Pattern Recognition 2018
Support Vector Machines

Ad Feelders

Universiteit Utrecht

Support Vector Machines

Overview

1. Separable Case
2. Kernel Functions
3. Allowing Errors (Soft Margin)
4. SVM's in `R`.

Linear Classifier for two classes

Linear model

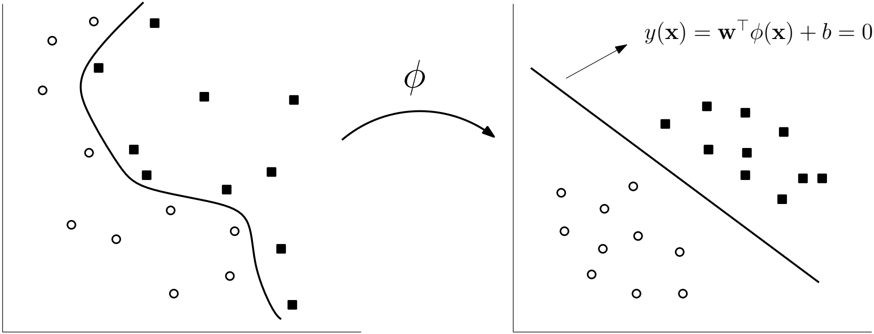
$$y(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b \tag{7.1}$$

with $t_n \in \{-1, +1\}$.

- Predict $t_0 = +1$ if $y(\mathbf{x}_0) \geq 0$ and $t_0 = -1$ otherwise.
- The decision boundary is given by $y(\mathbf{x}) = 0$.

This is a linear classifier in feature space $\phi(\mathbf{x})$.

Mapping



ϕ maps $\mathbf{x}$ into higher dimensional space where data is linearly separable.

Data linearly separable

Assume training data is linearly separable in feature space, so there is at least one choice of $\mathbf{w}, b$ such that:

1. $y(\mathbf{x}_n) > 0$ for $t_n = +1$;
2. $y(\mathbf{x}_n) < 0$ for $t_n = -1$;

that is, all training points are classified correctly.

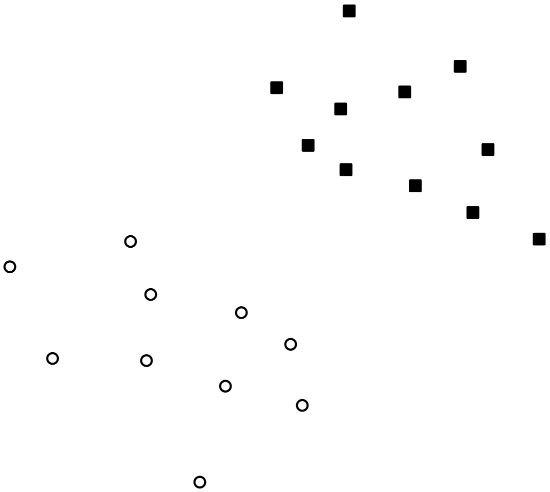
Putting 1. and 2. together:

$$t_n y(\mathbf{x}_n) > 0 \qquad \text{for } n = 1, \ldots, N$$

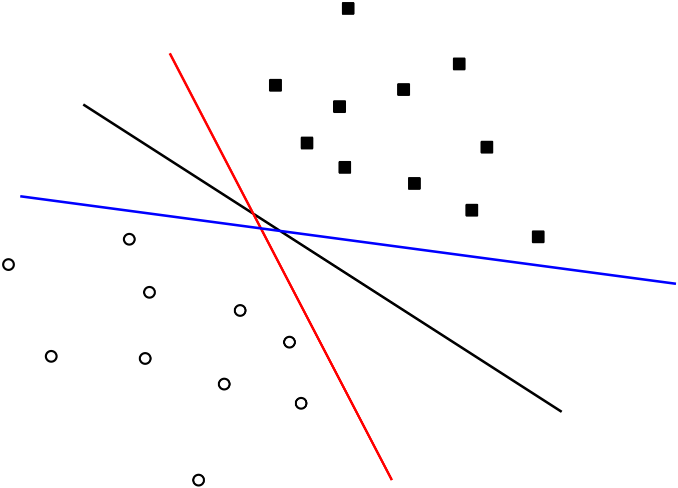
Maximum Margin

- There may be many solutions that separate the classes exactly.
- Which one gives smallest prediction error?
- SVM chooses line with maximal *margin*, where the margin is the distance between the line and the closest data point.
- In this way, it avoids "low confidence" classifications.

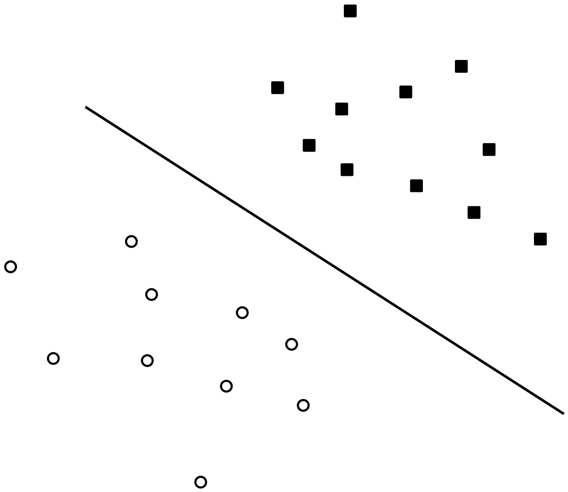
Two-class training data



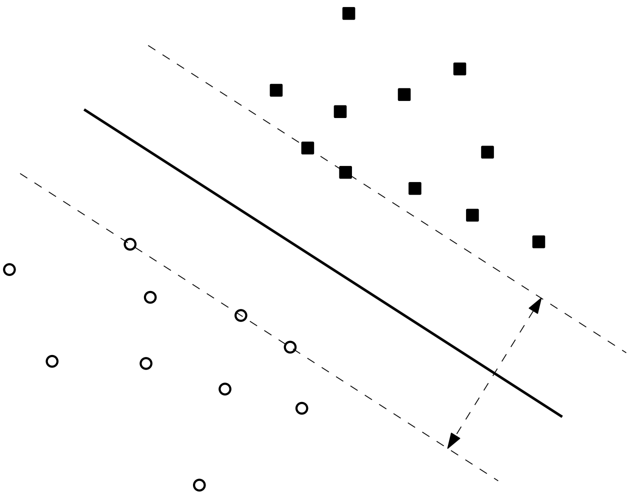
Many Linear Separators



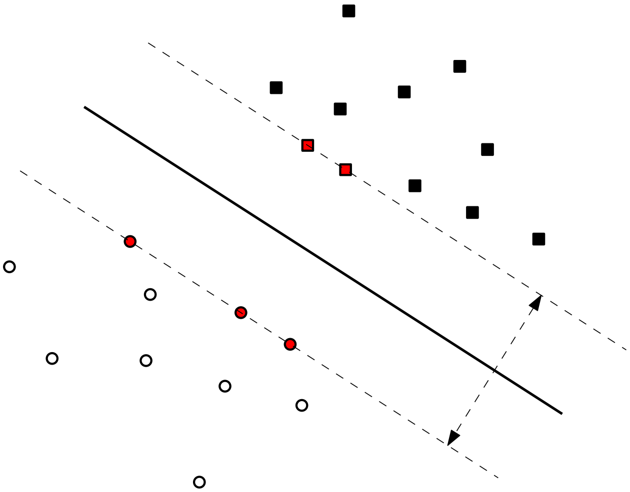
SVM Decision Boundary



Maximize Margin



Support Vectors



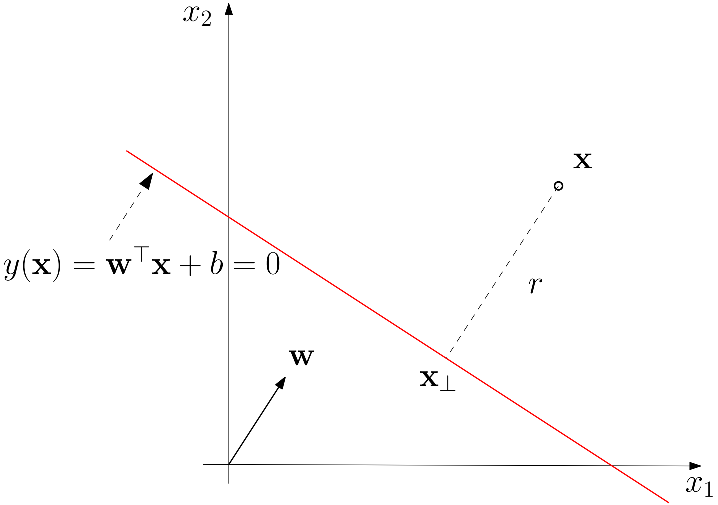
Weight vector is orthogonal to the decision boundary

Consider two points $\mathbf{x}_A$ and $\mathbf{x}_B$ both of which lie on the decision surface. Because $y(\mathbf{x}_A) = y(\mathbf{x}_B) = 0$, we have

$$(\mathbf{w}^\top \mathbf{x}_A + b) - (\mathbf{w}^\top \mathbf{x}_B + b) = \mathbf{w}^\top (\mathbf{x}_A - \mathbf{x}_B) = 0$$

and so the vector $\mathbf{w}$ is orthogonal to the decision surface.

Distance of a point to a line



Distance to decision surface ($\phi(\mathbf{x}) = \mathbf{x}$)

We have

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}. \tag{4.6}$$

where $\frac{\mathbf{w}}{\|\mathbf{w}\|}$ is the unit vector in the direction of $\mathbf{w}$, $\mathbf{x}_\perp$ is the orthogonal projection of $\mathbf{x}$ onto the line $y(\mathbf{x}) = 0$, and r is the (signed) distance of $\mathbf{x}$ to the line. Multiply (4.6) left and right by $\mathbf{w}^\top$ and add b:

$$\underbrace{\mathbf{w}^\top\mathbf{x} + b}_{y(\mathbf{x})} = \underbrace{\mathbf{w}^\top\mathbf{x}_\perp + b}_{0} + r\frac{\mathbf{w}^\top\mathbf{w}}{\|\mathbf{w}\|}$$

So we get

$$r = y(\mathbf{x})\frac{\|\mathbf{w}\|}{\|\mathbf{w}\|^2} = \frac{y(\mathbf{x})}{\|\mathbf{w}\|} \tag{4.7}$$

Distance of a point to a line

The signed distance of $\mathbf{x}_n$ to the decision boundary is

$$r = \frac{y(\mathbf{x}_n)}{\|\mathbf{w}\|}$$

For lines that separate the data perfectly, we have $t_n y(\mathbf{x}_n) = |y(\mathbf{x}_n)|$, so that the distance is given by

$$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n(\mathbf{w}^\top \phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|} \qquad (7.2)$$

Maximum margin solution

Now we are ready to define the optimization problem:

$$\arg\max_{\mathbf{w},b} \left\{ \min_n \left[\frac{t_n(\mathbf{w}^\top \phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|} \right] \right\}. \tag{7.3}$$

Since $\frac{1}{\|\mathbf{w}\|}$ does not depend on n, it can be moved outside of the minimization:

$$\arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n(\mathbf{w}^\top \phi(\mathbf{x}_n) + b)] \right\}. \tag{7.3}$$

- Direct solution of this problem would be rather complex.
- A more convenient representation is possible.

Canonical Representation

The hyperplane (decision boundary) is defined by

$$\mathbf{w}^\top \phi(\mathbf{x}) + b = 0$$

Then also

$$\kappa(\mathbf{w}^\top \phi(\mathbf{x}) + b) = \kappa \mathbf{w}^\top \phi(\mathbf{x}) + \kappa b = 0$$

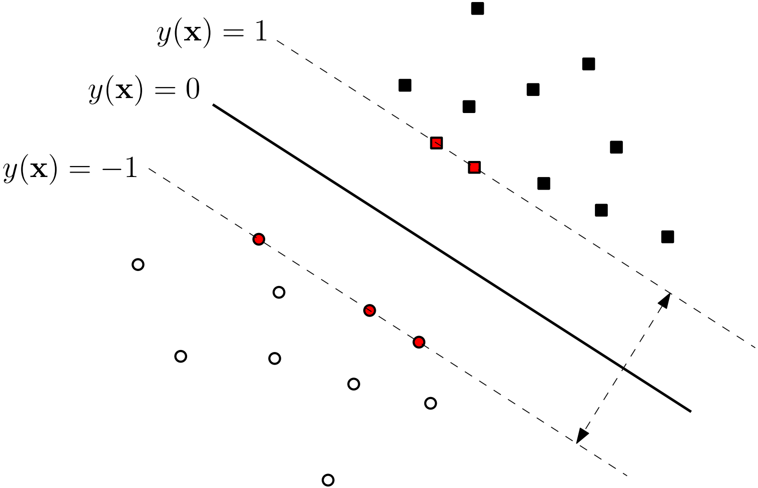
so rescaling $\mathbf{w} \rightarrow \kappa \mathbf{w}$ and $b \rightarrow \kappa b$ gives just another representation of the same decision boundary.

To resolve this ambiguity, we choose the scaling factor such that

$$t_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b) = 1 \tag{7.4}$$

for the points $\mathbf{x}_i$ closest to the decision boundary.

Canonical Representation (square=1,circle=−1)



Canonical Representation

In this case we have

$$t_n(\mathbf{w}^\top \phi(\mathbf{x}_n) + b) \geq 1 \qquad n = 1, \ldots, N \tag{7.5}$$

Quadratic program

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{7.6}$$

subject to the constraints (7.5).

This optimization problem has a unique global minimum.

Lagrangian Function

Introduce Lagrange multipliers $a_n \geq 0$ to get Lagrangian function

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \{t_n(\mathbf{w}^\top \phi(\mathbf{x}_n) + b) - 1\} \tag{7.7}$$

with

$$\frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)$$

Lagrangian Function

and for b:

$$\frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial b} = -\sum_{n=1}^{N} a_n t_n$$

Equating the derivatives to zero yields the conditions:

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n) \tag{7.8}$$

and

$$\sum_{n=1}^{N} a_n t_n = 0 \tag{7.9}$$

Dual Representation

Eliminating $\mathbf{w}$ and b from $L(\mathbf{w}, b, \mathbf{a})$ gives the *dual representation*.

$$
\begin{aligned}
L(\mathbf{w}, b, \mathbf{a}) &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \{t_n(\mathbf{w}^\top \phi(\mathbf{x}_n) + b) - 1\} \\
&= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n t_n \mathbf{w}^\top \phi(\mathbf{x}_n) - b \sum_{n=1}^{N} a_n t_n + \sum_{n=1}^{N} a_n \\
&= \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m) \\
&- \sum_{n=1}^{N} \sum_{m=1}^{N} a_n t_n a_m t_m \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m) + \sum_{n=1}^{N} a_n \\
&= \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n t_n a_m t_m \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m)
\end{aligned}
$$

Dual Representation

Maximize

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n,m=1}^{N} a_n t_n a_m t_m \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m) \tag{7.10}$$

with respect to **a** and subject to the constraints

$$a_n \geq 0, \qquad n = 1, \ldots, N \tag{7.11}$$

$$\sum_{n=1}^{N} a_n t_n = 0. \tag{7.12}$$

Kernel Function

- We map $\mathbf{x}$ to a high-dimensional space $\phi(\mathbf{x})$ in which data is linearly separable.
- Performing computations in this high-dimensional space may be very expensive.
- Use a kernel function k that computes a dot product in this space without making the actual mapping ("kernel trick"):

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$$

Example: polynomial kernel

Suppose $\mathbf{x} \in \mathbb{R}^3$ and $\phi(\mathbf{x}) \in \mathbb{R}^{10}$ with

$$\phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3)$$

Then

$$\begin{aligned}\phi(\mathbf{x})^\top\phi(\mathbf{z}) &= 1 + 2x_1z_1 + 2x_2z_2 + 2x_3z_3 + x_1^2z_1^2 + x_2^2z_2^2 + x_3^2z_3^2 \\ &+ 2x_1x_2z_1z_2 + 2x_1x_3z_1z_3 + 2x_2x_3z_2z_3\end{aligned}$$

But this can be written as

$$(1 + \mathbf{x}^\top\mathbf{z})^2 = (1 + x_1z_1 + x_2z_2 + x_3z_3)^2$$

which costs much less operations to compute.

Polynomial kernel: numeric example

Suppose $\mathbf{x} = (3, 2, 6)$ and $\mathbf{z} = (4, 1, 5)$.

Then

$$\begin{aligned} \phi(\mathbf{x}) &= (1, 3\sqrt{2}, 2\sqrt{2}, 6\sqrt{2}, 9, 4, 36, 6\sqrt{2}, 18\sqrt{2}, 12\sqrt{2}) \\ \phi(\mathbf{z}) &= (1, 4\sqrt{2}, 1\sqrt{2}, 5\sqrt{2}, 16, 1, 25, 4\sqrt{2}, 20\sqrt{2}, 5\sqrt{2}) \end{aligned}$$

Then

$$\phi(\mathbf{x})^\top \phi(\mathbf{z}) = 1 + 24 + 4 + 60 + 144 + 4 + 900 + 48 + 720 + 120 = 2025.$$

But

$$(1 + \mathbf{x}^\top \mathbf{z})^2 = (1 + (3)(4) + (2)(1) + (6)(5))^2 = 45^2 = 2025$$

is a more efficient way to compute this dot product.

Kernels

Linear kernel

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$$

Two popular non-linear kernels are the polynomial kernel (of degree M):

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + c)^M$$

and Gaussian (or radial) kernel:

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2), \tag{6.23}$$

or

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2),$$

where $\gamma = \frac{1}{2\sigma^2}$.

Dual Representation with kernels

Using $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$ we get dual representation:

Maximize

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n,m=1}^{N} a_n t_n a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \tag{7.10}$$

with respect to **a** and subject to the constraints

$$a_n \geq 0, \qquad n = 1, \ldots, N \tag{7.11}$$

$$\sum_{n=1}^{N} a_n t_n = 0. \tag{7.12}$$

Is this dual "easier" than the original problem?

Prediction

Recall that

$$y(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b \tag{7.1}$$

Substituting

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n) \tag{7.8}$$

into (7.1), we get

$$\boxed{y(\mathbf{x}) = b + \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n)} \tag{7.13}$$

Constrained Optimization

Minimize

$$f(x)$$

subject to

$$g_i(x) \geq 0$$

Lagrangian function:

$$L(x, \lambda) = f(x) - \sum_i \lambda_i g_i(x)$$

KKT conditions for solution:

1. $g_i(x) \geq 0$
2. $\lambda_i \geq 0$
3. $\lambda_i g_i(x) = 0$

Prediction: support vectors

KKT conditions:

$$a_n \geq 0 \tag{7.14}$$

$$t_n y(\mathbf{x}_n) - 1 \geq 0 \tag{7.15}$$

$$a_n\{t_n y(\mathbf{x}_n) - 1\} = 0 \tag{7.16}$$

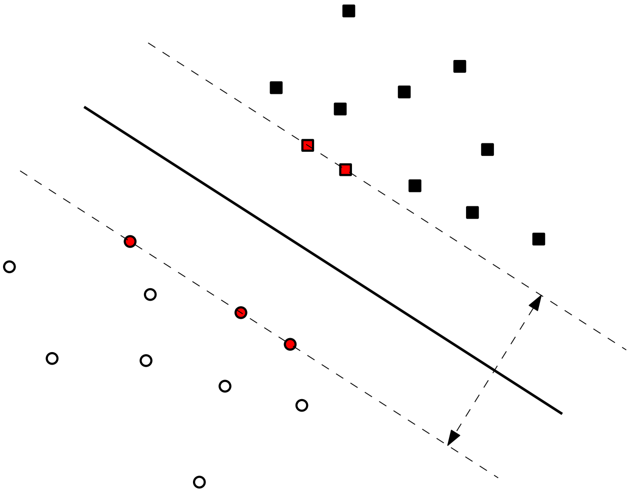
From (7.16) it follows that for every data point, either

1. $a_n = 0$, or
2. $t_n y(\mathbf{x}_n) = 1$.

The former play no role in making predictions (see 7.13), and the latter are the *support vectors* that lie on the maximum margin hyper planes.

Only the support vectors play a role in predicting the class of new attribute vectors!

Only the support vectors are important for prediction



Prediction: computing b

Since for any support vector $\mathbf{x}_n$ we have $t_n y(\mathbf{x}_n) = 1$, we can use (7.13) to get

$$t_n \left(b + \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right) = 1, \tag{7.17}$$

where $\mathcal{S}$ denotes the set of support vectors.

Hence we have

$$\begin{aligned} t_n b + t_n \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) &= 1 \\ t_n b &= 1 - t_n \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \end{aligned}$$

and since $t_n \in \{-1, +1\}$ and so $1/t_n = t_n$:

$$\boxed{b = t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m)} \tag{7.17a}$$

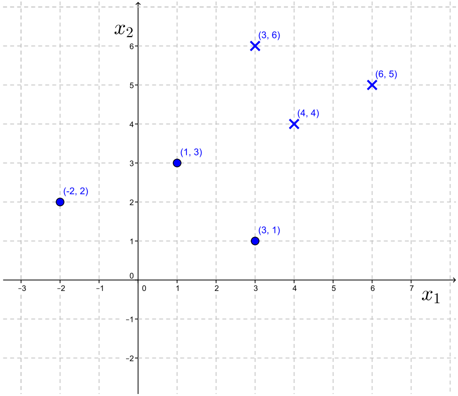
Prediction: Example

We receive the following output from the optimization software for fitting a support vector machine with linear kernel and perfect separation of the training data:

n	$x_{n,1}$	$x_{n,2}$	t_n	a_n
1	-2	2	-1	0
2	1	3	-1	$\frac{1}{8}$
3	3	1	-1	$\frac{1}{8}$
4	3	6	$+1$	0
5	4	4	$+1$	$\frac{1}{4}$
6	6	5	$+1$	0

Prediction: Example

The figure below is a plot of the same data set, where the dots represent points with class -1, and the crosses points with class $+1$.



Prediction: Example

(a) Compute the value of the SVM bias term b.
Data points with $a > 0$ are support vectors.
Let's take the point $x_1 = 4, x_2 = 4$ with class label $+1$:

$$b = t_m - \sum_{n=1}^{N} a_n t_n \mathbf{x}_m^\top \mathbf{x}_n = 1 + \tfrac{1}{8}[4 \;\; 4]\begin{bmatrix} 1 \\ 3 \end{bmatrix} + \tfrac{1}{8}[4 \;\; 4]\begin{bmatrix} 3 \\ 1 \end{bmatrix} - \tfrac{1}{4}[4 \;\; 4]\begin{bmatrix} 4 \\ 4 \end{bmatrix} = -3$$

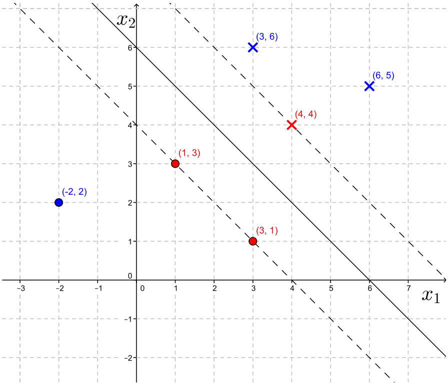
(b) Which class does the SVM predict for the data point $x_1 = 5, x_2 = 2$?

$$y(\mathbf{x}) = b + \sum_{n=1}^{N} a_n t_n \mathbf{x}^\top \mathbf{x}_n = -3 - \tfrac{1}{8}[5 \;\; 2]\begin{bmatrix} 1 \\ 3 \end{bmatrix} - \tfrac{1}{8}[5 \;\; 2]\begin{bmatrix} 3 \\ 1 \end{bmatrix} + \tfrac{1}{4}[5 \;\; 2]\begin{bmatrix} 4 \\ 4 \end{bmatrix} = \tfrac{1}{2}$$

Since the sign is positive, we predict class $+1$.

Prediction: Example

Decision boundary and support vectors.



Allowing Errors

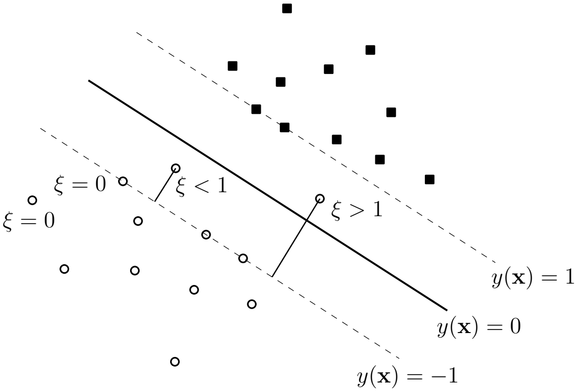
- So far we assumed that the training data points are linearly separable in feature space $\phi(\mathbf{x})$.
- Resulting SVM gives exact separation of training data in original input space $\mathbf{x}$, with non-linear decision boundary.
- Class distributions typically overlap, in which case exact separation of the training data leads to poor generalization (overfitting).

Allowing Errors

- 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 convenience we make this penalty a linear function of the distance to the margin boundary.
- Introduce slack variables $\xi_n \geq 0$ with one slack variable for each training data point.

Definition of Slack Variables

We define $\xi_n = 0$ for data points that are on the inside of the correct margin boundary and $\xi_n = |t_n - y(\mathbf{x}_n)|$ for all other data points.



New objective function

Our goal is to maximize the margin while softly penalizing points that lie on the wrong side of the margin boundary. We therefore minimize

$$\boxed{C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|\mathbf{w}\|^2} \tag{7.21}$$

where the parameter $C > 0$ controls the trade-off between the slack variable penalty and the margin. Alternative view (divide by C and put $\lambda = \frac{1}{2C}$):

$$\boxed{\sum_{n=1}^{N}\xi_n + \lambda\sum w_i^2}$$

First term represents lack-of-fit (hinge loss) and second term takes care of regularization.

Dual

Maximize

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n,m=1}^{N} a_n t_n a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \tag{7.32}$$

with respect to **a** and subject to the constraints

$$0 \leq a_n \leq C, \qquad n = 1, \ldots, N \tag{7.33}$$

$$\sum_{n=1}^{N} a_n t_n = 0. \tag{7.34}$$

The same as the separable case, except for the constraints $a_n \leq C$.

Model Selection

- As usual we are confronted with the problem of selecting the appropriate model complexity.
- The relevant parameters are C and any parameters of the chosen kernel function.

SVM in R

LIBSVM is available in package `e1071` in `R`.
It can also perform regression and non-binary classification.

Non-binary classification is performed as follows:

- Train $K(K-1)/2$ binary SVM's on all possible pairs of classes.
- To classify a new point, let it be classified by every binary SVM, and pick the class with the highest number of votes.
- This is done automatically by function `svm` in `e1071`.

How to in R: analysis of optdigits data

```
# SVM with radial kernel and gamma=1/62 and cost=1 (default settings)

> optdigits.svm <- svm(optdigits.train[, -c(1, 40,65)],optdigits.train[,65])

# make predictions on test set

> svm.pred <- predict(optdigits.svm,optdigits.test[, -c(1, 40,65)])

> table(optdigits.test[,65],svm.pred)
   svm.pred
      0   1   2   3   4   5   6   7   8   9
  0 177   0   0   0   1   0   0   0   0   0
  1   0 179   0   0   2   0   0   0   1   0
  2   0   7 167   0   3   0   0   0   0   0
  3   0   0   3 172   2   2   0   1   2   1
  4   0   1   0   0 179   0   0   0   1   0
  5   0   0   0   0   0 181   0   0   0   1
  6   1   0   0   0   1   0 179   0   0   0
  7   0   0   0   0   0   0   0 172   0   7
  8   0   6   0   0   3   0   0   0 159   6
  9   0   0   0   2   0   1   1   1   1 174

# accuracy on test sample is somewhat better than regularized multinomial logit
# which had "only" 95% accuracy

> sum(diag(table(optdigits.test[,65],svm.pred)))/nrow(optdigits.test)
[1] 0.967724
```

How to in R: : analysis of optdigits data

```
# tune cost parameter with cross-validation
> optdigits.svm.tune <- tune.svm(optdigits.train[, -c(1, 40,65)],
                                 optdigits.train[,65],cost=1:10)
Warning messages:
1: In svm.default(list(V2 = c(0L, 0L, 6L, 0L, 3L, 1L, 0L, 0L, 0L, 0L,  :
  Variable(s) V57 constant. Cannot scale data.
# V57 is almost always zero, let's remove it
> optdigits.svm.tune <- tune.svm(optdigits.train[, -c(1, 40,57,65)],
                                 optdigits.train[,65],cost=1:10)
# show performance for each value of the cost parameter
> optdigits.svm.tune$performances
   cost      error  dispersion
1     1 0.01490643 0.005227509
2     2 0.01202958 0.005934869
3     3 0.01229136 0.005376797
4     4 0.01176917 0.005119188
5     5 0.01150739 0.004803942
6     6 0.01150739 0.004803942
7     7 0.01176849 0.005115891
8     8 0.01150739 0.004803942
9     9 0.01176849 0.005115891
10   10 0.01176849 0.005115891
```

How to in R: : analysis of optdigits data

```
# cost=5 is the smallest among the parameter values that minimize
# cross-validation error
# We fit the SVM with this parameter value on the whole training set:

> optdigits.svm.tuned <- svm(optdigits.train[, -c(1, 40,57,65)],
                             optdigits.train[,65],cost=5)

# Generate predictions on the test set

> svm.tuned.pred <- predict(optdigits.svm.tuned,
                            optdigits.test[, -c(1, 40,57,65)])

# Compute accuracy on test set

> sum(diag(table(optdigits.test[,65],svm.tuned.pred)))/nrow(optdigits.test)
[1] 0.9727323
```