MSSC 6250 / Statistical Machine Learning

Instructor: Mehdi Maadooliat

A Gentle Introduction to

Support Vector Machines

in Biomedicine

Slides are borrowed mainly from:

Alexander Statnikov[*],
Douglas Hardin[#],
Isabelle Guyon[†],
Constantin F. Aliferis[*]

[*]New York University, [#]Vanderbilt University, [†]ClopiNet

Department of Mathematical and Statistical Sciences

PART I

- Introduction
- Necessary mathematical concepts
- Support vector machines for binary classification: classical formulation
- Basic principles of statistical machine learning

Part 1

INTRODUCTION

About this tutorial

- **Main goal**: Fully understand support vector machines (and important extensions) with a modicum of mathematics knowledge.

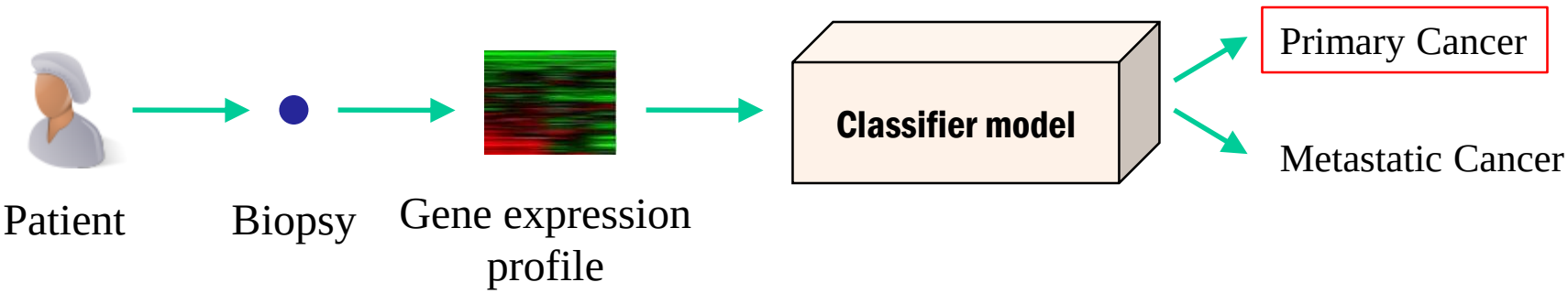
- This tutorial is both modest (it does not invent anything new) and ambitious (support vector machines are generally considered mathematically quite difficult to grasp).

- Tutorial approach:

  learning problem → main idea of the SVM solution →
  geometrical interpretation → math/theory →
  basic algorithms → extensions → case studies.

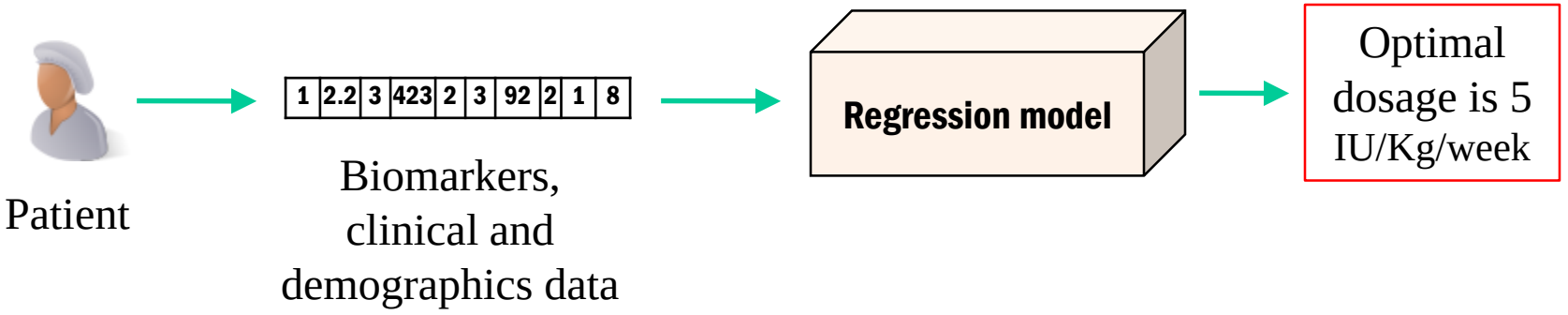
DATA-ANALYSIS PROBLEMS OF INTEREST

1. Build computational classification models (or "*classifiers*") that assign patients/samples into two or more classes.
   - Classifiers can be used for diagnosis, outcome prediction, and other classification tasks.
   - E.g., build a decision-support system to diagnose primary and metastatic cancers from gene expression profiles of the patients:



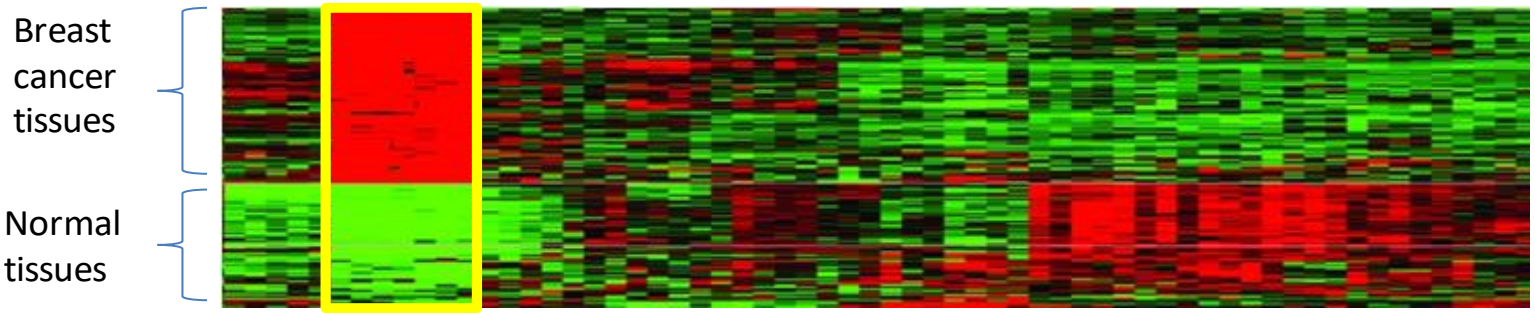
DATA-ANALYSIS PROBLEMS OF INTEREST

2. Build computational regression models to predict values of some continuous response variable or outcome.
   - Regression models can be used to predict survival, length of stay in the hospital, laboratory test values, etc.
   - E.g., build a decision-support system to predict optimal dosage of the drug to be administered to the patient. This dosage is determined by the values of patient biomarkers, and clinical and demographics data:



DATA-ANALYSIS PROBLEMS OF INTEREST

3. Out of all measured variables in the dataset, select the smallest subset of variables that is necessary for the most accurate prediction (classification or regression) of some variable of interest (e.g., phenotypic response variable).
   - E.g., find the most compact panel of breast cancer biomarkers from microarray gene expression data for 20,000 genes:

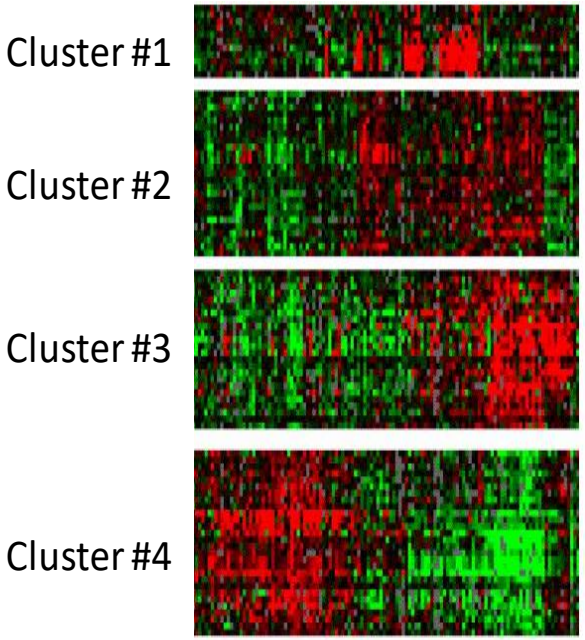


DATA-ANALYSIS PROBLEMS OF INTEREST

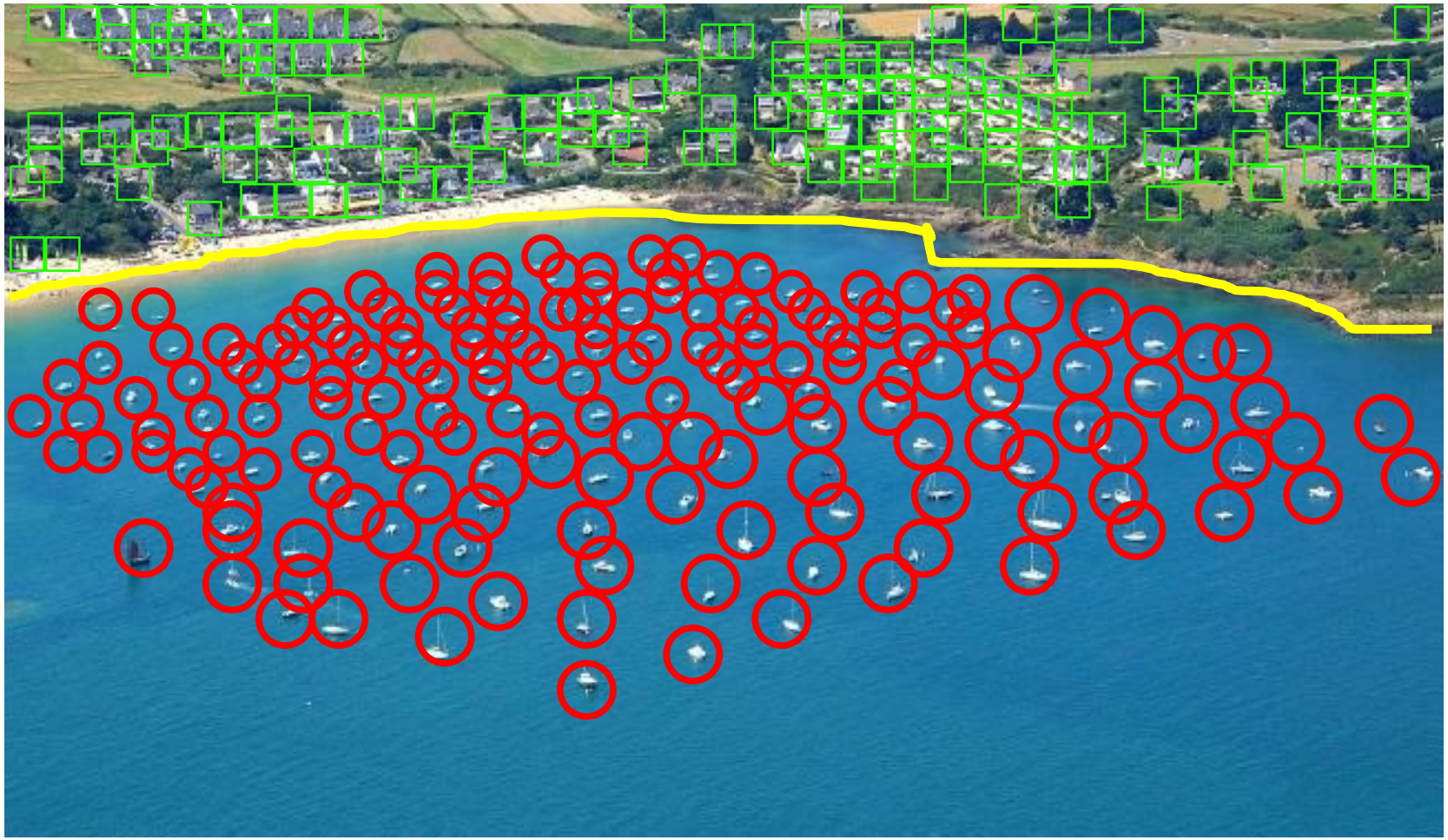
4. Build a computational model to identify novel or outlier patients/samples.
   - Such models can be used to discover deviations in sample handling protocol when doing quality control of assays, etc.
   - E.g., build a decision-support system to identify aliens.

DATA-ANALYSIS PROBLEMS OF INTEREST

5. Group patients/samples into several clusters based on their similarity.
   - These methods can be used to discovery disease sub-types and for other tasks.
   - E.g., consider clustering of brain tumor patients into 4 clusters based on their gene expression profiles. All patients have the same pathological sub-type of the disease, and clustering discovers new disease subtypes that happen to have different characteristics in terms of patient survival and time to recurrence after treatment.



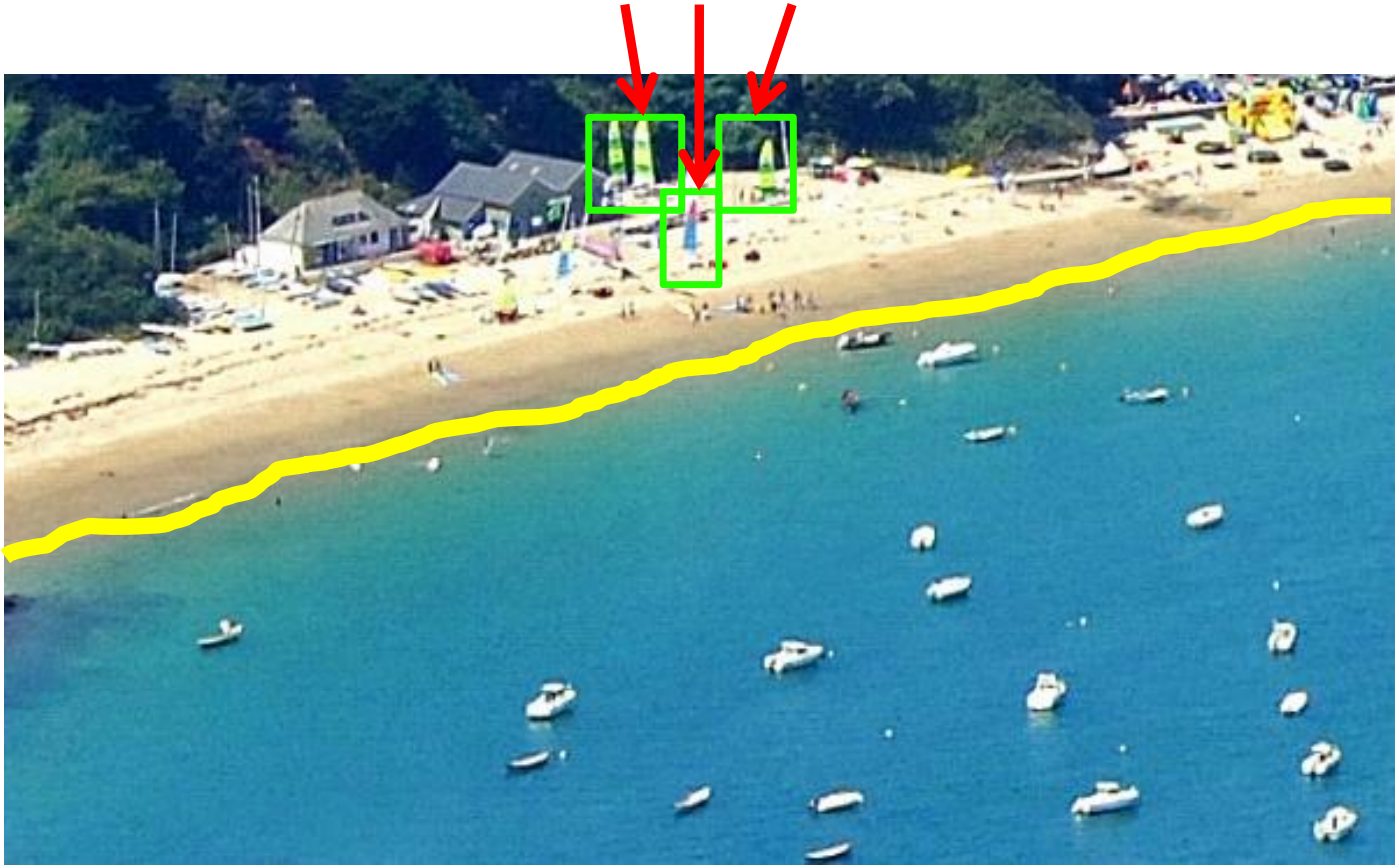
BASIC PRINCIPLES OF CLASSIFICATION



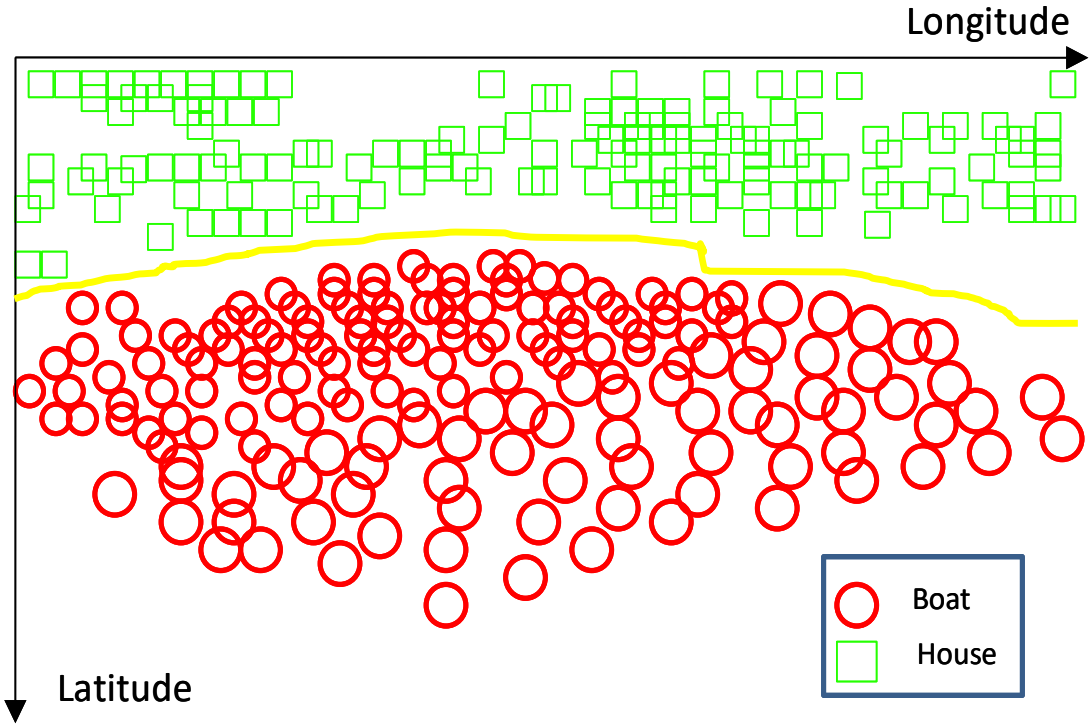
- Want to classify objects as boats and houses.
- Coast line serves as a *decision surface* that separates two classes.

BASIC PRINCIPLES OF CLASSIFICATION

These boats will be misclassified as houses

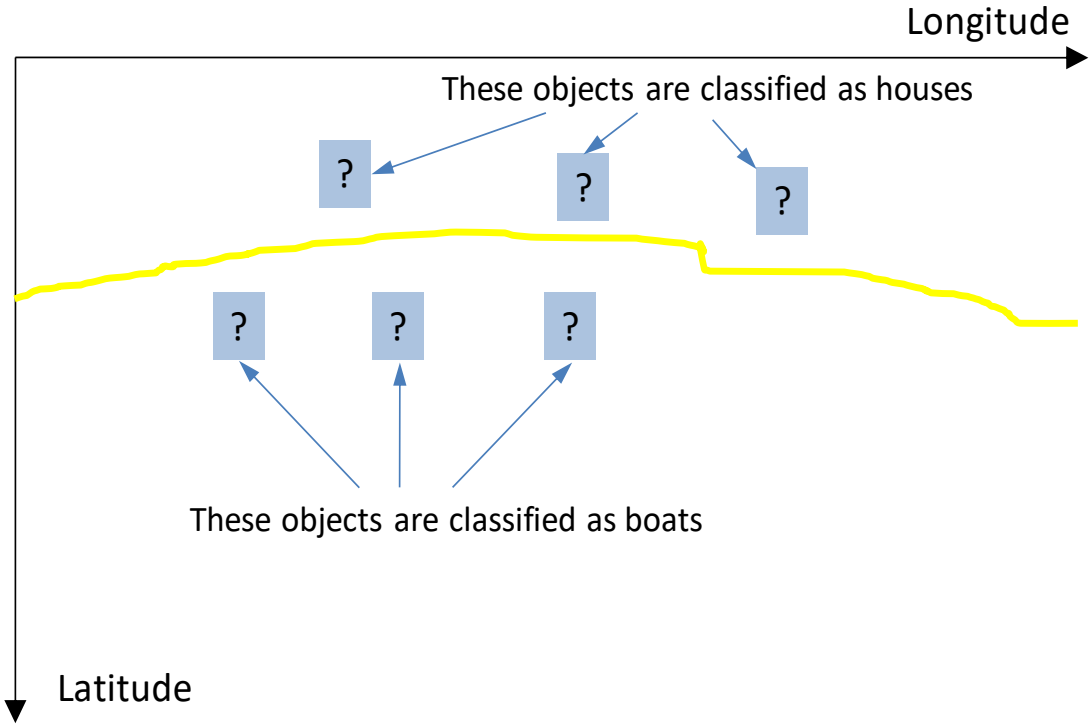


BASIC PRINCIPLES OF CLASSIFICATION



- The methods that build classification models (i.e., "classification algorithms") operate very similarly to the previous example.
- First all objects are represented geometrically.
- Then the algorithm seeks to find a decision surface that separates classes of objects

BASIC PRINCIPLES OF CLASSIFICATION

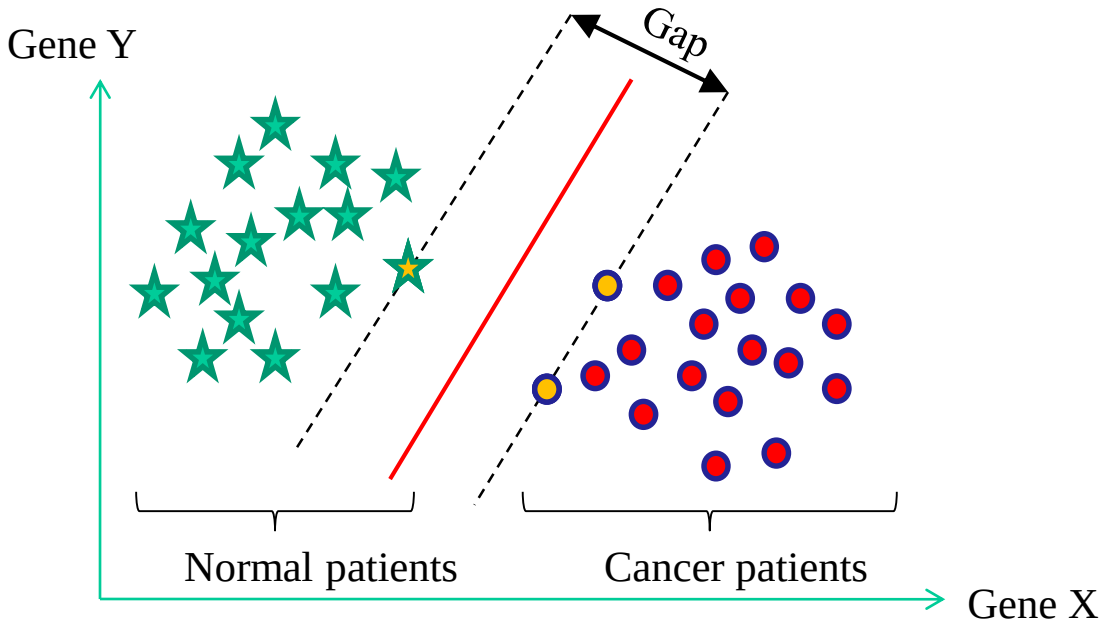


- Unseen (new) objects are classified as "boats" if they fall below the decision surface and as "houses" if the fall above it

The Support Vector Machine (SVM) approach

- Support vector machines (SVMs) is a binary classification algorithm that offers a solution to problem #1.
- Extensions of the basic SVM algorithm can be applied to solve problems #1-#5.
- SVMs are important because of
  a. theoretical reasons:
     - Robust to very large number of variables and small samples
     - Can learn both simple and highly complex classification models
     - Employ sophisticated mathematical principles to avoid overfitting
  b. superior empirical results.

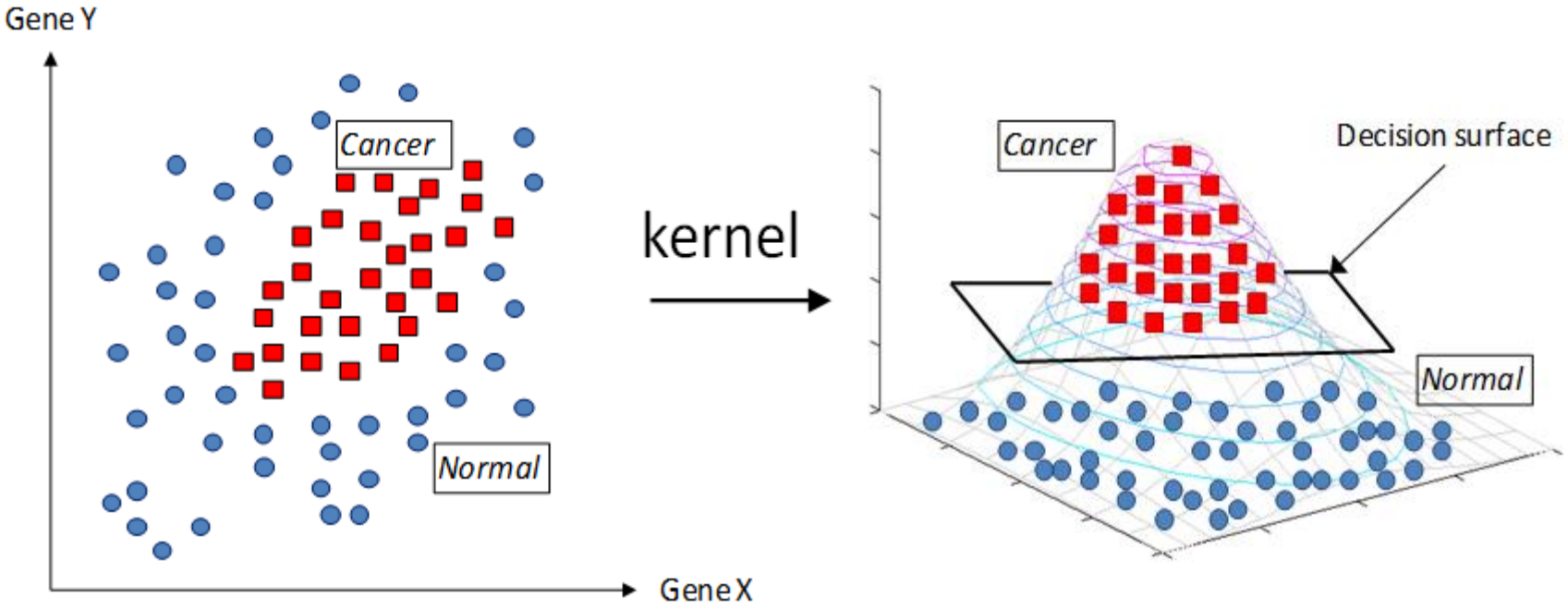
MAIN IDEAS OF SVMS



- Consider example dataset described by 2 genes, gene X and gene Y
- Represent patients geometrically (by "vectors")

- Find a linear decision surface ("hyperplane") that can separate patient classes and has the largest distance (i.e., largest "gap" or "margin") between border-line patients (i.e., "support vectors");

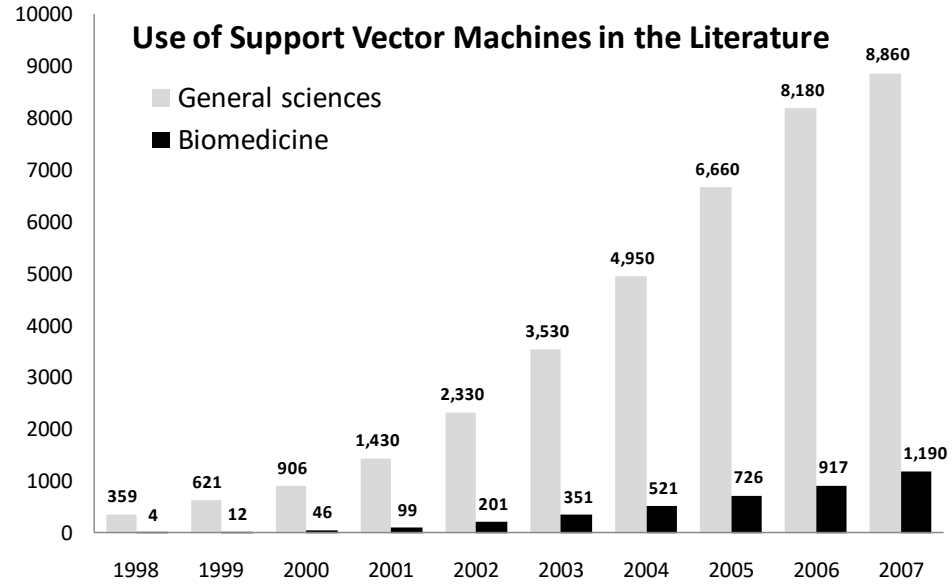
MAIN IDEAS OF SVMS

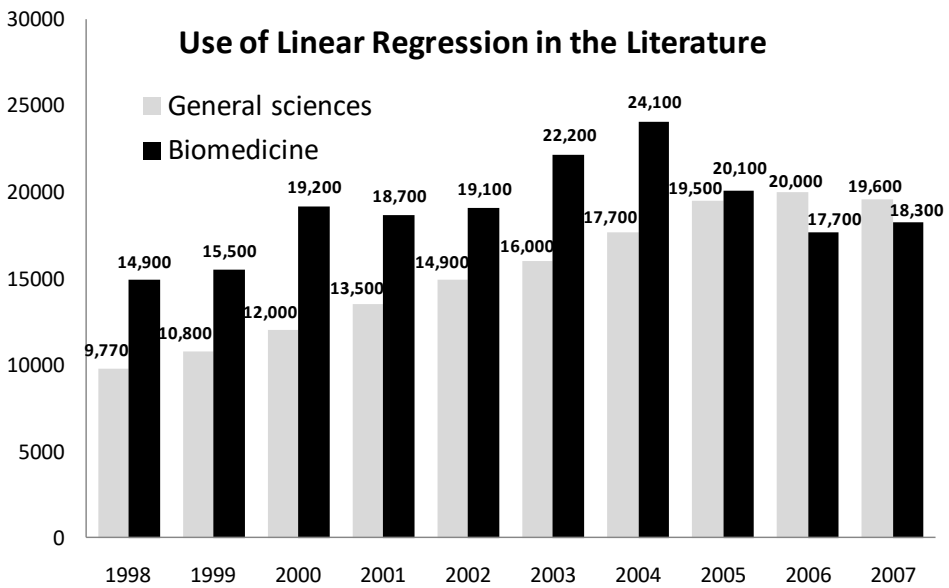


- If such linear decision surface does not exist, the data is mapped into a much higher dimensional space ("feature space") where the separating decision surface is found;
- The feature space is constructed via very clever mathematical projection ("kernel trick").

History of SVMs and usage in the literature

- Support vector machine classifiers have a long history of development starting from the 1960's.
- The most important milestone for development of modern SVMs is the 1992 paper by Boser, Guyon, and Vapnik ("*A training algorithm for optimal margin classifiers*")





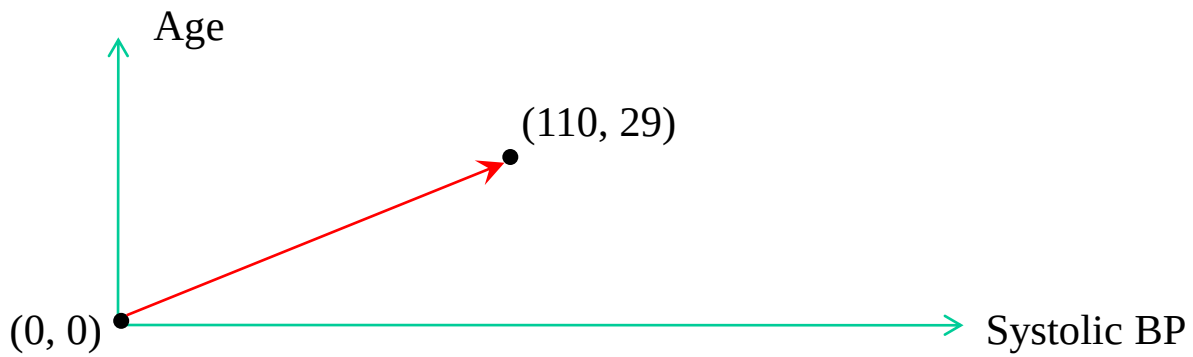
MARQUETTE UNIVERSITY
Be The Difference.

Part 1

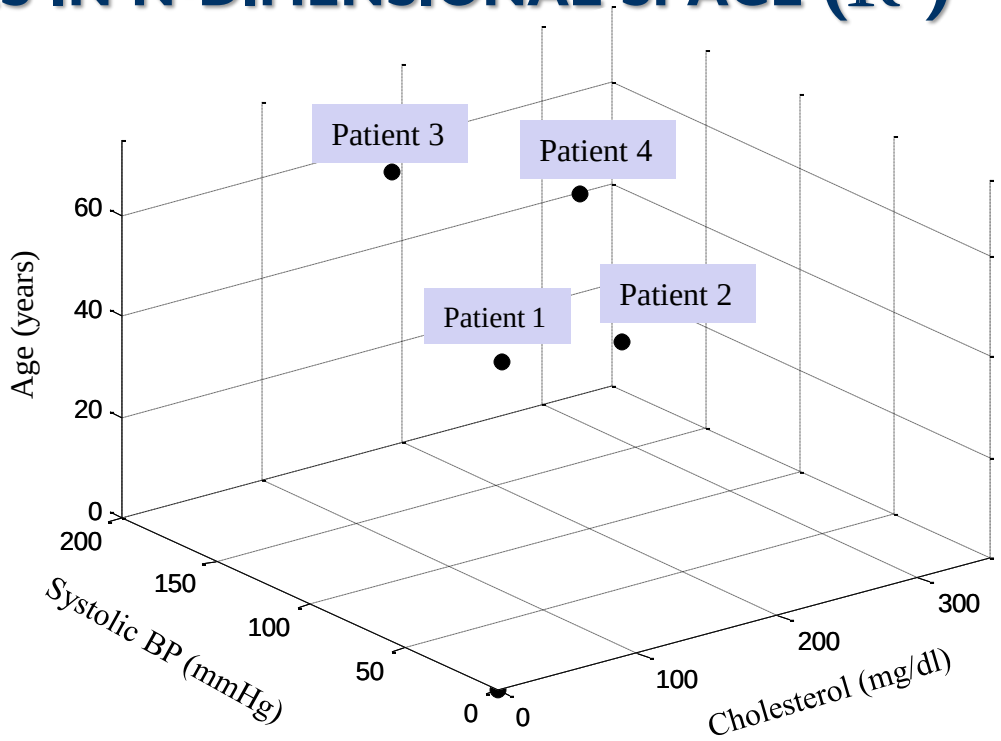
NECESSARY MATHEMATICAL CONCEPTS

How to represent samples geometrically? Vectors in n-dimensional space ($\mathbb{R}^n$)

- Assume that a sample/patient is described by *n* characteristics ("features" or "variables")
- **Representation:** Every sample/patient is a vector in $\mathbb{R}^n$ with tail at point with 0 coordinates and arrow-head at point with the feature values.
- **Example:** Consider a patient described by 2 features:
  - *Systolic BP* = 110 and *Age* = 29.
  - This patient can be represented as a vector in $\mathbb{R}^2$:



How to represent samples geometrically?
Vectors in n-dimensional space ($\mathbb{R}^n$)

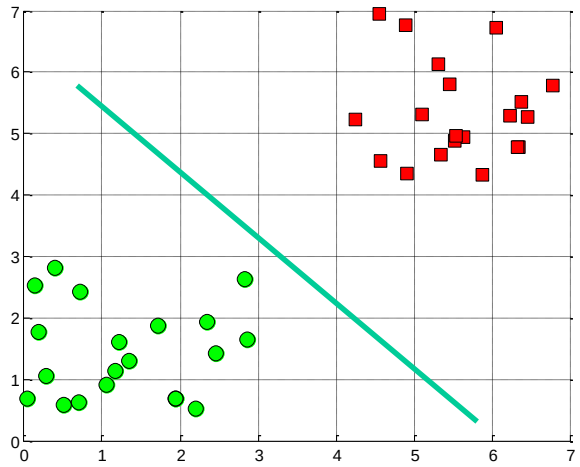


Patient id	Cholesterol (mg/dl)	Systolic BP (mmHg)	Age (years)	Tail of the vector	Arrow-head of the vector
1	150	110	35	(0,0,0)	(150, 110, 35)
2	250	120	30	(0,0,0)	(250, 120, 30)
3	140	160	65	(0,0,0)	(140, 160, 65)
4	300	180	45	(0,0,0)	(300, 180, 45)

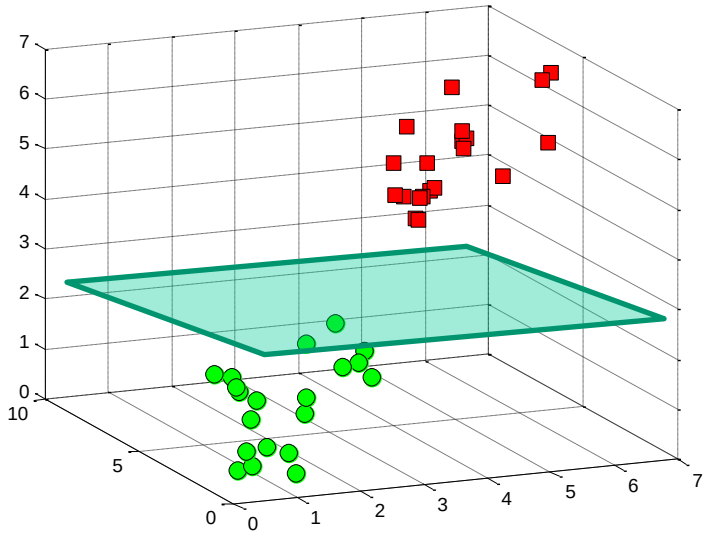
PURPOSE OF VECTOR REPRESENTATION

- Having represented each sample/patient as a vector allows now to geometrically represent the decision surface that separates two groups of samples/patients.

A decision surface in $\mathbb{R}^2$



A decision surface in $\mathbb{R}^3$



- In order to define the decision surface, we need to introduce some basic math elements…

BASIC OPERATION ON VECTORS IN $\mathbb{R}^N$

1. Multiplication by a scalar

Consider a vector $\vec{a} = (a_1, a_2, ..., a_n)$ and a scalar c

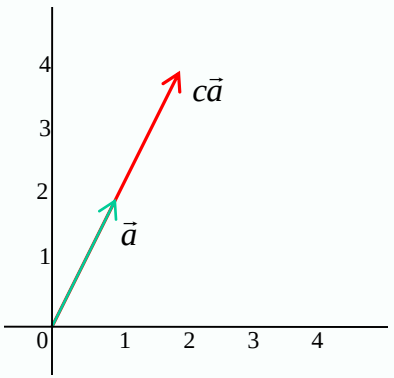
Define: $c\vec{a} = (ca_1, ca_2, ..., ca_n)$

When you multiply a vector by a scalar, you "stretch" it in the same or opposite direction depending on whether the scalar is positive or negative.

$\vec{a} = (1,2)$

$c = 2$

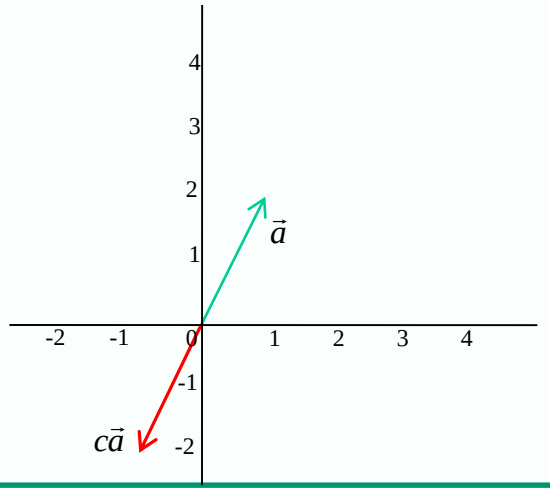
$\vec{c}a = (2,4)$



$\vec{a} = (1,2)$

$c = -1$

$\vec{c}a = (-1,-2)$



BASIC OPERATION ON VECTORS IN $\mathbb{R}^N$

2. Addition

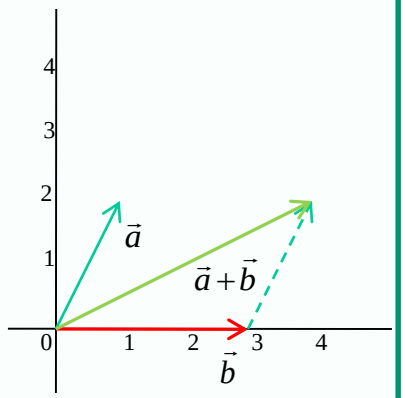
Consider vectors $\vec{a} = (a_1, a_2, ..., a_n)$ and $\vec{b} = (b_1, b_2, ..., b_n)$

Define: $\vec{a} + \vec{b} = (a_1 + b_1, a_2 + b_2, ..., a_n + b_n)$

$\vec{a} = (1,2)$

$\vec{b} = (3,0)$

$\vec{a} + \vec{b} = (4,2)$



Recall addition of forces in classical mechanics.

BASIC OPERATION ON VECTORS IN $\mathbb{R}^N$

3. Subtraction

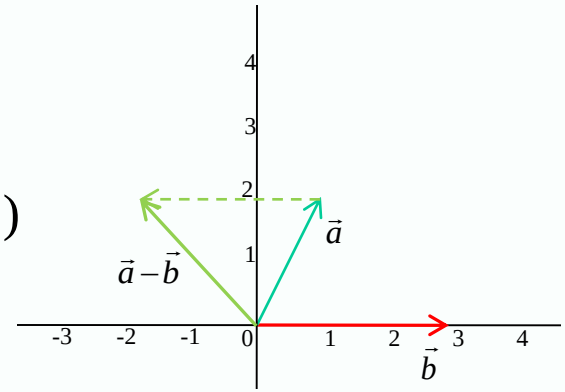
Consider vectors $\vec{a} = (a_1, a_2, ..., a_n)$ and $\vec{b} = (b_1, b_2, ..., b_n)$

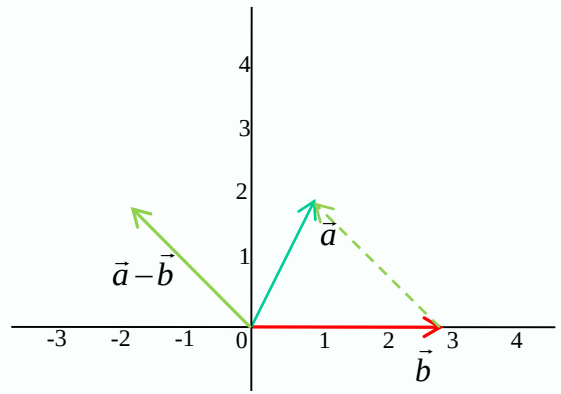
Define: $\vec{a} - \vec{b} = (a_1 - b_1, a_2 - b_2, ..., a_n - b_n)$

$\vec{a} = (1,2)$

$\vec{b} = (3,0)$

$\vec{a} - \vec{b} = (-2,2)$





BASIC OPERATION ON VECTORS IN $\mathbb{R}^N$

4. Euclidian length or L2-norm

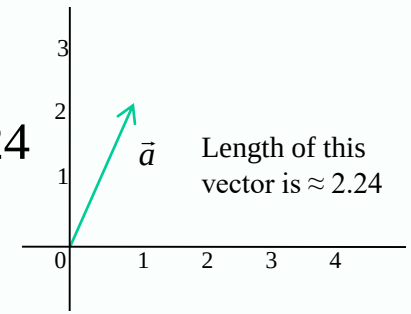
Consider a vector $\vec{a} = (a_1, a_2, ..., a_n)$

Define the L2-norm: $\|\vec{a}\|_2 = \sqrt{a_1^2 + a_2^2 + ... + a_n^2}$

We often denote the L2-norm without subscript, i.e. $\|\vec{a}\|$

$\vec{a} = (1,2)$

$\|\vec{a}\|_2 = \sqrt{5} \approx 2.24$



L2-norm is a typical way to measure length of a vector; other methods to measure length also exist.

BASIC OPERATION ON VECTORS IN $\mathbb{R}^N$

5. Dot product

Consider vectors $\vec{a} = (a_1, a_2, ..., a_n)$ and $\vec{b} = (b_1, b_2, ..., b_n)$

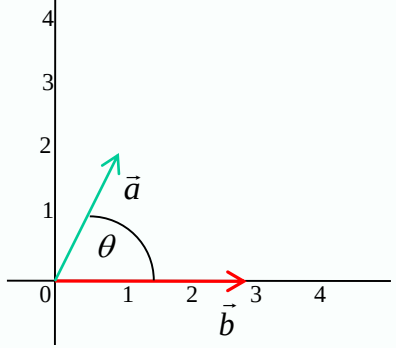
Define dot product: $\vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 + ... + a_n b_n = \sum_{i=1}^{n} a_i b_i$

- The law of cosines says that $\vec{a} \cdot \vec{b} = \| \vec{a} \|_2 \| \vec{b} \|_2 \cos\theta$ where θ is the angle between $\vec{a}$ and $\vec{b}$.
- Therefore, when the vectors are perpendicular $\vec{a} \cdot \vec{b} = 0$

$\vec{a} = (1,2)$

$\vec{b} = (3,0)$

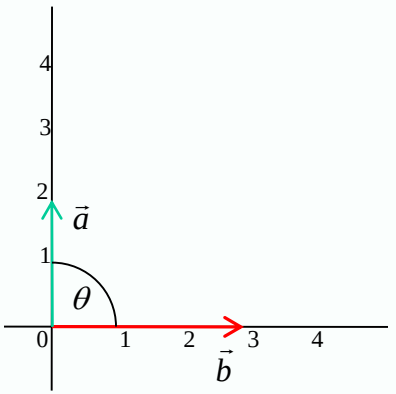
$\vec{a} \cdot \vec{b} = 3$



$\vec{a} = (0,2)$

$\vec{b} = (3,0)$

$\vec{a} \cdot \vec{b} = 0$



BASIC OPERATION ON VECTORS IN $\mathbb{R}^N$

5. Dot product (continued)

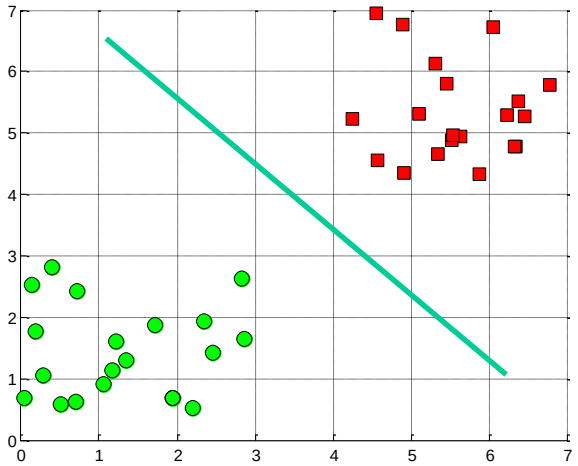
$$\vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n = \sum_{i=1}^{n} a_i b_i$$

- Property: $\vec{a} \cdot \vec{a} = a_1 a_1 + a_2 a_2 + \ldots + a_n a_n = \left\| \vec{a} \right\|_2^2$
- In the classical regression equation $y = \vec{w} \cdot \vec{x} + b$ the response variable *y* is just a dot product of the vector representing patient characteristics ($\vec{x}$) and the regression weights vector ($\vec{w}$) which is common across all patients plus an offset *b*.

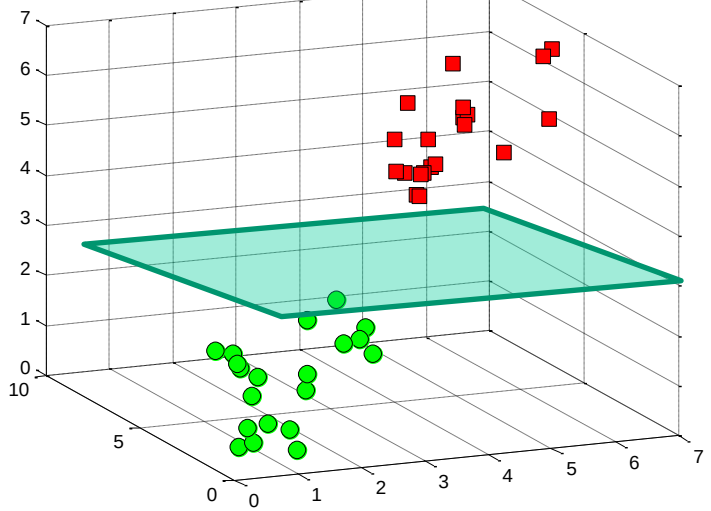
HYPERPLANES AS DECISION SURFACES

- A hyperplane is a linear decision surface that splits the space into two parts;
- It is obvious that a hyperplane is a binary classifier.

A hyperplane in $\mathbb{R}^2$ is a line



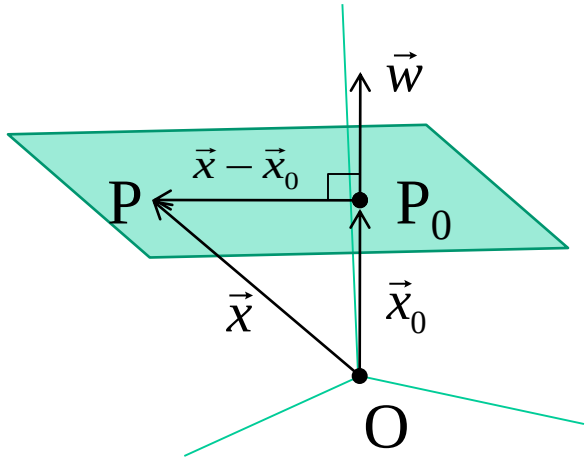
A hyperplane in $\mathbb{R}^3$ is a plane



A hyperplane in $\mathbb{R}^n$ is an n-1 dimensional subspace

Equation of a hyperplane

Consider the case of $\mathbb{R}^3$:



An equation of a hyperplane is defined by a point (P_0) and a perpendicular vector to the plane ($\vec{w}$) at that point.

Define vectors: $\vec{x}_0 = \overrightarrow{OP_0}$ and $\vec{x} = \overrightarrow{OP}$, where *P* is an arbitrary point on a hyperplane.

A condition for *P* to be on the plane is that the vector $\vec{x}-\vec{x}_0$ is perpendicular to $\vec{w}$:

$$\vec{w}\cdot(\vec{x}-\vec{x}_0)=0 \quad \text{or}$$

$$\vec{w}\cdot\vec{x}-\vec{w}\cdot\vec{x}_0=0 \quad \text{define} \quad b=-\vec{w}\cdot\vec{x}_0$$

$$\boxed{\vec{w}\cdot\vec{x}+b=0}$$

The above equations also hold for $\mathbb{R}^n$ when n>3.

Equation of a hyperplane

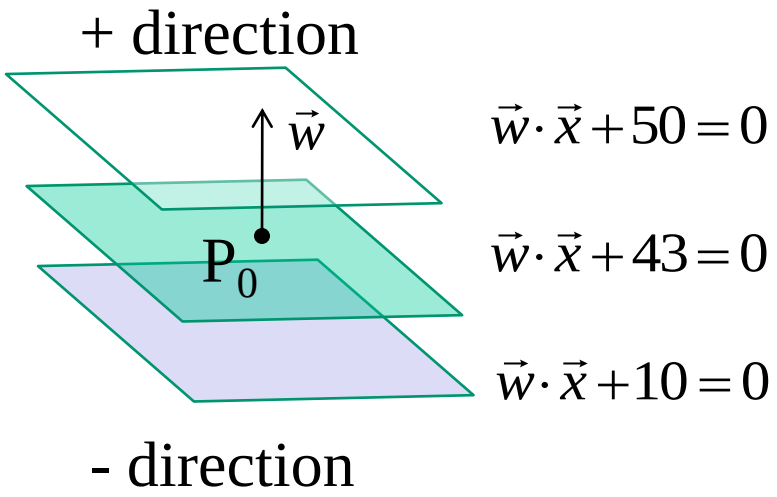
Example

$\vec{w} = (4,-1,6)$

$P_0 = (0,1,-7)$

$b = -\vec{w} \cdot \vec{x}_0 = -(0-1-42) = 43$

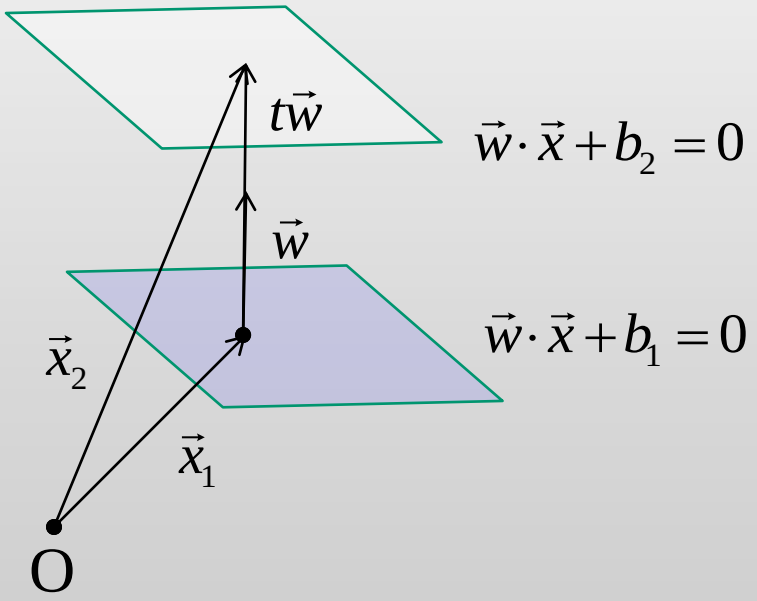
$$\Rightarrow \vec{w} \cdot \vec{x} + 43 = 0$$
$$\Rightarrow (4,-1,6) \cdot \vec{x} + 43 = 0$$
$$\Rightarrow (4,-1,6) \cdot (x_{(1)}, x_{(2)}, x_{(3)}) + 43 = 0$$
$$\Rightarrow 4x_{(1)} - x_{(2)} + 6x_{(3)} + 43 = 0$$



What happens if the *b* coefficient changes?
The hyperplane moves along the direction of $\vec{w}$.
We obtain "parallel hyperplanes".

Distance between two parallel hyperplanes $\vec{w} \cdot \vec{x} + b_1 = 0$ and $\vec{w} \cdot \vec{x} + b_2 = 0$ is equal to $D = |b_1 - b_2| / \|\vec{w}\|$.

(Derivation of the distance between two parallel hyperplanes)



$$\vec{w}\cdot\vec{x}+b_2=0$$

$$\vec{w}\cdot\vec{x}+b_1=0$$

$$\vec{x}_2=\vec{x}_1+t\vec{w}$$

$$D=\left\|t\vec{w}\right\|=\left|t\right|\left\|\vec{w}\right\|$$

$$\vec{w}\cdot\vec{x}_2+b_2=0$$

$$\vec{w}\cdot(\vec{x}_1+t\vec{w})+b_2=0$$

$$\vec{w}\cdot\vec{x}_1+t\left\|\vec{w}\right\|^2+b_2=0$$

$$(\vec{w}\cdot\vec{x}_1+b_1)-b_1+t\left\|\vec{w}\right\|^2+b_2=0$$

$$-b_1+t\left\|\vec{w}\right\|^2+b_2=0$$

$$t=(b_1-b_2)/\left\|\vec{w}\right\|^2$$

$$\Rightarrow D=\left|t\right|\left\|\vec{w}\right\|=\left|b_1-b_2\right|/\left\|\vec{w}\right\|$$

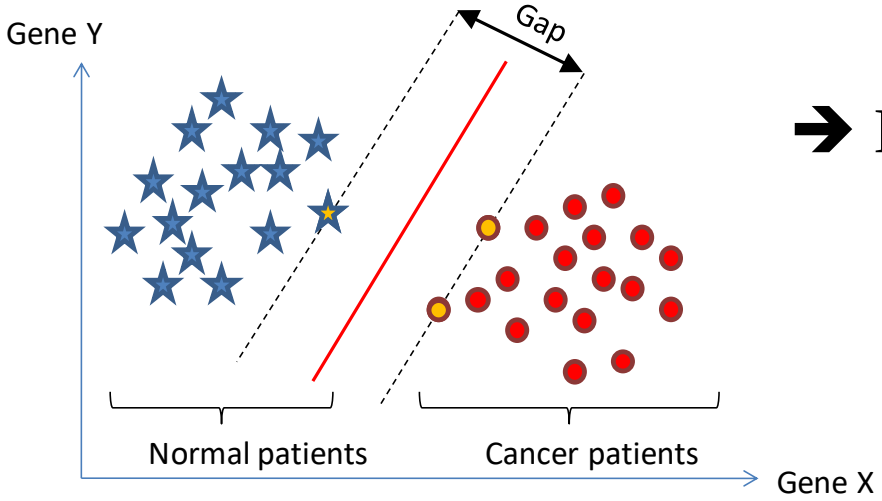
RECAP

We know…

- How to represent patients (as "vectors")
- How to define a linear decision surface ("hyperplane")

We need to know…

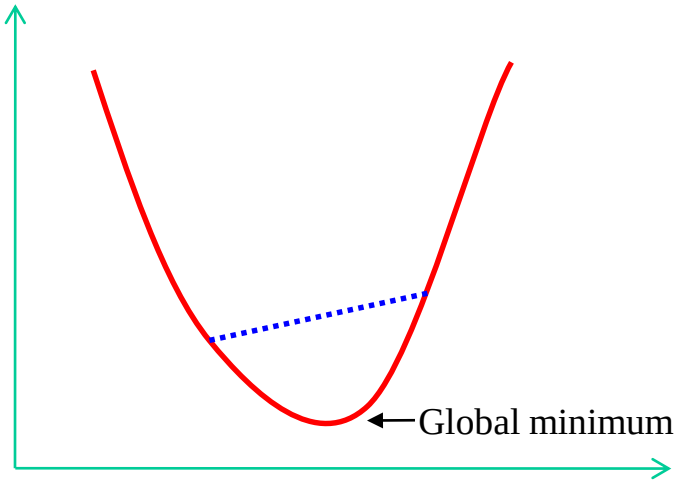
- How to efficiently compute the hyperplane that separates two classes with the largest "gap"?



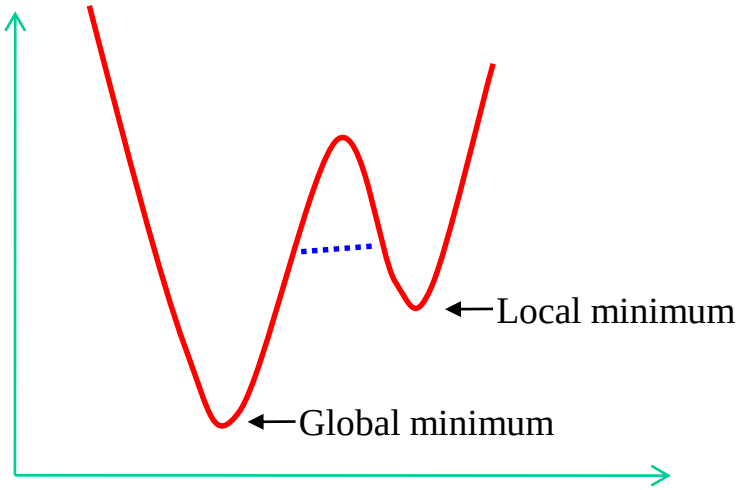
➔ Need to introduce basics of relevant optimization theory

Basics of optimization: Convex functions

- A function is called *convex* if the function lies below the straight line segment connecting two points, for any two points in the interval.
- Property: Any local minimum is a global minimum!



Convex function

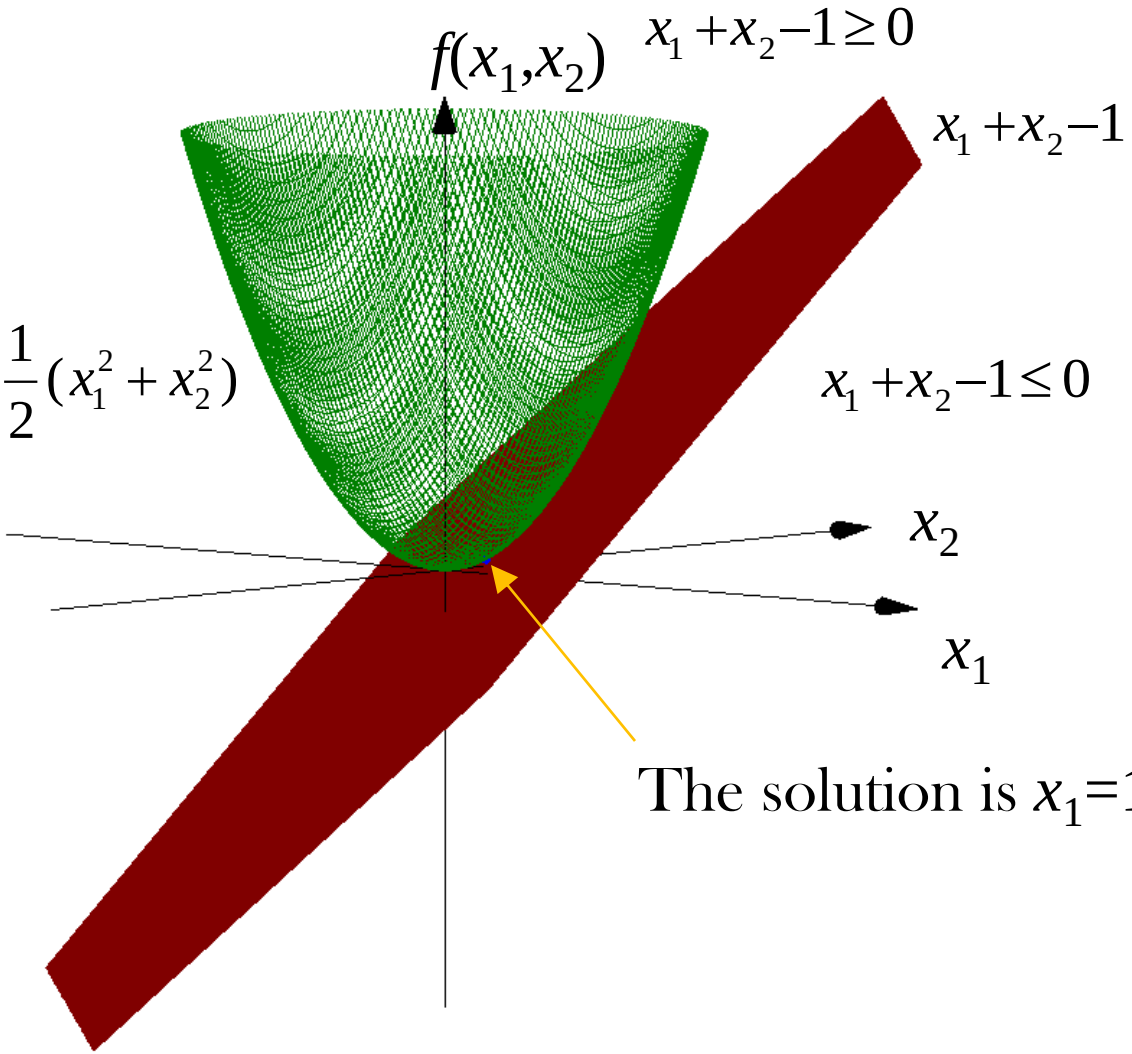


Non-convex function

BASICS OF OPTIMIZATION: QUADRATIC PROGRAMMING (QP)

- Quadratic programming (QP) is a special optimization problem: the function to optimize ("*objective*") is quadratic, subject to linear *constraints*.
- Convex QP problems have convex objective functions.
- These problems can be solved easily and efficiently by greedy algorithms (because every local minimum is a global minimum).
- Consider $\vec{x} = (x_1, x_2)$
- Minimize $\underbrace{\frac{1}{2}(x_1^2 + x_2^2)}_{\text{quadratic objective}}$ subject to $\underbrace{x_1 + x_2 - 1 \geq 0}_{\text{linear constraints}}$
- This is QP problem, and it is a convex QP as we will see later

BASICS OF OPTIMIZATION: EXAMPLE QP PROBLEM



The solution is x_1=1/2 and x_2=1/2.

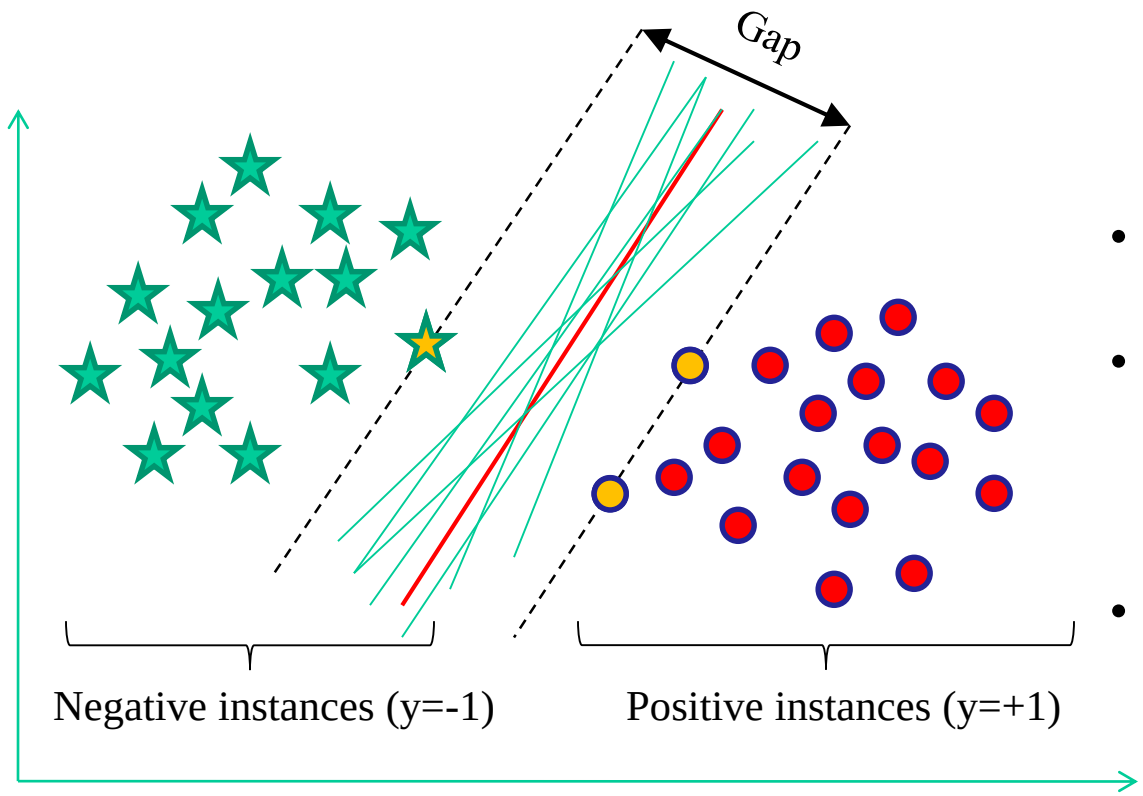
Part 1

SUPPORT VECTOR MACHINES FOR BINARY CLASSIFICATION: CLASSICAL FORMULATION

CASE 1: LINEARLY SEPARABLE DATA; "HARD-MARGIN" LINEAR SVM

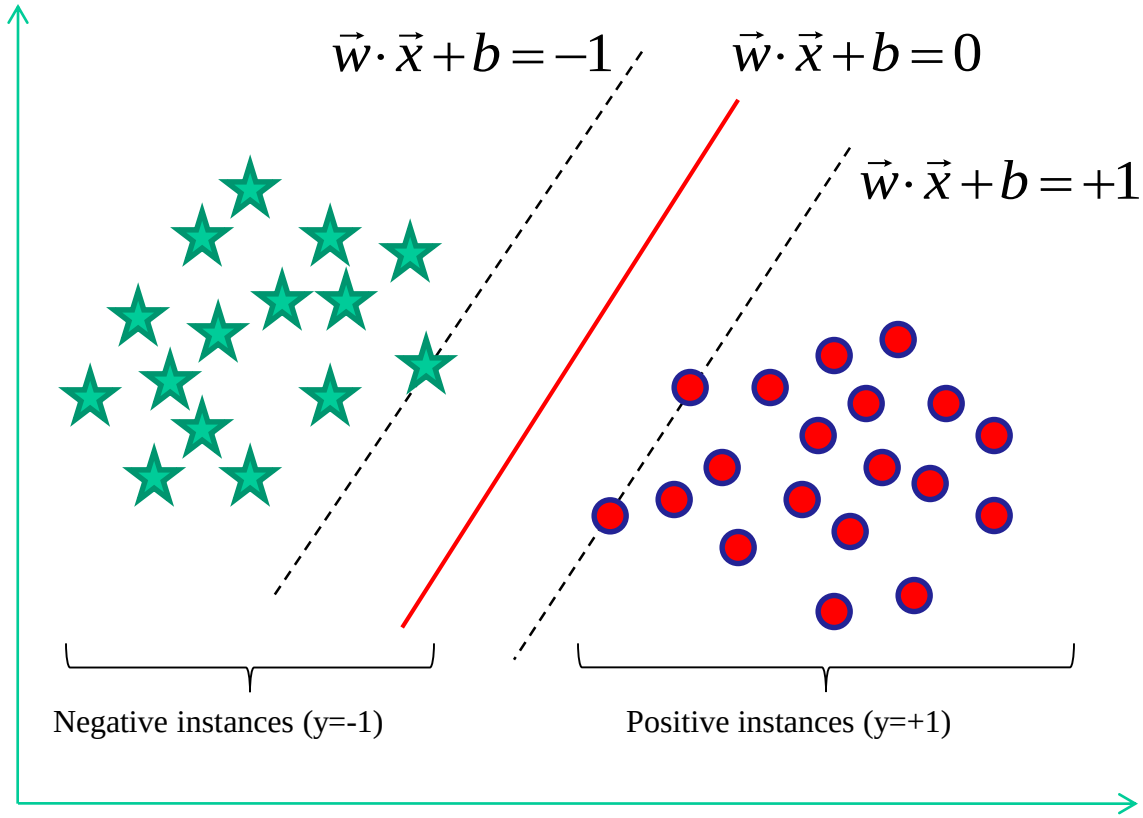
Given training data: $\vec{x}_1, \vec{x}_2, ..., \vec{x}_N \in R^n$

$y_1, y_2, ..., y_N \in \{-1, +1\}$



- Want to find a classifier (hyperplane) to separate negative instances from the positive ones.
- An infinite number of such hyperplanes exist.
- SVMs finds the hyperplane that maximizes the gap between data points on the boundaries (so-called "support vectors").
- If the points on the boundaries are not informative (e.g., due to noise), SVMs will not do well.

STATEMENT OF LINEAR SVM CLASSIFIER



The gap is distance between parallel hyperplanes:

$\vec{w}\cdot\vec{x}+b=-1$ and
$\vec{w}\cdot\vec{x}+b=+1$

- Or equivalently:

$\vec{w}\cdot\vec{x}+(b+1)=0$
$\vec{w}\cdot\vec{x}+(b-1)=0$

- We know that

$D=|b_1-b_2|/\|\vec{w}\|$

- Therefore:

$D=2/\|\vec{w}\|$

Since we want to maximize the gap,

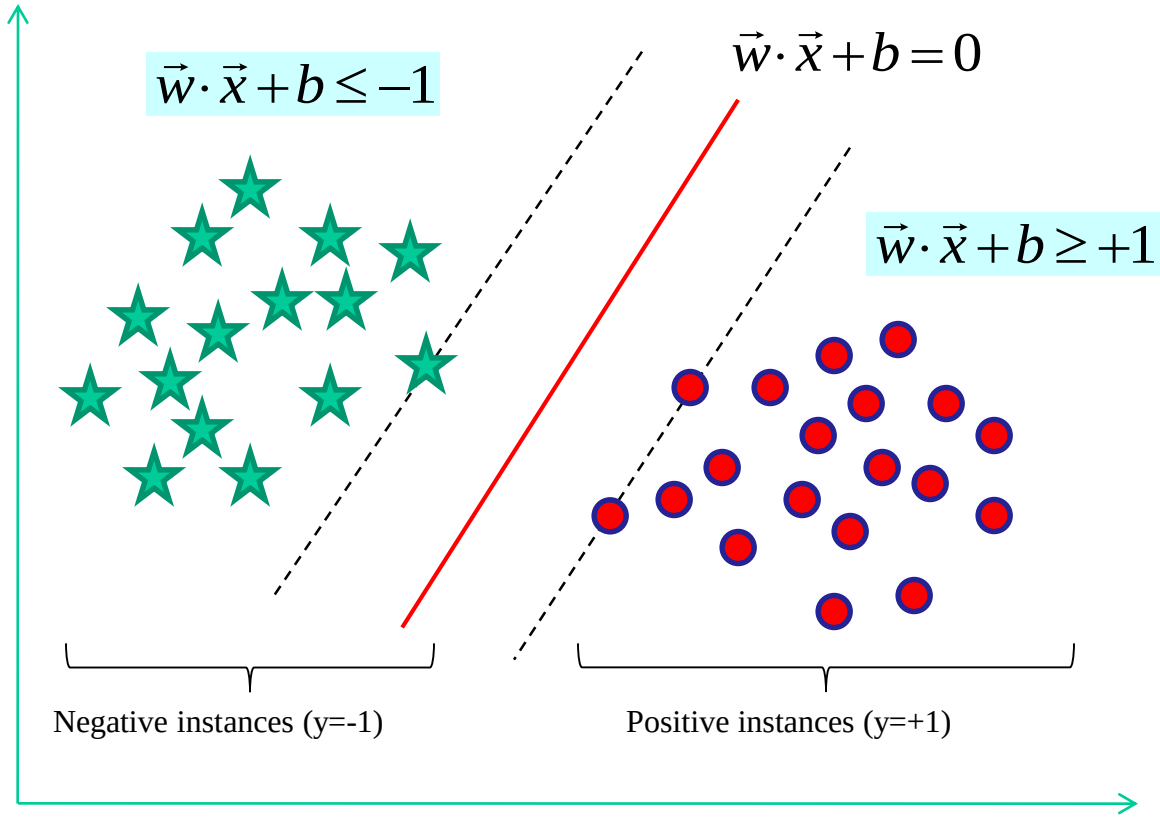
we need to minimize $\|\vec{w}\|$

or equivalently **minimize** $\frac{1}{2}\|\vec{w}\|^2$

($\frac{1}{2}$ is convenient for taking derivative later on)

MARQUETTE UNIVERSITY
Be The Difference.

STATEMENT OF LINEAR SVM CLASSIFIER



In addition we need to impose constraints that all instances are correctly classified. In our case:

$\vec{w}\cdot\vec{x}_i+b\leq -1$ if $y_i=-1$

$\vec{w}\cdot\vec{x}_i+b\geq +1$ if $y_i=+1$

- Equivalently:

$$y_i(\vec{w}\cdot\vec{x}_i+b)\geq 1$$

- In summary:
  Want to **minimize** $\frac{1}{2}\|\vec{w}\|^2$ subject to $y_i(\vec{w}\cdot\vec{x}_i+b)\geq 1$ for $i=1,\ldots,N$
- Then given a new instance x, the classifier is $f(\vec{x})=sign(\vec{w}\cdot\vec{x}+b)$

SVM OPTIMIZATION PROBLEM: PRIMAL FORMULATION

Minimize $\frac{1}{2}\sum_{j=1}^{p} w_j^2$ subject to $y_i(\vec{w}\cdot\vec{x}_i + b) - 1 \geq 0$ for $i = 1,\ldots,N$

Objective function

Constraints

- This is called "*primal formulation of linear SVMs*".
- It is a convex quadratic programming (QP) optimization problem with p variables ($w_j, i = 1, \ldots, p$), where p is the number of features in the dataset.

SVM OPTIMIZATION PROBLEM: DUAL FORMULATION

- The previous problem can be recast in the so-called "*dual form*" giving rise to "*dual formulation of linear SVMs*".

- It is also a convex quadratic programming problem but with N variables (α_i ,$i = 1,\ldots,N$), where N is the number of samples.

- **Maximize** $\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j$ subject to $\alpha_i \geq 0$ and $\sum_{i=1}^{N}\alpha_i y_i = 0$.

Objective function Constraints

- Then the w-vector is defined in terms of α_i: $\vec{w} = \sum_{i=1}^{N}\alpha_i y_i \vec{x}_i$

- And the solution becomes: $f(\vec{x}) = sign(\sum_{i=1}^{N}\alpha_i y_i \vec{x}_i \cdot \vec{x} + b)$

SVM OPTIMIZATION PROBLEM:
BENEFITS OF USING DUAL FORMULATION

1) No need to access original data, need to access only dot products.

Objective function: $\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j$

Solution: $f(\vec{x}) = sign(\sum_{i=1}^{N}\alpha_i y_i \vec{x}_i \cdot \vec{x} + b)$

2) Number of free parameters is bounded by the number of support vectors and not by the number of variables (beneficial for high-dimensional problems).

E.g., if a microarray dataset contains 20,000 genes and 100 patients, then need to find only up to 100 parameters!

(Derivation of dual formulation)

Minimize $\frac{1}{2}\sum_{j=1}^{p} w_j^2$ subject to $y_i(\vec{w}\cdot\vec{x}_i + b) - 1 \geq 0$ for $i = 1,\ldots,N$

Objective function

Constraints

https://en.wikipedia.org/wiki/Duality_(optimization)#Convex_problems

Convex problems [edit]

For a convex minimization problem with inequality constraints,

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \end{aligned}$$

the Lagrangian dual problem is

Apply the method of Lagrange multipliers.

Define Lagrangian $\Lambda_p(\vec{w}, b, \vec{\alpha}) = \frac{1}{2}\sum_{j=1}^{p} w_j^2 - \sum_{i=1}^{N} \alpha_i (y_i(\vec{w}\cdot\vec{x}_i + b) - 1)$

a vector with p elements

a vector with N elements

$$\Lambda(x, u) = \left(f(x) + \sum_{j=1}^{m} u_j g_j(x) \right)$$

We need to **maximize** this Lagrangian with respect to $\vec{w}, b, \vec{\alpha}$ and simultaneously require that the derivative with respect to $\vec{w}, b, \vec{\alpha}$ vanishes, all subject to the constraints that $\alpha_i \geq 0$.

$$\begin{aligned} \underset{x,u}{\text{maximize}} \quad & f(x) + \sum_{j=1}^{m} u_j g_j(x) \\ \text{subject to} \quad & \nabla f(x) + \sum_{j=1}^{m} u_j \nabla g_j(x) = 0 \\ & u_i \geq 0, \quad i = 1, \ldots, m \end{aligned}$$

(Derivation of dual formulation)

If we set the derivatives with respect to $\vec{w}, b$ to 0, we obtain:

$$\frac{\partial \Lambda_P(\vec{w}, b, \vec{\alpha})}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\frac{\partial \Lambda_P(\vec{w}, b, \vec{\alpha})}{\partial \vec{w}} = 0 \Rightarrow \vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i$$

https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

We substitute the above into the equation for $\Lambda_P(\vec{w}, b, \vec{\alpha})$ and obtain "*dual formulation of linear SVMs*":

$$\Lambda_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j$$

We seek to **maximize** the above Lagrangian with respect to $\vec{\alpha}$, subject to the constraints that $\alpha_i \geq 0$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$.

IMPLEMENTATION

- There exist several specialized algorithms for quickly solving the quadratic programming (QP) problem that arises from SVMs, mostly relying on heuristics for breaking the problem down into smaller, more manageable chunks.

- Another approach is to use an interior-point method that uses Newton-like iterations to find a solution of the Karush–Kuhn–Tucker conditions of the primal and dual problems.

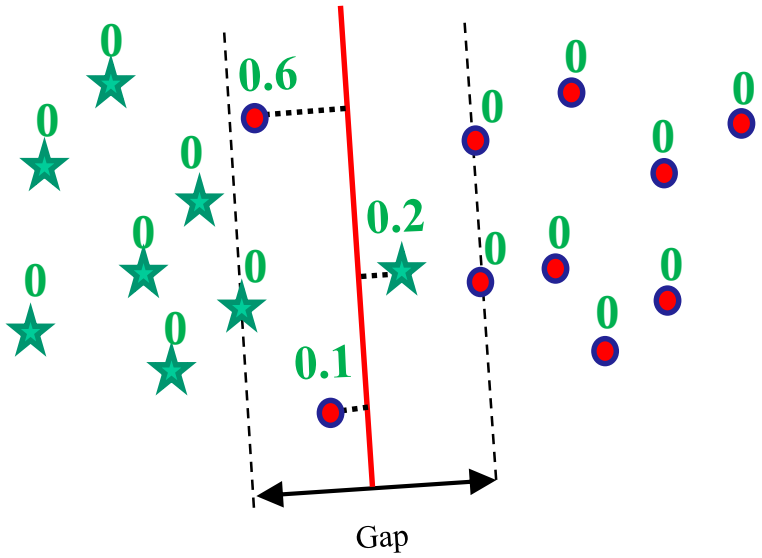
- Another common method is Platt's sequential minimal optimization (SMO) algorithm, which breaks the problem down into 2-dimensional sub-problems that are solved analytically, eliminating the need for a numerical optimization algorithm and matrix storage.

Case 2: Not linearly separable data; "Soft-margin" linear SVM

What if the data is not linearly separable?

- E.g., there are outliers or noisy measurements, or the data is slightly non-linear.

Want to handle this case without changing the family of decision functions.



Approach:

Assign a "slack variable" to each instance $\xi_i \geq 0$, which can be thought of distance from the separating hyperplane if an instance is misclassified and 0 otherwise.

Want to **minimize** $\frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{N}\xi_i$ subject to $y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 - \xi_i$ for $i = 1,\ldots,N$

Then given a new instance x, the classifier is $f(x) = sign(\vec{w}\cdot\vec{x} + b)$

Two formulations of soft-margin linear SVM

Primal formulation:

Minimize $\frac{1}{2}\sum_{i=1}^{n} w_i^2 + C\sum_{i=1}^{N} \xi_i$ subject to $y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 - \xi_i$ for $i = 1,\ldots,N$

Objective function — Constraints

Dual formulation:

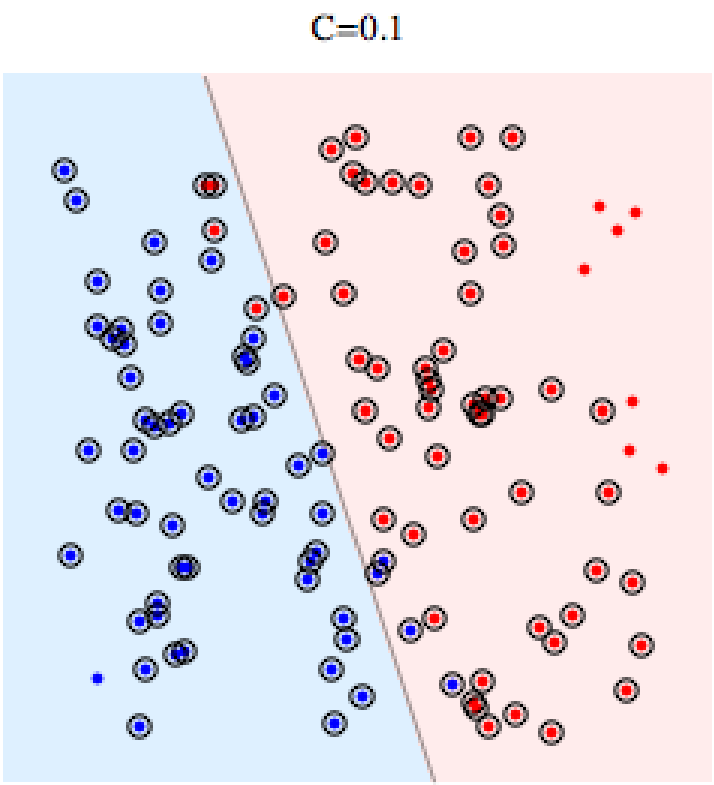
Maximize $\sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j$ subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$

Objective function — Constraints

for $i = 1,\ldots, N$.

PARAMETER C IN SOFT-MARGIN SVM

Minimize $\frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{N}\xi_i$ subject to $y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1-\xi_i$ for $i = 1,\ldots,N$

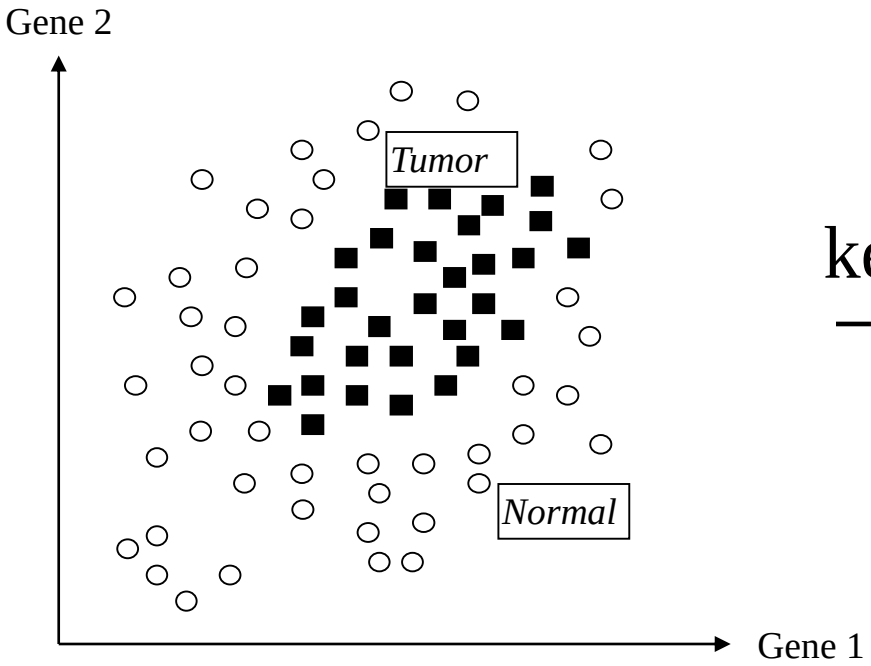


- When C is very large, the soft-margin SVM is equivalent to hard-margin SVM;
- When C is very small, we admit misclassifications in the training data at the expense of having w-vector with small norm;
- C has to be selected for the distribution at hand as it will be discussed later in this tutorial.

Maximize $\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \vec{x}_i\cdot\vec{x}_j$ subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N}\alpha_i y_i = 0$ for $i = 1,\ldots,N$.

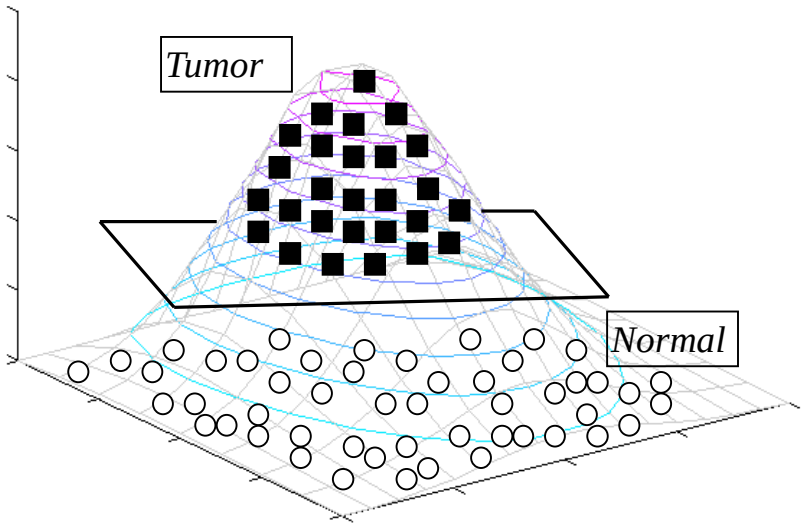
MARQUETTE UNIVERSITY
Be The Difference.

CASE 3: NOT LINEARLY SEPARABLE DATA; KERNEL TRICK



kernel

Φ



Data is not linearly separable in the input space

Data is linearly separable in the feature space obtained by a kernel

$$\Phi : \mathbf{R}^N \rightarrow \mathbf{H}$$

KERNEL TRICK

Original data $\vec{x}$ (in input space)

$$f(x) = sign(\vec{w} \cdot \vec{x} + b)$$

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i$$

Data in a higher dimensional feature space $\Phi(\vec{x})$

$$f(x) = sign(\vec{w} \cdot \Phi(\vec{x}) + b)$$

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\vec{x}_i)$$

$$f(x) = sign(\sum_{i=1}^{N} \alpha_i y_i \Phi(\vec{x}_i) \cdot \Phi(\vec{x}) + b)$$

$$f(x) = sign(\sum_{i=1}^{N} \alpha_i y_i K(\vec{x}_i, \vec{x}) + b)$$

Therefore, we do not need to know Φ explicitly, we just need to define function $K(\cdot, \cdot)$: $\mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$.

Not every function $\mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ can be a valid kernel; it has to satisfy so-called Mercer conditions. Otherwise, the underlying quadratic program may not be solvable.

POPULAR KERNELS

A kernel is a dot product in *some* feature space:

$$K(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j)$$

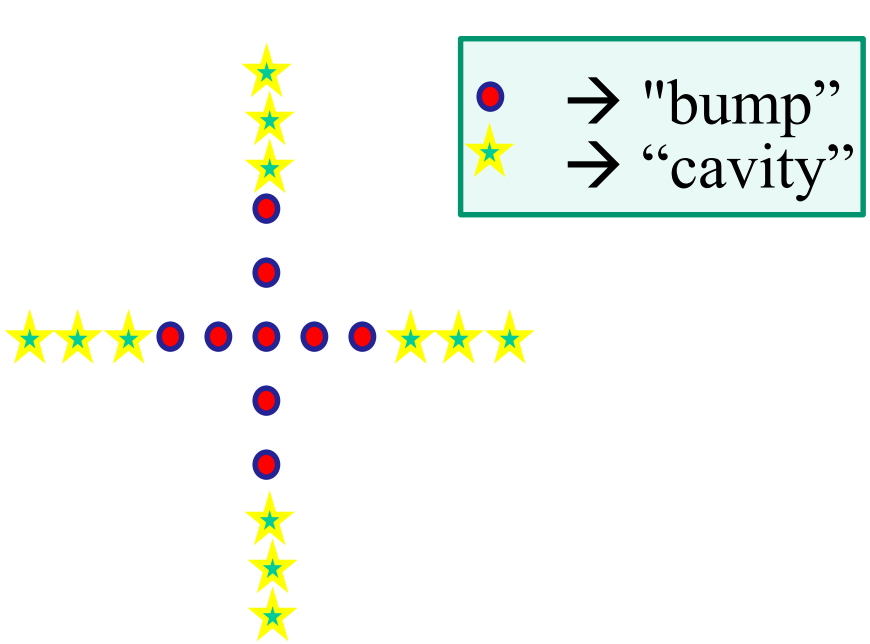
Examples:

$K(\vec{x}_i, \vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$	**Linear kernel**
$K(\vec{x}_i, \vec{x}_j) = \exp(-\gamma \left\| \vec{x}_i - \vec{x}_j \right\|^2)$	**Gaussian kernel**
$K(\vec{x}_i, \vec{x}_j) = \exp(-\gamma \left\| \vec{x}_i - \vec{x}_j \right\|)$	**Exponential kernel**
$K(\vec{x}_i, \vec{x}_j) = (p + \vec{x}_i \cdot \vec{x}_j)^q$	**Polynomial kernel**
$K(\vec{x}_i, \vec{x}_j) = (p + \vec{x}_i \cdot \vec{x}_j)^q \exp(-\gamma \left\| \vec{x}_i - \vec{x}_j \right\|^2)$	**Hybrid kernel**
$K(\vec{x}_i, \vec{x}_j) = \tanh(k\vec{x}_i \cdot \vec{x}_j - \delta)$	**Sigmoidal**

UNDERSTANDING THE GAUSSIAN KERNEL

Consider Gaussian kernel: $K(\vec{x}, \vec{x}_j) = \exp(-\gamma \left\| \vec{x} - \vec{x}_j \right\|^2)$

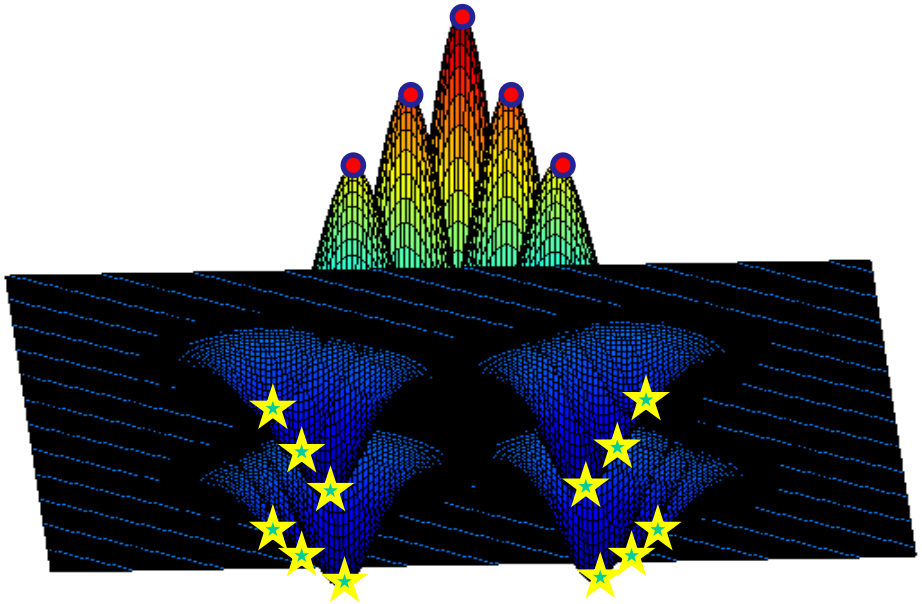
Geometrically, this is a "bump" or "cavity" centered at the training data point $\vec{x}_j$:

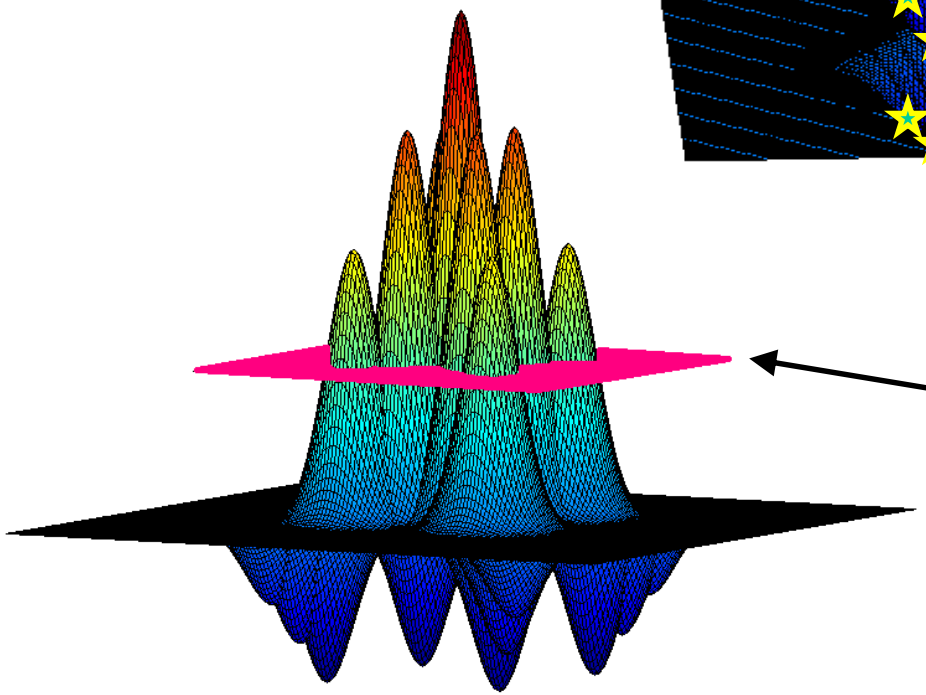


The resulting mapping function is a **combination** of bumps and cavities.

Understanding the Gaussian kernel

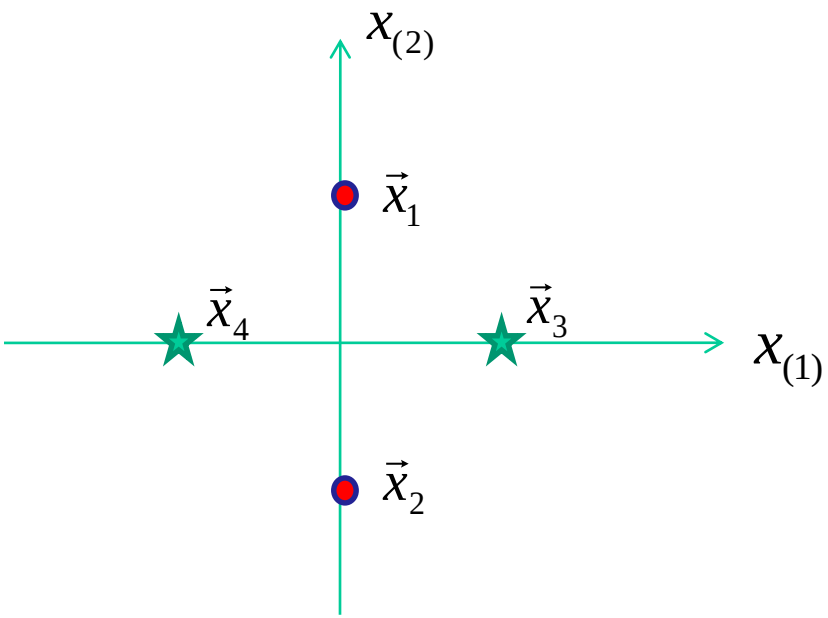
Several more views of the data is mapped to the feature space by Gaussian kernel





Linear hyperplane that separates two classes

UNDERSTANDING THE POLYNOMIAL KERNEL



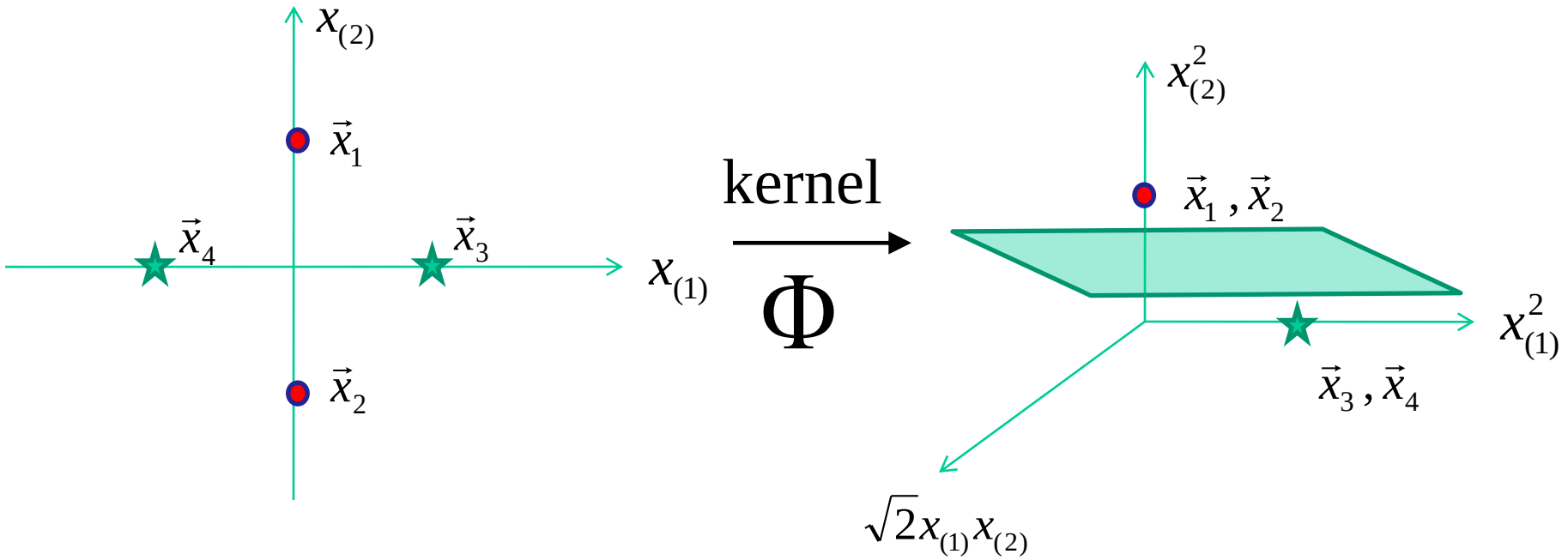
- Data is not linearly separable in the input space ($\mathbb{R}^2$).
- Apply kernel $K(\vec{x},\vec{z}) = (\vec{x}\cdot\vec{z})^2$ to map data to a higher dimensional space (3-dimensional) where it is linearly separable.

$$K(\vec{x},\vec{z}) = (\vec{x}\cdot\vec{z})^2 = \left[\begin{pmatrix} x_{(1)} \\ x_{(2)} \end{pmatrix}\cdot\begin{pmatrix} z_{(1)} \\ z_{(2)} \end{pmatrix}\right]^2 = \left[x_{(1)}z_{(1)} + x_{(2)}z_{(2)}\right]^2 =$$

$$= x_{(1)}^2 z_{(1)}^2 + 2x_{(1)}z_{(1)}x_{(2)}z_{(2)} + x_{(2)}^2 z_{(2)}^2 = \begin{pmatrix} x_{(1)}^2 \\ \sqrt{2}x_{(1)}x_{(2)} \\ x_{(2)}^2 \end{pmatrix}\cdot\begin{pmatrix} z_{(1)}^2 \\ \sqrt{2}z_{(1)}z_{(2)} \\ z_{(2)}^2 \end{pmatrix} = \Phi(\vec{x})\cdot\Phi(\vec{z})$$

UNDERSTANDING THE POLYNOMIAL KERNEL

Therefore, the explicit mapping is $\Phi(\vec{x}) = \begin{pmatrix} x_{(1)}^2 \\ \sqrt{2}x_{(1)}x_{(2)} \\ x_{(2)}^2 \end{pmatrix}$

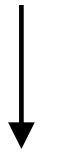


Example of benefits of using a kernel

Consider polynomial kernel: $K(\vec{x}_i, \vec{x}_j) = (1 + \vec{x}_i \cdot \vec{x}_j)^3$

Assume that we are dealing with 2-dimensional data (i.e., in $\mathbb{R}^2$). Where will this kernel map the data?

2-dimensional space $x_{(1)}$ $x_{(2)}$



kernel

10-dimensional space

1 $x_{(1)}$ $x_{(2)}$ $x_{(1)}^2$ $x_{(2)}^2$ $x_{(1)}x_{(2)}$ $x_{(1)}^3$ $x_{(2)}^3$ $x_{(1)}x_{(2)}^2$ $x_{(1)}^2x_{(2)}$

COMPARISON WITH METHODS FROM CLASSICAL STATISTICS & REGRESSION

- Need ≥ 5 samples for each parameter of the regression model to be estimated:

Number of variables	Polynomial degree	Number of parameters	Required sample
2	3	10	50
10	3	286	1,430
10	5	3,003	15,015
100	3	176,851	884,255
100	5	96,560,646	482,803,230

- SVMs do not have such requirement & often require much less sample than the number of variables, even when a high-degree polynomial kernel is used.

Part 1

BASIC PRINCIPLES OF STATISTICAL MACHINE LEARNING

Generalization and overfitting

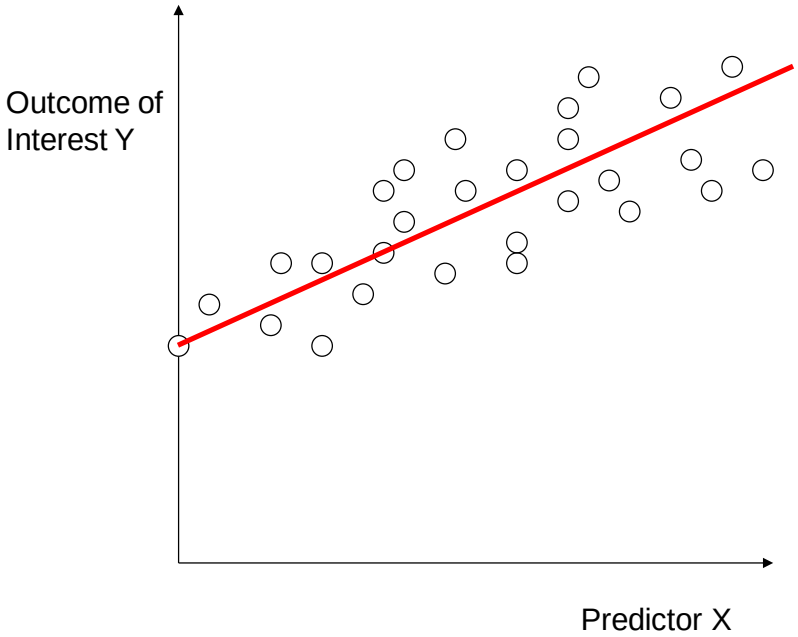
- **Generalization:** A classifier or a regression algorithm learns to correctly predict output from given inputs not only in previously seen samples but also in previously unseen samples.

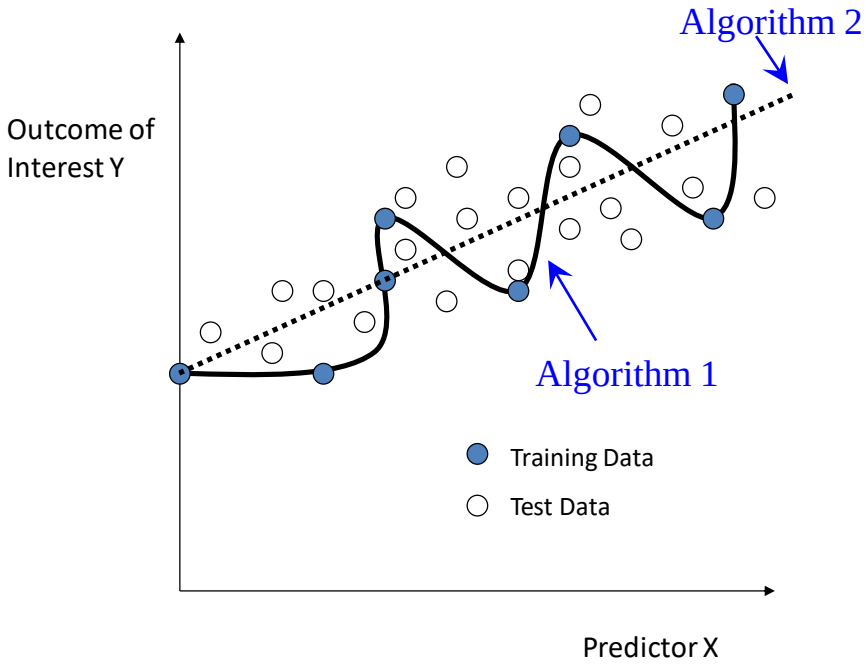
- **Overfitting:** A classifier or a regression algorithm learns to correctly predict output from given inputs in previously seen samples but fails to do so in previously unseen samples.

- **Overfitting → Poor generalization**.

Example of overfitting and generalization

There is a linear relationship between predictor and outcome (plus some Gaussian noise).





- Algorithm 1 learned non-reproducible peculiarities of the specific sample available for learning but did not learn the general characteristics of the function that generated the data. Thus, it is overfitted and has poor generalization.
- Algorithm 2 learned general characteristics of the function that produced the data. Thus, it generalizes.

"Loss + penalty" paradigm for learning to avoid overfitting and ensure generalization

- Many statistical learning algorithms (including SVMs) search for a decision function by solving the following optimization problem:

  Minimize (*Loss* + λ *Penalty*)

- ***Loss*** measures error of fitting the data
- ***Penalty*** penalizes complexity of the learned function
- λ is regularization parameter that balances ***Loss*** and ***Penalty***

SVMs IN "LOSS + PENALTY" FORM

SVMs build the following classifiers: $f(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b)$

Consider soft-margin linear SVM formulation:

Find $\vec{w}$ and b that

Minimize $\frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{N}\xi_i$ subject to $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i$ for $i = 1,\ldots,N$

This can also be stated as:

Find $\vec{w}$ and b that

Minimize $\sum_{i=1}^{N}[1 - y_i f(\vec{x}_i)]_+ + \lambda\|\vec{w}\|_2^2$

Loss ("hinge loss") *Penalty*

(in fact, one can show that $\lambda = 1/(2C)$).

Meaning of SVM loss function

Consider loss function: $\sum_{i=1}^{N}[1-y_i f(\vec{x}_i)]_+$

- Recall that $[\ldots]_+$ indicates the positive part
- For a given sample/patient *i*, the loss is non-zero if $1-y_i f(\vec{x}_i)>0$
- In other words, $y_i f(\vec{x}_i)<1$

- Since $y_i=\{-1,+1\}$, this means that the loss is non-zero if

$f(\vec{x}_i)<1$ for $y_i = +1$

$f(\vec{x}_i)>-1$ for $y_i = -1$

- In other words, the loss is non-zero if

$\vec{w}\cdot\vec{x}_i+b<1$ for $y_i = +1$

$\vec{w}\cdot\vec{x}_i+b>-1$ for $y_i = -1$

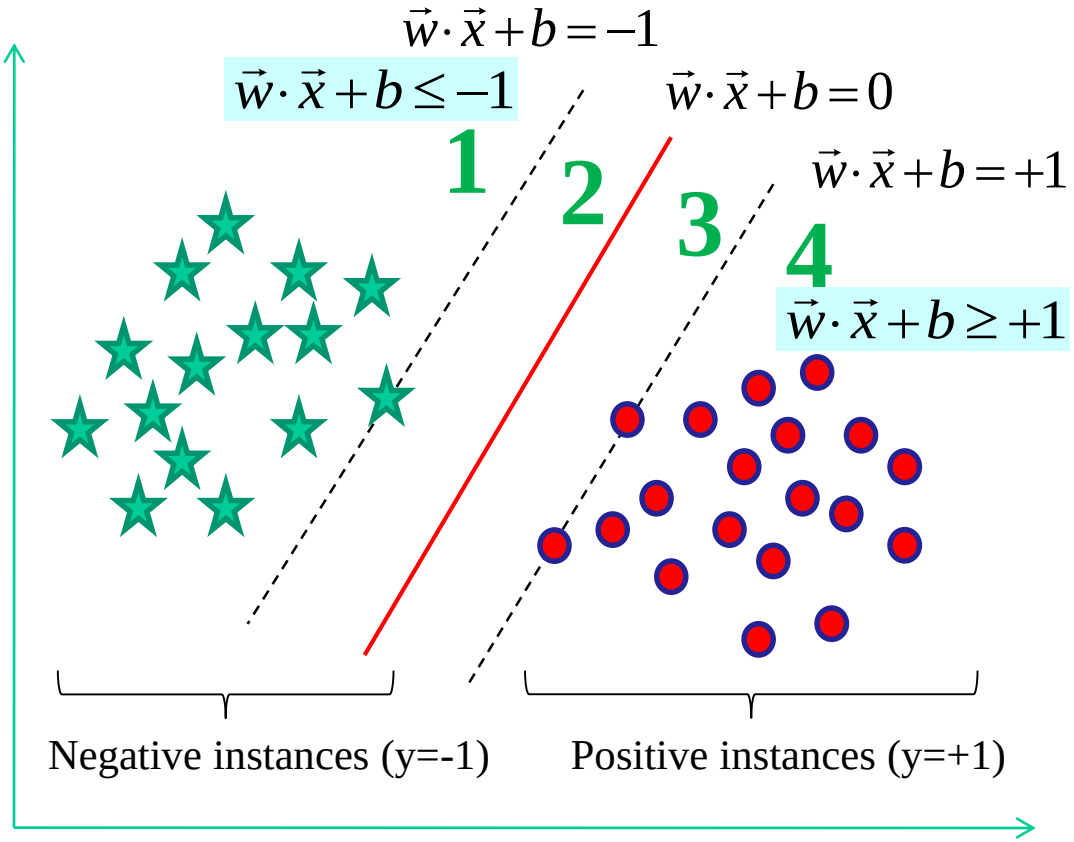
Meaning of SVM loss function

- In other words, the loss is non-zero if

$\vec{w} \cdot \vec{x}_i + b < 1$ for $y_i = +1$

$\vec{w} \cdot \vec{x}_i + b > -1$ for $y_i = -1$

- If the instance is positive, it is penalized only in regions 1,2,3
- If the instance is negative, it is penalized only in regions 2,3,4



Flexibility of "loss + penalty" framework

Minimize (*Loss* + λ *Penalty*)

Loss function	Penalty function	Resulting algorithm
Hinge loss: $\sum_{i=1}^{N}[1-y_i f(\vec{x}_i)]_+$	$\lambda\|\vec{w}\|_2^2$	**SVMs**
Mean squared error: $\sum_{i=1}^{N}(y_i - f(\vec{x}_i))^2$	$\lambda\|\vec{w}\|_2^2$	**Ridge regression**
Mean squared error: $\sum_{i=1}^{N}(y_i - f(\vec{x}_i))^2$	$\lambda\|\vec{w}\|_1$	**Lasso**
Mean squared error: $\sum_{i=1}^{N}(y_i - f(\vec{x}_i))^2$	$\lambda_1\|\vec{w}\|_1 + \lambda_2\|\vec{w}\|_2^2$	**Elastic net**
Hinge loss: $\sum_{i=1}^{N}[1-y_i f(\vec{x}_i)]_+$	$\lambda\|\vec{w}\|_1$	**1-norm SVM**

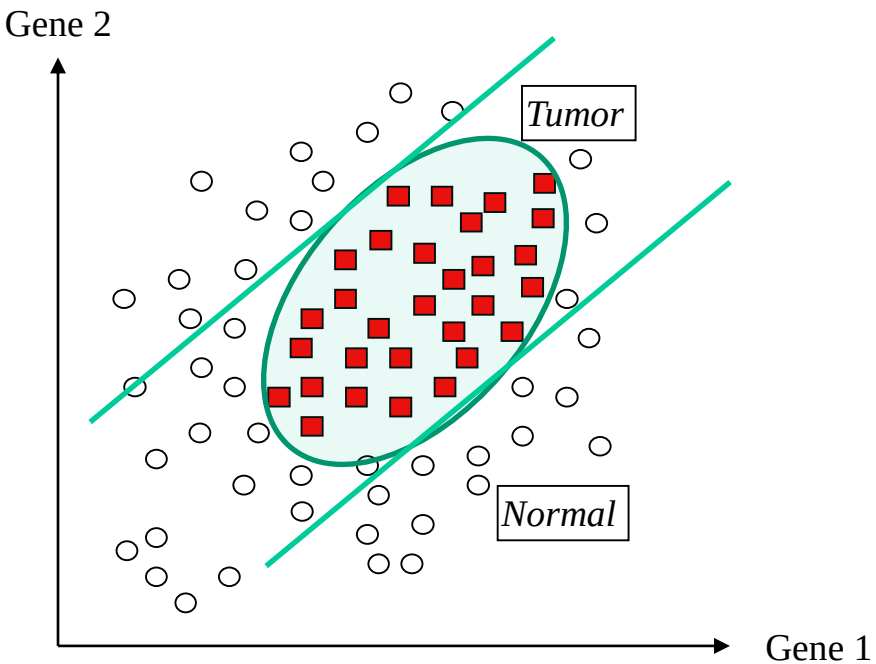
PART 2

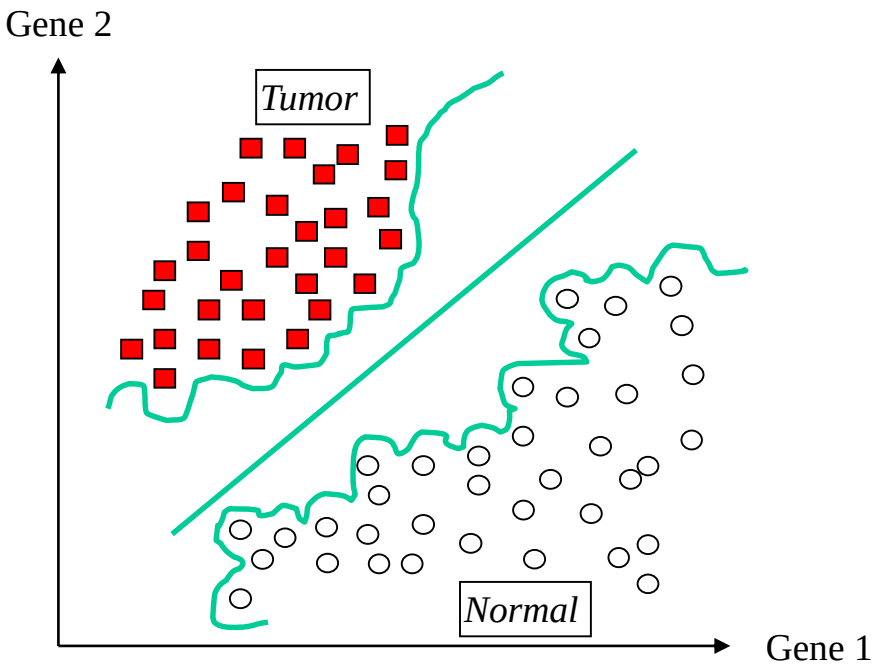
- Model selection for SVMs
- Extensions to the basic SVM model:
  1. SVMs for multicategory data
  2. Support vector regression
  3. Novelty detection with SVM-based methods
  4. Support vector clustering
  5. SVM-based variable selection
  6. Computing posterior class probabilities for SVM classifiers

Part 2

MODEL SELECTION FOR SVMS

Need for model selection for SVMs





- It is impossible to find a linear SVM classifier that separates tumors from normals!
- Need a non-linear SVM classifier, e.g. SVM with polynomial kernel of degree 2 solves this problem without errors.

- We should not apply a non-linear SVM classifier while we can perfectly solve this problem using a linear SVM classifier!

A DATA-DRIVEN APPROACH FOR MODEL SELECTION FOR SVMs

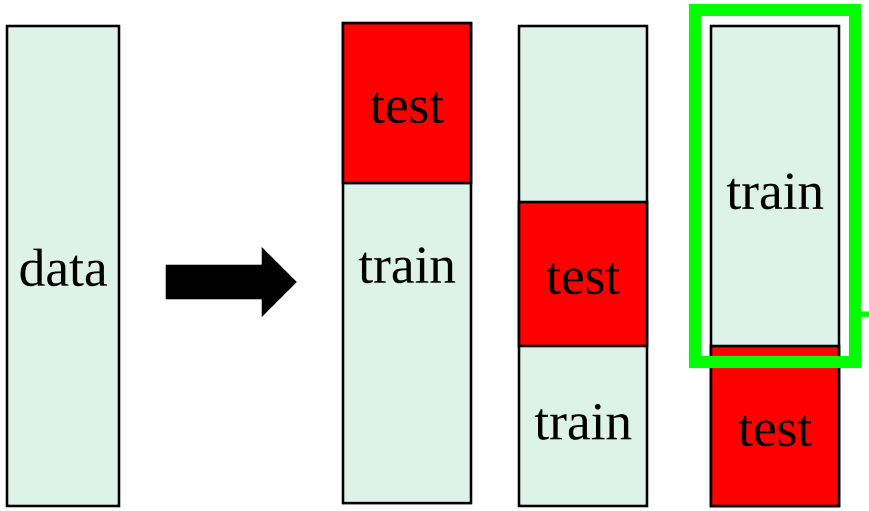
- Do not know *a priori* what type of SVM kernel and what kernel parameter(s) to use for a given dataset?
- Need to examine various combinations of parameters, e.g. consider searching the following grid:

	Polynomial degree *d*				
Parameter *C*	(0.1, 1)	(1, 1)	(10, 1)	(100, 1)	(1000, 1)
	(0.1, 2)	(1, 2)	(10, 2)	(100, 2)	(1000, 2)
	(0.1, 3)	(1, 3)	(10, 3)	(100, 3)	(1000, 3)
	(0.1, 4)	(1, 4)	(10, 4)	(100, 4)	(1000, 4)
	(0.1, 5)	(1, 5)	(10, 5)	(100, 5)	(1000, 5)

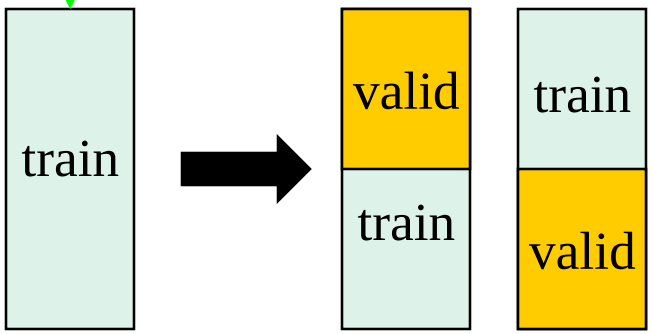
- How to search this grid while producing an unbiased estimate of classification performance?

NESTED CROSS-VALIDATION

Recall the main idea of cross-validation:



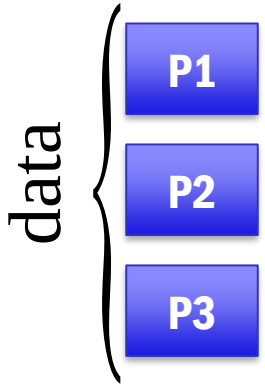
What combination of SVM parameters to apply on training data?



Perform "grid search" using another nested loop of cross-validation.

EXAMPLE OF NESTED CROSS-VALIDATION

Consider that we use 3-fold cross-validation and we want to optimize parameter C that takes values "1" and "2".



Outer Loop

Training set	Testing set	C	Accuracy	Average Accuracy
P1, P2	P3	1	89%	83%
P1,P3	P2	2	84%	
P2, P3	P1	1	76%	

Inner Loop

Training set	Validation set	C	Accuracy	Average Accuracy
P1	P2	1	86%	**85%**
P2	P1		84%	
P1	P2	2	70%	80%
P2	P1		90%	

choose **C**=1

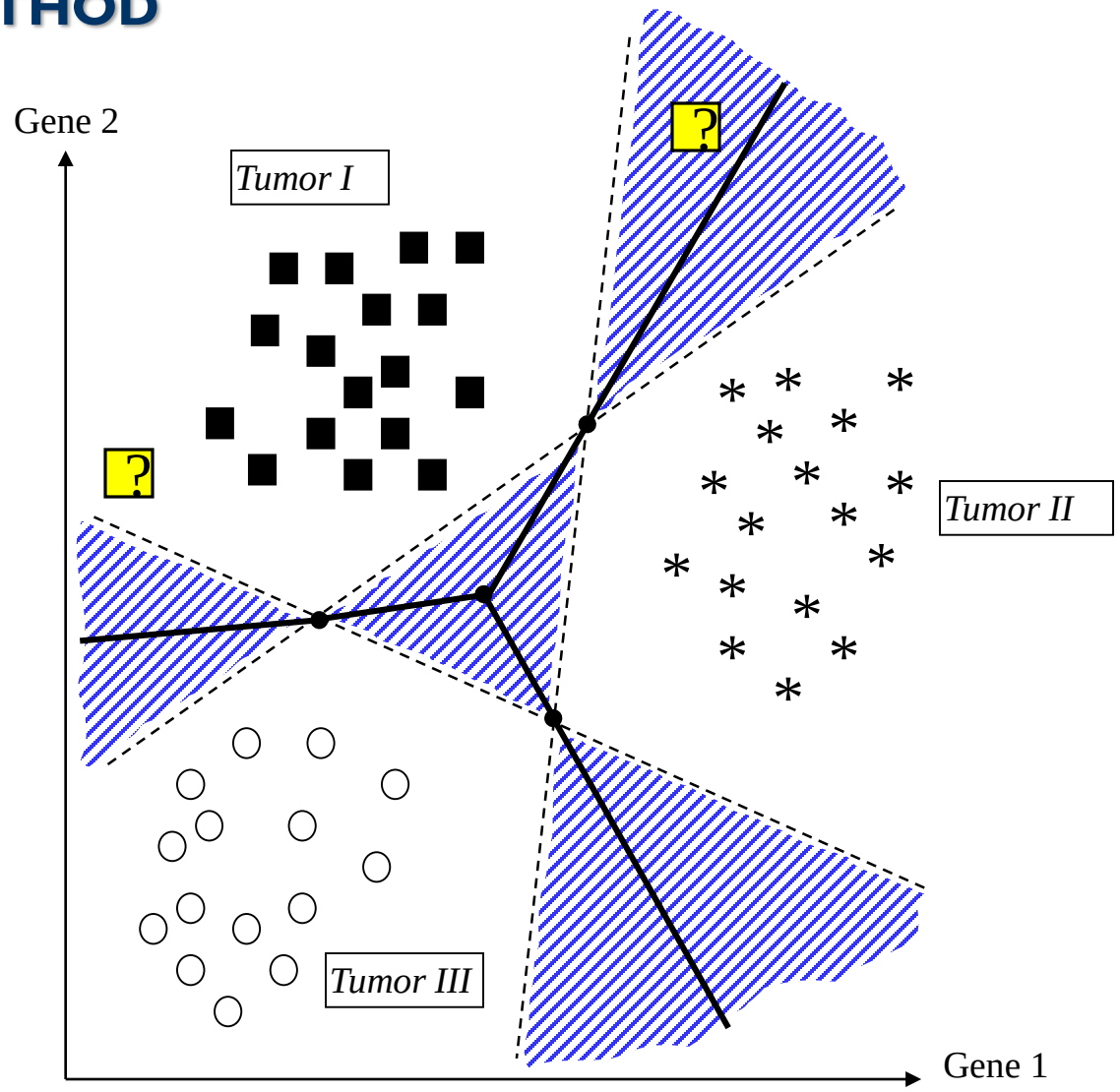
On use of cross-validation

- Empirically we found that cross-validation works well for model selection for SVMs in many problem domains;
- Many other approaches that can be used for model selection for SVMs exist, e.g.:
  - Generalized cross-validation
  - Bayesian information criterion (BIC)
  - Minimum description length (MDL)
  - Vapnik-Chernovenkis (VC) dimension
  - Bootstrap

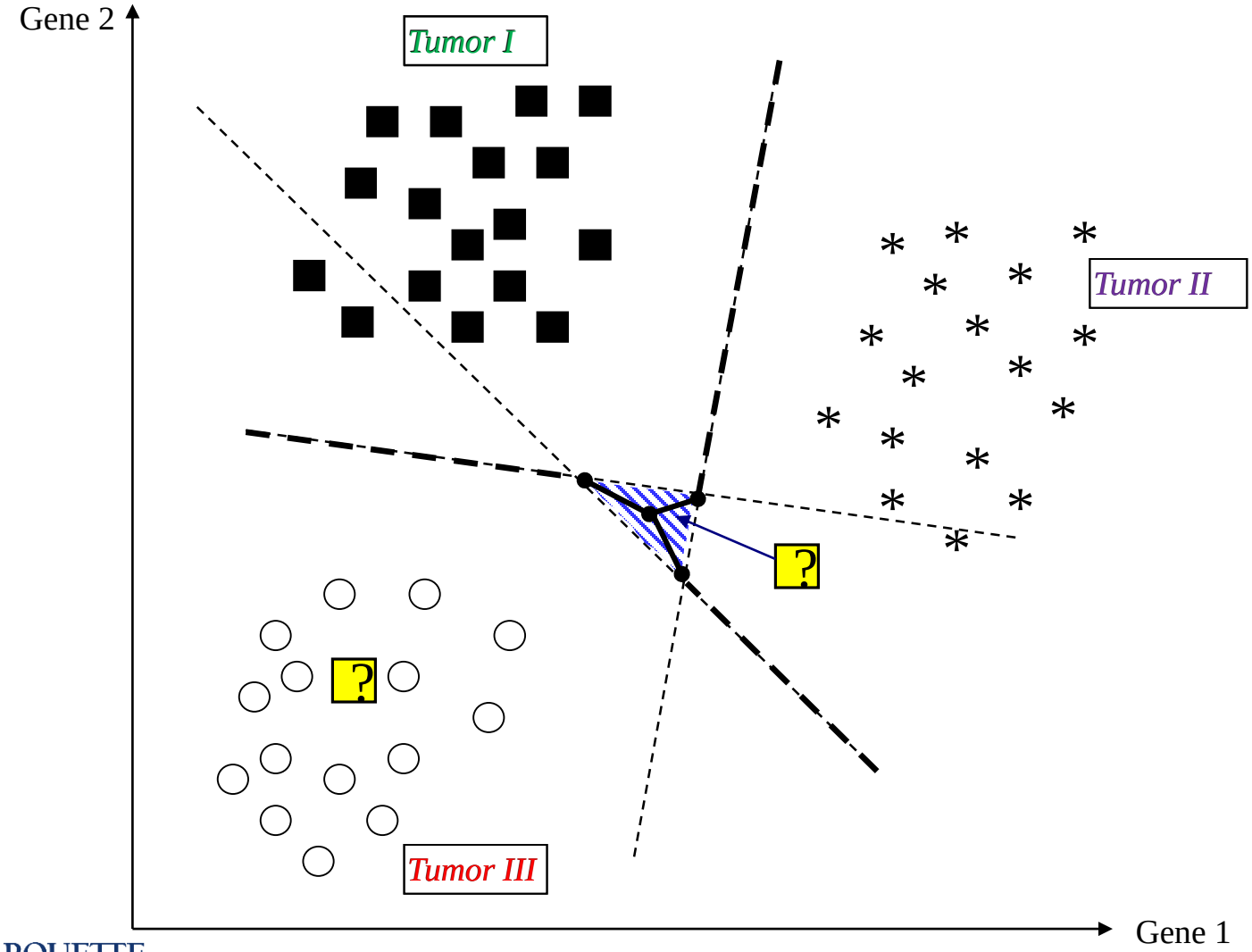
Part 2

SVMS FOR MULTICATEGORY DATA

ONE-VERSUS-REST MULTICATEGORY SVM METHOD



ONE-VERSUS-ONE MULTICATEGORY SVM METHOD



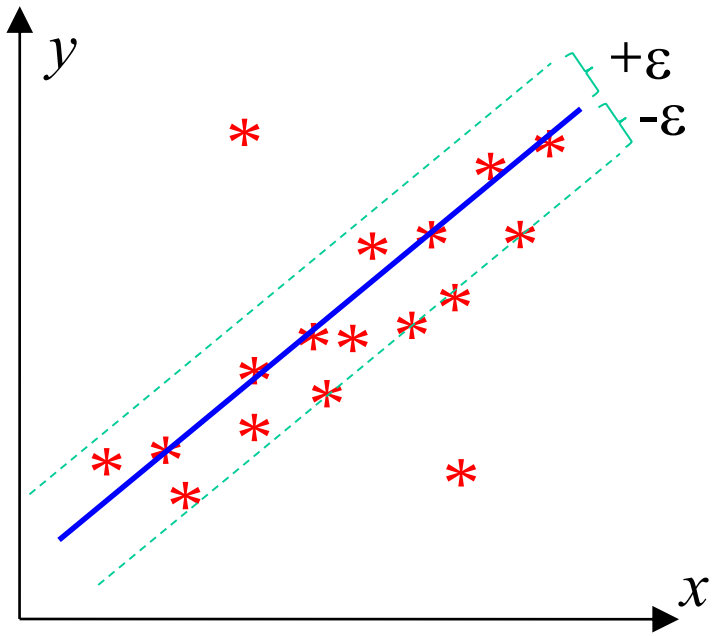
Part 2

SUPPORT VECTOR REGRESSION

ε-SUPPORT VECTOR REGRESSION (ε-SVR)

Given training data: $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N \in R^p$

$y_1, y_2, \ldots, y_N \in R$



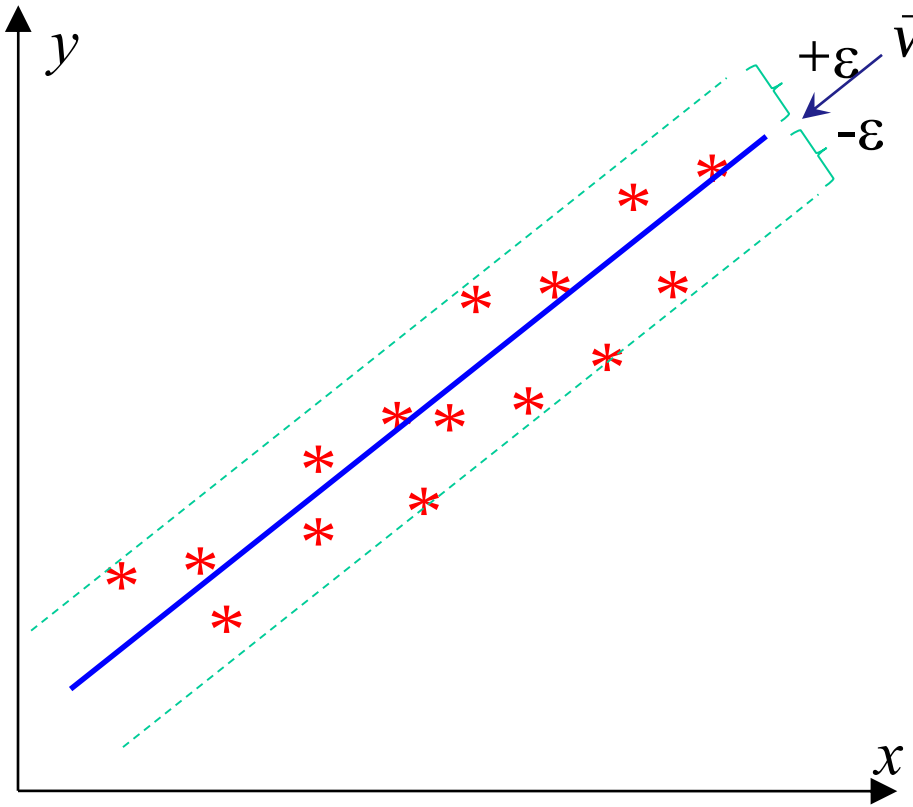
Main idea:

Find a function $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$ that **approximates** $y_1, \ldots, y_N$:

- it **has at most** ε **derivation from the true values** y_i
- it is as "flat" as possible (to avoid overfitting)

E.g., build a model to predict survival of cancer patients that can admit a one month error (= ε).

FORMULATION OF "HARD-MARGIN" ε-SVR



Find $f(\vec{x}) = \vec{w}\cdot\vec{x}+b$

by **minimizing** $\frac{1}{2}\|\vec{w}\|^2$

subject to constraints:

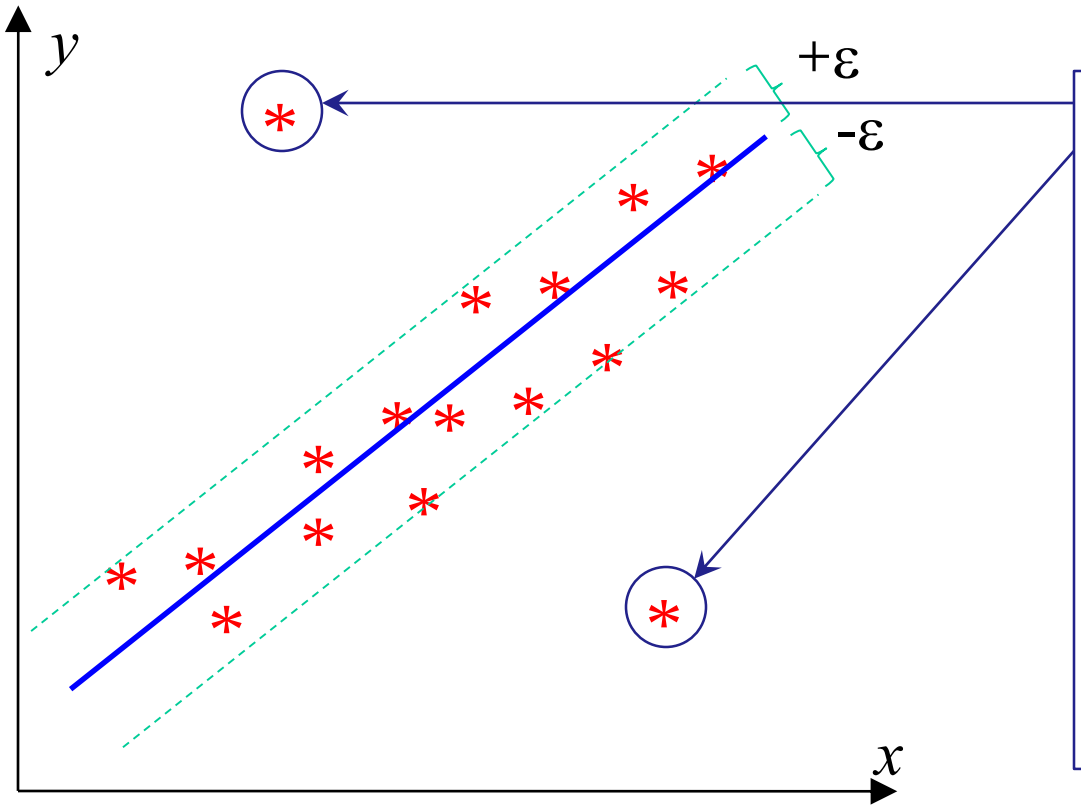
$$y_i - (\vec{w}\cdot\vec{x}+b) \le \varepsilon$$

$$y_i - (\vec{w}\cdot\vec{x}+b) \ge -\varepsilon$$

for $i = 1,\ldots,N$.

I.e., difference between y_i and the fitted function should be smaller than ε and larger than -ε ⇔ all points y_i should be in the "ε-ribbon" around the fitted function.

FORMULATION OF "SOFT-MARGIN" ε-SVR

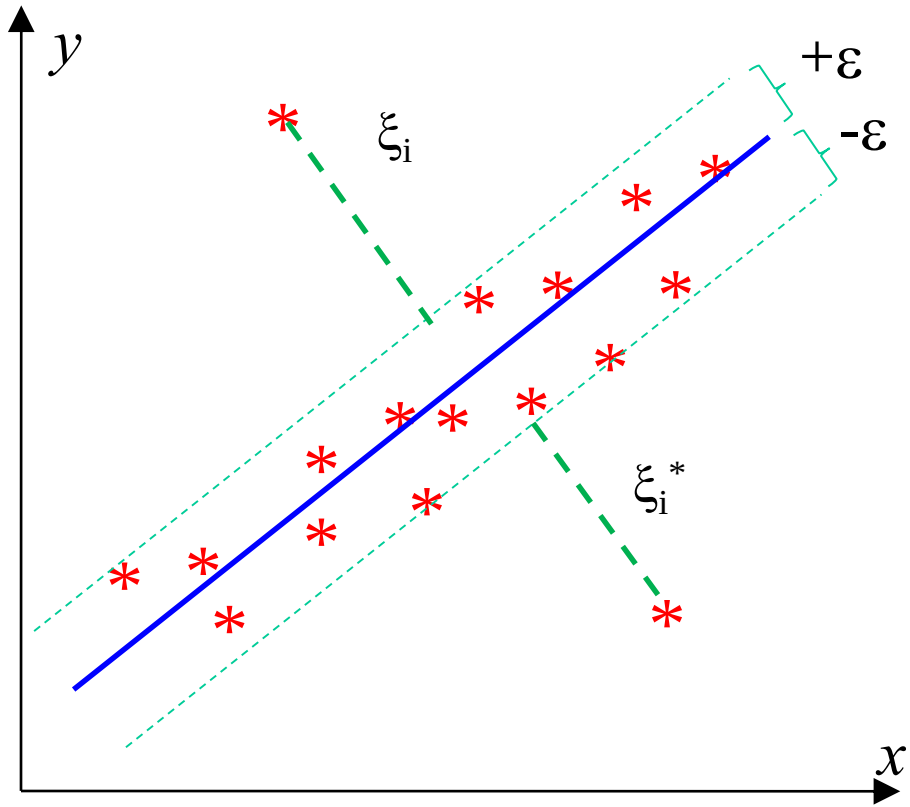


If we have points like this (e.g., outliers or noise) we can either:

a) increase ε to ensure that these points are within the new ε-ribbon, or

b) assign a penalty ("slack" variable) to each of this points (as was done for "soft-margin" SVMs)

FORMULATION OF "SOFT-MARGIN" ε-SVR



Find $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$

by **minimizing** $\frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$

subject to constraints:

$$y_i - (\vec{w} \cdot \vec{x} + b) \le \varepsilon + \xi_i$$

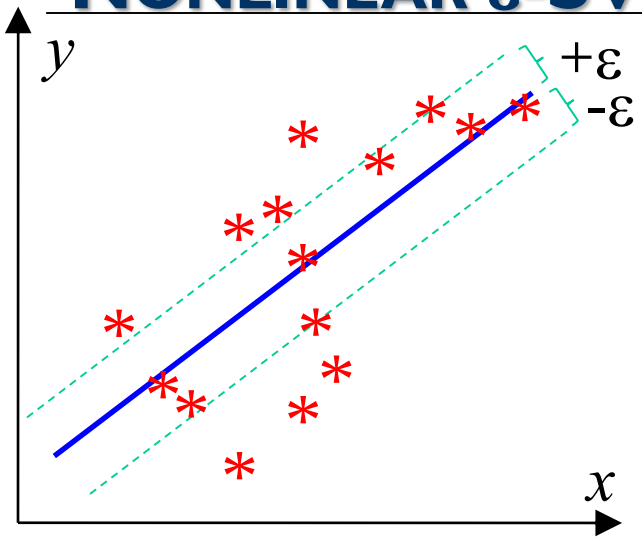
$$y_i - (\vec{w} \cdot \vec{x} + b) \ge -\varepsilon - \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$

for $i = 1,\ldots,N$.

Notice that only points outside ε-ribbon are penalized!

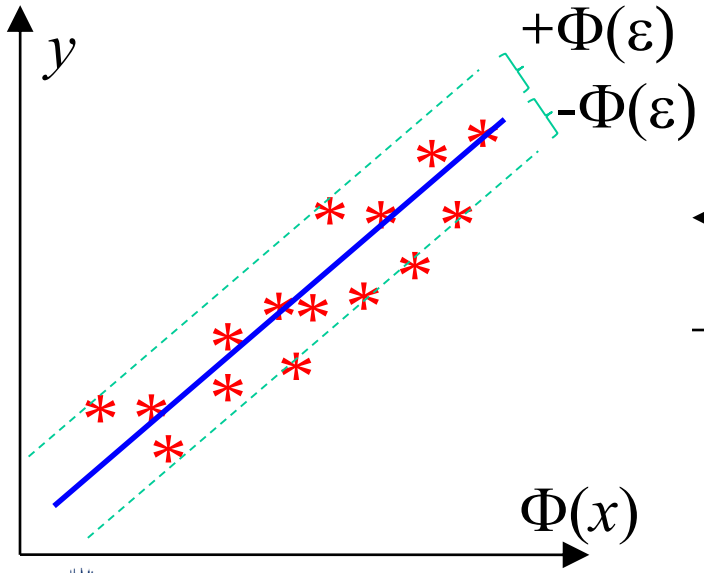
NONLINEAR ε-SVR



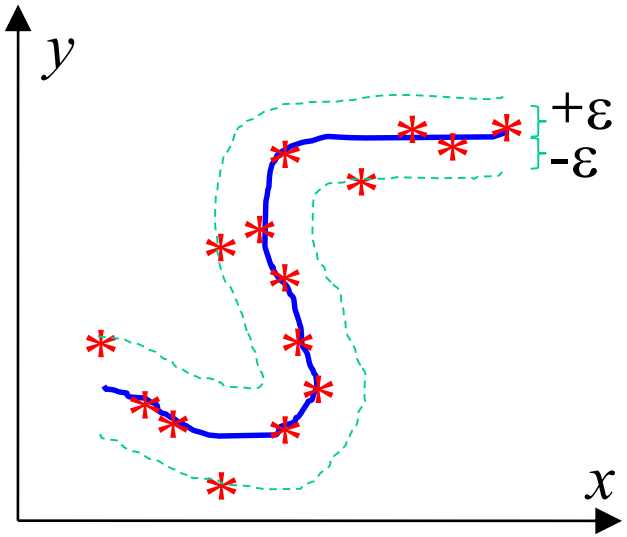
Cannot approximate well this function with small ε!

kernel

Φ



Φ^{-1}



MARQUETTE UNIVERSITY
Be The Difference.

ε-SUPPORT VECTOR REGRESSION IN "LOSS + PENALTY" FORM

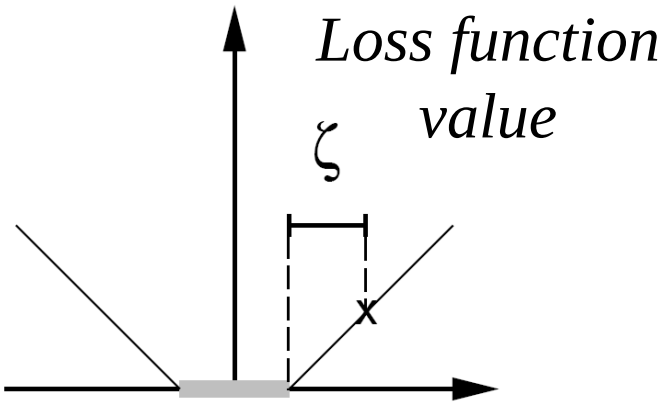
Build decision function of the form: $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$

Find $\vec{w}$ and b that

Minimize $\sum_{i=1}^{N} \max(0, |y_i - f(\vec{x}_i)| - \varepsilon) + \lambda \|\vec{w}\|_2^2$

Loss *Penalty*

(*"linear* ε-insensitive loss*")*

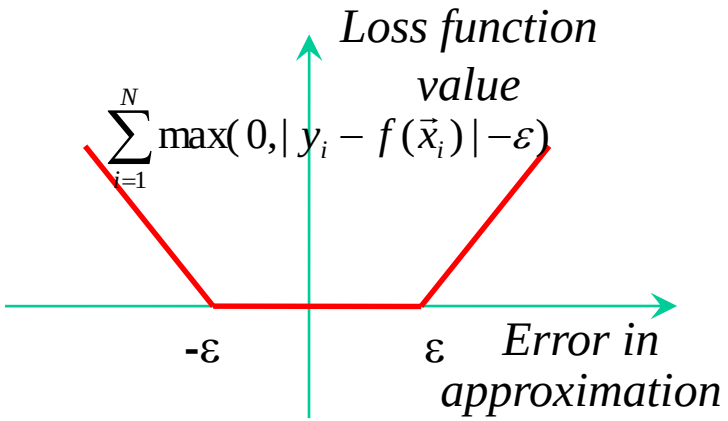


Comparing ε-SVR with popular regression methods

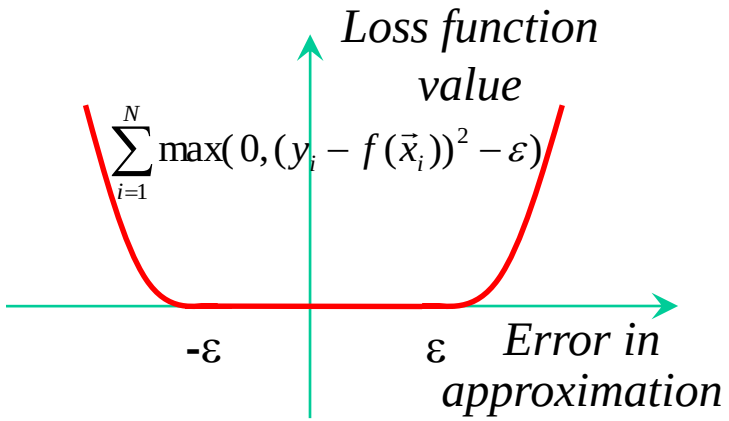
Loss function	Penalty function	Resulting algorithm
Linear ε-insensitive loss: $\sum_{i=1}^{N} \max(0, \lvert y_i - f(\vec{x}_i) \rvert - \varepsilon)$	$\lambda \lVert \vec{w} \rVert_2^2$	**ε-SVR**
Quadratic ε-insensitive loss: $\sum_{i=1}^{N} \max(0, (y_i - f(\vec{x}_i))^2 - \varepsilon)$	$\lambda \lVert \vec{w} \rVert_2^2$	**Another variant of ε-SVR**
Mean squared error: $\sum_{i=1}^{N} (y_i - f(\vec{x}_i))^2$	$\lambda \lVert \vec{w} \rVert_2^2$	**Ridge regression**
Mean linear error: $\sum_{i=1}^{N} \lvert y_i - f(\vec{x}_i) \rvert$	$\lambda \lVert \vec{w} \rVert_2^2$	**Another variant of ridge regression**

COMPARING LOSS FUNCTIONS OF REGRESSION METHODS

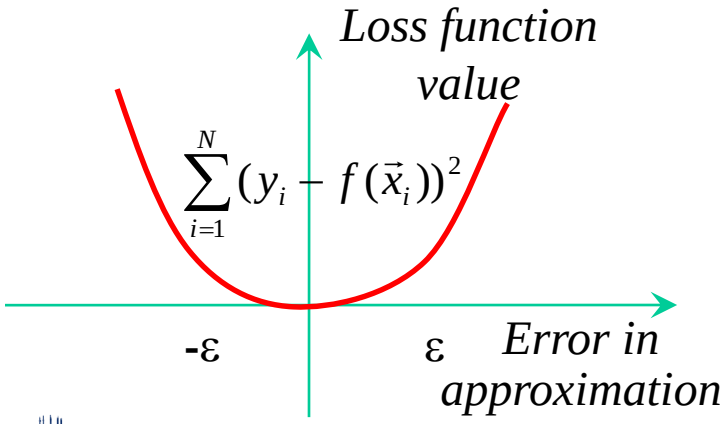
Linear ε-insensitive loss



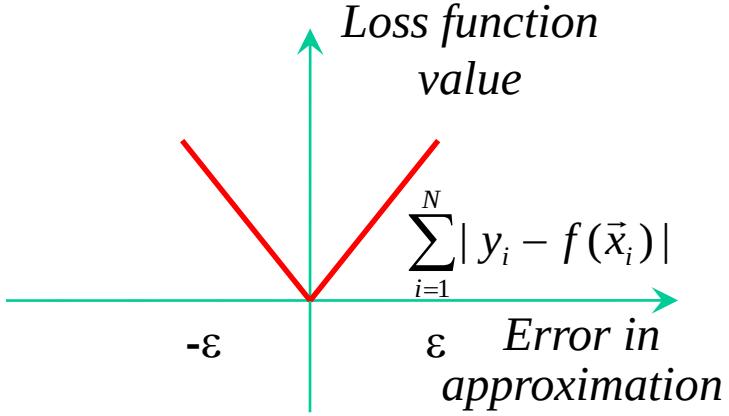
Quadratic ε-insensitive loss



Mean squared error



Mean linear error



MARQUETTE UNIVERSITY
Be The Difference.

APPLYING ε-SVR TO REAL DATA

- In the absence of domain knowledge about decision functions, it is recommended to optimize the following parameters
(e.g., by cross-validation using grid-search):

  - parameter C
  - parameter ε
  - kernel parameters (e.g., degree of polynomial)

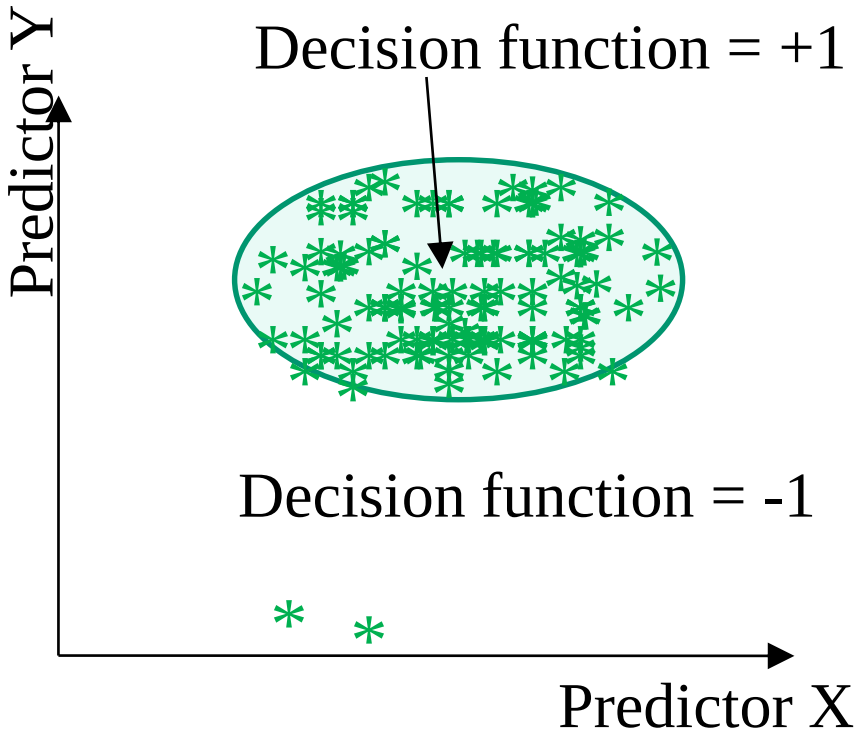
- Notice that parameter ε depends on the ranges of variables in the dataset; therefore it is recommended to normalize/re-scale data prior to applying ε-SVR.

Part 2

NOVELTY DETECTION WITH SVM-BASED METHODS

What is it about?

- Find the simplest and most compact region in the space of predictors where the majority of data samples "live" (i.e., with the highest density of samples).
- Build a decision function that takes value +1 in this region and -1 elsewhere.
- Once we have such a decision function, we can identify novel or outlier samples/patients in the data.



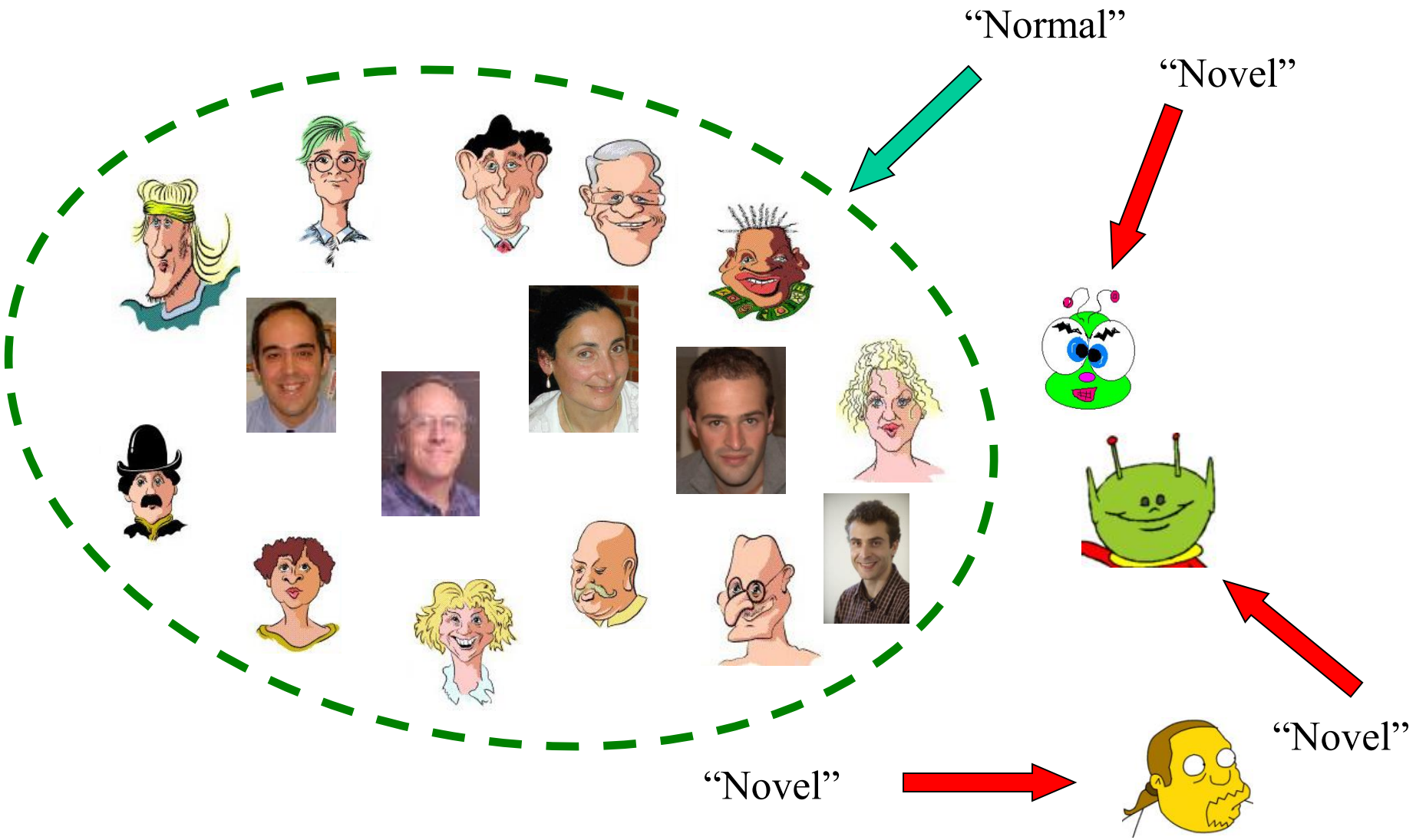
KEY ASSUMPTIONS

- We do not know classes/labels of samples (positive or negative) in the data available for learning
  ➔ this is not a classification problem

- All positive samples are similar but each negative sample can be different in its own way

Thus, do not need to collect data for negative samples!

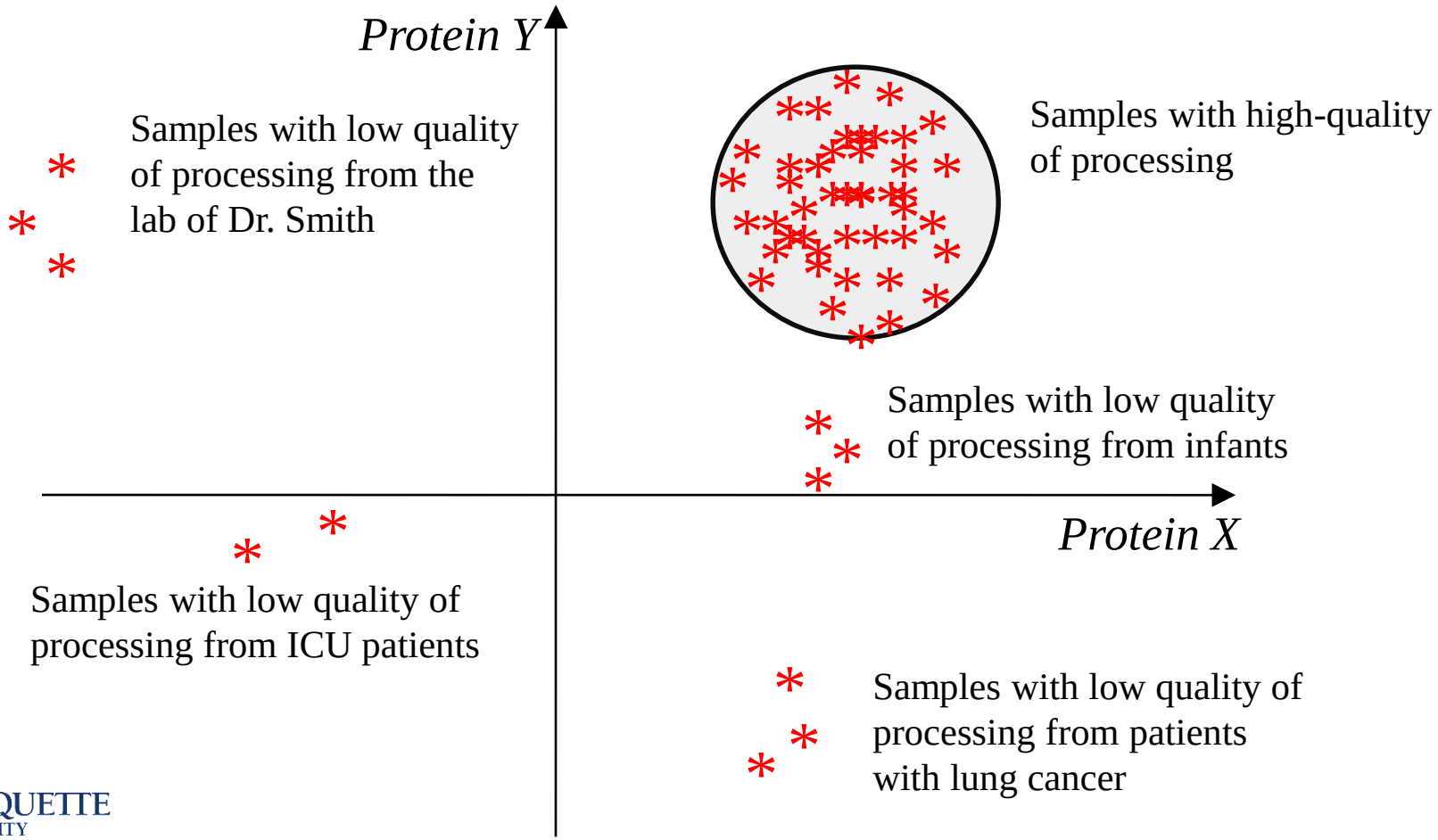
SAMPLE APPLICATIONS



Modified from: www.cs.huji.ac.il/course/2004/learns/NoveltyDetection.ppt

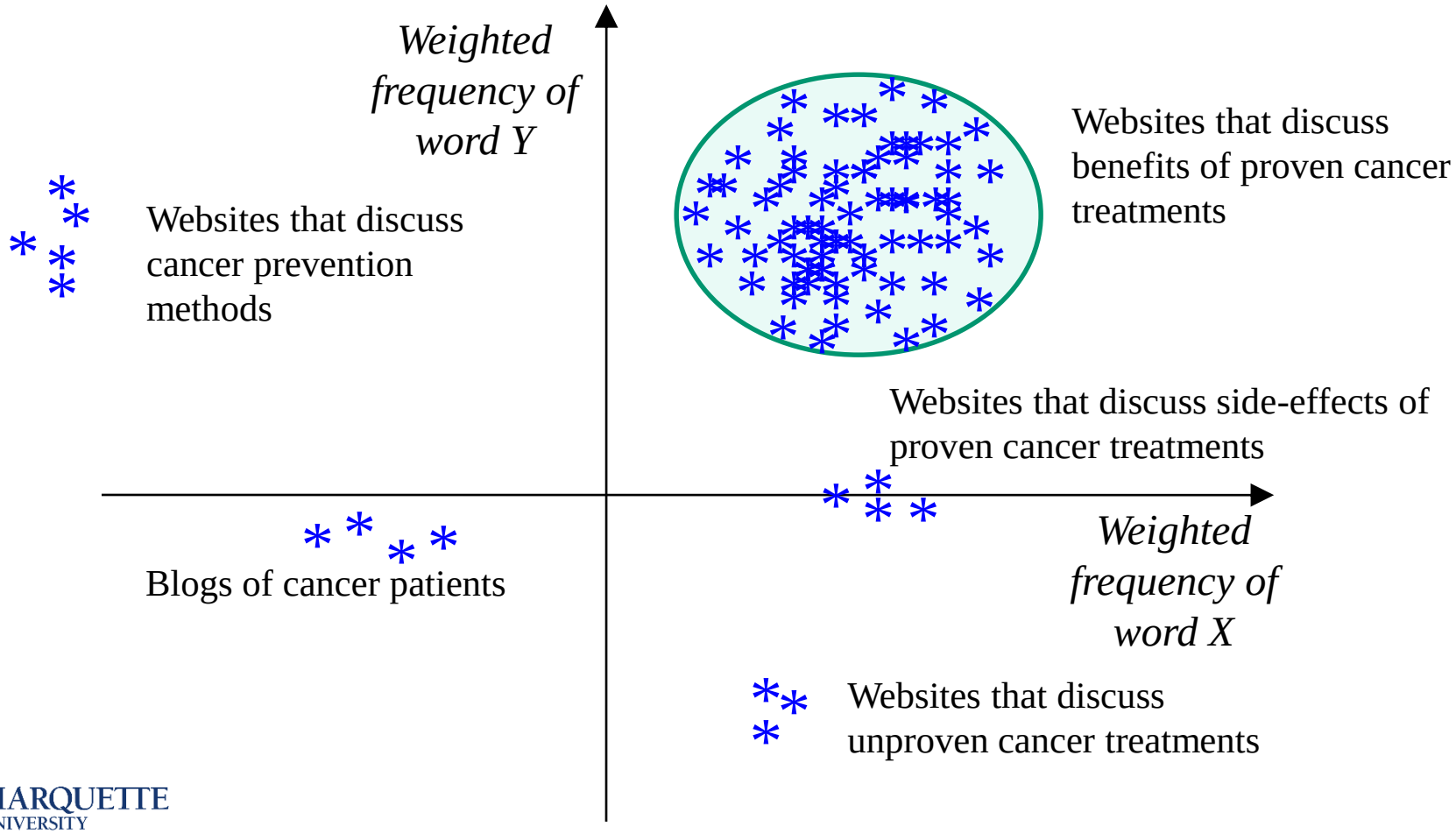
SAMPLE APPLICATIONS

Discover deviations in sample handling protocol when doing quality control of assays.



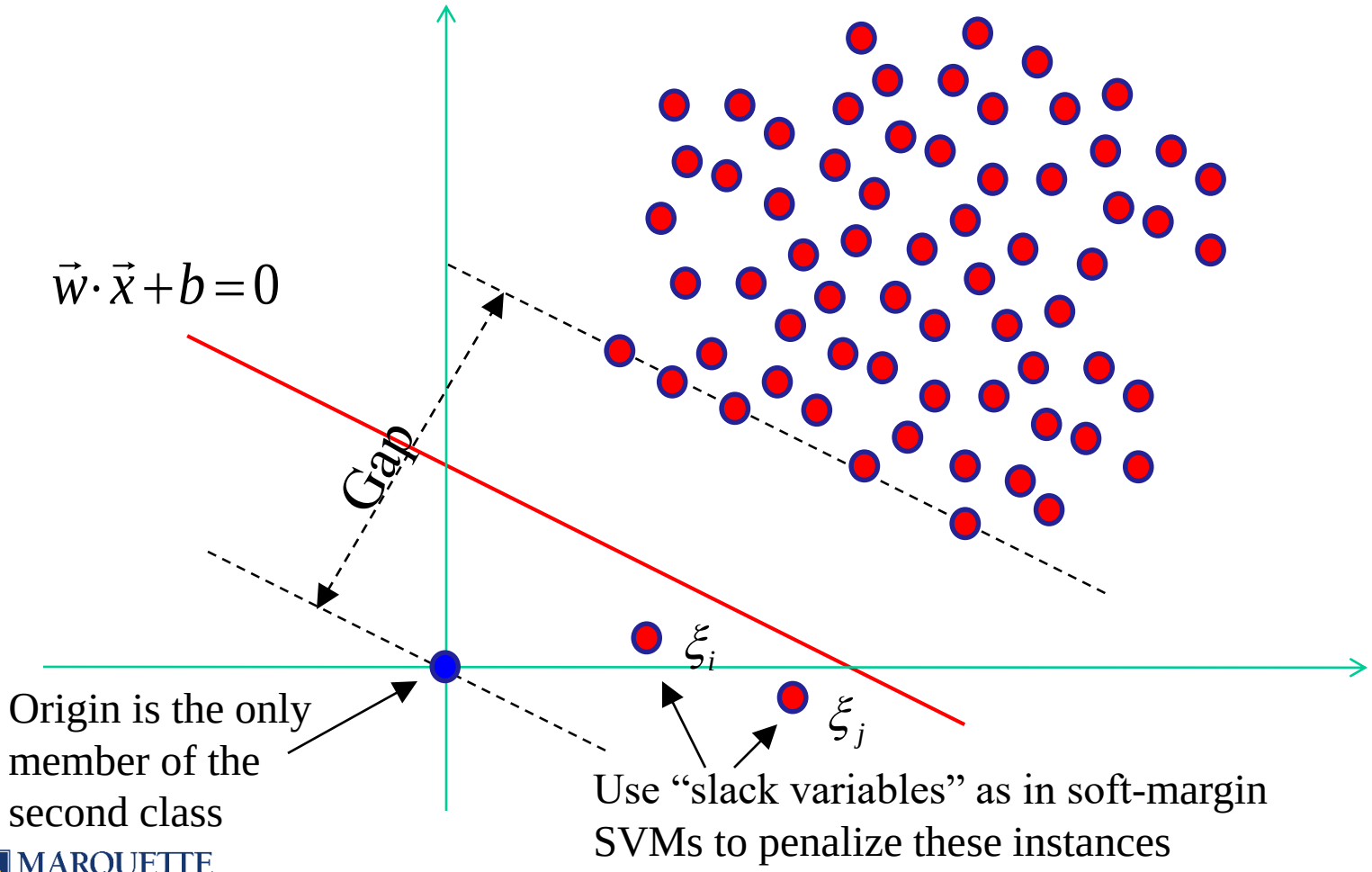
SAMPLE APPLICATIONS

Identify websites that discuss benefits of proven cancer treatments.



ONE-CLASS SVM

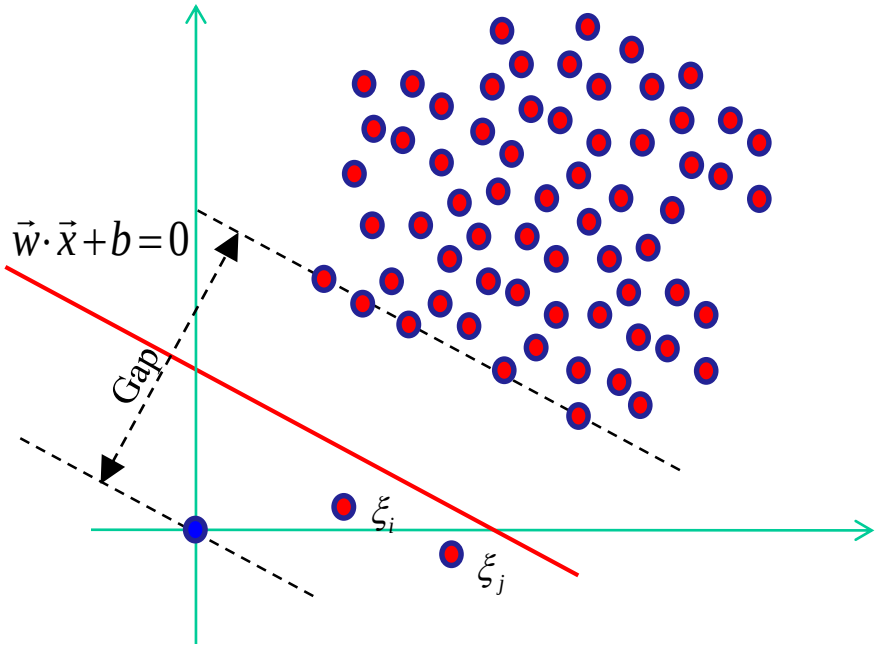
Main idea: Find the maximal gap hyperplane that separates data from the origin (i.e., the only member of the second class is the origin).



FORMULATION OF ONE-CLASS SVM:

LINEAR CASE

Given training data: $\vec{x}_1, \vec{x}_2, ..., \vec{x}_N \in R^n$



Find $f(\vec{x}) = sign(\vec{w}\cdot\vec{x}+b)$

by minimizing $\frac{1}{2}\|\vec{w}\|^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i + b$

subject to constraints:

$$\vec{w}\cdot\vec{x}+b \geq -\xi_i$$
$$\xi_i \geq 0$$
for $i = 1,\ldots,N$.

upper bound on the fraction of outliers (i.e., points outside decision surface) allowed in the data

i.e., the decision function should be positive in all training samples except for small deviations

FORMULATION OF ONE-CLASS SVM:

LINEAR AND NON-LINEAR CASES

Linear case

Find $f(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b)$

by **minimizing** $\frac{1}{2}\|\vec{w}\|^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i + b$

subject to constraints:

$$\vec{w} \cdot \vec{x} + b \geq -\xi_i$$

$$\xi_i \geq 0$$

for $i = 1,\ldots,N$.

Non-linear case (use "kernel trick")

Find $f(\vec{x}) = sign(\vec{w} \cdot \Phi(\vec{x}) + b)$

by **minimizing** $\frac{1}{2}\|\vec{w}\|^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i + b$

subject to constraints:

$$\vec{w} \cdot \Phi(\vec{x}) + b \geq -\xi_i$$

$$\xi_i \geq 0$$

for $i = 1,\ldots,N$.

More about one-class SVM

- One-class SVMs inherit most of properties of SVMs for binary classification (e.g., "kernel trick", sample efficiency, ease of finding of a solution by efficient optimization method, etc.);
- The choice of other parameter ν significantly affects the resulting decision surface.
- The choice of origin is arbitrary and also significantly affects the decision surface returned by the algorithm.

QUESTIONS?

- ANY QUESTION?