# Support Vector Machine

Zhongjian Lin
University of Georgia

October 1, 2024

## Support Vector Machine

- The *support vector machine* (SVM),is an approach for classification that was developed in the computer science community in the 1990s and that has grown in popularity since then.
- SVMs have been shown to perform well in a variety of settings, and are often considered one of the best "out of the box" classifiers.
- The support vector machine is a generalization of a simple and intuitive classifier called the *maximal margin classifier*.

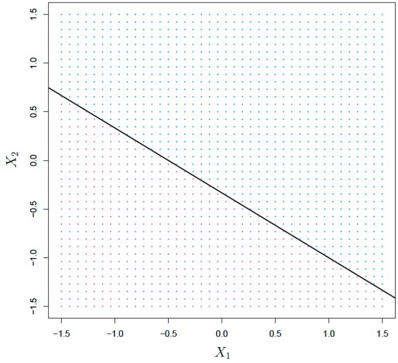
## Hyperplane

- In a p-dimensional space, a hyperplane is a flat affine subspace of dimension hyperplane $p-1$.
- For instance, in two dimensions, a hyperplane is a flat one-dimensional subspace-in other words, a line.
- In three dimensions, a hyperplane is a flat two-dimensional subspace-that is, a plane.
- In $p>3$ dimensions, it can be hard to visualize a hyperplane, but the notion of a $(p-1)$-dimensional flat subspace still applies.
- In the p-dimensional setting,

  $$\beta_0+\beta_1X_1+\beta_2X_2+\cdots+\beta_pX_p=0 \tag{1}$$

  defines a p-dimensional hyperplane, again in the sense that if a point $X=(X_1,X_2,\cdots,X_p)'$ in p-dimensional space (i.e. a vector of length p) satisfies Equation (1), then $X$ lies on the hyperplane.
- So we can think of the hyperplane as dividing p-dimensional space into two halves.

## Hyperplane Seperating



The hyperplane (line) defined by $1 + 2X_1 + 3X_2 = 0$.

## Seperating Hyperplane

Now suppose that we have an $n \times p$ data matrix $X$ that consists of $n$ training observations in p-dimensional space,

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \cdots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

and that these observations fall into two classes—that is, $y_1, \cdots, y_n \in \{-1, 1\}$ where $-1$ represents one class and 1 the other class. We have a test observation, a p-vector of observed features $x^* = (x_1^*, \cdots, x_p^*)'$. Our goal is to develop a classifier based on the training data that will correctly classify the test observation using its feature measurements. A *separating hyperplane* has the property that

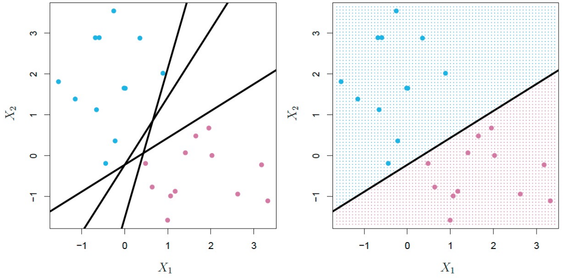
$$\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} > 0 \text{ if } y_i = 1,$$

and

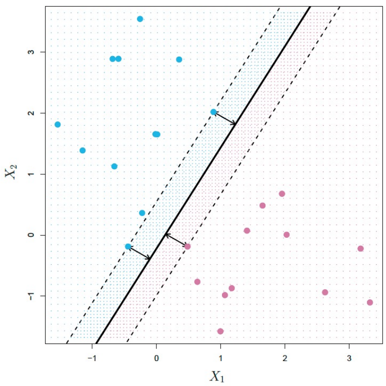
$$\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} < 0 \text{ if } y_i = -1.$$

## Maximal Margin Classifier

If a separating hyperplane exists, we can use it to construct a very natural classifier: a test observation is assigned a class depending on which side of the hyperplane it is located. We classify the test observation $x^*$ based on the sign of $f(x^*) = \beta_0 + \beta_1 x_1^* + \cdots + \beta_p x_p^*$. We can also make use of the magnitude of $f(x^*)$. In general, if our data can be perfectly separated using a hyperplane, then there will in fact exist an infinite number of such hyperplanes. This is because a given separating hyperplane can usually be shifted a tiny bit up or down, or rotated, without coming into contact with any of the observations. A natural choice is the *maximal margin hyperplane* (also known as the *optimal separating hyperplane*), which is the separating hyperplane that is farthest from the training observations.



# Maximal Margin Classifier



Three training observations are equidistant from the maximal margin hyperplane and lie along the dashed lines indicating the width of the margin. These three observations are known as *support vectors*, since they are vectors in p-dimensional space. They “support” the maximal margin hyperplane. The fact that the maximal margin hyperplane depends directly on only a small subset of the observations is an important property.

## Maximal Margin Classifier

We now consider the task of constructing the maximal margin hyperplane based on a set of $n$ training observations $x_1, \cdots, x_n \in R^p$ and associated class labels $y_1, \cdots, y_n \in \{-1, 1\}$. Briefly, the maximal margin hyperplane is the solution to the optimization problem

$$\max_{\beta_0, \beta_1, \cdots, \beta_p, M} M$$

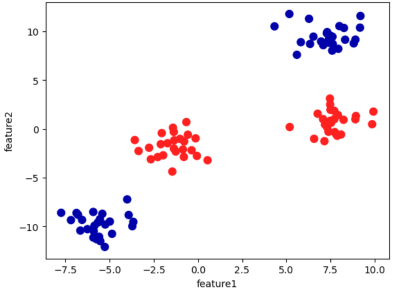
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

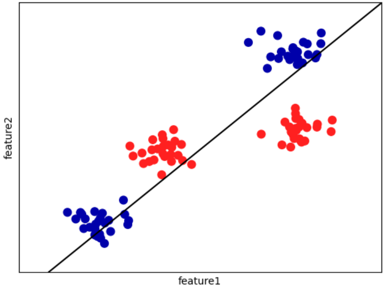
$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) > M, \ \forall i = 1, \cdots, n.$$

where $M$ is positive.

## The Non-separable Case

The maximal margin classifier is a very natural way to perform classification, if a separating hyperplane exists. However, in many cases no separating hyperplane exists, and so there is no maximal margin classifier.

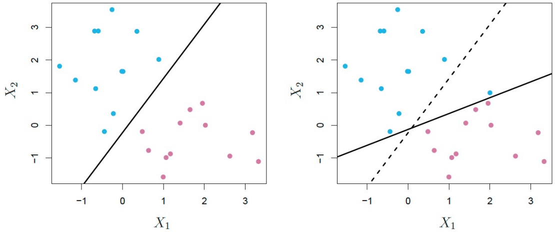




However, we can extend the concept of a separating hyperplane in order to develop a hyperplane that almost separates the classes, using a so-called *soft margin*. The generalization of the maximal margin classifier to the non-separable case is known as the *support vector classifier*.

## Support Vector Classifiers

In fact, even if a separating hyperplane does exist, then there are instances in which a classifier based on a separating hyperplane might not be desirable. The maximal margin hyperplane is extremely sensitive to a change in a single observation and this suggests that it may have overfit the training data.



In this case, we might be willing to consider a classifier based on a hyperplane that does not perfectly separate the two classes, in the interest of

- Greater robustness to individual observations, and
- Better classification of most of the training observations.

The *support vector classifier*, sometimes called a *soft margin classifier*, support vector classifier soft margin classifier does exactly this.

## Support Vector Classifiers

The hyperplane is chosen to correctlyseparate most of the training observations into the two classes, but may misclassify a few observations. It is the solution to the optimization problem

$$\max_{\beta_0,\beta_1,\cdots,\beta_p,\epsilon_1,\cdots,\epsilon_n,M} M$$

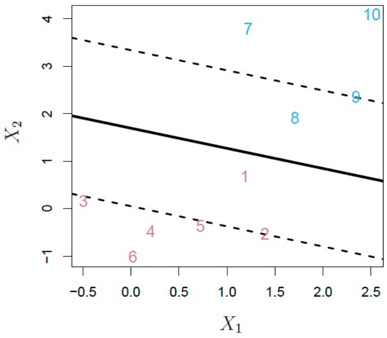
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

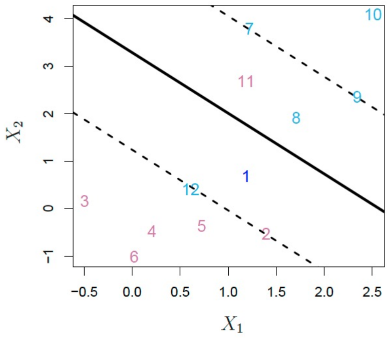
$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) > M(1 - \epsilon_i),\ \forall i = 1, \cdots, n,$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C,$$

where $C$ is a nonnegative tuning parameter. $\epsilon_1, \cdots, \epsilon_n$ are *slack variables* that allow individual observations to be on slack the wrong side of the margin or the hyperplane. If $\epsilon_i = 0$ then the $i$th observation is on the correct side of the margin. If $\epsilon_i > 0$ then the $i$th observation is on the wrong side of the margin, and we say that the $i$th observation has violated the margin. If $\epsilon_i > 1$ then it is on the wrong side of the hyperplane.

## Support Vector Classifiers

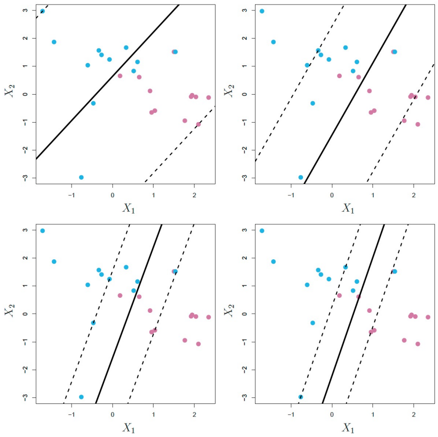




An observation that lies strictly on the correct side of the margin does not affect the support vector classifier! Changing the position of that observation would not change the classifier at all, provided that its position remains on the correct side of the margin. Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as *support vectors*.

## Tuning Parameter $C$

$C$ bounds the sum of the $\epsilon_i$'s, and so it determines the number and severity of the violations to the margin (and to the hyperplane) that we will tolerate. As the budget $C$ increases, we become more tolerant of violations to the margin, and so the margin will widen. Conversely, as $C$ decreases, we become less tolerant of violations to the margin and so the margin narrows.



## Nonlinear Decision Boundary

The support vector classifier is a natural approach for classification in the two-class setting, if the boundary between the two classes is linear. However, in practice we are sometimes faced with non-linear class boundaries. One way to tackle this problem is by enlarging the feature space using quadratic, cubic, and even higher-order polynomial functions of the predictors. For instance, rather than fitting a support vector classifier using $p$ features $X_1, \cdots, X_p$, we could instead fit a support vector classifier using $2p$ features$X_1, \cdots, X_p, X_1^2, \cdots, X_p^2$.

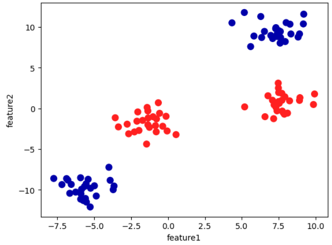
$$\max_{\beta_0, \beta_{11}, \beta_{12}, \cdots, \beta_{p1}, \beta_{p2}, \epsilon_1, \cdots, \epsilon_n, M} M$$

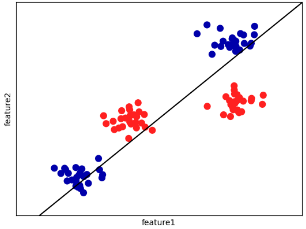
$$\text{subject to } \sum_{k=1}^{2} \sum_{j=1}^{p} \beta_{jk}^2 = 1$$

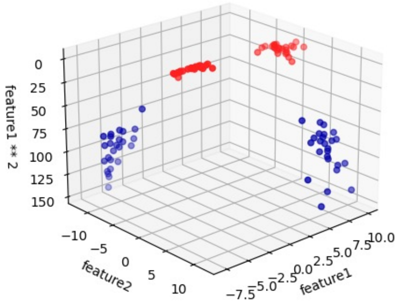
$$y_i \left( \beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \right) > M(1 - \epsilon_i), \ \forall i = 1, \cdots, n,$$

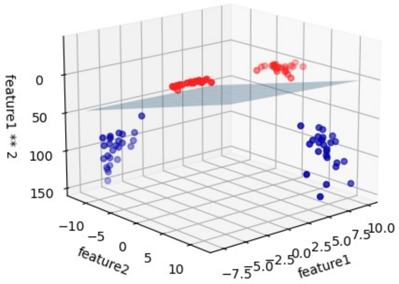
$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C,$$

## Nonlinear Features



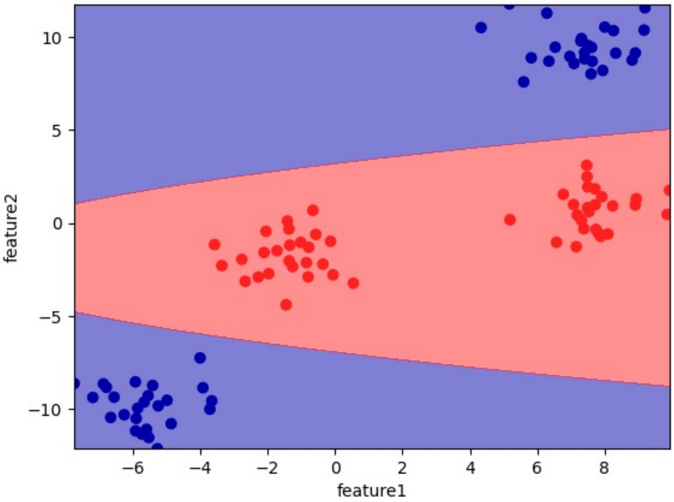






## Nonlinear Features

As a function of the original features, the linear SVM model is not actually linear anymore. It is not a line, but more of an ellipse.



## The Support Vector Machine

The support vector machine (SVM) is an extension of the support vector classifier that results from enlarging the feature space in a specific way, using kernels. The main idea is that we may want to enlarge our feature space in order to accommodate a non-linear boundary between the classes. The kernel approach here is simply an efficient computational approach for enacting this idea. Before going into more details, let's define the inner product, i.e., the inner product of two observations $x_i, x_{i'}$ is given by

$$<x_i, x_{i'}> = \sum_{j=1}^{p} x_{ij} x_{i'j}.$$

It can be shown that the linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i <x, x_i>.$$

To estimate the parameters $\alpha_1, \cdots, \alpha_n$ and $\beta_0$, all we need are the $\binom{n}{2}$ inner products $<x_i, x_{i'}>$ between all pairs of training observations.

## The Kernelized SVMs

Furthermore, it turns out that $\alpha_i$ is nonzero only for the support vectors in the solution-that is, if a training observation is not a support vector, then its $\alpha_i$ equals zero. Let $\mathcal{S}$ be the set of all support vectors, we then have

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}}^{n} \alpha_i \langle x, x_i \rangle.$$

To summarize, in representing the linear classifier $f(x)$, and in computing its coefficients, all we need are inner products. Now suppose that we replace the inner product with a generalization of the inner product of the form

$$K(x_i, x_{i'}),$$

where $K$ is some function that we will refer to as a kernel. A kernel is a function that quantifies the similarity of two observations. We now arrive at the *Kernelized SVMs*.

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}}^{n} \alpha_i K(x, x_i).$$

## Kernels

There are many possible Kernel functions

- Linear Kernel:

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

- Polynomial kernel of degree *d*:

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$

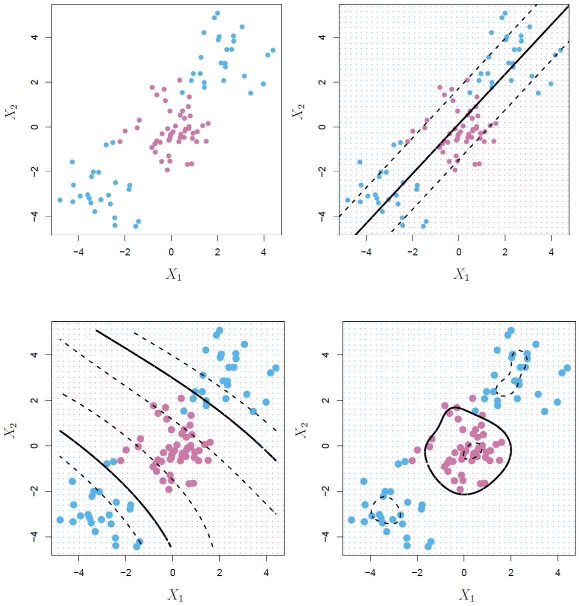
  Using such a kernel with $d > 1$, instead of the standard linear kernel, in the support vector classifier algorithm leads to a much more flexible decision boundary.

- The radial basis function (rbf) kernel, also known as Gaussian kernel.

$$k_{\text{rbf}}(x_1, x_2) = \exp(-\gamma ||x_1 - x_2||^2)$$

  The radial basis function kernel has very local behavior, in the sense that only nearby training observations have an effect on the class label of a test observation.

# Kernelized SVMs



## SVMs for More than Two Classes

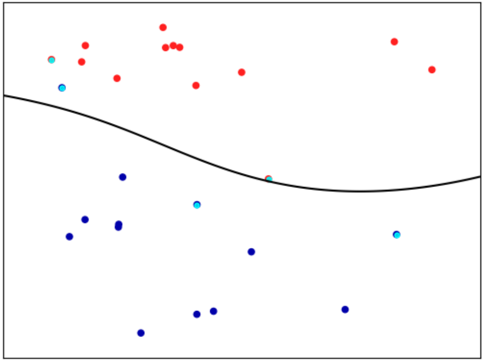
One-Versus-One Classification: suppose that we would like to perform classification using SVMs, and there are $K > 2$ classes. A one-versus-one or all-pairs approach constructs $\begin{pmatrix} K \\ 2 \end{pmatrix}$ SVMs, each of which compares a pair of classes. For example, one such one SVM might compare the $k$th class, coded as $+1$, to the $k'$th class, coded as $-1$. We classify a test observation using each of the $\begin{pmatrix} K \\ 2 \end{pmatrix}$ classifiers, and we tally the number of times that the test observation is assigned to each of the $K$ classes. The final classification is performed by assigning the test observation to the class to which it was most frequently assigned in these $\begin{pmatrix} K \\ 2 \end{pmatrix}$ pairwise classifications.

## One-Versus-All

The one-versus-all approach (also referred to as one-versus-rest) is an alternative procedure for applying SVMs in the case of $K > 2$ classes. We fit $K$ SVMs, each time comparing one of the $K$ classes to the remaining $K - 1$ classes. Let $\beta_{0k}, \beta_{1k}, \cdots, \beta_{pk}$ denote the parameters that result from fitting an SVM comparing the $k$th class (coded as $+1$) to the others (coded as $-1$). Let $x^*$ denote a test observation. We assign the observation to the class for which $\beta_{0k} + \beta_{1k}x_1^* + \cdots + \beta_{pk}x_k^*$ is largest, as this amounts to a high level of confidence that the test observation belongs to the $k$th class rather than to any of the other classes.

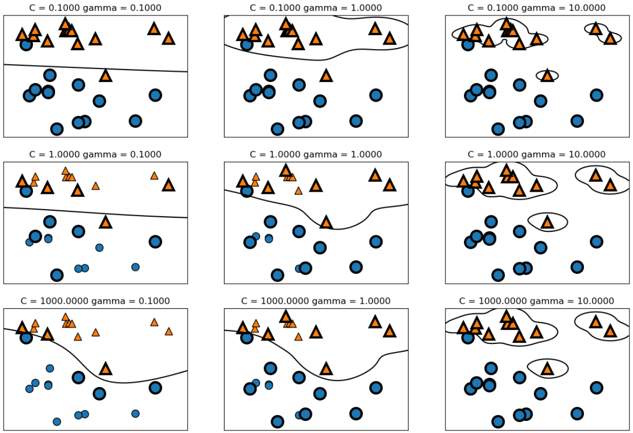
## Support Vectors Machine

```
from sklearn.svm import SVC
X, y = mglearn.tools.make_handcrafted_dataset()
svm = SVC(kernel='rbf', C=10, gamma=0.1).fit(X, y)
mglearn.plots.plot_2d_separator(svm, X, eps=.5)
# plot data
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=mglearn.cm2,s=20)
# plot support vectors
sv = svm.support_vectors_
plt.scatter(sv[:, 0], sv[:, 1], s=2, facecolors='cyan',zorder=2, linewidth=3)
```



## Parameter Choices

A small C means a very restricted model, where each data point can only have very limited influence. Increasing C, as shown on the bottom right, allows these points to have a stronger influence on the model, and makes the decision boundary bend to correctly classify them. The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'.


C = 0.1000 gamma = 0.1000
C = 0.1000 gamma = 1.0000
C = 0.1000 gamma = 10.0000
C = 1.0000 gamma = 0.1000
C = 1.0000 gamma = 1.0000
C = 1.0000 gamma = 10.0000
C = 1000.0000 gamma = 0.1000
C = 1000.0000 gamma = 1.0000
C = 1000.0000 gamma = 10.0000