Théorie et Algorithmes de l'Apprentissage Automatique

3 - Classification linéaire

Simon BERNARD
simon.bernard@univ-rouen.fr

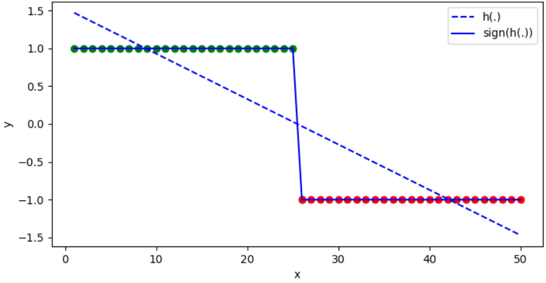
Introduction

- Modèle $h : \mathbb{R}^d \to \mathbb{R}$ de la forme

$$h(\mathsf{x}) = \sum_{i=1}^{d} w_i x^{(i)} + b = \mathsf{x}^\top \mathsf{w} + b = [\mathsf{x}^\top 1]\boldsymbol{\alpha}$$

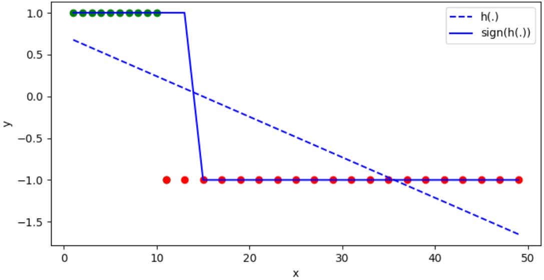
  avec
  - $\mathsf{w} \in \mathbb{R}^d$, un vecteur qui définit un hyperplan
  - $b \in \mathbb{R}$ un biais qui déplace la fonction perpendiculairement à l'hyperplan
  - $\boldsymbol{\alpha} = \begin{bmatrix} \mathsf{w} \\ b \end{bmatrix} \in \mathbb{R}^{d+1}$
- Régression linéaire : $h(\mathsf{x}) = \hat{y} \in \mathbb{R}$
- Classification linéaire : $h(\mathsf{x}) = ?$

- Classification binaire avec les classes $\lambda_1 = 1$ et $\lambda_2 = -1$
- Apprendre un modèle $h : \mathbb{R}^d \to \{-1, 1\}$
- Moindres carrés ordinaires + la classe prédite est donnée par le signe de $h(.)$



Plusieurs défauts

- Pas de transposition naturelle à + de 2 classes
- Les résidus n'ont pas de sens
- Le résultat est très sensible aux données d'apprentissage



- Alternative : Prédire $p(\lambda_i|\mathbf{x})$, la probabilité d'appartenance à la classe λ_i (probabilité *a posteriori* de λ_i)
- Par exemple, si on se limite toujours à deux classes :

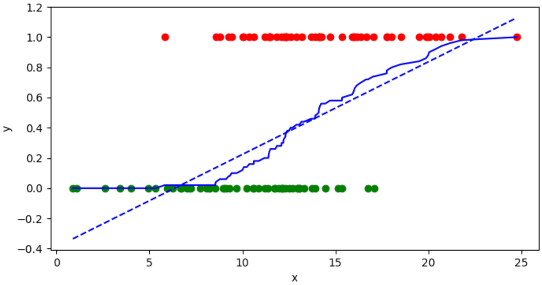
$$h(\mathbf{x}) = p(\lambda_1|\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b$$

  et

$$p(\lambda_2|\mathbf{x}) = 1 - p(\lambda_1|\mathbf{x}) = 1 - h(\mathbf{x})$$

- $\forall \mathbf{x} \in \mathcal{X}$, on prédit λ_1 si $p(\lambda_1|\mathbf{x}) \geq p(\lambda_2|\mathbf{x})$ (i.e. si $p(\lambda_1|\mathbf{x}) \geq 0,5$), λ_2 sinon

Problème : $h(\mathrm{x}) \notin [0,1]$: modèle linéaire non adapté pour modéliser une probabilité



Analyse Discriminante Linéaire

- L'analyse discriminante linéaire (LDA) s'appuit sur la règle de Bayes :

$$p(\lambda_i|\mathbf{x}) = \frac{p(\mathbf{x}|\lambda_i)p(\lambda_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\lambda_i)p(\lambda_i)}{\sum_{j=1}^{c} p(\mathbf{x}|\lambda_j)p(\lambda_j)}$$

  où

  - $p(\lambda_i)$ est la probabilité *a priori* de la classe λ_i
  - $p(\mathbf{x}|\lambda_i)$ est la vraisemblance de λ_i étant donné $\mathbf{x}$
- L'enjeu est d'estimer $p(\mathbf{x}|\lambda_i)$ pour déduire $p(\lambda_i|\mathbf{x})$
- Prédiction obtenue par :

$$\hat{y} = \arg\max_i p(\lambda_i|\mathbf{x}) = \arg\max_i p(\mathbf{x}|\lambda_i)p(\lambda_i)$$

- LDA suppose que les $\mathbf{x}$ sont issues de distributions gaussiennes $\mathcal{N}(\mu_i, \Sigma), \forall i$
- Les classes partagent la même matrice de covariance Σ (homoscédasticité)
- C'est ce qui permet d'aboutir à un modèle linéaire (sinon modèle quadratique)
- Probabilités conditionnelles :

$$p(\mathbf{x}|\lambda_i) = \det(2\pi\Sigma)^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_i)^\top \Sigma^{-1}(\mathbf{x} - \mu_i)\right)$$

- Les valeurs μ_i, Σ et $p_i = p(\lambda_i)$ sont estimées sur les données d'apprentissage

- Pour 2 classes ($\lambda_1 = 1$ et $\lambda_2 = -1$) la décision est donnée par :

$$p(\lambda_1|\mathsf{x}) \underset{\lambda_2}{\overset{\lambda_1}{\gtrless}} p(\lambda_2|\mathsf{x}) \quad \Leftrightarrow \quad p(\mathsf{x}|\lambda_1)p(\lambda_1) \underset{\lambda_2}{\overset{\lambda_1}{\gtrless}} p(\mathsf{x}|\lambda_2)p(\lambda_2)$$

- Fonction de décision basée sur le rapport de vraisemblances :

$$h(\mathsf{x}) = \log\left(\frac{p(\lambda_1|\mathsf{x})}{p(\lambda_2|\mathsf{x})}\right) = \log\left(\frac{p(\mathsf{x}|\lambda_1)p(\lambda_1)}{p(\mathsf{x}|\lambda_2)p(\lambda_2)}\right)$$

- $h(\mathsf{x})$ est positif si $p(\lambda_1|\mathsf{x}) > p(\lambda_2|\mathsf{x})$, négatif sinon

$$
\begin{aligned}
h(\mathbf{x}) &= \log\left(\frac{p(\lambda_1|\mathbf{x})}{p(\lambda_2|\mathbf{x})}\right) = \log\left(\frac{p(\mathbf{x}|\lambda_1)p(\lambda_1)}{p(\mathbf{x}|\lambda_2)p(\lambda_2)}\right)\\
&= \log\left(\frac{\det(2\pi\Sigma)^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu_1)^\top\Sigma^{-1}(\mathbf{x}-\mu_1)\right)p(\lambda_1)}{\det(2\pi\Sigma)^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu_2)^\top\Sigma^{-1}(\mathbf{x}-\mu_2)\right)p(\lambda_2)}\right)\\
&= -\frac{1}{2}(\mathbf{x}-\mu_1)^\top\Sigma^{-1}(\mathbf{x}-\mu_1) + \frac{1}{2}(\mathbf{x}-\mu_2)^\top\Sigma^{-1}(\mathbf{x}-\mu_2) + \log(p(\lambda_1)) - \log(p(\lambda_2))\\
&= \mathbf{x}^\top\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_1^\top\Sigma^{-1}\mu_1 - \mathbf{x}^\top\Sigma^{-1}\mu_2 + \frac{1}{2}\mu_2^\top\Sigma^{-1}\mu_2 + \log(p(\lambda_1)) - \log(p(\lambda_2))\\
&= \mathbf{x}^\top\Sigma^{-1}(\mu_1-\mu_2) + \frac{1}{2}(\mu_1+\mu_2)^\top\Sigma^{-1}(\mu_1-\mu_2) + \log(p(\lambda_1)) - \log(p(\lambda_2))\\
&= \mathbf{x}^\top\mathbf{w} + b
\end{aligned}
$$

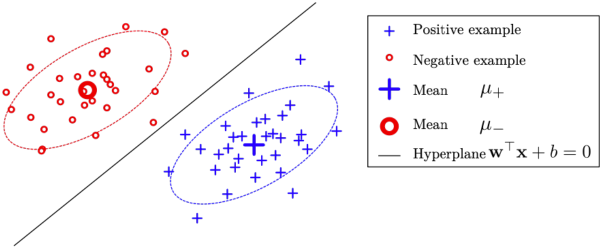
avec

$$
\mathbf{w} = \Sigma^{-1}(\mu_1-\mu_2) \qquad b = \frac{1}{2}\mathbf{w}^\top(\mu_1+\mu_2) + \log(p(\lambda_1)) - \log(p(\lambda_2))
$$

- Le modèle est un hyperplan séparateur ($\neq$ estimateur de $p(\lambda_1|\mathbf{x})$)

$$h(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b, \quad s.t.\ \mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_2), b = \frac{1}{2}\mathbf{w}^\top(\mu_1 + \mu_2) + \log(p(\lambda_1)) - \log(p(\lambda_2))$$

- Σ, μ_1, μ_2, $p(\lambda_1)$ et $p(\lambda_2)$ à estimer sur les données d'apprentissage



- Version régularisée :

$$\mathbf{w} = (\Sigma + \lambda \mathbf{I})^{-1}(\mu_1 - \mu_2), \quad b = \frac{1}{2}\mathbf{w}^\top(\mu_1 + \mu_2) + \log(p(\lambda_1)) - \log(p(\lambda_2))$$

  avec $\mathbf{I}$ la matrice identité et λ un paramètre de régularisation.
- Extensions aux cas multi-classes ($C > 2$) :
  - Approche *one-vs-all* ou *one-vs-one*
  - *Fisher Discriminant Analysis* : maximise la variance inter-classe et minimise la variance intra-classe
- LDA ne modélise pas directement $p(\lambda_i|\mathbf{x})$
- L'hypothèse de distributions gaussiennes et d'homoscédasticité est forte

Régression logistique

Objectifs

- Modèle linéaire
- Modéliser directement les $p(\lambda_i|\mathbf{x})$
- Pas d'hypothèse sur la distribution des données

Approche

- Modèle linéaire généralisé basé sur la fonction *logit*
- Maximiser la vraisemblance
- Résolution itérative (descente de gradient)

- Modèle linéaire généralisé de la forme :

$$g(y_i) = w_1 x_i^{(1)} + w_2 x_i^{(2)} + \cdots + w_d x_i^{(d)} + b$$

  où g est appelé fonction de lien
- Ce modèle est linéaire en ses coefficients mais il peut modéliser des relations non-linéaires grâce à g
- La régression logistique est un cas particulier de régression linéaire généralisée où g est judicieusement choisi pour modéliser $p(\lambda_1|\mathbf{x})$

- Le modèle de régression logistique s'écrit :

$$logit\,(p(\lambda_1|\mathbf{x})) = w_1 x_i^{(1)} + w_2 x_i^{(2)} + \cdots + w_d x_i^{(d)} + b$$

- La fonction de lien est la fonction *logit* :

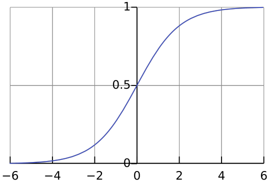
$$logit(p) = \ln\left(\frac{p}{1-p}\right)$$

- Donc

$$logit\,(p(\lambda_1|\mathbf{x})) = \ln\left(\frac{p(\lambda_1|\mathbf{x})}{1-p(\lambda_1|\mathbf{x})}\right) = \ln\left(\frac{p(\lambda_1|\mathbf{x})}{p(\lambda_2|\mathbf{x})}\right) = w_1 x_i^{(1)} + w_2 x_i^{(2)} + \cdots + w_d x_i^{(d)} + b$$

- Nous voulons modéliser $p(\lambda_1|\mathbf{x})$ et non *logit* $(p(\lambda_1|\mathbf{x}))$
- Il faut utiliser la réciproque de *logit*, appelée fonction logistique ou sigmoïde :

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



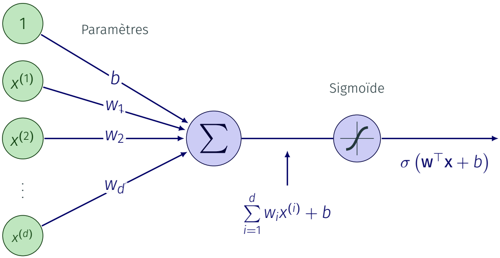
- Donc le modèle de régression logistique à estimer est :

$$p(\lambda_1|\mathbf{x}) = \sigma\left(w_1 x_i^{(1)} + w_2 x_i^{(2)} + \cdots + w_d x_i^{(d)} + b\right)$$

- Le modèle de régression logistique s'écrit :

$$h(\mathbf{x}) = \sigma\left(\mathbf{w}^\top \mathbf{x} + b\right) = \frac{1}{1 + \exp(-(\mathbf{w}^\top \mathbf{x} + b))} = \frac{\exp(\mathbf{w}^\top \mathbf{x} + b)}{1 + \exp(\mathbf{w}^\top \mathbf{x} + b)}$$

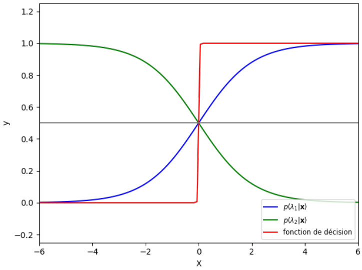
où $h(\mathbf{x})$ est une fonction de prédiction de $p(\lambda_1|\mathbf{x})$ (et non pas y)



- On peut prendre une décision si $p(\lambda_1|\mathbf{x}) > p(\lambda_2|\mathbf{x})$ ou l'inverse
- La frontière de décision est

$$p(\lambda_1|\mathbf{x}) = p(\lambda_2|\mathbf{x}) = 0.5 = \sigma\left(\mathbf{w}^\top \mathbf{x} + b\right)$$

soit quand $\mathbf{w}^\top \mathbf{x} + b = 0$



Maximum de vraisemblance

- On veut estimer $\boldsymbol{\alpha}$ [1] qui maximise la probabilité des y_i conditionnellement aux $\mathbf{x}_i$:

$$p(y_1, y_2, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n)$$

- La fonction à maximiser est appelée fonction de vraisemblance :

$$L(\boldsymbol{\alpha}) = p(y_1, y_2, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n; \boldsymbol{\alpha}) = \prod_{i=1}^{n} p(y_i | \mathbf{x}_i; \boldsymbol{\alpha})$$

où $p(y_i|\mathbf{x}_i; \boldsymbol{\alpha})$ peut s'interpréter comme la probabilité d'obtenir la vraie classe y_i pour $\mathbf{x}_i$ avec le modèle de paramètre $\boldsymbol{\alpha}$

1. On rappelle que $\boldsymbol{\alpha} = [\mathbf{w}^\top b]^\top$ auquel cas on considère que $\mathbf{x}_i$ est "augmenté"

- La meilleure estimation de $\boldsymbol{\alpha}$ est celle qui maximise la vraisemblance

$$\max_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}} \prod_{i=1}^{n} p(y_i|\mathbf{x}_i;\boldsymbol{\alpha})$$

- Pour simplifier les calculs, on préfère minimiser la log-vraisemblance :

$$\begin{aligned} LL(\boldsymbol{\alpha}) &= -\log L(\boldsymbol{\alpha}) \\ &= -\sum_{i=1}^{n} \log(p(y_i|\mathbf{x}_i;\boldsymbol{\alpha})) \end{aligned}$$

le $\boldsymbol{\alpha}$ qui minimise la log-vraisemblance est celui qui maximise la vraisemblance.

- Modèle de regression logistique :

$$p(\lambda_1|\mathbf{x}) = \frac{\exp(\mathbf{w}^\top \mathbf{x} + b)}{1 + \exp(\mathbf{w}^\top \mathbf{x} + b)} \qquad p(\lambda_2|\mathbf{x}) = \frac{1}{1 + \exp(\mathbf{w}^\top \mathbf{x} + b)}$$

- Donc :

$$\begin{aligned}
LL(\boldsymbol{\alpha}) &= -\sum_{i=1}^{n} \log(p(y_i|\mathbf{x}_i;\boldsymbol{\alpha})) \\
&= -\sum_{i \in \mathcal{I}_1} \log(p(\lambda_1|\mathbf{x}_i;\boldsymbol{\alpha})) - \sum_{i \in \mathcal{I}_2} \log(p(\lambda_2|\mathbf{x}_i;\boldsymbol{\alpha})) \\
&= -\sum_{i \in \mathcal{I}_1} \log\left(\frac{\exp(\mathbf{w}^\top \mathbf{x} + b)}{1 + \exp(\mathbf{w}^\top \mathbf{x} + b)}\right) - \sum_{i \in \mathcal{I}_2} \log\left(\frac{1}{1 + \exp(\mathbf{w}^\top \mathbf{x} + b)}\right)
\end{aligned}$$

où $\mathcal{I}_1$ (resp. $\mathcal{I}_2$) est l'ensemble des i tels que $y_i = \lambda_1$ (resp. $y_i = \lambda_2$)

- En posant $\mathbf{z} = \mathbf{w}^\top \mathbf{x} + b$, $\lambda_1 = 1$ et $\lambda_2 = 0$, on peut écrire[2] :

$$
\begin{aligned}
LL(\boldsymbol{\alpha}) &= -\sum_{i=1}^{n} y_i \log\left(\frac{\exp(\mathbf{z}_i)}{1+\exp(\mathbf{z}_i)}\right) - \sum_{i=1}^{n}(1-y_i)\log\left(\frac{1}{1+\exp(\mathbf{z}_i)}\right) \\
&= -\sum_{i=1}^{n} y_i \left(\log(\exp(\mathbf{z}_i)) - \log(1+\exp(\mathbf{z}_i))\right) + \sum_{i=1}^{n}(1-y_i)\log\left(1+\exp(\mathbf{z}_i)\right) \\
&= -\sum_{i=1}^{n} y_i \mathbf{z}_i + \sum_{i=1}^{n} \log(1+\exp(\mathbf{z}_i))
\end{aligned}
$$

2. on peut aboutir à une formulation équivalente en posant $\lambda_1 = 1$ et $\lambda_2 = -1$

- Le problème d'optimisation est donc :

$$\min_{\boldsymbol{\alpha}} \quad -\sum_{i=1}^{n} y_i \mathsf{z}_i + \sum_{i=1}^{n} \log(1+\exp(\mathsf{z}_i))$$

$$\min_{\boldsymbol{\alpha}} \quad -\sum_{i=1}^{n} y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) + \sum_{i=1}^{n} \log\left(1+\exp(\mathbf{w}^\top \mathbf{x}_i + b)\right)$$

- Fonction convexe : il existe une solution unique, celle qui annule le gradient

- Après calcul[3], on trouve le gradient

$$\begin{aligned}\frac{\partial J}{\partial \alpha_k} &= -\sum_{i=1}^{n} x_i^{(k)} \left(y_i - \frac{\exp(\mathbf{z}_i)}{1+\exp(\mathbf{z}_i)} \right) \\ &= -\sum_{i=1}^{n} x_i^{(k)} \left(y_i - \mathbf{p}_i \right)\end{aligned}$$

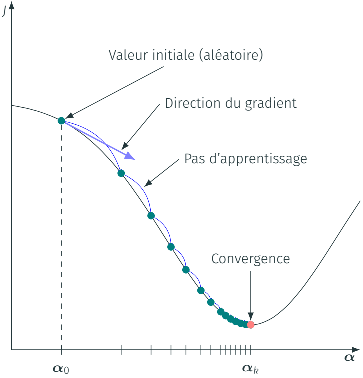
  avec $\mathbf{p}_i = \frac{\exp(\mathbf{z}_i)}{1+\exp(\mathbf{z}_i)}$
- Sous la forme matricielle :

$$\nabla_\alpha J(\boldsymbol{\alpha}) = -\mathbf{X}^\top (\mathbf{y} - \mathbf{p})$$

  où $\mathbf{p}$ est le vecteur des $\mathbf{p}_i = \frac{\exp(-\mathbf{z}_i)}{1+\exp(-\mathbf{z}_i)}$
- $\nabla_\alpha J(\boldsymbol{\alpha}) = 0$ ne permet pas d'obtenir de calcul direct de la solution

3. détail en annexe, à la fin de ce support

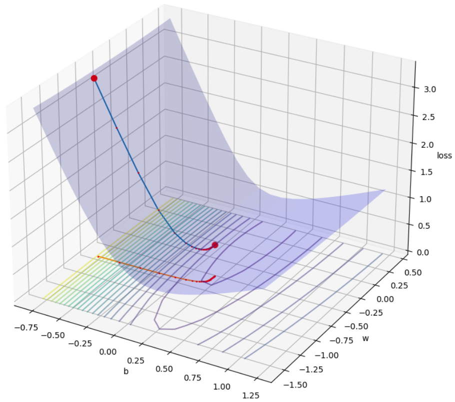
Solution : descente de gradient (cf. cours d'optimisation)



- Initialiser $\boldsymbol{\alpha}^{(0)}$ (e.g. valeurs aléatoires)
- À l'itération t, mise à jour de la solution :

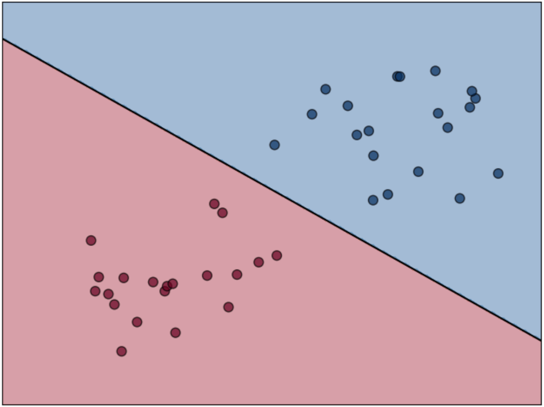
$$\begin{aligned}\boldsymbol{\alpha}^{(t)} &= \boldsymbol{\alpha}^{(t-1)} - \gamma_t \nabla_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}^{(t-1)}) \\ &= \boldsymbol{\alpha}^{(t-1)} - \gamma_t \mathsf{X}^\top (\mathsf{y} - \mathsf{p})\end{aligned}$$

  où $\mathbf{p}$ est calculé avec $\boldsymbol{\alpha}^{(t-1)}$
- Arrêt : convergence ou nombre d'itérations


3.0
2.5
2.0
loss
1.5
1.0
0.5
0.0
0.50
0.25
0.00
−0.25
−0.50
−0.75
w
−1.00
−1.25
−1.50
−0.75
−0.50
−0.25
0.00
0.25
b
0.50
0.75
1.00
1.25

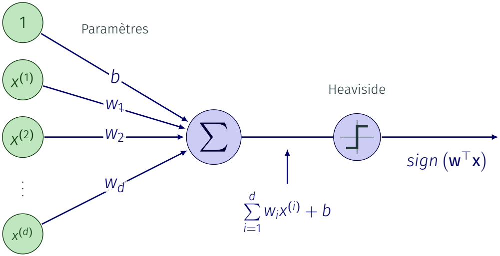
Autres méthodes de classification linéaires

- Classification binaire : $\lambda_1 = 1$ et $\lambda_2 = -1$
- Fonction de décision linéaire $h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ tel que $\hat{y} = sign(h(\mathbf{x}))$



Perceptron

- Le plus simple des réseaux de neurones (et le premier) = séparateur linéaire



Perceptron

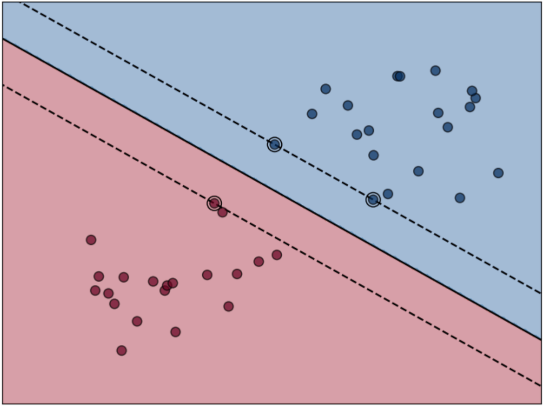
- Apprentissage itératif des $\mathbf{w}_i, \forall i = 0, \dots, n$
- Mise à jour pour corriger les erreurs en apprentissage les unes après les autres
- Revient à minimiser :

$$\begin{aligned} J(\mathbf{w}, b) &= -\sum_{\mathbf{x}_i \in \mathcal{M}} y_i(\mathbf{w}^\top \mathbf{x}_i + b) \\ &= \sum_{\mathbf{x}_i \in \mathcal{M}} \max(0, -y_i(\mathbf{w}^\top \mathbf{x}_i + b)) \quad \text{(hinge loss)} \end{aligned}$$

  où $\mathcal{M}$ est l'ensemble des données d'apprentissage mal classées à l'itération précédente
- Descente de gradient stochastique (gradient calculé pour un seul point à chaque itération)
- Problème si les données ne sont pas linéairement séparables...

Support Vector Machine (SVM)

- Toutes les explications dans le prochain chapitre



Classification multiclasse

- Les modèles précédents ne sont applicables qu'à la classification binaire ($c = 2$)
- Chaque méthode transposable à la classification multiclasse
- Pour la régression logistique : régression logistique multinomiale ou régression *softmax*
- On modélise $p(y = \lambda_k | \mathbf{x})$ avec un modèle qui lui est propre (stratégie *one-vs-all*)
- On utilise pour cela, une fonction *softmax* :

$$p(y = \lambda_k | \mathbf{x}) = \frac{\exp(\mathbf{x}^\top \mathbf{w}_k)}{\sum_{j=1}^{C} \exp(\mathbf{x}^\top \mathbf{w}_j)}$$

où C est le nombre de classes, et $\mathbf{w}_k$ est le vecteur paramètre du modèle pour la classe λ_k.

Plus de détails en TP...

Annexes

- La fonction objective est :

$$J(\mathbf{w}, b) = -\sum_{i=1}^{n} y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) + \sum_{i=1}^{n} \log\left(1 + \exp(\mathbf{w}^\top \mathbf{x}_i + b)\right)$$

- Calcul du gradient :

$$\begin{aligned}
\frac{\partial J}{\partial w_k} &= \frac{\partial}{\partial w_k}\left(-\sum_{i=1}^{n} y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) + \sum_{i=1}^{n} \log\left(1 + \exp(\mathbf{w}^\top \mathbf{x}_i + b)\right)\right) \\
&= -\frac{\partial}{\partial w_k}\sum_{i=1}^{n} y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) + \frac{\partial}{\partial w_k}\sum_{i=1}^{n} \log\left(1 + \exp(\mathbf{w}^\top \mathbf{x}_i + b)\right) \\
&= -\sum_{i=1}^{n} x_i^{(k)} y_i + \sum_{i=1}^{n} \frac{\frac{\partial}{\partial w_k}\left(1 + \exp(\mathbf{w}^\top \mathbf{x}_i + b)\right)}{1 + \exp(\mathbf{w}^\top \mathbf{x}_i + b)} \qquad \text{car } \log(u)' = \frac{u'}{u} \\
&= -\sum_{i=1}^{n} x_i^{(k)} y_i + \sum_{i=1}^{n} x_i^{(k)} \frac{\exp(\mathbf{w}^\top \mathbf{x}_i + b)}{1 + \exp(\mathbf{w}^\top \mathbf{x}_i + b)} \qquad \text{car } \exp(u)' = u' \exp(u) \\
&= -\sum_{i=1}^{n} x_i^{(k)} \left(y_i - \frac{\exp(\mathbf{w}^\top \mathbf{x}_i + b)}{1 + \exp(\mathbf{w}^\top \mathbf{x}_i + b)}\right) \\
&= -\sum_{i=1}^{n} x_i^{(k)} \left(y_i - \frac{\exp(\mathbf{z}_i)}{1 + \exp(\mathbf{z}_i)}\right)
\end{aligned}$$

avec $\mathbf{z}_i = \mathbf{w}^\top \mathbf{x}_i + b$