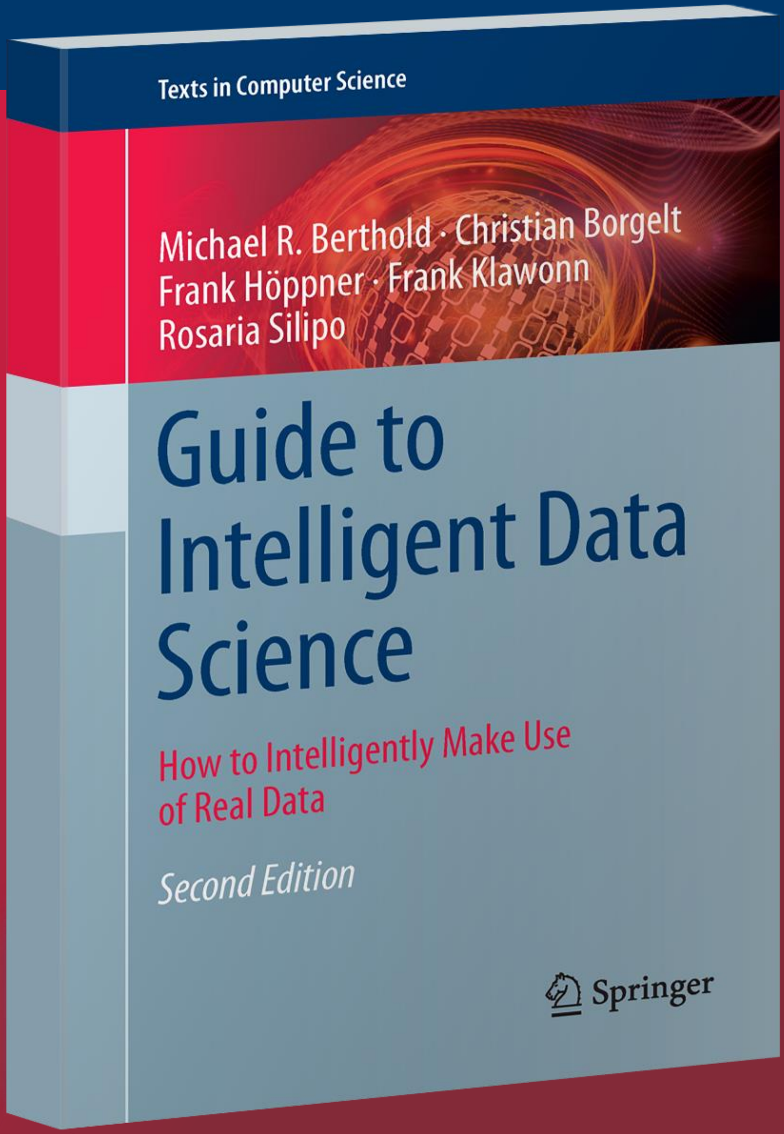
Support Vector Machines (SVM)



Summary of this lesson

“The key to artificial intelligence has always been the representation”
-Jeff Hawkins

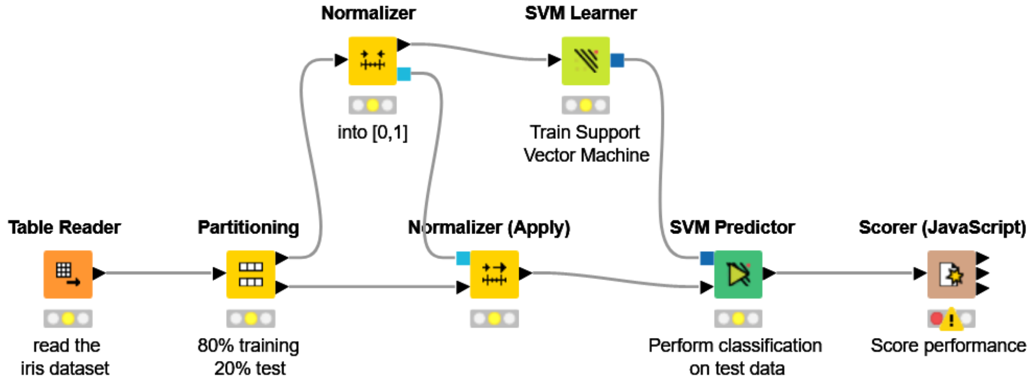
What are Support Vector Machines?

**This lesson refers to chapter 9 of the GIDS book*

What you will learn

- Support Vector Machines (SVM)
  - Overview
  - Dual Representation
  - Kernel Functions
  - Margin of Error

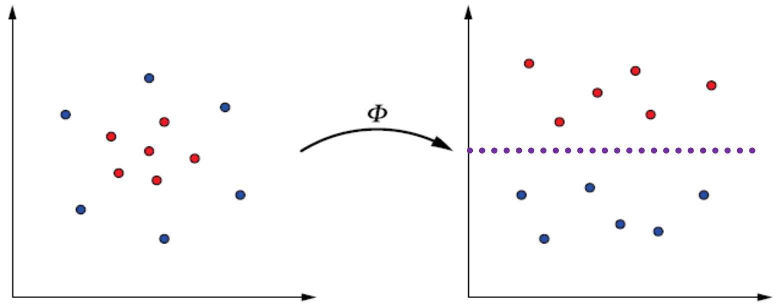
- Datasets used : iris dataset
- Example Workflows:
  - „SVM on iris dataset “ https://kni.me/w/DTfbNITUngKQVF8v
    - Normalization
    - SVM



General idea:

- For classification, linear separation is enough
- For regression, linear regression is enough

→ Given that data is transformed to a space where linear methods work



- Explicit transformation is not necessary
- All you need is a kernel function $\Phi(\cdot)$ describing the transformation to the linear space

Overview

- Embed data into suitable vector space
- Find linear classifier (or other linear pattern of interest) in new space
- Needed: a Mapping

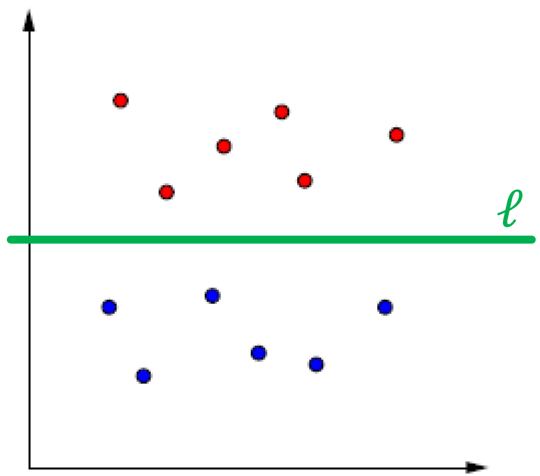
$$\Phi\colon\ x \in X \ \rightarrow\ \Phi(x) \in F$$

- **Kernel Trick**
- Information about relative position is often all that is needed by learning methods
- The inner products between points in the projected space can be computed in the original space using special functions (kernels).

- Consider a binary classification problem, with ±1 as the classes
- Linear discriminat function $f(\boldsymbol{x})$ return which side of the discrimating line ℓ a point $\boldsymbol{x}$ lies

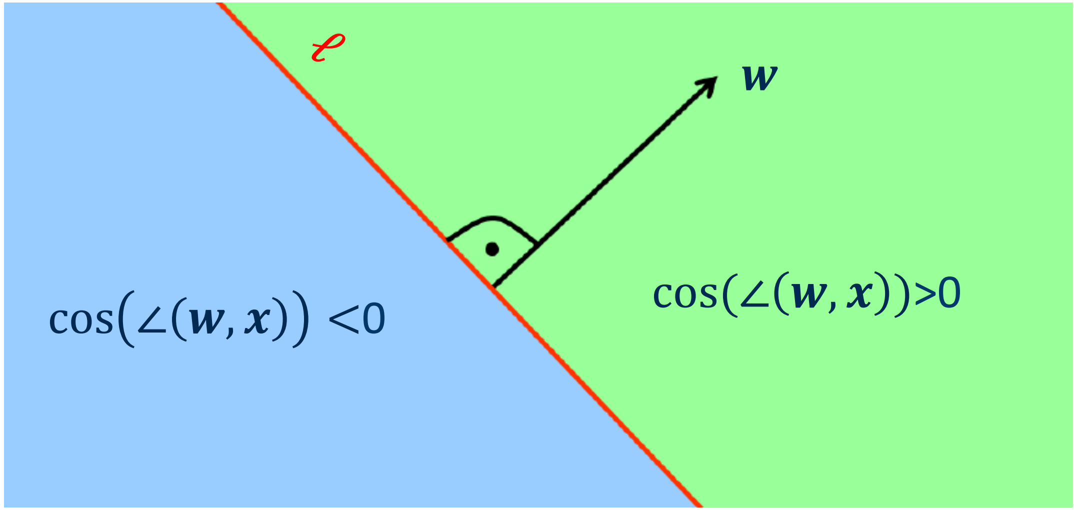
$$f(\boldsymbol{x}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b = b + \|\boldsymbol{w}\|\|\boldsymbol{x}\|\cos(\angle(\boldsymbol{w}, \boldsymbol{x}))$$

- Desicion function $h(\boldsymbol{x})$ classify $\boldsymbol{x}$ according to $h(\boldsymbol{x}) = sign(f(\boldsymbol{x}))$

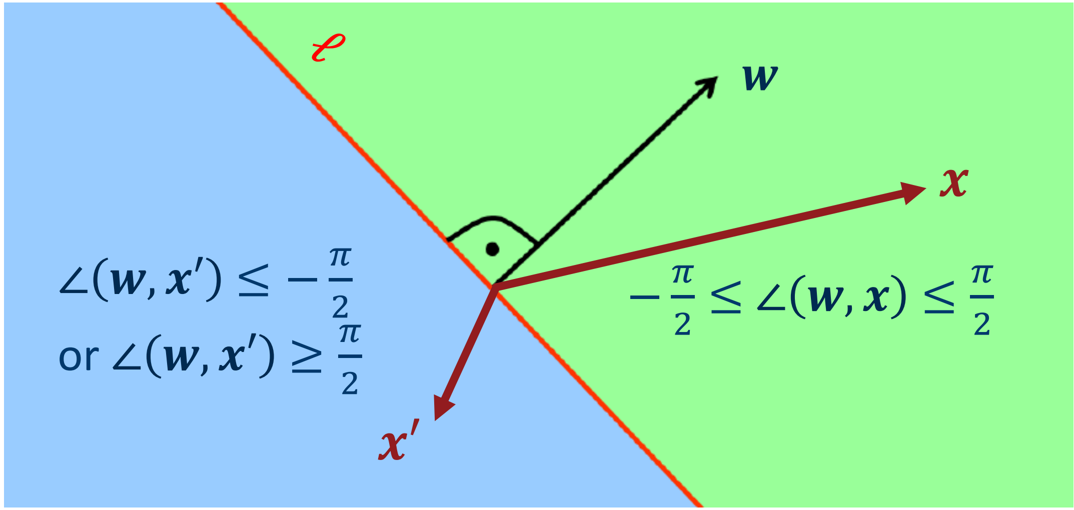


– Linear discriminants represent hyperplanes in feature space

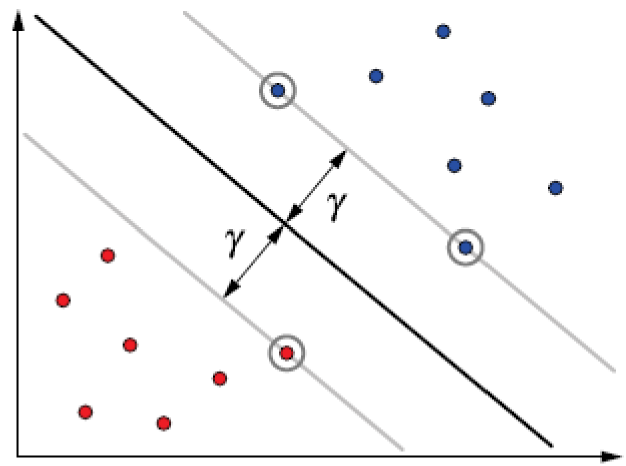
$$f(\boldsymbol{x}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b = b + \|\boldsymbol{w}\|\|\boldsymbol{x}\|\cos(\angle(\boldsymbol{w}, \boldsymbol{x}))$$



– $\boldsymbol{x}$ lies either side of the discriminant function depending on the angle $\angle(\boldsymbol{w}, \boldsymbol{x})$



- To represent $\boldsymbol{w}$, we only need points which lie closest to the separating hyperplane ➔ known as ***support vectors***
- The margin γ – distance between the hyperplane and a support vector
- The margin γ is calculated as: $\gamma = \max_{\boldsymbol{w}} \min_{j} \boldsymbol{w}^T \boldsymbol{x}_j$



Dual Representation

Dual Representation

- Weight vector $\boldsymbol{w}$ is a weighted sum of input $\boldsymbol{x_j}$

$$\boldsymbol{w} = \sum_{j=1}^{n} \alpha_j \cdot y_j \cdot \boldsymbol{x}_j$$

Where α_j represents how much $\boldsymbol{x_j}$ contributes to $\boldsymbol{w}$

- Large α_j: $\boldsymbol{x_j}$ is difficult to classify – higher information content
- Small or zero α_j: $\boldsymbol{x_j}$ easy to classify – smaller information content

→ This representation with α_j's – known as ***dual representation***

- We can now represent the discriminant function as

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b = \left(\sum_{j=1}^{n} \alpha_j \cdot y_j \cdot \boldsymbol{x}_j^T \boldsymbol{x}\right) + b$$

- Both α_j and b can be updated iteratively
- At iteration t,
- IF:

$$y_j \cdot \left(\sum_{j'} \alpha_{j'} \cdot y_{j'} \cdot \boldsymbol{x}_{j'}^T \boldsymbol{x}_j + b \right) < 0$$

- THEN:

$$\alpha_j^{(t+1)} = \alpha_j^{(t)} + y_j$$

$$b_j^{(t+1)} = b_j^{(t)} + y_j \cdot R^2$$

- Where $R = \max_j \lVert \boldsymbol{x}_j \rVert$

Dual Representation of Learning Algorithm:

Given a training set S

$\vec{\alpha} \leftarrow \mathbf{0}$; $b \leftarrow 0$

$R \leftarrow max_{1 \leq i \leq m} ||x_i||$

repeat

for $i = 1$ **to** m

if $y_i(\sum_{j=1}^{m} \alpha_j y_j \langle \vec{x_j}, \vec{x_i} \rangle + b) \leq 0$ **then**

$\alpha_i \leftarrow \alpha_i + 1$

$b \leftarrow b + y_i R^2$

end if

end for

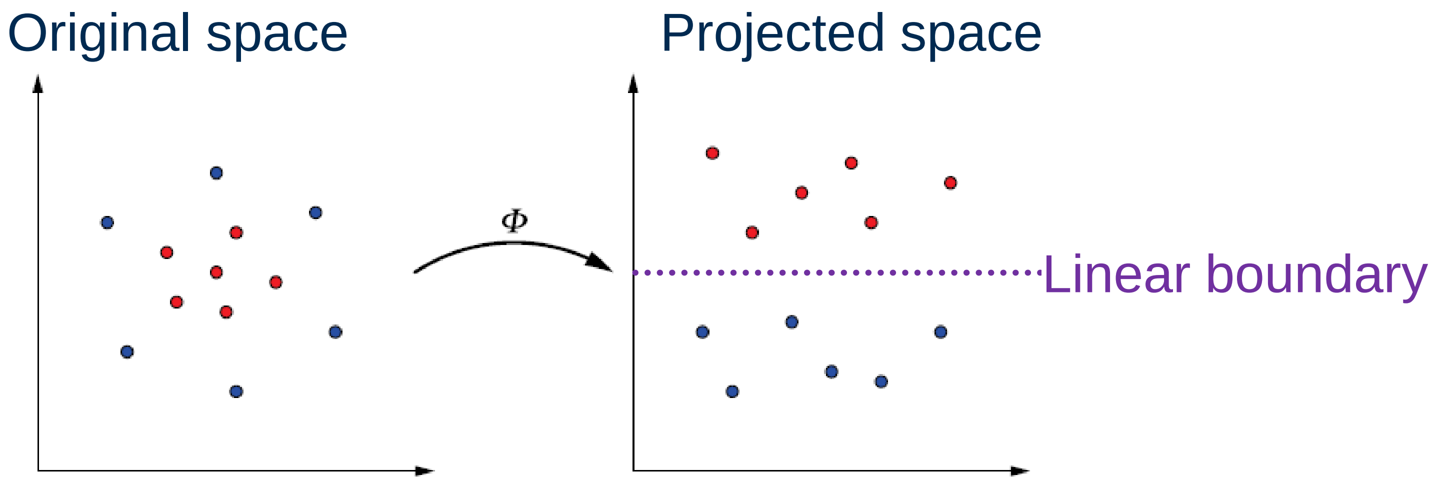
until no mistakes made within the *for* loop

return $(\vec{\alpha}, b)$

– So far, we have seen training via computation of inner products

→ Indicating which side of the linear decision boundary $\boldsymbol{x}$ falls into

– Say, it is hard to find a linear boundary in the original space



– Solution: project to another space, find the linear boundary in the projected space, classify in the projected space

Kernel Functions

- A **kernel function** K is the inner product of data projected by the function Φ

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \Phi(\boldsymbol{x}_1)^T \Phi(\boldsymbol{x}_2)$$

- It is not necessary to transform the original data into the projected space before learning linear SVM
- The discriminant function in the projected space

$$f(\boldsymbol{x}) = \left(\sum_{j=1}^{n} \alpha_j \cdot y_j \cdot \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{x}_j)\right) + b$$

- Or with the kernel function K

$$f(\boldsymbol{x}) = \left(\sum_{j=1}^{n} \alpha_j \cdot y_j \cdot K(\boldsymbol{x}, \boldsymbol{x}_j)\right) + b$$

All data necessary for

- The decision function $h(\boldsymbol{x})$
- The training of the coefficients

Can be pre-computed using a Gram matrix K

$$K = \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & K(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_m) \\ K(\boldsymbol{x}_2, \boldsymbol{x}_1) & K(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_2, \boldsymbol{x}_m) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_m, \boldsymbol{x}_1) & K(\boldsymbol{x}_m, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_m, \boldsymbol{x}_m) \end{pmatrix}$$

Rules for a Gram Matrix

For Gram Matrices, interesting observations hold

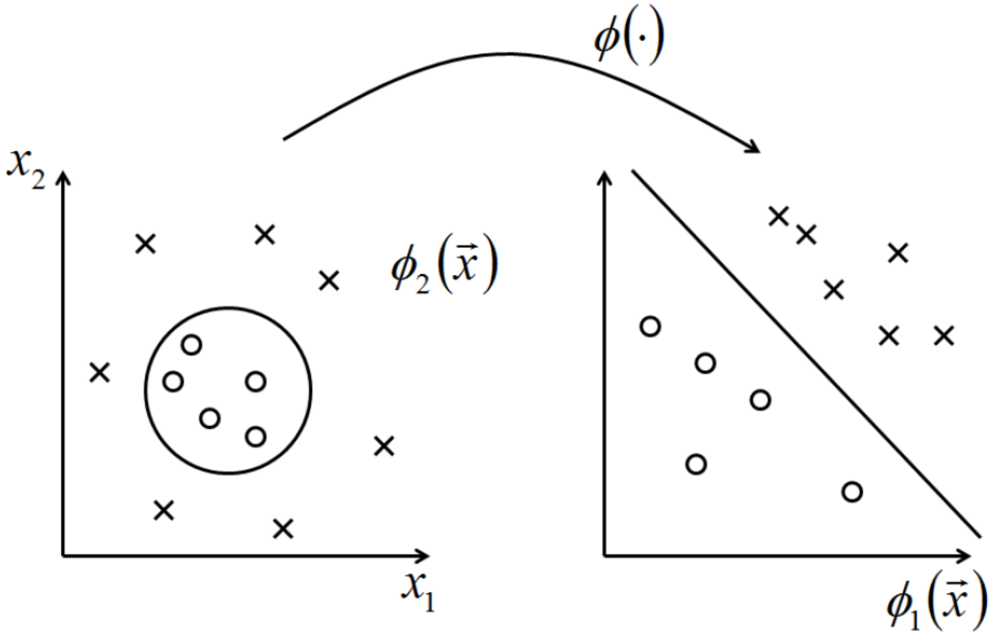
- Symmetric
- Positive definite
- Eigenvectors of the matrix correspond to the input vectors

Moreover,

- Every positive definite & symmetric matrix is a Gram matrix

– Polynomial kernel of degree d

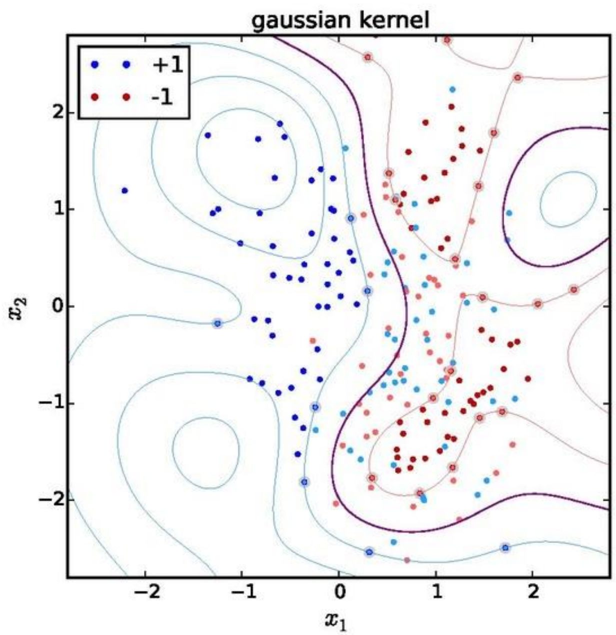
$$K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^T \boldsymbol{y} + c)^d$$



- Gaussian kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}}$$

- Also known as radial basis function (RBF) kernel

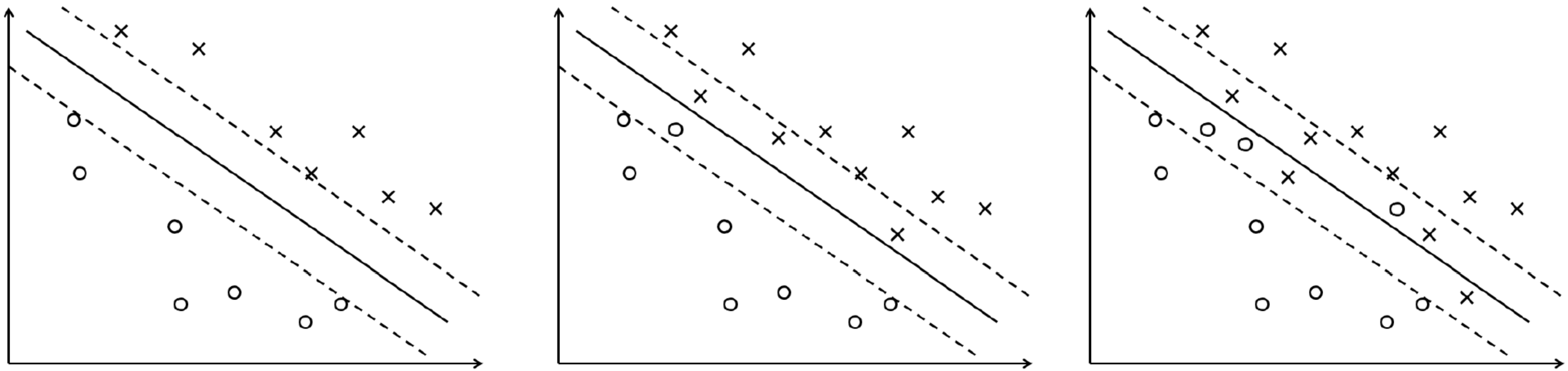


Margin of Error

– What can we do if no linear separating hyperplane exists?

– Solution: allow minor violations – also known as ***soft margins***

→ In contrast, avoiding any misclassifications ≡ ***hard margins***



Hard margins ⟷ ***Soft margins***

- How do we implement soft margins? → via ***slack variables*** ε_j
- Introducing the slack variables to the minimization constraint

$$\forall j = 1, \dots, n: \quad y_j \cdot \left(\boldsymbol{w}^T \boldsymbol{x}_j + b\right) \geq 1 - \varepsilon_j$$

- Misclassifications are allowed if slack $\varepsilon_j > 1$ is allowed
- The minimization problem is solved using Lagrange multipliers

$$\arg\min \frac{1}{2} ||w||^2 + C \sum_j \varepsilon_j$$

- Subject to: $y_j \cdot \left(\boldsymbol{w}^T \boldsymbol{x}_j + b\right) \geq 1 - \varepsilon_j$
- The regularization parameter $C > 0$ controls the “hardness” of the margins (large C → hard margins, small C → soft margins)

Multi-Class SVM

How do we separate more than two classes?

- Transform the problem into a set of binary classification problems
  - One class vs. all other classes
  - One class vs. another class, for all possible class pairs

- The class with the farthest distance from the hyperplane wins

- The key idea: change the optimization

$$\arg\min \frac{1}{2} ||w||^2$$

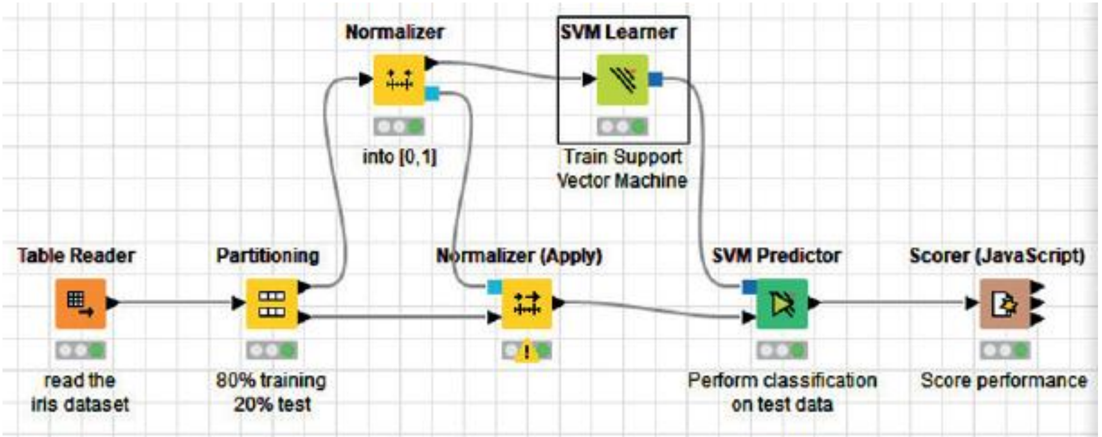
- Subject to: $y_j - \left(\boldsymbol{w}^T \boldsymbol{x}_j + b\right) \leq \varepsilon \quad \text{for } 1 \leq j \leq n$

- This require the prediction error to be within a margin ε
- We can introduce slack variables to tolerate larger errors

- Support Vector Machine
  - Classifier as weighted sum over inner products of training pattern (or only support vectors) and the new pattern.
  - Training analog
- Kernel-Induced feature space
  - Transformation into higher-dimensional space (where we will hopefully be able to find a linear separation plane).
  - Representation of solution through few support vectors (> 0).
- Maximum Margin Classifier
  - Reduction of Capacity (Bias) via maximization of margin (and not via reduction of degrees of freedom).
  - Efficient parameter estimation.
- Relaxations
  - Soft Margin for non separable problems.

Practical Examples with KNIME Analytics Platform

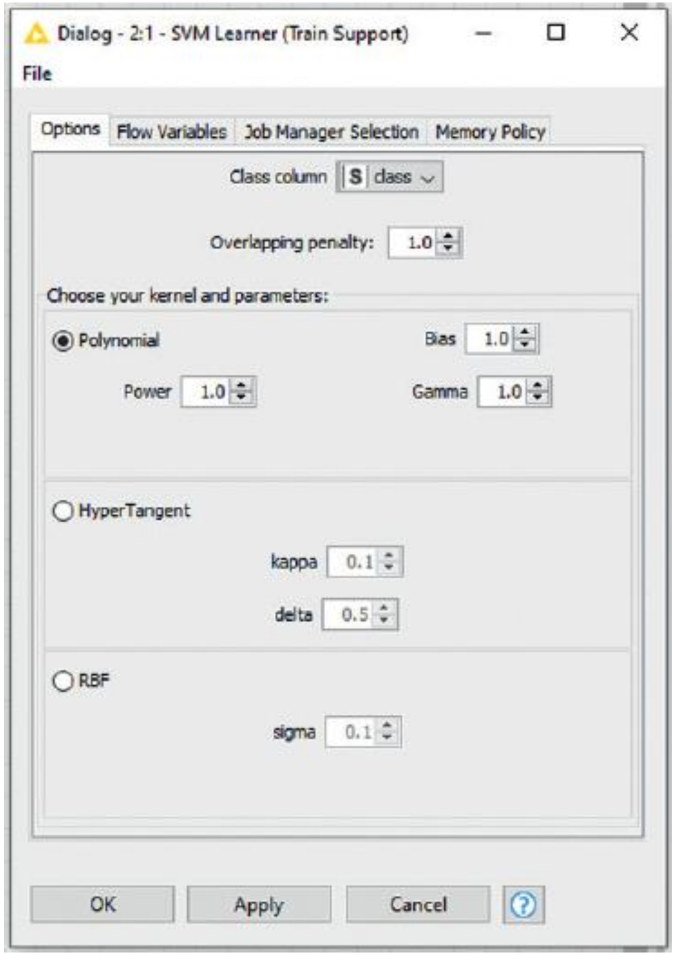
SVM on the Iris Data



– Workflow training an SVM model to classify the iris data set

SVM on the Iris Data

- The configuration window of the SVM Learner node
- Allows a selection of a kernel and the associated parameters
- Overlapping penalty controls the margin hardness



Thank you

For any questions please contact: education@knime.com