Numerical Optimization for Machine Learning

Momentum, Acceleration, and Second-Order Methods

Mark Schmidt

University of British Columbia

Summer 2022

Last Time: Convergence Rate of Gradient Descent

- We considered gradient descent, $w^{k+1} = w^k - \alpha_k \nabla f(w^k)$.
- If gradient is Lipschitz continuous and function is bounded below:
  - Needs $t = O(L/\epsilon)$ iterations to find a w with $\|\nabla f(w)\|^2 \leq \epsilon$.
- If gradient is Lipschitz continuous and function satisfies PL (or strong-convexity):
  - Needs $t = O((L/\mu)\log(1/\epsilon))$ iterations to a find a w with $f(w) - f(w^*) \leq \epsilon$.
- Above hold for fixed step size of $\alpha_k = 1/L$, and various practical methods.
  - Practical methods work better in practice since they often take bigger step sizes.
- Today: do algorithms that converge faster exist?

Oracle Model of Computation

- To analyze algorithms and *what is possible* we need two ingredients:
  1. Assumptions about the function like Lipschitz, PL, convexity, and so on.
     - If the set of functions is unrestricted, we can design impossible-to-optimize functions.
  2. Model of computation, restricting what the algorithm can do.
     - If the algorithm is unrestricted, then our algorithm could be: `return` w^*.

- Standard model of computation is the first-order oracle model:
  1. At each iteration the algorithm chooses a point w^k.
  2. The algorithm then receives $f(w^k)$ and $\nabla f(w^k)$.

- We analyze how many iterations are needed to make some quantity small.
  - Usually $\|\nabla f(w^k)\|$, $f(w^k) - f^*$, or $\|w^k - w^*\|$.

- Given assumptions and oracle model, many works:
  - Prove upper bounds on iteration complexity of specific algorithms.
  - Prove lower bounds on iteration complexity across algorithms.

Iteration Complexity Lower and Upper Bounds

- In first-order oracle model the algorithm itself is often unrestricted, but it can only learn about the function through evaluations at the chosen w^k.
- Often prove lower bounds by designing a "worst function" under the assumptions.
  - And show that you can only slowly discover the minimum location from oracle.
- Common variations on the first-order oracle model:
  - Zero-order oracles only return $f(w^k)$.
  - Second-order oracles also return $\nabla^2 f(w^k)$.
- Another variation is requiring the algorithm to be dimension independent:
  - The number of oracle calls does not directly depend on the dimension d.
    - Our gradient descent bounds were dimension independent.
  - It may depend on quantities L, that might grow as d increases.
    - But you can have infinite-dimensional problems where L is finite.

Outline

Gradient Descent and Quadratics

- Consider a minimizing a strongly-convex quadratic function,

$$f(w) = \frac{1}{2} w^T A w + b^T w + c,$$

where $\mu I \preceq A \preceq LI$ for positive μ and L.

  - So the quadratic is strongly-convex with a Lipschitz-continuous gradient.
  - Examples: least squares with independent features, L2-regularized least squares.

- With $\alpha_k = 1/L$ gradient descent satisfies

$$\|w^k - w^*\| = \left(1 - \frac{\mu}{L}\right)^k \|w^0 - w^*\|.$$

- The optimal step size is $\alpha_k = 2/(L+\mu)$ which gives

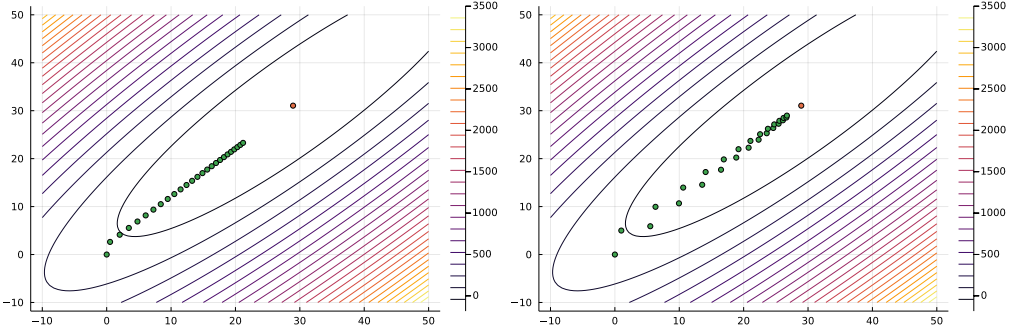
$$\|w^k - w^*\| = \left(\frac{L-\mu}{L+\mu}\right)^k \|w^0 - w^*\|,$$

which is faster because $(1 - \mu/L) = (L-\mu)/L < (L-\mu)/(L+\mu)$.

  - And descent lemma with $\nabla f(w^*) = 0$ implies $f(w^k) - f(w^*) \leq \frac{L}{2}\|w^k - w^*\|^2$.

Gradient Descent and Quadratics

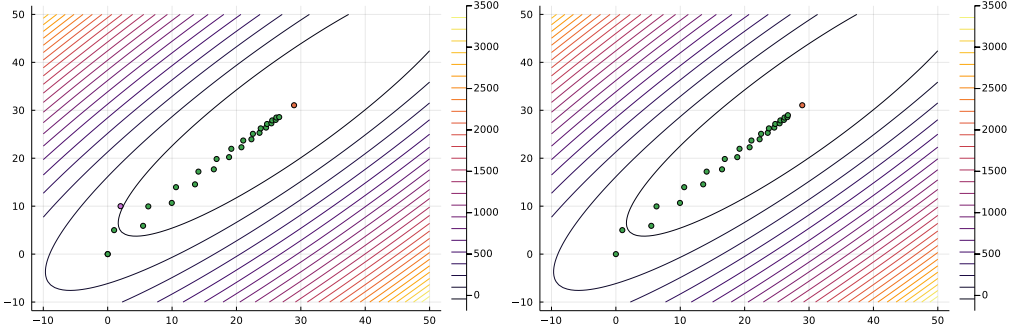
- Gradient descent with $\alpha_k = 1/L$ (left) and $\alpha_k = 2/(L+\mu)$ (right) on a quadratic:



- Both step-sizes satisfy $\alpha_k < 2/L$ so decrease function at each step.
  - Using $\alpha_k = 2/(L+\mu)$ takes bigger steps but requires knowing μ.
  - For the above function, $1/L \approx 0.26$ and $2/(L+\mu) \approx 0.5$.

Gradient Descent and Quadratics

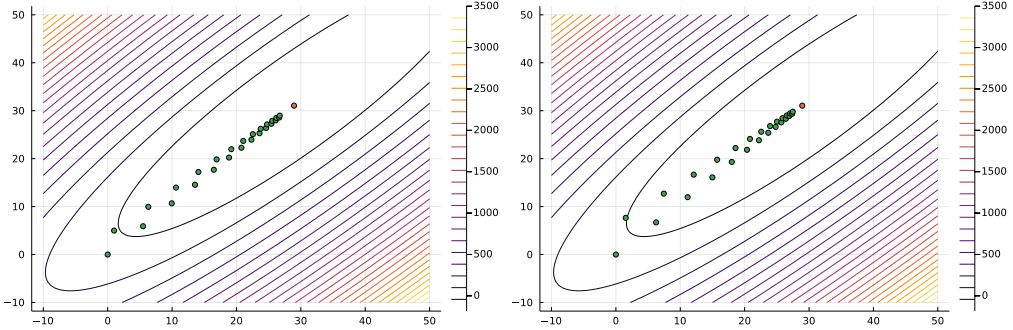
- Gradient descent with $\alpha_k = 1/\hat{L}$ (left) and $\alpha_k = 2/(L+\mu)$ (right) on a quadratic:



- The approximate Lipschitz constant $\hat{L}$ is 0.5 after the first iteration.
  - So it is close to $2/(L+\mu)$ without knowing L or μ.
  - But for other functions this step size may be better or worse.

Gradient Descent and Quadratics

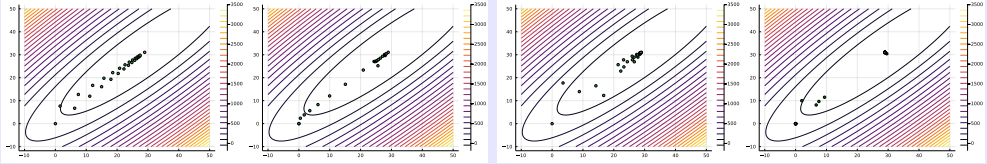
- Gradient descent with $\alpha_k = 2/(L+\mu)$ (left) and optimal α_k on a quadratic:



- For quadratic functions, you can solve for optimal step-size at each step.
  - Does not require knowing μ: $\alpha_k = \nabla f(w^k)^T \nabla f(w^k) / \nabla f(w^k)^T A \nabla f(w^k)$.
  - Above, $2/(L+\mu) \approx 0.5$, optimal step alternated between ≈ 0.37 and ≈ 0.76.

Gradient Descent and Quadratics

- The "optimal" step size may not make the most progress across iterations:



- Step sizes left to right: optimal, Malitsky-Mischenko, Polyak, Barzilai-Borwein.

- The Barzilai-Borwein step size leads to superlinear convergence for 2d quadatics.
  - Basically solves the above problem in 5 steps.
  - Convergence rate beyond 2d quadratics case not known

Heavy-Ball Method

- The $(L-\mu)/(L+\mu)$ rate is tight for gradient descent on SC quadratics
  - There exist quadratic where the method converges at exactly this rate.
  - Optimal dimension-independent rate for gradient descent.

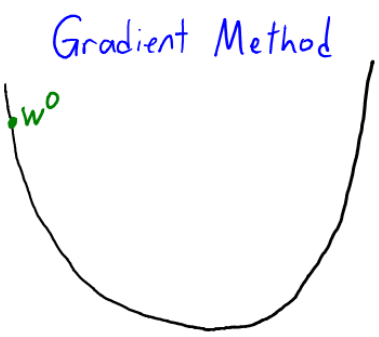
- But there exist faster algorithm for SC quadratics in the first-order oracle model.

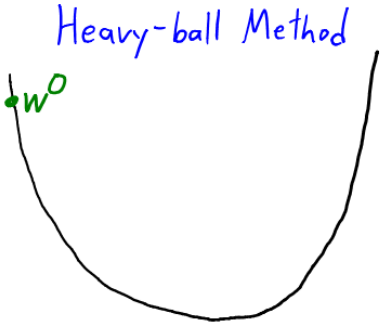
- A classic example is Polyak's heavy-ball method [1964],

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

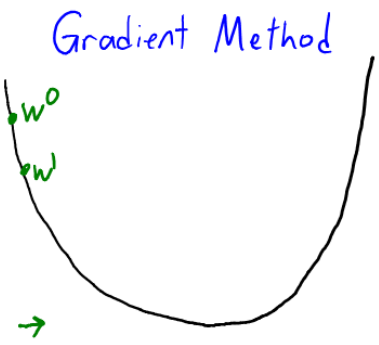
  which adds a momentum term to gradient descent for $k > 1$.
  - Extra term makes us "go further in the previous direction".
  - Has an extra momentum parameter $\beta_k \in [0, 1)$.

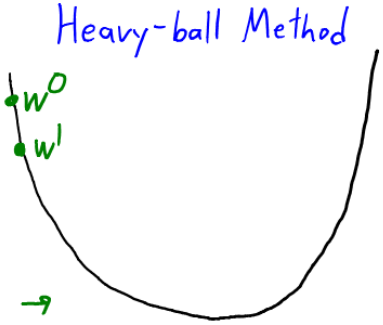
Heavy-Ball Method Method



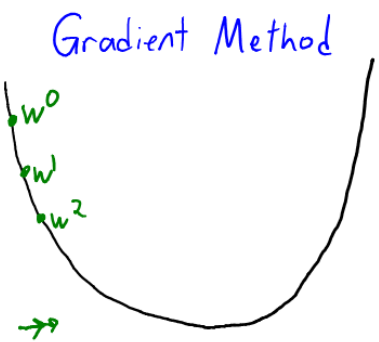


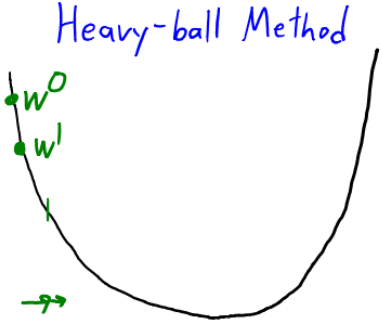
Heavy-Ball Method Method



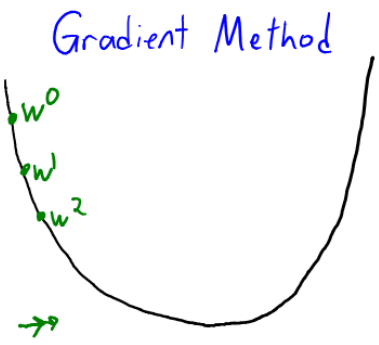


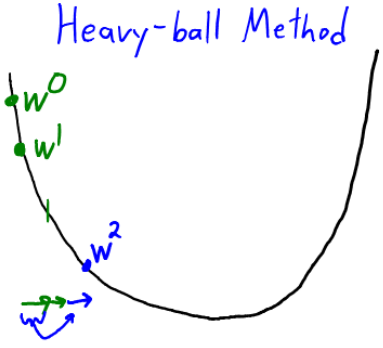
Heavy-Ball Method Method



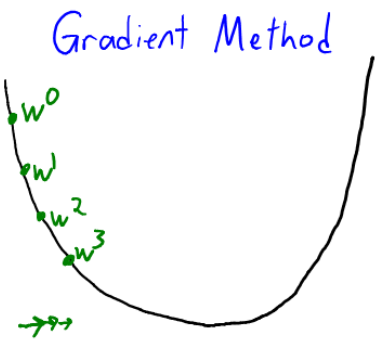


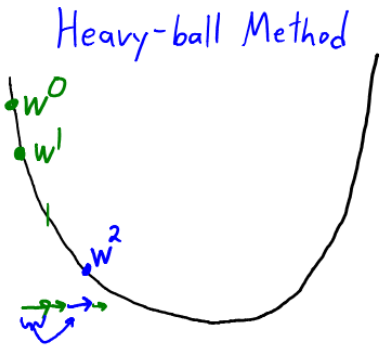
Heavy-Ball Method Method



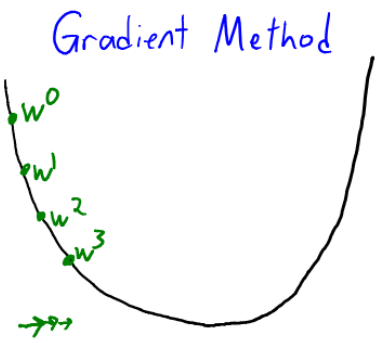


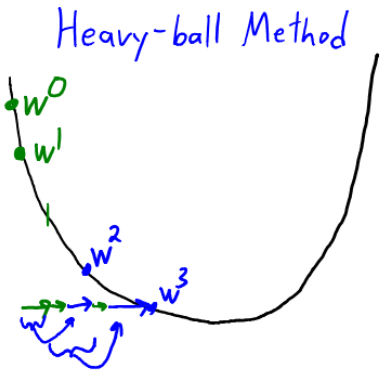
Heavy-Ball Method Method



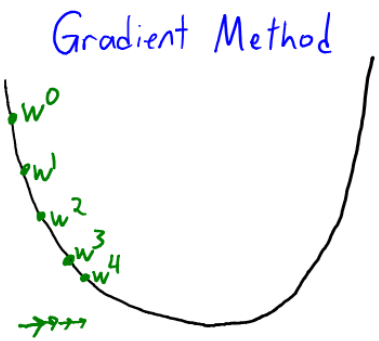


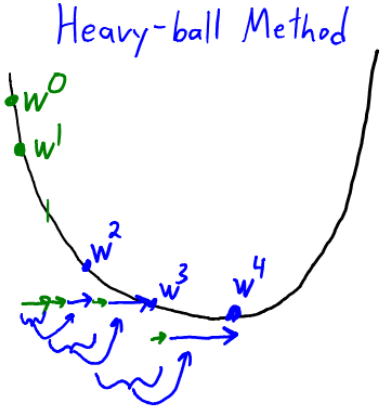
Heavy-Ball Method Method



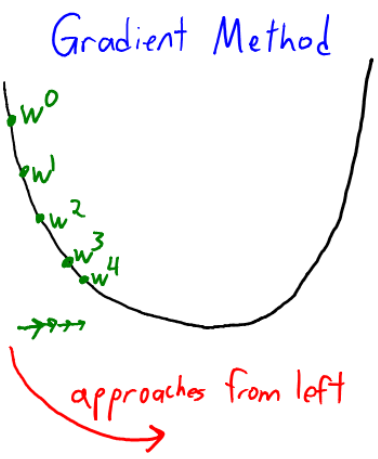


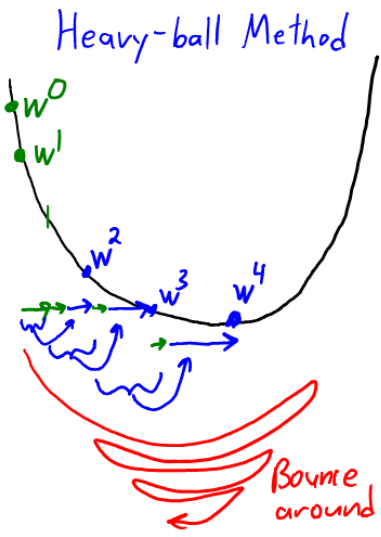
Heavy-Ball Method Method





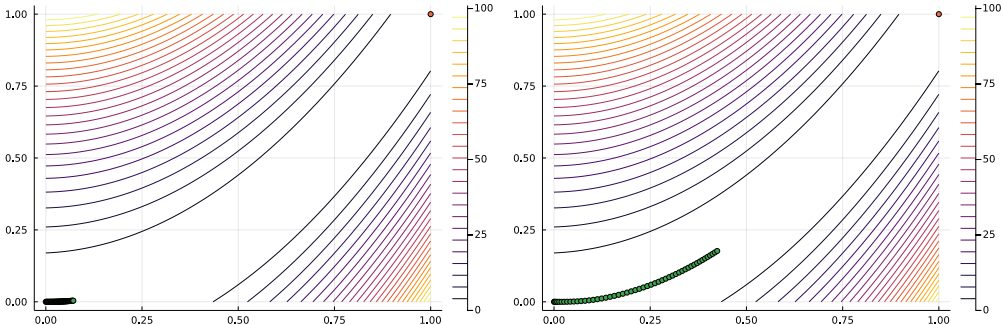
Heavy-Ball Method Method





Gradient Descent vs. Heavy Ball on Rosenbrock

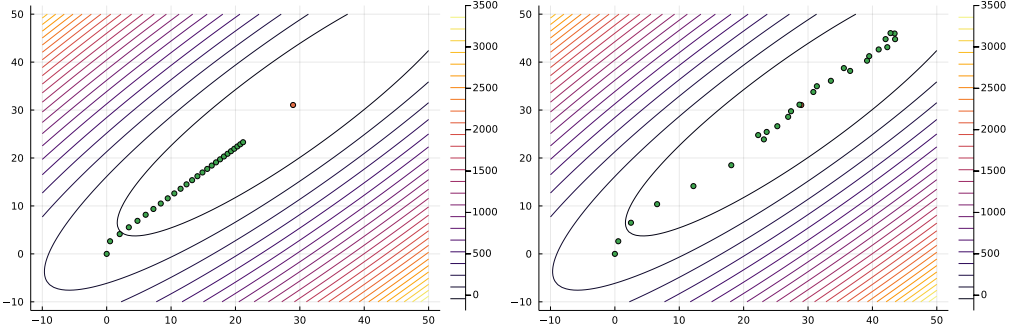
- Gradient descent ($\alpha_k = 1/L$) vs. heavy ball ($\alpha_k = 1/L$ and $\beta_k = 0.9$):



- Momentum in heavy-ball can significantly speed up gradient descent.

Gradient Descent vs. Heavy Ball on Quadratic

- Gradient descent ($\alpha_k = 1/L$) vs. heavy ball ($\alpha_k = 1/L$ and $\beta_k = 0.9$):



- Heavy-ball method can increase function and "overshoot" the optimum.
  - But iterations may be closer to solution on average.

Fast Convergence of Heavy-Ball Method on Quadratics

- Consider the heavy-ball method with the choices

$$\alpha_k = \frac{4}{(\sqrt{L}+\sqrt{\mu})^2}, \quad \beta_k = \left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2.$$

- Under these choices the heavy-ball method has

$$\|w^k - w^*\| \leq \left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}} + \epsilon_k\right)^k \|w^0 - w^*\|,$$

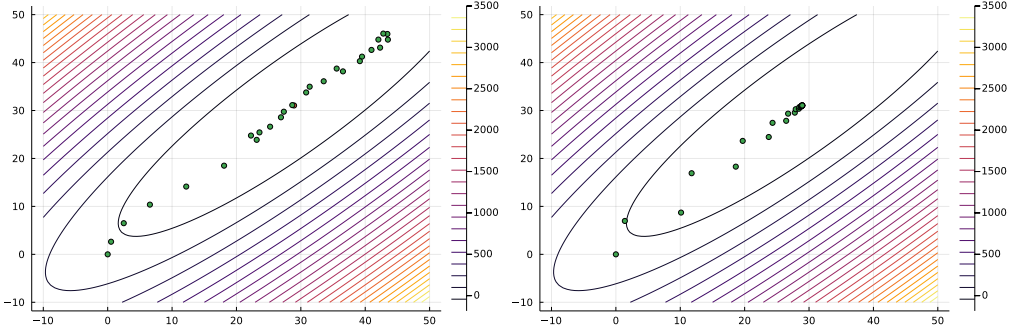
where $\epsilon_k \to 0$.

  - Instead of directly bounding $\|w^k - x^*\|$, proof bounds $\|w^k - w^*\|^2 + \|w^{k-1} - w^*\|^2$.
  - This is a special case of the Lyapunov potential function proof technique.
    - Show that a function "bigger than what you want" is converging at right rate.

- The optimal dimension-independent rate in first-oracle model is $\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$.
  - So with this choice the heavy-ball method is close to optimal.

Heavy Ball on Quadratic

- Heavy ball ($\alpha_k = 1/L$ and $\beta_k = 0.9$) and $\left(\alpha_k = \frac{4}{(\sqrt{L}+\sqrt{\mu})^2}, \beta_k = \left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2\right)$.



- For this problem, β_k on the left is ≈ 0.4.
  - Unfortunately, the setting on the right requires knowing μ.

Conjugate Gradient: Heavy-Ball with Optimal Parameters

- For quadratics, we could optimize α_k and β_k on each iteration.
  - At each iteration, choose α_k and β_k that maximally decrease f.
    - "Plane search" ("subspace optimization") along two directions instead of "line search".
- This "optimal heavy-ball" method is called the conjugate gradient (CG) method:

$$\alpha_k = \nabla f(w^k)^T d^k / d_k^T A d_k \qquad \text{(step size for gradient direction)}$$

$$\beta_k = \alpha_k \hat{\beta}_{k-1} / \alpha_{k-1} \qquad \text{(momentum parameter, } \beta_0 = 0\text{)}$$

$$w^{k+1} = w^k - \alpha_k \nabla f(w^k) + \beta_k (w^k - w^{k-1}) \qquad \text{(heavy-ball update)}$$

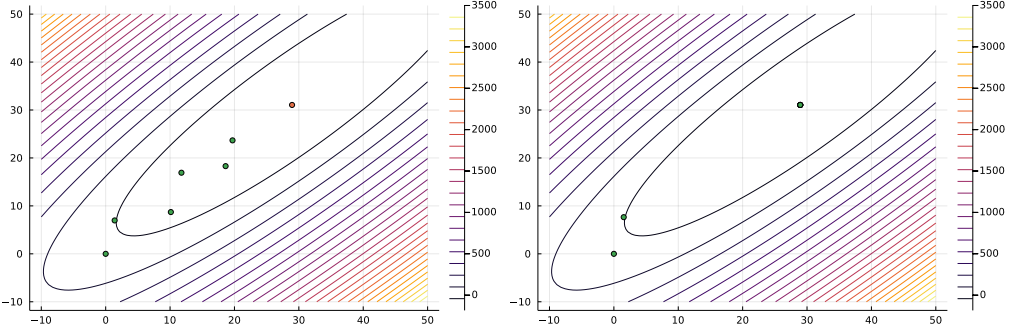
$$\hat{\beta}_k = \nabla f(w^{k-1})^T A d_{k-1} / d_{k-1}^T A d_{k-1}$$

$$d_k = -\nabla f(w^k) + \hat{\beta}_k d_{k-1} \qquad \text{(search direction, } d_0 = -\nabla f(w^0)\text{)}$$

- Properties:
  - Gradients between iterations are orthogonal, $\nabla f(w^k)^T \nabla f(w^{k-1}) = 0$.
  - Achieves optimal $(\sqrt{L} - \sqrt{\mu})/(\sqrt{L} + \sqrt{\mu})$ dimension-independent rate.
  - Faster dimension-dependent analysis (via Chebyshev polynomials).

Heavy Ball vs. Conjugate Gradient on Quadratic

- Heavy ball with $\left(\alpha_k = \frac{4}{(\sqrt{L}+\sqrt{\mu})^2}, \beta_k = \left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2\right)$ and conjugate gradient:



- Conjugate gradient method minimizes the two-dimensional quadratic in 2 steps.
  - You can show that CG minimizes a d-dimensional quadratic in d steps.
    - Tends not to happen on computers due to floating point issues.
  - Note that conjugate gradient does not need to know L or μ.

Outline

Convex Functions

- An important generalization of quadratics is convex functions.

- Fitting many models involves minimizing a non-quadratic convex function:
  - Robust regression with the Huber loss or L1-loss.
  - Binary and multi-class logistic regression.
  - Binary and multi-class support vector machines.
  - Density estimation with exponential family distributions like Gaussians.
  - Fitting various types of graphical models.
  - Adding L2-regularization or L1-regularization to any of the above.
    - With L2-regularization, they are all strongly-convex.

- Convexity implies that all stationary points are global optima.
  - So differentiable convex functions can be optimized with gradient descent.

- We are not going to review properties of convex functions during these lectures.
  - See the webpage for notes on convex sets and functions.

Converge Rates of First-Order Methods on [Strongly-]Convex Functions

- Convergence rates for gradient descent applied to convex functions:
  - With $\alpha_k = 1/L$, requires $O(1/k)$ iterations for convex.
    - Using telescoping argument as we did for non-convex functions.
  - With $\alpha_k = 1/L$, requires $O((1 - \mu/L)^{2k})$ for strongly-convex.
    - We showed a similar rate for the special case of PL functions.
  - With $\alpha_k = 2/(L + \mu)$, requires $O(((L - \mu)/(L + \mu)))^{2k})$ for strongly-convex.

- Dimension-independent lower bounds (Lipschitz gradient, first-order oracle):
  - There exist convex functions requiring $\Omega(1/k^2)$ iterations to have $f(w^k) - f^* \leq \epsilon$.
    - For any algorithm (so we can expect sublinear rates at best).
  - For strongly-convex functions we require $\Omega(((\sqrt{L} - \sqrt{\mu})/(\sqrt{L} + \sqrt{\mu}))^{2k})$.
    - The same speed we saw for strongly-convex quadratics.

- We call a first-order method accelerated if it either:
  - Has a $O(1/k^2)$ rate for convex functions.
  - Has a linear rate depending on $\sqrt{L}$ and $\sqrt{\mu}$ for strongly-convex functions.

Heavy Ball and Conjugate Gradient for Convex Functions?

- Is heavy-ball method an accelerated method?
  - No! For some convex functions heavy-ball is not faster than gradient descent.

- There are complications in generalizing conjugate gradient for acceleration:
  - For convex functions, need to optimize over a 3-dimensional subspace instead of 2.
    - For non-quadratic functions, usually have no fast way to optimize over subspaces.
  - For strongly-convex functions, also need to periodically restart the method.
    - You can restart by setting $\beta_k = 0$, to "clear" the memory.
    - But unfortunately the restart frequency depends on L/μ.

- Problems where we can optimize efficiently over the subspace:
  - Linear composition problems, $f(w) = g(Aw)$.
    - Assuming multiplication by A is bottleneck.
  - Some other functions that can be expressed as multi-linear maps.
    - Matrix factorization (PCA) with sufficiently low rank.
  - See the sequential subspace optimization (SESOP) papers and read Betty's thesis.
    - In practice, subspace optimization with gradient plus momentum works really well.

Nesterov's Accelerated Gradient Method

- In 1983 Nesterov proposed the first efficient accelerated gradient method:

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

- We can write the heavy-ball method in a similar form:

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

- Nesterov's method computes gradient after applying momentum.
  - If gradient descent is $w^{k+1} = \mathrm{GD}(w^k)$, then:
    - Momentum is $w^{k+1} = \mathrm{GD}(w^k) + \beta_k (w^k - w^{k-1})$.
    - Nesterov is $w^{k+1} = \mathrm{GD}(w^k + \beta_k (w^k - w^{k-1}))$.

Writing Nesterov's Algorithm with 3 Directions

- We can alternately write Nesterov's algorithm as:

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

  where we add "momentum of the gradient" to the heavy-ball method.

  (I use w^k above but this is technically the momentum sequence v^k)

- From this point of view, Nesterov's method is taking a step along 3 directions:
  - Gradient and momentum (like heavy-ball and CG) and old gradient direction.
    - Using the gradient difference can be viewed as approximating effect of Hessian.

- Consider optimizing a one-dimensional convex function:
  - If sign of gradient stays same, Nesterov's algorithm speeds up heavy-ball.
  - If sign of gradient changes (overshoot min), it "slows down" faster.

- Many accelerated variations exist, proofs are often not fun or enlightening.

Nesterov's Accelerated Gradient: Setting α and β (Theory)

- Nesterov's method is typically analyzed with $\alpha_k = 1/L$.
- For convex functions, accelerated rate can be achieved with

$$\beta_k = \frac{k-1}{k+2},$$

  a momentum that converges to 1.
- For strongly-convex functions, acceleration can be achieved with constant

$$\beta_k = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}},$$

  as in the heavy-ball method.
  - Notice that you need different parameters for different problems.
    - Using a momentum that converges to 1 for strongly-convex could be very slow.
  - Unlike gradient descent which adapts to problem with standard choices.
    - Using $\alpha_k = 1/L$ maintains rate for convex, strongly-convex, and non-convex.

Nesterov's Accelerated Gradient: Setting α and β (Practice)

- We can maintain the accelerated rate without knowing L:
  - Start with a small guess $\hat{L}$.
  - Given the momentum step v^k, test the inequality

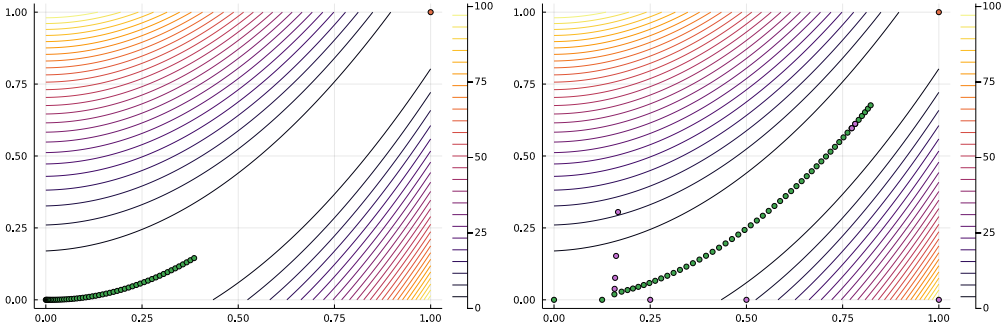
$$f(w^{k+1}) \leq f(v^k) - \frac{1}{2\hat{L}}\|\nabla f(v^k)\|^2,$$

    and double $\hat{L}$ if it is not satisfied.
    - As with gradient descent, can work much better than knowing L.

- Note that Nesterov's method is often typically non-monotonic.
  - We do not always have $f(w^{k+1}) < f(w^k)$.
  - As with momentum, this is not necessarily a bad thing.

Example: Knowing L vs. Approximation L

- $O(1/k^2)$ Nesterov with $\alpha_k = 1/L$ vs. $\alpha_k = 1/\hat{L}$:



- As with gradient descent, you can go faster with the approximation.
  - $\hat{L} = 16$ on iteration 1, then 256 for many iterations, 512 for one, and 1024 for rest.

Nesterov's Accelerated Gradient: Setting α and β (Practice)

- Do you need to know μ for strongly-convex problems?
  - On some problems $O(1/k^2)$ Nesterov is slower than gradient descent, since gradient descent adapts to best/local μ value.

- Common strategy is applying $O(1/k^2)$ algorithm with restarting:
  - Run the $O(1/k^2)$ algorithm (which increases momentum and does not need μ).
  - Occasionally stop the method and reset the momentum.
- Accelerated rate is achieved if we reset every $O(\sqrt{L/\mu})$ iterations.

- Various practical resetting strategies have been proposed:
  - Use a binary search for a best-performing fixed restart frequency.
  - Methods that check if the restart frequency is too long.
  - Methods that test whether a restart is needed.
    - Simplest method restarts if $f(w^k)$ increases.

Effect of μ Estimate

- Effect of different momentum updates on an accelerated gradient method:
  - $q = 1$ is gradient descent, $q = 0$ is $O(1/k^2)$ method, intermediate are SC methods.

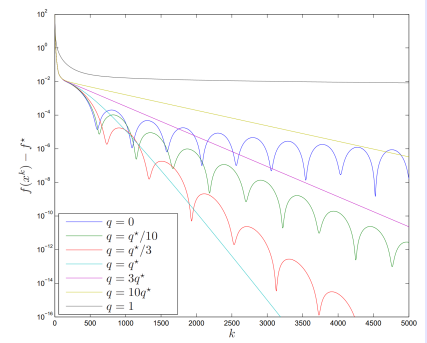


Figure 1: Convergence of Algorithm 1 with different estimates of q.

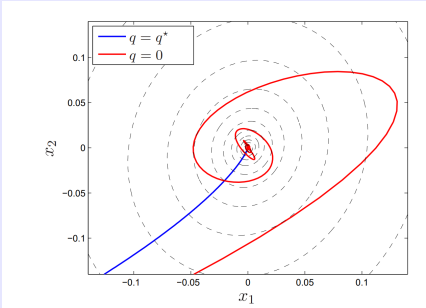


Figure 2: Sequence trajectories under Algorithm 1.

https://arxiv.org/pdf/1204.3982.pdf

- Performance degrades away from optimal update q^* (depending on L/μ).

Restarted Accelerated Gradient

- Effect of restarting on an accelerated gradient method:

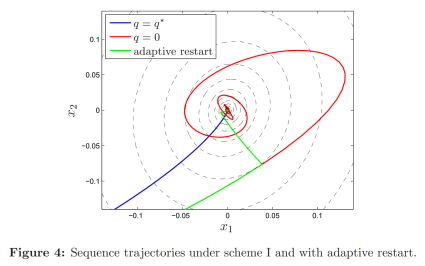
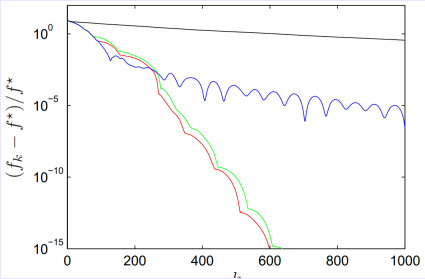


Figure 4: Sequence trajectories under scheme I and with adaptive restart.



https://arxiv.org/pdf/1204.3982.pdf

- Restarting often significantly improves performance.

Non-Linear Conjugate Gradient

- There are also various non-linear conjugate gradient methods.
  - These methods use the heavy-ball update with particular choices of β_k.
  - On each step they use a line search along the direction $d_k = g_k + \beta_k d_{k-1}$.
  - On quadratic functions with exact line search, equivalent to conjugate gradient.
  - They work best with a precise line-search along d_k.

- Many variations exist, with common variations being (with $g_k = \nabla f(w^k)$):
  - Fletcher-Reeves: $\beta_k = \langle g_k, g_k \rangle / \langle g_{k-1}, g_{k-1} \rangle$.
  - Polak-Ribiere: $\beta_k = \langle g_k, g_k - g_{k-1} \rangle / \langle g_{k-1}, g_{k-1} \rangle$.
  - Hestenes-Steifel: $\beta_k = \langle g_k, g_k - g_{k-1} \rangle / \langle g_k - g_{k-1}, d_{k-1} \rangle$.

- These methods often use restart mechasnisms.
  - Example: set $\beta_k = 0$ if directional derivative ($d_k^T \nabla f(w^k)$) is not sufficiently negative.

- These methods do not achieve accelerated rate.
- But for many problems they work amazingly well in practice.

Outline

Gradient Descent vs. Newton's Method

- Recall the second-order Taylor expansion representation of a function,

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

for some u between w and v.

- We analyze progress of gradient descent by upper-bounding last term,

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

and minimizing right side in terms of v gives gradient descent with $\alpha_k = 1/L$.

- Newton's method is obtained by minimizing a truncated Taylor series,

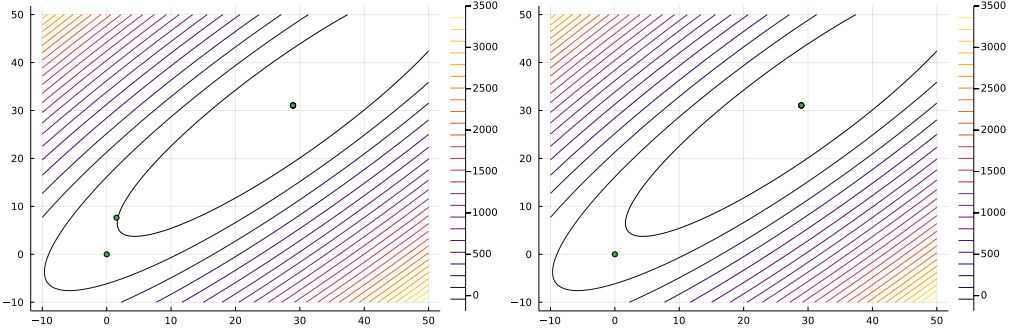
$$f(v) \approx f(w) + \nabla f(w)^T(v-w) + \frac{1}{2}(v-w)^T\nabla^2 f(w)(v-w),$$

which becomes exact as $\|v-w\|$ shrinks to 0.

  - We typically analyze Newton's method based on second-order oracle.

Example: Conjugate Gradient vs. Newton on Quadratic

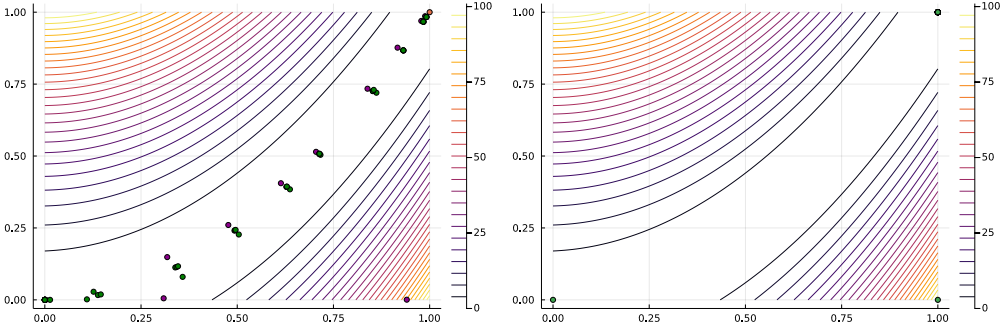
- Conjugate gradient and Newton on a 2d quadratic function:



- Newton's method finds minimizer of quadratics in one iteration.
  - The Hessian $\nabla^2 f(w)$ is constant so the approximation is exact.

Example: Gradient Descent vs. Newton on Quadratic

- Our best gradient descent method and Newton on Rosenbrock



- Newton's method finds exact minimizer in 2 iterations.
  - Though notice that first iteration increased the function a lot.

Implementation of Newton's Method (Strongly-Convex)

- If f is strongly-convex then $\nabla^2 f(w^k)$ is invertible and positive-definite.
- In this situation the minimizer in terms of v of

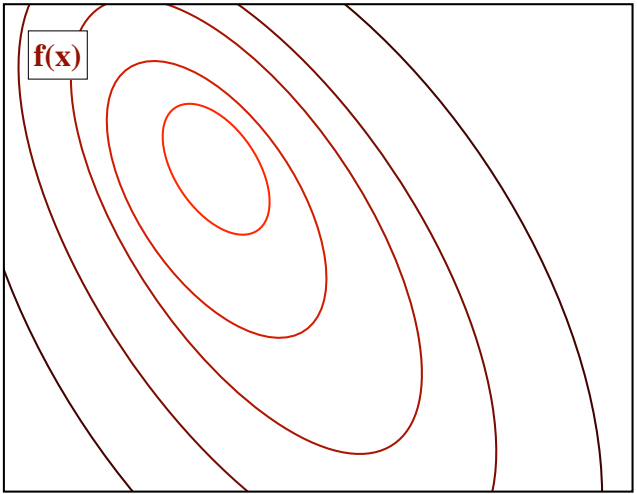
$$f(w^k) + \nabla f(w^k)^T(v - w) + \frac{1}{2}(v - w^k)^T \nabla^2 f(w^k)(v - w^k),$$

  is given by

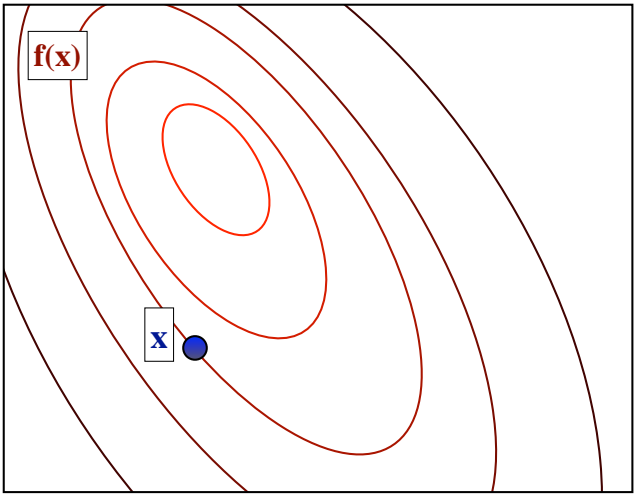
$$w^{k+1} = w^k - [\nabla^2 f(w^k)]^{-1} \nabla f(w^k),$$

  which is the Newton update.
- We do not compute the inverse Hessian, but use a Cholesky factorization.
  - Fast Gaussian elimination method for solving positive-definite linear systems.

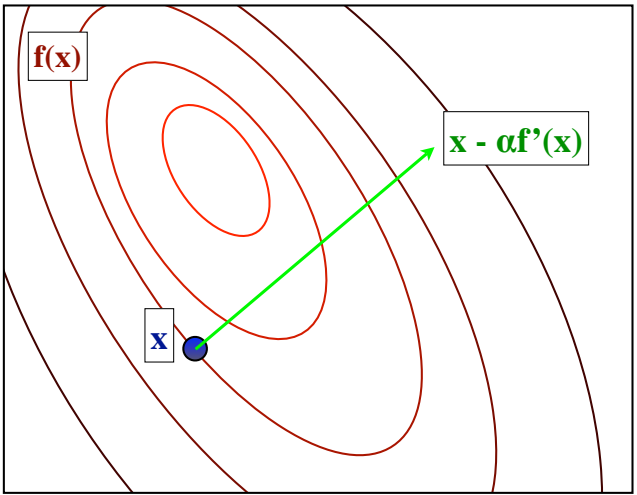
Newton's Method



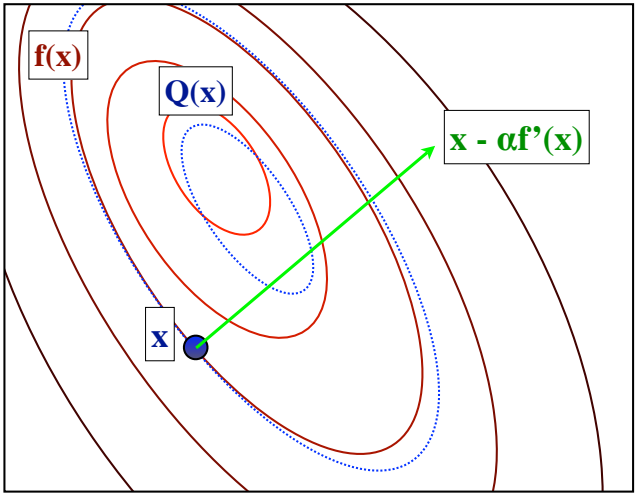
Newton's Method



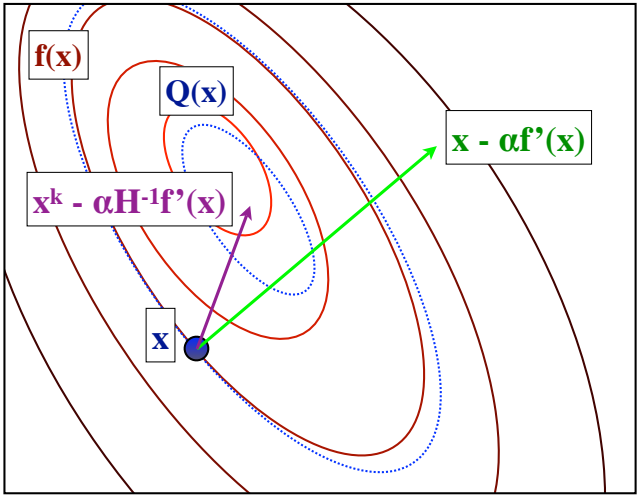
Newton's Method



Newton's Method



Newton's Method



Quadratic Convergence of Newton's Method

- Conisder a function with a Lipschitz-continuous Hessian,

$$\|\nabla^2 f(w) - \nabla^2 f(v)\| \leq M\|w - v\|,$$

  for some M and using the operator matrix norm (max singular value).

- If f is also μ-strongly convex then Newton's method has

$$\|w^{k+1} - w^*\| \leq \frac{M}{2\mu}\|w^k - w^*\|^2.$$

- If $\|w^k - w^*\|$ becomes sufficiently small, this implies quadratic convergence.
  - A form of superlinear convergence.

- Problem: there is no guarantee that Newton's method converges.
  - So $\|w^k - w^*\|$ may never become sufficiently small.

Damped Newton for Global Convergence

- Most common way to make Newton converge is to add a step size:

$$w^{k+1} = w^k - \alpha_k[\nabla^2 f(w^k)]^{-1}\nabla f(w^k),$$

sometimes called a damped Newton step.

- If gradient is Lipschitz and f is strongly-convex, then we obtain

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

by using damped Newton in the descent lemma with $\alpha_k = \mu/L$.

  - Notice that is a worse progress bound than with gradient descent.
  - We lose a factor of μ/L from using Newton instead of gradient direction.

- This leads to a global convergence rate of

$$f(w^{k+1}) - f^* \leq \left(1 - \frac{\mu^2}{L^2}\right)^k [f(w^0) - f^*],$$

depending on the squared condition number L/μ (opposite of acceleration).

Damped Newton with Armijo: Linear then Quadratic Rate

- We do not want to use $\alpha_k = \mu/L$, instead we might use the Armijo condition,

$$f(w^{k+1}) \leq f(w^k) - \gamma\alpha_k\langle\nabla f(w^k), d_k\rangle,$$

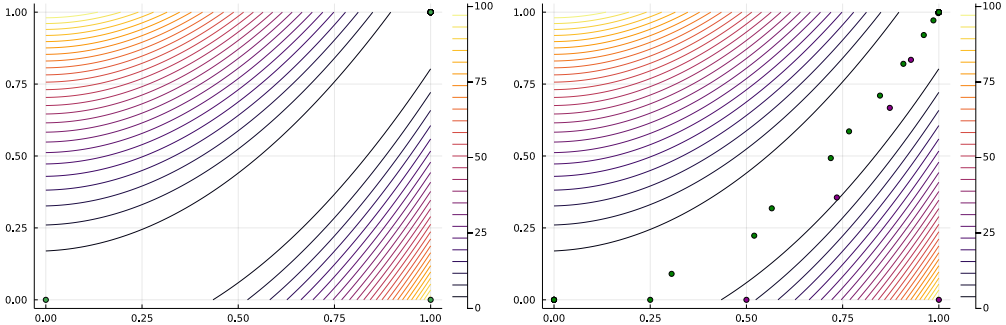
  where for the damped Newton the direction is $d_k = [\nabla^2 f(w^k)]^{-1}\nabla f(w^k)$.
  - This is a generalization of the earlier Armijo condition for a generic direction d_k.

- You can show $\alpha_k = 1$ satisfies the Armijo condition when close to solution.
  - So if we try $\alpha_k = 1$ first, we eventually have quadratic convergence.

- We can do a two-phase analysis of damped Newton for strongly convex f:
  - Far from solution, Lipschitz gradient guarantees slow linear rate.
    - Worse than gradient descent.
  - Close to solution, Lipschitz Hessian guarantees fast superlinear rate.
    - Assuming we eventually start trying $\alpha_k = 1$ first.

Example: Newton vs. Damped Newton

- Newton vs. damped Newton (with Armijo starting from $\alpha_k = 1$) on Rosenbrock:



- A step size of $\alpha_k = 1$ was used on most iterations.
  - But damped Newton takes around 16 iterations to reach machine precision.

Hessian Modification

- The Newton step "inverts" the Hessian, $\nabla^2 f(w^k)$.

- This causes problems if f is not strongly convex:
  - The Hessian matrix may be singular (no inverse exists).
  - The Newton direction may not be a descent direction.
    - The directional derivative might be 0 or positive, causing Armijo to fail.

- Common fix is to replace $\nabla^2 f(w^k)$ with a positive-definite approximation,

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

- For example, set $H^k = \nabla^2 f(w^k) + \lambda^k I$.
  - Where λ^k is set so that eigenvalues of H^k are at least some positive ϵ.
- More sophisticated approaches try to minimally modify Cholesky of $\nabla^2 f(w)$.
  - Works better than the λ^k approach, but still not ideal for non-convex.

Trust-Region Methods

- For a constant α_k and positive-definite approximation H^k, damped Newton is

$$w^{k+1} \in \underset{w \in \mathbb{R}^d}{\text{argmin}} \left\{ f(w^k) + \nabla f(w^k)^T (w - w^k) + \frac{1}{2\alpha_k} (w - w^k)^T H^k (w - w^k) \right\},$$

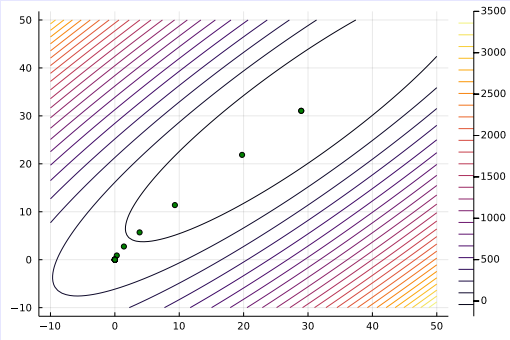
- For a constant Δ_k trust region methods instead try to compute

$$w^{k+1} \in \underset{w \mid \|w - w^k\| \leq \Delta_k}{\text{argmin}} \left\{ f(w^k) + \nabla f(w^k)^T (w - w^k) + \frac{1}{2} (w - w^k)^T \nabla^2 f(w^k) (w - w^k) \right\}$$

- The number Δ_k is called the trust region radius.
  - Intuitively, it is how far we "trust" the truncated Taylor series.
  - Radius is grown/shrunk by comparing expected progress to actual progress.
    - For example, you shrink it you are making less progress than expected.

Trust Region on Quadratic

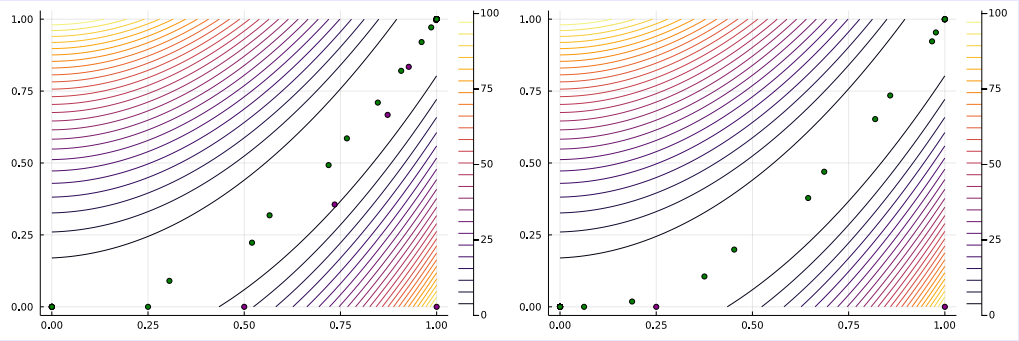
- Newton with trust region on quadratic starting from $\Delta_k = 1$.



- Does not converge in one step because trust region was too small.
  - Converges in one step once trust region contains Newton step.

Example: Line Search vs. Trust Region

- Armijo line-search (starting from $\alpha_k = 1$) vs. trust region (starting from $\Delta_k = 1$):



- δ_k shrinks to $1/8$ on first iteration, doubles on next two, then stays at $1/4$.
  - On this problem both methods take a similar number of iterations.

Cubic Regularization of Newton's Method

- Gradient descent ($\alpha_k = 1/L$) uses upper-bound on second-order term,

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

- Cubic regularization of Newton's method upper bounds third-order term,

$$f(w) \leq f(w^k) + \nabla f(w^k)^T(w^k - w) + \frac{1}{2}(w - w^k)\nabla^2 f(w^k)(w - w^k) + \frac{M}{6}\|w - w^k\|^3,$$

  where M is the Lipschitz constant of the Hessian.

- Minimizing this upper bound leads to guaranteed progress.
  - Similar to gradient descent with $\alpha_k = 1/L$.
  - Bound is tighter for small $\|w - w^k\|$, looser for large $\|w - w^k\|$.

- Can/should use backtracking to replace M by an approximation $\hat{M}$.
- There exist accelerated variants that achieve faster rates.
  - Accelerated method has error of $O(1/k^3)$ for convex functions.

Trust-Region and Cubic Regularization for Non-Convex

- Trust-region and cubic regularization may be better for non-convex problems.
  - These methods may move along directions of negative curvature.
    - These are directions that speed up progress.
    - Leads to "escaping" saddle points rather than converging to them.
- Trust-region and cubic regularization methods are more difficult to implement.
  - Computing the update is more complicated than solving a linear system.
  - But practical methods exist to compute steps guaranteeing sufficient progress.
    - Many have similar cost to solving a linear system.

Summary

- First-order oracle model of computation.
  - How many queries to an oracle returning function and gradient value?
- Momentum and heavy-ball method.
  - Optimal convergence rate for optimizing strongly-convex quadratics.
- Accelerated gradient method.
  - Near-optimal convergence rates for optimizing convex and strongly-convex functions.
  - Resetting strategies to avoid needing to know μ.
- Newton's method
  - Second-order method with local quadratic convergence.
  - Global convergence with line-search, trust-region, or cubic regularization.

- Next time: decreasing iteration cost instead of iteration complexity (SGD).

Conjugate Gradient Derivation and Implementation

- We said that conjugate gradient is an optimal heavy-ball method.
- But it is usually derived/analyzed as optimizing a growing set of subspaces.
  - In particular, you can show it optimizes over span of all previous gradients.
- CG is usually written/implemented/motivated in a different way:
  - Phrased as solving linear system ($Aw = -b$) for a positive-definite matrix A.
  - Using $g_k = Aw^k + b$ and $d_0 = -g_0$, the iteration can be written:

$$\alpha_k = -\frac{g_k^T d_k}{d_k^T A d_k}$$
$$w^{k+1} = w^k + \alpha_k d_k$$
$$\beta_k = \frac{g_{k+1}^T A d_k}{d_k^T A d_k}$$
$$d_{k+1} = -g_{k+1} + \beta_k d_k$$

  - Note that β_k above is not the β_k used in heavy-ball way of writing.
    - Momentum direction d_k is multiplied by both α_k and β_{k-1}.

Affine Invariance and Self-Concordance

- Newton's method is an affine invariant method:
  - Consider applying Newton to $f(w)$ and $g = f(Aw)$ for an invertible matrix A.
  - The method generates the same sequence of iterations, up to transformation by A.

- Despite affine invariance, iteration complexity depends on L, μ, and M.
  - These are changed by the above re-parameterization.
  - We could theoretically search for the A giving the "best" L, μ, and M.

- We have affine-invariant analyses for self-concordant functions.
  - Self-concordance bounds third derivative in terms of second derivative.
  - Iteration complexity of damped Newton that only depends on Armijo parameters.
    - No dependence on condition number.
    - We pick the Armijo parameters (can be the same across problems).