Rensselaer

why not change the world?®

Support Vector Machines (SVM) for classification

Ahmed Eleish

Data Analytics ITWS-4600/6600 CSCI-4960 MGMT 4600/6600

October 27th 2026

Tetherless World Constellation
Rensselaer Polytechnic Institute

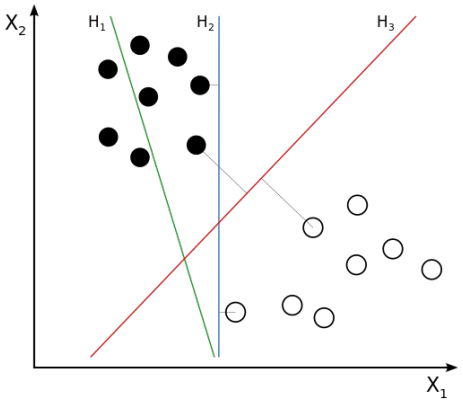


Support Vector Machines

- Rationale
- Hyperplanes, Margins and Support vectors
- Classification using SVM
- Linear Separability of classes (or not)
- Soft Margin SVM
- Kernels

Rationale

- If data points in *p*-dimensional space, belonging to 2 different classes can be separated by a (*p*-1)-dimensional hyperplane, this hyperplane can be used as a linear classifier.
- Example: in 2d space, a line could be linear classifier..
- The hyperplane representing the largest separation or “margin” between the classes maximizes the distance to the nearest data point from each class.
- SVM utilizes the maximum margin hyperplane to solve classification, regression and outlier detection problems.

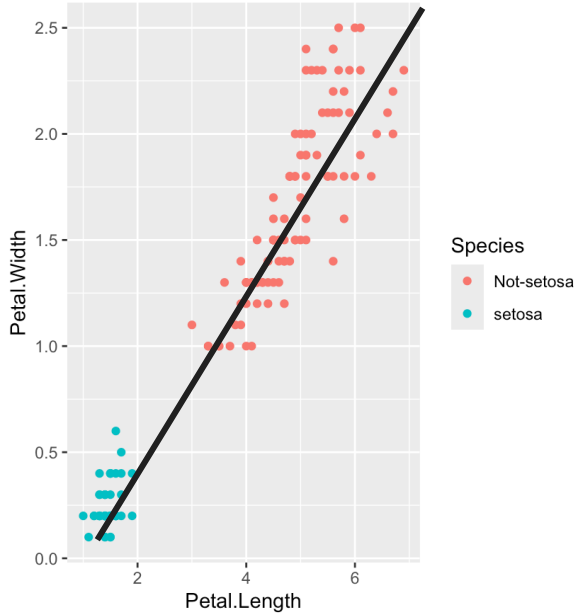


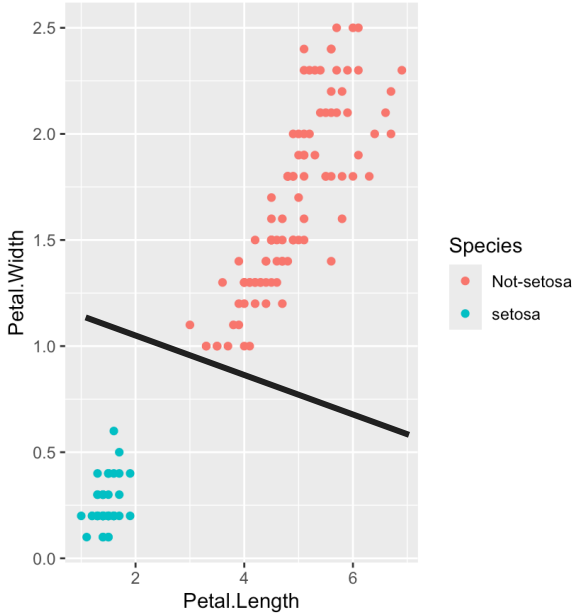
Hyperplane

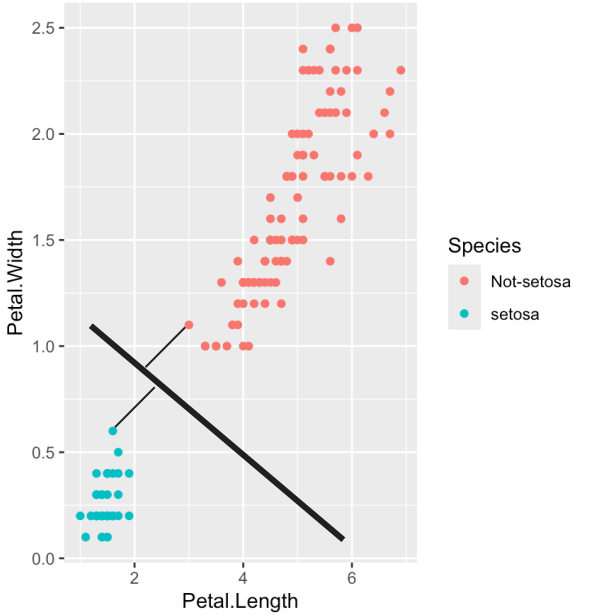
- A hyperplane is a plane of dimension *p-1* in a *p* dimensional space
- “a flat hypersurface, a subspace whose dimension is one less than that of the ambient space”
- “any codimension - 1 vector subspace of a vector space”

https://en.wikipedia.org/wiki/Hyperplane
https://mathworld.wolfram.com/Hyperplane.html

Hyperplanes







Margin

- The distance between the hyperplane (decision boundary) and the nearest points from each class.
- larger margin = greater confidence in the classifier
- SVMs find the hyperplane that maximizes the margin
    - “maximum-margin classifiers”

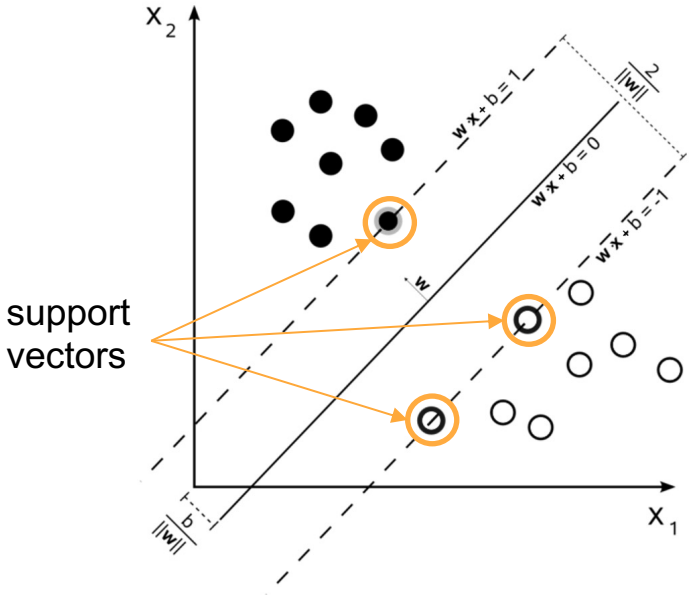
Support Vectors

- The points closest to the decision boundary.
- They determine the position and orientation of the hyperplane, i.e. define the decision boundary.
- They are used to calculate the margin.

Hyperplane:

$$w^T x + b = 0$$

w: weight vector (hyperplane orientation)
b: bias (shifts the hyperplane)



Support Vector Machines

- Given training dataset of points (x_i, y_i) where $\boldsymbol{y_i}$ is equal to **1** or **-1**

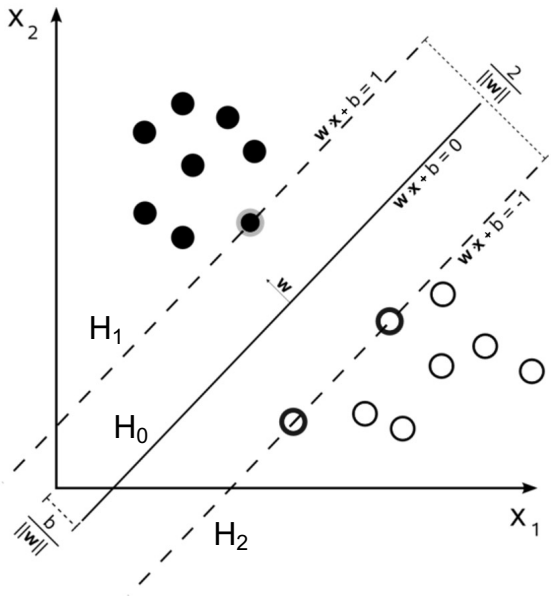
* Find the maximum-margin-hyperplane that divides the points x_i for which $y_i = 1$ from the points for which $y_i = -1$

Hyperplane:

$$w^T x + b = 0$$

Maximize the distance between *H1* and *H2*

where $$w = \sum_{i=1}^{n} \alpha_i \, y_i x_i$$



Support Vector Machines

- Maximize the distance between *H1* and *H2*
- For point x_0, y_0 and line $ax + by + c = 0$ the perpendicular distance from the point to the line is:

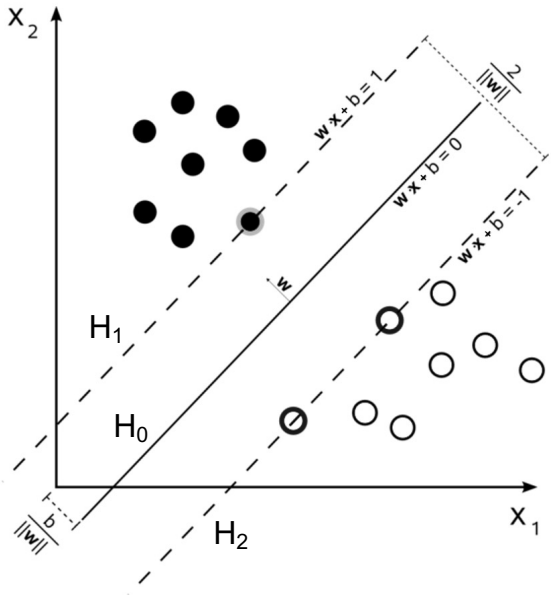
$$d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

- For hyperplane $w^\top x + b = 0$:

$$d = \frac{|w^\top x_0 + b|}{\|w\|}$$

- If x_0 is on the margin $w^\top x + b = \pm 1$, then the distance from it to the decision boundary is:

$$\frac{1}{\|w\|}$$



Support Vector Machines

- The total distance between *H1* and *H2*:

$$\frac{2}{\|w\|}$$

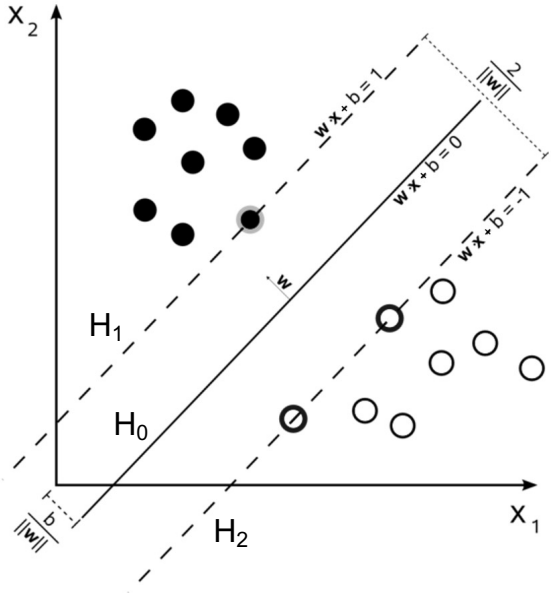
To find *w* and *b*:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

subject to

$$y_i(w^T x_i + b) \geq 1 \; \forall i \in \{1, \ldots, n\}$$

* The optimization problem is solved using gradient descent, quadratic programming, etc.



Gradient Descent Optimization

- The **gradient** of a function points in the direction of its **steepest increase**.
- Gradient descent minimizes the function e.g. $\boldsymbol{f(\theta_t)}$ by repeatedly taking a step in the opposite direction:

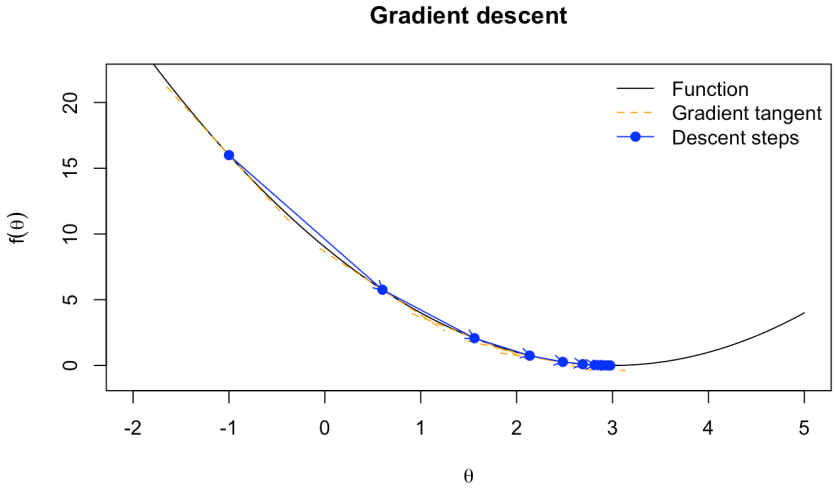
$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t)$$

where θ_t is the current value of the parameter, $\nabla f(\theta_t)$ is the function gradient at that point, and $\eta > 0$ is the *learning rate*

- Steps:
  - Start with an initial parameter value.
  - Evaluate the gradient at that value.
  - Step in the negative-gradient direction.
  - Repeat until the values stop changing much or a stopping limit is reached.

Gradient Descent example

- Function: $f(\theta) = (\theta - 3)^2$
- Function derivative: $f'(\theta) = 2(\theta - 3)$
- Step update: $\theta_{t+1} = \theta_t - \eta 2(\theta_t - 3)$



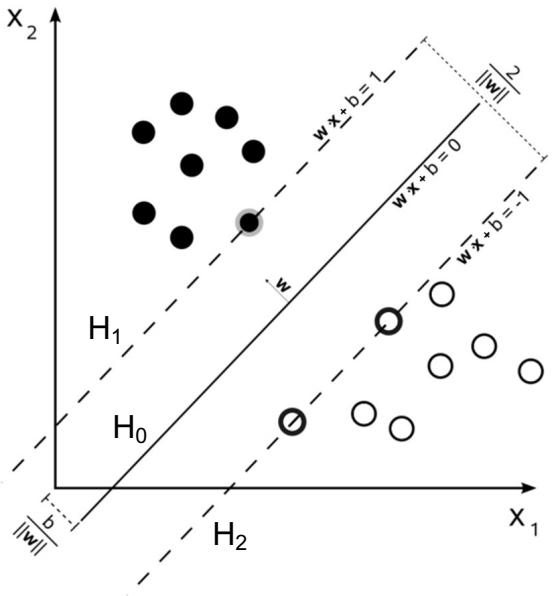
Classification with SVM

- Once the ***w*** and ***b*** are found, a point *x* is classified by finding on which side of the boundary it lies.
- First the SVM computes a score:

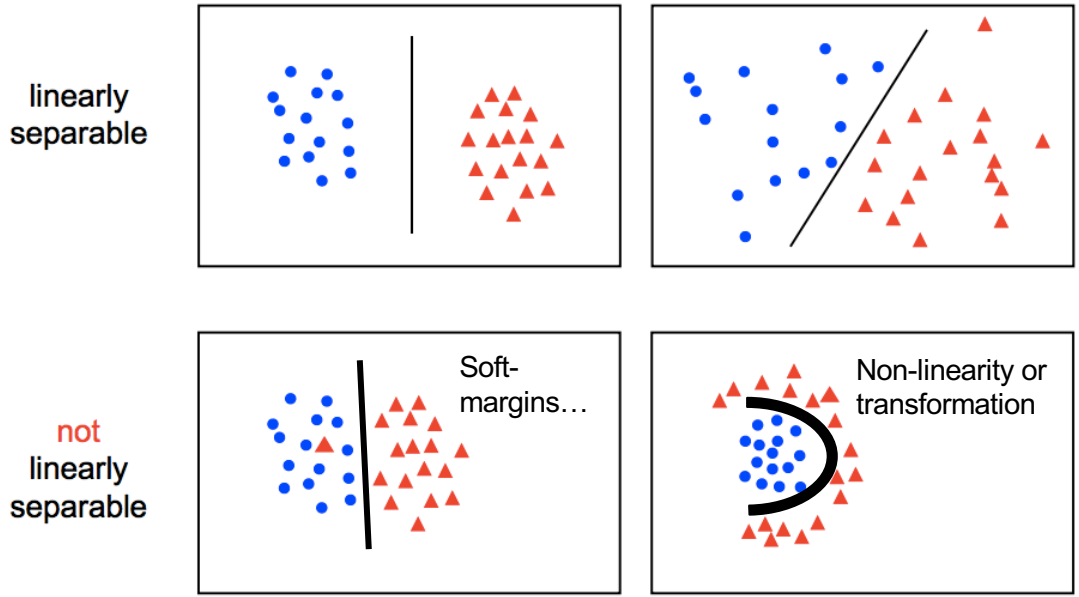
$$\mathrm{f}(x) = w^{\top} x_i + b$$

- Then the sign of the score determines orientation from the boundary:

$$\hat{y} = \mathrm{sign}\big(\mathrm{f}(x)\big) = \begin{cases} +1, & \mathrm{f}(x) > 0, \\ -1, & \mathrm{f}(x) < 0. \end{cases}$$



Linear Separability



Soft-margin SVM

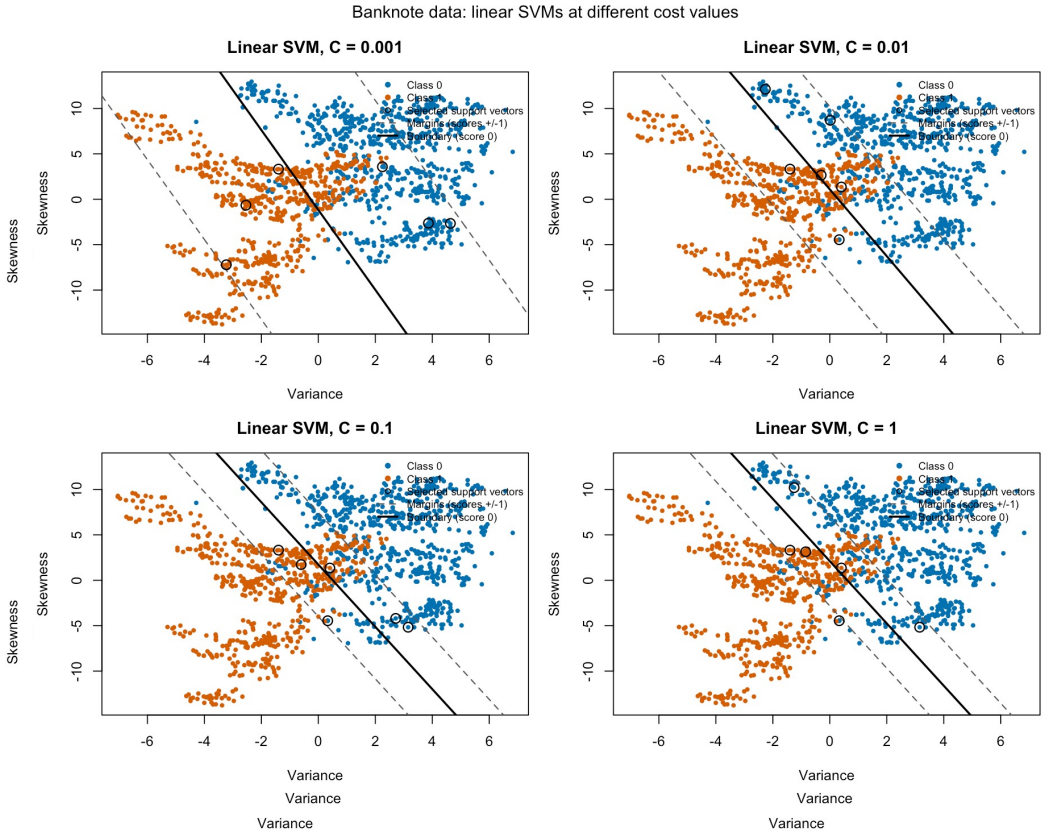
Allow for some margin violations controlled by the parameter *C*, the *regularization parameter*

$$\min_{w,b,z} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{l} z_i$$

Controls margin wideness

subject to

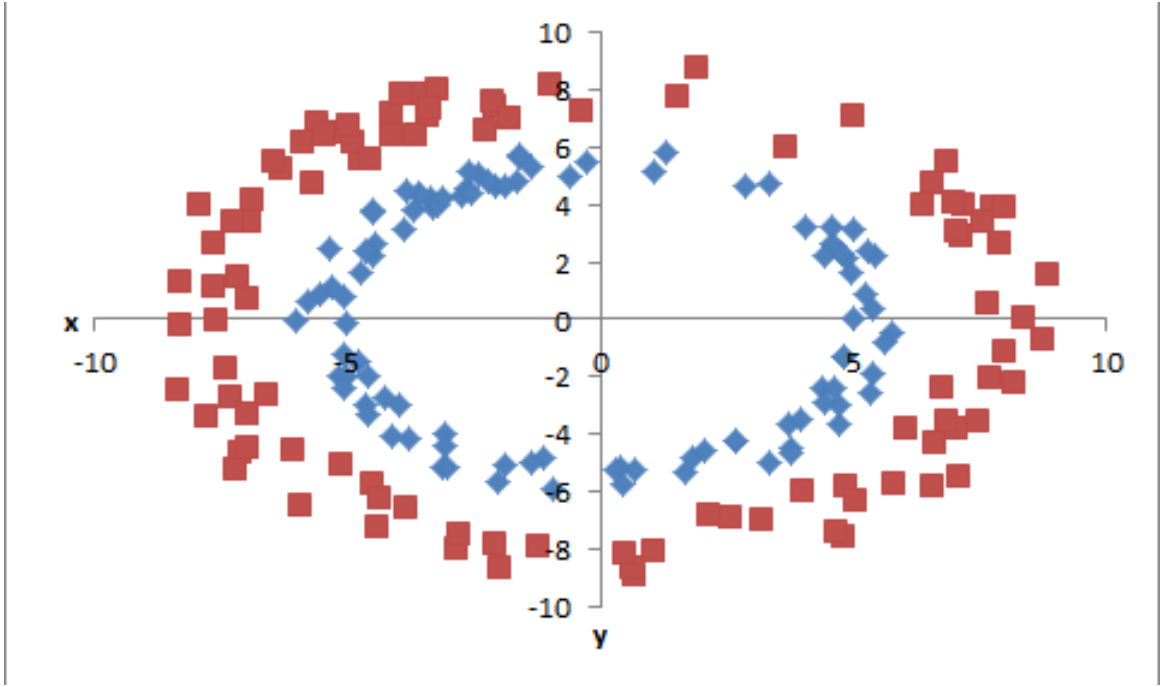
$$yi(w^\top x_i + b) \geq 1 - z_i, \qquad z_i \geq 0.$$



Top left: Highest C value, decreasing C narrows the margin

Non-linearity

What to do??



Non-linearity (ideal example)

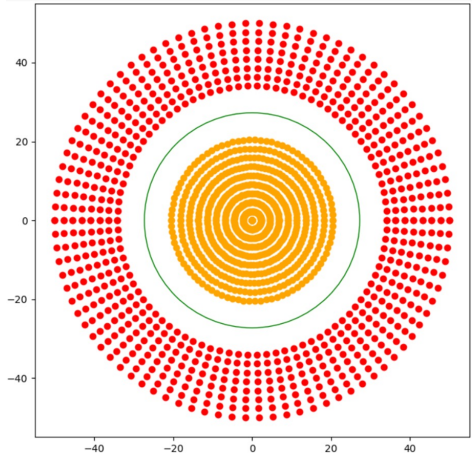
Transform the input:

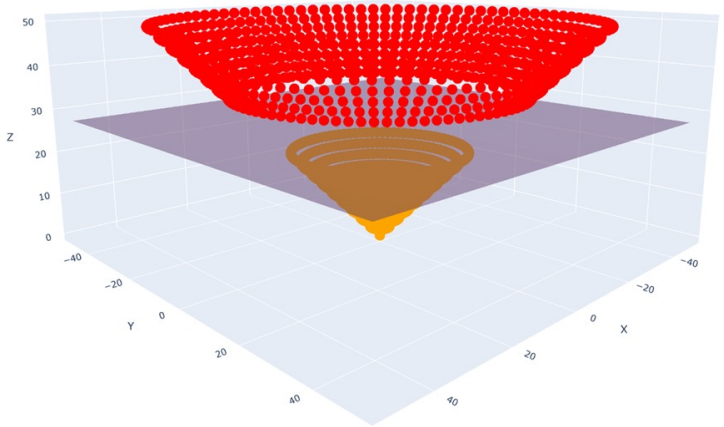
- Add a new dimension where the data are linearly separable

If are dataset contains variables X1, X2:
we can add X3 = f(X1,X2)

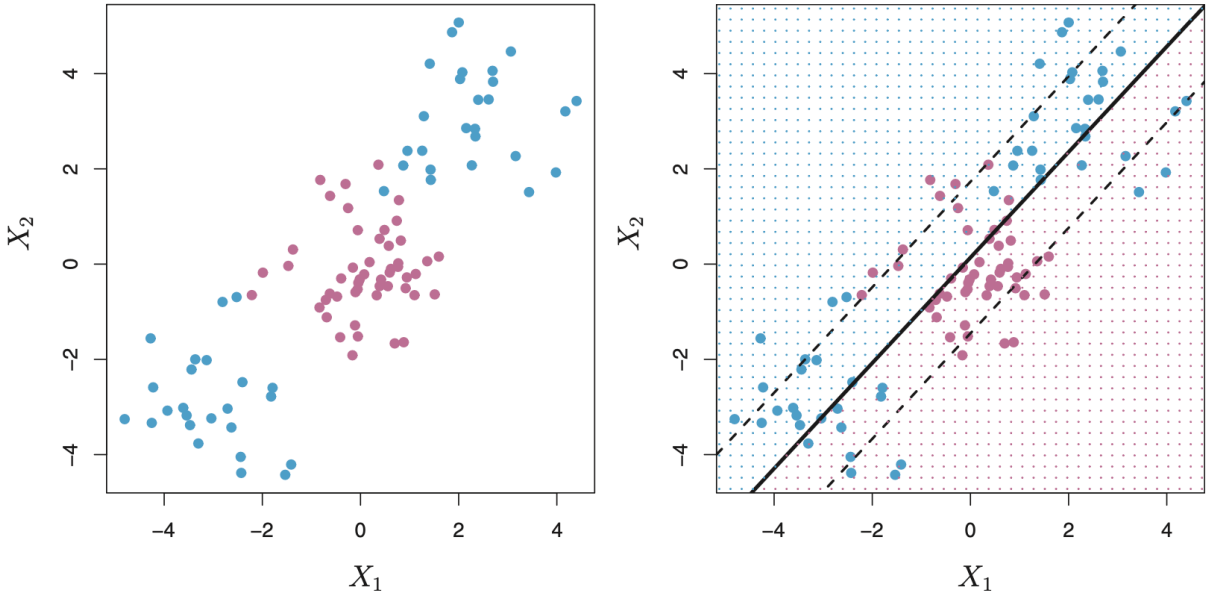
e.g. $X3 = (X1^2 + X2^2)^{(1/2)}$

- Computationally expensive!





Less ideally…



The Kernel Trick

- The SVM scoring function can be rewritten:

$$f(x) = \boldsymbol{w}^\top x_i + b = \sum_{i=1}^{n} \boldsymbol{\alpha_i y_i x_i}^\top x + b$$

Where $x_i{}^\top x$ is the dot product of a vector $\mathbf{x}$ and all training samples $\mathbf{x}_i$

- We replace the dot product $x_i{}^\top x$ with a kernel function:

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

Where $K(\mathbf{x}_i, \mathbf{x})$ is the kernel function and α_i is a weight coefficient

The Kernel Trick

- Kernel Functions:

1. Linear Kernel

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

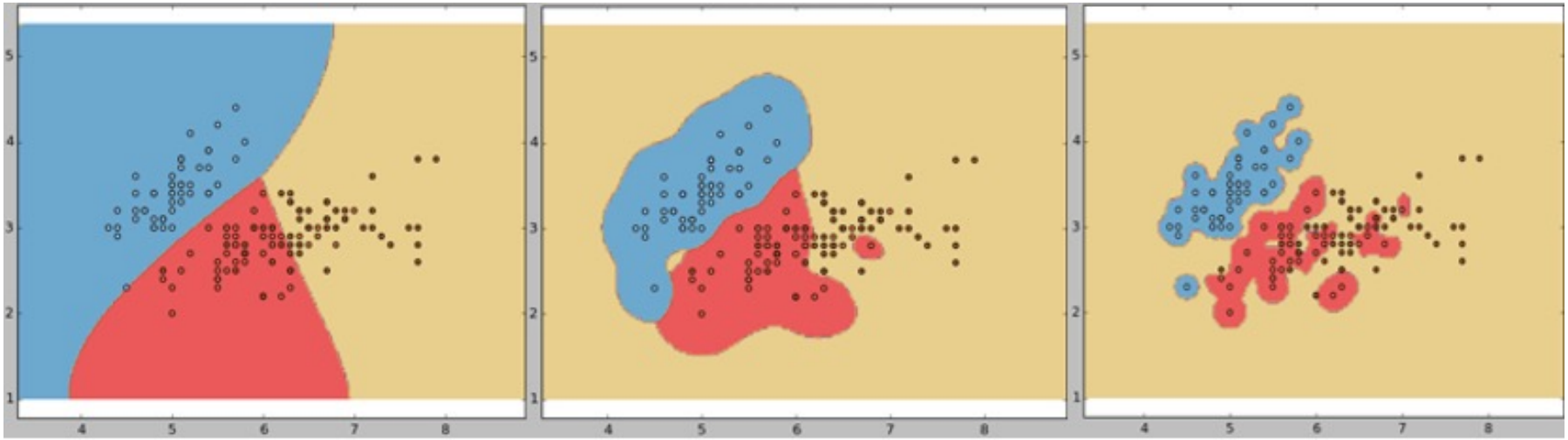
2. Radial Basis Function (RBF) Kernel

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

3. Polynomial Kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + c)^d$$

Parameter Gamma (γ) in RBF



γ = 0.1 γ = 10 γ = 100

Many Kernels

- Polynomial Kernel
- Gaussian Kernel
- Gaussian RBF Kernel
- Laplace RBF Kernel
- Hyperbolic Tangent Kernel
- Sigmoid Kernel
- Bessel function of first kind Kernel
- ANOVA radial basis Kernel
- Linear Splines Kernel

In-class exercise

https://rpi.box.com/s/uu52zsneoqks5ei5zl7he3tpn5orcqrk

Thanks!