Optimization for Data Science
Lecture 09: Accelerated Gradient

Kimon Fountoulakis

School of Computer Science
University of Waterloo

08/10/2019

Previous lecture: existence of an estimate sequence

- We proved that for convex and differentiable functions there exists an estimate sequence $\{\phi_k, \lambda_k, x_k\}_{k \in \mathbb{N}_0}$ with
- $\phi_0(x) := f(x_0) + \frac{L}{2}\|x - x_0\|_2^2$
- $\lambda_k \leq \frac{c}{k^2}$

Accelerated gradient

- We constructed a sequence in our attempt to prove this result. The sequence is given by the following algorithm:

Pick an arbitrary x_0
Set $\gamma_0 = 0$
Set $z_0 = x_0$
Set $\phi_0(x) := f(x_0) + \frac{L}{2}\|x - x_0\|_2^2$
while *termination criterion is not satisfied* **do**
  $y_{k-1} := (1 - \gamma_k)x_{k-1} + \gamma_k z_{k-1}$
  $\phi_k(x) := (1 - \gamma_k)\phi_{k-1} + \gamma_k(f(y_{k-1}) + \nabla f(y_{k-1})^T(x - y_{k-1}))$
  $z_k := \text{argmin}_{x \in \mathbb{R}^n}\ \phi_k(x)$
  $x_k := y_{k-1} - \frac{1}{L}\nabla f(y_{k-1})$
end

How to pick γ_k

- In this algorithm we cannot pick γ_k arbitrarily, it has to satisfy:
- $\gamma_k \in [0,1]$
- Let $\lambda_k := (1-\gamma_k)\lambda_{k-1}$ and $\gamma_0 = 1$, then γ_k must satisfy $\lambda_k/\gamma_k^2 \geq 1$.
- $\lambda_k \leq c/k^2$ for some constant $c > 0$.

How to pick γ_k

- It can be shown that by setting γ_k as
  - $\gamma_0 = \gamma_1 = \gamma_2 = \gamma_3 = 0$
  - $\gamma_k = 2/k$ for all $k \geq 4$
- Then the previous conditions are satisfied.

Simplified accelerated gradient

- However, accelerated algorithm is still complicated due to the step $z_k := \min_{x \in \mathbb{R}^n} \ \phi_k(x)$.

- This step can be simplified to

$$z_k := z_{k-1} - \frac{\gamma_k}{\lambda_k} \frac{1}{L} \nabla f(y_{k-1})$$

Simplified accelerated gradient

Set $\gamma_0 = 0$

Set $z_0 = x_0$

while *termination criterion is not satisfied* **do**

$$\gamma_k := \begin{cases} \frac{2}{k} & k \geq 4 \\ 0 & \text{otherwise} \end{cases}$$

$$y_{k-1} := (1 - \gamma_k) x_{k-1} + \gamma_k z_{k-1}$$

$$z_k := z_{k-1} - \frac{\gamma_k}{\lambda_k} \frac{1}{L} \nabla f(y_{k-1})$$

$$x_k := y_{k-1} - \frac{1}{L} \nabla f(y_{k-1})$$

end

Iteration complexity: smooth functions

- If $\nabla f(x)$ is Lipschitz continuous then

	Gradient Descent	**Accelerated Gradient**
Non-convex	$\mathcal{O}\left(\frac{L}{\epsilon}\right)$	-
Convex	$\mathcal{O}\left(\frac{L}{\epsilon}\right)$	$\mathcal{O}\left(\sqrt{\frac{L}{\epsilon}}\right)$
Strongly convex	$\mathcal{O}\left(\frac{L}{\delta}\log\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\sqrt{\frac{L}{\delta}}\log\frac{1}{\epsilon}\right)$

Iteration complexity: non-smooth functions

- These results were obtained by smoothing the objective function first. D is a constant due to smoothing.

	Gradient Descent	Accelerated Gradient
Non-convex	$\mathcal{O}\left(\frac{D}{\epsilon^2}\right)$	-
Convex	$\mathcal{O}\left(\frac{D}{\epsilon^2}\right)$	$\mathcal{O}\left(\frac{\sqrt{D}}{\epsilon}\right)$
Strongly convex	$\mathcal{O}\left(\frac{D}{\delta\epsilon}\log\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\sqrt{\frac{D}{\delta\epsilon}}\log\frac{1}{\epsilon}\right)$

Optimization for Data Science
Lecture 09: Stochastic Gradient

Kimon Fountoulakis

School of Computer Science
University of Waterloo

08/10/2019

Outline of this lecture

- ML loss functions
- Stochastic gradient
- Iteration complexity
- Practical issues

Data

- We are given n data points $(a_i, b_i)\ i = 1, \ldots, n$.
- $a_i \in \mathbb{R}^d$ is a sample/vector of length d, i.e., for each sample we have d features.
- b_i are the given labels. It can be real or binary or more generally an integer.

Examples of Loss Functions

- **Squared error:** $f_i(x) := \frac{1}{2}(a_i^T x - b_i)^2$, where $a_i \in \mathbb{R}^d$ is the i-th sample with d features and $b_i \in \mathbb{R}$ is the label.
- **Absolute error:** $|a_i^T x - b_i|$, robust to outliers.
- **Hinge loss:** $\max\{0, 1 - b_i a_i^T x\}$, better for binary labels b_i
- **Logistic loss:** $\log(1 + \exp(-b_i a_i^T x))$, better for binary labels b_i and also it's a smooth function.

Finite-Sum Optimization Problems

- We will consider optimization problems of this form

$$\min_{x \in \mathbb{R}^n} \; r(x) + \frac{1}{n} \sum_{i=1}^{n} \bar{f}_i(x)$$

- Function $r(x)$ is the regularization, e.g., $r(x) := \frac{\lambda}{2}\|x\|_2^2$, where $\lambda > 0$ is the regularization parameter.

- Functions $\bar{f}_i(x)$ measure loss w.r.t given labels

Standard Form

- By setting $f_i(x) := \bar{f}_i(x) + r(x)$, we will convert the above problem to the "standard" finite-sum form:

$$\min_{x \in \mathbb{R}^n} \underbrace{\frac{1}{n} \sum_{i=1}^{n} f_i(x)}_{f(x)}$$

Gradient Descent (again)

- Assuming that each f_i is differentiable then gradient descent is equivalent to:
- $$x_{k+1} := x_k - \frac{1}{L}\frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x)$$
- where L is the Lipschitz constant of $\frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x)$.
- iteration cost is linear in n, since we have to compute n gradients, one for each sample f_i.

Stochastic Gradient Method

- Assuming that each f_i is differentiable then stochastic gradient is equivalent to:

- Pick randomly a sample i

- $x_{k+1} := x_k - \alpha_k \nabla f_i(x)$

- iteration cost is now independent of n, but how many iterations?

Comments on Stochastic Gradient

- Proving the worst-case iteration complexity result also requires $\mathbb{E}[\|\nabla f_i x(_k)\|_2^2] \leq B^2$, where B^2 is a constant.

- There are multiple strategies for setting the step-sizes α_k and each one gives a different worst-case iteration complexity result.

Comments on Stochastic Gradient

- For example, for convex functions we have the following:
- If $B^2 = 0$, then by setting $\alpha_k = 1/L$, we get the usual rate $\mathcal{O}(1/t)$.
- If $\alpha_k = 1/k$, then we get rate $\mathcal{O}(1/\log t)$
- If $\alpha_k = 1/\sqrt{k}$, then we get rate $\mathcal{O}(\log t/\sqrt{t})$
- If $\alpha_k = \alpha$ for some constant α, then we get rate $\mathcal{O}(1/t + \alpha)$. This means implies that the minimum expected norm of the gradient will never go to zero and then algorithm only converges to a neighborhood of a stationary point. Also, this rate appears to be converging sub-linearly initially, but then stagnates!

Iteration complexity for smooth functions

- If $\nabla f(x)$ is Lipschitz continuous, and $\mathbb{E}\left[\|\nabla f_i(x_k)\|_2^2\right] \leq B^2$ (for stochastic gradient), then

	Gradient Descent	Accelerated Gradient	Stochastic Gradient
Non-convex	$\mathcal{O}\left(\frac{L}{\epsilon}\right)$	-	$\mathcal{O}\left(LB^2 e^{\frac{1}{\epsilon}}\right)$
Convex	$\mathcal{O}\left(\frac{L}{\epsilon}\right)$	$\mathcal{O}\left(\sqrt{\frac{L}{\epsilon}}\right)$	$\mathcal{O}\left(LB^2 e^{\frac{1}{\epsilon}}\right)$
Strongly convex	$\mathcal{O}\left(\frac{L}{\delta}\log\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\sqrt{\frac{L}{\delta}}\log\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{LB^2}{\delta}\frac{1}{\epsilon}\right)$

Comments on Stochastic Gradient

- Stochastic gradient has sub-linear rate even for strongly convex functions where full-gradient methods have linear convergence.

- However, each iteration is much cheaper since the iteration cost in independent of the number of samples n.

- Stochastic methods converge very fast to a low accuracy solution but then they stagnate.

Why would we consider stochastic methods?

- For two reasons

  - Extremely cheap iteration cost. Very good when we have millions (or more) data points and each data point is low-dimensional. This is good when we do not want to access all data points at each iteration.

  - Fast convergence to low accuracy. For most AI/ML problems a low accuracy solution to the optimization problem is good enough to obtain high accuracy w.r.t domain metrics of measuring performance, e.g., precision/recall.

Why would we consider stochastic methods?

- Stochastic Gradient is used everywhere in ML nowadays.
- In contrast to what theory says, practitioners use fixed step-sizes.
- Usually, you start with a large step-size which you keep for few iterations, then if there is no progress you decrease the step-size.

Why would we consider stochastic methods?

- Recently, there have been many variants of stochastic gradient that fix in practice the slow convergence issue. We will study some of these in the next lectures. Some examples are
  - Hybrid Deterministic-Stochastic
  - Mini-batching
  - Accelerated methods
  - Averaging gradients

Termination Criterion of Stochastic Gradient

- Usually we terminate an algorithm if the norm of the gradient is small.

- However, in this algorithm we only access one sample per iteration. This means that we cannot compute the norm of the whole gradient.

- If we do compute the norm of the whole gradient at each iteration, then this defeats the purpose of stochastic gradient.

Some options for terminating stochastic gradient

- Predefined maximum number of iterations
- Predefined upper bound for the running time
- Stop when the norm of the gradient of the chosen sample is small.
- Measure the norm of the whole gradient every n iterations, if it is small then terminate.
- Use your validation data and terminate the algorithm when precision/recall (or your preferred metric) are large enough.

Some options for terminating stochastic gradient

- Measure validation error every n iterations
- Stop if the validation error starts overfitting

Non-smooth loss functions

- What if the loss functions f_i are non-smooth? We have two options
  - Smooth them and then use stochastic-gradient (previous lectures)
  - Use stochastic sub-gradient method (this lecture)

Sub-differential and sub-gradient

- We can generalize the notion of a gradient by using the definition of convexity.

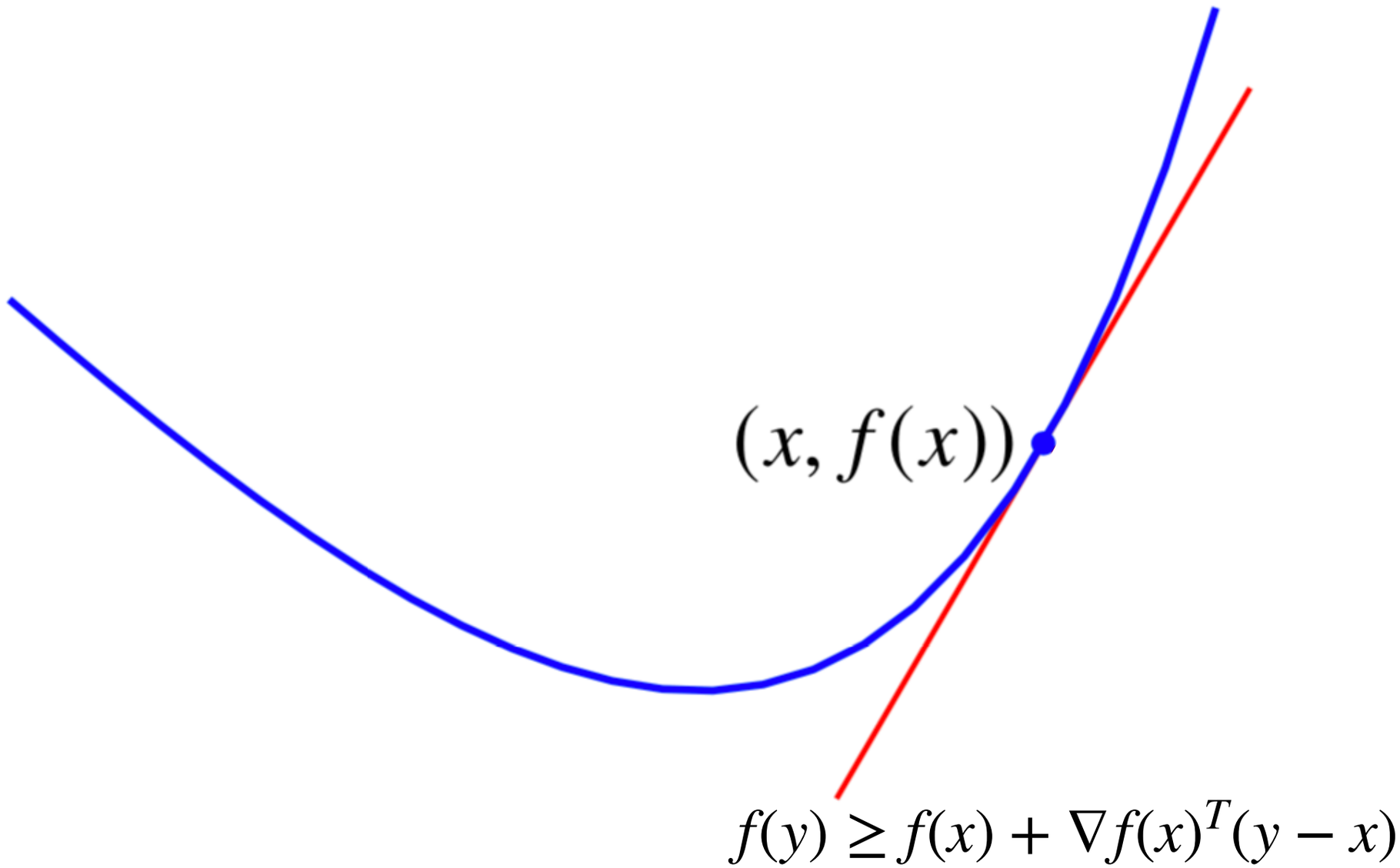
- Recall that the definition of convexity is:
$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

- We define the **sub-differential set at** x as**:**
$$\partial f(x) := \{g \in \mathbb{R}^n \mid f(y) \geq f(x) + g^T (y - x) \forall y \in \mathbb{R}^n\}$$

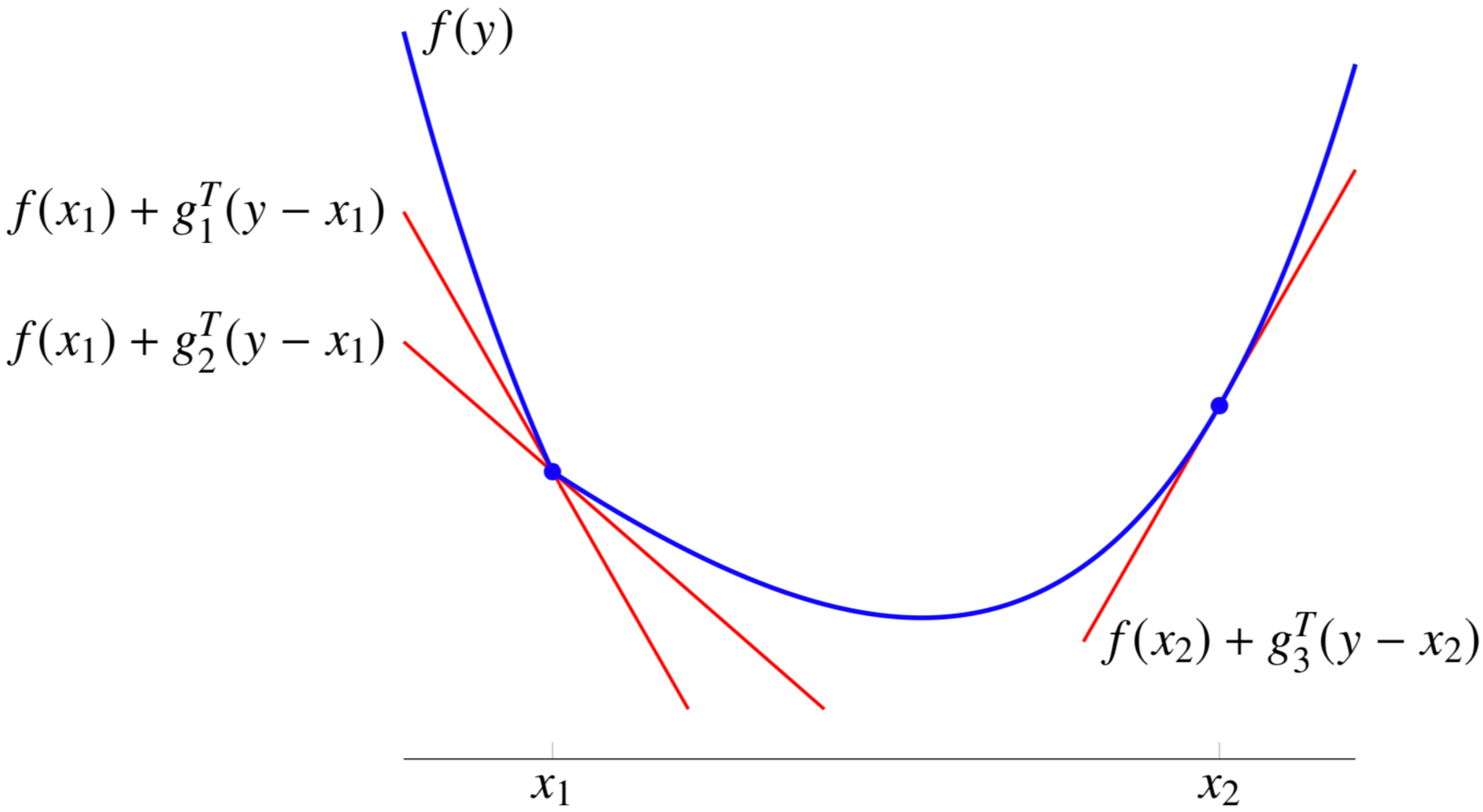
- We call an element $g \in \partial f(x)$ a **sub-gradient** of f at x.

This part of the lecture is mostly based on L. Vandenberghe lecture.

Visualization: lower bound of convex smooth function



Visualization: lower bound of convex non-smooth function



Optimality conditions for non-smooth functions

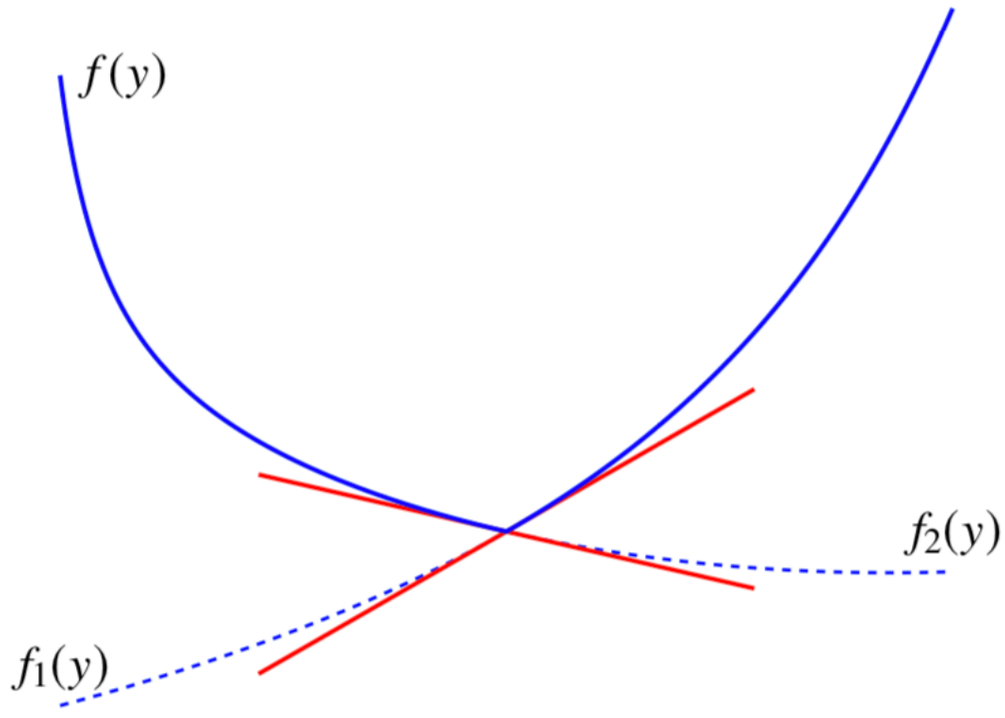
- A point x^* is a minimizer iff $0 \in \partial f(x^*)$.

- This means that there exists a sub-gradient at x^* which is equal to zero.

- This follows directly from the definition of sub-gradient:
  $f(y) \geq f(x^*) + 0^T(y - x^*)\ \forall y \iff 0 \in \partial f(x^*)$

Example

- Define $f(x) := \max(f_1(x), f_2(x))$, where f_1, f_2 are convex and differentiable.



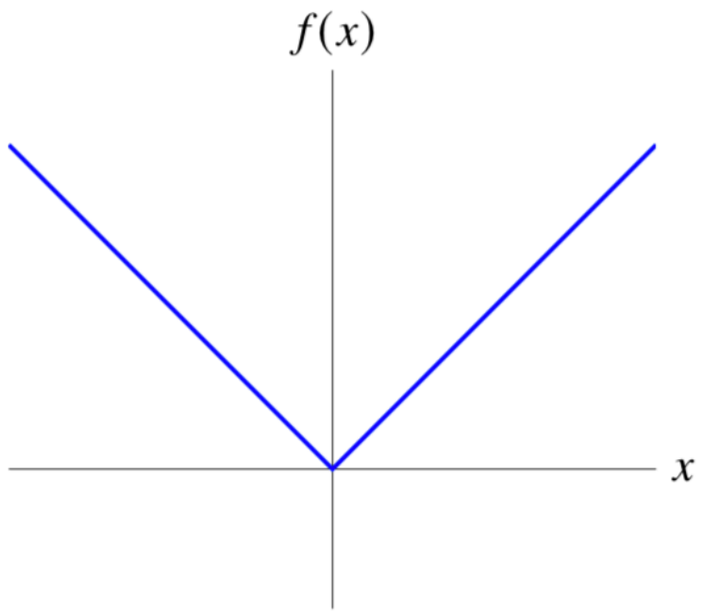
- If $f_1(x) = f_2(x)$ for some x, then
  $\partial f(x) := \left\{ g \in \mathbb{R}^n \mid t\nabla f_1(x) + (1-t)\nabla f_2(x) \ \forall t \in [0,1] \right\}$

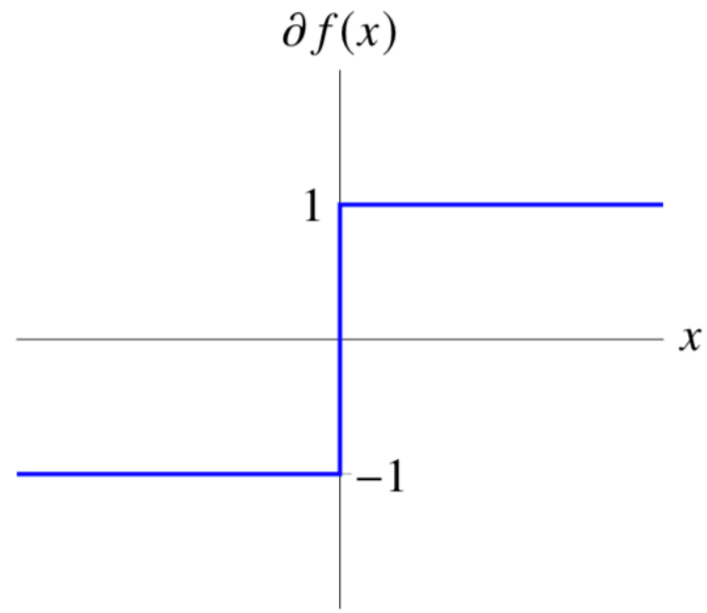
- If $f_1(x) > f_2(x)$ for some x, then $\partial f(x) := \nabla f_1(x)$

- If $f_1(x) < f_2(x)$ for some x, then $\partial f(x) := \nabla f_2(x)$

Example

- Define $f(x) := |x|$





- $\partial f(x) = 1$ if $x > 0$
- $\partial f(x) = -1$ if $x < 0$
- $\partial f(x) \in [-1,1]$ if $x = 0$

Sub-gradient calculus

- Weak sub-gradient calculus: rules for finding one sub-gradient
  - Sufficient for most algorithms for non-smooth convex problems
  - If you can evaluate $f(x)$, you can usually compute a sub-gradient
- Strong sub-gradient calculus: rules for finding $\partial f(x)$ (all sub-gradients)
  - Some algorithms, optimality conditions need the entire sub-differential.
  - Usually, it is difficult to compute

Next lecture

- More sub-gradient calculus and more examples
- Stochastic **sub**-gradient method.
- Iteration complexity
- Commentary on how to implement stochastic (sub)gradient methods.
- More advanced stochastic methods