Optimization for machine/deep learning

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function. If $x^\star$ is a local extremum then $\nabla f(x^\star) = 0$, where $\nabla f(x^\star)$ is the gradient of the function f at point $x^\star$:

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_d} \right)^T .$$

For instance, if $f : x \mapsto \langle a; x \rangle$, then $\nabla f(x) = a$ and if $f : x \mapsto x^T A x$ then $\nabla f(x) = (A + A^T)x$. The gradient is orthogonal to level sets where, for a function $f : \mathbb{R}^d \to \mathbb{R}$ and a value $c_\star \in \mathbb{R}$, the level set associated with $c_\star$ is given by $\mathcal{C}_{c_\star} = \{x \in \mathbb{R}^d, f(x) = c_\star\}$.

Deep learning optimization problems may be written as (see previous session for logistic regression and the multilayer perceptron):

$$\text{argmin}_{w \in \mathbb{R}^d} \left\{ f(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, \langle w; X_i \rangle) + \lambda g(w) \right\} ,$$

where ℓ is a given loss function and g is a penalization function such as $g : w \mapsto \|w\|_2^2$ or $g : w \mapsto \|w\|_1^2$. The objective of gradient based optimization algorithms is to produce a sequence of parameter estimates $(w_k)_{k \geqslant 0}$ which approximate this minimum.

Convergence of gradient descent algorithm

Starting from an initial vector $w^{(0)}$ and with a step size $\eta > 0$, the most simple gradient based algorithm writes, for all $k \geqslant 0$,

$$w^{(k+1)} = w^{(k)} - \eta \nabla f(w^{(k)}) .$$

If $f : \mathbb{R}^d \to \mathbb{R}$ is a L-smooth convex function, i.e. a convex function such that for all $(x, y) \in \mathbb{R}^d \times \mathbb{R}^d$, $\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|$ and let $w^\star$ be a minimum of f on $\mathbb{R}^d$. Then, Gradient Descent with step size $\eta \leqslant 1/L$ satisfies

$$f(w^{(k)}) - f(w^\star) \leqslant \frac{\|w^{(0)} - w^\star\|_2^2}{2\eta k} .$$

In particular, for $\eta = 1/L$,

$$L\|w^{(0)} - w^\star\|_2^2 / 2\varepsilon$$

iterations are sufficient to get an ε-approximation of the minimal value of f. The proof of this result relies on the descent lemma: if f is L-smooth, then for any $w, w' \in \mathbb{R}^d$

$$f(w') \leqslant f(w) + \langle \nabla f(w); w' - w \rangle + \frac{L}{2} \|w - w'\|_2^2 .$$

Assuming this holds, remark that

$$\text{argmin}_{w \in \mathbb{R}^d} \{ f(w^k) + \langle \nabla f(w^k); w - w^k \rangle + \frac{L}{2} \|w - w^k\|_2^2 \} = \text{argmin}_{w \in \mathbb{R}^d} \|w - (w^k - \frac{1}{L} \nabla f(w^k))\|_2^2 .$$

Hence, it is natural to choose

$$w^{k+1} = w^k - \frac{1}{L} \nabla f(w^k) .$$

Faster rate for strongly convex function

A function $f : \mathbb{R}^d \to R$ is μ-strongly convex if

$$x \mapsto f(x) - \frac{\mu}{2}\|x\|_2^2$$

is convex. If f is differentiable it is equivalent to assume that, for all $x \in \mathbb{R}^d$,

$$\lambda_{min}(\nabla^2 f(x)) \geqslant \mu\,,$$

where $\lambda_{min}(\nabla^2 f(x))$ is the smallest eigenvalue of $\nabla^2 f(x)$ which is the Hessain matrix of f at x ($d \times d$ matrix containing second order derivative of f at x). This is also equivalent to assume that, for all $x, y \in \mathbb{R}^d$,

$$f(y) \geqslant f(x) + \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}\|y - x\|_2^2.$$

If $f : \mathbb{R}^d \to \mathbb{R}$ is a L-smooth, μ strongly convex function and $w^\star$ is a minimum of f on $\mathbb{R}^d$, then, Gradient Descent with step size $\eta \leqslant 1/L$ satisfies

$$f(w^{(k)}) - f(w^\star) \leq \Big(1 - \eta\mu\Big)^k \|f(w^{(0)}) - f(w^\star)\|_2^2.$$

Choice of η

In practice, the algorithm performance is highly sensitive to the choice of η.

1. **Exact line search**. At each step, choose the best η by optimizing

   $$\eta^{(k)} = \operatorname*{argmin}_{\eta > 0} f(w - \eta \nabla f(w))\,.$$

   The computational cost of this approach makes it prohibitive.

2. **Backtracking line search**. First, fix a parameter $0 < \beta < 1$, then at each iteration, start with $\eta^{(k)} = 1$ and while

   $$f(w^{(k)} - \eta^{(k)} \nabla f(w^{(k)})) > f(w^{(k)}) - \frac{\eta^{(k)}}{2}\|\nabla f(w^{(k)})\|^2,$$

   update $\eta^{(k)} = \beta\eta^{(k)}$. This simple to implement approach is widespread and efficient in practice. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a L-smooth convex function and $w^\star$ be a minimum of f on $\mathbb{R}^d$. Then, Gradient Descent with backtracking line search satisfies

   $$f(w^{(k)}) - f(w^\star) \leqslant \frac{\|w^{(0)} - w^\star\|_2^2}{2k \min(1, \beta/L)}\,.$$

Stochastic gradient descent

In the case of machine/deep learning applications, the function to be minimized is of the form:

$$f : w \mapsto \frac{1}{n}\sum_{i=1}^n \ell(Y_i, \langle w; X_i\rangle) + \lambda g(w) = \frac{1}{n}\sum_{i=1}^n f_i(w)\,.$$

Previous methods are based on **full gradients**, since at each iteration they require to compute

$$\nabla f(w) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(w)\,,$$

which depends on the whole dataset. When processing very large datasets (n is large), highly prohibitive computational cost before doing a **unique step** towards the minimum. Stochastic

Gradient Descent algorithm aims at reducing this computational cost by replacing this gradient by an unbiased estimate. If I choose uniformly at random $I \in \{1, \dots, n\}$, then

$$\mathbb{E}[\nabla f_I(w)] = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w) = \nabla f(w) \,.$$

Therefore $\nabla f_I(w)$ is an **unbiased** estimate of the full gradient $\nabla f(w)$. The computation of $\nabla f_I(w)$ only requires the I-th line of data and its complexity is of order $O(d)$ Starting from $w^{(0)}$, and chosing the step size η_k at each time step, Stochastic Gradient Descent algorithm proceeds as follows. For all $k \geqslant 1$, pick at random (uniformly) I_k in $\{1, \dots, n\}$ and set

$$w^{(k)} = w^{(k-1)} - \eta_k \nabla f_{i_k}(w^{(k-1)}) \,.$$

Each iteration has complexity $O(d)$ instead of $O(nd)$ for the previous full gradient methods. Consider the projected stochastic gradient descent algorithm on the ball $B(0, R)$ with $R > 0$ fixed. Assume that f is convex and that there exists $M > 0$ satisfying, for all $x \in B(0, R)$,

$$\|\nabla f_i(x)\| \leqslant M$$

and assume that all minima of f belong to $B(0, R)$. Then, setting $\eta_k = 2R/(b\sqrt{k})$,

$$\mathbb{E}\Big[f\Big(\frac{1}{k}\sum_{j=1}^{k} w^{(j)}\Big)\Big] - f(w^\star) \leqslant \frac{3Rb}{\sqrt{k}} \,.$$

In the case where f is μ strongly convex, then, setting $\eta_k = 2/(\mu(k+1))$ yields

$$\mathbb{E}\Big[f\Big(\frac{2}{k(k+1)}\sum_{j=1}^{k} j w^{(j-1)}\Big)\Big] - f(w^\star) \leqslant \frac{2b^2}{\mu(k+1)} \,.$$

Stochastic Gradient Descent performance may be improved using variance reduction techniques, see for instance Stochastic Variance Reduced Gradient (SVRG) (Johnson et al. 2013), Stochastic Average Gradient (SAG) (Schmidt et al., 2017) and SAGA (Defazio et al., 2014).

Nesterov Accelerated Gradient Descent

Momentum based gradient descent algorithms aim at taking into account the previous updates as additional velocity to avoid convergence to local minima. Starting from $w^{(0)}$ and an initial velocity $v^{(0)} = 0$ and using a sequence of step sizes $(\eta_k)_{k\geqslant 0}$ and momentum $(\beta_k)_{k\geqslant 0}$ in $(0, 1)$, the Nesterov Accelerated Gradient Descent proceeds as follows:

$$\begin{aligned} v^{(k+1)} &= w^{(k)} - \eta \nabla f(w^{(k)}) \,, \\ w^{(k+1)} &= v^{(k+1)} + \beta_{k+1}(v^{(k+1)} - v^{(k)}) \,. \end{aligned}$$

If the target function f is a L-smooth, convex and if $w^\star$ is a minimum of f then, if $\beta_{k+1} = k/(k+3)$,

$$f(w^{(k)}) - f(w^\star) \leqslant \frac{2\|w^{(0)} - w^\star\|_2^2}{\eta(k+1)^2} \,.$$

When in addition f is μ strongly convex, choosing

$$\beta_k = \frac{1 - \sqrt{\mu/L}}{1 + \sqrt{\mu/L}} \,,$$

yields

$$f(w^{(k)}) - f(w^\star) \leqslant \frac{\|w^{(0)} - w^\star\|_2^2}{\eta} \Big(1 - \sqrt{\frac{\mu}{L}}\Big)^k \,.$$

It can be proved that the Nesterov Accelerated Gradient Descent reaches the optimal bounds among optimization algorithms such that for all $k \geqslant 0$,

$$w^{(k)} \in w^{(0)} + \text{Span}(\nabla f(w^{(0)}), \ldots, \nabla f(w^{(k-1)})).$$

This means that it is possible to design a L-smooth convex function such that any algorithm of this kind has an error term greater than $(k+1)^{-2}$ up to a multiplicative constant after k step of the algorithm.

Optimization methods for neural networks

AdaGrad The ADAptive GRADient algorithm introduced by (Duchi et al. 2011) starts from $w^{(0)}$ and uses a learning rate $\eta > 0$ and a momentum α and defines, for all $k \geqslant 0$ and all $j \in \{1, \ldots, d\}$,

$$w_j^{(k+1)} \leftarrow w_j^{(k)} - \frac{\eta}{\sqrt{\sum_{\tau=1}^{k} (\nabla f(w^{(\tau)}))_j^2}} (\nabla f(w^{(k)}))_j \,.$$

The rationale of this method is that different rates are used for all coordinates which is crucial for neural networks in which gradient at different layers can be of different order of magnitudes. It is proved in (Ward et al., 2018) that AdaGrad achieves the same convergence rate as gradient descent with optimal fixed stepsize up to a log factor. The adaptive step size grows with the inverse of the gradient magnitudes, so that large gradients have small learning rates and small gradients have large learning rates.

AdaDelta Introduced in (Zeiler, 2012), was introduced to reduce the sensitivity to initial conditions of AdaGrad. Indeed, if the initial gradients are large, the learning rates of AdaGrad will be low for all updates which can be overcome by increasing η, but making the AdaGrad method highly sensitive to the choice of η.

RMSprop optimizer Unpublished method, from the course of Geoff Hinton.

ADAM: Adaptive moment estimation Introduced in (Kingma et al., 2014) and considered as the state of the art to otpimize neural networks, the ADAM procedure update the parameter estimate as follows. Starting from $m_0 = 0$ and $v_0 = 0$ and choosing $\beta_1, \beta_2, \eta, \varepsilon \in (0, 1)$, compute first and second moment estimate

$$m_k = \beta_1 m_{k-1} + (1 - \beta_1) \nabla f(w^{(k)}) \quad \text{and} \quad v_k = \beta_2 v_{k-1} + (1 - \beta_2)(\nabla f(w^{(k)}))^2 \,,$$

then, compute the correction terms

$$\hat{m}_k = \frac{m_k}{1 - \beta_1^k} \quad \hat{v}_k = \frac{v_k}{1 - \beta_2^k} \,,$$

and update the parameter estimate with

$$w^{(k+1)} = w^{(k)} - \frac{\eta}{\sqrt{\hat{v}_k} + \varepsilon} \hat{m}_k \,.$$

First convergence results can be found in (Kingma et al., 2014) and examples where ADAM algorithm does not converge to the optimum are given in (Reddi et al., 2018). Recent analysis by (Barakat et al., 2018).