

# Signaux, Sons et Images pour l'Informaticien: Regression logistique

Diane Lingrand

Polytech SI3

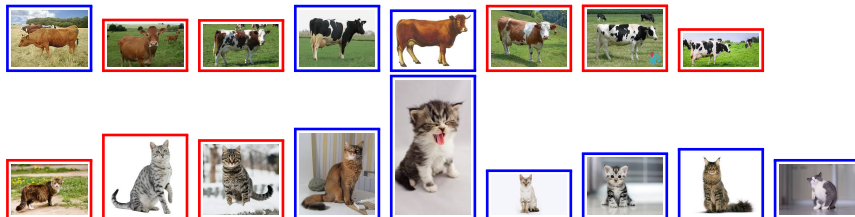
2017 - 2018

- 1 Retour sur la classification avec 2 *kmeans* : exemple des vaches et chats
- 2 Régression logistique
- 3 Retour sur le TP de classification

# Ensemble d'apprentissage : classification par kmean : $k_1 = 50$ , $k_2 = 2$ . 40 sur 63



# Ensemble de test : 6 reconnues sur 17



# Classification vaches / chats



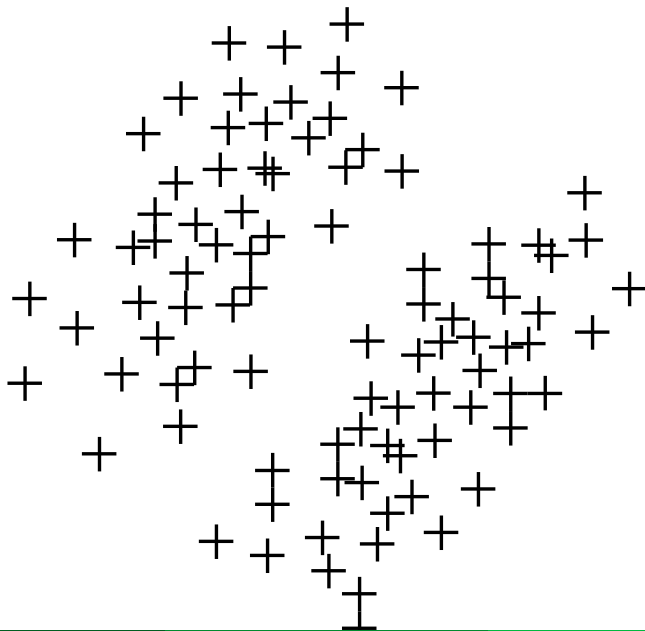
- Echec de la classification par 2 kmeans :
  - ensemble d'apprentissage : 40 / 63 Matrice de confusion :

		classes estimées	
		vache	chat
classes réelles	vache	21	10
	chat	13	19

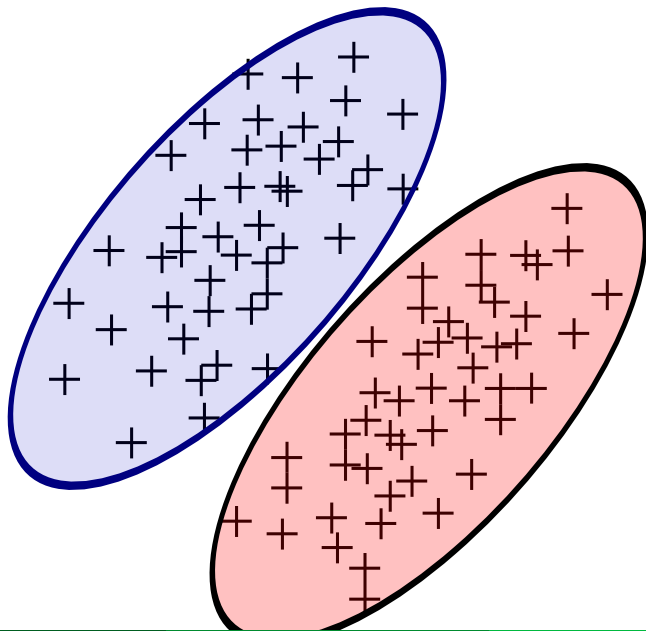
- ensemble de test : 6 / 17 Matrice de confusion :

		classes estimées	
		vache	chat
classes réelles	vache	3	5
	chat	6	3

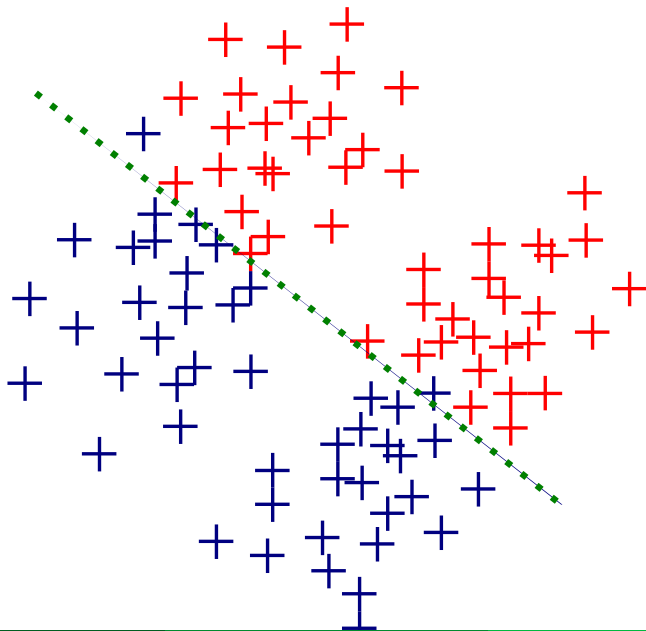
# Retour sur le *kmean* : points dans le plan



# Séparation en deux ensembles par *kmean*



# Utilisation des classes des points : régression logistique





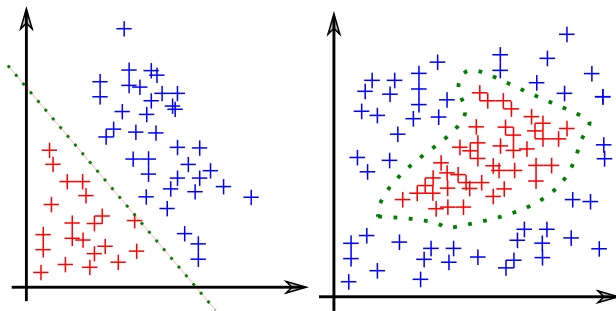
- 1 Retour sur la classification avec 2 *kmeans* : exemple des vaches et chats
- 2 Régresssion logistique
- 3 Retour sur le TP de classification

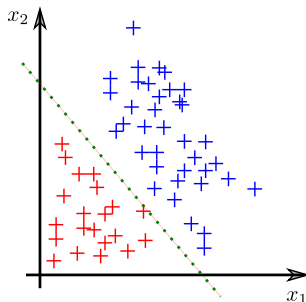
- Apprentissage supervisé : on apprend à partir de données et de leurs classes
- Apprentissage non supervisé : on apprend une représentation des données

Pour l'ensemble d'apprentissage, on connaît la solution : il faut l'utiliser !

# Régression logistique (*logistic regression*)

- Séparation de données selon leurs labels (0 ou 1).
  - Séparation linéaire : droite ou hyperplan
  - Séparation non-linéaire : polynomiale ou gaussienne
- Notations :
  - données :  $\mathbf{x} = [x_1 \ x_2 \dots]$
  - labels :  $y \in \{0, 1\}$
  - critère de décision  $h_\theta$  de paramètre  $\theta$ 
    - $\theta = [\theta_0 \ \theta_1 \ \dots]$



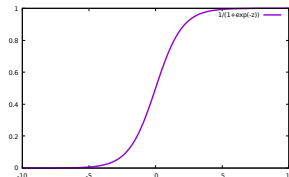


- Frontière de décision :
  - droite d'équation  $\theta_0 + \theta_1 x_1 + \theta_2 x_2$
  - s'écrit aussi :  $\theta^T \mathbf{x} = 0$
- Décision :
  - si  $\theta^T \mathbf{x} \geq 0$  alors  $y = 1$
  - si  $\theta^T \mathbf{x} < 0$  alors  $y = 0$

$$h_{\theta}(\mathbf{x}) = s(\theta^T \mathbf{x})$$

avec :

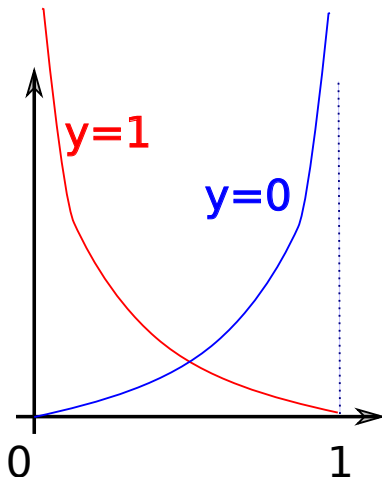
$$s(z) = \frac{1}{1 + e^{-z}}$$



- La décision devient :
  - si  $h_{\theta}(\mathbf{x}) \geq 0.5$  alors  $y = 1$
  - si  $h_{\theta}(\mathbf{x}) < 0.5$  alors  $y = 0$

- données d'apprentissage :  $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})$
- $m$  données d'apprentissage
- but de l'apprentissage : trouver  $\theta$
- méthode :
  - minimisation d'une erreur
  - descente de gradient (ou autre méthode de minimisation)

$$J = -\frac{1}{m} \sum_{i=1}^m (y^{(i)} \log(h_{\theta}(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(\mathbf{x}^{(i)})))$$



Pour toutes les composantes  $\theta_j$  de  $\theta$  :

$$\theta_j = \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$$

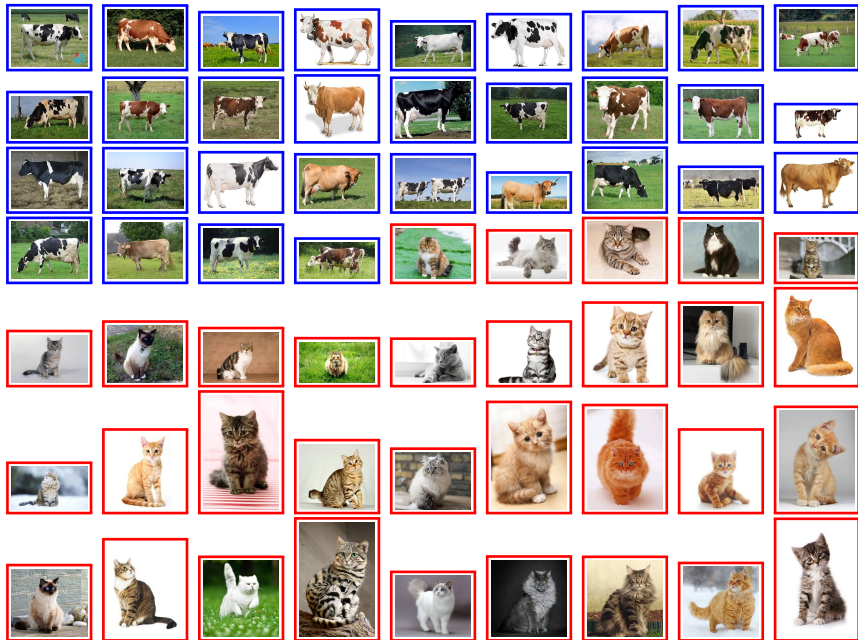
avec

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

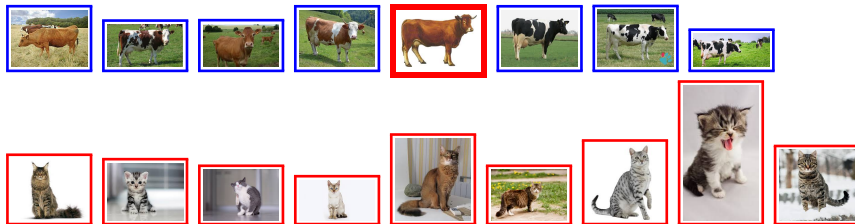


- 1 Retour sur la classification avec 2 *kmeans* : exemple des vaches et chats
- 2 Régression logistique
- 3 Retour sur le TP de classification

# Ensemble d'apprentissage, $k_1 = 50$ : score = 100%



# Ensemble de test : score 94%



Matrice de confusion :

		classes estimées	
		vache	chat
classes réelles	vache	7	1
	chat	0	8

Pour la base d'apprentissage :

- représentation de chaque image par un *bow*
- liste de tous les *bow* et de tous les labels
  - correspondances entre *bow* et label (0 ou 1)
- calcul du score sur la base d'apprentissage
- sauvegarde des paramètres du *kmean* et de la régression logistique

Pour la base de test :

- chargement des paramètres appris pour le *kmean* et la régression logistique
- calcul des bows pour chaque image
- prédiction des labels de chaque image et comparaison avec les vrais labels (calcul de score)

# Logistic regression en python

```
#création d'un objet de regression logistique
logisticRegr = LogisticRegression()
#apprentissage
logisticRegr.fit(bows, labels)
#calcul des labels prédits
labelsPredicted = logisticRegr.predict(bows)
#calcul et affichage du score
score = logisticRegr.score(bows, labels)
print("train score = ", score)
#sauvegarde de l'objet
with open('sauvegarde.logr', 'wb') as output:
    pickle.dump(logisticRegr, output, pickle.HIGHEST_PROTOCOL)
#chargement de l'objet
with open('sauvegarde.logr', 'rb') as input:
    logisticRegr = pickle.load(input)
```

# Une approche classique

