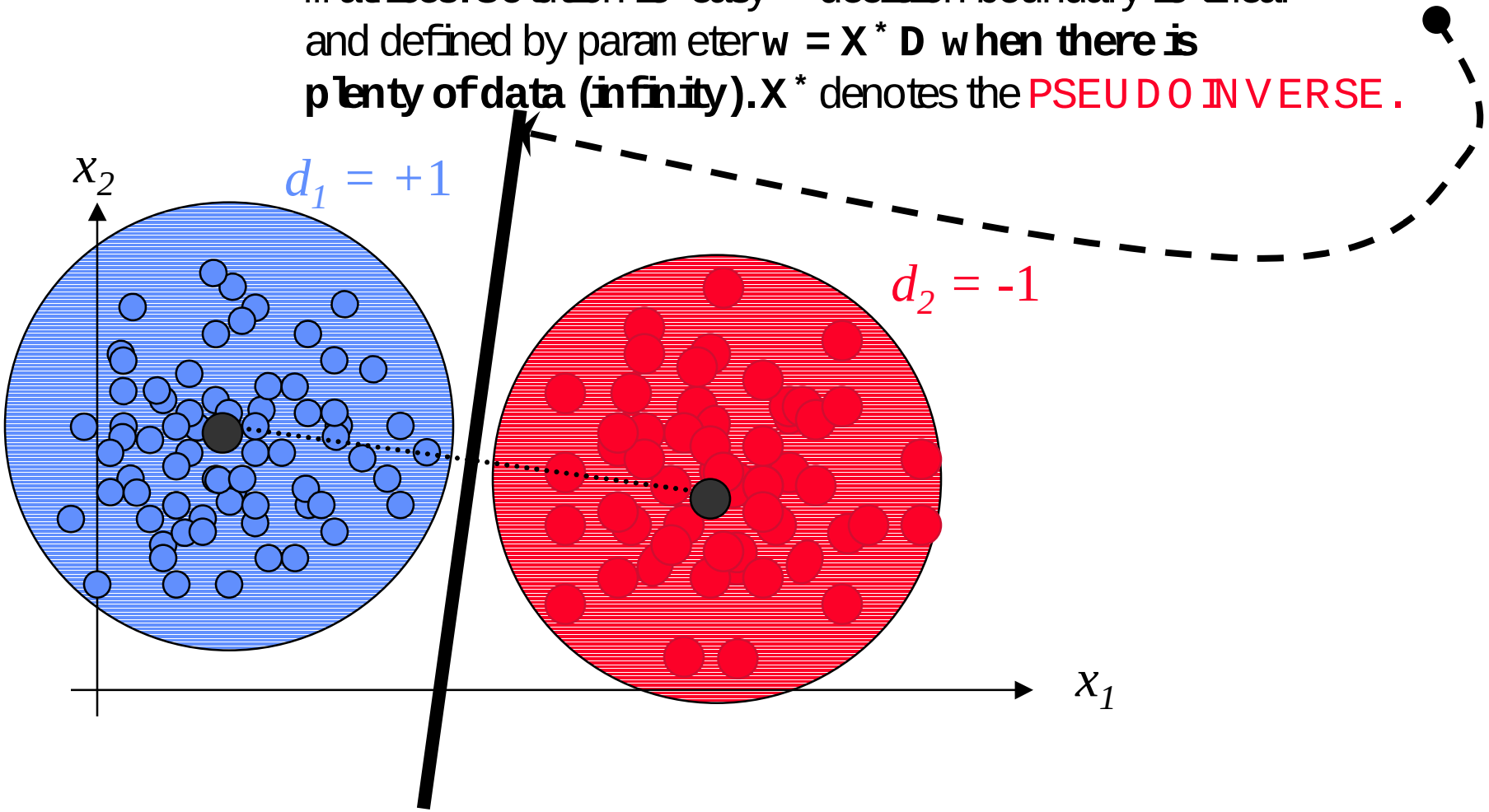
Recall - the redundancy provides the knowledge!

A lot of data - 'easy' problem.

LET'S EXEMPLIFY THE INFLUENCE OF A DATA SET SIZE ON THE SIMPLEST RECOGNITION PROBLEM BINARY CLASSIFICATION, i.e., DICHOTOMIZATION.

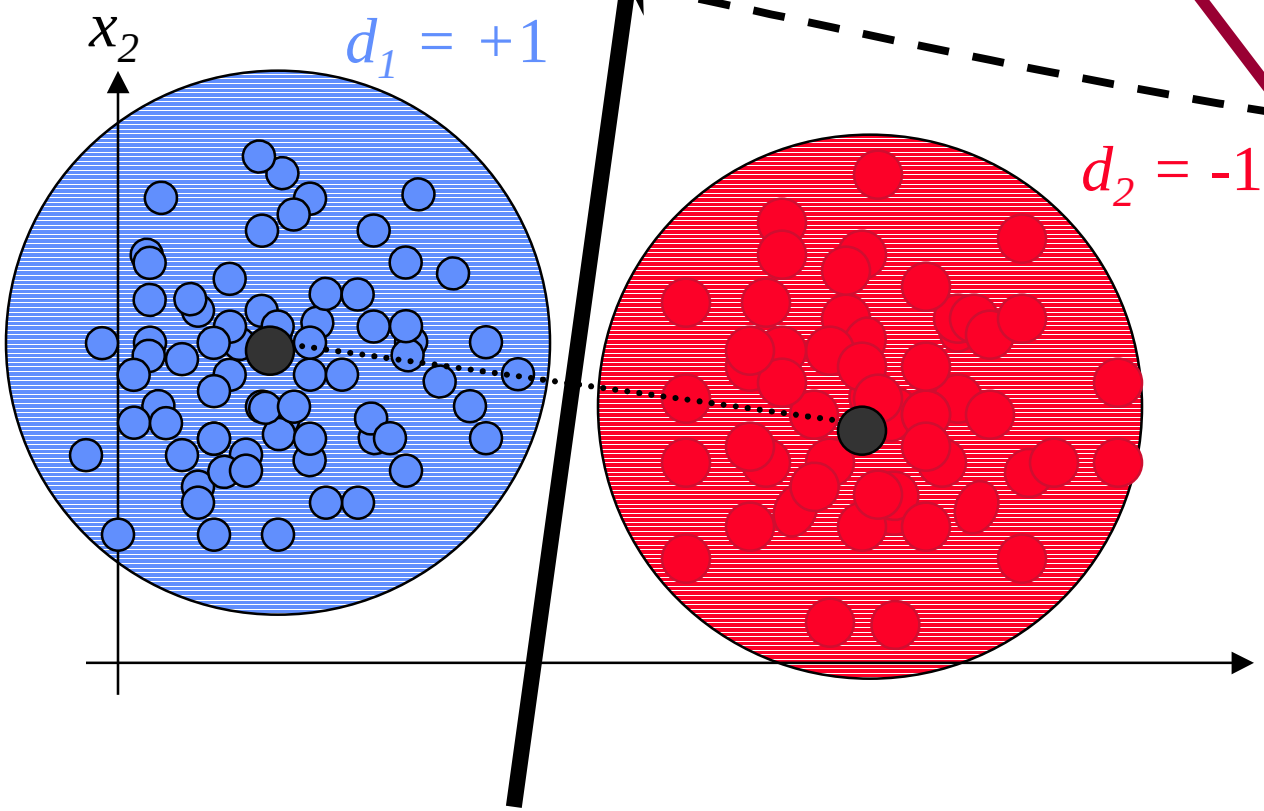
CLASSIFICATION or PATTERN RECOGNITION EXAMPLE

Assume - Normally distributed classes, same covariance matrices. Solution is 'easy' - decision boundary is linear and defined by parameter **w = X* D when there is plenty of data (infinity).** X* denotes the PSEUDOINVERSE.



CLASSIFICATION or PATTERN RECOGNITION EXAMPLE

Assume - Normally distributed classes, same covariance matrices. Solution is 'easy' - decision boundary is linear and defined by parameter **w = X* D when there is plenty of data (infinity).** X* denotes the PSEUDOINVERSE.



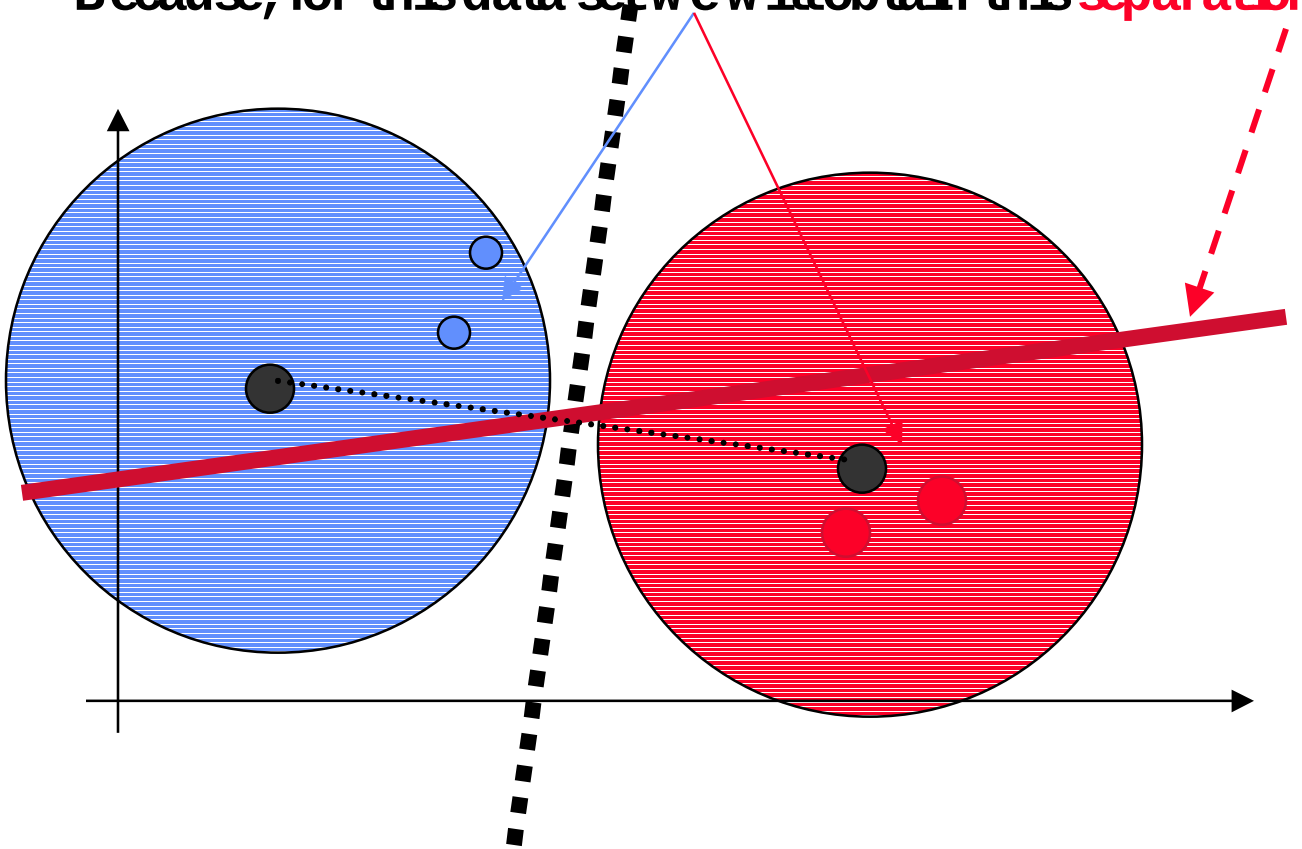
Note that this **solution** follows from the last two assumptions in classical inference!

Gaussian data and minimization of the sum-of-errors-squares!

However, for a small sample -

Solution defined by $w = X^* D$ is **NO LONGER GOOD ONE !!!**

Because, for this data set we will obtain this separation line,

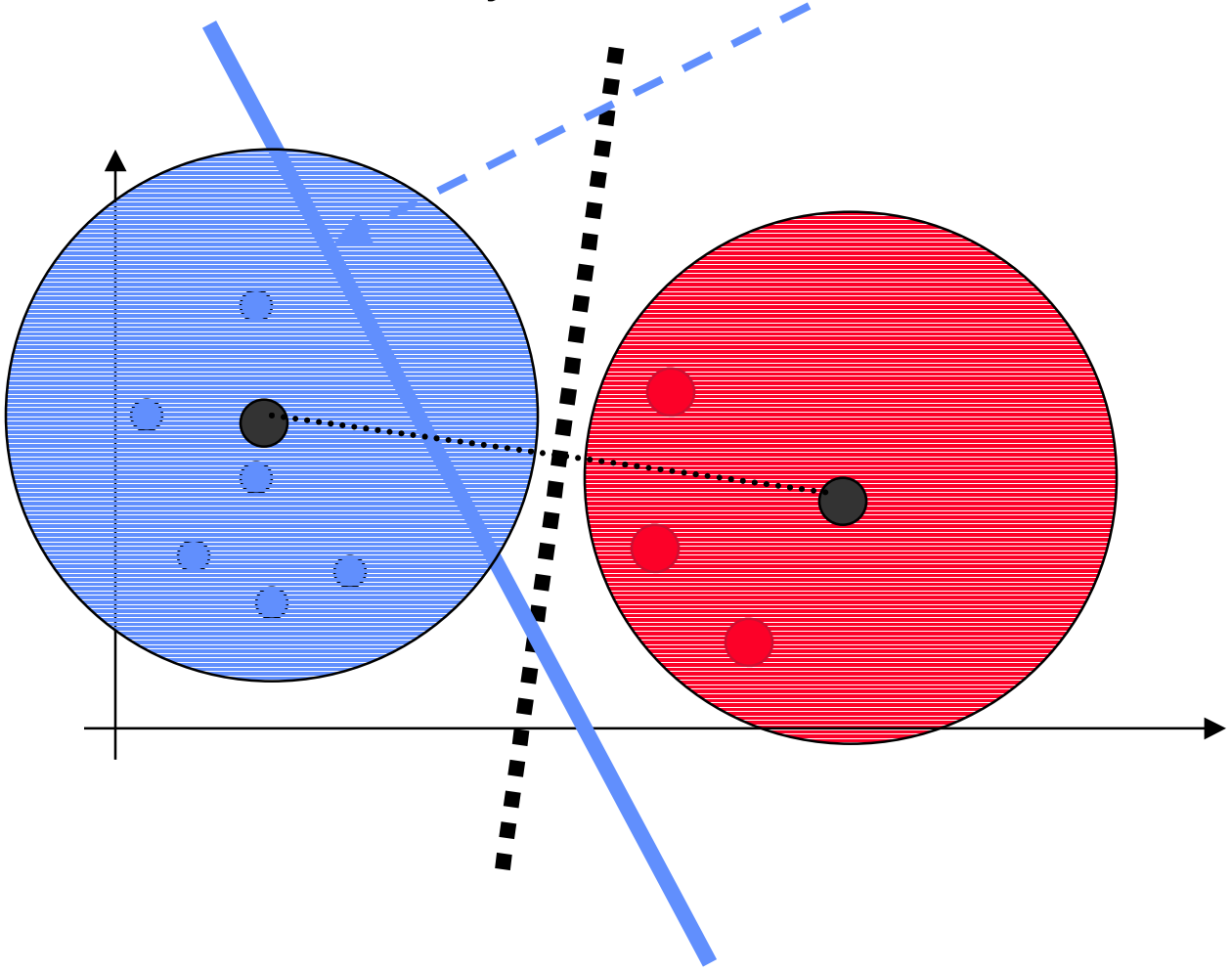


and,

for another data set we will obtain another separation line.

Again, for small sample -

a solution defined by $w = X^{*} D$ is **NO LONGER GOOD ONE !!!**

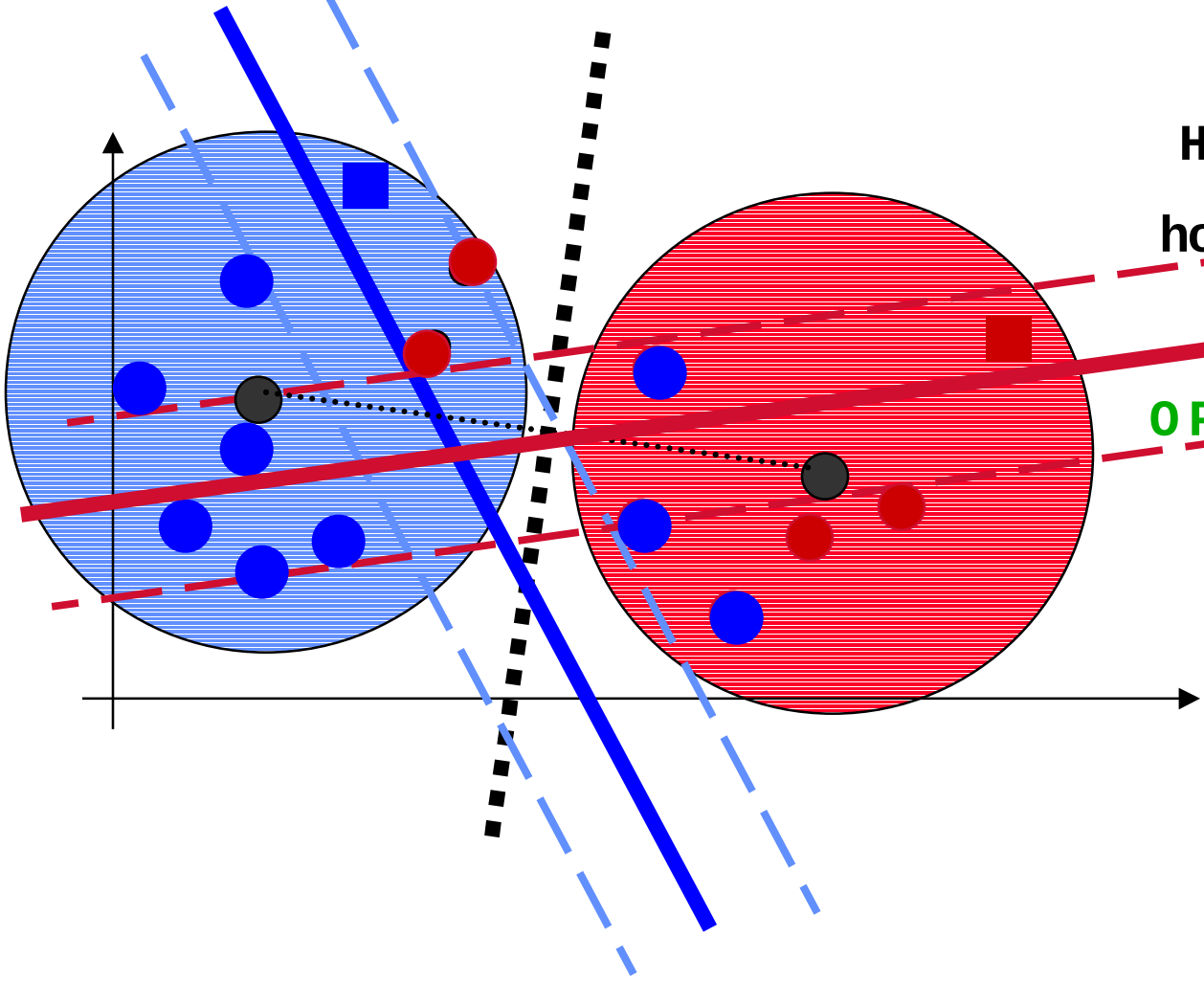


What is common for both separation lines the red and the blue one.

Both have a SMALL MARGIN.

WHAT'S WRONG WITH SMALL MARGIN?

It is very likely that the new examples (,) will be wrongly classified.



However, the question is how to DEFINE and FIND the

OPTIMAL SEPARATION HYPERPLANE

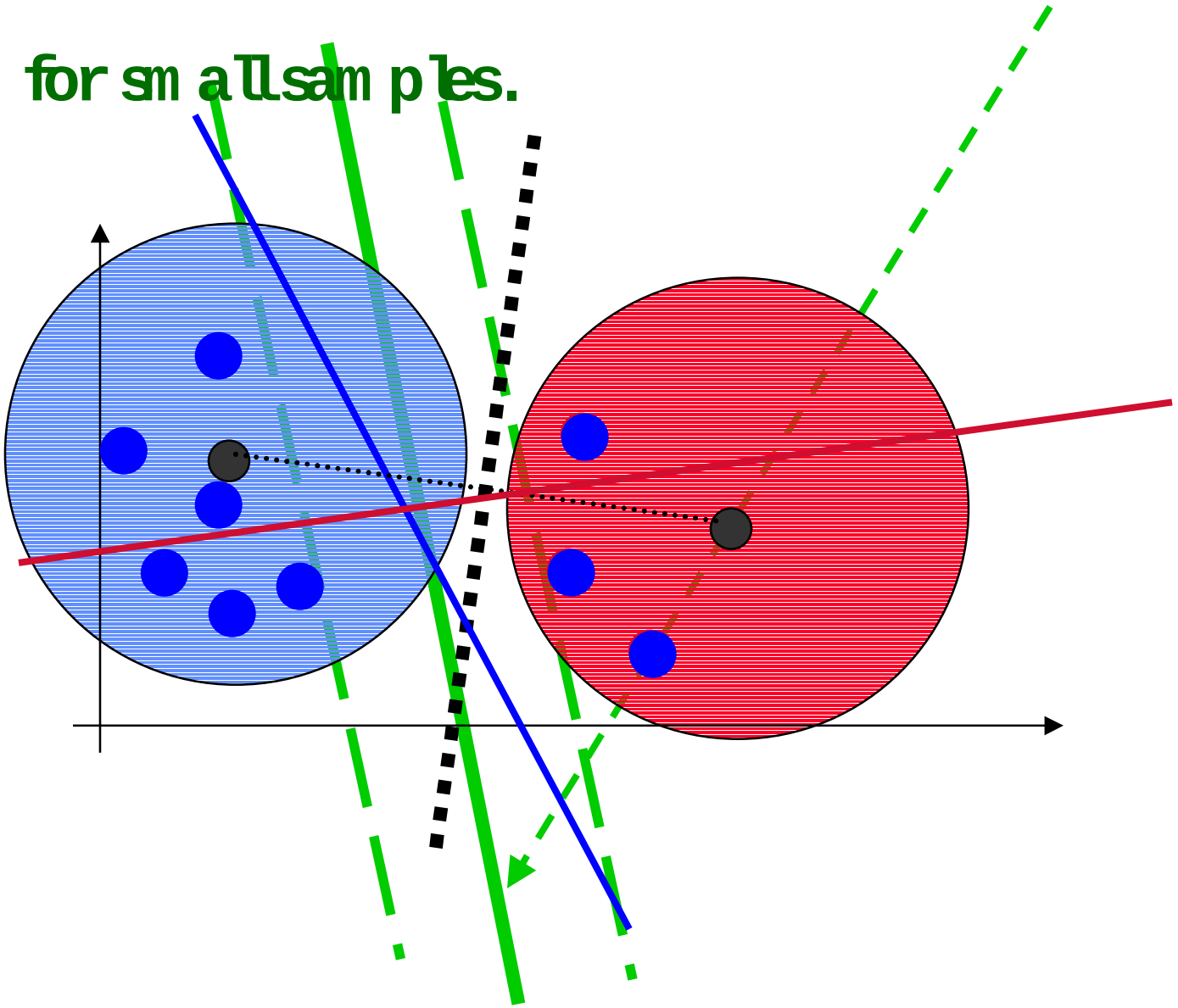
GIVEN (scarce)

DATA SAMPLES???

The **STATISTICAL LEARNING THEORY** IS DEVELOPED TO SOLVE

PROBLEMS of FINDING THE **OPTIMAL SEPARATION HYPERPLANE**

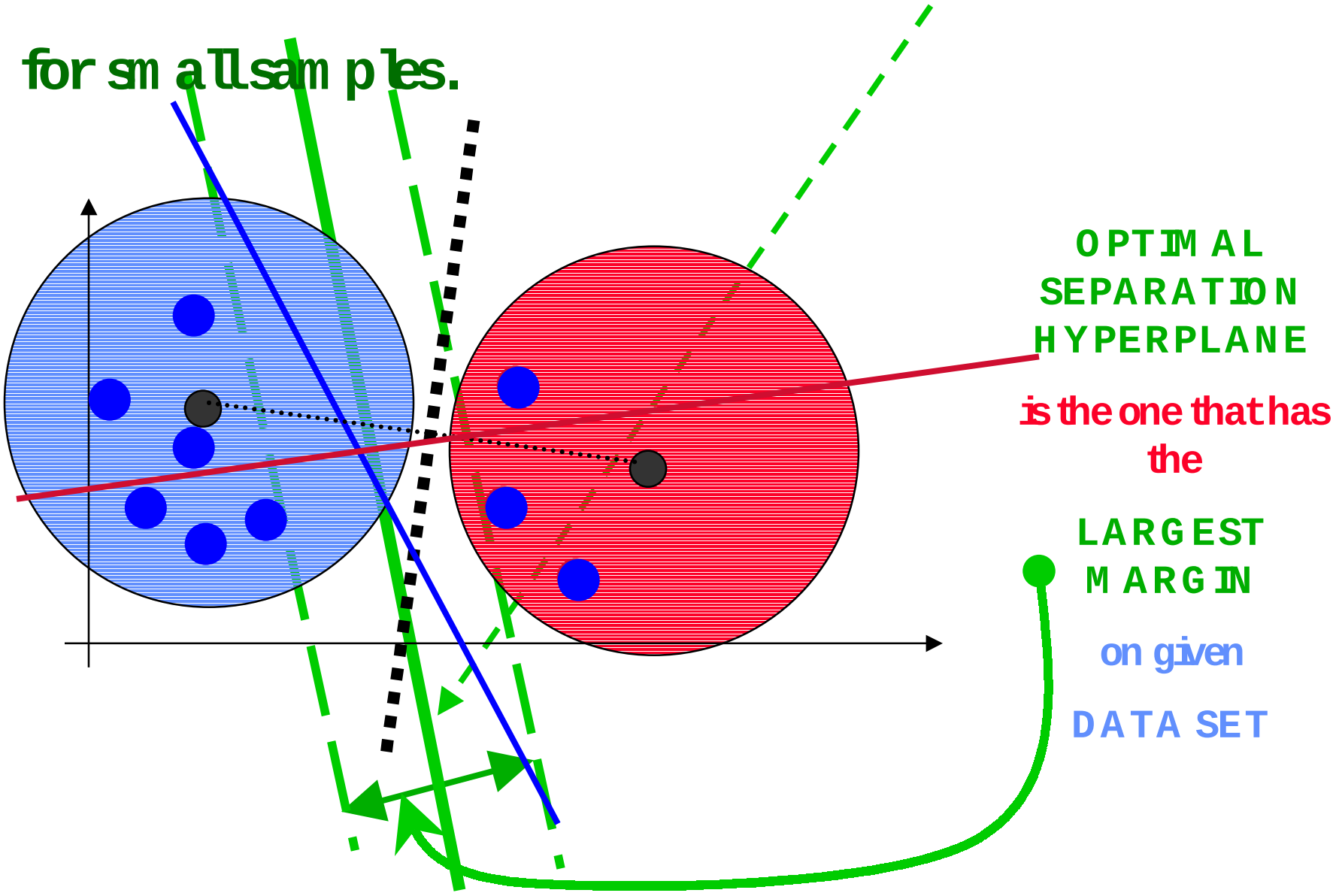
for small samples.



The STATISTICAL LEARNING THEORY IS DEVELOPED TO SOLVE

PROBLEMS of FINDING THE OPTIMAL SEPARATION HYPERPLANE

for small samples.



OPTIMAL SEPARATION HYPERPLANE

is the one that has the

LARGEST MARGIN

on given

DATA SET

Thus, let's show a little more formally,

the constructive part of

the statistical learning theory.

Vapnik and Chervonenkis introduced

a nested set of hypothesis (approximating or decision) functions.

Nested set means that

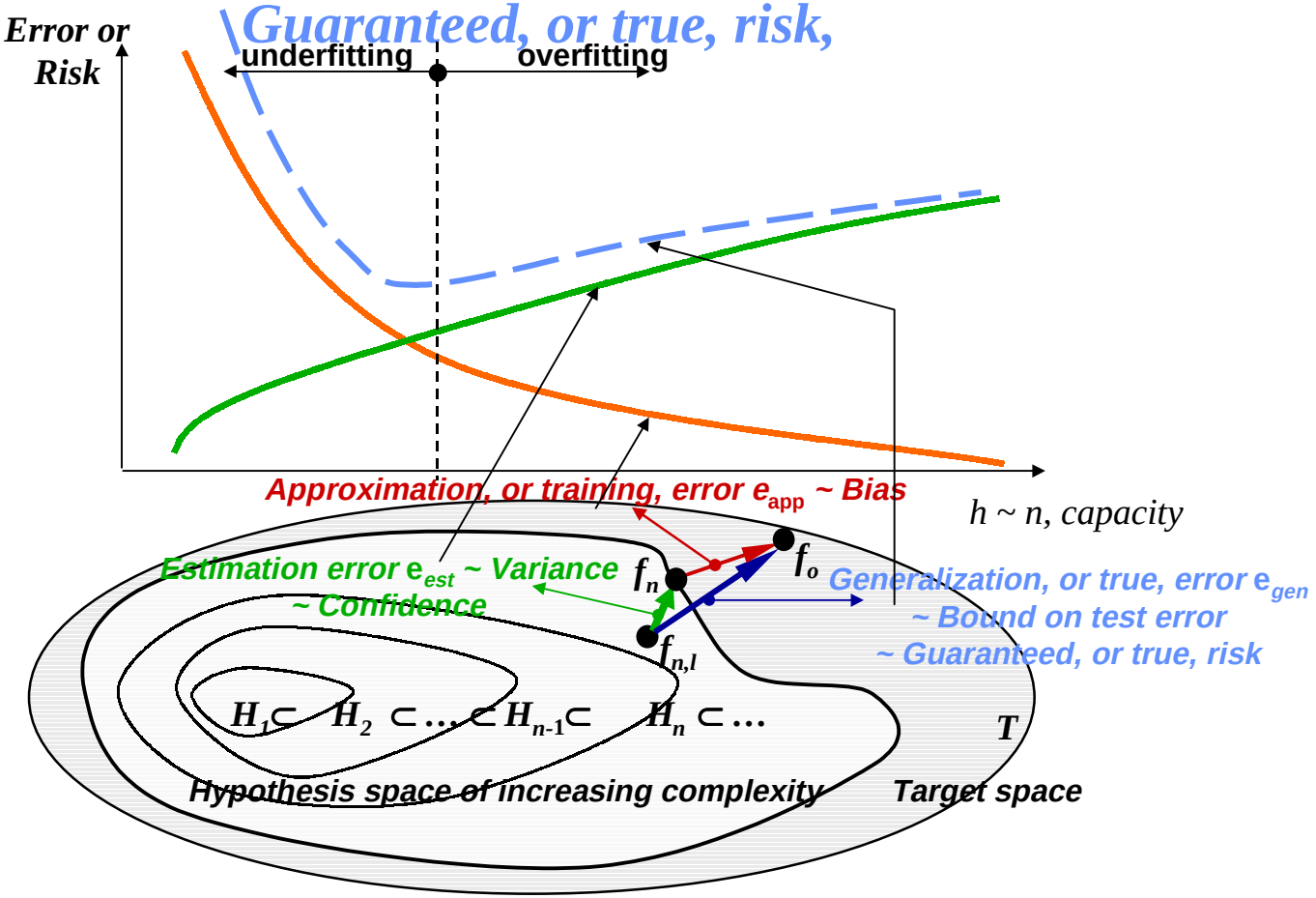
$$H_1 \subset \quad H_2 \subset \ldots \subset H_{n\text{-}1} \subset \quad H_n \subset \ldots$$

Before presenting the basic ideas, clarifications regarding terminology is highly desired because a lot of confusion is caused by a non-unified terminology:

Approximation, or training, error e_{app} ~ Empirical risk ~ Bias,

Estimation error e_{est} ~ Variance ~ Confidence on the training error ~ VC confidence interval,

Generalization (true, expected) error e_{gen} ~ Bound on test error ~ Guaranteed, or true, risk,



Another terminological Rashomons are the concepts of

Decision functions and/or hyperplanes and/or hypersurfaces,

Discriminant functions and/or hyperplanes and/or hypersurfaces,

Decision boundaries, (hyperplanes, hypersurfaces)

Separation lines, functions and/or hyperplanes and/or hypersurfaces,

Input space and feature space used to be the same
and then
SVM developers introduced feature space as the hidden layer or imaginary z-space

Another terminological Rashomons are the concepts of

Decision functions and/or hyperplanes and/or hypersurfaces,

Discriminant functions and/or hyperplanes and/or hypersurfaces,

Decision boundaries, (hyperplanes, hypersurfaces)

Separation lines, functions and/or hyperplanes and/or hypersurfaces,

Input space and feature space used to be the same
and then
SVM developers introduced feature space as the hidden layer or imaginary z-space

The final contributing confusing terminology comes with an indicator function that is basically thresholding function.

Another terminological Rashomons are the concepts of

Decision functions and/or hyperplanes and/or hypersurfaces,

Discriminant functions and/or hyperplanes and/or hypersurfaces,

Decision boundaries, (hyperplanes, hypersurfaces)

Separation lines, functions and/or hyperplanes and/or hypersurfaces,

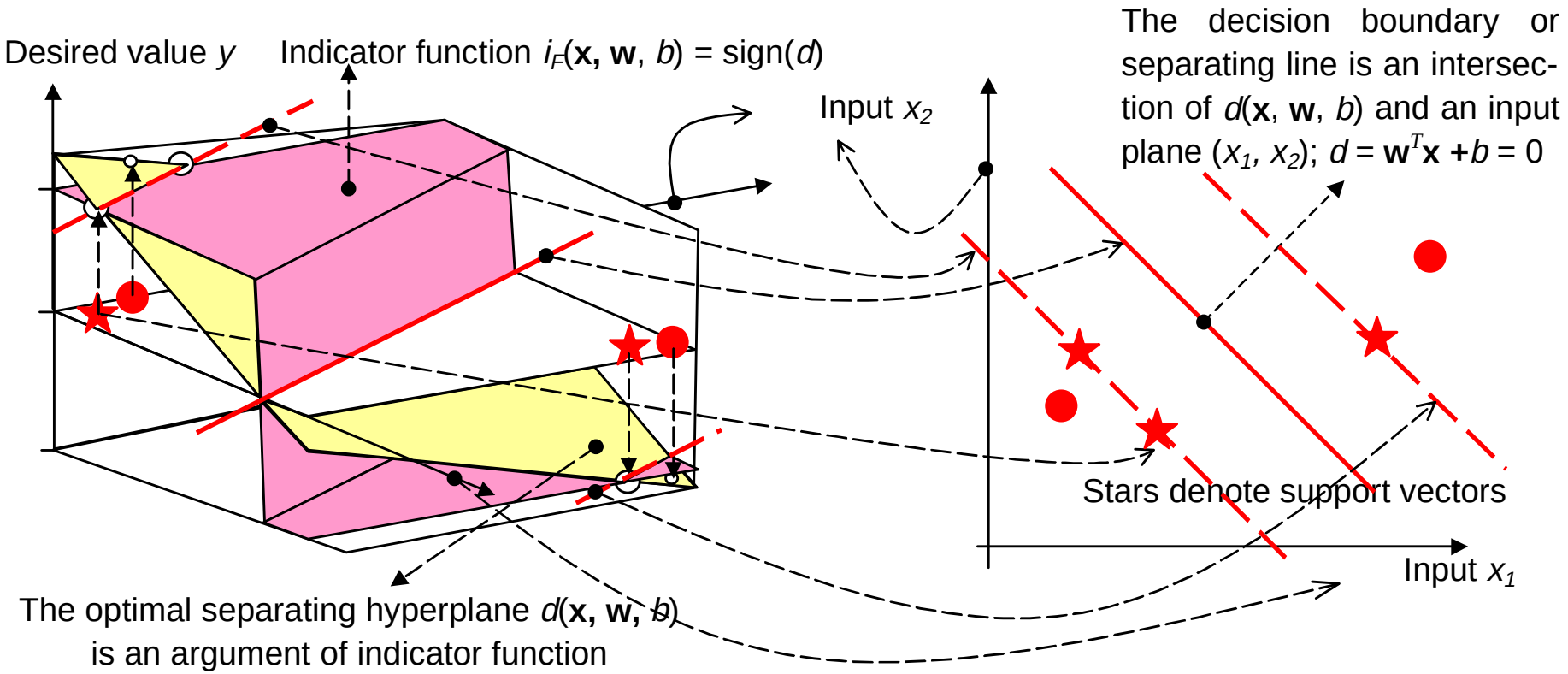
Input space and feature space used to be the same
and then
SVM developers introduced feature space as the hidden layer or imaginary z-space

The final contributing confusing terminology comes with an indicator function that is basically thresholding function.

And we must not forget a CANONICAL HYPERPLANE !

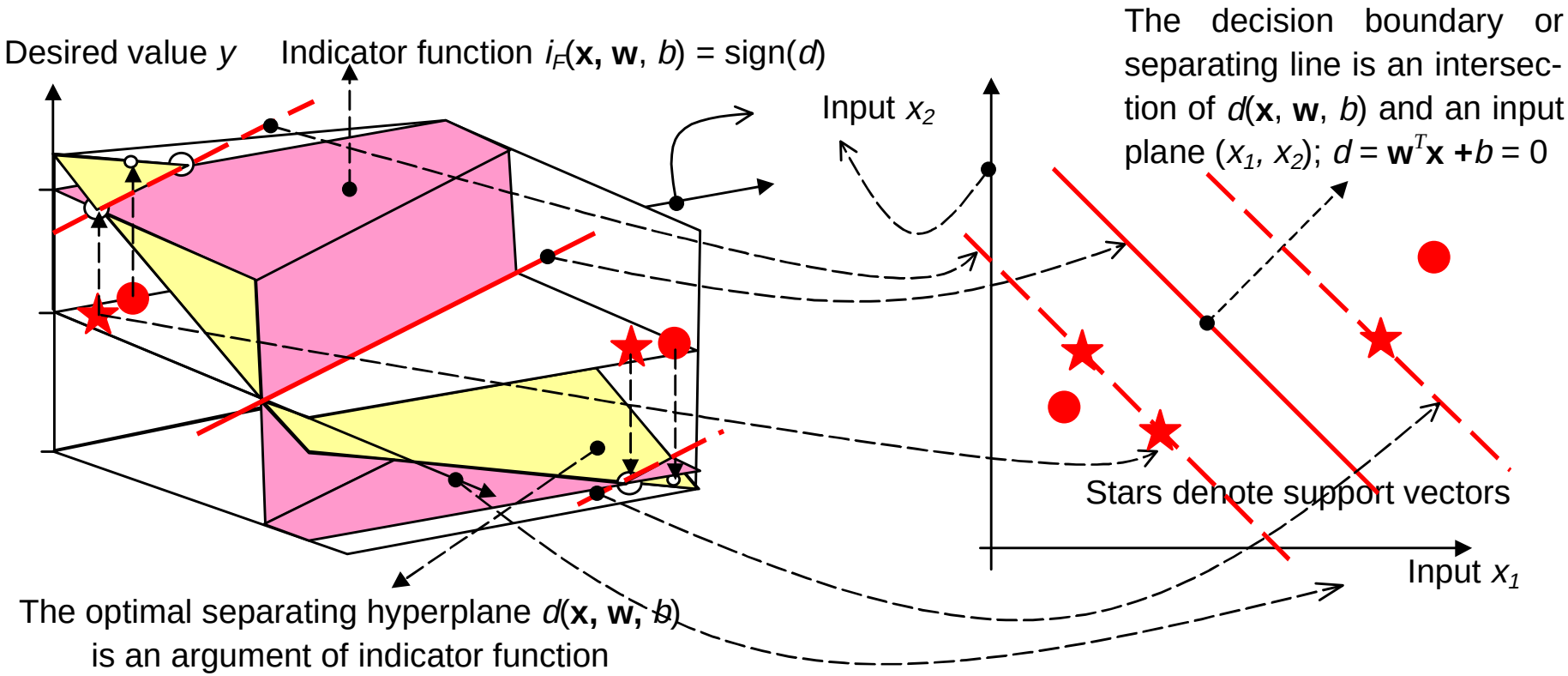
Therefore, in other to reduce a level of confusion at the first-time-readers, i.e., beginners, in the world of SVM, we developed the LEARNSC software that should elucidate all these mathematical objects, their relations and the spaces they are living in.

Thus, the software that can be downloaded from the same site will show the kind of figures below and a little more complex ones.



Therefore, in other to reduce a level of confusion at the first-time-readers, i.e., beginners, in the world of SVM, we developed the LEARNSC software that should elucidate all these mathematical objects, their relations and the spaces they are living in.

Thus, the software that can be downloaded from the same site will show the kind of figures below and a little more complex ones.



That's it! Let's fly hyperspaces!!!

However, before taking-off, a few more ‘similarities’

between NNs and SVMs follow

$$E = \sum_{i=1}^{P} \underbrace{(d_i - f(\mathbf{x}_i, \mathbf{w}))^2}_{Closeness\ to\ data}$$

Classical multilayer perceptron

$$E = \sum_{i=1}^{P} \underbrace{(d_i - f(\mathbf{x}_i, \mathbf{w}))^2}_{Closeness\ to\ data} + \lambda \underbrace{\|\mathbf{P}f\|^2}_{Smoothness}$$

Regularization (RBF) NN

$$E = \sum_{i=1}^{P} \underbrace{(d_i - f(\mathbf{x}_i, \mathbf{w}))^2}_{Closeness\ to\ data} + \underbrace{\Omega(l, h, \eta)}_{Capacity\ of\ a\ machine}$$

Support Vector Machines

In the last expression the SRM principle uses the VC dimension *h* (defining model capacity) as a controlling parameter for minimizing the generalization error E (i.e., risk R).

There are two basic, constructive approaches to the minimization of the right hand side of previous equations (Vapnik, 1995 and 1998):

-choose an appropriate structure (order of polynomials, number of HL neurons, number of rules in the FL model) and, keeping the confidence interval fixed in this way, minimize the training error (i.e., empirical risk), or

-keep the value of the training error fixed (equal to zero or equal to some acceptable level) and minimize the confidence interval.

Classical NNs implement the first approach (or some of its sophisticated variants) and SVMs implement the second strategy.
In both cases the resulting model should resolve the trade-off between under-fitting and over-fitting the training data.
The final model structure (order) should ideally
***match the learning machines capacity with training data complexity*.**

O.K.

Let us do some more

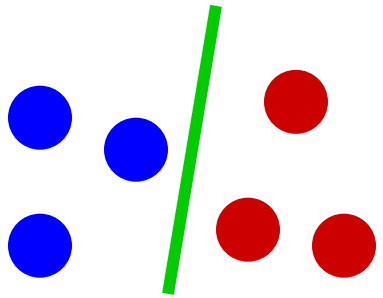
formal,

meaning,

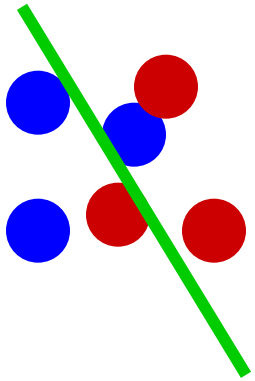
mathematical analysis of SVMs learning!

The presentation will follow an idea of a **gentle introduction, i.e., of a gradual proceeding** from the 'simple' cases to the more complex and involved ones!

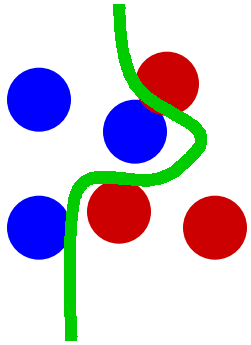
1) Linear Maximal Margin Classifier for Linearly Separable Data - no samples overlapping.

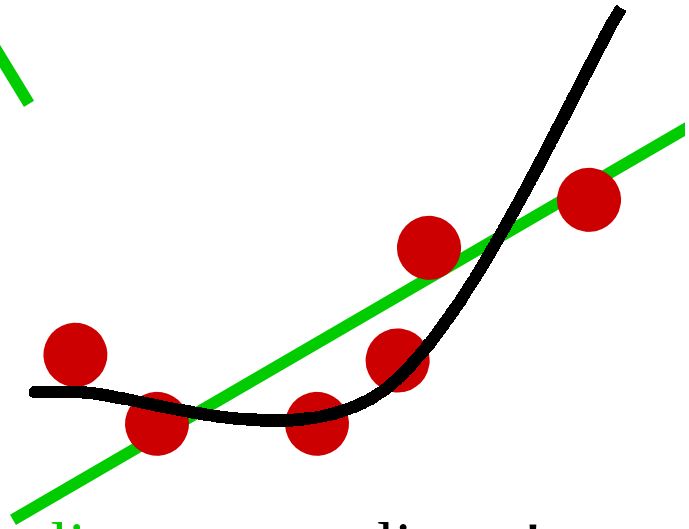


2) Linear Soft Margin Classifier for Overlapping Classes.



3) Nonlinear Classifier.





4) Regression by SV Machines that can be either linear or nonlinear!

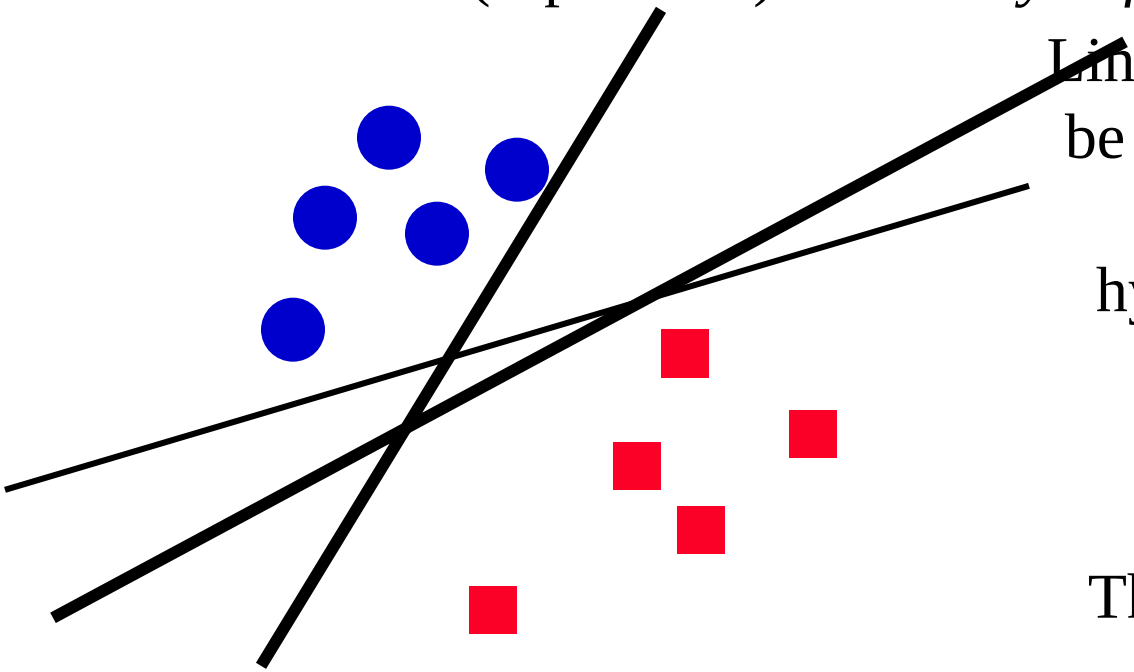
1) Linear Maximal Margin Classifier for Linearly Separable Data

Binary classification - no samples overlapping

Given some training data

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), \qquad y_i \in \{-1, +1\}$$

find the function $f(\mathbf{x}, \mathbf{w}_0) \in f(\mathbf{x}, \mathbf{w})$ which best approximates the unknown discriminant (separation) function $y = f(\mathbf{x})$.



Linearly separable data can be separated by in infinite number of linear hyperplanes that can be written as

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\mathbf{x} + b$$

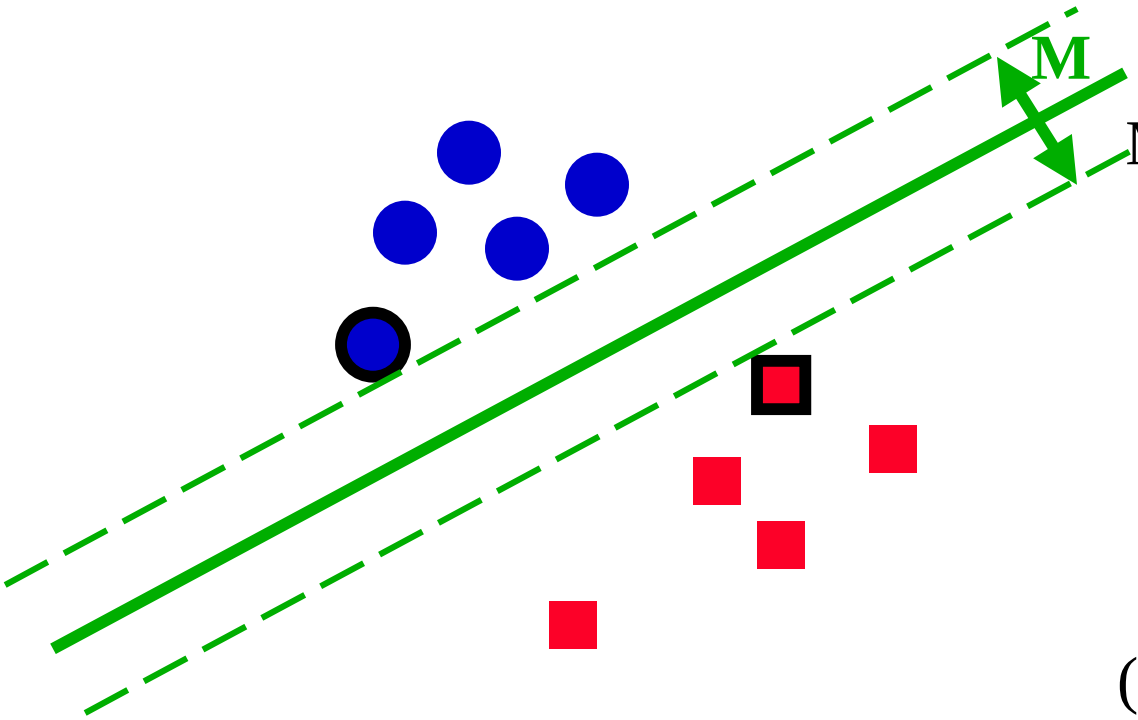
The problem is: find the optimal separating hyperplane

1) Optimal separating hyperplane is the one with

MAXIMAL MARGIN !

This hyperplane is uniquely determined by the vectors on the margin

the support vectors!



MARGIN IS DEFINED by **w** as follows:

$$M = \frac{2}{\|\mathbf{w}\|}$$

(Vapnik, Chervonenkis ‘74)

The optimal canonical separating hyperplane (OCSH), i.e., a separating hyperplane with the largest margin (defined by $M = 2 / \|\mathbf{w}\|$) specifies *support vectors*, i.e., training data points closest to it, which satisfy $y_j[\mathbf{w}^T\mathbf{x}_j + b] \equiv 1$, $j = 1, N_{SV}$. At the same time, the OCSH must separate data correctly, i.e., it should satisfy inequalities

$$y_i[\mathbf{w}^T\mathbf{x}_i + b] \geq 1, \qquad i = 1, l$$

where l denotes a number of training data and N_{SV} stands for a number of SV.

Note that maximization of M means a minimization of $\|\mathbf{w}\|$. Minimization of a norm of a hyperplane normal weight vector $\|\mathbf{w}\| = \sqrt{\mathbf{w}^T\mathbf{w}} = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}$ leads to a maximization of a margin M. Because $sqrt(f)$ is a monotonic function, its minimization is equivalent to a minimization of f.

Consequently, a minimization of norm $\|\mathbf{w}\|$ equals a minimization of

$$\mathbf{w}^T\mathbf{w} = w_1^2 + w_2^2 + \ldots + w_n^2$$

and this leads to a maximization of a margin M.

Thus the problem to solve is:

minimize

$$J = \mathbf{w}^T \mathbf{w} = \| \mathbf{w} \|^2$$

subject to constraints

$$y_i[\mathbf{w}^T \mathbf{x}_i + b] \geq 1$$

and this is a classic QP problem with constraints that ends in forming and solving of a primal and/or dual Lagrangian.

Thus the problem to solve is:

minimize

Margin maximization!

$$J = \mathbf{w}^T \mathbf{w} = \| \mathbf{w} \|^2$$

subject to constraints

Correct classification!

$$y_i[\mathbf{w}^T \mathbf{x}_i + b] \geq 1$$

and this is a classic QP problem with constraints that ends in forming and solving of a primal and/or dual Lagrangian.

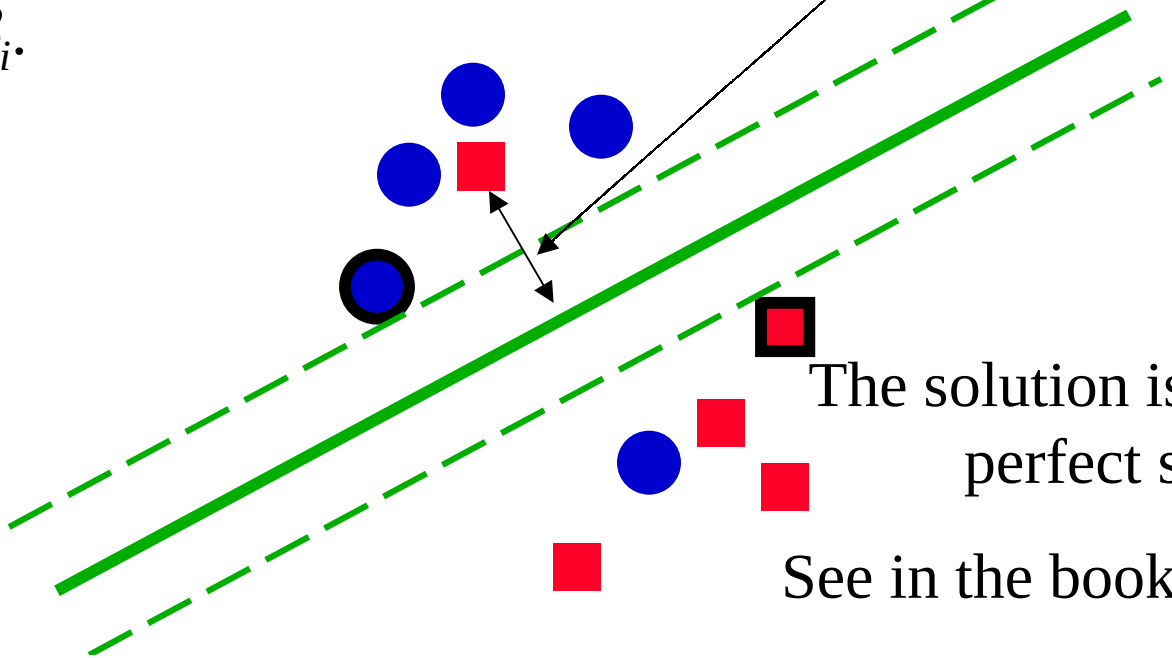
2) Linear Soft Margin Classifier for ***Overlapping Classes***

Now one minimizes: $J(\mathbf{w},\xi)=\frac{1}{2}\mathbf{w}^T\mathbf{w}+C(\sum_{i=1}^{l}\xi_i)^k$

s.t. $\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \geq +1 - \xi_i$, for $y_i = +1$,

$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \leq -1 + \xi_i$, for $y_i = -1$.

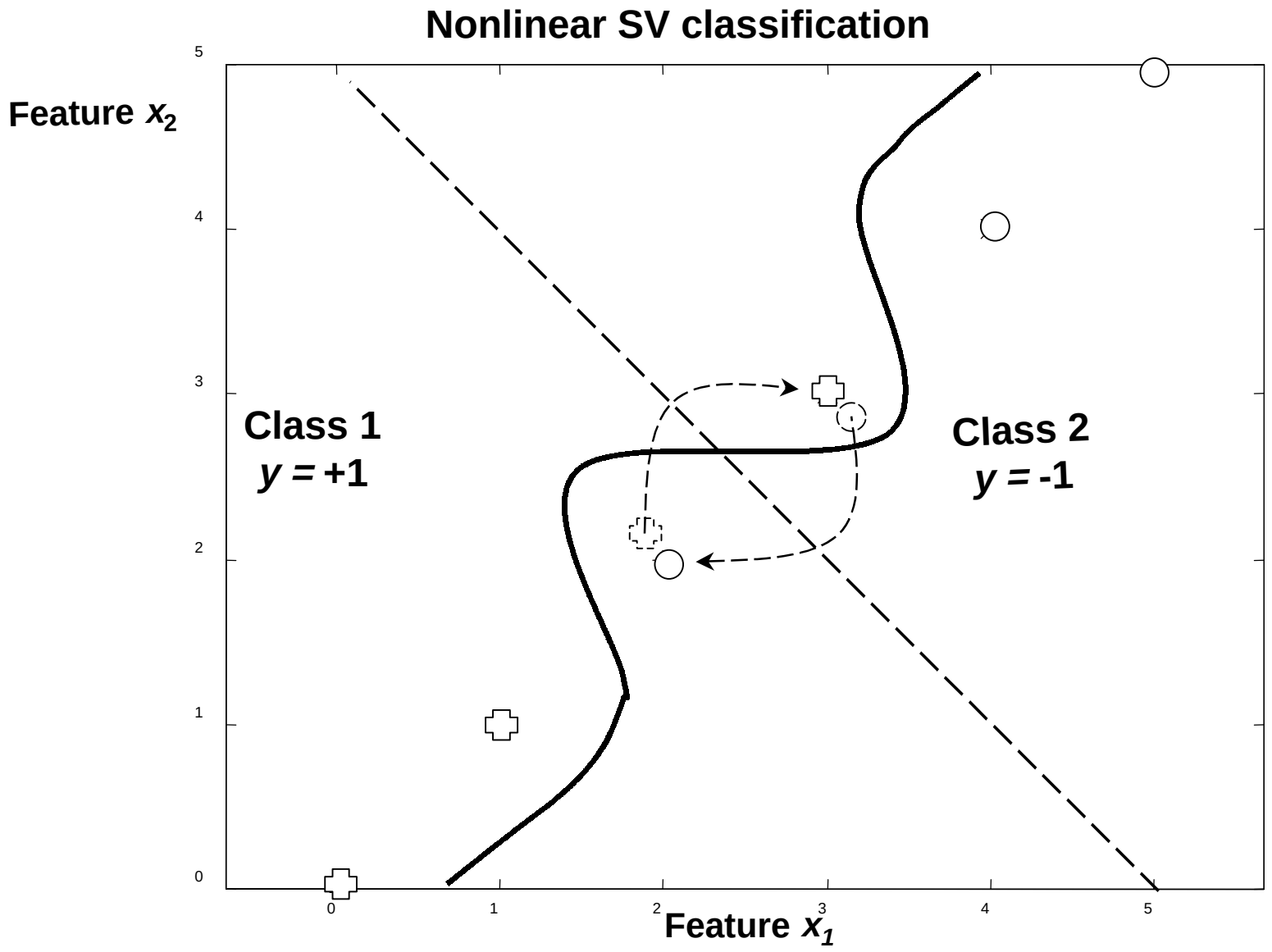
The problem is no longer convex and the solution is given by the saddle point of the primal Lagrangian $L_p(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ where α_i and β_i are the Lagrange multipliers. Again, we should find an *optimal* saddle point ($\mathbf{w}_o$, b_o, $\boldsymbol{\xi}_o$, $\boldsymbol{\alpha}_o$, $\boldsymbol{\beta}_o$) because the Lagrangian L_p has to be *minimized* with respect to $\mathbf{w}$, b and $\boldsymbol{\xi}$, and *maximized* with respect to nonnegative α_i and β_i.



The solution is a **hyperplane again**. No perfect separation however!

See in the book the details of the solution!

However, the hyperplanes cannot be the solutions when the decision boundaries are nonlinear.



Now, the SVM should be constructed by

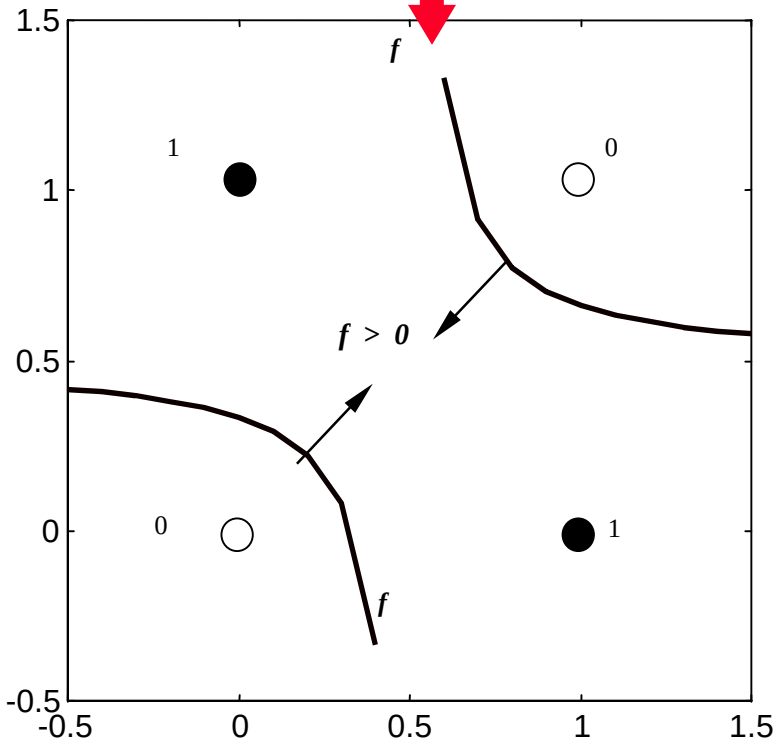
i) mapping input vectors nonlinearly into a high dimensional feature space and,

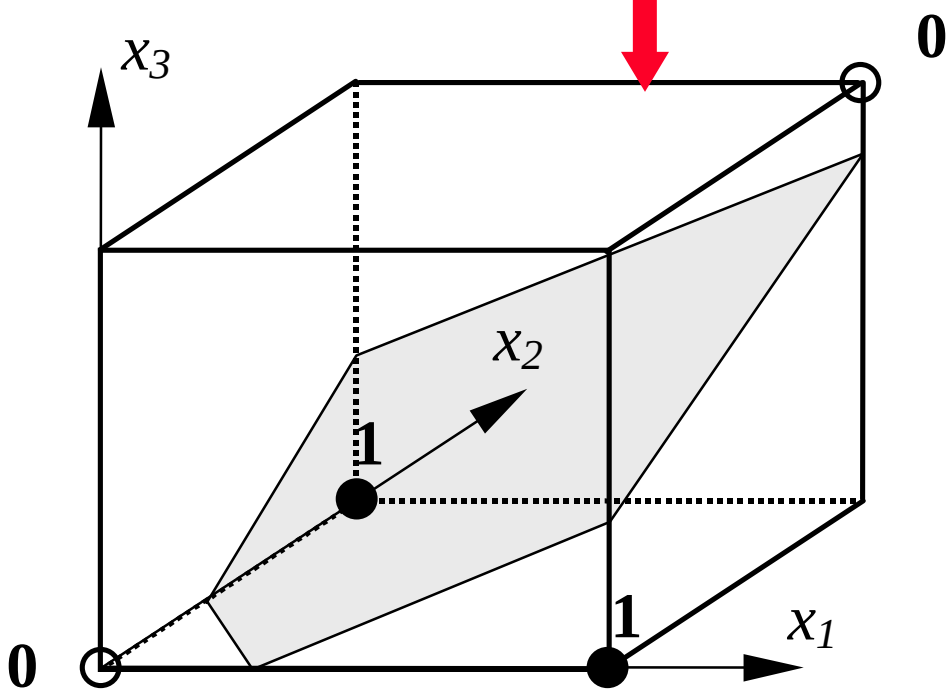
ii) by constructing the OCSH in the high dimensional feature space.

Let's show this mapping into feature space on a classic XOR (nonlinearly separable) problem!

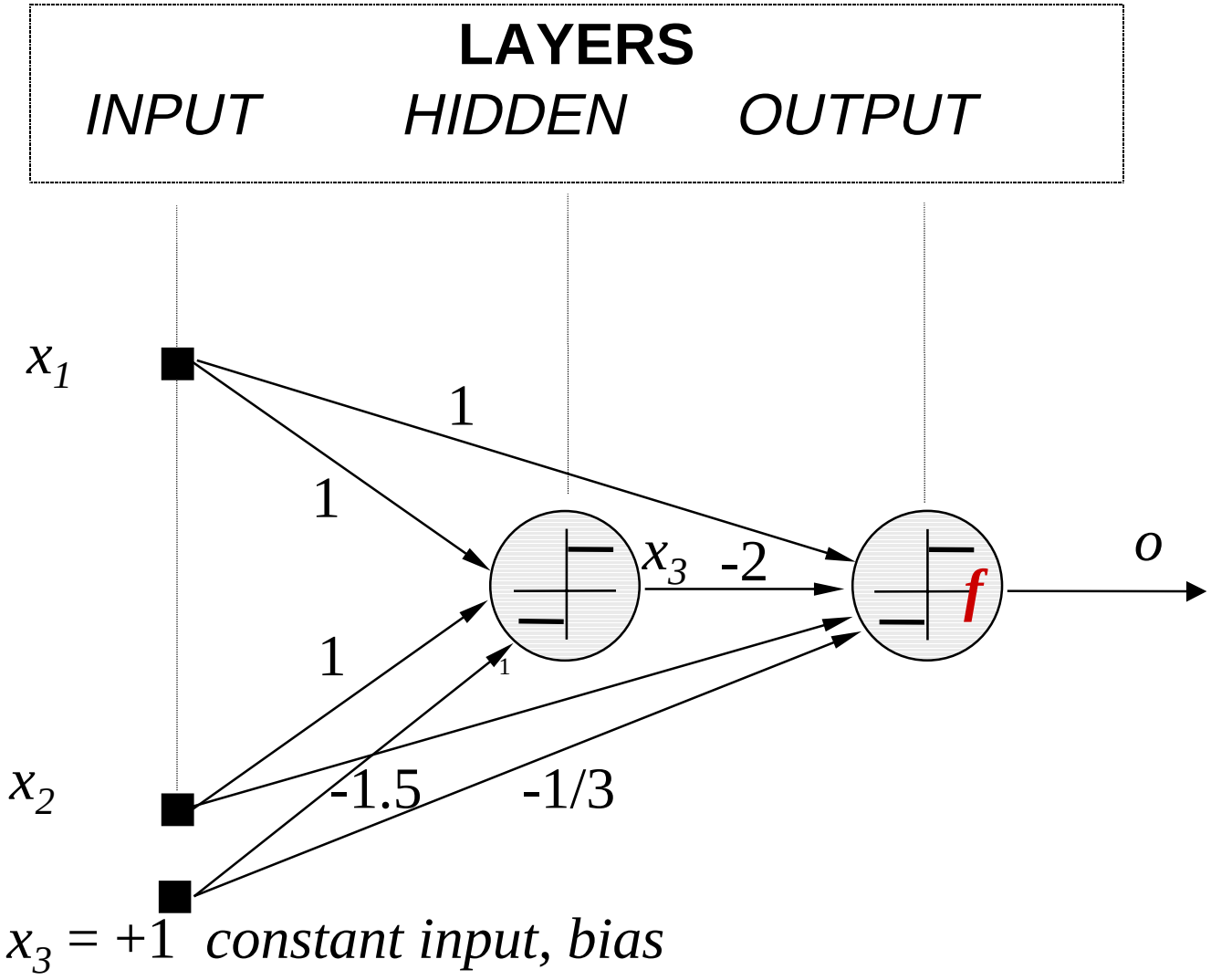
Many different **nonlinear discriminant functions** that separate 1-s from 0-s can be drawn in a feature plane. Suppose, the following one

$f(\mathbf{x}) = x_1 + x_2 - 2x_1x_2 - 1/3$, $x_3 = x_1x_2$, $f(\mathbf{x}) = x_1 + x_2 - 2x_3 - 1/3$

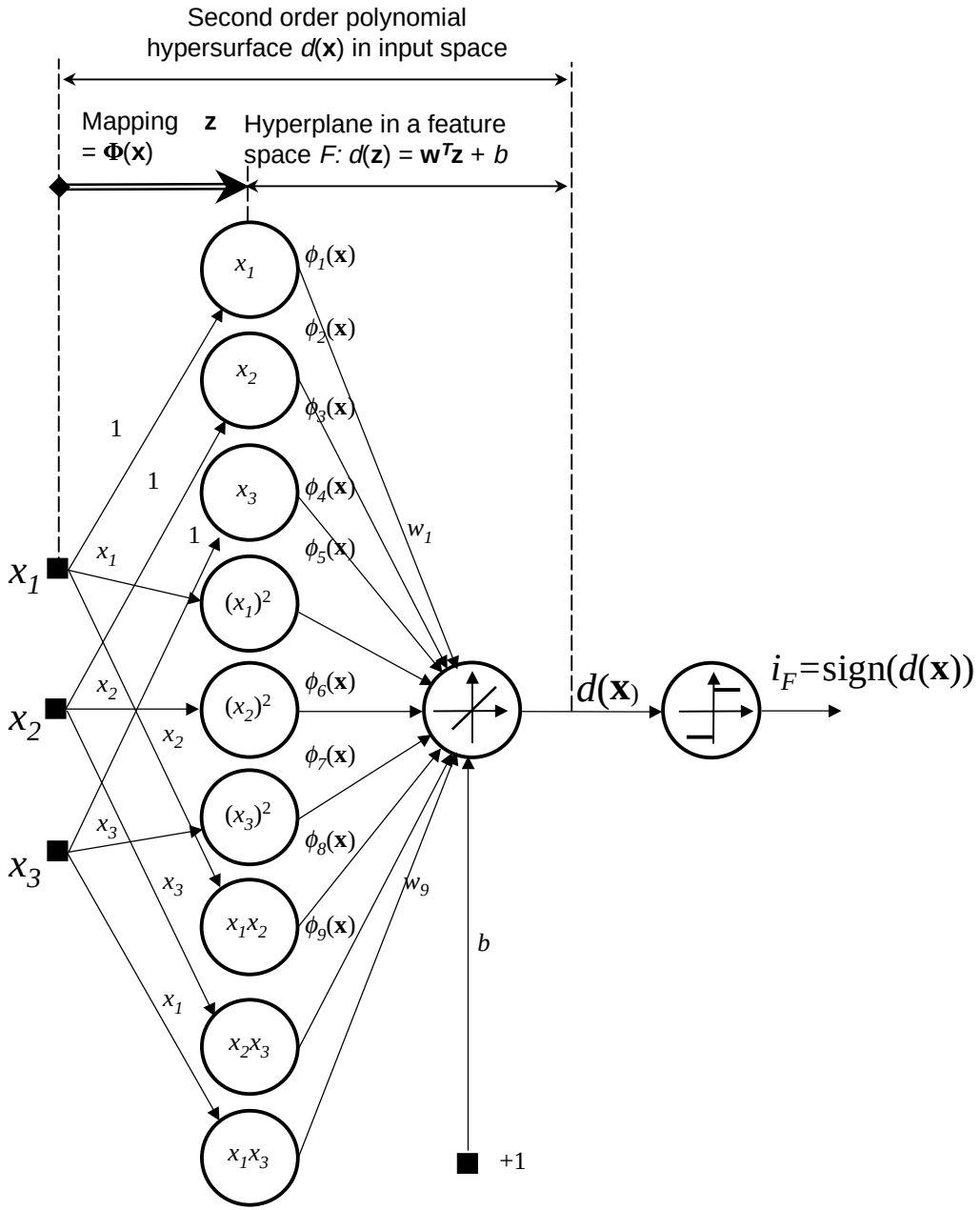




The last plane, in a 3-dimensional feature space, will be produced by the following NN structure.



SVMs arise from mapping input vectors $\mathbf{x} = [x_1\ x_2\ \ldots\ x_n]^T$ into feature vectors $\mathbf{z} = \Phi(\mathbf{x})$.



Now, we apply a '**kernel trick**'.

One basic idea in designing nonlinear SV machines is to map input vectors $\mathbf{x} \in \Re^n$ into vectors $\mathbf{z}$ of a higher dimensional *feature space* $F(\mathbf{z}) = \mathbf{\Phi}(\mathbf{x})$ where $\mathbf{\Phi}$ represents mapping: $\Re^n \rightarrow \Re^f$ and to

solve a linear classification problem in this feature space

$$\mathbf{x} \in \Re^n \rightarrow \mathbf{z}(\mathbf{x}) = [a_1\phi_1(\mathbf{x}), a_2\phi_2(\mathbf{x}), \ldots, a_f\phi_f(\mathbf{x})]^T \in \Re^f$$

The solution for an indicator function $i_F(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{z}(\mathbf{x}) + b)$, which is a linear classifier in a feature space F, will create a nonlinear separating hypersurface in the original input space given by

$$i_F(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{l} \alpha_i y_i \mathbf{z}^T(\mathbf{x})\mathbf{z}(\mathbf{x}_i) + b\right)$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{z}_i^T\mathbf{z}_j = \mathbf{\Phi}^T(\mathbf{x}_i)\mathbf{\Phi}(\mathbf{x}_j).$$

Note that a *kernel function* $K(\mathbf{x}_i, \mathbf{x}_j)$ is a function in input space.

Kernel functions	**Type of classifier**
$K(\mathbf{x}, \mathbf{x}_i) = [(\mathbf{x}^T\mathbf{x}_i) + 1]^d$	Polynomial of degree d
$K(\mathbf{x}, \mathbf{x}_i) = e^{-\frac{1}{2}[(\mathbf{x}-\mathbf{x}_i)^T \Sigma^{-1}(\mathbf{x}-\mathbf{x}_i)]}$	Gaussian RBF
$K(\mathbf{x}, \mathbf{x}_i) = \tanh[(\mathbf{x}^T\mathbf{x}_i) + b]^*$	Multilayer perceptron

*only for certain values of b

The learning procedure is the same as the construction of a ‘hard’ and ’soft’ margin classifier in **x**-space previously.

Now, in **z**-space, the dual Lagrangian that should be maximized is

$$L_d(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,\, j=1}^{l} y_i y_j \alpha_i \alpha_j \mathbf{z}_i^T \mathbf{z}_j$$

or,

$$L_d(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,\, j=1}^{l} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$

and the constraints are

$$\alpha_i \geq 0, \qquad i = 1, l$$

In a more general case, because of noise or generic class' features, there will be an overlapping of training data points. Nothing but constraints change as for the soft margin classifier above. Thus, the nonlinear 'soft' margin classifier will be the solution of the quadratic optimization problem given above subject to constraints

$$C \geq \alpha_i \geq 0, \qquad i = 1, l \qquad \text{and} \qquad \sum_{i=1}^{l} \alpha_i y_i = 0$$

The decision hypersurface is given by

$$d(\mathbf{x}) = \sum_{i=1}^{l} y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b$$

We see that the final structure of the SVM is equal to the NN model.
In essence it is a weighted linear combination of some kernel (basis) functions. We'll show this (hyper)surfaces in simulations later.

Regression by SVMs

Initially developed for solving classification problems, SV techniques can be successfully applied in regression, i.e., for a functional approximation problems (Drucker et al, (1997), Vapnik et al, (1997)).

Unlike pattern recognition problems (where the desired outputs y_i are discrete values e.g., Boolean), here we deal with *real valued* functions.

Now, the general regression learning problem is set as follows;

the learning machine is given l training data from which it attempts to learn the input-output relationship (dependency, mapping or function) $f(\mathbf{x})$.

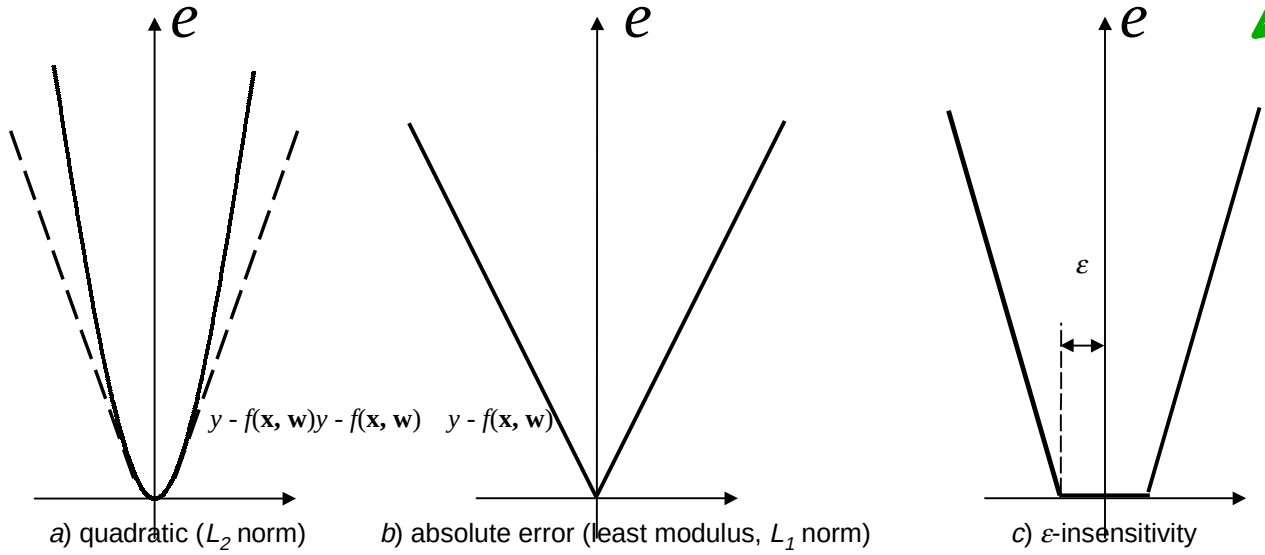
A training data set $D = \{[\mathbf{x}(i), y(i)] \in \Re^n \times \Re, i = 1,...,l\}$ consists of l pairs $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)$, where the inputs $\mathbf{x}$ are n-dimensional vectors $\mathbf{x} \in \Re^n$ and system responses $y \in \Re$, are continuous values. The SVM considers approximating functions of the form

$$f(\mathbf{x}, \mathbf{w}) = \sum^{N} w_i \varphi_i(\mathbf{x})$$

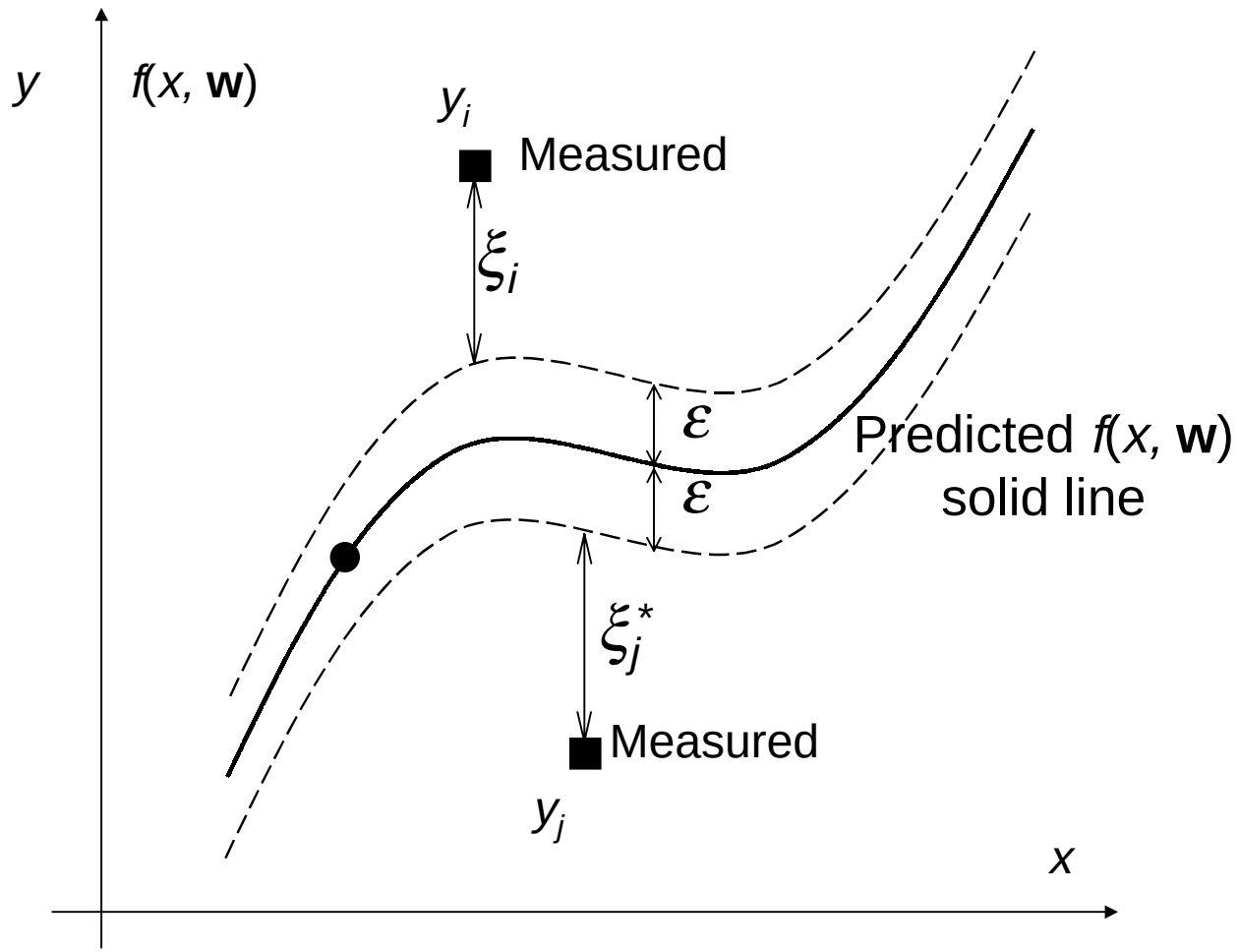
Vapnik introduced a more general error (loss) function - the so-called **ε-insensitivity loss function**

$$|\, y - f(\mathbf{x}, \mathbf{w}) \,|_{\varepsilon} = \begin{cases} 0 & \text{if } |\, y - f(\mathbf{x}, \mathbf{w}) \,| \le \varepsilon \\ |\, y - f(\mathbf{x}, \mathbf{w}) \,| - \varepsilon, & \text{otherwise.} \end{cases}$$

Thus, the loss is equal to 0 if the difference between the predicted $f(\mathbf{x}, \mathbf{w})$ and the measured value is less than ε. Vapnik's ε-insensitivity loss function **defines an ε tube** around $f(\mathbf{x}, \mathbf{w})$. If the predicted value is within the tube the loss (error, cost) is zero. For all other predicted points outside the tube, the loss equals the magnitude of the difference between the predicted value and the radius ε of the tube. **See the next figure.**



a) quadratic (L_2 norm) *b*) absolute error (least modulus, L_1 norm) *c*) ε-insensitivity



The parameters used in (1-dimensional) support vector regression.

Now, minimizing risk R equals

$$R_{\mathbf{w},\xi,\xi^*} = \left[\frac{1}{2}\|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{l}\xi + \sum_{i=1}^{l}\xi^*\right)\right]$$

and the constraints are,

$y_i - \mathbf{w}^T\mathbf{x}_i - b \le \varepsilon + \xi$, $\quad i = 1, l,$

$\mathbf{w}^T\mathbf{x}_i + b - y_i \le \varepsilon + \xi^*$, $\quad i = 1, l,$

$\xi \ge 0$ $\quad i = 1, l,$

$\xi^* \ge 0$ $\quad i = 1, l,$

where ξ and ξ^* are slack variables shown in previous figure for measurements **'above'** and **'below'** an ε-tube respectively. Both slack variables are positive values. Lagrange multipliers (that will be introduced during the minimization) α_i and α_i^* corresponding to ξ and ξ^* will be nonzero values for training points 'above' and 'below' an ε-tube respectively. Because no training data can be on both sides of the tube, either α_i or α_i^* will be nonzero. For data points inside the tube, both multipliers will be equal to zero.

Similar to procedures applied to SV classifiers, we solve this constrained optimization problem by forming a *primal variables Lagrangian* $L_p(\mathbf{w}, \xi, \xi^*)$

$$L_p(\mathbf{w}, b, \xi, \xi^*, \alpha_i, \alpha_i^*, \beta_i, \beta_i^*) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\left(\sum_{i=1}^{l}\xi + \sum_{i=1}^{l}\xi^* - \sum_{i=1}^{l}\alpha_i^*\left[y_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i - b + \varepsilon + \xi_i^*\right.\right.$$

$$\left.\left. - \sum_{i=1}^{l}\alpha_i\left[\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b - y_i + \varepsilon + \xi_i\right] - \sum_{i=1}^{l}(\beta_i^*\xi_i^* + \beta_i\xi_i)\right.\right.$$

A primal variables Lagrangian $L_p(w_i, b, \xi, \xi^*, \alpha, \alpha^*, \beta, \beta^*)$ has to be *minimized* with respect to primal variables **w**, b, ξ and ξ^* and *maximized* with respect to nonnegative LaGrange multipliers α, α^*, β and β^*. This problem can be solved again either in a ***primal space*** or in a ***dual*** one. Below, we consider a solution in a dual space. Applying Karush-Kuhn-Tucker (KKT) conditions for regression, we will *maximize a dual variables Lagrangian* $L_d(\alpha, \alpha^*)$

$$L_d(\alpha, \alpha^*) = -\varepsilon\sum_{i=1}^{l}(\alpha_i^* + \alpha_i) + \sum_{i=1}^{l}(\alpha_i^* - \alpha_i)\,y_i - \frac{1}{2}\sum_{i,\,j=1}^{l}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)\mathbf{x}_i^T\mathbf{x}_j$$

subject to constraints

$$\sum_{i=1}^{l} \alpha_i^* = \sum_{i=1}^{l} \alpha_i$$

$$0 \le \alpha_i^* \le C \qquad i = 1, l,$$

$$0 \le \alpha_i \le C \qquad i = 1, l.$$

Note that **a dual variables Lagrangian $L_d(\alpha, \alpha^*)$ is expressed in terms of LaGrange multipliers α and α^* only**, and that - the size of the problem, with respect to the size of an SV classifier design task, is doubled now.

There are $2l$ unknown multipliers for linear regression and the Hessian matrix **H** of the quadratic optimization problem in the case of regression is a $(2l, 2l)$ matrix.

The ***standard quadratic optimization problem*** above can be expressed in a ***matrix notation*** and formulated as follows:

Maximize

$$L_d(\alpha) = -0.5\alpha^T \mathbf{H}\, \alpha + \mathbf{f}^T \alpha,$$

subject to constraints above where for a **linear regression**,

$$\mathbf{H} = [\mathbf{x}^T\mathbf{x} + 1], \mathbf{f} = [\varepsilon - y_1\ \varepsilon - y_2, \ldots, \varepsilon - y_N, \varepsilon + y_1, \varepsilon + y_2, \ldots, \varepsilon + y_{2N}].$$

More interesting, common and challenging problem is to aim at solving the ***nonlinear regression tasks***. Here, similar as in the case of nonlinear classification, this will be achieved by considering a linear regression hyperplane in the so-called *feature space.*

Thus, we use the same basic idea in designing SV machines for creating a nonlinear regression function.

We map input vectors $\mathbf{x} \in \Re^n$ into vectors $\mathbf{z}$ of a higher dimensional *feature space F* ($\mathbf{z} = \Phi(\mathbf{x})$ where Φ represents mapping: $\Re^n \rightarrow \Re^f$) and **we solve a linear regression problem in this feature space.**

A mapping $\Phi(\mathbf{x})$ is again chosen in advance. Such an approach again leads to solving a quadratic optimization problem with inequality constraints in a $\mathbf{z}$-space. The solution for an regression hyperplane $f = \mathbf{w}^T\mathbf{z}(\mathbf{x}) + b$ which is linear in a feature space F, will create a nonlinear regressing hypersurface in the original input space. In the case of nonlinear regression, after calculation of LaGrange multiplier vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$, we can find an optimal desired weight vector of the *kernels expansion* as

$$\mathbf{w}_o = \boldsymbol{\alpha}^* - \boldsymbol{\alpha},$$

and an optimal bias b_o can be found from $b_o = \frac{1}{l}\sum_{i=1}^{l}(y_i - g_i)$.

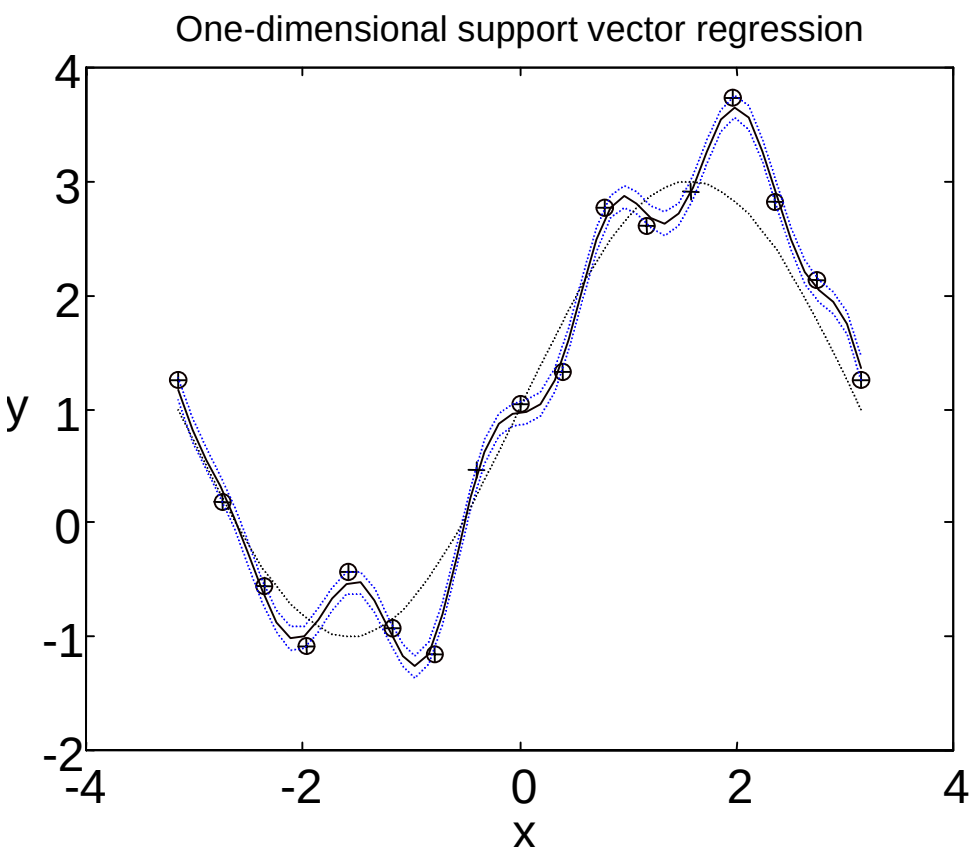
where $\mathbf{g} = \mathbf{G}\,\mathbf{w}_o$ and the matrix $\mathbf{G}$ is a corresponding design matrix of given RBF kernels.

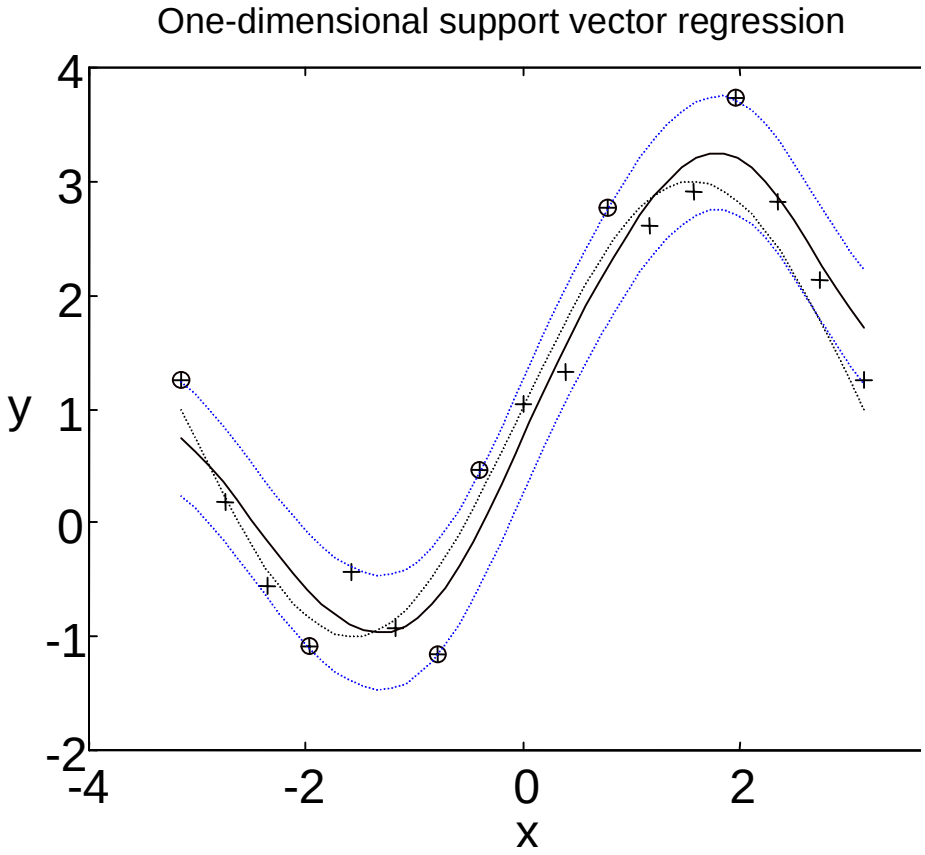
The best nonlinear regression hyperfunction is given by

$$z = f(\mathbf{x}, \mathbf{w}) = \mathbf{G}\mathbf{w} + b.$$

There are a few learning parameters in constructing SV machines for regression. The two most relevant are **the insensitivity zone *e*** and the **penalty parameter *C*** that determines the trade-off between the training error and VC dimension of the model. **Both parameters should be chosen by the user.**

Generally, an increase in an insensitivity zone *e* has smoothing effects on modeling highly noisy polluted data. Increase in *e* means a reduction in requirements on the accuracy of approximation. It decreases the number of SVs leading to data compression too. See the next figures.





The influence of **a insensitivity zone *e* on modeling quality.** *A* nonlinear SVM creates a regression function with **Gaussian kernels** and models a highly polluted (25% noise) sinus function (dashed). 17 measured training data points (plus signs) are used.

Left: *e* = 0.1. 15 SV are chosen (encircled plus signs).

Right: *e* = 0.5. 6 chosen SV produced a much better regressing function.

Some of the constructive problems:

The SV training works almost perfectly for not too large data basis.

However, when the number of data points is large (say $l > 2000$) the QP problem becomes extremely difficult to solve with standard methods. For example, a training set of **50,000 examples amounts to a Hessian matrix H with $2.5*10^9$ (2.5 billion) elements. Using an 8-byte floating-point representation we need 20,000 Megabytes = 20 Gigabytes of memory** (Osuna et al, 1997). This cannot be easily fit into memory of present standard computers.

There are three approaches that resolve the QP for large data sets. Vapnik in (**Vapnik, 1995**) proposed the ***chunking method*** that is the decomposition approach. Another decomposition approach is proposed in (**Osuna et all, 1997**). The sequential minimal optimization (**Platt, 1997**) algorithm is of different character and it seems to be an 'error back propagation' for a SVM learning.

There is an alternative approach in the calculation of the support vectors presented in the section 5.3.4 under the title:

OPTIMAL SUBSET SELECTION BY LINEAR PROGRAMMING (LP)

OPTIMAL SUBSET SELECTION BY LINEAR PROGRAMMING (LP)

At the moment, a lot of work on implementing **LP** approach in support vectors selection **is starting** (Smola et all **1998**, Bennett **1999**, Weston et all **1999**, and Graepel et all **1999**), although the early work on LP based classification algorithms dates back to the mid 1960s (see Mangasarian, **1965**).

We (Ph.D. student I. Hadzic + me) recently found that an analysis of relations between APPROXIMATION and LP was performed in (Cheney and Goldstein, 1958; Stiefel, 1960 and Rice, 1964).

Independently, we also started with an LP approach. Others are not lagging behind.

Here we present the LP approach in a regression problem as given in the book.

The detailed description of the LP approach in both the regression and the classification (pattern recognition) tasks can be found in:

Kecman V., Hadzic I., Support Vectors Selection by Linear Programming, Proceedings of the International Joint Conference on Neural Networks (IJCNN 2000), Vol. 5, pp. 193-199, Como, Italy, 2000

The problem is now the following:

Having the measurement pairs (**x,** y), place the basis function (kernels) G at each data points and perform approximation same as in SVM regression.
However, unlike in QP approach one minimizes the L_1 norm now!

Thus, we do not want to solve the equation below

$$\mathbf{y = Gw},\ \mathbf{w = ?}$$

(which leads to the interpolation). We rather design a parsimonious NN containing less neurons than data points.

In other words, we want to solve $\mathbf{y = Gw}$ such that $\| \mathbf{Gw - y} \|$ is small for some chosen norm. We reformulate the initial problem as follows: Find a weight vector

$$\mathbf{w} = \arg \|\mathbf{w}\|_1, \text{ subject to, } \| \mathbf{Gw - y} \|_\infty \leq \varepsilon\ ,$$

where ε defines the *maximally* allowed error (that is why we used L_∞ norm) and corresponds to the ε - insensitivity zone in SVM.

This constrained optimization problem can be transformed into a standard linear programming form as follows.

First, $\|\mathbf{w}\|_1 = \sum_{p=1}^{P} |w_p|$ is not an LP problem formulation where we typically minimize $\mathbf{c}^T\mathbf{w}$, and $\mathbf{c}$ is some known coefficient vector.

Thus, we use the standard trick by replacing w_p and $|\, w_p \,|$ as follows,

$$w_p = w^+ - w^-$$
$$|w_p| = w^+ + w^-$$

where w^+ and w^- are the two non-negative variables.

Second, the constraint is not in a standard formulation either and it should also be reformulated as follows. Note that

$\| \mathbf{Gw} - \mathbf{y} \|_\infty \leq \varepsilon$ above defines an ε-tube inside which should our approximating function reside.

Such a constraint can be rewritten as,

$$\mathbf{y} - \varepsilon\,\mathbf{1} \leq \mathbf{Gw} \leq \mathbf{y} + \varepsilon\,\mathbf{1}$$

where $\mathbf{1}$ is a (P, 1) column vector filled with ones.

Finally, this problem can be rewritten in a standard LP matrix notation,

$$\min \mathbf{c}^T\mathbf{w} = \min_{\mathbf{w}} [\underbrace{1\,1\ldots1}_{P\,columns}\ \underbrace{1\,1\ldots1}_{P\,columns}] \begin{bmatrix} w_1^+ \\ \vdots \\ w_P^+ \\ w_1^- \\ \vdots \\ w_P^- \end{bmatrix}$$

subject to,

$$\begin{bmatrix} G & -G \\ -G & G \end{bmatrix} \begin{bmatrix} \mathbf{w}^+ \\ \mathbf{w}^- \end{bmatrix} \leq \begin{bmatrix} \mathbf{y} + \varepsilon\mathbf{1} \\ -\mathbf{y} + \varepsilon\mathbf{1} \end{bmatrix}, \mathbf{w}^+ > 0, \mathbf{w}^- > 0$$

Despite the lack of a comparative analysis between QP an LP approaches yet, our first simulational results show that the LP subset selection may be more than a good alternative to the QP based algorithms when faced with a huge training data sets.

Saying that we primarily refer to the following possible benefits of applying LP based learning:

* **LP algorithms are faster and more robust than QP ones,**
* **they tend to minimize number of weights (meaning SV) chosen,**
* **because of the latter, LP share many good properties with an established statistical technique known as Basis Pursuit and finally,**
* **they naturally incorporate the use of kernels for creation of nonlinear separation and regression hypersurfaces in pattern recognition or function approximation.**

Let us conclude the presentation of SVMs by summarizing the basic constructive steps that lead to SV machine:

- **selection of the kernel function that determines the shape of the decision and regression function in classification and regression problems respectively,**

- **selection of the 'shape', i.e., 'smoothing' parameter in the kernel function (for example, polynomial degree and variance of the Gaussian RBF for polynomials and RBF kernels respectively),**

- **choice of the penalty factor *C* and selection of the desired accuracy by defining the insensitivity zone *e*,**

- **solution of the QP (or an LP) problem in *l* and 2*l* variables in the case of classification and regression problems respectively.**

Let us conclude the comparisons between the SVMs and NNs

- **both the NNs and SVMs learn from experimental data,**

- **both the NNs and SVMs are universal approximators in the sense that they can approximate any function to any desired degree of accuracy,**

- **after the learning they are given with the same mathematical model and they can be presented graphically with the same so-called NN's graph,**

- **they differ by the learning method used. While NNs typically use either EBP (or some more sophisticated gradient descent algorithm) or some other linear algebra based approach, the SVMs learn by solving the QP or LP problem.**