Optimization for Data Science Lecture 13: Proximal Gradient Methods (second part)

Kimon Fountoulakis

School of Computer Science
University of Waterloo

29/10/2019

Outline of this lecture

- Proximal mapping
- Optimality conditions for composite problems
- Proximal gradient method with fixed step-size
- Proximal gradient method with line-search
- Proof of convergence rate

Composite Optimization Problems

- We are interested in minimizing

$$\textbf{minimize}_{x \in \mathbb{R}^n} \; g(x) + f(x)$$

- $f(x)$ is smooth (differentiable)

- $g(x)$ is convex. This function is not-necessarily smooth.

Modelling Motivation

- Composite problems are very popular in machine learning because
  - f represents a loss function.
  - g represents a regularizer, i.e., $\|x\|_2^2$, $\|x\|_1$.
- Different regularizers often represent different prior information about the optimal solution.

Algorithmic Motivation

- So far we have seen two ways to solve non-smooth problems:
  - Smooth the objective function and apply a gradient-type method
  - Use a sub-gradient method on the non-smooth objective function

Algorithmic Motivation

- Smoothing makes the problem differentiable, but iteration complexity of gradient methods takes a hit.
- Sub-gradient methods are very slow and they require a lot of parameter tuning.
- There exists a **very** popular class of non-smooth problems for which we can apply a specialized gradient method without smoothing or using sub-gradients. Also, the rate is worse than the rate for smooth objective functions.

Recap: making of gradient descent for smooth functions

- Let’s try to derive gradient descent for smooth functions again and based on what we learn we will derive proximal gradient descent for non-smooth composite problems.

Recap: making of gradient descent for smooth functions

- Say that we want to minimize a smooth function f.
- We defined gradient descent as $x_{k+1} := x_k - \frac{1}{L} \nabla f(x_k)$
- This is equivalent to

  $x_{k+1} :=$ argmin$_{x \in \mathbb{R}^n}$ $f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{L}{2} \|x - x_k\|_2^2$
- Why? simply compute the optimality conditions of the above strongly convex problem: $\nabla f(x_k) + L(x - x_k) = 0$
- and solve w.r.t x.

Recap: making of gradient descent for smooth functions

- Let's work with this definition of gradient descent.

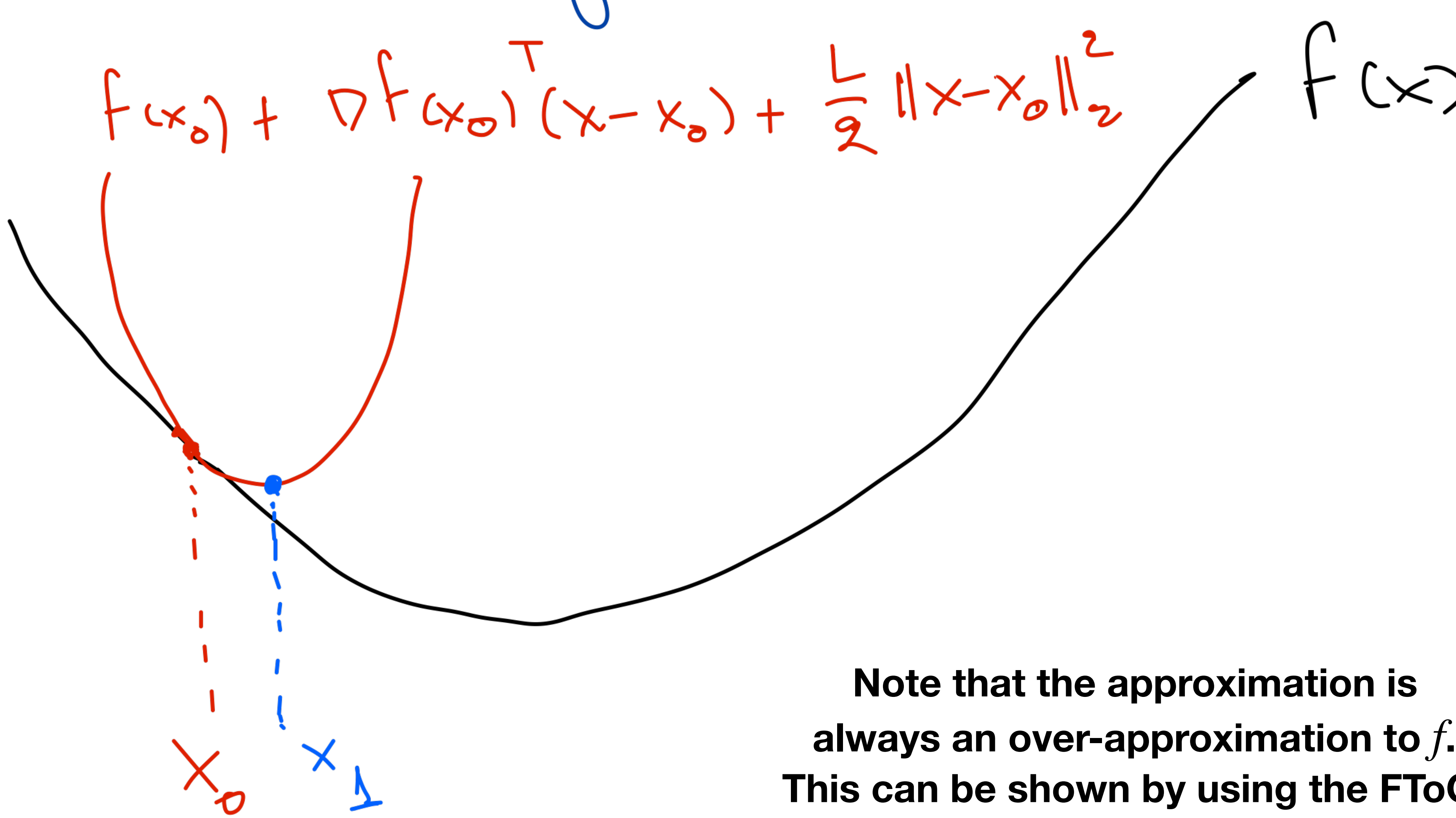
$$x_{k+1} := \text{argmin}_{x \in \mathbb{R}^n} \; f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{L}{2} \|x - x_k\|_2^2$$

This function looks like an approximation to $f(x)$: linearization of f at x_k + $\frac{L}{2} \|x - x_k\|_2^2$.

Recap: making of gradient descent for smooth functions

1st iteration of gradient descent

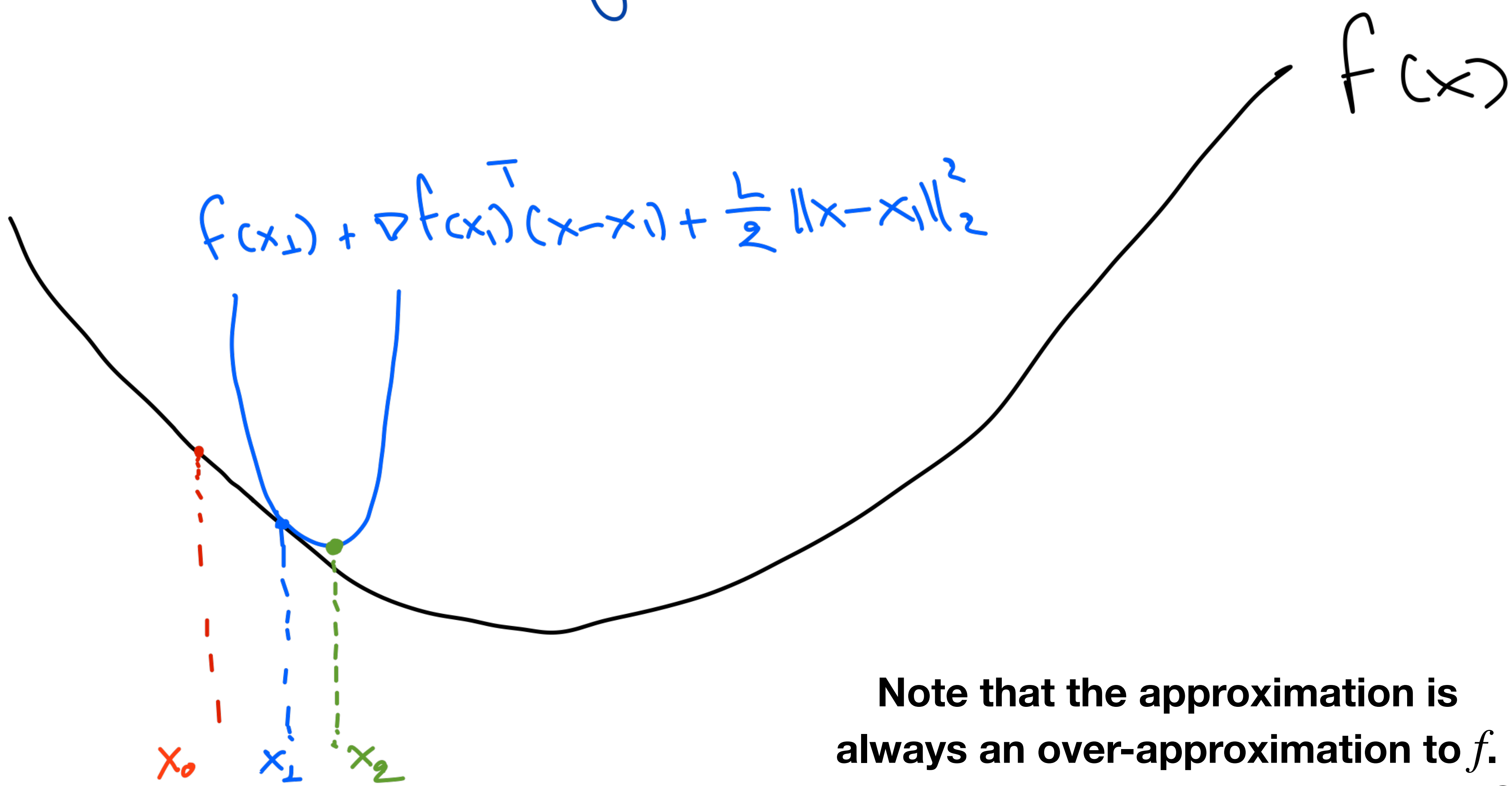
$$f(x_0) + \nabla f(x_0)^T (x - x_0) + \frac{L}{2} \|x - x_0\|_2^2$$



Note that the approximation is always an over-approximation to f. This can be shown by using the FToC.

Recap: making of gradient descent for smooth functions

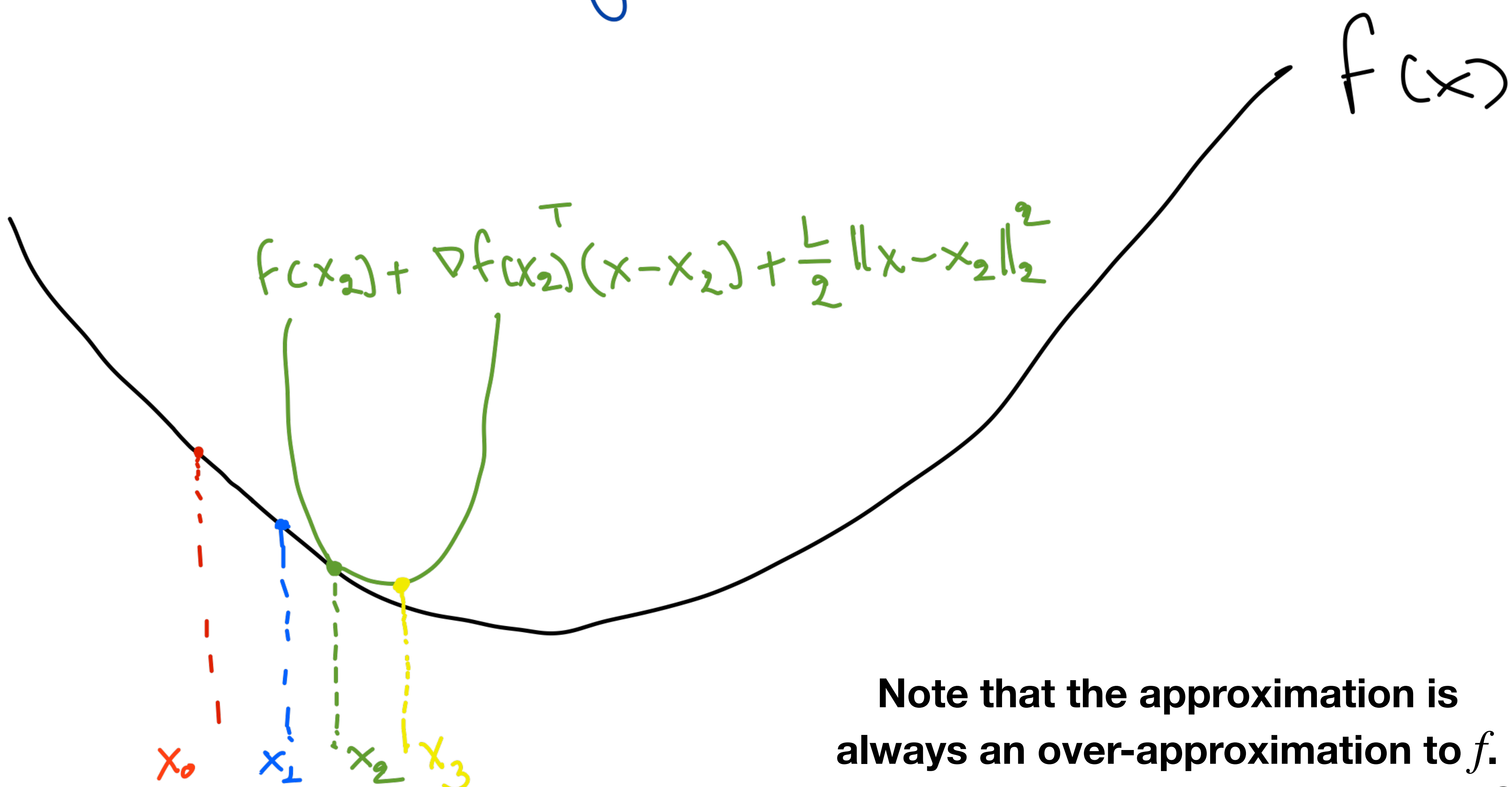
2nd iteration of gradient descent



Note that the approximation is always an over-approximation to f. This can be shown by using the FToC.

Recap: making of gradient descent for smooth functions

3rd iteration of gradient descent



Note that the approximation is always an over-approximation to f. This can be shown by using the FToC.

Recap: making of gradient descent for smooth functions

- This means that we can view gradient descent as a sequence of subproblems:

$$x_{k+1} := \text{argmin}_{x \in \mathbb{R}^n} \; f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{L}{2} \|x - x_k\|_2^2$$

- that are easier to solve than solving the original problem.

- (More generally this is true for **many** optimization algorithms)

What about composite problems?

- But now we have to minimize $g + f$, where g is not necessarily smooth.
- This means that we cannot compute $\nabla g + \nabla f$.
- We could compute a sub-gradient of g, but as we saw in previous lectures this does not result in efficient algorithms.
- So what do we do?

Proximal Gradient Descent: intuitive interpretation

- Simple idea: let's just add g in the sub-problem of gradient descent without approximating it.

$$x_{k+1} = \text{argmin}_{x\in\mathbb{R}^n} \quad \underbrace{g(x)}_{\text{new term}} \quad + \quad \underbrace{f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{L}{2}\|x - x_k\|_2^2}_{\text{similar to gradient descent for smooth } f}$$

- What are possible issues?

  - The sub-problem might not be "easy" to solve anymore.

  - This means that we do not have a closed form solution for this sub-problem like we had for gradient descent applied only on f.

Example

- Fortunately, for special cases of g, the sub-problem does have a closed form solution.

- Example: $g(x) = \lambda \|x\|_1$

- Then

- $$[x_{k+1}]_j = \begin{cases} u_j - \frac{\lambda}{L} & \text{if } u_j \geq \frac{\lambda}{L} \\ 0 & \text{if } |u_j| \leq \frac{\lambda}{L} \\ u_j + \frac{\lambda}{L} & \text{if } u_j \leq -\frac{\lambda}{L} \end{cases} \quad \forall j$$

- where $u = x_k - \frac{\lambda}{L} \nabla f(x_k)$.

- How do we obtain this? Through the optimality conditions of the sub-problem. (We did make a similar derivation when we were studying smoothing of the L1-norm).

What about non-constant step-sizes?

- In this case, the sub-problem simply is:

$$x_{k+1} = \text{argmin}_{x\in\mathbb{R}^n} \underbrace{g(x)}_{\text{new term}} + \underbrace{f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{2\alpha_k}\|x - x_k\|_2^2}_{\text{similar to gradient descent for smooth } f}$$

- Previously we had:

$$x_{k+1} = \text{argmin}_{x\in\mathbb{R}^n} \underbrace{g(x)}_{\text{new term}} + \underbrace{f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{L}{2}\|x - x_k\|_2^2}_{\text{similar to gradient descent for smooth } f}$$

Proximal Mapping

- We can generalize the previous technique by using proximal mapping.

- The **proximal mapping** or **proximal operator** of a convex function g is defined as
  $\text{prox}_g(x) = \text{argmin}_{u \in \mathbb{R}^n} \; g(u) + \dfrac{1}{2}\|u - x\|_2^2$

Proximal Gradient Method

- Using the definition of proximal mapping, proximal gradient descent can be written as:
$x_{k+1} = \text{prox}_{\alpha_k g}(x_k - \alpha_k \nabla f(x_k))$

- which is equivalent to

$$x_{k+1} = \text{argmin}_{x \in \mathbb{R}^n} \; g(x) + \frac{1}{2\alpha_k} \| x - x_k + \alpha_k \nabla f(x_k) \|_2^2$$

$$= \text{argmin}_{x \in \mathbb{R}^n} \; \underbrace{g(x)}_{\text{new term}} \; \underbrace{+ f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{\alpha_k 2} \| x - x_k \|_2^2}_{\text{similar to gradient descent for smooth } f}$$

Descent Lemma

- Let $0 < \alpha_k \leq 2/L$ then

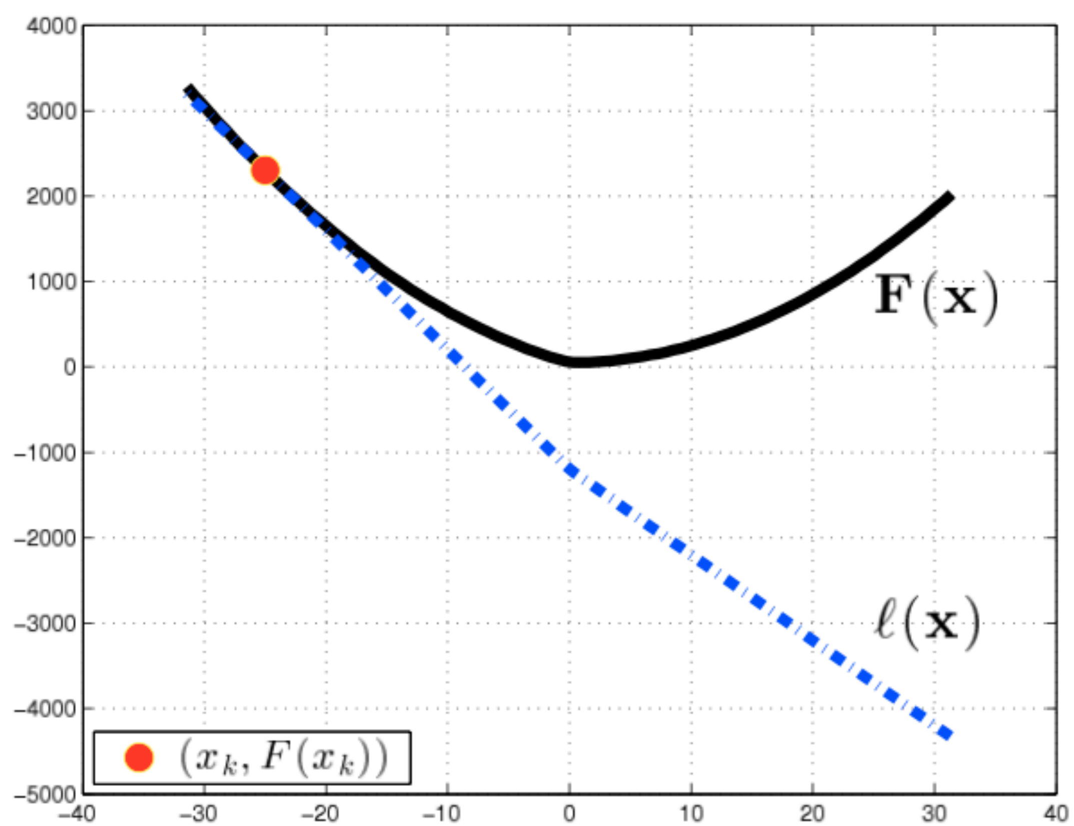
$$F(x_{k+1}) \leq F(x_k) + \left(\frac{L}{2} - \frac{1}{\alpha_k}\right)\|x_{k+1} - x_k\|_2^2.$$

Armijo Line-Search for Proximal Gradient

- Le us define $\ell(x) := g(x) + \underbrace{f(x_k) + \nabla f(x_k)^T(x - x_k)}_{\text{linearization of } f \text{ at } x_k}$

- Thus function $\ell(x)$ is a lower approximation to $g + f$ at x_k.

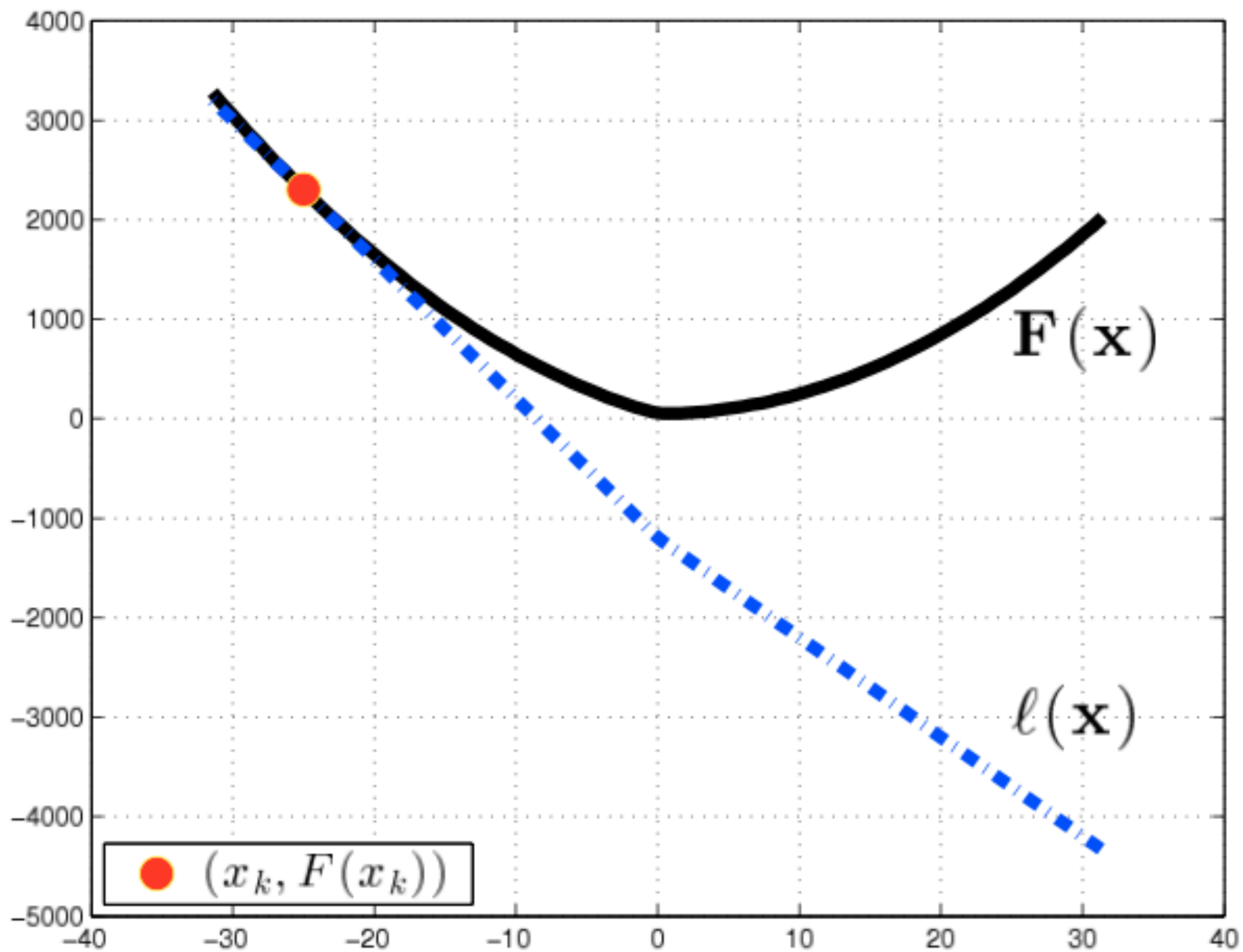
- Define $F(x) := g(x) + f(x)$, then this is how $\ell(x)$ looks like:



Armijo Line-Search for Proximal Gradient

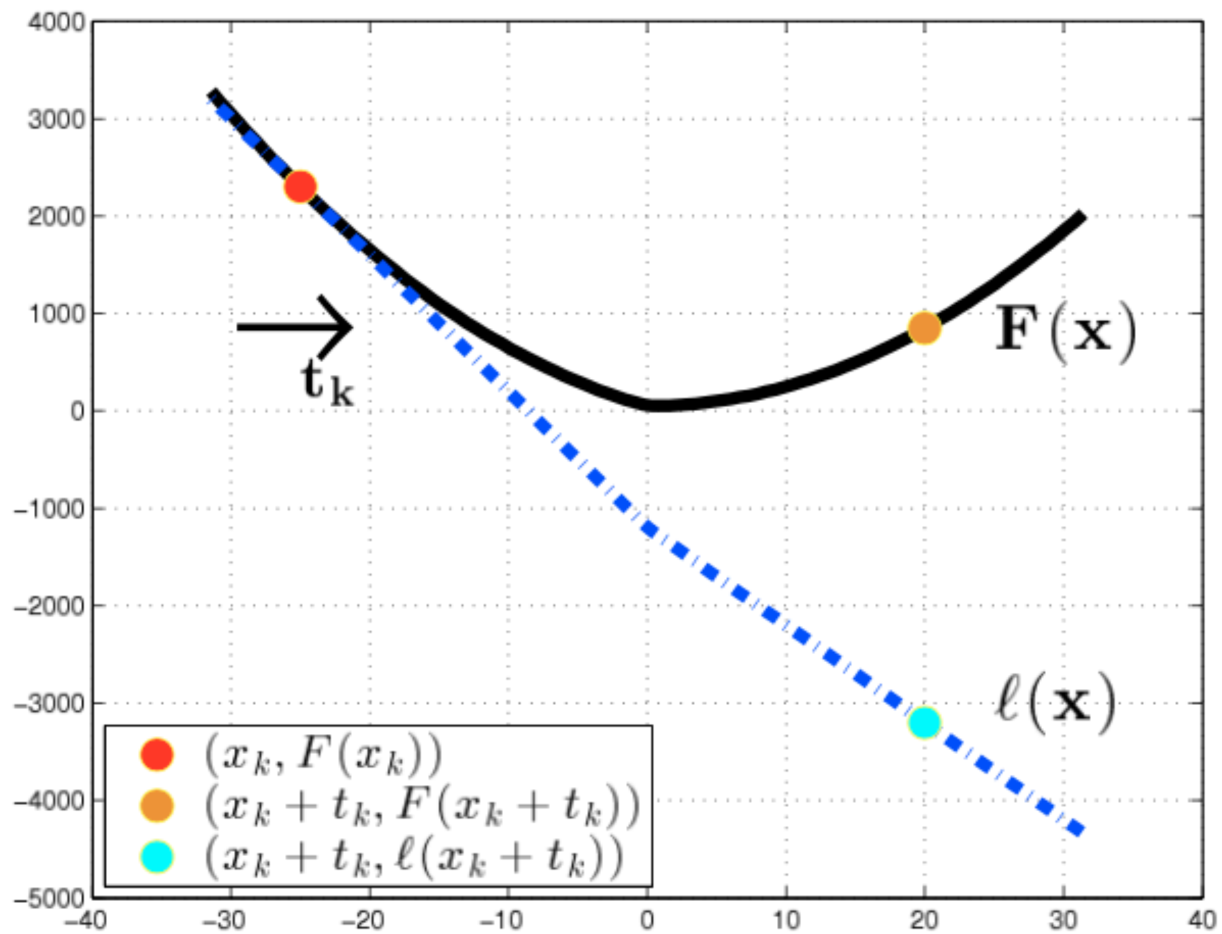
- Let $x(\alpha) := \text{prox}_{\alpha g}(x_k - \alpha \nabla f(x_k))$
- Start with a guess $\alpha = 1$
- Check if the objective is **sufficiently** decreased:
- $F(x(\alpha)) \leq F(x_k) - \theta \left(\ell(x_k) - \ell(x(\alpha) \right)$ for $\theta \in (0,1)$
- If not, then half α i.e., $\alpha \leftarrow \alpha/2$

Armijo Line-Search: Intuition



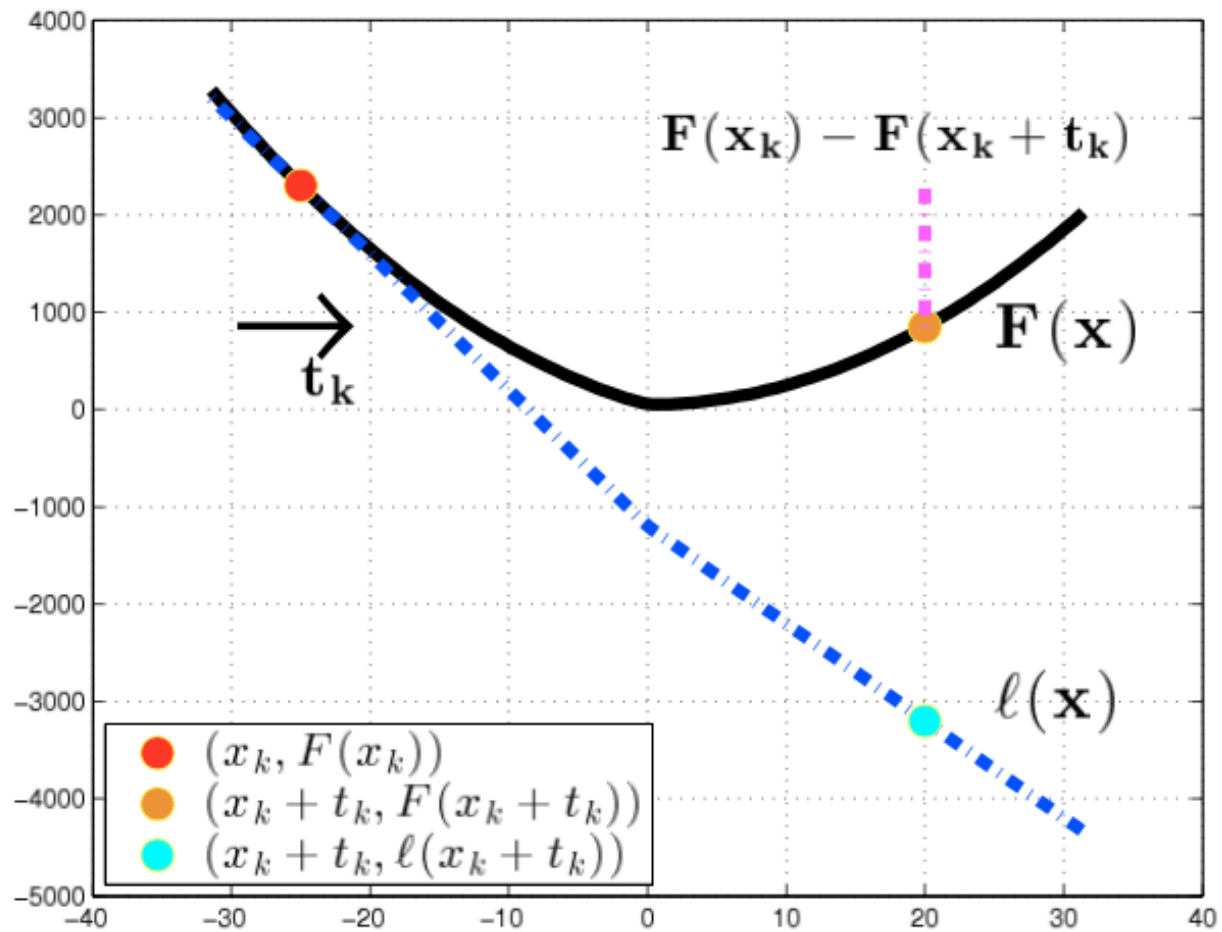
Armijo Line-Search: Intuition

- Say that the proximal gradient suggests that I move from x_k (red dot) the direction t_k to the left.
- I set $\alpha = 1$ and this take me to point $x_k + t_k = 20$. Is this point good enough?



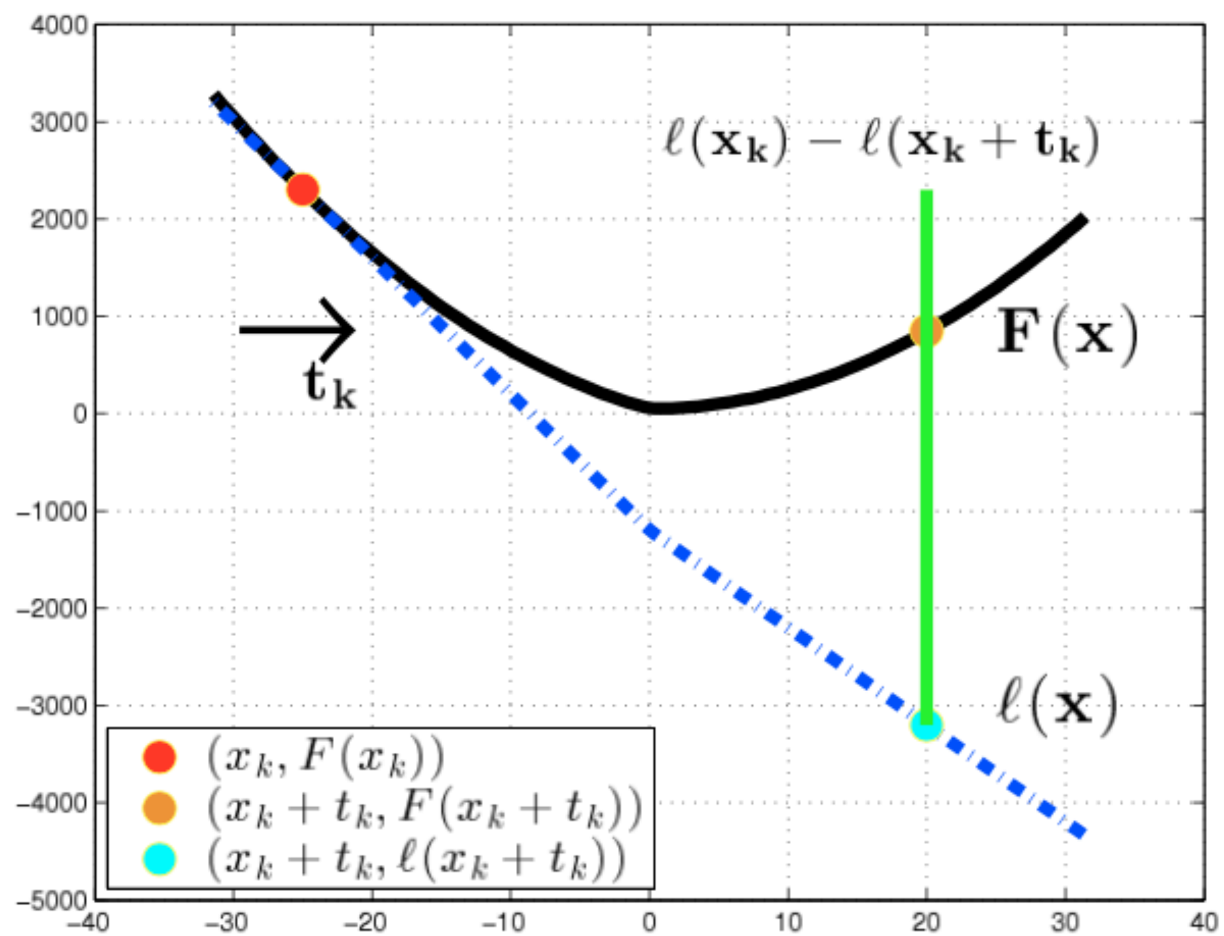
Armijo Line-Search: Intuition

- We measure the decrease in the objective function: $F(x_k) - F(x_k + t_k)$.



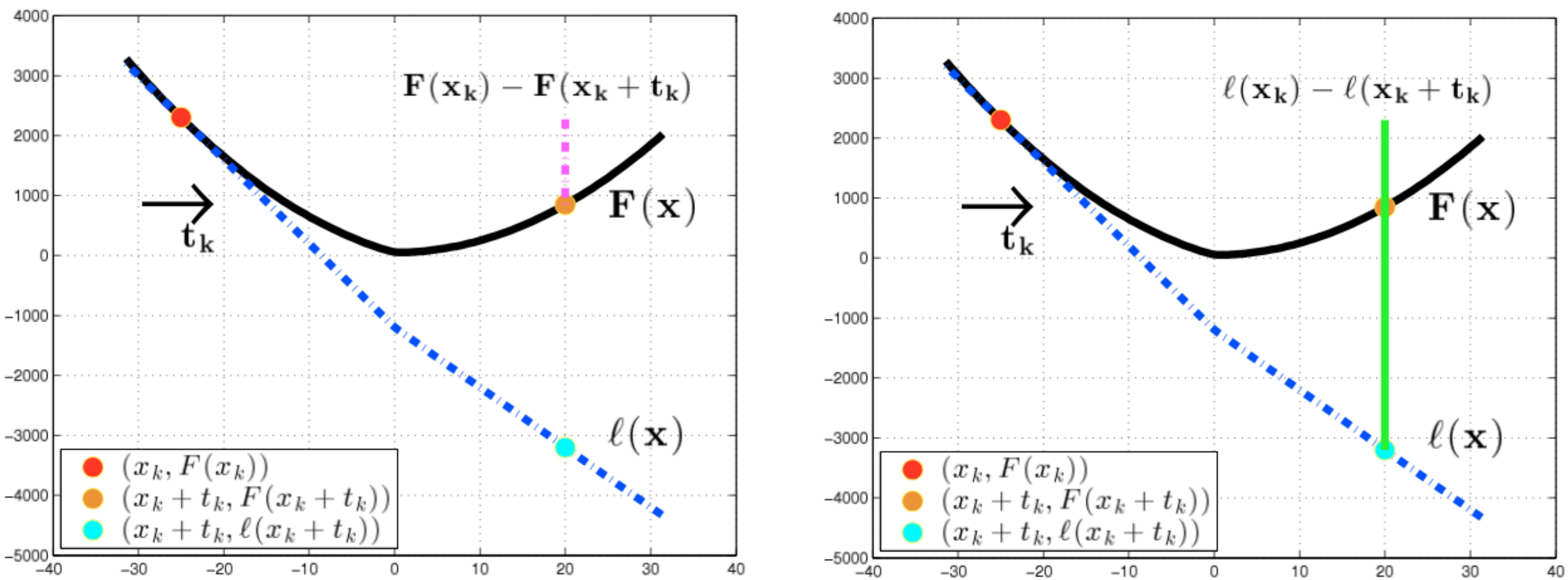
Armijo Line-Search: Intuition

- We measure the decrease in the approximation to the objective function: $\ell(x_k) - \ell(x_k + t_k)$.



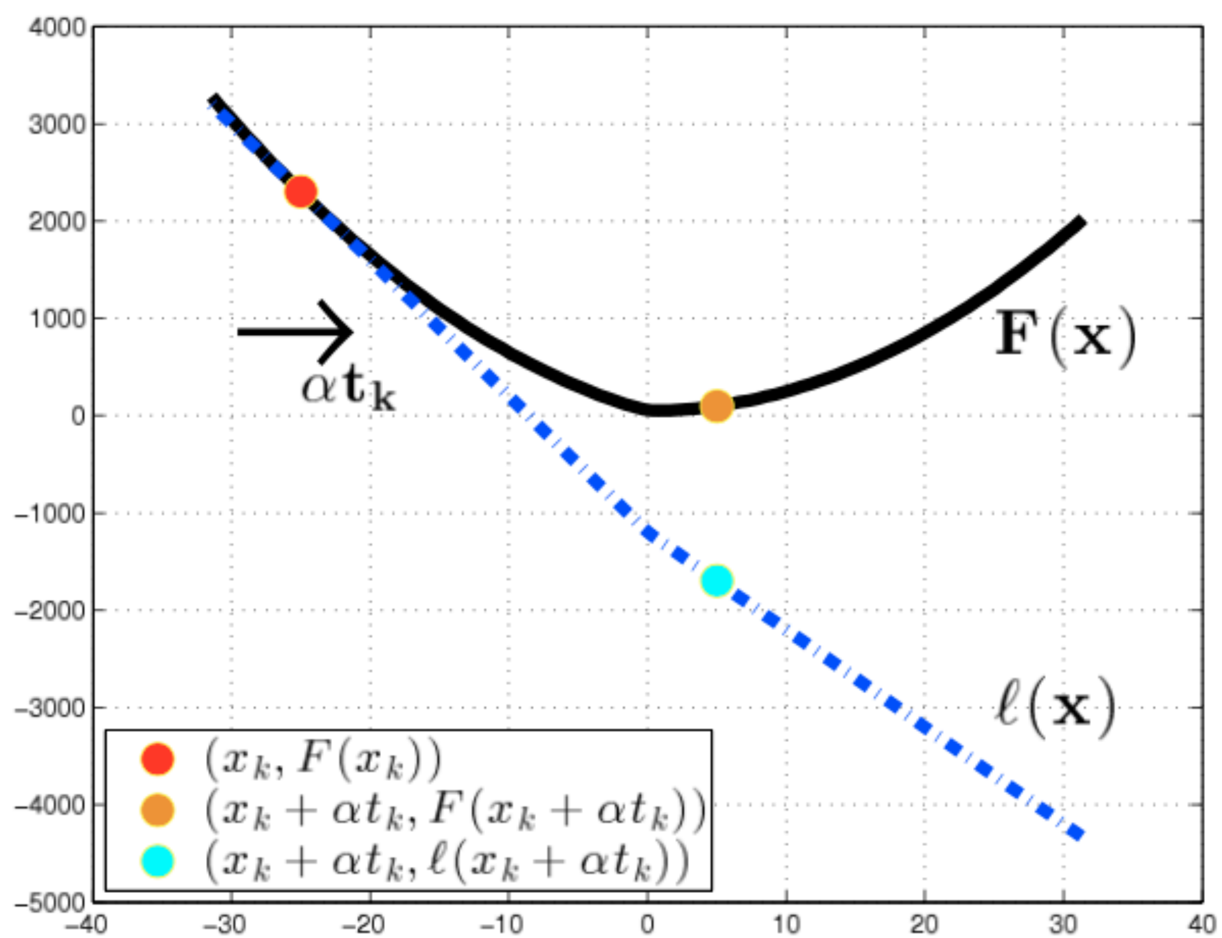
Armijo Line-Search: Intuition

- If $F(x_k) - F(x_k + t_k)$ (purple dashed line) is larger than $\theta\left(\ell(x_k) - \ell(x_k + t_k)\right)$(green solid line), then I stop.



Armijo Line-Search: Intuition

- Otherwise, I have to decrease α, and try again.



Comments on Armijo Line-Search

- In case you are curious, Armijo line-search for proximal gradient is a generalization of Armijo line-search for gradient descent.

- If you set $g(x) = 0$, then the procedure reduces to the same Armijo line-search that you know for gradient descent.

Termination of Armijo line-search

- Any $\alpha \leq \frac{2(1-\theta)}{L}$ satisfies the termination criterion of Armijo line-search for proximal gradient descent.

How do we terminate proximal gradient?

- Let's introduce the gradient mapping
$$G(x) := \frac{1}{\alpha}(x - x^+) = \frac{1}{\alpha}(x - \text{prox}_{\alpha g}(x - \alpha \nabla f(x))).$$
- where $\alpha > 0$.
- Use the norm of $\|G(x)\|_2$, to terminate proximal gradient when $\|G(x_k)\|_2 \leq \epsilon$.

How do we terminate proximal gradient?

- Why is $\|G(x)\|_2 \leq \epsilon$ a good metric for termination?
- This is because x^* is a stationary point if and only if $G(x^*) = 0$. (We proved this in the previous lecture).

Accelerated Proximal Gradient

- $x_k = \text{prox}_{\alpha_k g}\left(y_k - \alpha_k \nabla f(y_k)\right)$
- $t_{k+1} = \dfrac{1 + \sqrt{1 + 4t_k^2}}{2}$
- $y_{k+1} = x_k + \dfrac{t_k - 1}{t_{k+1}}(x_k - x_{k-1})$
- α_k can be computed by line-search
- This method is the same method as the one in Assignment 3 with the addition of the proximal operator.

Iteration Complexity

	Smoothing + Gradient Descent	Smoothing + Accelerated Gradient	Stochastic Sub-Gradient	Proximal Gradient	Accelerated Proximal Gradient
Non-convex	$\mathcal{O}\left(\frac{D}{\epsilon^2}\right)$	**??**	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$	$\mathcal{O}\left(\frac{L}{\epsilon}\right)$	**??**
Convex	$\mathcal{O}\left(\frac{D}{\epsilon^2}\right)$	$\mathcal{O}\left(\frac{\sqrt{D}}{\epsilon}\right)$	$\mathcal{O}\left(e^{\frac{\sigma^2}{\epsilon}}\right)$	$\mathcal{O}\left(\frac{L}{\epsilon}\right)$	$\mathcal{O}\left(\sqrt{\frac{L}{\epsilon}}\right)$
Strongly convex	$\mathcal{O}\left(\frac{D}{\delta\epsilon}\log\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\sqrt{\frac{D}{\delta\epsilon}}\log\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{G\sigma^2}{\delta^2}\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{L}{\delta}\log\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\sqrt{\frac{L}{\delta}}\log\frac{1}{\epsilon}\right)$

- Some constants might be different, but roughly they are of the same order.

References

- Book: First-order Methods in Optimization by A. Beck