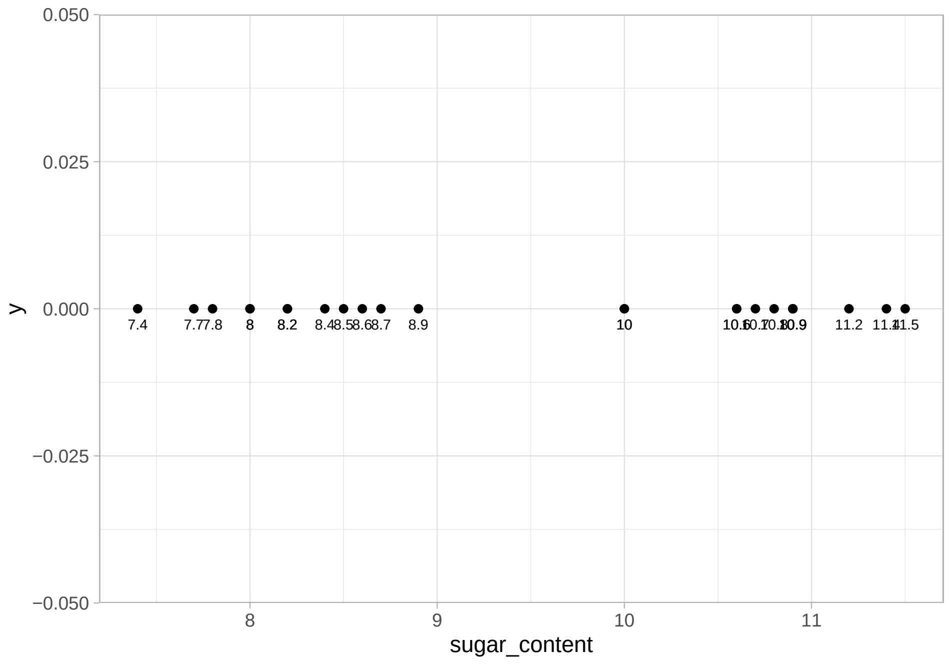
Support Vector Machines

STAT 671: Statistical Learning

Di & Shen

- Kernel Methods:
  - Supervised Learning
    - Kernel ridge regression
    - Kernel logistic regression
    - Large-margin classifiers
    - Interlude: convex optimization and duality
    - Support vector machines

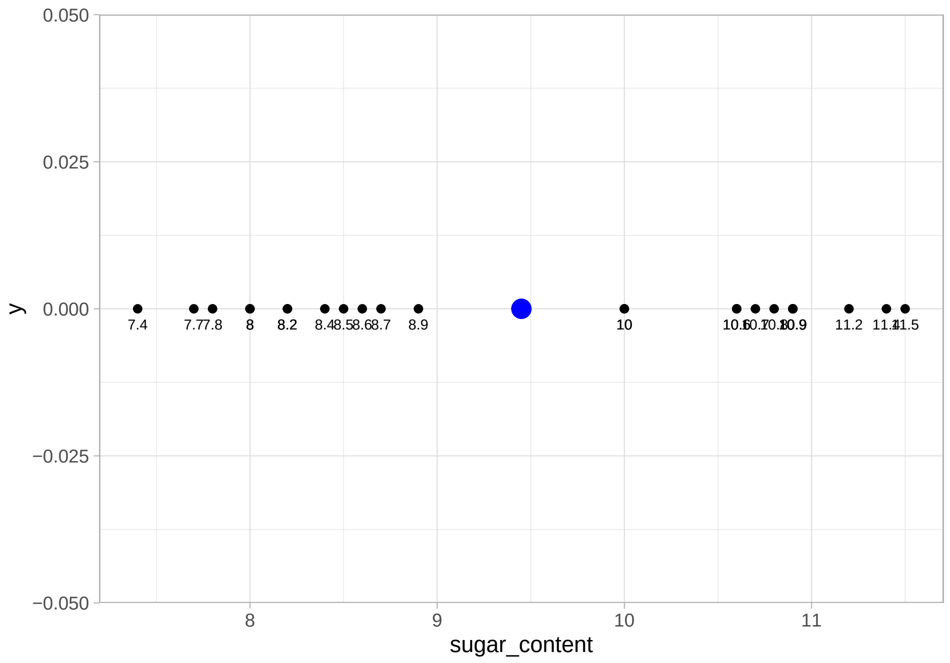
Motivation: A 1-D Example



25 soft drink sugar content measurements. The distinct clusters for identifying candidate decision boundaries.

The maximal margin separator

```r
mm_separator <- (8.9 + 10)/2
```



The maximal margin separator is at the midpoint of the two extreme points in each cluster.

Find the maximal margin separator

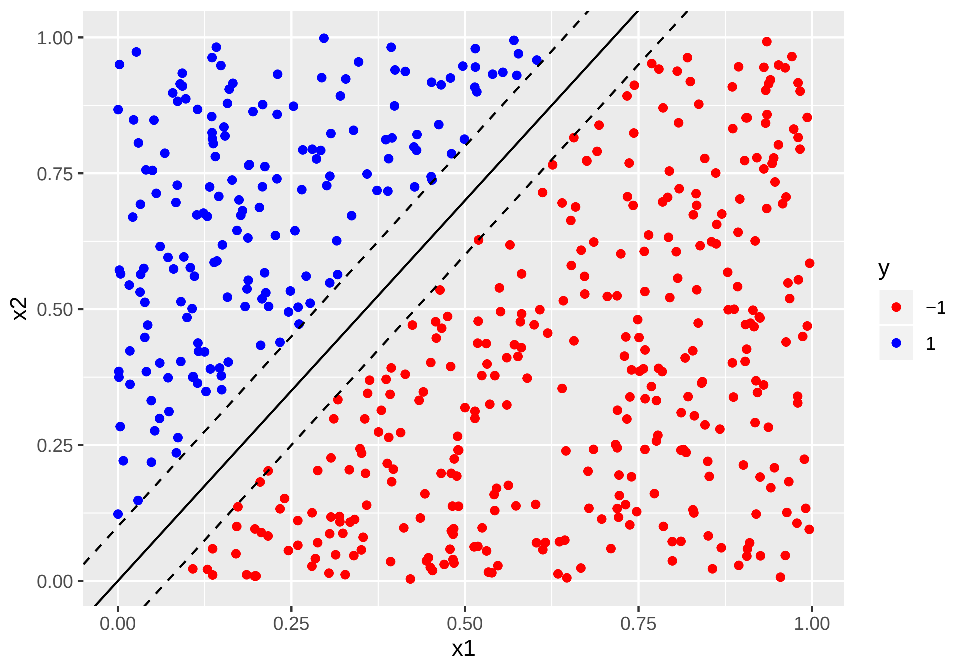
Identified two distinct clusters (classes).

A dataset in which the classes do not overlap is called separable, the classes being separated by a decision boundary.

The maximal margin separator is the decision boundary that is furthest from both classes.

The relevant points are the highest valued point in the low sugar content class and the lowest valued point in the high sugar content class.

Example 2: A 2d dataset.



A decision boundary and a margin. Add a class variable to that dataset by creating a variable y whose value is -1 or +1 depending on whether the point (x1, x2) lies below or above the straight line that passes through the origin and has slope 1.4.

Definition

Consider the ℓ_2 regularized empirical risk function

$$J(\vec{w}, \lambda) = \sum i = 1^N L(y_i, \hat{y}_i) + \lambda \|\vec{w}\|^2$$

where $\hat{y}_i = \vec{w}^T \vec{x}_i + w_0$.

If L is quadratic loss, this is equivalent to ridge regression.
If L is the log-loss, this is equivalent to logistic regression. If we replace the loss function with hinge loss function, we can ensure that the solution is sparse, so that predictions only depend on a subset of the training data, known as support vectors. This combination of the kernel trick plus a modified loss function is known as a support vector machine or SVM.

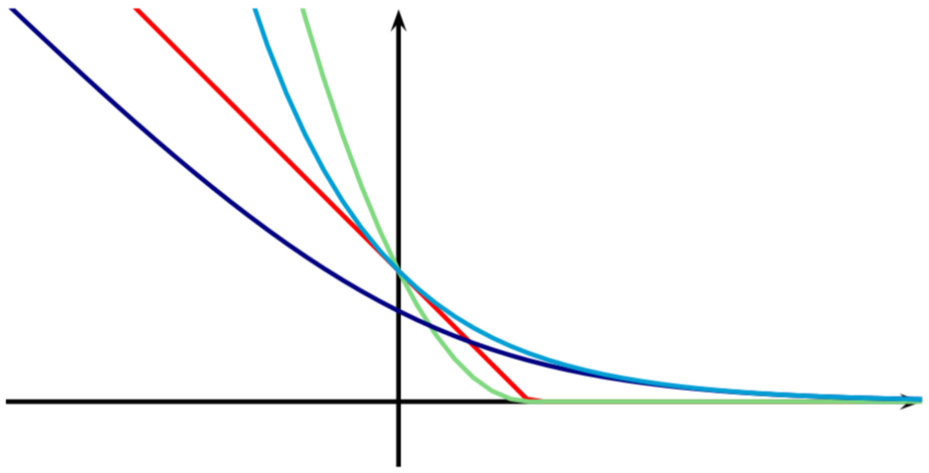
Hinge Loss

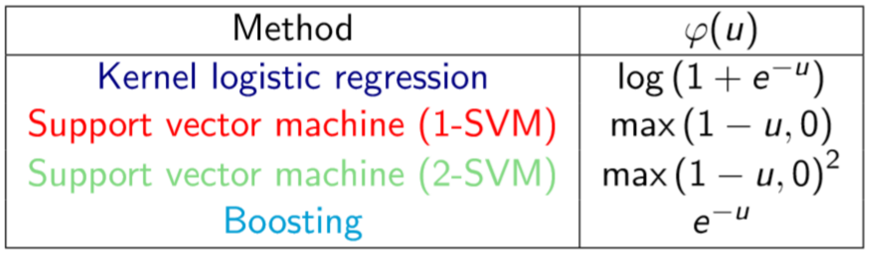
Linear support vector machines can also be interpreted as hinge loss minimization:

$$\min_{\vec{w}, b} \sum_{i=1}^{N} L(y_i, f(\vec{x}_i)) + \lambda \|\vec{w}\|^2$$

where $L(y, f(\vec{x}))$ is a hinge loss function:

$$L(y, f(\vec{x})) = \begin{cases} 1 - yf(x), & 1 - yf(x) > 0 \\ 0, & 1 - yf(x) \leqslant 0 \end{cases}$$





Method	$\varphi(u)$
Kernel logistic regression	$\log\left(1+e^{-u}\right)$
Support vector machine (1-SVM)	$\max\left(1-u,0\right)$
Support vector machine (2-SVM)	$\max\left(1-u,0\right)^2$
Boosting	e^{-u}

Definition

The hinge loss is the function $\mathbb{R} \to \mathbb{R}_+$:

$$\varphi_{hinge}(u) = \max(1-u, 0) = \begin{cases} 0 & \text{if } u \geq 1, \\ 1-u & \text{otherwise.} \end{cases}$$

SVM is the corresponding large-margin classifier, which solves:

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \varphi_{hinge}(y_i f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2$$

Support vectors

The training points with $\alpha_i \neq 0$ are called support vectors. Only support vectors are important for the classification of new points:

$$\forall x \in \mathcal{X}, \;\; f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x) = \sum_{i \in SV} \alpha_i K(x_i, x)$$

where SV is the set of support vectors.

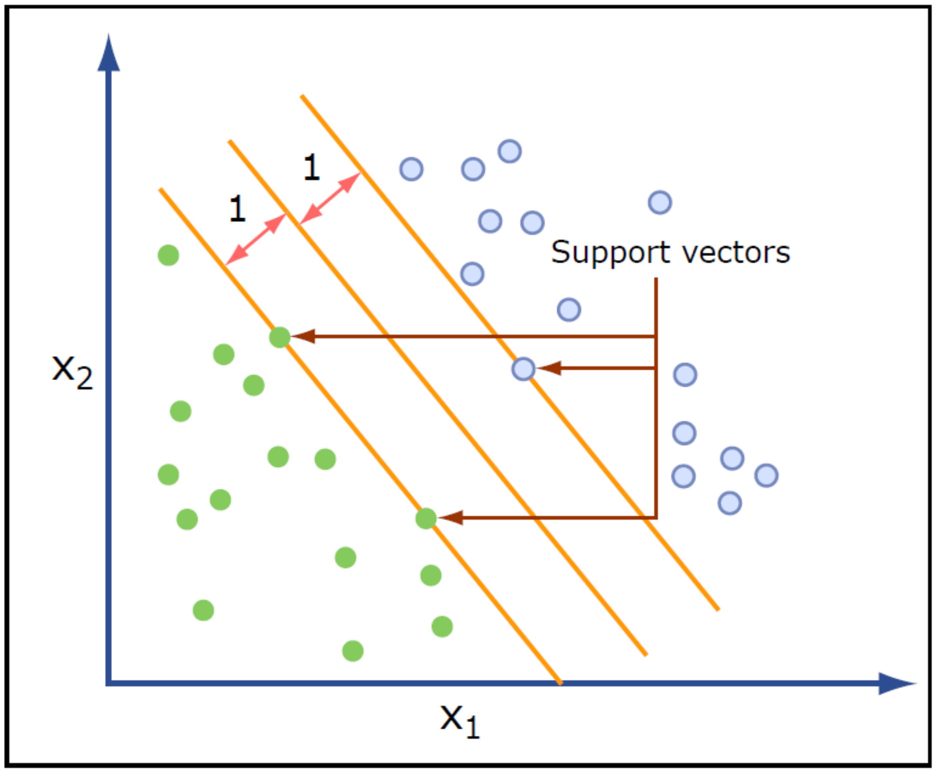


Image by MIT OpenCourseWare.

Primal form

Representation

$$\mathcal{H} : y = f(\vec{x}) = \operatorname{sign}(\vec{w}\vec{x} + b)$$

Evaluation

$$\min_{\vec{w},b} \frac{1}{2}\|\vec{w}\|^2 \quad \text{s.t.} \quad y_i(\vec{w}\vec{x}_i + b) \geqslant 1, i = 1, 2, \ldots, N$$

Primal form

By the representer theorem, the solution satisfies

$$\hat{f}(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(x_i, x)$$

, where $\hat{\alpha}$ solves

$$\min_{\alpha \in \mathbb{R}^{\times}} \frac{1}{n} \sum_{i=1}^{n} \varphi_{hinge}(y_i[K\alpha]_i) + \lambda \alpha^T K \alpha$$

This is a convex optimization problem But the objective function is not smooth (because of the hinge loss)

Solving the SVM problem

This is a classical quadratic program (minimization of a convex quadratic function with linear constraints) for which any out-of-the-box optimization package can be used.

The dimension of the problem and the number of constraints, however, are 2n where n is the number of points. General-purpose QP solvers will have difficulties when n exceeds a few thousands.

Solving the dual of this problem (also a QP) will be more convenient and lead to faster algorithms (due to the sparsity of the final solution).

Dual form

Representation $\mathcal{H} : y = f(\vec{x}) = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i y_i (\vec{x} \cdot \vec{x}_i) + b\right)$

Evaluation

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j) - \sum_{i=1}^{N} \alpha_i$$

s.t. $\sum_{i=1}^{N} \alpha_i y_i = 0 \quad \alpha_i \geqslant 0, i = 1, 2, \dots, N$

$$\max_{\alpha \in \mathbb{R}^{\ltimes}} 2 \sum_{i=1}^{n} \alpha_i y_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) = 2\alpha^T y - \alpha^T K \alpha$$

subject to: $0 \leq y_i \alpha_i \leq \frac{1}{2\lambda n}$, for $i = 1, ..., n$

Primal and Dual

$$\alpha^\star \left[1 - y_i f^\star(x_i)\right] = 0$$

$$\Longrightarrow \begin{cases} \alpha^\star > 0 & \Rightarrow y_i f^\star(x_i) = 1 \\ \alpha^\star = 0 & \Leftarrow y_i f^\star(x_i) > 1 \\ \alpha^\star < 0 & \text{not exist} \\ \text{not exist} & \Leftarrow y_i f^\star(x_i) < 1 \end{cases}$$

The examples in the first category, for which the scaled margin is 1 and the constraints are active are called support vectors. They are the closest to the decision boundary.

Slack variables

Let us introduce additional slack variables $\xi_1, ..., \xi_n \in \mathbb{R}$. The problem is equivalent to:

$$\min_{\alpha \in \mathbb{R}^{\ltimes}, \xi \in \mathbb{R}^{\ltimes}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda \alpha^T K \alpha \right\}$$

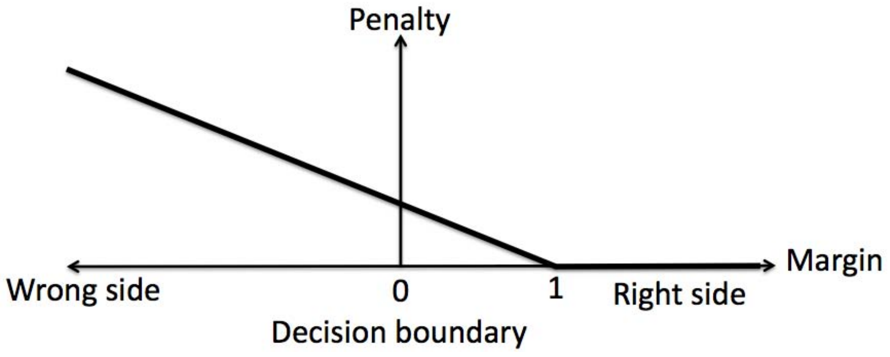
subject to:

$$\xi_i \geq \varphi_{hinge}(y_i [K\alpha]_i)$$

The objective function is now smooth, but not the constraints However it is easy to replace the non-smooth constraint by a cunjunction of two smooth constraints, because:

$$u \geq \varphi_{hinge}(v) \iff \begin{cases} u \geq 1 - v \\ u \geq 0 \end{cases}$$

slack variables



The solution is sparse in α, leading to fast algorithms for training (use of decomposition methods).

The classification of a new point only involves kernel evaluations with support vectors (fast).

Slack variables

Representation $\mathcal{H} : y = f(\vec{x}) = \text{sign}(\vec{w}\vec{x} + b)$

Evaluation $\min_{\vec{w},b} C \sum\limits_{i=1}^{N} \xi_i + \frac{1}{2}\|\vec{w}\|^2$

s.t.

$$y_i(\vec{w}\vec{x}_i + b) \geqslant 1 - \xi_i \quad \xi_i \geqslant 0, \quad i = 1, 2, \ldots, N$$

the SVM solution is $\hat{f}(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(x_i, x)$,

where $\hat{\alpha}$ solves: $\min\limits_{\alpha \in \mathbb{R}^{\ltimes}, \xi \in \mathbb{R}^{\ltimes}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda \alpha^T K \alpha \right\}$

subject to: $\begin{cases} (y_i[K\alpha]_i + \xi_i - 1 \geq 0, & \text{for} i = 1, ..., n, \\ \xi_i \geq 0, & \text{for} i = 1, ..., n \end{cases}$

Dual form with slack variables

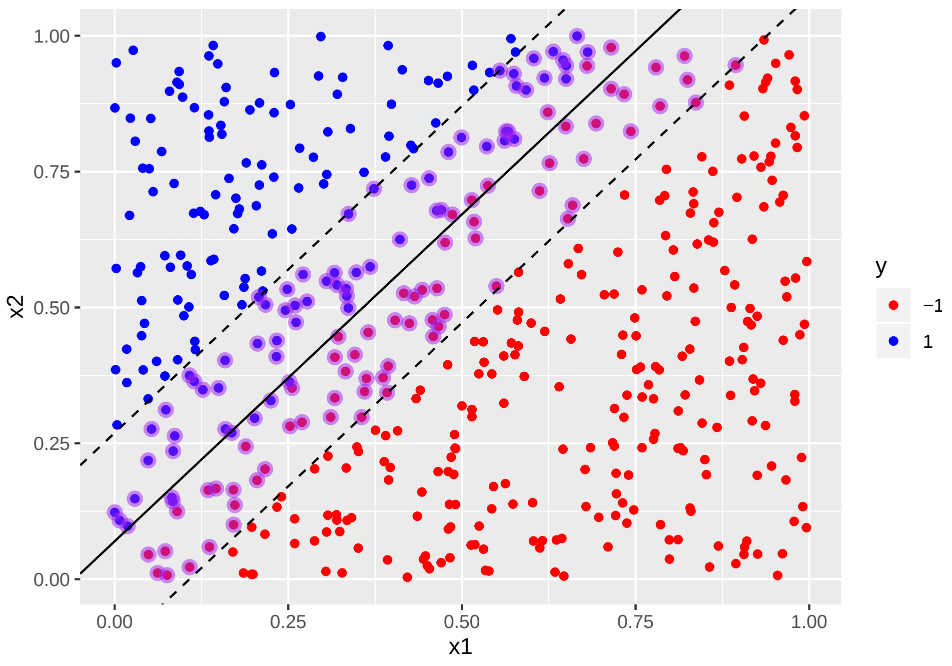
Representation $\mathcal{H} : y = f(\vec{x}) = \text{sign}\left(\sum\limits_{i=1}^{N} \alpha_i y_i (\vec{x} \cdot \vec{x}_i) + b \right)$

Evaluation $\min_{\alpha} \frac{1}{2} \sum\limits_{i=1}^{N} \sum\limits_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j) - \sum\limits_{i=1}^{N} \alpha_i$

s.t. $\sum\limits_{i=1}^{N} \alpha_i y_i = 0 \quad 0 \leqslant \alpha_i \leqslant C, i = 1, 2, \ldots, N$

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i(\vec{w} \cdot \vec{x}_i + b) \geqslant 1 \\ \alpha_i = C &\Rightarrow y_i(\vec{w} \cdot \vec{x}_i + b) \leqslant 1 \\ 0 < \alpha_i < C &\Rightarrow y_i(\vec{w} \cdot \vec{x}_i + b) = 1 \end{aligned}$$

Soft margin classifiers and nonseparatable Data



C-SVM

Often the SVM optimization problem is written in terms of a regularization parameter C instead of λ as follows:

$$\underset{f \in \mathcal{H}}{\arg\min} \left\{ \frac{1}{2} \|f\|^2_{\mathcal{H}} + C \sum_{i=1}^{n} L_{hinge}(f(x_i), y_i) \right\}$$

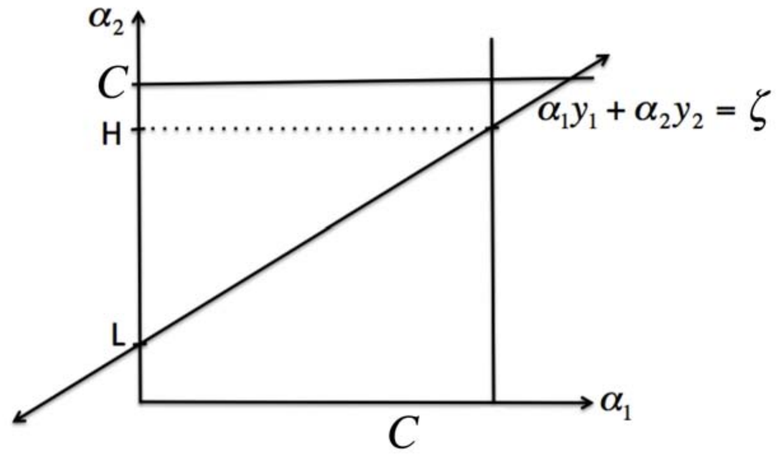
This is equivalent to our formulation with $C = \frac{1}{2\lambda n}$.

The SVM optimization problem is then:

$$\max_{\alpha \in \mathbb{R}} \left\{ 2 \sum_{i=1}^{n} \alpha_i y_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \right\}$$

subject to: $0 \leq y_i \alpha_i \leq C,$ for $i = 1, \ldots, n$ This formulation is often called C-SVM.

C-SVM



2-SVM

A variant of the SVM, sometimes called 2-SVM, is obtained by replacing the hinge loss by the square hinge loss:

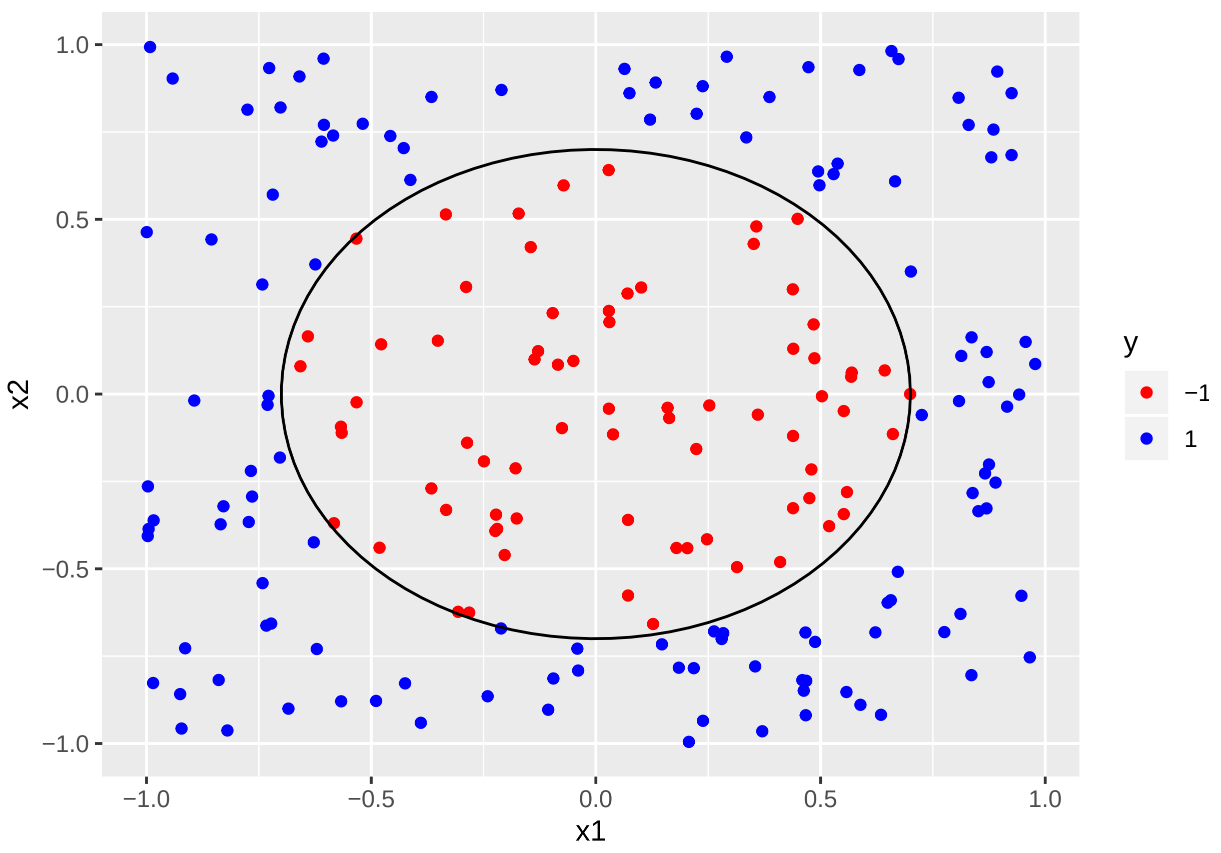
$$\min_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \varphi_{hinge}(y_i f(x_i))^2 + \lambda \|f\|_{\mathcal{H}}^2 \right\}$$

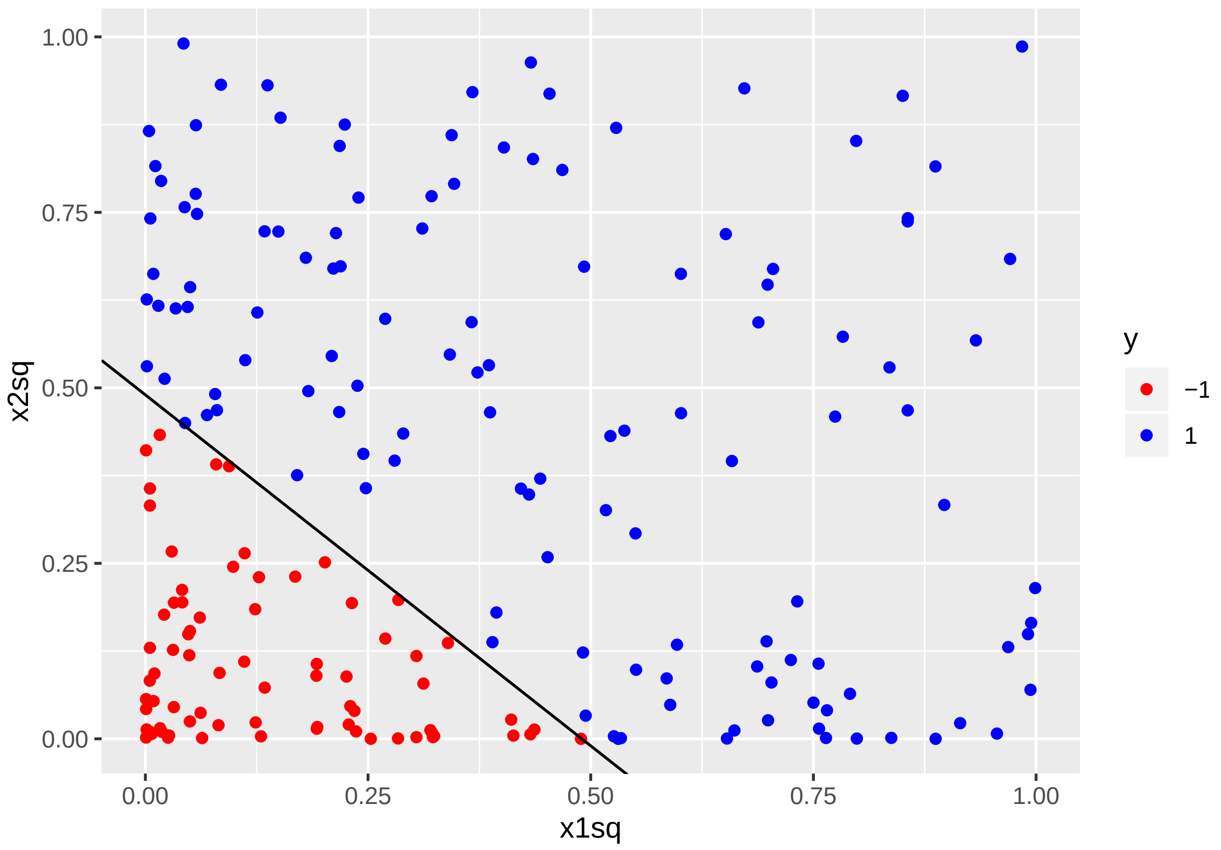
The dual problem of the 2-SVM is:

$$\max_{\alpha \in \mathbb{R}} \left\{ 2\alpha^T y - \alpha^T (K + n\lambda I)\alpha \right\}$$
$$0 \leq y_i \alpha_i, \text{ for } i = 1, ..., n$$

This is therefore equivalent to the previous SVM with the kernel $K + n\lambda I$ and $C = +\infty$

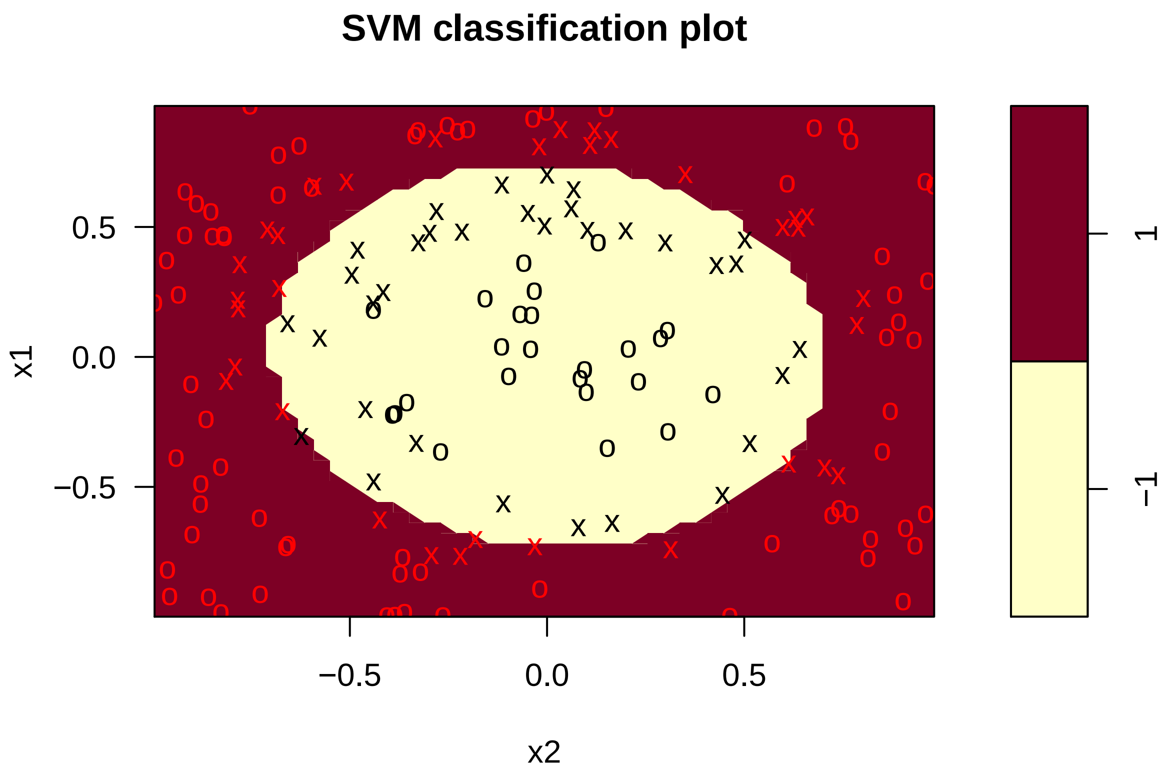
A radially 2d non-linear Example



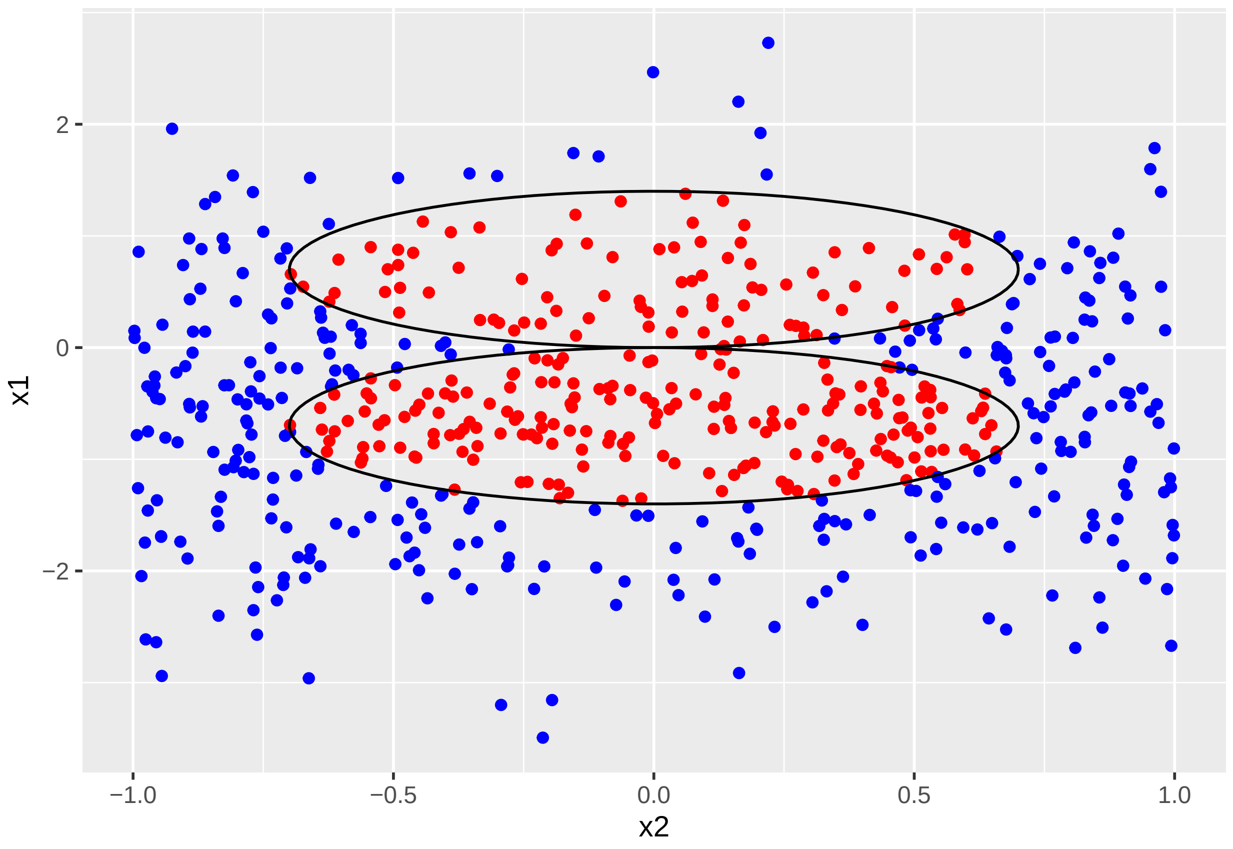


2 predictors x1 and x2, uniformly distributed between -1 and 1.

```
## [1] 0.949
```

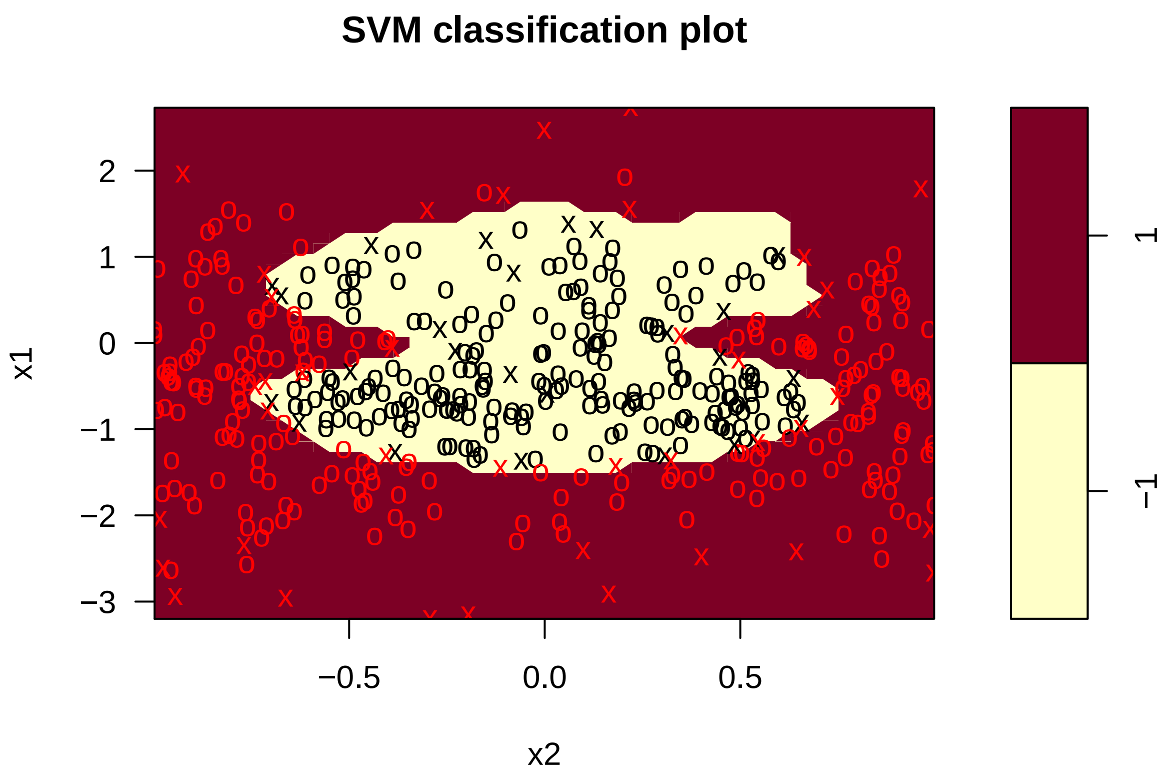


A more complex Example



600 points (x1,x2) x1 and x2 distributed differently

```
## [1] 0.964
```



Note

Historically the first “kernel method” for pattern recognition, still the most popular.

Often state-of-the-art in performance.

One particular choice of loss function (hinge loss).

Leads to a sparse solution, i.e., not all points are involved in the decomposition (compression).

Particular algorithm for fast optimization (decomposition by chunking methods).

Note

SVMs are very unnatural from a probabilistic point of view.

- First, they encode sparsity in the loss function rather than the prior.
- Second, they encode kernels by using an algorithmic trick, rather than being an explicit part of the model.
- Finally, SVMs do not result in probabilistic outputs, which causes various difficulties, especially in the multi-class classification setting.

It is possible to obtain sparse, probabilistic, multi-class kernel-based classifiers, which work as well or better than SVMs, using techniques such as the L1VM or RVM.

Summary

SVM classifiers involve three key ingredients: the kernel trick, sparsity, and the large margin principle .

The kernel trick is necessary to prevent underfitting, i.e., to ensure that the feature vector is sufficiently rich that a linear classifier can separate the data.

If the original features are already high dimensional, it suffices to use a linear kernel, $K(x, x') = x^T x'$, which is equivalent to working with the original features.

The sparsity and large margin principles are necessary to prevent overfitting, i.e., to ensure that we do not use all the basis functions. These two ideas are closely related to each other,and both arise (in this case) from the use of the hinge loss function.

However, there are other methods of achieving sparsity (such as ℓ_1), and also other methods of maximizing the margin(such as boosting).

Experiments

Chapelle, O. (2007). Training a support vector machine in the primal. Neural computation, 19(5), 1155-1178.

Algorithm 1 SVM primal training by Newton optimization

```
Function: $\boldsymbol{\beta}$ = PrimalSVM(K,Y,$\lambda$)
  $n \leftarrow$ length(Y) {Number of training points}
  if $n > 1000$ then
    $n_2 \leftarrow n/2$ {Train first on a subset to estimate the decision boundary}
    $\boldsymbol{\beta} \leftarrow$ PrimalSVM($K_{1..n_2,1..n_2}, Y_{1..n_2}, \lambda$)]
    sv $\leftarrow$ non zero components of $\boldsymbol{\beta}$
  else
    sv $\leftarrow \{1, \dots, n\}$.
  end if
  repeat
    $\boldsymbol{\beta}_{\mathsf{sv}} \leftarrow (K_{sv} + \lambda I_{n_{\mathsf{sv}}})^{-1} Y_{\mathsf{sv}}$
    Other components of $\boldsymbol{\beta} \leftarrow 0$
    sv $\leftarrow$ indices $i$ such that $y_i[K\boldsymbol{\beta}]_i < 1$
  until sv has not changed
```