Optimization Methods for Machine Learning
Variants of Gradient method for multilayer perceptron

Laura Palagi

http://www.dis.uniroma1.it/~palagi

Dipartimento di Ingegneria informatica automatica e gestionale A. Ruberti
Sapienza Università di Roma

Via Ariosto 25

SAPIENZA
UNIVERSITÀ DI ROMA

Gradient methods

$$\min_{\omega \in \mathbb{R}^d} E(\omega) = \frac{1}{P} \sum_{p=1}^{P} E_p$$

$$\omega^{k+1} = \omega^k + \eta^k d^k$$

- BP **batch**

$$d^k = -\nabla E(\omega^k) = -\frac{1}{P} \sum_{p=1}^{P} \nabla E_p(\omega^k)$$

- BP **mini-batch**

$$d^k = -\frac{1}{|\mathscr{B}^k|} \sum_{i \in \mathscr{B}^k} \nabla E_i(\omega^k) \text{ for some } \mathscr{B}^k \subset \{1, \ldots . P\}$$

- BP **on-line**

$$d^k = -\nabla E_{p^k}(\omega^k) \text{ for some } p^k \in \{1, \ldots . P\}$$

Convergence of gradient method for the regularized error

- Batch - constant stepsize $\eta^k = \bar{\eta}$
- Online - diminishing stepsize $\eta^k \to 0$ (costant stepsize: convergence to a limit cycle)

Practical and theoretical Comparison

METHOD	UPDATE speed	MEMORY requirement	ONLINE learning
BATCH	slow	high	No
MINI BATCH	medium	medium	Yes
ON LINE	high	low	Yes

If you have a small training set, use batch gradient descent ($P < 200$)

Comments cont'd

- Practical behaviour
  - online methods achieve in the early iterations a much faster convergence than the non incremental batch gradient
  - In the last iterations accuracy is worst and convergence slower
  - If some error in the accuracy of the solution can be tolerated, the online method is a good option.
  - the stepsize is fixed constant small or gradually decreased up to a small enough value such that the oscillation is of no concern.
  - or the degree of incrementalism is reduced over the time so that ultimately it becomes non incremental

[Y. Bengio, Practical Recommendations for Gradient-Based Training of Deep Architectures] `https://arxiv.org/pdf/1206.5533v2.pdf`

In practice

Widely used modifications to improve performance:

1. Momentum term
2. Iteration averaging
3. Diagonal scaling

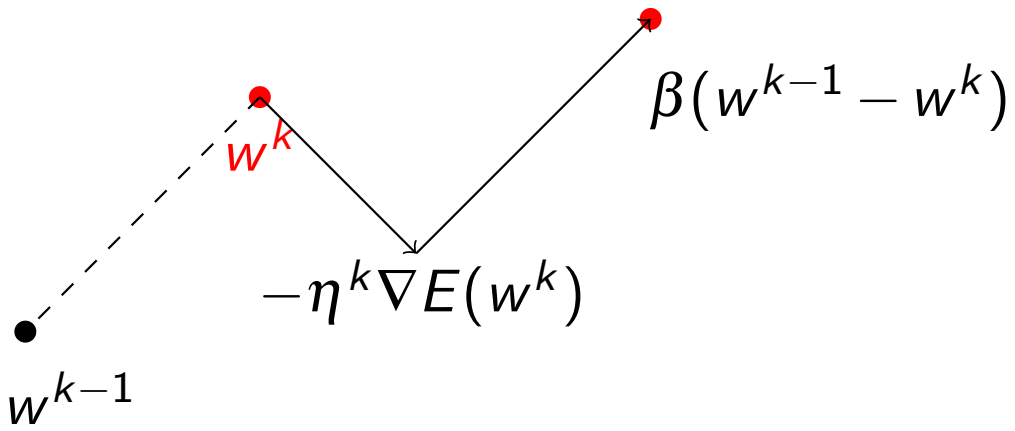
Momentum term

The momentum term represents an extrapolation along the difference of the two preceding iterates. It has been introduced for FG (*heavy ball method*) and extended to IGD/SGD.

$$w^{k+1} = w^k - \eta^k \nabla E(w^k) + \beta^k (w^k - w^{k-1})$$

with $w^{-1} = w^0$.

where $\beta > 0$ is a given scalar with (typical values $= 0.8 \pm 0.9$).



Momentum ≈ Averaging gradient

Momentum iteration
If $\beta^k = \beta$ and $\eta^k = \eta$

$$w^{k+1} = w^k - \eta \sum_{\ell=0}^{k-1} (\beta)^\ell \nabla E_{p^{k-\ell}}(w^{k-\ell})$$

Momentum uses a *geometric weighting* (exponentially decaying average) of the previous gradients.

Steps accumulate contributions in direction of persistent descent and tend to cancel directions of oscillations.

Fundamental role of momentum together with good starting point has been showed in Deep Networks [2].

Iterate averaging

SGD generates noisy iterates that tend to oscillate around the minimizers. Compute a corresponding *averaged iteration* (Polyak-Ruppert) that would automatically possess less noisy behavior

$$\overline{w}^k = \frac{1}{k+1} \sum_{j=0}^{k} w^j$$

or more in general $\overline{w}^k = 1/q^k \sum_{j=0}^{q^k} w^j$

- it allows longer stepsize; stepsizes diminishing at a slower rate $O\left(1/\sqrt{k}\right)$
- it does not deteriorate rate of convergence
- some the adverse effects caused by ill-conditioning disappear.

Diagonal scaling

- choosing a proper learning rate is difficult
- learning rate schedule adjust the learning rate but are unable to adapt to the dataset
- same learnig rate applies to all parameters

Adaptive learning rate methods
Diagonal scaling of the direction

$$d^k = A^k \begin{cases} -\nabla E(\omega^k), & \text{if batch} \\ -\nabla E_{\mathscr{B}}(\omega^k), & \text{mini batch.} \end{cases}$$

where A^k is a diagonal $d \times d$ matrix and

$$\omega^{k+1} = \omega^k + \eta d^k$$

Adagrad

$$A_{ii}^k = \frac{1}{\sqrt{\sum_{t=0}^{k-1} \nabla_{\omega_i} E(\omega^t)^2 + \varepsilon}}$$

$\varepsilon \tilde{1} 0^{-8}$ and he upedate is

$$\omega_i^{k+1} = \omega_i^k - \eta \frac{1}{\sqrt{\sum_{t=0}^{k-1} \nabla_{\omega_i} E(\omega^t)^2 + \varepsilon}} \nabla_{\omega_i} E(\omega^k)$$

Instead of using al the square norm of the preceding gradient, it uses only past K gradients and defines un updating decaying average as a fraction of the preceding accumulated square gradients

Adadelta/RMSPROP

$$A_{ii}^k = \gamma A_{ii}^{k-1} + (1-\gamma)\nabla_{\omega_i} E(\omega^t)^2 + \varepsilon$$

$\gamma \tilde{1}0^{-9}$ and he update is

$$\omega_i^{k+1} = \omega_i^k - \eta \frac{1}{\sqrt{\sum_{t=0}^{k-1} \nabla_{\omega_i} E(\omega^t)^2 + \varepsilon}} \nabla_{\omega_i} E(\omega^k)$$

Adam

Momentum modified direction

$$d^k = \frac{1}{1-\beta_a^k}\left(\beta_1^k d^{k-1} + (1-\beta_1^k)\nabla E(\omega^k)\right)$$

$$A_{ii}^k = \frac{1}{1-\beta_2^k}\left(\beta_2^k A_{ii}^{k-1} + (1-\beta_2^k)\nabla_{\omega_i} E(\omega^k)^2\right)$$

Nice slide at Ruder's page `https://www.slideshare.net/SebastianRuder/optimization-for-deep-learning`

Mark Schmidt, Nicolas Le Roux, and Francis Bach.
Minimizing finite sums with the stochastic average gradient.
Mathematical Programming, 162(1-2):83–112, 2017.

Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton.

On the importance of initialization and momentum in deep learning.
ICML (3), 28:1139–1147, 2013.