Gradient Descent

Lecturer: Ryan Tibshirani
Convex Optimization 10-725/36-725

Last time: canonical convex programs

- Linear program (LP): takes the form

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{subject to} \quad & Dx \leq d \\ & Ax = b \end{aligned}$$

- Quadratic program (QP): like an LP, but with a quadratic criterion
- Semidefinite program (SDP): like an LP, but with matrices
- Conic program: the most general form of all

Gradient descent

Consider unconstrained, smooth convex optimization

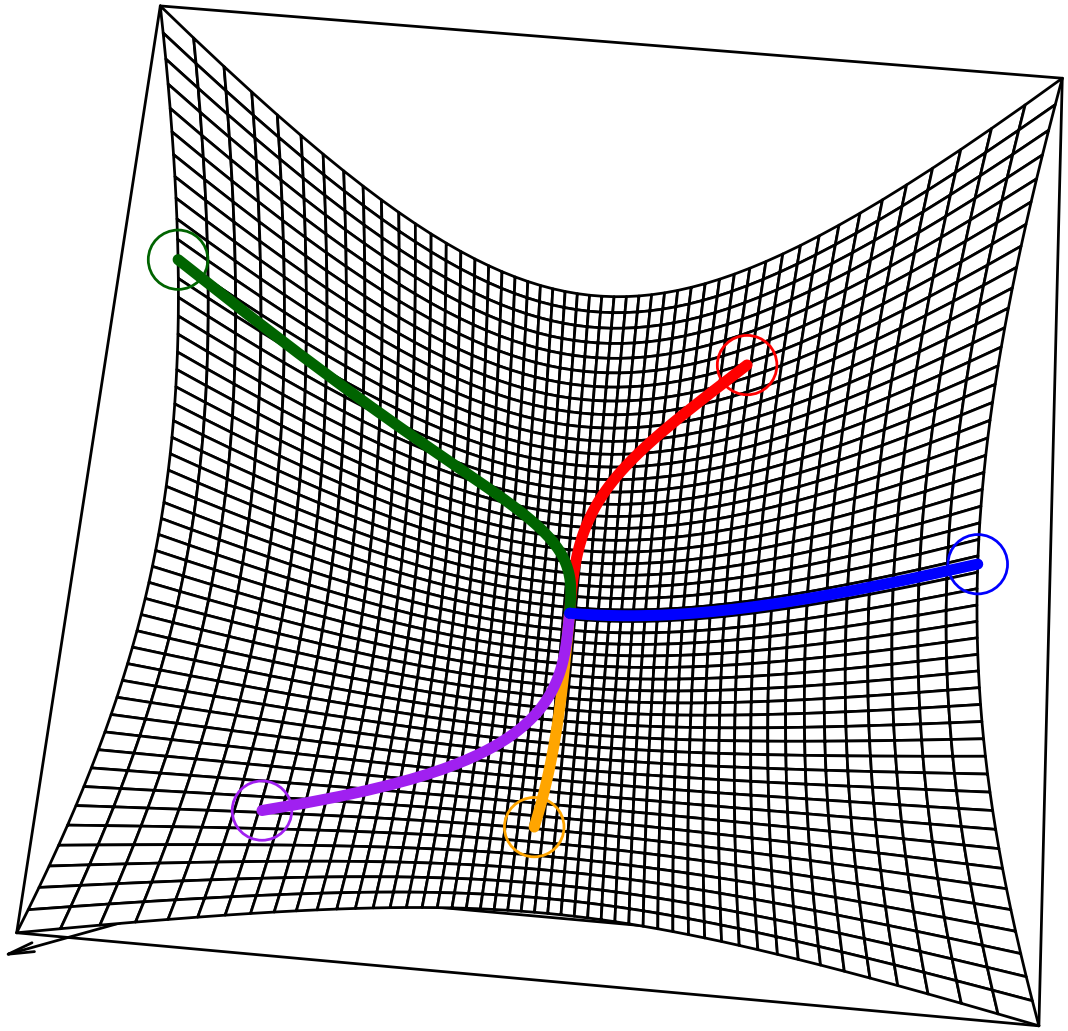
$$\min_x \ f(x)$$

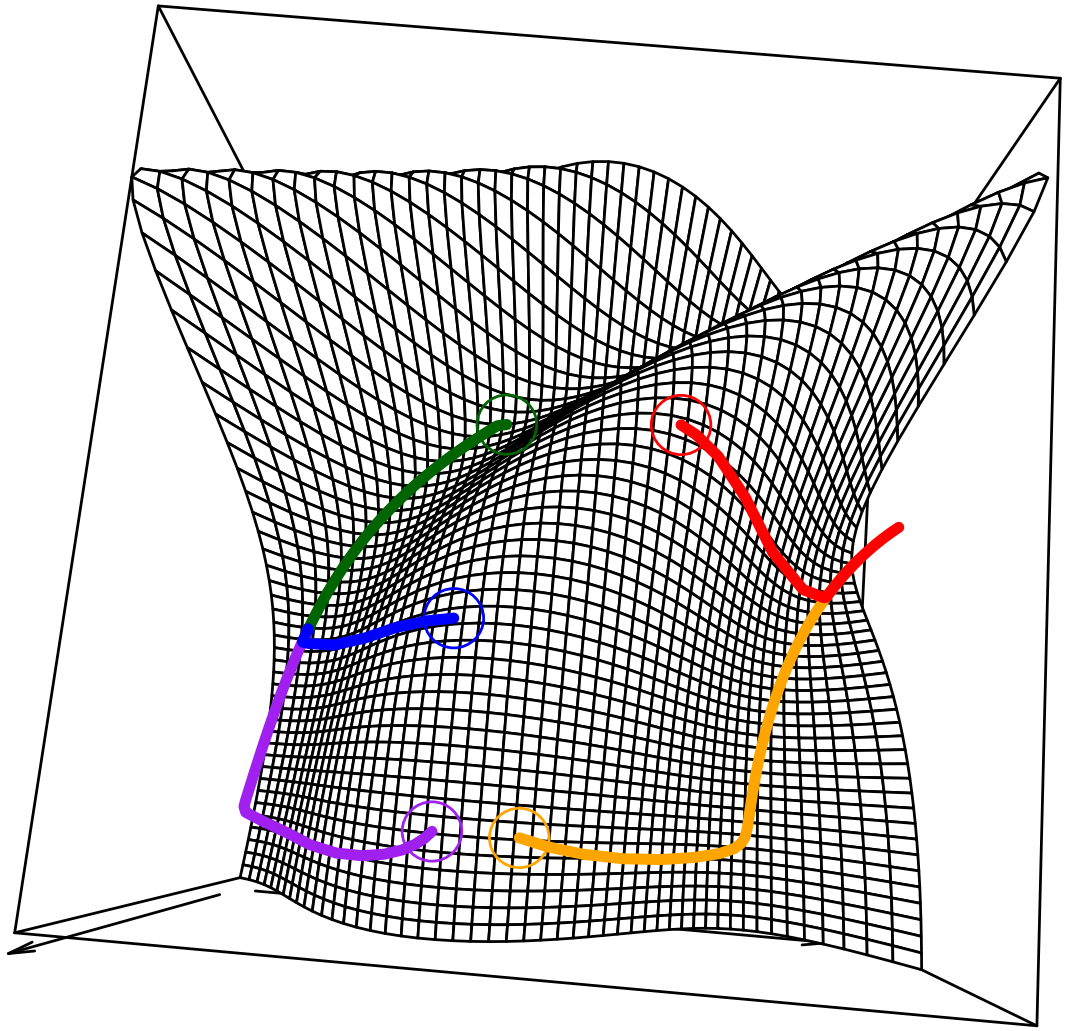
i.e., f is convex and differentiable with $\mathrm{dom}(f) = \mathbb{R}^n$. Denote the optimal criterion value by $f^\star = \min_x \ f(x)$, and a solution by $x^\star$

Gradient descent: choose initial point $x^{(0)} \in \mathbb{R}^n$, repeat:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

Stop at some point





Gradient descent interpretation

At each iteration, consider the expansion

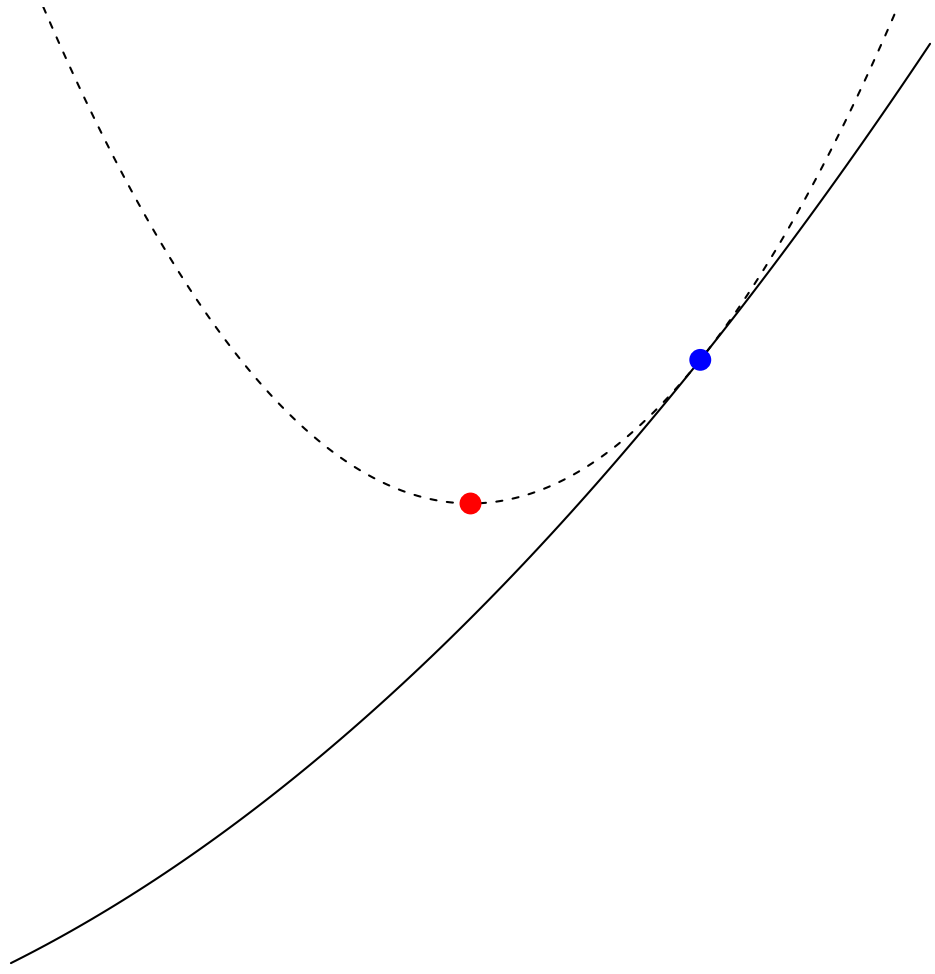
$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2t}\|y - x\|_2^2$$

Quadratic approximation, replacing usual Hessian $\nabla^2 f(x)$ by $\frac{1}{t} I$

$f(x) + \nabla f(x)^T (y - x)$	linear approximation to f
$\frac{1}{2t}\|y - x\|_2^2$	proximity term to x, with weight $1/(2t)$

Choose next point $y = x^+$ to minimize quadratic approximation:

$$x^+ = x - t\nabla f(x)$$



Blue point is x, red point is

$$x^+ = \operatorname*{argmin}_y \; f(x) + \nabla f(x)^T (y - x) + \frac{1}{2t} \|y - x\|_2^2$$

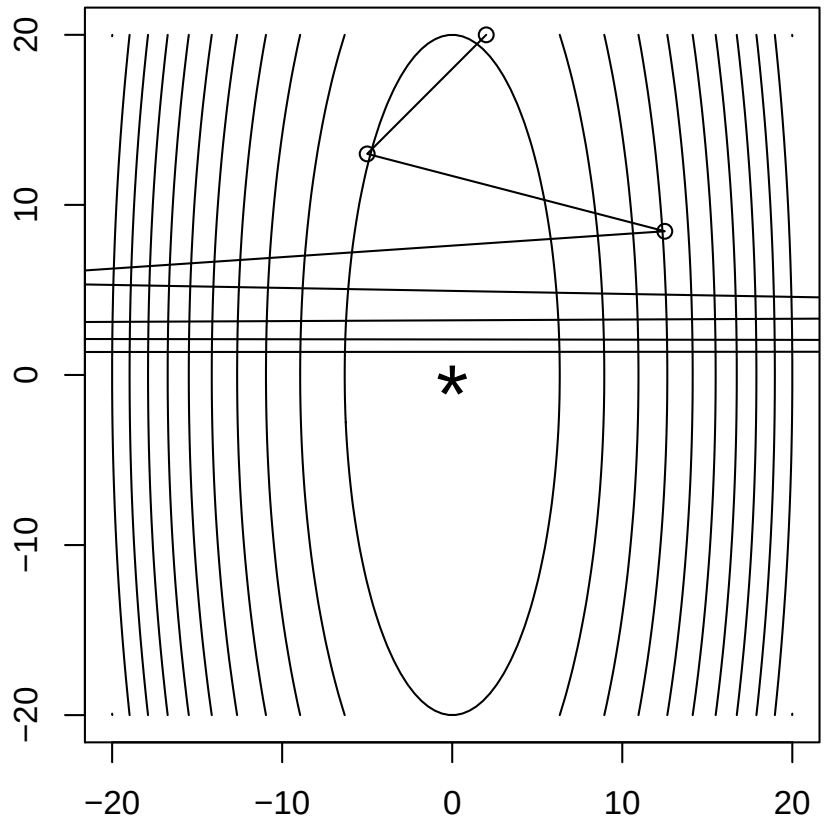
Outline

Today:

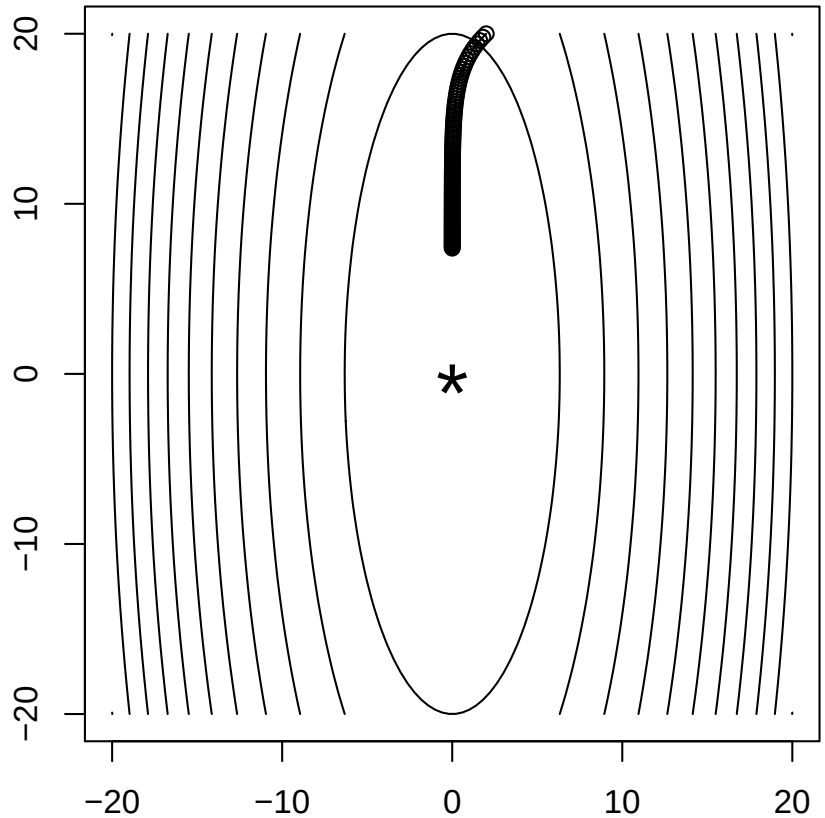
- How to choose step sizes
- Convergence analysis
- Gradient boosting
- Stochastic gradient descent

Fixed step size

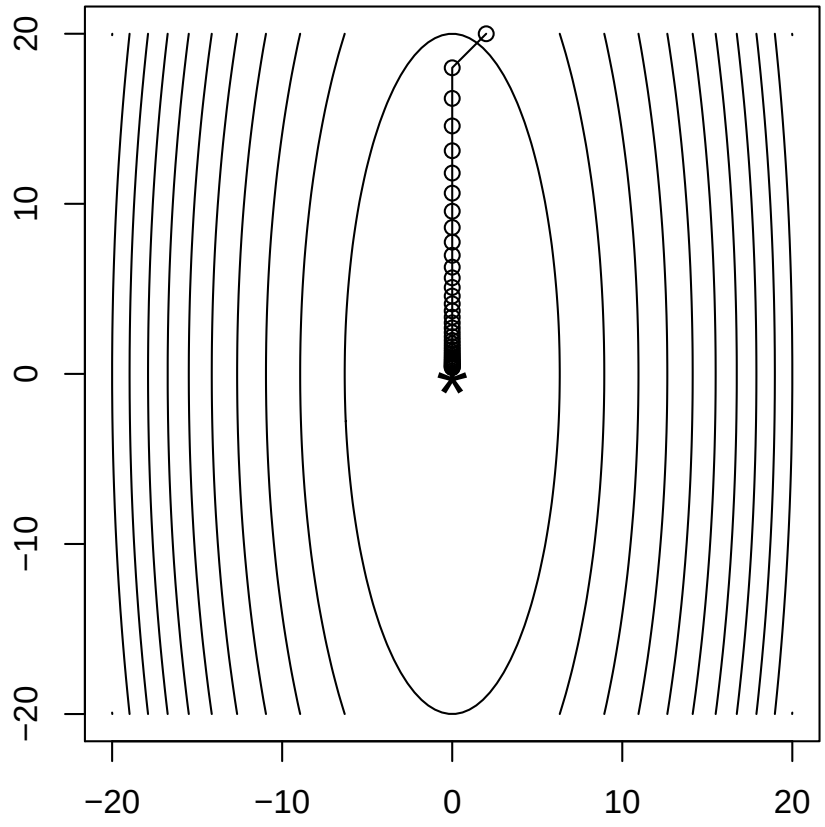
Simply take $t_k = t$ for all $k = 1, 2, 3, \ldots$, can diverge if t is too big.
Consider $f(x) = (10x_1^2 + x_2^2)/2$, gradient descent after 8 steps:



Can be slow if t is too small. Same example, gradient descent after 100 steps:



Converges nicely when t is "just right". Same example, gradient descent after 40 steps:



Convergence analysis later will give us a precise idea of "just right"

Backtracking line search

One way to adaptively choose the step size is to use backtracking line search:

- First fix parameters $0 < \beta < 1$ and $0 < \alpha \leq 1/2$
- At each iteration, start with $t = t_{\text{init}}$, and while

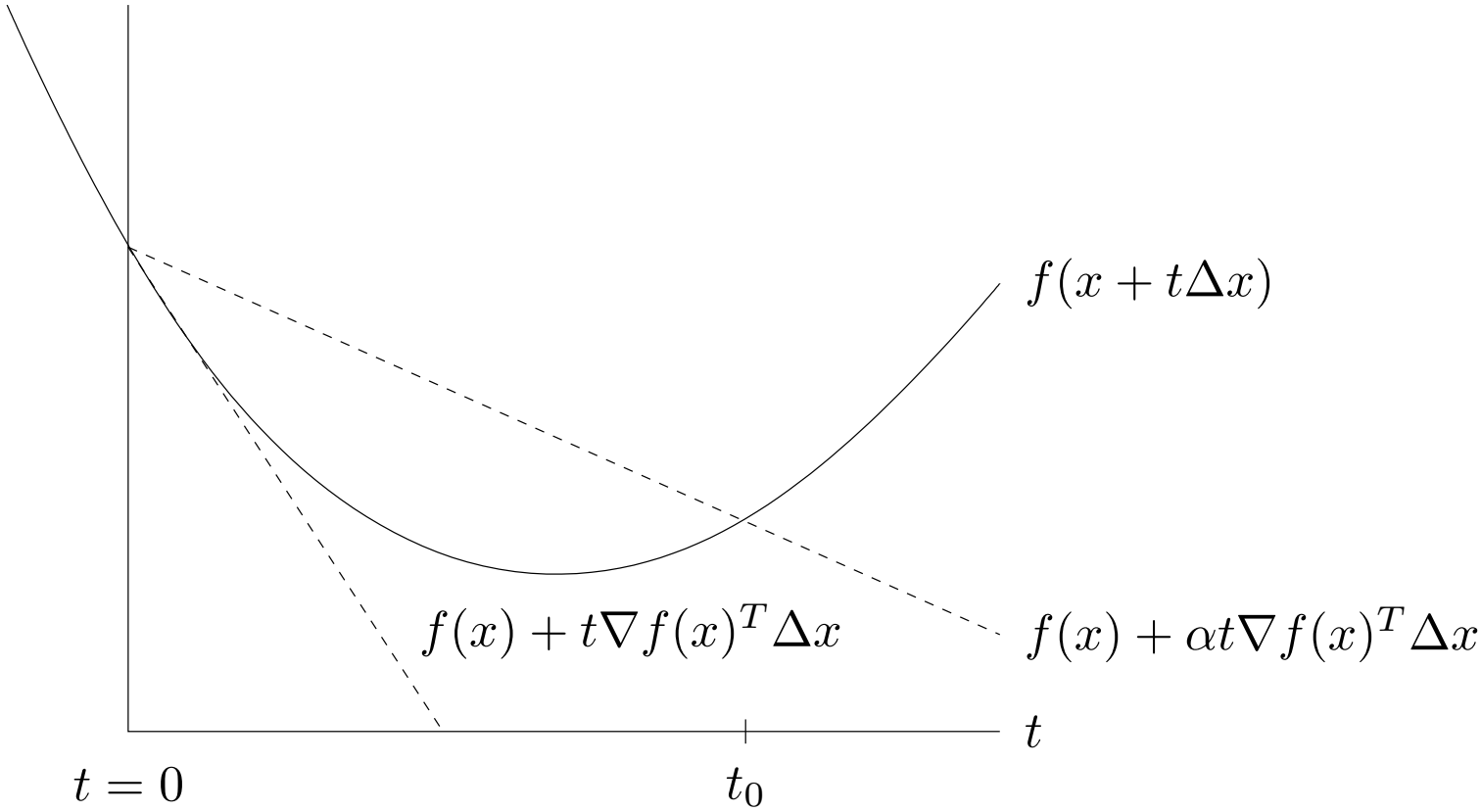
$$f(x - t\nabla f(x)) > f(x) - \alpha t \|\nabla f(x)\|_2^2$$

shrink $t = \beta t$. Else perform gradient descent update

$$x^+ = x - t\nabla f(x)$$

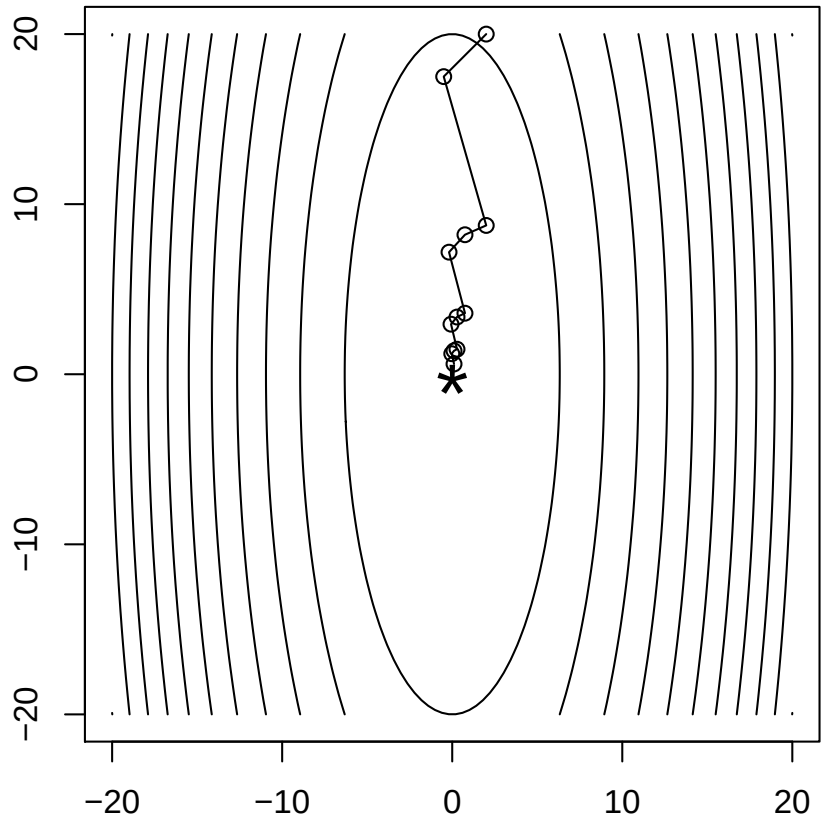
Simple and tends to work well in practice (further simplification: just take $\alpha = 1/2$)

Backtracking interpretation



For us $\Delta x = -\nabla f(x)$

Backtracking picks up roughly the right step size (12 outer steps, 40 steps total):



Here $\alpha = \beta = 0.5$

Exact line search

Could also choose step to do the best we can along direction of negative gradient, called exact line search:

$$t = \underset{s \geq 0}{\operatorname{argmin}} \; f(x - s\nabla f(x))$$

Usually not possible to do this minimization exactly

Approximations to exact line search are often not as efficient as backtracking, and it's usually not worth it

Convergence analysis

Assume that f convex and differentiable, with $\mathrm{dom}(f) = \mathbb{R}^n$, and additionally

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2 \quad \text{for any } x, y$$

I.e., ∇f is Lipschitz continuous with constant $L > 0$

Theorem: Gradient descent with fixed step size $t \leq 1/L$ satisfies

$$f(x^{(k)}) - f^\star \leq \frac{\|x^{(0)} - x^\star\|_2^2}{2tk}$$

We say gradient descent has convergence rate $O(1/k)$

I.e., to get $f(x^{(k)}) - f^\star \leq \epsilon$, we need $O(1/\epsilon)$ iterations

Proof

Key steps:

- ∇f Lipschitz with constant $L \Rightarrow$

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|_2^2 \quad \text{all } x, y$$

- Plugging in $y = x^+ = x - t\nabla f(x)$,

$$f(x^+) \leq f(x) - \Big(1 - \frac{Lt}{2}\Big)t\|\nabla f(x)\|_2^2$$

- Taking $0 < t \leq 1/L$, and using convexity of f,

$$\begin{aligned} f(x^+) &\leq f^\star + \nabla f(x)^T(x - x^\star) - \frac{t}{2}\|\nabla f(x)\|_2^2 \\ &= f^\star + \frac{1}{2t}\big(\|x - x^\star\|_2^2 - \|x^+ - x^\star\|_2^2\big) \end{aligned}$$

- Summing over iterations:

$$\sum_{i=1}^{k}(f(x^{(i)}) - f^\star) \leq \frac{1}{2t}\left(\|x^{(0)} - x^\star\|_2^2 - \|x^{(k)} - x^\star\|_2^2\right)$$
$$\leq \frac{1}{2t}\|x^{(0)} - x^\star\|_2^2$$

- Since $f(x^{(k)})$ is nonincreasing,

$$f(x^{(k)}) - f^\star \leq \frac{1}{k}\sum_{i=1}^{k}\left(f(x^{(i)}) - f^\star\right) \leq \frac{\|x^{(0)} - x^\star\|_2^2}{2tk}$$

Convergence analysis for backtracking

Same assumptions, f is convex and differentiable, $\mathrm{dom}(f) = \mathbb{R}^n$, and ∇f is Lipschitz continuous with constant $L > 0$

Same rate for a step size chosen by backtracking search

Theorem: Gradient descent with backtracking line search satisfies

$$f(x^{(k)}) - f^\star \leq \frac{\|x^{(0)} - x^\star\|_2^2}{2t_{\mathsf{min}} k}$$

where $t_{\mathsf{min}} = \min\{1, \beta/L\}$

If β is not too small, then we don't lose much compared to fixed step size (β/L vs $1/L$)

Convergence analysis under strong convexity

Reminder: strong convexity of f means $f(x) - \frac{2}{2}\|x\|_2^2$ is convex for some $m > 0$. If f is twice differentiable, then this is equivalent to

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{m}{2}\|y - x\|_2^2 \quad \text{all } x, y$$

Under Lipschitz assumption as before, and also strong convexity:

Theorem: Gradient descent with fixed step size $t \leq 2/(m + L)$ or with backtracking line search search satisfies

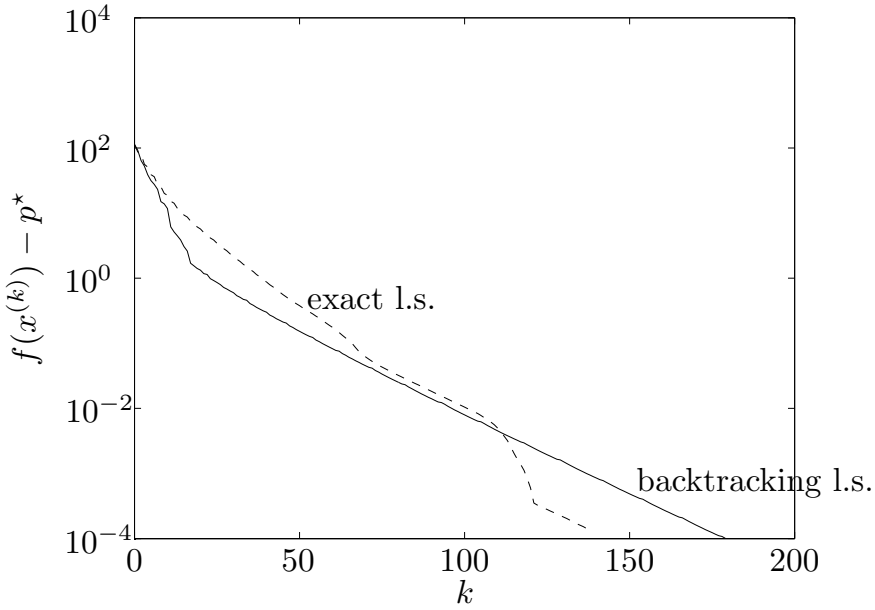
$$f(x^{(k)}) - f^\star \leq c^k \frac{L}{2}\|x^{(0)} - x^\star\|_2^2$$

where $0 < c < 1$

I.e., rate with strong convexity is $O(c^k)$, exponentially fast!

I.e., to get $f(x^{(k)}) - f^\star \leq \epsilon$, need $O(\log(1/\epsilon))$ iterations

Called linear convergence, because looks linear on a semi-log plot



(From B & V page 487)

Constant c depends adversely on condition number L/m (higher condition number $\Rightarrow$ slower rate)

A look at the conditions

A look at the conditions for a simple problem, $f(\beta) = \frac{1}{2}\|y - X\beta\|_2^2$

Lipschitz continuity of ∇f:

- This means $\nabla^2 f(x) \preceq LI$
- As $\nabla^2 f(\beta) = X^T X$, we have $L = \sigma_{\text{max}}^2(X)$

Strong convexity of f:

- This means $\nabla^2 f(x) \succeq mI$
- As $\nabla^2 f(\beta) = X^T X$, we have $m = \sigma_{\text{min}}^2(X)$
- If X is wide—i.e., X is $n \times p$ with $p > n$—then $\sigma_{\text{min}}(X) = 0$, and f can't be strongly convex
- Even if $\sigma_{\text{min}}(X) > 0$, can have a very large condition number $L/m = \sigma_{\text{max}}^2(X)/\sigma_{\text{min}}^2(X)$

A function f having Lipschitz gradient and being strongly convex satisfies:

$$mI \preceq \nabla^2 f(x) \preceq LI \quad \text{for all } x \in \mathbb{R}^n,$$

for constants $L > m > 0$

Think of f being sandwiched between two quadratics

May seem like a strong condition to hold globally (for all $x \in \mathbb{R}^n$). But a careful look at the proofs shows that we only need Lipschitz gradients/strong convexity over the sublevel set

$$S = \{x : f(x) \leq f(x^{(0)})\}$$

This is less restrictive (especially if S is compact)

Practicalities

Stopping rule: stop when $\|\nabla f(x)\|_2$ is small

- Recall $\nabla f(x^\star) = 0$ at solution $x^\star$
- If f is strongly convex with parameter m, then

$$\|\nabla f(x)\|_2 \leq \sqrt{2m\epsilon} \implies f(x) - f^\star \leq \epsilon$$

Pros and cons of gradient descent:

- Pro: simple idea, and each iteration is cheap (usually)
- Pro: fast for well-conditioned, strongly convex problems
- Con: can often be slow, because many interesting problems aren't strongly convex or well-conditioned
- Con: can't handle nondifferentiable functions

Can we do better?

Gradient descent has $O(1/\epsilon)$ convergence rate over problem class of convex, differentiable functions with Lipschitz gradients

First-order method: iterative method, updates $x^{(k)}$ in

$$x^{(0)} + \mathrm{span}\{\nabla f(x^{(0)}), \nabla f(x^{(1)}), \ldots \nabla f(x^{(k-1)})\}$$

Theorem (Nesterov): For any $k \leq (n-1)/2$ and any starting point $x^{(0)}$, there is a function f in the problem class such that any first-order method satisfies

$$f(x^{(k)}) - f^\star \geq \frac{3L\|x^{(0)} - x^\star\|_2^2}{32(k+1)^2}$$

Can attain rate $O(1/k^2)$, or $O(1/\sqrt{\epsilon})$? Answer: yes (we'll see)!

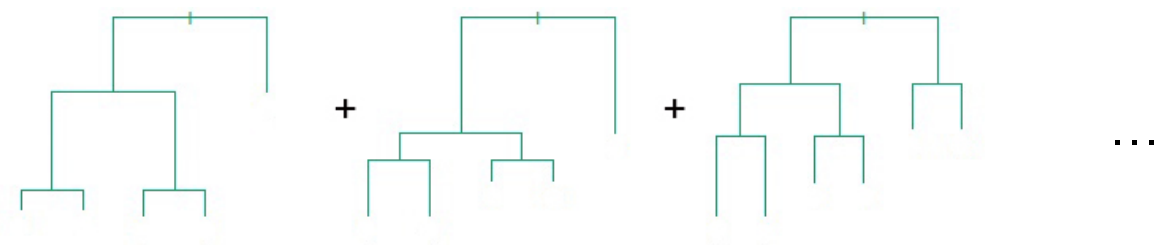
Gradient boosting

Given observations $y = (y_1, \dots y_n) \in \mathbb{R}^n$, predictor measurements $x_i \in \mathbb{R}^p$, $i = 1, \dots n$

Want to construct a flexible (nonlinear) model for outcome based on predictors. Weighted sum of trees:

$$u_i = \sum_{j=1}^{m} \beta_j \cdot T_j(x_i), \quad i = 1, \dots n$$

Each tree T_j inputs predictor measurements x_i, outputs prediction. Trees are grown typically pretty short



Pick a loss function L that reflects setting; e.g., for continuous y, could take $L(y_i, u_i) = (y_i - u_i)^2$

Want to solve

$$\min_{\beta} \sum_{i=1}^{n} L\Big(y_i, \sum_{j=1}^{M} \beta_j \cdot T_j(x_i)\Big)$$

Indexes all trees of a fixed size (e.g., depth $= 5$), so M is huge

Space is simply too big to optimize

Gradient boosting: basically a version of gradient descent that is forced to work with trees

First think of optimization as $\min_u \ f(u)$, over predicted values u, subject to u coming from trees

Start with initial model, e.g., fit a single tree $u^{(0)} = T_0$. Repeat:

- Compute negative gradient d at latest prediction $u^{(k-1)}$,

$$d_i = -\left[\frac{\partial L(y_i, u_i)}{\partial u_i}\right]\Bigg|_{u_i = u_i^{(k-1)}}, \quad i = 1, \ldots n$$

- Find a tree T_k that is close to a, i.e., according to

$$\min_{\text{trees } T} \sum_{i=1}^{n} (d_i - T(x_i))^2$$

  Not hard to (approximately) solve for a single tree

- Compute step size α_k, and update our prediction:

$$u^{(k)} = u^{(k-1)} + \alpha_k \cdot T_k$$

  Note: predictions are weighted sums of trees, as desired

Stochastic gradient descent

Consider minimizing a sum of functions

$$\min_x \sum_{i=1}^m f_i(x)$$

As $\nabla \sum_{i=1}^m f_i(x) = \sum_{i=1}^m \nabla f_i(x)$, gradient descent would repeat:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \sum_{i=1}^m \nabla f_i(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

In comparison, stochastic gradient descent or SGD (or incremental gradient descent) repeats:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f_{i_k}(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

where $i_k \in \{1, \ldots m\}$ is some chosen index at iteration k

Two rules for choosing index i_k at iteration k:

- Cyclic rule: choose $i_k = 1, 2, \ldots m, 1, 2, \ldots m, \ldots$
- Randomized rule: choose $i_k \in \{1, \ldots m\}$ uniformly at random

Randomized rule is more common in practice

What's the difference between stochastic and usual (called batch) methods? Computationally, m stochastic steps $\approx$ one batch step. But what about progress?

- Cyclic rule, m steps: $x^{(k+m)} = x^{(k)} - t \sum_{i=1}^m \nabla f_i(x^{(k+i-1)})$
- Batch method, one step: $x^{(k+1)} = x^{(k)} - t \sum_{i=1}^m \nabla f_i(x^{(k)})$
- Difference in direction is $\sum_{i=1}^m [\nabla f_i(x^{(k+i-1)}) - \nabla f_i(x^{(k)})]$

So SGD should converge if each $\nabla f_i(x)$ doesn't vary wildly with x

Rule of thumb: SGD thrives far from optimum, struggles close to optimum ... (we'll revisit in just a few lectures)

References and further reading

- D. Bertsekas (2010), "Incremental gradient, subgradient, and proximal methods for convex optimization: a survey"
- S. Boyd and L. Vandenberghe (2004), "Convex optimization", Chapter 9
- T. Hastie, R. Tibshirani and J. Friedman (2009), "The elements of statistical learning", Chapters 10 and 16
- Y. Nesterov (1998), "Introductory lectures on convex optimization: a basic course", Chapter 2
- L. Vandenberghe, Lecture notes for EE 236C, UCLA, Spring 2011-2012