# Machine Learning II
# A crash course on Optimization

Le Thi Khanh Hien
UMONS, thikhanhhien.le@umons.ac.be

Mons, March 2023

# Table of content

# I. Accelerated Proximal Gradient Method

**References:**

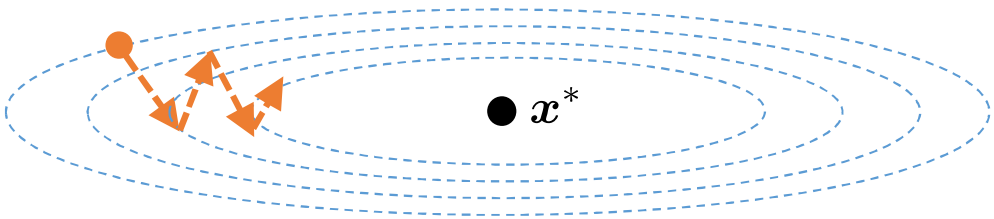
- N Parikh, S Boyd, "Proximal Algorithms", Foundations and Trends in Optimization 1(3), 2014. Link
- Amir Beck, "First-Order Methods in Optimization", MOS-SIAM Series on Optimization, 2017

# Accelerated gradient descent

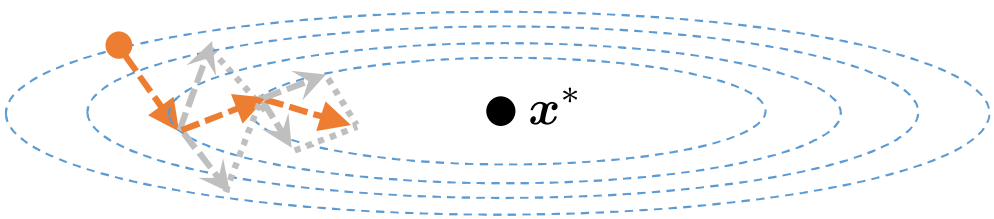
## Heavy ball method[1]

$$x^{k+1} = x^k - \lambda_k \nabla f(x^k) + \gamma_k (x^k - x^{k-1}).$$

(Photo credit: Yuxin Chen)




gradient descent




heavy-ball method

[1] B. Polyak. Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics, 4(5) : 1–17, 1964

## Nesterov accelerated gradient method [2]

$$y^k = x^k + \gamma_{k-1}(x^k - x^{k-1})$$
$$x^{k+1} = y^k - \lambda_k \nabla f(y^k)$$

Choice of extrapolation parameter:

- $\gamma_k = \frac{k}{k+3}$.
- $t_0 = 1, t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$, $\gamma_k = \frac{t_k - 1}{t_{k+1}}$.

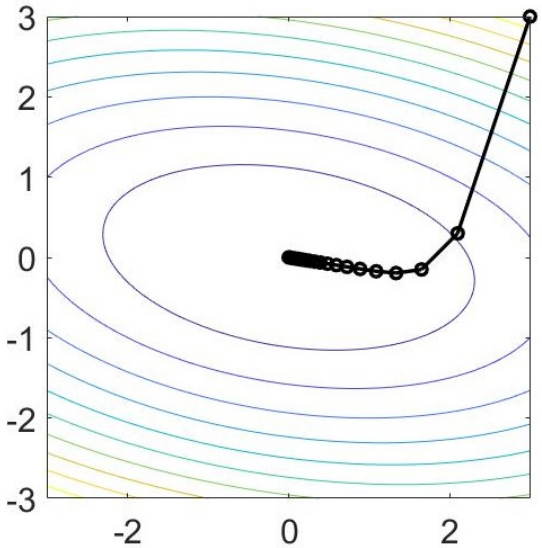
Other forms of accelerated gradient method:

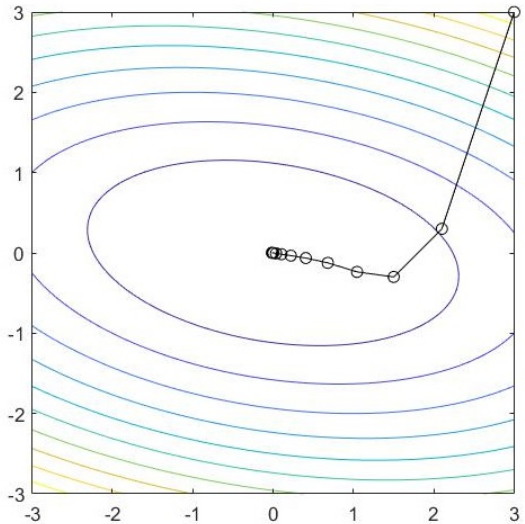
- Y. Nesterov. Smooth minimization of non-smooth functions. Math. Prog., 103(1):127–152, 2005.
- P. Tseng. On accelerated proximal gradient methods for convex-concave optimization, 2008.
- Y. Nesterov, Lectures on Convex Optimization, Springer Optimization and Its Applications book series, 2018.

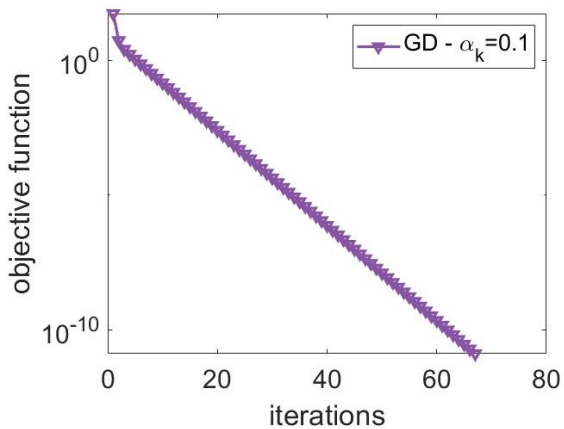
[2] [Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$, Soviet Mathematics Doklady, 27(2), 1983].

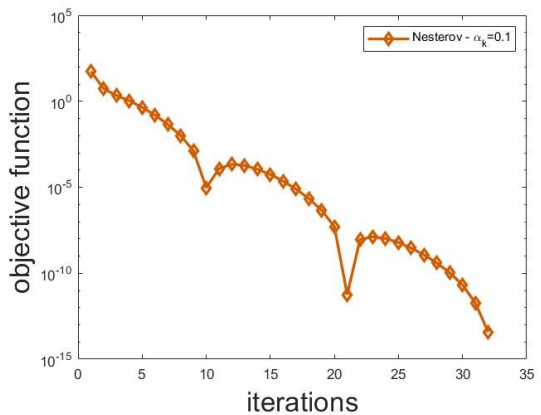
# Accelerated gradient descent

Example problem: $\min_{x \in \mathbb{R}^2} f(x_1, x_2) = x_1^2 + x_1 x_2 + 4x_2^2.$



# Problem setting

We consider the same convex composite optimization problem

$$\min_{x \in \mathbb{E}} F(x) = f(x) + g(x),$$

where $f : \mathbb{E} \to \mathbb{R}$ is a differentiable convex function and $g : \mathbb{E} \to \overline{\mathbb{R}}$ is a proper lower-semicontinuous convex function.

## Assumptions

- $f$ is $L$-smooth, which is equivalent to $x \mapsto \frac{L}{2}\|x\|^2 - f(x)$ is convex.
- There is a constant $\mu \geq 0$ such that $x \mapsto f(x) - \frac{\mu}{2}\|x\|^2$ is convex. When $m > 0$, we say $f$ is $\mu$-strongly convex.
- The optimal value $F^*$ is attained at $x^*$.

# Accelerated proximal gradient method - FISTA

**FISTA** [3]

$$t_0 = 1, t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$$

$$y^k = x^k + \frac{t_{k-1} - 1}{t_k}(x^k - x^{k-1}), x^{k+1} = \textbf{prox}_{\lambda_k g}(y^k - \lambda_k \nabla f(y^k)).$$

**Choice of step size.**

- Constant step size $\lambda_k = \frac{1}{L}$.
- Back-tracking line search invoking on the extrapolation point $y^k$: start at some $\lambda = \lambda^0$ and back-track $\lambda = C\lambda^0$, with $0 < C < 1$, until the following inequality holds

$$f\left(y^k - \lambda G_\lambda(y^k)\right) \leq f(y^k) - \lambda \left\langle \nabla f(y^k), G_\lambda(y^k) \right\rangle + \frac{\lambda}{2} \left\| G_\lambda(y^k) \right\|^2.$$

[3] [A. Beck and M. Teboulle. A Fast Iterative Shrinkage - Thresholding Algorithm for Linear Inverse Problems, SIAM Journal on Imaging Sciences, 2: 183–202, 2009.]

- Convergence rate $O(1/k^2)$ of FISTA. We have

$$F(x^k) - F^* \leq \frac{2L}{\alpha(k+1)^2} \left\|x^0 - x^*\right\|^2,$$

where $\alpha = 1$ in the constant stepsize choice and $\alpha = \min\{L\lambda^0, C\}$ in the back-tracking line search.

- Convergence for strongly convex problems.

$$y^k = x^k + \frac{\sqrt{L/\mu} - 1}{\sqrt{L/\mu} + 1}(x^k - x^{k-1}), x^{k+1} = \mathbf{prox}_{\frac{1}{L}g}(y^k - \frac{1}{L}\nabla f(y^k)).$$

Denote $\kappa = \frac{L}{\mu}$. We have

$$F(x^k) - F^* \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right)^k \left(F(x^0) - F^* + \frac{\mu}{2}\left\|x^0 - x^*\right\|^2\right).$$

# Summary

Table: Convergence rate and iteration complexity of accelerated proximal gradient method for convex $L$-smooth composite problem

| | Stepsize | Convergence rate | Iteration complexity |
|---|---|---|---|
| Convex | $\lambda_k = \frac{1}{L}$ | $O\left(\frac{1}{k^2}\right)$ | $O\left(\frac{1}{\sqrt{\varepsilon}}\right)$ |
| $m$-strongly convex | $\lambda_k = \frac{1}{L}$ | $O\left(\left(1 - \frac{1}{\sqrt{\kappa}}\right)^k\right)$ | $O\left(\sqrt{\kappa} \log \frac{1}{\varepsilon}\right)$ |

# Summary

Table: Convergence rate and iteration complexity of accelerated proximal gradient method for convex $L$-smooth composite problem

| | Stepsize | Convergence rate | Iteration complexity |
|---|---|---|---|
| Convex | $\lambda_k = \frac{1}{L}$ | $O\left(\frac{1}{k^2}\right)$ | $O\left(\frac{1}{\sqrt{\varepsilon}}\right)$ |
| $m$-strongly convex | $\lambda_k = \frac{1}{L}$ | $O\left(\left(1 - \frac{1}{\sqrt{\kappa}}\right)^k\right)$ | $O\left(\sqrt{\kappa} \log \frac{1}{\varepsilon}\right)$ |

Convergence rate and complexity of proximal gradient methods?

# II. Stochastic gradient method

## References:

- L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization Methods for Large-Scale Machine Learning", SIAM Review 2018.
- `https://leon.bottou.org/projects/sgd`

## Stochastic optimization

$$\min_x F(x) := E[f(x, \xi)],$$

where $\xi$ is a random variable with probability distribution $P$.

## Stochastic optimization

$$\min_x F(x) := E[f(x, \xi)],$$

where $\xi$ is a random variable with probability distribution $P$.

- Directly computing $E[f(x, \xi)]$ is not possible for most problems
- Let $\xi_i$, $i = 1, \ldots, n$ be independent, identically distributed realizations of $\xi$.

$$E[f(x, \xi)] \approx \frac{1}{n} \sum_{i=1}^{n} f(x, \xi_i)$$

## Stochastic optimization

$$\min_x F(x) := E[f(x, \xi)],$$

where $\xi$ is a random variable with probability distribution $P$.

- Directly computing $E[f(x, \xi)]$ is not possible for most problems
- Let $\xi_i$, $i = 1, \ldots, n$ be independent, identically distributed realizations of $\xi$.

$$E[f(x, \xi)] \approx \frac{1}{n} \sum_{i=1}^{n} f(x, \xi_i)$$

## Sample Average Approximation (SAA) Method:

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f(x, \xi_i)$$

## Stochastic optimization

$$\min_x F(x) := E[f(x, \xi)],$$

where $\xi$ is a random variable with probability distribution $P$.

- Directly computing $E[f(x, \xi)]$ is not possible for most problems
- Let $\xi_i$, $i = 1, \ldots, n$ be independent, identically distributed realizations of $\xi$.

$$E[f(x, \xi)] \approx \frac{1}{n} \sum_{i=1}^{n} f(x, \xi_i)$$

## Sample Average Approximation (SAA) Method:

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f(x, \xi_i)$$

Implementations:

- Sample-then-optimize: collect enough examples then solve an approximated deterministic optimization problem.

- Sample-and-optimize-concurrently: collect a sample set $S^k$ of a few samples of $\xi$, then at iteration $k$ update

$$g_k = \frac{1}{|S^k|} \sum_{i \in S^k} \nabla f(x^k, \xi_i), \quad x^{k+1} = x^k - \alpha_k g_k.$$

## Optimization problems in large-scale machine learning

- Minimizing expected risk/loss

$$\min_w E[\ell(h(x; w), y)],$$

where $(x, y)$ is a given input-output pair, $h(x; w)$ and $y$ are the predicted and true outputs, and $\ell(h(x; w), y)$ is the loss.

- Minimizing the empirical risk

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i; w), y_i).$$

## Optimization problems in large-scale machine learning

- Minimizing expected risk/loss

$$\min_w E[\ell(h(x; w), y)],$$

where $(x, y)$ is a given input-output pair, $h(x; w)$ and $y$ are the predicted and true outputs, and $\ell(h(x; w), y)$ is the loss.

- Minimizing the empirical risk

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i; w), y_i).$$

### Finite sum problem

$$\min_x F(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

## Optimization problems in large-scale machine learning

- Minimizing expected risk/loss

$$\min_w E[\ell(h(x; w), y)],$$

  where $(x, y)$ is a given input-output pair, $h(x; w)$ and $y$ are the predicted and true outputs, and $\ell(h(x; w), y)$ is the loss.

- Minimizing the empirical risk

$$\min_w \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i; w), y_i).$$

## Finite sum problem

$$\min_x F(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

## Example

- Linear regression: $f_i(x) = (a_i^T x - b_i)^2$.
- Logistic regression: $f_i(x) = \log(1 + \exp(b_i a_i^T x))$.

We consider the finite sum problem

$$\min_{x \in \mathbb{R}^d} F(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x).$$

When $n$ is large, evaluating $\nabla F(x)$ could be very expensive.

We consider the finite sum problem

$$\min_{x\in\mathbb{R}^d} F(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x).$$

When $n$ is large, evaluating $\nabla F(x)$ could be very expensive.

## Stochastic Gradient Method

- Initial point $x^0 \in \mathbb{R}^d$.
- For $k = 0, 1 \ldots$, choose $i_k \in [n]$ uniformly at random and update

$$x^{k+1} = x^k - \alpha_k \nabla f_{i_k}(x^k).$$

We consider the finite sum problem

$$\min_{x\in\mathbb{R}^d} F(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x).$$

When $n$ is large, evaluating $\nabla F(x)$ could be very expensive.

## Stochastic Gradient Method

- Initial point $x^0 \in \mathbb{R}^d$.
- For $k = 0, 1 \ldots$, choose $i_k \in [n]$ uniformly at random and update

$$x^{k+1} = x^k - \alpha_k \nabla f_{i_k}(x^k).$$

- SG estimates gradient using only one $f_{i_k}$ function. $\nabla f_{i_k}(x^k)$ is an unbiased estimate of $\nabla F(x^k)$:

$$\mathrm{E}[\nabla f_{i_k}(x^k)] =$$

We consider the finite sum problem

$$\min_{x\in\mathbb{R}^d} F(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x).$$

When $n$ is large, evaluating $\nabla F(x)$ could be very expensive.

## Stochastic Gradient Method

- Initial point $x^0 \in \mathbb{R}^d$.
- For $k = 0, 1 \ldots$, choose $i_k \in [n]$ uniformly at random and update

$$x^{k+1} = x^k - \alpha_k \nabla f_{i_k}(x^k).$$

- SG estimates gradient using only one $f_{i_k}$ function. $\nabla f_{i_k}(x^k)$ is an unbiased estimate of $\nabla F(x^k)$:

$$\mathrm{E}[\nabla f_{i_k}(x^k)] =$$

- SG is not always descent in general (stochastic gradient descent would be a misleading name).

# How to choose step-size?

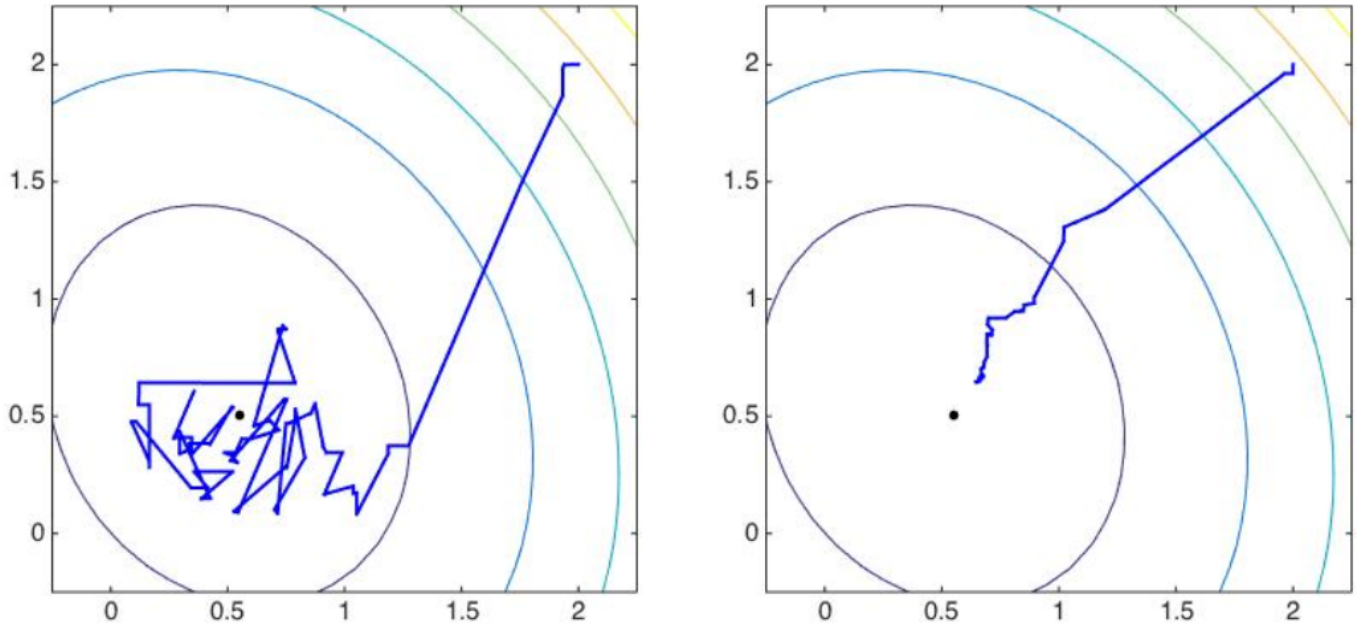



Figure: SG run with a fixed stepsize (left) vs. diminishing stepsizes (right)

The image is from F. Curtis, L. Bottou, J. Nocedal, "Beyond SG: Noise Reduction and Second-Order Methods," ICML 2016.

- For fixed step-size, SG generates a sequence approaching a noise ball that is proportional to the step-size .
- Intuition: use diminishing step-sizes (larger step-sizes initially and gradually decrease the step-sizes).

## Diminishing step-sizes

- An example of choosing step-size: $\alpha_k = \dfrac{1}{a(k_0 + k)}$ , where $a$ and $k_0$ are tuning parameters, see e.g., `https://scikit-learn.org/stable/modules/sgd.html`
- Common diminishing step-size choice is $O(1/k)$.
- Under some certain assumptions, the following condition is sufficient to have some convergence guarantee:

$$\sum_{k=1}^{\infty} \alpha_k = \infty, \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty.$$

Let us do a convergence analysis.

# Convergence analysis

## Proposition 2.1

*Suppose $F$ is $L$-smooth then we have*

$$\mathrm{E}_{i_k}[F(x^{k+1})] - F(x^k) \leq -\alpha_k \|\nabla F(x^k)\|^2 + \frac{1}{2}\alpha_k^2 L \mathrm{E}_{i_k}[\|\nabla f_{i_k}(x^k)\|^2]$$

# Convergence analysis

## Proposition 2.1

*Suppose $F$ is $L$-smooth then we have*

$$\mathrm{E}_{i_k}[F(x^{k+1})] - F(x^k) \leq -\alpha_k \|\nabla F(x^k)\|^2 + \frac{1}{2}\alpha_k^2 L \mathrm{E}_{i_k}[\|\nabla f_{i_k}(x^k)\|^2]$$

Proof. By the descent lemma, we have

$$F(x^{k+1}) - F(x^k) \leq \nabla F(x^k)^\top (x^{k+1} - x^k) + \frac{L}{2}\|x^{k+1} - x^k\|_2^2.$$

Hence,

$$F(x^{k+1}) - F(x^k) \leq -\alpha_k \nabla F(x^k)^\top \nabla f_{i_k}(x^k) + \frac{L}{2}\alpha_k^2 \|\nabla f_{i_k}(x^k)\|^2.$$

By taking expectation with respect to $i_k$ and noting that $\mathrm{E}_{i_k} \nabla f_{i_k}(x^k) = \nabla F(x^k)$, we obtain the result.

## Proposition 2.2

*Suppose $F$ is $L$-smooth, bounded from below and*

$$\mathrm{E}_{i_k}[\|\nabla f_{i_k}(x)\|^2] \leq \sigma^2,$$

*for some constant $\sigma$ and all $x$. Then we have*

$$\min_{k=0,1,\ldots,T-1} \left\{\mathrm{E}\|\nabla F(x^k)\|^2\right\} \leq \frac{F(x^0) - F^*}{\sum_{k=0}^{T-1} \alpha_k} + \frac{L\sigma^2}{2} \frac{\sum_{k=0}^{T-1} \alpha_k^2}{\sum_{k=0}^{T-1} \alpha_k}.$$

## Proposition 2.2

*Suppose $F$ is $L$-smooth, bounded from below and*

$$\mathrm{E}_{i_k}[\|\nabla f_{i_k}(x)\|^2] \leq \sigma^2,$$

*for some constant $\sigma$ and all $x$. Then we have*

$$\min_{k=0,1,\ldots,T-1} \left\{\mathrm{E}\|\nabla F(x^k)\|^2\right\} \leq \frac{F(x^0) - F^*}{\sum_{k=0}^{T-1} \alpha_k} + \frac{L\sigma^2}{2} \frac{\sum_{k=0}^{T-1} \alpha_k^2}{\sum_{k=0}^{T-1} \alpha_k}.$$

Proof. From Proposition 2.1, we have

$$\alpha_k \|\nabla F(x^k)\|^2 \leq F(x^k) - \mathrm{E}_{i_k}[F(x^{k+1})] + \alpha_k^2 \frac{L\sigma^2}{2}$$

Taking expectation with respect to $i_{k-1}$ and summing up the inequalities from $k = 0$ to $T - 1$, we obtain

$$\sum_{k=0}^{T-1} \alpha_k \mathrm{E}\|\nabla F(x^k)\|^2 \leq \sum_{k=0}^{T-1} \left(\mathrm{E}F(x^k) - \mathrm{E}[F(x^{k+1})]\right) + \sum_{k=0}^{T-1} \alpha_k^2 \frac{L\sigma^2}{2}.$$

The result follows.

# Convergence rates

For convex $L$-smooth function

- Full (batch) gradient descent

$$F(x^k) - F^* = O(1/k).$$

- Stochastic gradient

$$\mathrm{E}[F(x^k)] - F^* = O(1/\sqrt{k}).$$

For strongly convex $L$-smooth function

- Full (batch) gradient descent with suitable step-size

$$F(x^k) - F^* = O(\rho^k), \text{where } \rho \in (0, 1).$$

- Stochastic gradient with suitable assumptions

$$\mathrm{E}[F(x^k)] - F^* = O(1/k).$$

## Stochastic Gradient - general form

- Initial point $x^0 \in \mathbb{R}^d$.
- For $k = 0, 1, \ldots$
  - Generate a realization of random variable $\xi^k$.
  - Compute a stochastic vector $g(x^k, \xi^k)$ ( $g(x^k, \xi^k)$ is a stochastic estimate of $\nabla F(x^k)$).
  - Choose a step-size $\alpha_k$.
  - Update $x^{k+1} = x^k - \alpha_k g(x^k, \xi^k)$.

## Stochastic Gradient - general form

- Initial point $x^0 \in \mathbb{R}^d$.
- For $k = 0, 1, \ldots$
  - Generate a realization of random variable $\xi^k$.
  - Compute a stochastic vector $g(x^k, \xi^k)$ ( $g(x^k, \xi^k)$ is a stochastic estimate of $\nabla F(x^k)$).
  - Choose a step-size $\alpha_k$.
  - Update $x^{k+1} = x^k - \alpha_k g(x^k, \xi^k)$.

Bias:

$$\text{bias}(g(x^k, \xi^k)) := E_{\xi^k}[g(x^k, \xi^k)] - \nabla F(x^k).$$

Variance:

$$\text{Var}(g(x^k, \xi^k)) := E_{\xi^k}\left[\left\|g(x^k, \xi^k)\right\|_2^2\right] - \left\|E_{\xi^k}\left[g(x^k, \xi^k)\right]\right\|_2^2.$$

## Stochastic Gradient - general form

- Initial point $x^0 \in \mathbb{R}^d$.
- For $k = 0, 1, \ldots$
  - Generate a realization of random variable $\xi^k$.
  - Compute a stochastic vector $g(x^k, \xi^k)$ ( $g(x^k, \xi^k)$ is a stochastic estimate of $\nabla F(x^k)$).
  - Choose a step-size $\alpha_k$.
  - Update $x^{k+1} = x^k - \alpha_k g(x^k, \xi^k)$.

Bias:

$$\mathrm{bias}(g(x^k, \xi^k)) := E_{\xi^k}[g(x^k, \xi^k)] - \nabla F(x^k).$$

Variance:

$$\mathrm{Var}(g(x^k, \xi^k)) := E_{\xi^k}\left[\left\|g(x^k, \xi^k)\right\|_2^2\right] - \left\|E_{\xi^k}\left[g(x^k, \xi^k)\right]\right\|_2^2.$$

Let us view SG as a "gradient method with error"

$$x^{k+1} = x^k - \alpha_k(\nabla F(x^k) + e^k), \quad \text{where} \quad e^k = g(x^k, \xi^k) - \nabla F(x^k)$$

Note that $\mathrm{bias}(g(x^k, \xi^k)) = E_{\xi^k}[e^k]$ and $\mathrm{Var}(g(x^k, \xi^k)) = \mathrm{E}_{\xi^k}[\|e^k\|^2]$ for unbiased $g(x^k, \xi^k)$.

Minibatching Instead of using a single element, use an average of several of $f_i$

$$g(x^k, \xi^k) = \frac{1}{|S^k|} \sum_{j \in S^k} \nabla f_j(x^k), \quad x^{k+1} = x^k - \alpha_k g(x^k, \xi^k).$$

- an unbiased estimate of $\nabla F(x)$

$$E\big[\frac{1}{|S^k|} \sum_{j \in S^k} \nabla f_j(x^k)\big] = \nabla F(x^k).$$

- Minibatching is an intermediate approach between an SG and the full gradient. It is particularly useful for parallelization.

Minibatching Instead of using a single element, use an average of several of $f_i$

$$g(x^k, \xi^k) = \frac{1}{|S^k|} \sum_{j \in S^k} \nabla f_j(x^k), \quad x^{k+1} = x^k - \alpha_k g(x^k, \xi^k).$$

- an unbiased estimate of $\nabla F(x)$

$$E\big[\frac{1}{|S^k|} \sum_{j \in S^k} \nabla f_j(x^k)\big] = \nabla F(x^k).$$

- Minibatching is an intermediate approach between an SG and the full gradient. It is particularly useful for parallelization.

(See Lab 3) Compare the variance between standard SG and minibatch SG. Study the effect of the batch size.

Minibatching Instead of using a single element, use an average of several of $f_i$

$$g(x^k, \xi^k) = \frac{1}{|S^k|} \sum_{j \in S^k} \nabla f_j(x^k), \quad x^{k+1} = x^k - \alpha_k g(x^k, \xi^k).$$

- an unbiased estimate of $\nabla F(x)$

$$E\big[\frac{1}{|S^k|} \sum_{j \in S^k} \nabla f_j(x^k)\big] = \nabla F(x^k).$$

- Minibatching is an intermediate approach between an SG and the full gradient. It is particularly useful for parallelization.

(See Lab 3) Compare the variance between standard SG and minibatch SG. Study the effect of the batch size.

Stochastic Newton

$$g(x^k, \xi^k) = H^k \frac{1}{s} \sum_{j \in S^k} \nabla f_j(x^k), \quad x^{k+1} = x^k - \alpha_k g(x^k, \xi^k),$$

where $H^k$ is a positive definite scaling matrix.

## Improving the SG method

- Can we use better learning rate scheme instead of the diminishing learning rates?
- Can we better approximate the full gradient with smaller variance?

## Improving the SG method

- Can we use better learning rate scheme instead of the diminishing learning rates?
- Can we better approximate the full gradient with smaller variance?

## Accelerated Stochastic Gradient Methods

- Adjusting learning rate: Nesterov accelerated gradient, Adagrad, ADAM [4], etc.
- Variance reduction methods: SAG, SAGA [1], SVRG [2], SARAH [3], etc.

**Algorithm 5.1** SVRG Methods for Minimizing an Empirical Risk $R_n$

Choose an initial iterate $w_1 \in \mathbb{R}^d$, stepsize $\alpha > 0$, and positive integer $m$.
**for** $k = 1, 2, \ldots$ **do**
    Compute the batch gradient $\nabla R_n(w_k)$.
    Initialize $\tilde{w}_1 \leftarrow w_k$.
    **for** $j = 1, \ldots, m$ **do**
        Chose $i_j$ uniformly from $\{1, \ldots, n\}$.
        Set $\tilde{g}_j \leftarrow \nabla f_{i_j}(\tilde{w}_j) - (\nabla f_{i_j}(w_k) - \nabla R_n(w_k))$.
        Set $\tilde{w}_{j+1} \leftarrow \tilde{w}_j - \alpha \tilde{g}_j$.
    **end for**
    Option (a): Set $w_{k+1} = \tilde{w}_{m+1}$.
    Option (b): Set $w_{k+1} = \frac{1}{m} \sum_{j=1}^{m} \tilde{w}_{j+1}$.
    Option (c): Choose $j$ uniformly from $\{1, \ldots, m\}$ and set $w_{k+1} = \tilde{w}_{j+1}$.
**end for**

The description is from [5]

**Algorithm 1:** *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. $g_t^2$ indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With $\beta_1^t$ and $\beta_2^t$ we denote $\beta_1$ and $\beta_2$ to the power $t$.

**Require:** $\alpha$: Stepsize
**Require:** $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
**Require:** $f(\theta)$: Stochastic objective function with parameters $\theta$
**Require:** $\theta_0$: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
**while** $\theta_t$ not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
**end while**
**return** $\theta_t$ (Resulting parameters)

The description is from [6]

# Stochastic Subgradient Method

# References

[1] A. Defazio, F. Bach, and S. Lacoste-Julien. "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives", NeurIPS 2014.

[2] R. Johnson and T. Zhang. "Accelerating stochastic gradient descent using predictive variance reduction", NeurIPS 2013.

[3] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takac. "SARAH: A novel method for machine learning problems using stochastic recursive gradient". ICML 2017.

[4] `https://cs231n.github.io/neural-networks-3/`

[5] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization Methods for Large-Scale Machine Learning", SIAM Review 2018.

[6] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization", ICLR 2015