Momentum. Acceleration. Optimal methods

Optimization in ML

Aleksandr Beznosikov

Skoltech

28 November 2023

Questions from previous lectures

- We obtained an upper bound on the convergence of gradient descent for L-smooth and μ-strongly convex problems. **Question:** how many iterations/oracle calls should be made to find a ε-solution?

Questions from previous lectures

- We obtained an upper bound on the convergence of gradient descent for L-smooth and μ-strongly convex problems. **Question:** how many iterations/oracle calls should be made to find a ε-solution?

$$O\left(\frac{L}{\mu}\log\frac{\|x^0 - x^*\|_2}{\varepsilon}\right) \text{ iterations/oracle calls.}$$

Questions from previous lectures

- We obtained an upper bound on the convergence of gradient descent for L-smooth and μ-strongly convex problems. **Question:** how many iterations/oracle calls should be made to find a ε-solution?

$$O\left(\frac{L}{\mu}\log\frac{\|x^0 - x^*\|_2}{\varepsilon}\right) \text{ iterations/oracle calls.}$$

- The question we're going to answer today is: can we do better?

Heavy ball method

- B.T. Polyak proposed the heavy ball method in 1964.

Algorithm 1 Heavy ball method

Input: stepsizes $\{\gamma_k\}_{k=0} > 0$, momentums $\{\tau_k\}_{k=0} \in [0;1]$, starting point $x^0 = x^{-1} \in \mathbb{R}^d$, number of iterations K

1: **for** $k = 0, 1, \ldots, K-1$ **do**

2: Compute $\nabla f(x^k)$

3: $x^{k+1} = x^k - \gamma_k \nabla f(x^k) + \tau_k(x^k - x^{k-1})$

4: **end for**

Output: x^K

Heavy ball method

- B.T. Polyak proposed the heavy ball method in 1964.

Algorithm 2 Heavy ball method

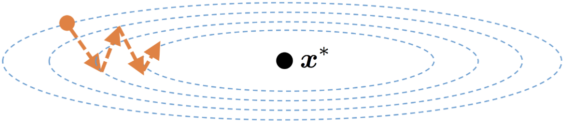
Input: stepsizes $\{\gamma_k\}_{k=0} > 0$, momentums $\{\tau_k\}_{k=0} \in [0;1]$, starting point $x^0 = x^{-1} \in \mathbb{R}^d$, number of iterations K

1: **for** $k = 0, 1, \ldots, K-1$ **do**
2: Compute $\nabla f(x^k)$
3: $x^{k+1} = x^k - \gamma_k \nabla f(x^k) + \tau_k(x^k - x^{k-1})$
4: **end for**

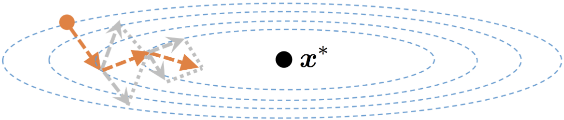
Output: x^K

- Let us add a momentum term to the gradient descent — assume that the point responsible for the current position value x^k has inertia.

Comparison of heavy ball and gradient descent

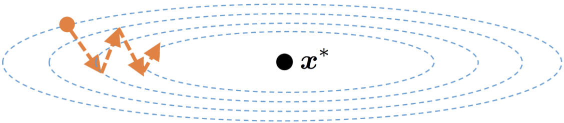


gradient descent

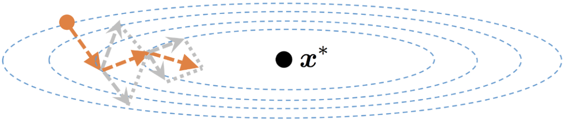


heavy-ball method

Comparison of heavy ball and gradient descent



gradient descent



heavy-ball method

An interactive illustration is available at the link.

Pros and cons

Question: what pros and cons of the heavy ball method do you see?

Pros and cons

Question: what pros and cons of the heavy ball method do you see?

Pros

- Understandable physics and intuition.
- Easy to implement.
- Cheapness of computation.

Cons

- Now we need to choose two parameters. Now we only know how to estimate γ_k in theory. Now we need to do something about τ_k.... Typically, τ_k is taken to be close to 1 or to limit to 1.
- We were going for the acceleration of gradient descent. Does it even exist in the general case?

Pros and cons

Question: what pros and cons of the heavy ball method do you see?

Pros

- Understandable physics and intuition.
- Easy to implement.
- Cheapness of computation.

Cons

- Now we need to choose two parameters. Now we only know how to estimate γ_k in theory. Now we need to do something about τ_k.... Typically, τ_k is taken to be close to 1 or to limit to 1.
- We were going for the acceleration of gradient descent. Does it even exist in the general case? No...

Accelerated gradient method

- Y.E. Nesterov proposed an accelerated gradient method in 1983.

Algorithm 3 Accelerated gradient method

Input: stepsizes $\{\gamma_k\}_{k=0} > 0$, momentums $\{\tau_k\}_{k=0} \in [0; 1]$, starting point $x^0 = y^0 \in \mathbb{R}^d$, number of iterations K

for $k = 0, 1, \ldots, K-1$ **do**

  Compute $\nabla f(y^k)$

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

  $y^{k+1} = x^{k+1} + \tau_k (x^{k+1} - x^k)$

end for

Output: x^K

Accelerated gradient and heavy ball methods

- **Question:** What is the key difference between Nesterov's method and the heavy ball?

  Heavy ball method:

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

  Accelearated gradient method:

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

Accelerated gradient and heavy ball methods

- **Question:** What is the key difference between Nesterov's method and the heavy ball?
  Heavy ball method:
  $$x^{k+1} = x^k - \gamma_k \nabla f(x^k) + \tau_k (x^k - x^{k-1})$$
  Accelearated gradient method:
  $$x^{k+1} = y^k - \gamma_k \nabla f(y^k)$$
  $$y^{k+1} = x^{k+1} + \tau_k (x^{k+1} - x^k)$$
- Let us rewrite the accelerated gradient method:
  $$x^{k+1} = x^k + \tau_k (x^k - x^{k-1}) - \gamma_k \nabla f(x^k + \tau_k (x^k - x^{k-1})).$$
  Momentum at the gradient counting point/«look ahead»/extrapolation

Accelerated gradient method

- The convergence of Nesterov's method is proved in the book.
- Now there are modifications of Nesterov's idea that also achieve the same result.

Algorithm 4 Linear coupling: inner loop

Input: stepsizes $\{\gamma_k\}_{k=0} > 0$ and $\{\eta_k\}_{k=0} > 0$, momentums $\{\tau_k\}_{k=0} \in [0;1]$, starting point $x^0 = y^0 = z^0 \in \mathbb{R}^d$, number of iterations K

1: **for** $k = 0, 1, \ldots, K-1$ **do**
2: Compute $\nabla f(x^k)$
3: $y^{k+1} = x^k - \eta_k \nabla f(x^k)$
4: $z^{k+1} = z^k - \gamma_k \nabla f(x^k)$
5: $x^{k+1} = \tau_k z^{k+1} + (1-\tau_k) y^{k+1}$
6: **end for**

Output: $\frac{1}{K}\sum_{k=0}^{K-1} x^k$

To prove we need

- The method itself (with fixed parameters):

Algorithm 5 Linear coupling: inner loop

Input: stepsizes $\gamma > 0$ and $\eta > 0$, momentums $\tau \in [0;1]$, starting point $x^0 = y^0 = z^0 \in \mathbb{R}^d$, number of iterations K

1: **for** $k = 0, 1, \dots, K-1$ **do**

2: Compute $\nabla f(x^k)$

3: $y^{k+1} = x^k - \eta \nabla f(x^k)$

4: $z^{k+1} = z^k - \gamma \nabla f(x^k)$

5: $x^{k+1} = \tau z^{k+1} + (1-\tau) y^{k+1}$

6: **end for**

Output: $\frac{1}{K}\sum_{k=0}^{K-1} x^k$

- μ-strong convexity and L-smoothness:

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

Proof

Use line 4 of Algorithm 5:

$$\begin{aligned}\|z^{k+1} - x^*\|_2^2 =& \|z^k - \gamma \nabla f(x^k) - x^*\|_2^2 \\ =& \|z^k - x^*\|_2^2 - 2\gamma \langle \nabla f(x^k), z^k - x^* \rangle + \gamma^2 \|\nabla f(x^k)\|^2 \\ =& \|z^k - x^*\|_2^2 - 2\gamma \langle \nabla f(x^k), x^k - x^* \rangle \\ & - 2\gamma \langle \nabla f(x^k), z^k - x^k \rangle + \gamma^2 \|\nabla f(x^k)\|^2. \end{aligned} \tag{1}$$

Proof

Use line 4 of Algorithm 5:

$$\begin{aligned}\|z^{k+1} - x^*\|_2^2 =& \|z^k - \gamma \nabla f(x^k) - x^*\|_2^2 \\ =& \|z^k - x^*\|_2^2 - 2\gamma\langle \nabla f(x^k), z^k - x^* \rangle + \gamma^2 \|\nabla f(x^k)\|^2 \\ =& \|z^k - x^*\|_2^2 - 2\gamma\langle \nabla f(x^k), x^k - x^* \rangle \\ & - 2\gamma\langle \nabla f(x^k), z^k - x^k \rangle + \gamma^2 \|\nabla f(x^k)\|^2. \end{aligned} \tag{1}$$

Let us estimate $\left[-\langle \nabla f(x^k), z^k - x^k \rangle\right]$ and $\|\nabla f(x^k)\|^2$.

Proof

Start with $\|\nabla f(x^k)\|^2$ and use smoothness

$$f(y^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), y^{k+1} - x^k \rangle + \frac{L}{2}\|y^{k+1} - x^k\|_2^2.$$

Proof

Start with $\|\nabla f(x^k)\|^2$ and use smoothness

$$f(y^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), y^{k+1} - x^k \rangle + \frac{L}{2}\|y^{k+1} - x^k\|_2^2.$$

Let us substitute the iterative step for y^{k+1} (line 3 Algorithm 5):

$$\begin{aligned} f(y^{k+1}) \leq & f(x^k) - \eta\|\nabla f(x^k)\|_2^2 + \frac{L\eta^2}{2}\|\nabla f(x^k)\|_2^2. \\ = & f(x^k) - \eta\left(1 - \frac{L\eta}{2}\right)\|\nabla f(x^k)\|_2^2. \end{aligned}$$

Proof

Start with $\|\nabla f(x^k)\|^2$ and use smoothness

$$f(y^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), y^{k+1} - x^k \rangle + \frac{L}{2}\|y^{k+1} - x^k\|_2^2.$$

Let us substitute the iterative step for y^{k+1} (line 3 Algorithm 5):

$$\begin{aligned} f(y^{k+1}) \leq & f(x^k) - \eta\|\nabla f(x^k)\|_2^2 + \frac{L\eta^2}{2}\|\nabla f(x^k)\|_2^2. \\ = & f(x^k) - \eta\left(1 - \frac{L\eta}{2}\right)\|\nabla f(x^k)\|_2^2. \end{aligned}$$

Let us choose $\eta \in \left(0; \frac{2}{L}\right)$, then

$$\|\nabla f(x^k)\|_2^2 \leq \frac{2}{\eta(2 - L\eta)}(f(x^k) - f(y^{k+1})). \tag{2}$$

Proof

Combine (1) and (2):

$$
\begin{aligned}
\|z^{k+1} - x^*\|_2^2 \leq & \|z^k - x^*\|_2^2 - 2\gamma\langle \nabla f(x^k), x^k - x^*\rangle \\
& + \frac{2\gamma^2}{\eta(2 - L\eta)}(f(x^k) - f(y^{k+1})) \\
& + 2\gamma\langle \nabla f(x^k), x^k - z^k\rangle. \qquad (3)
\end{aligned}
$$

Proof

Combine (1) and (2):

$$
\begin{aligned}
\|z^{k+1} - x^*\|_2^2 \leq & \|z^k - x^*\|_2^2 - 2\gamma\langle \nabla f(x^k), x^k - x^* \rangle \\
& + \frac{2\gamma^2}{\eta(2 - L\eta)}(f(x^k) - f(y^{k+1})) \\
& + 2\gamma\langle \nabla f(x^k), x^k - z^k \rangle.
\end{aligned} \tag{3}
$$

It remains $\left[-\langle \nabla f(x^k), z^k - x^k \rangle\right]$.

Proof

Use 5 of Algorithm 5:

$$
\begin{aligned}
\langle \nabla f(x^k), x^k - z^k \rangle =& \langle \nabla f(x^k), x^k - \frac{1}{\tau}(x^k - (1-\tau)y^k) \rangle \\
=& \frac{1-\tau}{\tau} \langle \nabla f(x^k), y^k - x^k \rangle.
\end{aligned}
$$

Proof

Use 5 of Algorithm 5:

$$\begin{aligned}\langle \nabla f(x^k), x^k - z^k \rangle =& \langle \nabla f(x^k), x^k - \frac{1}{\tau}(x^k - (1-\tau)y^k)\rangle \\ =& \frac{1-\tau}{\tau}\langle \nabla f(x^k), y^k - x^k \rangle.\end{aligned}$$

Next take into account:

$$\langle \nabla f(x^k), x^k - z^k \rangle \leq \frac{1-\tau}{\tau}(f(y^k) - f(x^k)). \tag{4}$$

Proof

Connect (3) and (4):

$$
\begin{aligned}
\|z^{k+1} - x^*\|_2^2 \leq & \|z^k - x^*\|_2^2 - 2\gamma\langle \nabla f(x^k), x^k - x^* \rangle \\
& + \frac{2\gamma^2}{\eta(2 - L\eta)}(f(x^k) - f(y^{k+1})) \\
& + 2\gamma \cdot \frac{1-\tau}{\tau}(f(y^k) - f(x^k)).
\end{aligned}
$$

Let us adjust the parameters as follows $\frac{\gamma}{\eta(2-L\eta)} = \frac{1-\tau}{\tau}$:

$$
\begin{aligned}
\|z^{k+1} - x^*\|_2^2 \leq & \|z^k - x^*\|_2^2 - 2\gamma\langle \nabla f(x^k), x^k - x^* \rangle \\
& + \frac{2\gamma^2}{\eta(2 - L\eta)}(f(y^k) - f(y^{k+1})).
\end{aligned}
$$

Proof

Rearrange:

$$
\begin{aligned}
2\gamma\langle \nabla f(x^k), x^k - x^* \rangle \leq & \|z^k - x^*\|_2^2 - \|z^{k+1} - x^*\|_2^2 \\
& + \frac{2\gamma^2}{\eta(2 - L\eta)}(f(y^k) - f(y^{k+1})).
\end{aligned}
$$

Proof

Rearrange:

$$2\gamma\langle \nabla f(x^k), x^k - x^* \rangle \leq \|z^k - x^*\|_2^2 - \|z^{k+1} - x^*\|_2^2 + \frac{2\gamma^2}{\eta(2 - L\eta)}(f(y^k) - f(y^{k+1})).$$

Next we use convexity:

$$2\gamma(f(x^k) - f(x^*)) \leq \|z^k - x^*\|_2^2 - \|z^{k+1} - x^*\|_2^2 + \frac{2\gamma^2}{\eta(2 - L\eta)}(f(y^k) - f(y^{k+1})).$$

Proof

Summing up by k and averaging:

$$
\begin{aligned}
\frac{2\gamma}{K}\sum_{k=0}^{K-1}(f(x^k)-f(x^*)) \leq & \frac{1}{K}\sum_{k=0}^{K-1}\left(\|z^k-x^*\|_2^2-\|z^{k+1}-x^*\|_2^2\right) \\
& + \frac{2\gamma^2}{\eta(2-L\eta)K}\sum_{k=0}^{K-1}(f(y^k)-f(y^{k+1})) \\
= & \frac{1}{K}\left(\|z^0-x^*\|_2^2-\|z^K-x^*\|_2^2\right) \\
& + \frac{2\gamma^2}{\eta(2-L\eta)K}(f(y^0)-f(y^K)) \\
\leq & \frac{\|x^0-x^*\|_2^2}{K} + \frac{2\gamma^2(f(y^0)-f(x^*))}{\eta(2-L\eta)K}.
\end{aligned}
$$

Proof

Substituting starting points: $x^0 = y^0 = z^0$ and using Jensens's inequality:

$$2\gamma \left[f\left(\frac{1}{K} \sum_{k=0}^{K-1} x^k \right) - f(x^*) \right] \leq \frac{\|x^0 - x^*\|_2^2}{K} + \frac{2\gamma^2 (f(x^0) - f(x^*))}{\eta(2 - L\eta)K}.$$

Proof

Substituting starting points: $x^0 = y^0 = z^0$ and using Jensens's inequality:

$$2\gamma \left[f\left(\frac{1}{K} \sum_{k=0}^{K-1} x^k \right) - f(x^*) \right] \leq \frac{\|x^0 - x^*\|_2^2}{K} + \frac{2\gamma^2 (f(x^0) - f(x^*))}{\eta(2 - L\eta)K}.$$

Next we use μ-strong convexity

$$\begin{aligned} f\left(\frac{1}{K} \sum_{k=0}^{K-1} x^k \right) - f(x^*) &\leq \frac{f(x^0) - f(x^*)}{2\mu\gamma K} + \frac{\gamma(f(x^0) - f(x^*))}{\eta(2 - L\eta)K} \\ &= \left(\frac{1}{2\mu\gamma K} + \frac{\gamma}{\eta(2 - L\eta)K} \right) (f(x^0) - f(x^*)). \end{aligned}$$

Proof

Optimizing estimation with the choice of $\eta = \frac{1}{L}$:

$$f\left(\frac{1}{K}\sum_{k=0}^{K-1} x^k\right) - f(x^*) \leq \left(\frac{1}{2\mu\gamma K} + \frac{\gamma L}{K}\right)(f(x^0) - f(x^*)).$$

Proof

Optimizing estimation with the choice of $\eta = \frac{1}{L}$:

$$f\left(\frac{1}{K}\sum_{k=0}^{K-1} x^k\right) - f(x^*) \leq \left(\frac{1}{2\mu\gamma K} + \frac{\gamma L}{K}\right)(f(x^0) - f(x^*)).$$

And with the choice of $\gamma = \frac{1}{\sqrt{2\mu L}}$:

$$f\left(\frac{1}{K}\sum_{k=0}^{K-1} x^k\right) - f(x^*) \leq \sqrt{\frac{2L}{\mu K^2}}(f(x^0) - f(x^*)).$$

Proof

And then $K = \sqrt{\frac{8L}{\mu}}$

$$f\left(\frac{1}{K}\sum_{k=0}^{K-1} x^k\right) - f(x^*) \leq \frac{1}{2}(f(x^0) - f(x^*)).$$

Proof

And then $K = \sqrt{\frac{8L}{\mu}}$

$$f\left(\frac{1}{K}\sum_{k=0}^{K-1} x^k\right) - f(x^*) \leq \frac{1}{2}(f(x^0) - f(x^*)).$$

Question: why?

Proof

And then $K = \sqrt{\frac{8L}{\mu}}$

$$f\left(\frac{1}{K}\sum_{k=0}^{K-1} x^k\right) - f(x^*) \leq \frac{1}{2}(f(x^0) - f(x^*)).$$

Question: why? for K iterations we are guaranteed to get 2 times closer to the solution. Then let this be one iteration of our new outer algorithm. That is, we run linear coupling for K iterations, and therefore restart with a new starting point $\frac{1}{K}\sum\limits_{k=0}^{K-1} x^k$ taken from the last coupling run. These are called restarts.

Proof

Then, after T restarts:

$$f\left(x^T\right) - f(x^*) \leq \frac{1}{2^T}(f(x^0) - f(x^*)).$$

From where we can immediately get oracle complexity:

$$f\left(x^T\right) - f(x^*) \leq \frac{1}{2^T}(f(x^0) - f(x^*)) \leq \varepsilon.$$

$$T \geq \log_2\left(\frac{f(x^0) - f(x^*)}{\varepsilon}\right)$$

$$K \cdot T = O\left(\sqrt{\frac{L}{\mu}} \log_2 \frac{f(x^0) - f(x^*)}{\varepsilon}\right) \quad \text{oracle calls.}$$

Convergence of linear coupling

Theorem onconvergence of linear coupling

Let the unconditional optimization problem with L-smooth, μ-simply convex objective function f be solved using restored linear kapling. Then with $\eta = \frac{1}{L}$, $\gamma = \frac{1}{\sqrt{2\mu L}}$ and $K = \sqrt{\frac{8L}{\mu}}$, to achieve accuracy ε on the function $(f(x) - f(x^*) \leq \varepsilon)$, we need

$$O\left(\sqrt{\frac{L}{\mu}} \log \frac{f(x^0) - f(x^*)}{\varepsilon}\right) \text{ oracle calls.}$$

Questions remain

- A better method than gradient descent.
- But can we do more?
- **Question:** how do we know if it can be better?

Questions remain

- A better method than gradient descent.
- But can we do more?
- **Question:** how do we know if it can be better? to get lower bounds.

Questions remain

- A better method than gradient descent.
- But can we do more?
- **Question:** how do we know if it can be better? to get lower bounds.
- To get lower bounds, we don't need to come up with a method, but a «bad» function that any method will take a «long» time to optimize. **Question:** what does «any method» mean here?

Class of algorithms

- An initial point x^0 is given. This initial point gives rise to some set M_0 – the set of all points reached so far (at a given step k). $M_0 = \{x^0\}$.

Class of algorithms

- An initial point x^0 is given. This initial point gives rise to some set M_0 – the set of all points reached so far (at a given step k). $M_0 = \{x^0\}$.
- On the current oracle call, the method can count the gradient of the function at the point x^k: $\nabla f(x^k)$, where $x^k \in M_k$, that is, the method can count the gradient at all the points it has already reached. Initially, we can only calculate the gradient at x^0.

Class of algorithms

- An initial point x^0 is given. This initial point gives rise to some set M_0 – the set of all points reached so far (at a given step k). $M_0 = \{x^0\}$.
- On the current oracle call, the method can count the gradient of the function at the point x^k: $\nabla f(x^k)$, where $x^k \in M_k$, that is, the method can count the gradient at all the points it has already reached. Initially, we can only calculate the gradient at x^0.
- $M_{k+1} = \text{span}\{x', \nabla f(x'')\}$ (linear envelope), where $x', x'' \in M_k$.

Class of algorithms

- An initial point x^0 is given. This initial point gives rise to some set M_0 – the set of all points reached so far (at a given step k). $M_0 = \{x^0\}$.
- On the current oracle call, the method can count the gradient of the function at the point x^k: $\nabla f(x^k)$, where $x^k \in M_k$, that is, the method can count the gradient at all the points it has already reached. Initially, we can only calculate the gradient at x^0.
- $M_{k+1} = \text{span}\{x', \nabla f(x'')\}$ (linear envelope), where $x', x'' \in M_k$.
- After K of oracle calls, the output of the method is some point of M_K.

Class of algorithms

- An initial point x^0 is given. This initial point gives rise to some set M_0 – the set of all points reached so far (at a given step k). $M_0 = \{x^0\}$.
- On the current oracle call, the method can count the gradient of the function at the point x^k: $\nabla f(x^k)$, where $x^k \in M_k$, that is, the method can count the gradient at all the points it has already reached. Initially, we can only calculate the gradient at x^0.
- $M_{k+1} = \text{span}\{x', \nabla f(x'')\}$ (linear envelope), where $x', x'' \in M_k$.
- After K of oracle calls, the output of the method is some point of M_K.

Question: do the methods we have studied fit this definition?

Class of algorithms

- An initial point x^0 is given. This initial point gives rise to some set M_0 – the set of all points reached so far (at a given step k). $M_0 = \{x^0\}$.
- On the current oracle call, the method can count the gradient of the function at the point x^k: $\nabla f(x^k)$, where $x^k \in M_k$, that is, the method can count the gradient at all the points it has already reached. Initially, we can only calculate the gradient at x^0.
- $M_{k+1} = \text{span}\{x', \nabla f(x'')\}$ (linear envelope), where $x', x'' \in M_k$.
- After K of oracle calls, the output of the method is some point of M_K.

Question: do the methods we have studied fit this definition? yes, gradient descent, heavy ball method, linear coupling, and accelerated gradient method.

Class of algorithms

- An initial point x^0 is given. This initial point gives rise to some set M_0 – the set of all points reached so far (at a given step k). $M_0 = \{x^0\}$.
- On the current oracle call, the method can count the gradient of the function at the point x^k: $\nabla f(x^k)$, where $x^k \in M_k$, that is, the method can count the gradient at all the points it has already reached. Initially, we can only calculate the gradient at x^0.
- $M_{k+1} = \text{span}\{x', \nabla f(x'')\}$ (linear envelope), where $x', x'' \in M_k$.
- After K of oracle calls, the output of the method is some point of M_K.

Question: do the methods we have studied fit this definition? yes, gradient descent, heavy ball method, linear coupling, and accelerated gradient method.

Question: are all the methods that count the gradient included here?

Class of algorithms

- An initial point x^0 is given. This initial point gives rise to some set M_0 – the set of all points reached so far (at a given step k). $M_0 = \{x^0\}$.
- On the current oracle call, the method can count the gradient of the function at the point x^k: $\nabla f(x^k)$, where $x^k \in M_k$, that is, the method can count the gradient at all the points it has already reached. Initially, we can only calculate the gradient at x^0.
- $M_{k+1} = \text{span}\{x', \nabla f(x'')\}$ (linear envelope), where $x', x'' \in M_k$.
- After K of oracle calls, the output of the method is some point of M_K.

Question: do the methods we have studied fit this definition? yes, gradient descent, heavy ball method, linear coupling, and accelerated gradient method.

Question: are all the methods that count the gradient included here? no, see the next lectures.

«Bad» function

A quadratic (its sufficient) function:

$$f(x) = \frac{L-\mu}{8} x^T A x + \frac{\mu}{2} x^T x - \frac{L-\mu}{4} e_1^T x,$$

where

$$A = \begin{pmatrix} 2 & -1 & & 0 & \\ -1 & 2 & -1 & & \\ & -1 & 2 & \ddots & \\ & & \ddots & \ddots & -1 \\ & 0 & & -1 & \zeta \end{pmatrix}, \quad e_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

ζ will be defined later.

«Bad» function

A quadratic (its sufficient) function:

$$f(x) = \frac{L-\mu}{8} x^T A x + \frac{\mu}{2} x^T x - \frac{L-\mu}{4} e_1^T x,$$

where

$$A = \begin{pmatrix} 2 & -1 & & 0 & \\ -1 & 2 & -1 & & \\ & -1 & 2 & \ddots & \\ & & \ddots & \ddots & -1 \\ & 0 & & -1 & \zeta \end{pmatrix}, \quad e_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

ζ will be defined later.

The function L is smooth and μ-strongly convex (homework problem).

«Bad» function: solution

Question: what can we say about the solution?

«Bad» function: solution

Question: what can we say about the solution? Strong convex problem — the only one solution.

«Bad» function: solution

Question: what can we say about the solution? Strong convex problem — the only one solution.
Question: how to find?

«Bad» function: solution

Question: what can we say about the solution? Strong convex problem — the only one solution.

Question: how to find? Optimality condition:

$$\nabla f(x^*) = 0$$

or

$$Ax^* + \frac{4\mu}{L-\mu}x^* - e_1 = 0$$

«Bad» function: solution

Question: what can we say about the solution? Strong convex problem — the only one solution.
Question: how to find? Optimality condition:

$$\nabla f(x^*) = 0$$

or

$$Ax^* + \frac{4\mu}{L-\mu}x^* - e_1 = 0$$

Let us rewrite it component by component. The first component:

$$2x_1^* - x_2^* + \frac{4\mu}{L-\mu}x_1^* - 1 = 0 \text{ или } \frac{2(L+\mu)}{L-\mu} \cdot x_1^* - x_2^* = 1$$

«Bad» function: solution

Question: what can we say about the solution? Strong convex problem — the only one solution.
Question: how to find? Optimality condition:

$$\nabla f(x^*) = 0$$

or

$$Ax^* + \frac{4\mu}{L-\mu}x^* - e_1 = 0$$

Let us rewrite it component by component. The first component:

$$2x_1^* - x_2^* + \frac{4\mu}{L-\mu}x_1^* - 1 = 0 \text{ или } \frac{2(L+\mu)}{L-\mu} \cdot x_1^* - x_2^* = 1$$

All coordiantes (not 1st and last):

$$-x_{k-1}^* + \frac{2(L+\mu)}{L-\mu}x_k^* - x_{k+1}^* = 0$$

«Bad» function: solution

Last coordinate:

$$-x^*_{d-1} + \zeta x^*_d + \frac{4\mu}{L-\mu} x^*_d = 0 \text{ или } -x^*_{d-1} + \left(\zeta + \frac{4\mu}{L-\mu}\right) x^*_d = 0$$

«Bad» function: solution

Last coordinate:

$$-x_{d-1}^* + \zeta x_d^* + \frac{4\mu}{L-\mu} x_d^* = 0 \text{ или } -x_{d-1}^* + \left(\zeta + \frac{4\mu}{L-\mu}\right) x_d^* = 0$$

We can see that all equations (except the 1st and last one) are simply linear recurrence. The solution is as follows if ζ is chosen correctly:

$$x_k^* = q^k, \qquad q = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$$

«Bad» function: moving towards a solution

Let us take the starting point $x^0 = (0 \ldots 0)^T$.

«Bad» function: moving towards a solution

Let us take the starting point $x^0 = (0 \dots 0)^T$.
Gradient:

$$\nabla f(x) = \frac{L-\mu}{4} A x + \mu x - \frac{L-\mu}{4} e_1$$

«Bad» function: moving towards a solution

Let us take the starting point $x^0 = (0 \ldots 0)^T$.
Gradient:

$$\nabla f(x) = \frac{L-\mu}{4} Ax + \mu x - \frac{L-\mu}{4} e_1$$

Note that $\nabla f(x^0) \in \text{span}(e_1)$,

«Bad» function: moving towards a solution

Let us take the starting point $x^0 = (0 \ldots 0)^T$.
Gradient:

$$\nabla f(x) = \frac{L-\mu}{4} Ax + \mu x - \frac{L-\mu}{4} e_1$$

Note that $\nabla f(x^0) \in \text{span}(e_1)$, so it turns out that for the first oracle call only the first coordinate of the method output can be non-zero

«Bad» function: moving towards a solution

Let us take the starting point $x^0 = (0 \dots 0)^T$.
Gradient:

$$\nabla f(x) = \frac{L-\mu}{4} Ax + \mu x - \frac{L-\mu}{4} e_1$$

Note that $\nabla f(x^0) \in \text{span}(e_1)$, so it turns out that for the first oracle call only the first coordinate of the method output can be non-zero
After the second oracle call: $\nabla f(x^1) \in \text{span}(e_1, e_2), \quad x^1 \in M_1$, that is, for 2 oracle calls, at most 2 of the first coordinates can be non-zero.

«Bad» function: moving towards a solution

Let us take the starting point $x^0 = (0 \ldots 0)^T$.
Gradient:

$$\nabla f(x) = \frac{L-\mu}{4} Ax + \mu x - \frac{L-\mu}{4} e_1$$

Note that $\nabla f(x^0) \in \text{span}(e_1)$, so it turns out that for the first oracle call only the first coordinate of the method output can be non-zero
After the second oracle call: $\nabla f(x^1) \in \text{span}(e_1, e_2), \quad x^1 \in M_1$, that is, for 2 oracle calls, at most 2 of the first coordinates can be non-zero.
After K oracle calls, only the first K of coordinates can be non-zero, the rest are exactly zero.

«Bad» function: guarantees

Let us take $d = 2K$, where K is the number of oracle calls. **Question:** why?

«Bad» function: guarantees

Let us take $d = 2K$, where K is the number of oracle calls. **Question:** why?
Initial distance to solution:

$$\|x^0 - x^*\|_2^2 = \sum_{i=1}^{2K} q^{2i} = (1 + q^{2K}) \sum_{i=1}^{K} q^{2i}$$

«Bad» function: guarantees

Let us take $d = 2K$, where K is the number of oracle calls. **Question:** why?
Initial distance to solution:

$$\|x^0 - x^*\|_2^2 = \sum_{i=1}^{2K} q^{2i} = (1 + q^{2K}) \sum_{i=1}^{K} q^{2i}$$

After K of oracle calls, the final output can be evaluated as follows (only the first K of coordinates are non-zero):

$$\|x^K - x^*\|^2 \geq \sum_{i=K+1}^{2K} q^{2i} = q^{2K} \sum_{i=1}^{K} q^{2i} = \frac{q^{2K}}{1 + q^{2K}} \|x^0 - x^*\|_2^2$$

$$\geq \frac{q^{2K}}{2} \|x^0 - x^*\|_2^2 = \left(1 - \frac{2\sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}\right)^{2K} \frac{\|x^0 - x^*\|_2^2}{2}$$

Lower bound on oracle complexity

Lower bound on oracle complexity

For any method from the class described above, there exists an unconditional optimization problem with L-smooth, μ-strongly convex objective function f such that to solve this problem the method needs to

$$\Omega\left(\sqrt{\frac{L}{\mu}}\log\frac{\|x^0 - x^*\|_2}{\varepsilon}\right) \text{ oracle calls.}$$

Lower bound on oracle complexity

Lower bound on oracle complexity

For any method from the class described above, there exists an unconditional optimization problem with L-smooth, μ-strongly convex objective function f such that to solve this problem the method needs to

$$\Omega\left(\sqrt{\frac{L}{\mu}}\log\frac{\|x^0 - x^*\|_2}{\varepsilon}\right) \text{ oracle calls.}$$

- Linear coupling is the optimal method in terms of oracle calls for L-smooth and μ-strongly convex problems.

Lower bound on oracle complexity

Lower bound on oracle complexity

For any method from the class described above, there exists an unconditional optimization problem with L-smooth, μ-strongly convex objective function f such that to solve this problem the method needs to

$$\Omega\left(\sqrt{\frac{L}{\mu}}\log\frac{\|x^0 - x^*\|_2}{\varepsilon}\right) \text{ oracle calls.}$$

- Linear coupling is the optimal method in terms of oracle calls for L-smooth and μ-strongly convex problems.
- For L-smooth and convex problems too.

Lower bound on oracle complexity

Lower bound on oracle complexity

For any method from the class described above, there exists an unconditional optimization problem with L-smooth, μ-strongly convex objective function f such that to solve this problem the method needs to

$$\Omega\left(\sqrt{\frac{L}{\mu}}\log\frac{\|x^0 - x^*\|_2}{\varepsilon}\right) \text{ oracle calls.}$$

- Linear coupling is the optimal method in terms of oracle calls for L-smooth and μ-strongly convex problems.
- For L-smooth and convex problems too.
- For the accelerated gradient method the results are the same.