Università degli Studi di Padova

Machine Learning
2024/2025

AMCO
ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND CONTROL RESEARCH GROUP

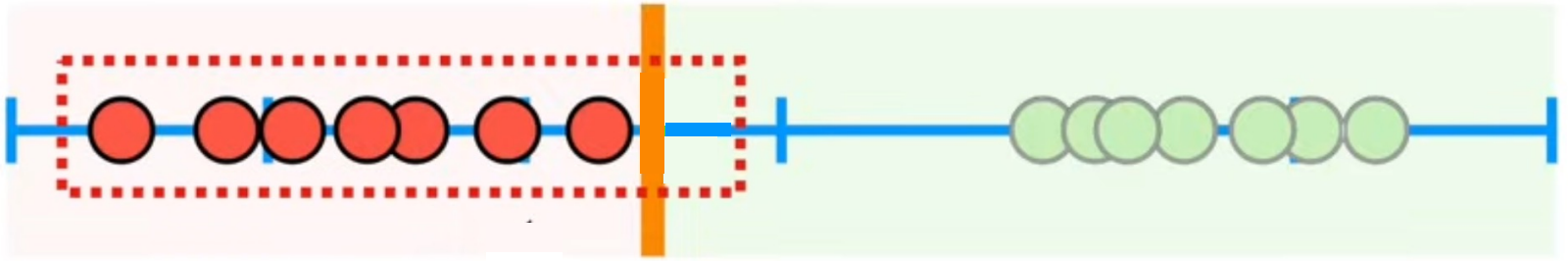
Lecture #22
Support Vector Machines (SVM)

Gian Antonio Susto

Disclaimer: SVM and exam

Support Vector Machines will only be evaluated on the theoretic part of the exam!

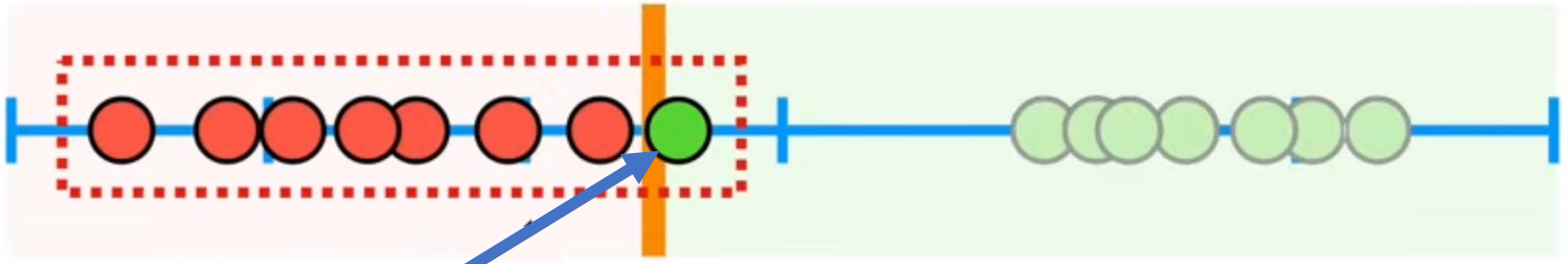
Let’s go back to supervised classification!



We are considering a binary classification problem (red vs green class)

Our classifier has defined a decision boundary which determines the decision between the two classes.

Let’s go back to supervised classification!



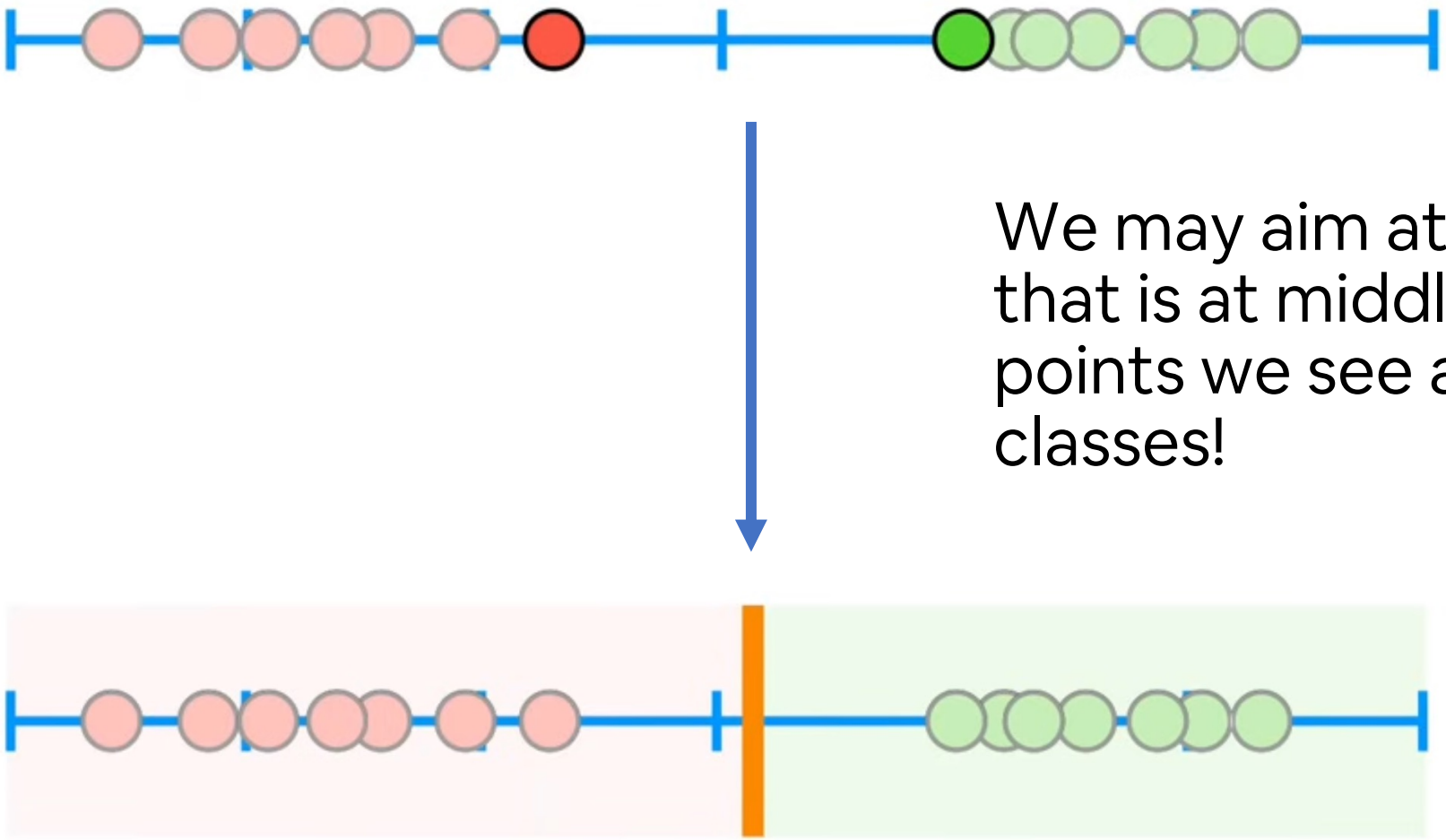
We are considering a binary classification problem (red vs green class)

Our classifier has defined a decision boundary which determines the decision between the two classes.

A new data point arrives: does the current boundary seems like a good idea?

Kudos to Joshua Starmer!

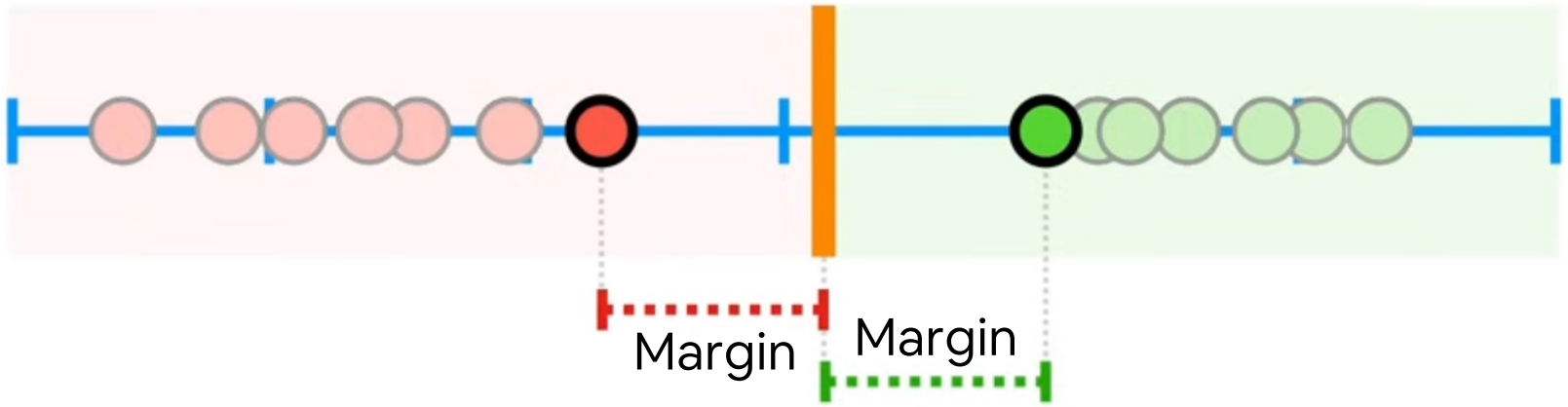
Let’s go back to supervised classification!



We may aim at a decision boundary that is at middle of the two data points we see at middle of the two classes!

Kudos to Joshua Starmer!

The Maximal Margin Classifier

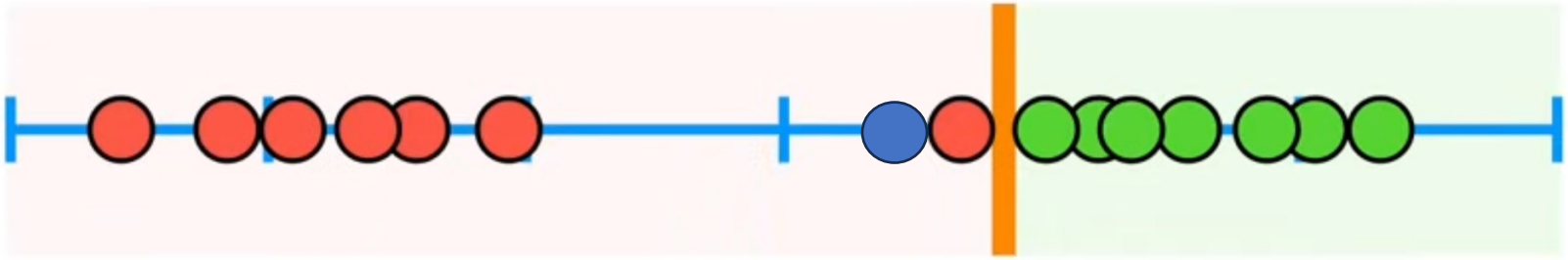


The distance between the boundary observations and the threshold is called the margin (in this example, the margin on the left and on the right are the same)

A **maximal margin classifier** is a type of linear classifier that aims to find the decision boundary (or hyperplane) that maximizes the margin between two classes

Kudos to Joshua Starmer!

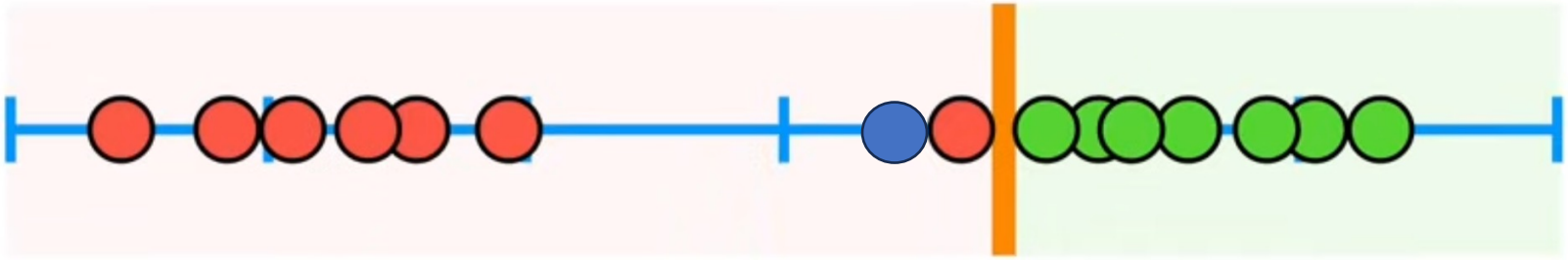
The Maximal Margin Classifier



Unfortunately, **maximal margin classifier** are really sensitive about outliers: what if we get the blue data point to be classified? 🔵

Kudos to Joshua Starmer!

The Maximal Margin Classifier



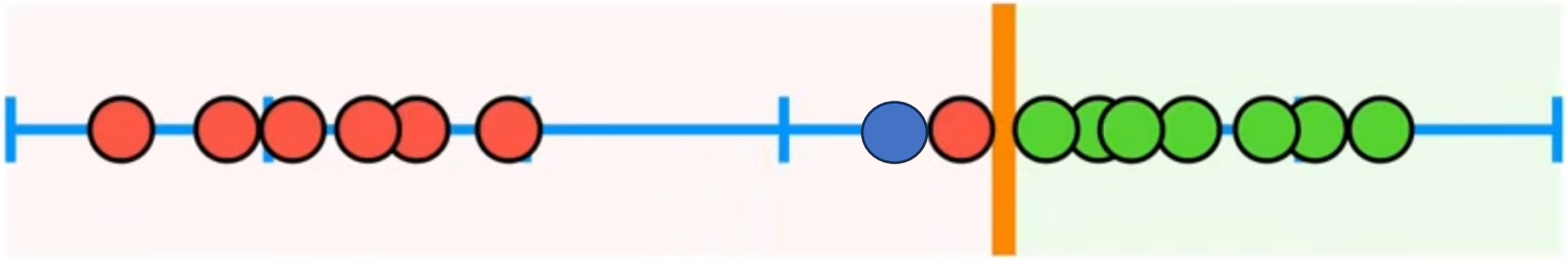
Unfortunately, **maximal margin classifier** are really sensitive about outliers: what if we get the blue data point to be classified? 🔵

This first threshold is really sensitive to historical data (overfitting, high variance!)

How to cope with this? Ideas?

Kudos to Joshua Starmer!

The Maximal Margin Classifier



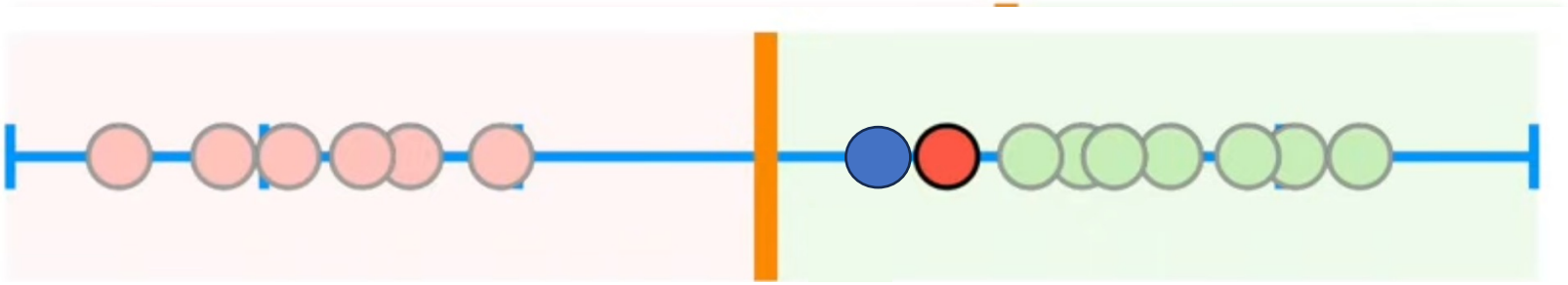
Unfortunately, **maximal margin classifier** are really sensitive about outliers: what if we get the blue data point to be classified? 🔵

This first threshold is really sensitive to historical data (overfitting, high variance!)

How to cope with this? Ideas?

Kudos to Joshua Starmer!

Moving beyond the Maximal Margin Classifier

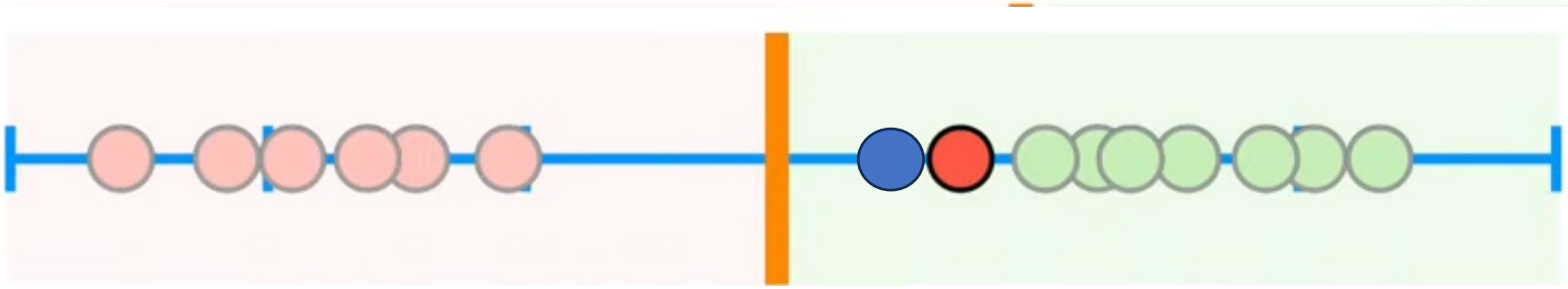


We can choose a threshold that allow misclassification in training!

This will allow us to properly classify the blue data point, more in general to tune the bias/variance tradeoff!

Kudos to Joshua Starmer!

Moving beyond the Maximal Margin Classifier



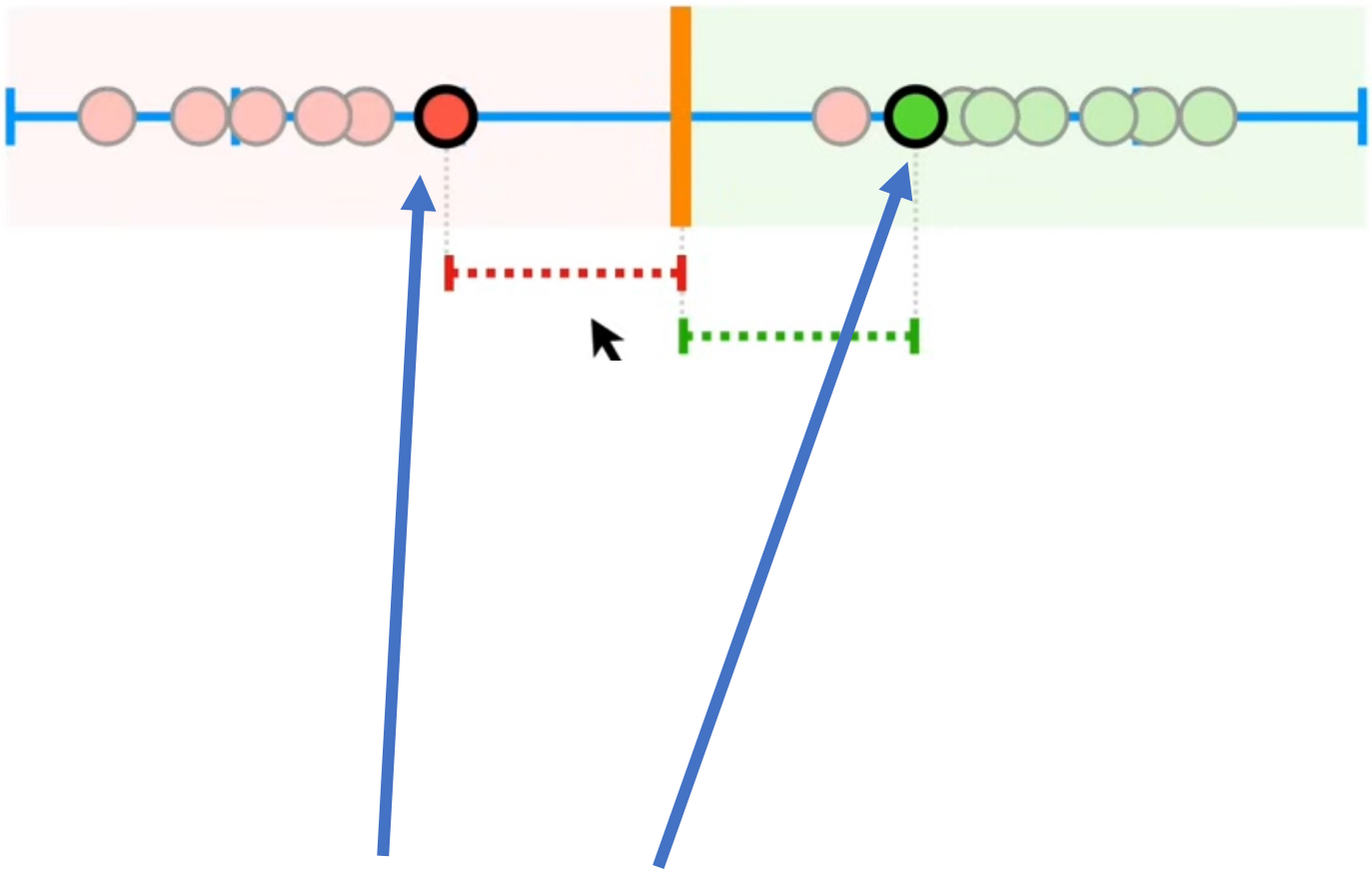
We can choose a threshold that allow misclassification in training!

This will allow us to properly classify the blue data point, more in general to tune the bias/variance tradeoff!

This second threshold may reduce variance at the expenses of some bias!

Kudos to Joshua Starmer!

Soft Margin & Support Vector Classifier

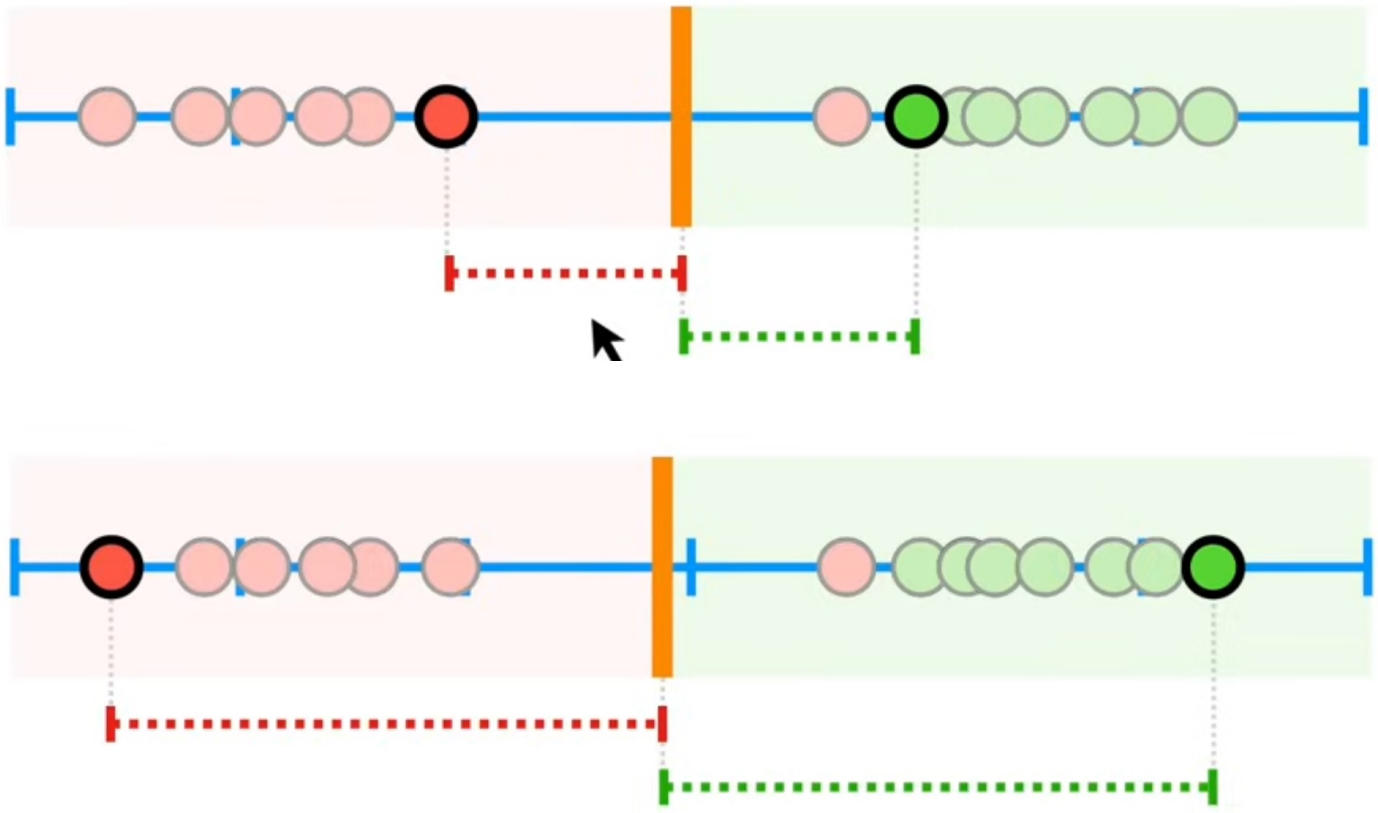


If we allow misclassification in training, the distance between the correctly classified observations ‘at the boundaries’ is the soft margin

The ‘boundary’ data points (the ones that defines the solutions) are called support vectors -> with Soft Margins, we are calling this solution Support Vector Classifier!

Kudos to Joshua Starmer!

Soft Margin & Support Vector Classifier



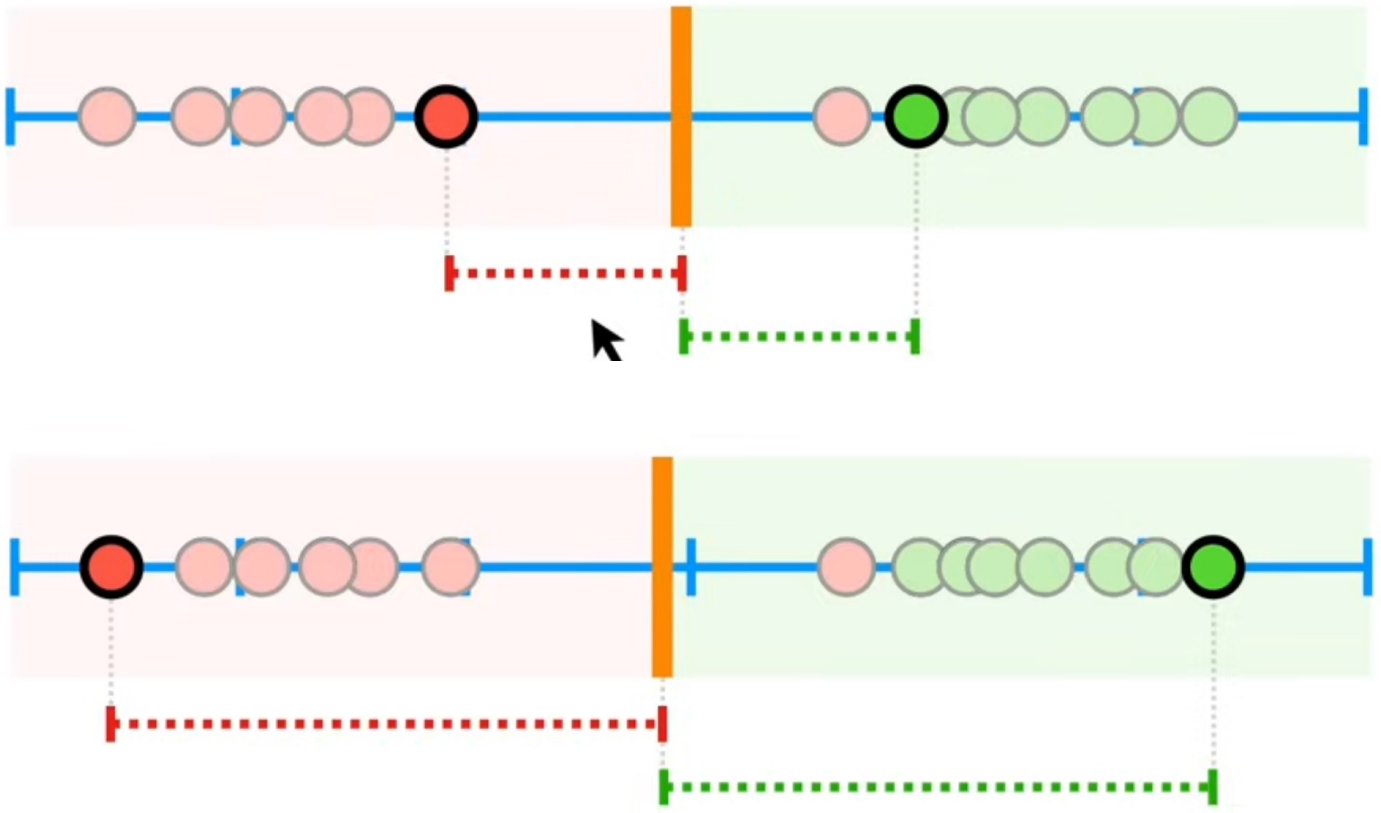
If we allow misclassification in training, the distance between the correctly classified observations ‘at the boundaries’ is the soft margin

How do we choose the ‘best’ soft margin?

The ‘boundary’ data points (the ones that defines the solutions) are called support vectors -> with Soft Margins, we are calling this solution Support Vector Classifier!

Kudos to Joshua Starmer!

Soft Margin & Support Vector Classifier



If we allow misclassification in training, the distance between the correctly classified observations ‘at the boundaries’ is the soft margin

How do we choose the ‘best’ soft margin?

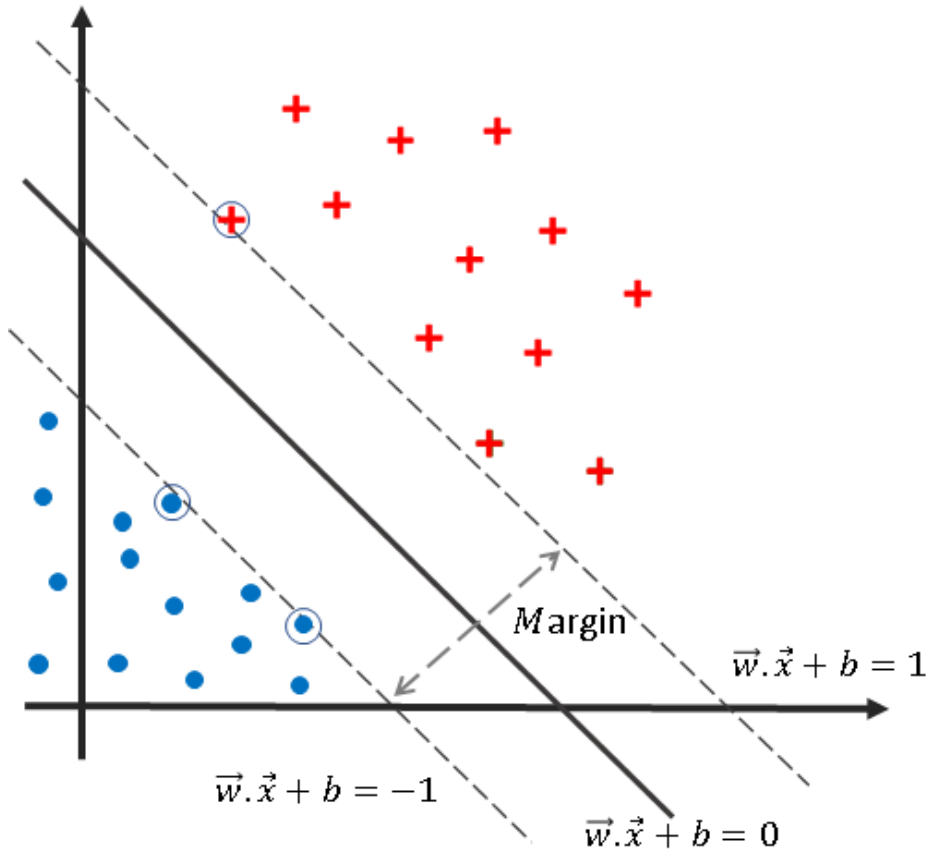
Cross-validation!

The ‘boundary’ data points (the ones that defines the solutions) are called support vectors -> with Soft Margins, we are calling this solution Support Vector Classifier (SVC)

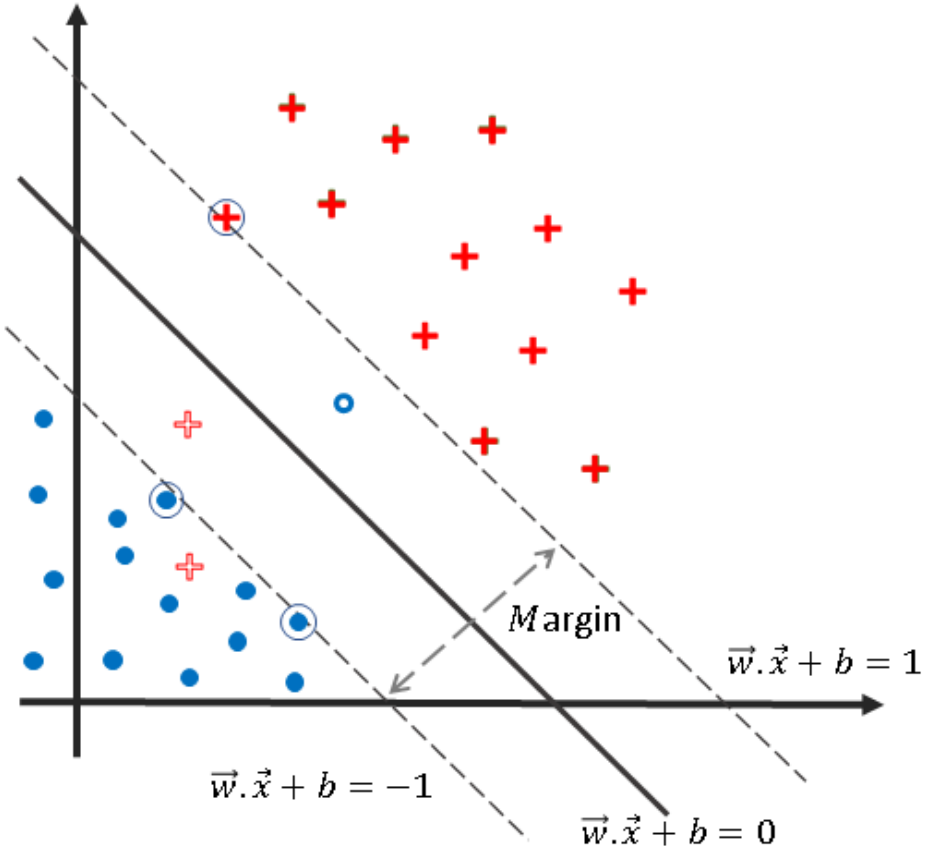
Kudos to Joshua Starmer!

In 2-dimensional data

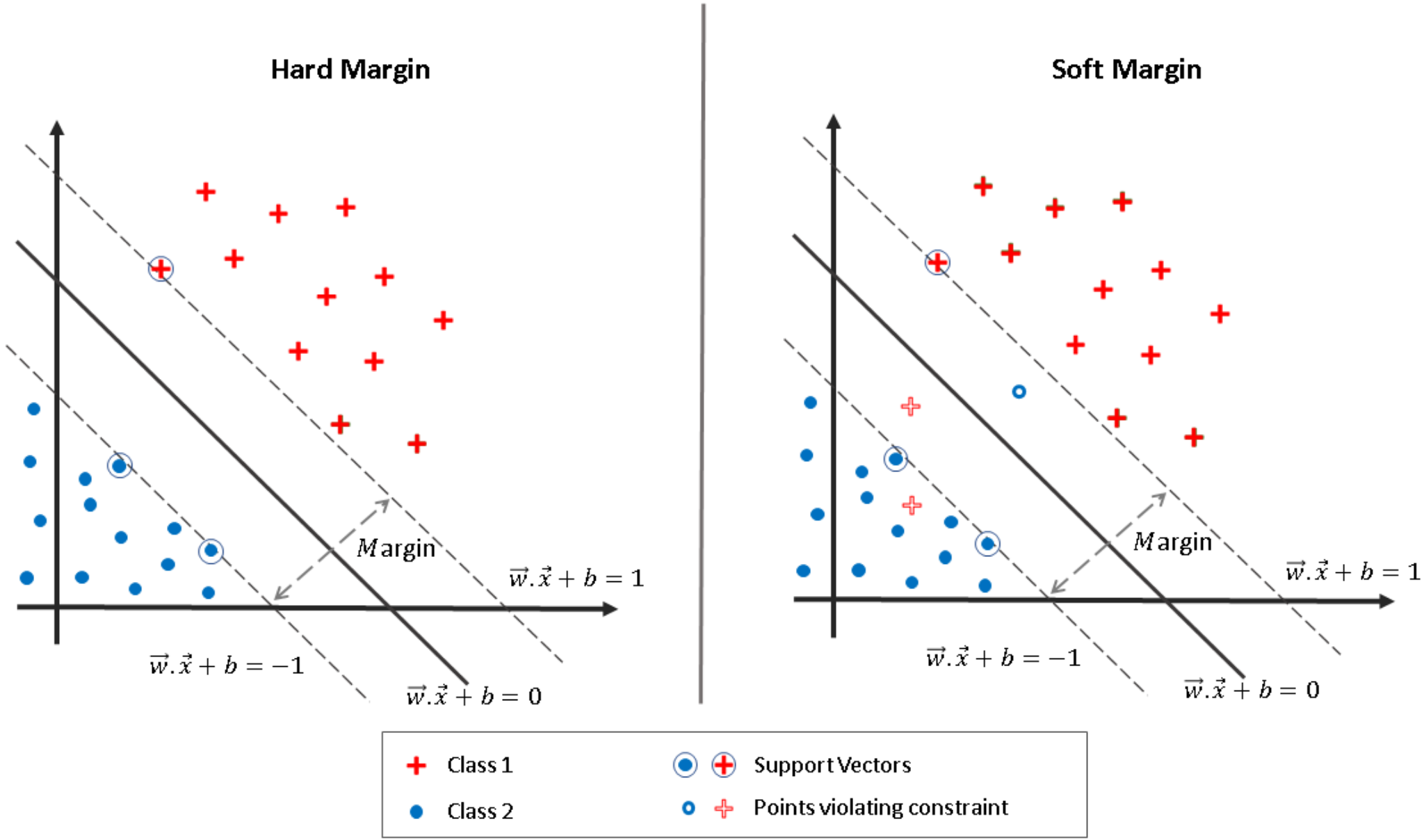
Data are linearly separable: I can find a plane that separates the two classes



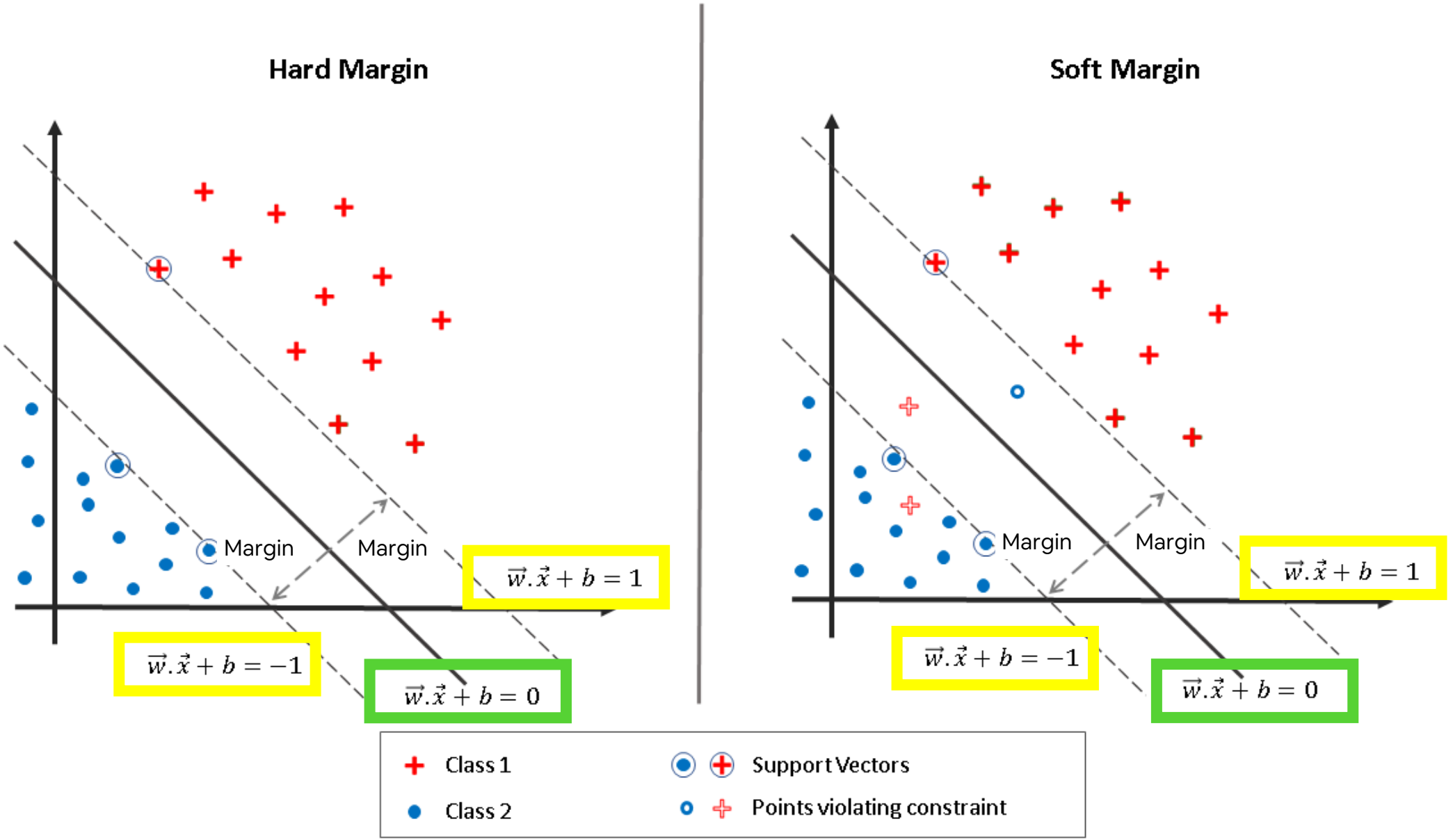
Data are not linearly separable: I cannot find a plane to separate the two classes



In 2-dimensional data



In 2-dimensional data



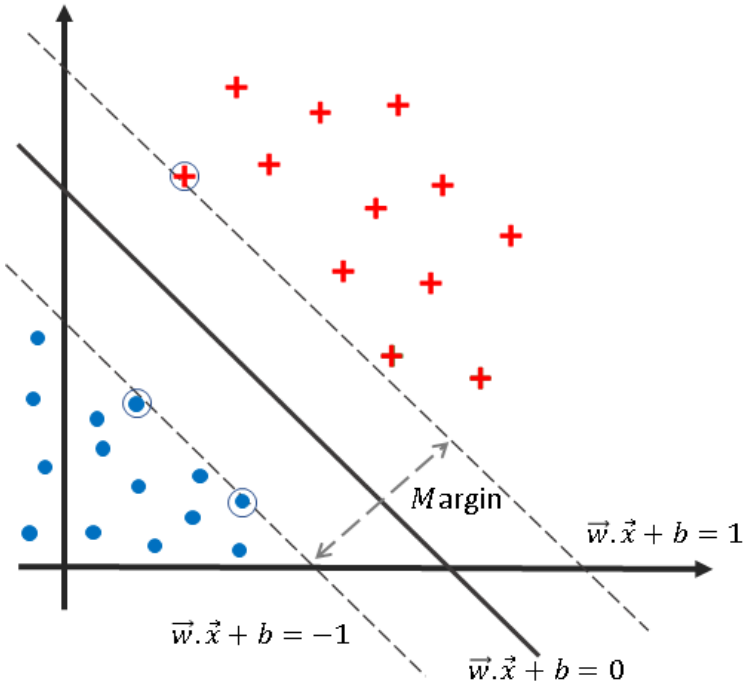
Notation

The margin is the distance between the decision boundary (the hyperplane $w \cdot x + b = 0$) and the closest support vectors.

💡 Key idea: The SVC creates two parallel hyperplanes:

- $w \cdot x + b = +1$
- $w \cdot x + b = -1$

These lie on either side of the decision boundary and touch the support vectors. The total margin is the distance between these two hyperplanes.



The distance from the center hyperplane to one of the margin lines:

$$\text{Margin} = \frac{1}{\|w\|}$$

Total margin width: $\frac{2}{\|w\|}$

How to find the best hyperplane?

SVM solves a convex optimization problem (linearly separable case):

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{Subject to} \qquad y_i(w \cdot x_i + b) \geq 1 \quad \forall i$$

How to find the best hyperplane?

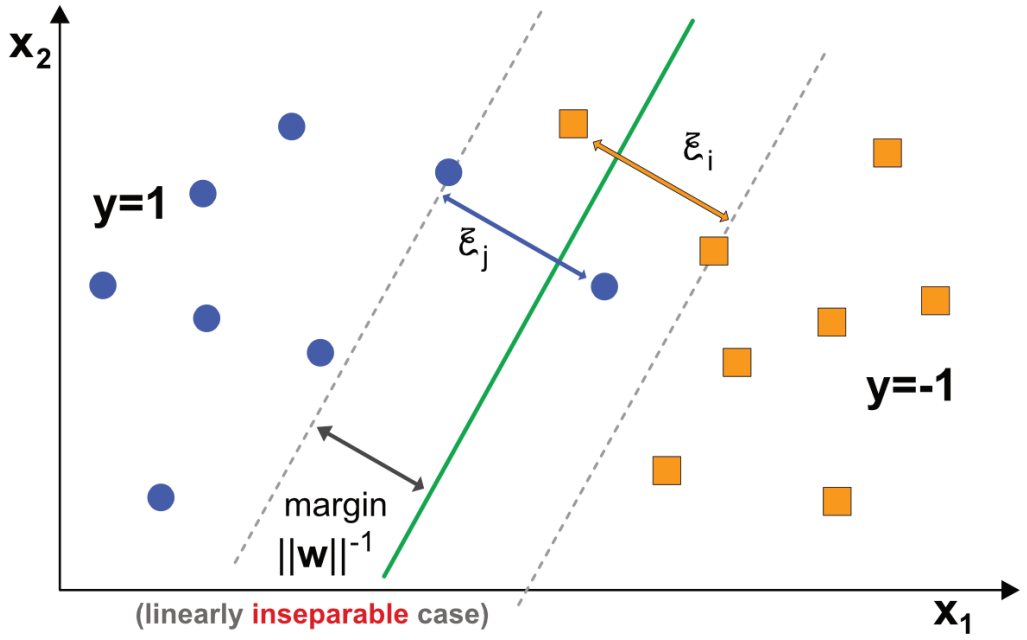
SVM solves a convex optimization problem (linearly separable case):

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{Subject to} \qquad y_i(w \cdot x_i + b) \geq 1 \quad \forall i$$

If it is not separable, we allow some "slack" (errors) by introducing variables ξ:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_i$$

Subject to $y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$



How to find the best hyperplane?

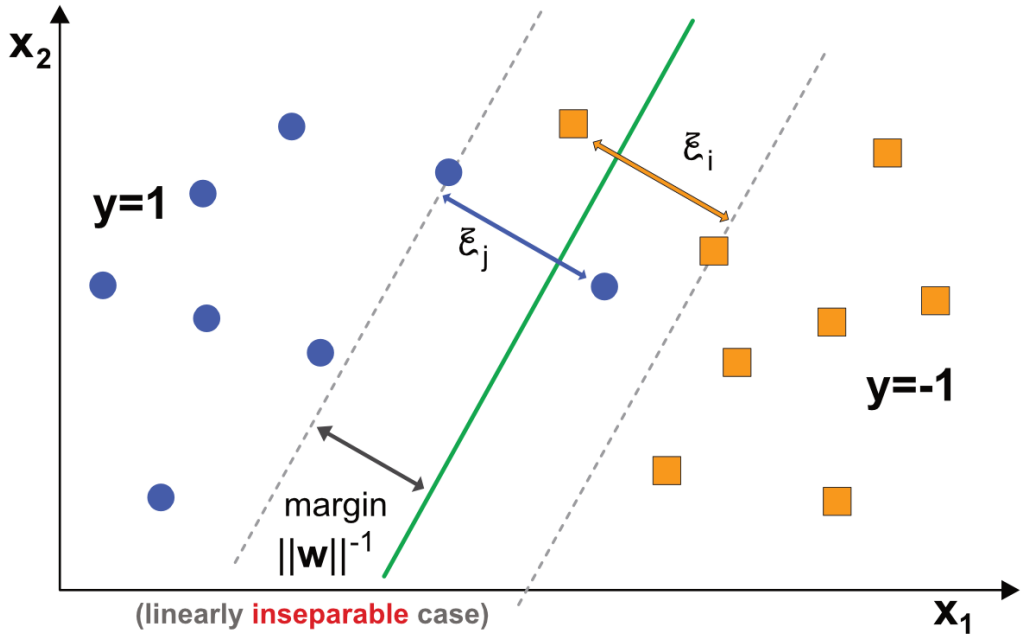
SVM solves a convex optimization problem (linearly separable case):

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{Subject to} \qquad y_i(w \cdot x_i + b) \geq 1 \quad \forall i$$

If it is not separable, we allow some "slack" (errors) by introducing variables ξ:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_i$$

Subject to $y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$

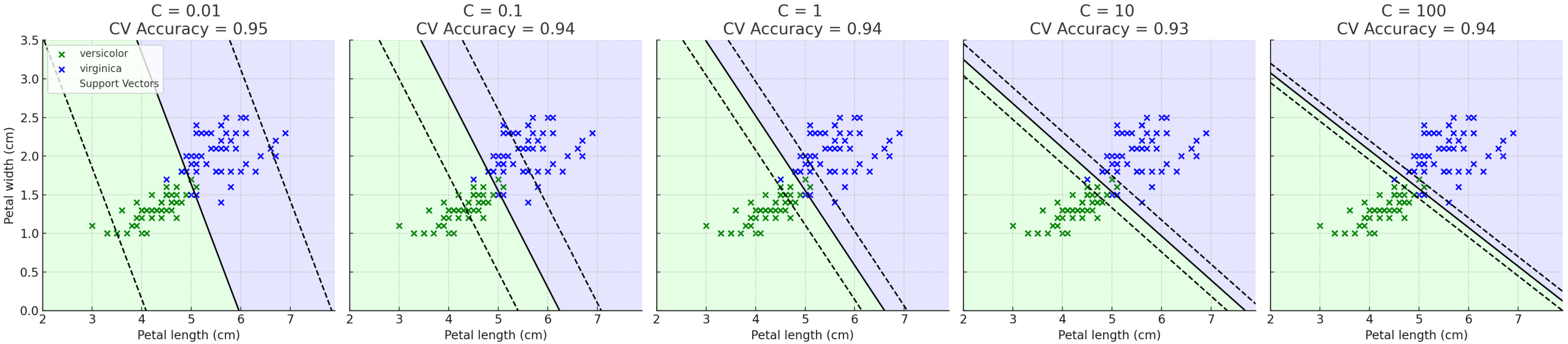


➔ ξ allows some points to be inside or on the wrong side of the margin

➔ C is the regularization parameter controlling how much you care about errors

On the Iris Dataset

Iris dataset: only virginica and versicolour, only petal length and width!



Here we allow the classifier to have some misclassification

Here we try to have all historical data point correctly classified

Can we solve it with Gradient Descent?

Yes! We can rewrite the SVC problem using a loss function, like the hinge loss:

$$\text{Loss} = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \max(0, 1 - y_i(w \cdot x_i + b))$$

This is called the primal form of SVM with hinge loss, and we can apply gradient descent to minimize it.

However, Gradient Descent is not the best choice for solving SVC:

- There are more efficient algorithms for SVCs like SMO (Sequential Minimal Optimization)

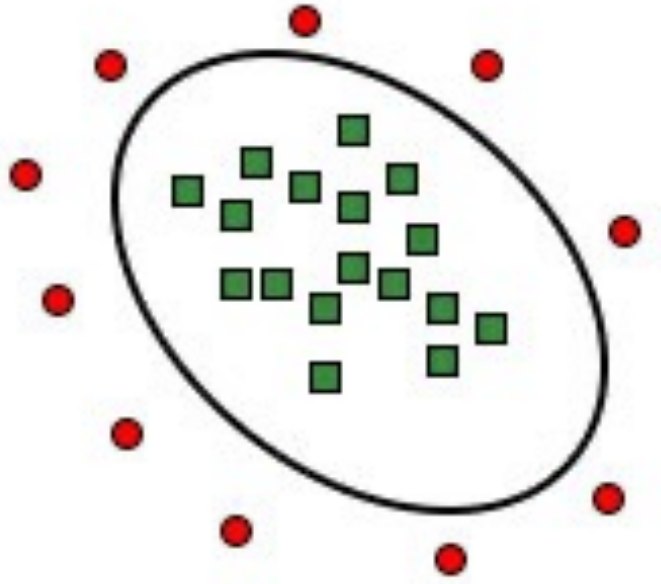
- SVC is convex, so specialized solvers (e.g., quadratic programming) can find the solution faster and exactly

- Gradient descent may be slow, especially for large or high-dimensional datasets

Moving beyond linear decision boundaries

What to do if data look like this?

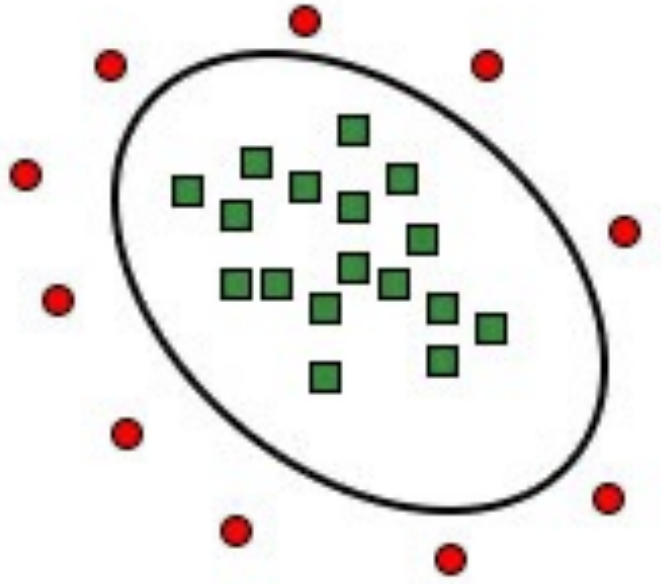
Or like this?



Moving beyond linear decision boundaries

What to do if data look like this?

Or like this?

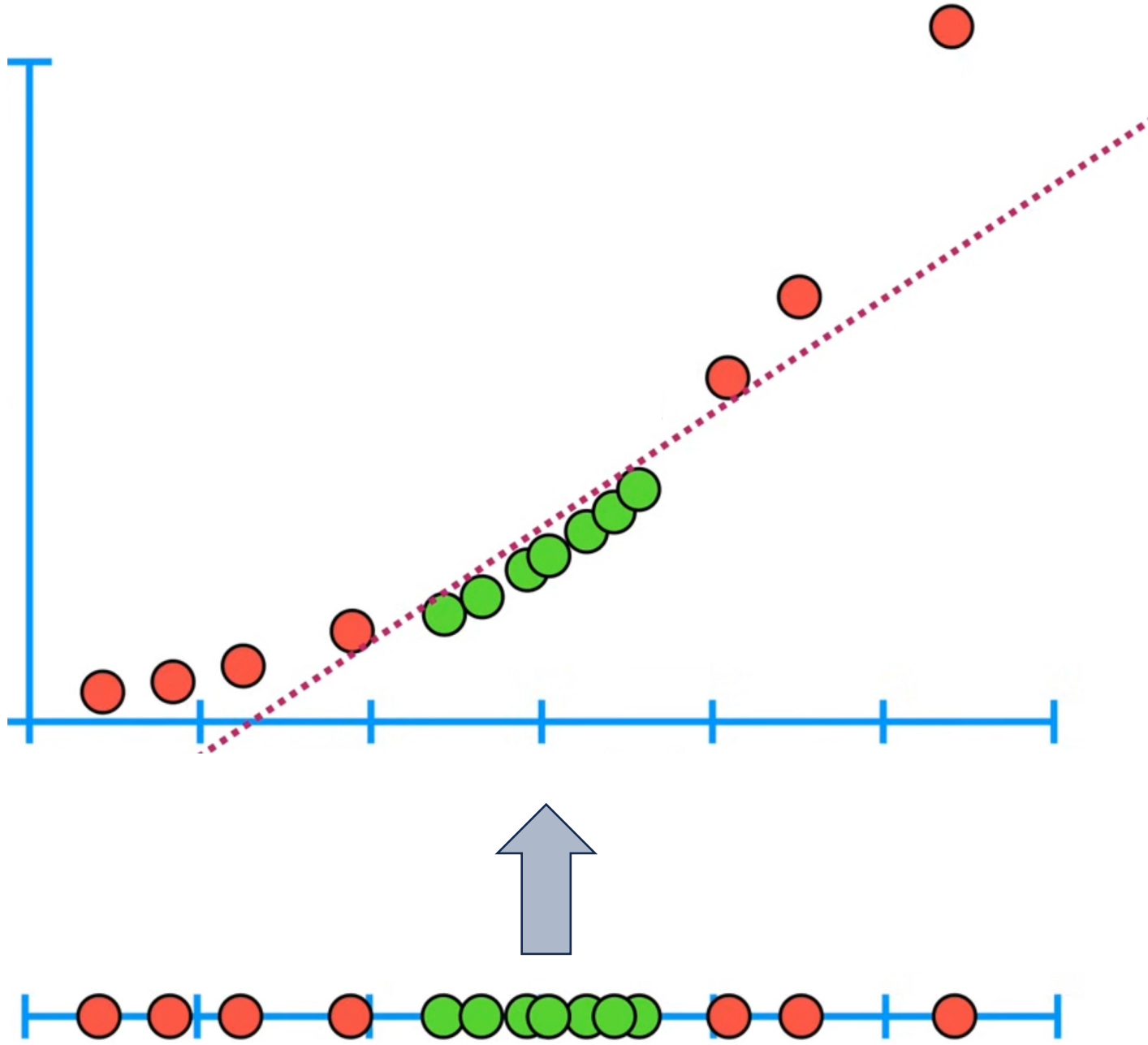


A hyperplane cannot be fit in this case, unless we are ‘transforming’ the data somehow… what if we add another dimension?

In higher dimensions

Let’s consider for example the square of the original data: we make a basis extension.

In this new two-dimensional space, the data are linearly separable!

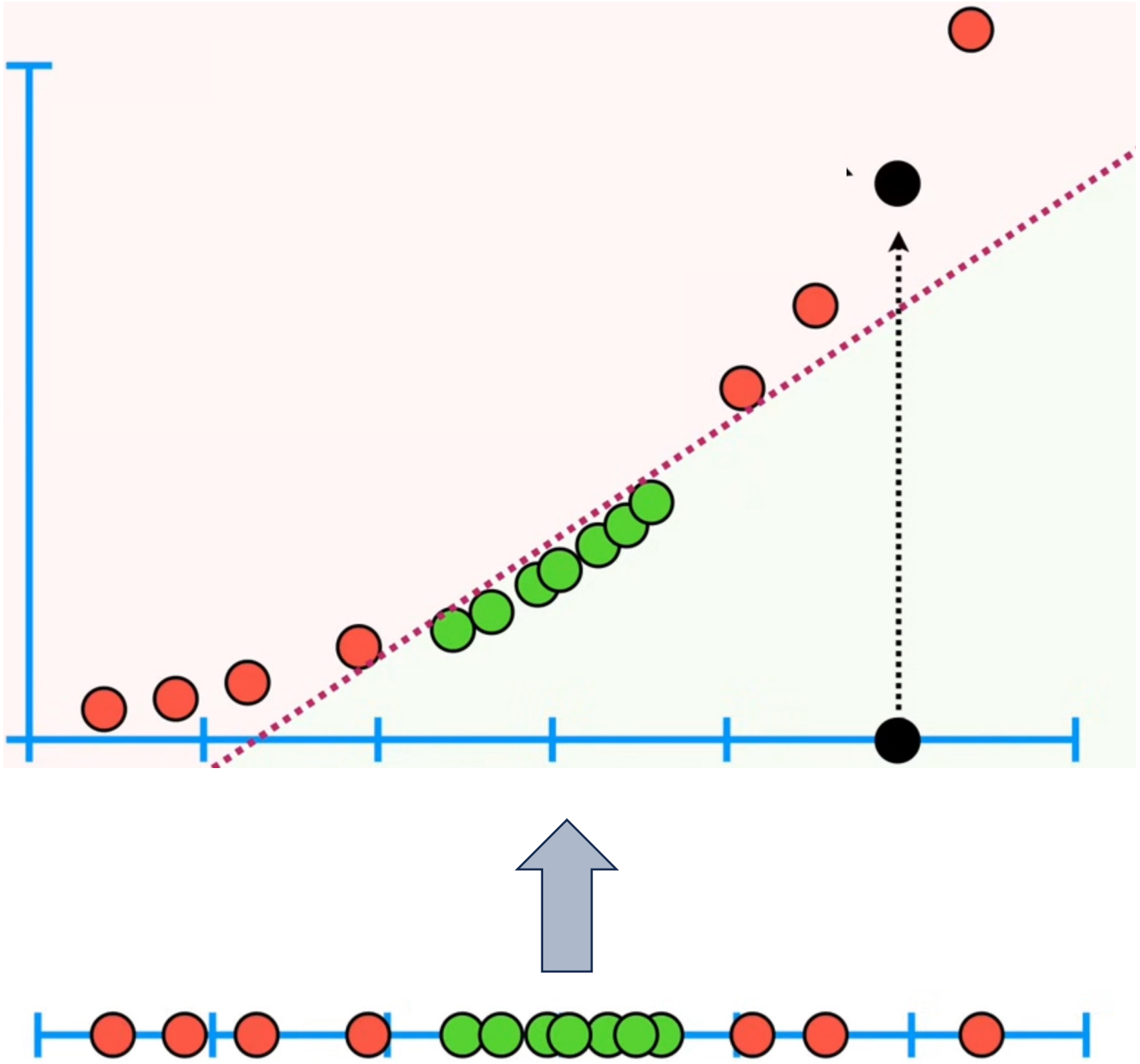


In higher dimensions

Let’s consider for example the square of the original data: we make a basis extension.

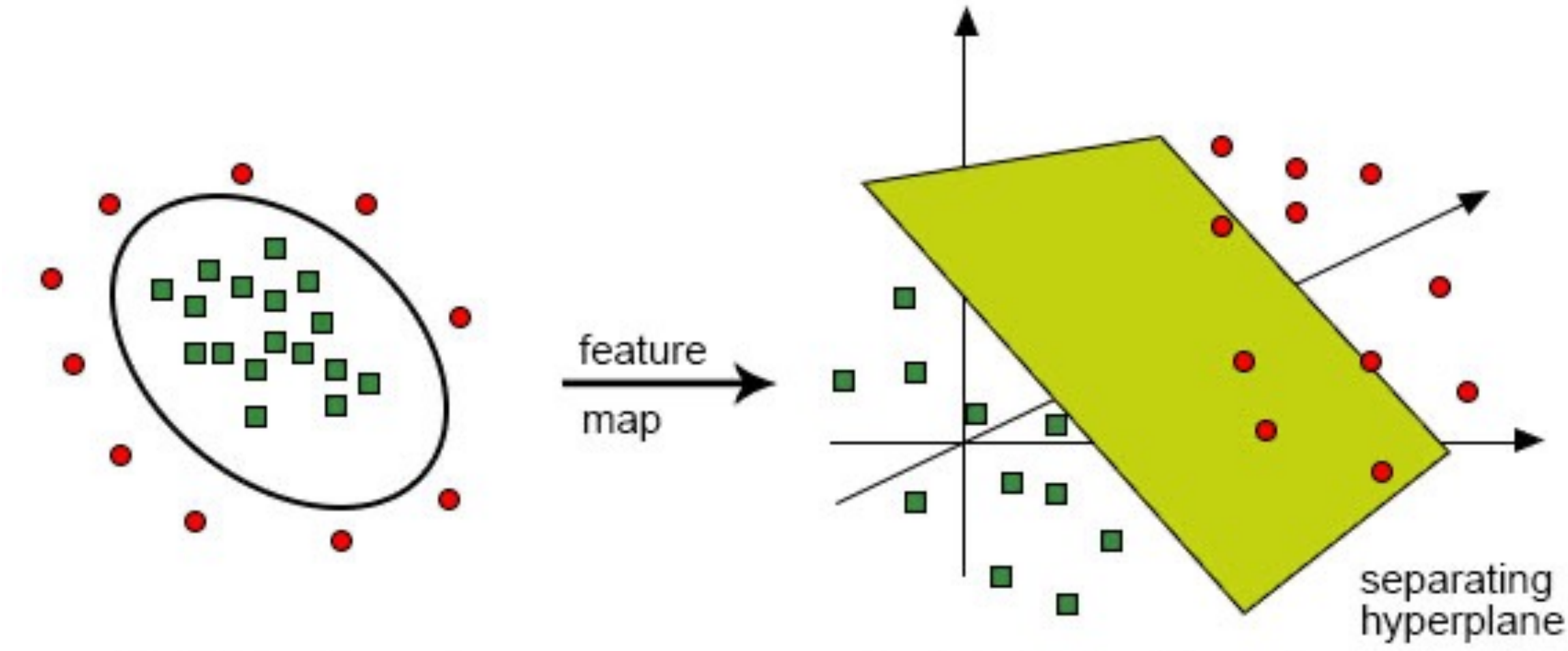
In this new two-dimensional space, the data are linearly separable!

We can then derive a new classifier that properly divides data



In higher dimensions

We can do the same thing for a p = 2, 3, … dimensional problem!



Computing basis expansion

Compute the basis expansion can be quite complex. Let's imagine you have *p = 2*

$$x = [x_1, x_2]$$

Now you want to transform it with a quadratic (degree-2) polynomial mapping. The explicit transformation might look like:

$$\phi(x) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1]$$

So, we are now dealing with a 6-dimensional problem!

Computing basis expansion

Compute the basis expansion can be quite complex. Let's imagine you have *p = 2*

$$x = [x_1, x_2]$$

Now you want to transform it with a quadratic (degree-2) polynomial mapping. The explicit transformation might look like:

$$\phi(x) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1]$$

So, we are now dealing with a 6-dimensional problem!

'Easy' with 2 features... but imagine you have 1,000 features and want a degree-3 polynomial. You'd end up with millions of dimensions! 😱

Support Vector Machines: the Kernel Trick

If instead of computing the whole function, we exploit the so-called kernel trick, by just computing – through a kernel function K – the dot product between 2 data points. Some examples:

- Linear kernel $K(x, x') = x^\top x'$
- Polynomial kernel $K(x, x') = (\gamma \cdot x^\top x' + r)^d$
- Radial Basis Function (RBF) / Gaussian Kernel $K(x, x') = \exp\left(-\gamma \|x - x'\|^2\right)$
- Hyperbolic Tangent Kernel $K(x, x') = \tanh(\gamma \cdot x^\top x' + r)$

When using the Kernel Trick, we are talking about Support Vector Machines (SVM)ns

Support Vector Machines: the Kernel Trick

If instead of computing the whole function, we exploit the so-called kernel trick, by just computing – through a kernel function K – the dot product between 2 data points. Some examples:

- Linear kernel $K(x, x') = x^\top x'$
- Polynomial kernel $K(x, x') = (\gamma \cdot x^\top x' + r)^d$
- Radial Basis Function (RBF) / Gaussian Kernel $K(x, x') = \exp\left(-\gamma \|x - x'\|^2\right)$
- Hyperbolic Tangent Kernel $K(x, x') = \tanh(\gamma \cdot x^\top x' + r)$

For all these functions:

- If x and x' are very similar, K(x, x') is large

- If they're very different, K(x,x') is small or zero

we are talking about Support Vector Machines (SVM)ns

Support Vector Machines: the Kernel Trick

If instead of computing the whole fur
just computing – through a kernel fur
points. Some examples:

Every kernel function behaves as if it were computing a dot product between vectors φ(x) and φ(x′) in some (possibly infinite) transformed space: $K(x, x') = \langle \phi(x), \phi(x') \rangle$

But you never need to compute φ(x) explicitly — the kernel function handles it for you!

- Linear kernel $K(x, x') = x^\top x'$
- Polynomial kernel $K(x, x') = (\gamma \cdot x^\top x' + r)^d$
- Radial Basis Function (RBF) / Gaussian Kernel $K(x, x') = \exp\left(-\gamma \|x - x'\|^2\right)$
- Hyperbolic Tangent Kernel $K(x, x') = \tanh(\gamma \cdot x^\top x' + r)$

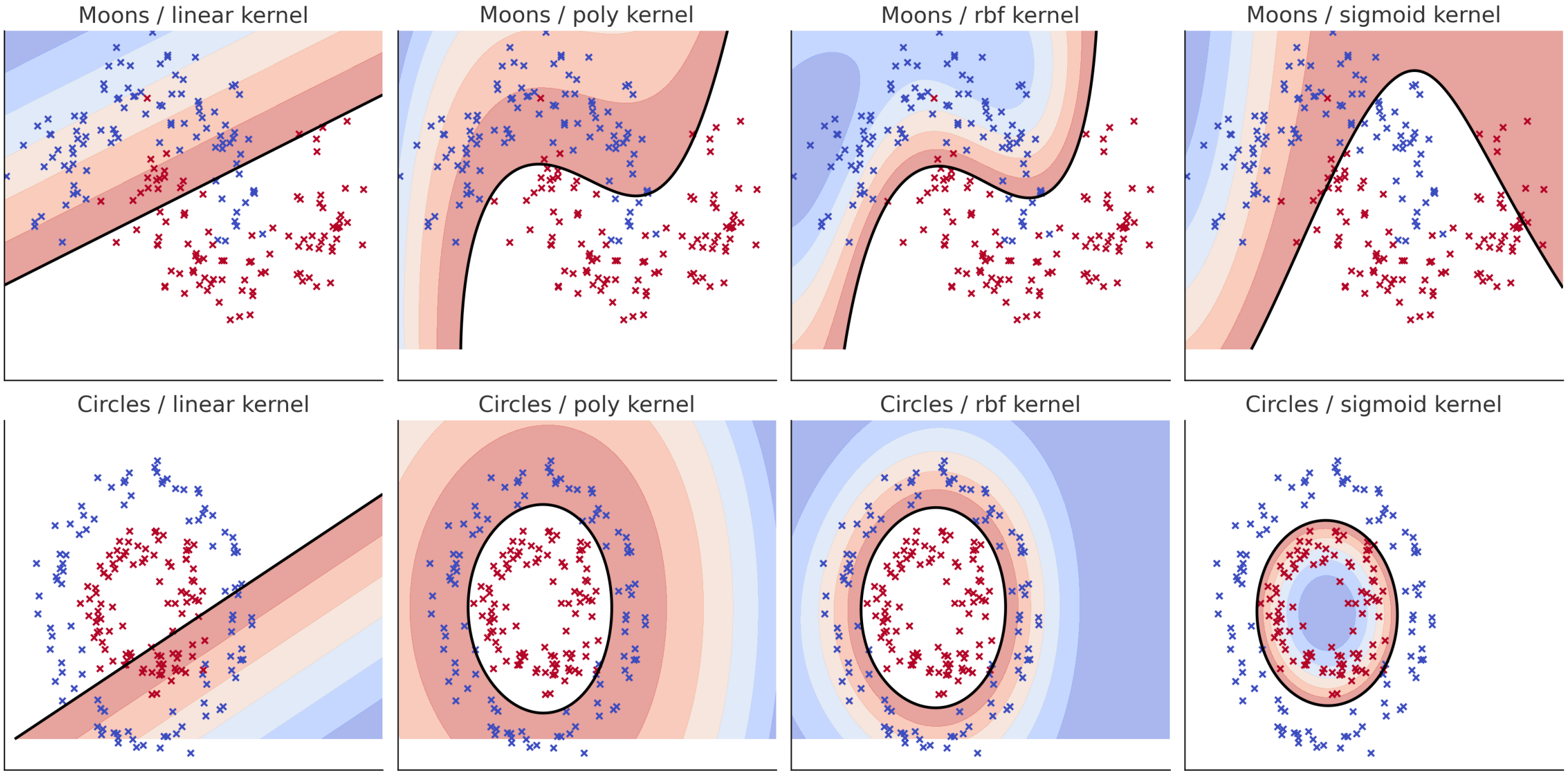
For all these functions:

- If x and x' are very similar, K(x, x') is large

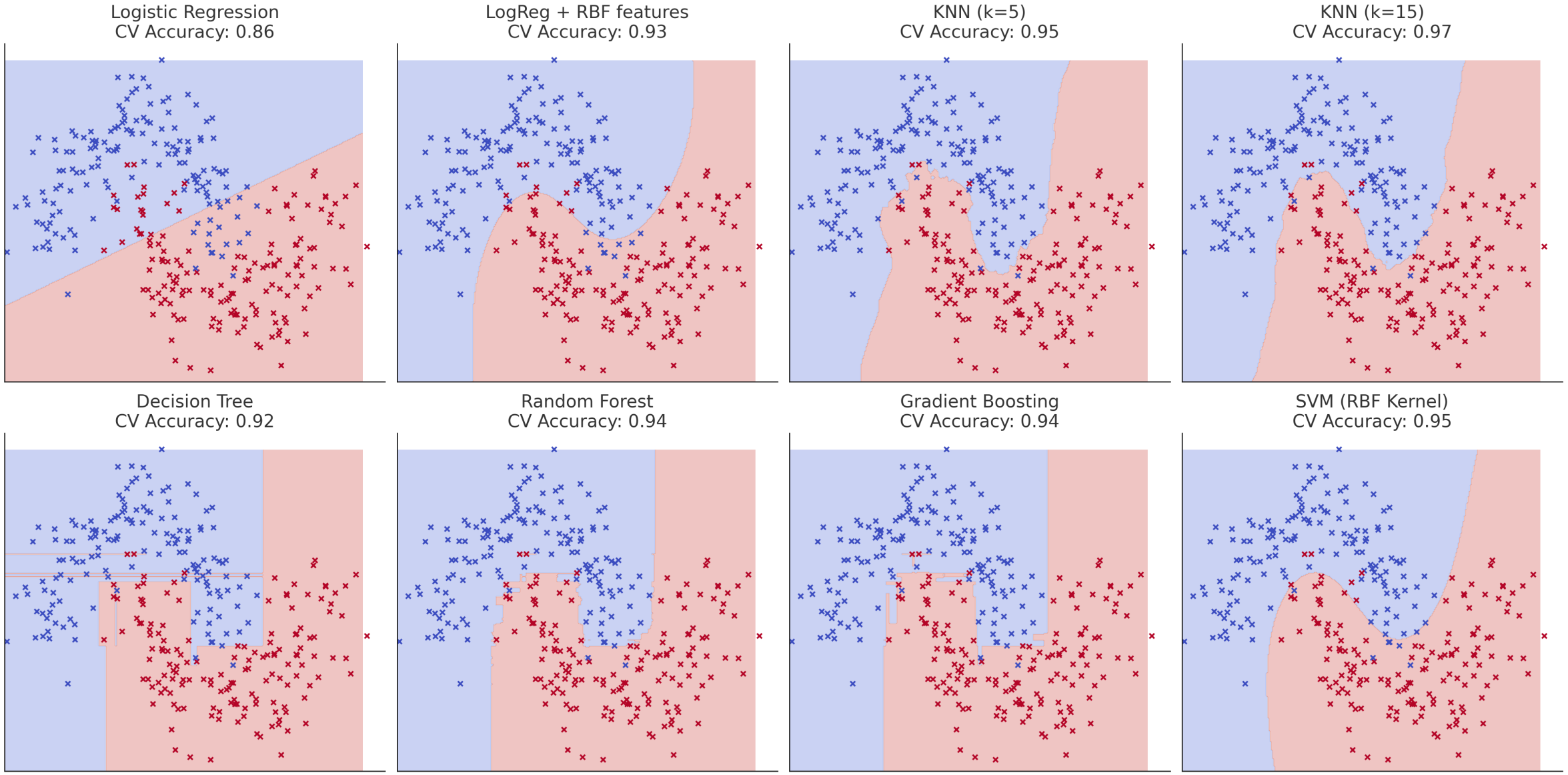
- If they're very different, K(x,x') is small or zero

we are talking about Support Vector Machines (SVM)ns

Kernels comparisons:

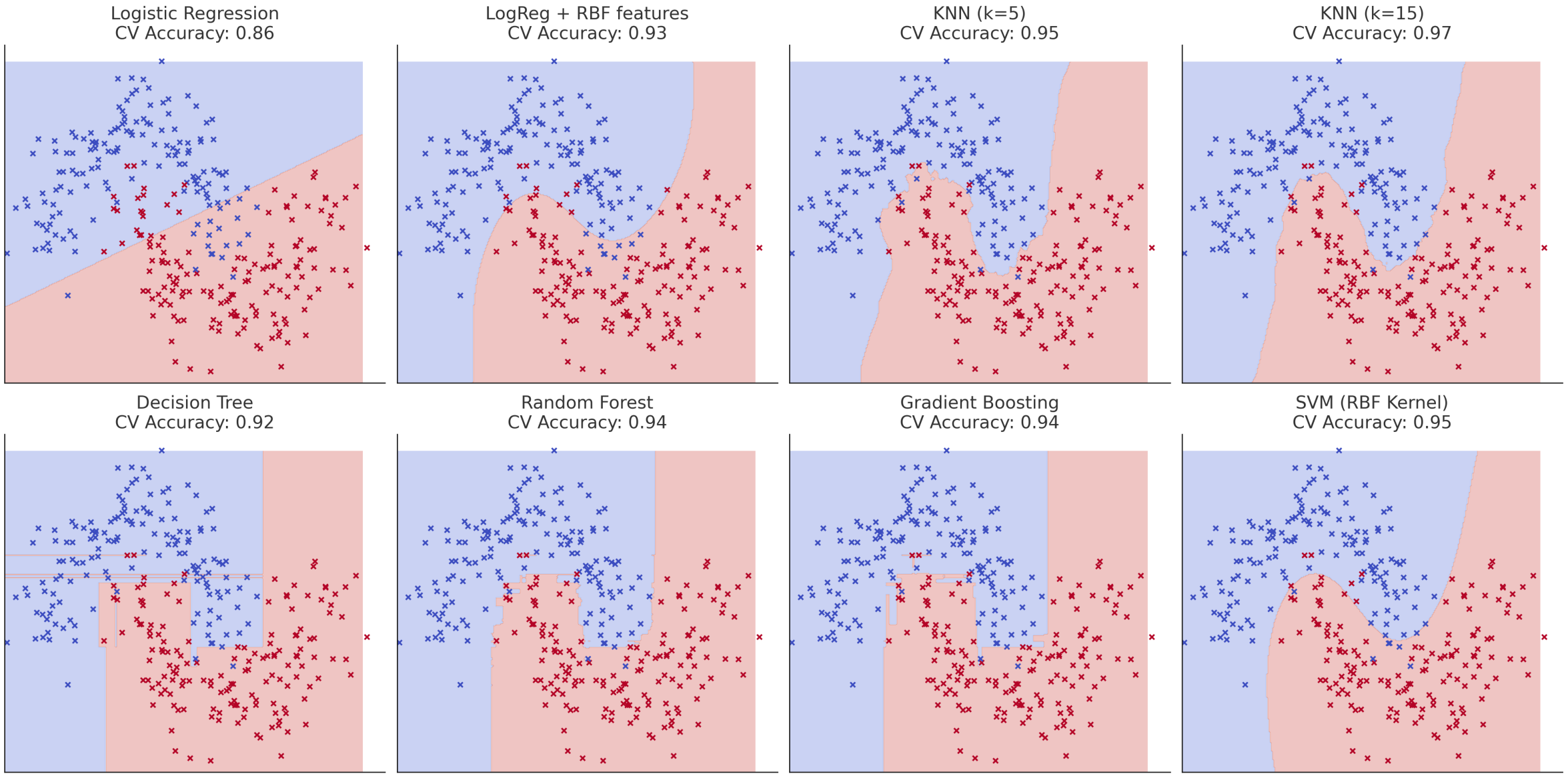


Kernels comparisons – Moons Dataset

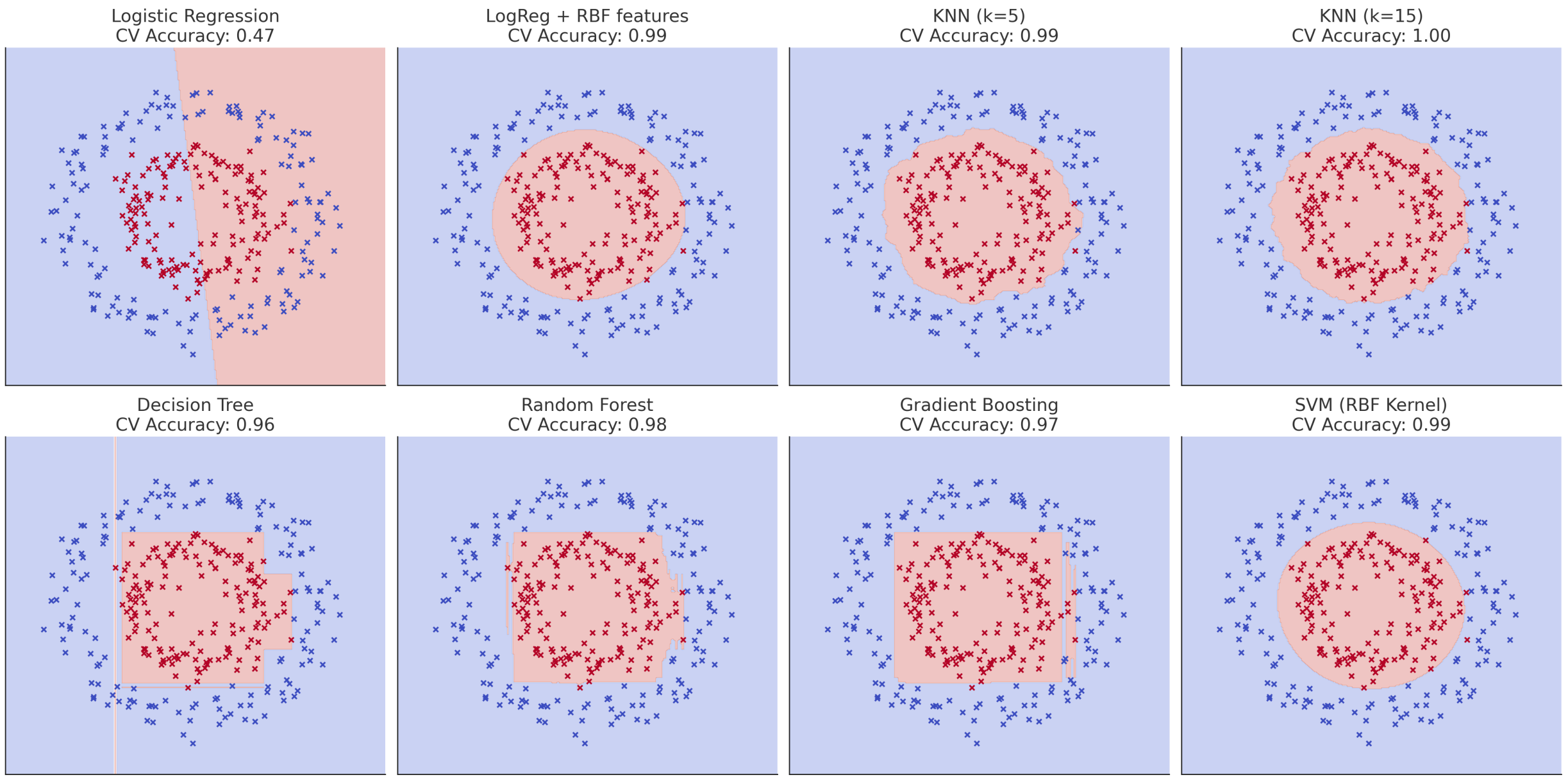


Kernels comparisons – Moon

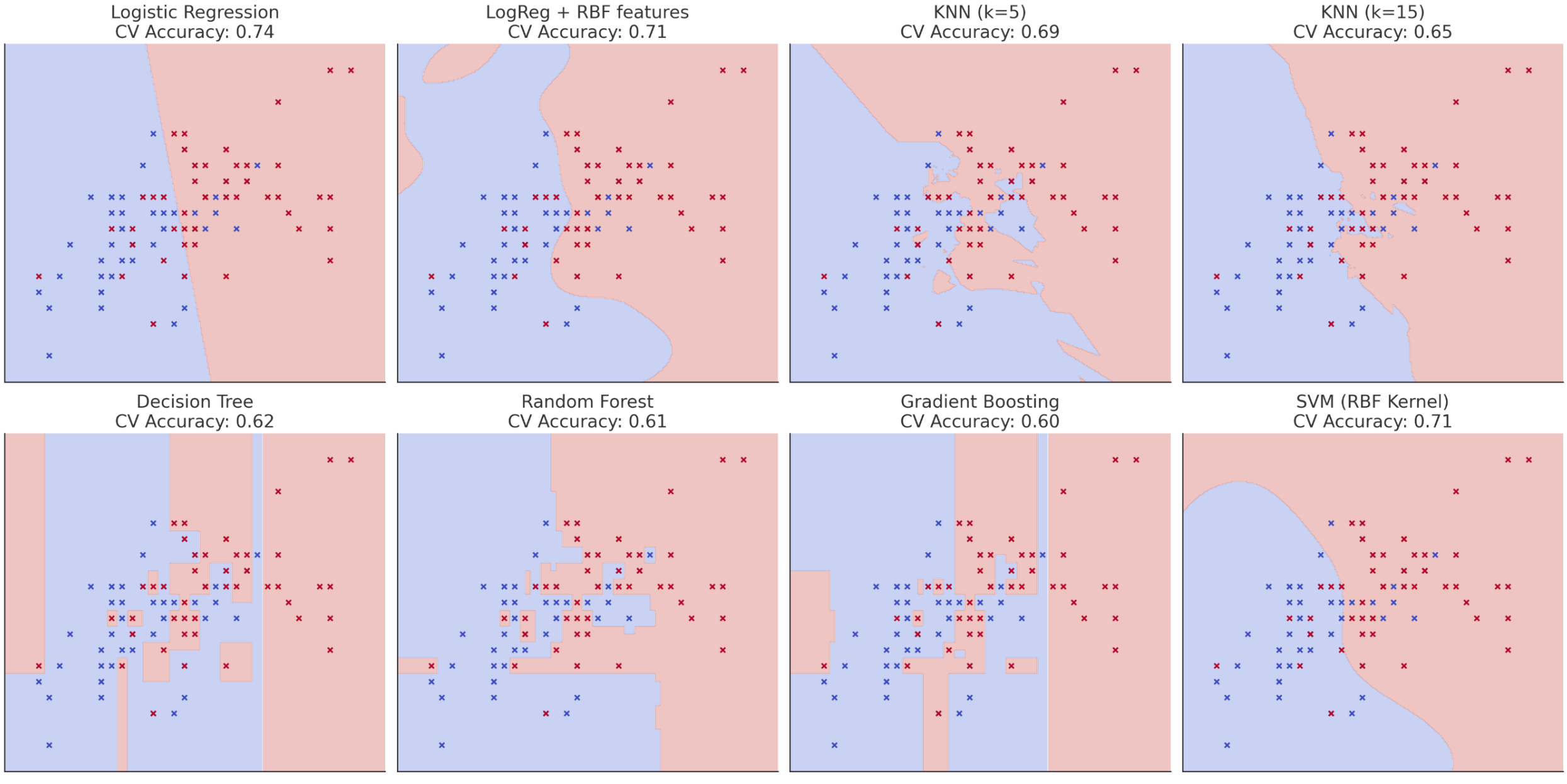
RBF features (in general, non-linear ones) can also be used in logistic regression!



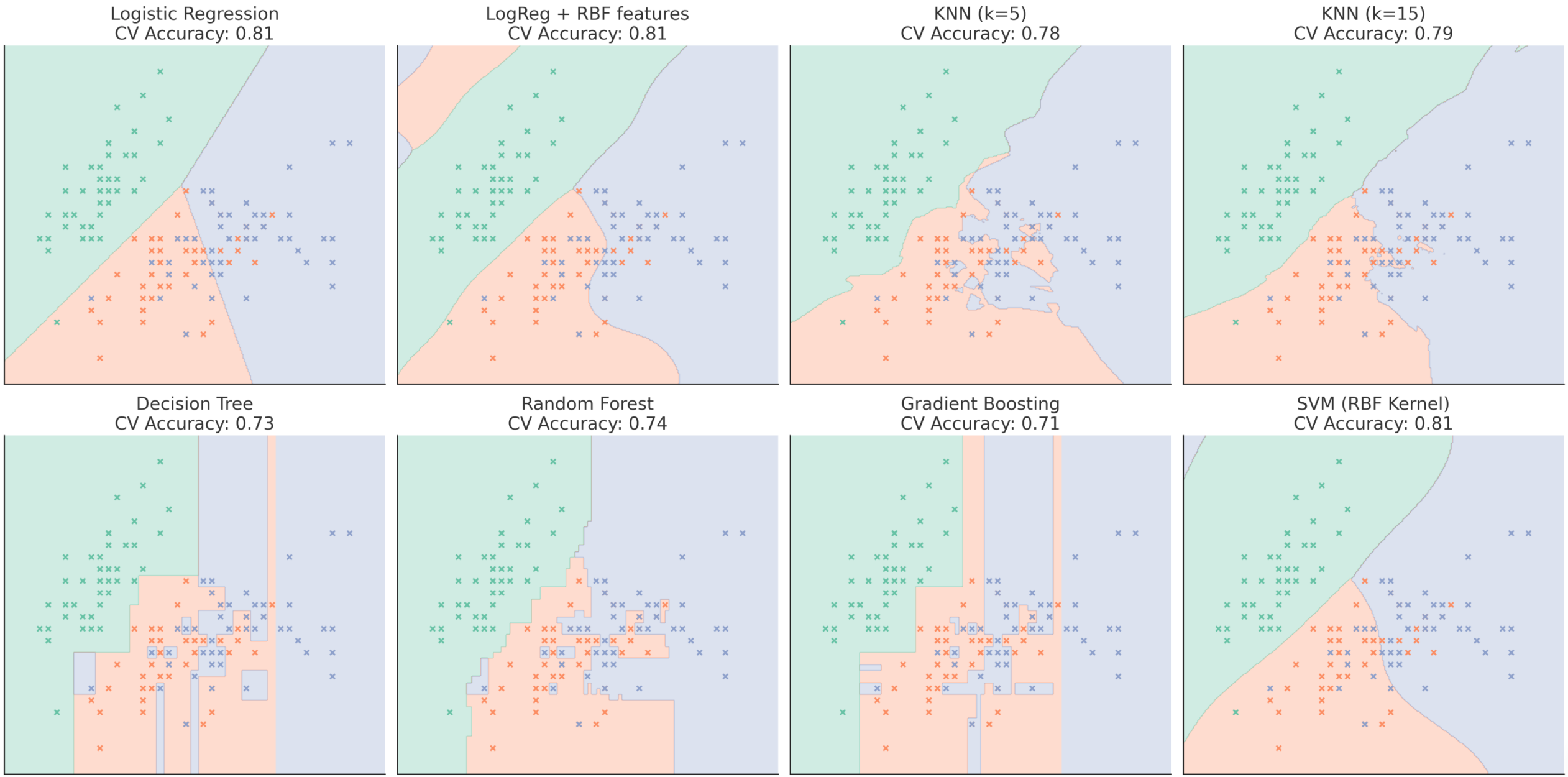
Kernels comparisons – Circle Dataset



Kernels comparisons – Iris Dataset (binary)



Kernels comparisons – Iris Dataset

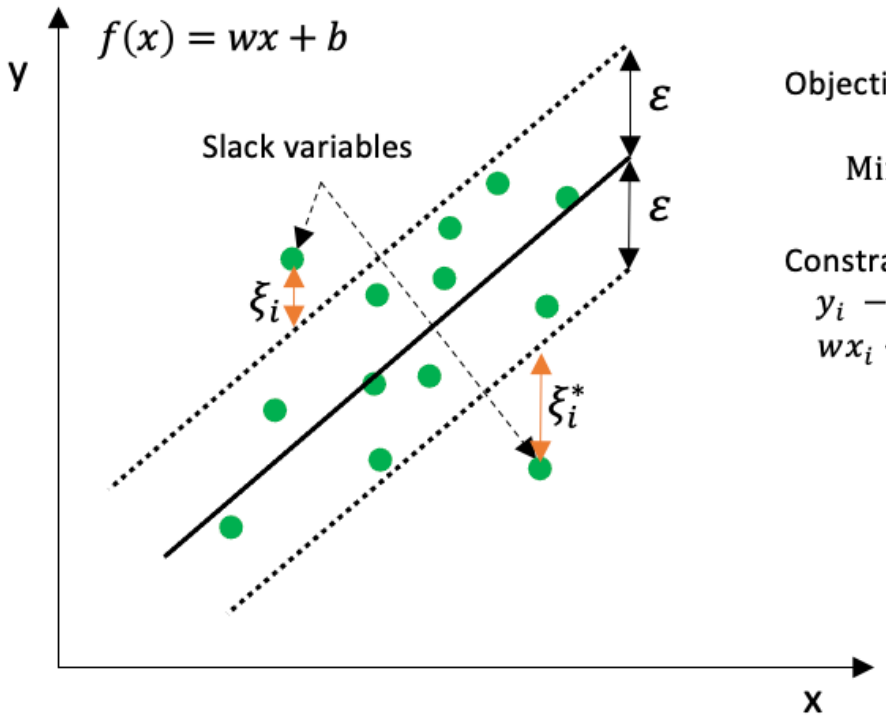


Support Vector Regression (SVR)

SVR tries to find a function (like a line or curve) that:

- Stays as close as possible to the data points
- Ignores small errors within a certain margin (called epsilon)
- Keeps the model as flat (simple) as possible, like in SVM classification.

So instead of separating classes with a wide margin, SVR fits a line (or surface) with a tube of tolerance around it.



Objective:

$$\text{Minimize: } \frac{1}{2}||w||^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$

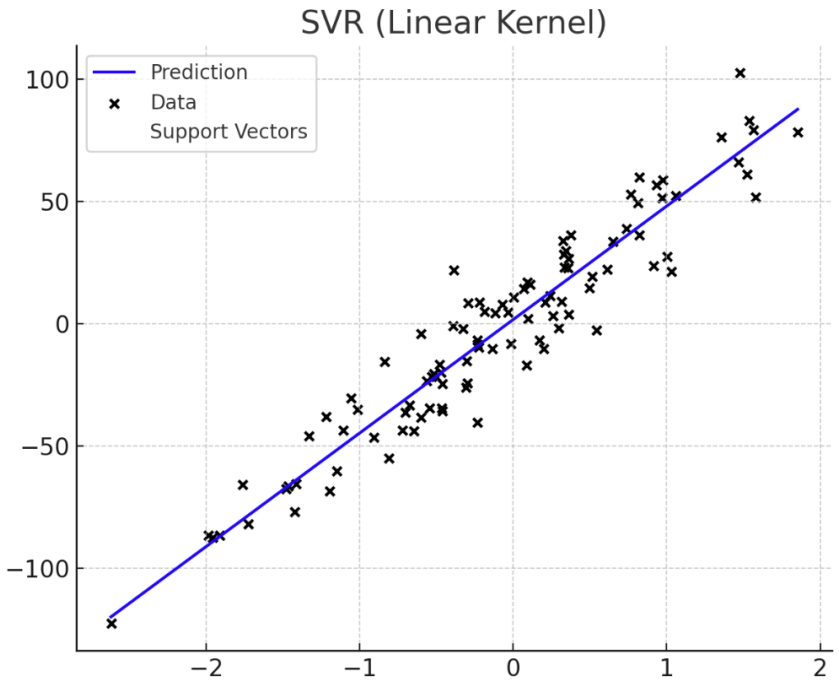
Constraints:

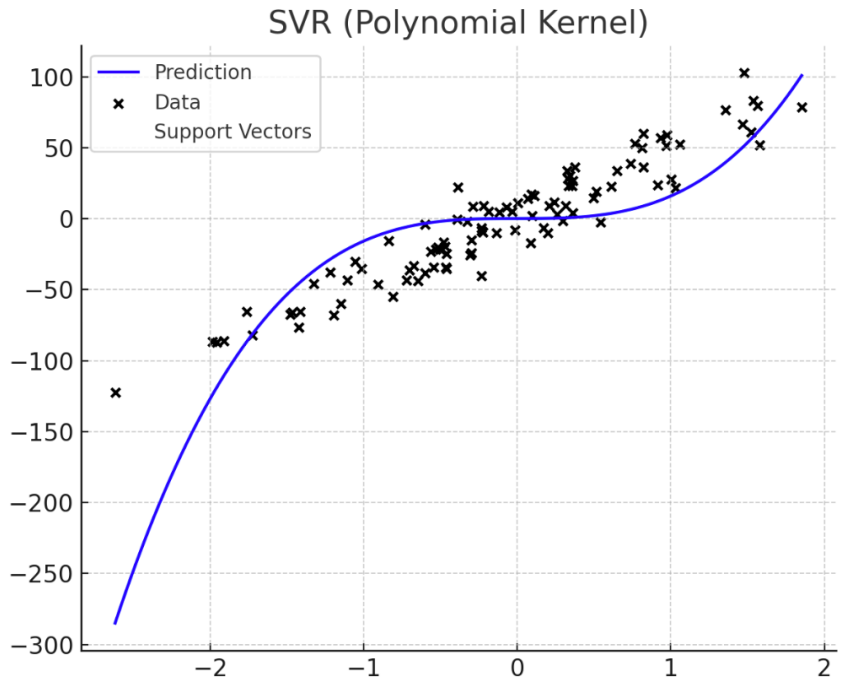
$$y_i - wx_i - b \leq \varepsilon + \xi_i$$
$$wx_i + b - y_i \leq \varepsilon + \xi_i^*$$
$$\xi_i^*, \xi_i \geq 0$$

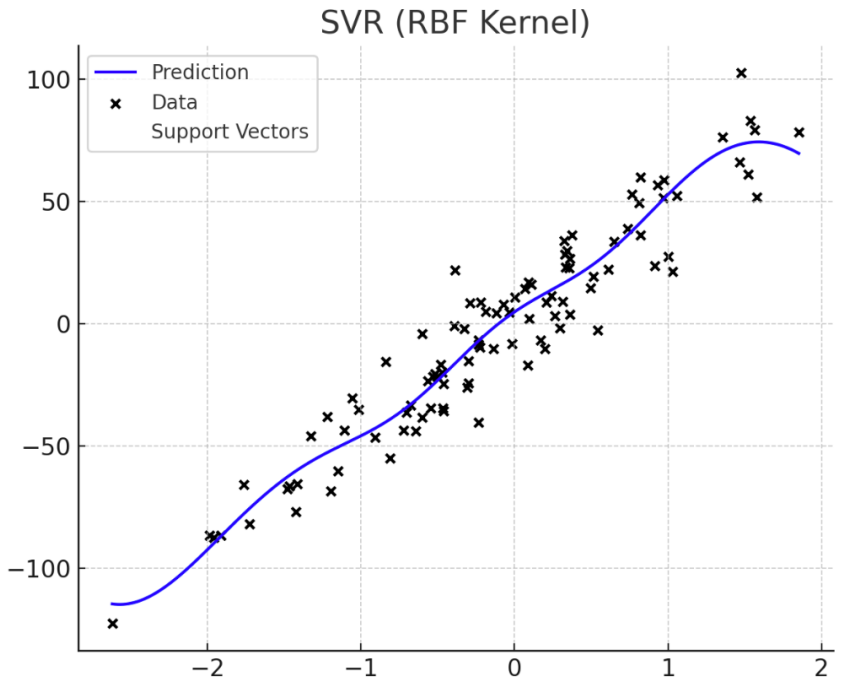
SVR tries to fit a line (or curve) through the data with a "margin of tolerance" (epsilon), and only penalizes the points that fall outside that margin.

Think of it as: a "soft-fit" regression line that balances accuracy with simplicity.

Support Vector Regression (SVR)

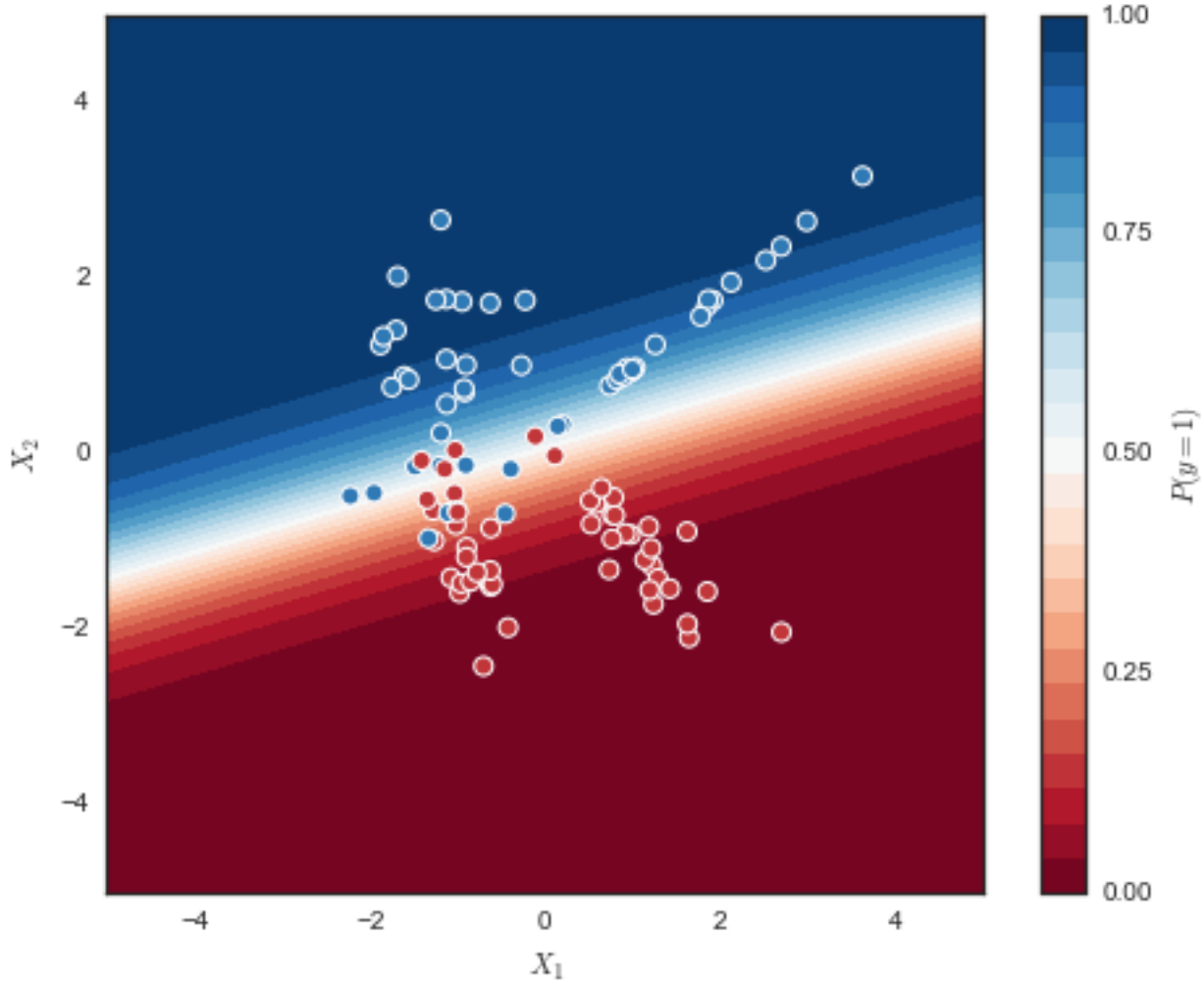






Back on binary classification…

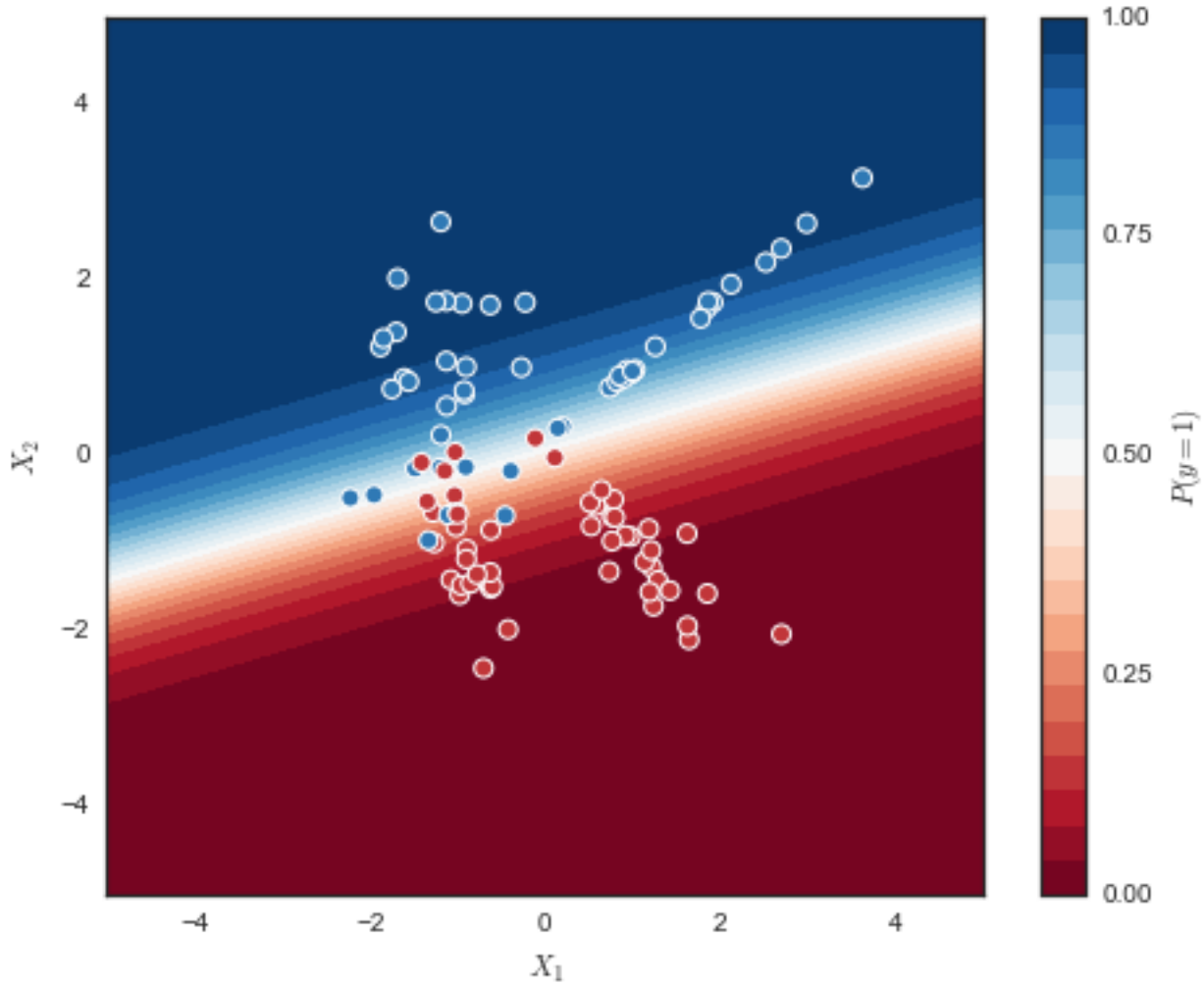
In many approaches (logistic regression, SVM, …) we can derive a distance from the decision boundary, a probability of being classified to one class or another…



Back on binary classification…

In many approaches (logistic regression, SVM, …) we can derive a distance from the decision boundary, a probability of being classified to one class or another…

But errors are not the same! We may prefer being wrong on one class instead of another one (false positive may be more acceptable than false negatives and vice versa)!

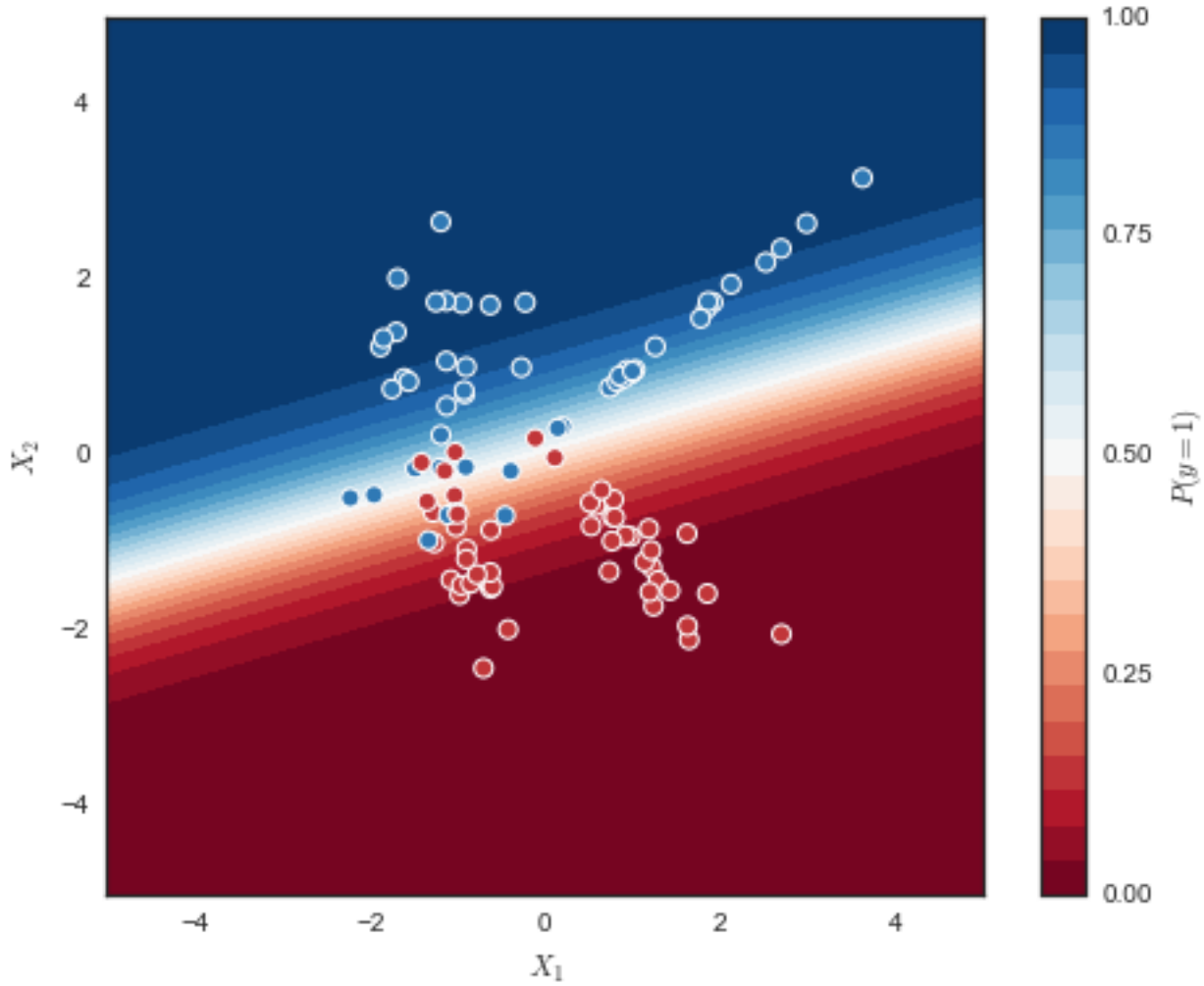


Back on binary classification…

In many approaches (logistic regression, SVM, …) we can derive a distance from the decision boundary, a probability of being classified to one class or another…

But errors are not the same! We may prefer being wrong on one class instead of another one (false positive may be more acceptable than false negatives and vice versa)!

How can we cope with this?



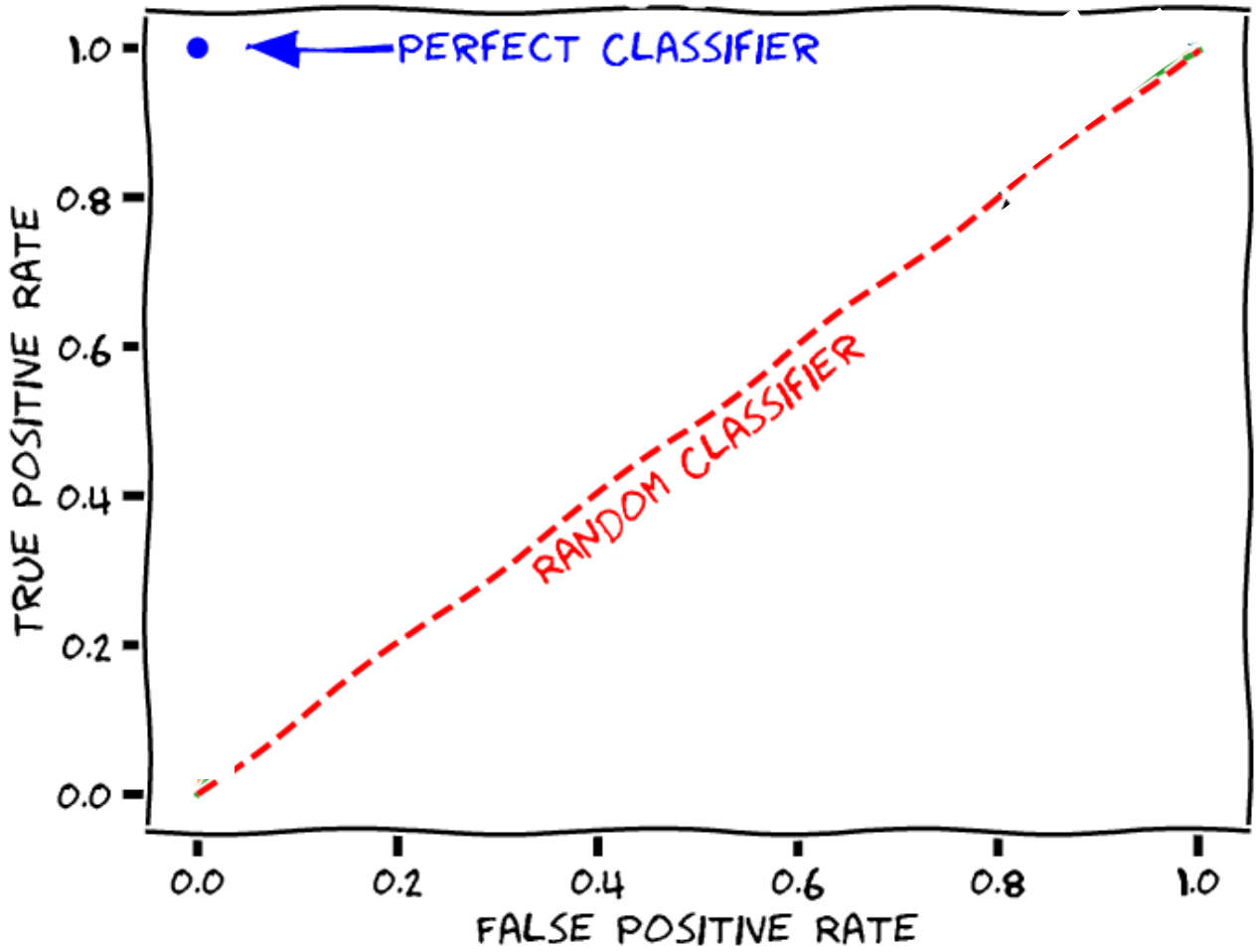
Back on binary classification…

By tuning the decision making on the probability/distance/etc… we can obtain different performances!

Let’s consider this plot with:

- True Positive Rate
- False Positive Rate

These two metrics are in contrast: when a classifier improves in one, it lowers its performances in the other one!



Back on binary classification…

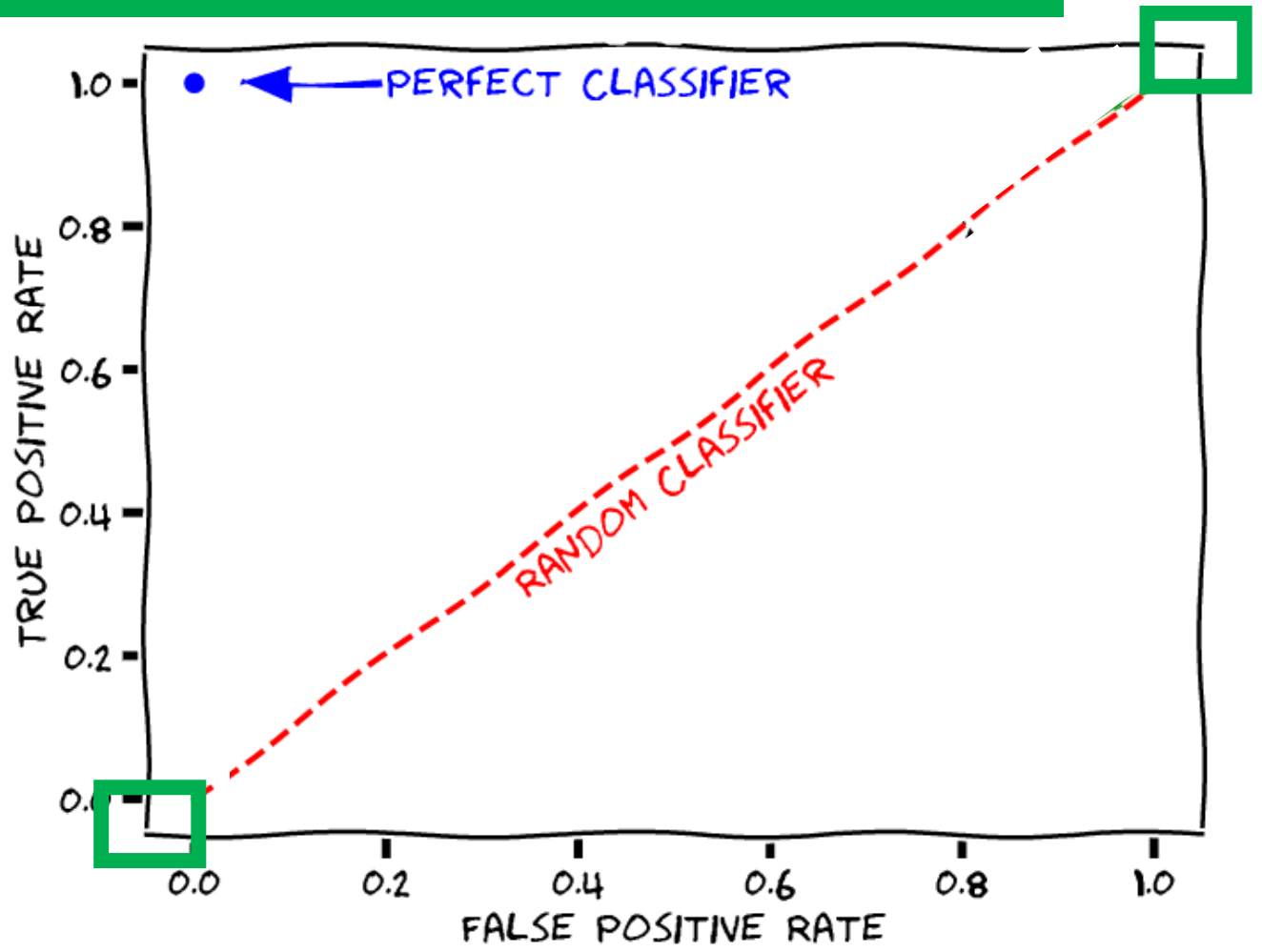
By tuning the decision making on the probability/distance/etc… we can obtain different performances!

Let’s consider this plot with:

- True Positive Rate
- False Positive Rate

These two metrics are in contrast: when a classifier improves in one, it lowers its performances in the other one!

Classifiers that only predict one class



The Receiver Operating Characteristic (ROC) Curve

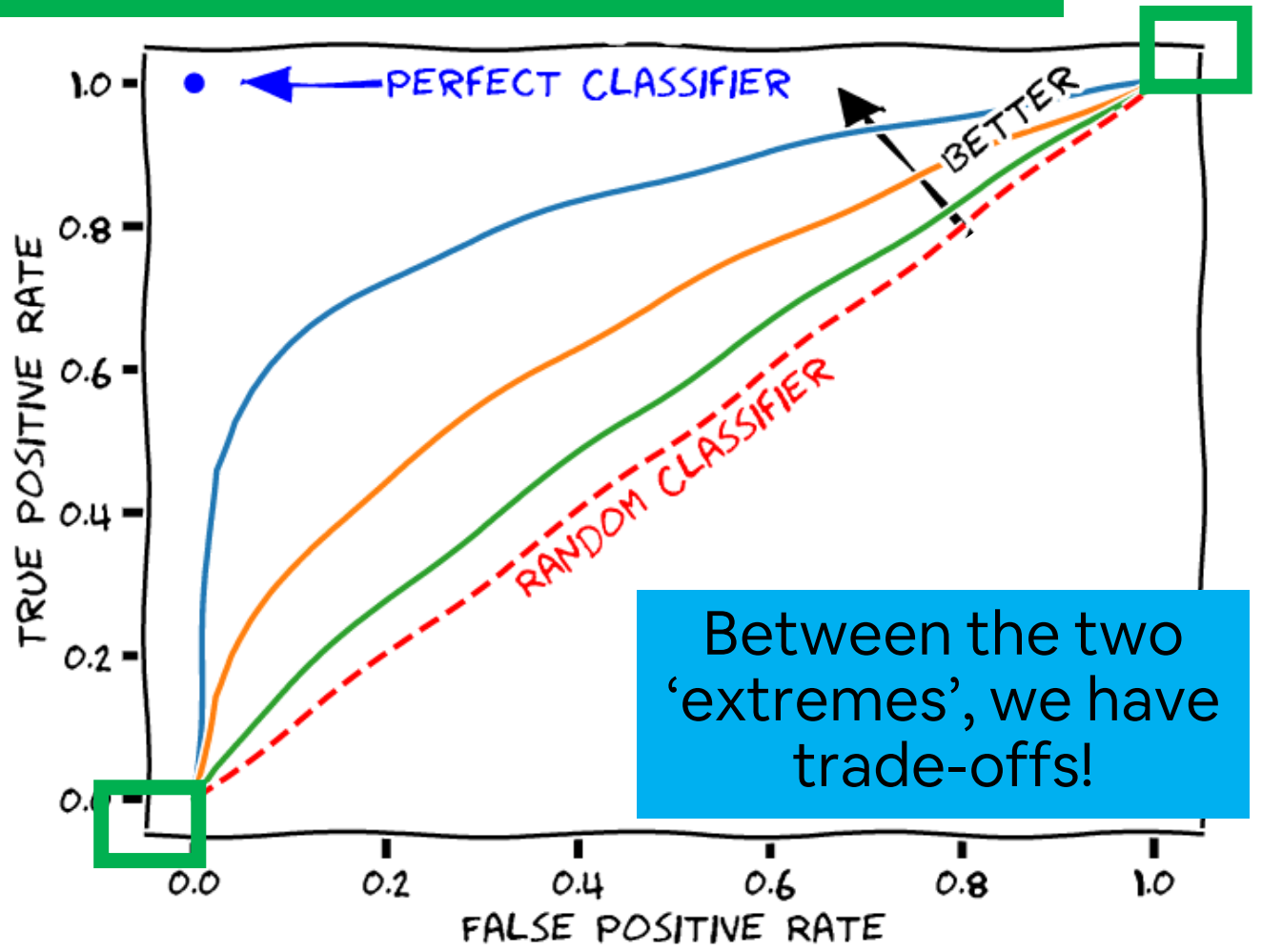
By tuning the decision making on the probability/distance/etc… we can obtain different performances!

Let’s consider this plot with:

- True Positive Rate
- False Positive Rate

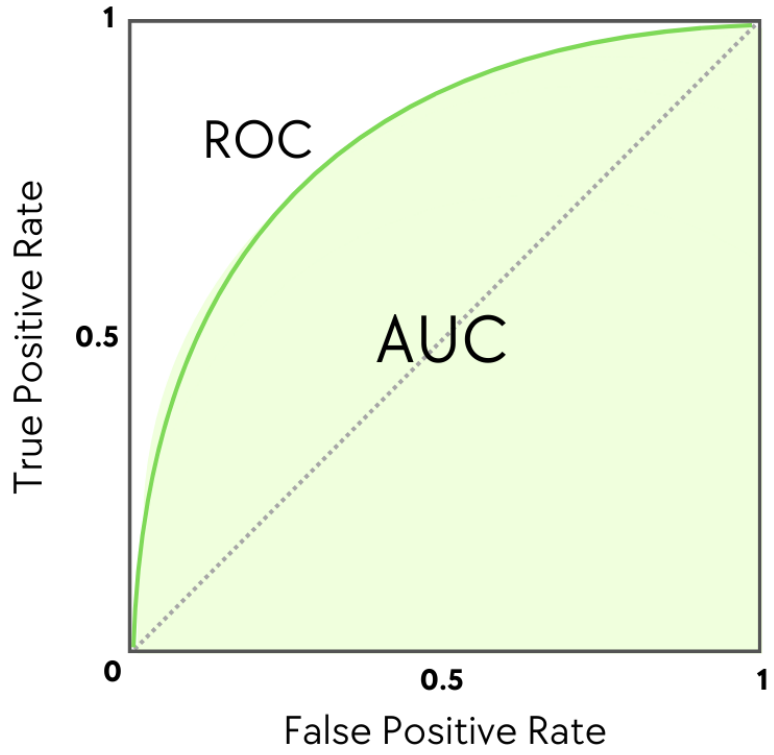
These two metrics are in contrast: when a classifier improves in one, it lowers its performances in the other one!

Classifiers that only predict one class



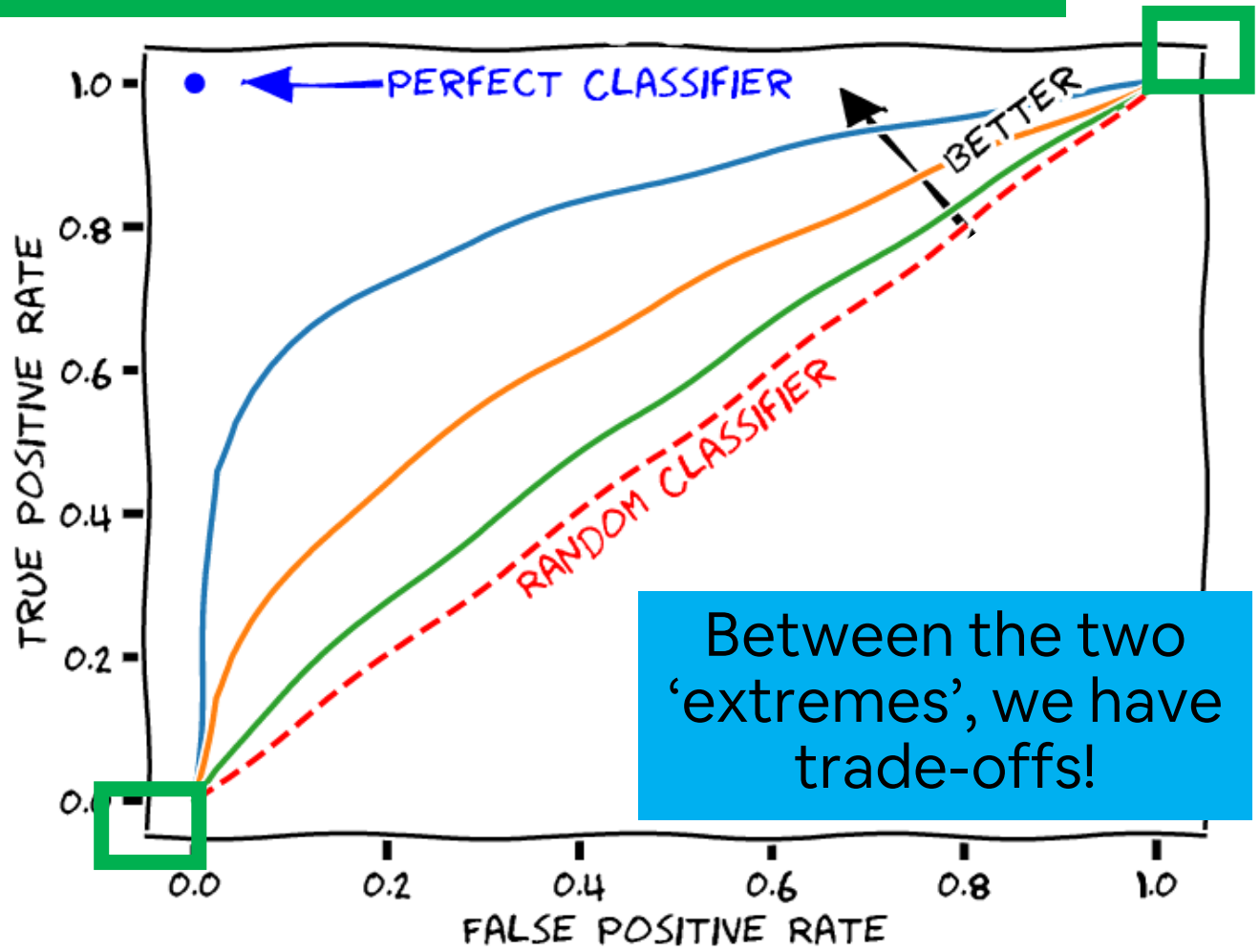
Between the two ‘extremes’, we have trade-offs!

The Receiver Operating Characteristic (ROC) Curve



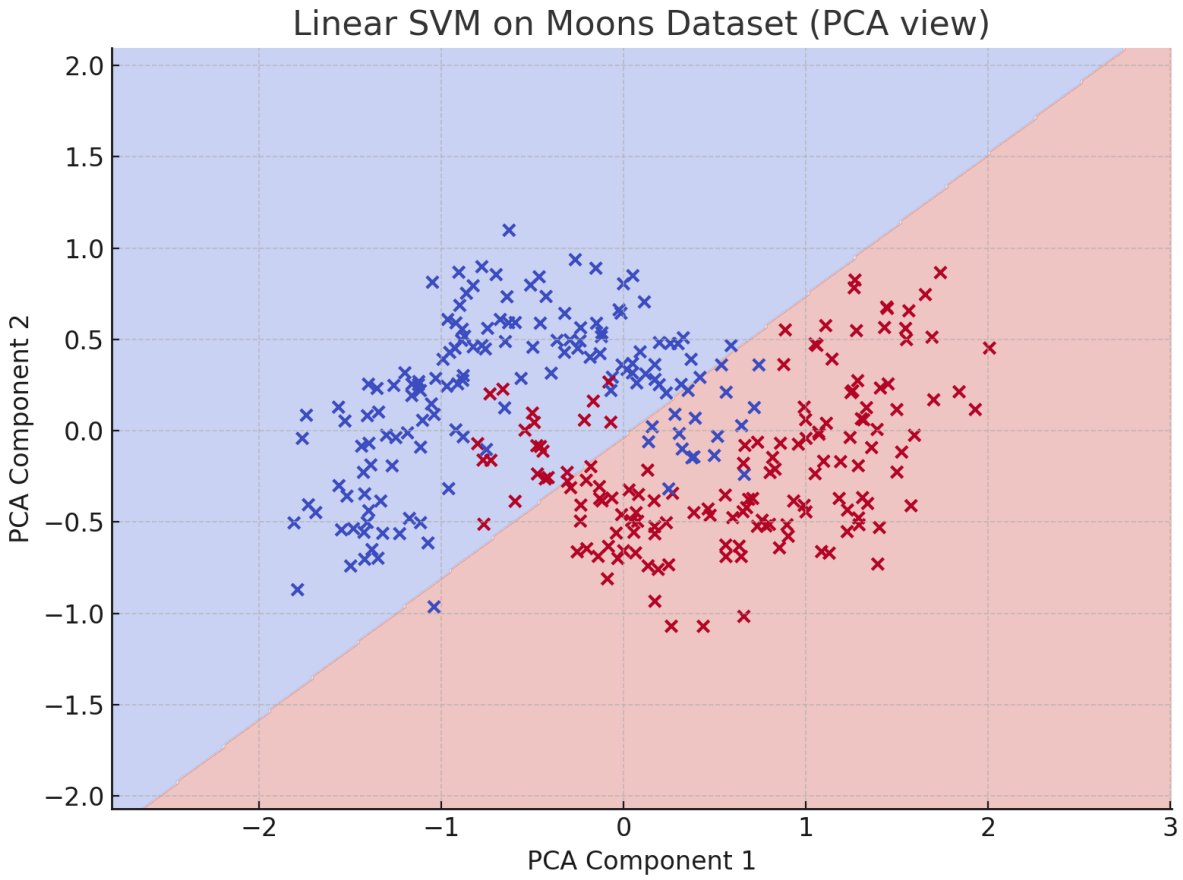
A related quantity is the Area Under the Curve (AUC): the integral of the area under the ROC

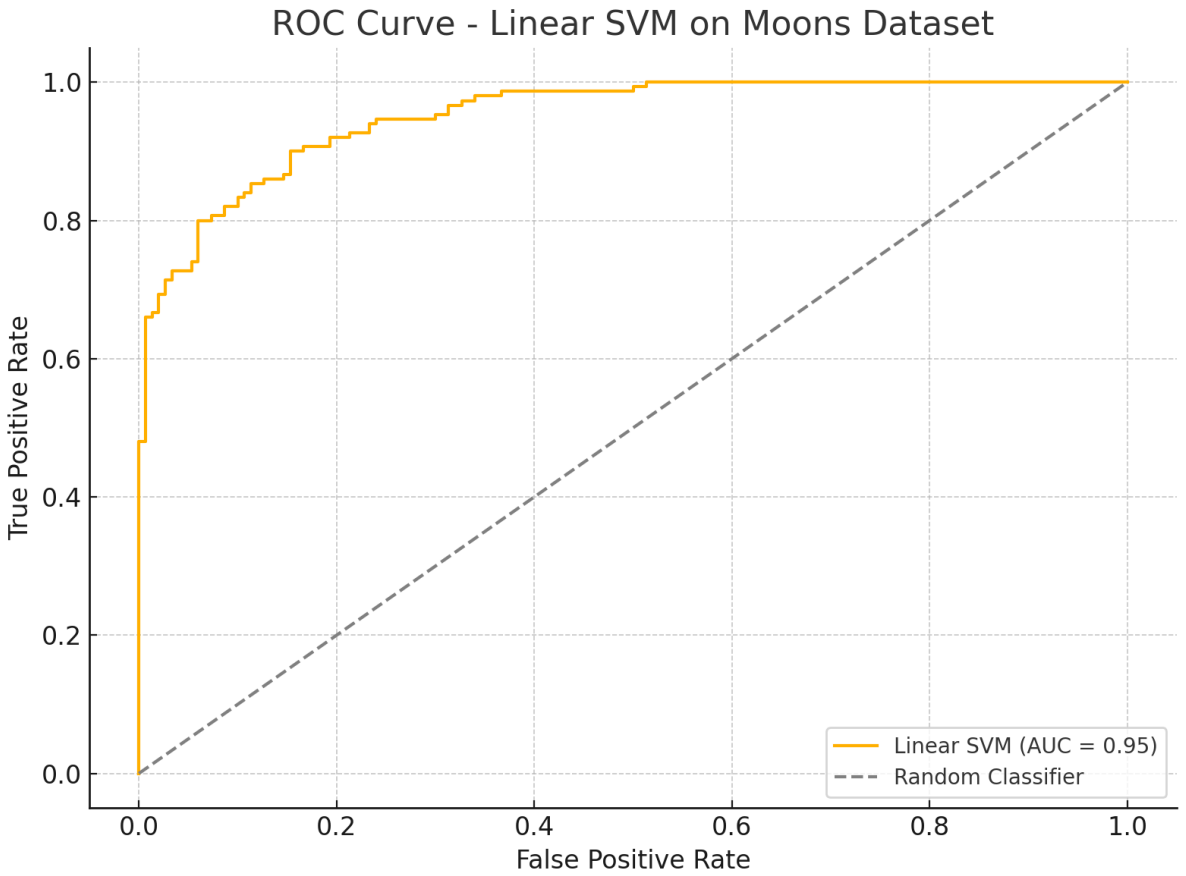
Classifiers that only predict one class



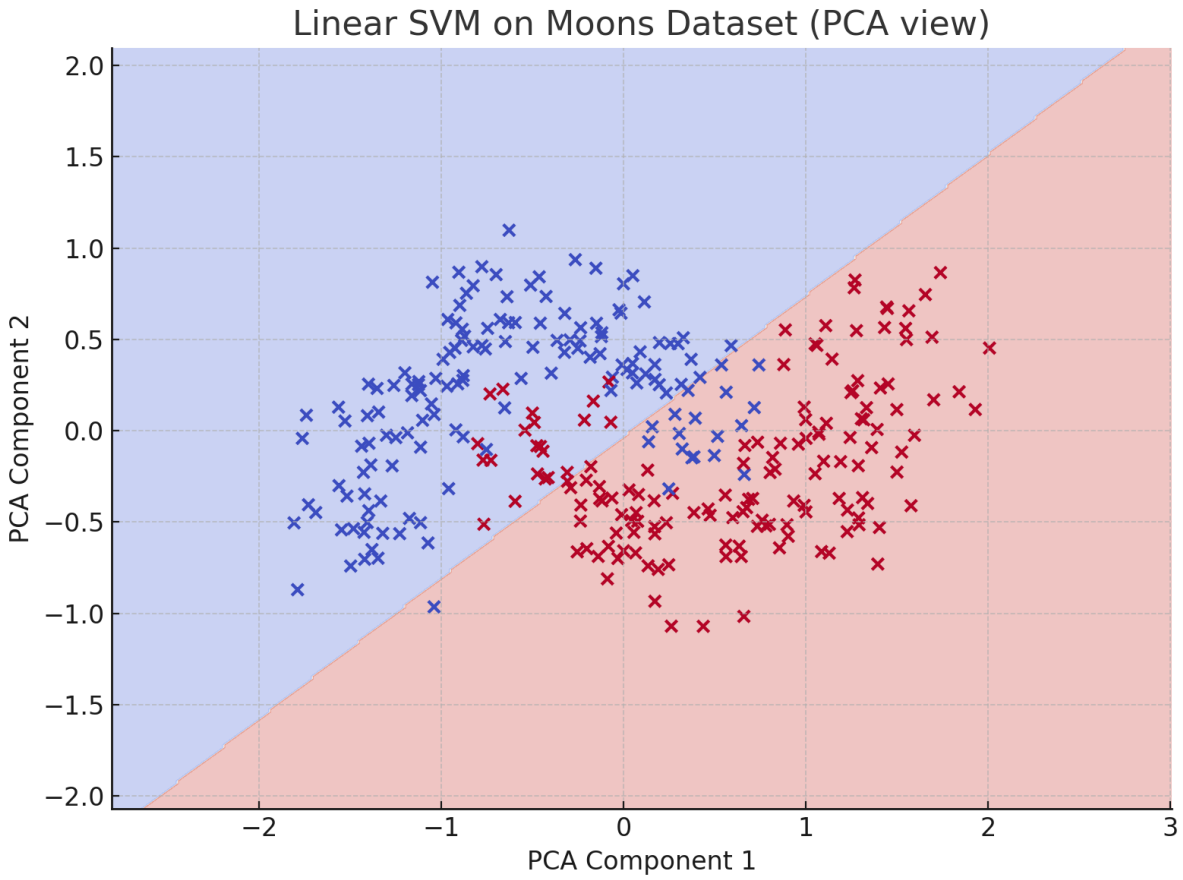
Between the two 'extremes', we have trade-offs!

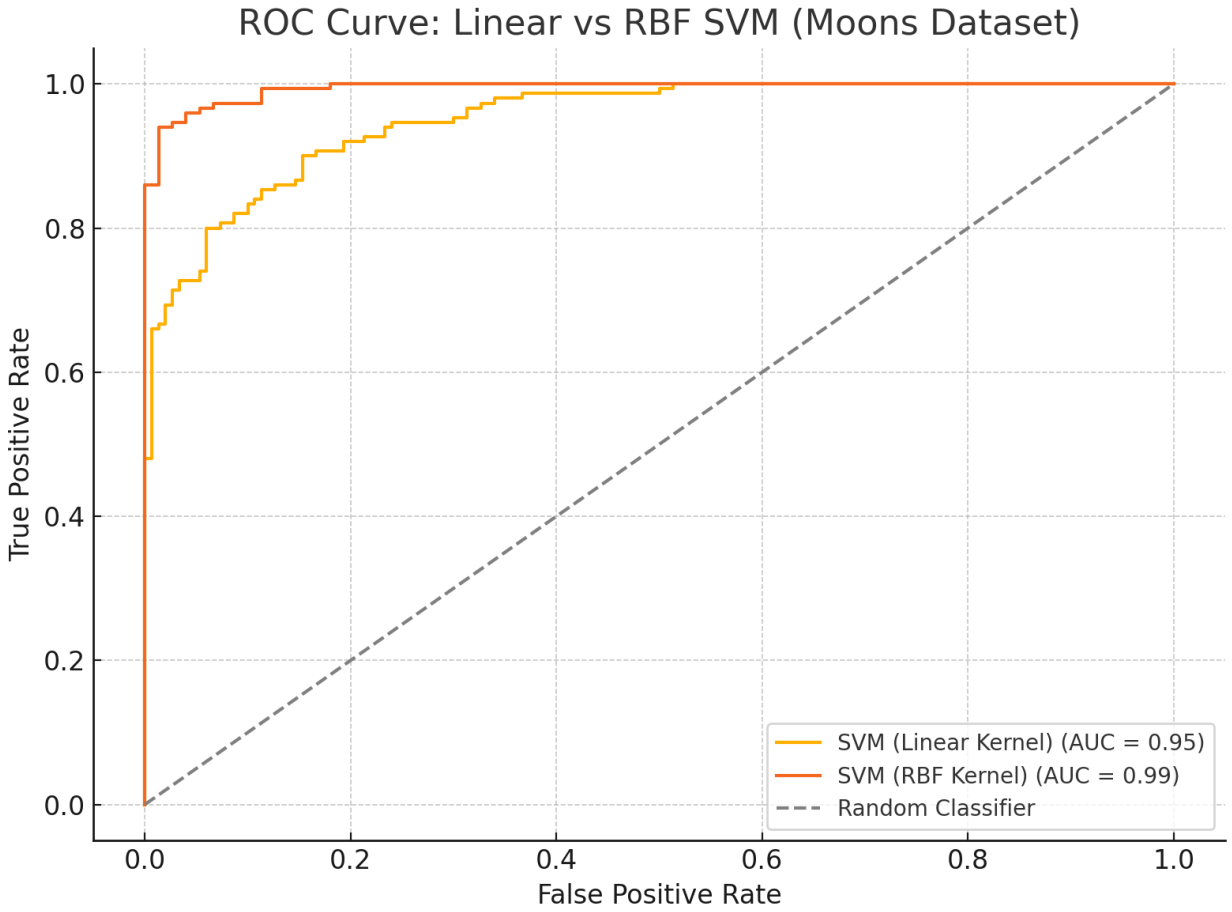
On the Moons Dataset





On the Moons Dataset





Thank you!

Gian Antonio Susto