First-Order Optimization Algorithms for Machine Learning

Convergence of Gradient Descent

Mark Schmidt

University of British Columbia

Summer 2020

Last Time: Progress Bound for Gradient Descent

- We discussed gradient descent,

$$w^{k+1} = w^k - \alpha_k \nabla f(w^k).$$

assuming that the gradient was Lipschitz continuous (weak assumption),

$$\|\nabla f(w) - \nabla f(v)\| \leq L\|w - v\|,$$

- We showed that setting $\alpha_k = 1/L$ gives a progress bound of

$$f(w^{k+1}) \leq f(w^k) - \frac{1}{2L}\|\nabla f(w^k)\|^2,$$

- We discussed practical α_k values that give similar bounds.
  - "Try a big step-size, and decrease it if isn't satisfying a progress bound."

Cost of L2-Regularizd Least Squares

- Two strategies from 340 for L2-regularized least squares:
  1. Closed-form solution,
  $$w = (X^TX + \lambda I)^{-1}(X^Ty),$$
  which costs $O(nd^2 + d^3)$.
     - This is fine for $d = 5000$, but may be too slow for $d = 1,000,000$.
  2. Run t iterations of gradient descent,
  $$w^{k+1} = w^k - \alpha_k \underbrace{(X^T(Xw^k - y) + \lambda w^k)}_{\nabla f(w^k)},$$
  which costs $O(ndt)$.
     - I'm using t as total number of iterations, and k as iteration number.

- Gradient descent is faster if t is not too big:
  - If we only need $t < \max\{d, d^2/n\}$ iterations.

Cost of Logistic Regression

- Gradient descent can also be applied to other models like logistic regression,

$$f(w) = \sum_{i=1}^{n} \log(1 + \exp(-y^i w^T x^i)),$$

which we can't formulate as a linear system.

  - Setting $\nabla f(w) = 0$ gives a system of transcendental equations.

- But this objective function is convex and differentiable.
  - So gradient descent converges to a global optimum.

- Alternately, another common approach is Newton's method.
  - Requires computing Hessian $\nabla^2 f(w^k)$, and known as "IRLS" in statistics.

Cost of Logistic Regression

- Gradient descent costs $O(nd)$ per iteration to for logistic regression.
- Newton costs $O(nd^2 + d^3)$ per iteration to compute and invert $\nabla^2 f(w^k)$.

- Newton typically requires substantially fewer iterations.

- But for datasets with very large d, gradient descent might be faster.
  - If $t < \max\{d, d^2/n\}$ then we should use the "slow" algorithm with fast iterations.

- So, how many iterations t of gradient descent do we need?

Outline

Convergence Rate of Gradient Descent

- In 340, we claimed that $\nabla f(w^k)$ converges to zero as k goes to ∞.
  - For convex functions, this means it converges to a global optimum.
  - However, we may not have $\nabla f(w^k) = 0$ for any finite k.

- Instead, we're usually happy with $\|\nabla f(w^k)\| \leq \epsilon$ for some small ϵ.
  - Given an ϵ, how many iterations does it take for this to happen?

- We'll first answer this question only assuming that
  1. Gradient ∇f is Lipschitz continuous (as before).
  2. Step-size $\alpha_k = 1/L$ (this is only to make things simpler).
  3. Function f can't go below a certain value f^* ("bounded below").

- Most ML objectives f are bounded below (like the squared error being at least 0).
  - We're not assuming convexity (but only showing convergence to a stationary point).

Convergence Rate of Gradient Descent

- Key ideas:
  1. We start at some $f(w^0)$, and at each step we decrease f by at least $\frac{1}{2L}\|\nabla f(w^k)\|^2$.
  2. But we can't decrease $f(w^k)$ below f^*.
  3. So $\|\nabla f(w^k)\|^2$ must be going to zero "fast enough".

- Let's start with our guaranteed progress bound,

$$f(w^k) \leq f(w^{k-1}) - \frac{1}{2L}\|\nabla f(w^{k-1})\|^2.$$

- Since we want to bound $\|\nabla f(w^k)\|$, let's rearrange as

$$\|\nabla f(w^{k-1})\|^2 \leq 2L(f(w^{k-1}) - f(w^k)).$$

Convergence Rate of Gradient Descent

- So for each iteration k, we have

$$\|\nabla f(w^{k-1})\|^2 \leq 2L[f(w^{k-1}) - f(w^k)].$$

- Let's sum up the squared norms of all the gradients up to iteration t,

$$\sum_{k=1}^{t} \|\nabla f(w^{k-1})\|^2 \leq 2L \sum_{k=1}^{t}[f(w^{k-1}) - f(w^k)].$$

- Now we use two tricks:
  1. On the left, use that all $\|\nabla f(w^{k-1})\|$ are at least as big as their minimum.
  2. On the right, use that this is a telescoping sum:

$$\sum_{k=1}^{t}[f(w^{k-1}) - f(w^k)] = f(w^0) - \underbrace{f(w^1) + f(w^1)}_{0} - \underbrace{f(w^2) + f(w^2)}_{0} - \dots f(w^t)$$
$$= f(w^0) - f(w^t).$$

Convergence Rate of Gradient Descent

- With these substitutions we have

$$\sum_{k=1}^{t} \underbrace{\min_{j \in \{0,\dots,t-1\}} \{\|\nabla f(w^j)\|^2\}}_{\text{no dependence on } k} \leq 2L[f(w^0) - f(w^t)].$$

- Now using that $f(w^t) \geq f^*$ we get

$$t \min_{k \in \{0,1,\dots,t-1\}} \left\{\|\nabla f(w^k)\|^2\right\} \leq 2L[f(w^0) - f^*],$$

and finally that

$$\min_{k \in \{0,1,\dots,t-1\}} \left\{\|\nabla f(w^k)\|^2\right\} \leq \frac{2L[f(w^0) - f^*]}{t} = O(1/t),$$

so if we run for t iterations, we'll find $\underbrace{\text{at least one } k}_{\text{the minimum}}$ with $\|\nabla f(w^k)\|^2 = O(1/t)$.

Convergence Rate of Gradient Descent

- Our "error on iteration t" bound:

$$\min_{k \in \{0,1,\dots,t-1\}} \left\{ \|\nabla f(w^k)\|^2 \right\} \leq \frac{2L[f(w^0) - f^*]}{t}.$$

- We want to know when the norm is below ϵ, which is guaranteed if:

$$\frac{2L[f(w^0) - f^*]}{t} \leq \epsilon.$$

- Solving for t gives that this is guaranteed for every t where

$$t \geq \frac{2L[f(w^0) - f^*]}{\epsilon},$$

so gradient descent requires $t = O(1/\epsilon)$ iterations to achieve $\|\nabla f(w^k)\|^2 \leq \epsilon$.

Outline

Discussion of $O(1/t)$ and $O(1/\epsilon)$ Results

- We showed that after t iterations, there will be a k such that

$$\|\nabla f(w^k)\|^2 = O(1/t).$$

- If we want to have a k with $\|\nabla f(w^k)\|^2 \leq \epsilon$, number of iterations we need is

$$t = O(1/\epsilon).$$

- So if computing gradient costs $O(nd)$, total cost of gradient descent is $O(nd/\epsilon)$.
  - $O(nd)$ per iteration and $O(1/\epsilon)$ iterations.

- This also be shown for practical step-size strategies from last time.
  - Just changes constants.

Discussion of $O(1/t)$ and $O(1/\epsilon)$ Results

- Our precise "error on iteration t" result was

$$\min_{k=0,1,\dots,t-1} \{\|\nabla f(w^k)\|^2\} \leq \frac{2L[f(w^0) - f^*]}{t}.$$

- This is a non-asymptotic result:
  - It holds on iteration 1, there is no "limit as $t \to \infty$" as in classic results.
  - But if t goes to ∞, argument can be modified to show that $\nabla f(w^t)$ goes to zero.

- This convergence rate is called "dimension-independent":
  - It does not directly depend on dimension d.
  - Though L might grow as dimension increases.

- Consider least squares with a fixed L and $f(w^0)$, and an accuracy ϵ:
  - There is dimension d beyond which gradient descent is faster than normal equations.

Discussion of $O(1/t)$ and $O(1/\epsilon)$ Results

- We showed that after t iterations, there is always a k such that

$$\min_{k=0,1,\dots,t-1}\{\|\nabla f(w^k)\|^2\} \leq \frac{2L[f(w^0) - f^*]}{t}.$$

- It isn't necessarily the last iteration t that achieves this.
  - But iteration t does have the lowest value of $f(w^k)$.

- For real ML problems optimization bounds like this are often very loose.
  - In practice gradient descent converges much faster.
  - There is a practical and theoretical component to developing optimization methods.

- This does not imply that gradient descent finds global minimum.
  - We could be minimizing an NP-hard function with bad local optima.

Faster Convergence to Global Optimum?

- What about finding the global optimum of a non-convex function?
- Fastest possible algorithms requires $O(1/\epsilon^d)$ iterations for Lipschitz-continuous f.
  - This is actually achieved by by picking w^k values randomly (or by "grid search").
  - You **can't** beat this with simulated annealing, genetic algorithms, Bayesian optim,...
- Without some assumption like Lipschitz f, getting within ϵ of f^* is impossible.
  - Due to real numbers being uncountable.
  - "Math with Bad Drawings" sketch of proof here.
- These issues are discussed in post-lecture bonus slides.

Convergence Rate for Convex Functions

- For convex functions we can get to a global optimum much faster.
- This is because $\nabla f(w) = 0$ implies w is a global optimum.
  - So gradient descent will converge to a global optimum.
- Using a similar proof (with telescoping sum), for convex f you can show

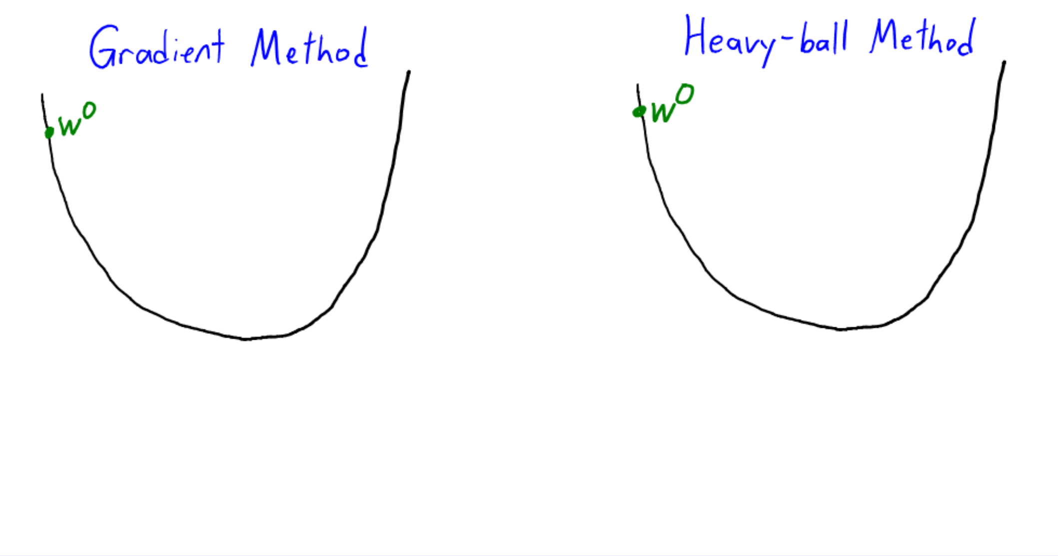
$$f(w^t) - f(w^*) = O(1/t),$$

if there exists a global optimum w^* and ∇f is Lipschitz.
  - So we need $O(1/\epsilon)$ iterations to get ϵ-close to global optimum, not $O(1/\epsilon^d)$.

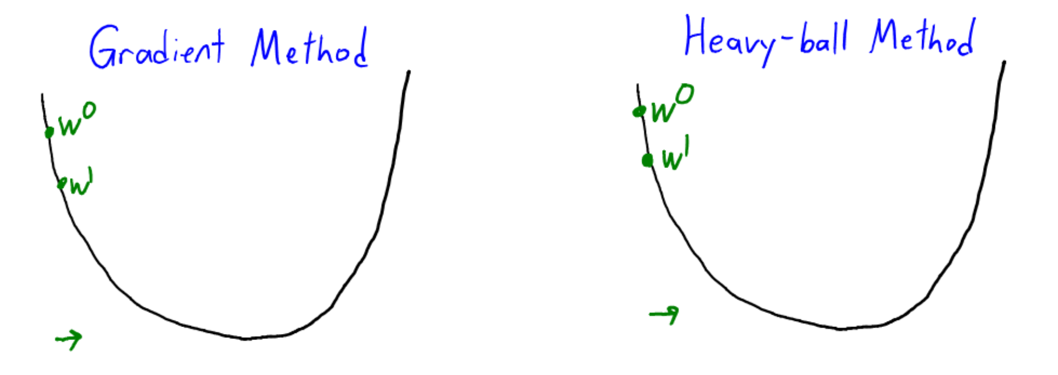
Faster Convergence to Global Optimum?

- Is $O(1/\epsilon)$ the best we can do for convex functions?
- No, there are algorithms that only need $O(1/\sqrt{\epsilon})$.
  - This is optimal for any algorithm based only on functions and gradients.
    - And restricting to dimension-independent rates.
- First algorithm to achieve this: Nesterov's accelerated gradient method.
  - A variation on what's known as the "heavy ball' method (or "momentum").

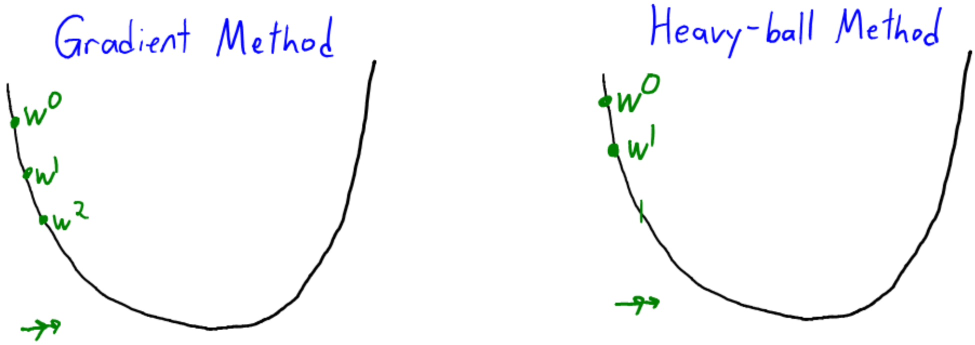
Heavy-Ball Method Method



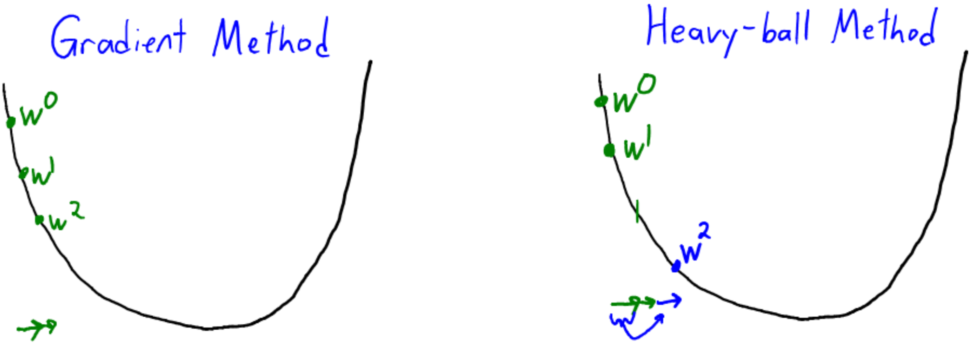
Heavy-Ball Method Method



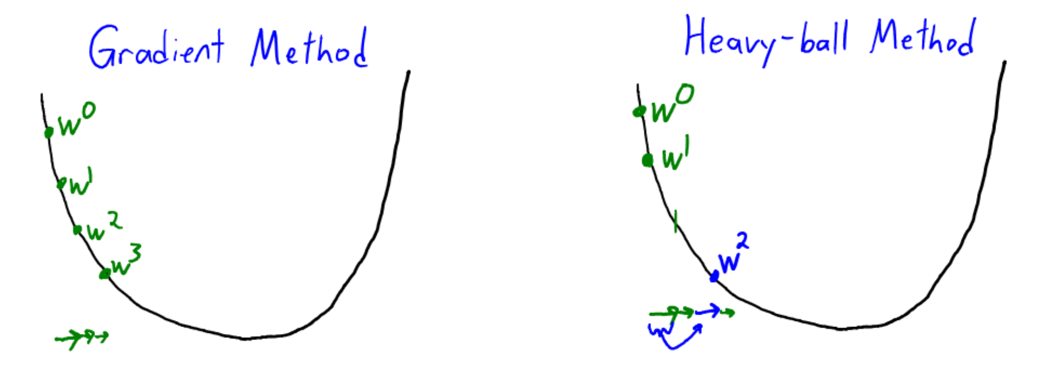
Heavy-Ball Method Method



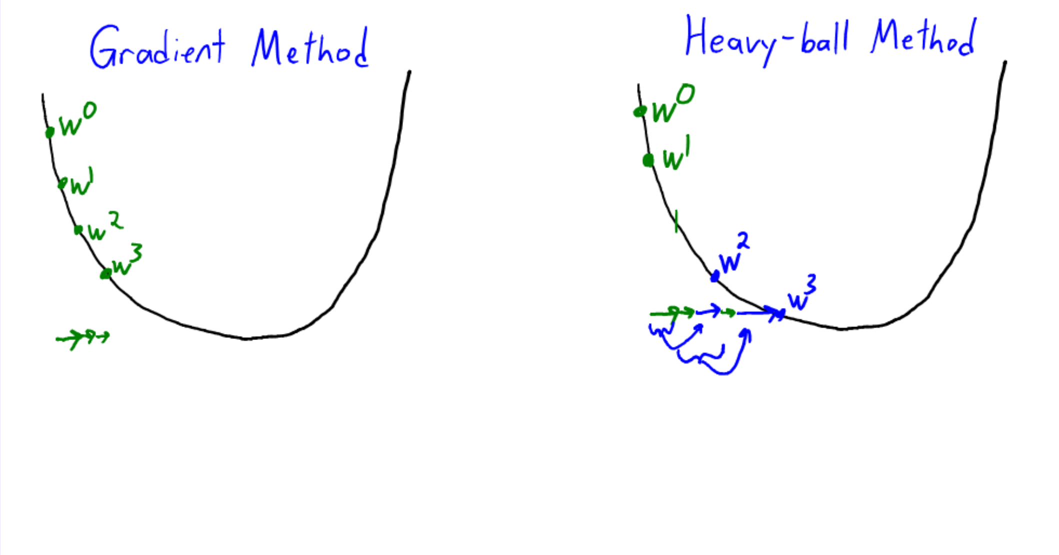
Heavy-Ball Method Method



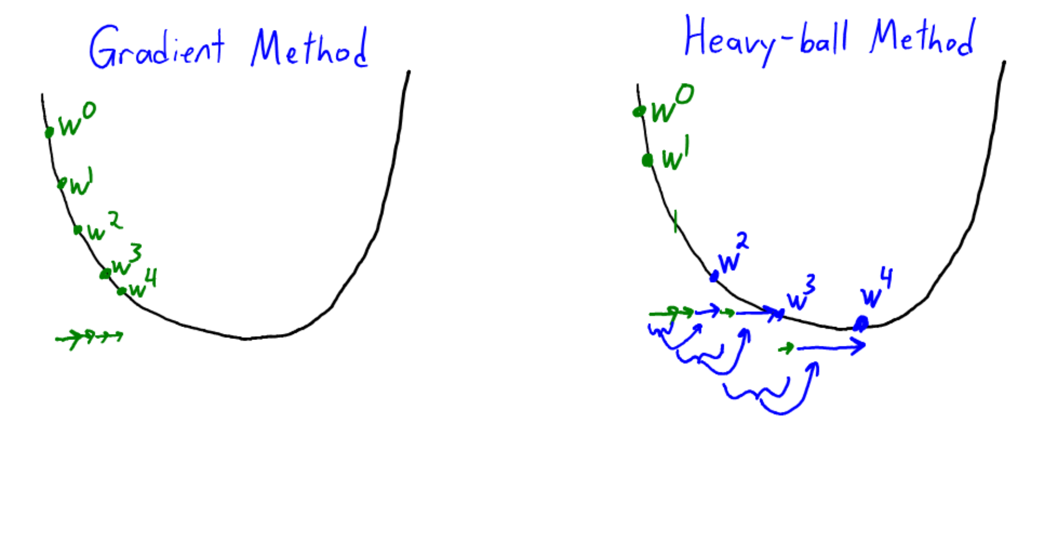
Heavy-Ball Method Method



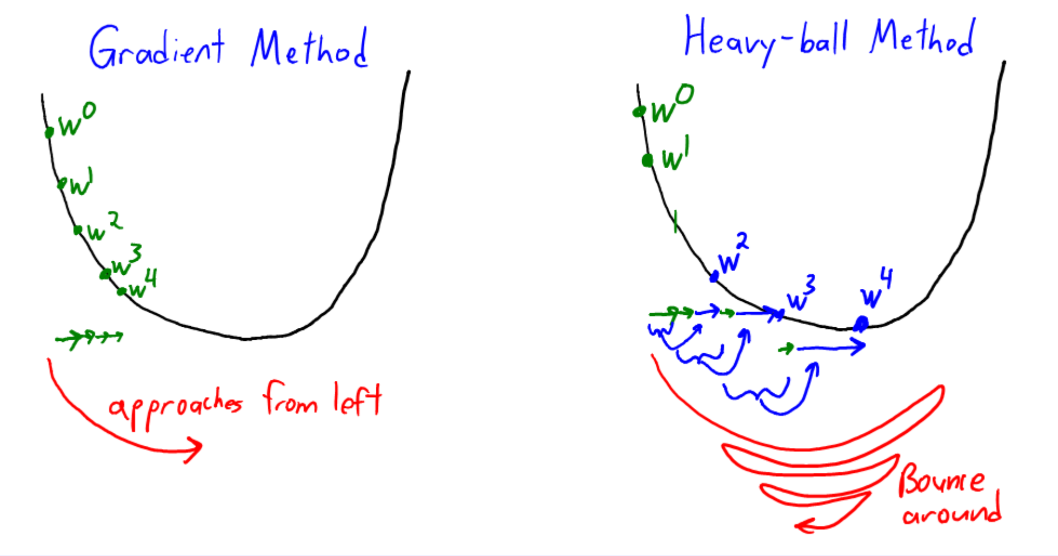
Heavy-Ball Method Method



Heavy-Ball Method Method



Heavy-Ball Method Method



Heavy-Ball, Momentum, CG, and Accelerated Gradient

- The heavy-ball method (called momentum in neural network papers) is

$$w^{k+1} = w^k - \alpha_k \nabla f(w^k) + \beta_k (w^k - w^{k-1}).$$

- For strictly-convex quadratics, achieves faster rate (for appropriate α_k and β_k).
  - With the optimal α_k and β_k, we obtain conjugate gradient.

- Variation is Nesterov's accelerated gradient method,

$$\begin{aligned} w^{k+1} &= v^k - \alpha_k \nabla f(v^k), \\ v^{k+1} &= w^k + \beta_k (w^{k+1} - w^k), \end{aligned}$$

- Has an error of $O(1/t^2)$ after t iterations instead of $O(1/t)$ for convex functions.
  - So it only needs $O(1/\sqrt{\epsilon})$ iterations to get within ϵ of global opt.
  - Can use $\alpha_k = 1/L$ and $\beta_k = \frac{k-1}{k+2}$ to achieve this.

Iteration Complexity

- Iteration complexity: smallest t such that algorithm guarantees ϵ-solution.

- Iteration complexities we have discussed so far:

Assumption	Quantity	Algorithm	Iteration Complexity
Lips. f, bounded domain	$f(w) - f^*$	Random	$O(1/\epsilon^d)$
Lips. ∇f, bounded below	$\|\nabla f(w)\|^2$	Gradient	$O(1/\epsilon)$
Lips. ∇f, convex f	$f(w) - f^*$	Gradient	$O(1/\epsilon)$
Lips. ∇f, convex f	$f(w) - f^*$	Nesterov	$O(1/\sqrt{\epsilon})$

- A lot of optimization research takes these types of forms:
  - Can we get a faster iteration complexity with more assumptions?
  - Can we get the same iteration complexity with fewer assumptions?
  - Can we get the same iteration complexity with cheaper iterations?

Iteration Complexity

- Think of $\log(1/\epsilon)$ as "number of digits of accuracy" you want.
    - We want iteration complexity to grow slowly with $1/\epsilon$.
- Is $O(1/\epsilon)$ a good iteration complexity?
- Not really, if you need 10 iterations for a "digit ''of accuracy then:
    - You might need 100 for 2 digits.
    - You might need 1000 for 3 digits.
    - You might need 10000 for 4 digits.
- We would normally call this exponential time.

Rates of Convergence

- A way to measure rate of convergence is by limit of the ratio of successive errors,

$$\lim_{k\to\infty} \frac{f(w^{k+1}) - f(w^*)}{f(w^k) - f(w^*)} = \rho.$$

- Different ρ values of give us different rates of convergence:
  1. If $\rho = 1$ we call it a sublinear rate.
  2. If $\rho \in (0, 1)$ we call it a linear rate.
  3. If $\rho = 0$ we call it a superlinear rate.

- Having $f(w^t) - f(w^*) = O(1/t)$ gives sublinear convergence rate:
  - "The longer you run the algorithm, the less progress it makes".

Sub/Superlinear Convergence vs. Sub/Superlinear Cost

- As a computer scientist, what would we ideally want?
  - Sublinear rate is bad, we don't want $O(1/t)$ ("exponential" time: $O(1/\epsilon)$ iterations).
  - Linear rate is ok, we're ok with $O(\rho^t)$ ("polynomial" time: $O(\log(1/\epsilon))$ iterations).
  - Superlinear rate is great, amazing to have $O(\rho^{2^t})$ ("constant": $O(\log(\log(1/\epsilon)))$).
- Notice that terminology is backwards compared to computational cost:
  - Superlinear cost is bad, we don't want $O(d^3)$.
  - Linear cost is ok, having $O(d)$ is ok.
  - Sublinear cost is great, having $O(\log(d))$ is great.
- Ideal algorithm: superlinear convergence and sublinear iteration cost.

Summary

- Error on iteration t of $O(1/t)$ for functions that are bounded below.
  - Implies that we need $t = O(1/\epsilon)$ iterations to have $\|\nabla f(x^k)\|^2 \leq \epsilon$.
- Convergence to global min for non-convex (slow) and convex (faster) functions.
  - Nesterov's accelerated gradient method has better bound than gradient descent.
- Iteration complexity measures number of iterations to reach accuracy ϵ.
- Sublinear/linear/superlinear convergence measure speed of convergence.

- Post-lecture slides: Cover various related issues.
  - L for logistic regression, non-convex iteration complexity, smoothing non-smooth?

- Next time: didn't I say that regularization makes gradient descent go faster?

Digression: Logistic Regression Gradient and Hessian

- With some tedious manipulations, gradient for logistic regression is

$$\nabla f(w) = X^T r.$$

where vector r has $r_i = -y^i h(-y^i w^T x^i)$ and h is the sigmoid function.

- We know the gradient has this form from the multivariate chain rule.
  - Functions for the form $f(Xw)$ always have $\nabla f(w) = X^T r$ (see bonus slide).

- With some more tedious manipulations we get

$$\nabla^2 f(w) = X^T D X.$$

where D is a diagonal matrix with $d_{ii} = h(y_i w^T x^i) h(-y^i w^T x^i)$.
  - The $f(Xw)$ structure leads to a $X^T D X$ Hessian structure.
  - For other problems D may not be diagonal.

Convexity of Logistic Regression

- Logistic regression Hessian is

$$\nabla^2 f(w) = X^T D X.$$

where D is a diagonal matrix with $d_{ii} = h(y_i w^T x^i) h(-y^i w^T x^i)$.

- Since the sigmoid function is non-negative, we can compute $D^{\frac{1}{2}}$, and

$$v^T X^T D X v = v^T X^T D^{\frac{1}{2}} D^{\frac{1}{2}} X v = (D^{\frac{1}{2}} X v)^T (D^{\frac{1}{2}} X v) = \|X D^{\frac{1}{2}} v\|^2 \geq 0,$$

so $X^T D X$ is positive semidefinite and logistic regression is convex.

  - It becomes strictly convex if you add L2-regularization, making solution unique.

Lipschitz Continuity of Logistic Regression Gradient

- Logistic regression Hessian is

$$\begin{aligned}\nabla^2 f(w) &= \sum_{i=1}^n \underbrace{h(y_i w^T x^i) h(-y^i w^T x^i)}_{d_{ii}} x^i (x^i)^T \\ &\preceq 0.25 \sum_{i=1}^n x^i (x^i)^T \\ &= 0.25 X^T X.\end{aligned}$$

- In the second line we use that $h(\alpha) \in (0,1)$ and $h(-\alpha) = 1 - \alpha$.
  - This means that $d_{ii} \leq 0.25$.
- So for logistic regression, we can take $L = \frac{1}{4}\max\{\text{eig}(X^T X)\}$.

Multivariate Chain Rule

- If $g : \mathbb{R}^d \mapsto \mathbb{R}^n$ and $f : \mathbb{R}^n \mapsto \mathbb{R}$, then $h(x) = f(g(x))$ has gradient

$$\nabla h(x) = \nabla g(x)^T \nabla f(g(x)),$$

  where $\nabla g(x)$ is the Jacobian (since g is multi-output).
- If g is an affine map $x \mapsto Ax + b$ so that $h(x) = f(Ax + b)$ then we obtain

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

- Further, for the Hessian we have

$$\nabla^2 h(x) = A^T \nabla^2 f(Ax + b) A.$$

First-Order Oracle Model of Computation

- Should we be happy with an algorithm that takes $O(\log(1/\epsilon))$ iterations?
  - Is it possible that algorithms *exist* that solve the problem faster?

- To answer questions like this, need a class of functions.
  - For example, strongly-convex with Lipschitz-continuous gradient.

- We also need a model of computation: what operations are allowed?

- We will typically use a first-order oracle model of computation:
  - On iteration k, algorithm choose an x^k and receives $f(x^k)$ and $\nabla f(x^k)$.
  - To choose x^k, algorithm can do anything that doesn't involve f.

- Common variation is zero-order oracle where algorithm only receives $f(x^k)$.

Complexity of Minimizing Real-Valued Functions

- Consider minimizing real-valued functions over the unit hyper-cube,

$$\min_{x \in [0,1]^d} f(x).$$

- You can use any algorithm you want.

  (simulated annealing, gradient descent + random restarts, genetic algorithms, Bayesian optimization,...)
- How many zero-order oracle calls t before we can guarantee $f(x^t) - f(x^*) \leq \epsilon$?
  - Impossible!

- Given any algorithm, we can construct an f where $f(x^k) - f(x^*) > \epsilon$ forever.
  - Make $f(x) = 0$ except at x^* where $f(x) = -\epsilon - 2^{\text{whatever}}$.

(the x^* is algorithm-specific)

- To say anything in oracle model we need assumptions on f.

Complexity of Minimizing Lipschitz-Continuous Functions

- One of the simplest assumptions is that f is Lipschitz-continuous,

$$|f(x) - f(y)| \leq L\|x - y\|.$$

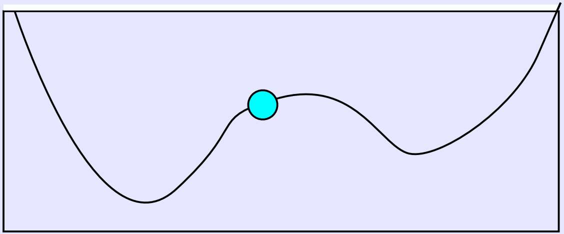
- Function can't change arbitrarily fast as you change x.

Complexity of Minimizing Lipschitz-Continuous Functions

- One of the simplest assumptions is that f is Lipschitz-continuous,

$$|f(x) - f(y)| \leq L\|x - y\|.$$

- Function can't change arbitrarily fast as you change x.

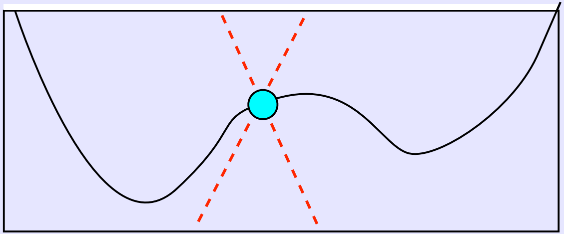


Complexity of Minimizing Lipschitz-Continuous Functions

- One of the simplest assumptions is that f is Lipschitz-continuous,

$$|f(x) - f(y)| \leq L\|x - y\|.$$

- Function can't change arbitrarily fast as you change x.

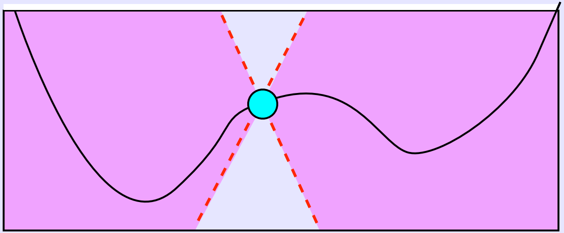


Complexity of Minimizing Lipschitz-Continuous Functions

- One of the simplest assumptions is that f is Lipschitz-continuous,

$$|f(x) - f(y)| \leq L\|x - y\|.$$

- Function can't change arbitrarily fast as you change x.

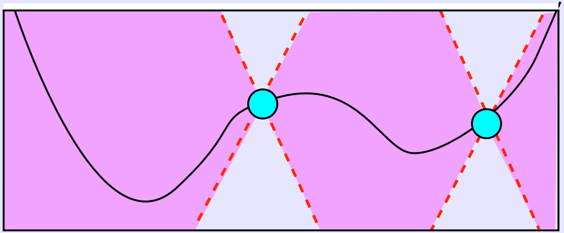


Complexity of Minimizing Lipschitz-Continuous Functions

- One of the simplest assumptions is that f is Lipschitz-continuous,

$$|f(x) - f(y)| \leq L\|x - y\|.$$

- Function can't change arbitrarily fast as you change x.



Complexity of Minimizing Lipschitz-Continuous Functions

- One of the simplest assumptions is that f is Lipschitz-continuous,

$$|f(x) - f(y)| \leq L\|x - y\|.$$

- Function can't change arbitrarily fast as you change x.

- Under only this assumption, any algorithm requires at least $\Omega(1/\epsilon^d)$ iterations.
- An optimal $O(1/\epsilon^d)$ worst-case rate is achieved by a grid-based search method.

- You can also achieve optimal rate in expectation by random guesses.
  - Lipschitz-continuity implies there is a ball of ϵ-optimal solutions around x^*.
  - The radius of the ball is $\Omega(\epsilon)$ so its area is $\Omega(\epsilon^d)$.
  - If we succeed with probability $\Omega(\epsilon^d)$, we expect to need $O(1/\epsilon^d)$ trials.

(mean of geometric random variable)

Complexity of Minimizing Convex Functions

- Life gets better if we assume convexity.
  - We'll consider first-order oracles and rates with no dependence on d.
- Subgradient methods (next week) can minimize convex functions in $O(1/\epsilon^2)$.
  - This is optimal in dimension-independent setting.
- If the gradient is Lipschitz continuous, gradient descent requires $O(1/\epsilon)$.
  - With Nesterov's algorithm, this improves to $O(1/\sqrt{\epsilon})$ which is optimal.
  - Here we don't yet have strong-convexity.
- What about the CPSC 340 approach of smoothing non-smooth functions?
  - Gradient descent still requires $O(1/\epsilon^2)$ in terms of solving original problem.
  - Nesterov improves to $O(1/\epsilon)$ in terms of original problem.