Statistical Learning for Data Science

MSDS534

Lecture 04: Support Vector Machines

Department of Statistics & Biostatistics
Rutgers University

Acknowledgement

- Some of the figures in this presentation are taken from *An Introduction to Statistical Learning, with applications in R* (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.
- Some of the figures in this presentation are taken from *Elements of Statistical Learning* (Springer, 2009) with permission from the authors: T. Hastie, R. Tibshirani and J. Friedman.
- Some of the Lab codes in this presentation are taken from *An Introduction to Statistical Learning, with applications in R* (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

Reading Assignments

- CO: §5.4.1, §5.4.2
- CO: §5.5.1 (I did not talk about it, but you may want to read), §5.5.2, §5.5.3
- ESL: §12.1, §12.2
- ISL: §9.1, §9.2

Outline

Standard Form Optimization and the Lagrangian Function

- Consider the minimization problem

$$\begin{array}{lll} \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 0 & i = 1, 2, \ldots, m \\ & h_i(x) = 0 & i = 1, 2, \ldots, p \end{array} \qquad \text{(P)}$$

  - This is called the primal problem.
  - The function f_0 is called the cost or objective function.
  - The optimization variable $x \in \mathbb{R}^n$.
  - The domain $\mathcal{D}$ of the problem is the set of x on which all the functions are defined (with finite values).
  - A $x \in \mathcal{D}$ is called feasible if it satisfies all the constraints.
  - The problem (P) is called feasible if there exists at least one feasible x.
  - Denote by p^* be the optimal value of the primal problem (P).
- Define the Lagrangian function

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{n} \nu_i h_i(x).$$

Dual Function and the Dual Problem

- The dual function is

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu).$$

  - The dual function $g(\lambda, \nu)$ is concave. This is TRUE no matter whether (P) is convex or not.
  - When $\lambda \succeq 0$, it holds that $g(\lambda, \nu) \leq p^*$.

- The dual problem is

$$\begin{array}{ll} \text{maximize} & g(\lambda, \nu) \\ \text{subject to} & \lambda \succeq 0. \end{array} \tag{D}$$

  - The dual function $g(\lambda, \nu)$ is concave, so the dual problem is convex.
  - Notice that $g(\lambda, \nu)$ can take value $-\infty$, so the dual feasible set of the problem (D) is

$$\mathcal{Q} = \{(\lambda, \nu) : \ \lambda \succeq 0, \ g(\lambda, \nu) > -\infty\}.$$

- Let d^* be the optimal value of (D). The fact $d^* \leq p^*$ is called weak duality. The nonnegative difference $p^* - d^*$ is called the duality gap.

Strong Duality and Slater's Condition

- If $d^* = p^*$, we say strong duality holds. In general, it does NOT hold!
- In particular, consider a convex problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0 \quad i = 1, 2, \ldots, m \\ & Ax = b \end{array} \tag{cP}$$

  where f_i are convex functions, and $A \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^p$.
- Slater's condition: there exists a x that belongs to the relative interior of $\mathcal{D}$, such that

$$f_i(x) < 0, \ i = 1, 2, \ldots, m \quad \text{and} \quad Ax = b.$$

- For the convex problem (cP), Slater's condition guarantees strong duality. Furthermore, the dual optimal value is attained.
- Any strict inequality $f_i(x) < 0$ in Slater's condition can be weakened to $f_i(x) \leq 0$ if f_i is linear.
- In particular, if the domain of f_0 is open, and the constraints are all linear equalities or inequalities, then Slater's condition reduces to feasibility condition.

Certificate

- A dual feasible point (λ, ν) provides a proof or certificate: $p^* \geq g(\lambda, \nu)$.
  - Strong duality means there exist arbitrarily good certificates.
- It also allows to bound how optimal a given feasible point x is:

$$f_0(x) - p^* \leq f_0(x) - g(\lambda, \nu).$$

- The gap between the primal and dual objectives $f_0(x) - g(\lambda, \nu)$ is called the duality gap associated with the primal feasible point x and the dual feasible point (λ, ν).
- If $f_0(x) = g(\lambda, \nu)$, then x is primal optimal and (λ, ν) is dual optimal, and we can think of (λ, ν) as a certificate that proves x is primal optimal.
- Suppose an optimizing algorithm produces a sequence of primal feasible x_k and dual optimal (λ_k, ν_k) for $k = 1, 2, \ldots$, and let ϵ_{abs} be a given required absolute accuracy. Then the stopping criterion

$$f_0(x_k) - g(\lambda_k, \nu_k) \leq \epsilon_{\text{abs}}$$

guarantees that x_k is ϵ_{abs}-suboptimal.

Complementary Slackness

- Suppose there exist x^* and (λ^*, ν^*) which are primal and dual optimal respectively, and $f(x^*) = g(\lambda^*, \nu^*)$, then

$$\begin{aligned} f_0(x^*) &= g(\lambda^*, \nu^*) \\ &= \inf_x \left(f_0(x) + \sum_{i=1}^m \lambda_i^* f_i(x) + \sum_{i=1}^p \nu_i^* h_i(x) \right) \\ &\leq f_0(x^*) + \sum_{i=1}^m \lambda_i^* f_i(x^*) + \sum_{i=1}^p \nu_i^* h_i(x^*) \\ &\leq f_0(x^*). \end{aligned}$$

- It implies that x^* minimizes $L(x, \lambda^*, \nu^*)$.
- Another important conclusion is that

$$\lambda_i^* f_i(x^*) = 0, \qquad \text{for } i = 1, \ldots, m,$$

which is referred to as the complementary slackness condition.

Karush-Kuhn-Tucker (KKT) Conditions

- For the problem (P), assume the functions $f_0, \ldots, f_m$ and $h_1, \ldots, h_p$ are all differentiable with open domains.
- Suppose x^* is primal optimal, (λ^*, ν^*) is dual optimal, and there is zero duality gap, then
  1. $f_i(x^*) \leq 0$, $i = 1, \ldots, m$ and $h_i(x^*) = 0$, $i = 1, \ldots, p$;
  2. $\lambda^* \succeq 0$;
  3. $\nabla f_0(x^*) + \sum_{i=1}^m \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^p \nu_i^* \nabla h_i(x^*) = 0$;
  4. $\lambda_i^* f_i(x^*) = 0$, $i = 1, 2, \ldots, m$.
- These four conditions, combined together, are called the Karush-Kuhn-Tucker (KKT) conditions.
- For the convex problem (cP), the converse is also true: for (cP), if x^* and (λ^*, ν^*) satisfy the KKT conditions, then x^* and (λ^*, ν^*) are primal and dual optimal, and the duality gap is zero.
- Consider the convex problem (cP), and assume Slater's condition holds. Then the point x^* is primal optimal if and only if there exists some (λ^*, ν^*) that, together with x^*, satisfy the KKT conditions.
- For the convex problem (cP), the differentiability of $f_0, \ldots, f_m$ can be dropped, and condition 3 can be re-formulated using sub differential.

KKT Conditions: A Special Case

Proposition (Optimality conditions for convex cost and linear constraints)

Consider the problem

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & \alpha_i' x \leq a_i, \ i = 1, \ldots, m, \\ & \beta_j' x = b_j, \ j = 1, \ldots, p, \end{aligned} \tag{1}$$

where α_i, β_j and a_i, b_j are given vectors and scalars, respectively, and $f : \mathbb{R}^p \to \mathbb{R}$ is convex and continuously differentiable. Then x^* is a global minimum if and only if there exist scalars λ_i^*, $1 \leq i \leq m$ and ν_j^*, $1 \leq j \leq p$ such that

$$\alpha_i' x \leq a_i, \ \beta_j' x = b_j, \qquad \text{(Primal Feasibility)}$$

$$\lambda_i^* \geq 0, \ 1 \leq i \leq m, \qquad \text{(Dual Feasibility)}$$

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \alpha_i + \sum_{j=1}^{p} \nu_j^* \beta_j = \mathbf{0}, \qquad \text{(Lagrangian Optimality)}$$

$$\lambda_i^* (\alpha_i' x^* - b_i) = 0, \ 1 \leq i \leq m, \qquad \text{(Complementary Slackness).}$$

KKT Conditions for Lasso

- For the convex problem (cP), the differentiability of $f_0, \ldots, f_m$ can be dropped, and Condition 3 can be re-formulated using sub differential.
- Lasso.

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 \\ \text{subject to} \quad & \|\boldsymbol{\beta}\|_1 \leq t. \end{aligned}$$

- This is a convex problem in standard form. When $t > 0$, Slater's condition guarantees the strong duality.
- $L(\boldsymbol{\beta}, \lambda) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda(\|\boldsymbol{\beta}\|_1 - t)$.
- KKT conditions: $\boldsymbol{\beta}^*$ is optimal if and only if there exists a $\lambda^* \geq 0$ such that

$$\begin{aligned} &\boldsymbol{X}'(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^*) - \lambda^* \operatorname{sign}(\boldsymbol{\beta}) = \mathbf{0}, \\ &\lambda^*(\|\boldsymbol{\beta}^*\|_1 - t) = 0. \end{aligned}$$

The Dual of a Quadratic Programming

- Consider the quadratic programming problem

$$\begin{array}{rl} \text{minimize} & \frac{1}{2}x'\boldsymbol{D}x + c'x \\ \text{subject to} & \boldsymbol{A}'x \preceq b, \end{array} \qquad \text{(QP)}$$

- $\boldsymbol{D}$ is a $n \times n$ positive definite matrix, $\boldsymbol{A}$ is a given $n \times m$ matrix, and $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ are given vectors.
- The dual function is

$$g(\lambda) = \inf_x \left[\tfrac{1}{2}x'\boldsymbol{D}x + c'x + \lambda'(\boldsymbol{A}'x - b)\right] = -\tfrac{1}{2}(\lambda'\boldsymbol{A}' + c')\boldsymbol{D}^{-1}(\boldsymbol{A}\lambda + c) - b'\lambda.$$

- The dual problem (convert maximization to minimization) can be written as

$$\begin{array}{rl} \text{minimize} & \frac{1}{2}\lambda'\left(\boldsymbol{A}'\boldsymbol{D}^{-1}\boldsymbol{A}\right)\lambda + \left(\boldsymbol{A}'\boldsymbol{D}^{-1}c + b\right)'\lambda \\ \text{subject to} & \lambda \succeq \mathbf{0}. \end{array} \qquad \text{(DQP)}$$

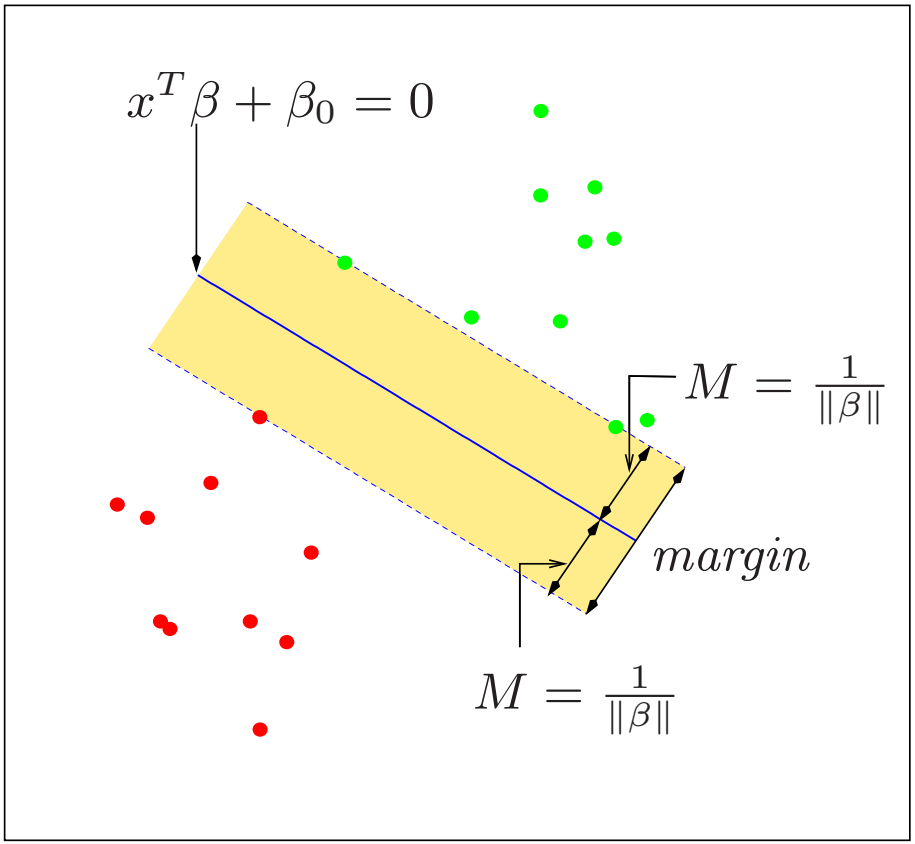
- This is also a quadratic programming problem, but it has simpler constraints and will be easier to solve computationally if $m < n$.

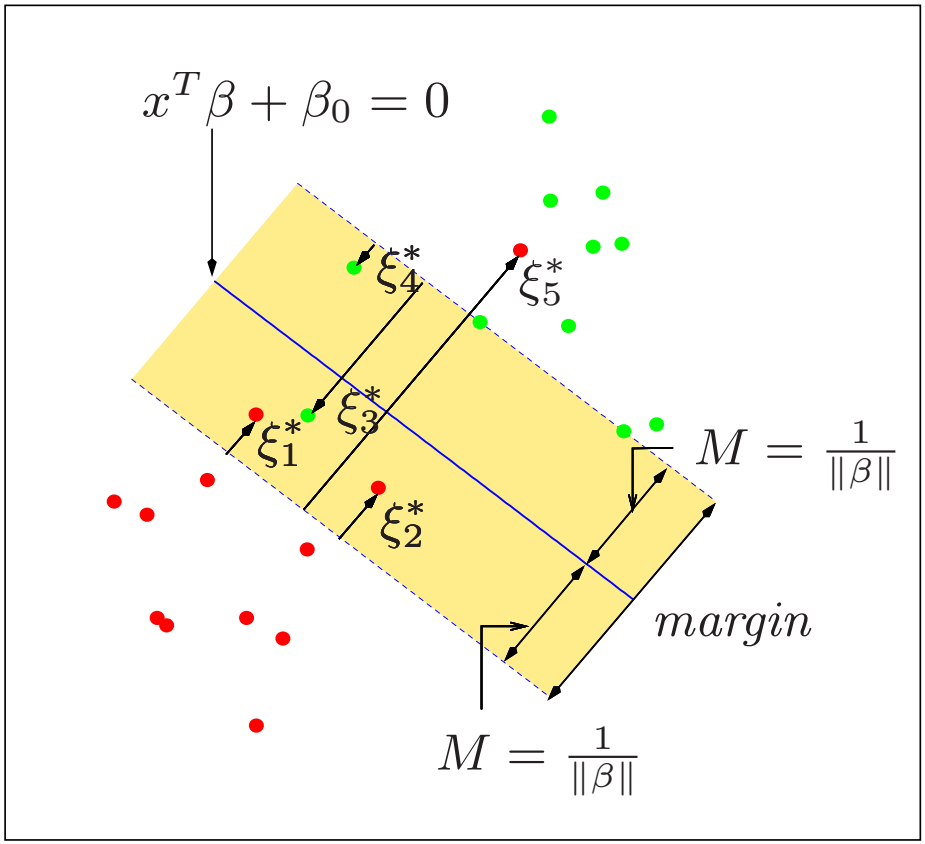
Outline

Support Vector Classifers: Separable Case

- We first assume the training data $(\boldsymbol{x}_1, g_1), \ldots, (\boldsymbol{x}_N, g_N)$ is separable.
- The linear classifier
$$f(x) = \beta_0 + \boldsymbol{x}'\boldsymbol{\beta}$$
assigns label $+1$ and -1 when $f(x) > 0$ and $f(x) < 0$ respectively.
- Recall that if $(\boldsymbol{x}_i, g_i)$ is classified correctly, than the geometric margin $g_i f(\boldsymbol{x}_i)/\|\boldsymbol{\beta}\|$ is the distance of the i-th training input $\boldsymbol{x}_i$ to the separating hyperplane in the input space.
- The support vector classifier is defined by the optimal separating hyperplane
$$\max_{\beta_0, \boldsymbol{\beta}} \min_{1 \le i \le N} g_i f(\boldsymbol{x}_i)/\|\boldsymbol{\beta}\|.$$
- Equivalently, it corresponds to the solution of the following optimization problem
$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\|\boldsymbol{\beta}\|^2 \\ \text{subject to} \quad & g_i(\beta_0 + \boldsymbol{x}_i'\boldsymbol{\beta}) \ge 1,\ 1 \le i \le N. \end{aligned}$$

Support Vector Classifiers





Dual Problem and KKT Conditions

- From now on we use $q()$ to denote the dual function.
- The Lagrangian function is

$$L(\beta_0, \boldsymbol{\beta}, \lambda) = \tfrac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta} + \sum_{i=1}^{N} \lambda_i [1 - g_i(\beta_0 + \boldsymbol{x}_i'\boldsymbol{\beta})], \quad \text{where } \lambda_i \geq 0,\ 1 \leq i \leq N.$$

- In order for $q(\lambda)$ to be finite, it must hold that $\sum_{i=1}^{N} \lambda_i g_i = 0$.
- The dual problem is

$$\text{maximize} \quad \left[-\tfrac{1}{2} \sum_{i,i'=1}^{N} \lambda_i \lambda_{i'} g_i g_{i'} \langle \boldsymbol{x}_i, \boldsymbol{x}_{i'} \rangle + \sum_{i=1}^{N} \lambda_i \right]$$

$$\text{subject to} \quad \sum_{i=1}^{N} \lambda_i g_i = 0, \quad \text{and} \quad \lambda_i \geq 0,\ 1 \leq i \leq N.$$

- The KKT conditions for the optimality of the primary solution $(\beta_0^*, \boldsymbol{\beta}^*)$ and the dual solution λ^* are (we omit the feasibility conditions):

$$\boldsymbol{\beta}^* - \sum_{i=1}^{N} \lambda_i^* g_i \boldsymbol{x}_i = 0, \quad \sum_{i=1}^{N} \lambda_i^* g_i = 0;$$

$$\lambda_i^* > 0 \Longrightarrow g_i(\beta_0^* + \boldsymbol{x}_i'\boldsymbol{\beta}^*) = 1. \quad \text{(support vector)}$$

Support Vector Classifers: Non-separable Case

- In many cases, the training data is not separable.
- Even when it is separable, we may want to regularize the classifier so that it is robust to outliers.
- Consider the following variant

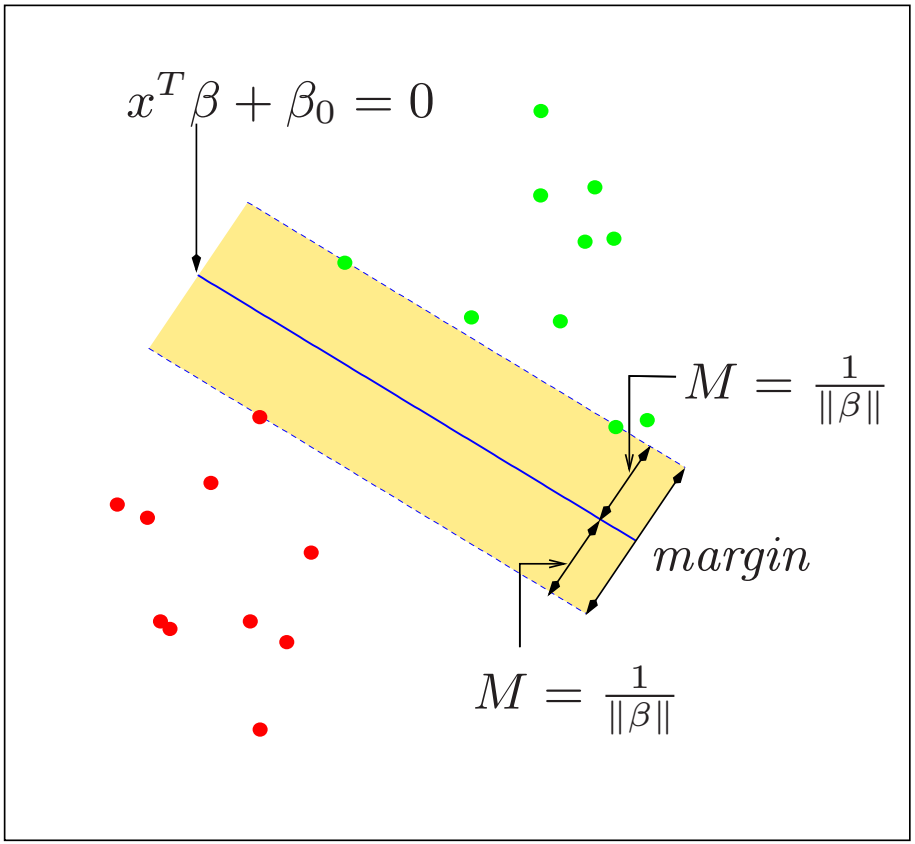
$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & g_i(\beta_0 + \boldsymbol{x}_i'\boldsymbol{\beta}) \geq 1 - \xi_i, \quad \text{and} \quad \xi_i \geq 0,\ 1 \leq i \leq N, \end{aligned}$$

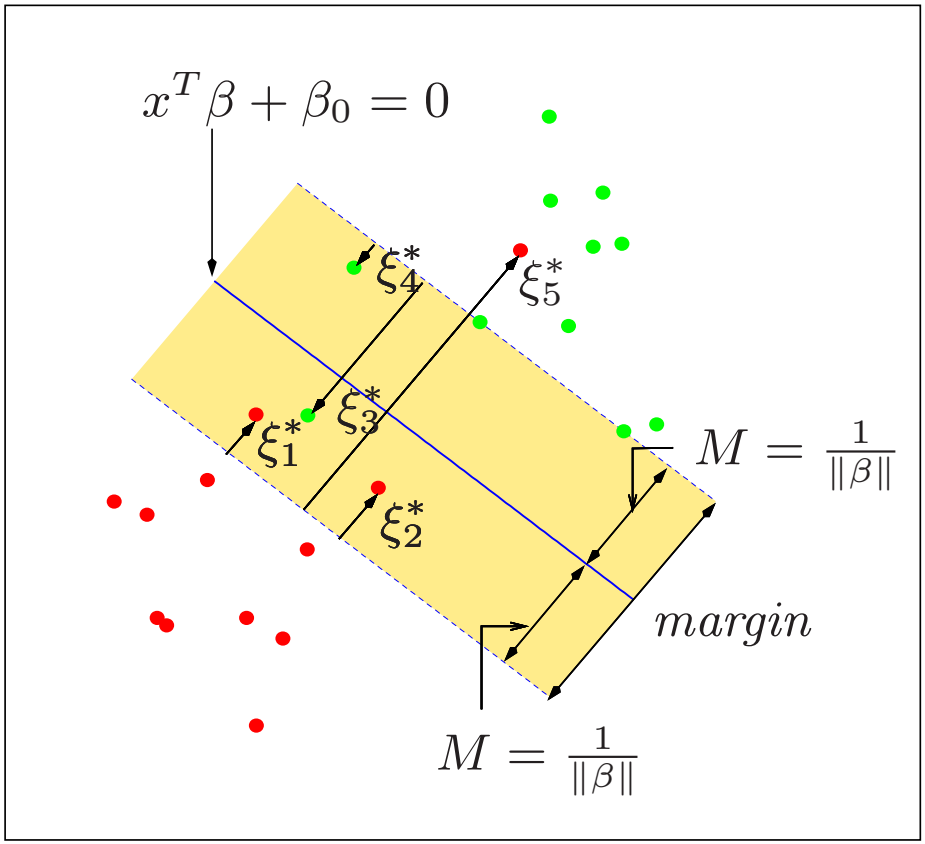
  where C is a regularization (tuning) parameter.
- The Lagrangian function is

$$L(\beta_0, \boldsymbol{\beta}, \xi; \lambda, \eta) = \tfrac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta} + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\lambda_i[1 - \xi_i - g_i(\beta_0 + \boldsymbol{x}_i'\boldsymbol{\beta})] - \sum_{i=1}^{N}\eta_i\xi_i,$$

  where $\lambda_i \geq 0,\ \eta_i \geq 0,\ 1 \leq i \leq N$.
- For the dual function $g(\lambda)$ to be finite, it must hold that $\sum_{i=1}^{N}\lambda_i g_i = 0$ and $\lambda_i + \eta_i = C$ for each $1 \leq i \leq N$.

Support Vector Classifiers





Dual Problem and KKT Conditions

- The dual problem is

$$\text{maximize} \quad \left[-\tfrac{1}{2} \sum_{i,i'=1}^{N} \lambda_i \lambda_{i'} g_i g_{i'} \langle \boldsymbol{x}_i, \boldsymbol{x}_{i'} \rangle + \sum_{i=1}^{N} \lambda_i \right]$$

$$\text{subject to} \quad \sum_{i=1}^{N} \lambda_i g_i = 0, \quad \text{and} \quad 0 \le \lambda_i \le C, \; 1 \le i \le N.$$

- The Lagrangian optimality condition for the primary solution $(\beta_0^*, \boldsymbol{\beta}^*)$ and the dual solution (λ^*, η^*) is

$$\boldsymbol{\beta}^* - \sum_{i=1}^{N} \lambda_i^* g_i \boldsymbol{x}_i = 0, \quad \sum_{i=1}^{N} \lambda_i^* g_i = 0, \quad \lambda_i^* + \eta_i^* = C, \; 1 \le i \le N;$$

- The complementary slackness condition is

$$\begin{aligned} \lambda_i^* = 0 &\Longrightarrow g_i(\beta_0^* + \boldsymbol{x}_i' \boldsymbol{\beta}^*) \ge 1, \\ \lambda_i^* = C &\Longrightarrow g_i(\beta_0^* + \boldsymbol{x}_i' \boldsymbol{\beta}^*) \le 1, \quad \text{support vector} \\ 0 < \lambda_i^* < C &\Longrightarrow g_i(\beta_0^* + \boldsymbol{x}_i' \boldsymbol{\beta}^*) = 1, \quad \text{support vector.} \end{aligned}$$

Understanding SVM

- In practice, C is treated as a tuning parameter that is generally chosen via cross-validation.
- When C is large, seek narrow margins that are rarely violated; this amounts to a classifier that is highly fit to the data, which may have low bias but high variance.
- When C is small, the margin is wider and we allow more violations to it; this amounts to fitting the data less hard and obtaining a classifier that is potentially more biased but may have lower variance.
- An observation that lies strictly on the correct side of the margin does not affect the support vector classifier!
- Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as support vectors.
  - When C is large, then the margin is narrow, and there will be few observations violating the margin, so there are few support vectors and the resulting classifier will have low bias but high variance.
  - When C is small, then the margin is wide, many observations violate the margin, and so there are many support vectors. Low variance but potentially high bias.

Derived Inputs

- Using derived inputs can lead to a nonlinear decision boundary and a possibly better classifier (*e.g.* LDA in enlarged input space).
- Let $\boldsymbol{x} \in \mathbb{R}^p$ be the original input. Denote $h(\boldsymbol{x}) = [h_1(\boldsymbol{x}), \ldots, h_Q(\boldsymbol{x})]'$ the derived input. We can apply the support vector classifier using $h(\boldsymbol{x})$.
- The dual function becomes

$$-\frac{1}{2}\sum_{k,l=1}^{N}\lambda_k\lambda_l g_k g_l \langle h(\boldsymbol{x}_k), h(\boldsymbol{x}_l)\rangle + \sum_{i=1}^{N}\lambda_i.$$

- In order to classify an object with original input $\boldsymbol{x}_0$, we compute

$$f(\boldsymbol{x}_0) = \beta_0^* + h(\boldsymbol{x}_0)'\boldsymbol{\beta}^* = \beta_0^* + \sum_{i=1}^{N}\lambda_i^* g_i \langle h(\boldsymbol{x}_i), h(\boldsymbol{x}_0)\rangle,$$

  and assign a label according to its sign.
- If the i-th object $(g_i, \boldsymbol{x}_i)$ is on the margin, $g_i[\beta_0^* + h(\boldsymbol{x}_i)'\boldsymbol{\beta}^*] = 1$, and

$$\beta_0^* = g_i - h(\boldsymbol{x}_i)'\boldsymbol{\beta}^* = g_i - \sum_{j=1}^{N}\lambda_j^* g_j \langle h(\boldsymbol{x}_i), h(\boldsymbol{x}_j)\rangle.$$

- When classifying an object, NO need to compute $\boldsymbol{\beta}^*$.
- This is particular important when using kernels, because $\boldsymbol{\beta}^*$ may be infinite dimensional.

Kernels

- We can go one step further by allowing $h(\boldsymbol{x})$ to be infinite dimensional!
- The idea is to replace the inner product $\langle h(\boldsymbol{x}_k), h(\boldsymbol{x}_l)\rangle$ by a *kernel function* $K(\boldsymbol{x}_k, \boldsymbol{x}_l)$.
- Generally, linear boundaries in the enlarged (infinite dimensional) input space achieves better training separation.
- The linear decision boundary becomes nonlinear when it is projected to the original input space.
- The following are some frequently used kernel functions.

$$
\begin{aligned}
d\text{-th degree polynomial: } & K(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\text{scale} \cdot \langle \boldsymbol{x}_1, \boldsymbol{x}_2\rangle + \text{offset})^d; \\
\text{Radial basis or Gaussian: } & K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp\left(-\alpha \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2\right), \quad \text{for some } \alpha > 0; \\
\text{Hyperbolic tangent: } & K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \tanh\left(\text{scale} \cdot \langle \boldsymbol{x}_1, \boldsymbol{x}_2\rangle + \text{offset}\right).
\end{aligned}
$$

- It is not difficult to verify the first two are indeed kernels. The third one is actually not a kernel, but it is linked to neural network and has been used in practice as well.

Mercer's Theorem

In general, the following theorem characterizes what kind of kernel functions can be represented as inner products.

Theorem (Mercer's Theorem)

Let $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ be a symmetric function. The following two statements are equivalent.

(i) *There exists an infinite dimensional function $h(\boldsymbol{x}) = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots]$ such that for any pair $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^p$, it holds that $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \langle h(\boldsymbol{x}_1), h(\boldsymbol{x}_2) \rangle$.*

(ii) *For any $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$, the $n \times n$ kernel matrix $[K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})]_{i,i'=1}^n$ is positive semi-definite.*

Sequential Minimal Optimization Algorithm

- The sequential minimal optimization (SMO) algorithm uses the idea of coordinate ascent.
- Due to the constraint $\sum_{i=1}^{n} \lambda_i g_i = 0$ of the dual problem, it is not allowed to optimize over one coordinate λ_i, while holding all other coordinates as fixed.
- SMO selects a pair (λ_i, λ_j), fixes all other $N-2$ coordinates, and expresses λ_j as a function of λ_i

$$\lambda_j = -g_j g_i \lambda_i - g_j \sum_{k \neq i,j} g_k \lambda_k.$$

- Now the dual function is a uni-variate quadratic function of λ_i, so it can be optimized easily.
- Although this is still a constrained problem, because both λ_i and λ_j have to be between 0 and C; it can be solved explicitly to produce an update on the pair (λ_i, λ_j).
- The algorithm will then check the convergence criterion (KKT), and proceed to select the next pair if it is not satisfied within some tolerance.

Example: Gaussian Mixtures

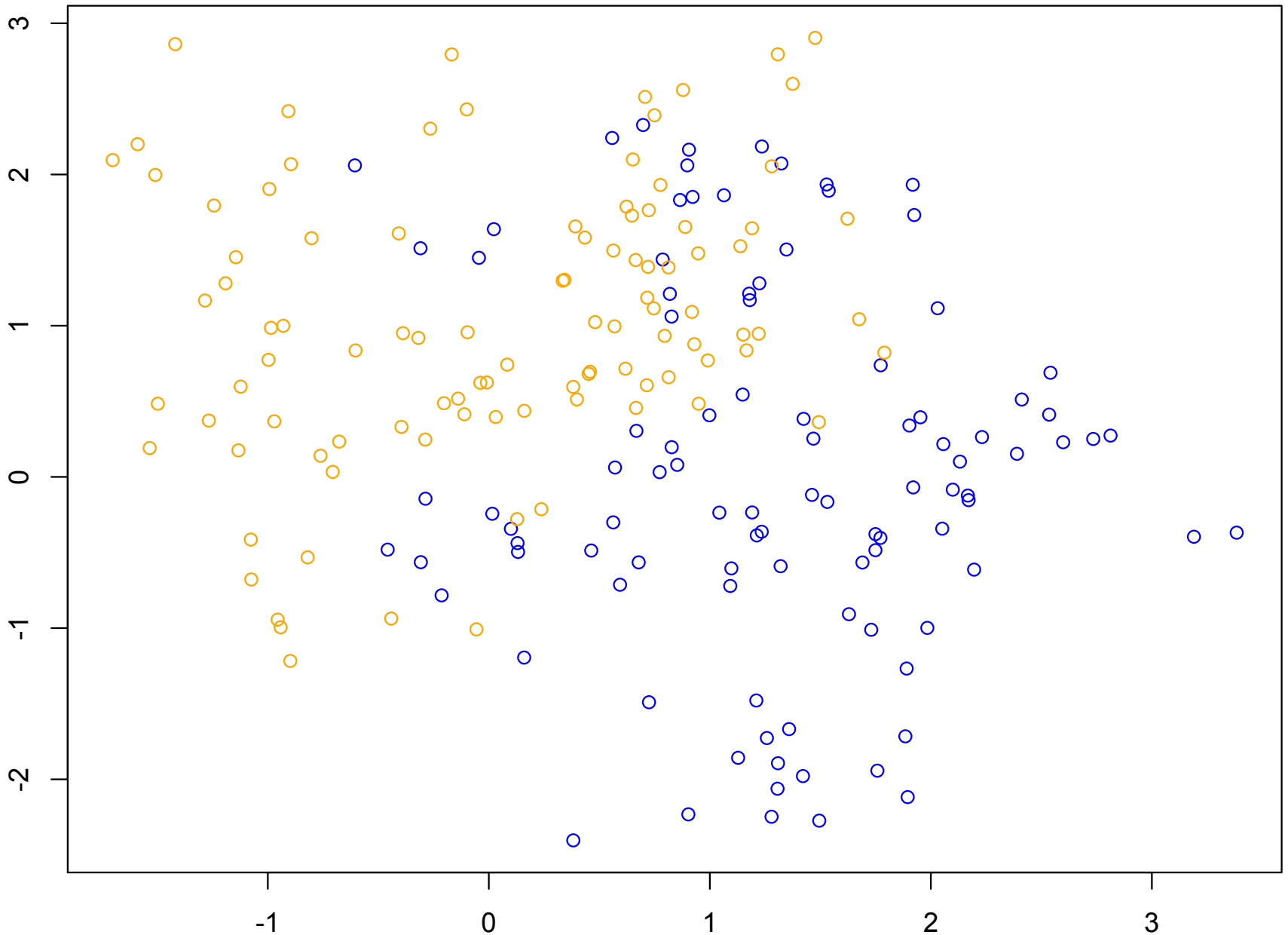
- Generate 10 means $\boldsymbol{\mu}_k$ from the bivariate Gaussian distribution $\mathcal{N}_2[(1,0)', \boldsymbol{I}]$ and label this class 1.
- Generate another 10 means from the distribution $\mathcal{N}_2[(0,1)', \boldsymbol{I}]$ and label this class -1.
- For each class, generate 100 observations as follows: for each observation, pick a $\boldsymbol{\mu}_k$ at random with probability $1/10$, and then generate a $\mathcal{N}_2(\boldsymbol{\mu}_k, \boldsymbol{I}/5)$.
- For each class, generate 5000 observations for testing.

Example: Gaussian mixtures

```
library(e1071)
library(stringr)
library(MASS)
library(mvtnorm)

###################### Gaussian mixtures
set.seed(123)
n.train=100
n.test=5000
ind.tr=c(1:n.train,(n.test+n.train+1):(n.test+2*n.train))
ind.te=c((n.train+1:n.test+n.train),(n.test+2*n.train+1):(2*n.test+2*n.train))
y=c(rep(1,n.test+n.train),rep(-1,n.test+n.train))
y=as.factor(y)
X=array(0,dim=c(length(y),2))
m1=rmvnorm(10,c(1,0),diag(2))
m2=rmvnorm(10,c(0,1),diag(2))
for (i in 1:(n.train+n.test)){
  X[i,]=rmvnorm(1,m1[sample(1:10,1),],diag(2)/5)
  X[n.train+n.test+i,]=rmvnorm(1,m2[sample(1:10,1),],diag(2)/5)
}
GM=data.frame(y,X)
par(mar=c(2,2,.5,.5))
plot(X[ind.tr,1],X[ind.tr,2],type="n",xlab="X1",ylab="X2")
points(X[1:n.train,1],X[1:n.train,2],col="blue")
points(X[(n.test+n.train+1):(n.test+2*n.train),1],
       X[(n.test+n.train+1):(n.test+2*n.train),2],col="orange")
```

Training set



Logistic regression

```
> GM.logit=glm(y~.,data=GM[ind.tr,],family="binomial")
> pred.logit=predict(GM.logit,GM[ind.tr,],type="response")
> sum(sign(pred.logit-.5)!=y[ind.tr])/(2*n.train)
[1] 0.185
> pred.logit=predict(GM.logit,GM[ind.te,],type="response")
> sum(as.factor(sign(pred.logit-.5))!=y[ind.te])/(2*n.test)
[1] 0.1886
> GM.logit=glm(y~X1+X2+I(X1*X2)+I(X1^2)+I(X2^2),data=GM[ind.tr,]
                ,family="binomial")
> pred.logit=predict(GM.logit,GM[ind.tr,],type="response")
> sum(sign(pred.logit-.5)!=y[ind.tr])/(2*n.train)
[1] 0.195
> pred.logit=predict(GM.logit,GM[ind.te,],type="response")
> sum(as.factor(sign(pred.logit-.5))!=y[ind.te])/(2*n.test)
[1] 0.1737
```

Linear discriminant analysis

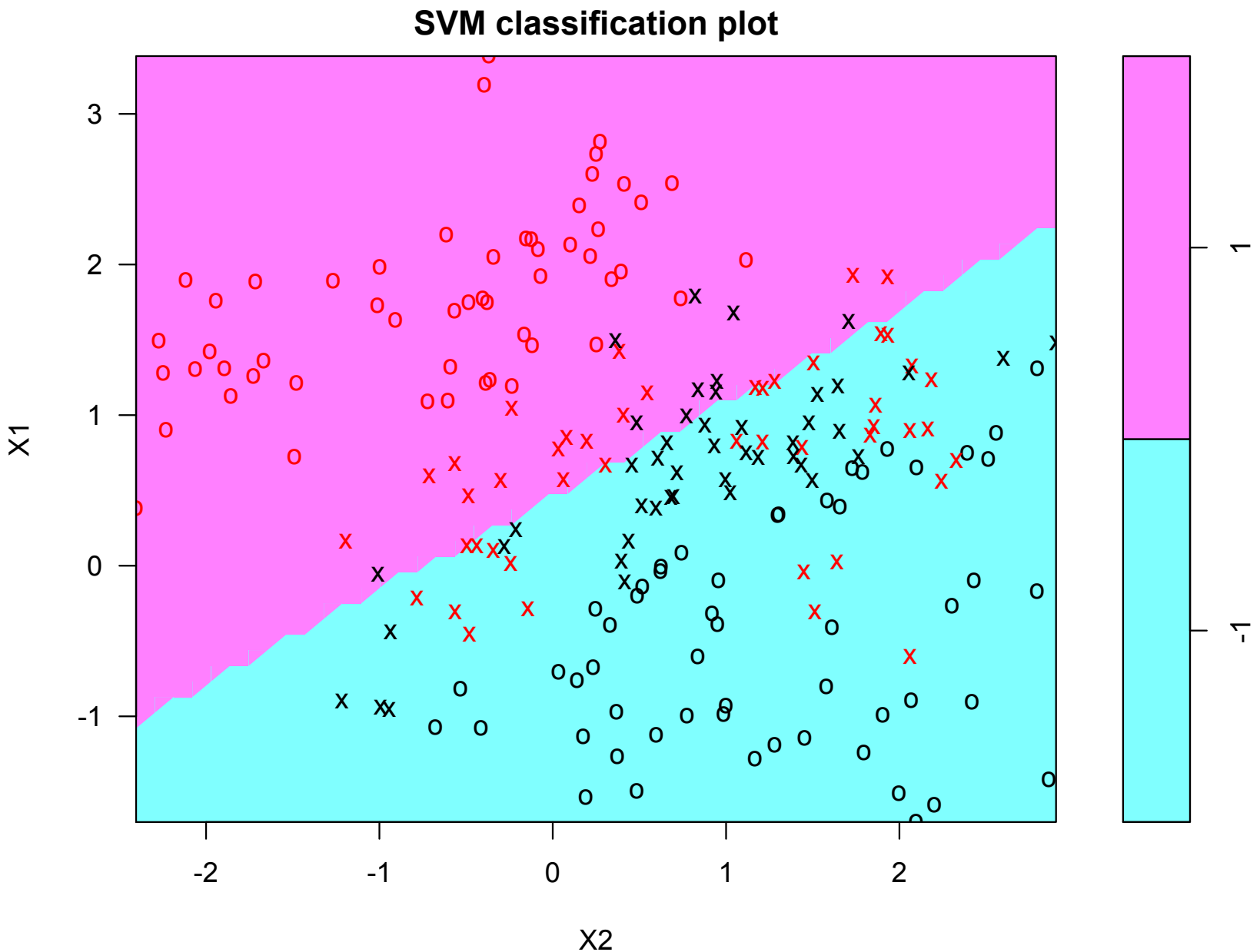
```
> GM.lda=lda(y~.,data=GM[ind.tr,])
> pred.lda=predict(GM.lda,GM[ind.tr,])
> sum(pred.lda$class!=y[ind.tr])/(2*n.train)
[1] 0.185
> pred.lda=predict(GM.lda,GM[ind.te,])
> sum(pred.lda$class!=y[ind.te])/(2*n.test)
[1] 0.1866
>
> GM.lqa=lda(y~X1+X2+I(X1*X2)+I(X1^2)+I(X2^2),data=GM[ind.tr,])
> pred.lqa=predict(GM.lqa,GM[ind.tr,])
> sum(pred.lqa$class!=y[ind.tr])/(2*n.train)
[1] 0.17
> pred.lqa=predict(GM.lqa,GM[ind.te,])
> sum(pred.lqa$class!=y[ind.te])/(2*n.test)
[1] 0.1818
```

Support vector machine

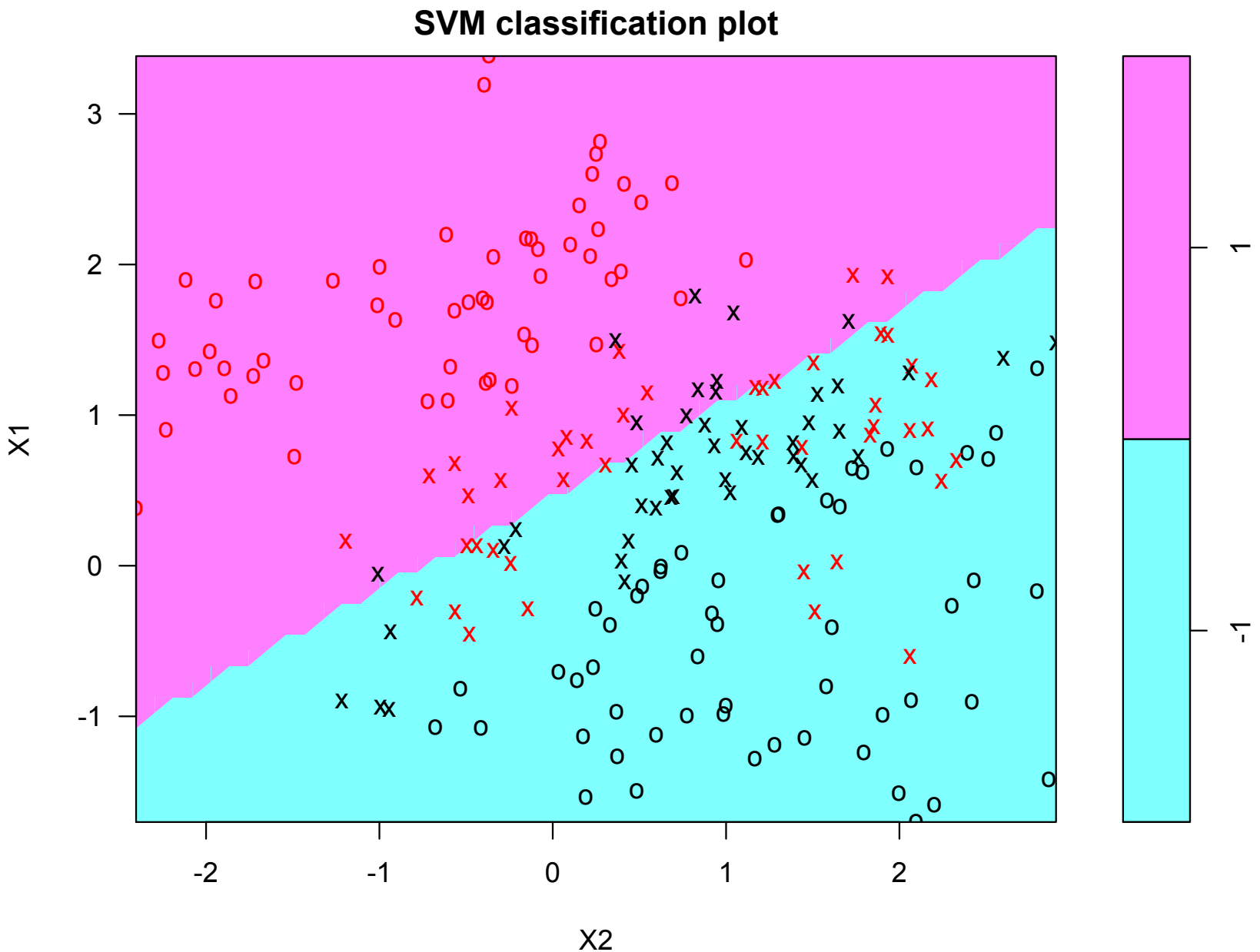
```
> GM.svm=svm(y~.,cost=1000,gamma=1,type="C-class",kernel="linear"
             ,data=GM[ind.tr,])
> pred.svm=predict(GM.svm,newdata=GM[ind.tr,])
> sum(pred.svm!=GM[ind.tr,1])/(2*n.train)
[1] 0.175
> pred.svm=predict(GM.svm,newdata=GM[ind.te,])
> sum(pred.svm!=GM[ind.te,1])/(2*n.test)
[1] 0.1788
> plot(GM.svm, GM[ind.tr,])

> GM.svm=svm(y~.,cost=10,gamma=1,type="C-class",kernel="linear"
             ,data=GM[ind.tr,])
> pred.svm=predict(GM.svm,newdata=GM[ind.tr,])
> sum(pred.svm!=GM[ind.tr,1])/(2*n.train)
[1] 0.175
> pred.svm=predict(GM.svm,newdata=GM[ind.te,])
> sum(pred.svm!=GM[ind.te,1])/(2*n.test)
[1] 0.1785
> plot(GM.svm, GM[ind.tr,])
```

Linear kernel: heavy penalty



Linear kernel: light penalty

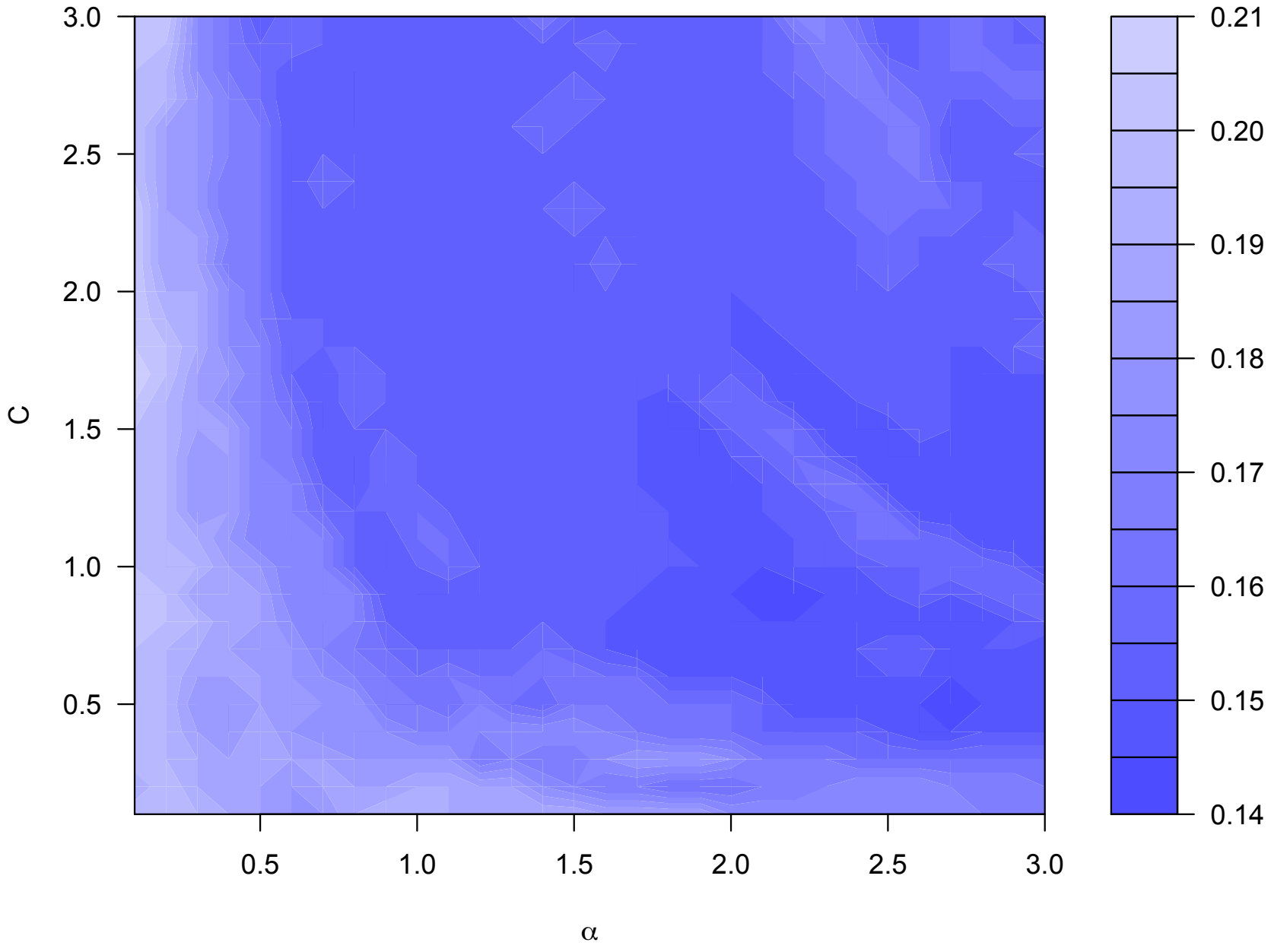


Example: Gaussian mixtures

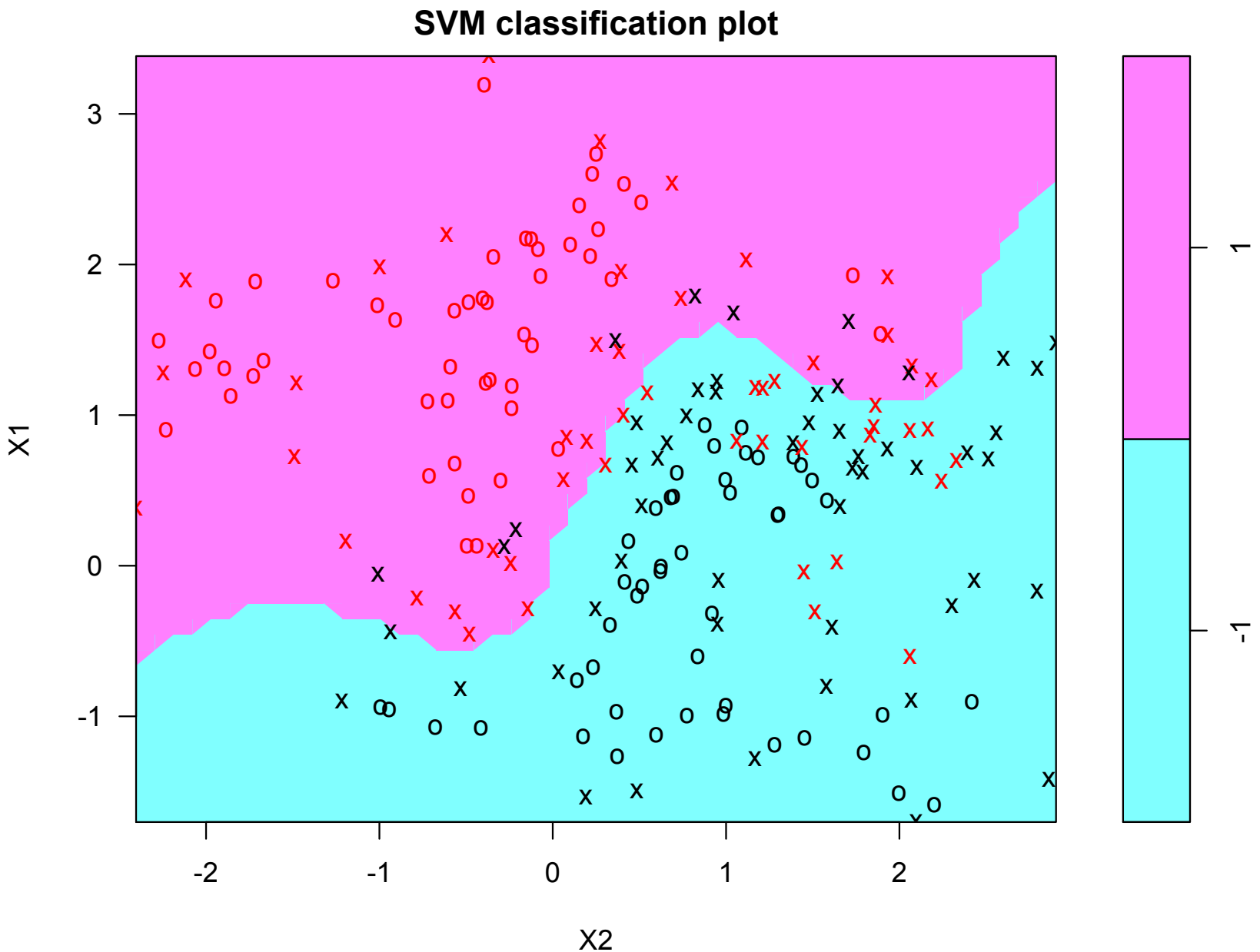
```
set.seed(123)
GM.tune=tune(svm,y~.,type="C-class",kernel="radial",
   ranges = list(gamma = 1:30/10, cost = 1:30/10),data=GM[ind.tr,])
summary(GM.tune)
```

```
> par(mar=c(4.3,4,.5,0))
> plot(GM.tune,main="",xlab=expression(alpha),ylab="C")
> GM.svm=svm(y~.,cost=.9,gamma=2.1,type="C-class",kernel="radial"
             ,data=GM[ind.tr,])
> pred.svm=predict(GM.svm,newdata=GM[ind.tr,])
> sum(pred.svm!=GM[ind.tr,1])/(2*n.train)
[1] 0.135
> pred.svm=predict(GM.svm,newdata=GM[ind.te,])
> sum(pred.svm!=GM[ind.te,1])/(2*n.test)
[1] 0.1658
> plot(GM.svm, GM[ind.tr,])
```

Choosing tuning parameter



Radial basis kernel



SVM for multiple classes

- 1-1 classification. Compare each pair of the K classes. Assign the test observation to the most frequent label. i.e. Each pair of classes compete to get the observation, the class with most wins gets it!
  - The `R` package `e1071` adopts this approach.
- 1-All classification. Each class compete with all other classes together. The class which gives the largest distance to the boundary will win!

Example: Lab

- Read and run §9.6 of ISL.