Régression logistique

Alexandre Allauzen

Université Paris-Sud / LIMSI-CNRS

Plan

1. Apprentissage / optimisation

2. Couche cachée / représentation de mots

Programme d'optimisation

Soient $\mathcal{D} = (\boldsymbol{x}_i, \tilde{y}_i)_{i=1}^{n} = (\mathcal{X}, \tilde{\mathcal{Y}})$ et un modèle de regression logistique, $\boldsymbol{\theta} = (w_0, \mathbf{w})$.

Fonction de perte à optimiser (minimiser)

$$\mathcal{L}(\boldsymbol{\theta}) = -log(P(\tilde{\mathcal{Y}}|\mathcal{X}, \boldsymbol{\theta})) = -\Big(\sum_{i=1}^{n} \tilde{y}_i log(\pi_i) + (1 - \tilde{y}_i) log(1 - \pi_i)\Big)$$

Avec:

$$\pi_i = \sigma(w_0 + \mathbf{w}^T \boldsymbol{x}_i) = \frac{1}{1 + e^{-(w_0 + \mathbf{w}^T \boldsymbol{x}_i)}}$$

Apprentissage = optimisation

Comment trouver $\boldsymbol{\theta}$ de manière à minimiser $\mathcal{L}(\boldsymbol{\theta})$?

Minimisation de fonction

Mauvaise nouvelle

Il n'existe pas de solution analytique a notre problème d'optimisation.

Bonnes nouvelles

- La fonction à optimiser est convexe.
- Les fonction convexe sont "faciles" à minimiser

$\rightarrow$ il existe un un algorithme efficace, facile et générique :

la descente de gradient (stochastique) et plein de variantes.

Méthodes de descentes - intuition

Comment atteindre le minimum ? Comment le reconnaître ?
$\rightarrow$ Fonction convexe : ouf !

Attention :

On ne “voit” pas la courbe ! On ne dispose que d'information locale. Comment faire ?

Dérivée partielle

Soit une fonction d'un ensemble de paramètres:

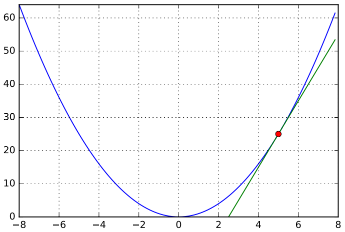
$$l(\boldsymbol{\theta}),\ \boldsymbol{\theta} = (w_0, \mathbf{w}) = (w_0, w_1, ..., w_K)$$

La dérivée partielle de cette fonction par rapport au paramètres w_i:

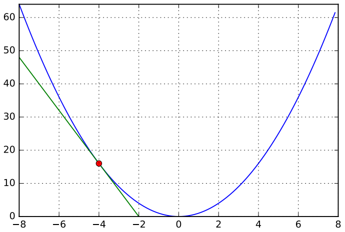
$$\frac{\partial l(\boldsymbol{\theta})}{\partial w_i}$$

- $\rightarrow$ la dérivée de $l(\boldsymbol{\theta})$ où seul w_i peut varier.
- $\boldsymbol{\theta}$ est fixé, sauf w_i : $l(\boldsymbol{\theta}) \rightarrow l(w_i)$

Interprétation de la dérivée partielle



$$w'_i = w_i + \eta \frac{\partial l(\boldsymbol{\theta})}{\partial w_i}$$
$$l(w'_i) > l(w_i)$$



$$w'_i = w_i - \eta \frac{\partial l(\boldsymbol{\theta})}{\partial w_i}$$
$$l(w'_i) < l(w_i)$$

$\eta > 0$, assez petit

Minimisation de l selon w_i

Idée

Partir de quelque part, choisir une direction de descente et l'emprunter (un peu). Puis recommencer !

Algorithme

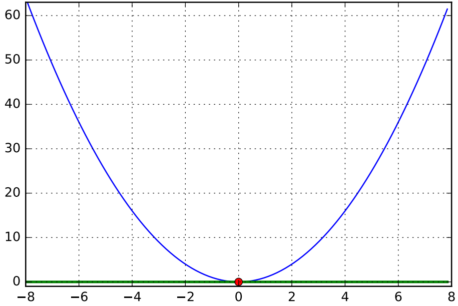
- Choisir η et une valeur initiale de de w_i.
- Tant que :
  - calcul de $l(\boldsymbol{\theta})$
  - calcul de la dérivée partielle puis mise à jour de w_i

$$w_i' = w_i - \eta \frac{\partial l(\boldsymbol{\theta})}{\partial w_i}$$

“Convergence”

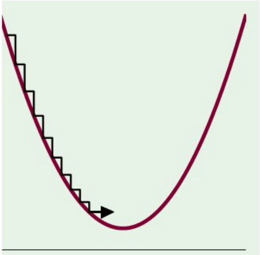
Au minimum:

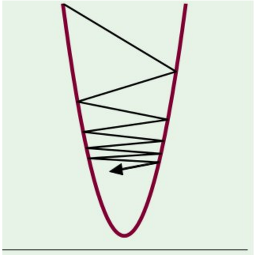
$$\frac{\partial l(\boldsymbol{\theta})}{\partial w_i} = 0$$

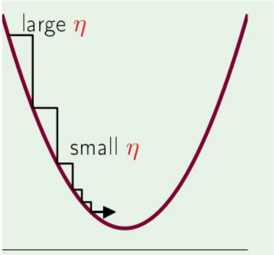


Choix du pas de descente : η

- des petits pas : approximation valable localement mais alors on avance tout doucement
- des grands pas : oscillation, divergence, ...







- Gradient à pas constant
- Gradient à pas optimal
- De toutes façon, c'est juste *un* pas, pourquoi s'embêter ?
- De très nombreuses méthodes pour optimiser le pas.

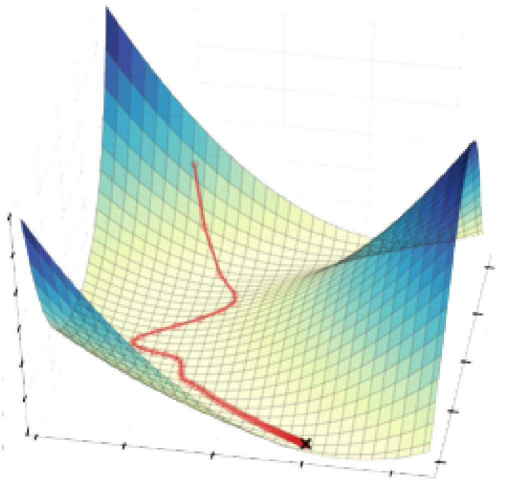
Généralisation : le gradient

- Mettons tous les paramètres dans un vecteur :
  $\boldsymbol{\theta} = (w_0, \mathbf{w}) = (w_0, w_1, w_2, ..., w_K)$
- calculons toutes les dérivées partielles que l'on met dans un vecteur:

$$\begin{pmatrix} \partial l(\boldsymbol{\theta})/\partial w_0 \\ \partial l(\boldsymbol{\theta})/\partial w_1 \\ \partial l(\boldsymbol{\theta})/\partial w_2 \\ ... \\ \partial l(\boldsymbol{\theta})/\partial w_K \end{pmatrix} = \nabla_{\boldsymbol{\theta}} l$$

$\nabla_{\boldsymbol{\theta}} l$ est le (vecteur) gradient de l selon $\boldsymbol{\theta}$.

Descente de gradient: 2D



Descente de gradient

Algorithme

- Choisir η et une valeur initiale de de $\boldsymbol{\theta}$.
- Tant que :
    - calcul de $l(\boldsymbol{\theta})$
    - calcul du gradient et mise à jour :

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} l$$

Apprentissage par descente de gradient

Algorithme Batch

$$\mathcal{L}(\boldsymbol{\theta}) = -\big(\sum_{i=1}^{n} \tilde{y}_i log(\pi_i) + (1-\tilde{y}_i)log(1-\pi_i) = \sum_{i=1}^{n} l(\boldsymbol{\theta}; \boldsymbol{x}_i, \tilde{y}_i)$$

$$l(\boldsymbol{\theta}; \boldsymbol{x}_i, \tilde{y}_i) = -\tilde{y}_i log(\pi_i) + (1-\tilde{y}_i)log(1-\pi_i)$$

Objectif, minimiser : $\mathcal{L}(\boldsymbol{\theta})$

Tant que:

Pour $i = 1...n$:

$$\left.\begin{array}{l} \mathcal{L}(\boldsymbol{\theta}) += l(\boldsymbol{\theta}; \boldsymbol{x}_i, \tilde{y}_i) \\ \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) += \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}; \boldsymbol{x}_i, \tilde{y}_i) \end{array}\right\} \text{une époque}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$$

Apprentissage par descente de gradient

Descente de gradient stochastique

Tant que:

$$\left.\begin{array}{l} \text{Mélanger } \mathcal{D} \\ \text{Pour } i = 1...n : \\ \qquad \mathcal{L}(\boldsymbol{\theta}) += l(\boldsymbol{\theta}; \boldsymbol{x}_i, \tilde{y}_i) \\ \qquad \boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}; \boldsymbol{x}_i, \tilde{y}_i) \end{array}\right\} \text{une époque}$$

Variante mini-batch (batch-size $= b << n$)

Tant que:

Mélanger $\mathcal{D}$

Pour $i = 1...(n/b)$:

$$\begin{array}{ll} \mathbf{u}_i = \mathbf{0} & \\ \mathcal{L}(\boldsymbol{\theta}) += l(\boldsymbol{\theta}; \boldsymbol{x}_j, \tilde{y}_j), & \text{pour } j = (i*b)...(i*(b+1)-1) \\ \mathbf{u}_i += \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}; \boldsymbol{x}_j, \tilde{y}_j), & \text{pour } j = (i*b)...(i*(b+1)-1) \\ \boldsymbol{\theta} = \boldsymbol{\theta} - \eta \mathbf{u}_i & \end{array}$$

Plan

1. Apprentissage / optimisation

2. Couche cachée / représentation de mots

Séparation linéaire et sac binaire

$$w_0 + \mathbf{w}^T \boldsymbol{x} = w_0 + \begin{pmatrix} 0 \\ \mathbf{1} \\ 0 \\ \mathbf{1} \\ \mathbf{1} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{pmatrix} = w_0 + w_2 + w_4 + w_5$$

La classe est positive ($y = 1$) si

$$\begin{aligned} w_0 + w_2 + w_4 + w_5 &> 0 \\ w_2 + w_4 + w_5 &> -w_0 \\ w_{this} + w_{long} + w_{great} &> \text{seuil} \end{aligned}$$

Représenter les mots dans un espace de dimension K

$$\begin{array}{l} \textit{the} \\ \textbf{this} \\ \textit{awesome} \\ \textbf{long} \\ \textbf{great} \end{array} \quad \begin{pmatrix} 0 \\ \mathbf{1} \\ 0 \\ \mathbf{1} \\ \mathbf{1} \end{pmatrix} \rightarrow \underbrace{(\mathbf{v}_2, \mathbf{v}_4, \mathbf{v}_5)}_{\text{un vecteur par mot présent}} \rightarrow \underbrace{(\mathbf{v}_2 + \mathbf{v}_4 + \mathbf{v}_5)}_{\text{document : une somme}}$$

Motivation

- the cat is walking in the bedroom
- the dog is running in the room

Le modèle doit pouvoir apprendre des représentations ($\mathbf{v}$) similaires pour des mots d'usage similaire.

Regréssion logistique modifiée

Représentation d'un document:

$$\mathbf{R} \times \boldsymbol{x} = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{v}_4 & \mathbf{v}_5 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \times \begin{pmatrix} 0 \\ \mathbf{1} \\ 0 \\ \mathbf{1} \\ \mathbf{1} \end{pmatrix} = \mathbf{v}_2 + \mathbf{v}_4 + \mathbf{v}_5 = \mathbf{d}$$

Régression logistique:

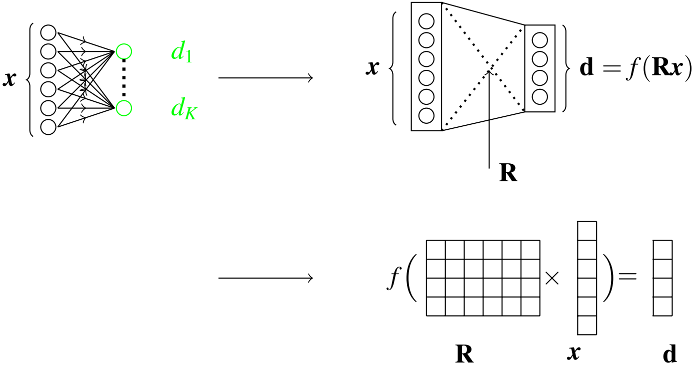
$$P(y = 1|\boldsymbol{x}) = \sigma(w_0 + \mathbf{w}^T\mathbf{d})$$

Les paramètres sont :

$$\boldsymbol{\theta} = (\mathbf{R}, w_0, \mathbf{w})$$

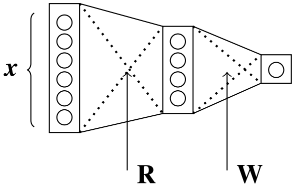
$\rightarrow$ à apprendre

Représentation neuronale



- Le vecteur d'entrée x : couche d'entrée, composée de neurone recepteur, n'a pas de paramètres
- Le vecteur $\mathbf{d}$: une couche neuronale *cachée*, de paramètres $\mathbf{R}$

Premier réseau de neurones



$$P(y = 1|\boldsymbol{x}; \boldsymbol{\theta}) = \sigma(\boldsymbol{W} \times \mathbf{d} + w_0)$$

- $\boldsymbol{x}$: $(|\mathcal{V}|, 1)$
- $\mathbf{R}$: $(K, |\mathcal{V}|)$
- $\mathbf{d}$: $(K, 1)$
- $\mathbf{W}$: $(1, K)$
- y : $(1, 1)$

Apprentisssage et notion de couche cachée

Apprentissage de **W**

- Connaissant **R**, on sait l'apprendre
- Calcul du gradient de $\mathcal{L}(\boldsymbol{\theta})$ par rapport à **W** et mise à jour de **W**

Apprentissage de **R**

- Il faut calculer le gradient de $\mathcal{L}(\boldsymbol{\theta})$ par rapport à **R**
- **R** "ne voit pas" $\mathcal{L}(\boldsymbol{\theta})$, elle est cachée
- Back-propagation du gradient