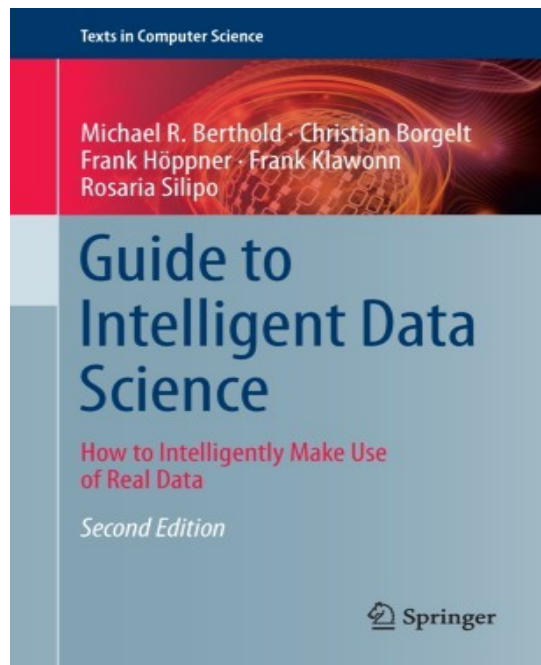




Big Data ACUA6002 Lecture 8

Support Vector Machines

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

What are Support Vector Machines?

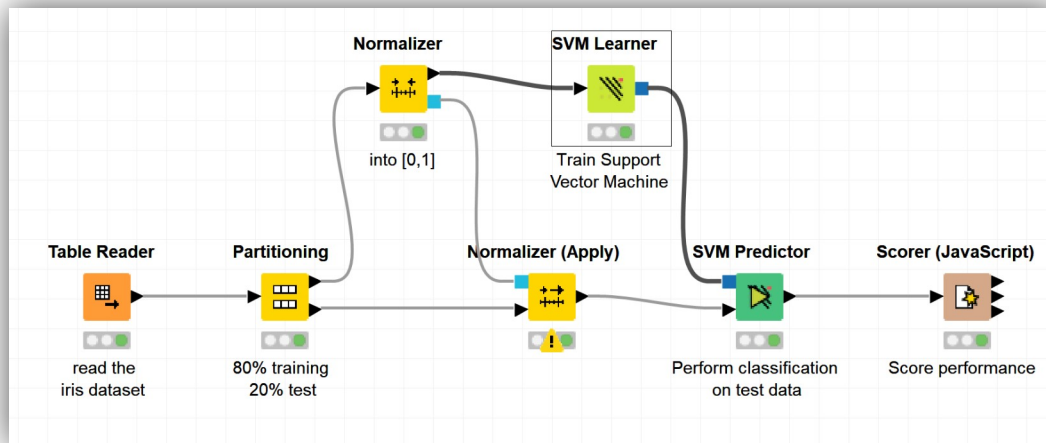
**This lesson refers to chapter 9 of the GIDS book*

Support Vector Machines (more generally – Kernel Machines)

- Motivation
- Linear Classifiers
 - Rosenblatt Learning Rule
- Kernel Methods and Support Vector Machines
 - Dual Representation
 - Maximal Margins
 - Kernels
- Margin of Error and Variations
 - Soft and Hard Margin Classifiers
 - Multi-Class SVM
 - Support Vector Regression

Datasets

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



Motivation

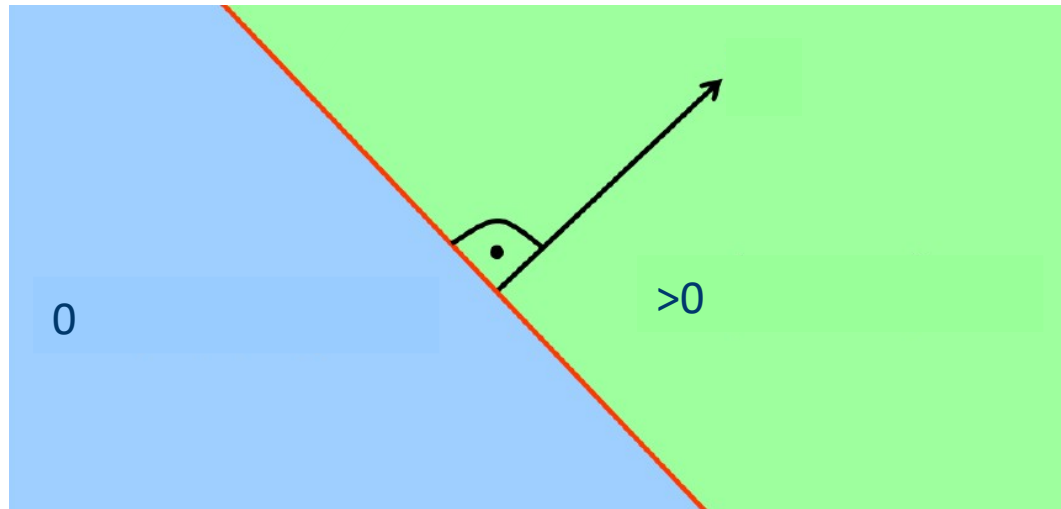
- Main idea of Kernel Methods
 - Embed data into suitable vector space
 - Find linear classifier (or other linear pattern of interest) in new space
- Needed: a Mapping
- Key Assumptions:
 - Information about relative position is often all that is needed by learning methods
 - The inner products between points in the projected space can be computed in the original space using special functions (kernels).

Linear Classifiers

- Simple linear, binary classifier:
- Class A if $z > 0$
- Class B if $z < 0$
- e.g. $z = w^T x + b$ is the decision function

Linear Discriminant Function

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



- Linear discriminants represent hyperplanes in feature space

Training a “Perceptron”

- Classification using a Perceptron
 - Represents a (hyper-) plane:
 - Left of hyperplane: class 0
 - Right of hyperplane: class 1
- Training a Perceptron
 - Learn the “correct” weights to distinguish the two classes
 - Iterative adaption of weights
 - Rotation of the hyperplane defined by $\mathbf{w}$ and $\mathbf{b}$ in small direction of $\mathbf{w}$ if is not yet on the correct side of the hyperplane.

- Rosenblatt (1959) introduced a simple learning algorithm for linear discriminants ("perceptrons"):
- Given a linearly separable training set S

```
repeat
  for to
    if then

        end if
    end for
until no mistakes made within the for loop
return
```

- Algorithm is
 - *On-line* (pattern by pattern approach)
 - *Mistake driven* (updates only in case of wrong classification)
- Algorithm converges guaranteed if a hyperplane exists which classifies all training data correctly (data is linearly separable)
- Learning rule:
- One observation:
 - Weight vector (if initialized properly) is simply a weighted sum of input vectors (b is even more trivial).

Dual Representation (of discriminant function)

- Weight vector is a weighted sum of input

$$= \sum_{i=1}^n \cdot \cdot$$

Where represents how much contributes to

- Large : is difficult to classify – higher information content
 - Small or zero : easy to classify – smaller information content
- This representation with 's – known as **dual representation**

- We can now represent the discriminant function as

$$\left(\right) = + = \left(\sum_{i=1}^n \cdot \cdot \right) +$$

Dual Representation

- Dual Representation of Learning Algorithm:
- Given a training set S

```
repeat
  for to
    if then

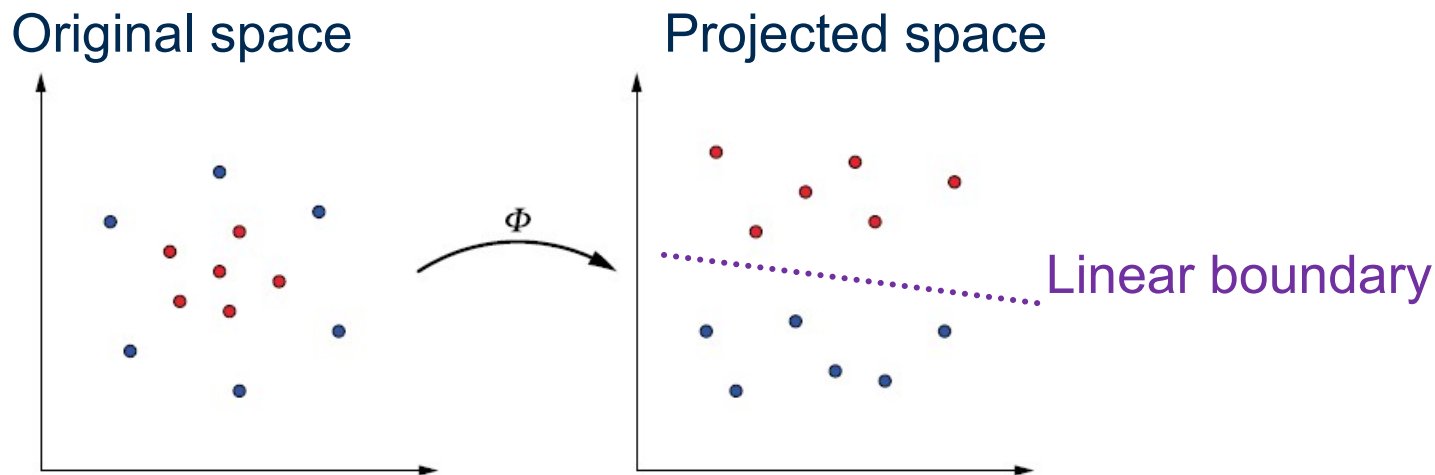
        end if
    end for
until no mistakes made within the for loop
return
```

- Both $\mathbf{w}$ and $\mathbf{b}$ can be updated iteratively
- Learning Rule at iteration

where

- Harder to learn examples having larger α
- The information about training examples enters algorithm only through the inner products (which we could pre-compute)

- So far, we have seen training via computation of inner products
- Indicating which side of the linear decision boundary 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 Methods and Support Vector Machines

- A **kernel function** is a function k , such that for all

where ϕ is a mapping from $\mathcal{X}$ to an (inner product) feature space $\mathcal{H}$.

- It is not necessary to transform the original data into the projected space before learning linear SVM
- The kernel k allows us to compute the inner product of two points x and y in the projected space without even entering that space

- The discriminant function in the projected space
- Or with the kernel function

All data necessary for

- the decision function
- the training of the coefficients

can be pre-computed using a Gram matrix

Rules for a Gramm Matrix

- Let S be a non empty set. A function k is a valid kernel in S if for all $x, y \in S$ and all n it produces a Gram matrix G , which is:
- Symmetric
- Positive semi-definite
- Eigenvectors of the matrix correspond to the input vectors

Moreover,

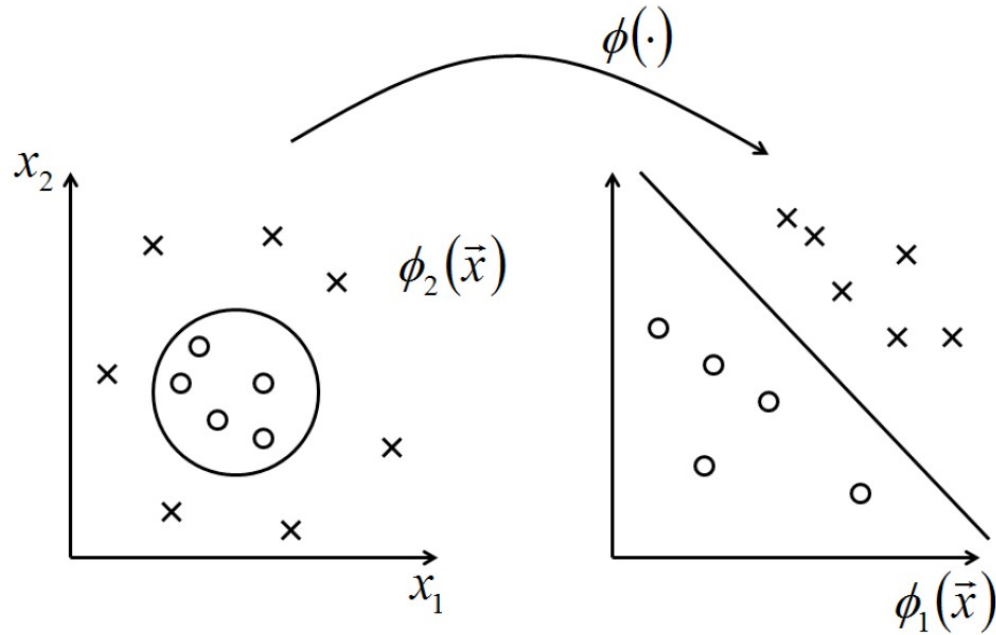
- Every positive definite & symmetric matrix is a Gram matrix

- A simple kernel is
- And the corresponding projected space:
- Since

- A few less simple kernels are
- And the corresponding projected spaces are of dimension
- But computing the inner products in the projected space can quickly become expensive

- Polynomial kernel of degree

$$(\cdot, \cdot) = (\cdot + \cdot)$$



Kernels

- Note that we do not need to know the projection .
- It is sufficient to prove that is a Kernel.

A few notes:

- Kernels are modular and closed: we can compose new Kernels based on existing ones
- Kernels can be defined over non numerical objects:
 - Text: e.g. string matching kernel
 - Images, trees, graphs...
- A good kernel is crucial
 - Gram Matrix diagonal: classification easy and useless

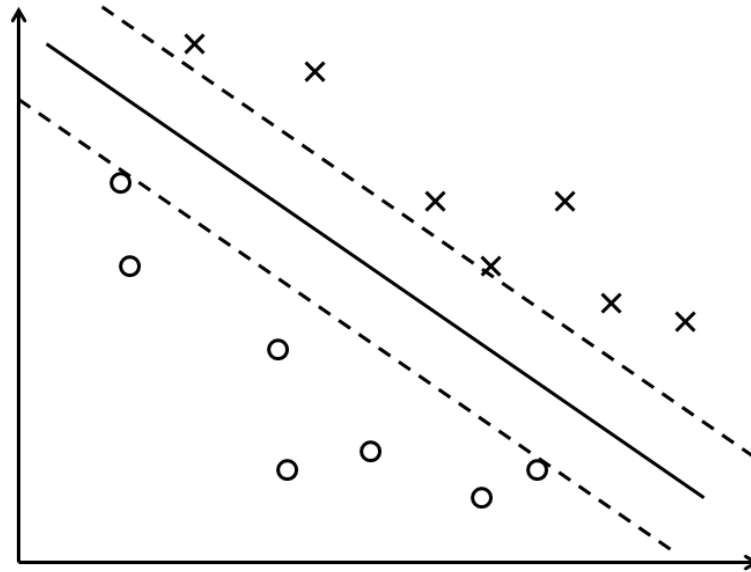
Finding Linear Discriminants

- Finding the hyperplane (in any space) still leaves lots of room for variations
- We can define “margins” of individual training examples:

appropriately normalized this is a “geometrical” margin

- The margin of a hyperplane (with respect to a training set):
- And a maximal margin of all training examples indicates the maximum margin over all hyperplanes

(maximum) Margin of a Hyperplane



Finding Linear Discriminants

- The original objective function
- Is reformulated slightly:
- The decision line is still defined by
- And in addition the upper and lower margins are defined by
- The distance between those two hyperplanes is

Finding Linear Discriminants

- Finding the maximum margin now turns into a minimization problem:
 - Minimize (in
 - subject to (for any)

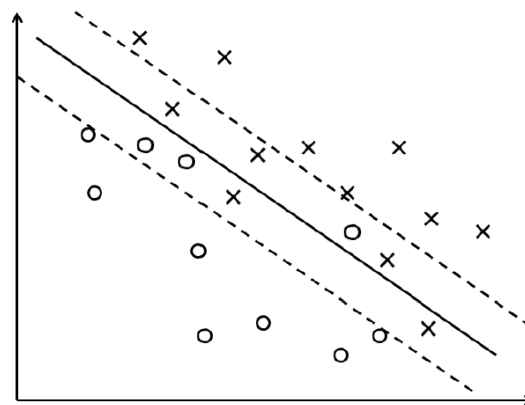
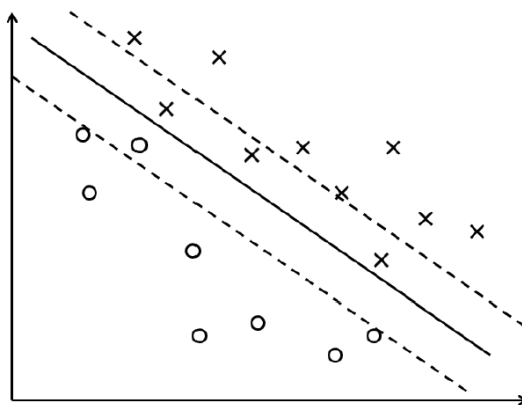
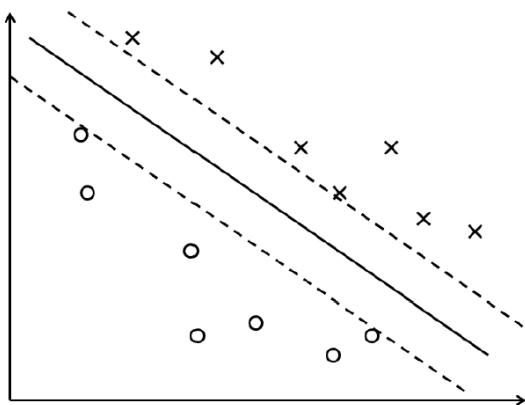
Solution sketch:

- Solutions depend on , the norm of which involves a square root
- Convert into a quadratic form by substituting with without changing the solution
- Using Lagrange multipliers this turns into a standard quadratic programming problem

Margin of Error and Variations

Soft and Hard Margin Classifiers

- What can we do if no linear separating hyperplane exists?
- Solution: allow minor violations – also known as ***soft margins***
 - In contrast, avoiding any misclassifications $\equiv$ ***hard margins***



Hard margins



Soft margins

- How do we implement soft margins? → via ***slack variables***
- Introducing the slack variables to the minimization constraint
- Misclassifications are allowed if slack is allowed
- The minimization problem is solved using Lagrange multipliers
- Subject to:
- The regularization parameter controls the “hardness” of the margins (large → hard margins, small → soft margins)

How do we separate more than two classes?

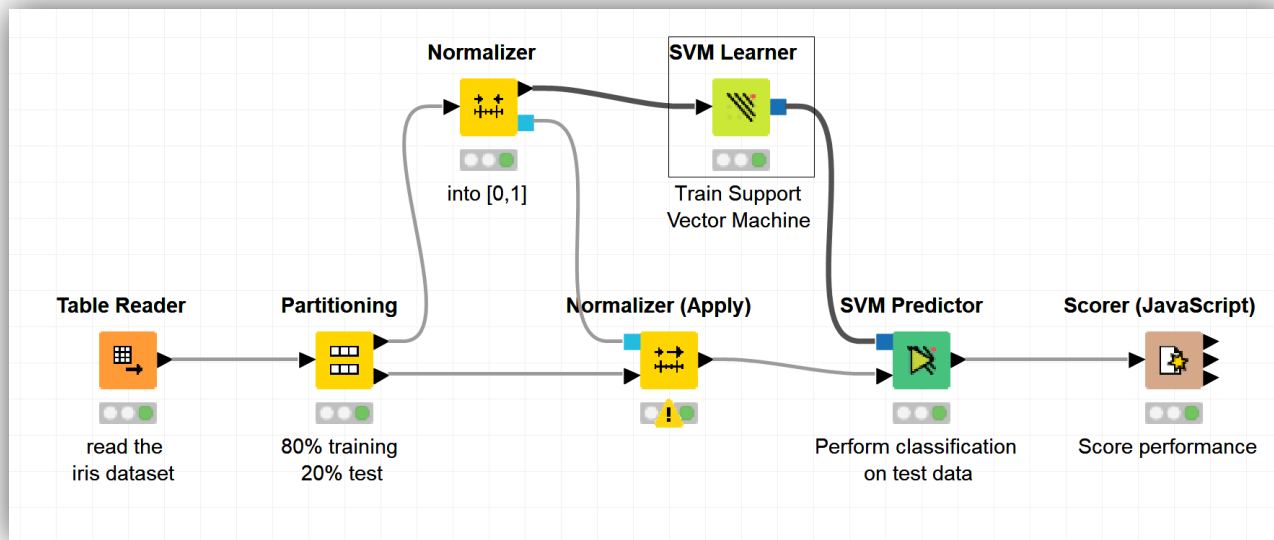
- Transform the problem into a set of binary classification problems
 - One class vs. all other classes
 - One class vs. another class, for all possible class pairs
- The class with the farthest distance from the hyperplane wins

- The key idea: change the optimization
- Subject to:
- This require the prediction error to be within a margin
- 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 ($\cdot$).
- **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

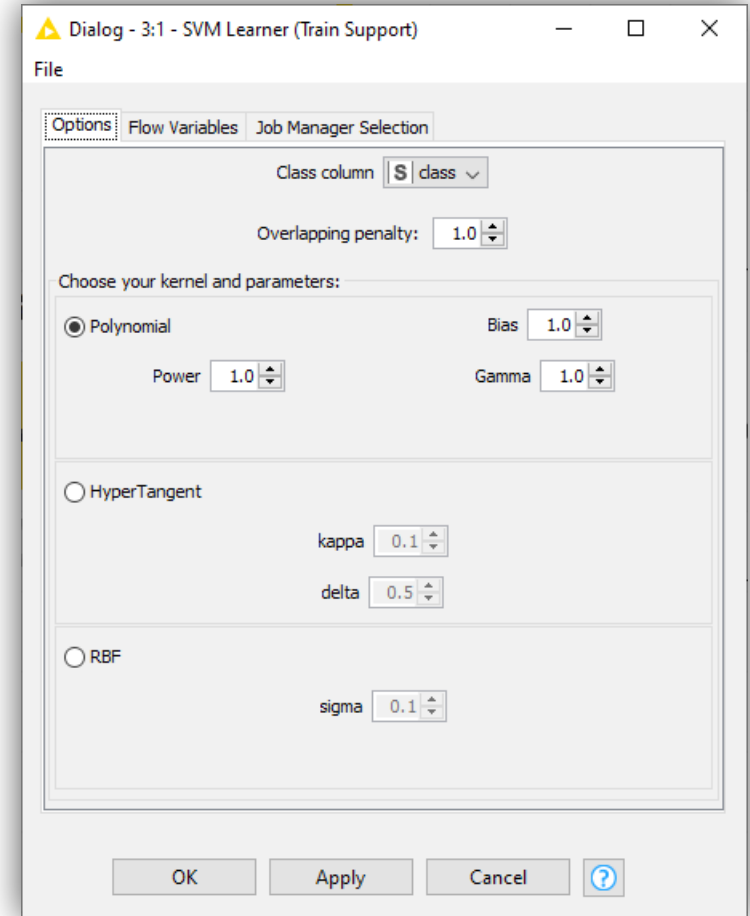
SVM on the Iris Data



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

SVM on the Iris Data

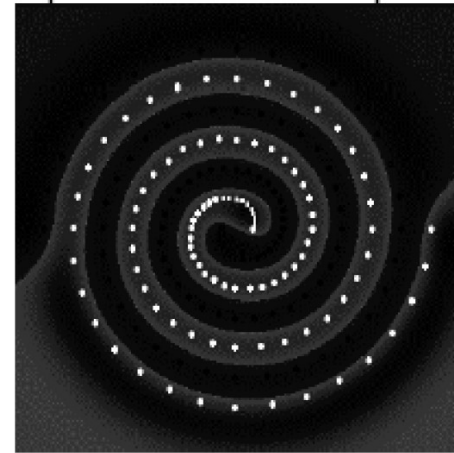
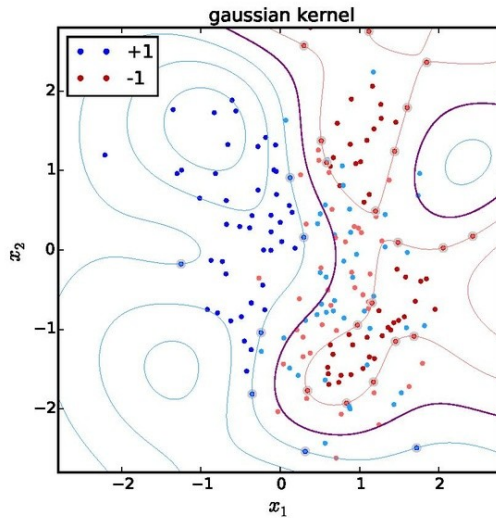
- The configuration window of the SVM Learner node
- Allows a selection of a kernel and the associated parameters
- Overlapping penalty controls the margin hardness



- Gaussian kernel

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

- Also known as radial basis function (RBF) kernel



Berthold, M.R., Borgelt, C., Höppner, F., Klawonn, F. and Silipo, R. (2020). *Guide to Intelligent Data Science: How to Intelligently Make Use of Real Data*. 2nd ed. [online] www.springer.com. Springer International Publishing.

Available at:

<https://www.springer.com/gp/book/9783030455736> [Accessed 17 Jun. 2021].

Questions?