Predictive Modelling - VI

Support Vector Machines

Rita P. Ribeiro

Data Mining I - 2023/2024

U.PORTO FC FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

[dcc] DEPARTAMENTO DE CIÊNCIA DE COMPUTADORES FACULDADE DE CIÊNCIAS DA UNIVERSIDADE DO PORTO

Summary

- Support Vector Machines
  - Linear SVMs
  - Non-Linear SVMs
  - SVMs for Multi-class Classification
  - SVMs for Regression

Predictive Modelling: Where we at?

- Distance-based Approaches
  - e.g. kNN
- Probabilistic Approaches
  - e.g. Naive Bayes, Bayesian Networks
- Mathematical Formulae
  - e.g. multiple linear regression
- Logical Approaches
  - e.g. CART
- Optimization Approaches
  - e.g. SVM, ANN
- Ensemble Approaches

Support Vector Machines

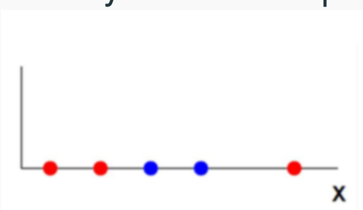
Support Vector Machines (SVM)

- Introduced in 1992
- Based on statistical learning theory
- Have a strong mathematical foundation
- Originally designed to binary classification and regression tasks
- Gave origin to a new class of algorithms named kernel machines
- A good reference on SVMs:
  - N. Cristianini and J. Shawe-Taylor: An introduction to Support Vector Machines. Cambridge University Press, 2000.

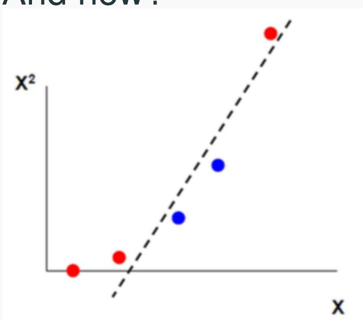
Support Vector Machines (SVM)

Why?

A linear classifier cannot classify these examples



And now?



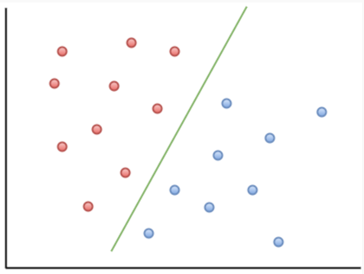
- Nonlinear decision boundary (in the original feature space X)
- Linear decision boundary (in the extended feature space of X: X^2)

Support Vector Machines: Linear SVMs

Setting

- Given a data set $D = \{< \mathrm{x}_i, y_i >\}_{i=1}^{N}$, where x_i is a feature vector and $y_i \in Y$ is the value of the nominal variable in $\{-1, +1\}$
- Every feature vector x_i is a point in a high-dimensional space.
- D is linearly separable if there is an hyperplane: $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ that divides the input space, such that,

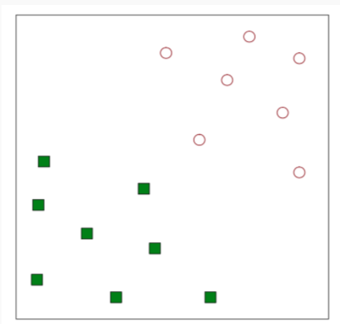
$$g(\mathbf{x}) = sgn(h(\mathbf{x})) = \begin{cases} +1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ -1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b < 0 \end{cases}$$

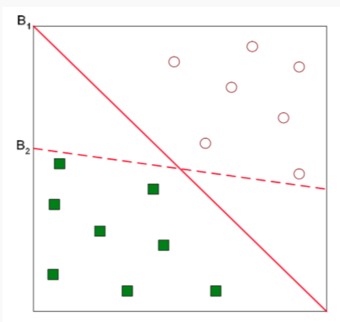


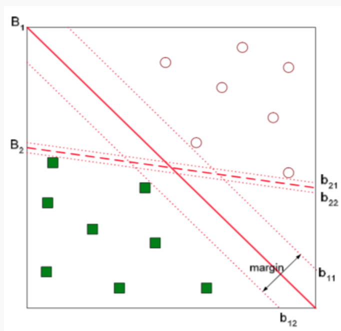
Support Vector Machines: Linear SVMs

The intuition

- Find a decision boundary to separate data.
- Many solutions exist.
- Which solution is better?
- Find a decision boundary that maximizes the margin.

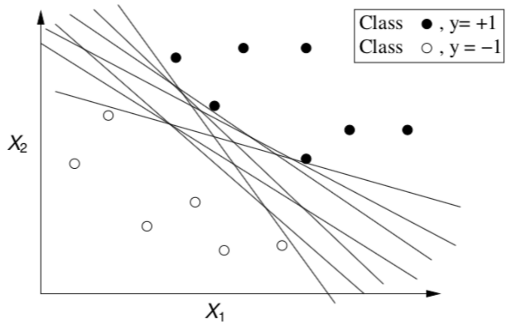


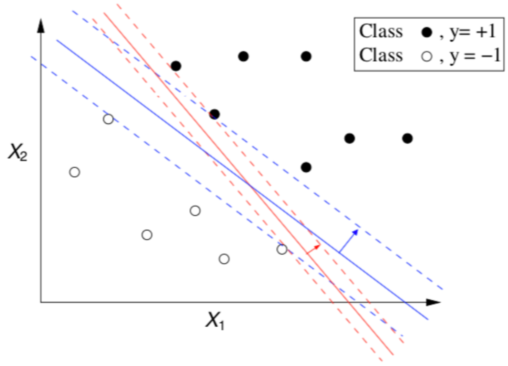




Support Vector Machines: Linear SVMs

- There is an inifinite number of hyperplanes $h(\mathbf{x})$
- Which one is better?
- Maximize generalization ability
- Ensure a better accuracy on unseen data
- SVMs approach this problem: search for the maximum margin hyperplane





Support Vector Machines: Linear SVMs

Maximum Margin Hyperplane

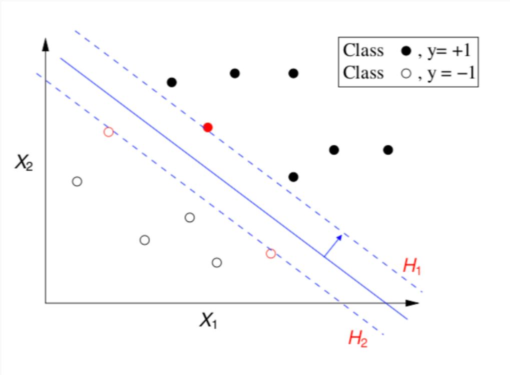
- The hyperplane that separates the examples from the two classes with the maximum margin.
- Largest margin $\rightarrow$ better generalization
- The goal is to find $\mathbf{w}$ and b given that

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b \geq +1 & \text{if } y_i = +1 \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 & \text{if } y_i = -1 \end{cases}$$

- Equivalent to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0, \forall (\mathbf{x}_i, y_i) \in D$

Support Vector Machines: Linear SVMs

The Support Vectors



$$H_1: \quad g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = 1$$
$$H_2: \quad g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = -1$$

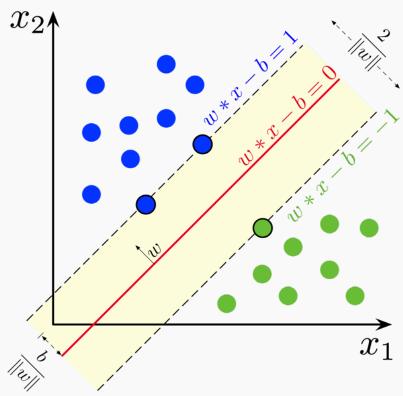
- All cases that fall on the hyperplanes H_1 and H_2 are called the support vectors.
- Removing all other cases would not change the solution!

Support Vector Machines: Linear SVMs

The Optimal Hyperplane

- The distance between the two hyperplanes H_1 and H_2 is $\frac{2}{||\mathbf{w}||}$
- Goal: Maximize this margin, i.e.

$$\min_{\mathbf{w},b} \frac{||\mathbf{w}||^2}{2} = \frac{1}{2}\sum_{i=1}^{n} w_i^2$$
$$\text{s.t.} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0, \forall i = 1, \cdots, N$$



- A constrained optimization problem that can be solved using Lagrange multiplier method

Support Vector Machines: Linear SVMs

These are Hard Margin SVMs

- Works well when data is linearly separable
- On real-world data this is hardly the case
- Does not take into account presence of noise

Solution

- Tolerate some misclassification errors to increase the size of the separation margin, so that other points can still be classified correctly.
- These points allowed inside the margin are referred to as “slack variables”.

Support Vector Machines: Linear SVMs

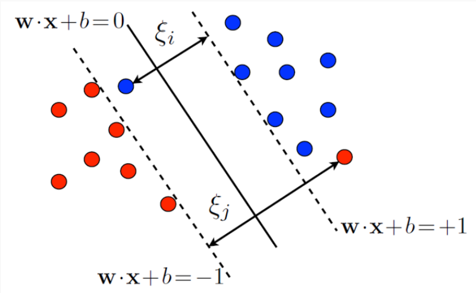
Soft Margin SVMs

- Regularization term C: trade-off between maximizing the margin and minimizing the misclassification errors
- When C is small, misclassification errors are given less importance, the focus is on maximizing the margin
- When C is large, misclassification more costly, the focus is on avoiding them at the expense of keeping the margin small

$$\min_{\mathbf{w},b,\xi} \quad \frac{||\mathbf{w}||^2}{2} + C\left(\sum_{i=1}^{N}\xi_i\right)$$

$$\text{s.t.} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0, \forall i = 1, \cdots, N$$

- ξ_i are the slack variables



Support Vector Machines: Non-Linear SVMs

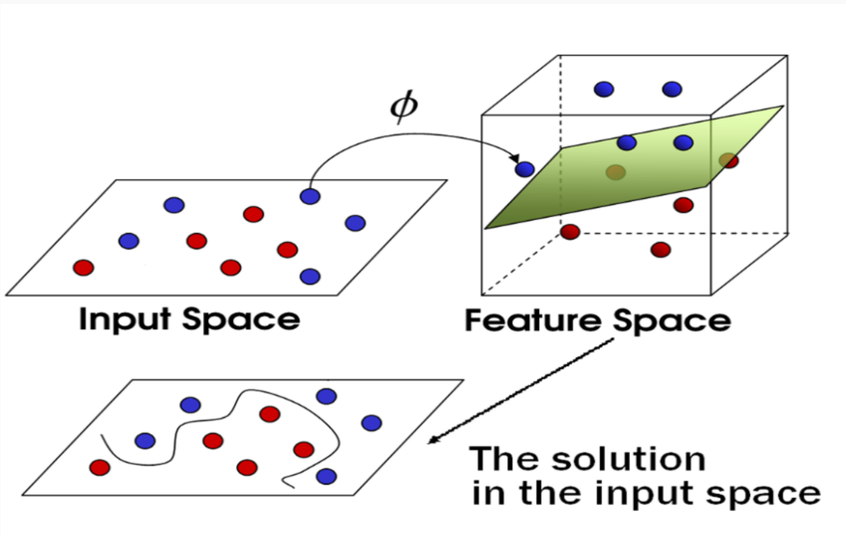
Non-Linear SVMs

- Most real world problems have inherent nonlinearity
- SVMs solve this by “moving” into a extended input space where classes are linearly separable
- This means the maximum margin hyperplane needs to be found on this new very high dimensional space

Support Vector Machines: Non-Linear SVMs

Example

- Input space $\mathbf{x}$ mapped to high-dimensional feature space $\phi(\mathbf{x})$ where the classes are linearly separable.



Support Vector Machines: Non-Linear SVMs

Main idea

- Map the original data into a new (higher dimension) coordinates system where the classes are linearly separable
- Same optimization problem, but involving $\phi(\mathbf{x})$ instead of $\mathbf{x}$
- Still, the solution to the optimization equation involves dot products between feature vectors, $\mathbf{x}_i$ and $\mathbf{x}_j$, that are computationally heavy on high-dimensional spaces
- Calculate the image of $\phi(\mathbf{x})$ of each input vector $\mathbf{x}$ and then do the dot product can be quite expensive.

Support Vector Machines: Non-Linear SVMs

The Kernel Trick:

- It was demonstrated that the result of these complex calculations is equivalent to the result of applying certain functions (kernel functions) in the space of the original variables.
- The kernel function takes as its inputs vectors in the original space and returns the dot product of the vectors in the feature space;
- Using kernels, we do not need to embed the data into the space explicitly, because a number of algorithms only require the inner products between image vectors!
- We never need the coordinates of the data in the feature space!

Support Vector Machines: Non-Linear SVMs

The Kernel Trick: (cont.)

- instead of calculating the dot products in a high dimensional space
- take advantage of the proof that $K(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$
- perform operations in the original space (without a feature transformation!)
- replace the complex dot products by these simpler and efficient calculations
- use a linear optimization solution to solve a non-linear problem

Support Vector Machines: Non-Linear SVMs

An Example

- $\mathbf{x}_i = (x_{i1}, x_{i2}, x_{i3})$, $\mathbf{x}_j = (x_{j1}, x_{j2}, x_{j3})$.
- $\phi(\mathbf{x}) = (x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3)$, a mapping from 3-dimensional to 9-dimensional space.
- the kernel is $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^2$.
- $\mathbf{x}_i = (1, 2, 3)$; $\mathbf{x}_j = (4, 5, 6)$.
- $\phi(\mathbf{x}_i) = (1, 2, 3, 2, 4, 6, 3, 6, 9)$,
  $\phi(\mathbf{x}_j) = (16, 20, 24, 20, 25, 30, 24, 30, 36)$
- $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$
- if we use the kernel instead: $K(\mathbf{x}_i, \mathbf{x}_j) = (4 + 10 + 18)^2 = 32^2 = 1024$
- Same result, but this calculation is so much easier!

Support Vector Machines: Non-Linear SVMs

Mercer's theorem: what functions can be kernels?

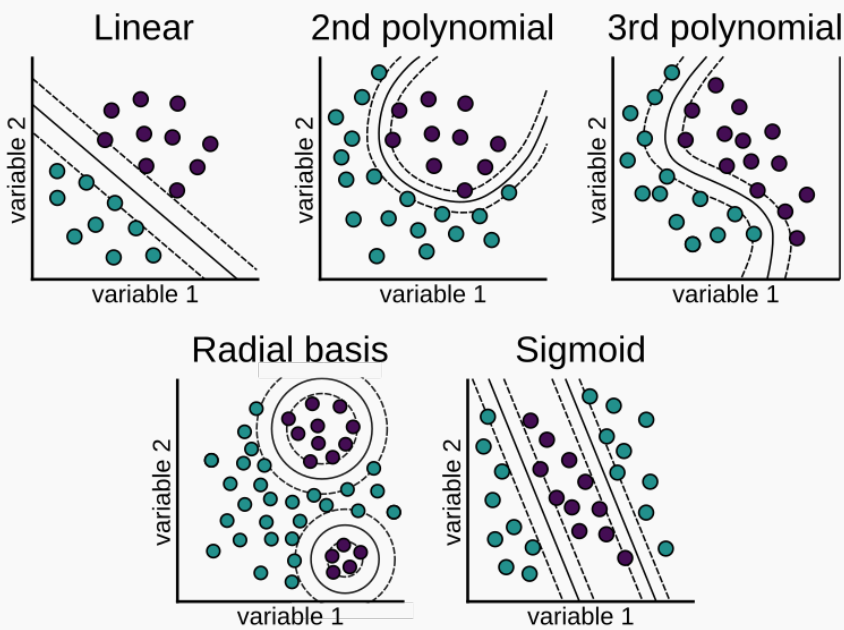
- *every semi-positive definite symmetric function is a kernel*

Examples of Kernel Functions

- Linear:
  - $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
- Polynomial of power p:
  - $K(\mathbf{x}_i, \mathbf{x}_j) = (\delta(\mathbf{x}_i \cdot \mathbf{x}_j) + \kappa)^p$, if $p = 1, \delta = 1, \kappa = 0$ is equivalent to Linear
- Gaussian (radial-basis function network):
  - $K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\sigma||\mathbf{x}_i - \mathbf{x}_j||^2)$
- Sigmoidal:
  - $K(\mathbf{x}_i, \mathbf{x}_j) = tanh(\delta(\mathbf{x}_i \cdot \mathbf{x}_j) + k)$

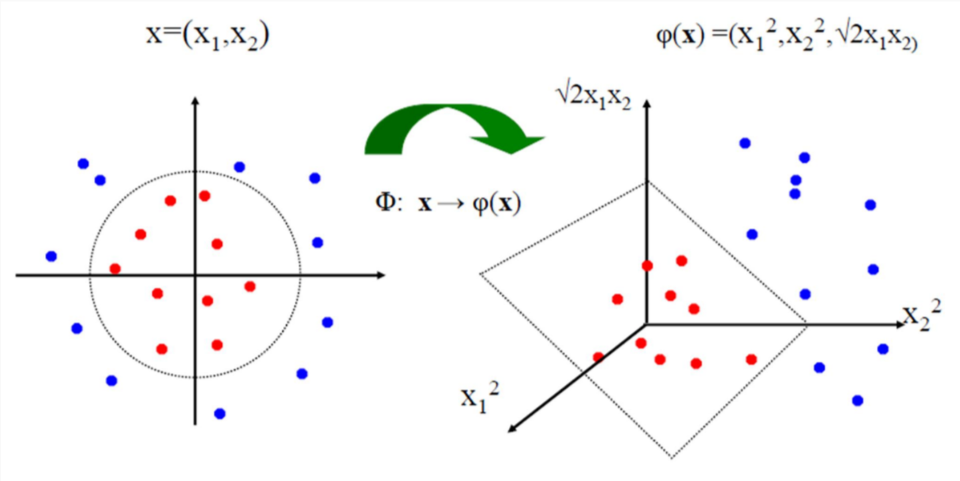
Support Vector Machines: Non-Linear SVMs

Examples of different kernel functions



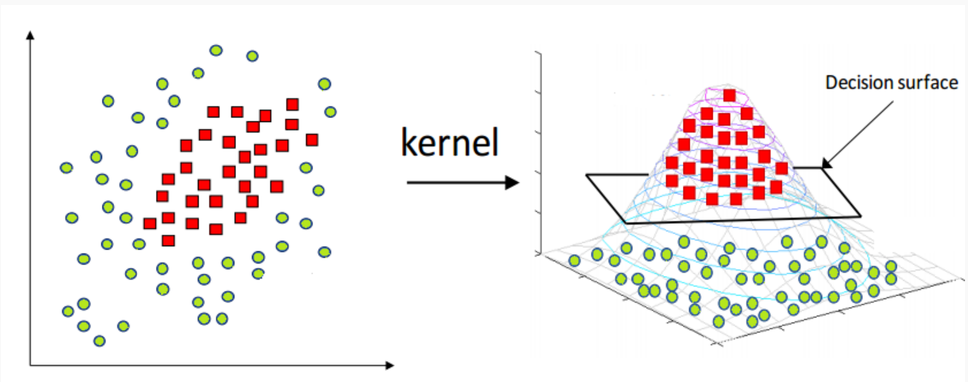
Support Vector Machines: Non-Linear SVMs

Examples of a Polynomial Kernel function



Support Vector Machines: Non-Linear SVMs

Example of a Gaussian RBF Kernel function



Support Vector Machines: Multiclass Classification

How to handle more than 2 classes?

- Solve several binary classification tasks
- Essentially, find the support vectors that separate each class from all others

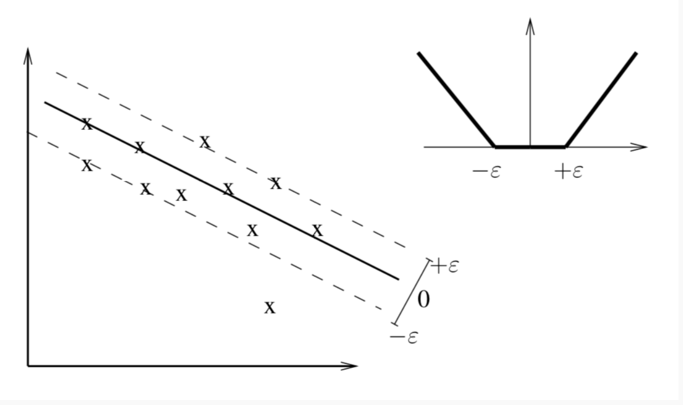
The Algorithm

- Given a m classes task
- Obtain m SVM classifiers, one for each class
- Given a test case assign it to the class whose separating hyperplane is more distant from the test case

Support Vector Machines: Regression

- Vapnik (1995) proposed the ε-SVR (Support Vector Regression)
- Find a linear function $h(\mathbf{x})$ that approximates the training cases with a precision of ε
- ε-SVR uses the following ε-insensitive loss function,

$$|\xi|_\varepsilon = \begin{cases} 0 & |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases}$$



Support Vector Machines: Regression

The theoretical development of this idea leads to the following optimization problem,

$$\min_{\mathbf{w},b,\xi,\xi^\star} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^\star)$$

$$\text{s.t.} \begin{cases} y_i - \mathbf{w}\cdot\mathbf{x} - b & \leq \varepsilon + \xi_i \\ \mathbf{w}\cdot\mathbf{x} + b - y_i & \leq \varepsilon + \xi_i^\star \\ \xi_i, \xi_i^\star & \geq 0 \end{cases}$$

where C corresponds to the cost to pay for each violation of the error limit ε and ξ and $\xi^\star$ are slack variables

Support Vector Machines: Regression

- In summary, by the use of the $|\xi|_\varepsilon$ loss function we reach a very similar optimization problem to find the support vectors
- As within classification, for a non-linear regression problem, we use the kernel trick to map a non-linear problem into a high dimensional space where we solve the same quadratic optimization problem as in the linear case

Support Vector Machines: Summary

- As problems are usually non-linear on the original feature space, move into a high-dimensional space where linear separability is possible
- Find the optimal separating hyperplane on this new space using quadratic optimization algorithms
- Avoid the heavy computational costs of the dot products using the kernel trick

Support Vector Machines: Key Issues

- Choice of kernel
  - Gaussian or polynomial kernel is default
  - if ineffective, more elaborate kernels are needed
  - domain experts can give assistance in formulating appropriate similarity measures
- Choice of kernel parameters
  - e.g. σ in Gaussian kernel, i.e. the distance between the closest points with different classifications
  - in the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.
- Optimization criterion
  - Hard margin vs Soft margin
  - a lengthy series of experiments with various parameters tested

Support Vector Machines: Wrap-up

- Pros:
  - Models with strong theoretical foundations
  - Sparse solution for large dataset: only support vectors are used to specify the separating hyperplane
  - Handles large feature spaces: complexity does not depend on its dimensionality
  - Overfitting is controlled by soft margin
  - A simple convex optimization problem which is guaranteed to converge to a single global solution

Support Vector Machines: Wrap-up

- Cons:
  - Original technique can only deal with binary classification tasks
  - Very sensitive to hyper-parameter values
  - With large number of support vectors, complexity (storage+computation) is high
  - Produces black-box models

References

References

Aggarwal, Charu C. 2015. *Data Mining, the Texbook*. Springer.

Gama, João, André Carlos Ponce de Leon Ferreira de Carvalho, Katti Faceli, Ana Carolina Lorena, and Márcia Oliveira. 2015. *Extração de Conhecimento de Dados: Data Mining -3rd Edition*. Edições Sílabo.

Han, Jiawei, Micheline Kamber, and Jian Pei. 2011. *Data Mining: Concepts and Techniques*. 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Moreira, João, Andre Carvalho, and Tomás Horvath. 2018. *Data Analytics: A General Introduction*. Wiley.

Smola, Alex J., and Bernhard Schölkopf. 2004. "A Tutorial on Support Vector Regression." *Statistics and Computing* 14 (3): 199–222.

Tan, Pang-Ning, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. 2018. *Introduction to Data Mining*. 2nd ed. Pearson.