

Optimization for Machine Learning

CS-439

Lecture 11: Gradient free and adaptive methods

Nicolas Flammarion

EPFL – github.com/epfml/OptML_course

May 16, 2025

Chapter XI.1

Zero-Order Optimization

Look mom no gradients!

Can we optimize $\min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x})$ without access to gradients?

meet the newest fanciest optimization algorithm,...

Random search

pick a random direction $\mathbf{d}_t \in \mathbb{R}^d$

$\gamma := \operatorname{argmin}_{\gamma \in \mathbb{R}} f(\mathbf{x}_t + \gamma \mathbf{d}_t)$ (line-search)

$\mathbf{x}_{t+1} := \mathbf{x}_t + \gamma \mathbf{d}_t$

Convergence Rate for Derivative-free Random Search

Theorem: Converges same as gradient descent - up to a slow-down factor d .

Proof. Assume that f is a L -smooth convex, differentiable function. For any γ , by smoothness, we have:

$$f(\mathbf{x}_t + \gamma \mathbf{d}_t) \leq f(\mathbf{x}_t) + \gamma \mathbf{d}_t^\top \nabla f(\mathbf{x}_t) + \frac{\gamma^2 L}{2} \|\mathbf{d}_t\|^2$$

Minimizing the upper bound, there is a step size $\bar{\gamma}$ for which

$$f(\mathbf{x}_t + \bar{\gamma} \mathbf{d}_t) \leq f(\mathbf{x}_t) - \frac{1}{L} \left(\frac{\mathbf{d}_t^\top}{\|\mathbf{d}_t\|} \nabla f(\mathbf{x}_t) \right)^2$$

The step size γ we actually took (based on f directly) can only be better:

$$f(\mathbf{x}_t + \gamma \mathbf{d}_t) \leq f(\mathbf{x}_t + \bar{\gamma} \mathbf{d}_t) .$$

Taking expectations, and using the Lemma $\mathbb{E}_{\mathbf{r}}(\mathbf{r}^\top \mathbf{g})^2 = \frac{1}{d} \|\mathbf{g}\|^2$ for $\mathbf{r} \sim \text{sphere} \subseteq \mathbb{R}^d$:

$$\mathbb{E}[f(\mathbf{x}_t + \gamma \mathbf{d}_t)] \leq \mathbb{E}[f(\mathbf{x}_t)] - \frac{1}{Ld} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] .$$

Convergence Rate for Derivative-free Random Search

Same as what we obtained for [gradient descent](#),
now with an **extra factor of d** . d can be huge!!!

Can do the same for different function classes, as before

- ▶ For convex functions, we get a rate of $\mathcal{O}(dL/\varepsilon)$.
- ▶ For strongly convex, we get $\mathcal{O}(dL/\mu \log(1/\varepsilon))$.

Always d times the complexity of gradient descent on the function class.

credits to Moritz Hardt

Without the Linear Search: Two Function Evaluations

Without the line search, when a function can be evaluated at two points per iteration, a gradient estimate can be obtained by

$$g_{\alpha}(\mathbf{x}) = \frac{f(\mathbf{x} + \alpha \mathbf{u}) - f(\mathbf{x})}{\alpha} \mathbf{u} \quad (1)$$

where \mathbf{u} is a random direction sample from the normal distribution $N(0, I_d)$. This can be then used as a gradient estimate in the gradient descent update.

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t g_{\alpha}(\mathbf{x}_t) \quad (2)$$

A simplified analysis: We show a preliminary analysis of the algorithm at the limit of $\alpha \rightarrow 0$. Assuming the function f is differentiable, we have,

$$g_0(\mathbf{x}) = \left(\nabla f(\mathbf{x})^{\top} \mathbf{u} \right) \mathbf{u} \quad (3)$$

Simplified Analysis of the Two Function Evaluation

First note that the gradient estimate is unbiased, i.e.,

$$\mathbb{E}[g_0(\mathbf{x}_t) \mid \mathbf{x}_t = \mathbf{x}] = \nabla f(\mathbf{x}). \quad (4)$$

Thus Eq. (2) is equivalent to stochastic gradient descent. Next, note second moment can be bounded as follows:

$$\begin{aligned} \mathbb{E}[\|g_0(\mathbf{x}_t)\|^2 \mid \mathbf{x}_t = \mathbf{x}] &= \mathbb{E}[\|\mathbf{u}\|^2 \nabla f(\mathbf{x}_t)^\top \mathbf{u} \mathbf{u}^\top \nabla f(\mathbf{x}_t) \mid \mathbf{x}_t = \mathbf{x}] \\ &= \nabla f(\mathbf{x})^\top \mathbb{E}[\|\mathbf{u}\|^2 \mathbf{u} \mathbf{u}^\top] \nabla f(\mathbf{x}). \end{aligned}$$

For $u \sim N(0, I_d)$, $\mathbb{E}[\|\mathbf{u}\|^2 \mathbf{u} \mathbf{u}^\top] \preceq (d+3)I_d$. Using this the second moment writes:

$$\mathbb{E}[\|g_0(\mathbf{x}_t)\|^2 \mid \mathbf{x}_t = \mathbf{x}] \leq (d+3)\|\nabla f(\mathbf{x})\|^2. \quad (5)$$

In the case of stochastic gradient descent, we have studied the convergence with stochastic gradient oracle with $E[\|g(\mathbf{x}_t)\|^2] \leq B$. Here, we have a *better* control on the second moment which we use in the following analysis.

Analysis of the Two Function Evaluation Method

For a L -smooth convex function f with optimum at \mathbf{x}_* , using the vanilla analysis,

$$\begin{aligned}\mathbb{E}[\|\mathbf{x}_{t+1} - \mathbf{x}_*\|^2] &= \mathbb{E}[\|\mathbf{x}_t - \mathbf{x}_*\|^2] - 2\eta_t \mathbb{E}\left[g_0(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{x}_*)\right] + \eta_t^2 \mathbb{E}[\|g_0(\mathbf{x}_t)\|^2] \\ &\leq \mathbb{E}[\|\mathbf{x}_t - \mathbf{x}_*\|^2] - 2\eta_t \mathbb{E}\left[\nabla f(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{x}_*)\right] + (d+3)\eta_t^2 \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2],\end{aligned}$$

where the inequality follows from the gradient estimate (4) and the second moment bound (5).

Analysis of the Two Function Evaluation Method

Smoothness implies $\|\nabla f(\mathbf{x}_t)\|^2 \leq 2L(f(\mathbf{x}_t) - f(\mathbf{x}_*))$, with convexity of f , we have $f(\mathbf{x}_t) - f(\mathbf{x}_*) \leq \nabla f(\mathbf{x}_t)^\top (\mathbf{x}_t - \mathbf{x}_*)$.

$$\begin{aligned}\mathbb{E}[\|\mathbf{x}_{t+1} - \mathbf{x}_*\|^2] &\leq \mathbb{E}[\|\mathbf{x}_t - \mathbf{x}_*\|^2] - 2\eta_t(1 - (d+3)\eta_t L)\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{x}_*)], \\ 2\eta_t(1 - (d+3)\eta_t L)\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{x}_*)] &\leq \mathbb{E}[\|\mathbf{x}_t - \mathbf{x}_*\|^2] - \mathbb{E}[\|\mathbf{x}_{t+1} - \mathbf{x}_*\|^2].\end{aligned}$$

Summing over t ,

$$\sum_{t=0}^T 2\eta_t(1 - (d+3)\eta_t L)\mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{x}_*)] \leq \|\mathbf{x}_0 - \mathbf{x}_*\|^2 - \mathbb{E}[\|\mathbf{x}_{T+1} - \mathbf{x}_*\|^2].$$

With a stepsize $\eta_t = \eta < \frac{1}{(d+3)L}$, we get a rate of convergence of $\mathcal{O}(d/T)$.

(Extra) Beyond Simplified Analysis

For the general case of gradient estimate $g_\alpha(\mathbf{x})$, a similar analysis can be following by computing the expectation and second moment as

$$\begin{aligned}\mathbb{E}[g_\alpha(\mathbf{x})] &= \nabla f(\mathbf{x}) + \mathcal{O}(\alpha), \\ \mathbb{E}[\|g_\alpha(\mathbf{x})\|^2] &\leq \mathcal{O}(\|\nabla f(\mathbf{x})\|^2 + \alpha^2).\end{aligned}$$

The analysis can be followed as before with the correction terms involving α . For a careful analysis, refer to the paper [DJWW15].

Applications for Derivative-free Random Search

Applications

- ▶ can be used for **Reinforcement learning**
- ▶ memory and communication advantages: never need to store a gradient
- ▶ hyperparameter optimization, and other difficult problems like discrete optimization problems
- ▶ finding adversarial examples

Reinforcement Learning

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_t).$$

where \mathbf{s}_t is the **state** of the system, \mathbf{a}_t is the control **action**, and \mathbf{e}_t is some random **noise**. We assume that f is fixed, but unknown.

We search for a control 'policy'

$$\mathbf{a}_t := \pi(\mathbf{a}_1, \dots, \mathbf{a}_{t-1}, \mathbf{s}_0, \dots, \mathbf{s}_t).$$

which takes a trajectory of the dynamical system and outputs a new control action.
Want to maximize overall **reward**

$$\begin{aligned} \max_{\mathbf{a}_t} \mathbb{E}_{\mathbf{e}_t} \left[\sum_{t=0}^N R_t(\mathbf{s}_t, \mathbf{a}_t) \right] \\ \text{s.t. } \mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_t) \\ (\mathbf{s}_0 \text{ given}) \end{aligned}$$

Examples: Simulations, Games (e.g. Atari), Alpha Go

Chapter XI.2

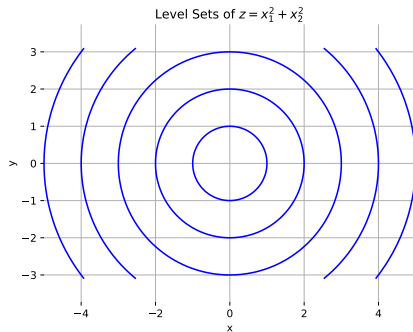
Adaptive Methods

Some problems with GD

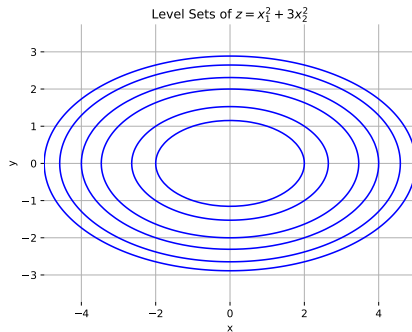
Consider the following function:

$$f(\mathbf{x}) = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 \end{bmatrix} \begin{bmatrix} \mu & 0 \\ 0 & L \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}$$

The level sets of the functions for different choices of μ and L can be seen as follows:



(a) Circular level sets when $L, \mu = 1$



(b) Elliptical level sets when $L, \mu = 3, 1$

Some problems with GD

- ▶ For skewed functions, $\mu \ll L$, the level sets are ellipses, in these cases GD can be very slow along some directions.
- ▶ A solution to this problem is to use a **preconditioner** P .

Preconditioned Gradient Descent

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t P_t \nabla f(\mathbf{x}_t),$$

for some semi-definite matrix P_t which is often called as preconditioner.

Intuition: Consider a function $g(\mathbf{y}) = f(R\mathbf{y})$, for some matrix R , now gradient descent on g is given by:

$$\begin{aligned}\mathbf{y}_{t+1} &= \mathbf{y}_t - \eta_t \nabla g(\mathbf{y}_t) = \mathbf{y}_t - \eta_t R^\top \nabla f(R\mathbf{y}_t), \\ R\mathbf{y}_{t+1} &= R\mathbf{y}_t - \eta_t R R^\top \nabla f(R\mathbf{y}_t) \\ \mathbf{x}_{t+1} &= \mathbf{x}_t - \eta_t R R^\top \nabla f(\mathbf{x}_t).\end{aligned}$$

Hence, we can choose $P_t = R R^\top$, and the preconditioned gradient descent is equivalent to the gradient descent on the function $g(\mathbf{y}) = f(R\mathbf{y})$.

How to choose the preconditioner?

- ▶ The optimal preconditioner is the inverse of the Hessian, $P = (\nabla^2 f(\mathbf{x}))^{-1}$ (similar to the Newton method).
- ▶ However, for a function f in d -dimensional space, the Hessian is a $d \times d$ matrix, making its inversion computationally expensive and often impractical.
- ▶ As a result, practical approaches rely on approximations, such as using a diagonal matrix to estimate the Hessian.

Adagrad

Adagrad is an adaptive variant of SGD.

Pick a stochastic gradient \mathbf{g}_t . For all i ,

$$\text{Update } [G_t]_i := \sum_{s=0}^t ([\mathbf{g}_s]_i)^2$$

$$\text{Diagonal preconditioner: } P_t = \begin{bmatrix} \sqrt{[G_t]_1} & 0 & \cdots & 0 \\ 0 & \sqrt{[G_t]_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{[G_t]_d} \end{bmatrix}^{-1},$$

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma P_t \mathbf{g}_t, \implies [\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \frac{\gamma}{\sqrt{[G_t]_i}} [\mathbf{g}_t]_i.$$

Adagrad

- ▶ chooses an **adaptive, coordinate-wise** learning rate
- ▶ To select the preconditioner, intuitively first the Hessian is approximated by the sum of gradient outer product matrices, $G_t = \sum_{s=0}^t \mathbf{g}_s \mathbf{g}_s^\top$ and then only the diagonal entries are used.
- ▶ strong performance in practice; theoretical guarantees for convergence in convex setting with better performance than GD especially for sparse gradients [DHS11].
- ▶ However, Adagrad often leads to diminishing learning rates as the aggregates square gradients, G_t , quickly become large.
- ▶ Variants: Adadelata, RMSprop, **Adam**

RMSprop is a variant of Adagrad that uses an exponential moving average with a parameter β of the squared gradients instead of the sum.

Pick a stochastic gradient \mathbf{g}_t . For all i ,

$$\text{Update } [G_t]_i := (\beta)[G_{t-1}]_i + (1 - \beta)([\mathbf{g}_t]_i)^2$$

$$\mathbf{x}_{t+1} := [\mathbf{x}_t]_i - \frac{\gamma}{\sqrt{[G_t]_i}}[\mathbf{g}_t]_i.$$

Momentum SGD

Before presenting Adam, let's discuss another momentum variant of SGD, a classical technique proposed by Polyak in 1964 to accelerate gradient descent.

pick a stochastic gradient \mathbf{g}_t

$$\mathbf{m}_{t+1} := \beta \mathbf{m}_t + (1 - \beta) \mathbf{g}_t \quad (\text{momentum term})$$

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma \mathbf{m}_{t+1}$$

(standard choice of $\mathbf{g}_t := \nabla f_j(\mathbf{x}_t)$ for sum-structured objective functions $f = \sum_j f_j$)

- ▶ momentum from previous gradients
- ▶ is a variant of the Nesterov acceleration seen before
- ▶ key element of deep learning optimizers, necessary for top accuracy

Polyak Momentum and Nesterov Acceleration

Polyak Momentum

At iteration t ,

$$\mathbf{m}_{t+1} = \beta \mathbf{m}_t + \eta \nabla f(\mathbf{x}_t),$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{m}_{t+1}$$

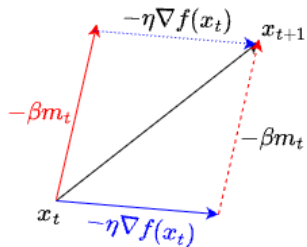


Figure: Aggregating the momentum and the gradient

Nesterov Acceleration

At iteration t ,

$$\mathbf{m}_{t+1} = \beta \mathbf{m}_t + \eta \nabla f(\mathbf{x}_t - \beta \mathbf{m}_t),$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{m}_{t+1}$$

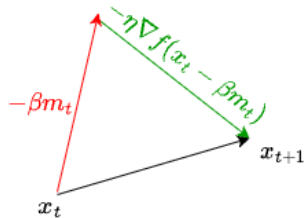


Figure: 'Look ahead' gradient. Add the momentum and take a gradient step from the new position.

Convergence of Momentum and Nesterov Acceleration

- ▶ For **quadratic** functions, both methods converge at an accelerated rate of $\exp\{-t/\sqrt{\kappa}\}$, where κ is the condition number of the function.
- ▶ For **strongly convex** functions, Nesterov acceleration converges at a rate of $\exp\{-t/\kappa\}$, while Polyak momentum does not provably converge at a faster rate.

Adam

Adam is a momentum variant of Adagrad

pick a stochastic gradient \mathbf{g}_t

$$\mathbf{m}_t := \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t \quad (\text{momentum term})$$

$$[\mathbf{v}_t]_i := \beta_2 [\mathbf{v}_{t-1}]_i + (1 - \beta_2) ([\mathbf{g}_t]_i)^2 \quad \forall i \quad (\text{2nd-order statistics})$$

$$[\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \frac{\gamma}{\sqrt{[\mathbf{v}_t]_i}} [\mathbf{m}_t]_i \quad \forall i$$

- ▶ faster forgetting of older weights
- ▶ momentum from previous gradients (see acceleration)
- ▶ (simplified version, without correction for initialization of $\mathbf{m}_0, \mathbf{v}_0$)
- ▶ strong performance in practice, e.g. for self-attention based networks like transformers

Convergence Issues with Adam

- ▶ While RMSprop and Adam solve the problem of diminishing learning rates, they can lead to convergence issues particularly due to the "short-term" memory of the exponential moving average of the squared gradients.
- ▶ EMA of the squared gradients approximately limits the update to only a few past gradients. Hence, particularly in the case where a few minibatches provide large gradients, their influence dies out quickly due to the EMA.

Convergence Issues with Adam

Problem Setup: Consider the functions $f_t(x) = \begin{cases} Cx & \text{for } t \bmod 3 = 1, \\ -x & \text{otherwise.} \end{cases}$, for $C > 2$

over the domain $\mathcal{F} = [-1, 1]$. We are interested in the following quantity (regret) over the course of optimization

$$R_T = \frac{1}{T} \sum_{t=1}^T (f_t(x_t)) - \frac{1}{T} \min_{x \in \mathcal{F}} \sum_{t=1}^T f_t(x)$$

Note that $x = -1$ gives the minimum regret, i.e., $\min_{x \in \mathcal{F}} \sum_{t=1}^T f_t(x)$. However, Adam provably converges to highly suboptimal $x = +1$ [RKK18].

Convergence Issues with Adam

Note the gradient of the function $f_t(x)$ is given by:

$$\nabla f_t(x) = \begin{cases} C & \text{for } t \bmod 3 = 1, \\ -1 & \text{otherwise.} \end{cases},$$

i.e., one large gradient every 3 iterations. However, the EMA of squared gradients nullify this large gradient. It can be shown that for the update of Adam,

Lemma

Consider the Adam algorithm with appropriate choice of β_1 and β_2 starting with $x_1 = 1$, then

$$x_t = 1 \text{ for every } t \text{ such that } t \bmod 3 = 1$$

Proof can be done by induction following [RKK18].

Convergence Issues with Adam

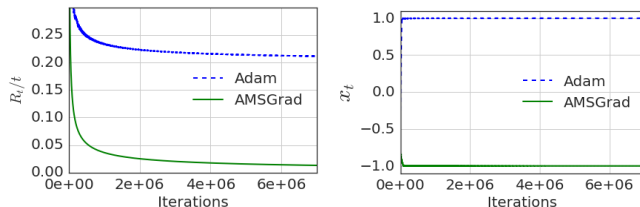


Figure: Non-convergence of ADAM on a simple one dimensional convex problem $f_t(x)$. The second plot shows that ADAM converges to $x = 1$. AMSGrad is the algorithm proposed in [RKK18]. Image is taken from [RKK18].

Chapter XI.3

Efficient variants of SGD

SignSGD

Only use the sign (one bit) of each gradient entry:

SignSGD is a communication efficient variant of SGD.

pick a stochastic gradient \mathbf{g}_t

$$[\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \gamma_t \text{sign}([\mathbf{g}_t]_i) \quad \forall i$$

(with possible rescaling of γ_t with $\|\mathbf{g}_t\|_1$)

- ▶ communication efficient for distributed training
- ▶ convergence issues

ClippedSGD




Clip the gradients to a predefined maximum length $c > 0$.

$$\begin{aligned} &\text{pick a stochastic gradient } \mathbf{g}_t \\ &\mathbf{x}_{t+1} := \mathbf{x}_t - \gamma_t \text{clip}_c(\mathbf{g}_t) \quad \forall i \end{aligned}$$

with $\text{clip}_c(\mathbf{g}) := \min\left(1, \frac{c}{\|\mathbf{g}\|}\right) \cdot \mathbf{g}$

- ▶ used to avoid instabilities in deep learning training (such as LLMs)
- ▶ used with differential privacy (adding noise of comparable magnitude)
- ▶ convergence non-trivial

Bibliography

-  John Duchi, Elad Hazan, and Yoram Singer.
Adaptive subgradient methods for online learning and stochastic optimization.
Journal of machine learning research, 12(7), 2011.
-  John C Duchi, Michael I Jordan, Martin J Wainwright, and Andre Wibisono.
Optimal rates for zero-order convex optimization: The power of two function evaluations.
IEEE Transactions on Information Theory, 61(5):2788–2806, 2015.
-  Sashank J Reddi, Satyen Kale, and Sanjiv Kumar.
On the convergence of adam and beyond.
In *ICLR*, 2018.