Introduction aux réseaux de neurones

Fabrice Rossi

Projet AxIS, INRIA Rocquencourt

2007

Plan

1. Introduction

2. Méthodes linéaires
   - Perceptron
   - Machine à vecteurs de support
   - Analyse discriminante de Fisher
   - Régression linéaire

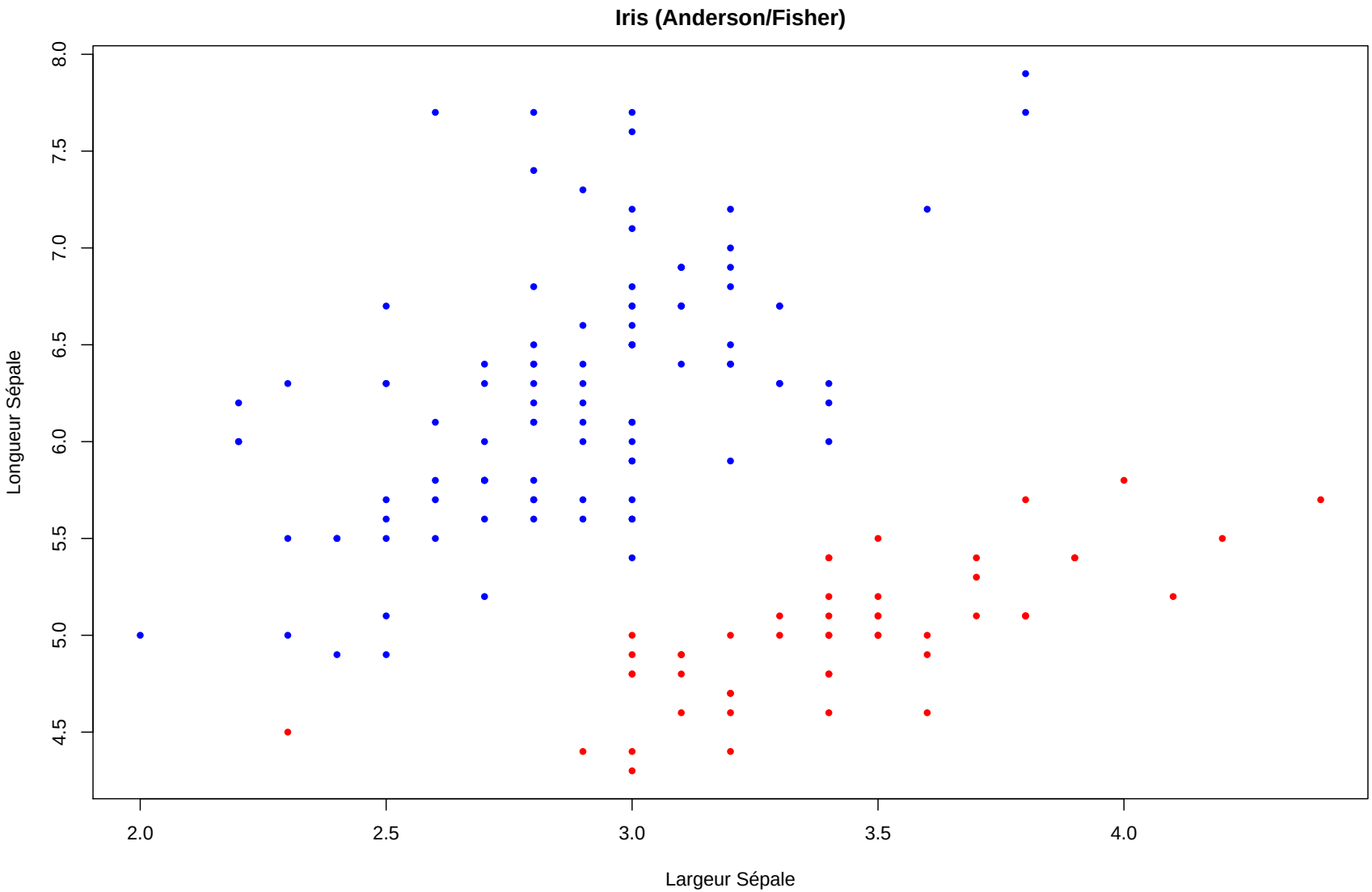
Tentative de définition

- Ensemble de méthodes d'analyse et de traitement des données vaguement issues de considérations biologiques, aux frontières très floues
- Éléments caractéristiques :
  - une combinaison (réseau) d'éléments simples (neurones)
  - un modèle de calcul par propagation de résultats partiels
  - les neurones peuvent être « non linéaires »
  - basé sur des mécanismes d'apprentissage (presque toujours)
  - essentiellement numériques, par opposition à l'IA symbolique (« systèmes experts »)
- Cadre général : apprentissage automatique

Applications

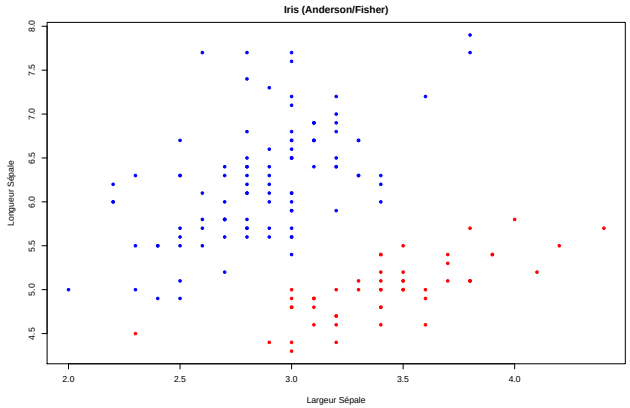
- Analyse exploratoire de données :
  - **classification** : recherche de groupes homogènes (*clustering*)
  - **visualisation**
- Modélisation :
  - **discrimination** : affectation de nouvelles observations à des groupes connus (*classification*)
  - **régression** : prédiction de valeurs numériques à partir de nouvelles observations

Exemple de discrimination



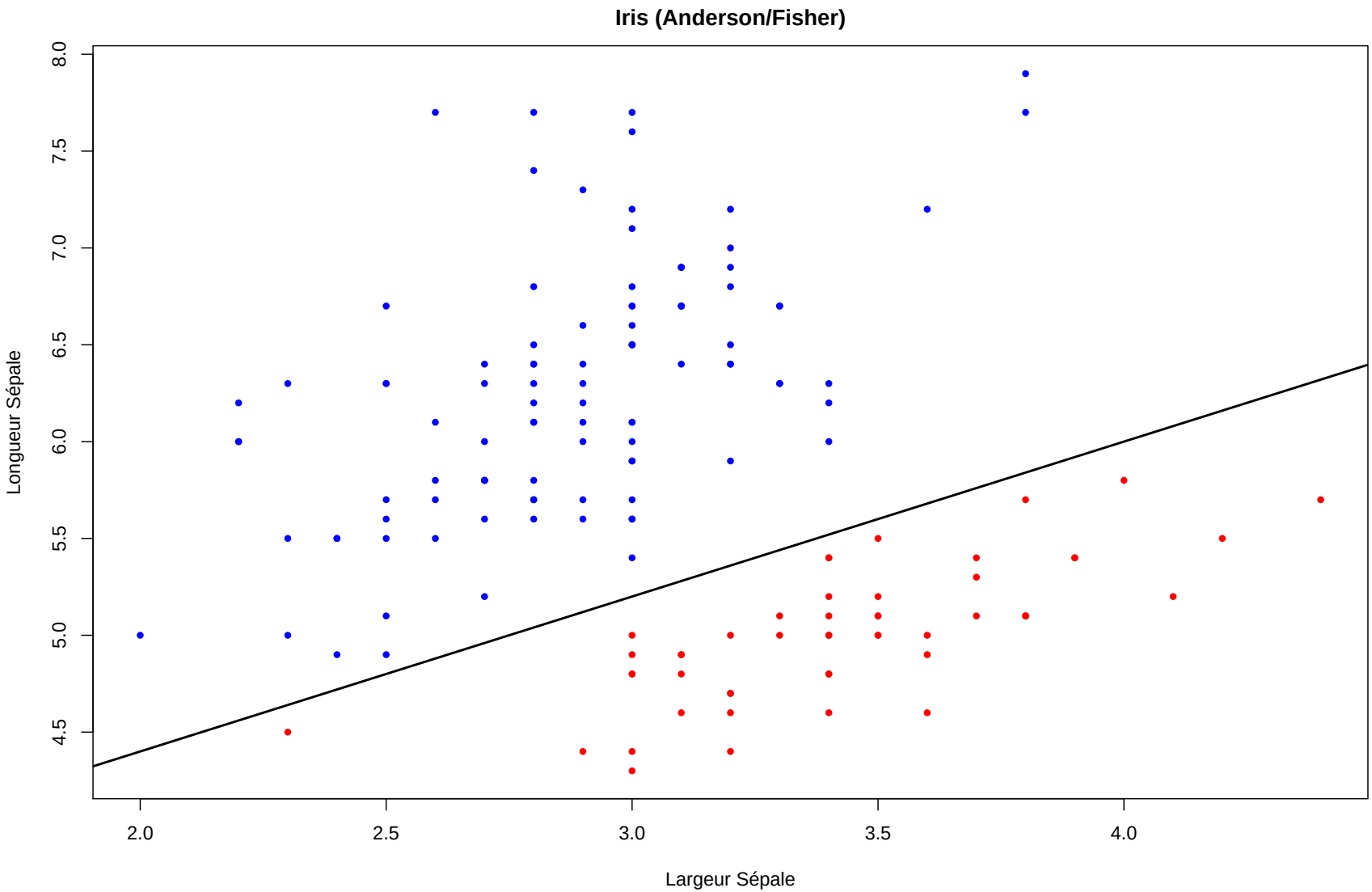
Exemple de discrimination

- But :
  - séparer les points rouges des points bleus (couleur : information de classe)
  - règle de décision basée sur les variables (ici positions)
  - applicable à des nouveaux points (sans couleur)



- Mathématiquement :
  - fonction de $\mathbb{R}^2$ dans $\{\text{bleu}, \text{ rouge}\}$
  - règle de décision
  - choisie de sorte à minimiser les mauvais classements (rouge pour bleu et vice versa)

Exemple de discrimination

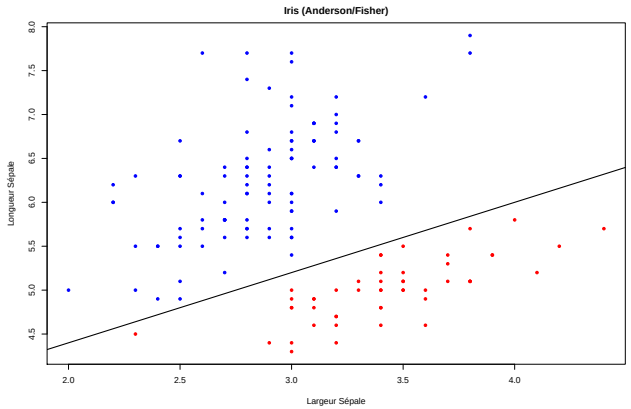


Exemple de discrimination

- Discrimination linéaire :
  - règle de la forme

$$(x_1, x_2) \mapsto \text{signe}(w_1 x_1 + w_2 x_2 + b)$$

  - classes $\{-1, 1\}$



- Apprentissage :
  - plus généralement

$$\mathbf{x} \mapsto \text{signe}\left(\phi(\mathbf{w}, \mathbf{x})\right)$$

  - apprentissage : choisir $\mathbf{w}$ à partir d'exemples déjà classés
  - vocabulaire statistique : estimation des paramètres d'un modèle

Plan

1 Introduction

2 Méthodes linéaires
- Perceptron
- Machine à vecteurs de support
- Analyse discriminante de Fisher
- Régression linéaire

Le neurone de McCulloch et Pitts (1943)

Caractéristiques :

- p entrées : $\mathbf{x} = (x_1, \ldots, x_p)$ (binaires dans le modèle d'origine)
- une sortie "binaire" obtenue par :

$$T\left(\sum_{j=1}^{p} w_j x_j + b\right)$$

avec pour T la fonction de Heaviside

$$T(u) = \begin{cases} 0 & u < 0 \\ 1 & u \geq 0. \end{cases}$$

T est la fonction d'**activation** (ou de transfert) du neurone

Le neurone de McCulloch et Pitts (1943)

Caractéristiques :

- p entrées : $\mathbf{x} = (x_1, \ldots, x_p)$ (binaires dans le modèle d'origine)
- une sortie "binaire" obtenue par :

$$T\left(\sum_{j=1}^{p} w_j x_j + b\right)$$

- p poids synaptiques $\mathbf{w} = (w_1, \ldots, w_p)$
- un seuil $-b$: le neurone est actif si le seuil est atteint
- correspond à un modèle de discrimination linéaire

Neurone MCP

Interprétation biologique :

- sortie : axone
- entrées : dendrites
- poids synaptiques w_j :
  - synapses
  - positif : excitatrice
  - négatif : inhibitrice
  - pondération $w_j x_j$: potentiel post-synaptique
- valeurs des entrées et de la sortie : potentiel d'action
- seuil $-b$: seuil d'excitabilité

Modèle relativement pertinent, excepté l'aspect temporel

Discrimination par un neurone de MCP

- Données :
  - N observations
  - p variables : observations dans $\mathbb{R}^p$
  - classe connue pour chaque observation : $\mathbf{x}_i = (x_{ij}, \ldots, x_{ip})$ et $y_i \in \{-1, 1\}$ (pour deux classes)
- Discrimination neuronale :
  - un neurone de MCP à p entrées (sortie dans $\{-1, 1\}$)
  - apprentissage : trouver des valeurs des poids synaptiques telles que

$$T\left(\sum_{j=1}^{p} w_j x_{ij} + b\right) = y_i,$$

pour beaucoup de i dans $\{1, \ldots, N\}$ (tous, si possible)

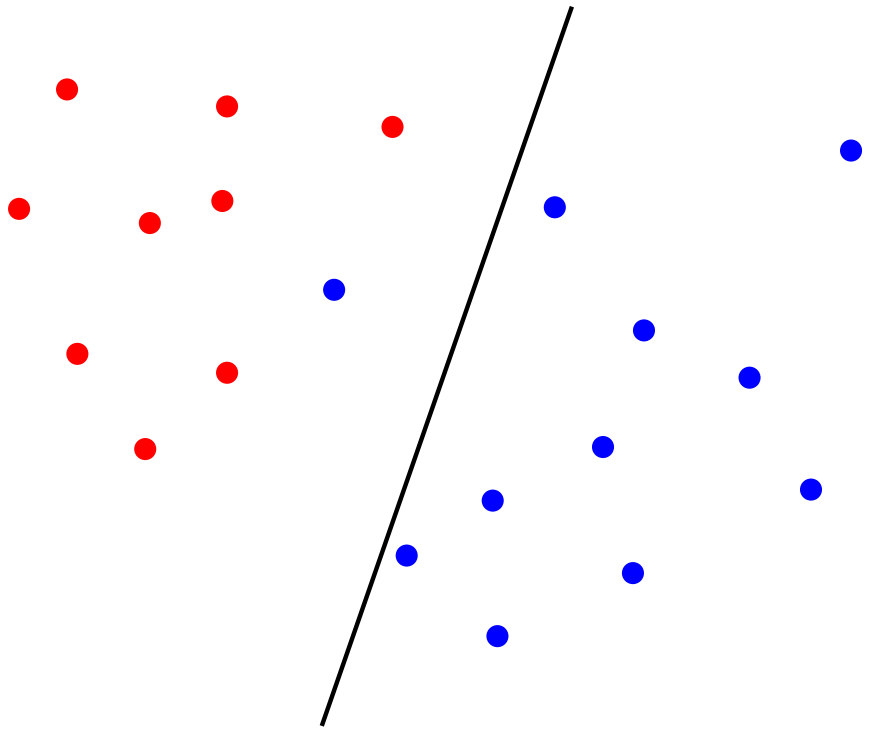
Le perceptron de Rosenblatt

Frank Rosenblatt utilise le neurone de McCulloch et Pitts dans les années 50 :

- sortie dans $\{-1, 1\}$
- première implémentation matérielle
- discrimination d'images (noir et blanc) en deux classes *a priori*
- premier algorithme d'apprentissage qui cherche à minimiser le nombre d'erreurs de classement
- algorithme *on line* : correction des poids synaptiques après l'étude de chaque observation
- converge si les données sont linéairement séparables

Principe de l'algorithme du perceptron

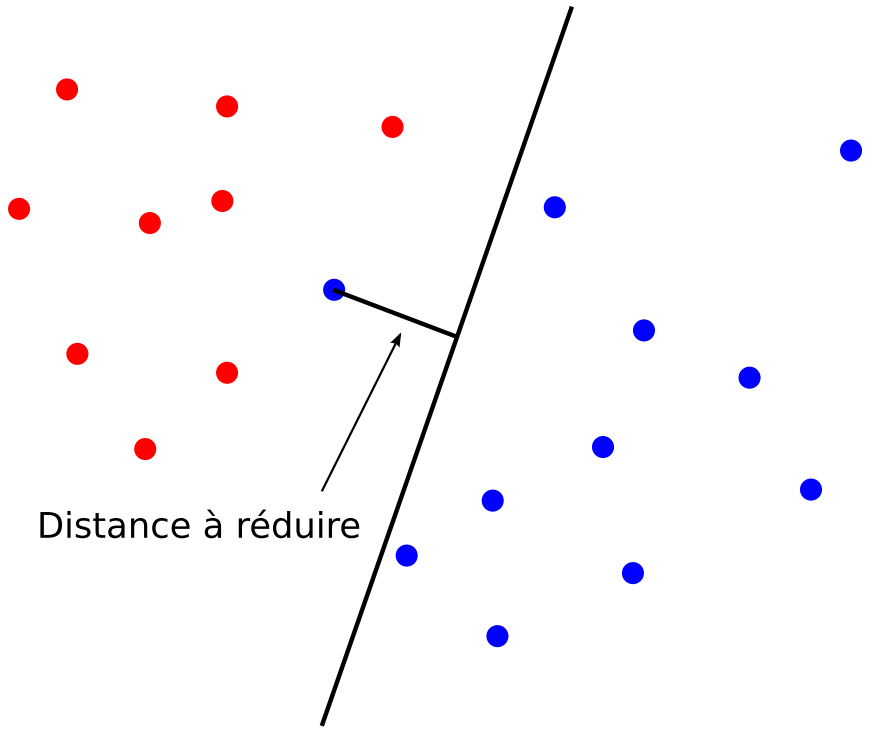
En cas d'erreur pour $(\mathbf{x}, y)$:



Principe de l'algorithme du perceptron

En cas d'erreur pour $(\mathbf{x}, y)$:

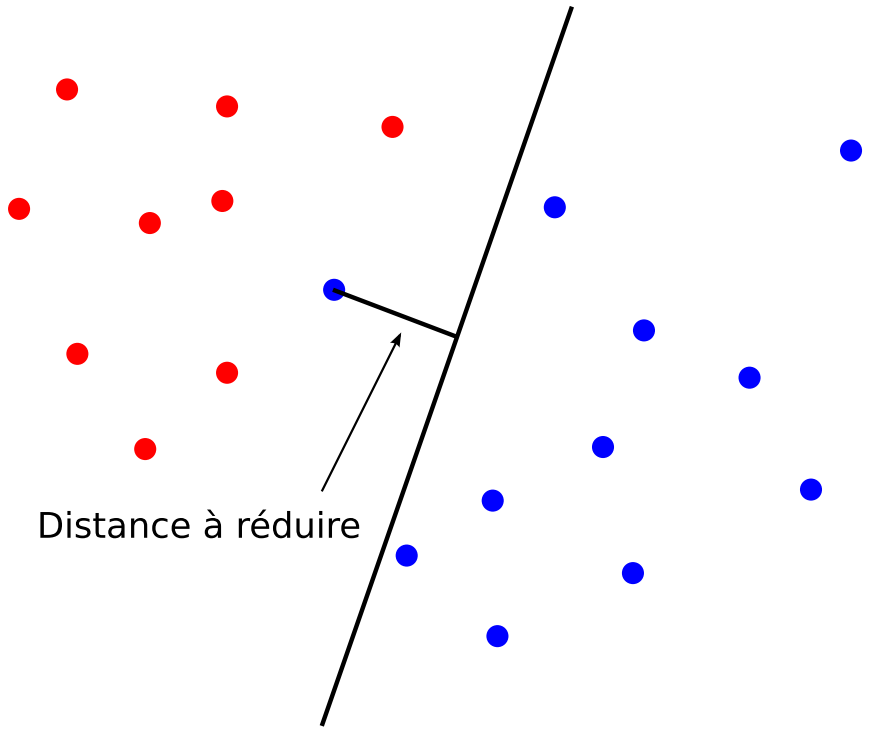
- $\frac{|w_1 x_1 + w_2 x_2 + b|}{\sqrt{w_1^2 + w_2^2}}$



Principe de l'algorithme du perceptron

En cas d'erreur pour $(\mathbf{x}, y)$:

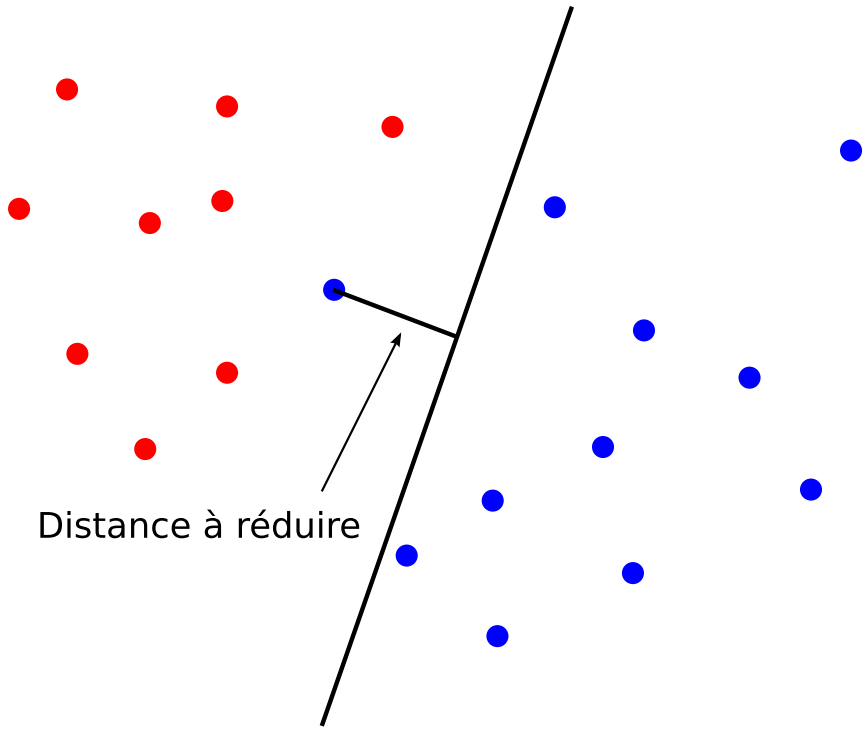
- $\frac{|w_1 x_1 + w_2 x_2 + b|}{\sqrt{w_1^2 + w_2^2}}$
- rapprocher $w_1 x_1 + w_2 x_2 + b$ de zéro :
  - si $w_1 x_1 + w_2 x_2 + b > 0$ alors $y < 0$
  - si $w_1 x_1 + w_2 x_2 + b < 0$ alors $y > 0$



Principe de l'algorithme du perceptron

En cas d'erreur pour $(\mathbf{x}, y)$:

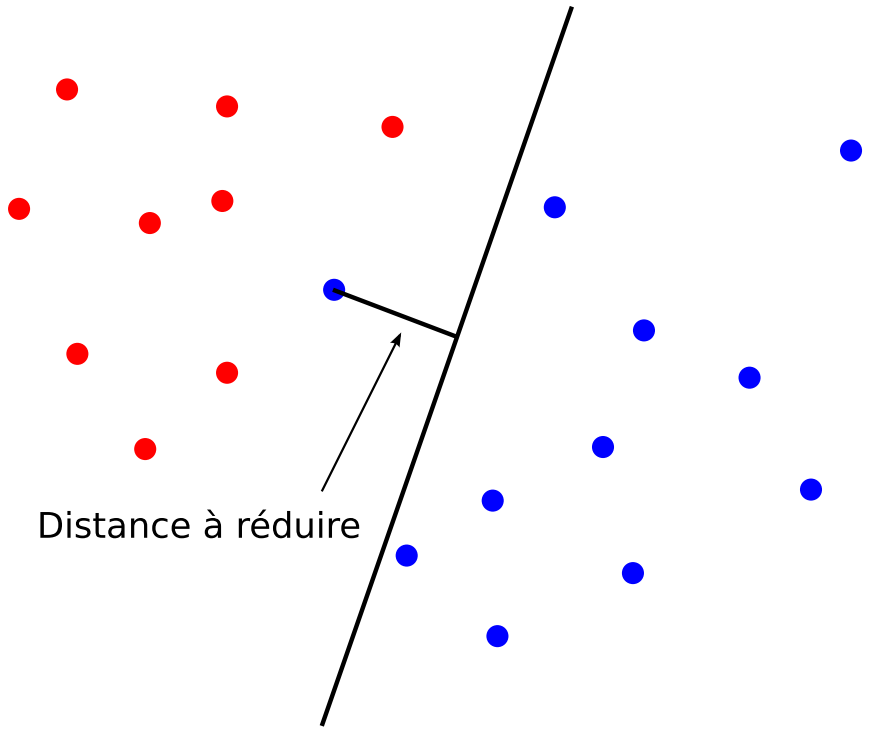
- $\frac{|w_1 x_1 + w_2 x_2 + b|}{\sqrt{w_1^2 + w_2^2}}$
- rapprocher $w_1 x_1 + w_2 x_2 + b$ de zéro :
  - si $w_1 x_1 + w_2 x_2 + b > 0$ alors $y < 0$
  - si $w_1 x_1 + w_2 x_2 + b < 0$ alors $y > 0$
- minimiser $|w_1 x_1 + w_2 x_2 + b|$



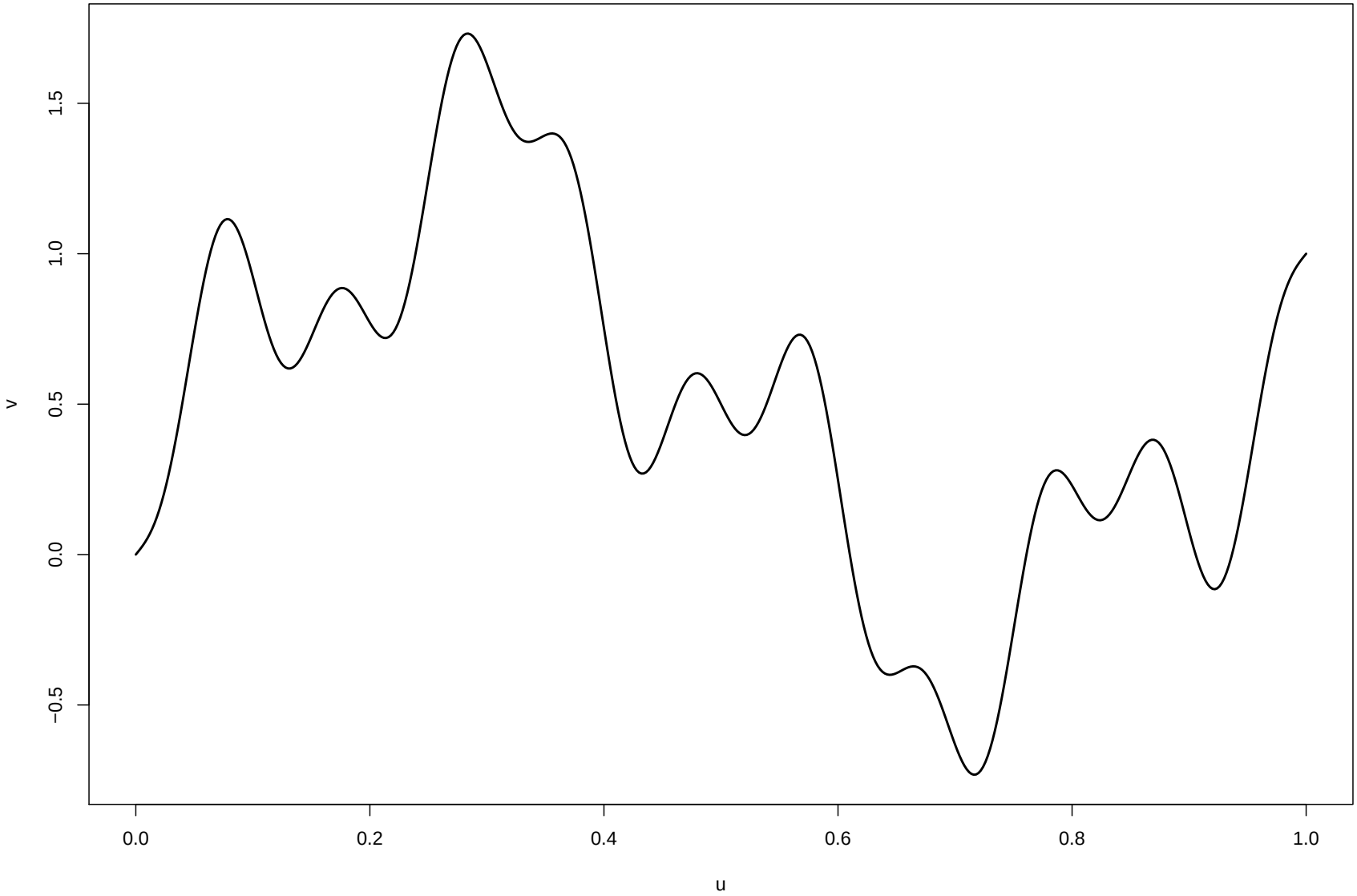
Principe de l'algorithme du perceptron

En cas d'erreur pour $(\mathbf{x}, y)$:

- $\frac{|w_1 x_1 + w_2 x_2 + b|}{\sqrt{w_1^2 + w_2^2}}$
- rapprocher $w_1 x_1 + w_2 x_2 + b$ de zéro :
  - si $w_1 x_1 + w_2 x_2 + b > 0$ alors $y < 0$
  - si $w_1 x_1 + w_2 x_2 + b < 0$ alors $y > 0$
- minimiser $|w_1 x_1 + w_2 x_2 + b|$
- Descente de gradient



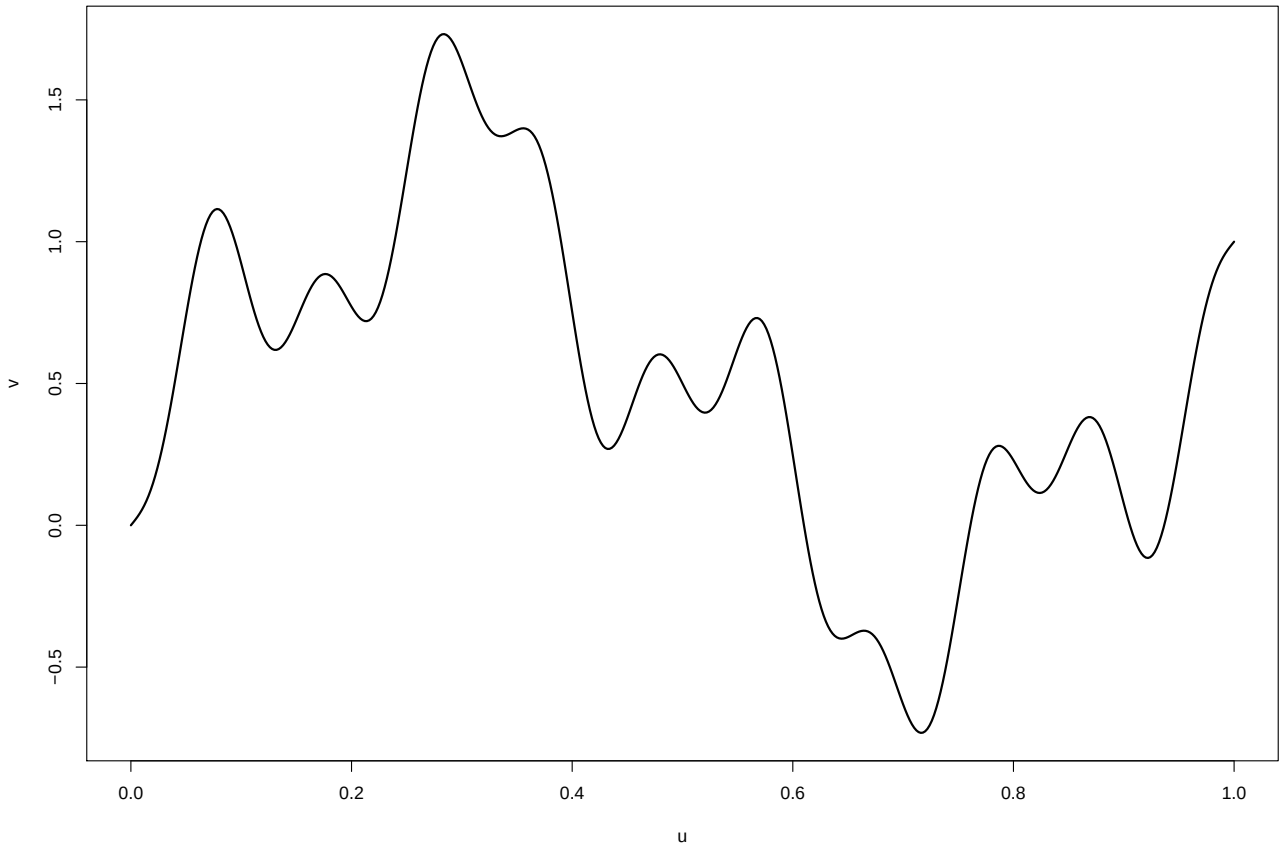
Descente de gradient



Descente de gradient

Minimisation de f :

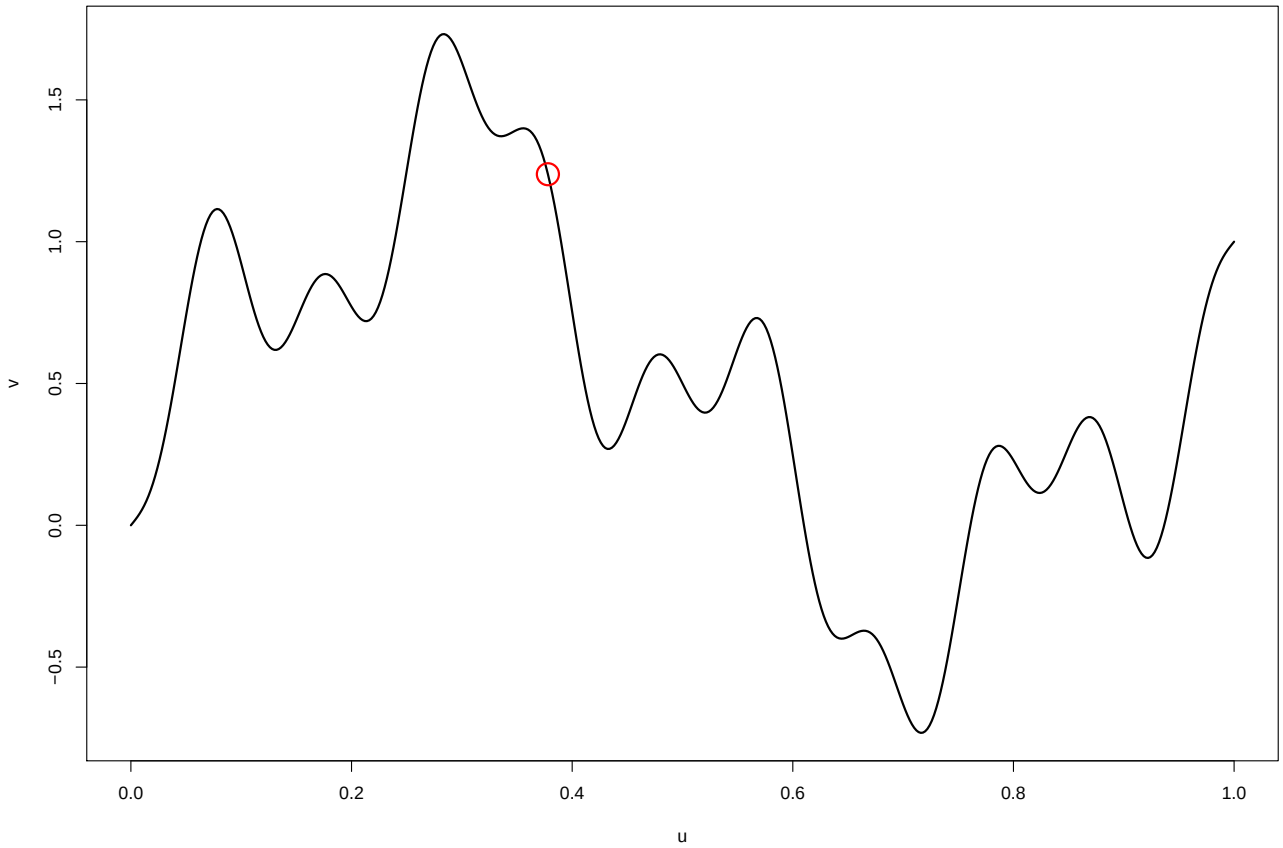
- Méthode itérative



Descente de gradient

Minimisation de f :

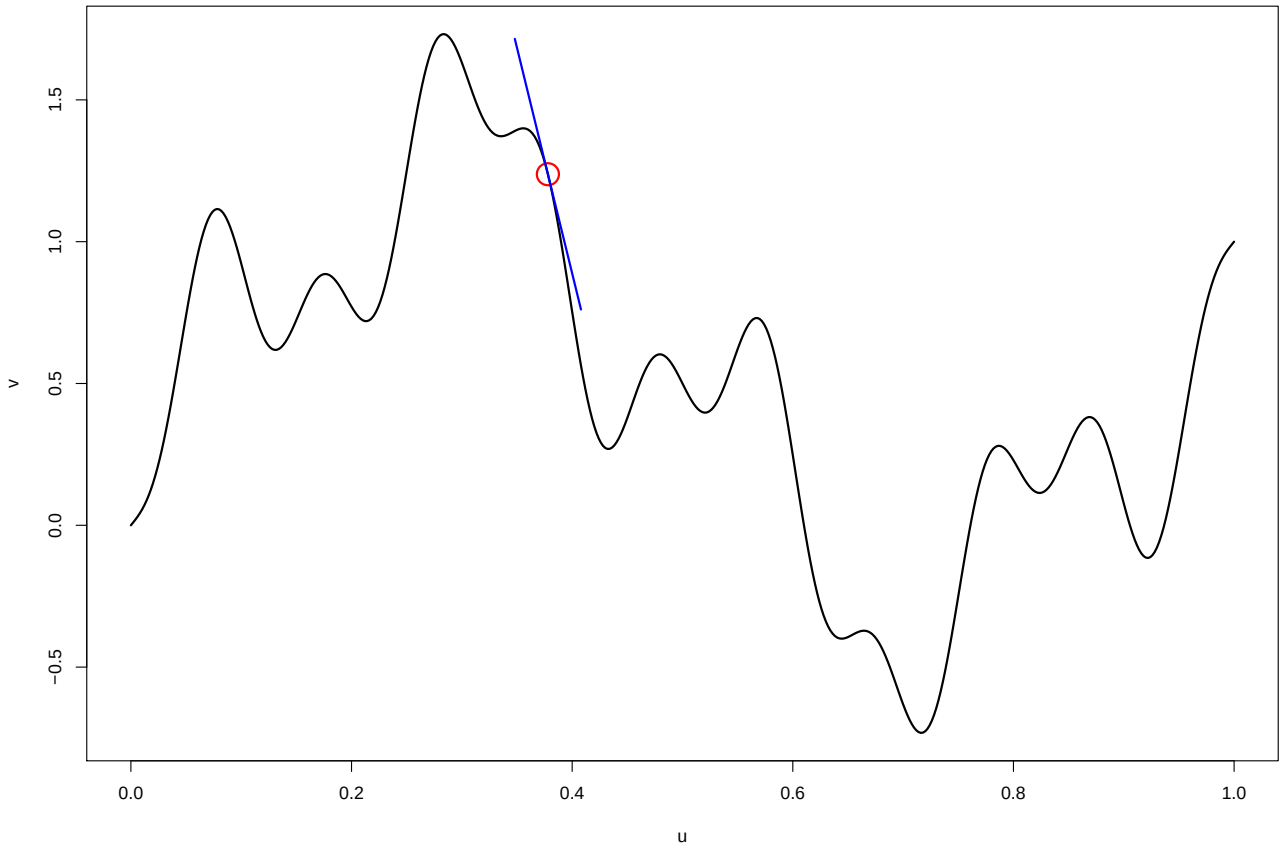
- Méthode itérative
- Point initial u_0



Descente de gradient

Minimisation de f :

- Méthode itérative
- Point initial u_0
- Dérivée : direction pour réduire

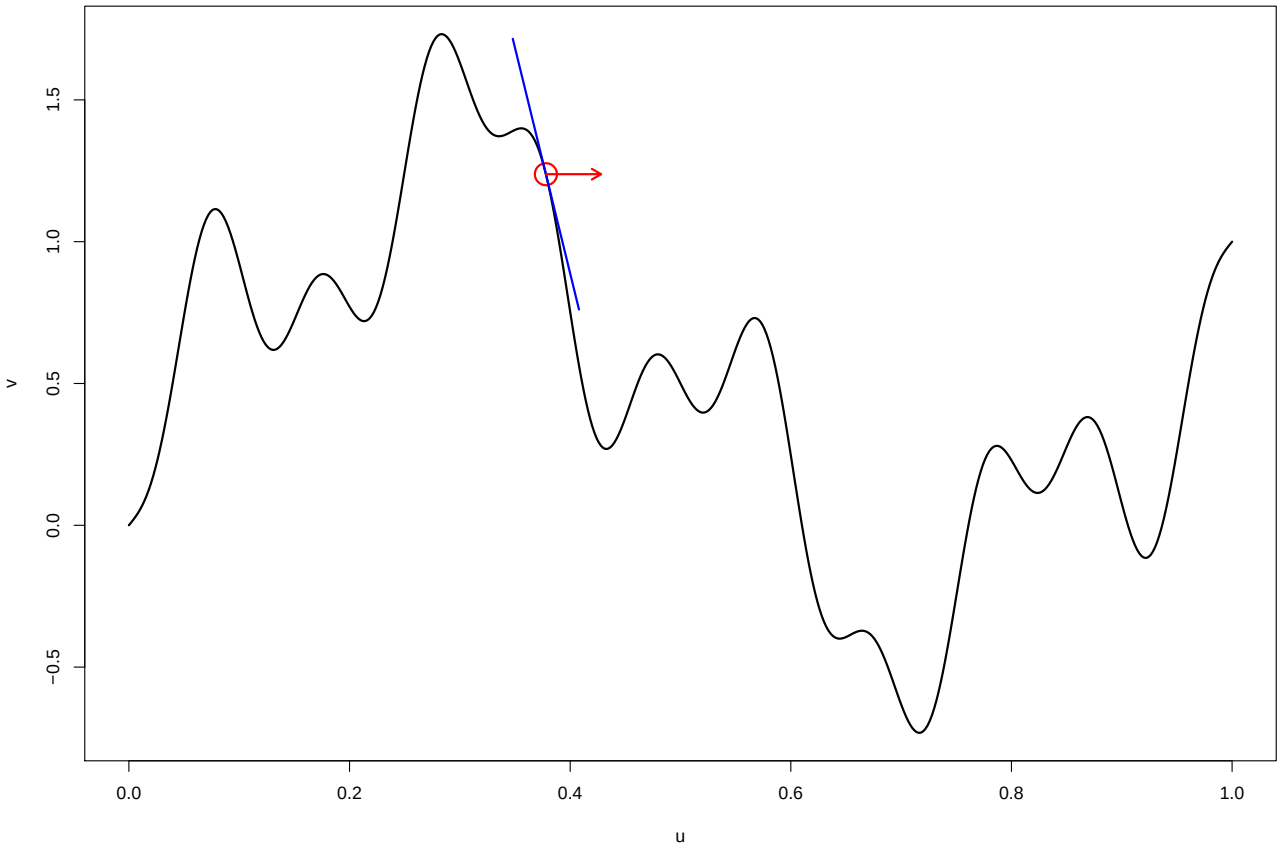


Descente de gradient

Minimisation de f :

- Méthode itérative
- Point initial u_0
- Dérivée : direction pour réduire
- Mouvement

$$u_{i+1} = u_i - \epsilon f'(u_i)$$



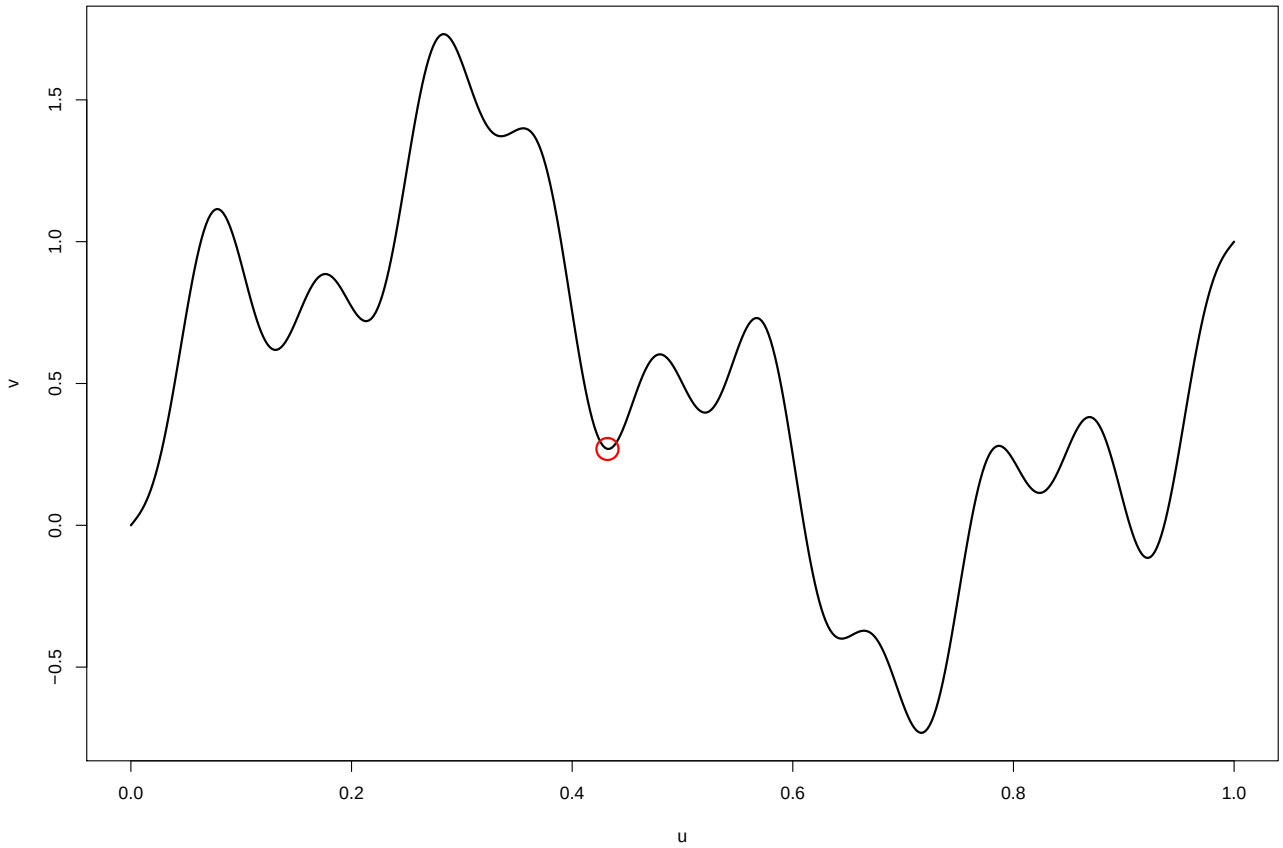
Descente de gradient

Minimisation de f :

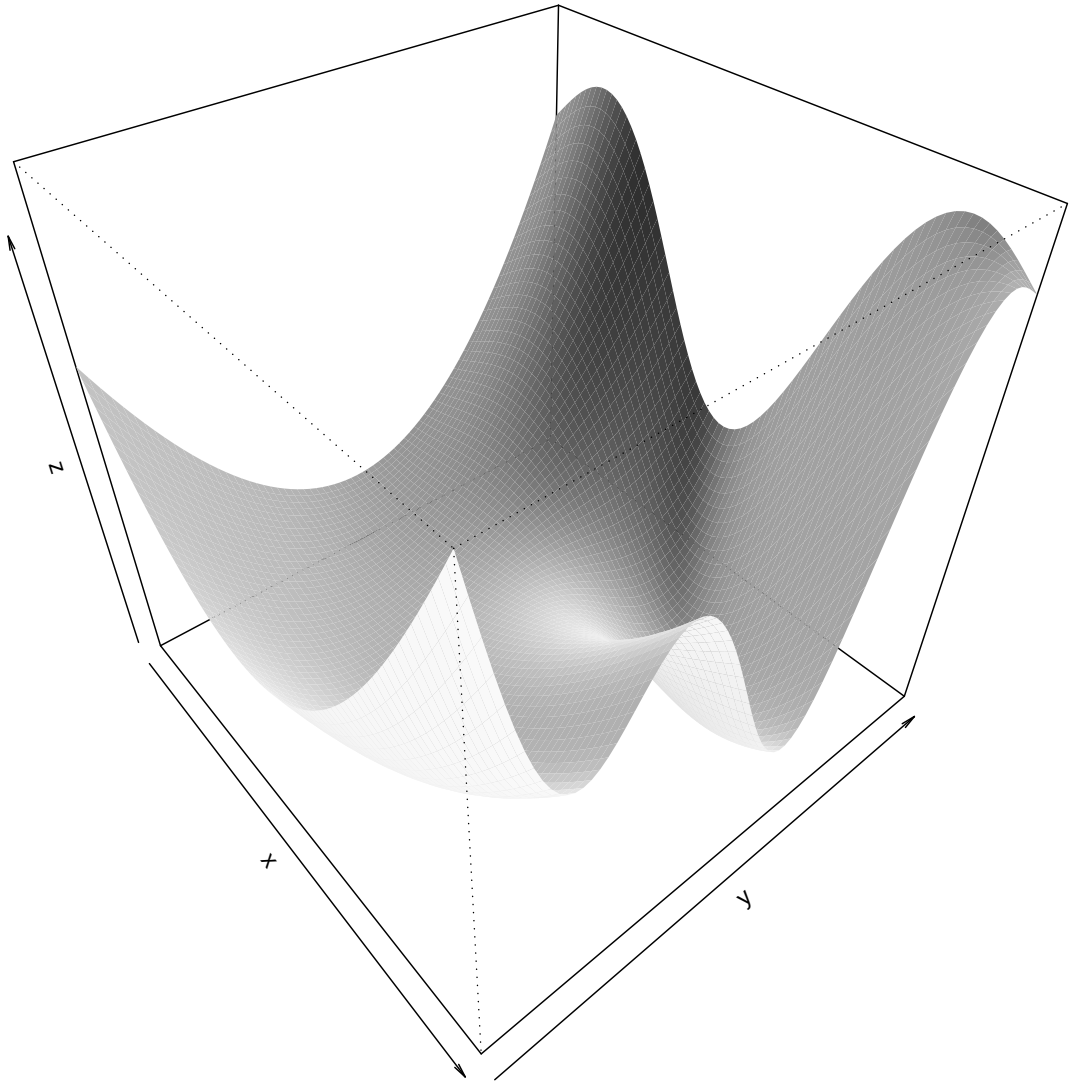
- Méthode itérative
- Point initial u_0
- Dérivée : direction pour réduire
- Mouvement

$$u_{i+1} = u_i - \epsilon f'(u_i)$$

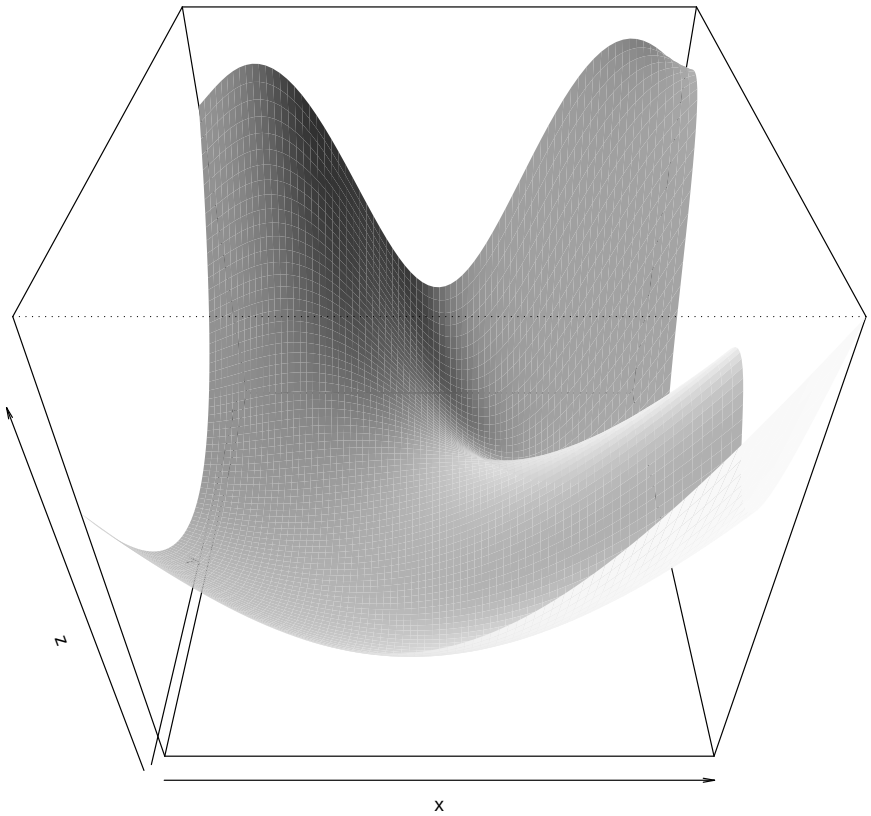
- Converge vers un minimum local

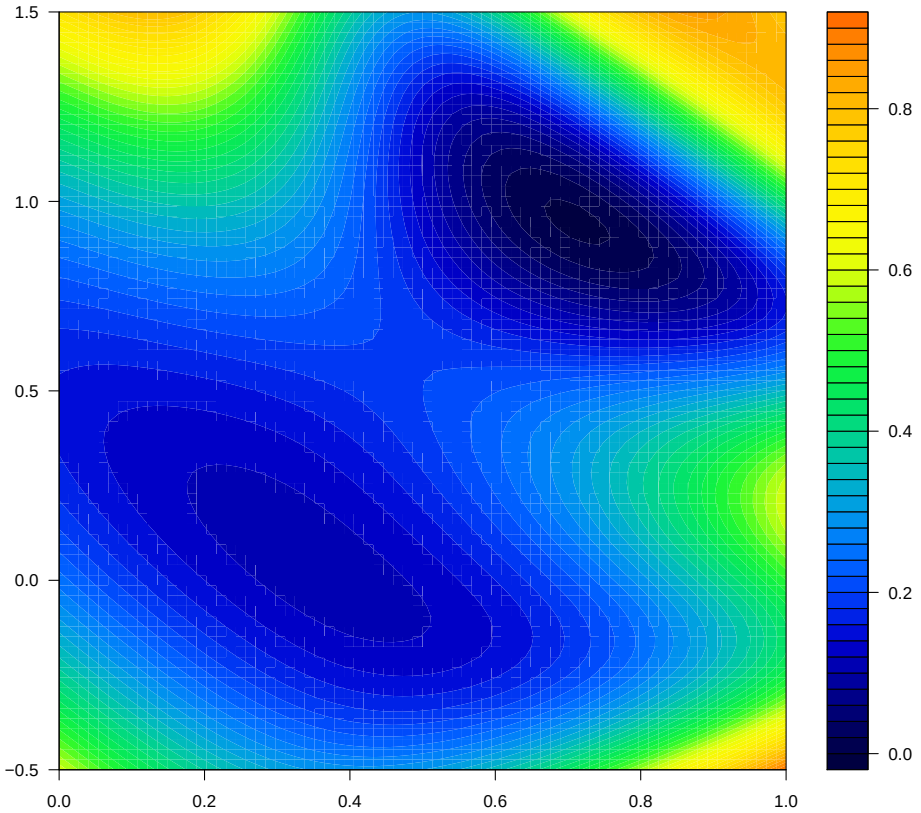


Deux dimensions

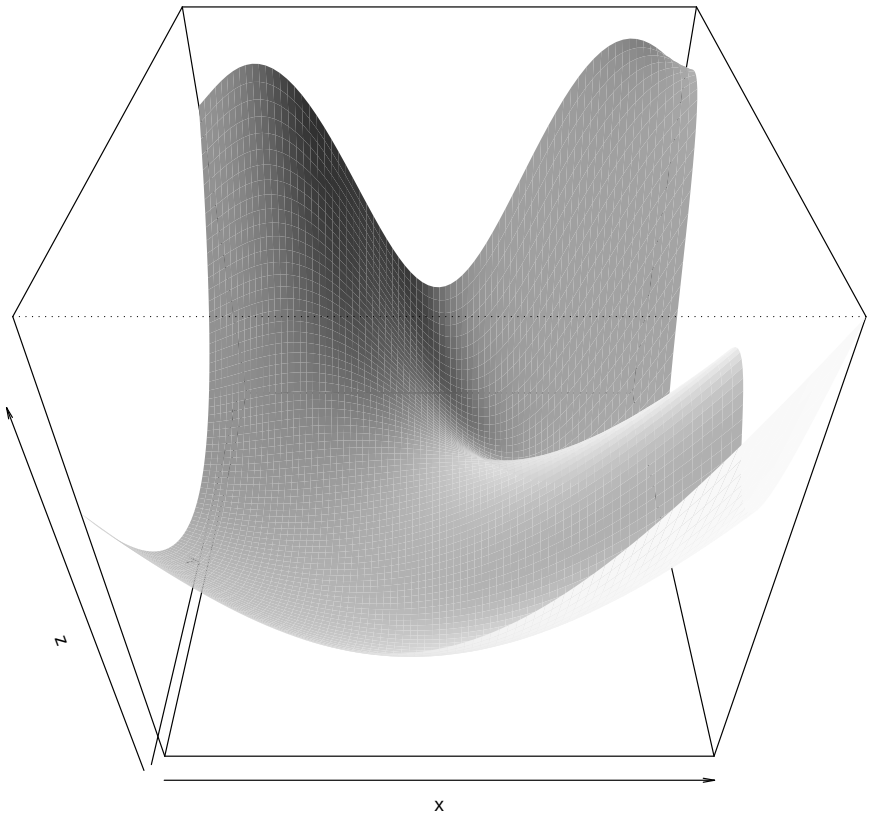


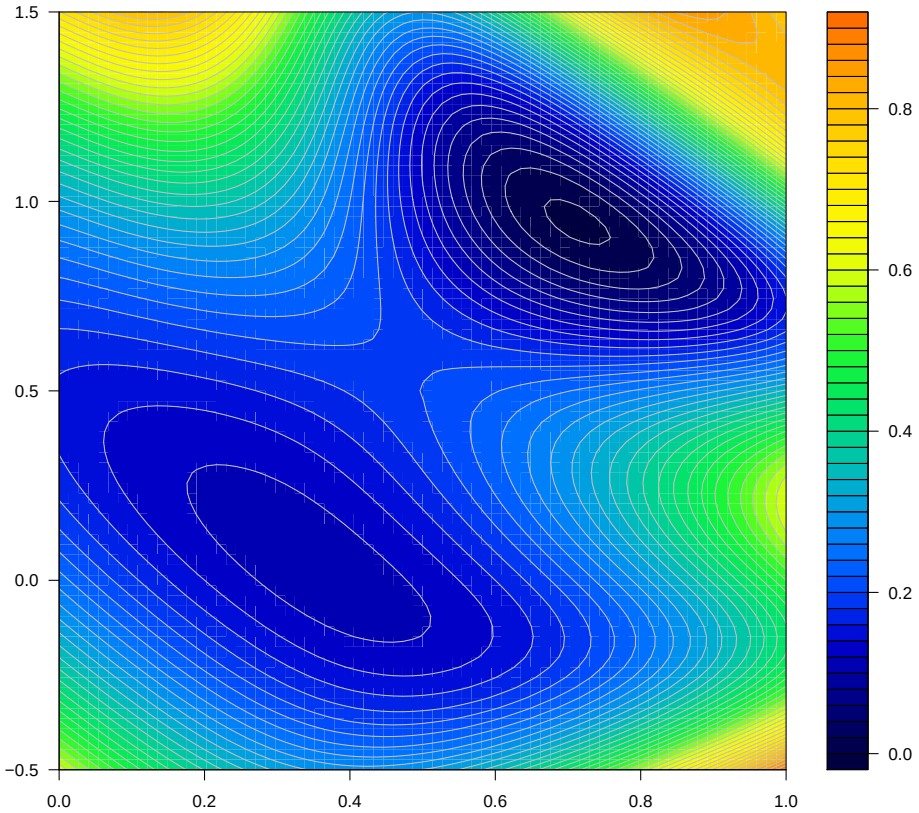
Deux dimensions



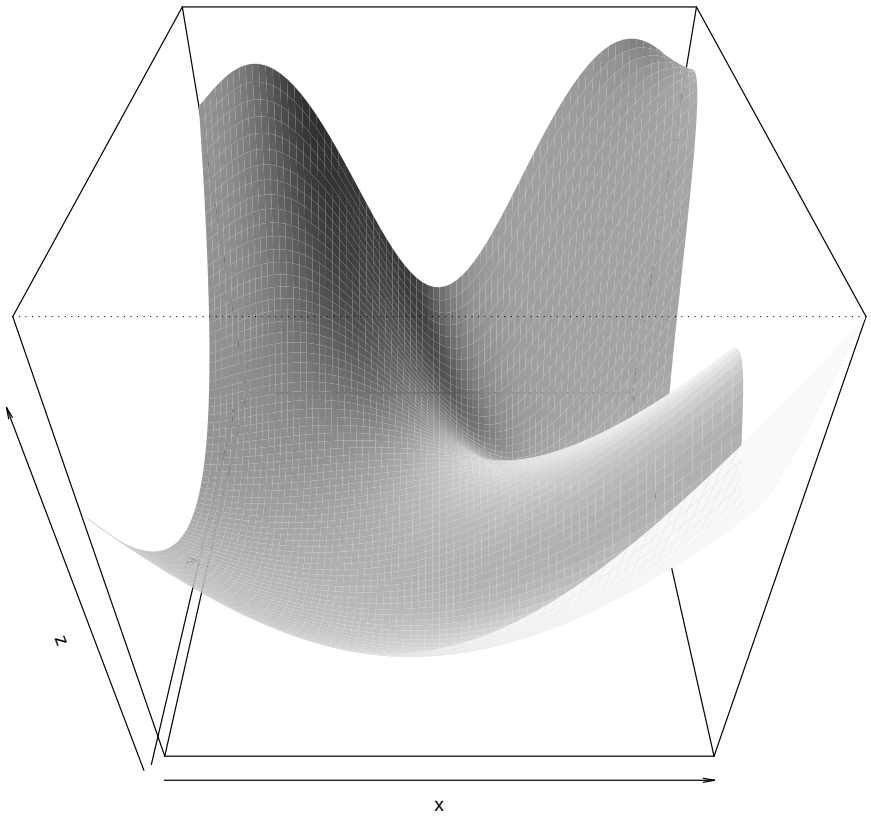


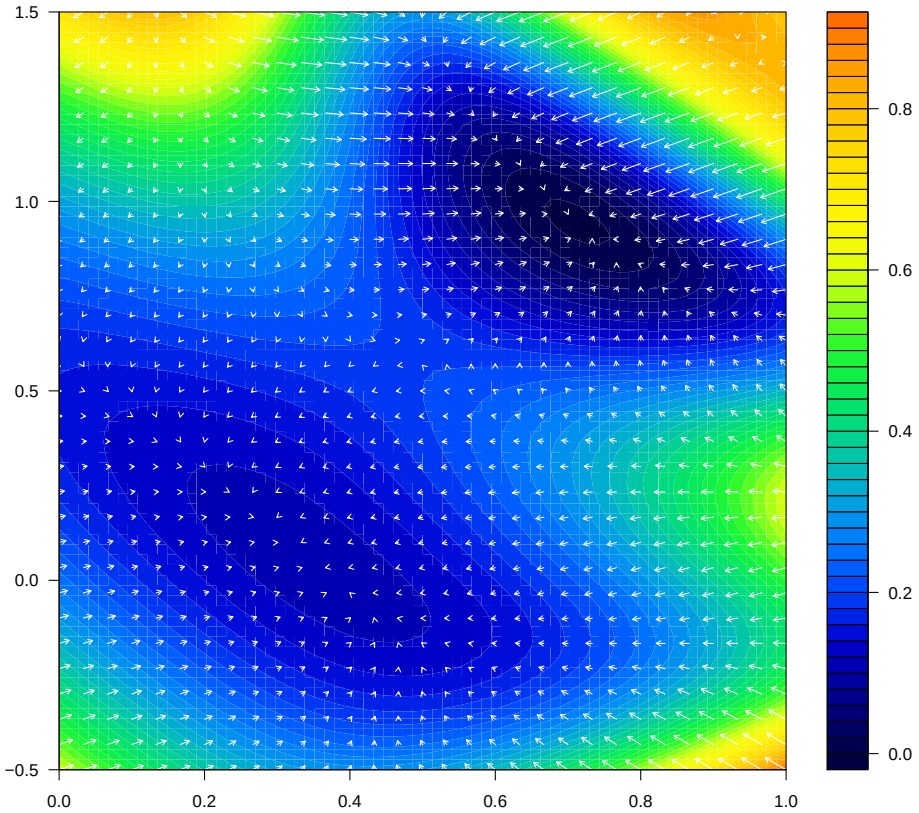
Deux dimensions



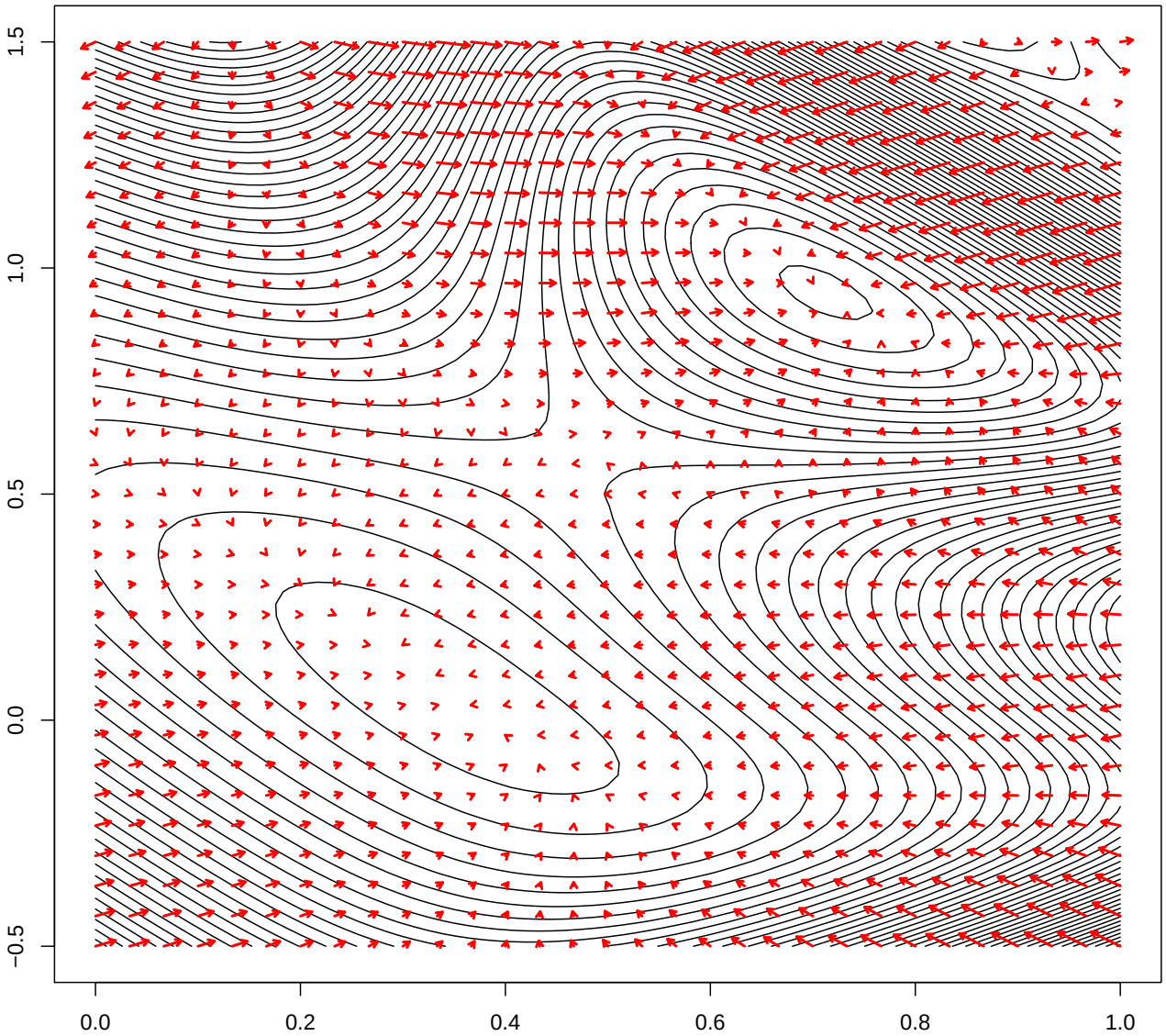


Deux dimensions





Deux dimensions



- une fonction f de $\mathbb{R}^I$ dans $\mathbb{R}$ à minimiser
- direction de plus grande pente : gradient $-\nabla f$
- rappel :

$$\nabla f(\mathbf{w}) = \left(\frac{\partial f}{\partial w_1}(\mathbf{w}), \ldots, \frac{\partial f}{\partial w_I}(\mathbf{w}) \right)^T$$

- algorithme du gradient simple :
  1. point initial aléatoire $\mathbf{w}^{(0)}$
  2. $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \epsilon^{(t)} \nabla f(\mathbf{w}^{(t)})$
  3. retour en 1 si $\left\| \nabla f(\mathbf{w}^{(t+1)}) \right\|$ est « grand »
- converge vers un minimum local (sous de nombreuses hypothèses)

Perceptron et gradient

- si **x** est mal classé :
  - signe $\left(b + \sum_{j=1}^{p} w_j x_j\right) = -y$
  - on cherche à minimiser $f(\mathbf{w}) = \left|b + \sum_{j=1}^{p} w_j x_j\right|$
  - si $y > 0$ alors $f(\mathbf{w}) = -\left(b + \sum_{j=1}^{p} w_j x_j\right)$, sinon $f(\mathbf{w}) = b + \sum_{j=1}^{p} w_j x_j$
  - donc $f(\mathbf{w}) = -y\left(b + \sum_{j=1}^{p} w_j x_j\right)$
- gradient

$$\nabla\left(f(\mathbf{w})\right) = \left(-y, -yx_1, \ldots, -yx_p\right)^T$$

- $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \epsilon^{(t)} \nabla f(\mathbf{w}^{(t)})$ s'écrit :
  - $b^{(t+1)} = b^{(t)} + \epsilon^{(t)} y$
  - $w_j^{(t+1)} = w_j^{(t)} + \epsilon^{(t)} y x_j$

Algorithme du perceptron

Algorithme :

1. poids initiaux aléatoires : $\mathbf{w}^{(0)}$
2. boucle sur les N observations $(\mathbf{x}_i, y_i)$:
   1. si $\mathbf{x}_i$ est bien classé, $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)}$
   2. sinon, mise à jour par :
      - $b^{(t+1)} = b^{(t)} + \epsilon y_i$
      - $w_j^{(t+1)} = w_j^{(t)} + \epsilon y_i x_{ij}$, pour tout j
3. s'il reste des observations mal classées, retour en 2

Remarques :

- $\epsilon \in]0, 1[$
- converge dans le cas linéairement séparable (i.e., il existe des poids pour lesquels le perceptron ne fait aucune erreur)
- comportement cyclique dans les autres cas

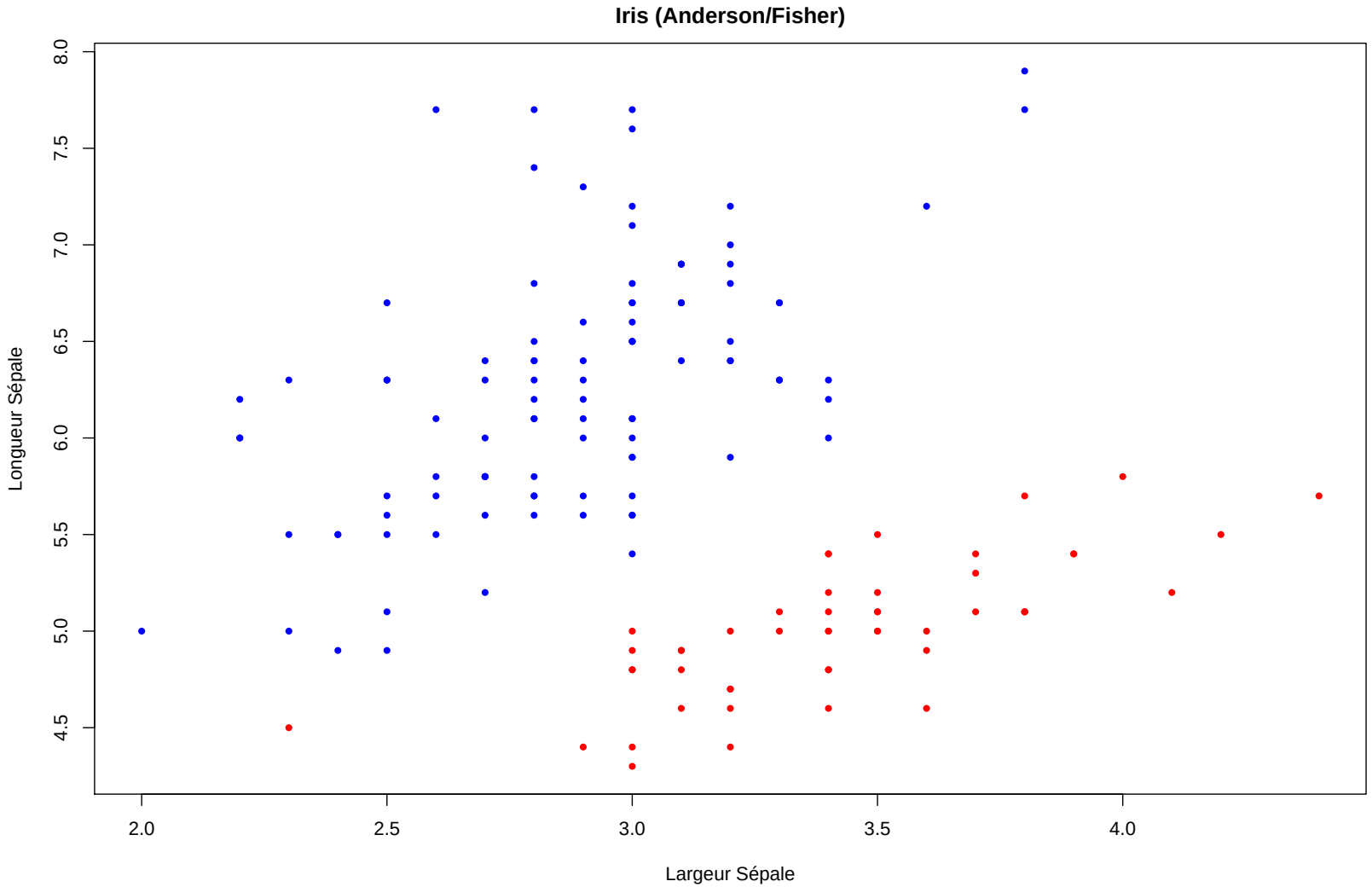
Interprétation biologique

Principe du renforcement :

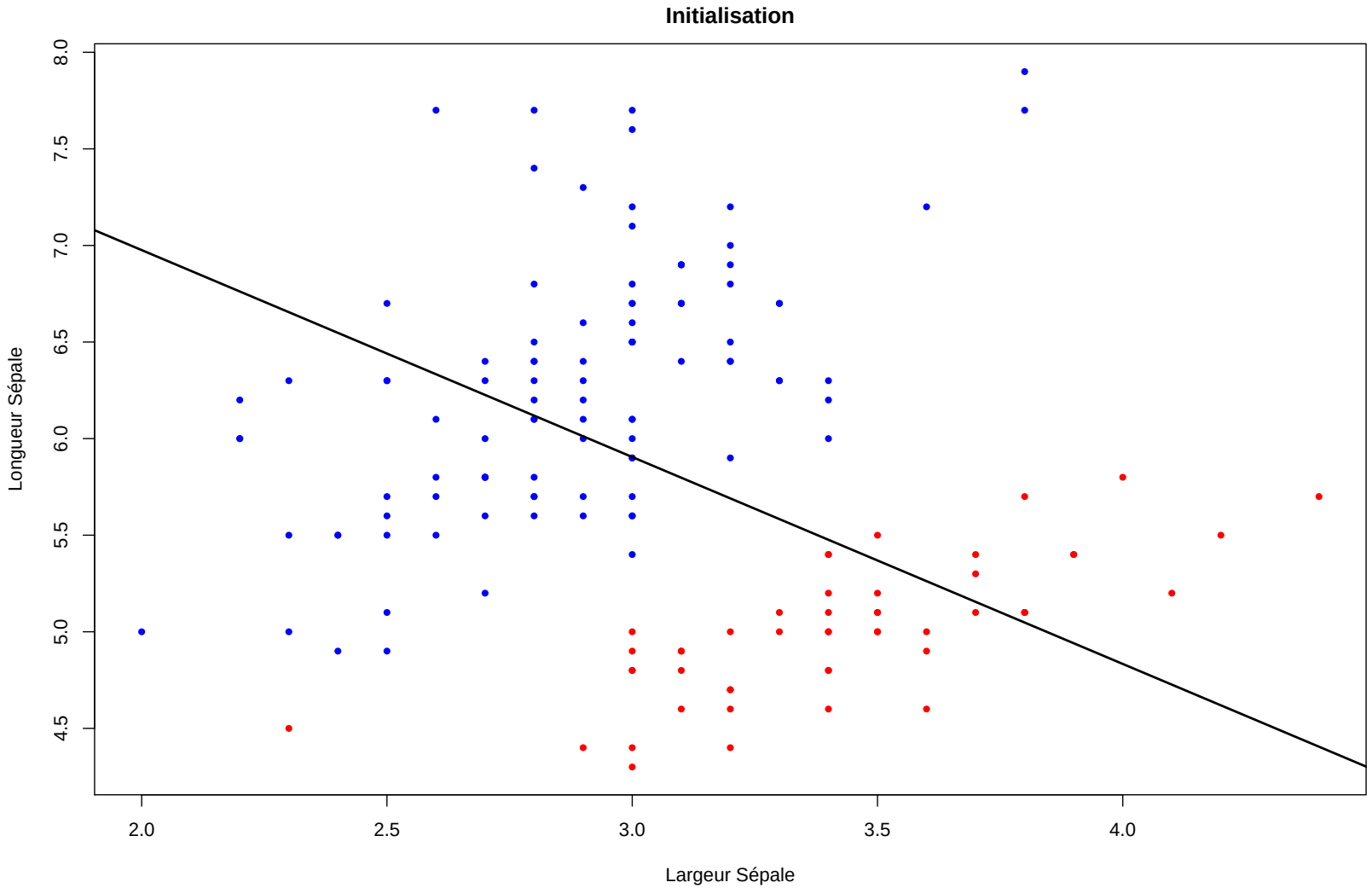
- corrélation d'activité $\Rightarrow$ renforcement des liens
- règle de Hebb (1949) : $\Delta w_j = \lambda x_j y$
- idée sous-jacente :
  - si x_j est **positif** quand y est **positif**, il faut que la synapse w_j soit **excitatrice**
  - si x_j est **positif** quand y est **négatif**, il faut que la synapse w_j soit **inhibitrice**
- l'algorithme du perceptron est « Hebbien »

Exemple de mise en œuvre

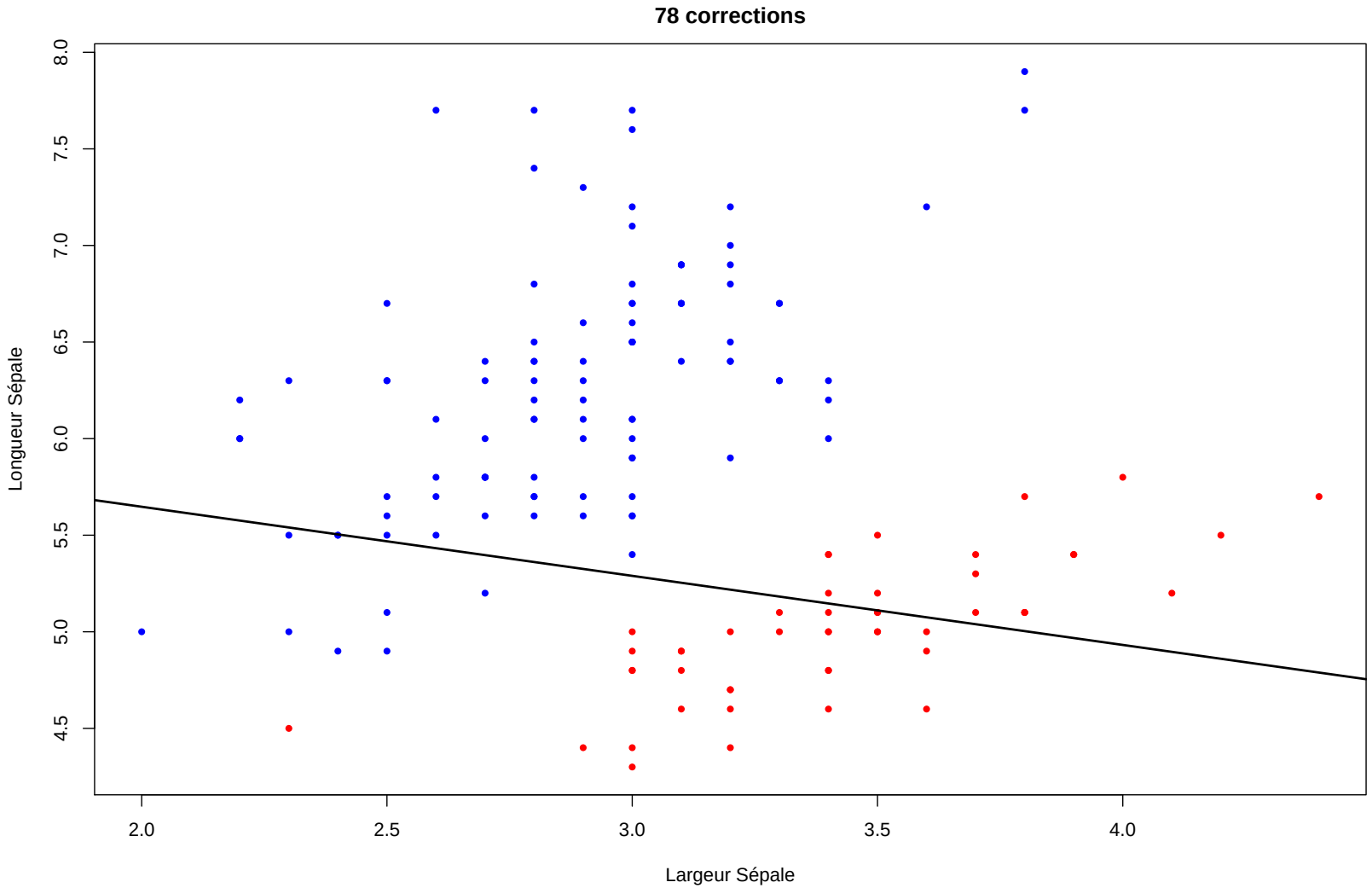
Données



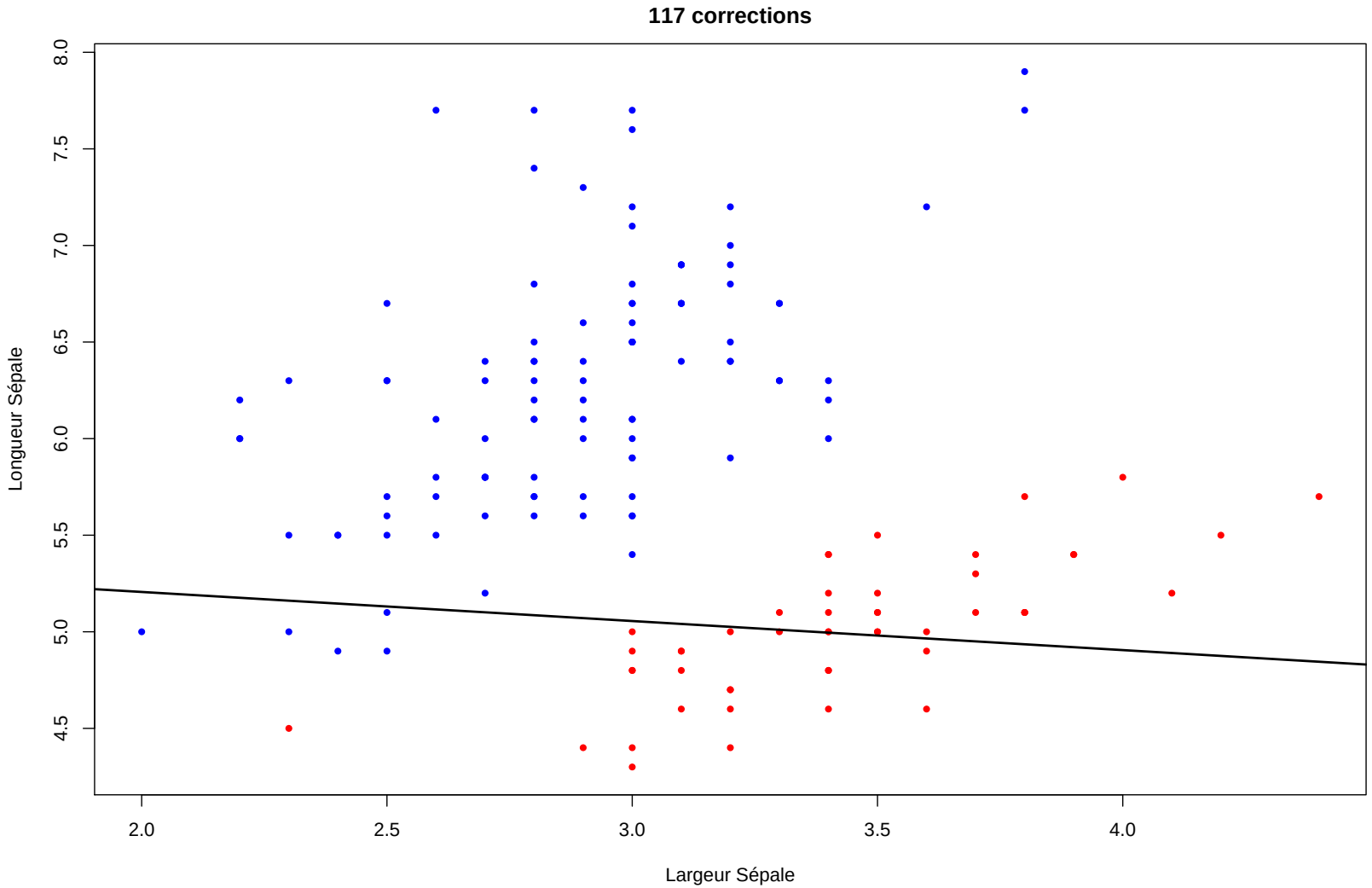
Exemple de mise en œuvre



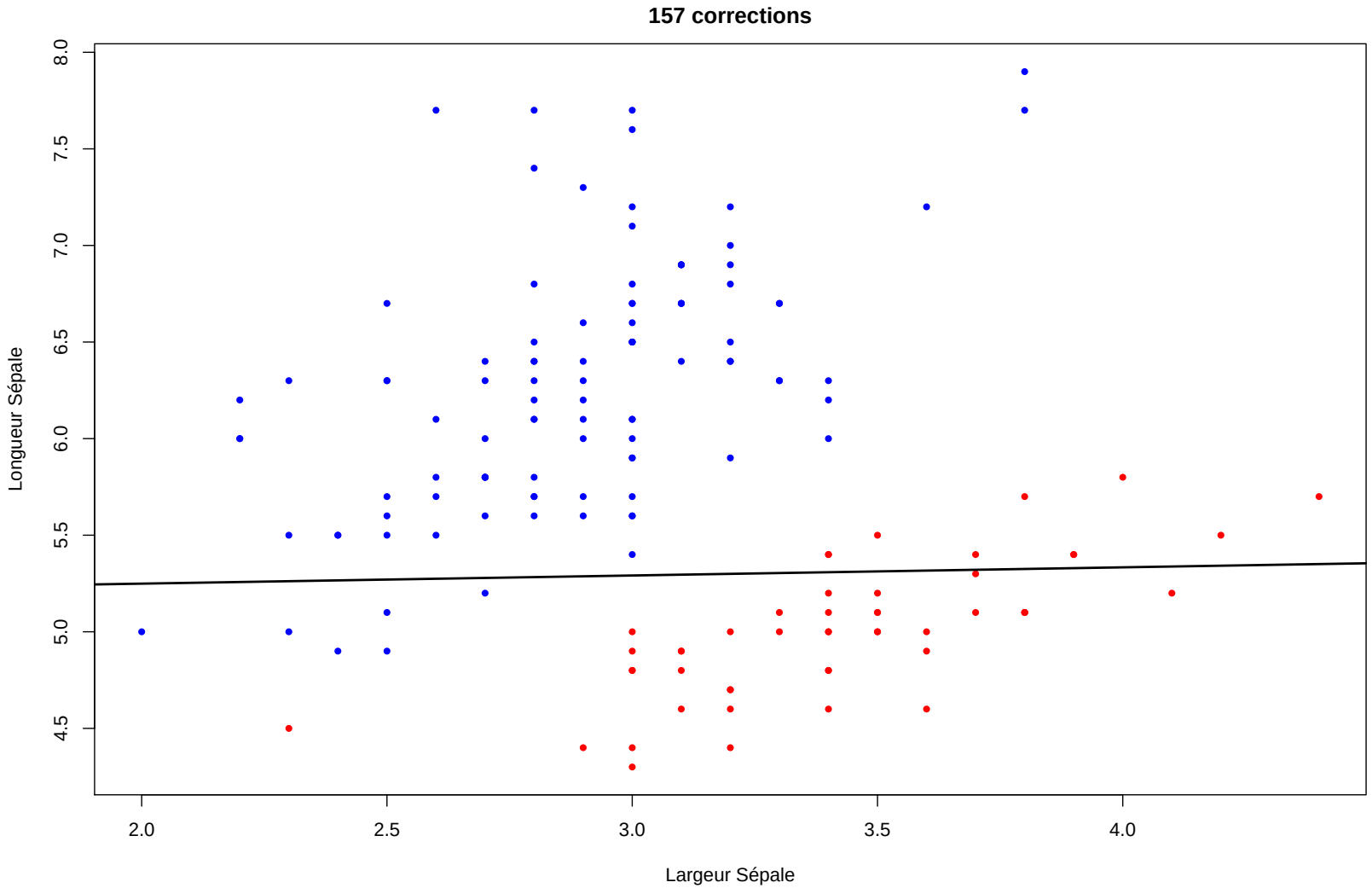
Exemple de mise en œuvre



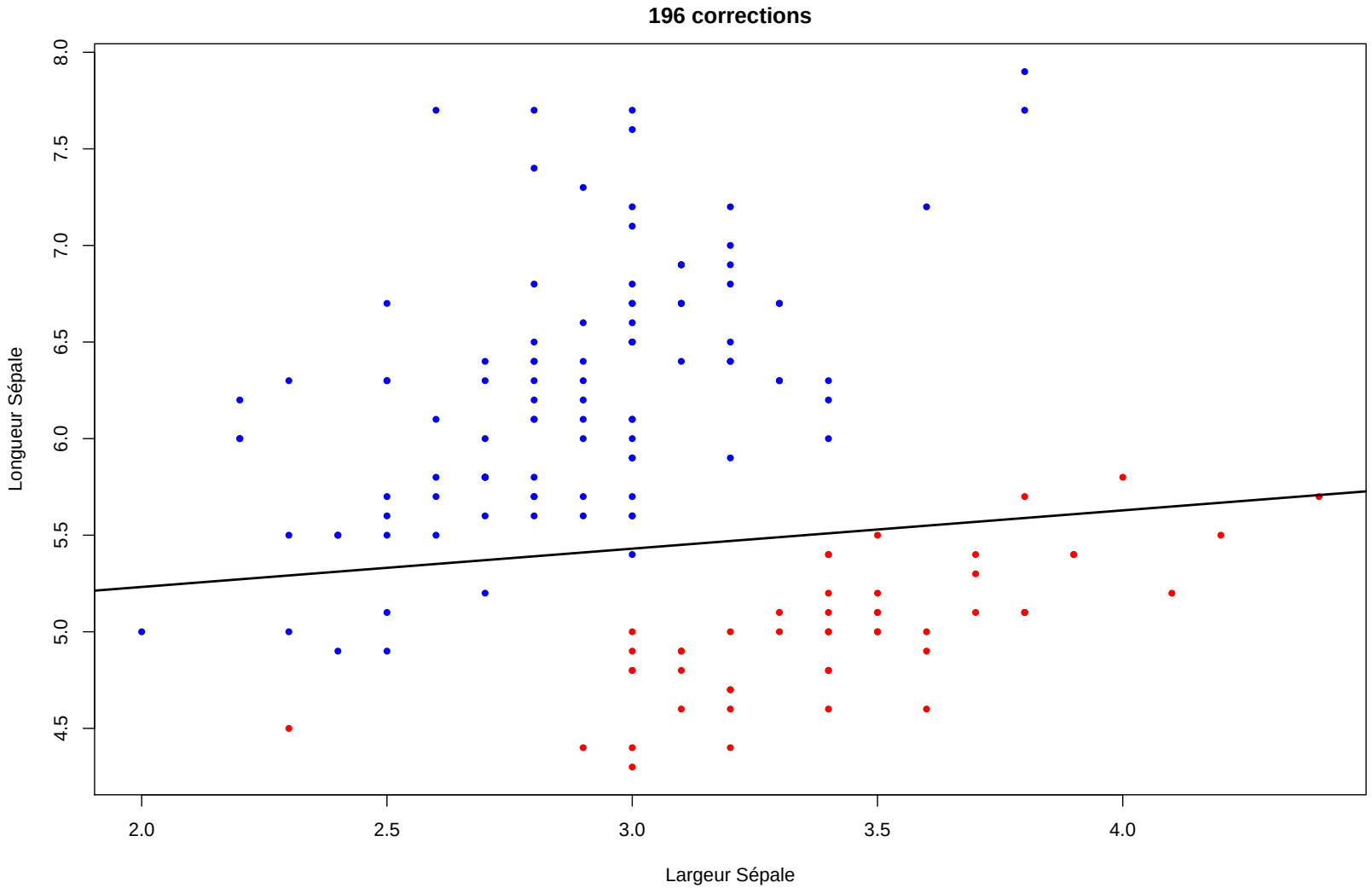
Exemple de mise en œuvre



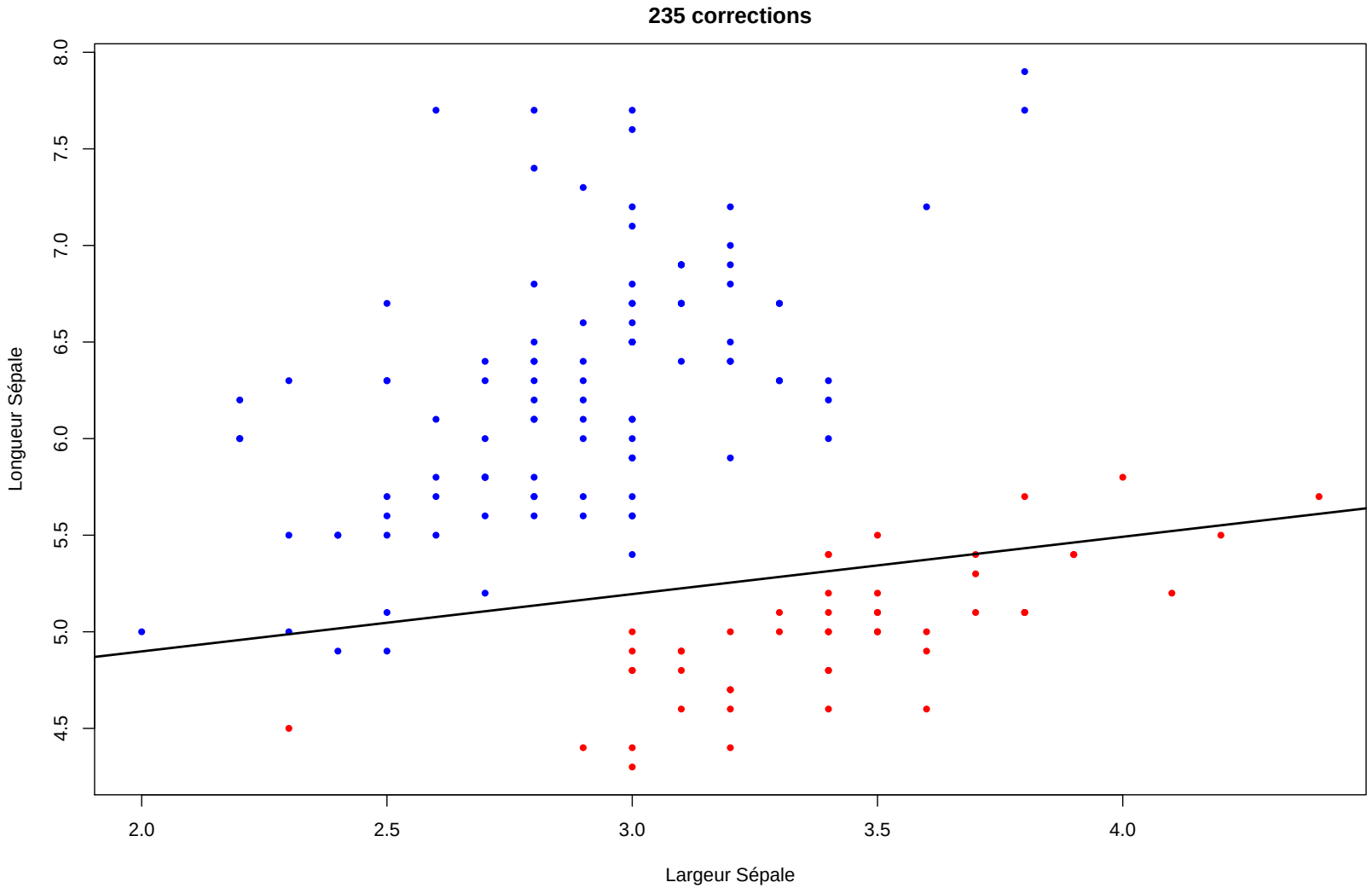
Exemple de mise en œuvre



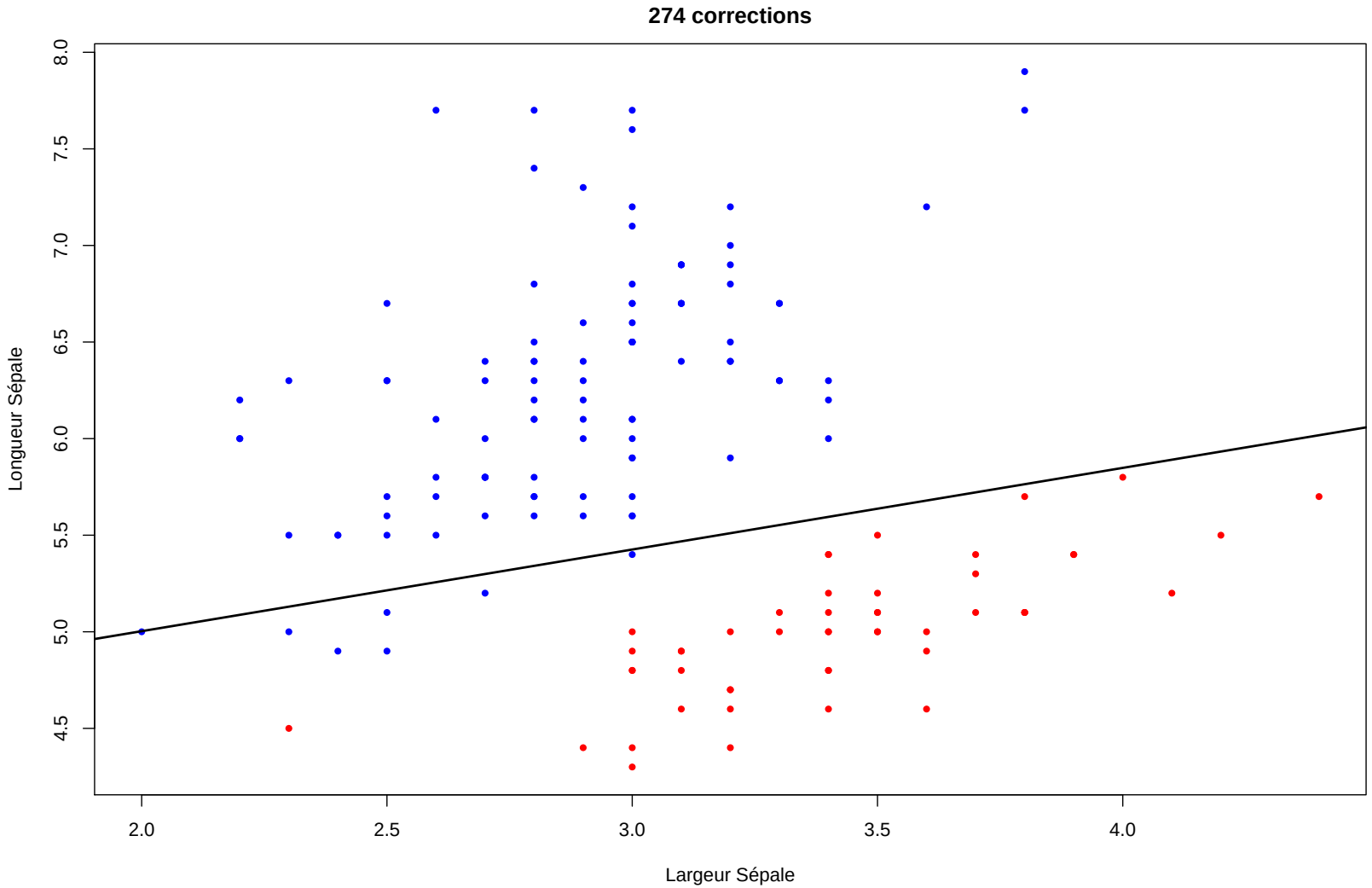
Exemple de mise en œuvre



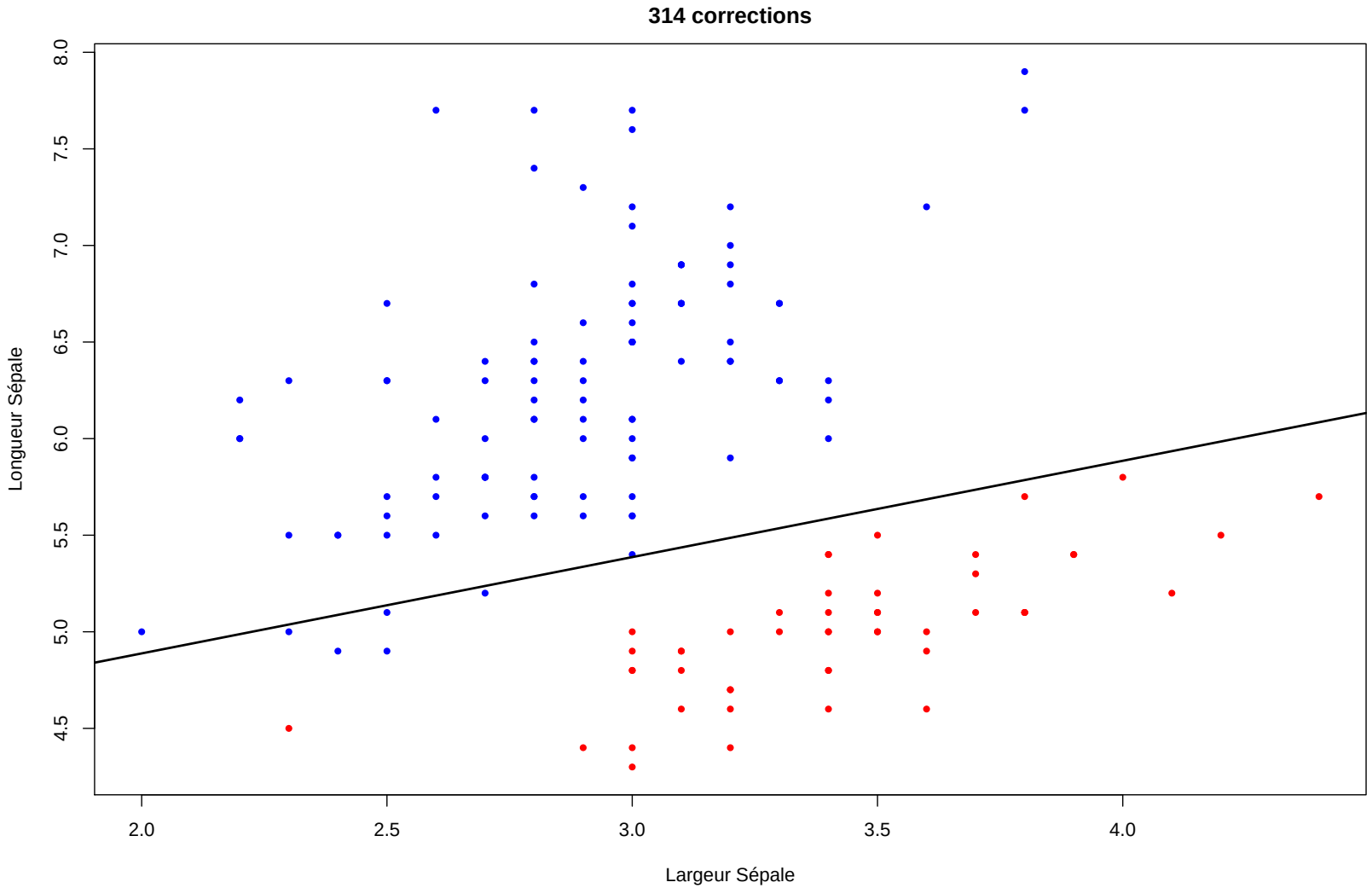
Exemple de mise en œuvre



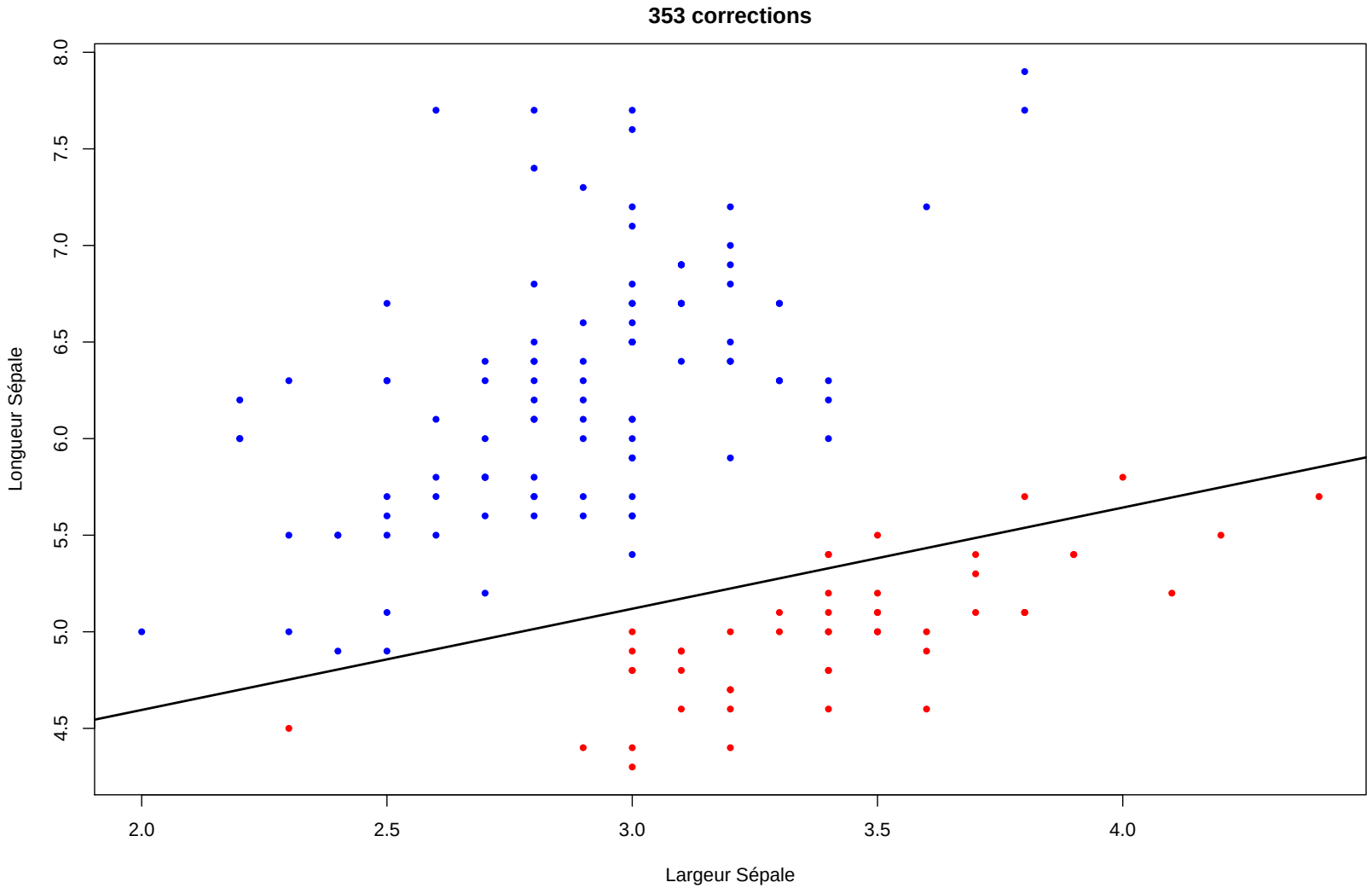
Exemple de mise en œuvre



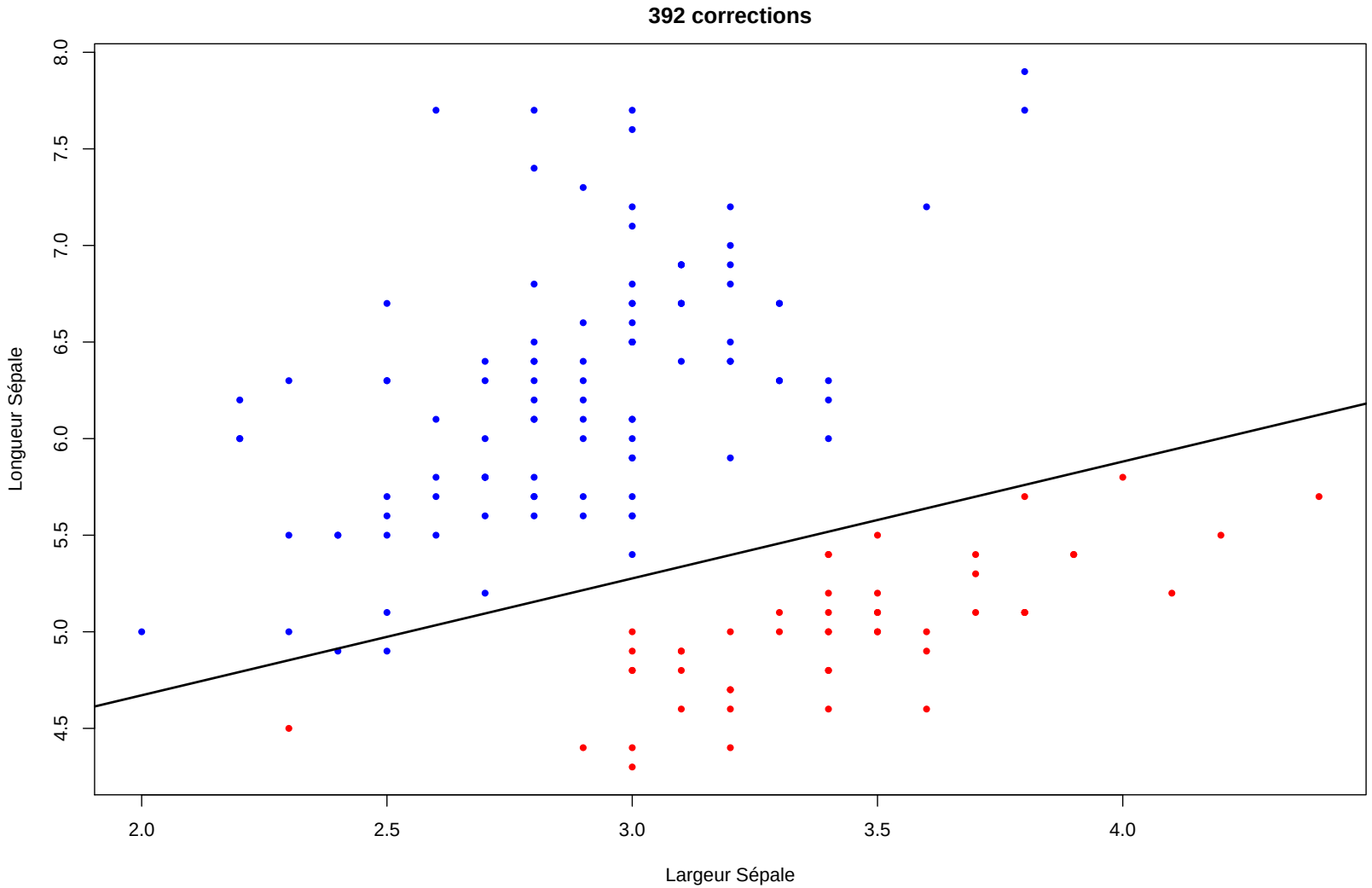
Exemple de mise en œuvre



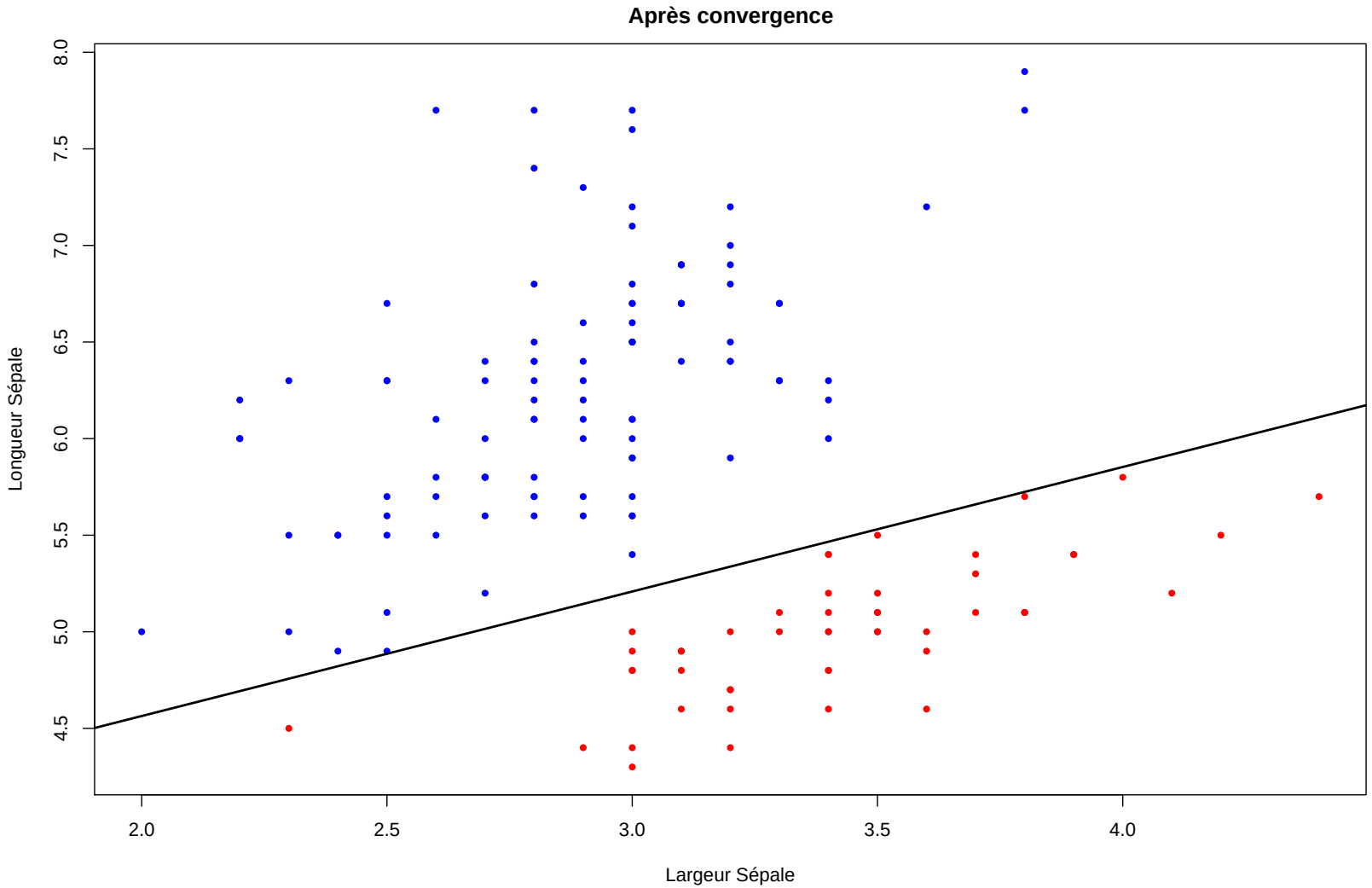
Exemple de mise en œuvre



Exemple de mise en œuvre



Exemple de mise en œuvre



Gradient stochastique

- la fonction optimisée dépend de la donnée considérée
  $f(\mathbf{w}) = -y\left(b + \sum_{j=1}^{p} w_j x_j\right)$
- algorithme de gradient stochastique :
  - fonction de la forme $f(\mathbf{w}) = \sum_{i=1}^{N} f_i(\mathbf{w})$
  - optimisation grâce à la règle

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \epsilon^{(t)} \nabla f_{i(t)}(\mathbf{w}^{(t)}),$$

  avec $i^{(t)}$ choisi aléatoirement dans $[1, N]$
- pour le perceptron :

$$f(\mathbf{w}) = \sum_{i=1}^{N} c_+\left(-y_i\left(b + \sum_{j=1}^{p} w_j x_{ij}\right)\right),$$

avec $c_+(u) = 0$ si $u < 0$ et $c_+(u) = u$ sinon

Perceptron

Résumé

- méthode neuronale pour la discrimination en deux classes
- algorithme simple à programmer
- justifications :
  - biologique (règle de Hebb)
  - mathématique (optimisation par gradient stochastique)
- limité aux données linéairement séparables
- ne converge pas en cas de problème
- aucun contrôle sur le séparateur linéaire obtenu

Perceptron

Résumé

- méthode neuronale pour la discrimination en deux classes
- algorithme simple à programmer
- justifications :
  - biologique (règle de Hebb)
  - mathématique (optimisation par gradient stochastique)
- limité aux données linéairement séparables
- ne converge pas en cas de problème
- aucun contrôle sur le séparateur linéaire obtenu

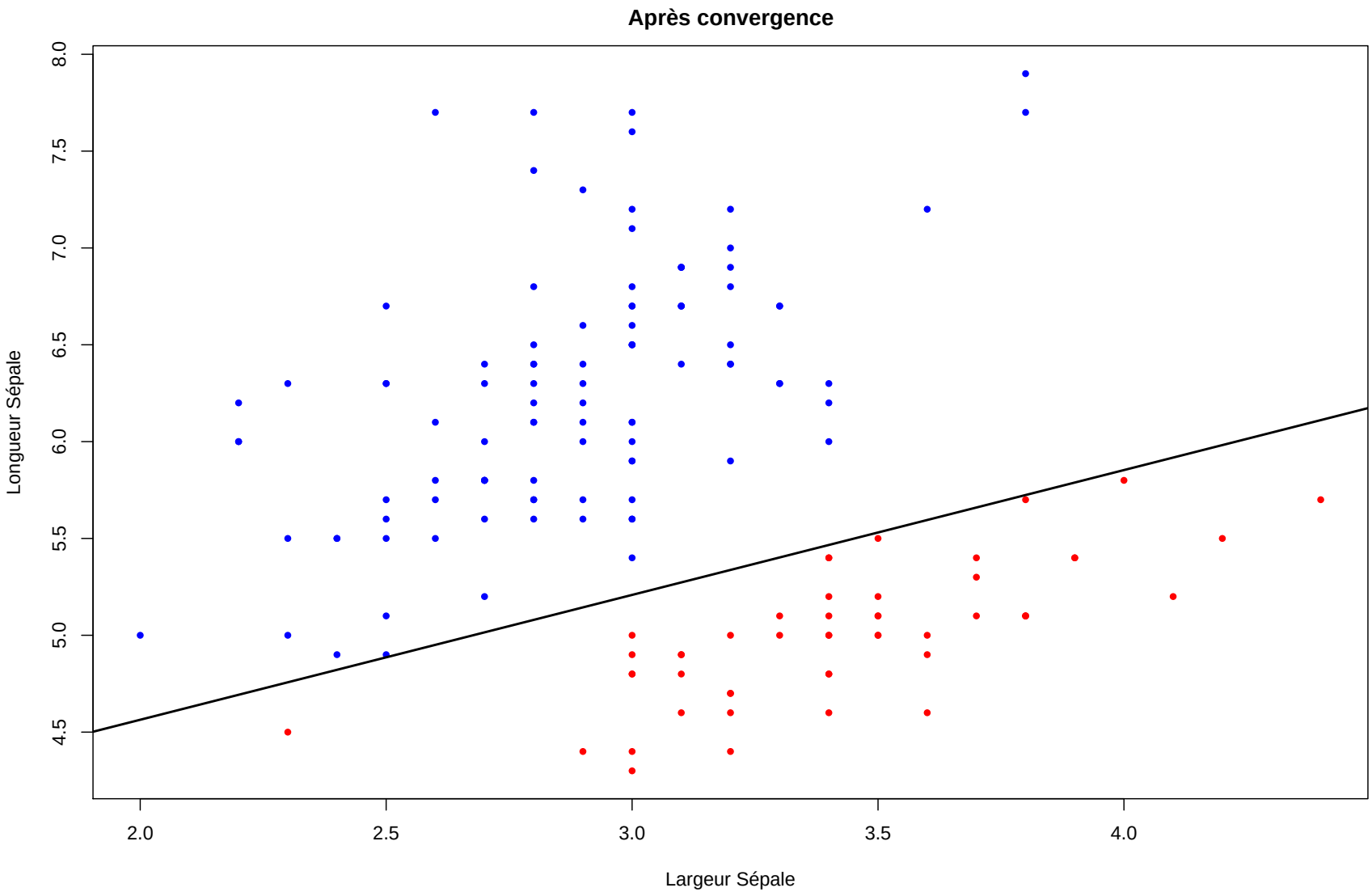
Choix d'un séparateur linéaire

Deux problèmes principaux :

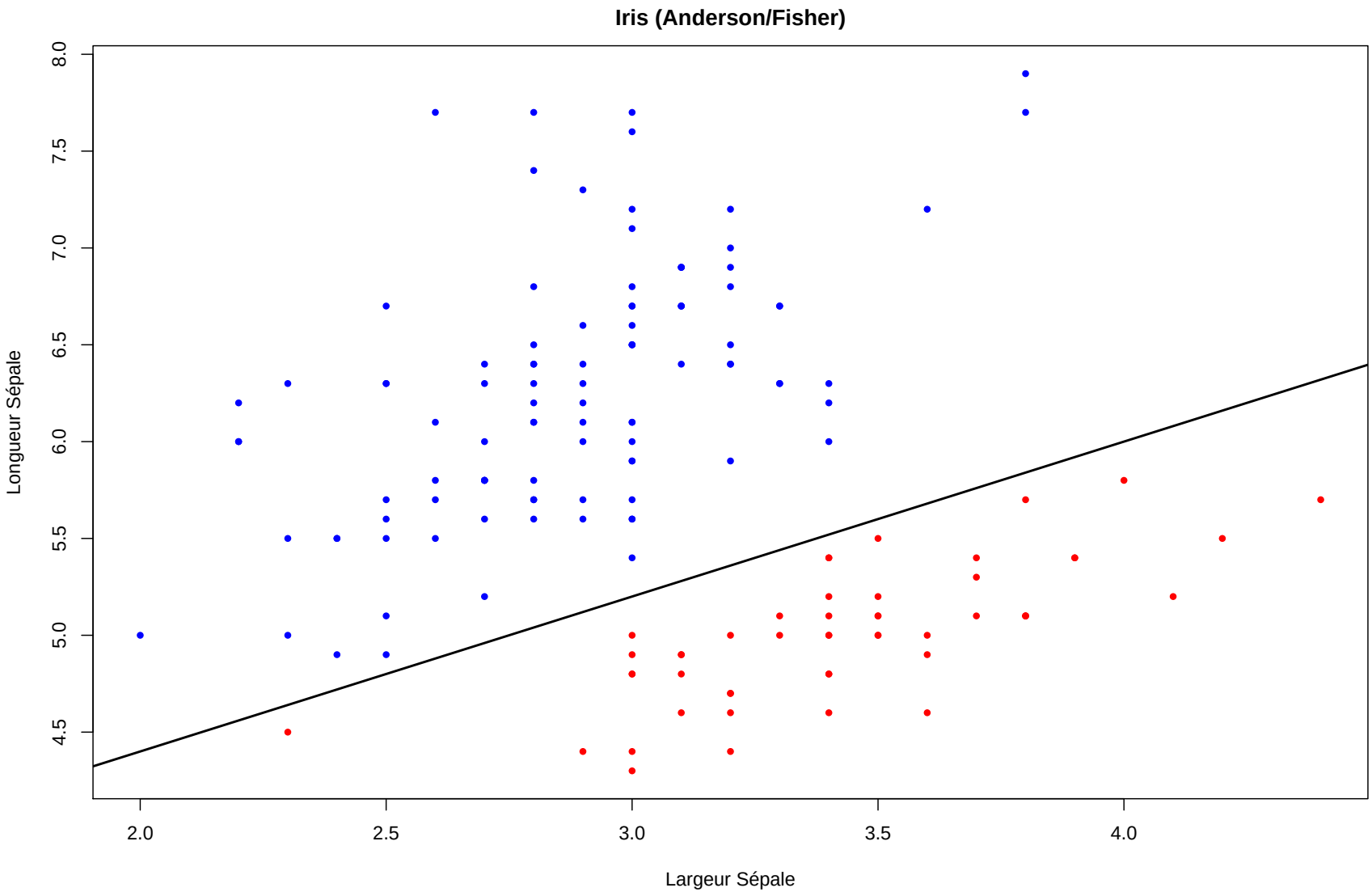
1. cas séparable : en général une infinité de solutions, comment choisir ?
   - critère additionnel : choix parmi les solutions exactes (avec aucune erreur)
   - critère alternatif : optimisation d'une autre grandeur (pas le nombre d'erreurs)
2. cas non séparable : comment minimiser le nombre d'erreurs ?
   - algorithme alternatif (autre que celui du perceptron)
   - critère alternatif (bis)

Question subsidiaire : le cas non linéaire est-il fréquent ?

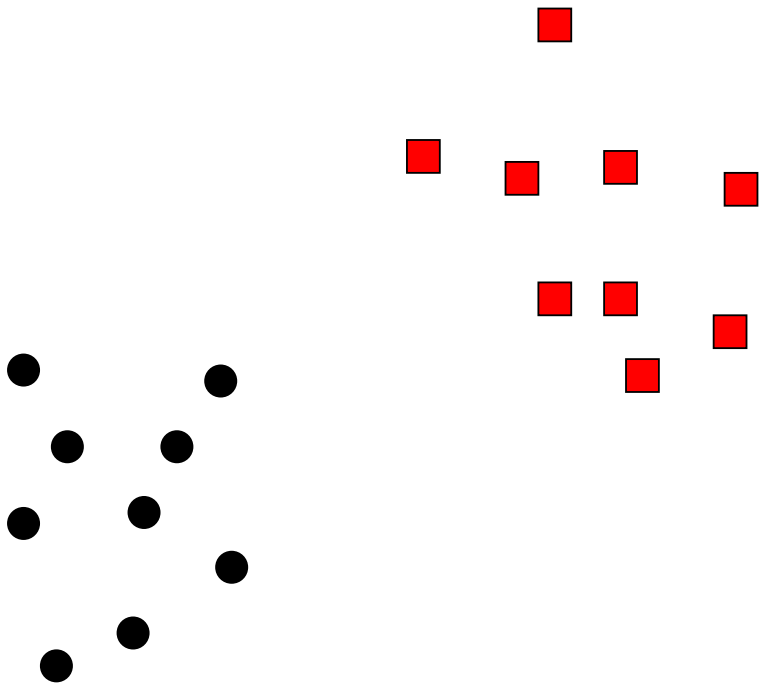
Problème de la robustesse



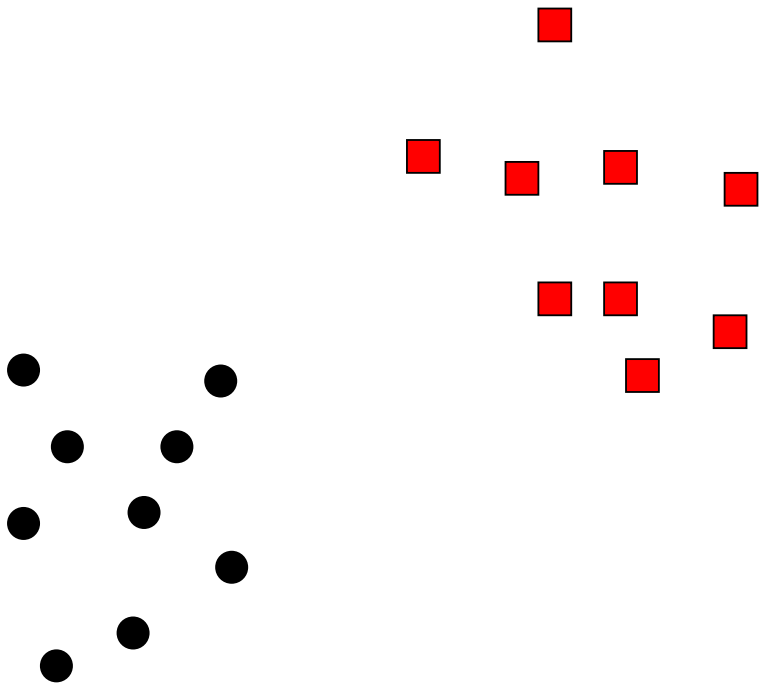
Problème de la robustesse



Maximisation de la marge

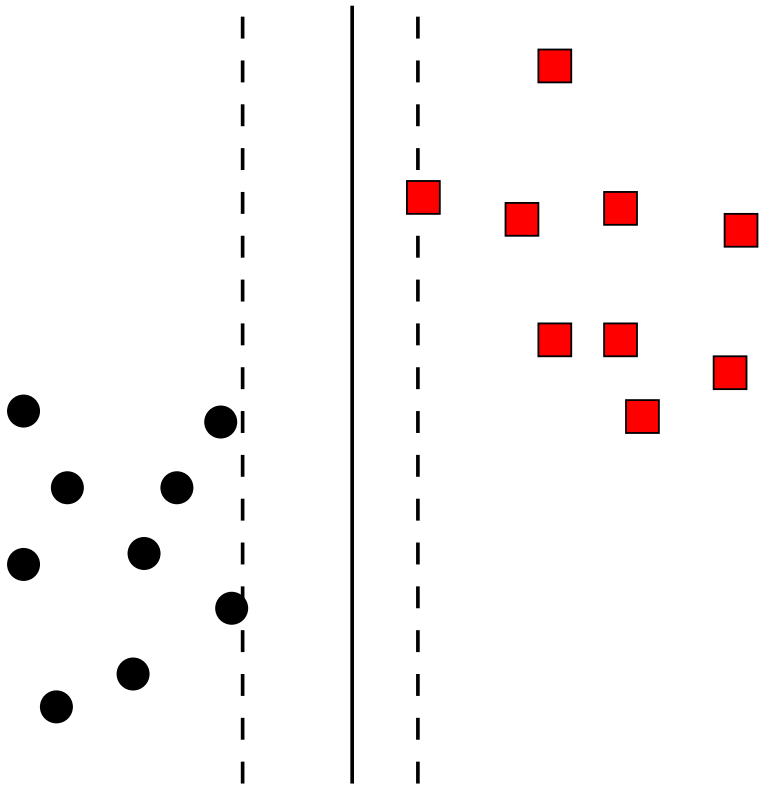


Maximisation de la marge



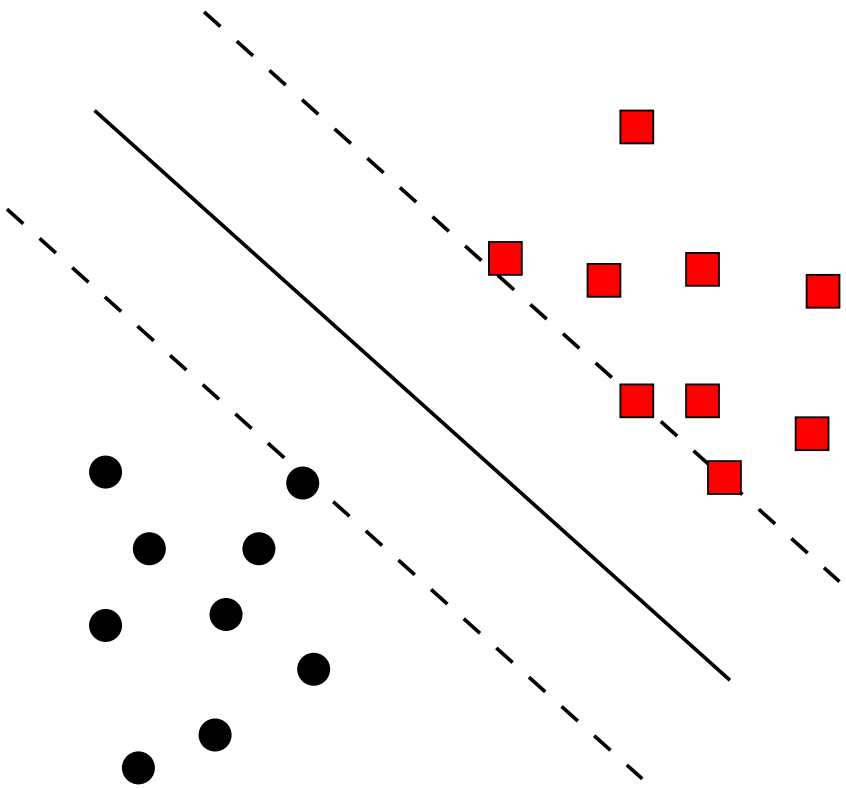
- Données linéairement séparables : une infinité de choix possibles

Maximisation de la marge



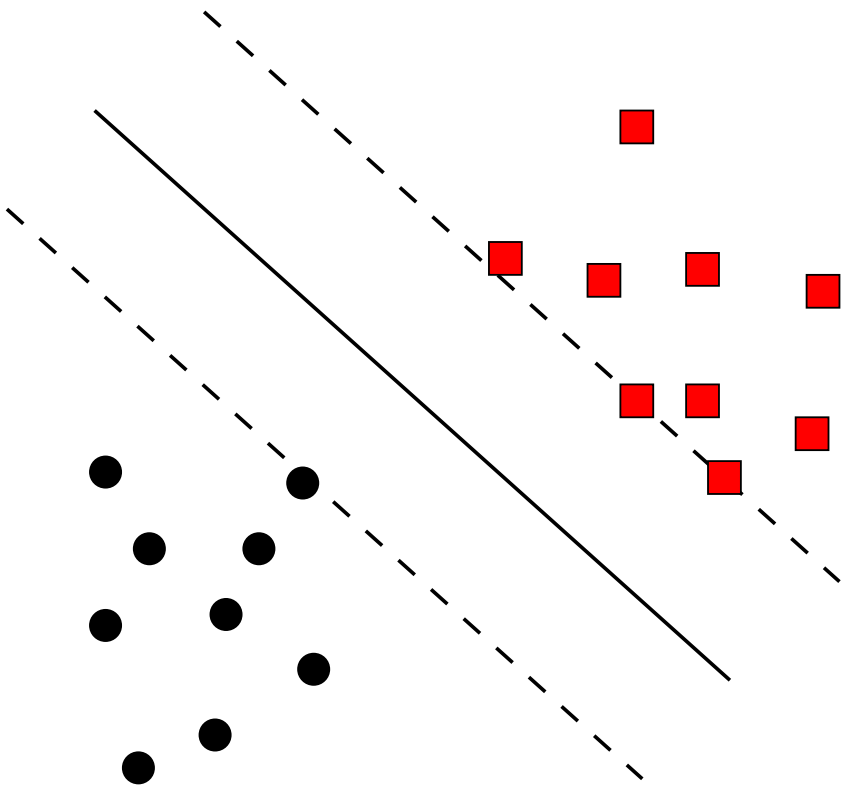
- Données linéairement séparables : une infinité de choix possibles
- Données proches du séparateur : petite « marge » $\Rightarrow$ faible robustesse

Maximisation de la marge



- Données linéairement séparables : une infinité de choix possibles
- Données proches du séparateur : petite « marge » $\Rightarrow$ faible robustesse
- Un critère de choix possible : maximiser la marge

Maximisation de la marge



- Données linéairement séparables : une infinité de choix possibles
- Données proches du séparateur : petite « marge » $\Rightarrow$ faible robustesse
- Un critère de choix possible : maximiser la marge
- Machine à vecteurs de support

Formulation du problème

- marge : distance entre le séparateur et l'observation la plus proche
- notation $\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{j=1}^{p} w_j x_j$
- marge :

$$\min_i \frac{|\langle \mathbf{w}, \mathbf{x}_i \rangle + b|}{\langle \mathbf{w}, \mathbf{w} \rangle} = \min_i \frac{y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)}{\langle \mathbf{w}, \mathbf{w} \rangle},$$

  en l'absence d'erreur, c.-à-d., avec $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) > 0$ pour tout i
- normalisation par $\min_i y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$:

$$(P_0) \quad \min_{\mathbf{w},b} \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle,$$
$$\text{sous les contraintes } y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \ 1 \leq i \leq N.$$

- problème d'optimisation quadratique sous contraintes linéaires
- formulation duale plus simple

Optimisation sous contraintes

- ajout de contraintes $\Rightarrow$ problème généralement plus difficile
- mais ici la fonction à optimiser est quadratique
- stratégie générale :
  - point initial $\mathbf{w}^{(0)}$ admissible (c.-à-d., qui vérifie les contraintes)
  - amélioration progressive de la fonction à optimiser en restant admissible
- nombreuses stratégies possibles
- concept important : contraintes saturées (ici quand $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) = 1$)
- principale difficulté ici : trouver un point initial admissible

Formulation duale

- Lagrangien (multiplicateurs $\alpha_i \geq 0$) :

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle - \sum_{i=1}^{N} \alpha_i \left(y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 \right)$$

- minimum par rapport à $(\mathbf{w}, b)$ et maximum par rapport à $\boldsymbol{\alpha}$ (point selle)
- à un point selle, $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0$ et $\frac{\partial \mathcal{L}}{\partial b} = 0$, soit $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$ et $\sum_{i=1}^{N} \alpha_i y_i = 0$
- les conditions de Karush-Kuhn-Tucker impliquent aussi (à l'optimum)

$$\alpha_i \left(y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 \right) = 0$$

Formulation duale

- (P_0) est donc équivalent à

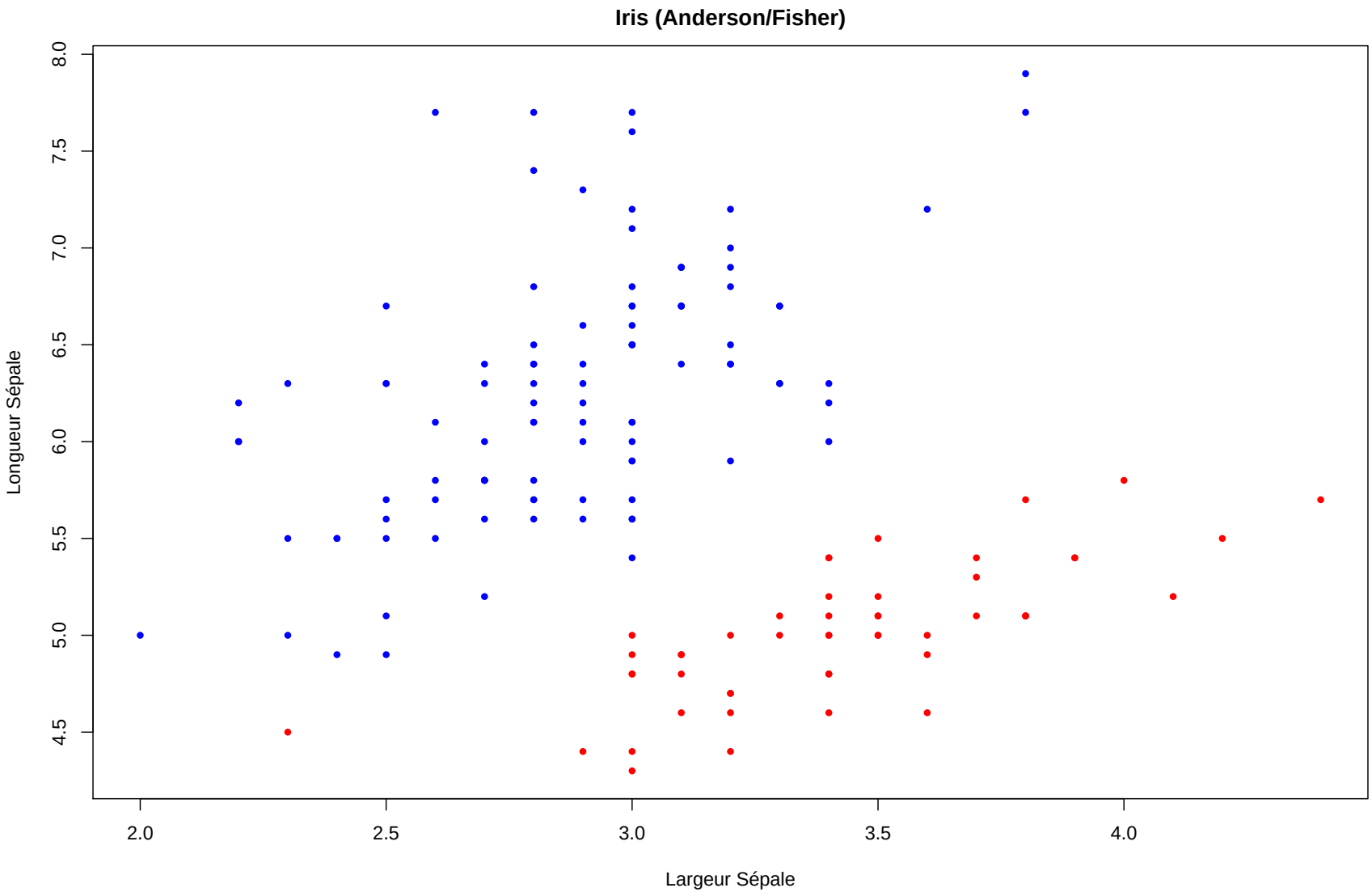
$$(D_0)\ \max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$
$$\text{sous les contraintes } \sum_{i=1}^{N} \alpha_i y_i = 0 \text{ et } \alpha_i \geq 0$$

- problème plus facile à résoudre :
  - toujours quadratique
  - contraintes plus simples
- quand $(y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1) > 0$, $\alpha_i = 0$:
  - les observations éloignées du séparateur n'interviennent pas dans la solution
  - la solution dépend uniquement des observations « sur la marge » : les vecteurs de support (contraintes saturées)
  - $\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{\alpha_i \neq 0} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle$

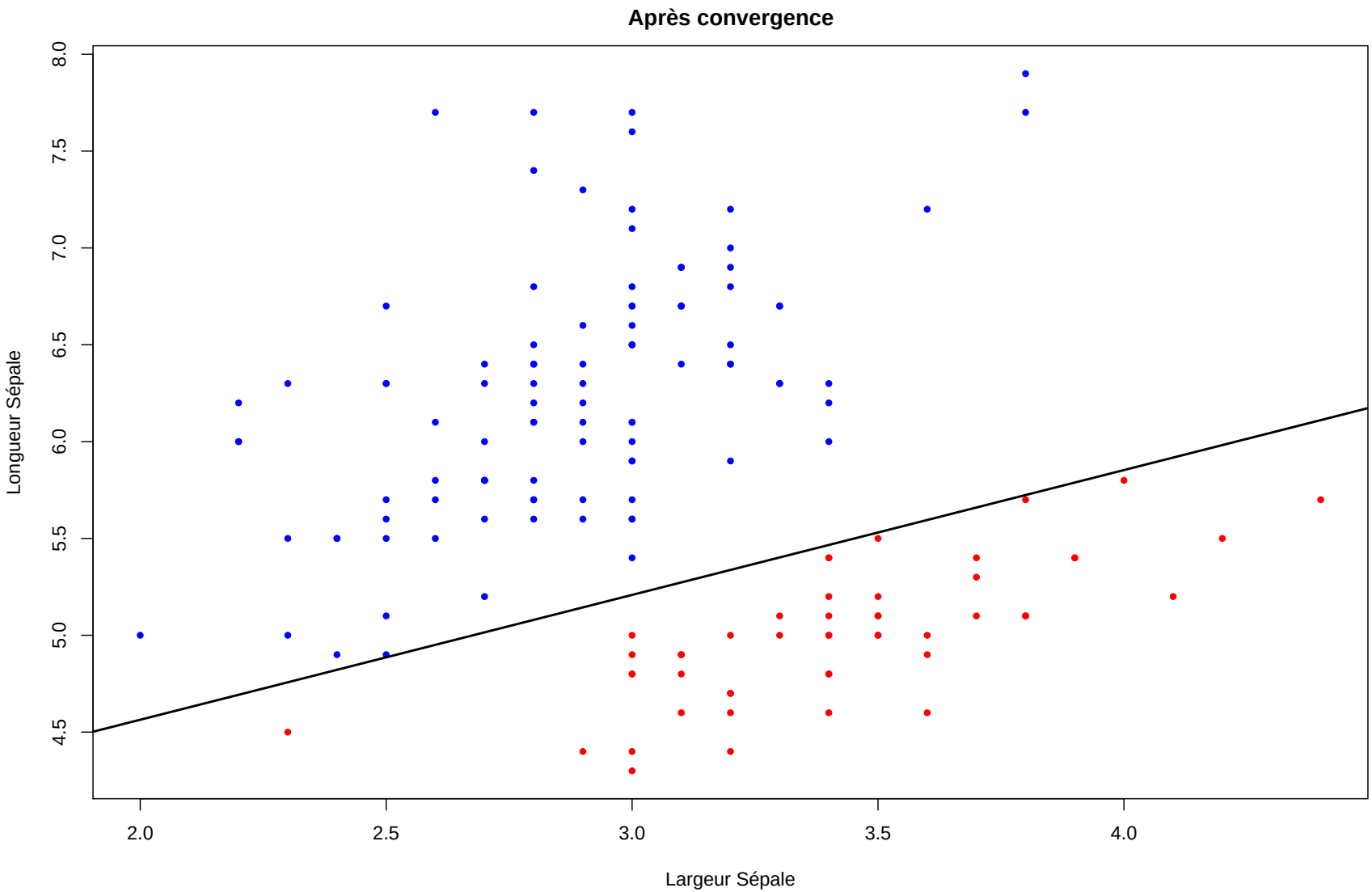
Algorithme SMO

- *Sequential Minimal Optimization*
- mise à jour de deux multiplicateurs $\alpha_i^{(t)}$ et $\alpha_j^{(t)}$ simultanément à chaque itération (les autres sont fixés)
- contraintes réduites :
  - $\alpha_i^{(t+1)} y_i + \alpha_j y_j^{(t+1)} = \alpha_i^{(t)} y_i + \alpha_j y_j^{(t)}$
  - $\alpha_i^{(t+1)} \geq 0$ et $\alpha_j^{(t+1)} \geq 0$
- solution calculable exactement
- j et j sont choisis par exemple parmi les multiplicateurs qui ne vérifient pas la contrainte $\alpha_i \left(y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1\right) = 0$
- cf `http://research.microsoft.com/~jplatt/abstracts/smo.html`

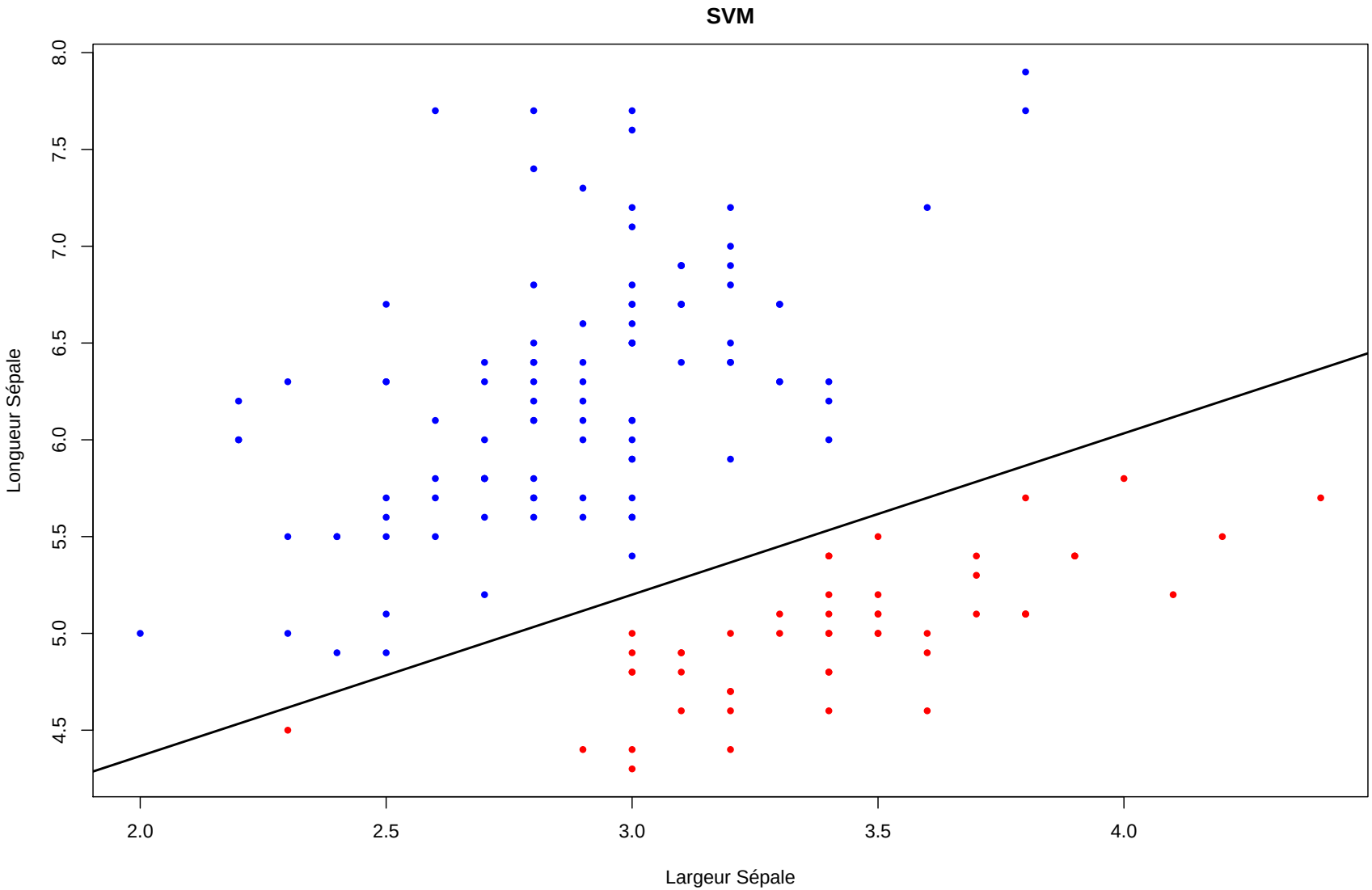
Exemple


Iris (Anderson/Fisher)
Longueur Sépale
Largeur Sépale
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
2.0
2.5
3.0
3.5
4.0

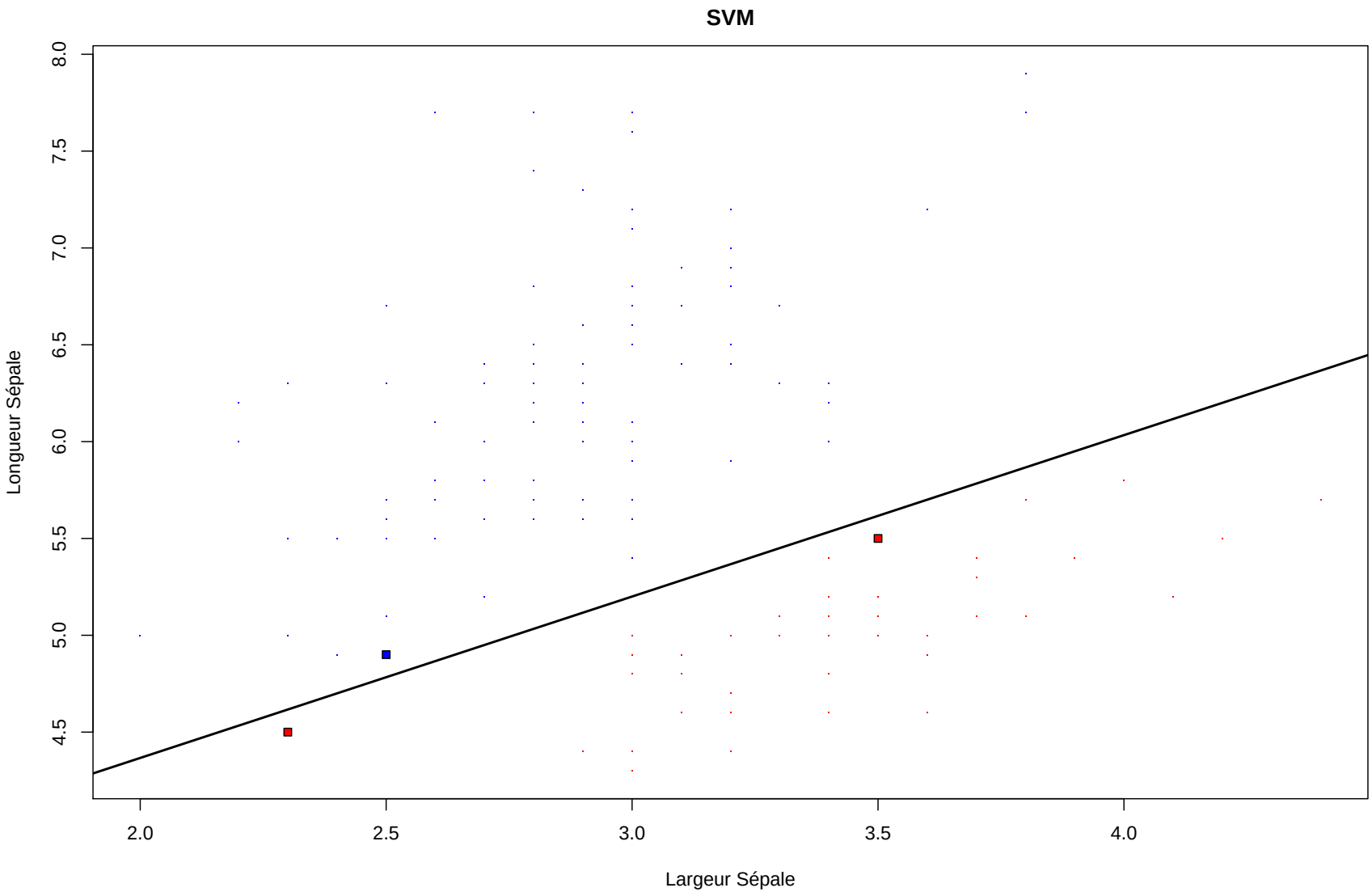
Exemple


Après convergence
Longueur Sépale
Largeur Sépale
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
2.0
2.5
3.0
3.5
4.0

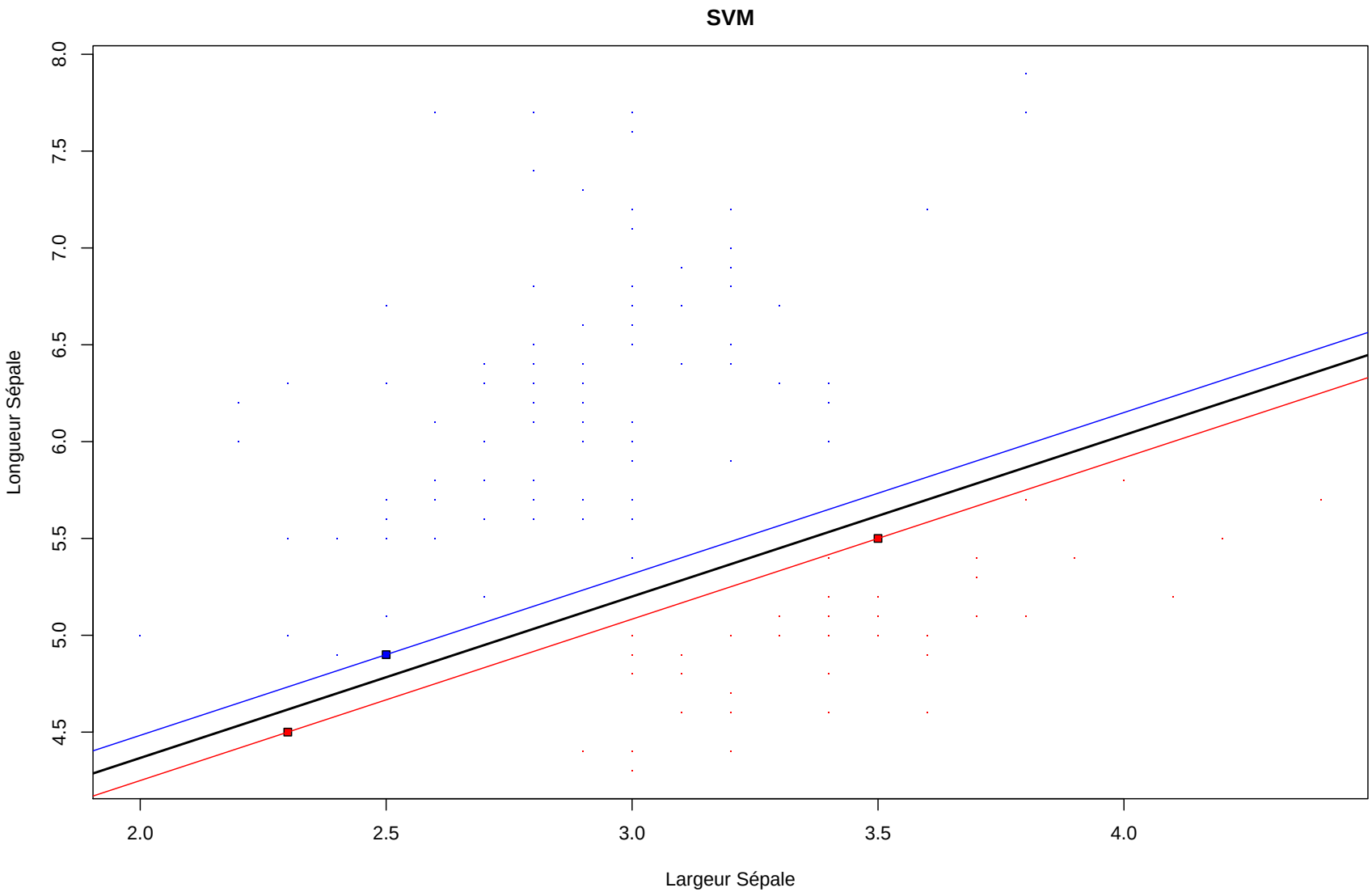
Exemple


SVM
Longueur Sépale
Largeur Sépale
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
2.0
2.5
3.0
3.5
4.0

Exemple


SVM
Longueur Sépale
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
2.0
2.5
3.0
3.5
4.0
Largeur Sépale

Exemple


SVM
Longueur Sépale
Largeur Sépale
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
2.0
2.5
3.0
3.5
4.0

Cas non linéairement séparable

- le problème P_0 n'a pas de solution : pas de point admissible
- assouplir les contraintes :
  - autoriser des erreurs de classement
  - conserver la notion de marge pour les points bien classés
  - $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$ avec $\xi_i \geq 0$
  - les ξ_i sont les « variables ressort »
- nouveau problème :

$$
\begin{array}{ll}
(P_C) & \min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{i=1}^{N} \xi_i, \\
& \text{avec} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \ 1 \leq i \leq N, \\
& \qquad \xi_i \geq 0, \ 1 \leq i \leq N.
\end{array}
$$

- variantes possibles (par exemple $C \sum_{i=1}^{N} \xi_i^2$)

Interprétation

- (P_C) s'écrit aussi

$$\begin{array}{rl}(P_C) & \min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\langle \mathbf{w},\mathbf{w}\rangle + C\sum_{i=1}^{N}\xi_i, \\ & \text{avec} \quad \xi_i \geq 1 - y_i(\langle \mathbf{w},\mathbf{x}_i\rangle + b),\ 1 \leq i \leq N, \\ & \qquad \xi_i \geq 0,\ 1 \leq i \leq N.\end{array}$$

- de façon équivalente :

$$(P_C)\ \min_{\mathbf{w},b} \frac{1}{2}\langle \mathbf{w},\mathbf{w}\rangle + C\sum_{i=1}^{N}\max\left(1 - y_i(\langle \mathbf{w},\mathbf{x}_i\rangle + b), 0\right)$$

- interprétation de C :
  - compromis entre erreurs et marge
  - C grand : erreurs interdites, au détriment de la marge (le modèle « colle » aux données)
  - C petit : marge maximisée, au détriment des erreurs
  - choix de C délicat : cf la suite du cours

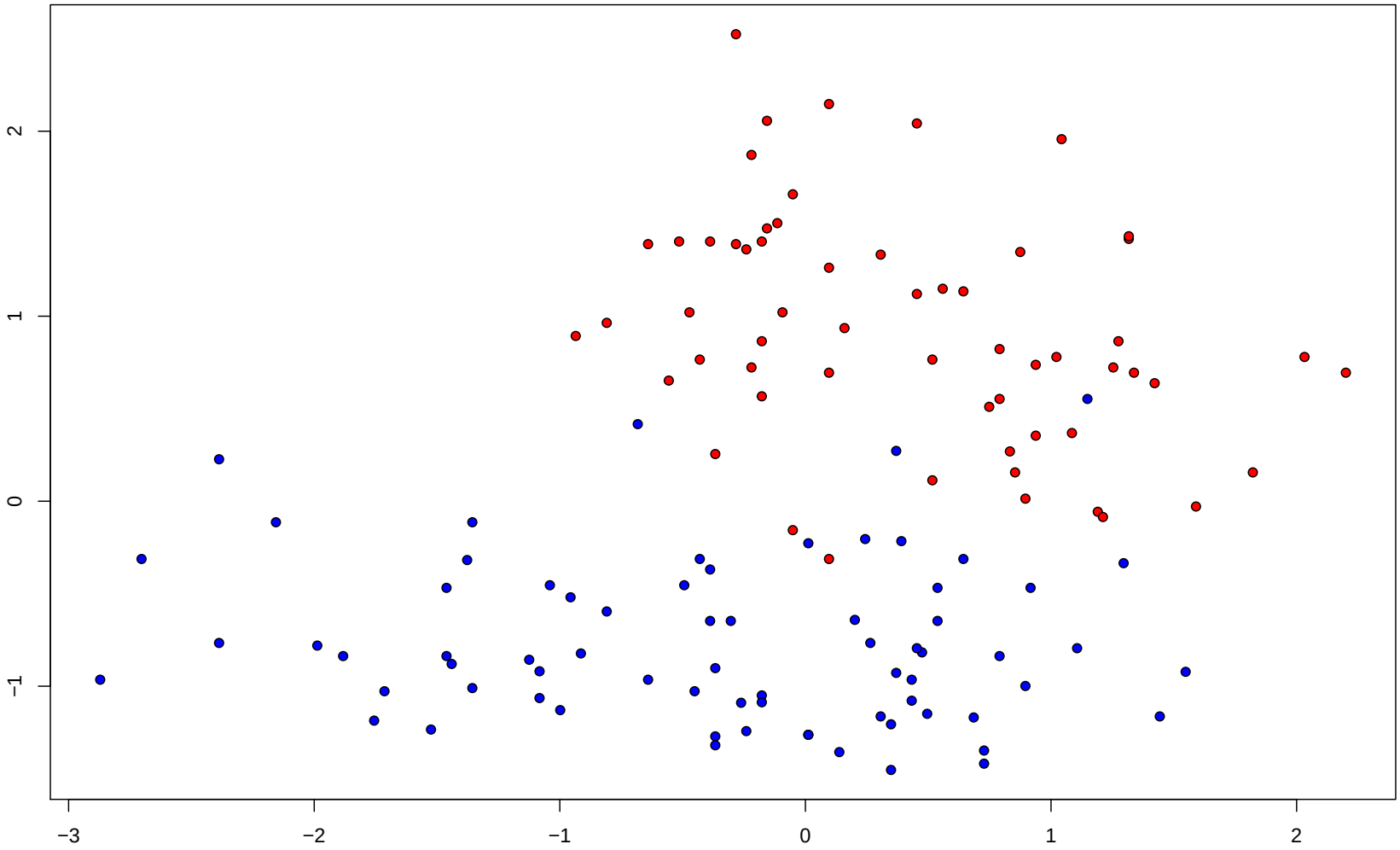
Problème dual

- (P_C) est équivalent à

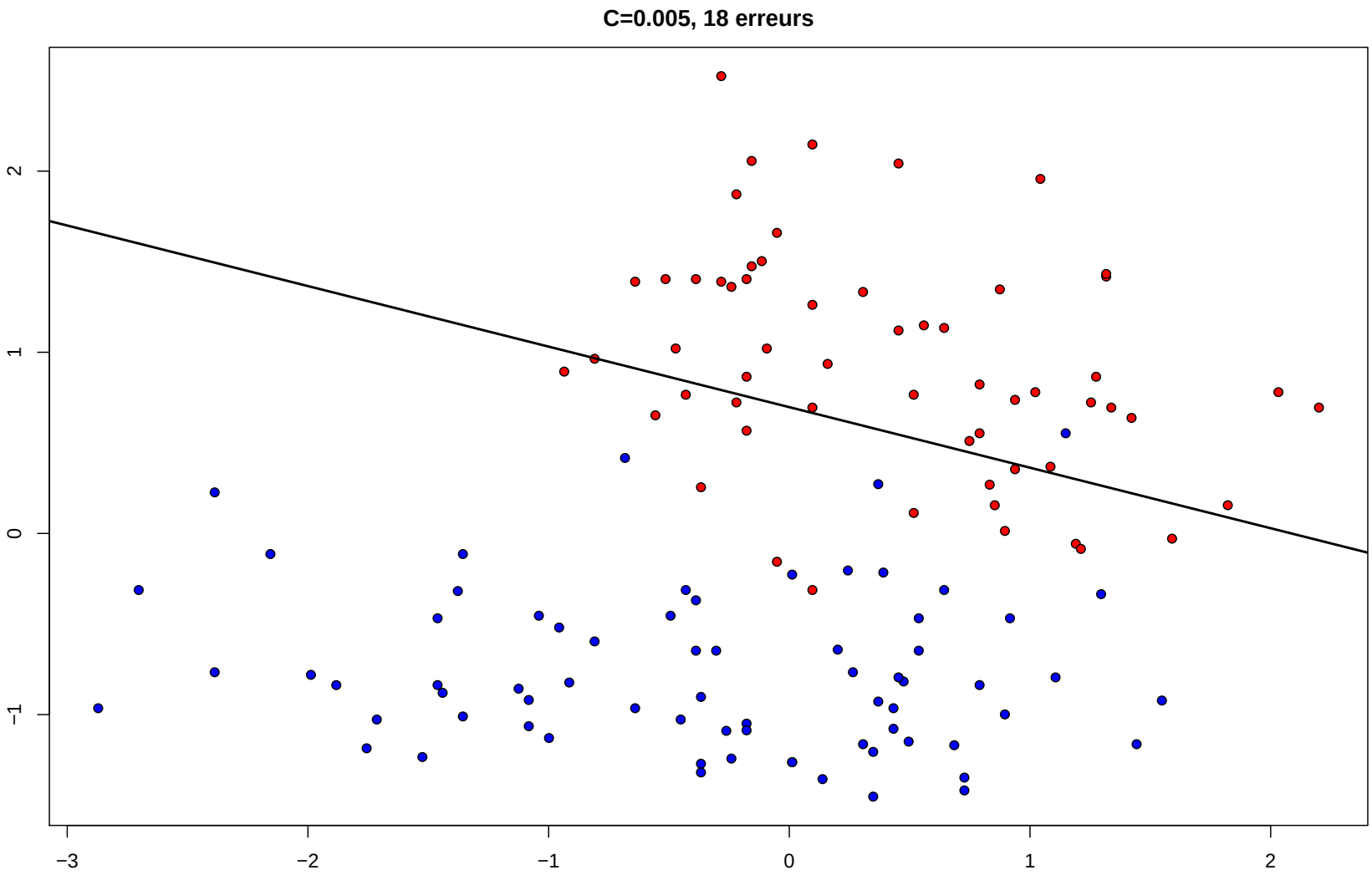
$$(D_C)\ \max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$
$$\text{sous les contraintes } \sum_{i=1}^{N} \alpha_i y_i = 0 \text{ et } 0 \leq \alpha_i \leq C$$

- seul changement : valeur maximale sur les multiplicateurs
- SMO toujours applicable

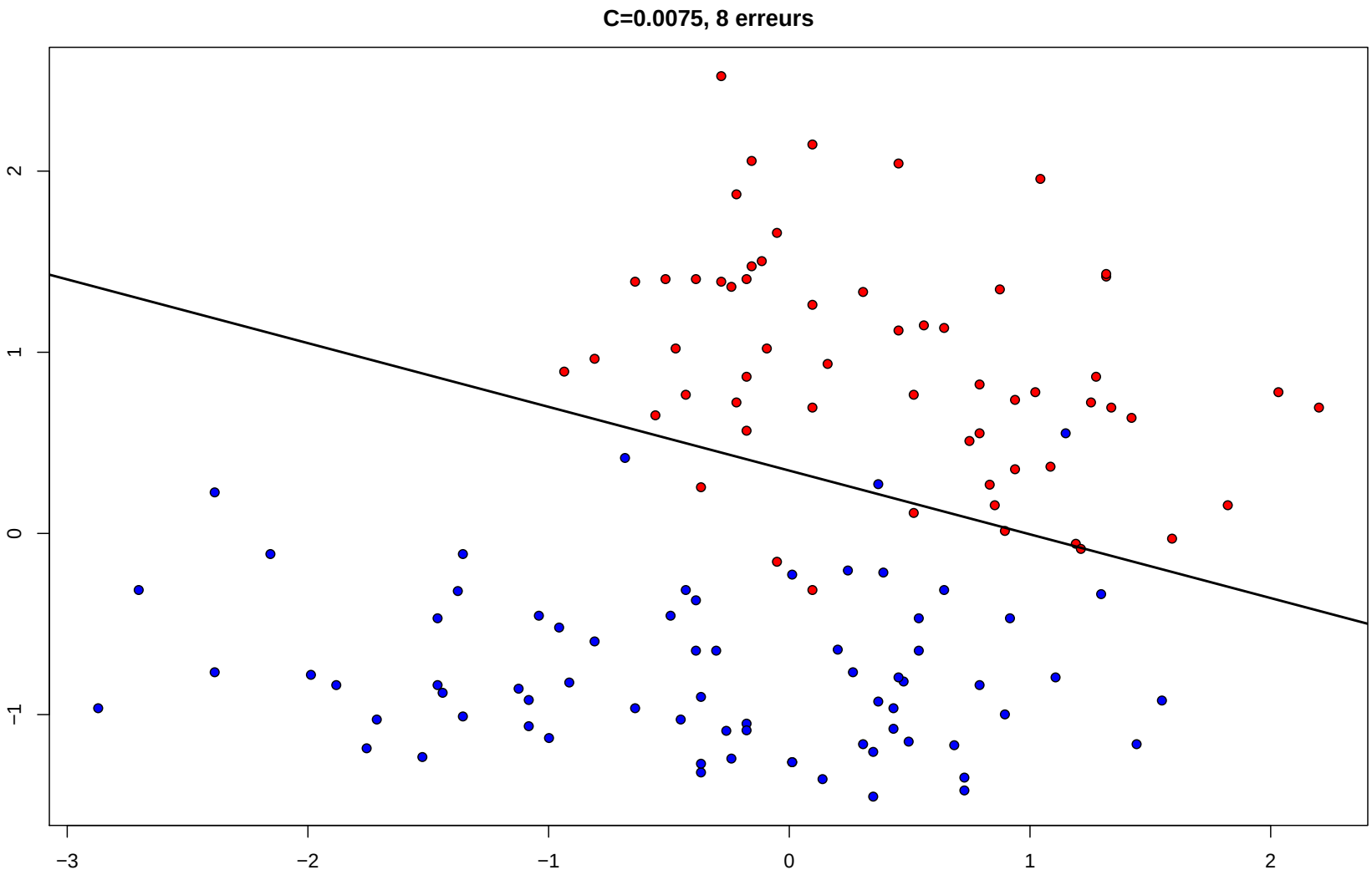
Exemple



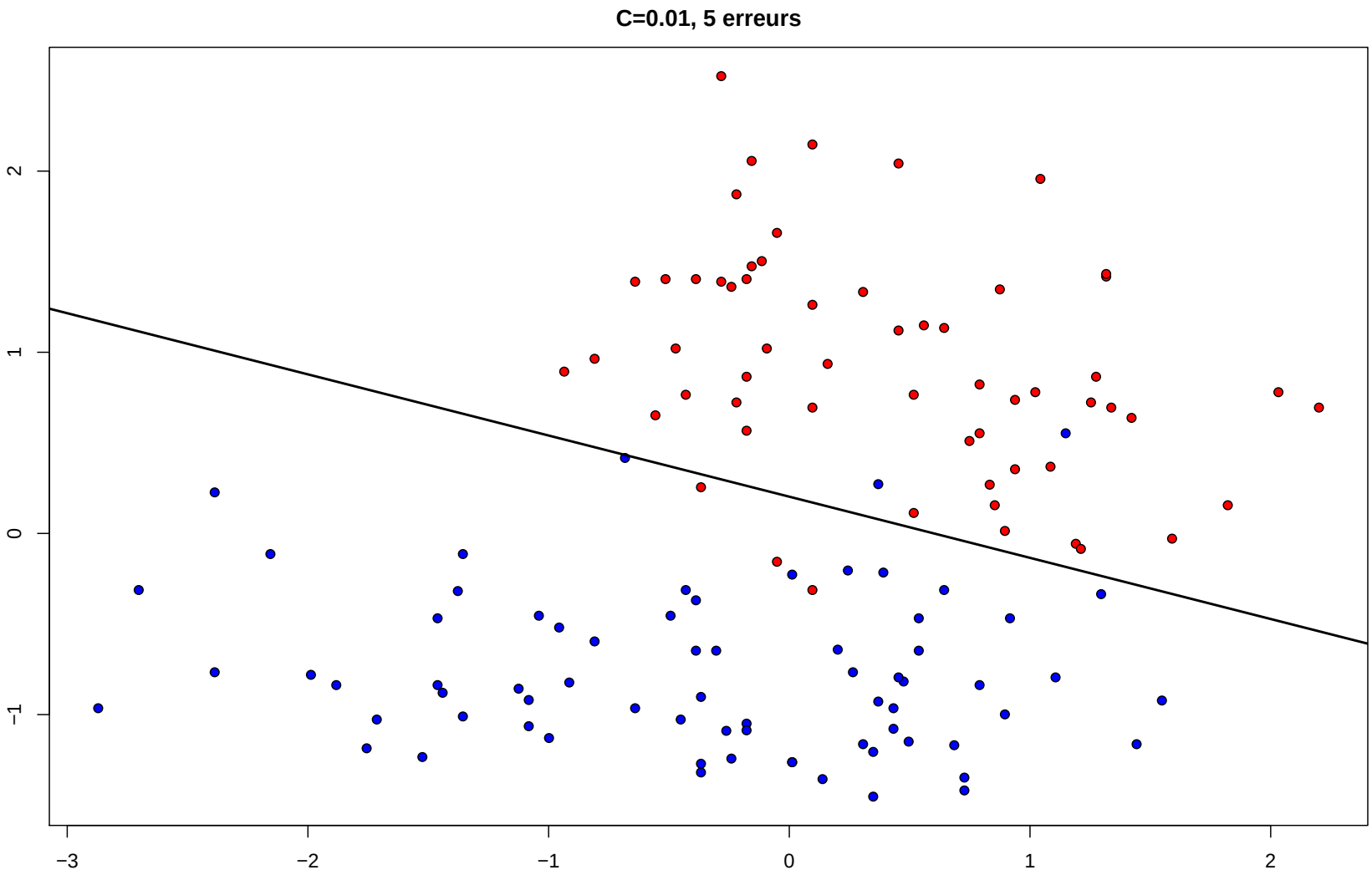
Exemple



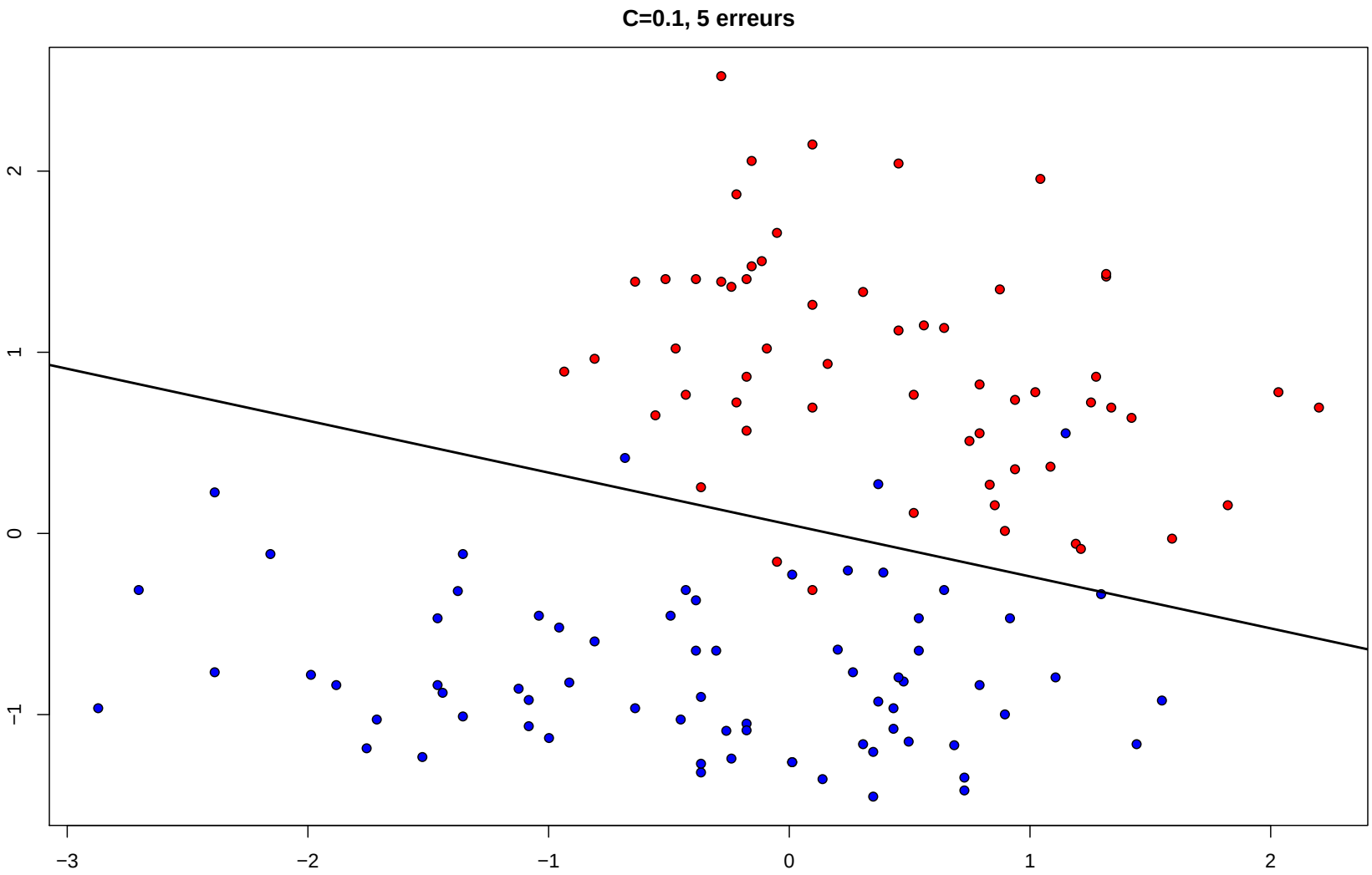
Exemple



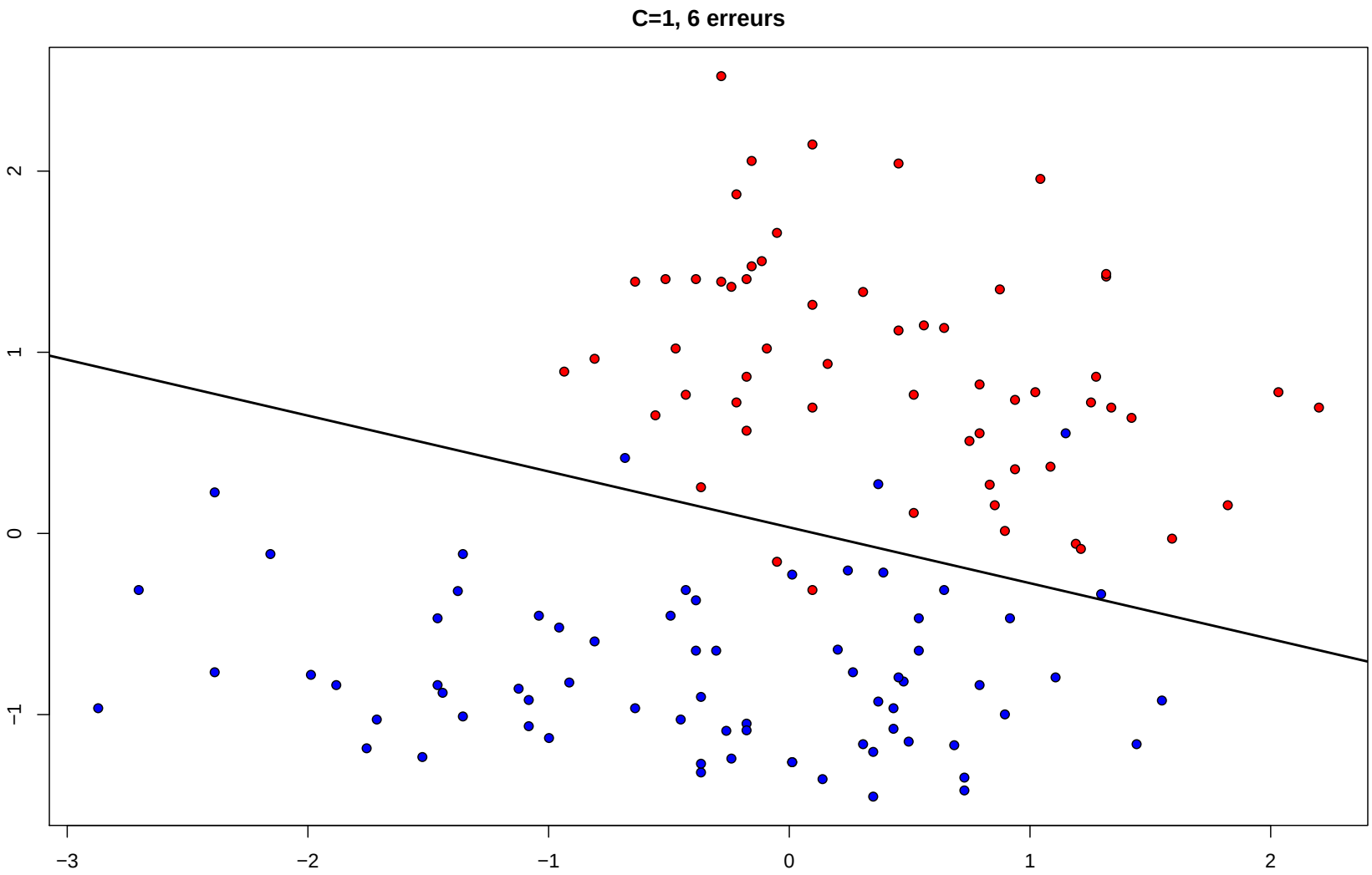
Exemple



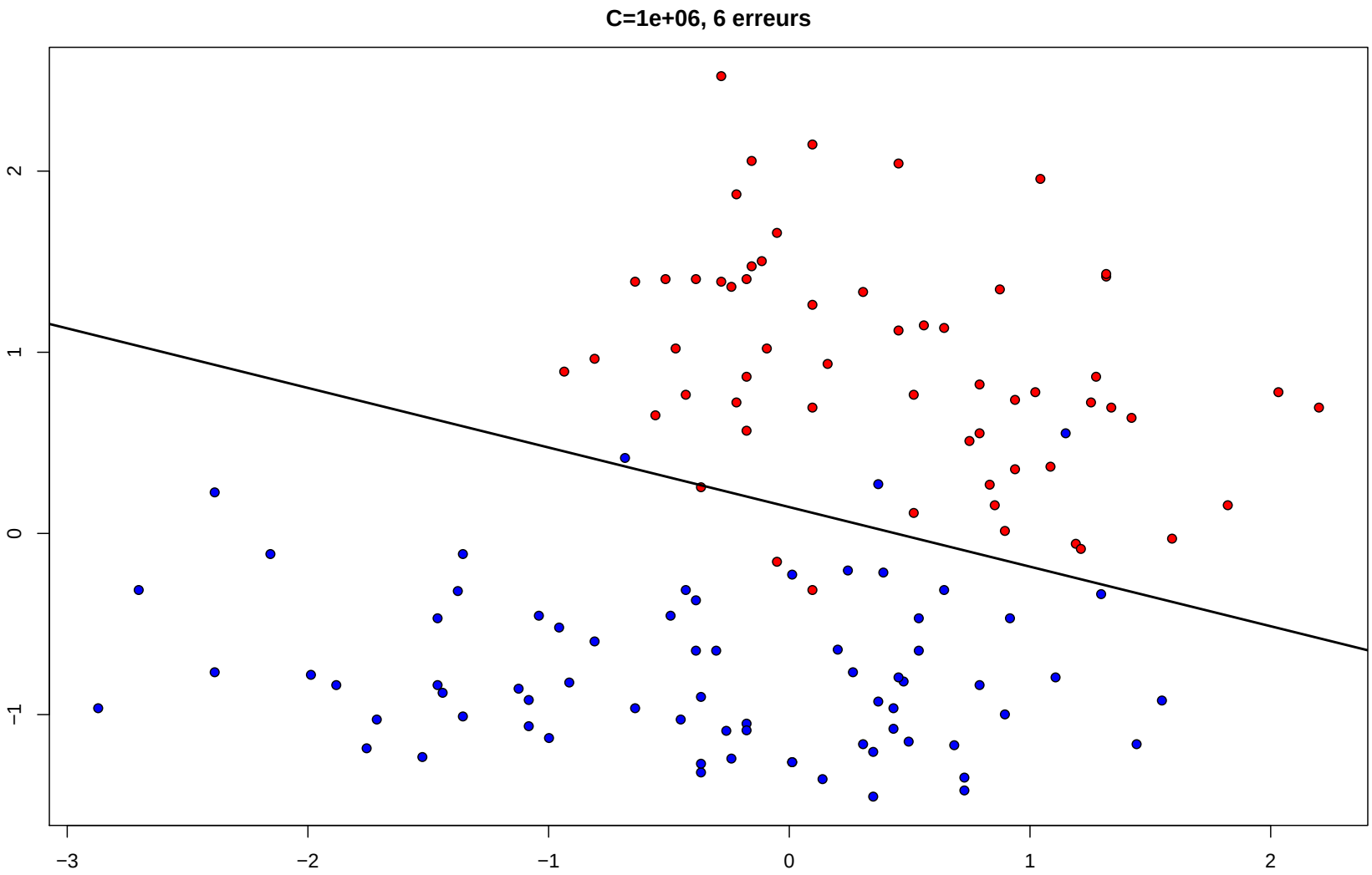
Exemple



Exemple



Exemple



Machine à vecteurs de support

Résumé

Cas linéaire :

- discrimination à deux classes
- algorithme alternatif à celui du Perceptron pour un neurone MCP
- solution robuste (marge maximale)
- facilement extensible au cas non linéairement séparable
- algorithme d'optimisation relativement complexe
- choix de C délicat

Dans la suite du cours :

- extension au cas non linéaire
- choix de C

Projection linéaire

Autre solution pour le neurone MCP :

- si on remplace la fonction de Heaviside par $T(u) = u$, on obtient un modèle linéaire (affine) de sortie

$$\sum_{k=1}^{n} w_k x_k + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

- réalise une projection de $\mathbb{R}^n$ dans $\mathbb{R}$
- comment optimiser cette projection ?
  - bien regrouper les projetés d'une même classe (variance intra petite)
  - bien éloigner les projetés de classes différences (variance inter grande)

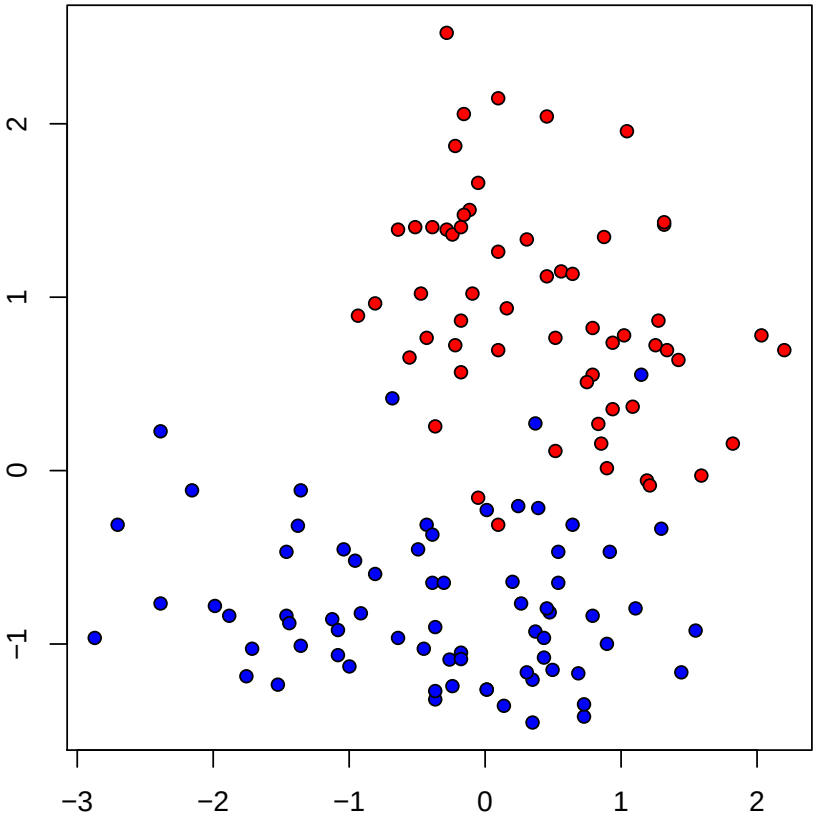
Variances

- on ne considère que la projection $\langle \mathbf{w}, \mathbf{x} \rangle$
- décomposition de la covariance :
  - covariance totale $T = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T$ (μ moyenne des $\mathbf{x}$)
  - covariances intraclasse $W_k = \frac{1}{N_k} \sum_{i \in C_k} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^T$ (μ_k moyenne des $\mathbf{x}$ de la classe C_k)
  - covariance interclasse $B = \frac{1}{N} \sum_{k=1}^{M} N_k (\mu_k - \mu)(\mu_k - \mu)^T$
  - $T = B + W$, avec $W = \frac{1}{N} \sum_{k=1}^{M} N_k W_k$
- projection = « multiplication » par $\mathbf{w}$
  - intraclasse : $\mathbf{w}^T W \mathbf{w}$
  - interclasse : $\mathbf{w}^T B \mathbf{w}$

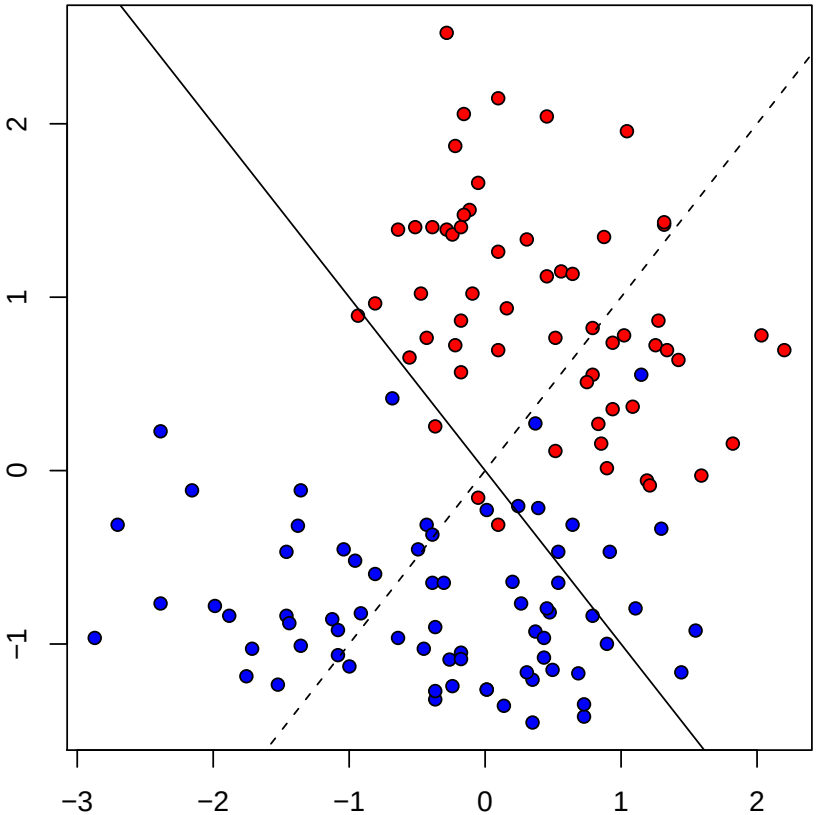
Critère de Fisher

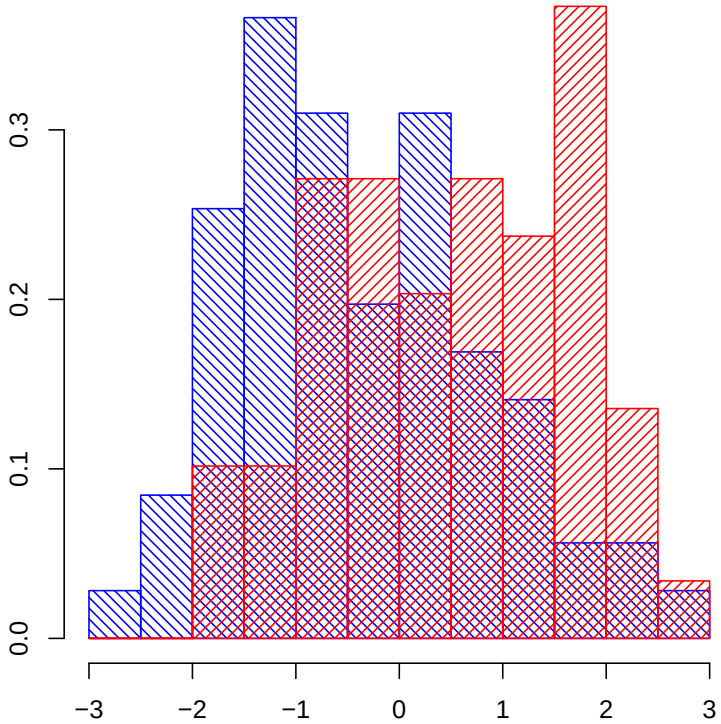
- Critère de Fisher : maximiser $\frac{\mathbf{w}^T B \mathbf{w}}{\mathbf{w}^T W \mathbf{w}}$
- si $\mathbf{w}$ maximise le critère, on montre qu'il existe λ tel que $B\mathbf{w} = \lambda W\mathbf{w}$ (problème de valeur propre généralisé)
- en général W est inversible et $\mathbf{w}$ est donc vecteur propre de $W^{-1}B$ (associé à la plus grande valeur propre)
- algorithme basique (méthode de la puissance itérée) :
  - $\mathbf{w}^{(0)}$ aléatoire
  - $\mathbf{w}^{(t+1)} = \frac{1}{\|W^{-1}B\mathbf{w}^{(t)}\|} W^{-1}B\mathbf{w}^{(t)}$
  - converge vers un vecteur propre associé à la plus grande valeur propre
- puis on ajoute un seuil b optimal (sous une hypothèse de distribution gaussienne)

Exemple

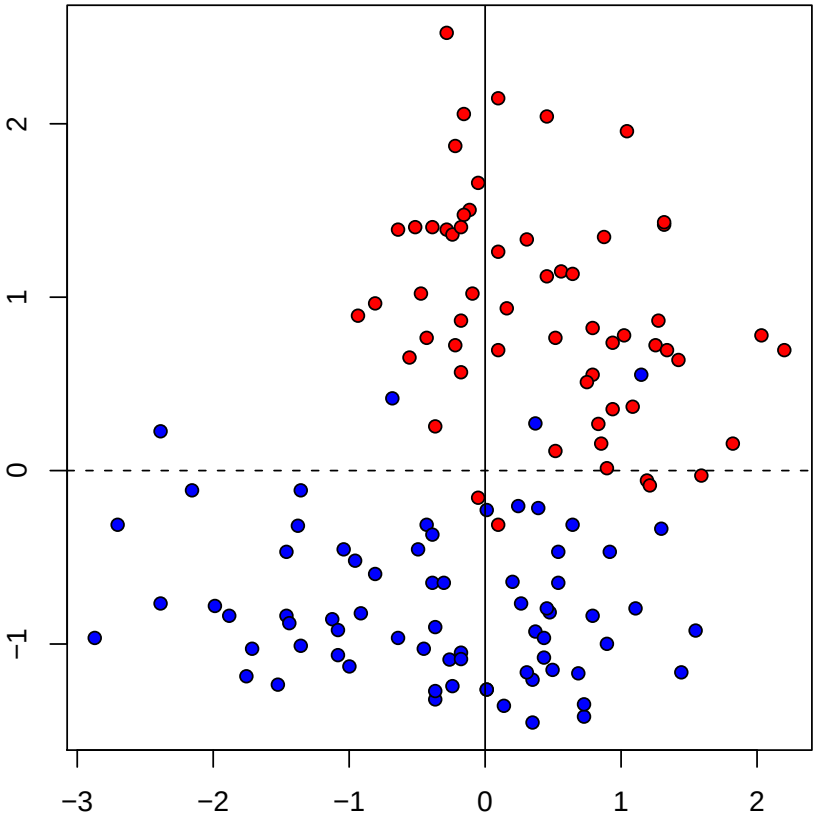


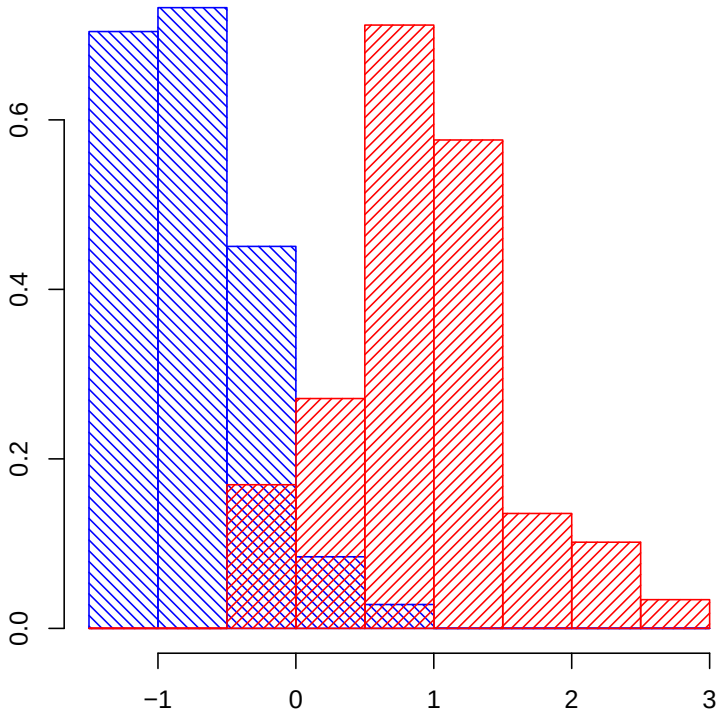
Exemple



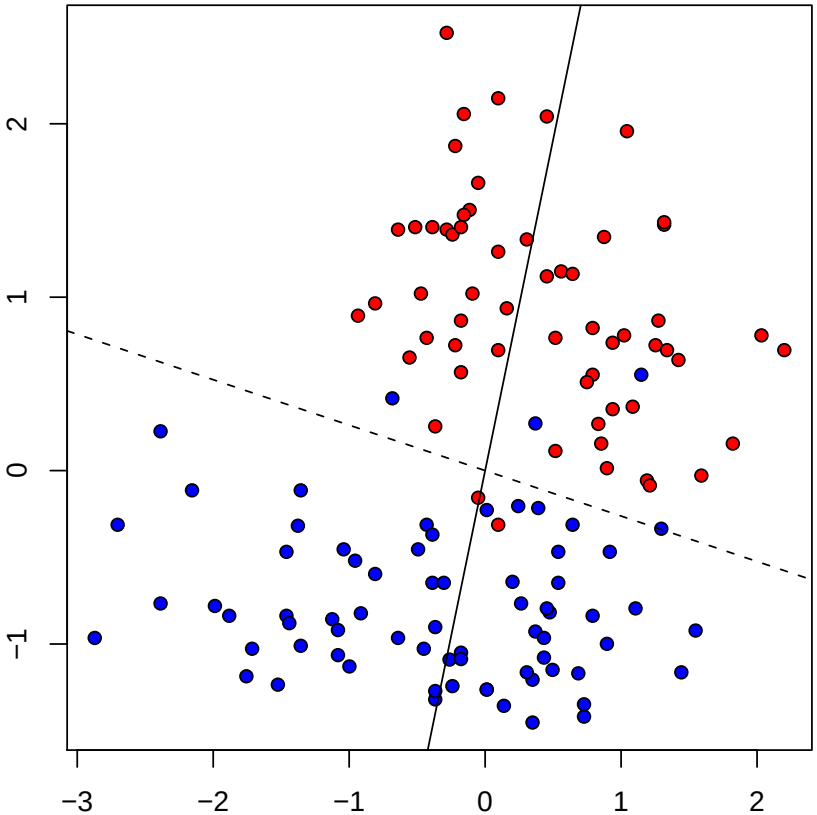


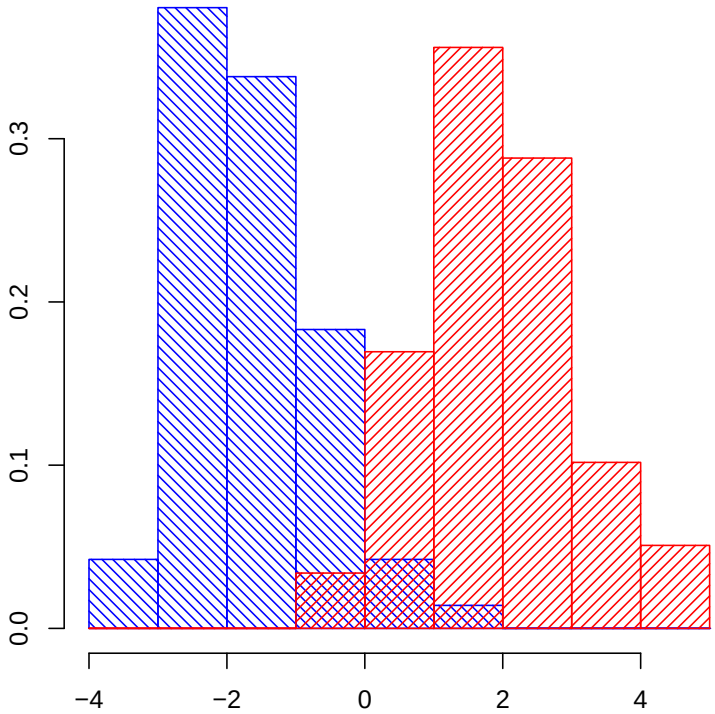
Exemple



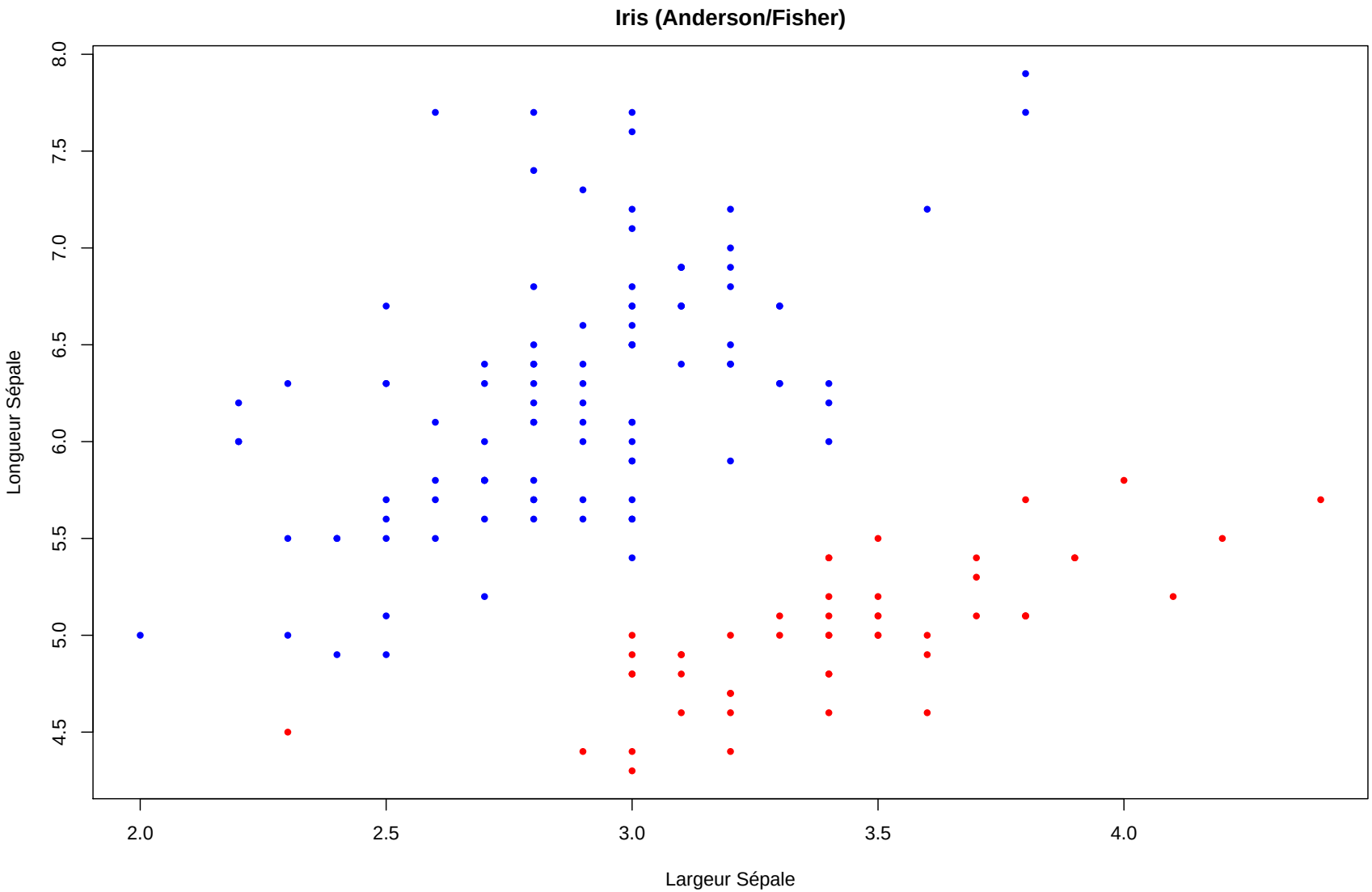


Exemple

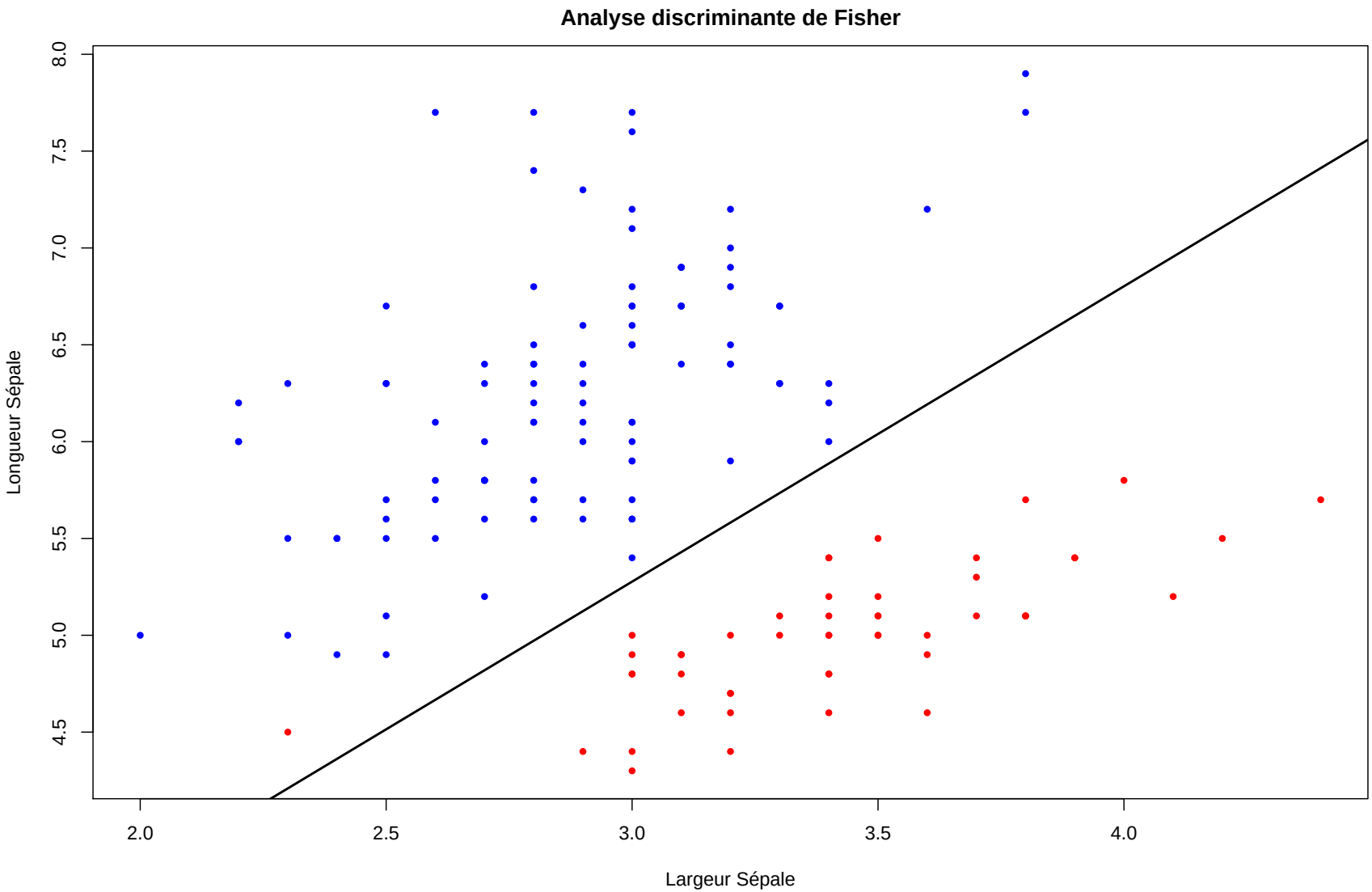




Autre exemple



Autre exemple



Régression linéaire

Solution alternative au critère de Fisher :

- optimisation du critère des moindres carrés :

$$\mathcal{E}(\mathbf{w}, b) = \sum_{i=1}^{N} (y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle + b)^2$$

- régression linéaire classique
- pour deux classes de mêmes effectifs : régression linéaire $\simeq$ analyse discriminante de Fisher

Régression linéaire simple

Cas le plus simple (régression linéaire avec $p = 1$) :

- on dispose de N couples de réels $(x_i, y_i)_{1 \leq i \leq N}$
- on cherche a et b deux réels tels que $y_i \simeq a x_i + b$
- résolution des moindres carrés :
  - on minimise

$$\mathcal{E}(a, b) = \sum_{i=1}^{N} (a x_i + b - y_i)^2$$

  - on a

$$\frac{\partial \mathcal{E}}{\partial b} = 2N \left(b + \frac{1}{N} \sum_{i=1}^{N} (a x_i - y_i) \right)$$

Régression linéaire simple

- on a

$$\frac{\partial \mathcal{E}}{\partial a} = 2\left(a\sum_{i=1}^{N}(x_i)^2 + \sum_{i=1}^{N} x_i(b - y_i)\right)$$

- $\overline{x}$: la moyenne (empirique) des x_i, $\overline{y}$: moyenne des y_i, $\overline{x^2}$: moyenne des $(x_i)^2$ et $\overline{xy}$: moyenne du produit $x_i y_i$
- on trouve :

$$\begin{aligned}\widehat{a} &= \frac{\overline{xy} - \overline{x}.\overline{y}}{\overline{x^2} - (\overline{x})^2} \\ \widehat{b} &= \frac{\overline{x^2}.\overline{y} - \overline{x}.\overline{xy}}{\overline{x^2} - (\overline{x})^2}\end{aligned}$$

Cas général

Même principe :

- on pose $\langle \mathbf{w}, \mathbf{x}_i \rangle + b = \langle \mathbf{v}, \mathbf{z} \rangle$, avec $\mathbf{z} = (\mathbf{x}, 1)$ et $\mathbf{v} = (\mathbf{w}, b)$
- $\mathcal{E} = \sum_{i=1}^{N} \left(\langle \mathbf{v}, \mathbf{z}_i \rangle - y_i \right)^2$
- la différentielle de $\mathcal{E}$ est donnée par

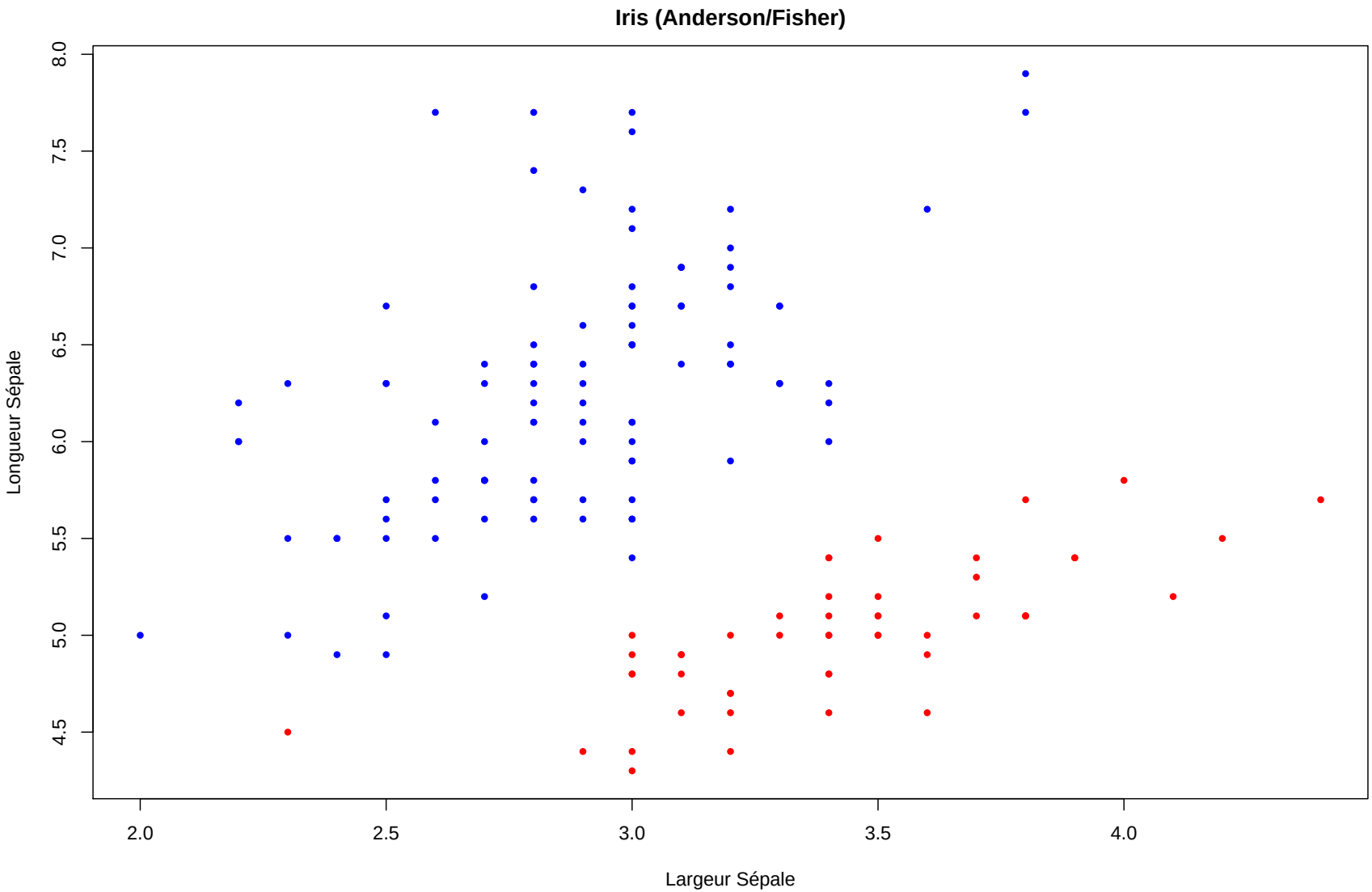
$$\mathrm{d}\mathcal{E} = 2 \sum_{i=1}^{N} \left(\langle \mathbf{v}, \mathbf{z}_i \rangle - y_i \right) \mathbf{z}_i = 2(\mathbf{v}\mathbf{Z} - \mathbf{Y})\mathbf{Z}^T,$$

  avec $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_N)$ et $\mathbf{Y} = (y_1, \ldots, y_N)$
- $\mathrm{d}\mathcal{E} = 0$ est donc équivalent à

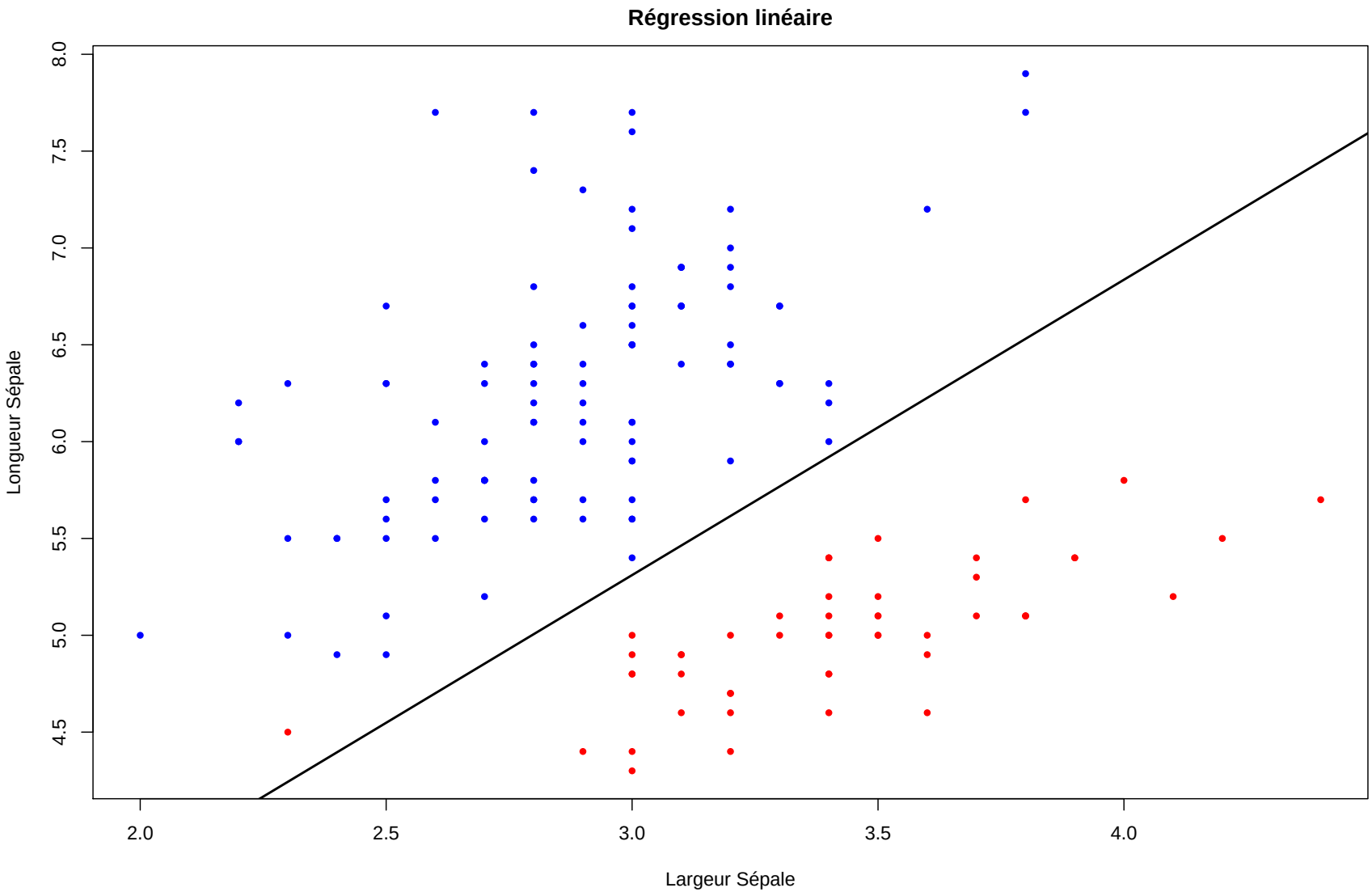
$$\mathbf{Z}\mathbf{Z}^T\mathbf{v}^T = \mathbf{Z}\mathbf{Y}^T$$

- résolution par (pseudo)-inversion de $\mathbf{Z}\mathbf{Z}^T$

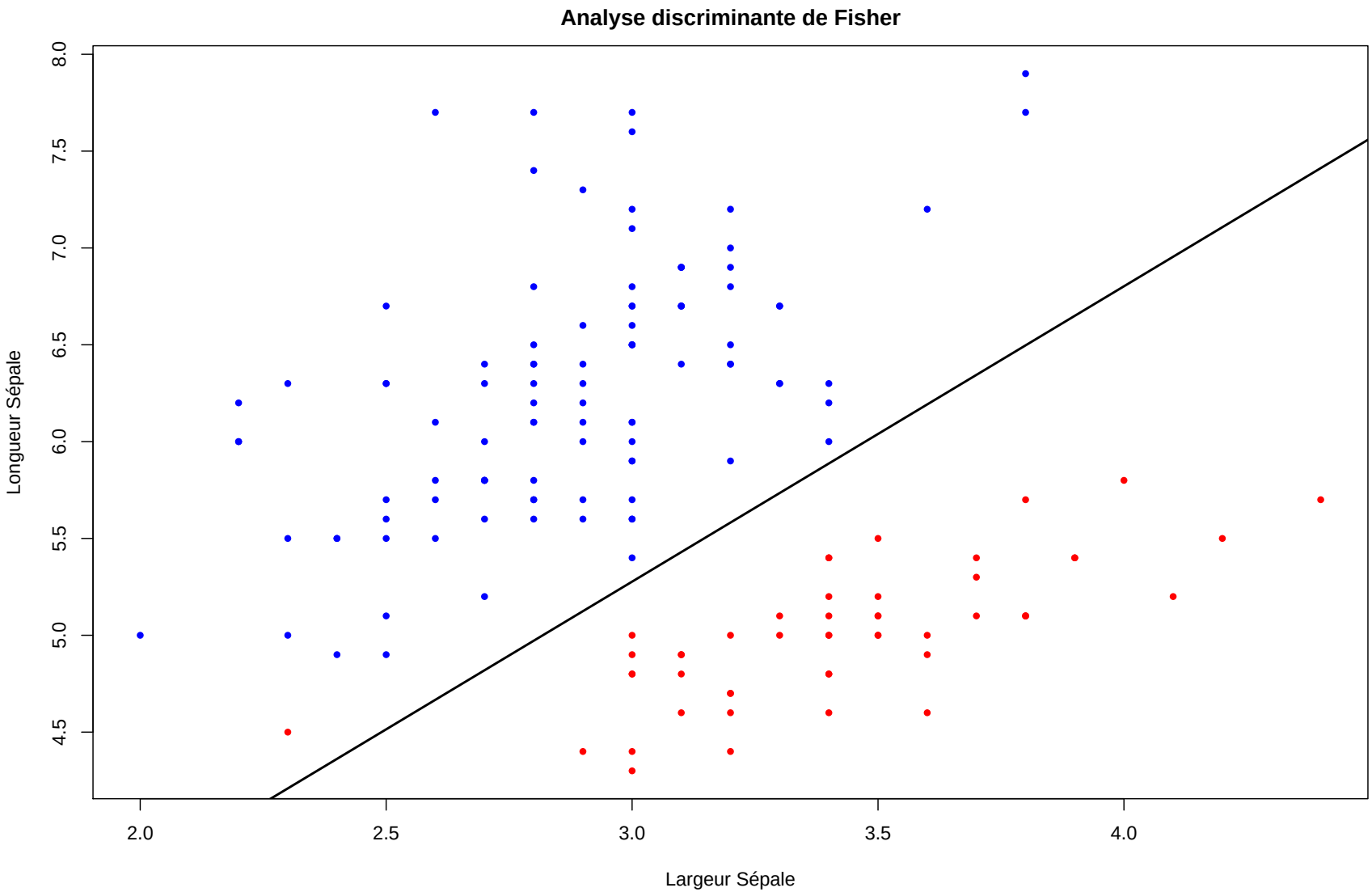
Exemple


Iris (Anderson/Fisher)
Longueur Sépale
Largeur Sépale
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
2.0
2.5
3.0
3.5
4.0

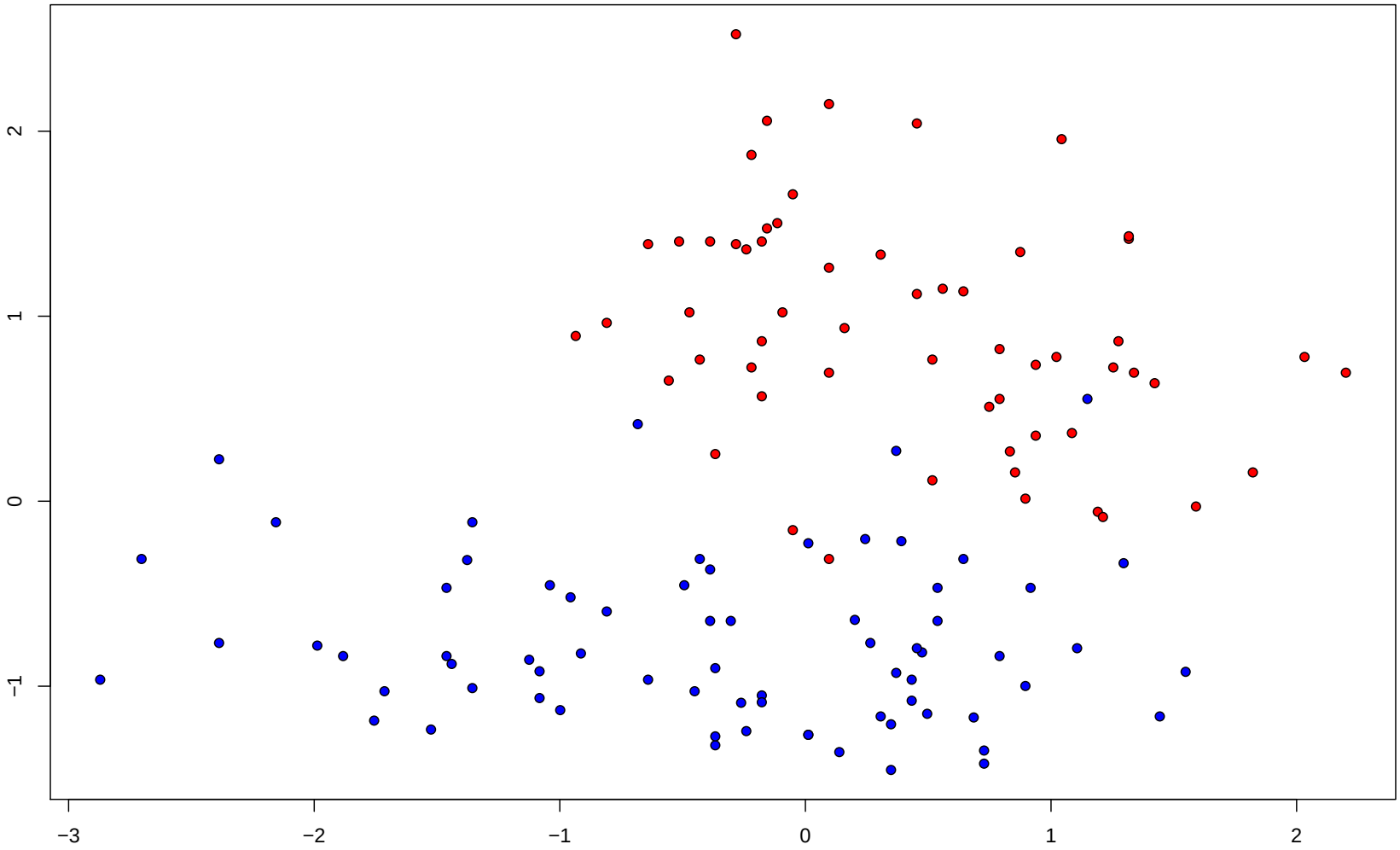
Exemple


Régression linéaire
Longueur Sépale
Largeur Sépale
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
2.0
2.5
3.0
3.5
4.0

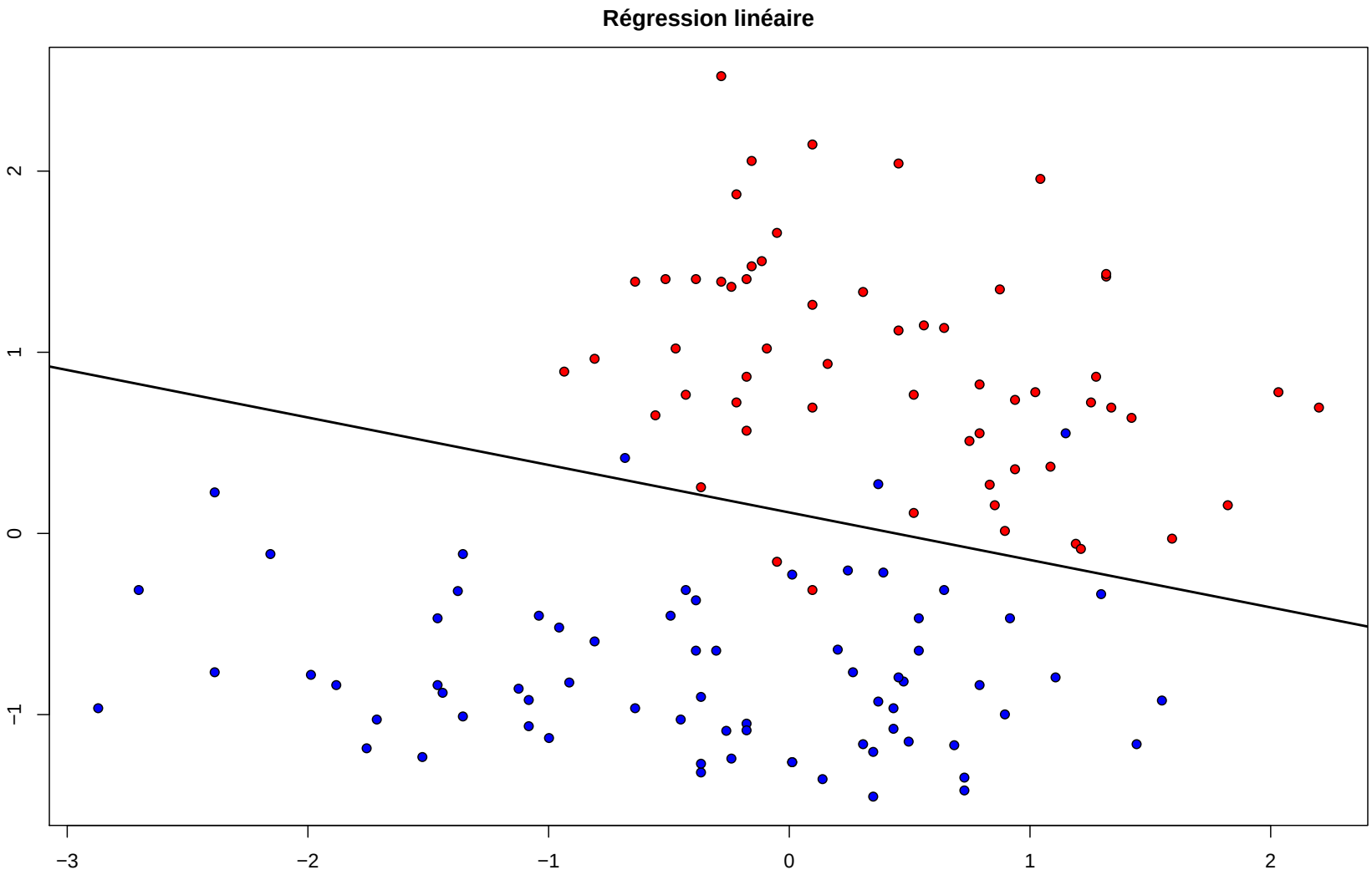
Exemple


Analyse discriminante de Fisher
Longueur Sépale
Largeur Sépale
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
2.0
2.5
3.0
3.5
4.0

Autre exemple



Autre exemple



Méthodes quadratiques

Résumé

Régression linéaire et analyse de Fisher :

- discrimination à deux classes
- algorithmes alternatifs à ceux du Perceptron et des MVS pour un neurone MCP
- facile à mettre en œuvre
- s'appliquent au cas non linéairement séparable
- ne trouvent pas la solution sans erreur si elle existe
- extension simple à plus de deux classes
- extension simple à la régression

Dans la suite du cours :

- régression

Méthodes linéaires

Résumé

- un modèle neuronal unique : le neurone de McCulloch et Pitts
- deux fonctions d'activation : Heaviside ou identité
- quatre algorithmes :
  1. Perceptron (linéairement séparable seulement)
  2. Machine à vecteurs de support (marge maximale)
  3. Analyse de Fisher (séparation des classes)
  4. Régression (moindres carrés)
- quelques problèmes restants :
  - problèmes non linéaires
  - plus de deux classes
  - choix de l'algorithme