CS 287 Advanced Robotics (Fall 2019)
Lecture 6: Unconstrained Optimization

Pieter Abbeel
UC Berkeley EECS

Many slides and figures adapted from Stephen Boyd

[optional] Boyd and Vandenberghe, Convex Optimization, Chapters 9 – 11

[optional] Betts, Practical Methods for Optimal Control Using Nonlinear Programming

Bellman's Curse of Dimensionality

- n-dimensional state space
- Number of states grows exponentially in n (for fixed number of discretization levels per coordinate)
- In practice
  - Discretization is considered only computationally feasible up to 5 or 6 dimensional state spaces even when using
    - Variable resolution discretization
    - Highly optimized implementations

Optimization for Optimal Control

- Goal: find a sequence of control inputs (and corresponding sequence of states) that solves:

$$
\begin{aligned}
\min_{u,x} \quad & \sum_{t=0}^{H} g(x_t, u_t) \\
\text{subject to} \quad & x_{t+1} = f(x_t, u_t) \quad \forall t \\
& u_t \in \mathcal{U}_t \quad \forall t \\
& x_t \in \mathcal{X}_t \quad \forall t
\end{aligned}
$$

- Generally hard to do. Exception: convex problems, which means g is convex, the sets U_t and X_t are convex, and f is linear.

- Note: iteratively applying LQR is one way to solve this problem but can get a bit tricky when there are constraints on the control inputs and state.

- In principle (though not in our examples), u could be parameters of a control policy rather than the raw control inputs.

Outline

- **Convex optimization problems**
- Unconstrained minimization
  - Gradient Descent
  - Newton’s Method
  - Natural Gradient / Gauss-Newton
  - Momentum, RMSprop, Aam

Convex Functions

- A function f is convex if and only if

$$\forall x_1, x_2 \in \mathrm{Domain}(f), \forall t \in [0,1]:$$
$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

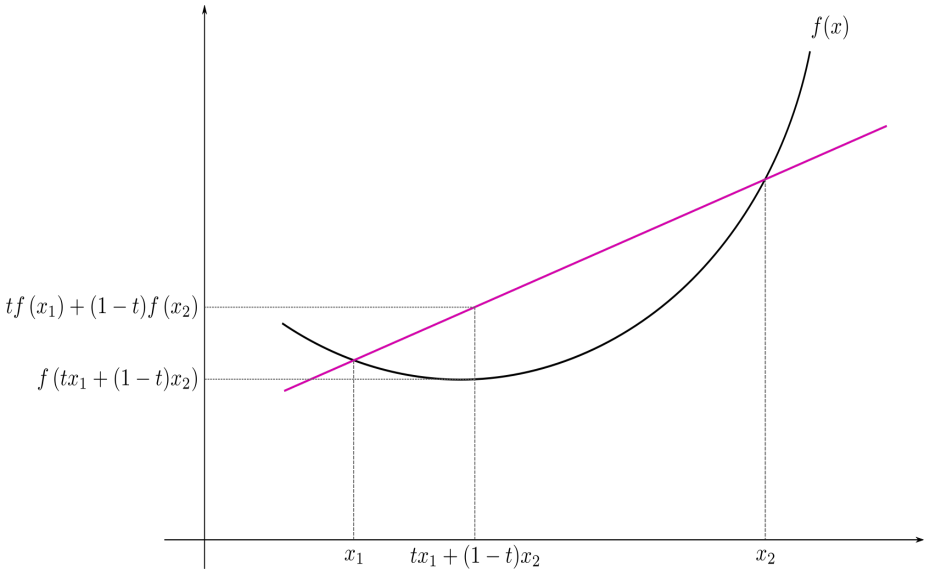
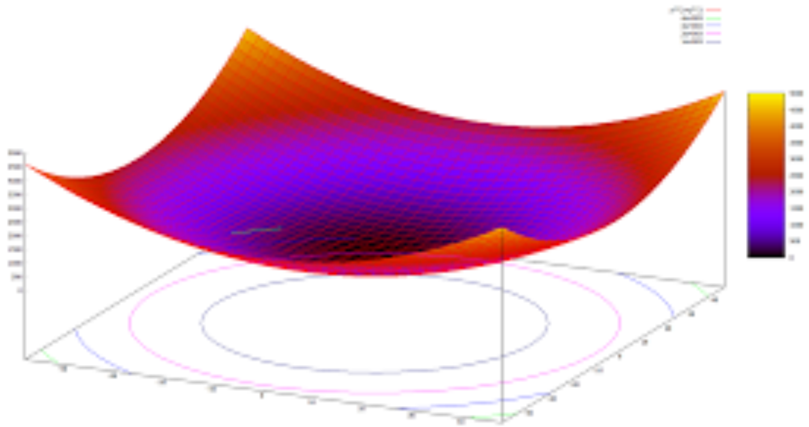


Image source: wikipedia

Convex Functions



- Unique minimum
- Set of points for which f(x) <= a is convex

Source: Thomas Jungblut’s Blog

Convex Optimization Problems

- Convex optimization problems are a special class of optimization problems, of the following form:

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} f_0(x) \\
\text{s.t.} \quad & f_i(x) \leq 0 \qquad i = 1, \ldots, n \\
& Ax = b
\end{aligned}
$$

  with $f_i(x)$ convex for i = 0, 1, …, n

- A function f is convex if and only if

$$
\begin{aligned}
&\forall x_1, x_2 \in \text{Domain}(f), \forall \lambda \in [0, 1] \\
&f(\lambda x_1 + (1-\lambda) x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)
\end{aligned}
$$

Outline

- Convex optimization problems
- ***Unconstrained minimization***
  - ***Gradient Descent***
  - Newton's Method
  - Natural Gradient / Gauss-Newton
  - Momentum, RMSprop, Aam

Unconstrained Minimization

$$\boxed{\min_x f(x) \qquad (1)}$$

(Implicitly assumed x can be chosen from the entire domain of f, often $\mathbb{R}^n$.)

- x* is a local minimum of (differentiable) f than it has to satisfy:

$$\begin{aligned} \nabla_x f(x^*) &= 0 \quad (2) \\ \nabla_x^2 f(x^*) &\succeq 0 \quad (3) \end{aligned}$$

- In simple cases we can directly solve the system of n equations given by (2) to find candidate local minima, and then verify (3) for these candidates.
- In general however, solving (2) is a difficult problem. Going forward we will consider this more general setting and cover numerical solution methods for (1).

Steepest Descent

- Idea:
  - Start somewhere
  - Repeat: Take a step in the steepest descent direction

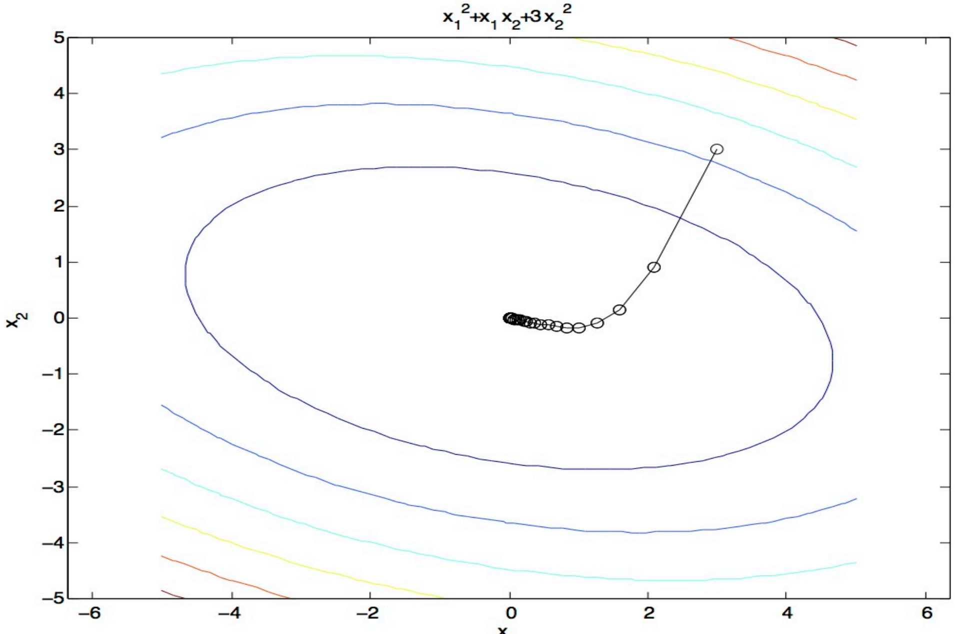


Figure source: Mathworks

Steepest Descent Algorithm

1. Initialize x

2. Repeat

   1. Determine the steepest descent direction Δx

   2. Line search: Choose a step size $t > 0$.

   3. Update: $x := x + t\, \Delta x$.

3. Until stopping criterion is satisfied

What is the Steepest Descent Direction?

Assuming a smooth function, we have that

$$f(x_0 + \Delta x) \approx f(x_0) + \nabla_x f(x_0)^\top \Delta x$$

The (locally at x_0) direction of steepest descent is given by:

$$\begin{aligned} \Delta x^* &= \arg \min_{\Delta x : \|\Delta x\|_2 = 1} f(x_0) + \nabla_x f(x_0)^\top \Delta x \\ &= \arg \min_{\Delta x : \|\Delta x\|_2 = 1} \nabla_x f(x_0)^\top \Delta x \end{aligned}$$

As we have all $a, b \in \mathbb{R}^n$ that $\min_{b:\|b\|_2=1} a^\top b$ is achieved for $b = -\frac{a}{\|a\|_2}$, we have that the steepest descent direction

$$\Delta x^* = -\nabla_x f(x_0)$$

→ Steepest Descent = Gradient Descent

Stepsize Selection: Exact Line Search

$$t = \arg\min_{s \geq 0} f(x + s\Delta x)$$

Used when the cost of solving the minimization problem with one variable is low compared to the cost of computing the search direction itself.

Stepsize Selection: Backtracking Line Search

- Inexact: step length is chose to approximately minimize f along the ray {x + t Δx | t > 0}

Backtracking Line Search.
given a descent direction Δx for f at $x \in \mathbf{dom} f$, $\alpha \in (0, 0.5), \beta \in (0, 1)$.
$t := 1$
while $f(x + t\Delta x) > f(x) + \alpha t \nabla f(x)^\top \Delta x, t := \beta t$.

Stepsize Selection: Backtracking Line Search

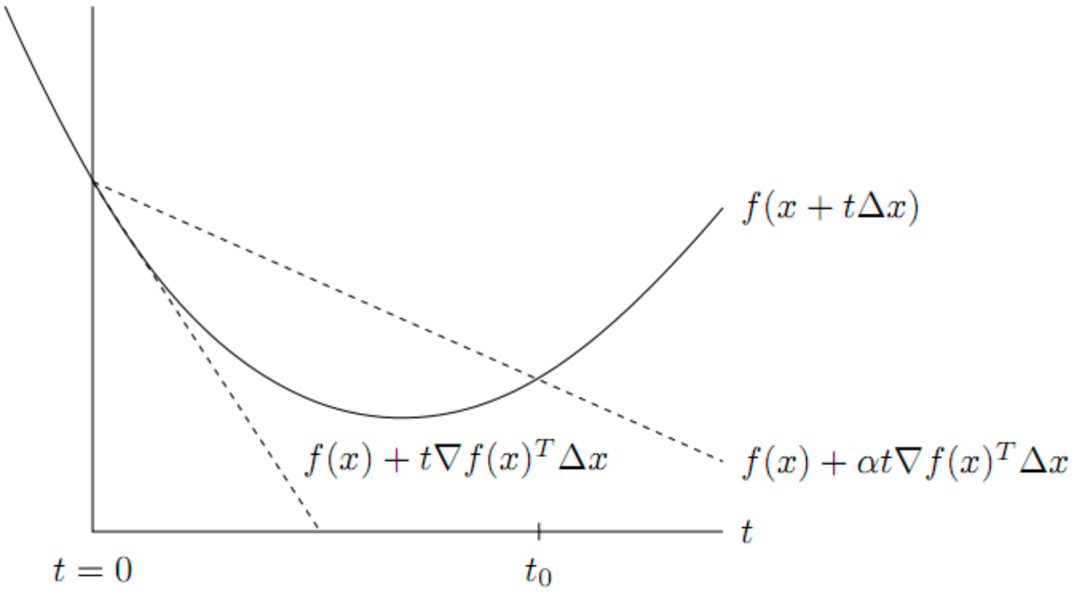


Figure 9.1 *Backtracking line search.* The curve shows f, restricted to the line over which we search. The lower dashed line shows the linear extrapolation of f, and the upper dashed line has a slope a factor of α smaller. The backtracking condition is that f lies below the upper dashed line, *i.e.*, $0 \le t \le t_0$.

Figure source: Boyd and Vandenberghe

Steepest Descent (= Gradient Descent)

Algorithm 9.3 *Gradient descent method.*

given a starting point $x \in \textbf{dom}\, f$.

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

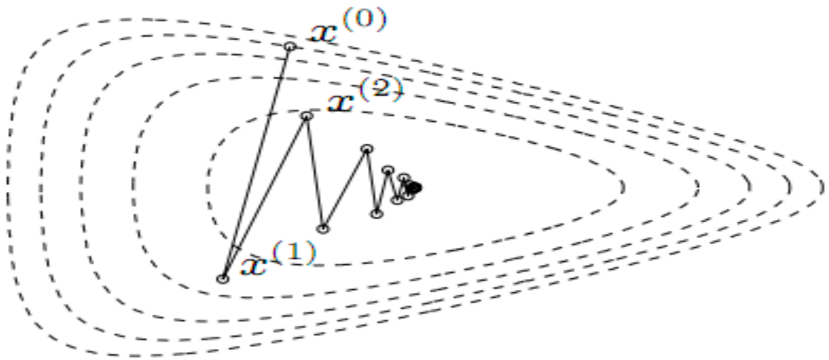
until stopping criterion is satisfied.

The stopping criterion is usually of the form $\|\nabla f(x)\|_2 \leq \eta$, where η is small and positive. In most implementations, this condition is checked after step 1, rather than after the update.

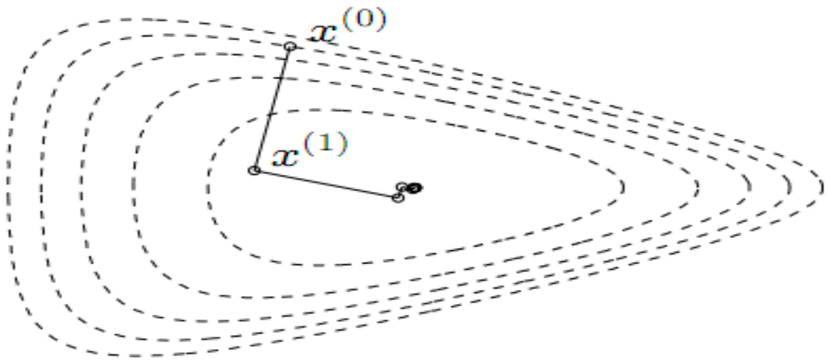
Source: Boyd and Vandenberghe

Gradient Descent: Example 1

$$f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$



backtracking line search



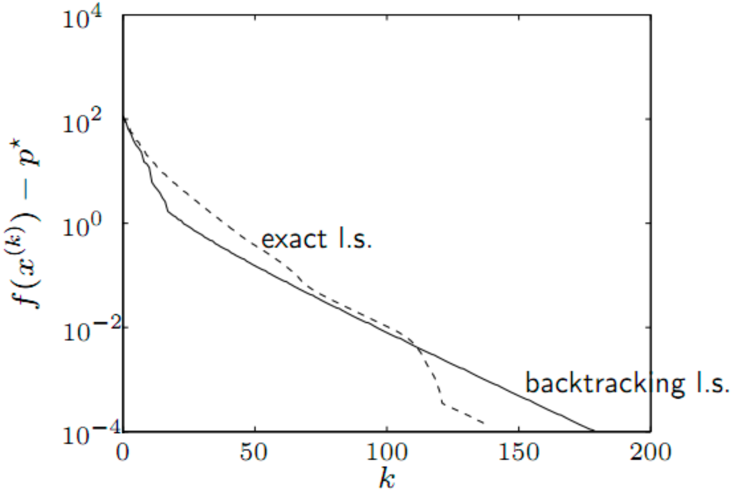
exact line search

Figure source: Boyd and Vandenberghe

Gradient Descent: Example 2

a problem in $\mathbf{R}^{100}$

$$f(x) = c^T x - \sum_{i=1}^{500} \log(b_i - a_i^T x)$$



'linear' convergence, *i.e.*, a straight line on a semilog plot

Figure source: Boyd and Vandenberghe

Gradient Descent: Example 3

$$f(x) = (1/2)(x_1^2 + \gamma x_2^2) \qquad (\gamma > 0)$$

with exact line search, starting at $x^{(0)} = (\gamma, 1)$:

$$x_1^{(k)} = \gamma \left(\frac{\gamma - 1}{\gamma + 1}\right)^k, \qquad x_2^{(k)} = \left(-\frac{\gamma - 1}{\gamma + 1}\right)^k$$

- very slow if $\gamma \gg 1$ or $\gamma \ll 1$
- example for $\gamma = 10$:

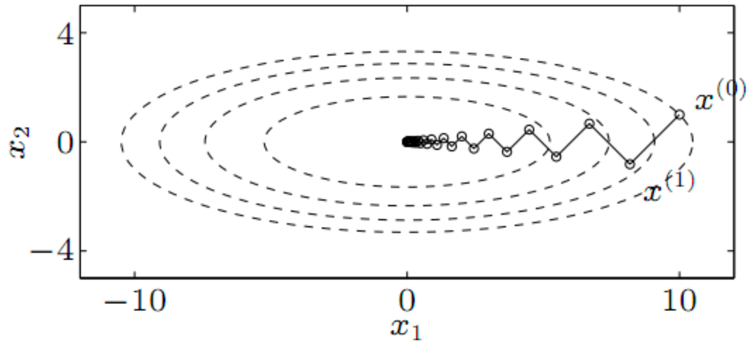
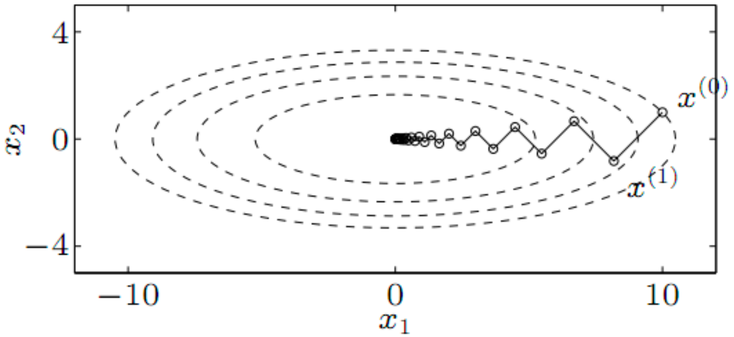
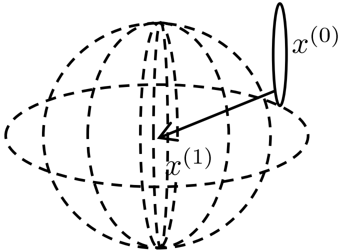


Figure source: Boyd and Vandenberghe

Gradient Descent Convergence



Condition number = 10



Condition number = 1

- For quadratic function, convergence speed depends on ratio of highest second derivative over lowest second derivative ("condition number")
- In high dimensions, almost guaranteed to have a high (=bad) condition number
- Rescaling coordinates (as could happen by simply expressing quantities in different measurement units) results in a different condition number

Outline

- Convex optimization problems
- ***Unconstrained minimization***
  - Gradient Descent
  - ***Newton's Method***
  - Natural Gradient / Gauss-Newton
  - Momentum, RMSprop, Aam

Newton's Method

- 2^{nd} order Taylor Approximation rather than 1^{st} order:

$$f(x+\Delta x) \approx f(x) + \nabla f(x)^\top \Delta x + \frac{1}{2}\Delta x^\top \nabla^2 f(x) \Delta x$$

assuming $\nabla^2 f(x) \succeq 0$ (which is true for convex f) the minimum of the 2^{nd} order approximation is achieved at:

$$\Delta x_{\rm nt} = -\left(\nabla^2 f(x)\right)^{-1} \nabla f(x)$$

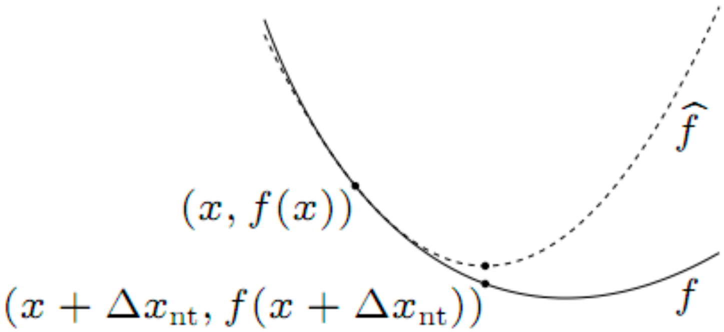


Figure source: Boyd and Vandenberghe

Newton's Method

Algorithm 9.5 *Newton's method.*

given a starting point $x \in \mathbf{dom}\, f$, tolerance $\epsilon > 0$.

repeat

1. *Compute the Newton step and decrement.*
   $\Delta x_{\mathrm{nt}} := -\nabla^2 f(x)^{-1} \nabla f(x); \quad \lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x).$
2. *Stopping criterion.* **quit** if $\lambda^2/2 \leq \epsilon$.
3. *Line search.* Choose step size t by backtracking line search.
4. *Update.* $x := x + t\Delta x_{\mathrm{nt}}$.

Figure source: Boyd and Vandenberghe

Affine Invariance

- Consider the coordinate transformation $y = A^{-1} x \quad (x = Ay)$
- If running Newton's method starting from $x^{(0)}$ on $f(x)$ results in

  $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$

- Then running Newton's method starting from $y^{(0)} = A^{-1} x^{(0)}$ on $g(y) = f(Ay)$, will result in the sequence

  $y^{(0)} = A^{-1} x^{(0)}, y^{(1)} = A^{-1} x^{(1)}, y^{(2)} = A^{-1} x^{(2)}, \ldots$

Exercise: try to prove this!

Affine Invariance --- Proof

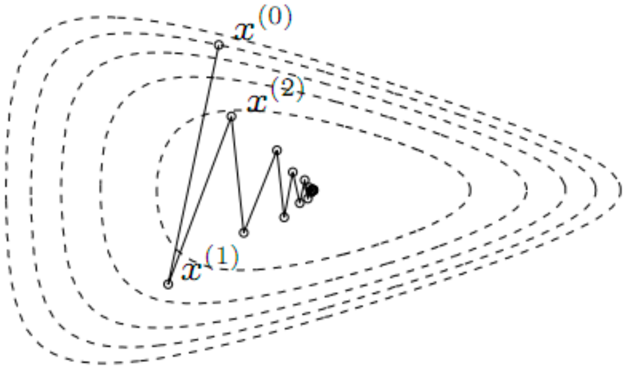
$$
\begin{aligned}
\frac{\partial g}{\partial y_i} &= \sum_j \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial y_i} \\
&= \sum_j \frac{\partial f}{\partial x_j} A_{ji} \\
&= (A^\top)_{i,:} \nabla f \\
\nabla g &= A^\top \nabla f
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial^2 g}{\partial y_k \partial y_i} &= \frac{\partial}{\partial y_i} \left(\sum_j \frac{\partial f}{\partial x_j} A_{j,i} \right) \\
&= \sum_j \frac{\partial}{\partial y_k} \left(\frac{\partial f}{\partial x_j} \right) A_{j,i} \\
&= \sum_j \sum_l \frac{\partial^2 f}{\partial x_l \partial x_j} \frac{\partial x_l}{\partial y_k} A_{j,i} \\
&= \sum_j \sum_l \frac{\partial^2 f}{\partial x_l \partial x_j} A_{l,k} A_{j,i} \\
\nabla^2 g &= A^\top \nabla^2 f A
\end{aligned}
$$

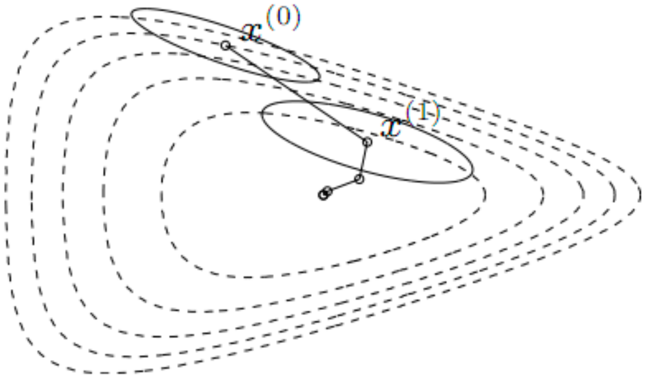
$$
\begin{aligned}
\Delta y &= -\left(\nabla^2 g\right)^{-1} \nabla g \\
&= -\left(A^\top \nabla^2 f A\right)^{-1} A^\top \nabla f \\
&= -A^{-1} \left(\nabla^2 f\right)^{-1} A^{-\top} A^\top \nabla f \\
&= -A^{-1} \left(\nabla^2 f\right)^{-1} \nabla f \\
&= A^{-1} \Delta x
\end{aligned}
$$

Example 1

$$f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$



gradient descent with
backtracking line search



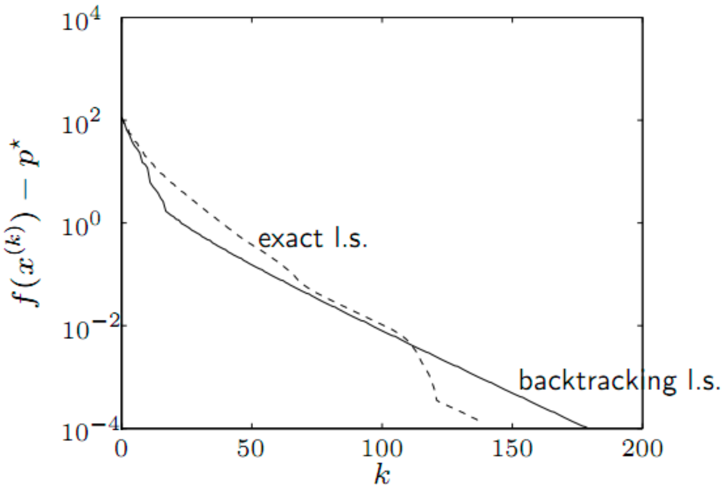
Newton's method with
backtracking line search

Figure source: Boyd and Vandenberghe

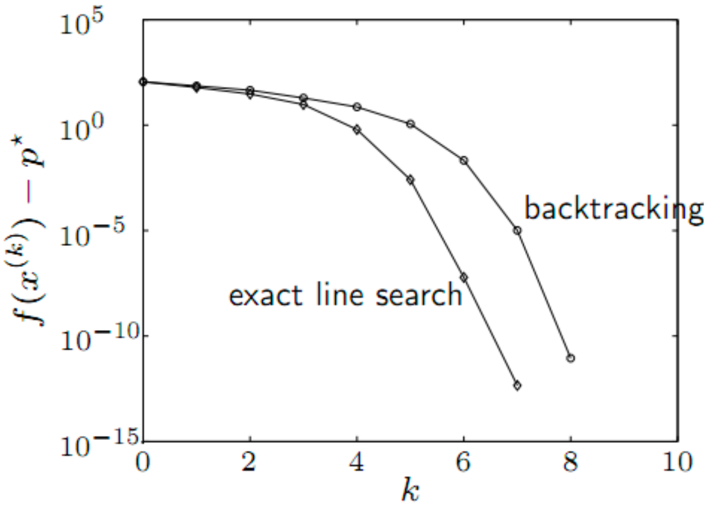
Example 2

a problem in $\mathbf{R}^{100}$

$$f(x) = c^T x - \sum_{i=1}^{500} \log(b_i - a_i^T x)$$



gradient descent



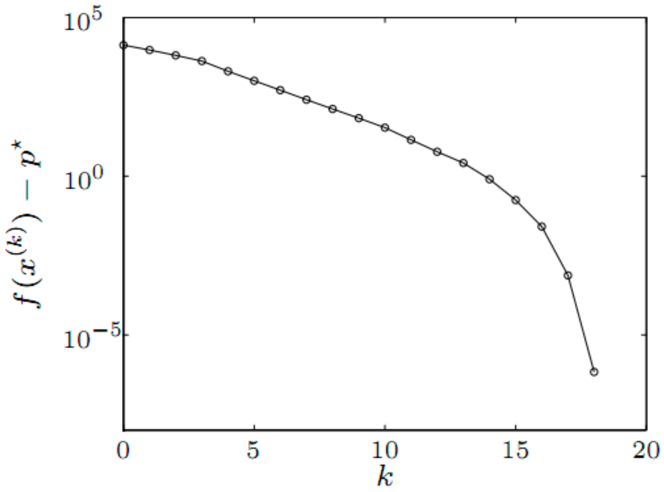
Newton's method

Figure source: Boyd and Vandenberghe

Larger Version of Example 2

example in $\mathbf{R}^{10000}$ (with sparse a_i)

$$f(x) = -\sum_{i=1}^{10000} \log(1 - x_i^2) - \sum_{i=1}^{100000} \log(b_i - a_i^T x)$$



- backtracking parameters $\alpha = 0.01$, $\beta = 0.5$.
- performance similar as for small examples

Figure source: Boyd and Vandenberghe

Gradient Descent: Example 3

$$f(x) = (1/2)(x_1^2 + \gamma x_2^2) \qquad (\gamma > 0)$$

with exact line search, starting at $x^{(0)} = (\gamma, 1)$:

$$x_1^{(k)} = \gamma \left(\frac{\gamma - 1}{\gamma + 1}\right)^k, \qquad x_2^{(k)} = \left(-\frac{\gamma - 1}{\gamma + 1}\right)^k$$

- very slow if $\gamma \gg 1$ or $\gamma \ll 1$
- example for $\gamma = 10$:

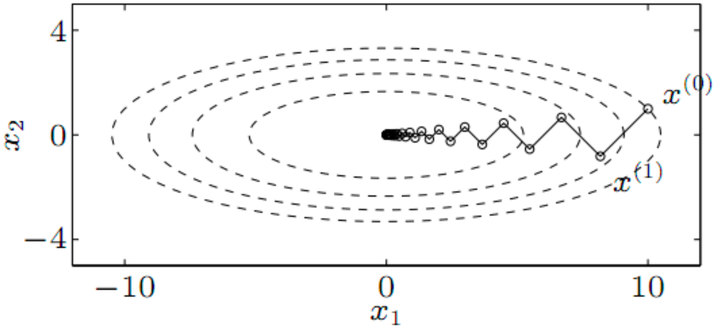
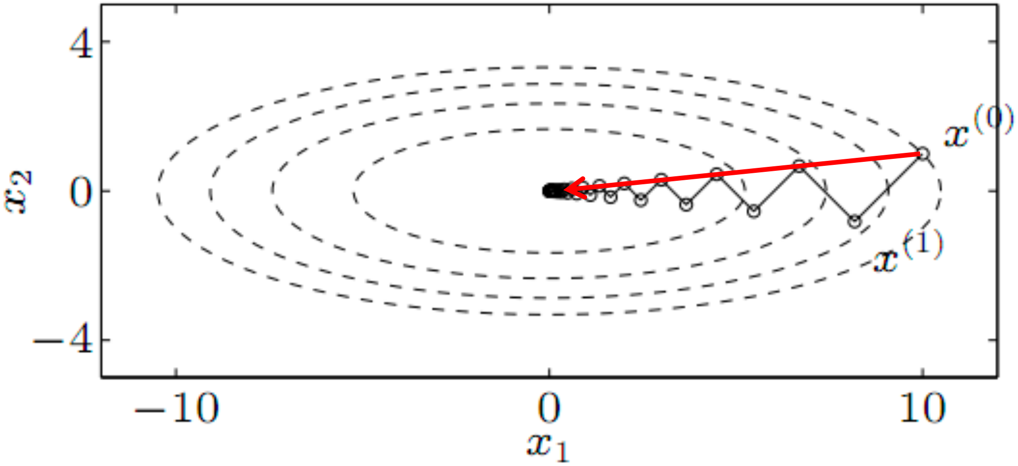


Figure source: Boyd and Vandenberghe

Example 3



- Gradient descent
- Newton's method (converges in one step if f convex quadratic)

Quasi-Newton Methods

- Quasi-Newton methods use an approximation of the Hessian
  - Example 1: Only compute diagonal entries of Hessian, set others equal to zero. Note this also simplifies computations done with the Hessian.
  - Example 2: Natural gradient --- see next slide

Outline

- Convex optimization problems
- ***Unconstrained minimization***
  - Gradient Descent
  - Newton's Method
  - ***Natural Gradient / Gauss-Newton***
  - Momentum, RMSprop, Aam

Natural Gradient

- Consider a standard maximum likelihood problem: $$\max_\theta f(\theta) = \max_\theta \sum_i \log p(x^{(i)};\theta)$$

- Gradient: $$\frac{\partial f(\theta)}{\partial \theta_p} = \sum_i \frac{\partial \log p(x^{(i)};\theta)}{\partial \theta_p} = \sum_i \frac{\partial p(x^{(i)};\theta)}{\partial \theta_p} \frac{1}{p(x^{(i)};\theta)}$$

- Hessian: $$\frac{\partial^2 f(\theta)}{\partial \theta_q \partial \theta_p} = \sum_i \frac{\partial^2 p(x^{(i)};\theta)}{\partial \theta_q \partial \theta_p} \frac{1}{p(x^{(i)};\theta)} - \frac{\partial p(x^{(i)};\theta)}{\partial \theta_q} \frac{1}{p(x^{(i)};\theta)} \frac{\partial p(x^{(i)};\theta)}{\partial \theta_p} \frac{1}{p(x^{(i)};\theta)}$$

$$\nabla^2 f(\theta) = \sum_i \frac{\nabla^2 p(x^{(i)};\theta)}{p(x^{(i)};\theta)} - \left(\nabla \log p(x^{(i)};\theta)\right)\left(\nabla \log p(x^{(i)};\theta)\right)^\top$$

- Natural gradient: $$= \left(\sum_i \left(\nabla \log p(x^{(i)};\theta)\right)\left(\nabla \log p(x^{(i)};\theta)\right)^\top\right)^{-1} \left(\sum_i \nabla \log p(x^{(i)};\theta)\right)$$

only keeps the 2nd term in the Hessian. Benefits: (1) faster to compute (only gradients needed); (2) guaranteed to be negative definite; (3) found to be superior in some experiments; (4) invariant to re-parameterization

Natural Gradient

- Property: Natural gradient is invariant to parameterization of the family of probability distributions $p(x ; \theta)$

- Hence the name.

- Note this property is stronger than the property of Newton's method, which is invariant to affine re-parameterizations only.

- Exercise: Try to prove this property!

Natural Gradient Invariant to Reparametrization --- Proof

- Natural gradient for parametrization with θ:

$$\bar{g}_\theta = \left(\sum_i \left(\nabla_\theta \log p(x^{(i)}; \theta) \right) \left(\nabla_\theta \log p(x^{(i)}; \theta) \right)^\top \right)^{-1} \left(\sum_i \nabla_\theta \log p(x^{(i)}; \theta) \right)$$

- Let Φ = f(θ), and let $J = \frac{\partial \theta}{\partial \phi}$ i.e., $J_{i,j} = \frac{\partial \theta_i}{\partial \phi_j}$

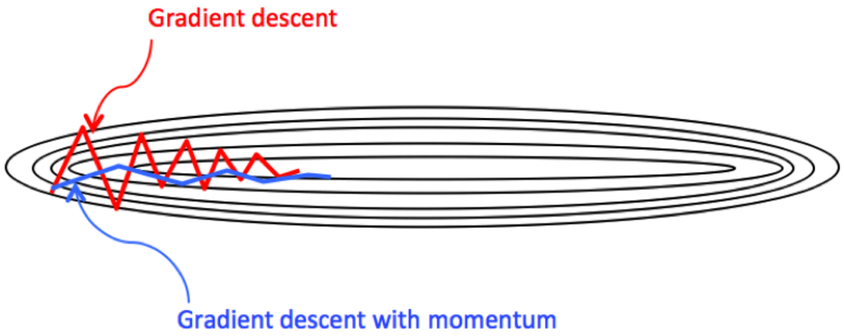
$$\begin{aligned}
\bar{g}_\phi &= \left(\sum_i \left(\nabla_\phi \log p(x^{(i)}; \phi) \right) \left(\nabla_\phi \log p(x^{(i)}; \phi) \right)^\top \right)^{-1} \left(\sum_i \nabla_\phi \log p(x^{(i)}; \phi) \right) \\
&= \left(\sum_i \left(J^\top \nabla_\theta \log p(x^{(i)}; \phi) \right) \left(J^\top \nabla_\theta \log p(x^{(i)}; \phi) \right)^\top \right)^{-1} \left(J^\top \sum_i \nabla_\theta \log p(x^{(i)}; \phi) \right) \\
&= J^\top \bar{g}_\theta
\end{aligned}$$

→ the natural gradient direction is the same independent of the (invertible, but otherwise not constrained) reparametrization f

Outline

- Convex optimization problems
- ***Unconstrained minimization***
  - Gradient Descent
  - Newton’s Method
  - Natural Gradient / Gauss-Newton
  - ***Momentum, RMSprop, Aam***

Gradient Descent with Momentum



Gradient Descent

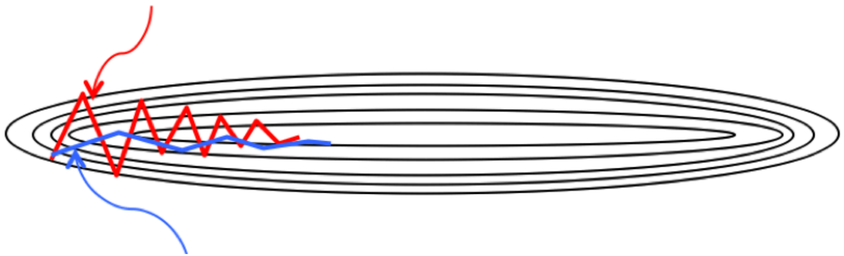
$$x \leftarrow x - \alpha \nabla_x f(x)$$

Gradient Descent with Momentum

$$v \leftarrow \beta v + (1 - \beta) \nabla_x f(x)$$
$$x \leftarrow \alpha v$$

Typically beta = 0.9
v = exponentially weighted avg of gradient

RMSprop



Gradient Descent

$$x \leftarrow x - \alpha \nabla_x f(x)$$

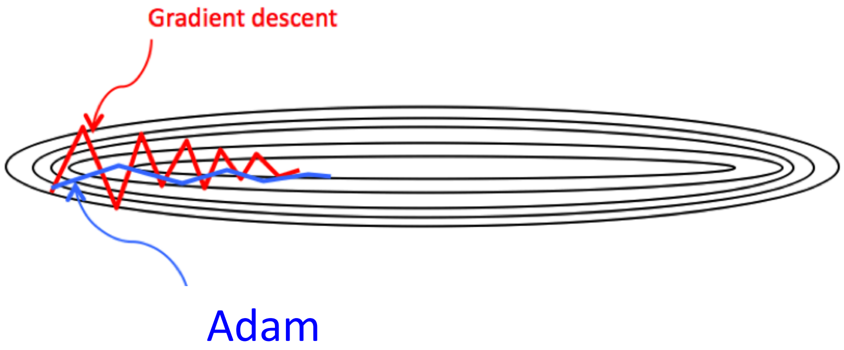
RMSprop (Root Mean Square propagation)

$$s \quad \leftarrow \beta s + (1-\beta)(\nabla_x f(x))^2$$
$$x \quad \leftarrow \alpha \nabla_x f(x) / (\sqrt{s + \epsilon})$$

Typically beta = 0.999

s = exponentially weighted avg of squared gradients

Adam



Gradient Descent

$$x \leftarrow x - \alpha \nabla_x f(x)$$

Adam (Adaptive momentum estimation)

$$v \leftarrow \left(\beta_1 v + (1-\beta_1)\nabla_x f(x)\right) / (1-\beta_1^t)$$

$$s \leftarrow \left(\beta_2 s + (1-\beta_2)(\nabla_x f(x))^2\right) / (1-\beta_2^t)$$

$$x \leftarrow \alpha v / (\sqrt{s+\epsilon})$$

Typically beta1= 0.9; beta2=0.999; eps=1e-8
s = exponentially weighted avg of squared gradients
v= momentum