Introduction to Support Vector Machines

Support Vector Machines

Support vector machines (SVMs) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis.

Support Vector Machines

Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier.

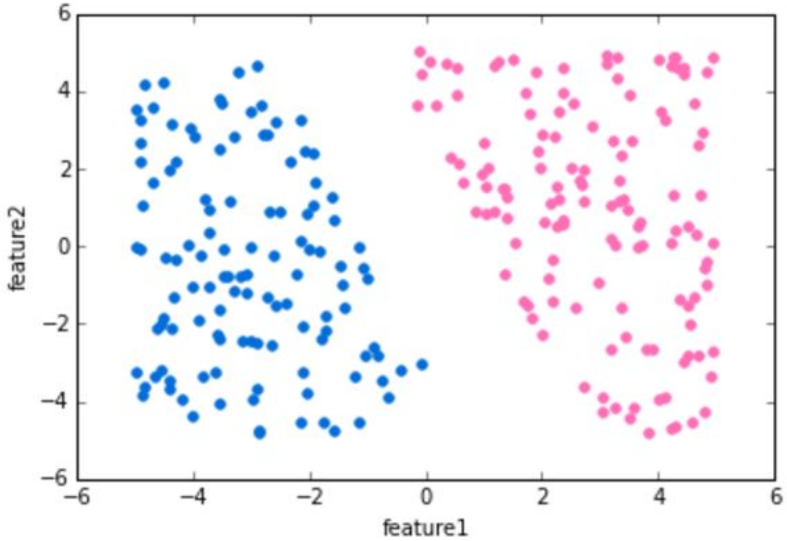
Support Vector Machines

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.

New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

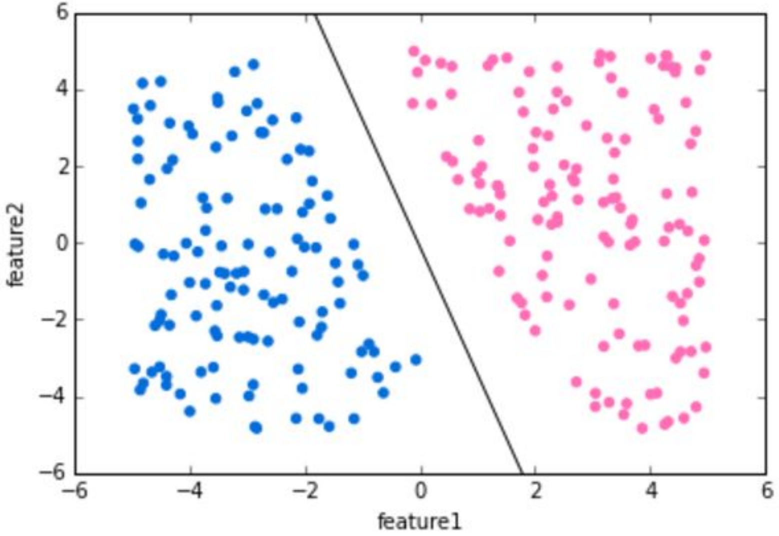
Support Vector Machines

Let's show the basic intuition behind SVMs. Imagine the labeled training data below:



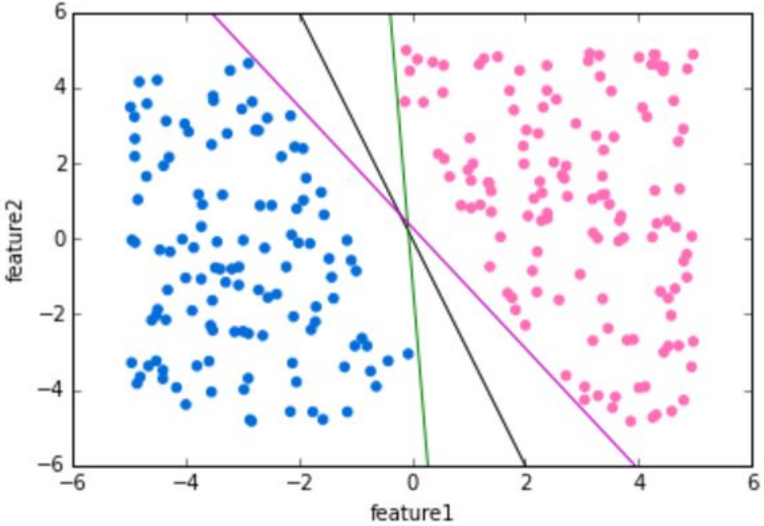
Support Vector Machines

We can draw a separating "hyperplane" between the classes.



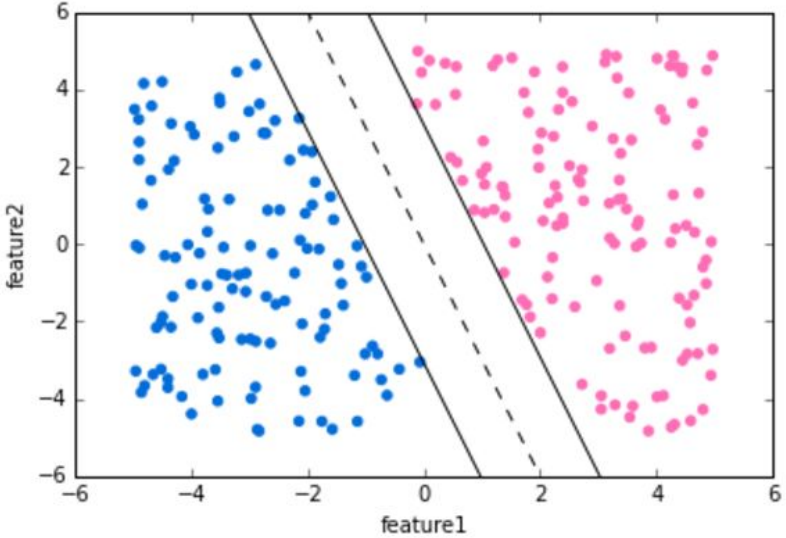
Support Vector Machines

But we have many options of hyperplanes that separate perfectly...



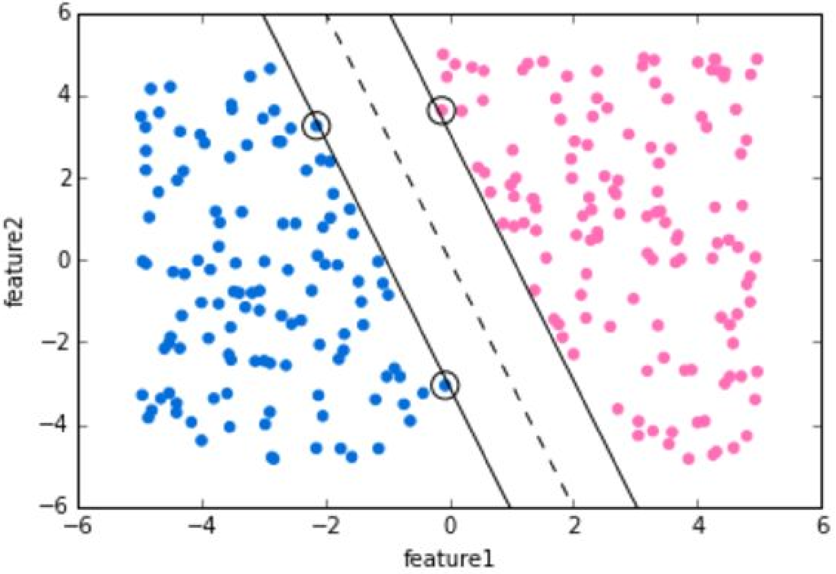
Support Vector Machines

We would like to choose a hyperplane that maximizes the margin between classes



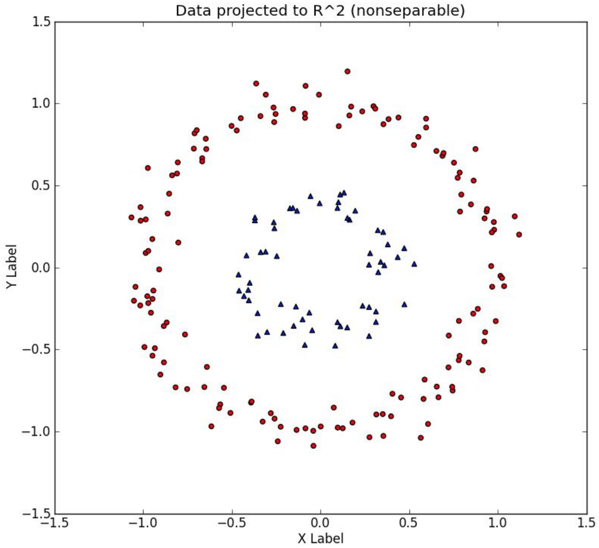
Support Vector Machines

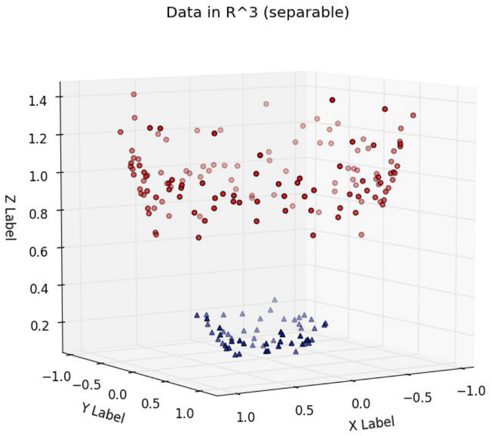
The vector points that the margin lines touch are known as Support Vectors.



Support Vector Machines

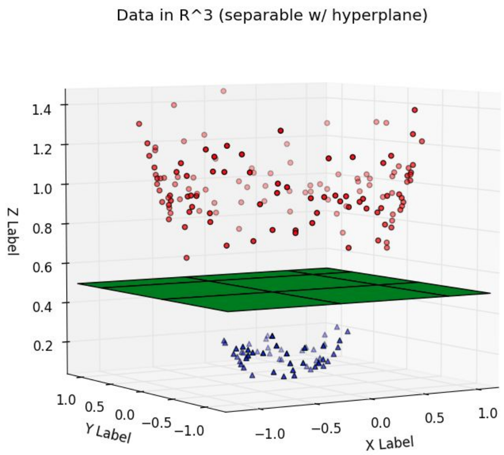
We can expand this idea to non-linearly separable data through the “kernel trick”.

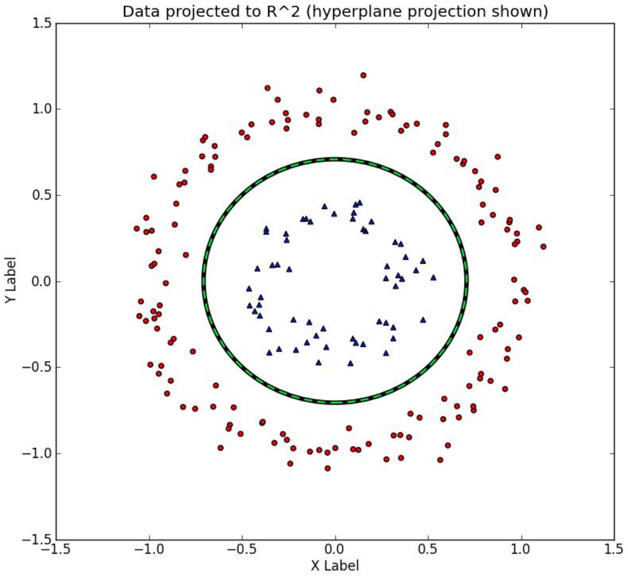




Support Vector Machines

Check out YouTube for nice 3D Visualization videos explaining this idea. Refer to reading for math behind this.





MACHINE LEARNING

SUPPORT VECTOR MACHINES

Support vector machine

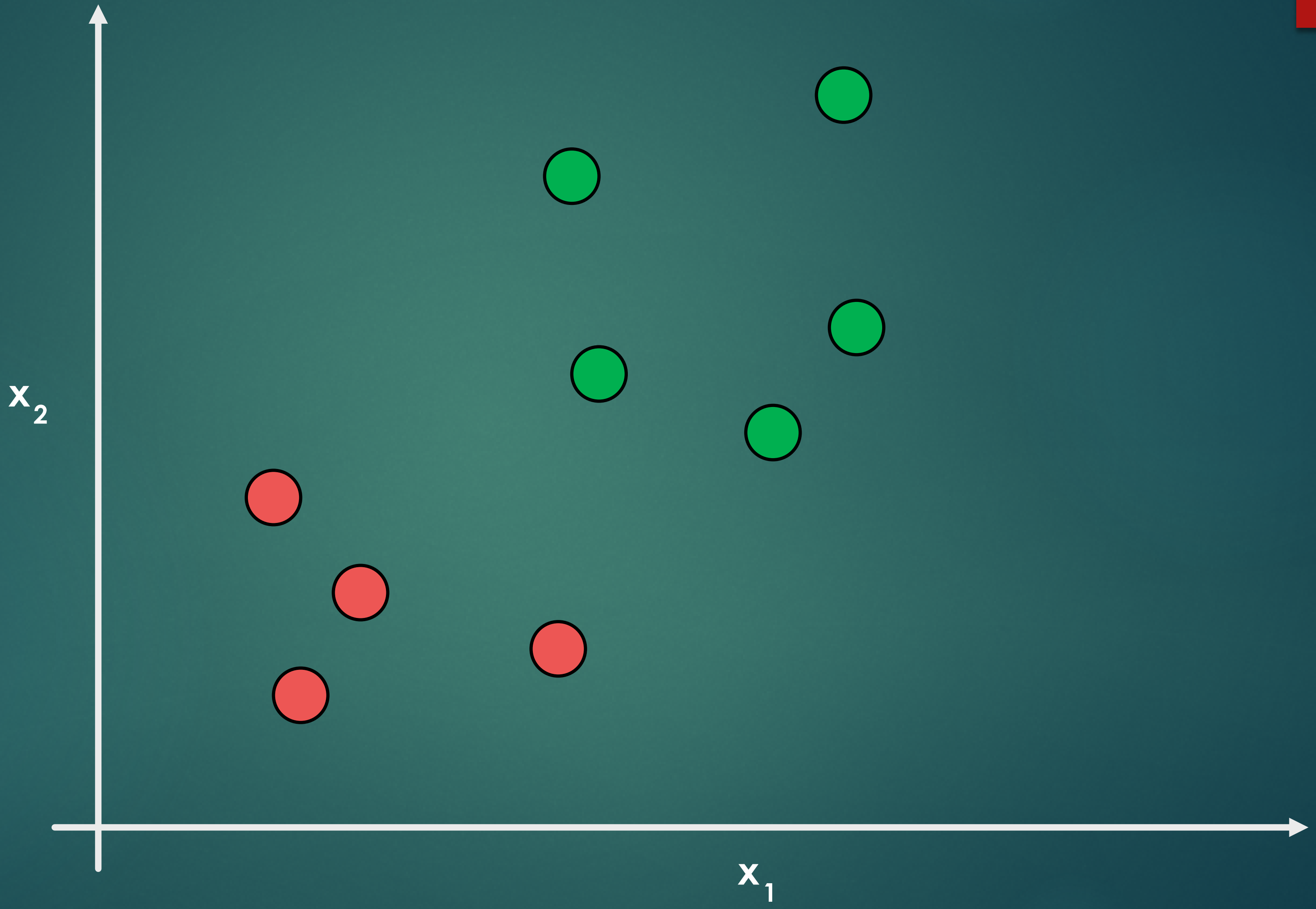
- Very popular and widely used supervised learning classification algorithm
- The great benefit: it can operates even in infinite dimensions !!!
- It defines a margin / boundary → between the data points in multidimensional space
- Goal: find a flat boundary („hyperplane”) that leads to a homogeneous partition of the data
- A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class since in general the larger the margin the lower the generalization error of the classifier
- So we have to maximize the margin

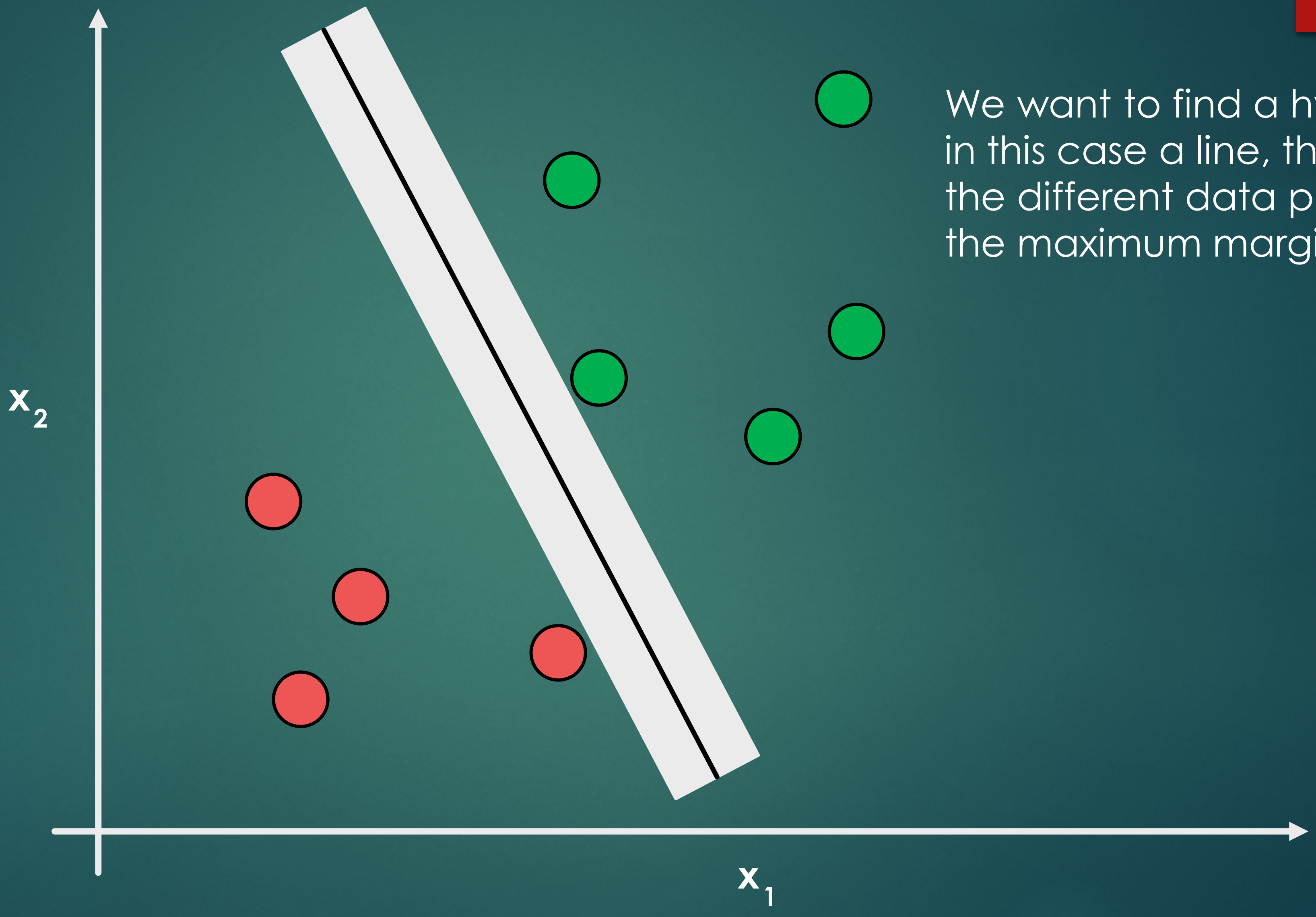
Support vector machine

- Can be applied to almost everything
- Classifications or numerical predictions
- Widely used in pattern recognition
  - Identify cancer or genetic diseases
  - Text classification: classify texts based on the language
  - Detecting rare events: earthquakes or engine failures

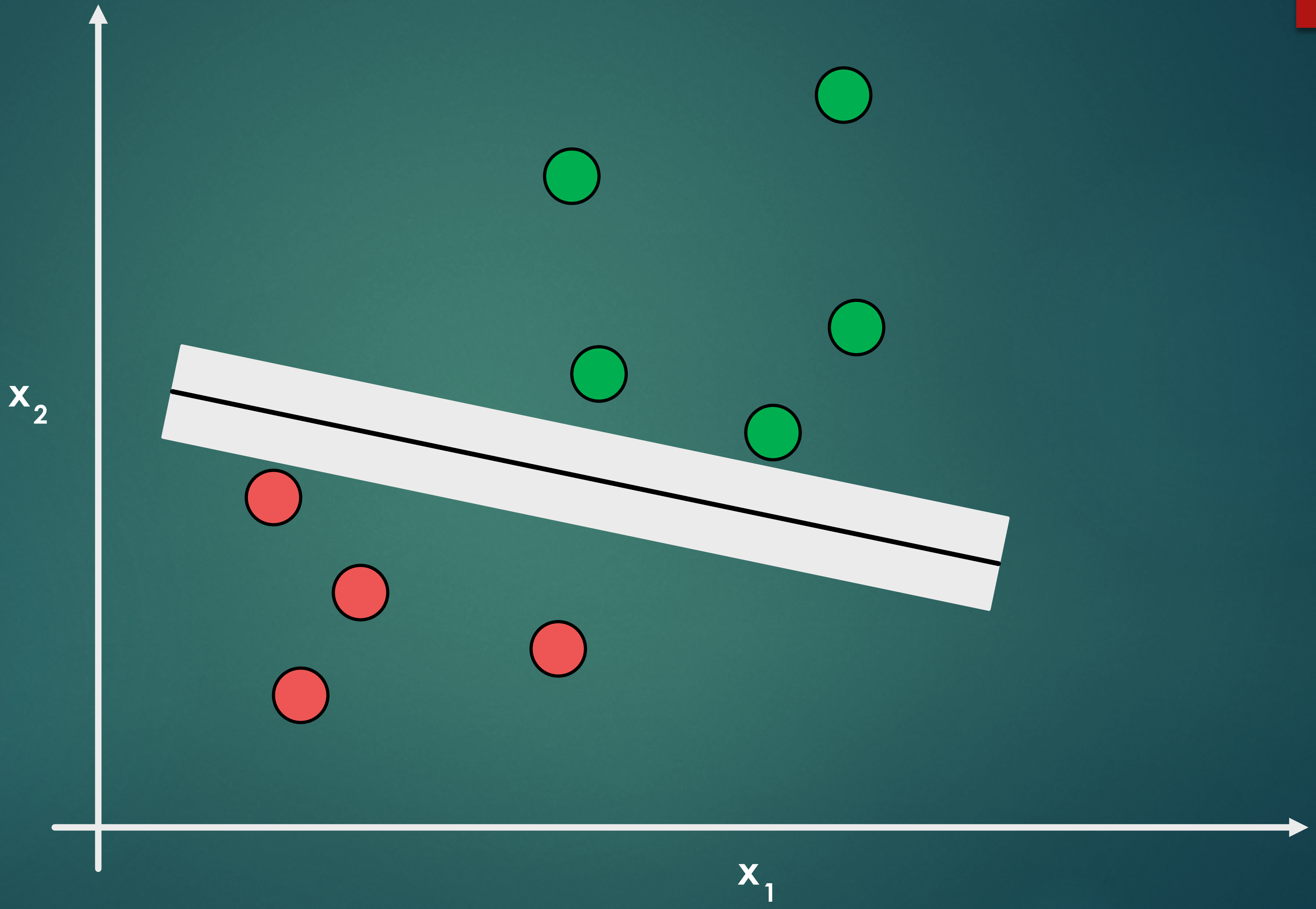
Linearly separable problem

We have two features $(\mathbf{x}_1, \mathbf{x}_2)$ and some data points

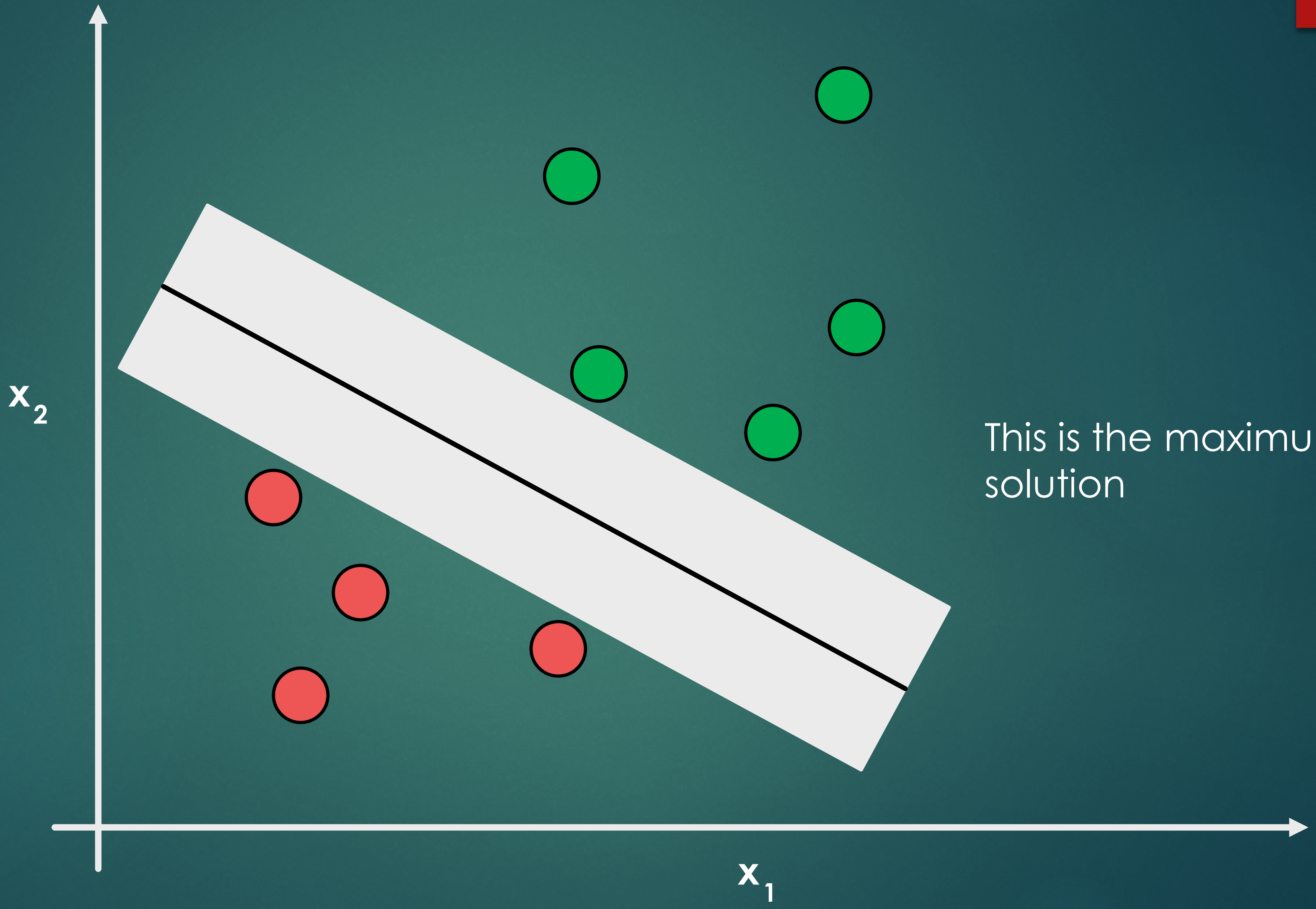




We want to find a hyperplane, in this case a line, that separates the different data points with the maximum margin

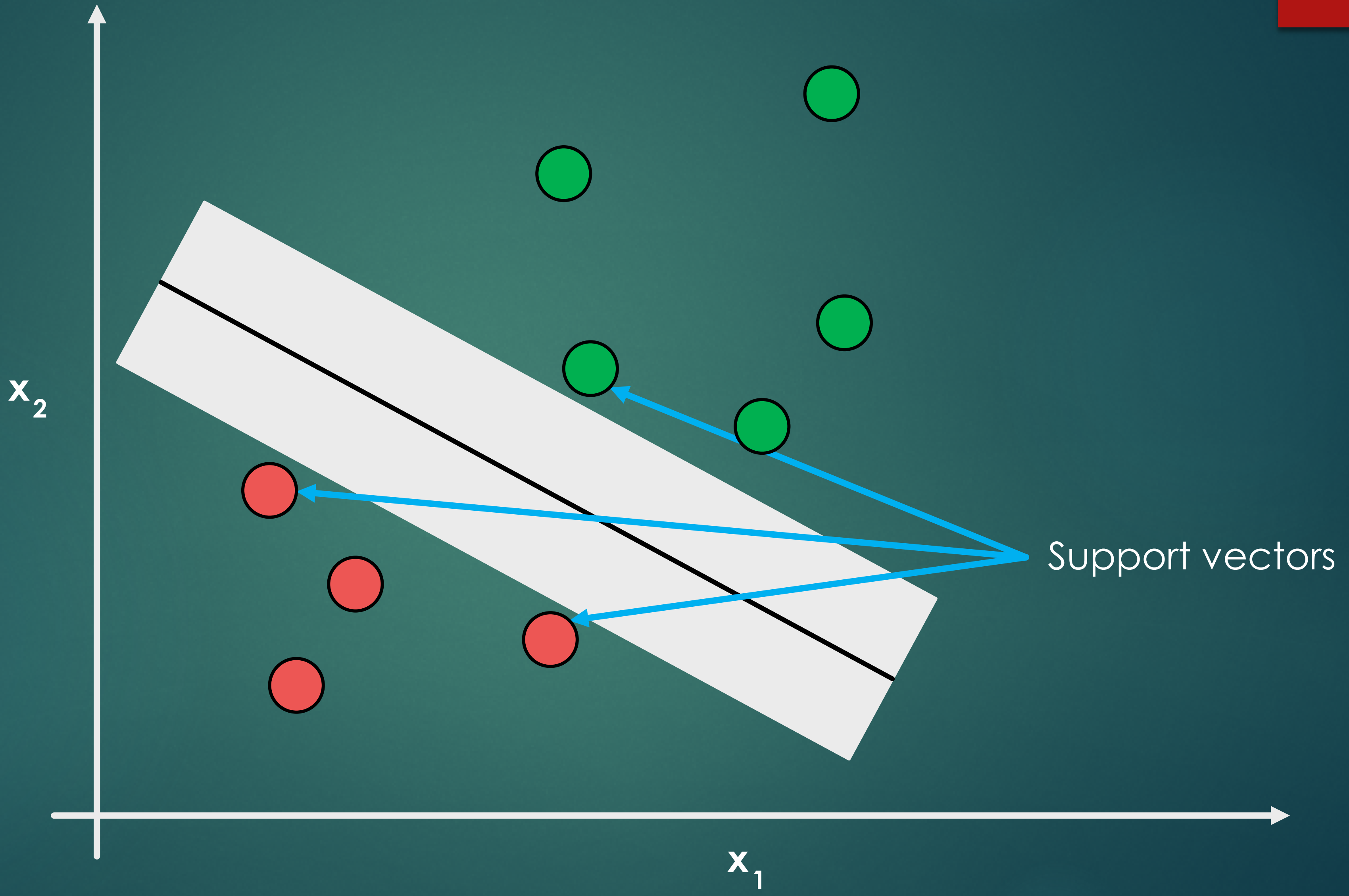


x_2
x_1

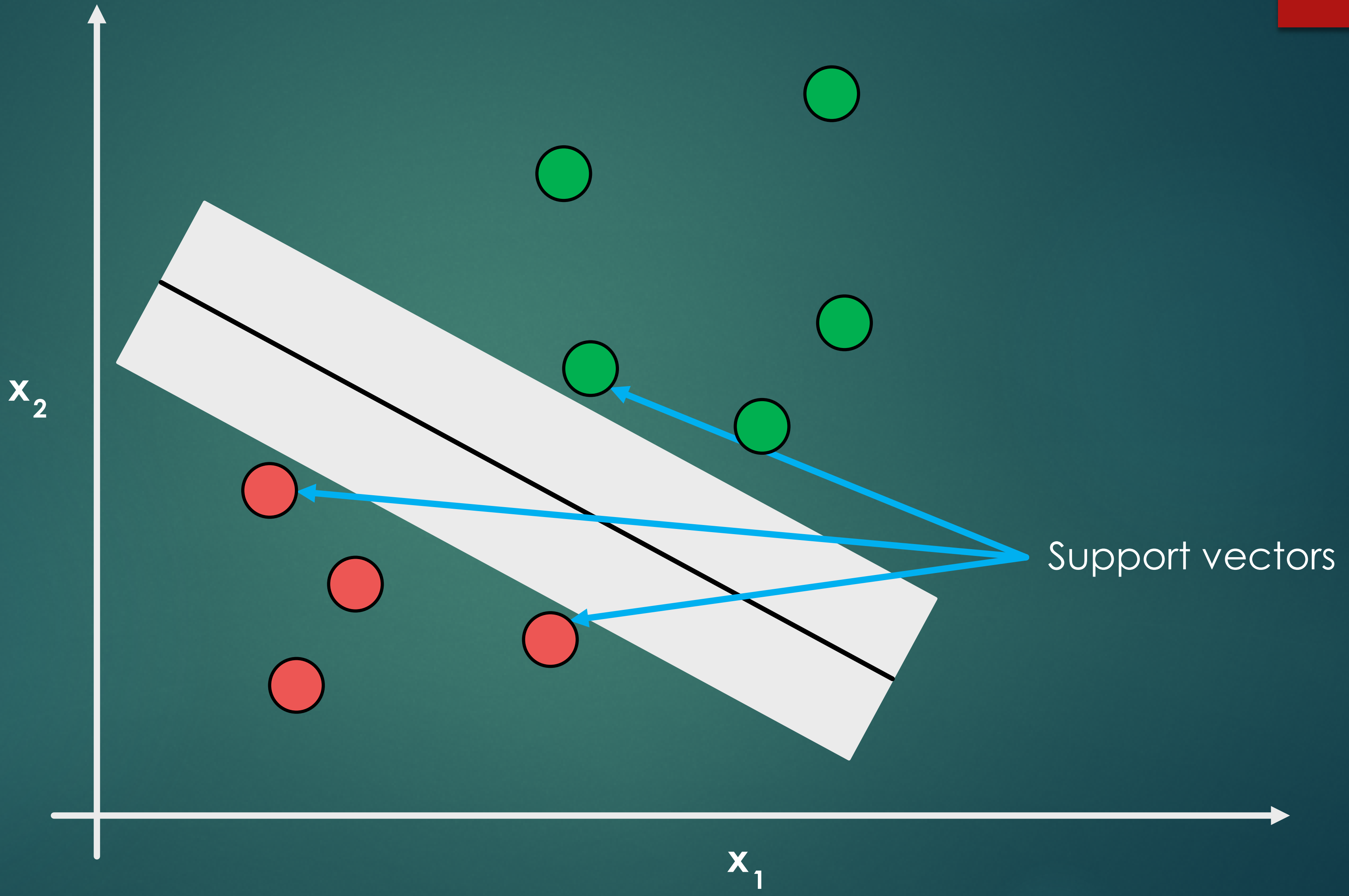


This is the maximum margin solution

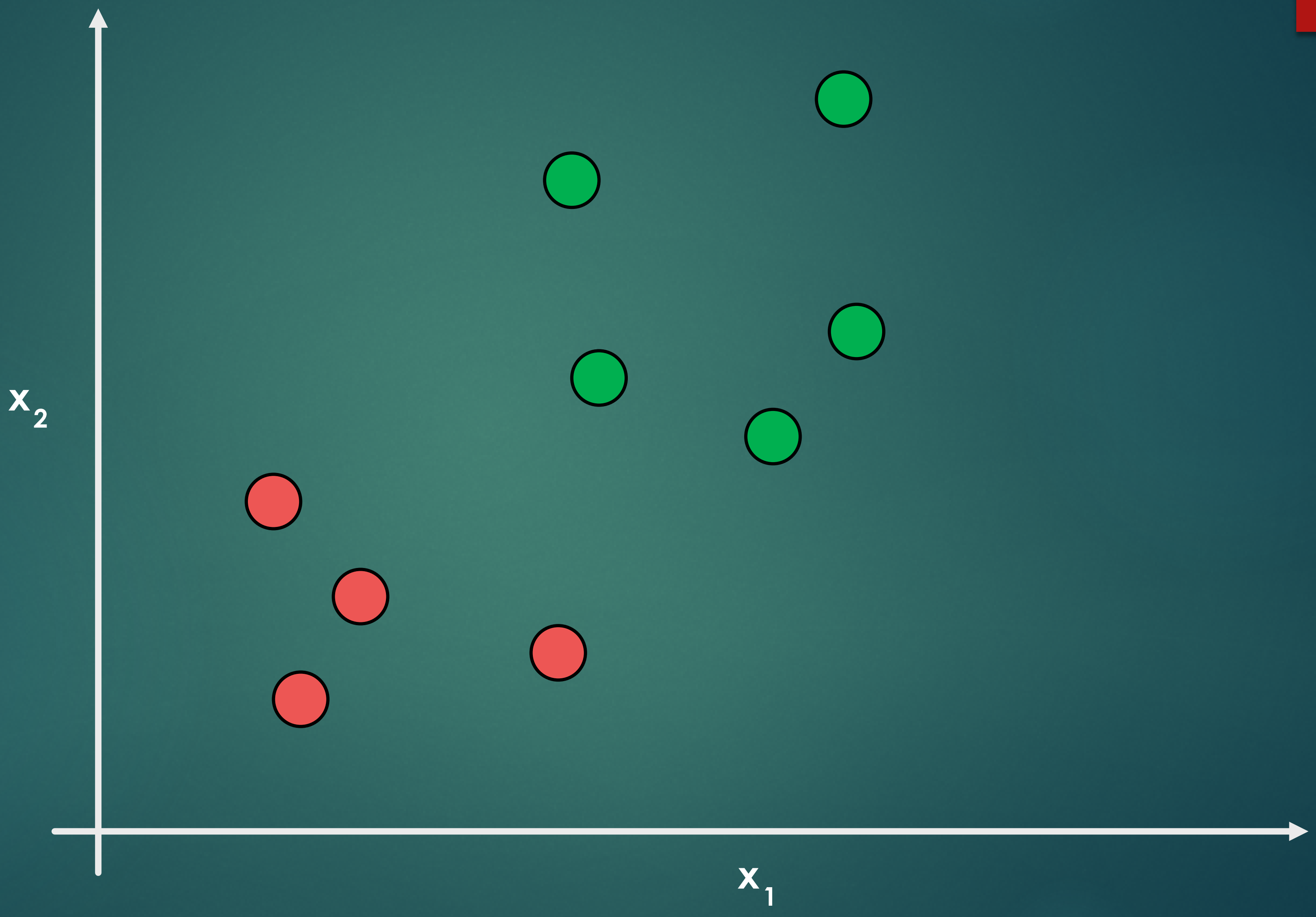
Support vectors: the points from each class that are closest to the maximum margin hyperplane // each class have at least **1** support vector



With the support vectors alone it is possible to reconstruct the hyperplane: it is good !!!
We can store the classification model even when we have millions of features

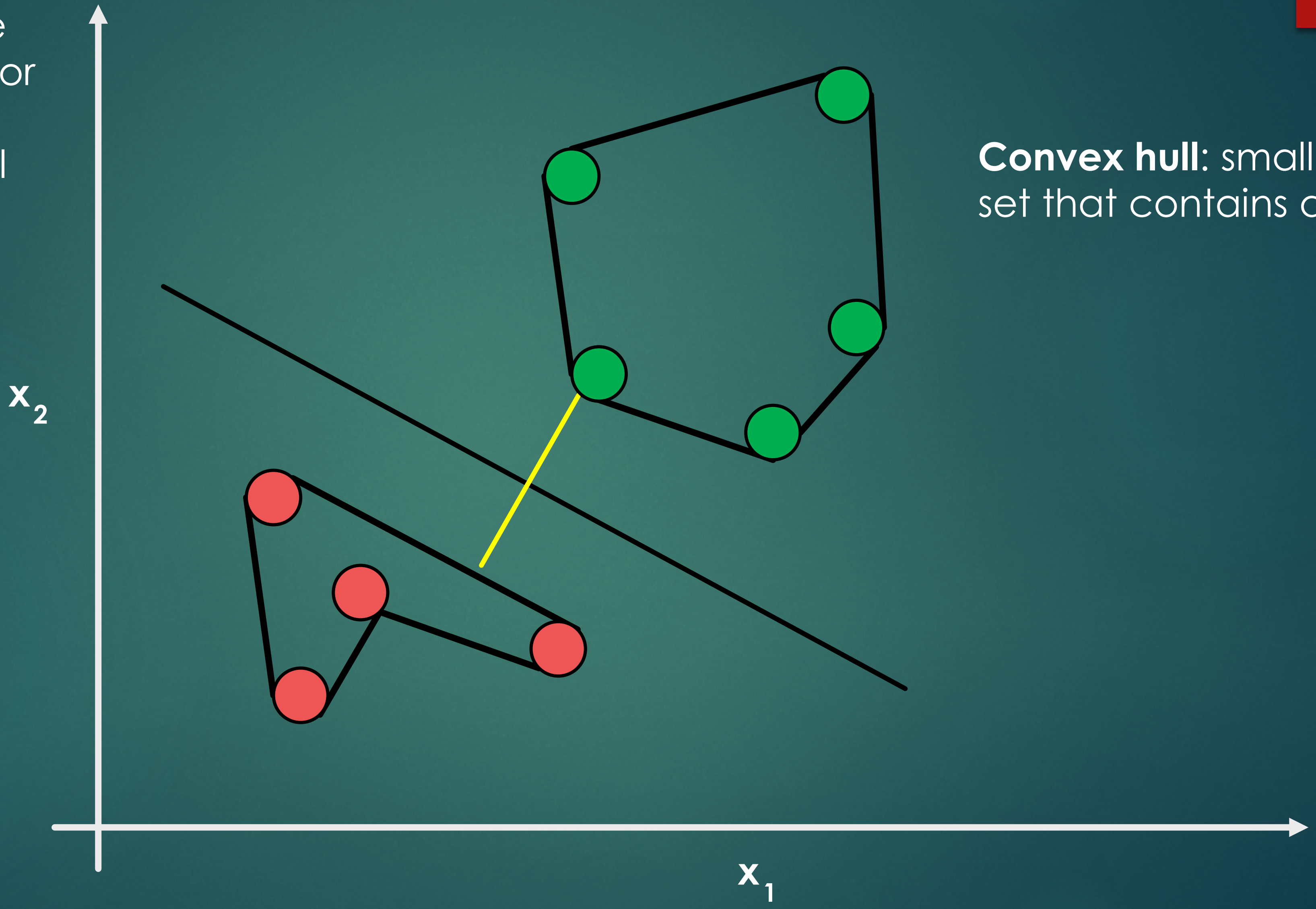


How to find the hyperplane when the problem is linearly separable? With convex hulls



How to find the hyperplane when the problem is linearly separable? With convex hulls

The hyperplane is the perpendicular bisector of the shortest line between the two hull



Convex hull: smallest convex set that contains all the points

Mathematical approach

$$\vec{w} * \vec{x} + b = 0$$

the equation of a hyperplane in **n**-dimensions

In **2D**: $y = m*x + b$

$w_1 \quad w_2 \quad \ldots \quad w_n$

we have the so called weights

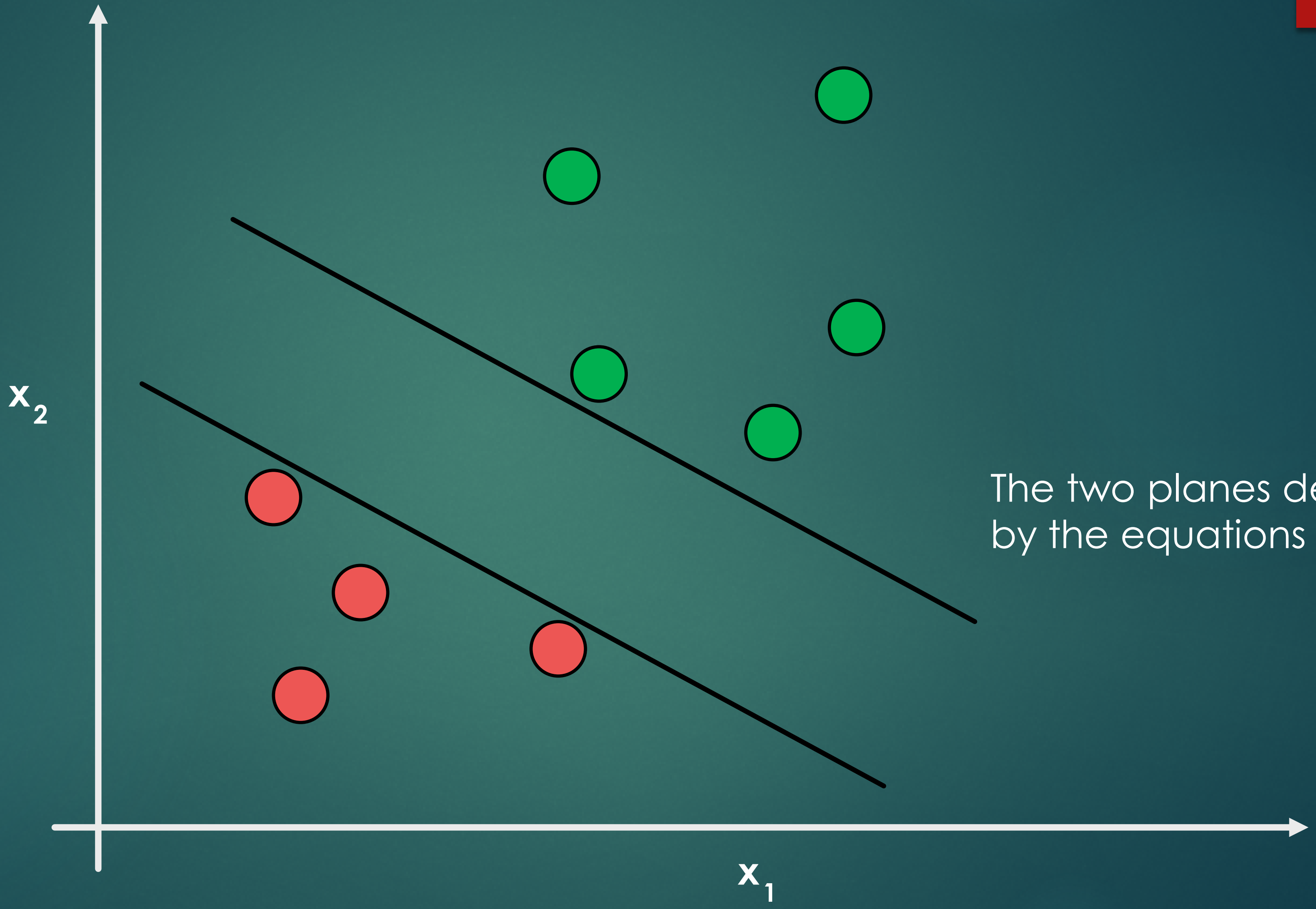
$x_1 \quad x_2 \quad \ldots \quad x_n$

The aim of the **SVM** algorithm is to find the w_n weights so that the data points will be separated accordingly:

$$w * x + b \geq +1$$

$$w * x + b \leq -1$$

How to find the hyperplane in **2D**? With convex hulls



The two planes defined by the equations

Mathematical approach

Vector geometry defines, that the distance between the two planes:

$$\frac{2}{||\vec{w}||}$$ Euclidean-norm (distance from 0)

We want to make the distance as large as possible → so we want to minimize the norm of the w

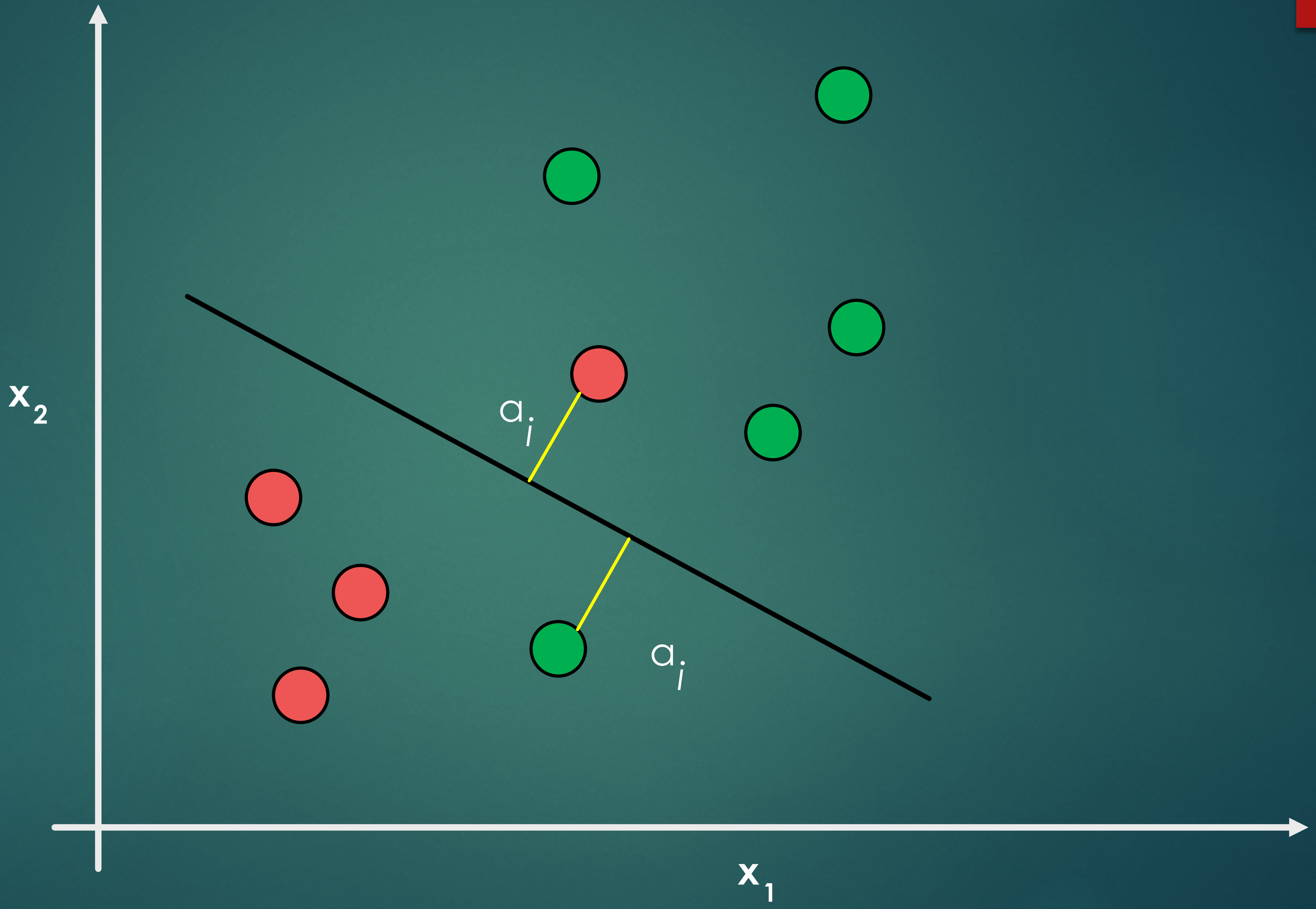
We usually minimize: $$\frac{1}{2}||\vec{w}||^2$$

Quadratic optimization solve this problem !!!

Non-linear spaces

- In many real-world applications, the relationships between variables are non-linear
- A key feature of **SVMs** is their ability to map the problem into a higher dimensional space using a process known as the ***„kernel trick”***
- Non-linear relationship may suddenly appears to be quite linear

We have to use slack variables, it is a non-linearly separable problem



Mathematical approach

We minimize: $$\frac{1}{2}\,||\vec{w}||^2 + C \sum_i a_i$$

C: cost parameter to all points that violate the constraints

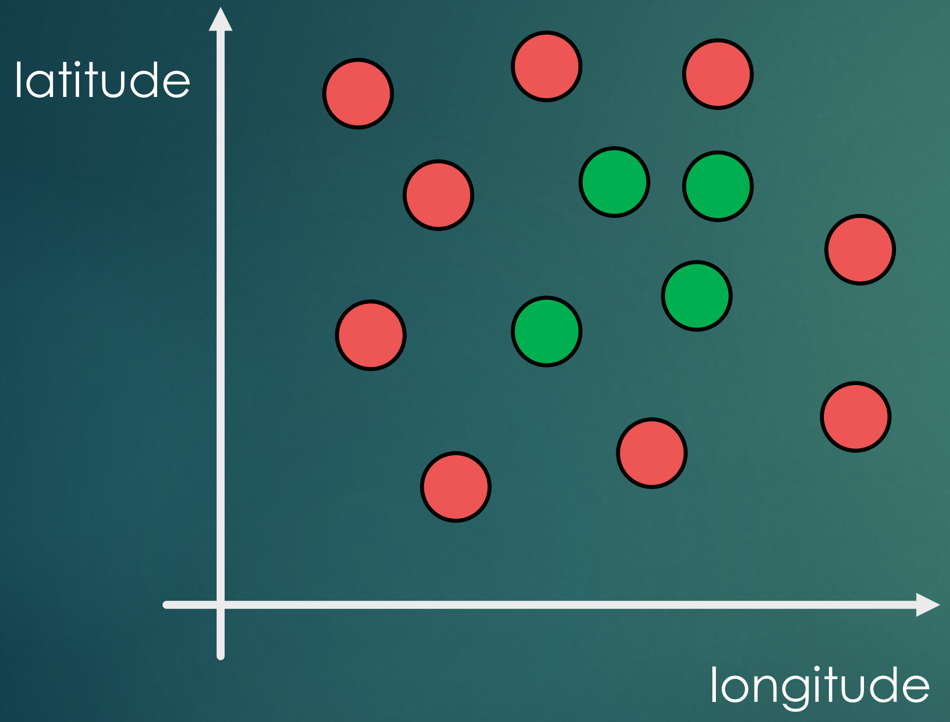
We make our optimization on this cost function

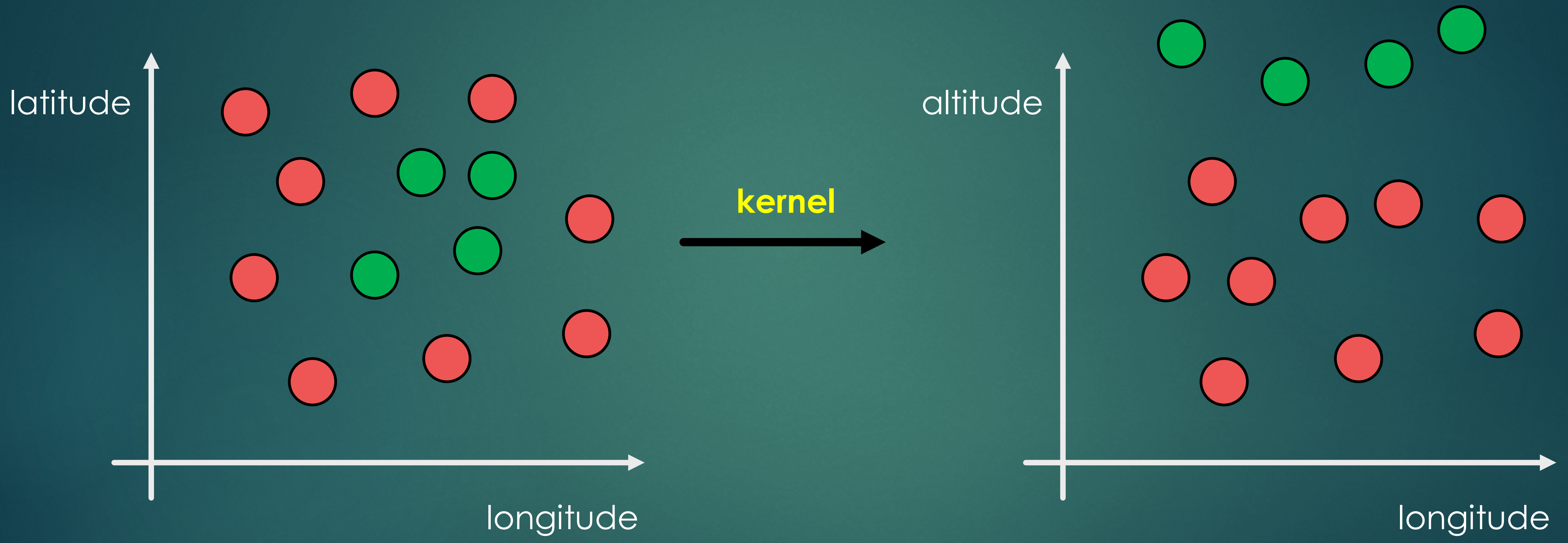
We can tune the **C** parameter: we can modify the penalty for the data points that are misclassified

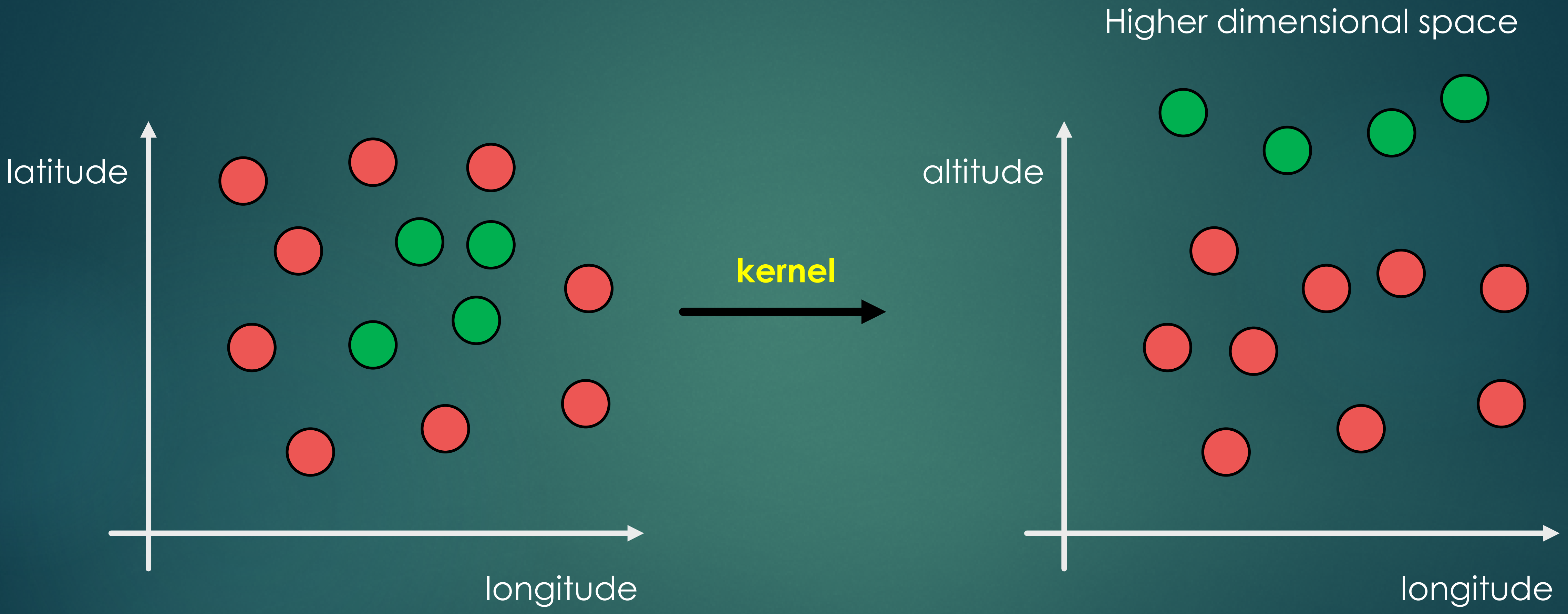
C is very large → the algorithm tries to find a **100%** separation
C is low → wider overall margin is allowed with more misclassified data points

Kernels

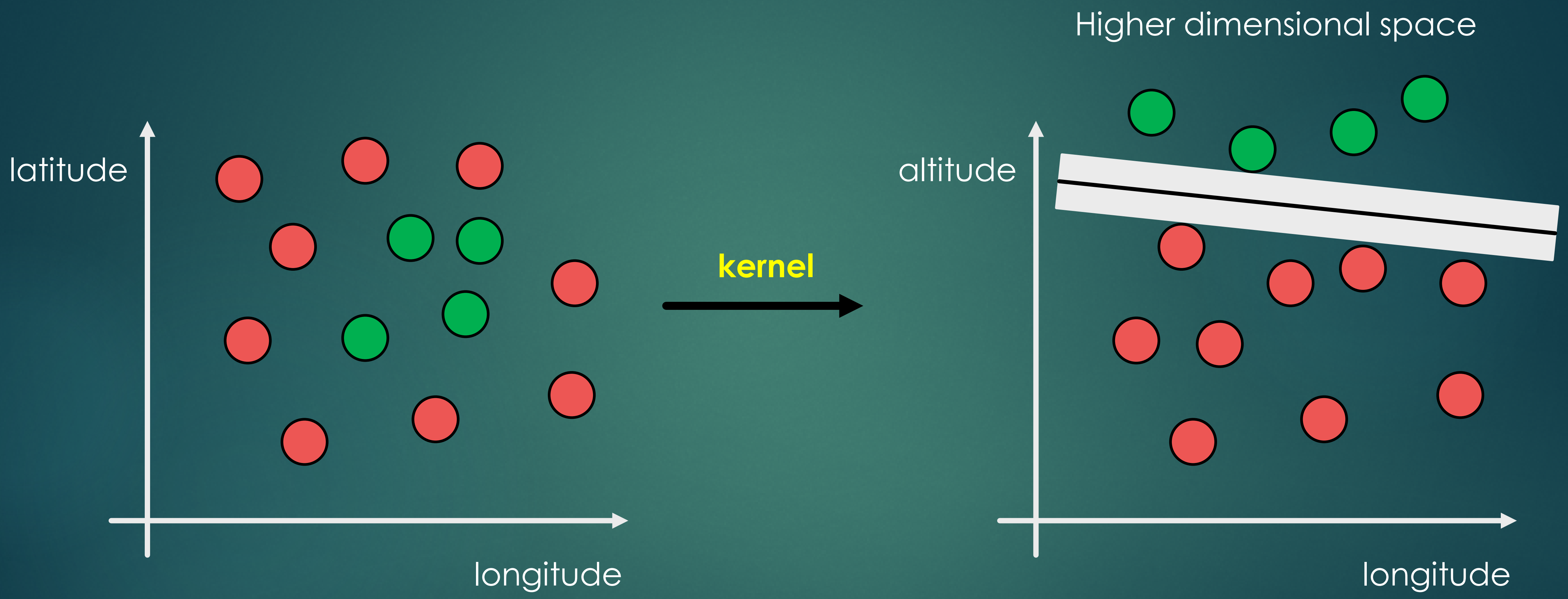
It can be weather classes: sunny and snowy



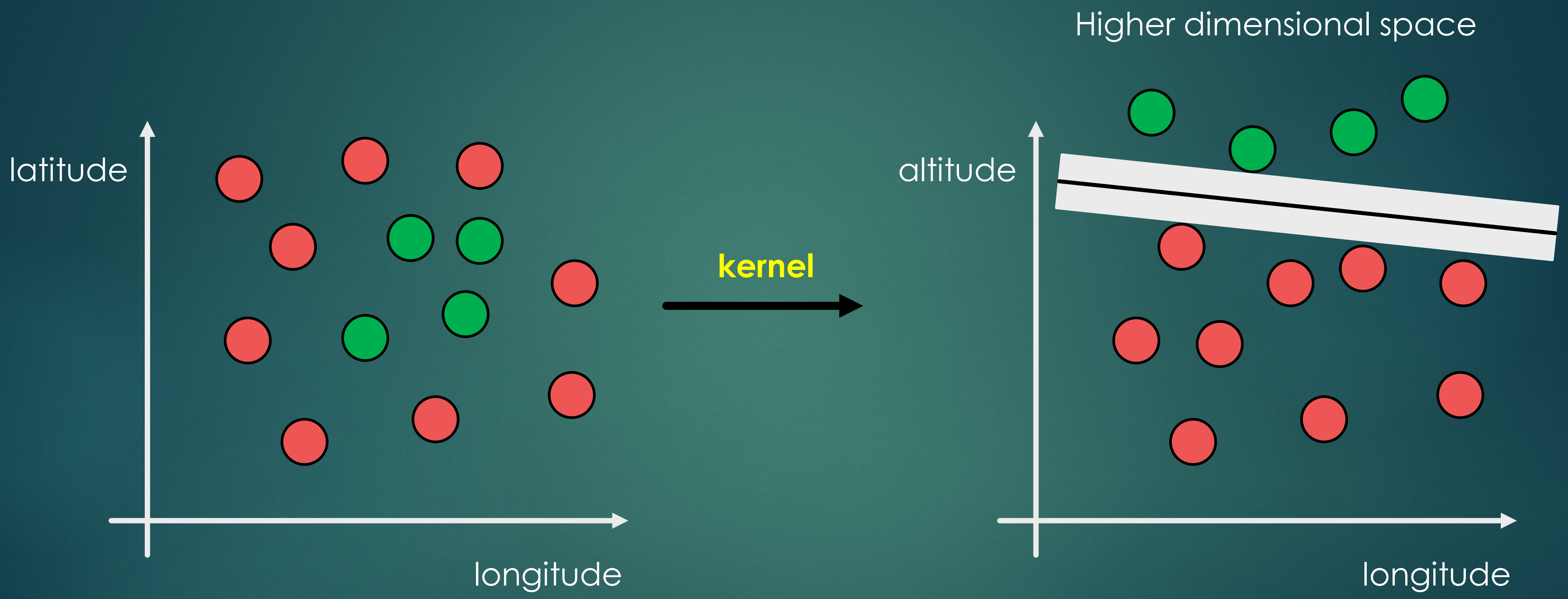

latitude
longitude
kernel
altitude
longitude



With the ***kernel function*** we can transform the problem into linearly separable one !!! (slack variable: altitude)



With the ***kernel function*** we can transform the problem into linearly separable one !!! (slack variable: altitude)



SVM learns concepts that were not explicitly measured in the original data !!!

Kernel functions

Φ(x) „phi function"

This is the mapping of data **x** into an other space

$K(\underline{x}_i, \underline{x}_j)$ this is the kernel function

$$K(\underline{x}_i, \underline{x}_j) = \underline{x}_i * \underline{x}_j$$

linear kernel: does not transform the data

$$K(\underline{x}_i, \underline{x}_j) = (\underline{x}_i * \underline{x}_j + 1)^d$$

polynomial kernel

$$K(\underline{x}_i, \underline{x}_j) = \exp \frac{-\left\| \underline{x}_i - \underline{x}_j \right\|^2}{2*\sigma^2}$$

gaussian **RBF** kernel

Support vector machines

- **SVMs** with non-linear kernels add additional dimensions to the data in order to create separation in this way
- Kernel trick → process of adding new features that express mathematical relationships between measured characteristics
- This allows the **SVM** to learn concepts that were not explicitly measured in the original data

Advantages

- **SVM** can be used for regression problems as well as for classifications
- Not overly influenced by noisy data
- Easier to use than neural networks

Disadvantages

- Finding the best model requires testing of various combinations of kernels and model parameters
- Quite slow → especially when the input dataset has a large number of features
- Black box model: very hard to understand !!!