Modélisation et reconnaissance des formes
Gradient stochastique

1 Présentation de la méthode

L'algorithme du gradient stochastique est une méthode de descente de gradient utilisée pour la minimisation d'une fonction objectif qui est écrite comme une somme de fonctions différentiables.

C'est un cas très fréquent dans l'estimation aux moindres carrés par exemple ($min_{a,b} \sum (y(i) - ax_i - b)^2$ ainsi que dans le domaine de la classification supervisée où on minimise la somme des distances entre valeur attendue et valeur prédite par le système sur l'ensemble des données disponibles.

La plupart du temps on cherche à trouver la valeur de w minimisant le risque empirique :

$$Q(w) = \frac{1}{n} \sum_{1}^{n} Q_i(w)$$

Une méthode de descente de gradient est le plus souvent employée. Il s'agit d'une méthode itérative construisant une suite w_n définie par

$$w_{n+1} = w_n - \mu \nabla Q(w) = w_n - \mu \frac{1}{n} \sum_{1}^{n} \nabla Q_i(w)$$

η est le pas d'itération. On l'appelle fréquemment *Taux d'apprentissage*.

En apprentissage, le nombre n de données utilisées peut être très grand (des millions de données...). Dans ce cas le calcul de la somme des gradients peut prendre un temps prohibitif.

Dans la méthode de descente du gradient stochastique (SGD), on procède à une simplification drastique. Au lieu de calculer le gradient du risque empirique ∇Q exactement, chaque itération estime ce gradient sur la base d'une seule mesure. L'algorithme réalise ainsi une mise à jour après chaque exemple.

— choix de w_0 et de η
  — for i=1 :n do
    choisir aléatoirement un exemple noté $l(i)$
    $w_{n+1} = w_n - \mu \nabla Q_{l(i)}(w)$
  — fin do

2 Mise en oeuvre sur un exemple jouet

Nous allons tester cette méthode sur un exemple très simple du calcul du minimum d'une fonction constituée de deux termes $f = \frac{1}{2}(f_1 + f_2)$. Au lieu de calculer le gradient de f et d'appliquer directement une descente de gradient, nous allons utiliser SGD et ainsi choisir aléatoirement une des deux fonctions et appliquer la descente de gradient soit sur f_1 soit sur f_2 en fonction du tirage effectué. La structure grossière de l'algo est écrite dans Algorithme 1).

Appliquer cet algorithme avec $f_1(x) = (x+1)^2$ et $f_2(x) = (x-1)^2$. Visualiser sur un graphique le tableau des x_i générés.

Répéter cette étape avec 5 initialisations différentes et afficher les résultats sur un même graphique.

Ajouter sur ce graphique le résultat obtenu par une descente de gradient *classique*.

Algorithm 1 Minimisation par gradient stochastique

$niter = 1000$
choisir aléatoirement une valeur initiale x_0
for $i = 1 : niter$ **do**
  choisir aléatoirement un nombre u entre 0 et 1
  $numfonc = (u > .5)$
  appliquer une itération de descente de gradient à partir de x_i avec $f1$ ou f_2 en utilisant $numfonc * f_1 + (1 - numfonc) * f_2$. Choisir $\mu = 1/(20 + i)$.
end for

3 Application à la classification logistique

Nous allons nous intéresser maintenant à un problème de classification en deux classes. Les données traitées sur le site (*load('dataBreast.mat')*). Il s'agit à l'origine de plus de n=569 données collectées dans le cadre de traitements du cancer du sein. Chaque ligne correspond à $p = 19$ mesures faites pour chaque patiente. Ces données ont été centrées réduites (le vérifier). Le vecteur y contient le caractère invasif (valeur 1) ou bénin (valeur -1) de la tumeur.

On cherche ici à construire un classifieur linéaire pour séparer les tumeurs bénignes et invasives. Dans l'espace R^p, on cherche un hyperplan défini par $w_1, ...w_p$ séparant au mieux les données. Si les données étaient parfaitement séparables, on pourrait trouver $w \in R^p$ avec $< w, x(i) > \geq 0$ pour les données bénignes et $< w, x(i) >< 0$ pour les données invasives. Ces données ne sont malheureusement pas parfaitement séparables... Et on va donc chercher à maximiser la qualité de la classification.

On cherche ainsi à déterminer w minimisant

$$Q(w) = \frac{1}{n} \sum_{i=1^n} S(< w, x(i) >, y(i))$$

où $x(i)$ est le vecteur de R^p concernant la i^{ieme} patiente et S est la fonction définie par $S(s, y) = log(1 + e^{-sy})$.

1. Tracer la fonction $log(1 + exp(-x))$ sur l'ensemble des réels positifs.
2. Justifier que la minimisation de Q conduit à séparer au mieux les classes étiquetées -1 ou 1 (noter que $(< w, x(i) >$ et $y(i)$ sont de meme signe si la classification est correcte).
3. Dans un premier temps nous allons déterminer la solution avec une méthode de **descente de gradient classique**. Il nous faut donc calculer ∇Q. Vérifier analytiquement que les fonctions suivantes dont le code est donné en matlab permettent bien de calculer ∇Q.
   L = @(s,y) 1/n * sum(log(1 + exp(-s.*y))) ;
   Q = @(w,X,y) L(X*w,y) ;
   theta = @(v)1 ./ (1+exp(-v)) ;
   nablaL = @(s,y)- 1/n * y.* theta(-s.*y) ;
   nablaQ = @(w,X,y)X'*nablaL(X*w,y) ;
   En vous servant de ces fonctions, implémenter la descente de gradient c-a-d une suite effectuant

   $$w_{k+1} = w_k - \mu \nabla Q(w_k, X, y)$$

   Utiliser $\mu = .5$ pour la descente de gradient. Afficher l'évolution de la suite w_k au cours des itérations (faire 5000 itérations). Stocker les valeurs de $Q(w_k)$ dans un tableau. Cela sera utile pour la suite pour comparer la convergence.

4. Nous allons maintenant nous intéresser à l'algorithme de **descente de gradient stochastique (SGD)**. Ceci est utile lorsqu'on a beaucoup de données à traiter. Comme dans le cas de la question 1, nous allons pour chaque itération k choisir aléatoirement et selon une loi uniforme une mesure $l(k)$ parmi $[1...n]$ qui sera utilisée pour approcher le gradient. Une étape du SGD sera alors

$$w_{k+1} = w_k - \mu_k \nabla Q_{l(k)}(w_k)$$

avec $\nabla Q_i(w) = S(< w, x_i >, y(i))$

— Vérifier par le calcul que l'expression du gradient est bien
nablaQi = @(w,i)-y(i) .* X(i, :)' * theta(-y(i) * (X(i, :)*w)) ;

— Le choix de μ_k est sensible. Il doit tendre vers 0 quand k est grand, mais il ne doit cependant par tendre trop vite vers 0 pour obtenir la convergence. Adopter le choix suivant de $\mu_k = \frac{\mu_0}{1+\frac{k}{k_0}}$ avec $k_0 = 100$ et $\mu_O = .05$

— implanter SGD avec les valeurs proposées pour μ. Afficher l'évolution de $Q(w_k)$ au cours du temps. Relancer plusieurs fois (10 par exemple) l'algorithme et afficher le résultat sur le même graphique. S'agissant d'un algorithme probabiliste, vous ne devriez pas trouver la même trajectoire. Comparer les valeurs de w.

— Expliquer pourquoi la complexité d'une itération de descente de gradient est en $\mathcal{O}(np)$ et celle d'une itération de SGD en $\mathcal{O}(p)$. Avec cette constatation, superposer grossièrement la courbe de minimisation avec une descente de gradient classique et celle avec un gradient stochastique sur les 5000 premières itérations du gradient stochastique. Qu'en concluez vous sur l'efficacité du SGD ? Voir l'article de L. Bottou en référence pour plus de renseignements sur les prorpiétés de ces algorithmes.

— faites varier les valeurs de μ_0 et de k_0 et regarder l'influence sur la convergence

4 Références

Stochastic Gradient Descent Tricks. Léon Bottou.Voir pdf ici

5 Annexe : dérivation de fonctions composées

Soient f et g deux fonctions de R dans R. On note $g \circ f$ la fonction définie par $g \circ f(x) = g(f(x))$.
Soient $g(x) = x^2$ et $f(x) = log(x)$. Alors $g \circ f(x) = (log(x))^2$ et $f \circ g(x) = log(x^2) = 2log(x)$.
Dérivation d'une fonction composée :
Si f est dérivable au point x et g est dérivable au point $f(x)$ alors$g \circ f$ est déribale en x et

$$(g \circ f)'(x) = g'(f(x)) \times f'(x)$$

Dans l'exemple précedent :
$(g \circ f)'(x) = 2f(x) * f'(x) = 2log(x) * 1/x$
$f \circ g(x) = 1/g(x) * g'(x) = 1/x^2 * 2x = 2/x$

Ce théorème s'étend au cas de fonctions dans des espace R^p par l'intermédiaire des matrices jacobiennes. Ainsi, si $f : R^n \to R^m$ ey $g : R^m \to R^p$.

Alors $g \circ f : R^n \to R^p$. Si on note $J_f|_x$ la jacobienne de f calculée au point x, alors

$$J_{g \circ f}|_x = J_g|_{f(x)} \times J_f|_x$$