Descente de gradient

Principe de la descente de gradient pour l'apprentissage supervisé

Application à la régression linéaire et la régression logistique

Ricco Rakotomalala

Université Lumière Lyon 2

Contexte big data – Volumétrie des données

- Contexte big data : la taille des bases à traiter devient un enjeu essentiel pour les algorithmes de machine learning
- Il faut développer des stratégies d'apprentissage qui permettent d'appréhender les grandes bases (en particulier en nombre de descripteurs)
- En réduisant notamment la taille des structures de données à maintenir en mémoire
- Tout en obtenant des résultats de qualité satisfaisante (comparables à ceux produits par les algorithmes standards ex. régression logistique)
- Descente de gradient (stochastique) n'est pas un concept nouveau (cf. ADALINE, 1960), mais elle connaît un très grand intérêt aujourd'hui, en particulier pour l'entraînement des réseaux de neurones profonds (deep learning). Elle permet aussi de revisiter des approches statistiques existantes (ex. régression dans ce document)

Plan

1. Algorithme du gradient
2. Régression linéaire multiple
3. Descente de gradient stochastique
4. Taux d'apprentissage
5. Régression logistique
6. Logiciels
7. Conclusion

Démarche itérative pour la minimisation d'une fonction

ALGORITHME DU GRADIENT

Optimisation d'une fonction différentiable et convexe

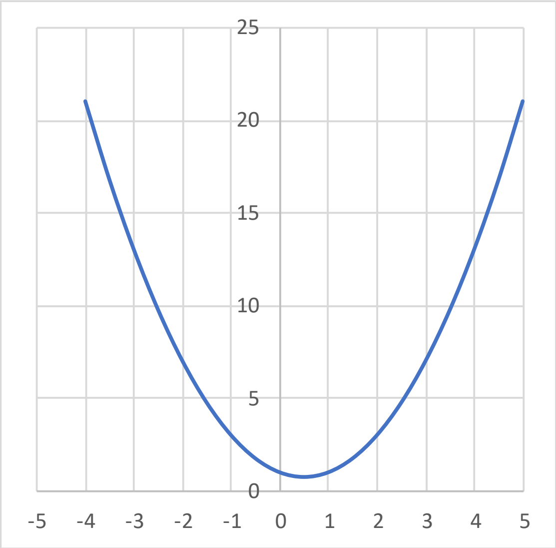
Maximiser ou minimiser une fonction est un problème usuel dans de nombreux domaines.

Ex. $f(x) = x^2 - x + 1$; à minimiser par rapport à x

$$\min_x f(x) = x^2 - x + 1$$

f() est la fonction à minimiser

x joue le rôle de paramètre ici c.-à-d. on recherche la valeur de x qui minimise f()



La solution analytique passe par :

$$f'(x) = 0$$

En s'assurant que $f''(x) > 0$

$$f'(x) = 2x - 1 = 0 \Rightarrow x^* = \frac{1}{2}$$

Algorithme du gradient (descente de gradient)

Parfois la résolution analytique n'est pas possible, parce que le nombre de paramètres est élevé par exemple, ou parce que le calcul serait trop coûteux ➔ approximation avec une approche itérative.

Quasiment impossible de suggérer des valeurs « intelligentes »

Algorithme du gradient

1. Initialiser avec x_0 (au hasard)
2. Répéter

$$x_{t+1} = x_t - \eta \times \nabla f(x_t)$$

3. Jusqu'à convergence

Le « gradient », généralisation multidimensionnelle de la dérivée [si un seul paramètre], au point x_t. Indique la direction et l'importance de la pente au voisinage de x_t.

— parce qu'on cherche à minimiser f(), on prendrait + sinon.

η est un paramètre qui permet de moduler la correction (η trop faible, lenteur de convergence ; η trop élevé, oscillation)

Nombre d'itérations fixé, ou différence entre valeurs successives x_t, ou $\|\nabla f(x_t)\|$ très petit

Algorithme du gradient - Exemple

$$\begin{cases} f(x) = x^2 - x + 1 \\ \nabla f(x) = \dfrac{\partial f(x)}{\partial x} = f'(x) = 2x - 1 \end{cases}$$

Il n’y a qu’un seul paramètre, la dérivée partielle est égale à la dérivée.

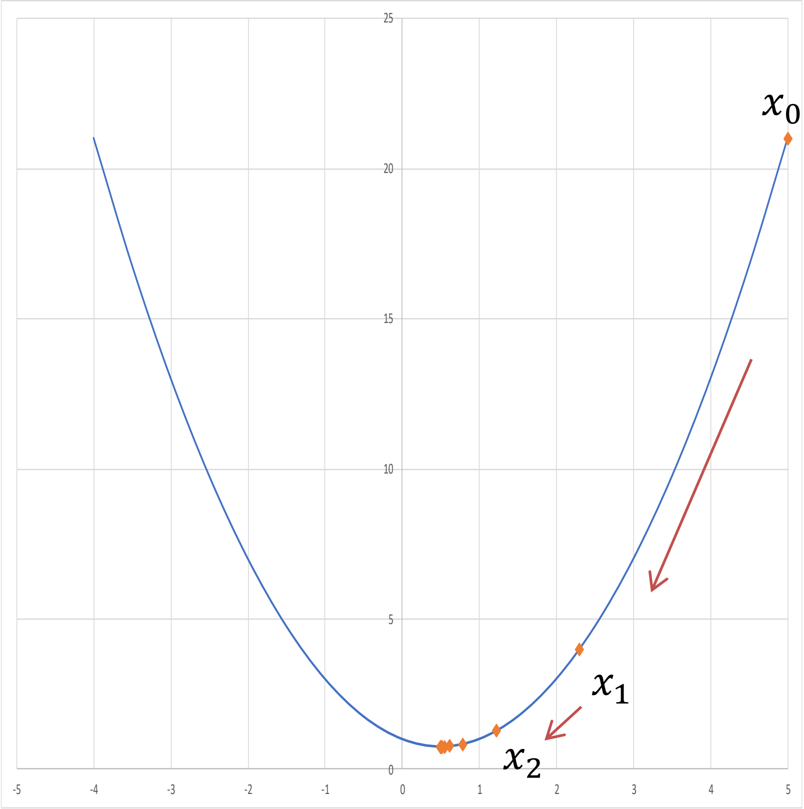
$\eta = 0.3$

eta	0.3

$x_0 = 5$

x	f'(x_t)	f(x)
5.0000		21.0000
2.3000	9.0000	3.9900
1.2200	3.6000	1.2684
0.7880	1.4400	0.8329
0.6152	0.5760	0.7633
0.5461	0.2304	0.7521
0.5184	0.0922	0.7503
0.5074	0.0369	0.7501
0.5029	0.0147	0.7500
0.5012	0.0059	0.7500
0.5005	0.0024	0.7500
0.5002	0.0009	0.7500
0.5001	0.0004	0.7500
0.5000	0.0002	0.7500

$$x_{t+1} = x_t - \eta \times \nabla f(x_t)$$



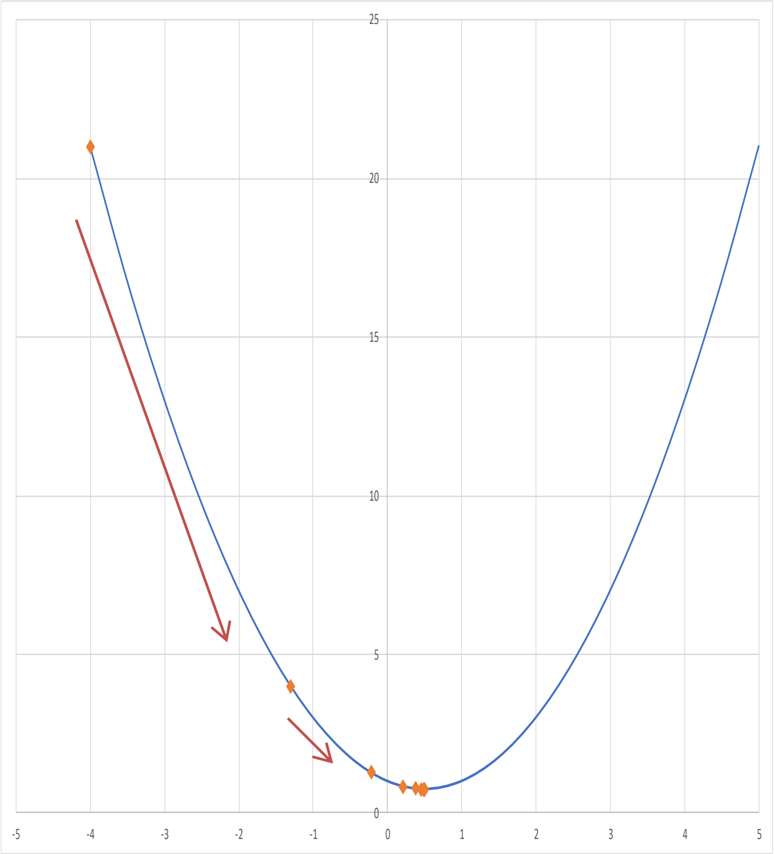
$\ldots x_2 < x_1 < x_0$

Algorithme du gradient – Exemple (2)

On aurait pu partir de l'autre côté…

eta	0.3

x	f'(x_t)	f(x)
-4.0000		21.0000
-1.3000	-9.0000	3.9900
-0.2200	-3.6000	1.2684
0.2120	-1.4400	0.8329
0.3848	-0.5760	0.7633
0.4539	-0.2304	0.7521
0.4816	-0.0922	0.7503
0.4926	-0.0369	0.7501
0.4971	-0.0147	0.7500
0.4988	-0.0059	0.7500
0.4995	-0.0024	0.7500
0.4998	-0.0009	0.7500
0.4999	-0.0004	0.7500
0.5000	-0.0002	0.7500



$x_0 < x_1 < x_2 \ldots$

Quel rapport avec le machine learning ?...

Application de la descente de gradient à la régression linéaire multiple

RÉGRESSION LINÉAIRE MULTIPLE

Régression linéaire multiple

Modéliser y (quantitative) à partir de p variables explicatives $X = (x_1, x_2, ..., x_p)$ quantitatives

$$y_i = a_0 + a_1 x_{i1} + \cdots + a_p x_{ip} + \varepsilon_i$$

Sur un échantillon de taille n, on cherche à minimiser le critère

$$S = \frac{1}{2n} \sum_{i=1}^{n} \varepsilon_i^2$$

Critère des moindres carrés

On a un problème de minimisation par rapport aux paramètres $a = (a_0, a_1, \cdots, a_p)$

$$\min_{a_0, a_1, \ldots, ap} S = \sum_{i=1}^{n} (y_i - \langle a, x_i \rangle)^2$$

Où $x_i = (x_{oi}, x_{i1}, \ldots, x_{ip})$ et (constante) $x_{oi} = 1$, $\forall i$

Il existe une solution « exacte »

$$\hat{a} = (X'X)^{-1} X'Y$$

Régression linéaire multiple – Descente de gradient

$$\hat{a} = (X'X)^{-1}X'Y$$

Nécessite la manipulation de la matrice (X'X) de taille (p+1, p+1), ingérable dès que p est élevé (millier de variables, grandes dimensions). D'autres approches plus sophistiquées existent, mais elle nécessitent toujours la manipulation d'une matrice de taille (p+1,p+1) durant les calculs.

Descente de gradient

$$a^{t+1} = a^t - \eta \times \nabla S^t$$

Taux d'apprentissage (êta, learning rate)

Où $\nabla S^t =$

$$\left\{\begin{array}{l}
\dfrac{\partial S^t}{\partial a_0} = \dfrac{1}{n}\sum_{i=1}^{n}(-1) \times (y_i - \langle a^t, x_i \rangle) \\
\dfrac{\partial S^t}{\partial a_1} = \dfrac{1}{n}\sum_{i=1}^{n}(-x_{i1}) \times (y_i - \langle a^t, x_i \rangle) \\
\vdots \\
\dfrac{\partial S^t}{\partial a_p} = \dfrac{1}{n}\sum_{i=1}^{n}(-x_{ip}) \times (y_i - \langle a^t, x_i \rangle)
\end{array}\right.$$

Le vecteur gradient de taille (p+1,1) est composé des dérivées partielles de S par rapport à chaque paramètre du modèle.

1. Manipuler un vecteur de taille (p+1,1) plutôt qu'une matrice (p+1,p+1), voilà tout l'intérêt de la descente de gradient pour les grandes dimensions (il y en a d'autres, cf. les réseaux de neurones)
2. Mais elle nécessite de parcourir plusieurs fois la base (avec n observations)

Régression linéaire multiple – Exemple

x0	x1	x2	y
1	0.72	0.32	6.93
1	0.75	0.12	5.99
1	0.53	0.65	1.46
1	0.27	0.82	1.44
1	0.49	0.15	4.51
1	0.02	0.19	1.25
1	0.35	0.87	2.53
1	0.99	0.71	6.88
1	0.98	0.92	6.25
1	0.73	0.19	6.36

Il est préférable d'harmoniser les données (normalisation, standardisation) pour éviter les problèmes d'échelles.

$$a^0 = (0.1, 0.1, 0.1)$$

$$a^{30} = (1.658, 6.618, -2.314)$$

$$a^{solution} = (1.424, 7.173, -2.523)$$

eta	1.05

t	a0	a1	a2	S	dS/d_ao	dS/d_a1	dS/d_a2
0	0.100	0.100	0.100	224.72	-4.152	-3.003	-1.889
1	4.460	3.253	2.083	127.71	3.026	1.483	1.892
2	1.283	1.697	0.097	77.53	-2.040	-1.633	-0.825
3	3.425	3.411	0.963	50.95	1.529	0.609	1.045
4	1.819	2.771	-0.135	36.36	-0.992	-0.931	-0.315
5	2.860	3.749	0.196	27.95	0.783	0.193	0.606
6	2.039	3.546	-0.440	22.79	-0.472	-0.564	-0.078
7	2.534	4.138	-0.358	19.39	0.410	0.002	0.373
8	2.104	4.135	-0.750	17.00	-0.216	-0.367	0.026
9	2.330	4.521	-0.778	15.22	0.222	-0.079	0.245
10	2.097	4.604	-1.035	13.83	-0.090	-0.257	0.067
11	2.191	4.874	-1.105	12.72	0.127	-0.108	0.171
12	2.058	4.987	-1.284	11.80	-0.029	-0.191	0.078
13	2.088	5.188	-1.365	11.04	0.078	-0.112	0.125
14	2.006	5.306	-1.497	10.41	0.000	-0.149	0.075
15	2.006	5.463	-1.576	9.87	0.052	-0.106	0.096
16	1.951	5.574	-1.676	9.42	0.013	-0.121	0.068
17	1.938	5.701	-1.747	9.03	0.038	-0.096	0.075
18	1.898	5.802	-1.826	8.71	0.018	-0.100	0.059
19	1.879	5.907	-1.888	8.43	0.030	-0.085	0.060
20	1.847	5.996	-1.951	8.20	0.019	-0.084	0.050
21	1.827	6.084	-2.003	8.00	0.025	-0.074	0.048
22	1.801	6.161	-2.054	7.83	0.019	-0.071	0.042
23	1.782	6.236	-2.098	7.68	0.021	-0.064	0.039
24	1.759	6.303	-2.139	7.56	0.018	-0.061	0.035
25	1.741	6.367	-2.175	7.46	0.018	-0.055	0.032
26	1.722	6.425	-2.209	7.37	0.016	-0.052	0.029
27	1.705	6.479	-2.239	7.29	0.016	-0.047	0.026
28	1.688	6.529	-2.267	7.23	0.015	-0.044	0.024
29	1.673	6.575	-2.292	7.17	0.014	-0.041	0.022
30	1.658	6.618	-2.314	7.12	0.013	-0.038	0.019

Convergence lente parce petit effectifs (n=10).

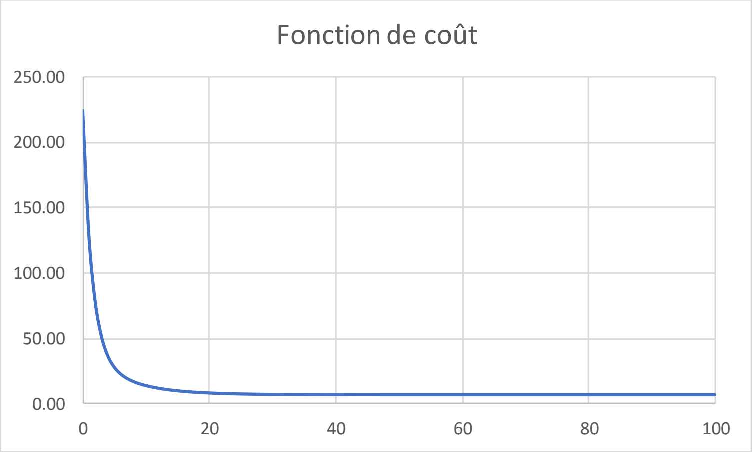
Régression linéaire multiple – Exemple (suite)

	eta	1.05					
t	a0	a1	a2	S	dS/d_ao	dS/d_a1	dS/d_a2
0	0.100	0.100	0.100	224.72	-4.152	-3.003	-1.889
1	4.460	3.253	2.083	127.71	3.026	1.483	1.892
2	1.283	1.697	0.097	77.53	-2.040	-1.633	-0.825
3	3.425	3.411	0.963	50.95	1.529	0.609	1.045

$$a^0 = (0.1,0.1,0.1) \Rightarrow \nabla S^0 = \begin{pmatrix} -4.152 \\ -3.003 \\ -1.889 \end{pmatrix}$$

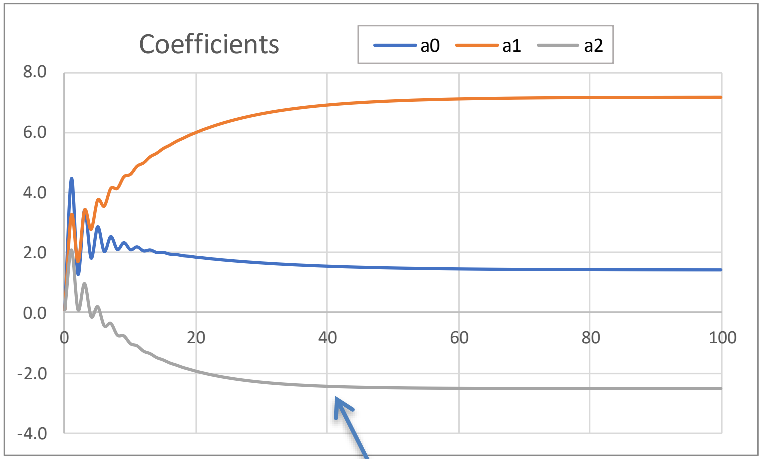
$$\begin{pmatrix} 0.1 \\ 0.1 \\ 0.1 \end{pmatrix} - 1.05 \times \begin{pmatrix} -4.152 \\ -3.003 \\ -1.889 \end{pmatrix} \rightarrow \begin{pmatrix} 4.460 \\ 3.253 \\ 2.083 \end{pmatrix} = a^1$$

Evolution de S au fil des itérations (t)



Baisse constante de la fonction de coût.

Evolution des coefficients au fil des itérations (t)



Oscillations au départ parce η=1.05 choisi très élevé

Solution acceptable dès t ≈ 40

Approche incrémentale pour le traitement des très grandes bases

DESCENTE DE GRADIENT STOCHASTIQUE

Correction par observation (online)

Gradient stochastique est une approximation de la descente de gradient, applicable lorsque la fonction objectif s'écrit comme une somme de fonctions dérivables : c'est très souvent le cas en apprentissage supervisé (ou on s'arrange pour que ce soit le cas)

Exemple de la régression linéaire multiple via les moindres carrés

$$S = \sum_{i=1}^{n} (y_i - \langle a, x_i \rangle)^2 \Rightarrow (y_i - \langle a, x_i \rangle)^2$$

Est dérivable par rapport au paramètres (a_j)

Il est possible de corriger les paramètres estimés pour le passage de chaque observation

$$\Rightarrow a \coloneqq a - \eta \times \nabla S_i$$

Où $\frac{\partial S_i}{\partial a_j} = \left(-x_{ij}\right) \times (y_i - \langle a, x_i \rangle)$

Correction par observation (online) - Exemple

eta	0.5

x0	x1	x2	y
1	0.72	0.32	6.93
1	0.75	0.12	5.99
1	0.53	0.65	1.46
1	0.27	0.82	1.44
1	0.49	0.15	4.51
1	0.02	0.19	1.25
1	0.35	0.87	2.53
1	0.99	0.71	6.88
1	0.98	0.92	6.25
1	0.73	0.19	6.36

i	a0	a1	a2	S	dS/d_ao	dS/d_a1	dS/d_a2
0	0.100	0.100	0.100	224.72	-6.726	-4.843	-2.152
1	3.463	2.521	1.176	47.83	-0.495	-0.371	-0.059
2	3.710	2.707	1.206	56.63	4.469	2.369	2.905
3	1.476	1.523	-0.247	81.22	0.245	0.066	0.201
4	1.354	1.490	-0.347	89.71	-2.479	-1.215	-0.372
5	2.593	2.097	-0.161	35.50	1.354	0.027	0.257
6	1.916	2.083	-0.290	50.15	-0.137	-0.048	-0.119
7	1.984	2.107	-0.230	47.19	-2.973	-2.943	-2.111
8	3.471	3.579	0.825	51.62	1.487	1.457	1.368
9	2.727	2.850	0.141	27.12	-1.526	-1.114	-0.290
10	3.490	3.407	0.286	39.93	-0.896	-0.645	-0.287
11	3.938	3.729	0.429	61.57	0.796	0.597	0.096
12	3.540	3.431	0.382	43.13	4.146	2.197	2.695
13	1.467	2.332	-0.966	68.10	-0.136	-0.037	-0.111
14	1.534	2.350	-0.910	63.90	-1.960	-0.961	-0.294
15	2.515	2.831	-0.763	27.57	1.176	0.024	0.223
16	1.927	2.819	-0.875	39.08	-0.378	-0.132	-0.329
17	2.116	2.885	-0.710	32.38	-2.413	-2.389	-1.713
18	3.322	4.079	0.146	39.90	1.204	1.180	1.108
19	2.720	3.489	-0.408	21.36	-1.170	-0.854	-0.222
20	3.305	3.917	-0.297	30.72	-0.900	-0.648	-0.288
21	3.755	4.241	-0.153	50.09	0.927	0.695	0.111
22	3.291	3.893	-0.208	31.51	3.759	1.992	2.444
23	1.412	2.897	-1.430	61.38	-0.419	-0.113	-0.343
24	1.621	2.953	-1.258	49.38	-1.631	-0.799	-0.245
25	2.436	3.353	-1.136	22.86	1.038	0.021	0.197
26	1.918	3.342	-1.235	32.14	-0.517	-0.181	-0.450
27	2.176	3.433	-1.010	24.43	-2.023	-2.002	-1.436
28	3.187	4.434	-0.292	32.71	1.014	0.994	0.933
29	2.680	3.937	-0.758	18.09	-0.950	-0.693	-0.180

1. La décroissance de S au passage de chaque observation n'est pas assurée mais, en moyenne, sur l'ensemble des observations, sa convergence est effective (Wikipedia).
2. Dans l'exemple précédent (descente de gradient), après 3 passages sur les observations nous avions **S = 50.95**. Ici nous obtenons **S = 18.09** (η n'est pas le même non plus ceci étant dit, mais le commentaire reste valable).

Stratégies – Gradient stochastique

1. Descente de Gradient classique (batch gradient descent). On fait passer la totalité des observations, le gradient est calculé, les coefficients sont corrigés. Etc.

2. Online. Gradient calculé pour chaque observation, correction des coefficients. Etc.

3. Mini-batch (traitement par lots). On fait passer un groupe (effectif = paramètre de l'algorithme) d'observations. Calcul du gradient. Correction des coefficients. Etc.

- Le traitement par lots permet d'améliorer la convergence en réduisant le nombre de passage sur la base entière.
- Il permet également de se contenter de charger partiellement les données en mémoire au fur et à mesure.

Taux fixe ou taux décroissant au fil du processus d’apprentissage

CHOIX DU TAUX D’APPRENTISSAGE

Importance du taux d'apprentissage (learning rate)

- η détermine la vitesse de convergence du processus d'apprentissage
- Améliorer le dispositif en faisant évoluer η au fil des itérations (fort au début pour accélérer la convergence, faible à la fin pour améliorer la précision)

Exemple : SGDRegressor du package "scikit-learn" (Python)

```
class sklearn.linear_model. SGDRegressor (loss='squared_loss', penalty='l2', alpha=0.0001, l1_ratio=0.15,
fit_intercept=True, max_iter=None, tol=None, shuffle=True, verbose=0, epsilon=0.1, random_state=None,
learning_rate='invscaling', eta0=0.01, power_t=0.25, warm_start=False, average=False, n_iter=None)        [source]
```

learning_rate : string, optional

The learning rate schedule:

- 'constant': eta = eta0
- 'optimal': eta = 1.0 / (alpha * (t + t0)) [default]
- 'invscaling': eta = eta0 / pow(t, power_t)

where t0 is chosen by a heuristic proposed by Leon Bottou.

eta0 : double, optional

The initial learning rate [default 0.01].

power_t : double, optional

The exponent for inverse scaling learning rate [default 0.25].

(t_0 ???) La documentation n'est pas très disserte à ce sujet.

$$\eta = \frac{\eta_0}{t^{0.25}}$$

0.25 étant lui-même modifiable

En tous les cas, il est acquis que faire évoluer η au fil des itérations sur la base (t) améliore l'efficacité du dispositif.

Application de la descente de gradient à la régression logistique

RÉGRESSION LOGISTIQUE

Régression logistique binaire

Nous sommes dans le cadre de l'apprentissage supervisé
où la variable cible y est binaire c.-à-d. $y \in \{1,0\}$

Fonction de perte :

Binary cross-entropy

$$J(a) = -\frac{1}{n}\sum_{i=1}^{n} y_i \ln p_i + (1 - y_i)\ln(1 - p_i)$$

p_i est la proba. conditionnelle P(Y/X) modélisée avec la régression logistique

$$p_i = \frac{1}{1 + e^{-(a_0 + a_1 x_{i1} + \cdots + a_p x_{ip})}}$$

J(a) = - LL, où LL est la log-vraisemblance du modèle binomial

Gradient :

Signal envoyé par x_j (cf. cours Perceptron)

$$\frac{\partial J}{\partial a_j} = \frac{1}{n}\sum_{i=1}^{n} (x_{ij}) \times (y_i - p_i)$$

Amplitude de l'erreur

Gradient stochastique :

La fonction de perte s'écrit comme une somme de fonctions dérivables, l'approche du gradient stochastique peut s'appliquer

!

Régression logistique multinomiale

Y est catégorielle et peut prendre K modalités c.-à-d. $y \in \{y_1, \dots, y_k, \dots, y_K\}$

Codage : Codage en K indicatrices 0/1

Ex.

Y	Y_A	Y_B	Y_C
A	1	0	0
A	1	0	0
B	0	1	0
A	1	0	0
C	0	0	1
A	1	0	0

Fonction de classement : a devient une matrice de dimension (K, p+1)

Cf. cours sur le « Perceptron »

Fonction de perte :

Categorical cross-entropy

$$J(a) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik} \ln p_{ik}$$

Gradient : Le vecteur gradient est un vecteur de dimension (K x (p+1), 1)

Quelques packages pour Python et R, entres autres….

LOGICIELS

Python – Librairie « scikit-learn »

scikit learn

Home | Installation | Documentation | Examples

Google Custom Search

Previous 1.4. Support … | Next 1.5. Nearest … | Up 1. Supervised…

scikit-learn v0.19.1
Other versions

Please **cite us** if you use the software.

1.5. Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) Support Vector Machines and Logistic Regression. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention just recently in the context of large-scale learning.

SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. Given that the data is sparse, the classifiers in this module easily scale to problems with more than 10^5 training examples and more than 10^5 features.

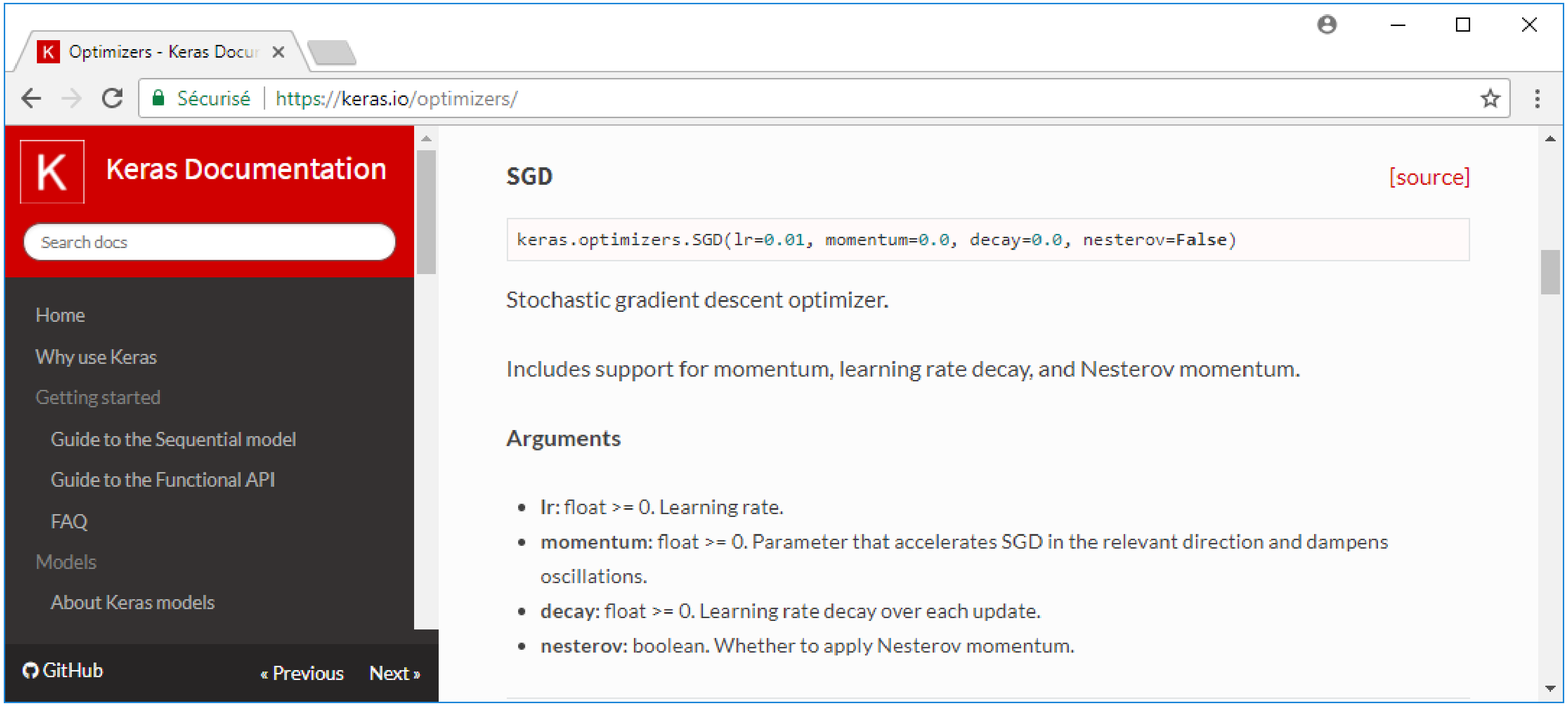
The advantages of Stochastic Gradient Descent are:

- Efficiency.
- Ease of implementation (lots of opportunities for code tuning).

The disadvantages of Stochastic Gradient Descent include:

- SGD requires a number of hyperparameters such as the regularization parameter and the number of iterations.
- SGD is sensitive to feature scaling.

Python – Librairie « tensorflow / keras »



« Stochastic gradient descent » est utilisé – parmi d'autres – pour l'apprentissage des réseaux de neurones profonds (deep learning).

R – Librairie « gradDescent », et d'autres…

CRAN - Package gradDe

https://cran.r-project.org/web/packages/gradDescent/index.html

gradDescent: Gradient Descent for Regression Tasks

An implementation of various learning algorithms based on Gradient Descent for dealing with regression tasks. The variants of gradient descent algorithm are : Mini-Batch Gradient Descent (MBGD), which is an optimization to use training data partially to reduce the computation load. Stochastic Gradient Descent (SGD), which is an optimization to use a random data in learning to reduce the computation load drastically. Stochastic Average Gradient (SAG), which is a SGD-based algorithm to minimize stochastic step to average. Momentum Gradient Descent (MGD), which is an optimization to speed-up gradient descent learning. Accelerated Gradient Descent (AGD), which is an optimization to accelerate gradient descent learning. Adagrad, which is a gradient-descent-based algorithm that accumulate previous cost to do adaptive learning. Adadelta, which is a gradient-descent-based algorithm that use hessian approximation to do adaptive learning. RMSprop, which is a gradient-descent-based algorithm that combine Adagrad and Adadelta adaptive learning ability. Adam, which is a gradient-descent-based algorithm that mean and variance moment to do adaptive learning. Stochastic Variance Reduce Gradient (SVRG), which is an optimization SGD-based algorithm to accelerates the process toward converging by reducing the gradient. Semi Stochastic Gradient Descent (SSGD),which is a SGD-based algorithm that combine GD and SGD to accelerates the process toward converging by choosing one of the gradients at a time. Stochastic Recursive Gradient Algorithm (SARAH), which is an optimization algorithm similarly SVRG to accelerates the process toward converging by accumulated stochastic information. Stochastic Recursive Gradient Algorithm+ (SARAHPlus), which is a SARAH practical variant algorithm to accelerates the process toward converging provides a possibility of earlier termination.

Version:	3.0
Published:	2018-01-25
Author:	Galih Praja Wijaya, Dendi Handian, Imam Fachmi Nasrulloh, Lala Septem Riza, Rani Megasari, Enjun Junaeti
Maintainer:	Lala Septem Riza <lala.s.riza at upi.edu>
License:	GPL-2 | GPL-3 | file LICENSE [expanded from: GPL (≥ 2) | file LICENSE]
URL:	https://github.com/drizzersilverberg/gradDescentR
NeedsCompilation:	no
In views:	MachineLearning
CRAN checks:	gradDescent results

Downloads:

Reference manual:	gradDescent.pdf
Package source:	gradDescent_3.0.tar.gz
Windows binaries:	r-prerel: gradDescent_3.0.zip, r-release: gradDescent_3.0.zip, r-oldrel: gradDescent_3.0.zip
OS X binaries:	r-prerel: gradDescent_3.0.tgz, r-release: gradDescent_3.0.tgz
Old sources:	gradDescent archive

Linking:

Please use the canonical form https://CRAN.R-project.org/package=gradDescent to link to this page.

Et toutes celles qui implémentent les réseaux de neurones dont les perceptrons simples et multicouches (ex. Tensorflow / Keras, nnet, etc.) et qui s'appuient sur la descente de gradient (stochastique).

CONCLUSION

Pourquoi Descente de Gradient pour l'apprentissage supervisé ?

- Approches et surtout implémentations classiques des méthodes de data mining impuissantes par rapport aux très grandes volumétries
- L'algorithme du gradient / gradient stochastique permet de les appréhender sans nécessiter de ressources machines prohibitives
- Possibilité de parallélisation des algorithmes

Attention cependant…

- Grand nombre de paramètres pas toujours faciles à appréhender, qui influent sur le comportement de l'algorithme
- Toujours ramener les variables sur la même échelle (normalisation, standardisation) pour éviter que les disparités faussent le déroulement de l'optimisation

RÉFÉRENCES

Références

- Wikipedia, « Gradient descent »
- Wikipedia, « Stochastic Gradient Descent »
- E. Biernat, M. Lutz, « Data Science : fondamentaux et études de cas », Eyrolles, 2015 ; chapitres 3 et 4.
- S. Ruder, « An overview of gradient descent optimization algorithms », version 09.02.2018.
- Scikit-learn, « Stochastic Gradient Descent », version 0.19.1 ; section 15.1.