TP gradient stochastique – MDI341

Aurélien Bellet, Alexandre Gramfort, Joseph Salmon

30 janvier 2019

Introduction

Le but de cette séance est la mise en œuvre d'algorithmes de type gradient stochastique (en anglais : *Stochastic Gradient Descent*, SGD). Dans un premier temps, afin de se familiariser avec le SGD, on va mettre en œuvre l'algorithme dans le cadre classique de la classification binaire.

Un fichier Python est fourni (`SGD_classification.py`) pour guider les réponses, avec néanmoins certaines parties manquantes qu'il faut compléter.

Définitions et notations

On rappelle ici le cadre de la classification binaire supervisée, et l'on présente les notations utilisées :

- $\mathcal{Y}$ l'ensemble des étiquettes des données (*labels* en anglais). On traite ici le cas binaire pas simplicité, il n'y a donc que deux classes. Il est confortable de raisonner avec $\mathcal{Y} = \{-1, 1\}$ pour représenter les étiquettes (on va considérer des signes au cours de ce travail).
- $\mathbf{x} = (x_1, \dots, x_p)^\top \in \mathcal{X} \subset \mathbb{R}^p$ est une observation, un exemple, un point (*sample* en anglais). La $j$ème coordonnée de $\mathbf{x}$ est la valeur prise par la $j$ème variable (*feature* en anglais).
- $\mathcal{D}_n = \{(\mathbf{x}_i, y_i), i = 1, \dots n\}$ est un ensemble d'apprentissage contenant n exemples et leurs étiquettes,
- Il existe un modèle probabiliste qui gouverne la génération des observations selon des variables aléatoires X et Y : $\forall i \in \{1, \dots, n\}, (\mathbf{x}_i, y_i) \overset{i.i.d}{\sim} (X, Y)$.
- On cherche à construire à partir de l'ensemble d'apprentissage $\mathcal{D}_n$ une fonction appelée classifieur, $\hat{f} : \mathcal{X} \mapsto \{-1, 1\}$ qui pour un nouveau point $\mathbf{x}_{\text{new}} \in \mathcal{X}$ fournit une étiquette $\hat{f}(\mathbf{x}_{\text{new}})$.
- On mesure la performance d'un classifieur, pour une perte $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, par le risque $\mathbb{E}(\ell(\hat{f}(\mathbf{x}), y))$. En pratique, cette quantité n'est pas calculable, on se sert donc de la contrepartie empirique du type $\frac{1}{n} \sum_{i=1}^{n} \ell(\hat{f}(\mathbf{x}_i), y_i)$. On note $\frac{\partial \ell}{\partial x_1}$ la dérivée partielle de ℓ par rapport à la première variable.

On se place dans le cas où la famille de classifieurs est indexée par un paramètre $\mathbf{w} \in \mathbb{R}^p$, c'est à dire $\hat{f}_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ (si l'on souhaite rajouter un paramètre de translation, on rajoutera par exemple une variable constante supplémentaire). Une manière de procéder pour obtenir un $\mathbf{w}$ satisfaisant est donc de choisir celui qui minimise le risque empirique. Quand n est très grand il peut-être bon d'utiliser, non pas une descente de gradient, mais plutôt une version stochastique décrite ci-dessous :

Algorithm 1 Algorithme du gradient stochastique

Données : les observations et leurs étiquettes $\mathcal{D}_n = \{(\mathbf{x}_i, y_i) : 1 \leq i \leq n\}$
Nombre maximal d'itérations : T et suite de pas de gradient : $(\gamma_t)_{t=1,\dots,T}$ (aussi appelé *learning rate*)
Résultat : $\mathbf{w}_T$

Initialiser (aléatoirement) $\mathbf{w}_0 \in \mathbb{R}^p$; initialiser $t = 0$
Tant que $t \leq T$:
  tirer aléatoirement i dans $\{1, \dots, n\}$
  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t \mathbf{x}_i \frac{\partial \ell}{\partial x_1}(\mathbf{w}_t^\top x_i, y_i)$
  $t \leftarrow t + 1$
optionnellement : $\bar{\mathbf{w}}_T \leftarrow \dfrac{1}{T} \displaystyle\sum_{t=1}^{T} \mathbf{w}_t$

Remarque : dans le cas où l'algorithme renvoie $\bar{\mathbf{w}}_T \leftarrow \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$, c'est à dire la moyenne des itérés plutôt que la dernière valeur, on parle de gradient stochastique moyenné (*averaged stochastic gradient descent* en anglais.)

Exemple sur données synthétiques

On pourra utiliser la fonction `stochastic_gradient` fournie dans `SGD_classification.py`.

1. On considère ici la perte quadratique : $\ell(a,b) = (a-b)^2/2$. Calculer $\frac{\partial \ell}{\partial x_1}(a,b)$.
2. Implémenter l'algorithme SGD avec le choix d'un pas constant γ et d'un nombre d'itérations T permettant de minimiser le critère et d'aboutir à une solution correcte.
3. Prendre comme modèle jouet : n vecteurs de $\mathbb{R}^p$ tiré de manière i.i.d selon une loi gaussienne, centrée réduite (prendre $n = 1000, p = 100$). On note $w^\star = (1, \dots, 1)^\top$, et l'on définit $\forall i \in \{1, \dots, n\}, y_i = \mathbf{x}_i^\top w^\star$. Donner le minimiseur de $\frac{1}{n}\sum_{i=1}^{n} \ell(\hat{f}_w(\mathbf{x}_i), y_i)$.
4. Afficher l'évolution de la valeur de l'objectif en fonction du nombre d'itérations, c'est-à-dire afficher la fonction $t \mapsto \frac{1}{n}\sum_{i=1}^{n} \ell(\hat{f}_{\mathbf{w}_t}(\mathbf{x}_i), y_i)$. On pourra considérer une échelle semi-logarithmique.
5. Utiliser la fonction `stochastic_gradient` fournie. Proposer le choix d'un pas constant γ et d'un nombre d'itérations T permettant de minimiser le critère et d'aboutir à une solution correcte.
6. Régulariser la difficulté en optimisant cette fois la fonction objectif $\frac{1}{n}\sum_{i=1}^{n} \ell(\hat{f}_{\mathbf{w}}(\mathbf{x}_i), y_i) + \frac{\alpha}{2}\|\mathbf{w}\|_2^2$, où $\alpha > 0$ est un paramètre de régularisation.
7. Peut-on faire diminuer encore plus la fonction objectif avec un pas non-constant $(\gamma_t)_{t=1,\dots,T}$ qui décroit au cours des itérations ? Si oui, quel problème cela pose-t-il en pratique ?

Exemple sur données réelles

8. Utiliser la fonction `stochastic_gradient` fournie. Proposer le choix d'un pas constant γ et d'un nombre d'itérations T permettant de minimiser le critère et d'aboutir à une solution correcte. On utilisera pour l'instant la perte quadratique : $\ell(a,b) = (a-b)^2/2$ et la base de données Iris.
9. Ajouter l'étape de moyennage optionnelle dans votre fonction. Comparer visuellement l'évolution de la fonction objectif $(\frac{1}{n}\sum_{i=1}^{n} \ell(\hat{f}_{\mathbf{w}}(\mathbf{x}_i), y_i))$ en fonction des itérations pour l'algorithme avec et sans cette étape. La mise à jour peut se faire de manière récursive en remarquant que $\bar{\mathbf{w}}_{T+1} = \frac{T}{T+1}\bar{\mathbf{w}}_T + \frac{1}{T+1}\mathbf{w}_{T+1}$. On implémentera aussi la même technique, mais en ne moyennant qu'après t_0 itérations, c'est à dire en considérant $\frac{1}{T-t_0}\sum_{t=t_0+1}^{T} \mathbf{w}_t$ [1].
10. Régulariser le problème en optimisant cette fois la fonction objectif $\frac{1}{n}\sum_{i=1}^{n} \ell(\hat{f}_{\mathbf{w}}(\mathbf{x}_i), y_i) + \frac{\alpha}{2}\|\mathbf{w}\|_2^2$, où $\alpha > 0$ est un paramètre de régularisation.
11. Adapter l'algorithme quand on prend comme fonction de perte la fonction "hinge" : $\ell(a,b) = \max(0, 1-ab)$.
12. Comparer vos implémentations avec les résultats donnés par `SGDRegressor` et `SGDClassifier` du package `Scikit-Learn`.

1. si besoin `http://research.microsoft.com/pubs/192769/tricks-2012.pdf`