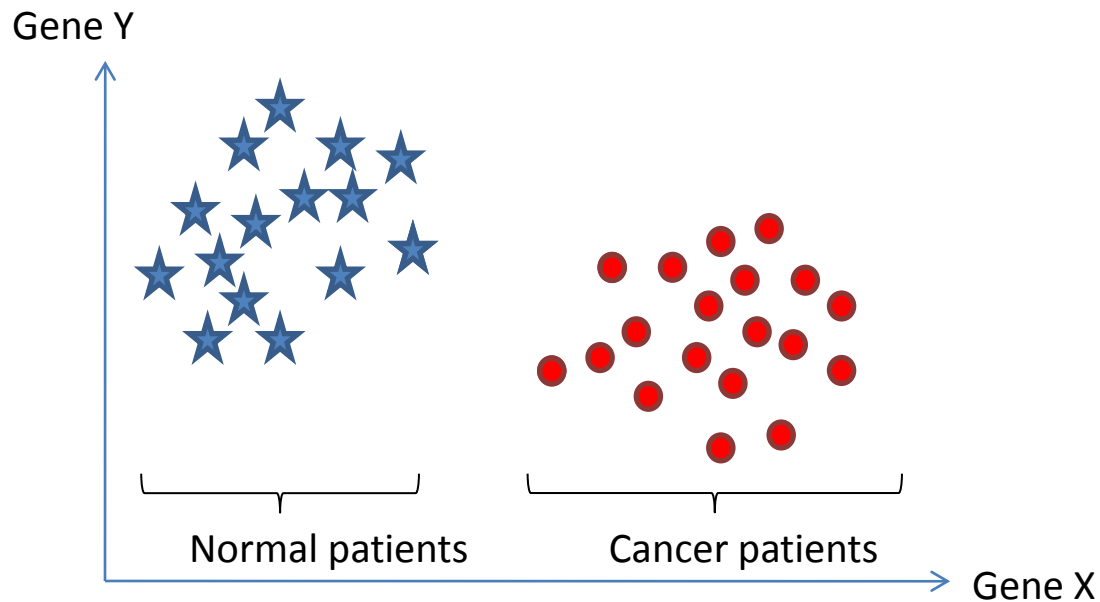
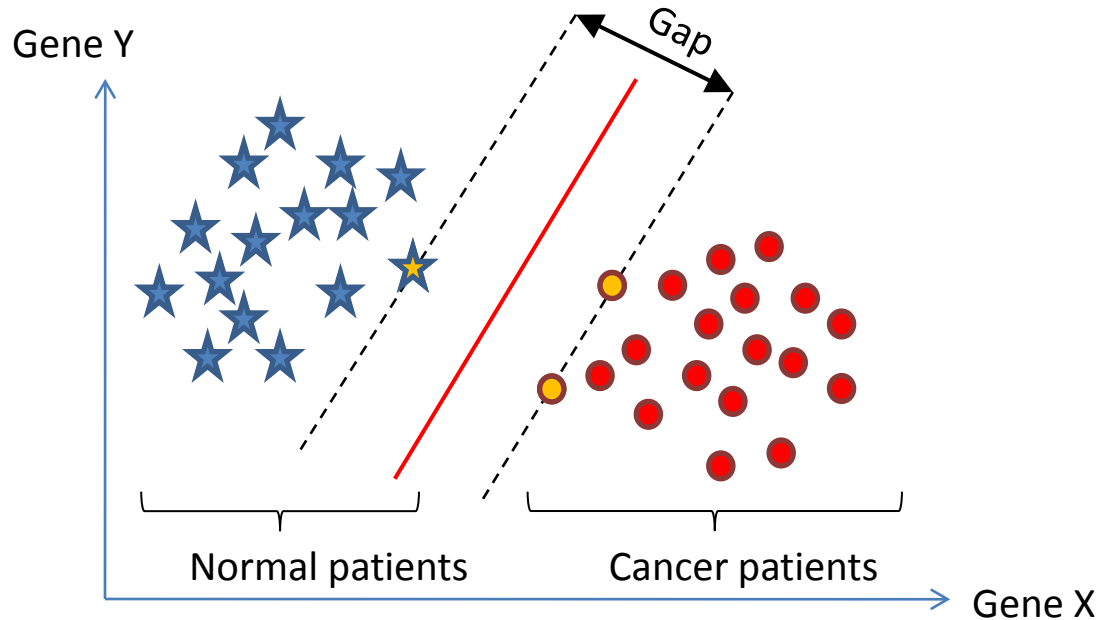


Main ideas of SVMs



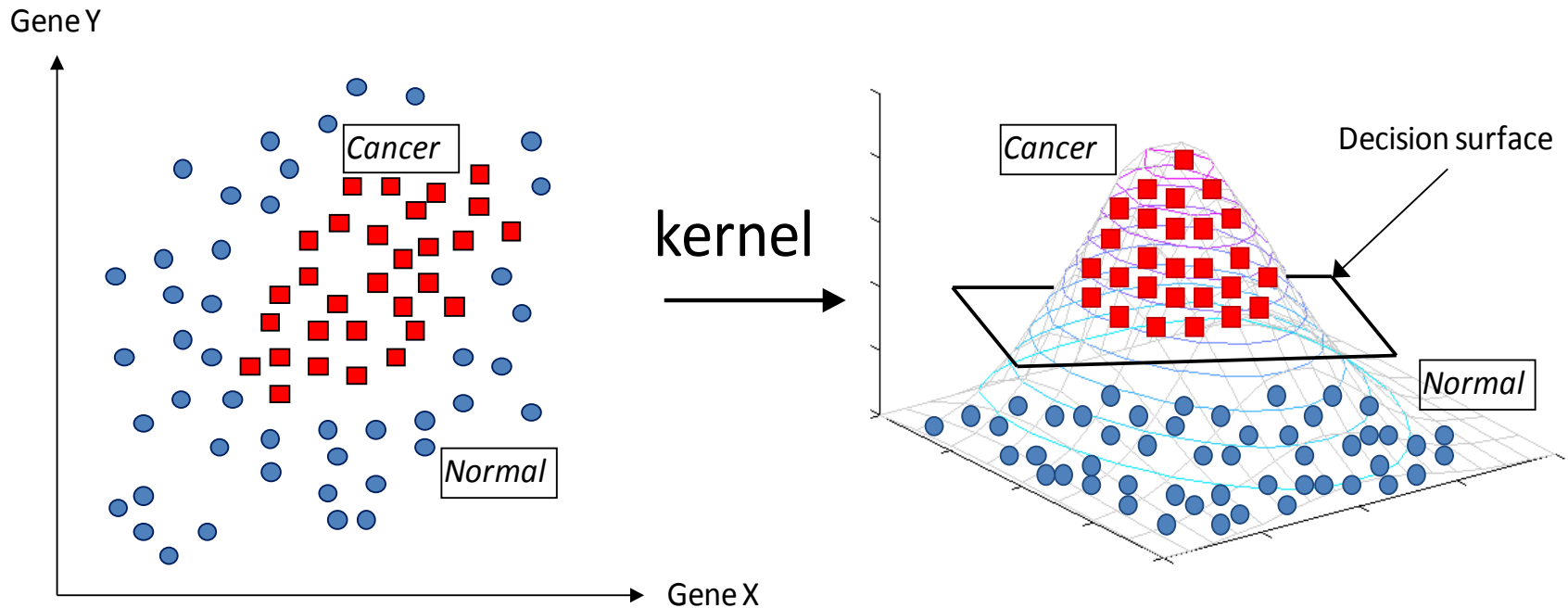
- Consider example dataset described by 2 genes, gene X and gene Y
- Represent patients geometrically (by “vectors”)

Main ideas of SVMs



- 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”);

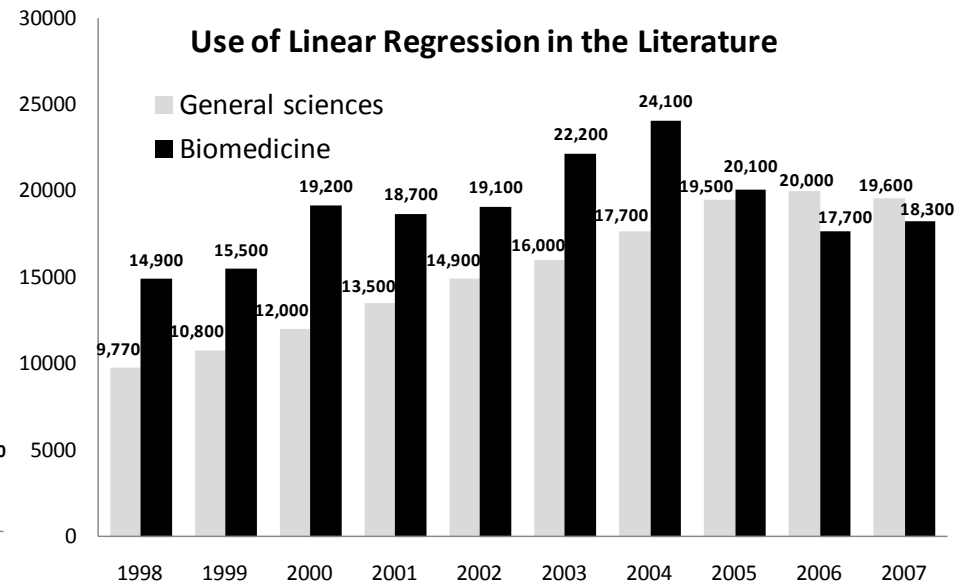
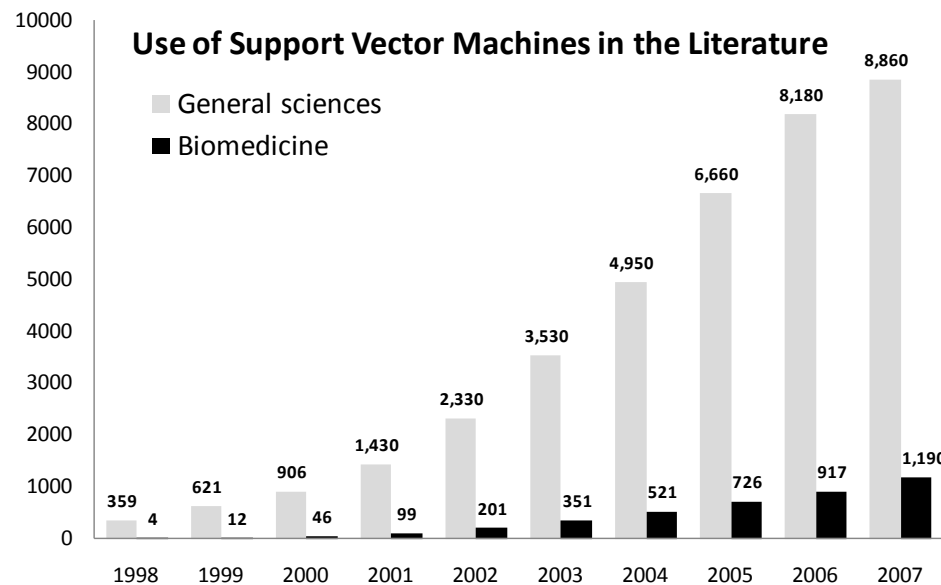
Main ideas of SVMs



- 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*")



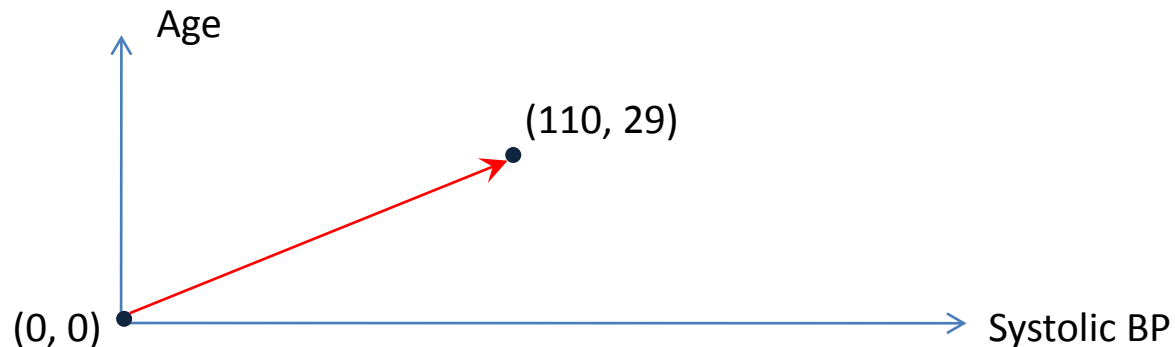
Necessary mathematical concepts

How to represent samples geometrically?

Vectors in n -dimensional space (\mathbb{R}^n)

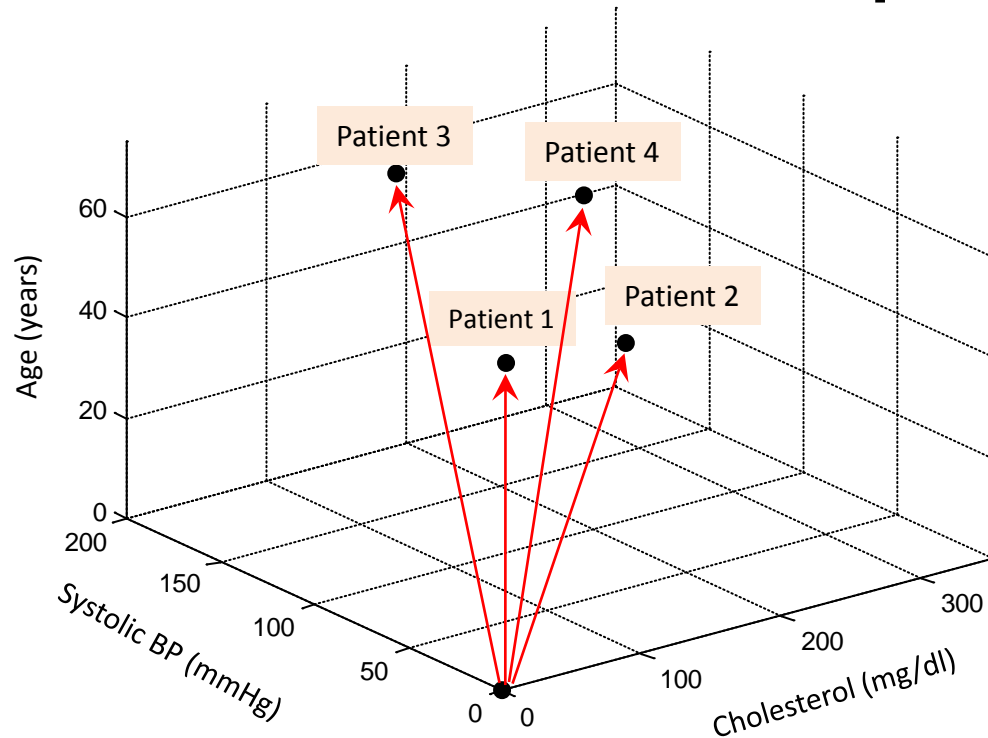
- Assume that a sample/patient is described by n characteristics (“features” or “variables”)
- **Representation:** Every sample/patient is a vector in \mathbb{R}^n with tail at point with 0 coordinates and arrow-head at point with the feature values.
- **Example:** Consider a patient described by 2 features: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

This patient can be represented as a vector in \mathbb{R}^2 :



How to represent samples geometrically?

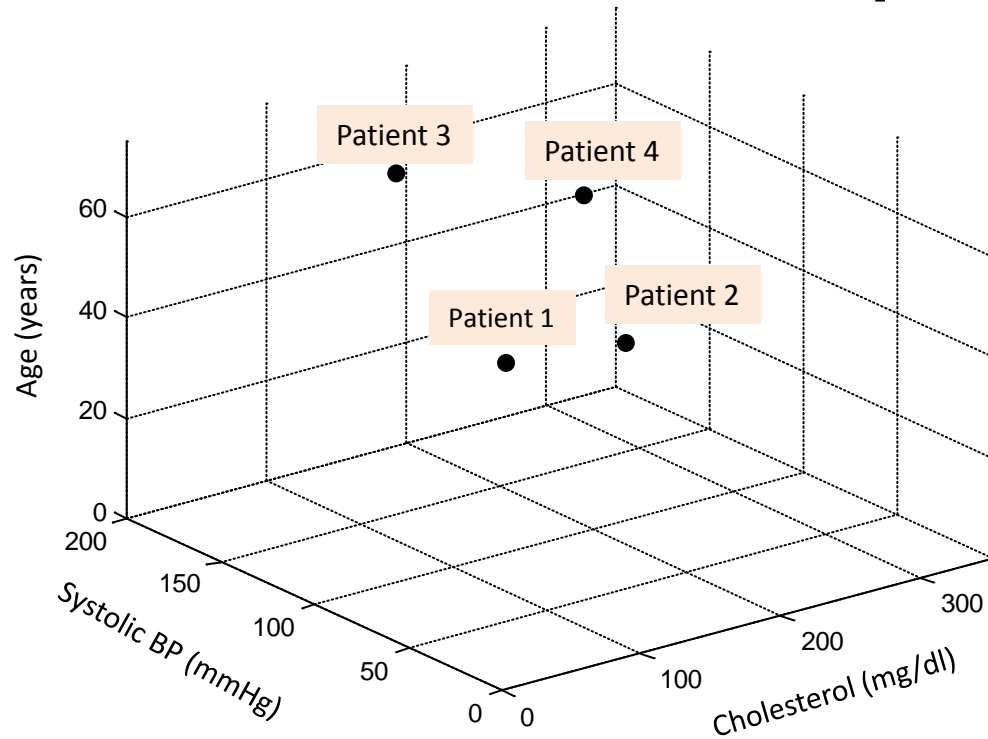
Vectors in n-dimensional space (\mathbb{R}^n)



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)

How to represent samples geometrically?

Vectors in n-dimensional space (\mathbb{R}^n)

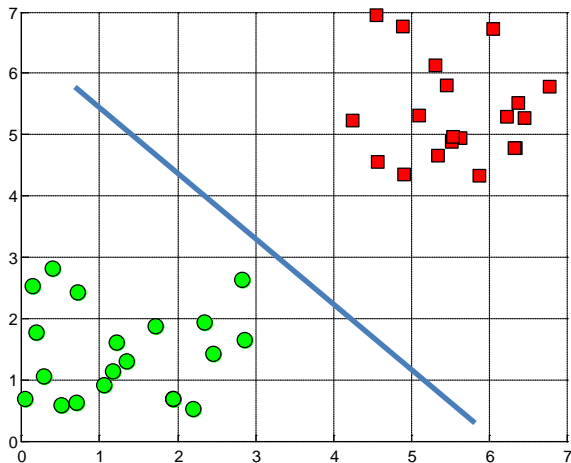


Since we assume that the tail of each vector is at point with 0 coordinates, we will also depict vectors as points (where the arrow-head is pointing).

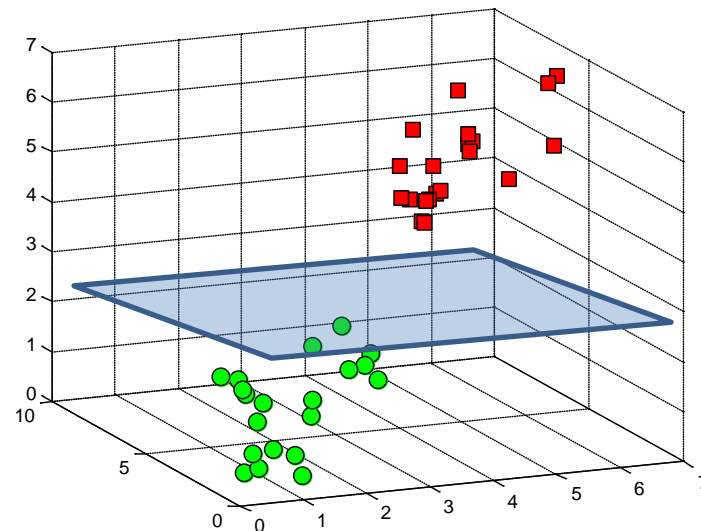
Purpose of vector representation

- 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...

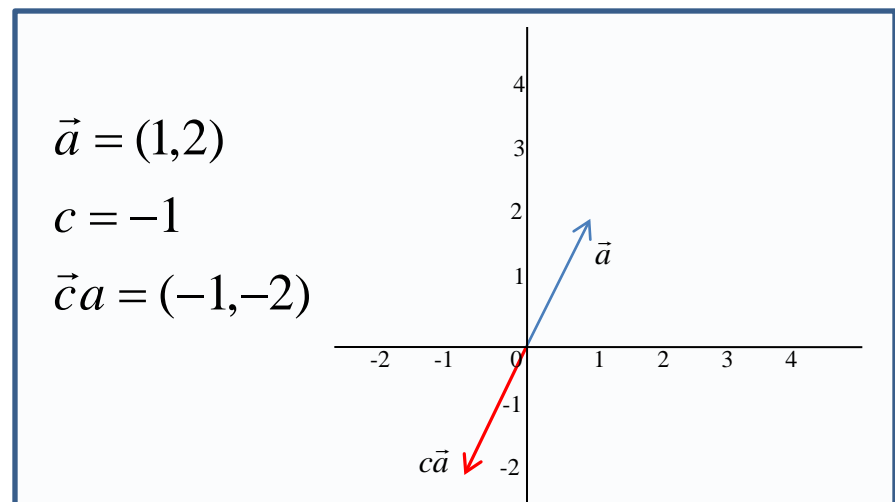
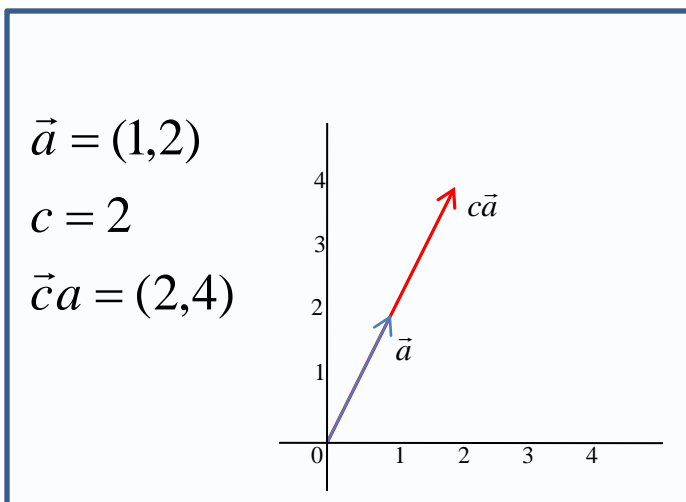
Basic operation on vectors in \mathbb{R}^n

1. Multiplication by a scalar

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

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

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.

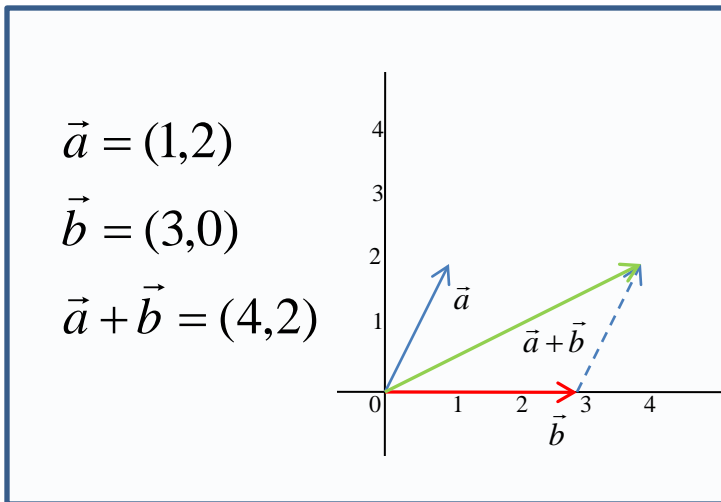


Basic operation on vectors in \mathbb{R}^n

2. Addition

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

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



Recall addition of forces in classical mechanics.

Basic operation on vectors in \mathbb{R}^n

3. Subtraction

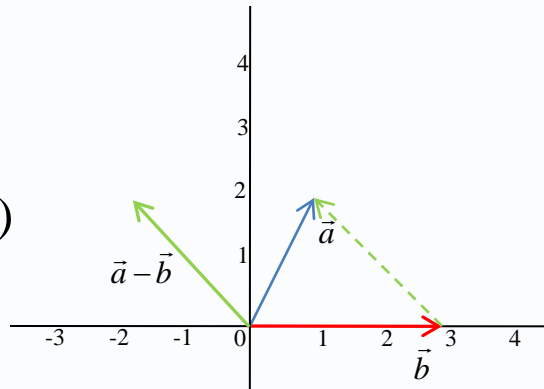
Consider vectors $\vec{a} = (a_1, a_2, \dots, a_n)$ and $\vec{b} = (b_1, b_2, \dots, b_n)$

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

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

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

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



What vector do we need to add to \vec{b} to get \vec{a} ? I.e., similar to subtraction of real numbers.

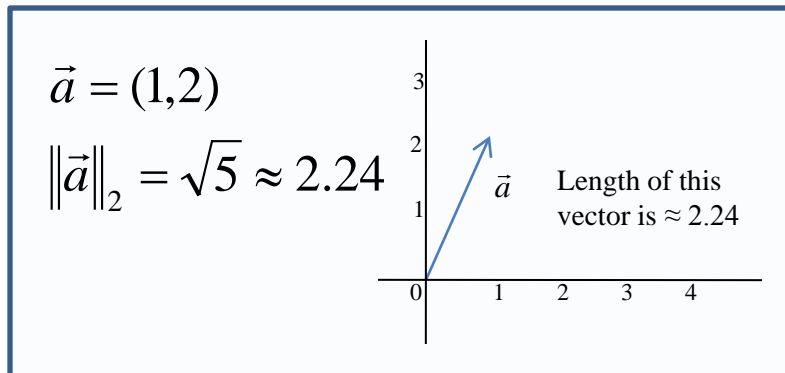
Basic operation on vectors in \mathbb{R}^n

4. Euclidian length or L2-norm

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

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

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



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

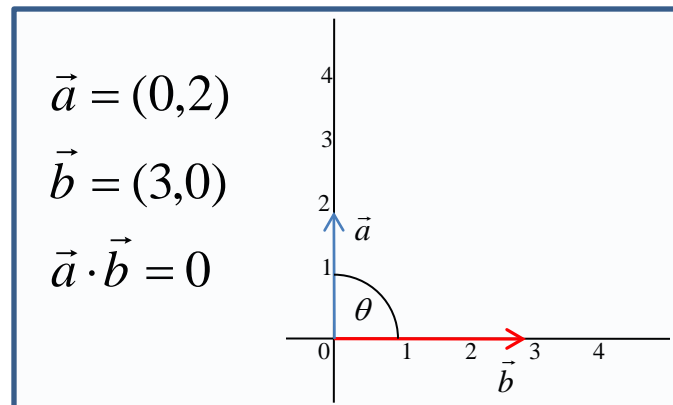
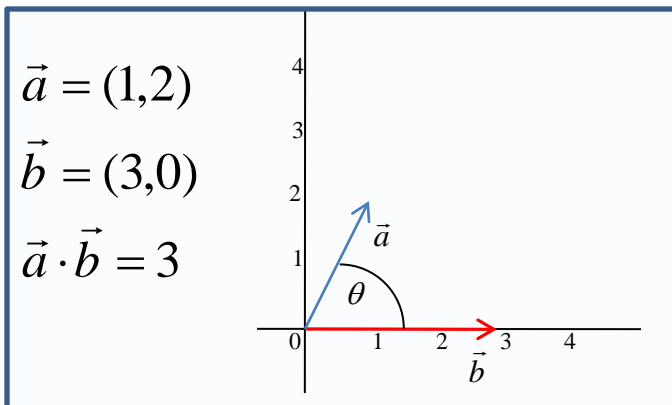
Basic operation on vectors in \mathbb{R}^n

5. Dot product

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

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

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



Basic operation on vectors in \mathbb{R}^n

5. Dot product (continued)

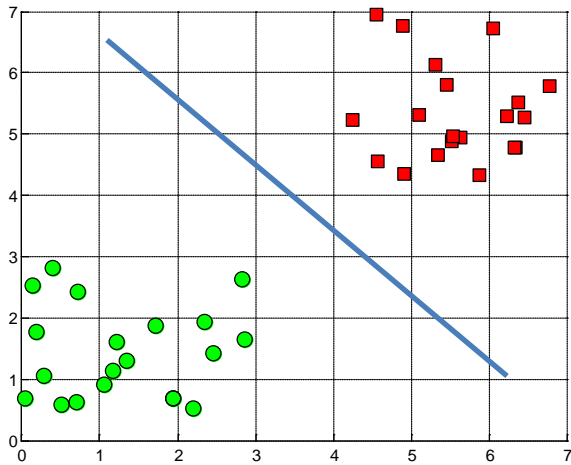
$$\vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 + \dots + a_n b_n = \sum_{i=1}^n a_i b_i$$

- Property: $\vec{a} \cdot \vec{a} = a_1 a_1 + a_2 a_2 + \dots + a_n a_n = \|\vec{a}\|_2^2$
- 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 .

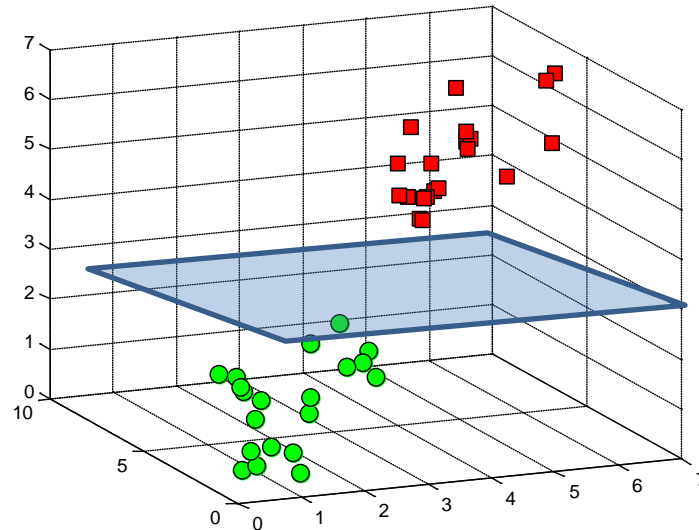
Hyperplanes as decision surfaces

- 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



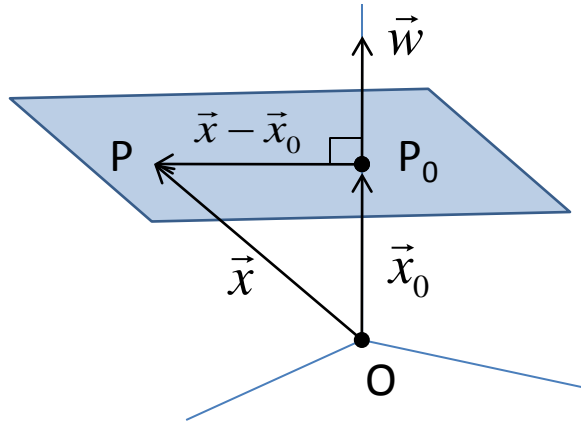
A hyperplane in \mathbb{R}^3 is a plane



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} :

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

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

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

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

Equation of a hyperplane

Example

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

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

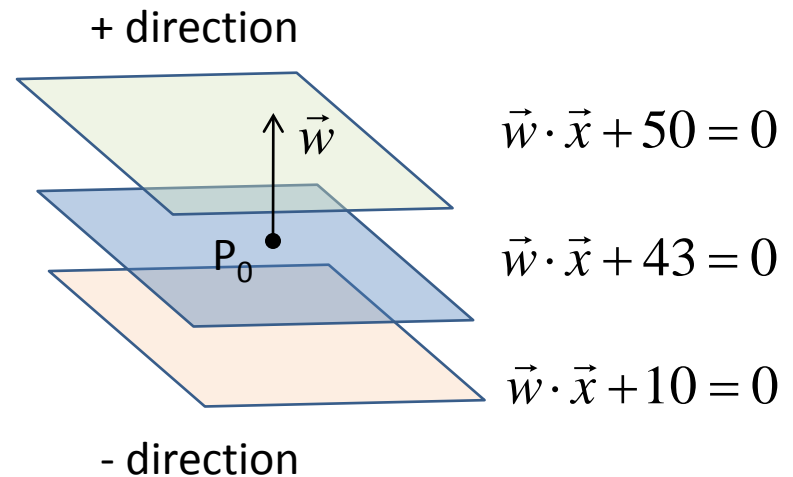
$$b = -\vec{w} \cdot \vec{x}_0 = -(0 - 1 - 42) = 43$$

$$\Rightarrow \vec{w} \cdot \vec{x} + 43 = 0$$

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

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

$$\Rightarrow 4x_{(1)} - x_{(2)} + 6x_{(3)} + 43 = 0$$



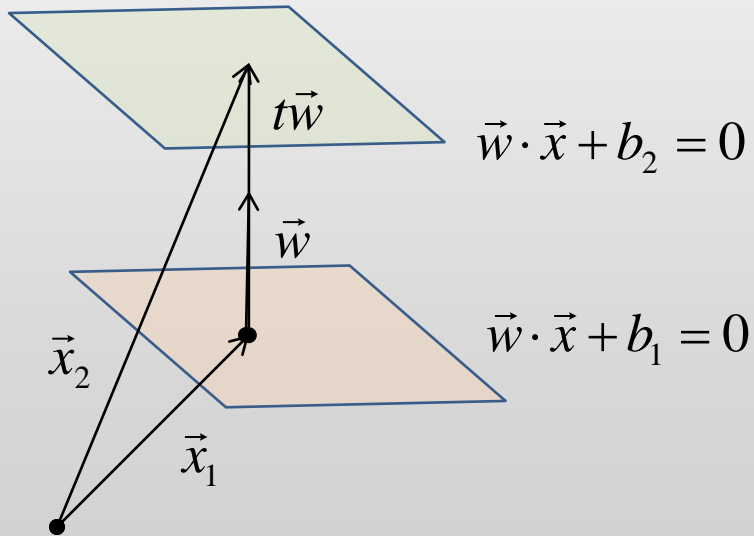
What happens if the b coefficient changes?

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

(Derivation of the distance between two parallel hyperplanes)



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

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

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

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

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

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

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

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

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

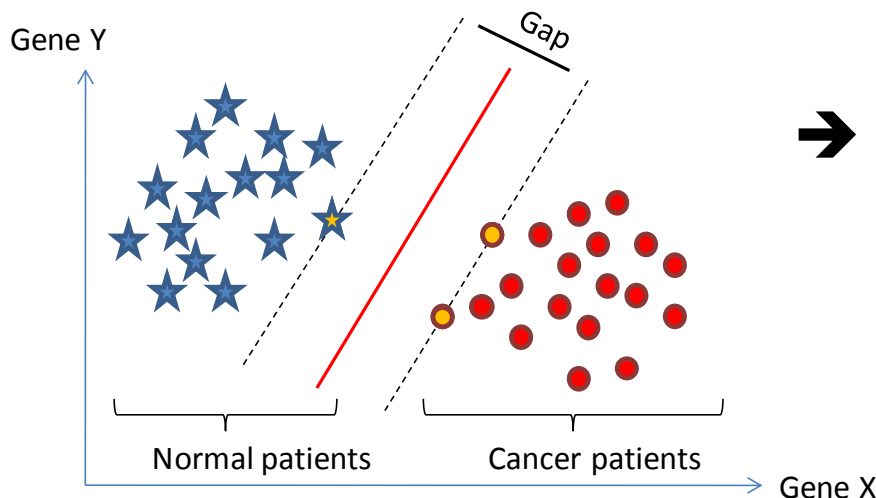
Recap

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”?

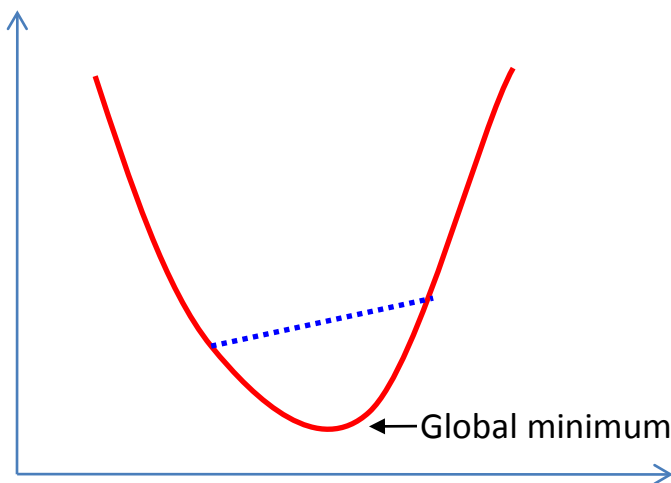


➔ Need to introduce basics of relevant optimization theory

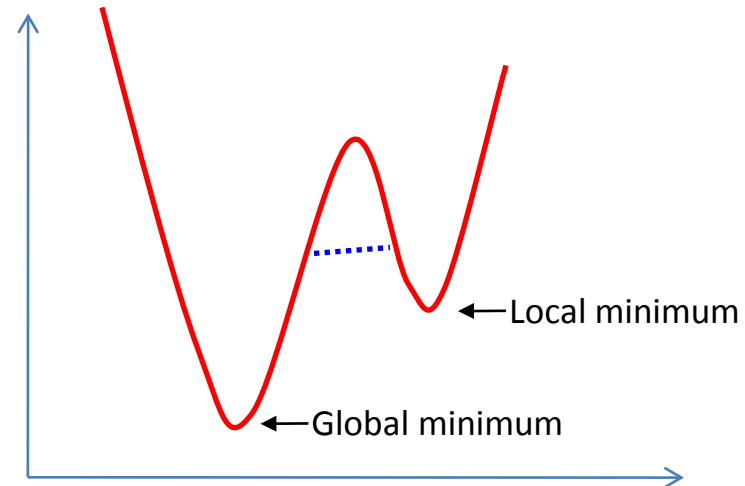
Basics of optimization:

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

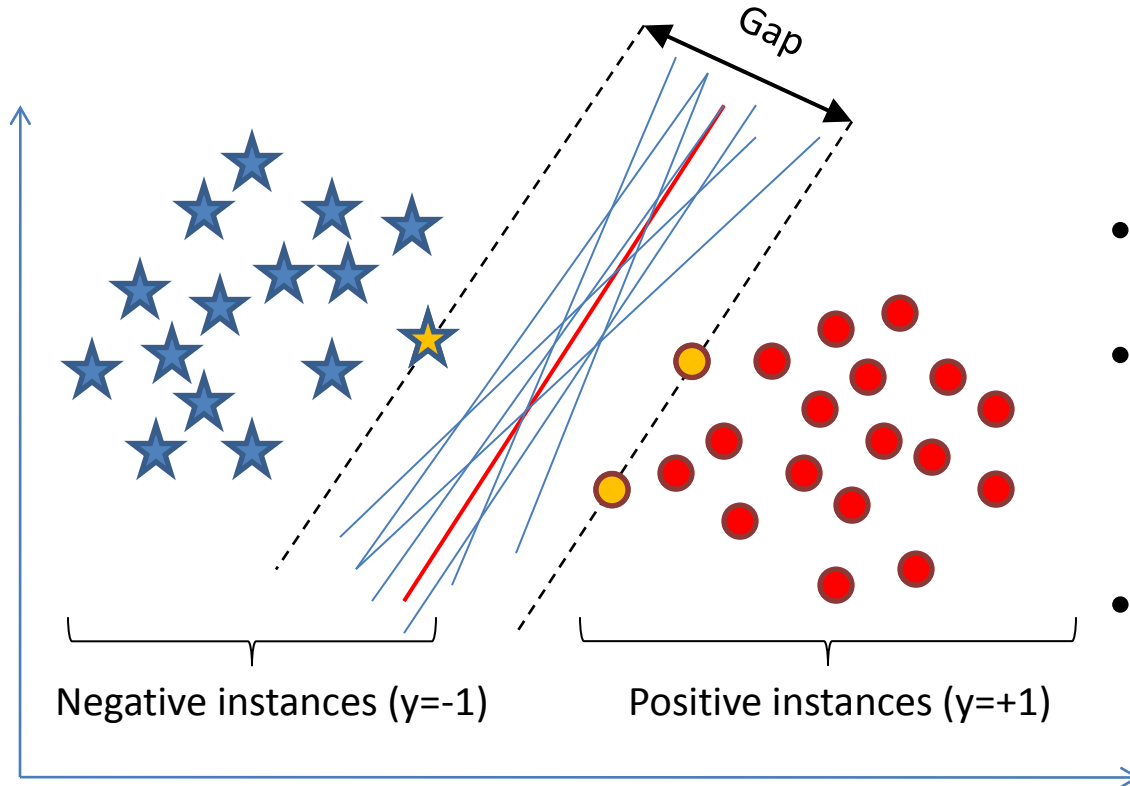


Non-convex function

Support vector machines for binary classification: classical formulation

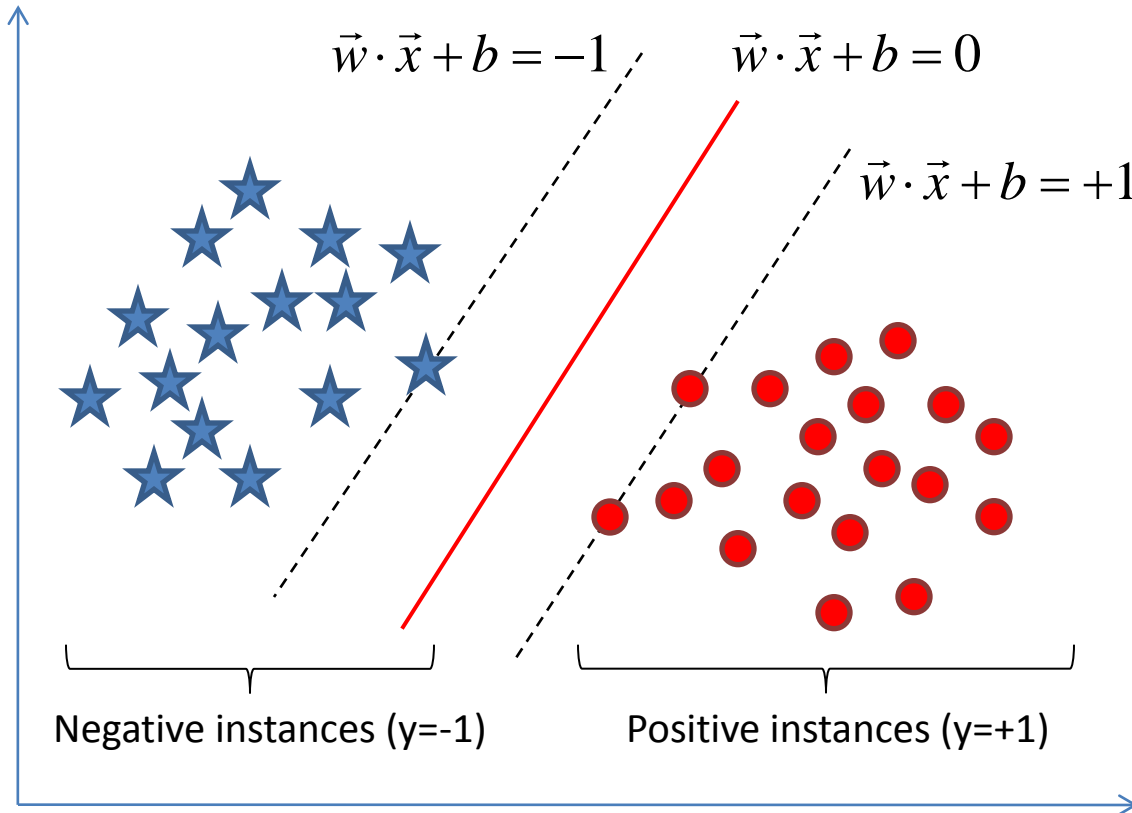
Case I: Linearly separable data; “Hard-margin” linear SVM

Given training data: $\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N \in R^n$
 $y_1, y_2, \dots, y_N \in \{-1, +1\}$



- Want to find a classifier (hyperplane) to separate negative instances from the positive ones.
- An infinite number of such hyperplanes exist.
- SVMs find 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:

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

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:

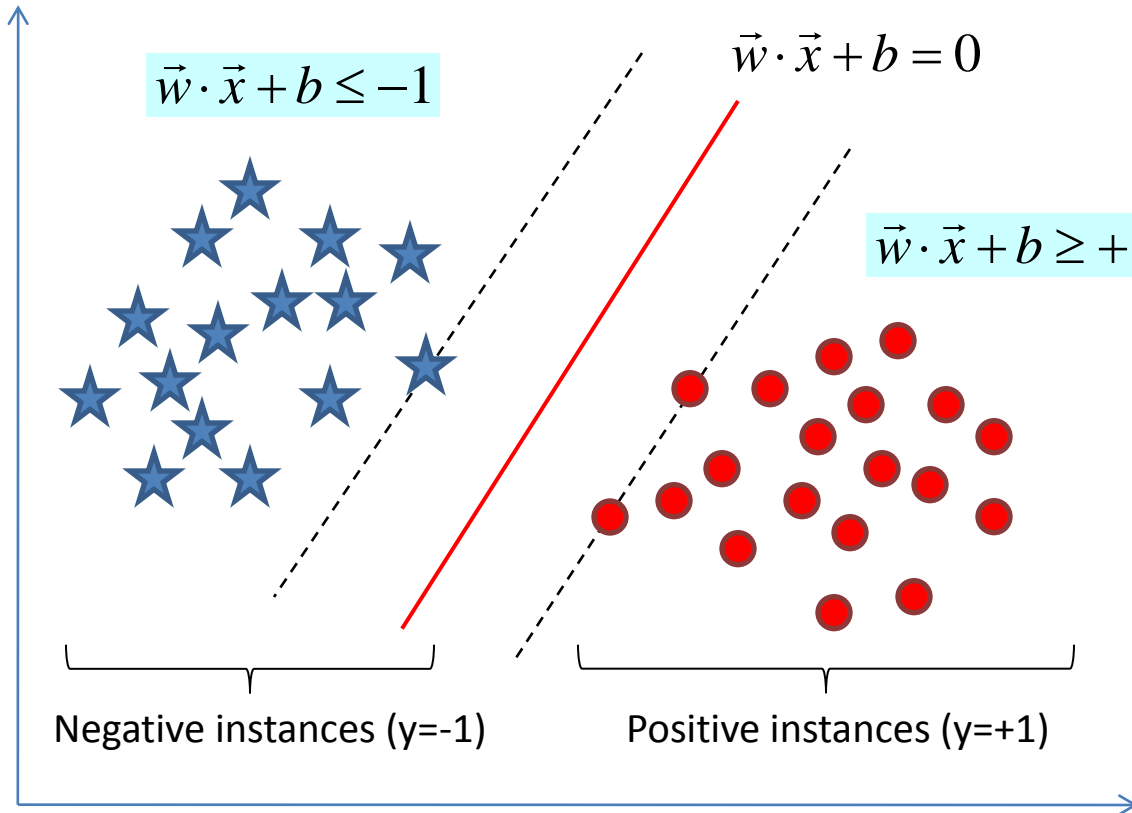
$$D = 2 / \|\vec{w}\|$$

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)

Statement of linear SVM classifier



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

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

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

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 $i = 1, \dots, N$

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

SVM optimization problem: Primal formulation

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

Objective function Constraints

- This is called “primal formulation of linear SVMs”.
- It is a convex quadratic programming (QP) optimization problem with n variables ($w_i, i = 1, \dots, n$), where n 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 ($\alpha_i, i = 1, \dots, N$), where N is the number of samples.

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

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

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

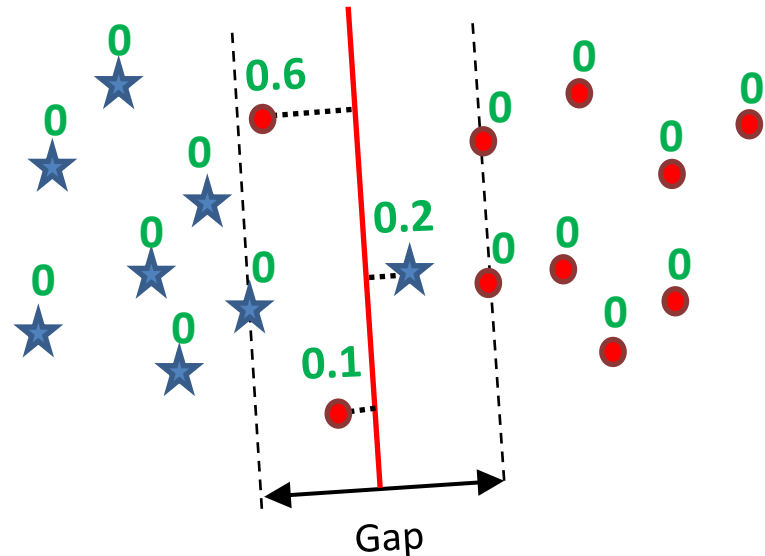
- 2) Number of free parameters is bounded by the number of support vectors and not by the number of variables (beneficial for high-dimensional problems).

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

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 $i = 1, \dots, N$

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

Two formulations of soft-margin linear SVM

Primal formulation:

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

Objective function Constraints

Dual formulation:

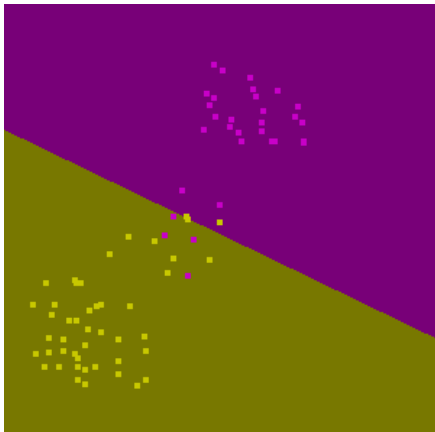
$$\text{Minimize } \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j \quad \text{subject to } 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^N \alpha_i y_i = 0$$

Objective function Constraints

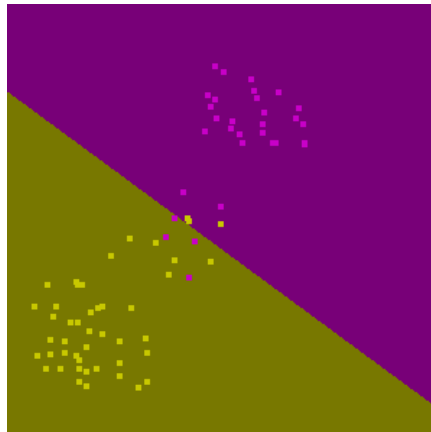
for $i = 1, \dots, N$.

Parameter C in soft-margin SVM

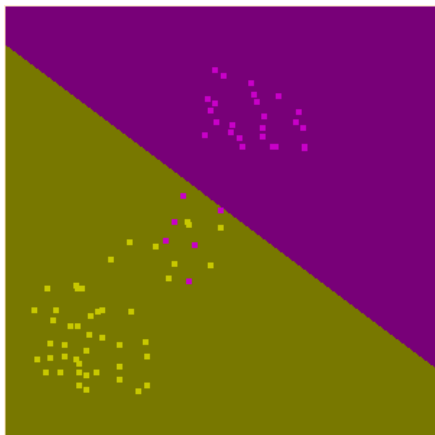
Minimize $\frac{1}{2} \|\vec{w}\|^2 + \boxed{C} \sum_{i=1}^N \xi_i$ subject to $y_i (\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i$ for $i = 1, \dots, N$



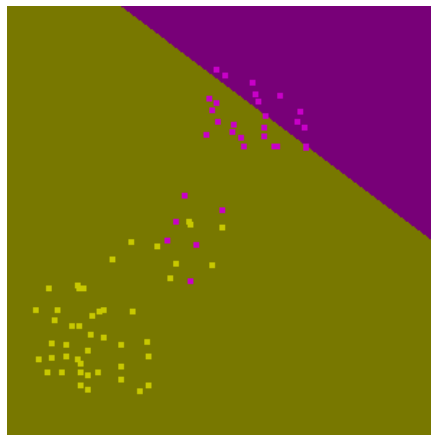
$C=100$



$C=1$



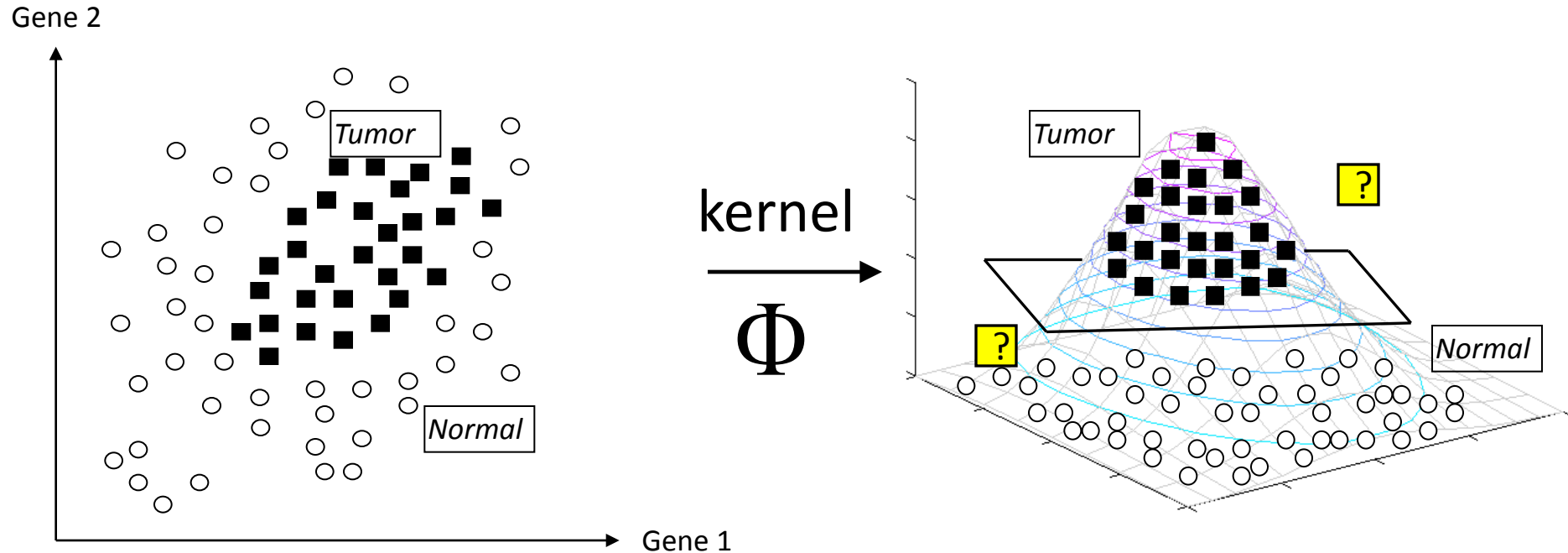
$C=0.15$



$C=0.1$

- 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.

Case 3: Not linearly separable data; Kernel trick



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)

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

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

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

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

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

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

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

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

$$K(\vec{x}_i, \vec{x}_j) = \exp(-\gamma \|\vec{x}_i - \vec{x}_j\|^2)$$

Gaussian kernel

$$K(\vec{x}_i, \vec{x}_j) = \exp(-\gamma \|\vec{x}_i - \vec{x}_j\|)$$

Exponential kernel

$$K(\vec{x}_i, \vec{x}_j) = (p + \vec{x}_i \cdot \vec{x}_j)^q$$

Polynomial kernel

$$K(\vec{x}_i, \vec{x}_j) = (p + \vec{x}_i \cdot \vec{x}_j)^q \exp(-\gamma \|\vec{x}_i - \vec{x}_j\|^2)$$

Hybrid kernel

$$K(\vec{x}_i, \vec{x}_j) = \tanh(k\vec{x}_i \cdot \vec{x}_j - \delta)$$

Sigmoidal