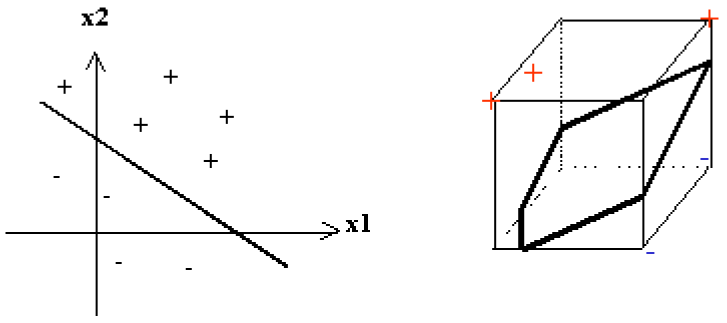
Brief Introduction to Support Vector Machines

1. Linearly Separable models

If all patterns in a dataset can be separated by a straight line or a hyper-plane, the dataset is said to be **linearly separable**.



However, there are many problems, such as XOR, which are not linearly separable.

2. SVM Approach

SVM uses linear models to implement nonlinear class boundaries. It transforms the input space using a nonlinear mapping into a new space (F feature space). Then a linear model constructed in the new space can represent a nonlinear decision boundary in the original space:

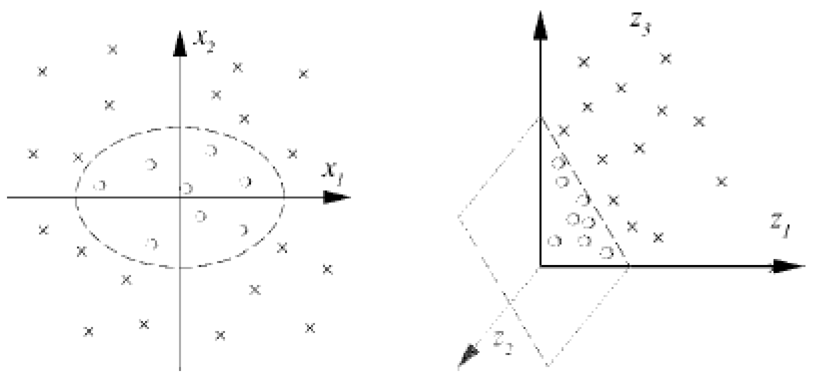


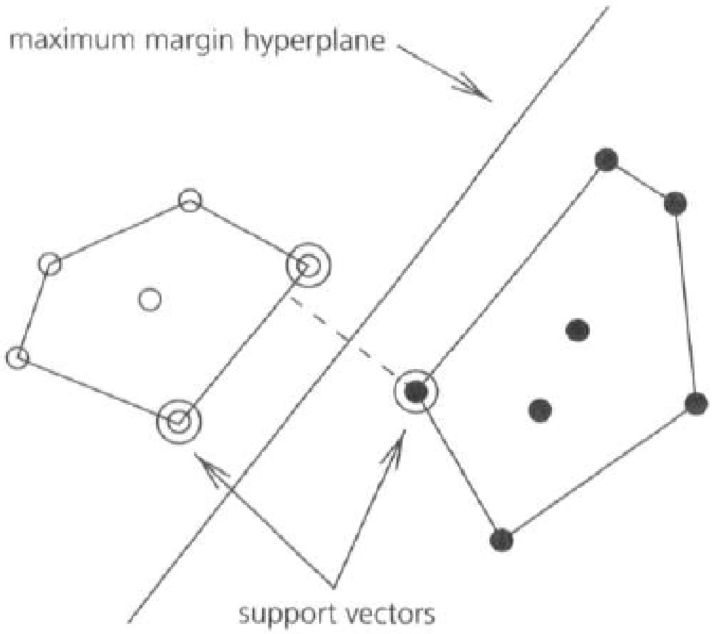
Fig. 4. Two-dimensional classification example. (a) Using the second-order monomials x_1^2, $\sqrt{2}\,x_1 x_2$ and x_2^2 as features a separation in feature space can be found using a *linear* hyperplane. (b) In input space this construction corresponds to a *nonlinear* ellipsoidal decision boundary (figure from [48]).

3. The maximum margin hyperplane

Another component in the SVM approach is the maximum margin hyperplane.

a. What is a maximum margin hyperplane?

Suppose a two-class dataset is **linearly separable.** The maximum margin hyperplane is the one that gives the greatest separation between the classes. Among all hyperplanes that separate the classes, the maximum margin hyperplane is the one that is as far away as possible from the two convex hulls, each of which is formed by connecting the instances of a class.



The instances that are closest to the maximum hyperplane are called **support vectors.** There is at least one support vector for each class, and often there are more.

A set of support vectors can uniquely defines the maximum margin hyperplane for the learning problem. All other training instances are irrelevant; they can be removed without changing the position and orientation of the hyperplane.

b. The equation of the maximum margin hyperplane

In general, a hyperplane separating the two classes in the two-attribute case, can be written as

$F(x) = w_0 + w_1 a_1 + w_2 a_2$, where a_1 and a_2 are attribute values, and w_0, w_1, w_2 are weights.

The equation for the maximum margin hyperplane can also be written in terms of support vectors.

$$f(x) = b + \sum_{i=1}^{l} \alpha_i \, y_i < x_i \bullet x >$$

where

i is the support vector;
y_i is the class value of a training instance: either 1 or -1;
x_i is the *i*th support vector;
x is a vector which represents a test instance;
$< x_i \bullet x >$ denotes the dot product of x and x_i; and
b and α's are parameters that determine the hyperplane (just as the weights w_0, w_1, w_2).

Finding support vectors and the maximum margin hyperplane (i.e., b and α's) belongs to a standard class of optimization problem known as **constraint quadratic optimization**. There are several off-the-shelf tools for solving those problems.

4. Kernel Functions

According to the equation of the maximum margin hyperplane above, every time an instance is classified, its dot product with all support vectors must be calculated. In a high-dimensional space resulting after a non-linear transformation is applied to the instances, the number of attributes in the new space could become huge and the computation of dot product becomes very expensive. The same problem occurs during training as well. Working in such a high-dimensional space becomes intractable very quickly.

$$f(x) = b + \sum_{i=1}^{l} \alpha_i \, y_i < \phi(x_i) \bullet \phi(x) > \qquad (1)$$

where ϕ is a non-linear transformation function.

Fortunately, we don't need to explicitly construct a high dimensional features space. By using the kernel function, it is possible to calculate the dot product before the non-linear transformation is performed. In other words, a kernel function can be applied to instances in the original input space which brings out the same effect as the linear transformation without expanding the feature space by non-linear transformation – *"kernel trick"*.

Implicit mapping into the transformed feature space

A kernel is a function K such that, for all x, z € X (input space)
K(x, z) = $< \phi(x\ ) \bullet \phi(z) >$

By using K, the equation (1) can be written as:

$$f(x) = b + \sum_{i=1}^{l} \alpha_i \, y_i K(x_i, x) \qquad (2)$$

Polynomial Kernel

The function $< x \bullet z >^n$ represents a polynomial function of degree n applied to the do product of two vector x and z. In the case when n = 2 and the number of attributes is 2, the function becomes:

$$
\begin{aligned}
&< x \bullet z >^2 \\
&= (\sum_{i=1}^{2} x_i z_i)^2 \\
&= (x_1 z_1 + x_2 z_2)^2 \\
&= (x_1^2 z_1^2 + 2 x_1 x_2 z_1 z_2 + x_2^2 z_2^2) \\
&=< (x_1^2, \sqrt{2} x_1 x_2, x_2^2) \bullet (z_1^2, \sqrt{2} z_1 z_2, z_2^2) > \\
&=< \phi(x) \bullet \phi(z) >
\end{aligned}
$$

Therefore, the kernel K becomes

K(x, z) $=< x \bullet z >^2 = < \phi(x) \bullet \phi(z) >$, where $\phi(x) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$.

Other Kernels

In addition to the polynomial kernel, **RBF (radial basis function)** and **sigmoid kernels** are often used. Using which kernel depends on the specific data and applications.

5. Advantage & Disadvantages of SVM

Advantages:

1. Produce very accurate classifiers.
2. Less overfitting, robust to noise.

Disadvantages:

1. SVM is a binary classifier. To do a multi-class classification, pair-wise classifications can be used (one class against all others, for all classes).
2. Computationally expensive, thus runs slow.