Lecture 10 Machine Learning and Optimization

- Examples of computing subgradients
- Example of using Fermat's condition to solve a nonsmooth, constrained optimization problem
- Gradient descent
- Three notions of convergence
- $\beta$-smoothness
- Iteration complexity & convergence rates
- Projected gradient descent
- Subgradient descent

We'll start with two examples to practice computing subgradients and using them to solve optimization problems.

**Ex**

Compute subgradient of $\ell_1$-regularized objective

$$f(\omega) = \frac{1}{n}\sum_{i=1}^{n}(1 - y_i \omega^T x_i)_+ + \lambda \|\omega\|_1$$

First consider the regularizer

$$r(\omega) = \|\omega\|_1 = \sum_{i=1}^{d} |\omega_i| = \sum_{i=1}^{d} |e_i^T \omega|$$

Consider a single term $|e_i^T \omega|$

Use the affine transformation rule on $g(\omega) = |e_i^T \omega|$ and our knowledge that if $m(x) = |x|$, then

$$\partial m(x) = \begin{cases} -1 & x < 0 \\ 1 & x > 0 \\ [-1,1] & x = 0 \end{cases} = \text{sign}(x)$$

to conclude

$$\partial g(\omega) = e_i \, \text{sign}(e_i^T \omega) = e_i \, \text{sign}(\omega_i)$$

Now use sum rule of subdifferentials to conclude

$$\partial r(\omega) = \sum_{i=1}^{d} e_i \, \text{sign}(\omega_i)$$

$$\partial \left\| \begin{bmatrix} -3 \\ 2 \\ 0 \end{bmatrix} \right\|_1 = \left\{ v \;\middle|\; v = \begin{bmatrix} -1 \\ 1 \\ [-1,1] \end{bmatrix} \right\}$$

Consider a single data fitting term

$$q(\omega) = (1 - y_i \omega^T x_i)_+$$

Aside: the affine transformation rule,

if $f$ convex, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^n$ and $g(x) = f(Ax+b)$

then

$$\partial g(x) = A^T \partial f(Ax+b)$$

is a generalization of the fact that if $f$ is differentiable,

$$\nabla g(x) = A^T \nabla f(Ax+b)$$

Pf

$$[\nabla g(x)]_i = \frac{\partial g}{\partial x_i}(x) = \sum_{j=1}^{m} \frac{\partial f}{\partial z_j}(Ax+b) \frac{\partial}{\partial x_i}(Ax+b)_j$$

$$= \sum_{j=1}^{m} [\nabla f(Ax+b)]_j \frac{\partial (Ax)_j}{\partial x_i}$$

and $(Ax)_j = \sum_{l=1}^{n} A_{jl} x_l$, so $\frac{\partial}{\partial x_i}(Ax)_j = A_{ji}$, and

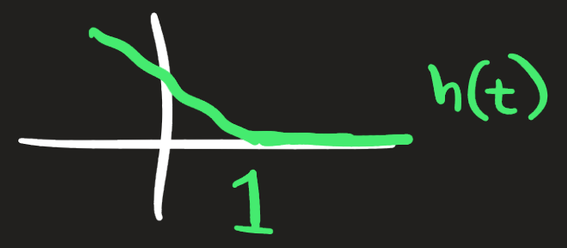
$$= \sum_{j=1}^{m} A_{ji} [\nabla f(Ax+b)]_j = \sum_{j=1}^{m} (A^T)_{ij} [\nabla f(Ax+b)]_j$$

$$= [A^T \nabla f(Ax+b)]_i$$

∎

We use the affine transformation rule again. Let $h(x) = (1-x)_+$ and recall

$$\partial h(x) = \begin{cases} -1 & x < 1 \\ [-1,0] & x = 1 \\ 0 & x > 1 \end{cases}$$




Then since $g(\omega) = h(y_i x_i^T \omega)$, we have that

$$\partial g(\omega) = y_i x_i \begin{cases} -1 & y_i x_i^T \omega < 1 \\ [-1,0] & y_i x_i^T \omega = 1 \\ 0 & y_i^T x_i^T \omega > 0 \end{cases}$$

Now we combine the data-fitting terms & regularizer by noting $f$ is a positively weighted sum of these terms

$$\partial f(\omega) = \frac{1}{n} \sum_{i=1}^{n} y_i x_i \begin{cases} -1 & y_i x_i^T \omega < 1 \\ [-1,0] & y_i x_i^T \omega = 1 \\ 0 & y_i x_i^T \omega = 0 \end{cases} + \lambda \sum_{i=1}^{d} e_i \operatorname{sign}(\omega_i)$$

Consider the contribution of a single training point to the subdifferential

$$(*)\quad y_i x_i \begin{cases} -1 & y_i x_i^T \omega < 1 \\ [-1, 0] & y_i x_i^T \omega = 1 \\ 0 & y_i x_i^T \omega > 1 \end{cases}$$

$y_i x_i^T \omega < 1$ — prediction is bad
$y_i x_i^T \omega = 1$ — prediction is ok
$y_i x_i^T \omega > 1$ — prediction is good

We update the model by picking a subgradient $v$ from the subdifferential and moving in the negative direction:

$$\omega_{t+1} \leftarrow \omega_t - \alpha v \qquad \text{where } v = y_i x_i c \text{ according to } (*)$$

We can interpret this update as improving the performance on $(x_i, y_i)$:

$$y_i x_i^T \omega_{t+1} = y_i x_i^T (\omega_t - \alpha v) = y_i x_i^T \omega_t - \alpha y_i x_i^T v$$
$$= y_i x_i^T \omega_t - \alpha y_i^2 \|x_i\|_2^2 c = y_i x_i^T \omega_t - \alpha c \|x_i\|_2^2$$

If:

$y_i x_i^T \omega < 1$ $(c = -1)$ : $y_i x_i^T \omega_{t+1} = y_i x_i^T \omega_t + \alpha \|x_i\|_2^2 > y_i x_i^T \omega$ prediction improved!

$y_i x_i^T \omega > 1$ $(c = 1)$ : $y_i x_i^T \omega_{t+1} = y_i x_i^T \omega_t$ prediction stayed good

**Ex** Compute the projection onto the $\|\cdot\|_\infty$ ball

$$P_C(x) = \underset{\|z\|_\infty \le 1}{\operatorname{argmin}} \ \|x - z\|_2^2$$


First, convert to an unconstrained optimization problem

$$z^* = P_C(x) = \underset{z}{\operatorname{argmin}} \ \frac{1}{2}\|x - z\|_2^2 + i_C(z)$$

where $C = \{v \mid \|v\|_\infty \le 1\}$. Apply Fermat's condition,

$$0 \in z^* - x + \partial i_C(z^*)$$

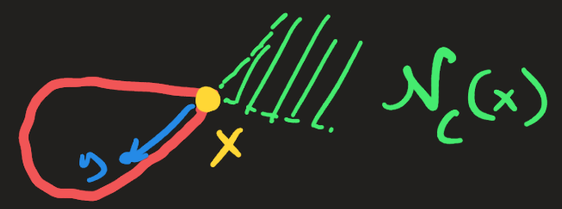
This means if we can find a closed form solution to this equation, we have an expression for $P_C(x)$

$$i_C(y) \geq \langle v, y-x \rangle$$

Recall that if $C$ is a convex set, then

$$\partial i_C(x) = \{ v : \langle v, y-x \rangle \leq 0 \text{ for all } y \in C \}$$

$$= N_C(x)$$




This looks difficult to apply: given an arbitrary $z \in C$, how to compute $N_C(z)$?

Luckily, this problem is <u>separable</u>

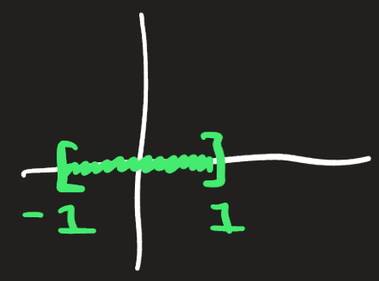
$$z^* = \underset{\|z\|_\infty \leq 1}{\operatorname{argmin}} \|x - z\|_2^2 \iff z_i^* = \underset{|z| \leq 1}{\operatorname{argmin}} |x_i - z|_2^2$$

$$\iff 0 \in z_i^* - x_i + \partial I_{[-1,1]}(z_i^*)$$

$$\vdots$$

and

$$\mathcal{N}_{[-1,1]}(z) = \begin{cases} \mathbb{R}_+ & z = 1 \\ -\mathbb{R}_+ & z = -1 \\ 0 & z \in (-1,1) \end{cases}$$




Now we know that $z_i^* \in [-1,1]$ so there are three cases to consider

<u>Maybe $z_i^* = 1$</u>

This would be the case if

$$0 \in 1 - x_i + \mathbb{R}_+$$
$$\Leftrightarrow \quad x_i - 1 \in \mathbb{R}_+$$
$$\Leftrightarrow \quad x_i \geq 1$$

<u>Maybe $z_i^* = -1$</u>

This requires

$$0 \in -1 - x_i - \mathbb{R}_+$$
$$\Leftrightarrow \quad x_i + 1 \in -\mathbb{R}_+$$
$$\Leftrightarrow \quad x_i \leq -1$$

<u>Maybe $z_i^* \in (-1,1)$</u>

This requires

$$0 = z_i^* - x_i$$
$$\Rightarrow \quad x_i = z_i^* \in (-1,1)$$

putting these cases together,

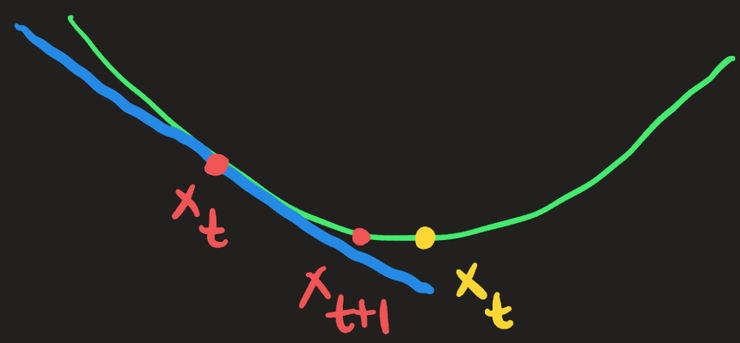
$$z_i^* = P_C(x)_i = \begin{cases} 1 & x_i \geq 1 \\ -1 & x_i \leq -1 \\ x_i & \text{otherwise} \end{cases}$$

## Gradient descent

$f$ – convex, smooth

want to solve

$$x^* \in \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} f(x)$$




The basic idea is to note by TE expansion argument

$$f(x_{t+1}) \approx f(x_t) + \langle \nabla f(x_t), x_{t+1} - x_t \rangle$$

when $\|x_t - x_{t+1}\|_2$ is small

$$= f(x_t) + \langle \nabla f(x_t), \Delta \rangle$$

Take $\Delta = -\alpha \nabla f(x_t)$, where stepsize $\alpha$ is chosen so that $\|\Delta\|_2$ is small, then

$$f(x_{t+1}) \approx f(x_t) - \alpha \|\nabla f(x_t)\|_2^2 < f(x_t)$$

## Gradient Descent Algorithm

For solving

$$(U) \min_{x \in \mathbb{R}^d} f(x)$$ where $f$ is smooth & convex

**Input**

$\nabla f$ — gradient oracle

$T$ — iterations

$x_0$ — starting guess

$\alpha_1, \dots, \alpha_T$ — step sizes for each iteration

**NB:** This alg aims at minimizing $f$, doesn't necessarily give a good estimate of $x^*$!

**Output**

$x_1, \dots, x_T \in \mathbb{R}^d$ estimates of $x^*$

**Alg**

For $t = 1$ to $T$

$$x_t = x_{t-1} - \alpha_t \nabla f(x_{t-1})$$

We want to know reasonable assumptions on $f$ under which GD converges, so we can:

1) choose our stepsizes
2) choose the number of steps $T$ to reach approximate convergence

## Three notions of approximate convergence

1) $f(x_t) < f(x^*) + \epsilon$

   approximate convergence in function value

$\epsilon$-suboptimality - the focus for convex optimization

2) $\|x_t - x^*\|_2^2 < \epsilon$

   in parameter space

3) $\|\nabla f(x_t)\|_2 < \epsilon$

   of gradient to zero

$\epsilon$-stationarity - the focus for non-convex optim

The most basic condition for GD to work is that $f$ is $\beta$-smooth:

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq \beta \|x - y\|_2$$

i.e. not only is $f$ differentiable, its gradient is Lipschitz-continuous w/ Lipschitz constant $\beta$.

Ex of a $\beta$-smooth function

$$f(x) = \frac{1}{2}\|Ax - b\|_2^2 \qquad \nabla f(x) = A^T(Ax - b)$$

$$\|\nabla f(x) - \nabla f(y)\|_2 = \|A^TA(x-y)\|_2 \leq \|A^TA\|_2 \|x-y\|_2$$

so $f$ is $\beta$-smooth with $\beta = \|A^TA\|_2$

**Aside** The matrix (spectral, or operator, or 2-> norm

$$\|M\|_2 := \max_{\|x\|_2=1} \|Mx\|_2$$

measures the maximum amount by which $M$ stretches any input.

**Fact**

From the definition of $\|M\|_2$, we have that for any vector $v$,

$$\|Mv\|_2 = \|v\|_2 \left\|M \frac{v}{\|v\|_2}\right\|_2 \leq \|M\|_2 \|v\|_2,$$

so

$$\|Mv\|_2 \leq \|M\|_2 \|v\|_2$$

which says $\|M\|_2$ stretches any vector by at most $\|M\|_2$

Ex of a function that is not $\beta$-smooth for any $\beta$

$$f(x) = e^x \qquad f'(x) = e^x$$

Let $\beta$ be fixed. Take $x = \beta + 1$ and $y = \beta + 2$, then by the mean value theorem,

$$e^x - e^y = e^z(x-y) \quad \text{for some } z \in [x, y]$$

This means

$$|e^x - e^y| = e^z|x-y| > e^\beta |x-y| > \beta|x-y|$$

so we have shown $f$ is not $\beta$-smooth for any $\beta$.

## Consequences of $\beta$-smoothness

1) $\|\nabla f(x)\|_2 \leq \beta \|x - x^*\|_2$ if we are close to a minimizer, the gradient is small

2) We have a concrete bound on the accuracy of the 1st order TS-approximation everywhere

$$f(y) \approx f(x) + \nabla f(x)^T (y-x) \quad \text{when } \|y-x\|_2 \text{ small}$$

becomes

$$|f(y) - [f(x) + \nabla f(x)^T (y-x)]| \leq \frac{\beta}{2} \|x-y\|_2^2$$

This follows from a simple TS argument. See Chapter 3 of Convex Optimization, Bubeck if interested.

This allows us to show convergence of GD when we use the constant stepsize $\alpha_t = \frac{1}{\beta}$

**Descent Lemma**

When $f$ is $\beta$-smooth and $\alpha_t = \frac{1}{\beta}$, GD ensures that

$$f(x_t) - f(x_{t-1}) \leq -\frac{1}{2\beta} \|\nabla f(x_{t-1})\|_2^2$$

**Pf**

Take $y = x_t$ and $x = x_{t-1}$ in the previous slide and recall GD takes $x_t = x_{t-1} - \alpha \nabla f(x_{t-1})$ to see that

$$f(x_t) - \left[f(x_{t-1}) + \langle \nabla f(x_{t-1}), -\alpha \nabla f(x_{t-1}) \rangle\right]$$

$$\leq \frac{\beta}{2} \|-\alpha \nabla f(x_{t-1})\|_2^2$$

Simplifying both sides,

$$f(x_t) - f(x_{t-1}) + \alpha \|\nabla f(x_{t-1})\|_2^2 \leq \frac{\beta}{2}\alpha^2 \|\nabla f(x_{t-1})\|_2^2$$

$$\Rightarrow \quad f(x_t) - f(x_{t-1}) \leq \left(\frac{\beta}{2}\alpha^2 - \alpha\right) \|\nabla f(x_{t-1})\|_2^2$$

$$\Rightarrow \quad f(x_t) - f(x_{t-1}) \leq \left(\frac{1}{2\beta} - \frac{1}{\beta}\right) \|\nabla f(x_{t-1})\|_2^2$$

$$= \frac{-1}{2\beta} \|\nabla f(x_{t-1})\|_2^2$$

as claimed.

**Fact** If $f$ is convex and $\beta$-smooth, then gradient descent with $\alpha = \alpha_t = \frac{1}{\beta}$ ensures that

$$f(x_T) - f(x^*) \leq \frac{2\beta \|x_0 - x^*\|_2^2}{T}$$

We say that GD has $T = \frac{2\beta \|x_0 - x^*\|_2^2}{\epsilon}$

iteration complexity $O\left(\frac{\beta}{\epsilon}\right)$ — this is the number of iterations it takes to reach $\epsilon$-suboptimality

convergence rate $O\left(\frac{\beta}{T}\right)$ — this is how fast the $\epsilon$-suboptimality gap goes down, as a function of the number of iterations

What about constrained optimization?

## Projected Gradient Descent

$$(C) \quad \min_{C} f(x)$$

### Inputs

$\nabla f$ — gradient oracle

$P_C$ — projection oracle

$\alpha_1, \ldots, \alpha_T$ — step sizes

$x_0$ — initial iterate

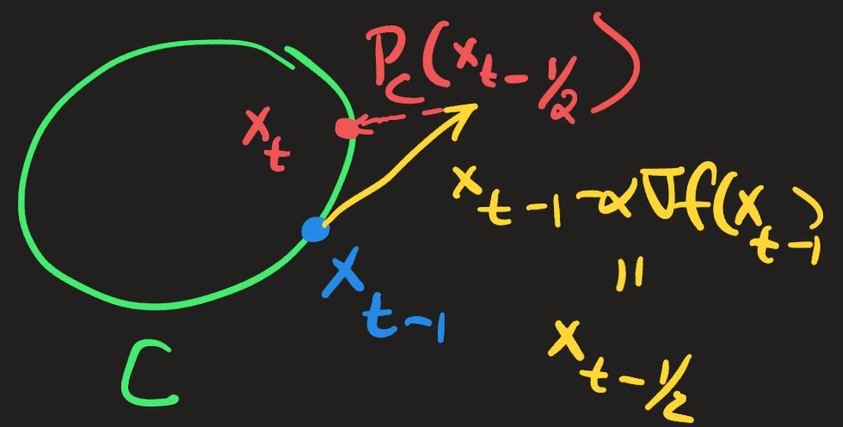
### Output

$x_1, \ldots, x_T$ — iterates

### Algorithm

For $t = 1$ to $T$:

$$x_t = P_C(x_{t-1} - \alpha_t \nabla f(x_{t-1}))$$



What about nonsmooth convex optimization?

## Subgradient Descent

$f$ - cvx, nonsmooth

$x^* \in \operatorname{argmin}_x f(x)$

### Inputs

$\partial f$ - subgradient oracle

$\alpha_1, \dots, \alpha_T$ - stepsizes

$x_0$ - starting iterate

### Alg

For $t = 1$ to $T$:

$g_{t-1} \leftarrow$ a vector in $\partial f(x_{t-1})$

$x_t \leftarrow x_{t-1} - \alpha_t g_{t-1}$

### Outputs

(use Polyak averaging)

$$\hat{x} = \frac{1}{T} \sum_{t=1}^{T} x_t$$

### Guarantee

$$f(\hat{x}) - f(x^*) \leq O\left(\frac{1}{\sqrt{T}}\right)$$