Stochastic Gradient Descent with Momentum and Line Searches

David Nardi
MSc student in Artificial Intelligence, Univeristy of Florence

3rd February 2024

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Contents

1 Introduction

1.1 Classification task

1.2 Optimization problem

Given the following dataset

$$\mathcal{D} = \{(x^{(i)}, y^{(i)}) \mid x^{(i)} \in \mathcal{X}, y^{(i)} \in \mathcal{Y}, i = 1, 2, \dots, N\}$$

the general Machine Learning optimization problem for the supervised learning subset is formulated as follows

$$\min_{w \in \mathbb{R}^p} f(w) = L(w) + \lambda \Omega(w) = \begin{cases} L(w) = \frac{1}{N} \sum_{i=1}^N \ell_i(w) \\ \Omega_{\ell_2} = \|w\|_2^2 \end{cases}$$

where $L(w)$ is called *loss function*, and it is a finite-sum, and $\Omega(w)$ it's the *regularization term* with its coefficient λ. There are three regularization possible choices, the ℓ_2 regularization was chosen for this problem.

The task performed is the binary classification, using the Logistic Regression model. Every machine learning model has its own loss function, the logistic regression uses the *log-loss*, for one single dataset sample $\ell_i = \log\bigl(1 + \exp(-y^{(i)} w^T x^{(i)})\bigr)$, follows the resulting optimization problem

$$\min \frac{1}{N} \sum_{i=1}^{N} \log\bigl(1 + \exp(-y^{(i)} w^T x^{(i)})\bigr) + \lambda \|w\|^2 \tag{1}$$

where $i = 1, \dots, N$ are the dataset samples, $\mathcal{Y} = \{-1, 1\}$ is the set of the possible values for the response variable, corresponding to the negative or positive class, $\mathcal{X} \subseteq \mathbb{R}^{(p+1)}$ are dataset examples, $(p+1)$ means that there are p features and the intercept. The $1/N$ term isn't always used, we choose to use that term for scaling issues.

We defined the matrix associated to the dataset as follows

$$X^T = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_p^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_p^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \dots & x_p^{(N)} \end{pmatrix} \in \mathbb{R}^{N \times (p+1)} \qquad x^{(i)} = \begin{pmatrix} 1 \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_N^{(i)} \end{pmatrix} \in \mathbb{R}^{(p+1)}$$

$$\nabla f(w) = \frac{1}{N} X^T r + 2\lambda w, \quad r_i = -y^{(i)} \sigma(-y^{(i)} w^T x^{(i)})$$

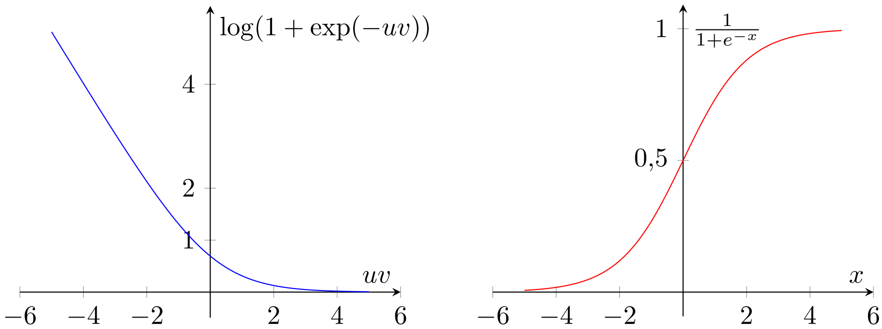
$$\nabla^2 f(w) = \frac{1}{N} X D X^T + 2\lambda I, \quad d_{ii} = \sigma(y^{(i)} w^T x^{(i)}) \sigma(-y^{(i)} w^T x^{(i)})$$

$r \in \mathbb{R}^N$, $D \in \mathbb{R}^{N \times N}$

Proposition 1. *Problem* (1) *admits a unique optimal solution.*

$$w = \begin{pmatrix} b \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{pmatrix} \in \mathbb{R}^{(p+1)}$$

$$y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{pmatrix} \in \{-1, 1\}$$



(a) *Log-loss* **(b)** *Sigmoid function*

Figure 1: Log-loss and sigmoid function plots

- $uv \gg 0$: the example is labelled correctly
- $uv \ll 0$: the class assigned to the example is the wrong one
- the hessian matrix is positive defined $\forall w$, this means that the objective function, which is quadratic, is coercive and for the continuity that function admits global minimum, so $f(w)$ has finite inferior limit
- the hessian matrix being positive defined implies also that the objective function is strictly convex (on the other hand the loss function is just convex, due to its hessian matrix being positive semi-defined), this implies that if the global minimum exists, that solution is unique
- a global minimum is a point that satisfy $\nabla f(w^*) = 0$, which is a sufficient condition implied by the convexity of the problem, see figure 1a on the preceding page
- the ℓ_2 regularization implies that the objective function is strongly convex, this speeds up the convergence
- further more we can assume that $\nabla f(w)$ is Lipschitz-continuous with constant L

2 Mini-batch gradient descent variants

see algorithm 1

2.1 Fixed step-size

Algorithm 1: Mini-batch Gradient Descent with fixed or decreasing step-size

```
given $w^0 \in \mathbb{R}^n$, $k = 0$ e $\{\alpha_k\} \mid \alpha_k = \alpha \vee \alpha_k = \frac{\alpha_0}{k+1}$
while ($\|\nabla f(w^k)\| > \varepsilon(1 + |f(w)|)$)
 shuffle $\{1, \ldots, N\}$ and split $B_1, \ldots, B_{N/M}$ such that $1 < |B_t| = M \ll N$
 set $y_0 = w^k$
 for $t = 1, \ldots, N/M$
  get mini-batch indices from $B_t$
  approximate true gradient $\nabla f_{i_t}(w) = \frac{1}{M} \sum_{j \in B_t} \nabla f_j(y_{t-1})$
  compute direction $d_t = -\nabla f_{i_t}(y_{t-1})$
  make a (internal) step $y_t = y_{t-1} + \alpha_k d_t$
 end for
 update weights $w^{k+1} = y_{N/M}$
 epoch ends $k = k + 1$
end while
```

2.2 Stochastic line search

Algorithm 2: Mini-batch Gradient Descent with Armijo line search

```
given $w^0 \in \mathbb{R}^{p+1}$, $\gamma \in (0,1)$, $\delta \in (0,1)$, $\alpha_0 \in \mathbb{R}^+$
$k = 0$
while ($\|\nabla f(w^k)\| > \varepsilon(1 + |f(w)|)$)
 shuffle $\{1, \ldots, N\}$ and split $B_1, \ldots, B_{N/M}$ such that $1 < |B_t| = M \ll N$
 set $y_0 = w^k$
 for $t = 1, \ldots, N/M$
  get mini-batch indices $i_t$ from $B_t$
  approximate true gradient $\nabla f_{i_t}(w) = \frac{1}{M} \sum_{j \in B_t} \nabla f_j(y_{t-1})$
  compute direction $d_t = -\nabla f_{i_t}(y_{t-1})$
  $\alpha = \mathrm{reset}()$, $q = 0$
  compute potential next step $y_t = y_{t-1} + \alpha d_t$
  while ($f_{i_t}(y_t) > f_{i_t}(y_{t-1}) + \gamma\alpha\nabla f_{i_t}(y_{t-1})^T d_t$)
   reduce step-size $\alpha = \delta\alpha$
   rejections counter $q = q + 1$
  end while
  set optimal mini-batch step-size $\alpha_t = \alpha$
  make a (internal) step $y_t = y_{t-1} + \alpha_t d_t$
 end for
 update weights $w^{k+1} = y_{N/M}$
 epoch ends $k = k + 1$
end while
```

2.3 Fixed momentum term

Algorithm 3: Mini-batch Gradient Descent with fixed Momentum term and fixed step-size

```
given w^0 ∈ R^{p+1}, {α_k} = α, {β_k} = β ∈ (0,1)
k = 0
while (‖∇f(w^k)‖ > ε(1 + |f(w)|))
 shuffle {1,...,N} and split B_1,...,B_{N/M} such that 1 < |B_t| = M ≪ N
 set y_0 = w^k, d_0 = 0
 for t = 1,...,N/M
  get mini-batch indices i_t from B_t
  approximate true gradient ∇f_{i_t}(w) = (1/M) Σ_{j∈B_t} ∇f_j(y_{t-1})
  compute direction d_t = −((1 − β)∇f_{i_t}(y_{t-1}) + βd_{t-1})
  make a (internal) step y_t = y_{t-1} + α_k d_t
 end for
 update weights w^{k+1} = y_{N/M}
 epoch ends k = k + 1
end while
```

Algorithm 4: Mini-batch Gradient Descent with Armijo line search for step-size and Momentum correction

```
given w^0 ∈ R^{p+1}, γ ∈ (0,1), δ_a ∈ (0,1), α_0 ∈ R^+, δ_m ∈ (0,1), β_0 ∈ (0,1)
k = 0
while (‖∇f(w^k)‖ > ε(1 + |f(w)|))
 shuffle {1,...,N} and split B_1,...,B_{N/M} such that 1 < |B_t| = M ≪ N
 set y_0 = w^k, d_0 = 0
 for t = 1,...,N/M
  get mini-batch indices i_t from B_t
  approximate true gradient ∇f_{i_t}(w) = (1/M) Σ_{j∈B_t} ∇f_j(y_{t-1})
  compute potential next direction d_t = −((1 − β)∇f_{i_t}(y_{t-1}) + βd_{t-1})
  q_m = 0
  while (∇f_{i_t}(y_{t-1})^T d_t ≥ 0)
   reduce momentum term β = δ_m β
   rejections counter q_m = q_m + 1
  end while
  set optimal mini-batch momentum term β_t = β
  α = reset(), q_a = 0
  compute potential next step y_t = y_{t-1} + αd_{t-1}
  while (f_{i_t}(y_t) > f_{i_t}(y_{t-1}) + γα∇f_{i_t}(y_{t-1})^T d_t)
   reduce step-size α = δ_a α
   rejections counter q_a = q_a + 1
  end while
  set optimal mini-batch step-size α_t = α
  make a (internal) step y_t = y_{t-1} + α_t d_t
 end for
update weights w^{k+1} = y_{N/M}
epoch ends k = k + 1
end while
```

Algorithm 5: Mini-batch Gradient Descent with Armijo line search for step-size and Momentum restart

```
given w^0 ∈ R^{p+1}, γ ∈ (0,1), δ_a ∈ (0,1), α_0 ∈ R^+, δ_m ∈ (0,1), {β_k} = β ∈ (0,1)
k = 0
while (‖∇f(w^k)‖ > ε(1 + |f(w)|))
 shuffle {1,...,N} and split B_1,...,B_{N/M} such that 1 < |B_t| = M ≪ N
 set y_0 = w^k, d_0 = 0
 for t = 1,...,N/M
  get mini-batch indices i_t from B_t
  approximate true gradient ∇f_{i_t}(w) = (1/M) Σ_{j∈B_t} ∇f_j(y_{t-1})
  compute potential next direction d_t = −((1 − β)∇f_{i_t}(y_{t-1}) + βd_{t-1})
  if (∇f_{i_t}(y_{t-1})^T d_t ≥ 0)
   restart direction d_t = d_0
  end if
  α = reset(), q_a = 0
  compute potential next step y_t = y_{t-1} + αd_{t-1}
  while (f_{i_t}(y_t) > f_{i_t}(y_{t-1}) + γα∇f_{i_t}(y_{t-1})^T d_t)
   reduce step-size α = δ_a α
   rejections counter q_a = q_a + 1
  end while
  set optimal mini-batch step-size α_t = α
  make a (internal) step y_t = y_{t-1} + α_t d_t
 end for
 update weights w^{k+1} = y_{N/M}
 epoch ends k = k + 1
end while
```

3 Experiments