Machine learning: basi e sue applicazioni

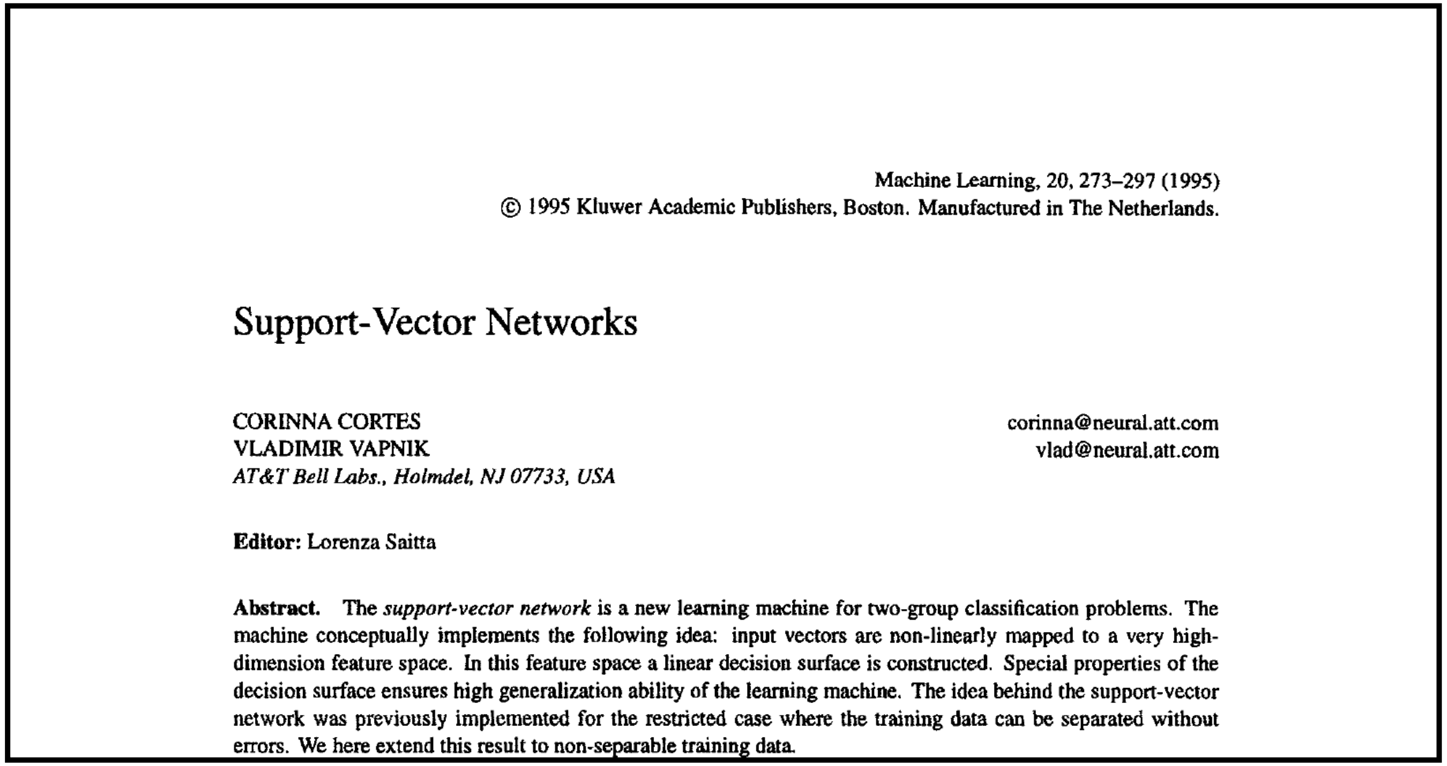
Christian Salvatore
Scuola Universitaria Superiore IUSS Pavia

christian.salvatore@iusspavia.it

CLASSIFICATION

SUPPORT VECTOR MACHINE

Support Vector Machine



Machine Learning, 20, 273–297 (1995)
© 1995 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Support-Vector Networks

CORINNA CORTES corinna@neural.att.com
VLADIMIR VAPNIK vlad@neural.att.com
AT&T Bell Labs., Holmdel, NJ 07733, USA

Editor: Lorenza Saitta

Abstract. The *support-vector network* is a new learning machine for two-group classification problems. The machine conceptually implements the following idea: input vectors are non-linearly mapped to a very high-dimension feature space. In this feature space a linear decision surface is constructed. Special properties of the decision surface ensures high generalization ability of the learning machine. The idea behind the support-vector network was previously implemented for the restricted case where the training data can be separated without errors. We here extend this result to non-separable training data.

SVM is a binary classifier which aims at generating a predictive model for the discrimination of new samples.

Support Vector Machine

Theory introduced by Vapnik (statistical learning theory) in 1965.
Improved later by Vapnik himself and others in 1995.

Vapnik suggests determining decision surfaces between classes (classification boundaries) to separate them.

SVM was born as a binary classifier (2 classes), with different degrees of complexity:

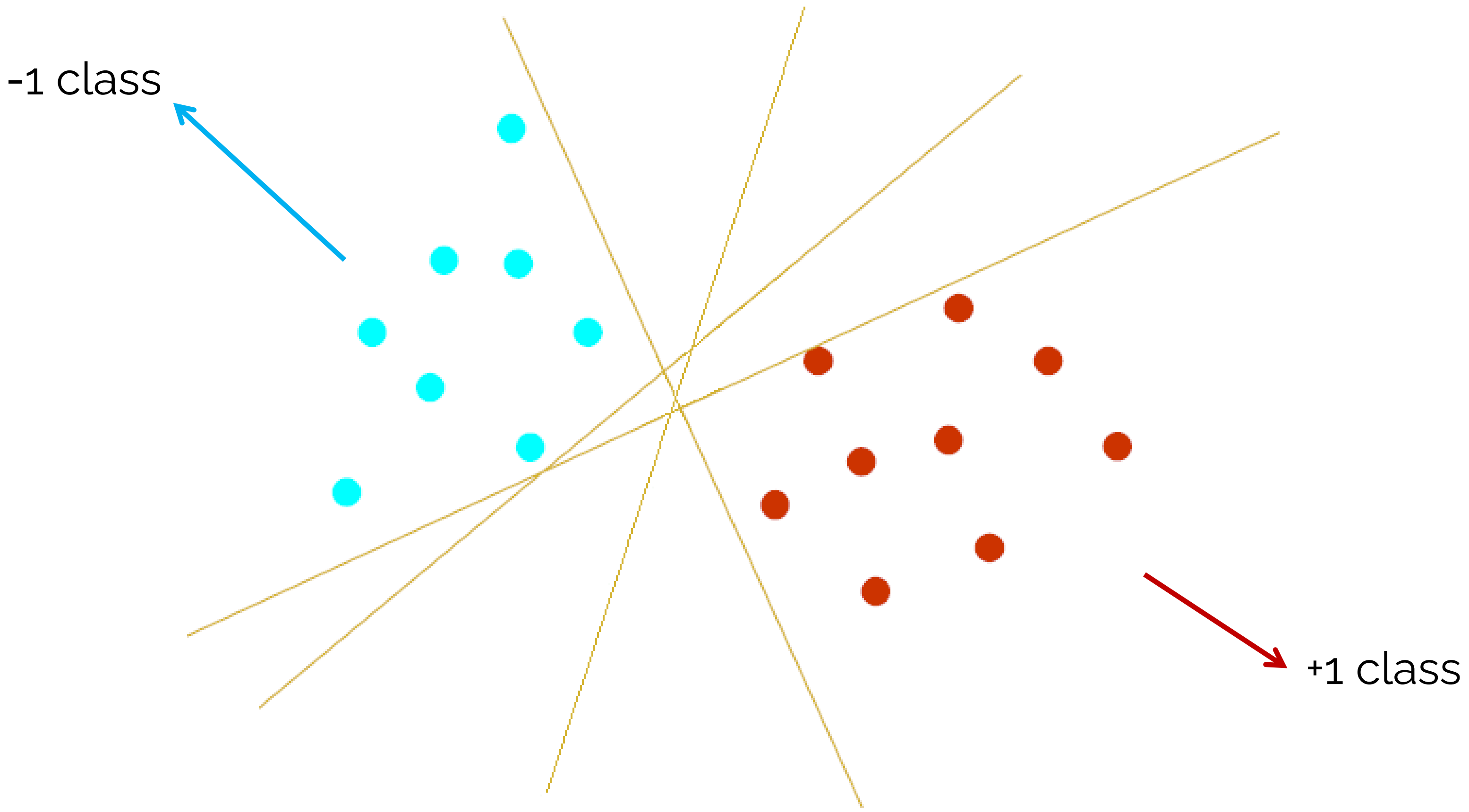
- Linear SVM (i.e., the separation surface is a hyperplane) and the training set patterns are linearly separable (i.e., there is by hypothesis at least one hyperplane capable of separating them).

- Linear SVM and non-linearly separable patterns. There are no classification errors hyperplane capable of separating patterns.

- Non-linear SVM (i.e., complex separation surface) without hypothesis on the separability of pattern.

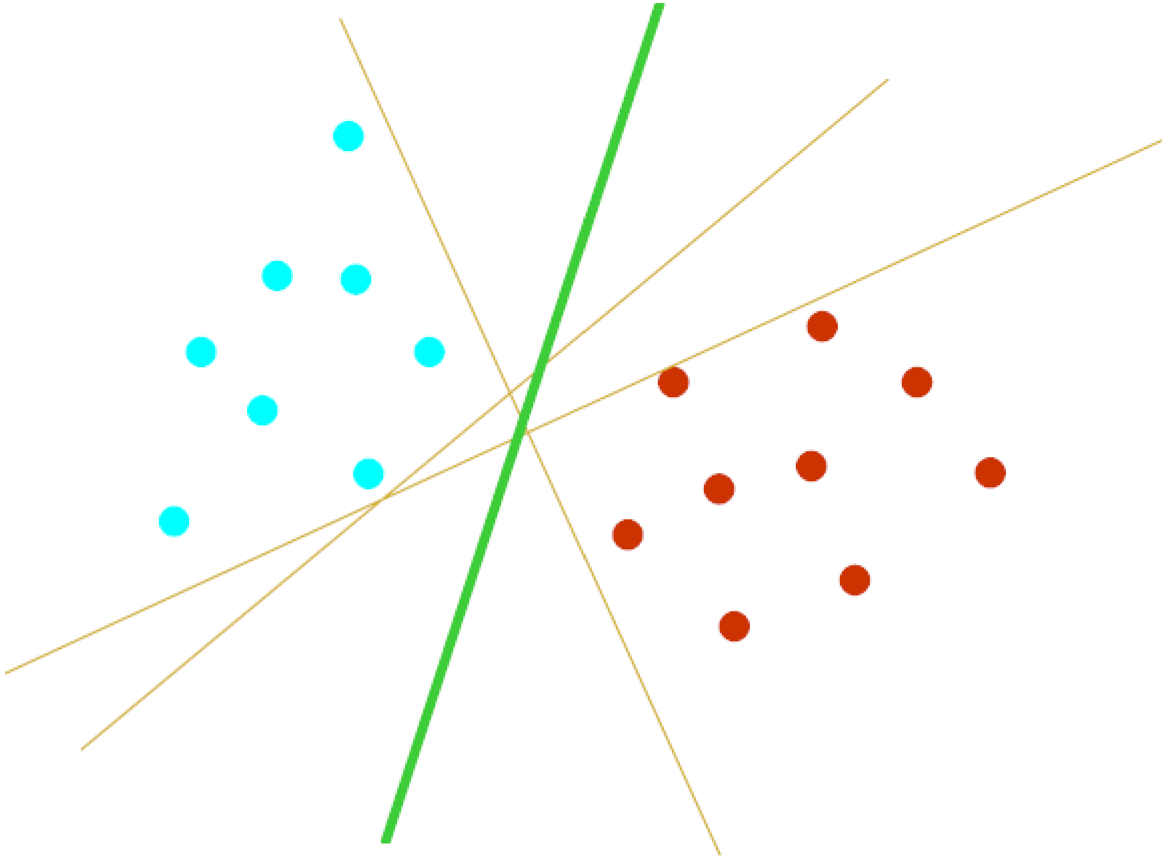
- SVM extension to multi-class.

Support Vector Machine



Let us suppose to have a set of training data consisting of a vector of N samples belonging to two classes and the corresponding vector of class labels (e.g. -1 and +1 for control and patient class, respectively)

Support Vector Machine



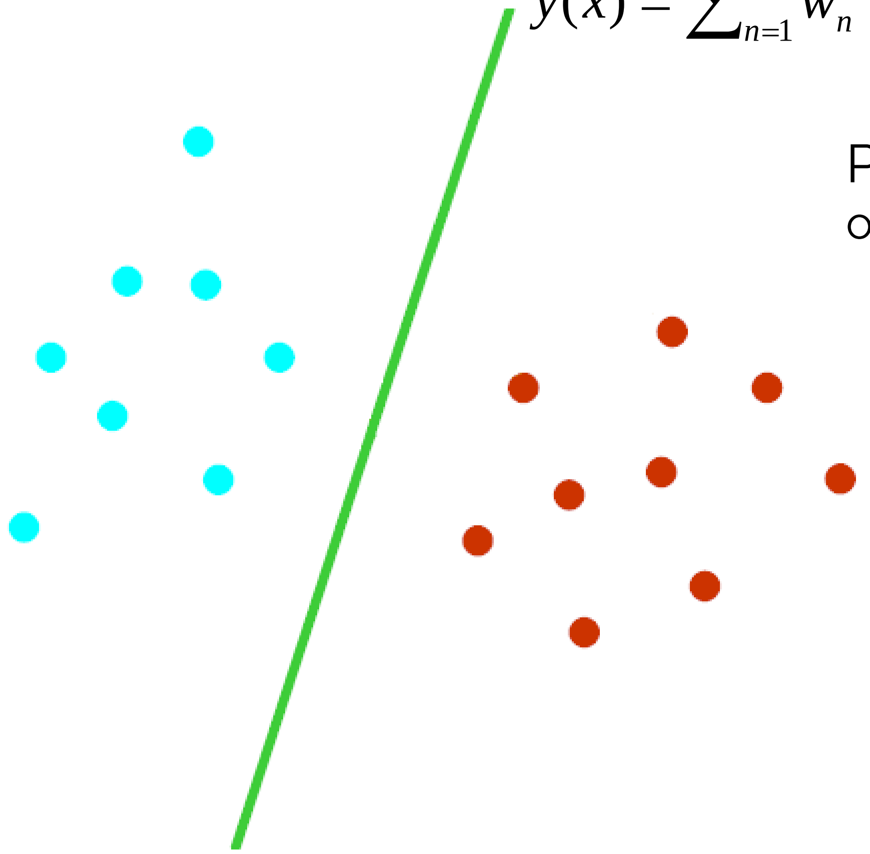
SVM generates a hyper-plane able to discriminate between the two classes of training dataset

Support Vector Machine

$$y(x)=\sum_{n=1}^{N} w_n \cdot t_n \cdot k(x,x_n)+b$$

=

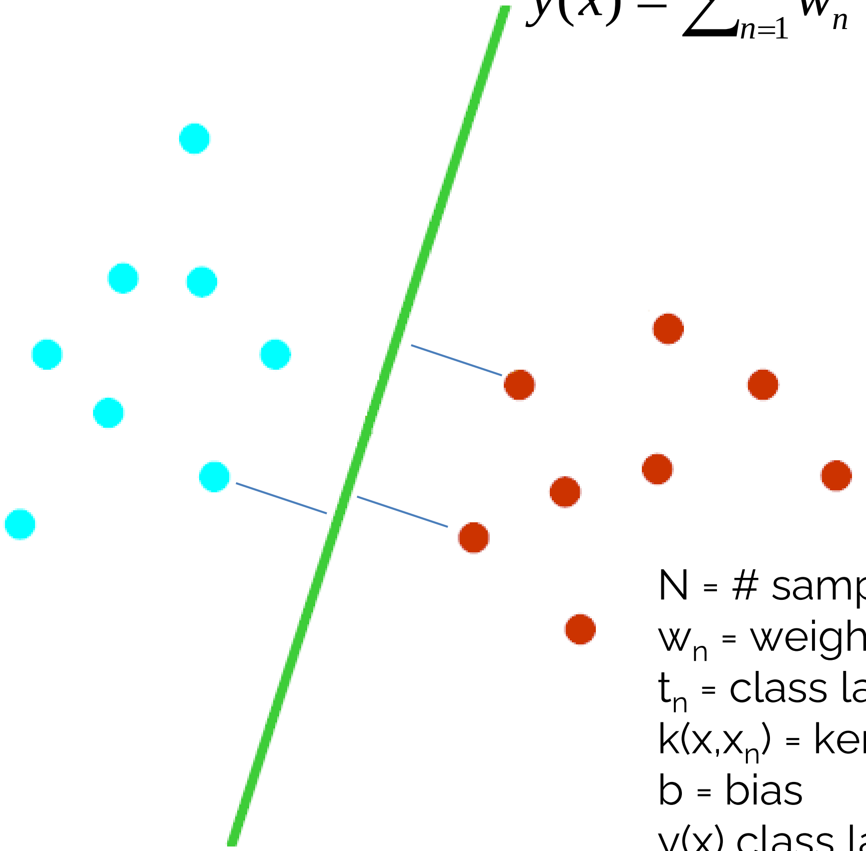
Predictive model
or decision function



SVM generates a hyper-plane able to discriminate between the two classes of training dataset

Support Vector Machine

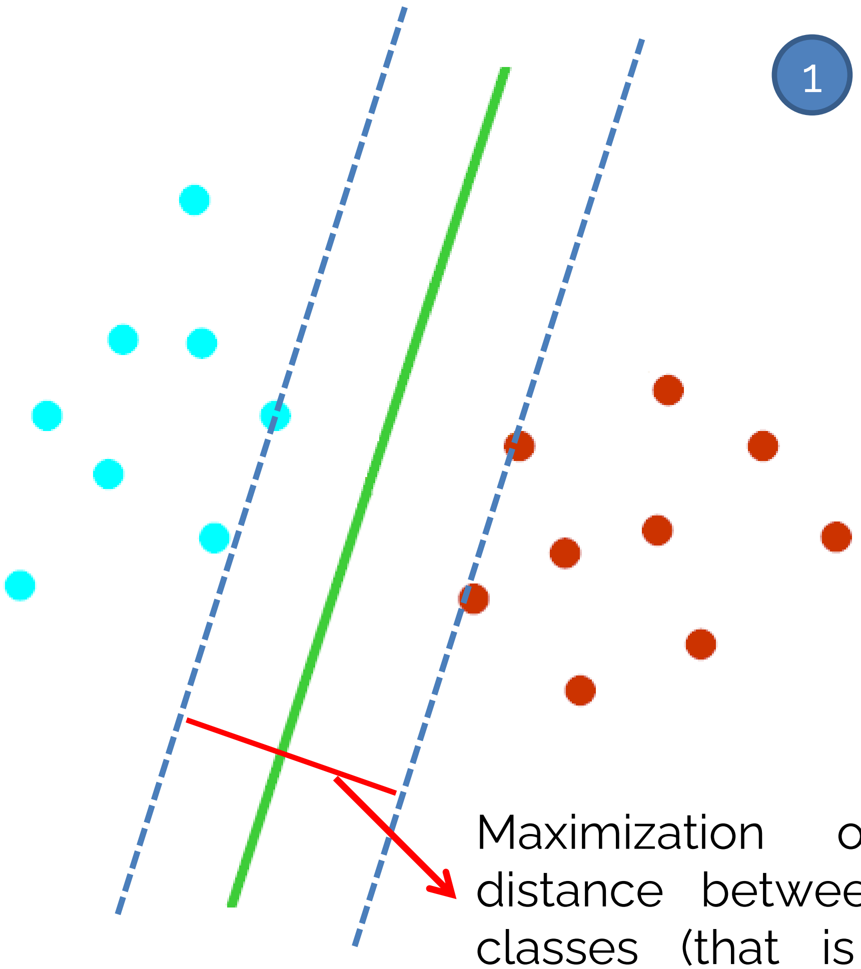
$$y(x) = \sum_{n=1}^{N} w_n \cdot t_n \cdot k(x, x_n) + b$$



N = # samples in the training dataset
w_n = weight assigned by SVM
t_n = class label of the training sample n
$k(x,x_n)$ = kernel function
b = bias
y(x) class label for unseen sample x

The weight assigned by SVM during the training phase to each training sample reflects its importance for classes discrimination

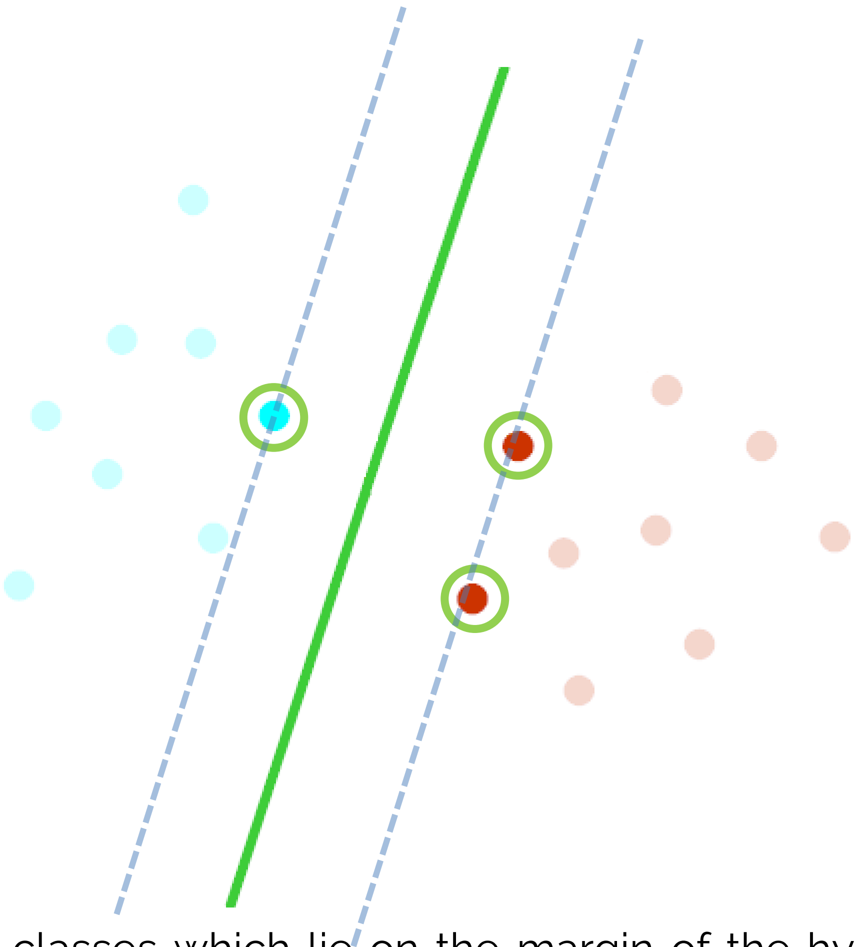
Support Vector Machine



1 Minimization of the empirical classification error

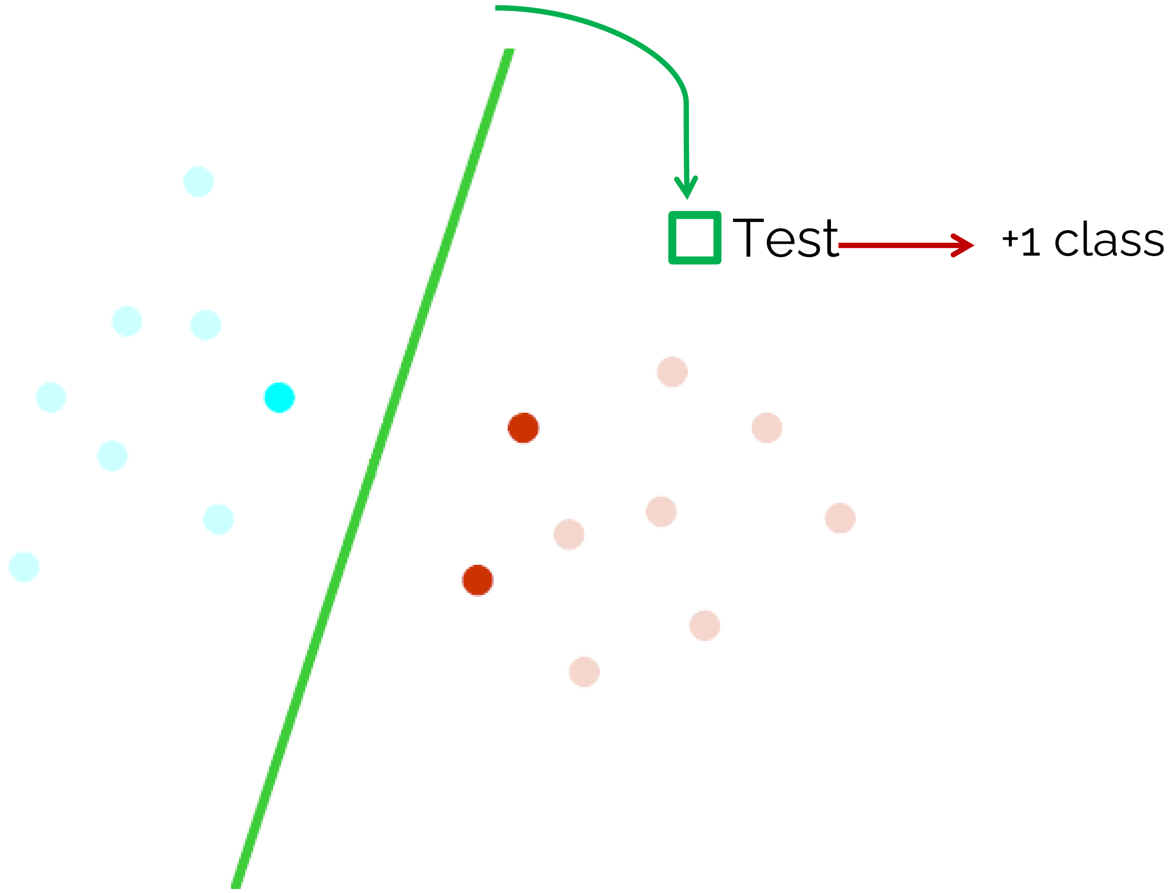
2 Maximization of the separation distance between the two training classes (that is why the decision function is often called maximum margin hyper-plane)

Support Vector Machine



Samples of the two classes which lie on the margin of the hyper-plane are called support vector, and they are the only samples for which the assigned weight w is non-zero.

Support Vector Machine



Once the hyper-plane is defined, the model is able to predict the class label for unseen samples.

Support Vector Machine | Proprietà degli Iperpiani

$D(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \cdot \mathbf{x} + b$

w: vettore normale all'iperpiano

$b / ||\mathbf{w}||$: distanza dall'origine

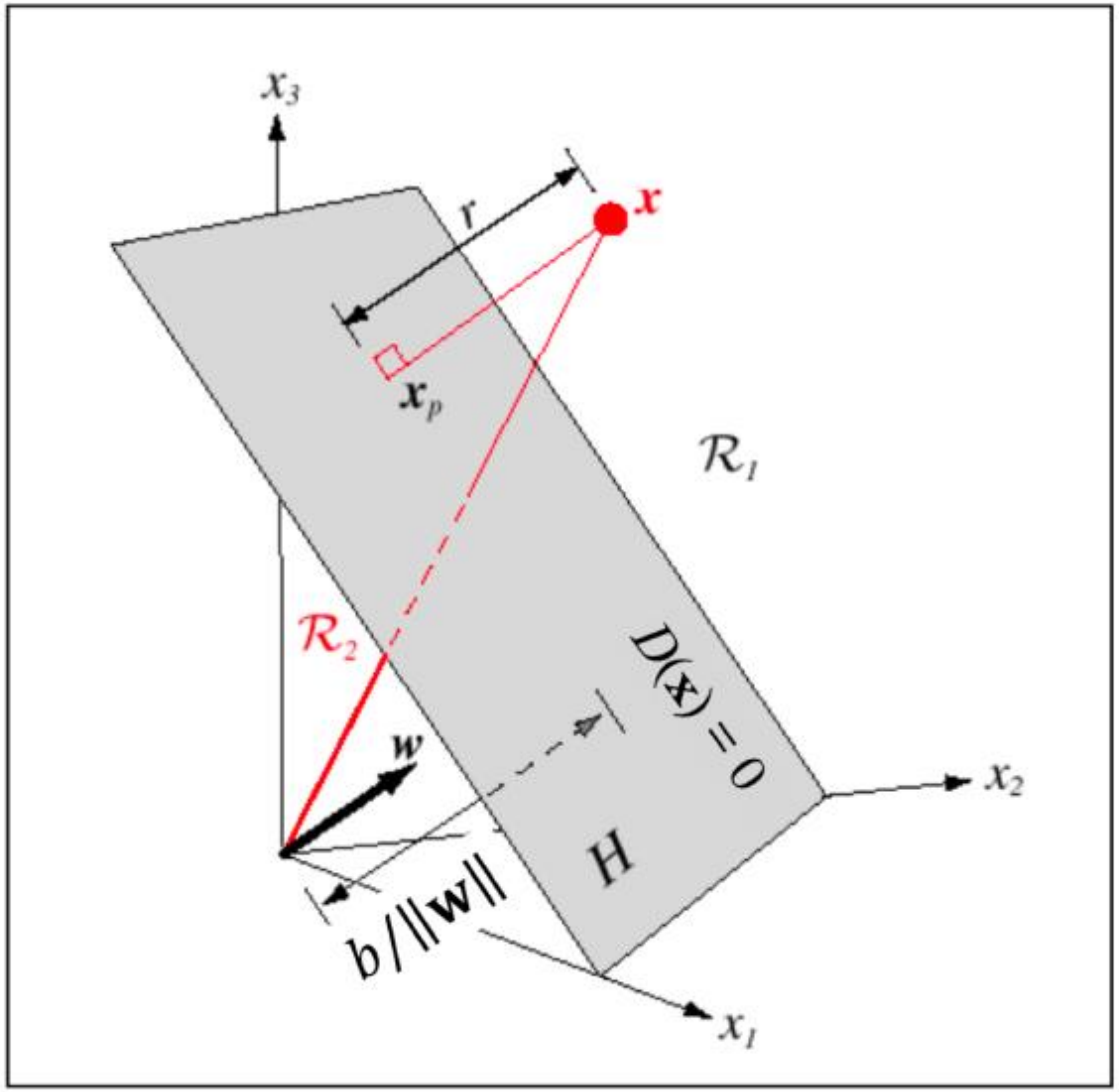
$D(\mathbf{x}) = 0$: luogo dei vettori sull'iperpiano

$D(\mathbf{x}_{\mathrm{p}}) = 0$

$\mathbf{x} = \mathbf{x}_p + \mathrm{r}\ \mathbf{w} / ||\mathbf{w}||$

$D(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \cdot \mathbf{x} + b = \mathrm{r}\ ||\mathbf{w}||$

distanza di un vettore **x** dall'iperpiano r = $D(\mathbf{x}) / ||\mathbf{w}||$



Support Vector Machine

Imagine we are given N observations, each one consisting of a pair: an input vector $x_n \in R^N$, n = 1, ...N and the corresponding target value $t_n \in \{\pm 1\}$, given to us by a trusted source. Data are assumed to be *iid*, i.e., independently drawn from an unknown probability distribution P(x; t) and identically distributed. In our example, $x_n \in R^N$ might be a vector of pixel values (representing images of patients) and $t_n \in \{\pm 1\}$ would be +1 for images belonging to the feature class, -1 for images belonging to the normal class. The aim is to estimate a function that will correctly classify unseen examples (x; t), i.e., we want f(x) = t for samples (x; t) that were also generated from P(x; t). This problem reduces to the goal of separating the two classes by a function which is induced from available examples, in order to produce a classifier, based on input-output data training, which will work well on unseen samples. Let us now make the further assumption that the training data set is linearly separable in feature space and consider the example in Figure 1.3.1: here there are many possible linear classifiers that can separate the data, but there is only one that maximizes the margin (i.e., the distance between it and the nearest data point of each class). This classifier is termed the optimal separating hyper-plane. Intuitively, we would expect this boundary to generalize well as opposed to the other possible boundaries (Figure 1.3.1).

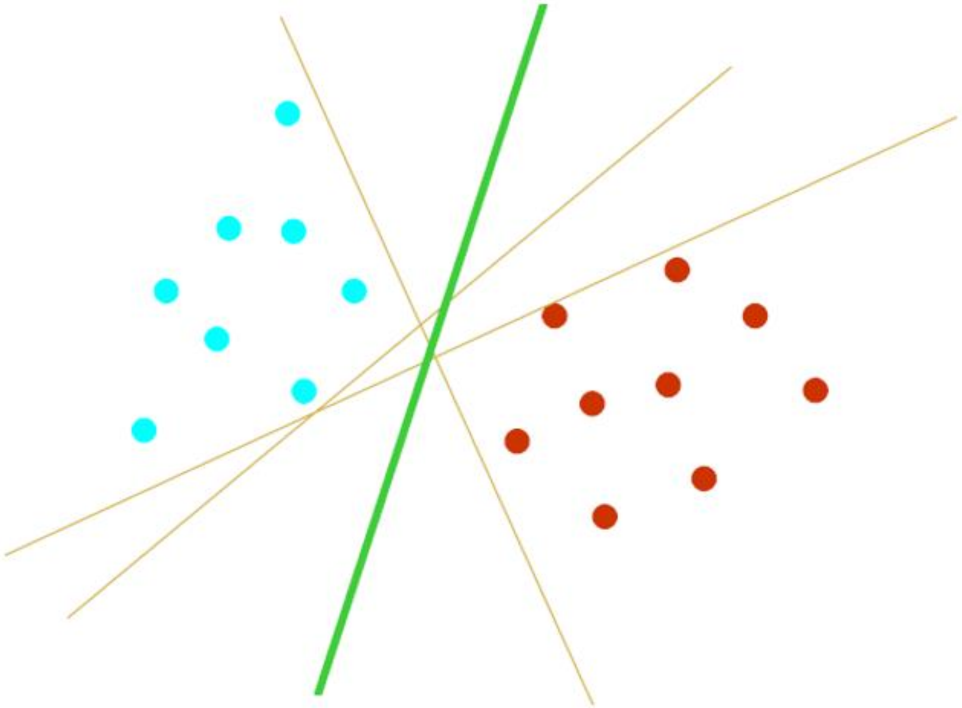


Figure 1.3.1 *Optimal separating hyper-plane*

Support Vector Machine

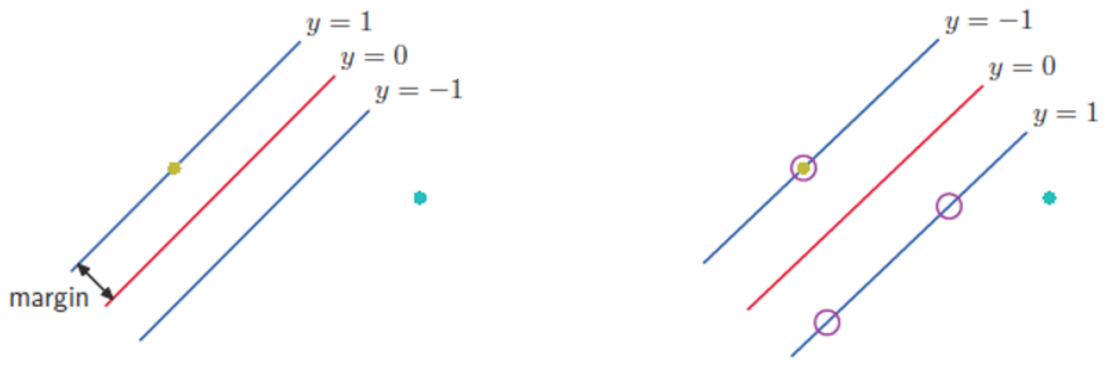


***Figure 1.3.2** Margin and support vectors*

If we consider the following class of hyper-planes (linear model)

$$y(x) = \omega^T \varphi(x) + b \qquad (1.3.1)$$

Where $\varphi(x)$ denotes a fixed feature-space transformation and the explicit parameter b is called bias parameter, then new data points x can be classified according to the sign of $y(x)$. In fact, as we assumed the training data set to be linearly separable in feature space, by definition there exists at least one choice of the parameters w and b such that a function of the form 1.3.1 satisfies $y(x_n) > 0$ for points having $t_n = +1$ and $y(x_n) < 0$ for points having $t_n = -1$. Furthermore, $t_n \cdot y(x_n) > 0$ for all training data points. As we said before, there may exist many such solutions that separate the classes exactly. This problem can be solved by introducing the concept of the margin, which is defined to be the smallest distance between the decision boundary and any of the samples, as illustrated in Figure 1.3.2.

In SVM, the decision boundary is chosen to be the one for which the margin is maximized. The maximum margin solution can be motivated using computational learning theory, also known as statistical learning theory or VC (Vapnik-Chervonenkis) theory.

Support Vector Machine

It must be noted that the margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left part of Figure 1.3.2. Maximizing the margin leads to a particular choice of decision boundary, as shown on the right part of Figure 1.3.2. The location of this boundary is determined by a subset of the data points, known as support vectors, which are indicated by the circles. The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left part of Figure 1.3.2. Maximizing the margin leads to a particular choice of decision boundary, as shown on the right of Figure 1.3.2. The location of this boundary is determined by a subset of the data points, known as support vectors, which are indicated by the circles.

In general, the perpendicular distance of a point x from a hyper-plane defined by y(x) = 0, where y(x) takes the form specified in equation 1.3.1, is given by $|y(x)|/||w||$. Furthermore, in this case we are only interested in solutions for which all data points are correctly classified, so that $t_n \cdot y(x_n) > 0$ for all n. Thus the distance of a point x_n to the decision surface is given by

$$\frac{t_n \cdot y(x_n)}{\|w\|} = \frac{t_n \cdot (\omega^T \varphi(x) + b)}{\|w\|} \qquad (1.3.2)$$

Support Vector Machine

The margin is given by the perpendicular distance to the closest point x_n from the data set, and we wish to optimize the parameters w and b in order to maximize this distance. Thus the maximum margin solution is found by solving

$$argmax_{W,b}\left\{\frac{1}{||w||}min_n[t_n\cdot(\omega^T\varphi(x)+b)]\right\} \tag{1.3.3}$$

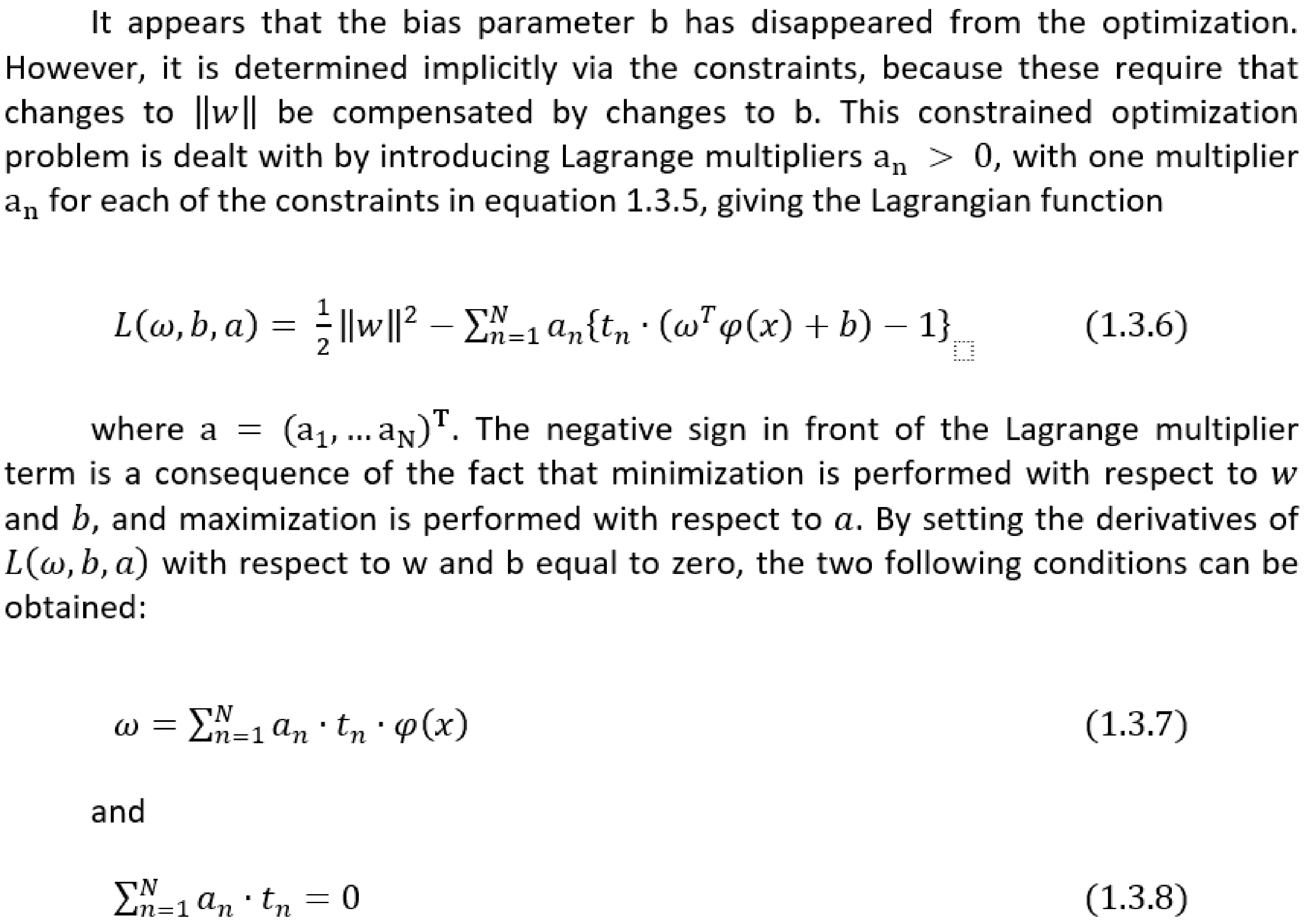
where we have taken the factor $\frac{1}{||w||}$ outside the optimization over n because w does not depend on n. Solving this optimization problem simply reduces to the requirement of minimizing $||w||^2$

$$argmax_{W,b}\left\{\frac{1}{2}||w||^2\right\} \tag{1.3.4}$$

(the factor of $\frac{1}{2}$ is included for later convenience), subject to the constraints

$$t_n\cdot(\omega^T\varphi(x)+b\ \geq 1, n=1,\ldots N \tag{1.3.5}$$

Support Vector Machine



It appears that the bias parameter b has disappeared from the optimization. However, it is determined implicitly via the constraints, because these require that changes to $\|w\|$ be compensated by changes to b. This constrained optimization problem is dealt with by introducing Lagrange multipliers $\mathrm{a_n} > 0$, with one multiplier $\mathrm{a_n}$ for each of the constraints in equation 1.3.5, giving the Lagrangian function

$$L(\omega, b, a) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N} a_n\{t_n \cdot (\omega^T\varphi(x) + b) - 1\} \tag{1.3.6}$$

where $\mathrm{a} = (\mathrm{a_1}, \ldots \mathrm{a_N})^{\mathrm{T}}$. The negative sign in front of the Lagrange multiplier term is a consequence of the fact that minimization is performed with respect to w and b, and maximization is performed with respect to a. By setting the derivatives of $L(\omega, b, a)$ with respect to w and b equal to zero, the two following conditions can be obtained:

$$\omega = \sum_{n=1}^{N} a_n \cdot t_n \cdot \varphi(x) \tag{1.3.7}$$

and

$$\sum_{n=1}^{N} a_n \cdot t_n = 0 \tag{1.3.8}$$

Support Vector Machine

Eliminating w and b from $\mathrm{L}(\omega, \mathrm{b}, \mathrm{a})$ using these conditions, then, gives the dual representation of the maximum margin problem in which we maximize

$$\tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_m a_n t_m t_n k(x_n, x_m) \qquad (1.3.9)$$

with respect to the constraints

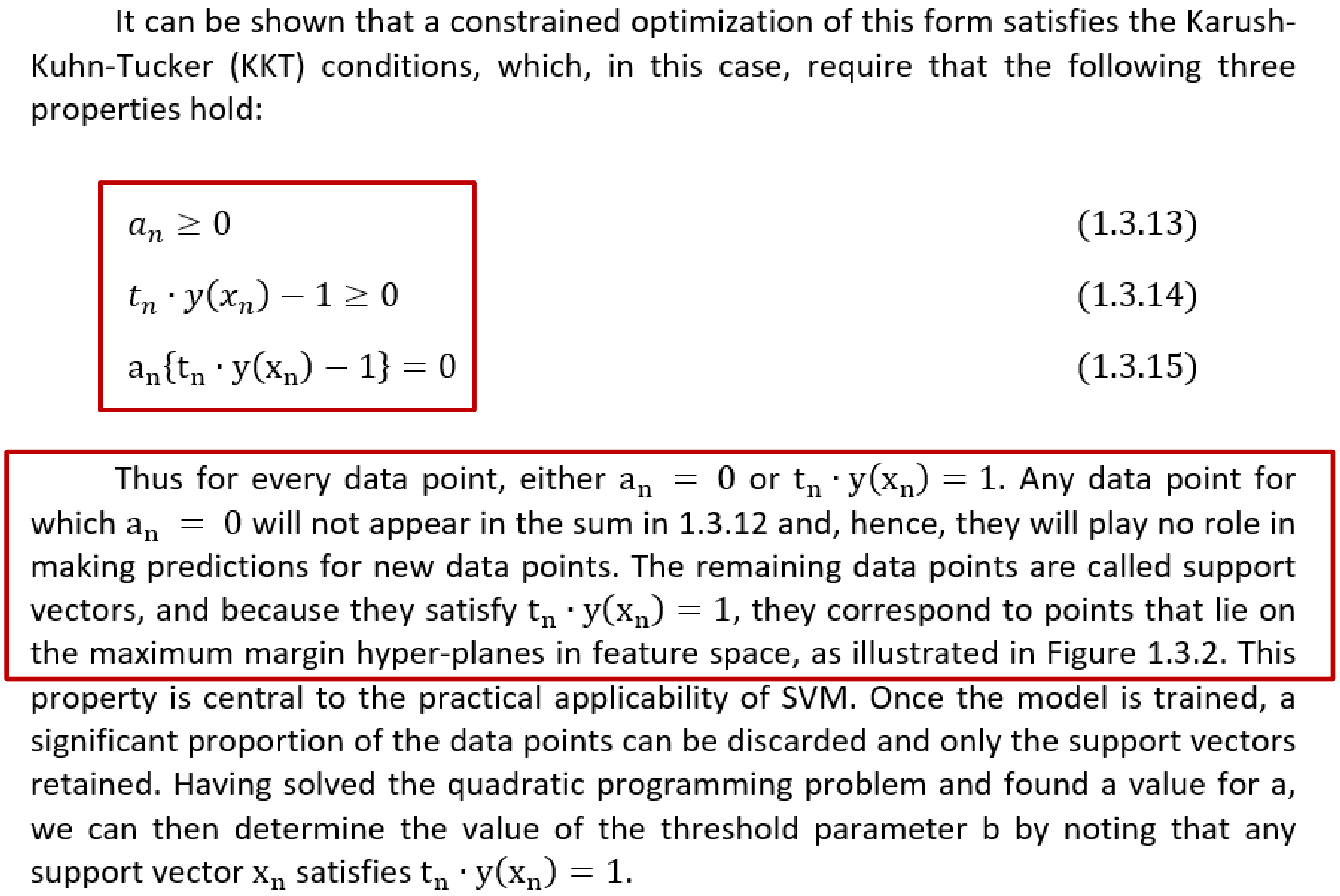
$$a_n \geq 0, n = 1, \dots N \qquad (1.3.10)$$

$$\sum_{n=1}^{N} a_n \cdot t_n = 0 \qquad (1.3.11)$$

Here, the kernel function is defined by $k(x, x') = \varphi(x)^T \cdot \varphi(x)'$: this kernel formulation makes clear the role of the constraint that the kernel function $\mathrm{k}(\mathrm{x}, \mathrm{x}')$ be positive definite, because this ensures that the Lagrangian function $\tilde{L}(a)$ is bounded below, giving rise to a well-defined optimization problem. In order to classify new data points using the trained model, we evaluate the sign of y(x) defined in equation 1.3.1. This can be expressed in terms of the parameters an and the kernel function by substituting for ω using 1.3.7 to give the hyper-plane decision function

$$\boxed{y(x) = \sum_{n=1}^{N} a_n \cdot t_n \cdot k(x, x_n) + b} \qquad (1.3.12)$$

Support Vector Machine



It can be shown that a constrained optimization of this form satisfies the Karush-Kuhn-Tucker (KKT) conditions, which, in this case, require that the following three properties hold:

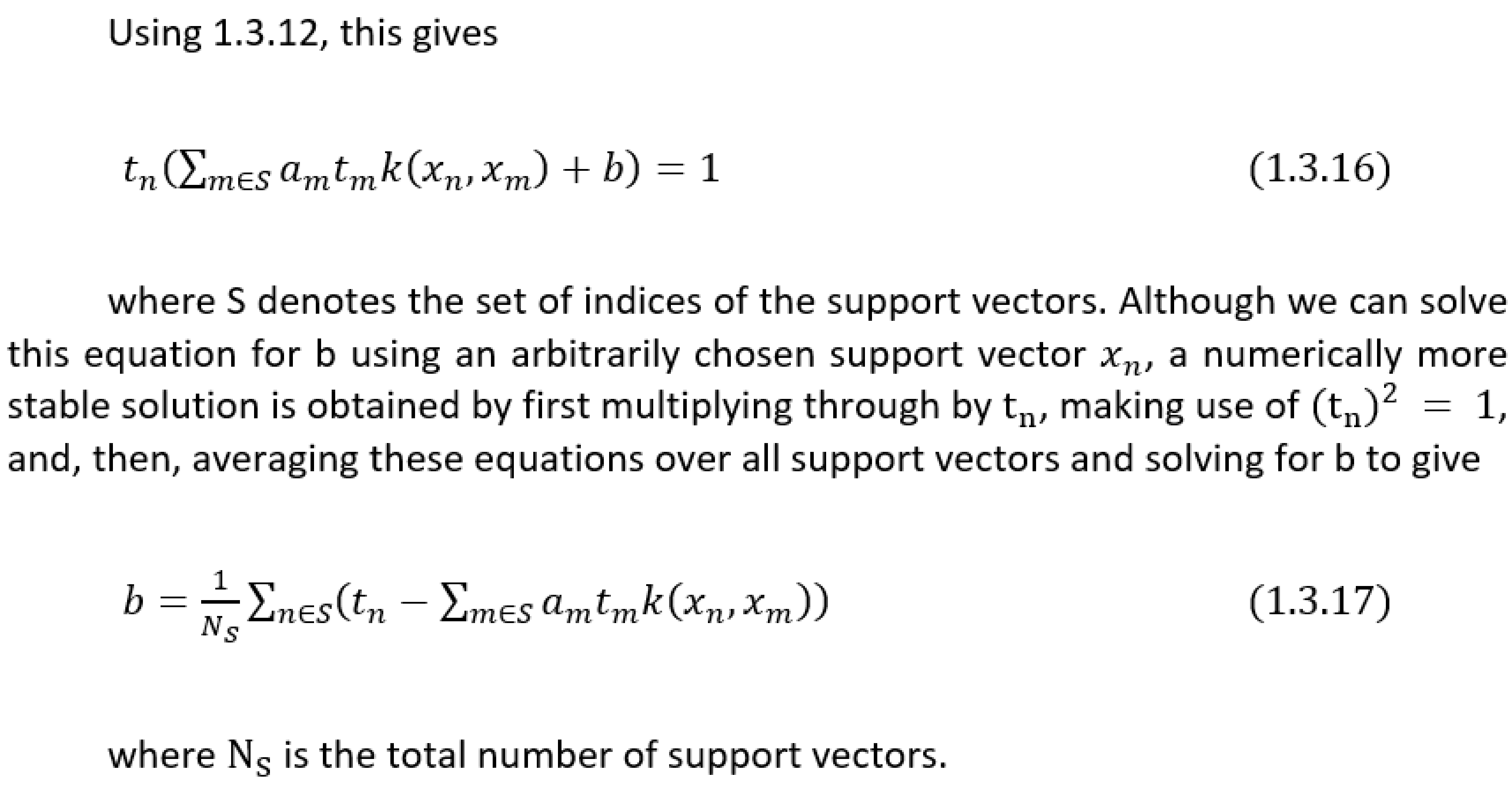
$$a_n \geq 0 \tag{1.3.13}$$

$$t_n \cdot y(x_n) - 1 \geq 0 \tag{1.3.14}$$

$$\mathrm{a_n\{t_n \cdot y(x_n) - 1\} = 0} \tag{1.3.15}$$

Thus for every data point, either $\mathrm{a_n} = 0$ or $\mathrm{t_n \cdot y(x_n) = 1}$. Any data point for which $\mathrm{a_n} = 0$ will not appear in the sum in 1.3.12 and, hence, they will play no role in making predictions for new data points. The remaining data points are called support vectors, and because they satisfy $\mathrm{t_n \cdot y(x_n) = 1}$, they correspond to points that lie on the maximum margin hyper-planes in feature space, as illustrated in Figure 1.3.2. This property is central to the practical applicability of SVM. Once the model is trained, a significant proportion of the data points can be discarded and only the support vectors retained. Having solved the quadratic programming problem and found a value for a, we can then determine the value of the threshold parameter b by noting that any support vector $\mathrm{x_n}$ satisfies $\mathrm{t_n \cdot y(x_n) = 1}$.

Support Vector Machine



Using 1.3.12, this gives

$$t_n\left(\sum_{m\in S} a_m t_m k(x_n, x_m) + b\right) = 1 \qquad (1.3.16)$$

where S denotes the set of indices of the support vectors. Although we can solve this equation for b using an arbitrarily chosen support vector x_n, a numerically more stable solution is obtained by first multiplying through by $\mathrm{t_n}$, making use of $(\mathrm{t_n})^2 = 1$, and, then, averaging these equations over all support vectors and solving for b to give

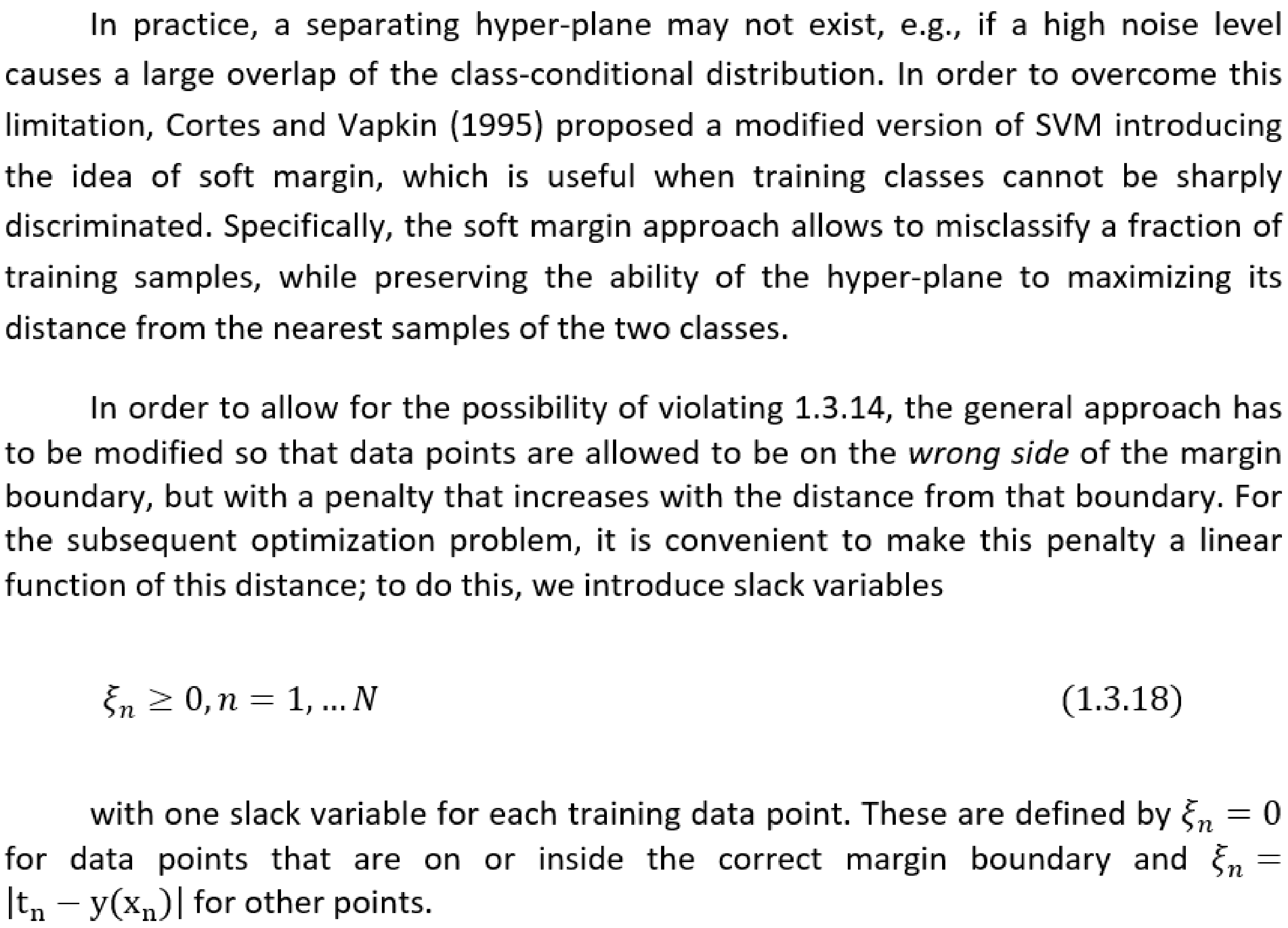
$$b = \frac{1}{N_S}\sum_{n\in S}\left(t_n - \sum_{m\in S} a_m t_m k(x_n, x_m)\right) \qquad (1.3.17)$$

where $\mathrm{N_S}$ is the total number of support vectors.

Support Vector Machine

In practice…

Support Vector Machine



In practice, a separating hyper-plane may not exist, e.g., if a high noise level causes a large overlap of the class-conditional distribution. In order to overcome this limitation, Cortes and Vapkin (1995) proposed a modified version of SVM introducing the idea of soft margin, which is useful when training classes cannot be sharply discriminated. Specifically, the soft margin approach allows to misclassify a fraction of training samples, while preserving the ability of the hyper-plane to maximizing its distance from the nearest samples of the two classes.

In order to allow for the possibility of violating 1.3.14, the general approach has to be modified so that data points are allowed to be on the *wrong side* of the margin boundary, but with a penalty that increases with the distance from that boundary. For the subsequent optimization problem, it is convenient to make this penalty a linear function of this distance; to do this, we introduce slack variables

$$\xi_n \geq 0, n = 1, \dots N \qquad (1.3.18)$$

with one slack variable for each training data point. These are defined by $\xi_n = 0$ for data points that are on or inside the correct margin boundary and $\xi_n = |\mathrm{t_n} - \mathrm{y(x_n)}|$ for other points.

Support Vector Machine

Thus, a data point that is on the decision boundary $y(x_n) = 0$ will have $\xi_n = 1$ and points with $\xi_n > 1$ will be misclassified. In this case, classification constraints relax to

$$\boxed{t_n \cdot y(x_n) \geq 1 - \xi_n, n = 1, \ldots N} \qquad (1.3.19)$$

in which the slack variables are constrained to satisfy $\xi_n \geq 0$. Data points for which $\xi_n = 0$ are correctly classified and are either on the margin or on the correct side of the margin. Points for which $0 < \xi_n \leq 1$ lie inside the margin, but on the correct side of the decision boundary, and those data points for which $\xi_n \geq 1$ lie on the wrong side of the decision boundary and are misclassified, as illustrated in Figure 1.3.3. This is sometimes described as relaxing the hard margin constraint to give a soft margin and it allows some of the training set data points to be misclassified. Note that while slack variables allow for overlapping class distributions, this framework is still sensitive to outliers, because the penalty for misclassification increases linearly with ξ. In order to realize a soft margin classifier, we now have to minimize

$$\frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n \qquad (1.3.20)$$

subject to the constraints 1.3.18 and 1.3.19; here, the parameter C > 0 controls the trade-off between the slack variable penalty and the margin.

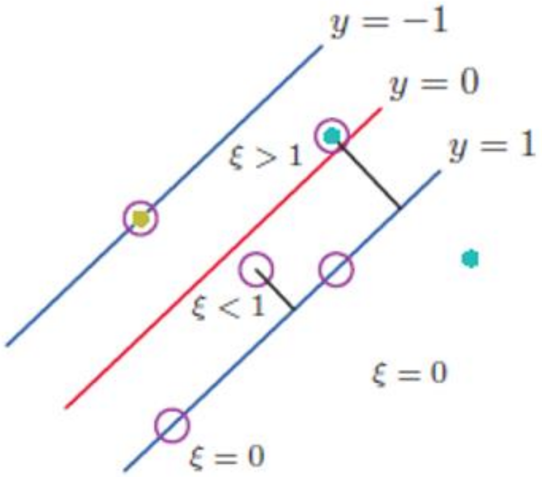


Figure 1.3.3 *Slack variables in support vector classifiers objective function. Data points with circles around them are support vectors.*

Support Vector Machine

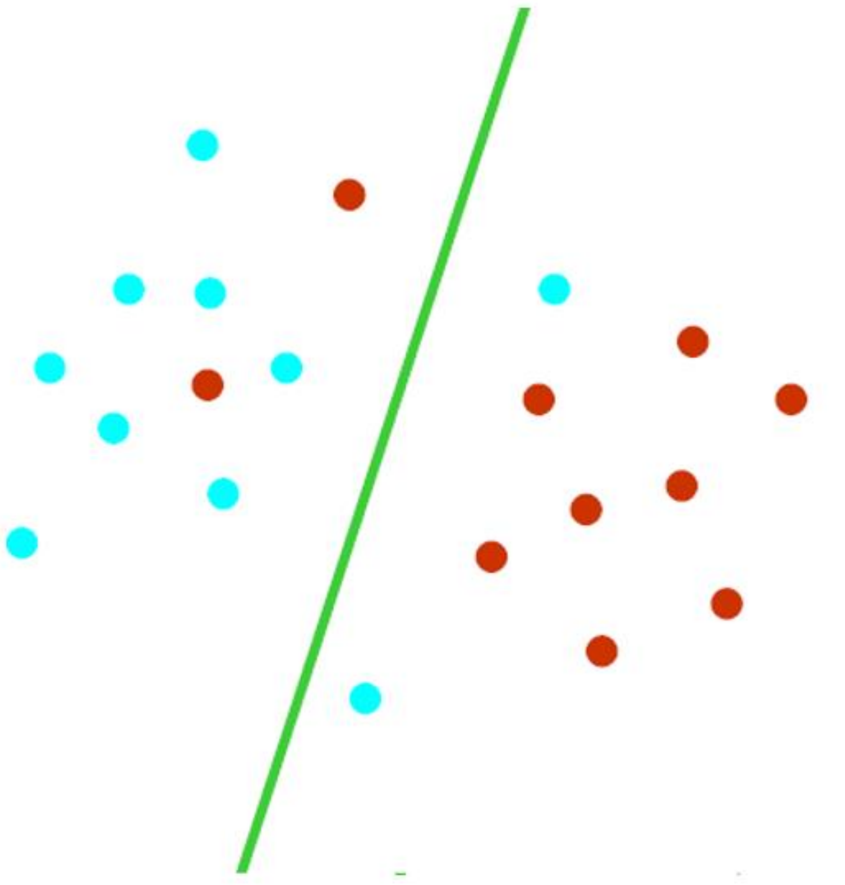
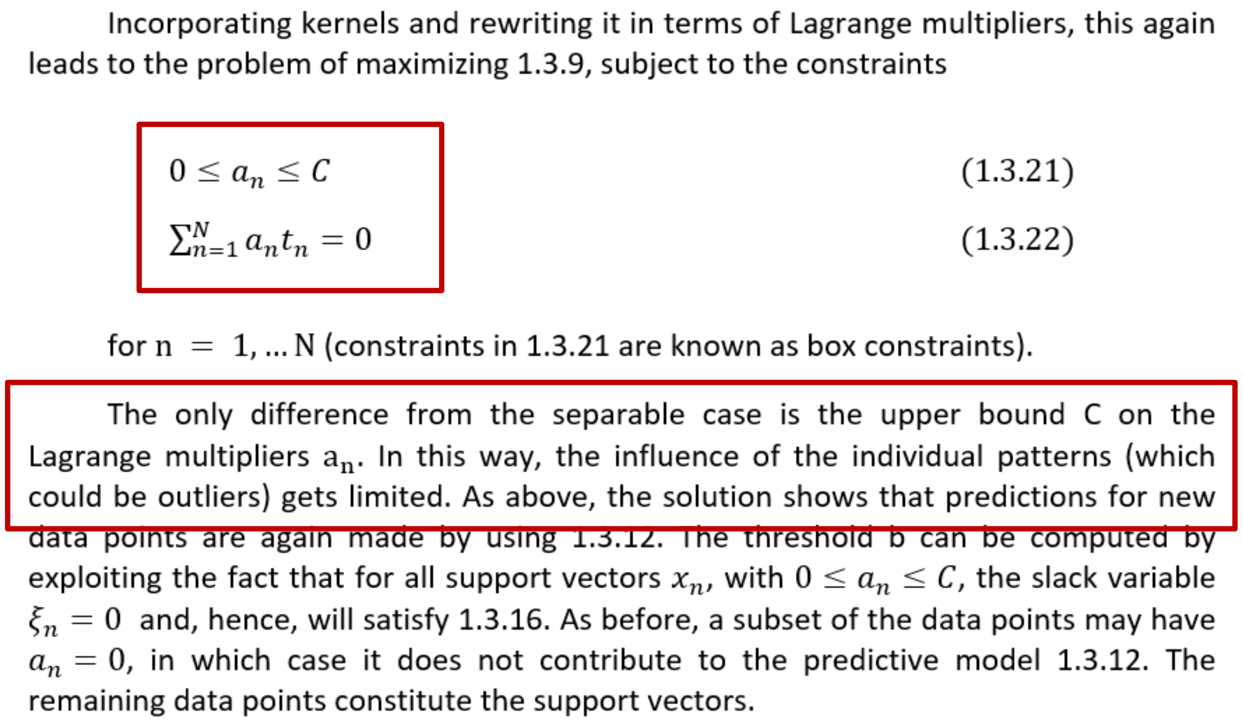


Figure 1.3.4 Generalized optimal separating hyper-plane



Incorporating kernels and rewriting it in terms of Lagrange multipliers, this again leads to the problem of maximizing 1.3.9, subject to the constraints

$$0 \le a_n \le C \tag{1.3.21}$$

$$\sum_{n=1}^{N} a_n t_n = 0 \tag{1.3.22}$$

for $n = 1, \ldots N$ (constraints in 1.3.21 are known as box constraints).

The only difference from the separable case is the upper bound C on the Lagrange multipliers a_n. In this way, the influence of the individual patterns (which could be outliers) gets limited. As above, the solution shows that predictions for new data points are again made by using 1.3.12. The threshold b can be computed by exploiting the fact that for all support vectors x_n, with $0 \le a_n \le C$, the slack variable $\xi_n = 0$ and, hence, will satisfy 1.3.16. As before, a subset of the data points may have $a_n = 0$, in which case it does not contribute to the predictive model 1.3.12. The remaining data points constitute the support vectors.

Support Vector Machine

In practice, when dealing with a SVM-based classifier, the main parameter to be set for a study is the kernel function. The most popular kernels in literature are the linear, the polynomial and the Gaussian Radial Basis Function (RBF) kernels:

$$k(x_i, x_j) = (x_i \cdot x_j) \tag{1.3.23}$$

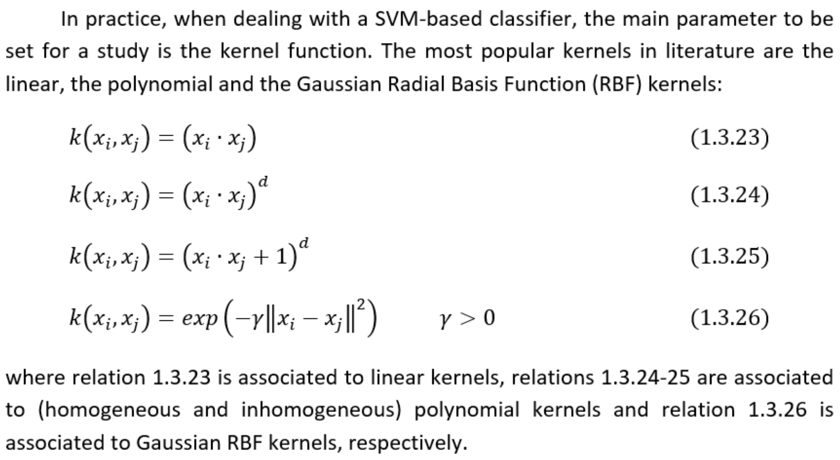
$$k(x_i, x_j) = (x_i \cdot x_j)^d \tag{1.3.24}$$

$$k(x_i, x_j) = (x_i \cdot x_j + 1)^d \tag{1.3.25}$$

$$k(x_i, x_j) = exp\left(-\gamma \|x_i - x_j\|^2\right) \qquad \gamma > 0 \tag{1.3.26}$$

where relation 1.3.23 is associated to linear kernels, relations 1.3.24-25 are associated to (homogeneous and inhomogeneous) polynomial kernels and relation 1.3.26 is associated to Gaussian RBF kernels, respectively.

Support Vector Machine



In practice, when dealing with a SVM-based classifier, the main parameter to be set for a study is the kernel function. The most popular kernels in literature are the linear, the polynomial and the Gaussian Radial Basis Function (RBF) kernels:

$$k(x_i, x_j) = (x_i \cdot x_j) \qquad (1.3.23)$$

$$k(x_i, x_j) = (x_i \cdot x_j)^d \qquad (1.3.24)$$

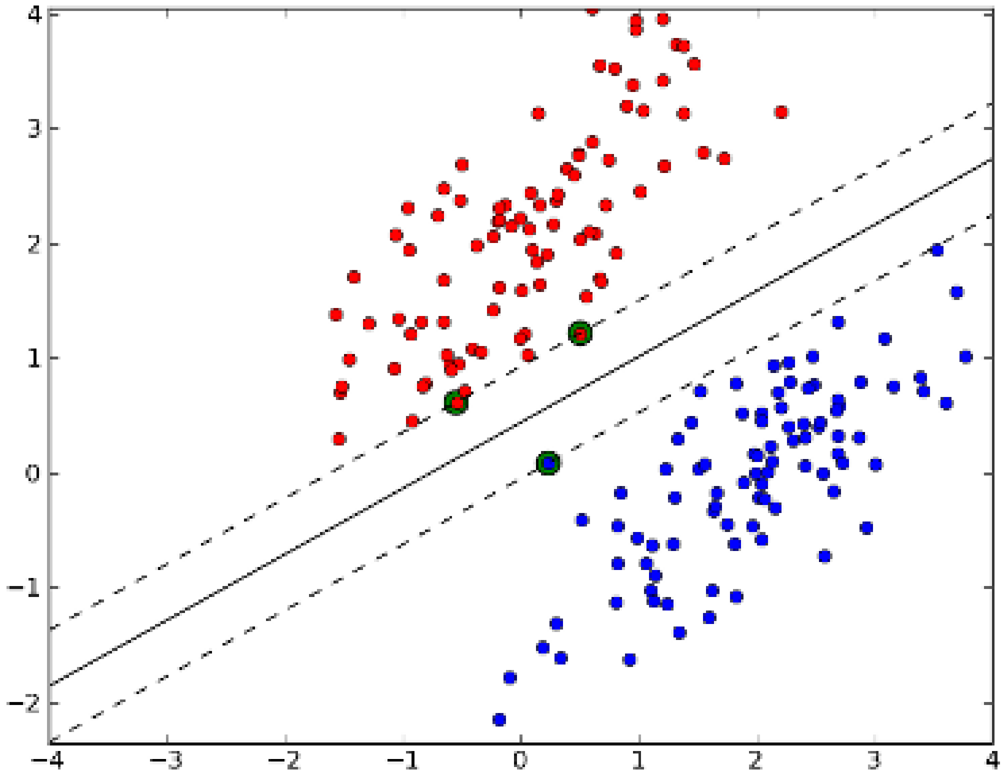
$$k(x_i, x_j) = (x_i \cdot x_j + 1)^d \qquad (1.3.25)$$

$$k(x_i, x_j) = exp\left(-\gamma \|x_i - x_j\|^2\right) \qquad \gamma > 0 \qquad (1.3.26)$$

where relation 1.3.23 is associated to linear kernels, relations 1.3.24-25 are associated to (homogeneous and inhomogeneous) polynomial kernels and relation 1.3.26 is associated to Gaussian RBF kernels, respectively.

The main parameter to be set in a SVM classifier is the kernel function, which is mostly set as linear, polynomial or Gaussian Radial Basis Function (RBF). Linear kernels are defined as

$$k(x_i, x_j) = (x_i \cdot x_j)$$

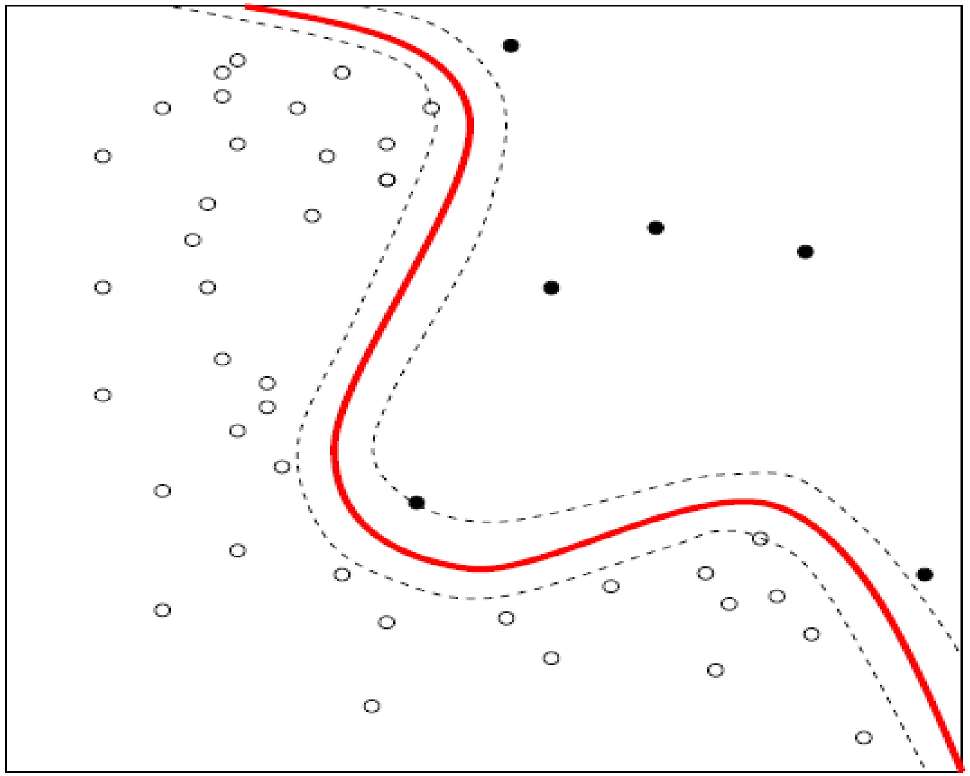


Support Vector Machine

while (homogeneous and inhomogeneous) polynomial kernels are given by

$$k(x_i, x_j) = (x_i \cdot x_j)^d$$

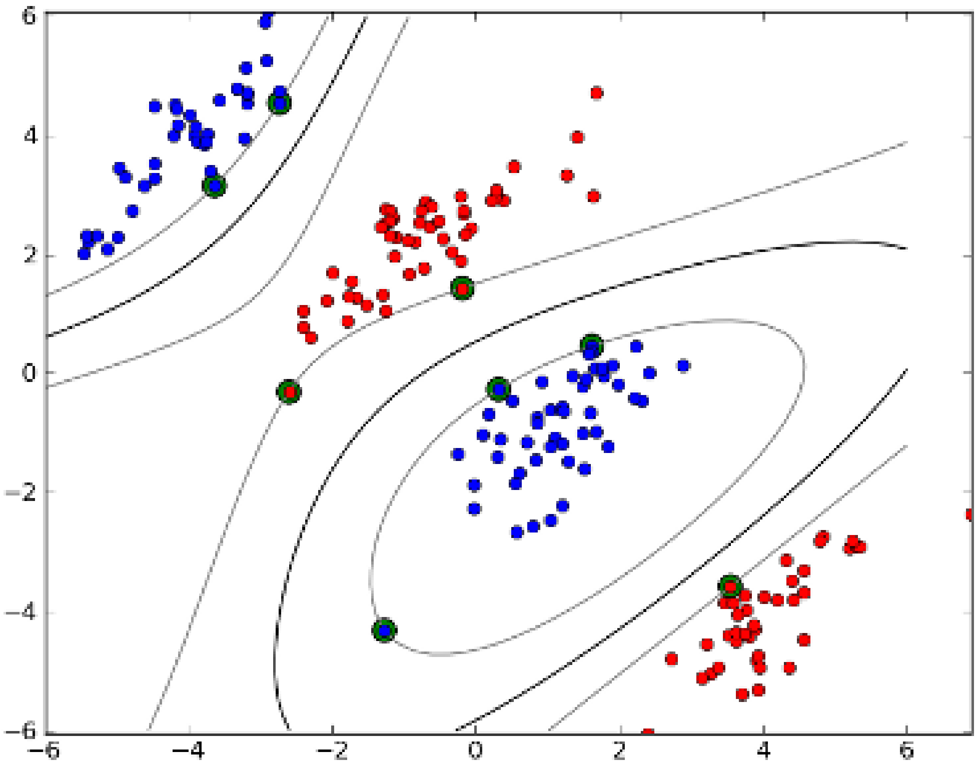
$$k(x_i, x_j) = (x_i \cdot x_j + 1)^d$$



Support Vector Machine

and Gaussian RBF kernels
have the following form

$$k(x_i, x_j) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right) \qquad \gamma > 0$$



Support Vector Machine | C HYPERPARAMETER

The C parameter is related to “how much you want to avoid misclassification” of each training example

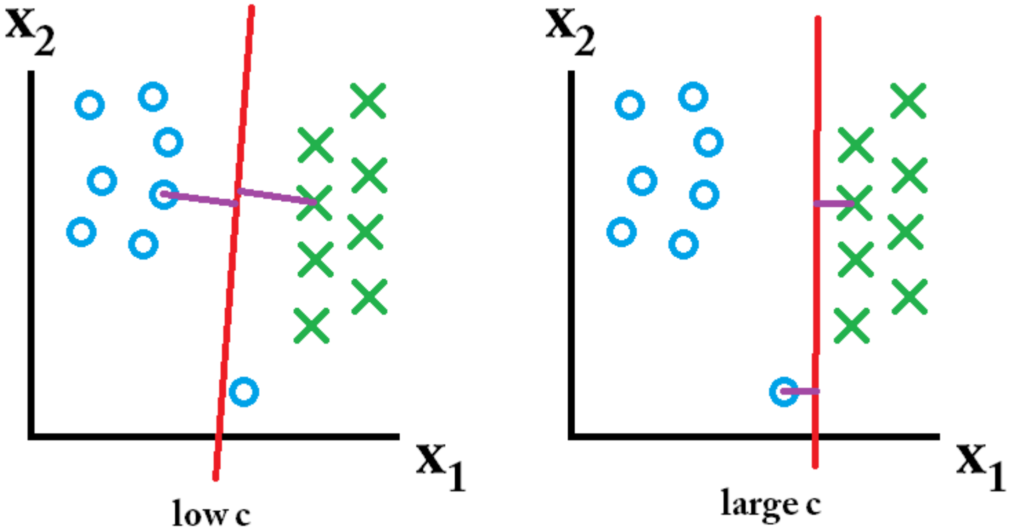
LARGE VALUES of C correspond to a smaller-margin separating hyperplane if that hyperplane correctly classifies all training data

SMALL VALUES of C correspond to a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

CONSIDER THAT an optimal C parameter should be larger when you scale down your data, smaller when data are not scaled

C	PENALIZATION	MARGIN	MODEL COMPLEXITY
↑	↑	↓	↑
↓	↓	↑	↓

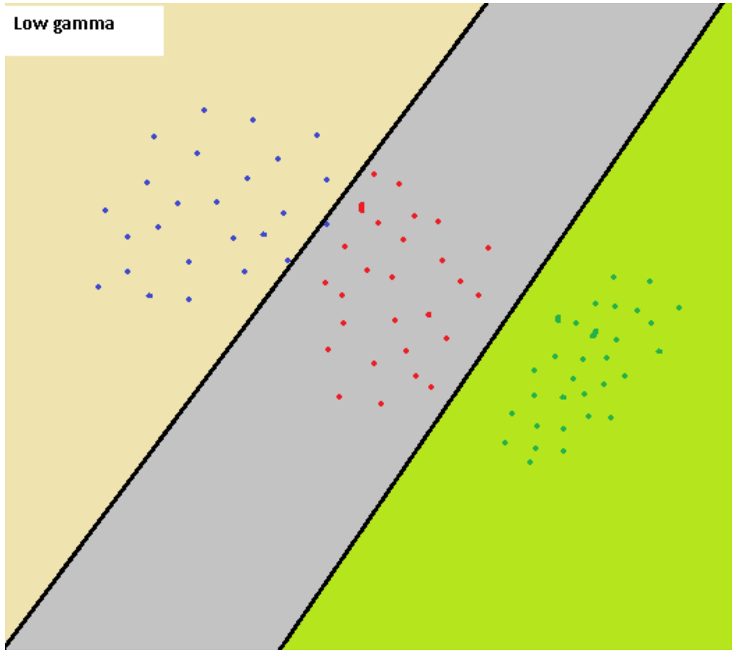
Support Vector Machine | C HYPERPARAMETER

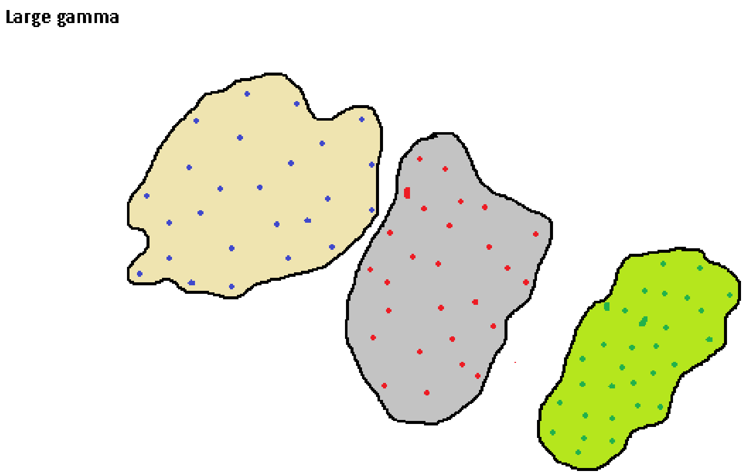


C	PENALIZATION	MARGIN	MODEL COMPLEXITY
↑	↑	↓	↑
↓	↓	↑	↓

Support Vector Machine | GAMMA HYPERPARAMETER

“Gamma parameter of RBF controls the distance of influence of a single training point. Low values of gamma indicates a large similarity radius which results in more points being grouped together. For high values of gamma, the points need to be very close to each other in order to be considered in the same group (or class). Therefore, models with very large gamma values tend to overfit.”





If gamma is large, the effect of c becomes negligible.
If gamma is small, c affects the model just like how it affects a linear model.

$0.0001 < \text{gamma} < 10$

$0.1 < c < 100$

Support Vector Machine

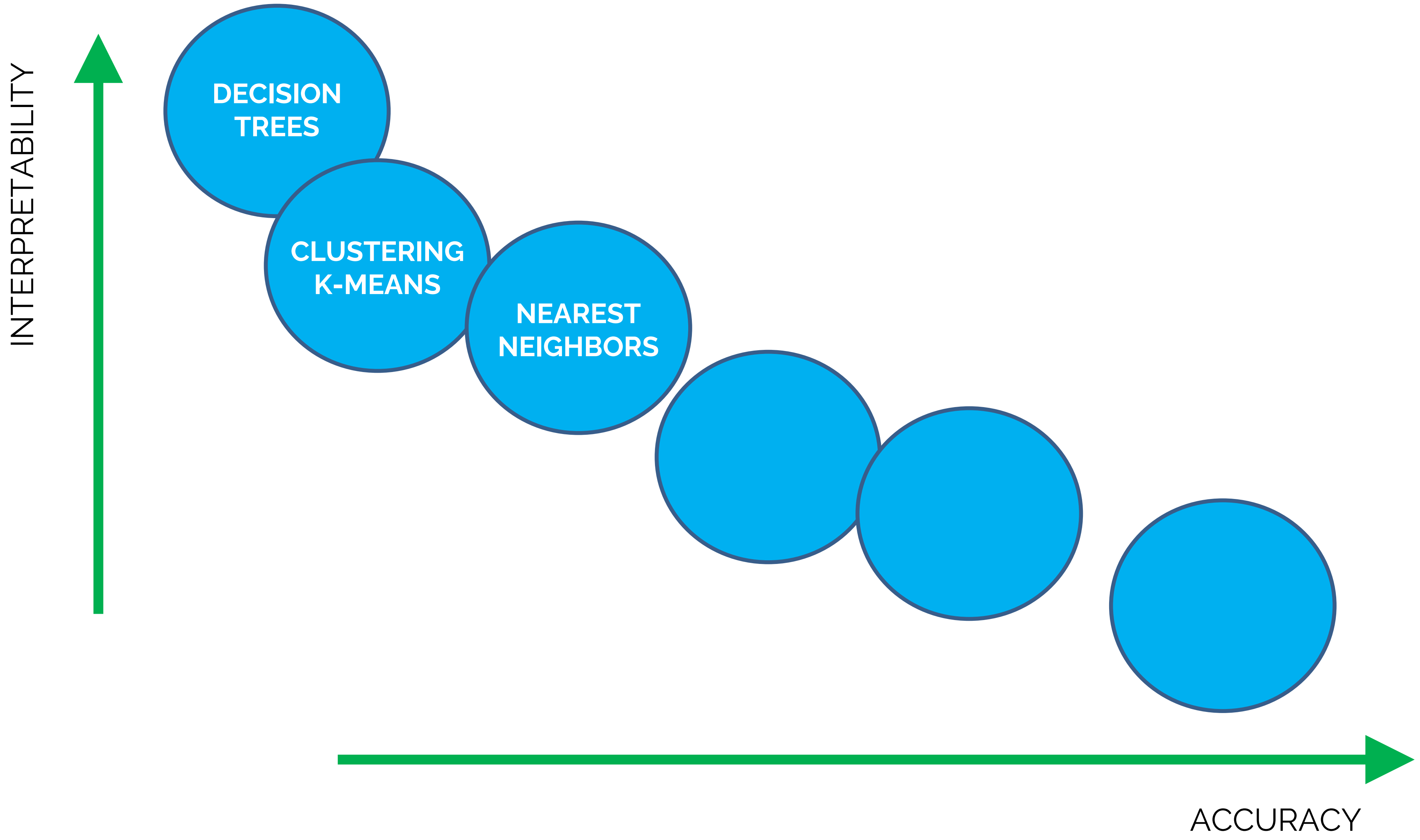
Advantages:

- Very powerful framework
- SVMs work well in practice (even with small datasets)

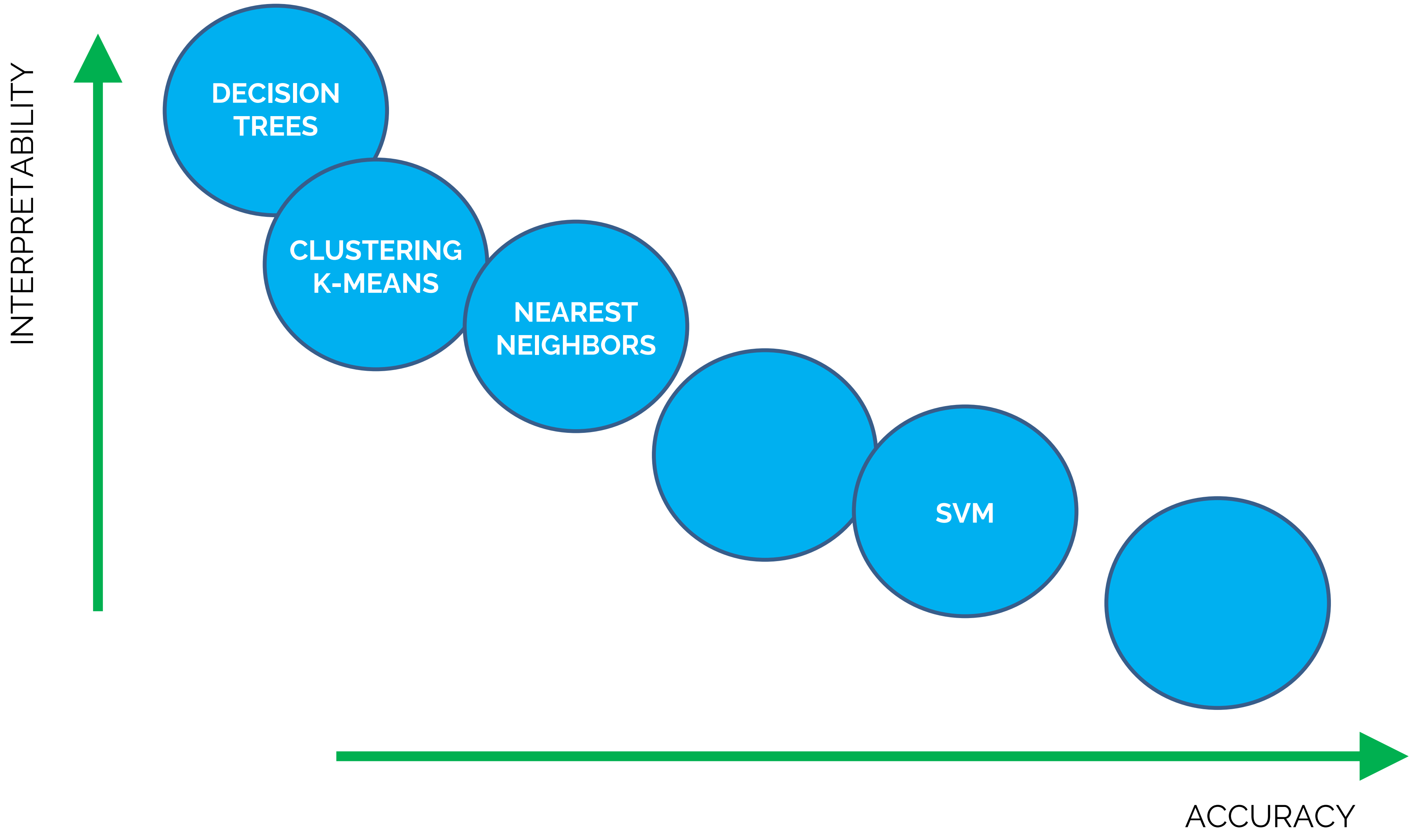
Drawbacks:

- High computational costs for training
- SVMs only allow direct binary classification
- Classification [+1/-1] is the only output (no probability-of-classification is given)

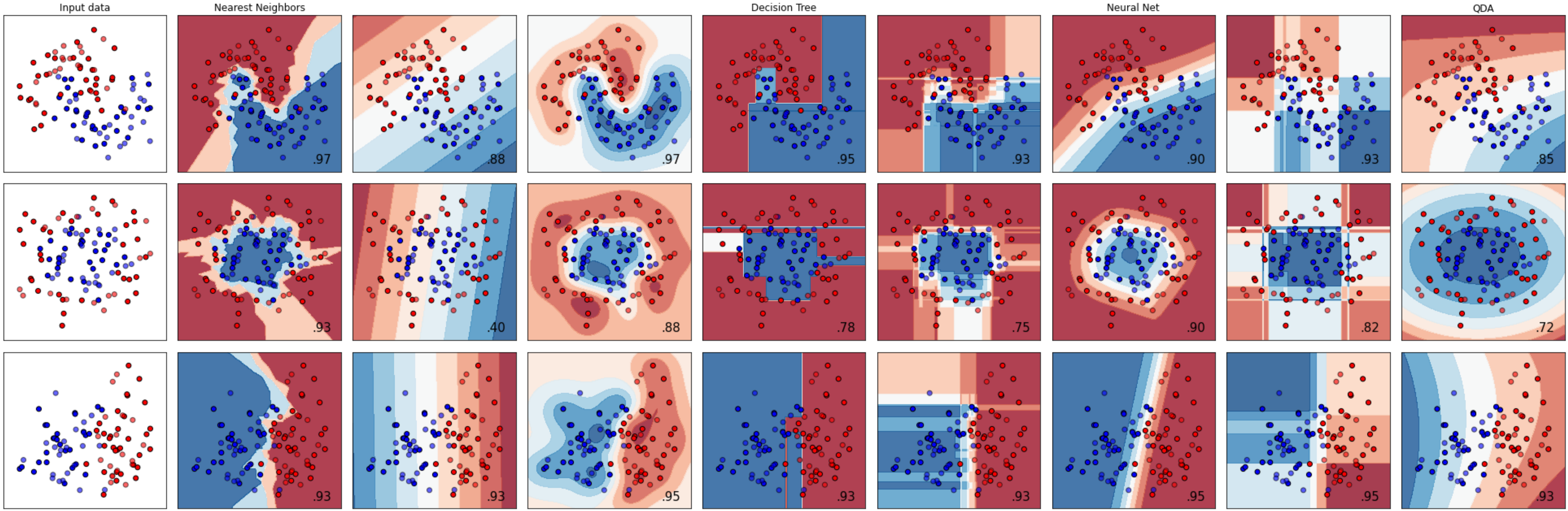
Interpretability-Accuracy TRADEOFF



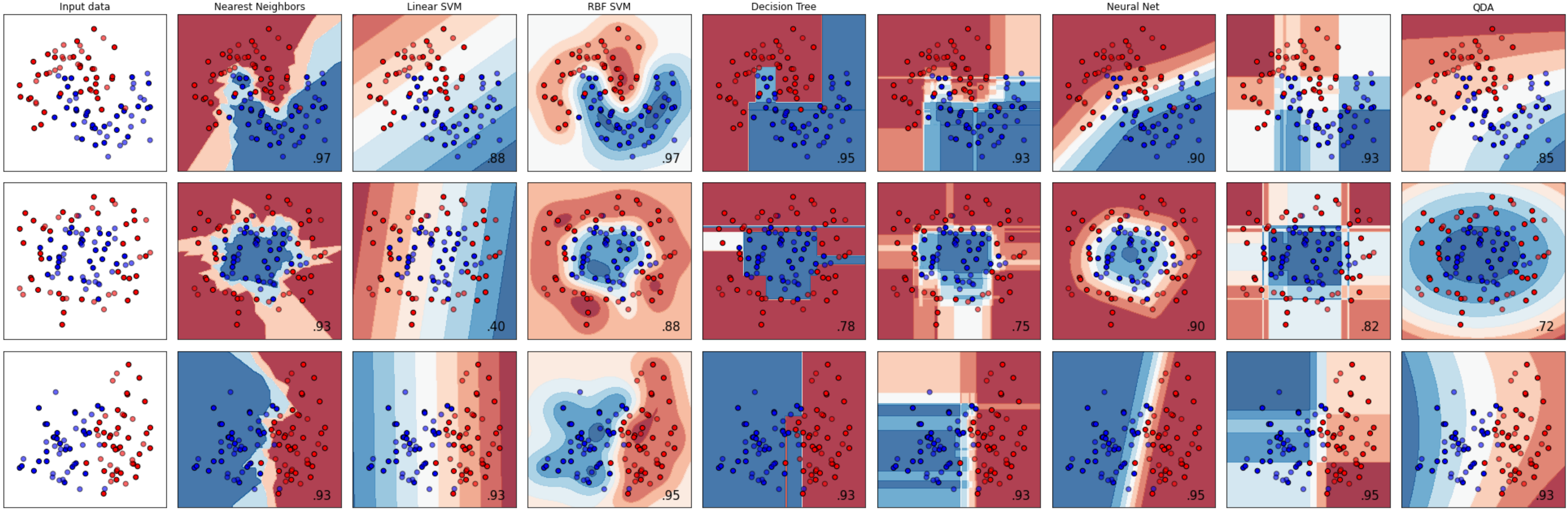
Interpretability-Accuracy TRADEOFF



Which is which | Decision Function



Which is which | Decision Function


Input data
Nearest Neighbors
Linear SVM
RBF SVM
Decision Tree
Neural Net
QDA