Support Vector Machines

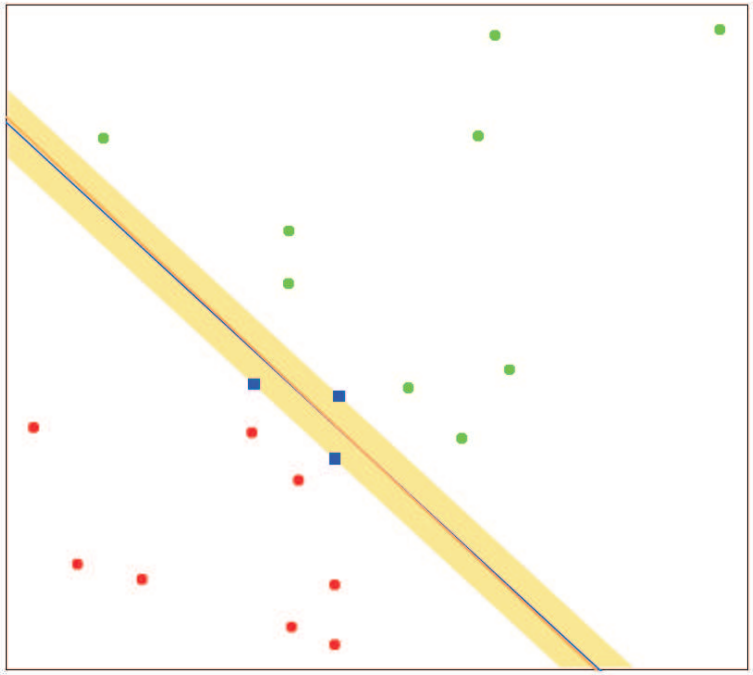
(HTF Sections 12.2, 12.3)

STA 414/2104, March 17 ♣, 2009

Jean-François Plante

Flashback

- Recall slides from February 3: Separating hyperplanes.



- Find the line (or hyper-plane) that separates the two groups with the largest possible margin.

More formally

Training data: $(x_1, y_1), (x_1, y_1), \ldots, (x_N, y_N)$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$ (a label for the class).

- Classification rule: $G(x) = \text{sign}(x^T\beta + \beta_0)$.
- Separating hyperplane

$$\max_{\beta, \beta_0, ||\beta||=1} M$$

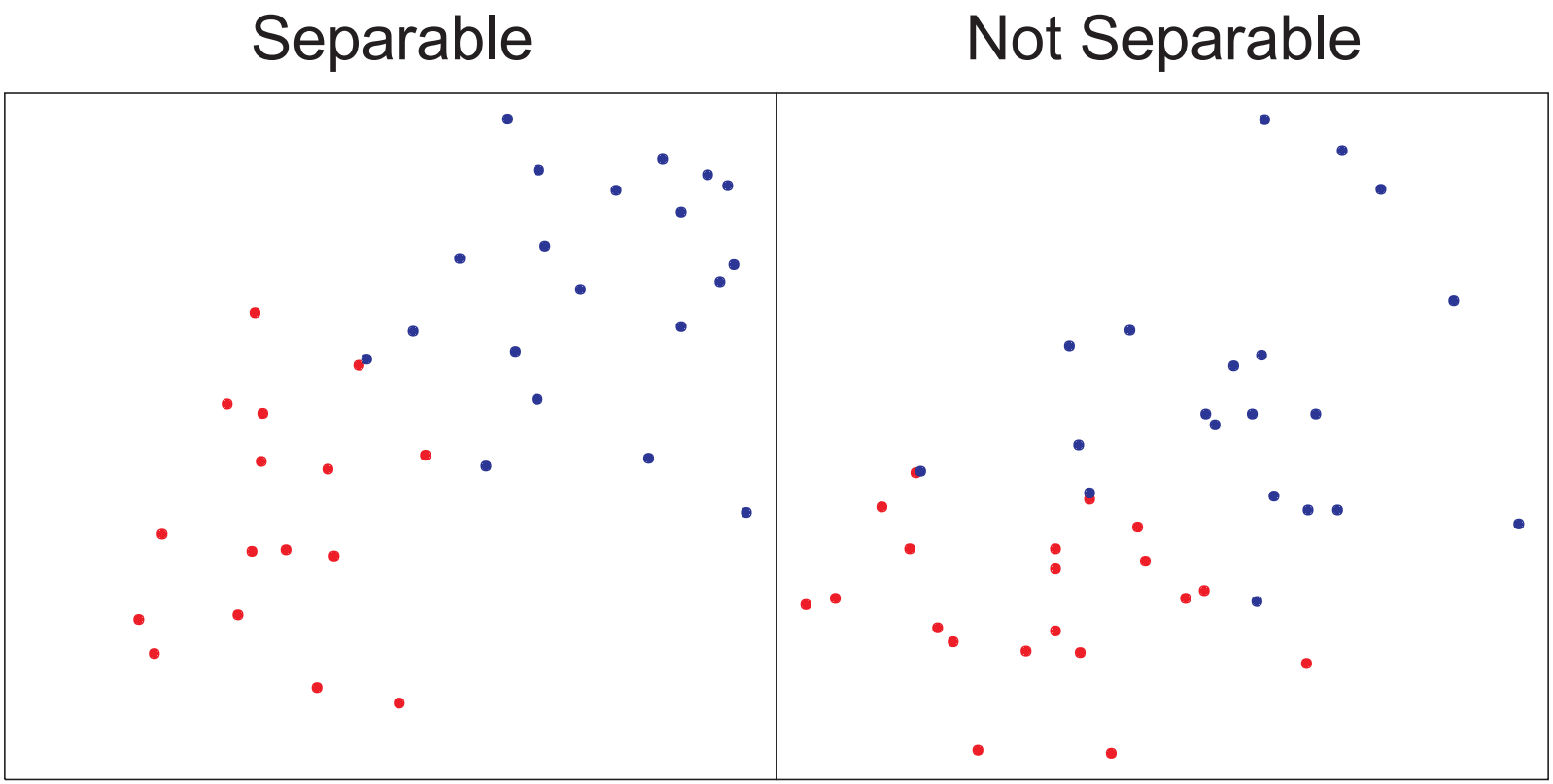
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq M, \quad i = 1, \ldots, N$$

- The sizes of M and $||\beta||$ are linked together. Without loss of generality, we can fix $M = 1/||\beta||$ rather than $||\beta|| = 1$.

$$\min_{\beta, \beta_0} ||\beta||^2$$

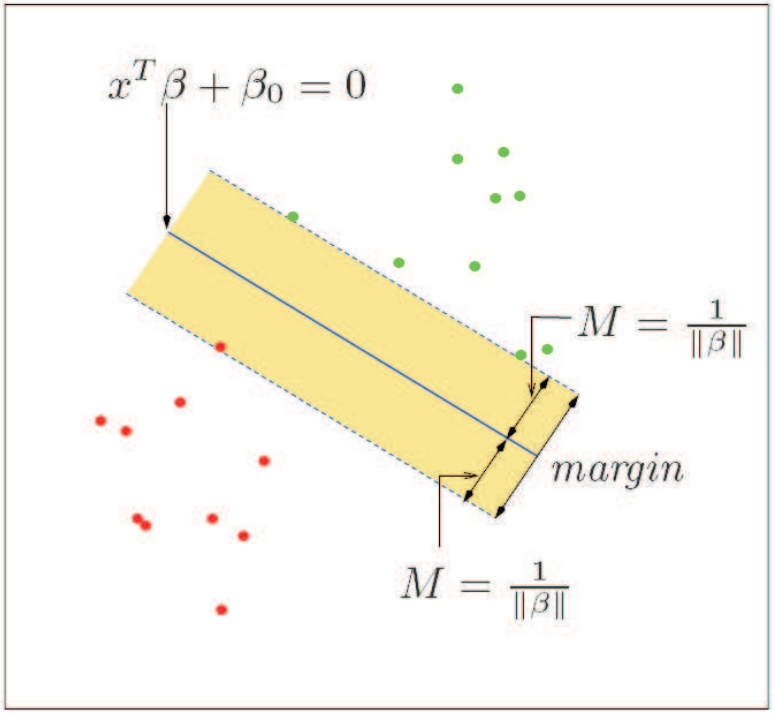
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1, \quad i = 1, \ldots, N$$

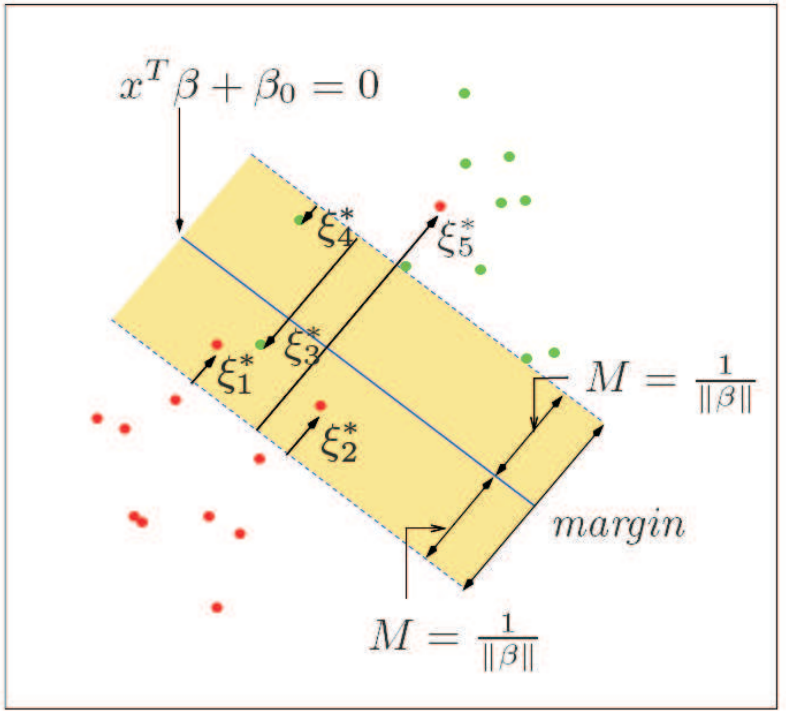
What should we do if the groups overlap?



Support Vector Classifier

We allow some points to be misclassified by introducing *slack variables* (ξ_i) that measure the distance between the misclassified point and the decision boundary. The total slack is bounded by an arbitrary constant chosen by the user.





Program

We use the constraints $y_i(x_i^T\beta + \beta_0) \geq M(1-\xi_i)$ with $\xi_i \geq 0$ and $\sum_{i=1}^N \xi_i \leq$ constant. The program is

$$\min ||\beta|| \quad \text{subject to} \begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \quad \forall i, \\ \xi_i \geq 0, \quad \sum_{i=1}^N \xi_i \leq \text{constant}. \end{cases}$$

This more convenient form is however used:

$$\min \frac{1}{2}||\beta||^2 + C\sum_{i=1}^N \xi_i$$

$$\text{subject to } \xi_i \geq 0, \quad y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \quad \forall i$$

where C is a user-defined cost parameter.

Lagrange multipliers

Lagrange multipliers lead to:

$$L_P = \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta + \beta_0) - (1-\xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i$$

with $\alpha_i, \mu_i, \xi_i \geq 0$ and must be minimized with respect to β, β_0, ξ_i.
Taking derivatives:

$$\begin{aligned} \beta &= \sum_{i=1}^{N}\alpha_i y_i x_i \\ 0 &= \sum_{i=1}^{N}\alpha_i y_i \\ \alpha_i &= C - \mu_i. \end{aligned}$$

Therefore the solution to the problem is of the form $\hat{\beta} = \sum_{i=1}^{N}\hat{\alpha}_i y_i x_i$.

Tool #1: Duality

The dual of a minimization problem is a maximization problem that shares the same solution, but may be easier to solve.

Here is an example from Strang (1986):

Primal problem:

Find the shortest distance between a line ℓ and a point p in $\mathbb{R}^3$.

Dual problem:

Find the longest distance between p and a plane containing ℓ.

Both optimization problems find the same distance, but the dual is easier, even if it involves two optimizations.

Wolfe (1961) found a dual applicable to our optimization problem.

G. Strang (1986). *Introduction to Applied Mathematics*, Wellesley-Cambridge Press.

P. Wolfe (1961). A Duality Theorem for Nonlinear Programming, *Quarterly of Applied Mathematics*, 19:3, 239-244.

Wolfe's Dual for SVM

Maximizing the function L_D below is equivalent to minimizing L_P.

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'}$$

subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$.

Much easier: Quadratic positive (semi-?)definite function of α_i constrained to an hyper-cube.

Note: Can rewrite as $L_D = -\frac{1}{2} \boldsymbol{\alpha}^T \mathrm{diag}(\mathbf{y})(\mathbf{x}^T \mathbf{x})\mathrm{diag}(\mathbf{y})\boldsymbol{\alpha} + \boldsymbol{\alpha}^T \mathbf{1}$.

Unconstrained solution: $\hat{\boldsymbol{\alpha}} = [\mathrm{diag}(\mathbf{y})(\mathbf{x}^T \mathbf{x})\mathrm{diag}(\mathbf{y})]^{-1} \mathbf{1}$.

Tool #2: Karush-Kuhn-Tucker (simplified version)

$$\min f(\mathbf{x})$$

$$\text{subject to } g_i(\mathbf{x}) \leq 0, \quad i = 1\ldots, m$$

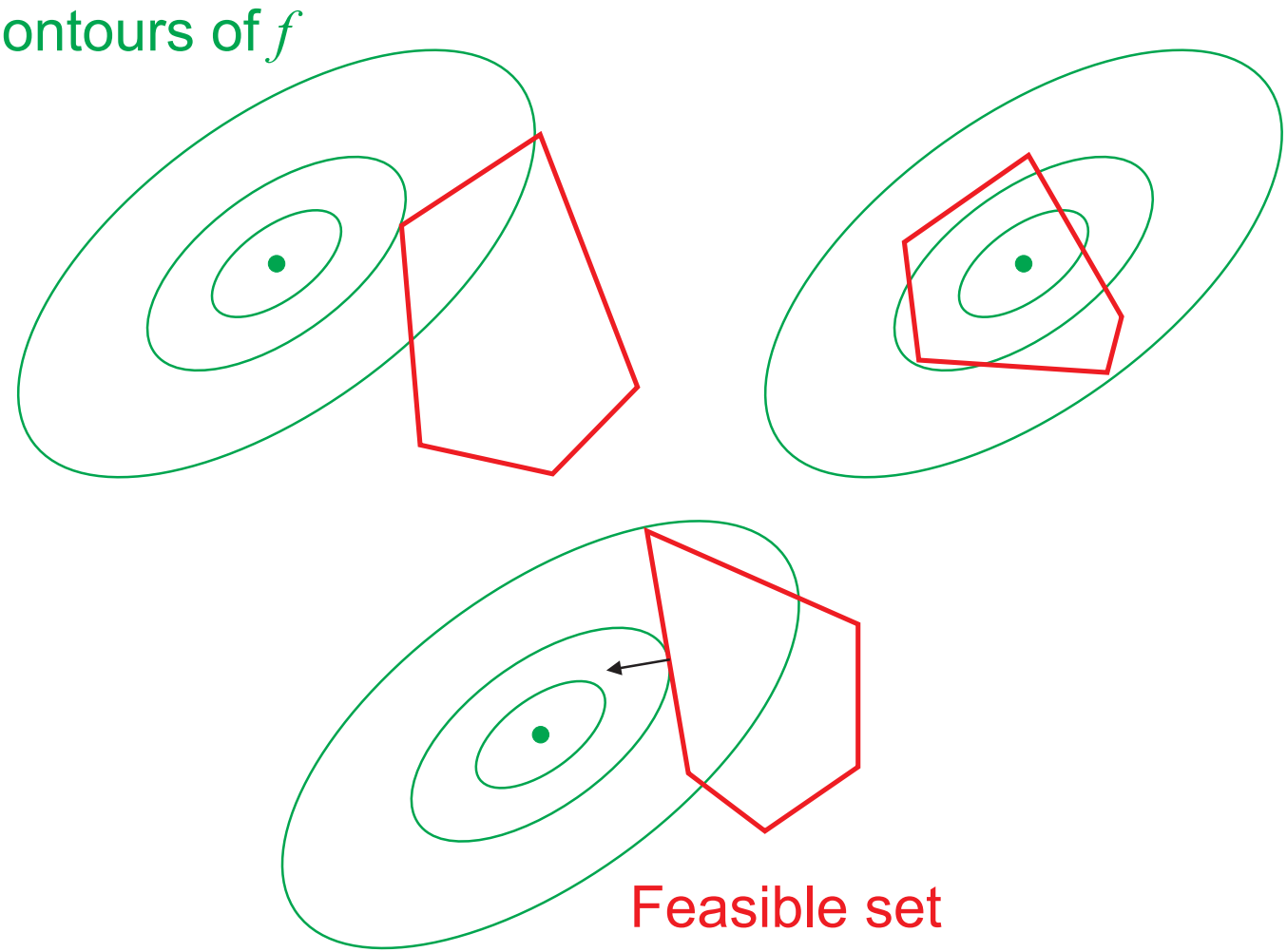
Necessary conditions: for a feasible point $\mathbf{x}^*$ to be minimum: $\exists \lambda \geq \mathbf{0}$ in $\mathbb{R}^m$ such that

$$\boldsymbol{\nabla} f(\mathbf{x}^*) + \lambda^T \boldsymbol{\nabla} \mathbf{g}(\mathbf{x}^*) = 0 \qquad \text{and} \qquad \lambda^T \mathbf{g}(\mathbf{x}^*) = 0$$

i.e. $\mathbf{x}^*$ is an interior point where the derivative of f is zero, or a point on the boundary of the feasible set. In addition, λ_i is positive only if the condition g_i is active.

Sufficient condition: if in addition, the Hessian associated to $\boldsymbol{\nabla} f(\mathbf{x}) + \lambda^T \boldsymbol{\nabla} \mathbf{g}(\mathbf{x})$ is positive semidefinite on the tangent subspace of the active constraints at $\mathbf{x}^*$.

Three cases:



Karush-Kuhn-Tucker for SVM

According tp HTF, KKT bring the additional constraints:

$$\begin{aligned} \alpha_i[y_i(x_i^T\beta + \beta_0) - (1 - \xi_i)] &= 0 \\ \mu_i\xi_i &= 0 \\ y_i(x_i^T\beta + \beta_0) - (1 - \xi_i) &\geq 0 \end{aligned}$$

Why Support Vector Machine

The first equation above implies that either x_i is within the margin, or $\alpha_i = 0$. Therefore, the estimate $\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i$ will be based only on a few data points (with $\alpha_i > 0$) called the *support vectors*.

Decision

We now have estimates of β and β_0. Given a new x, a decision is made based on

$$\hat{G}(x) = \text{sign}(x^T \hat{\beta} + \hat{\beta}_0)$$

Decision for Multiple Classes

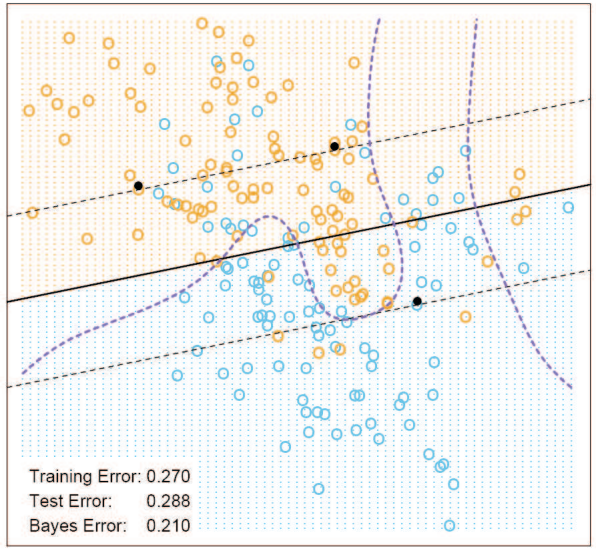
When y has $m > 2$ classes, a SVM is fitted to all of the $m(m-1)/2$ pairs of groups. Each model will allocate the new x to a category. A majority vote is used to determine the decision.

Choice of C

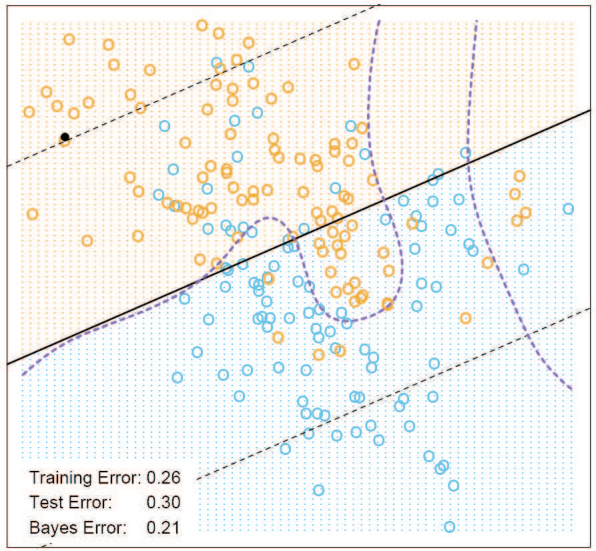
The amount of slack (C) has to be decided by the user.

A large C forces a tight margin, but a low C encourages overfitting.

$C = \infty$ requires perfect separation.



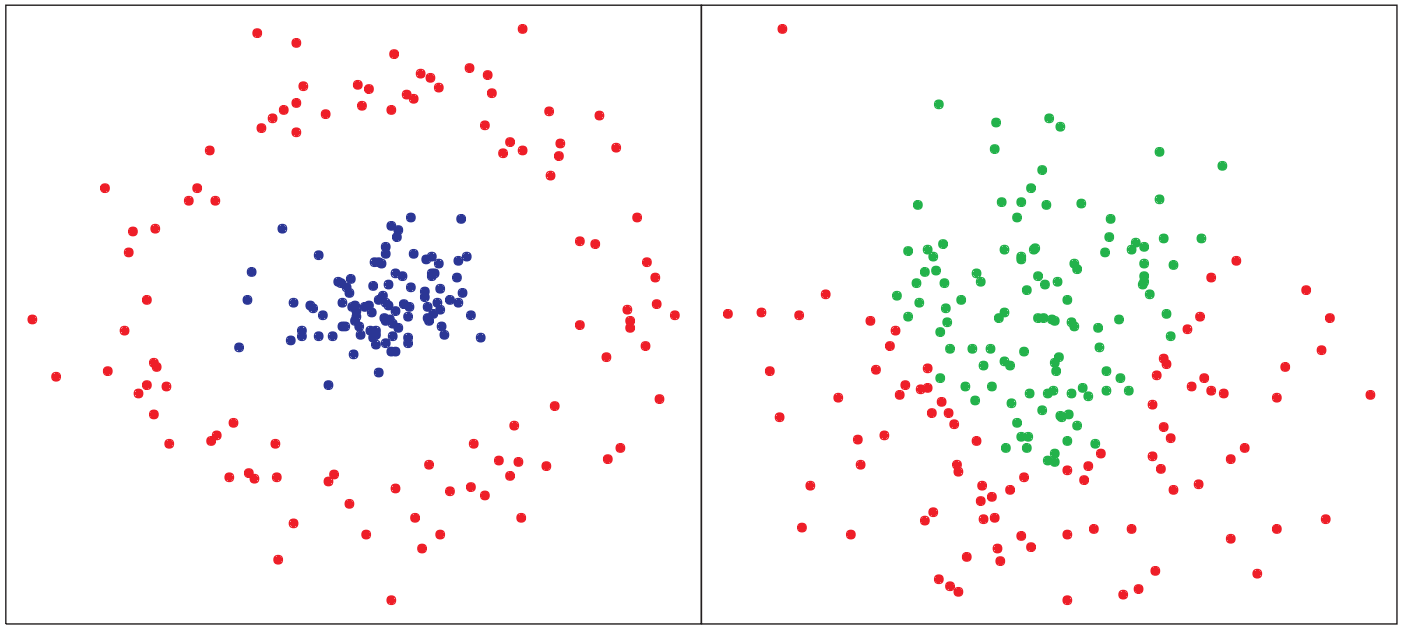
$C = 10000$



$C = 0.01$

Extensions

What can we do when the data is separable, but not by an hyperplane?



Adding Bases

We can expand the space of functions used to classify the data by adding additional bases.

Suppose $x_i = [u_i, v_i]$.

We can use $x_i = [u_i, v_i, u_i^2, v_i^2, uv]$ instead, yielding more flexibility.

The Kernel Trick

To fit the SVM, we must optimize

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'}$$

subject to $0 \leq \alpha_i \leq C$ and $\sum \alpha_i y_i = 0$.

The Kernel Trick (continued)

We can simply replace the inner product by different kernels.
For instance:

$$\begin{aligned} d\text{th-Degree polynomial}: &\quad K(x,x') = (1+\langle x,x'\rangle)^d, \\ \text{Radial basis}: &\quad K(x,x') = \exp(-\gamma||x-x'||^2), \\ \text{Neural network}: &\quad K(x,x') = \tanh(\kappa_1\langle x,x'\rangle + \kappa_2). \end{aligned}$$

With $d=2$, and $x=[u,v]$, the polynomial kernel expands into

$$K(x,x') = 1 + 2uu' + 2vv' + (uu')^2 + (vv')^2 + 2uu'vv'$$

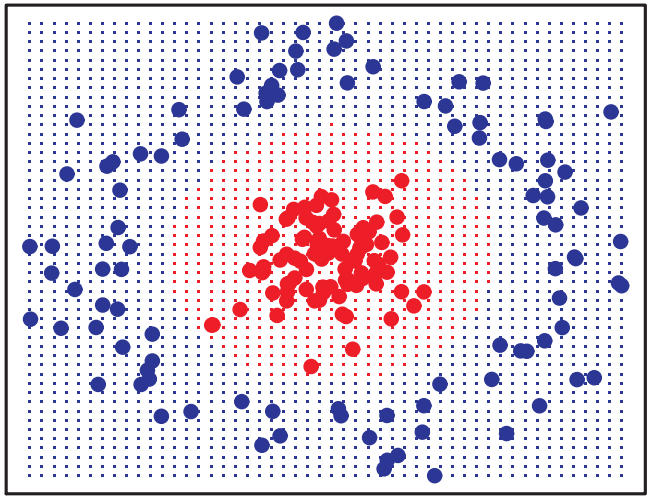
which is equivalent to using the additional bases

$$[1, \sqrt{2}u, \sqrt{2}v, u^2, v^2, \sqrt{2}uv].$$

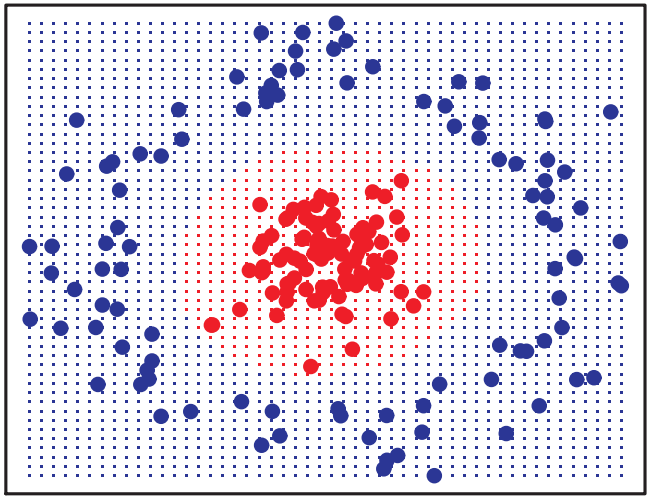
Kernels should be positive symmetric (semi-)definite functions.

Examples

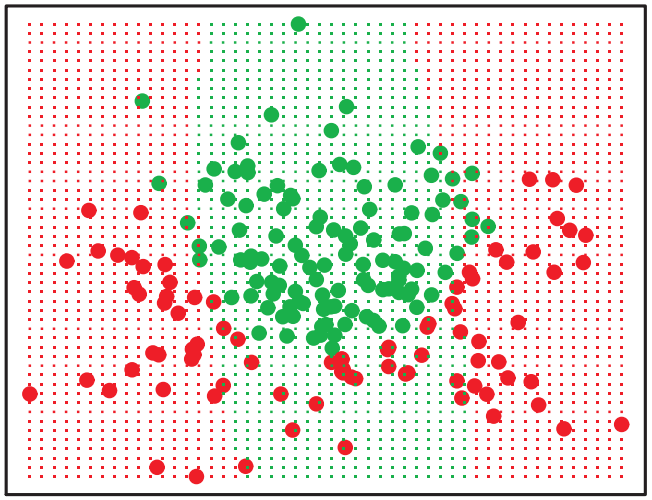
Polynomial Kernel ($d=2$)



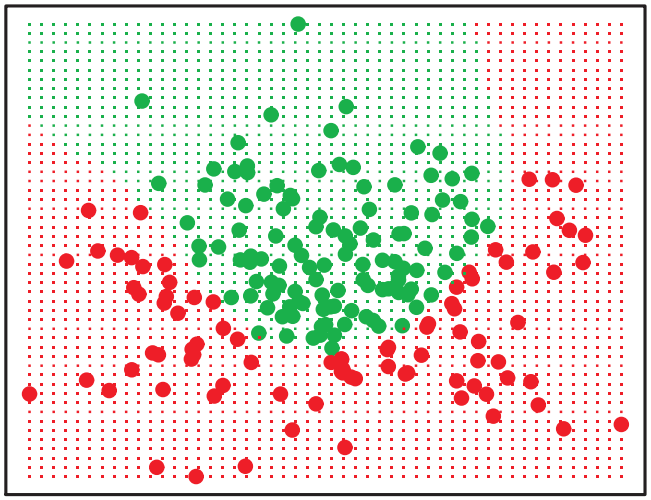
Radial basis



Polynomial Kernel ($d=2$)



Adding x^2 only



Choice of C vs Overfitting

The expression makes more sense when many bases are used. When more slack is allowed, it is easier to use a larger number of bases and therefore overfit.

Curse of Dimensionality

Adding more features or using kernels with more bases will not necessarily lead to better separation. In fact, it can make good classifiers in a subspace harder to find.

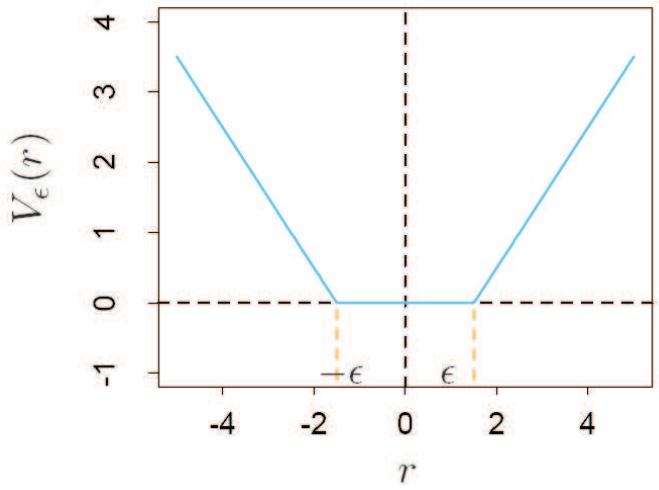
		Test Error (SE)	
	Method	No Noise Features	Six Noise Features
1	SV Classifier	0.450 (0.003)	0.472 (0.003)
2	SVM/poly 2	0.078 (0.003)	0.152 (0.004)
3	SVM/poly 5	0.180 (0.004)	0.370 (0.004)
4	SVM/poly 10	0.230 (0.003)	0.434 (0.002)
5	BRUTO	0.084 (0.003)	0.090 (0.003)
6	MARS	0.156 (0.004)	0.173 (0.005)
	Bayes	0.029	0.029

Continuous Response

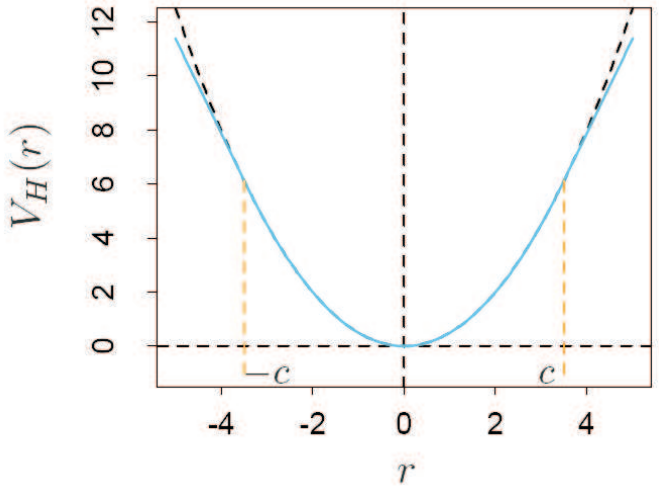
SVM can be adapted to accommodate regression (continuous y).
We consider the minimization of

$$H(\beta, \beta_0) = \sum_{i=1}^{N} V\{y_i - f(x_i)\} + \frac{\lambda}{2}||\beta||^2 \qquad \text{with } V =$$

Function for extending SVM



Huber's robust regression



The resulting optimization problem is similar to SVM.
The Kernel trick is still applicable.

Function `svm` from Library `e1071`

svm(e1071) R Documentation

Support Vector Machines

Description

`svm` is used to train a support vector machine. It can be used to carry out general regression and classification (of nu and epsilon-type), as well as density-estimation. A formula interface is provided.

Usage

```
## S3 method for class 'formula':
svm(formula, data = NULL, ..., subset, na.action =
na.omit, scale = TRUE)
## Default S3 method:
svm(x, y = NULL, scale = TRUE, type = NULL, kernel =
"radial", degree = 3, gamma = if (is.vector(x)) 1 else 1 / ncol(x),
coef0 = 0, cost = 1, nu = 0.5,
class.weights = NULL, cachesize = 40, tolerance = 0.001, epsilon = 0.1,
shrinking = TRUE, cross = 0, probability = FALSE, fitted = TRUE,
..., subset, na.action = na.omit)
```

Arguments

`formula`	a symbolic description of the model to be fit.
`data`	an optional data frame containing the variables in the model. By default the variables are taken from the environment which 'svm' is called from.
`x`	a data matrix, a vector, or a sparse matrix (object of class `matrix.csr` as provided by the package **SparseM**).
`y`	a response vector with one label for each row/component of `x`. Can be either a factor (for classification tasks) or a numeric vector (for regression).

`scale`

Data is normalized

A logical vector indicating the variables to be scaled. If `scale` is of length 1, the value is recycled as many times as needed. Per default, data are scaled internally (both `x` and `y` variables) to zero mean and unit variance. The center and scale values are returned and used for later predictions.

`type`

`svm` can be used as a classification machine, as a regresson machine, or for novelty detection. Depending of whether `y` is a factor or not, the default setting for `type` is `C-classification` or `eps-regression`, respectively, but may be overwritten by setting an explicit value.
Valid options are:

Methods we discussed →

- `C-classification`
- `nu-classification`
- `one-classification` (for novelty detection)
- → `eps-regression`
- `nu-regression`

`kernel`

Choose a kernel

the kernel used in training and predicting. You might consider changing some of the following parameters, depending on the kernel type.

linear:
*u'*v*
polynomial:
*(gamma*u'*v + coef0)^degree*
radial basis:
exp(-gamma|u-v|^2)*
sigmoid:
*tanh(gamma*u'*v + coef0)*

Tuning kernel {

`degree`

parameter needed for kernel of type `polynomial` (default: 3)

`gamma`

parameter needed for all kernels except `linear` (default: 1/(data dimension))

`coef0`

parameter needed for kernels of type `polynomial` and `sigmoid` (default: 0)

C `cost`

cost of constraints violation (default: 1)—it is the 'C'-constant of the regularization term in the Lagrange formulation.

`nu`

parameter needed for `nu-classification`, `nu-regression`, and `one-classification`