CPSC 540: Machine Learning

Rates of Convergence

Mark Schmidt

University of British Columbia

Winter 2017
Admin

- **Auditing/registration forms:**
  - Submit them at end of class, pick them up end of next class.
  - I need your prereq form before I’ll sign registration forms.
  - I wrote comments on the back of some forms.

- **Assignment 1** due tonight at midnight (Vancouver time).
  - 1 late day to hand in Monday, 2 late days for Wednesday.

- Monday I may be late, if so then Julie Nutini will start lecture.
Gradient descent:
- Iterative method for finding stationary point \( \nabla f(w) = 0 \) of differentiable function.
- For convex functions if converges to a global minimum (if one exists).

Start with \( w^0 \), apply
\[
 w^{k+1} = w^k - \alpha_k \nabla f(w^k),
\]
for step-size \( \alpha_k \).

Cost of algorithm scales linearly with number of variables \( d \).
- Costs \( O(ndt) \) for \( t \) iterations for least squares and logistic regression.
- For \( t < d \), faster than \( O(nd^2 + d^3) \) of normal equations or Newton’s method.
We discussed gradient descent,

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

assuming that the gradient was \textit{Lipschitz continuous} (weak assumption),

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

We showed that setting \( \alpha_k = \frac{1}{L} \) gives a \textit{progress bound} of

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

We discussed practical \( \alpha_k \) values that give similar bounds.

"Try a big step-size, and decrease it if isn't satisfying a progress bound."
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

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

- 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 $\infty$, argument can be modified to show that $\nabla f(w^t)$ goes to zero.

- This convergence rate is 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 $\epsilon$:
  - 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
  \[
  \min_{k=1,2,...,t} \{ \| \nabla f(w^k) \|^2 \} \leq \frac{2L[f(w^0) - f^*]}{t}.
  \]

- 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.
  - So there is a practical and theoretical component to research.

- 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 picking $w^t$ 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 $\epsilon$ 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 $f$

  \[ f(w^t) - f(w^*) = O(1/t), \]

  if there exists a global optimum $w^*$ and $\nabla f$ is Lipschitz.
  - So we need $O(1/\epsilon)$ iterations to get $\epsilon$-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

Gradient Method

Heavy-ball Method

$w^0$
Heavy-Ball Method Method

Gradient Method

Heavy-ball Method
Heavy-Ball Method Method
Heavy-Ball Method Method
Rates of Convergence

Linear Convergence of Gradient Descent

Heavy-Ball Method

Gradient Method

Heavy-ball Method
Heavy-Ball Method Method

Gradient Method

Heavy-ball Method

Rates of Convergence

Linear Convergence of Gradient Descent
Heavy-Ball Method Method

Gradient Method

Heavy-ball Method
Rates of Convergence

Heavy-Ball Method Method

Gradient Method

Heavy-ball Method

approaches from left

Bounce around
Heavy-Ball, Momentum, CG, and Accelerated Gradient

- The **heavy-ball** method (called **momentum** in neural network papers) is
  \[ w^{k+1} = w^t - \alpha_k \nabla f(w^k) + \beta_k (w^k - w^{k-1}) \].

- Faster rate for strictly-convex quadratic functions with appropriate \( \alpha_k \) and \( \beta_k \).
  - With the optimal \( \alpha_k \) and \( \beta_k \), we obtain **conjugate gradient**.

- Variation is **Nesterov’s accelerated gradient method**,
  \[ w^{k+1} = v^k - \alpha_k \nabla f(v^k), \]
  \[ v^{k+1} = w^k + \beta_k (w^{k+1} - w^k) \],

- Which has an **error of** \( O(1/t^2) \) after \( t \) iterations instead of \( O(1/t) \).
  - So it only needs \( O(1/\sqrt{\epsilon}) \) iterations to get within \( \epsilon \) of global opt.
  - Can use \( \alpha_k = 1/L \) and \( \beta_k = \frac{k-1}{k+2} \) to achieve this.
Iteration Complexity

- The smallest $t$ such that we’re within $\epsilon$ is called 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.
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 \( \rho \) 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.
Outline

1. Rates of Convergence

2. Linear Convergence of Gradient Descent
Polyak-Łojasiewicz (PL) Inequality

- For least squares, we have linear cost but we only showed sublinear rate.

- For many “nice” functions $f$, gradient descent actually has a linear rate.

- For example, for functions satisfying the Polyak-Łojasiewicz (PL) inequality,

\[
\frac{1}{2} \| \nabla f(w) \|^2 \geq \mu (f(w) - f^*),
\]

for all $w$ and some $\mu > 0$.

- “Gradient grows as a quadratic function as we increase $f$”.

Linear Convergence under the PL Inequality

- Recall our guaranteed progress bound

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

- Under the PL inequality we have \(-\|\nabla f(w^k)\|^2 \leq -2\mu(f(w^k) - f^*)\), so

\[ f(w^{k+1}) \leq f(w^k) - \frac{\mu}{L} (f(w^k) - f^*). \]

- Let’s subtract \(f^*\) from both sides,

\[ f(w^{k+1}) - f^* \leq f(w^k) - f^* - \frac{\mu}{L} (f(w^k) - f^*), \]

and factorizing the right side gives

\[ f(w^{k+1}) - f^* \leq \left(1 - \frac{\mu}{L}\right) (f(w^k) - f^*). \]
Linear Convergence under the PL Inequality

Applying this recursively:

\[ f(w^k) - f^* \leq \left( 1 - \frac{\mu}{L} \right) [f(w^{k-1}) - f(w^*)] \]

\[ \leq \left( 1 - \frac{\mu}{L} \right) \left[ \left( 1 - \frac{\mu}{L} \right) [f(w^{k-2}) - f^*] \right] \]

\[ = \left( 1 - \frac{\mu}{L} \right)^2 [f(w^{k-2}) - f^*] \]

\[ \leq \left( 1 - \frac{\mu}{L} \right)^3 [f(w^{k-3}) - f^*] \]

\[ \leq \left( 1 - \frac{\mu}{L} \right)^k [f(w^0) - f^*] \]

We’ll always have \( \mu \leq L \) so we have \( (1 - \mu/L) < 1 \).

- So PL implies a linear convergence rate: \( f(w^k) - f^* = O(\rho^k) \) for \( \rho < 1 \).
Linear Convergence under the PL Inequality

- We’ve shown that
  \[ f(w^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k [f(w^0) - f^*] \]

- By using the inequality that
  \[ (1 - \gamma) \leq \exp(-\gamma), \]
  so we have
  \[ f(w^k) - f^* \leq \exp\left(-k\frac{\mu}{L}\right)[f(w^0) - f^*], \]
  which is why linear convergence is sometimes called “exponential convergence”.

- We’ll have \( f(w^t) - f^* \leq \epsilon \) for any \( t \) where
  \[ t \geq \frac{L}{\mu} \log((f(w^0) - f^*)/\epsilon) = O(\log(1/\epsilon)). \]
Discussion of Linear Convergence under the PL Inequality

- PL is satisfied for many standard convex models like least squares (bonus).
  - So cost of least squares is $O(nd \log(1/\epsilon))$.

- PL is also satisfied for some non-convex functions like $w^2 + 3 \sin^2(w)$.
  - It's satisfied for PCA on a certain “Riemann manifold”.
  - But it's not satisfied for many models, like neural networks.

- The PL constant $\mu$ might be terrible.
  - For least squares $\mu$ is the smallest non-zero eigenvalue of the Hessian.

- It may be hard to show that a function satisfies PL.
  - But regularizing a convex function gives a PL function with non-trivial $\mu$...
**Strong Convexity**

- We say that a function $f$ is **strongly convex** if the function

\[ f(w) - \frac{\mu}{2} ||w||^2, \]

is a convex function for some $\mu > 0$.

  - “If you ‘un-regularize’ by $\mu$ then it’s still convex.”

- For $C^2$ functions this is equivalent to assuming that

\[ \nabla^2 f(w) \succeq \mu I, \]

that the eigenvalues of the Hessian are at least $\mu$ everywhere.

- **Two nice properties of strongly-convex functions:**
  - A **unique solution** exists.
  - $C^1$ strong-convex functions **satisfy the PL inequality**.
Strong Convexity Implies PL Inequality

- As before, from Taylor’s theorem we have for $C^2$ functions that

$$f(v) = f(w) + \nabla f(w)^T (v - w) + \frac{1}{2} (v - w)^T \nabla^2 f(u) (v - w).$$

- By strong-convexity, $d^T \nabla^2 f(u) d \geq \mu \|d\|^2$ for any $d$ and $u$.

$$f(v) \geq f(w) + \nabla f(w)^T (v - w) + \frac{\mu}{2} \|v - w\|^2$$

- Treating right side as function of $v$, we get a quadratic lower bound on $f$. 

![Graph showing quadratic lower bound](image-url)
Strong Convexity Implies PL Inequality

- As before, from Taylor’s theorem we have for $C^2$ functions that
  \[ f(v) = f(w) + \nabla f(w)^T (v - w) + \frac{1}{2} (v - w)^T \nabla^2 f(u) (v - w). \]

- By strong-convexity, $d^T \nabla^2 f(u) d \geq \mu \|d\|^2$ for any $d$ and $u$.
  \[ f(v) \geq f(w) + \nabla f(w)^T (v - w) + \frac{\mu}{2} \|v - w\|^2. \]

- Treating right side as function of $v$, we get a quadratic lower bound on $f$.

- Minimize both sides in terms of $v$ gives
  \[ f^* \geq f(w) - \frac{1}{2\mu} \|\nabla f(w)\|^2, \]
  which is the PL inequality (bonus slides show for $C^1$ functions).
Combining Lipschitz Continuity and Strong Convexity

- Lipschitz continuity of gradient gives **guaranteed progress**.
- Strong convexity of functions gives **maximum sub-optimality**.

Progress on each iteration will be at least a fixed fraction of the sub-optimality.
Effect of Regularization on Convergence Rate

- We said that $f$ is strongly convex if the function

$$f(w) - \frac{\mu}{2} \|w\|^2,$$

is a convex function for some $\mu > 0$.

- If we have a convex loss $f$, adding L2-regularization makes it strongly-convex,

$$f(w) + \frac{\lambda}{2} \|w\|^2,$$

with $\mu$ being at least $\lambda$.

- So adding L2-regularization can improve rate from sublinear to linear.
  - Go from exponential $O(1/\epsilon)$ to polynomial $O(\log(1/\epsilon))$ iterations.
  - And guarantees a unique solution.
Effect of Regularization on Convergence Rate

- Our convergence rate under PL was
  \[ f(w^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k f(w^0) - f^* \]

- For L2-regularized least squares we have
  \[ L = \frac{\max\{\text{eig}(X^T X)\} + \lambda}{\min\{\text{eig}(X^T X)\} + \lambda} \]

- So as \( \lambda \) gets larger \( \rho \) gets closer to 0 and we converge faster.

- The number \( \frac{L}{\mu} \) is called the condition number of \( f \).
  - For least squares, it’s the “matrix condition number” of \( \nabla^2 f(w) \).
Nesterov, Newton, and Newton Approximations

- There are **accelerated gradient methods** for strongly-convex functions.
  - They improve the rate to
    \[ f(w^k) - f^* \leq \left(1 - \sqrt{\frac{\mu}{L}}\right)^k [f(w^0) - f^*], \]
  - which is a faster linear convergence rate.
    - Nearly achieves optimal possible dimension-independent rate.

- Alternately, **Newton’s method** achieves **superlinear convergence rate**.
  - Under strong-convexity and using both \(\nabla f\) and \(\nabla^2 f\) being Lipschitz.
  - But unfortunately this gives a **superlinear iteration cost**.

- There are also **linear-time approximations to Newton** (see bonus):
  - Barzilai-Borwein step-size for gradient descent (findMin.jl).
  - Limited-memory Quasi-Newton methods like L-BFGS.
  - Hessian-free Newton methods.
  - Work amazing for many problems, but don’t achieve superlinear convergence.
Sublinear/linear/superlinear convergence measure speed of convergence.

Polyak-Łojasiewicz inequality leads to linear convergence of gradient descent.

- Only needs $O(\log(1/\epsilon))$ iterations to get within $\epsilon$ of global optimum.

Strongly-convex differentiable functions satisfy PL-inequality.

- Adding L2-regularization makes gradient descent go faster.

Next time: why does L1-regularization set variables to 0?
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 $t$, algorithm choose an $x^t$ and receives $f(x^t)$ and $\nabla f(x^t)$.
  - To choose $x^t$, algorithm can do anything that doesn’t involve $f$.

- Common variation is zero-order oracle where algorithm only receives $f(x^t)$. 
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 \( f(x^t) - f(x^*) > \epsilon \) 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 $x$. 
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 $x$. 
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 $x$. 

![Diagram of a function with a point and tangent lines indicating Lipschitz continuity](image-url)
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 $x$. 
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 \textit{Lipschitz-continuous},
  \[
  |f(x) - f(y)| \leq L\|x - y\|.
  \]

- Function can’t change arbitrarily fast as you change \( x \).

- Under only this assumption, \textit{any algorithm requires at least \( \Omega(1/\epsilon^d) \) iterations.}

- An optimal \( O(1/\epsilon^d) \) \textit{worst-case rate is achieved by a grid-based search method.}

- You can also achieve \textit{optimal rate in expectation} by \textit{random guesses}.
  - Lipschitz-continuity implies there is a ball of \( \epsilon \)-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)
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.
For strongly-convex functions:
- Sub-gradient methods achieve optimal rate of $O(1/\epsilon)$.
- If $\nabla f$ is Lipschitz continuous, we’ve shown that gradient descent has $O(\log(1/\epsilon))$.

Nesterov’s algorithms improves this from $O\left(\frac{L}{\mu} \log(1/\epsilon)\right)$ to $O\left(\sqrt{\frac{L}{\mu}} \log(1/\epsilon)\right)$.
- Corresponding to linear convergence rate with $\rho = (1 - \sqrt{\frac{\mu}{L}})$.
- This is close to the optimal dimension-independent rate of $\rho = \left(\frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}\right)^2$. 

**Complexity of Minimizing Strongly-Convex Functions**
Why is $\mu \leq L$?

The descent lemma for functions with $L$-Lipschitz $\nabla f$ is that

$$f(v) \leq f(w) + \nabla f(w)^T(v - w) + \frac{L}{2} \|v - w\|^2.$$ 

Minimizing both sides in terms of $v$ (by taking the gradient and setting to 0 and observing that it’s convex) gives

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

So with PL and Lipschitz we have

$$\frac{1}{2\mu} \|\nabla f(w)\|^2 \geq f(w) - f^* \geq \frac{1}{2L} \|\nabla f(w)\|^2,$$

which implies $\mu \leq L$. 
$C^1$ Strongly-Convex Functions satisfy PL

If $g(x) = f(x) - \frac{\mu}{2} \|x\|^2$ is convex then from $C^1$ definition of convexity

$$g(y) \geq g(x) + \nabla g(x)^T(y - x)$$

or that

$$f(y) - \frac{\mu}{2} \|y\|^2 \geq f(x) - \frac{\mu}{2} \|x\|^2 + (\nabla f(x) - \mu x)^T(y - x),$$

which gives

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\mu}{2} \|y\|^2 - \mu x^Ty + \frac{\mu}{2} \|x\|^2$$

$$= f(x) + \nabla f(x)^T(y - x) + \frac{\mu}{2} \|y - x\|^2, \quad \text{(complete square)}$$

the inequality we used to show $C^2$ strongly-convex function $f$ satisfies PL.
Linear Convergence without Strong-Convexity

- The least squares problem is convex but not strongly convex.
  - We could add a regularizer to make it strongly-convex.
  - But if we really want the MLE, are we stuck with sub-linear rates?

- Many conditions give linear rates that are weaker than strong-convexity:
  - 1993: Error bounds.

- Least squares satisfies all of the above.

- Do we need to study any of the newer ones?
  - No! All of the above imply PL except for QG.
  - But with only QG gradient descent may not find optimal solution.
PL Inequality for Least Squares

Least squares can be written as $f(x) = g(Ax)$ for a $\sigma$-strongly-convex $g$ and matrix $A$, we'll show that the PL inequality is satisfied for this type of function.

The function is minimized at some $f(y^*)$ with $y^* = Ax$ for some $x$, let's use $\mathcal{X}^* = \{ x | Ax = y^* \}$ as the set of minimizers. We'll use $x_p$ as the “projection” (defined next lecture) of $x$ onto $\mathcal{X}^*$.

$$f^* = f(x_p) \geq f(x) + \langle \nabla f(x), x_p - x \rangle + \frac{\sigma}{2} ||A(x_p - x)||^2$$

$$\geq f(x) + \langle \nabla f(x), x_p - x \rangle + \frac{\sigma \theta(A)}{2} ||x_p - x||^2$$

$$\geq f(x) + \min_y \left[ \langle \nabla f(x), y - x \rangle + \frac{\sigma \theta(A)}{2} ||y - x||^2 \right]$$

$$= f(x) - \frac{1}{2\theta(A)\sigma} ||\nabla f(x)||^2.$$

The first line uses strong-convexity of $g$, the second line uses the “Hoffman bound” which relies on $\mathcal{X}^*$ being a polyhedral set defined in this particular way to give a constant $\theta(A)$ depending on $A$ that holds for all $x$ (in this case it’s the smallest non-zero singular value of $A$), and the third line uses that $x_p$ is a particular $y$ in the min.
Linear Convergence for “Locally-Nice” Functions

- For linear convergence it’s sufficient to have

\[ L[f(x^{t+1}) - f(x^t)] \geq \frac{1}{2} \| \nabla f(x^t) \|^2 \geq \mu[f(x^t) - f^*], \]

for all \( x^t \) for some \( L \) and \( \mu \) with \( L \geq \mu > 0 \).

(technically, we could even get rid of the connection to the gradient)

- Notice that this only needs to hold for all \( x^t \), not for all possible \( x \).
  - We could get linear rate for “nasty” function if the iterations stay in a “nice” region.
  - We can get lucky and converge faster than the global \( L/\mu \) would suggest.

- Arguments like this give linear rates for some non-convex problems like PCA.
Convergence of Iterates

- Under strong-convexity, you can also show that the iterations converge linearly.

- With a step-size of $1/L$ you can show that

  $$\|w^{k+1} - w^*\| \leq \left(1 - \frac{\mu}{L}\right)\|w^k - w^*\|.$$  

- If you use a step-size of $2/(\mu + L)$ this improves to

  $$\|w^{k+1} - w^*\| \leq \left(\frac{L - \mu}{L + \mu}\right)\|w^k - w^*\|.$$  

- Under PL, the solution $w^*$ is not unique.
  - You can show linear convergence of $\|w^k - w^k_p\|$, where $w^k_p$ is closest solution.
We showed that we require $O(1/\epsilon)$ iterations for gradient descent to get norm of gradient below $\epsilon$ in the non-convex setting.

Is it possible to improve on this with a gradient-based method?

Yes, in 2016 it was shown that a gradient method can improve this to $O(1/\epsilon^{3/4})$:
- Combination of acceleration and trying to estimate a “local” $\mu$ value.
Newton’s Method

- **Newton’s method** is a second-order strategy.
  
  (also called IRLS for functions of the form $f(Ax)$)

- Modern form uses the update

  $$x^{t+1} = x^t - \alpha_t d^t,$$

  where $d^t$ is a solution to the system

  $$\nabla^2 f(x^t)d^t = \nabla f(x^t).$$

  (Assumes $\nabla^2 f(x^t) \succ 0$)

- Equivalent to minimizing the quadratic approximation:

  $$f(y) \approx f(x^t) + \nabla f(x^t)^T(y - x^t) + \frac{1}{2\alpha_t}(y - x^t)\nabla^2 f(x^t)(y - x^t).$$

- We can generalize the Armijo condition to

  $$f(x^{t+1}) \leq f(x^t) + \gamma \alpha \nabla f(x^t)^T d^t.$$

- Has a natural step length of $\alpha = 1$.
  
  (always accepted when close to a minimizer)
Newton’s Method
Newton’s Method

\[ f(x) \]

\[ x \]
Newton’s Method

\[ f(x) \]

\[ x - \alpha f'(x) \]
Newton’s Method

\[ Q(x) \]

\[ f(x) \]

\[ x - \alpha f'(x) \]
Newton’s Method

\[ f(x) \]

\[ x - \alpha \nabla f(x) \]

\[ x^k - \alpha H^{-1} \nabla f(x) \]
Convergence Rate of Newton’s Method

- If $\mu I \preceq \nabla^2 f(x) \preceq LI$ and $\nabla^2 f(x)$ is Lipschitz-continuous, then close to $x^*$ Newton’s method has local superlinear convergence:
  
  $$f(x^{t+1}) - f(x^*) \leq \rho_t[f(x^t) - f(x^*)],$$

  with $\lim_{t \to \infty} \rho_t = 0$.

- Converges very fast, use it if you can!

- But Newton’s method is expensive if dimension $d$ is large:
  - Requires solving $\nabla^2 f(x^t) d^t = \nabla f(x^t)$.

- “Cubic regularization” of Newton’s method gives global convergence rates.
Practical Approximations to Newton’s Method

- **Practical Newton-like** methods (that can be applied to large-scale problems):
  - **Diagonal** approximation:
    - Approximate Hessian by a diagonal matrix $D$ (cheap to store/invert).
    - A common choice is $d_{ii} = \nabla^2_{ii} f(x^t)$.
    - This sometimes helps, often doesn’t.
  - **Limited-memory quasi-Newton** approximation:
    - Approximates Hessian by a diagonal plus low-rank approximation $B^t$, 
      \[ B^t = D + UV^T, \]
    - which supports fast multiplication/inversion.
    - Based on “quasi-Newton” equations which use differences in gradient values.
      \[ (\nabla f(x^t) - \nabla f(x^{t-1})) = B^t(x^t - x^{t-1}). \]
    - A common choice is L-BFGS.
Practical Approximations to Newton’s Method

- Practical Newton-like methods (that can be applied to large-scale problems):
  - Barzilai-Borwein approximation:
    - Approximates Hessian by the identity matrix (as in gradient descent).
    - But chooses step-size based on least squares solution to quasi-Newton equations.
      \[ \alpha_t = -\alpha_t \frac{v^T \nabla f(x^t)}{\|v\|^2}, \quad \text{where} \quad v = \nabla f(x^t) - \nabla f(x^{t-1}). \]
    - Works better than it deserves to (findMind.jl).
    - We don’t understand why it works so well.
Practical Approximations to Newton’s Method

- **Practical Newton-like methods** (that can be applied to large-scale problems):
  - **Hessian-free Newton**:
    - Uses conjugate gradient to approximately solve Newton system.
    - Requires Hessian-vector products, but these cost same as gradient.
    - If you’re lazy, you can numerically approximate them using
      \[
      \nabla^2 f(x^t)d \approx \frac{\nabla f(x^t + \delta d) - \nabla f(x^t)}{\delta}.
      \]
    - If \( f \) is analytic, can compute exactly by evaluating gradient with complex numbers.
      (look up “complex-step derivative”)
  - A related approach to the above is **non-linear conjugate gradient**.
Result after 25 evaluations of limited-memory solvers on 2D rosenbrock:

x1 = 0.0000, x2 = 0.0000 (starting point)
x1 = 1.0000, x2 = 1.0000 (optimal solution)

x1 = 0.3654, x2 = 0.1230 (minFunc with gradient descent)
x1 = 0.8756, x2 = 0.7661 (minFunc with Barzilai-Borwein)
x1 = 0.5840, x2 = 0.3169 (minFunc with Hessian-free Newton)
x1 = 0.7478, x2 = 0.5559 (minFunc with preconditioned Hessian-free Newton)
x1 = 1.0010, x2 = 1.0020 (minFunc with non-linear conjugate gradient)
x1 = 1.0000, x2 = 1.0000 (minFunc with limited-memory BFGS - default)
Superlinear Convergence in Practice?

- You get **local superlinear convergence** if:
  - Gradient is Lipschitz-continuous and $f$ is strongly-convex.
  - Function is in $C^2$ and Hessian is **Lipschitz continuous**.
  - Oracle is second-order and method **asymptotically uses Newton's direction**.

- But the **practical Newton-like methods** don't achieve this:
  - Diagonal scaling, Barzilai-Borwein, and L-BFGS don't converge to Newton.
  - Hessian-free uses conjugate gradient which isn't superlinear in high-dimensions.

- Full quasi-Newton methods achieve this, but require $\Omega(d^2)$ memory/time.