CS-GY 6763: Lecture 6
Gradient Descent and Projected Gradient Descent

NYU Tandon School of Engineering, Prof. Christopher Musco

Administrative

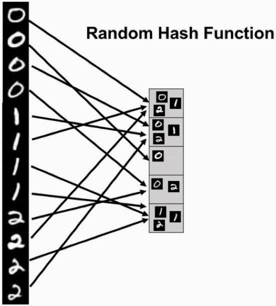
- Homework 3 due on Monday.
- Exam on Friday. 1 hour 15 minutes, cheat sheet allowed.

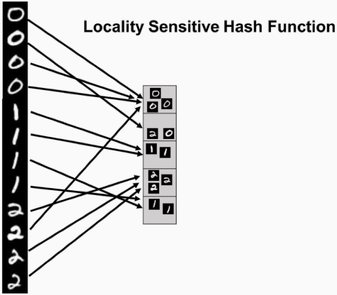
FINISH UP LSH + NEAR NEIGHBOR SEARCH

Let $h : \mathbb{R}^d \to \{1, \ldots, m\}$ be a random hash function.

We call h locality sensitive for similarity function $s(\mathbf{q}, \mathbf{y})$ if $\Pr[h(\mathbf{q}) == h(\mathbf{y})]$ is:

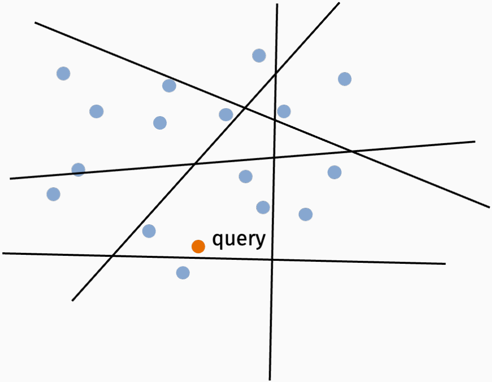
- Higher when $\mathbf{q}$ and $\mathbf{y}$ are more similar, i.e. $s(\mathbf{q}, \mathbf{y})$ is higher.
- Lower when $\mathbf{q}$ and $\mathbf{y}$ are more dissimilar, i.e. $s(\mathbf{q}, \mathbf{y})$ is lower.





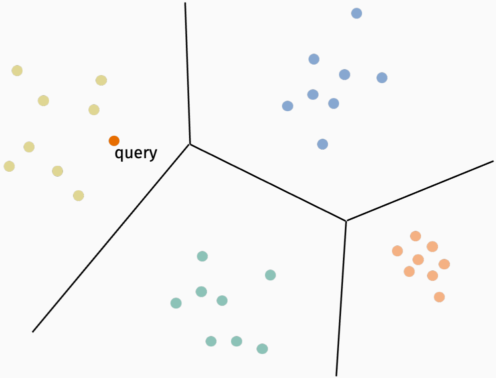
LSH is widely used in practice, but is starting to get replaced by other methods. Most of these are data dependent in some way.

Starting point: Think of LSH as a randomized space-partitioning method.

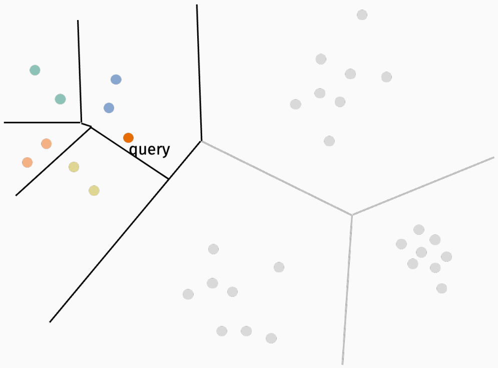


In practice, we can often get partitions with better margin but partitioning in a data-dependent way.

Common approach: Split data using k-means clustering.

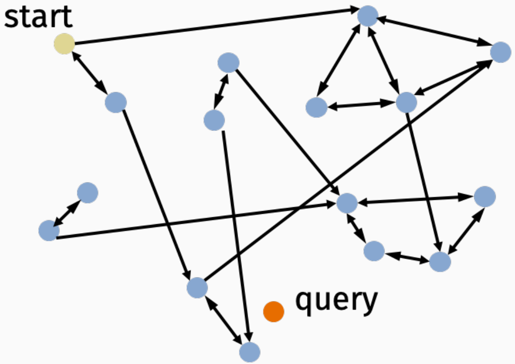


Common approach: Split data using k-means clustering.



Main approach behind "k-means tree" and "inverted file index" based near-neighbor search methods like Meta's FAISS library and Google's SCANN.

New kid on the block: Graph-based nearest neighbor search.



Idea behind methods like NSG, HNSW, DiskANN, etc. Inspired by Milgram's famous "small-world" experiments from the 1960's.

Can we better explain the success of data-dependent nearest-neighbor search methods?

LSH Forest: Practical Algorithms Made Theoretical

Alexandr Andoni, Columbia University
Ilya Razenshteyn, MIT CSAIL
Negev Shekel Nosatzki, Columbia University

Beyond Locality–Sensitive Hashing

Alexandr Andoni, Microsoft Research SVC
Piotr Indyk, MIT
Huy L. Nguyễn, Princeton
Ilya Razenshteyn, MIT

Abstract

We present a new data structure for the c–approximate near neighbor problem (ANN) in the Euclidean space. For n points in $\mathbb{R}^d$, our algorithm achieves $O_c(dn^\rho)$ query time and $O_c(n^{1+\rho} + nd)$ space, where $\rho \leq 7/(8c^2)$ … over the result by Andoni and Indyk (FC… a locality–sensitive hashing lower bound … a standard reduction we obtain a data … $\rho \leq 7/(8c) + O(1/c^{3/2}) + o_c(1)$, which is … Motwani (STOC 1998).

Worst-case Performance of Popular Approximate Nearest Neighbor Search Implementations: Guarantees and Limitations

Piotr Indyk
MIT
indyk@mit.edu

Haike Xu
MIT
haikexu@mit.edu

OPTIMIZATION

Have some function $f : \mathbb{R}^d \to \mathbb{R}$. Want to find $\mathbf{x}^*$ such that:

$$f(\mathbf{x}^*) = \min_{\mathbf{x}} f(\mathbf{x}).$$

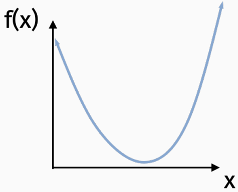
Or at least $\hat{\mathbf{x}}$ which is close to a minimum. E.g.

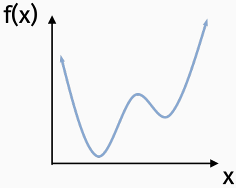
$$f(\hat{\mathbf{x}}) \leq \min_{\mathbf{x}} f(\mathbf{x}) + \epsilon.$$

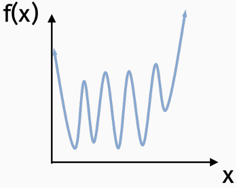
Often we have some additional constraints:

- $\mathbf{x} > 0$.
- $\|\mathbf{x}\|_2 \leq R$, $\|\mathbf{x}\|_1 \leq R$.
- $\mathbf{a}^T\mathbf{x} = c$.

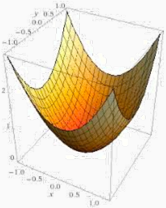
Dimension $d = 1$:

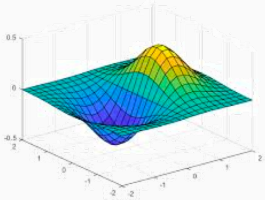


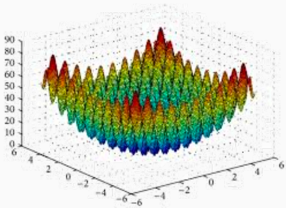




Dimension $d = 2$:







Continuouos optimization is the foundation of modern machine learning.

Supervised learning: Want to learn a model that maps <u>inputs</u>

- numerical data vectors
- images, video
- a sequence of tokens/works

to <u>predictions</u>

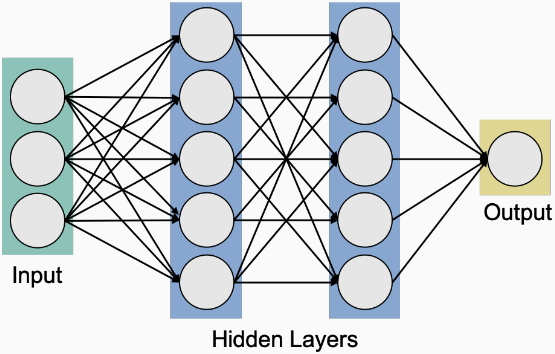
- numerical value (probability stock price increases)
- label (does the image contain a car? what is the next token in the sequence?)
- decision (turn car left, rotate robotic arm)

Let $M_{\mathbf{x}}$ be a model with parameters $\mathbf{x} = \{x_1, \ldots, x_k\}$, which takes as input a data vector $\mathbf{a}$ and outputs a prediction.

Example:

$$M_{\mathbf{x}}(\mathbf{a}) = \operatorname{sign}(\mathbf{a}^T \mathbf{x})$$

Example:



$x \in \mathbb{R}^{(\text{\# of connections})}$ is the parameter vector containing all the network weights.

Classic approach in supervised learning: Find a model that works well on data that you already have the answer for (labels, values, classes, etc.).

- Model $M_{\mathbf{x}}$ parameterized by a vector of numbers $\mathbf{x}$.
- Dataset $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(n)}$ with outputs $y^{(1)}, \ldots, y^{(n)}$.

Want to find $\hat{\mathbf{x}}$ so that $M_{\hat{\mathbf{x}}}(\mathbf{a}^{(i)}) \approx y^{(i)}$ for $i \in 1, \ldots, n$.

How do we turn this into a function minimization problem?

Loss function $L(M_{\mathsf{x}}(\mathsf{a}), y)$**:** Some measure of distance between prediction $M_{\mathsf{x}}(\mathsf{a})$ and target output y. Increases if they are further apart.

- Squared (ℓ_2) loss: $|M_{\mathsf{x}}(\mathsf{a}) - y|^2$
- Absolute deviation (ℓ_1) loss: $|M_{\mathsf{x}}(\mathsf{a}) - y|$
- Hinge loss: 1 - $y \cdot M_{\mathsf{x}}(\mathsf{a})$
- Cross-entropy loss (log loss).

Empirical risk minimization: Given a training dataset $(\mathbf{a}^{(1)}, y^{(1)}) \ldots, (\mathbf{a}^{(n)}, y^{(n)})$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} L\left(M_{\mathbf{x}}(\mathbf{a}^{(i)}), y^{(i)}\right)$$

Solve the optimization problem $\min_{\mathbf{x}} f(\mathbf{x})$.

- $M_{\mathbf{x}}(\mathbf{a}) = \mathbf{x}^T\mathbf{a}$. $\mathbf{x}$ contains the regression coefficients.
- $L(z, y) = |z - y|^2$.
- $f(\mathbf{x}) = \sum_{i=1}^{n} |\mathbf{x}^T\mathbf{a}^{(i)} - y^{(i)}|^2$

$$f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2$$

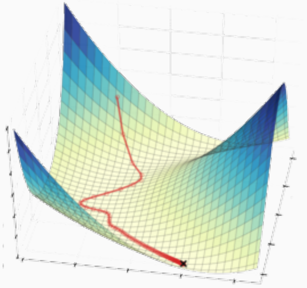
where $\mathbf{A}$ is a matrix with $\mathbf{a}^{(i)}$ as its i^{th} row and $\mathbf{y}$ is a vector with $y^{(i)}$ as its i^{th} entry.

The choice of algorithm to minimize $f(\mathbf{x})$ will depend on:

- The form of $f(\mathbf{x})$ (is it linear, is it quadratic, does it have finite sum structure, etc.)
- If there are any additional constraints imposed on $\mathbf{x}$. E.g. $\|\mathbf{x}\|_2 \leq c$.

What are some example algorithms for continuous optimization?

Gradient descent: A greedy algorithm for minimizing functions of multiple variables that often works amazingly well.



Runtime generally scales linearly with the dimension of $\mathbf{x}$ (although this is a bit of an over-simplification).

- Cutting plane methods (e.g. center-of-gravity, ellipsoid)
- Interior point methods

Faster and more accurate in low-dimensions, slower in very high dimensions. Generally runtime scales polynomially with the dimension of $\mathbf{x}$ (e.g., $O(d^3)$).

For $i = 1, \ldots, d$, let x_i be the i^{th} entry of $\mathbf{x}$. Let $\mathbf{e}^{(i)}$ be the i^{th} standard basis vector.

Partial derivative:

$$\frac{\partial f}{\partial x_i}(\mathbf{x}) = \lim_{t \to 0} \frac{f(\mathbf{x} + t\mathbf{e}^{(i)}) - f(\mathbf{x})}{t}$$

Directional derivative:

$$D_{\mathbf{v}} f(\mathbf{x}) = \lim_{t \to 0} \frac{f(\mathbf{x} + t\mathbf{v}) - f(\mathbf{x})}{t}$$

Gradient:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \frac{\partial f}{\partial x_2}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_d}(\mathbf{x}) \end{bmatrix}$$

Directional derivative:

$$D_{\mathbf{v}} f(\mathbf{x}) = \lim_{t \to 0} \frac{f(\mathbf{x} + t\mathbf{v}) - f(\mathbf{x})}{t} = \nabla f(\mathbf{x})^T \mathbf{v}.$$

Given a function f to minimize, assume we have:

- **Function oracle**: Evaluate $f(\mathbf{x})$ for any $\mathbf{x}$.
- **Gradient oracle**: Evaluate $\nabla f(\mathbf{x})$ for any $\mathbf{x}$.

We view the implementation of these oracles as black-boxes, but they can often require a fair bit of computation.

Linear least-squares regression:

- Given $\mathbf{a}^{(1)}, \ldots \mathbf{a}^{(n)} \in \mathbb{R}^d$, $y^{(1)}, \ldots y^{(n)} \in \mathbb{R}$.
- Want to minimize:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left(\mathbf{x}^T \mathbf{a}^{(i)} - y^{(i)}\right)^2 = \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2.$$

What is the time complexity to implement a function oracle for $f(\mathbf{x})$?

Linear least-squares regression:

- Want to minimize:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left(\mathbf{x}^T \mathbf{a}^{(i)} - y^{(i)}\right)^2 = \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2.$$

$$\frac{\partial f}{\partial x_j} = \sum_{i=1}^{n} 2\left(\mathbf{x}^T \mathbf{a}^{(i)} - y^{(i)}\right) \cdot a_j^{(i)} = 2\boldsymbol{\alpha}^{(j)T}(\mathbf{A}\mathbf{x} - \mathbf{y})$$

where $\boldsymbol{\alpha}^{(j)}$ is the j^{th} <u>column</u> of $\mathbf{A}$.

Linear least-squares regression:

$$\frac{\partial f}{\partial x_j} = \sum_{i=1}^{n} 2\left(\mathbf{x}^T\mathbf{a}^{(i)} - y^{(i)}\right) \cdot a_j^{(i)} = 2\boldsymbol{\alpha}^{(j)^T}(\mathbf{A}\mathbf{x} - \mathbf{y})$$

where $\boldsymbol{\alpha}^{(j)}$ is the j^{th} column of $\mathbf{A}$.

$$\nabla f(\mathbf{x}) = 2\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{y})$$

What is the time complexity of a gradient oracle for $\nabla f(\mathbf{x})$?

Greedy approach: Given a starting point $\mathbf{x}$, make a small adjustment that decreases $f(\mathbf{x})$. In particular, $\mathbf{x} \leftarrow \mathbf{x} + \eta\mathbf{v}$.

What property do I want in $\mathbf{v}$?

Leading question: When η is small, what's an approximation for $f(\mathbf{x} + \eta\mathbf{v}) - f(\mathbf{x})$?

$$f(\mathbf{x} + \eta\mathbf{v}) - f(\mathbf{x}) \approx$$

$$D_{\mathbf{v}} f(\mathbf{x}) = \lim_{t \to 0} \frac{f(\mathbf{x} + t\mathbf{v}) - f(\mathbf{x})}{t} = \nabla f(\mathbf{x})^T \mathbf{v}.$$

So:

$$f(\mathbf{x} + \eta \mathbf{v}) - f(\mathbf{x}) \approx \eta \cdot \nabla f(\mathbf{x})^T \mathbf{v}.$$

How should we choose $\mathbf{v}$ so that $f(\mathbf{x} + \eta \mathbf{v}) < f(\mathbf{x})$?

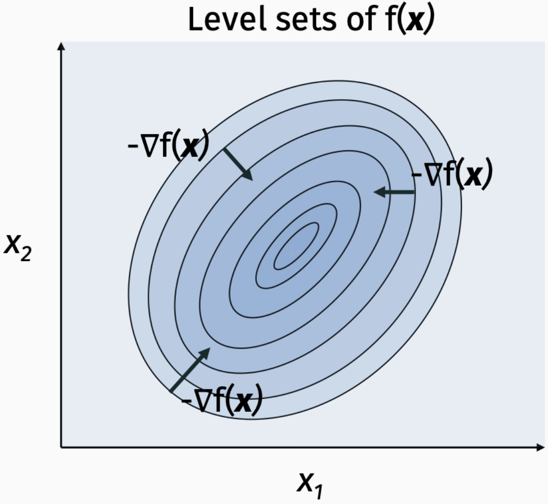
Prototype algorithm:

- Choose starting point $\mathbf{x}^{(0)}$.
- For $i = 0, \ldots, T$:
    - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f(\mathbf{x}^{(i)})$
- Return $\mathbf{x}^{(T)}$.

η is a step-size parameter, which is often adapted on the go. For now, assume it is fixed ahead of time.

1 dimensional example:

2 dimensional example:



For a convex function $f(\mathbf{x})$: For sufficiently small η and a sufficiently large number of iterations T, gradient descent will converge to a **near global minimum**:

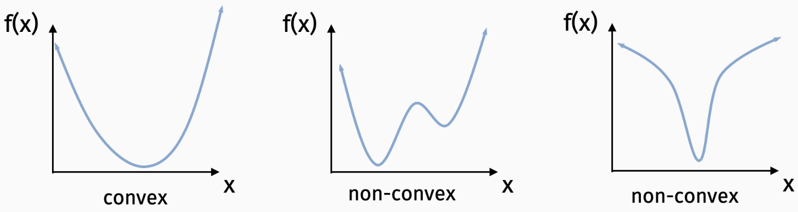
$$f(\mathbf{x}^{(T)}) \leq f(\mathbf{x}^*) + \epsilon.$$

Examples: least squares regression, logistic regression, kernel regression, SVMs.

For a non-convex function $f(\mathbf{x})$: For sufficiently small η and a sufficiently large number of iterations T, gradient descent will converge to a **near stationary point**:

$$\|\nabla f(\mathbf{x}^{(T)})\|_2 \leq \epsilon.$$

Examples: neural networks, matrix completion problems, mixture models.



One issue with non-convex functions is that they can have local minima. Even when they don't, convergence analysis requires different assumptions than convex functions.

We care about how fast gradient descent and related methods converge, not just that they do converge.

- Bounding iteration complexity requires placing some assumptions on $f(\mathbf{x})$.
- Stronger assumptions lead to better bounds on the convergence.

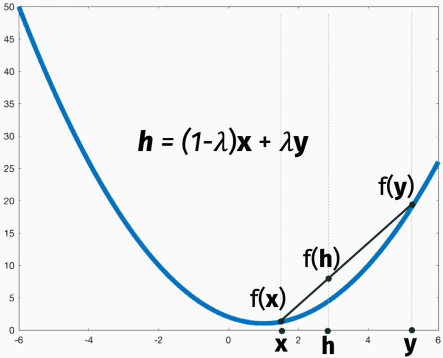
Understanding these assumptions can help us design faster variants of gradient descent (there are many!).

Today, we will start with **convex** functions.

Definition (Convex)

A function f is convex iff for any $\mathbf{x}, \mathbf{y}, \lambda \in [0, 1]$:

$$(1-\lambda) \cdot f(\mathbf{x}) + \lambda \cdot f(\mathbf{y}) \geq f\left((1-\lambda) \cdot \mathbf{x} + \lambda \cdot \mathbf{y}\right)$$



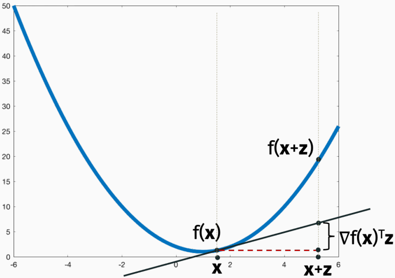
Definition (Convex)

A function f is convex if and only if for any $\mathbf{x}, \mathbf{y}$:

$$f(\mathbf{x}+\mathbf{z}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{z}$$

Equivalently:

$$f(\mathbf{x}) - f(\mathbf{y}) \leq \nabla f(\mathbf{x})^T (\mathbf{x}-\mathbf{y})$$



It is easy but not obvious how to prove the equivalence between these definitions. A short proof can be found in Karthik Sridharan's lecture notes here:

http://www.cs.cornell.edu/courses/cs6783/2018fa/lec16-supplement.pdf

Assume:

- f is convex.
- Lipschitz function: for all $\mathbf{x}$, $\|\nabla f(\mathbf{x})\|_2 \leq G$.
- Starting radius: $\|\mathbf{x}^* - \mathbf{x}^{(0)}\|_2 \leq R$.

Gradient descent:

- Choose number of steps T.
- Starting point $\mathbf{x}^{(0)}$. E.g. $\mathbf{x}^{(0)} = \vec{0}$.
- $\eta = \frac{R}{G\sqrt{T}}$
- For $i = 0, \ldots, T$:
  - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f(\mathbf{x}^{(i)})$
- Return $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}^{(i)}} f(\mathbf{x}^{(i)})$.

Claim (GD Convergence Bound)

If we run GD for $T \geq \frac{R^2 G^2}{\epsilon^2}$ *iterations then* $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$.

Proof is made tricky by the fact that $f(\mathbf{x}^{(i)})$ does not improve monotonically. We can "overshoot" the minimum.

Claim (GD Convergence Bound)

If we run GD for $T \geq \frac{R^2 G^2}{\epsilon^2}$ iterations with step-size $\eta = \frac{R}{G\sqrt{T}}$, then $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$.

Proof is made tricky by the fact that $f(\mathbf{x}^{(i)})$ does not improve monotonically. We can "overshoot" the minimum.

We will prove that the average solution value is low after $T = \frac{R^2 G^2}{\epsilon^2}$ iterations. I.e. that:

$$\frac{1}{T}\sum_{i=0}^{T-1}\left[f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*)\right] \leq \epsilon$$

Of course the best solution found, $\hat{\mathbf{x}}$ is only better than the average.

Claim (GD Convergence Bound)

If we run GD for $T \geq \frac{R^2G^2}{\epsilon^2}$ iterations with step-size $\eta = \frac{R}{G\sqrt{T}}$, then $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$.

Claim 1: For all $i = 0, \ldots, T$,

$$f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*) \leq \frac{\|\mathbf{x}^{(i)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(i+1)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

Claim 1(a): For all $i = 0, \ldots, T$,

$$\nabla f(\mathbf{x}^{(i)})^T(\mathbf{x}^{(i)} - \mathbf{x}^*) \leq \frac{\|\mathbf{x}^{(i)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(i+1)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

Claim 1 follows from Claim 1(a) by definition of convexity.

Claim (GD Convergence Bound)

If we run GD for $T \geq \frac{R^2G^2}{\epsilon^2}$ iterations with step size $\eta = \frac{R}{G\sqrt{T}}$, then $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$.

Claim 1(a): For all $i = 0, \ldots, T$,

$$\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(i+1)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2} \geq \nabla f(\mathbf{x}^{(i)})^T(\mathbf{x}^{(i)} - \mathbf{x}^*)$$

Claim (GD Convergence Bound)

If $T \geq \frac{R^2G^2}{\epsilon^2}$ and $\eta = \frac{R}{G\sqrt{T}}$, then $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$.

Claim 1: For all $i = 0, \ldots, T$,

$$f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*) \leq \frac{\|\mathbf{x}^{(i)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(i+1)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

Telescoping sum:

$$\begin{aligned}
\sum_{i=0}^{T-1} \left[f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*)\right] \leq\ & \frac{\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2} \\
&+ \frac{\|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(2)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2} \\
&+ \frac{\|\mathbf{x}^{(2)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(3)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2} \\
&\vdots \\
&+ \frac{\|\mathbf{x}^{(T-1)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(T)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}
\end{aligned}$$

Claim (GD Convergence Bound)

If $T \geq \frac{R^2G^2}{\epsilon^2}$ and $\eta = \frac{R}{G\sqrt{T}}$, then $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$.

Telescoping sum:

$$\sum_{i=0}^{T-1}\left[f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*)\right] \leq \frac{\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(T)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{T\eta G^2}{2}$$

$$\frac{1}{T}\sum_{i=0}^{T-1}\left[f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*)\right] \leq \frac{R^2}{2T\eta} + \frac{\eta G^2}{2}$$

Claim (GD Convergence Bound)

If $T \geq \frac{R^2G^2}{\epsilon^2}$ and $\eta = \frac{R}{G\sqrt{T}}$, then $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$.

Final step:

$$\frac{1}{T}\sum_{i=0}^{T-1}\left[f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*)\right] \leq \epsilon$$

$$\left[\frac{1}{T}\sum_{i=0}^{T-1} f(\mathbf{x}^{(i)})\right] - f(\mathbf{x}^*) \leq \epsilon$$

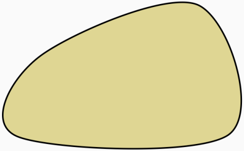
We always have that $f(\hat{\mathbf{x}}) = \min_i f(\mathbf{x}^{(i)}) \leq \frac{1}{T}\sum_{i=0}^{T-1} f(\mathbf{x}^{(i)})$, which gives the final bound:

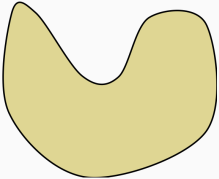
$$f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon.$$

Typical goal: Solve a convex minimization problem with additional convex constraints.

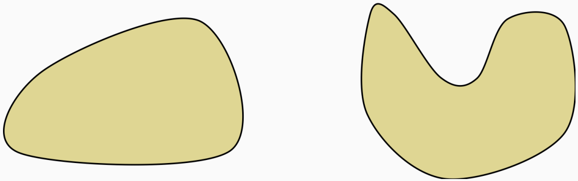
$$\min_{\mathbf{x} \in \mathcal{S}} f(\mathbf{x})$$

where $\mathcal{S}$ is a **convex set**.





Which of these is convex?



Definition (Convex set)

A set $\mathcal{S}$ is convex if for any $\mathbf{x}, \mathbf{y} \in \mathcal{S}, \lambda \in [0, 1]$:

$$(1 - \lambda)\mathbf{x} + \lambda\mathbf{y} \in \mathcal{S}.$$

Examples:

- **Norm constraint:** minimize $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ subject to $\|\mathbf{x}\|_2 \leq \lambda$. Used e.g. for regularization, finding a sparse solution, etc.
- **Positivity constraint:** minimize $f(\mathbf{x})$ subject to $\mathbf{x} \geq 0$.
- **Linear constraint:** minimize $\mathbf{c}^T\mathbf{x}$ subject to $\mathbf{A}\mathbf{x} \leq \mathbf{b}$. Linear program used in training support vector machines, industrial optimization, subroutine in integer programming, etc.

Gradient descent:

- For $i = 0, \ldots, T$:
  - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f(\mathbf{x}^{(i)})$
- Return $\hat{\mathbf{x}} = \arg\min_i f(\mathbf{x}^{(i)})$.

Even if we start with $\mathbf{x}^{(0)} \in \mathcal{S}$, there is no guarantee that $\mathbf{x}^{(0)} - \eta \nabla f(\mathbf{x}^{(0)})$ will remain in our set.

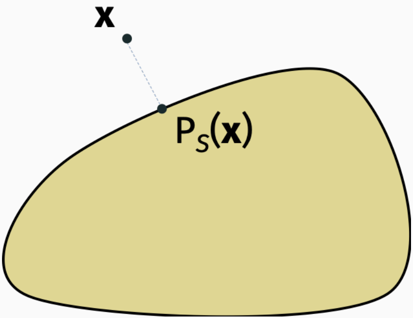
Extremely simple modification: Force $\mathbf{x}^{(i)}$ to be in $\mathcal{S}$ by **projecting** onto the set.

Given a function f to minimize and a convex constraint set $\mathcal{S}$, assume we have:

- **Function oracle**: Evaluate $f(\mathbf{x})$ for any $\mathbf{x}$.
- **Gradient oracle**: Evaluate $\nabla f(\mathbf{x})$ for any $\mathbf{x}$.
- **Projection oracle**: Evaluate $P_{\mathcal{S}}(\mathbf{x})$ for any $\mathbf{x}$.

$$P_{\mathcal{S}}(\mathbf{x}) = \arg\min_{\mathbf{y}\in\mathcal{S}} \|\mathbf{x} - \mathbf{y}\|_2$$

- How would you implement $P_{\mathcal{S}}$ for $\mathcal{S} = \{\mathbf{y} : \|\mathbf{y}\|_2 \leq 1\}$.
- How would you implement $P_{\mathcal{S}}$ for $\mathcal{S} = \{\mathbf{y} : \mathbf{y} = \mathbf{Qz}\}$.



Given function $f(\mathbf{x})$ and set $\mathcal{S}$, such that $\|\nabla f(\mathbf{x})\|_2 \leq G$ for all $\mathbf{x} \in \mathcal{S}$ and starting point $\mathbf{x}^{(0)}$ with $\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2 \leq R$.

Projected gradient descent:

- Select starting point $\mathbf{x}^{(0)}$, $\eta = \frac{R}{G\sqrt{T}}$.
- For $i = 0, \ldots, T$:
  - $\mathbf{z} = \mathbf{x}^{(i)} - \eta \nabla f(\mathbf{x}^{(i)})$
  - $\mathbf{x}^{(i+1)} = P_{\mathcal{S}}(\mathbf{z})$
- Return $\hat{\mathbf{x}} = \arg\min_i f(\mathbf{x}^{(i)})$.

Claim (PGD Convergence Bound)

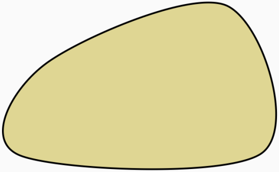
If $f, \mathcal{S}$ are convex and $T \geq \frac{R^2 G^2}{\epsilon^2}$, then $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$.

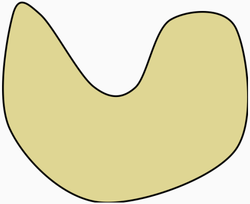
Analysis is almost identical to standard gradient descent! We just need one additional claim:

Claim (Contraction Property of Convex Projection)

If $\mathcal{S}$ is convex, then for <u>any</u> $\mathbf{y} \in \mathcal{S}$,

$$\|\mathbf{y} - P_{\mathcal{S}}(\mathbf{x})\|_2 \leq \|\mathbf{y} - \mathbf{x}\|_2.$$





Claim (PGD Convergence Bound)

If $f, \mathcal{S}$ are convex and $T \geq \frac{R^2G^2}{\epsilon^2}$, then $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$.

Claim 1: For all $i = 0, \ldots, T$, let $\mathbf{z}^{(i)} = \mathbf{x}^{(i)} - \eta \nabla f(\mathbf{x}^{(i)})$. Then:

$$f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*) \leq \frac{\|\mathbf{x}^{(i)} - \mathbf{x}^*\|_2^2 - \|\mathbf{z}^{(i)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$
$$\leq \frac{\|\mathbf{x}^{(i)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(i+1)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

Same telescoping sum argument:

$$\left[\frac{1}{T}\sum_{i=0}^{T-1} f(\mathbf{x}^{(i)})\right] - f(\mathbf{x}^*) \leq \frac{R^2}{2T\eta} + \frac{\eta G^2}{2}.$$

Conditions:

- **Convexity:** f is a convex function, $\mathcal{S}$ is a convex set.
- **Bounded initial distant:**

$$\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2 \leq R$$

- **Bounded gradients (Lipschitz function):**

$$\|\nabla f(\mathbf{x})\|_2 \leq G \text{ for all } \mathbf{x} \in \mathcal{S}.$$

Theorem (GD Convergence Bound)

(Projected) Gradient Descent returns $\hat{\mathbf{x}}$ *with* $f(\hat{\mathbf{x}}) \leq \min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x}) + \epsilon$ *after*

$$T = \frac{R^2 G^2}{\epsilon^2} \text{ iterations.}$$

The previous bounds are optimal for convex first order optimization in general.

But in practice, the dependence on $1/\epsilon^2$ is pessimistic: gradient descent typically requires far fewer steps to reach ϵ error.

Previous bounds only make a very weak first order assumption:

$$\|\nabla f(x)\|_2 \leq G.$$

In practice, many function satisfy stronger assumptions.

Often possible to place assumptions on the second derivative of f.

In particular, we say that a scalar function f is α-strongly convex and β-smooth if for all x:

$$\alpha \leq f''(x) \leq \beta.$$

We will give an appropriate generalization of these conditions to multi-dimensional functions shortly.

Take away: Having either an upper and lower bound on the second derivative helps convergence. Having both helps a lot.

Take away: Having either an upper and lower bound on the second derivative helps convergence. Having both helps a lot.

Number of iterations for ϵ error:

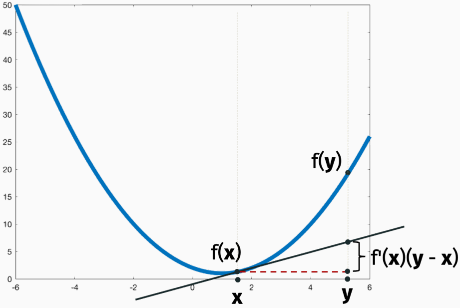
	G-Lipschitz	β-smooth
R bounded start	$O\left(\frac{G^2R^2}{\epsilon^2}\right)$	$O\left(\frac{\beta R^2}{\epsilon}\right)$
α-strong convex	$O\left(\frac{G^2}{\alpha\epsilon}\right)$	$O\left(\frac{\beta}{\alpha}\log(1/\epsilon)\right)$

As we defined them so far, smoothness and strong convexity require f to be twice differentiable. On the other hand, gradient descent only requires first order differentiability.

Equivalent conditions:

$$f''(x) \le \beta \iff [f(y) - f(x)] - f'(x)(y - x) \le \frac{\beta}{2}(y - x)^2$$

$$f''(x) \ge \alpha \iff [f(y) - f(x)] - f'(x)(y - x) \ge \frac{\alpha}{2}(y - x)^2$$



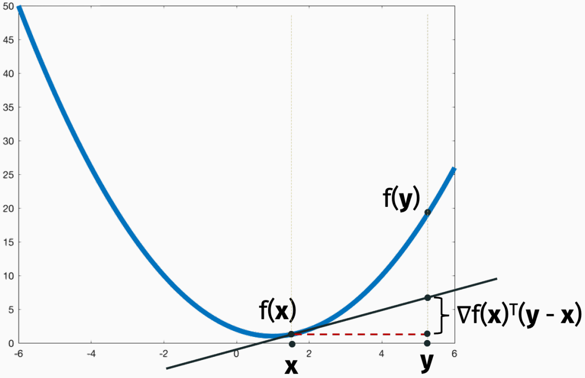
Recall: For all convex functions $[f(y) - f(x)] - f'(x)(y - x) \ge 0$.

Proof that $f''(x) \leq \beta \Rightarrow [f(y) - f(x)] - f'(x)(y - x) \leq \frac{\beta}{2}(y - x)^2$:

Proof for α-strongly convex is similar, as are the other directions.

A function is α-strongly convex and β-smooth if for all $\mathbf{x}$, $\mathbf{y}$:

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



Definition (β-smoothness)

A function f is β smooth if and only if, for all $\mathbf{x}, \mathbf{y}$

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

I.e., the gradient function is a β-Lipschitz function.

We won't use this definition directly, but it's good to know. Easy to prove equivalency to previous definition (see Lem. 3.4 in Bubeck's book).

Theorem (GD convergence for β-smooth functions.)

Let f be a β smooth convex function and assume we have $\|\mathbf{x}^* - \mathbf{x}^{(0)}\|_2 \leq R$. If we run GD for T steps, we have:

$$f(\mathbf{x}^{(T)}) - f(\mathbf{x}^*) \leq \frac{2\beta R^2}{T}$$

Corollary: If $T = O\left(\frac{\beta R^2}{\epsilon}\right)$ we have $f(\mathbf{x}^{(T)}) - f(\mathbf{x}^*) \leq \epsilon$.

Compare this to $T = O\left(\frac{G^2 R^2}{\epsilon^2}\right)$ without a smoothness assumption.

Why do you think gradient descent might be faster when a function is β-smooth?

Previously learning rate/step size η depended on G. Now choose it based on β:

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \frac{1}{\beta}\nabla f(\mathbf{x}^{(t)})$$

Progress per step of gradient descent:

1. $\left[f(\mathbf{x}^{(t+1)}) - f(\mathbf{x}^{(t)})\right] - \nabla f(\mathbf{x}^{(t)})^T(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}) \leq \frac{\beta}{2}\|\mathbf{x}^{(t)} - \mathbf{x}^{(t+1)}\|_2^2$.
2. $\left[f(\mathbf{x}^{(t+1)}) - f(\mathbf{x}^{(t)})\right] + \frac{1}{\beta}\|\nabla f(\mathbf{x}^{(t)})\|_2^2 \leq \frac{\beta}{2}\|\frac{1}{\beta}\nabla f(\mathbf{x}^{(t)})\|_2^2$.
3. $f(\mathbf{x}^{(t)}) - f(\mathbf{x}^{(t+1)}) \geq \frac{1}{2\beta}\|\nabla f(\mathbf{x}^{(t)})\|_2^2$.

Once we have the bound from the previous page, proving a convergence result isn't hard, but not obvious. A concise proof can be found in Page 15 in Garrigos and Gower's notes.

Theorem (GD convergence for β-smooth functions.)

Let f be a β smooth convex function and assume we have $\|\mathbf{x}^* - \mathbf{x}^{(1)}\|_2 \leq R$. If we run GD for T steps with $\eta = \frac{1}{\beta}$ we have:

$$f(\mathbf{x}^{(T)}) - f(\mathbf{x}^*) \leq \frac{2\beta R^2}{T}$$

Corollary: If $T = O\left(\frac{\beta R^2}{\epsilon}\right)$ we have $f(\mathbf{x}^{(T)}) - f(\mathbf{x}^*) \leq \epsilon$.

Where did we use convexity in this proof?

Progress per step of gradient descent:

1. $\left[f(\mathbf{x}^{(t+1)}) - f(\mathbf{x}^{(t)})\right] - \nabla f(\mathbf{x}^{(t)})^T(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}) \leq \frac{\beta}{2}\|\mathbf{x}^{(t)} - \mathbf{x}^{(t+1)}\|_2^2$.

2. $\left[f(\mathbf{x}^{(t+1)}) - f(\mathbf{x}^{(t)})\right] + \frac{1}{\beta}\|\nabla f(\mathbf{x}^{(t)})\|_2^2 \leq \frac{\beta}{2}\|\frac{1}{\beta}\nabla f(\mathbf{x}^{(t)})\|_2^2$.

3. $f(\mathbf{x}^{(t)}) - f(\mathbf{x}^{(t+1)}) \geq \frac{1}{2\beta}\|\nabla f(\mathbf{x}^{(t)})\|_2^2$.

Definition (Stationary point)

For a differentiable function f, a stationary point is any $\mathbf{x}$ with:

$$\nabla f(\mathbf{x}) = \mathbf{0}$$

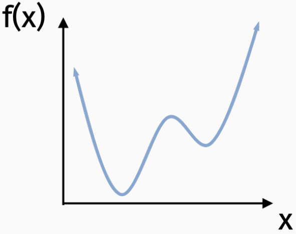
local/global minima - local/global maxima - saddle points

Theorem (Convergence to Stationary Point)

For any β-smooth differentiable function f (convex or not), if we run GD for T steps, we can find a point $\hat{\mathbf{x}}$ such that:

$$\|\nabla f(\hat{\mathbf{x}})\|_2^2 \leq \frac{2\beta}{T}\left(f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)\right)$$

Corollary: If $T \geq \frac{2\beta}{\epsilon}$, then $\|\nabla f(\hat{\mathbf{x}})\|_2^2 \leq \epsilon\left(f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)\right)$.



Theorem (Convergence to Stationary Point)

For any β-smooth differentiable function f (convex or not), if we run GD for T steps, we can find a point $\hat{\mathbf{x}}$ such that:

$$\|\nabla f(\hat{\mathbf{x}})\|_2^2 \leq \frac{2\beta}{T}\left(f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)\right)$$

We have that $\frac{1}{2\beta}\|\nabla f(\mathbf{x}^{(t)})\|_2^2 \leq f(\mathbf{x}^{(t)}) - f(\mathbf{x}^{(t+1)})$. So:

$$\sum_{t=0}^{T-1} \frac{1}{2\beta}\|\nabla f(\mathbf{x}^{(t)})\|_2^2 \leq f(\mathbf{x}^{(0)}) - f(\mathbf{x}^{(t)})$$

$$\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla f(\mathbf{x}^{(t)})\|_2^2 \leq \frac{2\beta}{T}\left(f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)\right)$$

$$\min_t \|\nabla f(\mathbf{x}^{(t)})\|_2^2 \leq \frac{2\beta}{T}\left(f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)\right)$$

I said it was a bit tricky to prove that $f(\hat{\mathbf{x}}) - f(\mathbf{x}^*) \leq \frac{2\beta R^2}{T}$ for convex functions. But we just easily proved that $\|\nabla f(\hat{\mathbf{x}})\|_2^2$ is small. Why doesn't this show we are close to the minimum?

Definition (α-strongly convex)

A convex function f is α-strongly convex if, for all $\mathbf{x}, \mathbf{y}$

$$[f(\mathbf{y}) - f(\mathbf{x})] - \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) \geq \frac{\alpha}{2}\|\mathbf{x} - \mathbf{y}\|_2^2$$

Compare to smoothness condition.

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

For a twice-differentiable scalar function f, equivalent to $f''(x) \geq \alpha$.

When f is convex, we always have that $f''(x) \geq 0$, so larger values of α correspond to a "stronger" condition.

Gradient descent for strongly convex functions:

- Choose number of steps T.
- For $i = 0, \ldots, T$:
  - $\eta = \frac{2}{\alpha \cdot (i+1)}$
  - $x^{(i+1)} = x^{(i)} - \eta \nabla f(x^{(i)})$
- Return $\hat{x} = \arg\min_{x^{(i)}} f(x^{(i)})$.

Theorem (GD convergence for α-strongly convex functions.)

Let f be an α-strongly convex function and assume we have that, for all $\mathbf{x}$, $\|\nabla f(\mathbf{x})\|_2 \leq G$. If we run GD for T steps (with adaptive step sizes) we have:

$$f(\hat{\mathbf{x}}) - f(\mathbf{x}^*) \leq \frac{2G^2}{\alpha T}$$

Corollary: If $T = O\left(\frac{G^2}{\alpha\epsilon}\right)$ we have $f(\hat{\mathbf{x}}) - f(\mathbf{x}^*) \leq \epsilon$

We could also have that f is both β-smooth and α-strongly convex.

Theorem (GD for β-smooth, α-strongly convex.)

Let f be a β-smooth and α-strongly convex function. If we run GD for T steps (with step size $\eta = \frac{1}{\beta}$) we have:

$$\|\mathbf{x}^{(T)} - \mathbf{x}^*\|_2^2 \leq e^{-T\frac{\alpha}{\beta}} \|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2^2$$

$\kappa = \frac{\beta}{\alpha}$ is called the "condition number" of f.

Is it better if κ is large or small?

Converting to more familiar form: Using that fact the $\nabla f(\mathbf{x}^*) = \mathbf{0}$ along with

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

we have:

$$\|\mathbf{x}^{(T)} - \mathbf{x}^*\|_2^2 \geq \frac{2}{\beta}\left[f(\mathbf{x}^{(T)}) - f(\mathbf{x}^*)\right].$$

We also assume

$$\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2^2 \leq R^2.$$

Corollary (GD for β-smooth, α-strongly convex.)

Let f be a β-smooth and α-strongly convex function. If we run GD for T steps (with step size $\eta = \frac{1}{\beta}$) we have:

$$f(\mathbf{x}^{(T)}) - f(\mathbf{x}^*) \leq \frac{\beta}{2} e^{-T\frac{\alpha}{\beta}} \cdot R^2$$

Corollary: If $T = O\left(\frac{\beta}{\alpha}\log(R\beta/\epsilon)\right)$ we have:

$$f(\mathbf{x}^{(T)}) - f(\mathbf{x}^*) \leq \epsilon$$

Only depend on $\log(1/\epsilon)$ instead of on $1/\epsilon$ or $1/\epsilon^2$!

We are going to prove the guarantee on the previous page for the special case of:

$$f(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

Goal: Get some of the key ideas across, introduces important concepts like the Hessian, and show the connection between conditioning and linear algebra.