IE510 Applied Nonlinear Programming

Lecture 2: Gradient Methods III: Convergence Rate

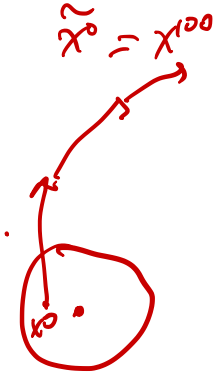
Ruoyu Sun

Feb 1, 2018

Outline

Questions for Last Time

- **Q1:** You use GD with constant stepsize to solve a problem, but it doesn't converge. (function values don't converge)
  1) What are the possible reasons? (0) Function goes to $-\infty$.
  (i) Stepsize too large.
  (ii) L doesn't exist. e.g. $(xy-1)^2$.
  2) How to make it work?
  - Reduce stepsize to smaller amount;
  - Armijo rule.



- **Q2** You use GD with diminishing stepsize to solve a problem. Recall the conditions $\sum_r \alpha_r = \infty$ and $\alpha_r \to 0$.

  This time, you set a lower bound 0.01; e.g., $\alpha_r = 1/r + 0.01$.

  Will this algorithm converge? Most students: No. Answer: Yes — if $0.01 < \frac{2}{L}$

- **Q3** Use GD to solve $\min_x x^6$. How to guarantee convergence?
  use level-set analysis

Last Time summary

- **Prop. 1**: Minimizing a function f with L-Lipschitcz gradient, GD with constant stepsize $\in (0, 2/L)$ works; i.e., every limit point is a stationary point.

- Subtleties of convergence:
  - function value may go to $-\infty$ → remember this one.
  - iterates may go unbounded

- **Prop. 1b:** Under same condition to Prop. 1, GD with bounded stepsize in $(\epsilon, 2/L - \epsilon)$ or diminishing stepsize satisfying $\sum_r \alpha_r = \infty$ and $\alpha_r \to 0$ works.

- **Prop. 2:** (line search) GD with line search (Armijo rule, minimization, limited minimization) works.

  Remark: no assumption on L-Lipschitz gradient.

Today

- Convergence Rate Analysis of GD
- After today's course, you will be able to
  - Describe the convergence result of GD for strongly convex problems
  - Show how to analyze quadratic problems

    Advanced: Show how to analyze strongly convex problems

  For engineers:
  - Explain why preprocessing data is useful

Remark: Learn something is better than nothing.
At least, try to understand simple cases.

What is Convergence Rate Analysis

optimal value
||
$\epsilon = 10^{-3}$, $\epsilon = 10^{-7}$.

- Define an ϵ optimal solution as $\{\mathbf{x}_\epsilon \mid f(\mathbf{x}_\epsilon) - f^* \leq \epsilon\}$
- **Convergence Rate** (convex case):
  1. Measures the number of iterations required to get an ϵ optimal solution
     e.g. it takes 500 iterations to get $\epsilon = 10^{-4}$ error.
  2. Gives global behavior of the algorithm
  3. Important measure for evaluating algorithms in big data applications
- Question: What determines the convergence rate?

Illustration

Figure: What Determines Convergence Rate?

Illustration

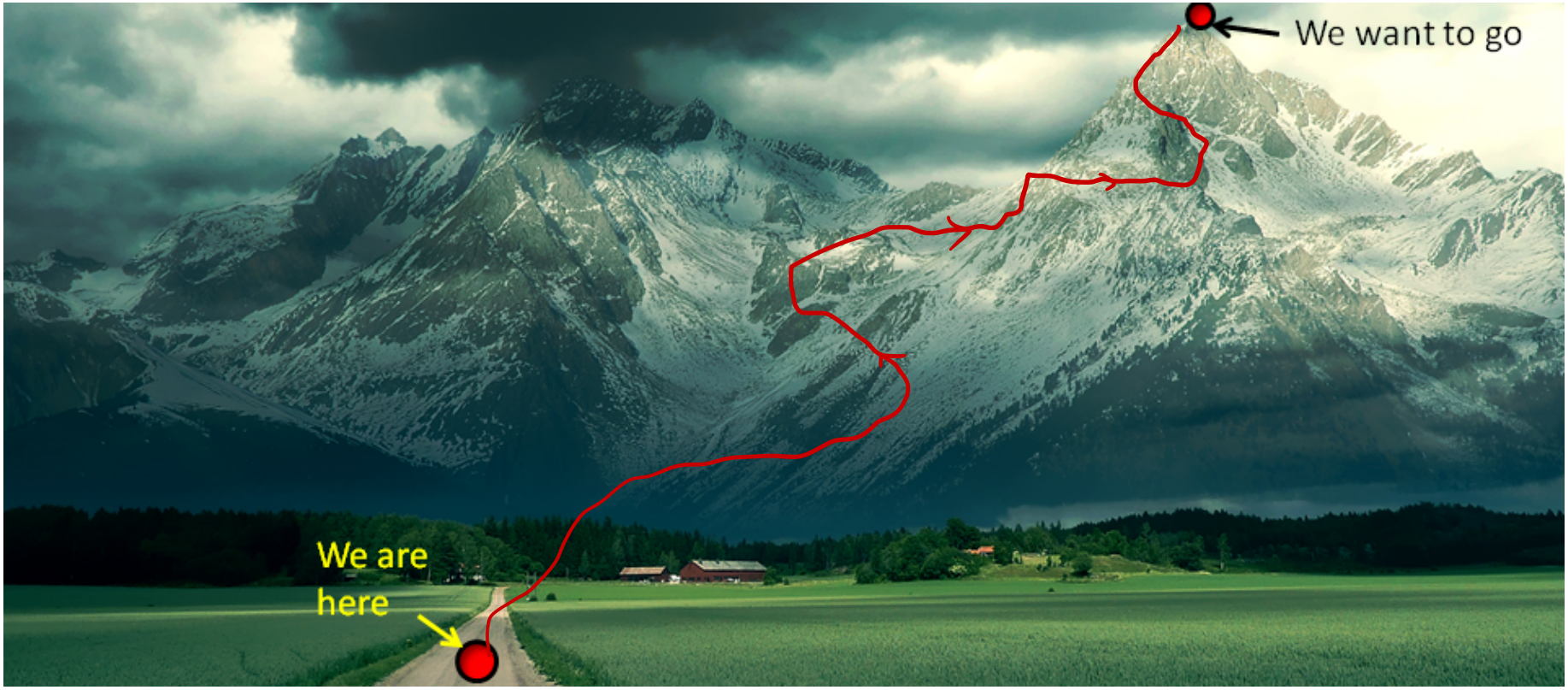


Figure: What Determines Convergence Rate?

10 days or 2 hours to the top ?

Example: Basketball ag-again

- p is position, x is the force. $\min \frac{1}{2}(p - cx)^2$
- GD: $x^+ = (1 - \alpha c^2)x + \alpha c p$. What is the speed of convergence?

  Simple case: $p = 0$, $x^+ = (1 - \alpha c^2)x$.

  Speed depends on $1 - \alpha c^2$

  1, 0.5, 0.25, ... $x^+ = 0.5x$,

  1, 0.1, 0.01, 0.001, ... $x^+ = 0.1x$.

  if $\alpha = 1/c^2$ 0.
- The best possible speed: pick $\alpha = 1/c^2$, then converge in 1 step.
- 1-dim quadratic problem not that interesting.

A Simple Example in Regression

- Let's take a look at a simple example in predicting the house price giving the living areas

Living Area (feet^2)	Price (1000$)
5719	567
3241	345
1139	141
1572	167
1101	287
2576	227
⋮	⋮

- Let's build a regression model and use gradient descent to solve the problem

A Simple Example in Regression

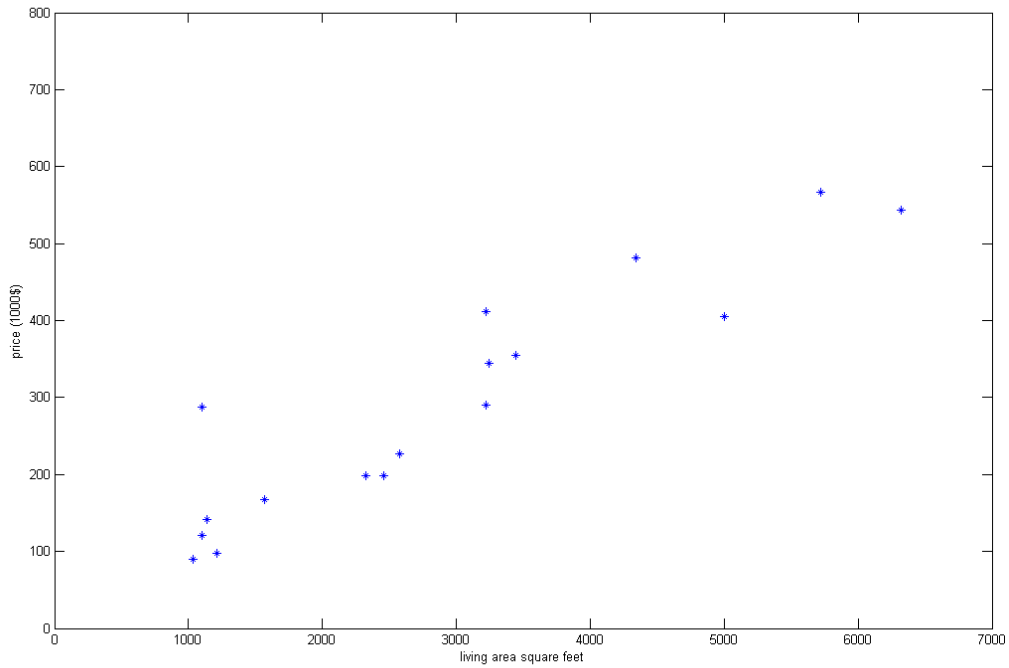
- The data is first plotted below (A total of 17 data points)
- Construct the regression model: variables x_1 (slope), x_0 (intercept)
- Data matrix $\mathbf{A} \in \mathbb{R}^{17\times 2}$; first column the area data, second column all 1
- Data vector $\mathbf{b} \in \mathbb{R}^{17\times 1}$, being the housing price.
- Solve the following problem $\min \frac{1}{2}\|\mathbf{Ax}-\mathbf{b}\|^2$

$b = a_1 x_1 + x_0$

price, area, constant

$A = \begin{bmatrix} 5719 & 1 \\ 3615 & 1 \\ \vdots & \vdots \\ 1 & 1 \end{bmatrix}$

$b = \begin{bmatrix} 56 \\ 345 \\ \vdots \\ 1 \end{bmatrix}$



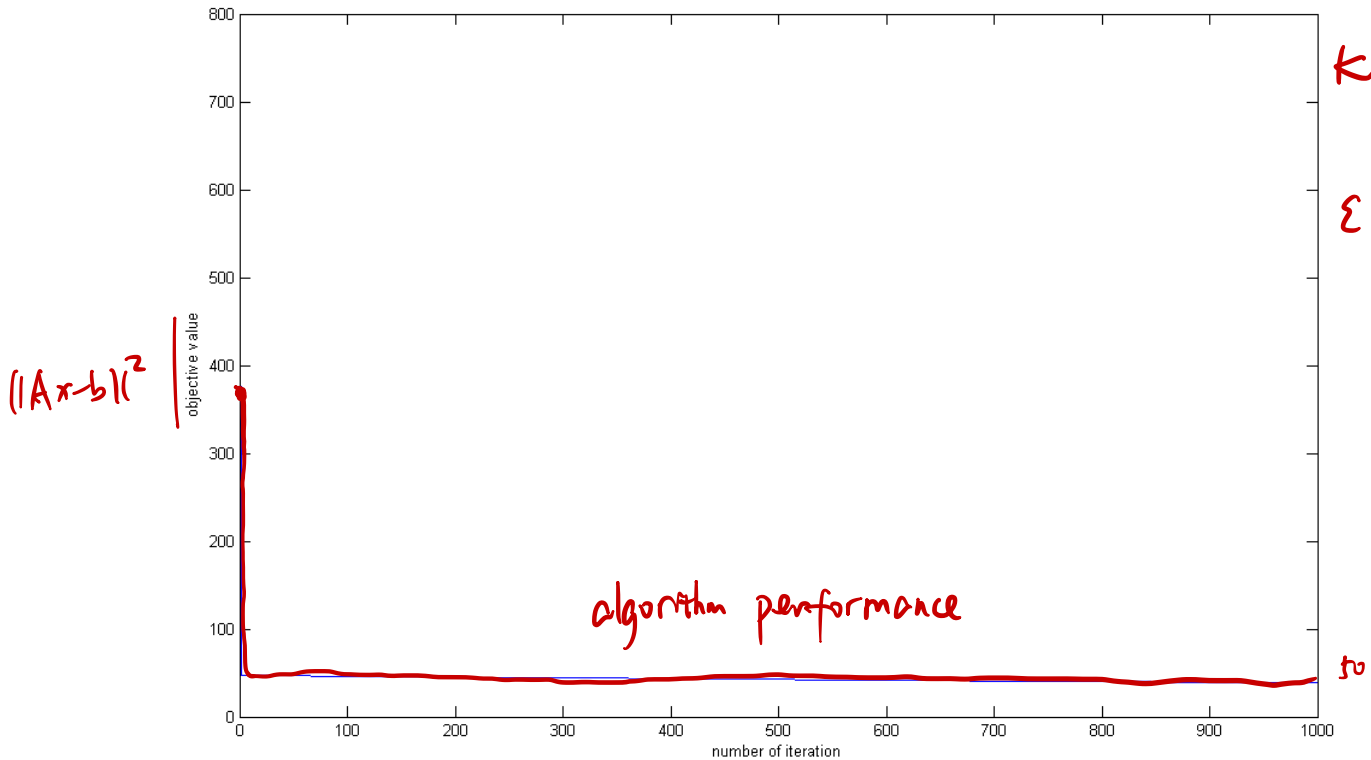
A Simple Example in Regression

- Use the gradient with constant stepsize (how to compute the stepsize?)
- **First Try**: Scale the data so that all the areas are multiplied by 0.01, and all the prices are multiplied by 0.1 – scaling the data
- Run the steepest gradient descent for 1000 iterations; initial $(10, 50)$
- Eigenvalues of $\mathbf{A}'\mathbf{A}$ are 0.0004 and 1.856

$$K = \frac{1.856}{0.0004} \approx 4000.$$

$K \log\frac{1}{\varepsilon} \approx 4000 \times 11.5 \approx 46000,$

$\varepsilon = 10^{-5}$.



$\|Ax-b\|^2$

algorithm performance

to

A Simple Example in Regression

- Use the gradient with constant stepsize (how to compute the stepsize?)
- **First Try**: Scale the data so that all the areas are multiplied by 0.01, and all the prices are multiplied by 0.1 – scaling the data
- Run the steepest gradient descent for 1000 iterations; initial $(10, 50)$
- Eigenvalues of $\mathbf{A}'\mathbf{A}$ are 0.0004 and 1.856

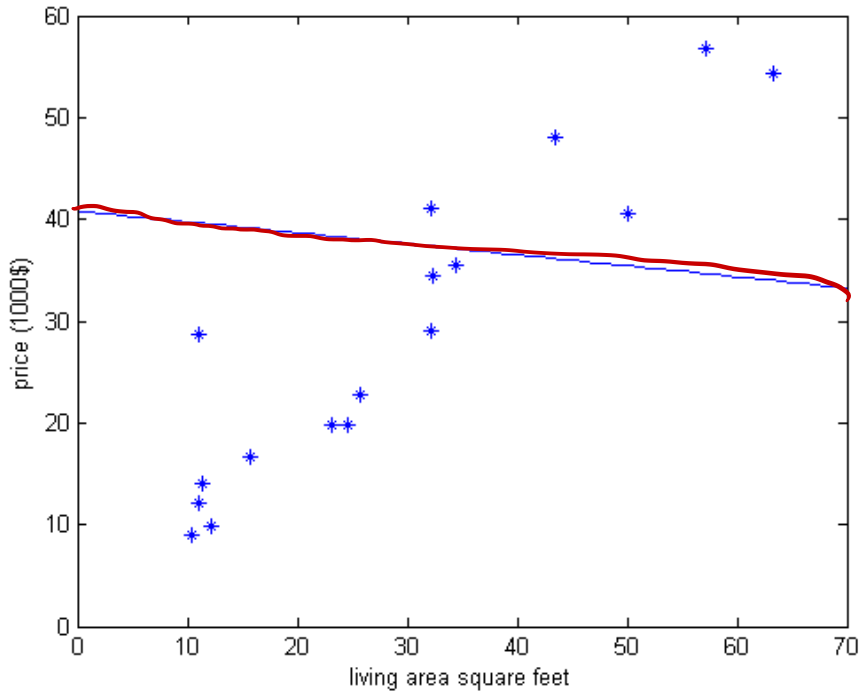
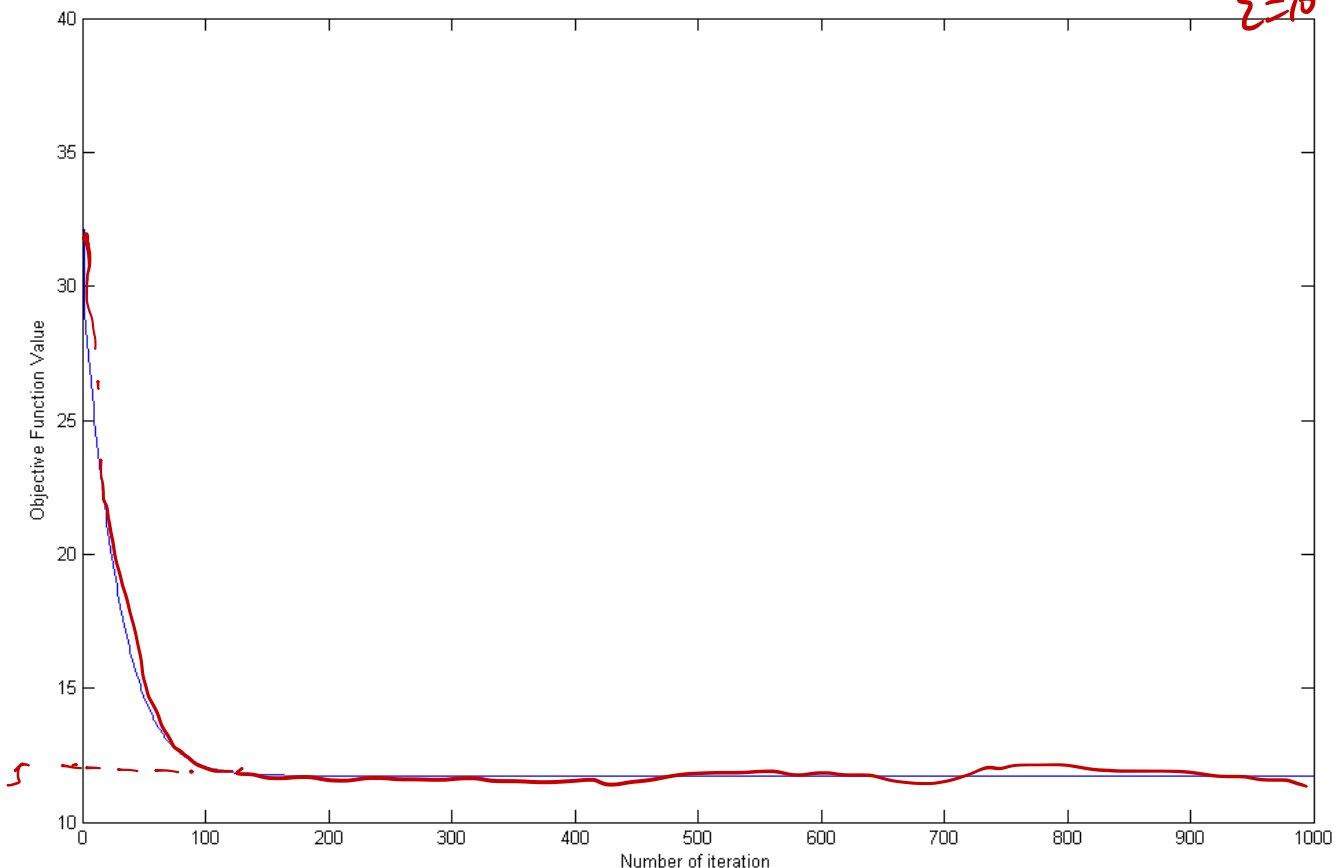


Figure: The final fitting.

A Simple Example in Regression

- Use the gradient with constant stepsize
- **Second Try**: Scale the data so that all the areas are multiplied by 0.0001, and all the prices are multiplied by 0.1 – scaling the data
- Run the steepest gradient descent for 1000 iterations; initial $(10, 50)$
- Eigenvalues of $\mathbf{A}'\mathbf{A}$ are 0.4 and 18.448

$K = \frac{18.448}{0.4} \simeq 40$,
$\varepsilon = 10^{-5}$, $K \cdot \log\frac{1}{\varepsilon} \simeq 460$.

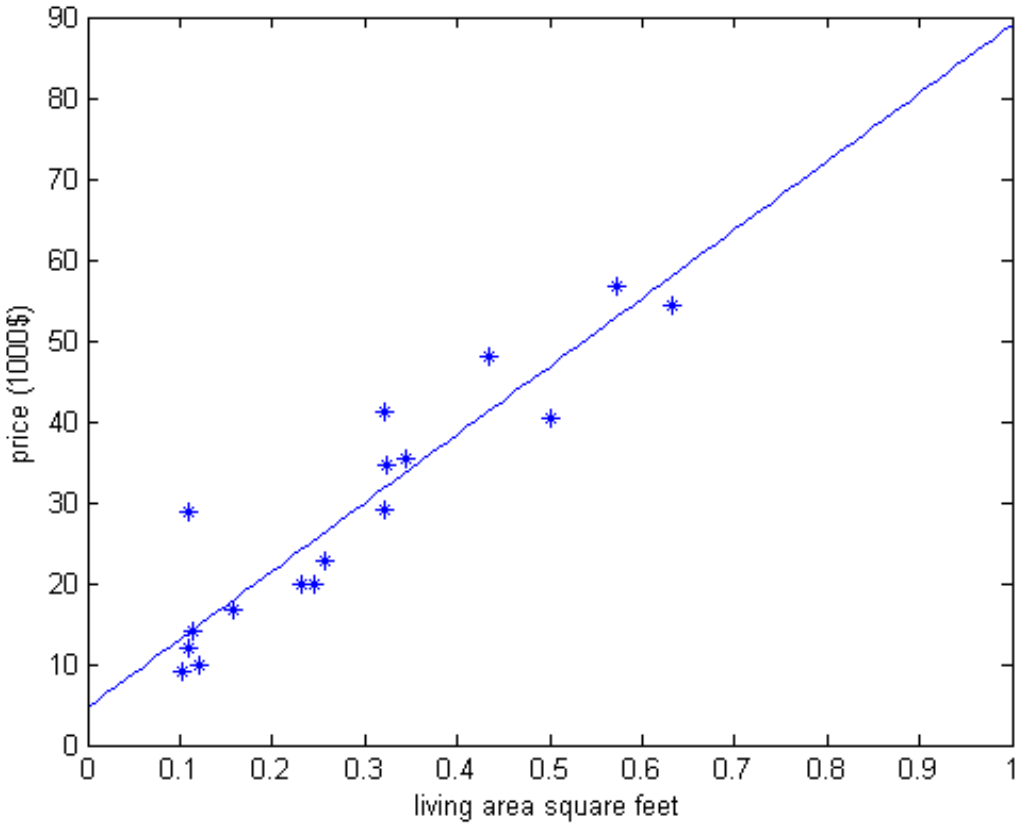


A Simple Example in Regression

- Use the gradient with constant stepsize
- **Second Try**: Scale the data so that all the areas are multiplied by 0.0001, and all the prices are multiplied by 0.1 – scaling the data
- Run the steepest gradient descent for 1000 iterations; initial $(10, 50)$
- Eigenvalues of $\mathbf{A}'\mathbf{A}$ are 0.4 and 18.448

$A = \begin{bmatrix} 5719 & 1 \\ 3241 & 1 \\ \vdots & \vdots \end{bmatrix}$, new $A = \begin{bmatrix} 0.5719 & 1 \\ 0.3241 & 1 \\ \vdots & \vdots \end{bmatrix}$

new $b = \begin{bmatrix} 56.7 \\ 34.5 \\ \vdots \end{bmatrix}$



Outline

Discussion on Example

- Seems that scaling the data gives much different performance on the gradient descent algorithm
- Quora: When does logistic regression not converge?

  Top Answer: Common reasons:
  1. Code/Optimization method has bug.
  2. Learning rate too large
  3. Feature not normalized (values of different feature have totally different scale, just ran into today).

- **Story**: When I learned Torch, the first "trick" is to scale data properly.
  It looks magic first, then I realize it should be explained by optimization theory.

Analysis of 2-dim Example

$\min_x \frac{1}{2} x^T Q x$.

$Q = \begin{bmatrix} L & 0 \\ 0 & 1 \end{bmatrix}$.

- Start from simple case $\min_{x_1,x_2} \frac{1}{2}(Lx_1^2 + x_2^2)$.
- $\nabla f(x) = \begin{pmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{pmatrix} = \begin{pmatrix} Lx_1 \\ x_2 \end{pmatrix}$.
- GD: $x^+ = x - \alpha \nabla f(x) = x - \alpha \begin{pmatrix} Lx_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} (1-\alpha L)x_1 \\ (1-\alpha) x_2 \end{pmatrix}$.
  i.e., $\begin{pmatrix} x_1^+ \\ x_2^+ \end{pmatrix} = \begin{pmatrix} (1-\alpha L)x_1 \\ (1-\alpha) x_2 \end{pmatrix}$
- Stepsize choice $\alpha = 1/L$, then

$$x_1^+ = (1-\alpha L)x_1 = 0,$$
$$x_2^+ = (1-\alpha) x_2 = (1 - \tfrac{1}{L}) x_2.$$

- Convergence speed depends on $1 - \frac{1}{L}$

e.g. $L=2$, $1-\frac{1}{L} = 0.5$,

$L=100$, $1-\frac{1}{L} = 0.99$.

$1 - \frac{1}{L}$ small $\rightarrow$ fast ?

$1 - \frac{1}{L}$ large $\rightarrow$ slow ? (L large).

Analysis of Quadratic Problems

- Next, consider $\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\mathbf{x}^T Q\mathbf{x}$.
  For simplicity, assume Q is symmetric positive-definite.
- GD: $x^+ = x - \alpha \nabla f(x) = x - \alpha \cdot Qx = (I - \alpha Q)x$.
- Convergence speed?
  If $\|I - \alpha Q\| < 1$, converge (Lecture 0); so pick $\alpha < \frac{2}{\lambda_{max}(Q)}$.
  or by Prop. 1 last time.
  Simple case: $Q = \text{diag}(\lambda_1, \ldots, \lambda_n)$, $\lambda_1 \geq \cdots \geq \lambda_n$. Pick $\alpha = \frac{1}{\lambda_1}$.

$$\mathbf{x}^+ = \left[I - \alpha \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}\right] x$$

$$\Rightarrow \begin{pmatrix} x_1^+ \\ \vdots \\ x_n^+ \end{pmatrix} = \begin{pmatrix} (1-\alpha\lambda_1)x_1 \\ \vdots \\ (1-\alpha\lambda_n)x_n \end{pmatrix}.$$

Speed depends on $\max_{1\leq i\leq n} \{1-\alpha\lambda_i\}$
$= 1 - \alpha\lambda_n = 1 - \frac{\lambda_n}{\lambda_1}$.

obtain $\frac{\|x^r\|}{\|x^0\|} \leq \left(1 - \frac{\lambda_n}{\lambda_1}\right)^r$.

Analysis of Quadratic Problems (cont'd)

- **General case**: How to connect Q with diagonal matrix?

  Eigen-decomposition $Q = U^T D U$, where D diagonal; assume $D = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$.
  U orthonormal.

- Then $\mathbf{x}^+ = (I - \alpha U^T D U)\mathbf{x}$. Trick:

  $$U\mathbf{x}^+ = (U - \alpha \underbrace{U U^T}_{I} D U)\mathbf{x}$$
  $$= (U - \alpha D U)\mathbf{x} = \cdots = U\mathbf{x}.$$

  or $\mathbf{y}^+ = (I - \alpha D)\mathbf{y}$.

- Obtain convergence rate $(\alpha = 1/\lambda_1)$.

  relative error $\frac{\|\mathbf{y}^r\|}{\|\mathbf{y}^0\|} \le \left(1 - \frac{\lambda_n}{\lambda_1}\right)^r$.

  $$\|\mathbf{y}\|^2 = \|U\mathbf{x}\|^2 = \mathbf{x}^T U^T U \mathbf{x} = \|\mathbf{x}\|^2.$$

  $$\Rightarrow \quad \frac{\|\mathbf{x}^r\|}{\|\mathbf{x}^0\|} \le \left(1 - \frac{\lambda_n}{\lambda_1}\right)^r.$$

General Convex Quadratic Problem

- What about general quadratic problem $\min_{\mathbf{x}} \mathbf{x}^T Q\mathbf{x} + 2\mathbf{b}^T\mathbf{x}$?
- Try to relate to $\min_{\mathbf{x}} \mathbf{x}^T Q\mathbf{x}$.

$x^* = -Q^{-1}b.$

$$x^+ = x - \alpha \nabla f(x)$$
$$= x - \alpha(Qx + b).$$
$$\Rightarrow \underbrace{x^+ - x^*}_{y^+} = \underbrace{(x - x^*)}_{y} - \alpha Q \underbrace{(x - x^*)}_{y}.$$

Reduce to previous page.

- **Analysis chain**:
diagonal $\rightarrow$ pure quadratic $\rightarrow$ quadratic plus linear term.

Result for Strongly Convex Quadratic Case

- **Proposition 3**: Suppose A is symmetric PD. Consider solving

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \triangleq \mathbf{x}^T Q\mathbf{x} + 2\mathbf{b}^T x + \mathbf{c}.$$

Suppose GD with stepsize $1/L$, where $L = \lambda_{\max}(Q)$, generates a sequence $\mathbf{x}^r$. Then $\frac{\|x^r - x^*\|}{\|x^0 - x^*\|} \le \left(1 - \frac{1}{\kappa}\right)^r$.

$$\frac{e^r}{e^0} = \frac{f(\mathbf{x}^r) - f*}{f(\mathbf{x}^0) - f*} \le \left(1 - \frac{1}{\kappa}\right)^r.$$

where the condition number $\kappa = \frac{\lambda_{max}(Q)}{\lambda_{min}(Q)}$.

- **Remark 1**: For solving $\min_{\mathbf{x}\in\mathbb{R}^n} \|Ax - b\|^2$, $\kappa = \frac{\lambda_{max}(A^TA)}{\lambda_{min}(A^TA)}$.

Remark 2: Stepsize $0 < \alpha < \frac{2}{L}$, some analysis works; same order of convergence speed, different constant. Consider $1/L$ for simplicity.

Transform Rate to Number of Iterations

$\epsilon = 10^{-5}$.

- How many iterations needed for $e(\mathbf{x}^r)$ to reach below ϵ?
- Relative error $e(\mathbf{x}^r)/e(\mathbf{x}^0) = \beta^r \leq \epsilon$, where $\beta = 1 - 1/\kappa$, so we require:

$$r \ln \beta \leq \ln \epsilon \Rightarrow r \ln \frac{1}{\beta} \geq \ln \frac{1}{\epsilon} \quad (>0);$$

$$r \geq \frac{\log 1/\epsilon}{\log 1/\beta} = \frac{\log 1/\epsilon}{\log\left(\frac{1}{1-1/\kappa}\right)}.$$

- For large κ, # of iterations

$$r \gtrsim \kappa \log \frac{1}{\epsilon}.$$

e.g. $\epsilon = 10^{-5}$, $\log \frac{1}{\epsilon} \approx 5 \log 10 \approx 11.5$

# of iterations $11.5\,\kappa$.

- κ big: ill-conditioned (slow convergence for GD)
- κ small: well-conditioned (fast convergence for GD)

Transform Rate to Time Complexity

- We only considered # of iterations; but as said before, Newton method takes 1 iteration.
- Should consider computation complexity: total amount of computation.
- **Unit operation**: we define as one $+$ or one $\times$.
- $\mathbf{u} \times \mathbf{v}$ for n-dim vector: time
- Compute gradient $\nabla f(\mathbf{x}) = A^T(Ax)$ takes time
- **Result:** GD solves $\|Ax - b\|^2$ in time $O(\quad)$

  Benchmark: Matrix inverse time $O(\quad)$ (best? OPEN)

core of CS theory; famous P=NP question.

time 1.

$2n$.

$= a_1b_1 + \cdots + a_nb_n$
$(n-1)$ "+"'s
n "x"'s

$O(n^2)$.

do this first

$n^2 \cdot k \log\frac{1}{\varepsilon} \approx O(n^2 \cdot k)$.

n^3

A^{-1}

Conjecture: n^2.

best known: $n^{2.4238\ldots}$

Recent paper: 2.4241 → 2.4238..

Application to Data Preprocessing

- The condition number being large, or **ill-conditioning**, usually arises when the optimization variables are scaled incoherently
- Example:
  1. x_1: gas flow, lbs/hr, nominal value $11,000$
  2. x_2: waste buildup: lbs, nominal value 6×10^{-4}
  3. x_3: steam thermal resistance, $\texttt{BTU/(hr.}ft^2\texttt{.F)}$, nominal value 100
- First step: "data preprocessing".
  Should rescale the units so that the nominal values are balanced
- **Lesson:** Normalizing data is reducing condition number, which makes GD faster.

Application to Data Preprocessing (cont'd)

- What is the effect of scaling data?
- Linear regression $\|A_1x_1 + \cdots + A_nx_n - \mathbf{b}\|^2$. Speed depends on $\kappa(A^TA)$.

$A = [A_1\ A_2\ \cdots\ A_n]$ — $\times d_1$, $\times d_2$, $\cdots \times d_n$

- Scale feature i by d_i, i.e., scale column A_i by d_i.

  Change matrix A to $\hat{A} =$ AD;

  For this $\hat{A}$, all of its diagonals are 1, if $d_i = \frac{1}{\|A_i\|}$. (normalize each column)

- Speed now depends on $\kappa(\hat{A}^T\hat{A}) = \kappa(DA^TAD)$.
- **Question**: did we change to a new problem?
- **Answer**: No, because $\|A_1x_1 + \cdots + A_nx_n - b\|^2$

  $\|A_1d_1\frac{x_1}{d_1} + \cdots + A_nd_n\frac{x_n}{d_n} - b\|^2$.

e.g. $A^TA = \begin{bmatrix} 1000 & \times & \times \\ 52 & 10 & \times \\ 250 & 13 & 10^{-5} \end{bmatrix}$,

$\hat{A}^T\hat{A} = \begin{bmatrix} 1 & \times & \times \\ \times & 1 & \times \\ \times & \times & 1 \end{bmatrix}$.

- Called Jacobi preconditioning in numerical linear algebra.

Open question: how much improvement?

Strongly Convex Function

- How much can we generalize?
- **Textbook** "nonquadratic cost function": near a nonsingular local-min (Hessian is PD), the analysis holds. Local analysis.

  The convergence speed depends on the condition number.
- **Nesterov's lecture notes**: discuss in detail the analysis of strongly convex functions –our next focus. Global analysis

Strongly Convex Function

- Recall that f is continuously differentiable, f is convex iff

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle.$$

- If f is twice continuously differentiable, then

$$f \text{ is convex} \Leftrightarrow \nabla^2 f(\mathbf{x}) \succeq 0, \text{ for all } \mathbf{x}$$

- **A New Notion**: f is strongly convex iff exists $\sigma > 0$

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\sigma}{2} \|\mathbf{x} - \mathbf{y}\|^2.$$

- If f is twice continuously differentiable, then exists $\sigma > 0$

$$f \text{ is strongly convex} \Leftrightarrow \nabla^2 f(\mathbf{x}) \succeq \sigma \mathbf{I}, \text{ for all } \mathbf{x}$$

- **Example**: A quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x}$ with strictly positive definite $\mathbf{A}$, i.e., $\mathbf{A} \succeq \sigma \mathbf{I}$; Here σ is the smallest eigenvalue for $\mathbf{A}$

GD for SC Function: Step 1

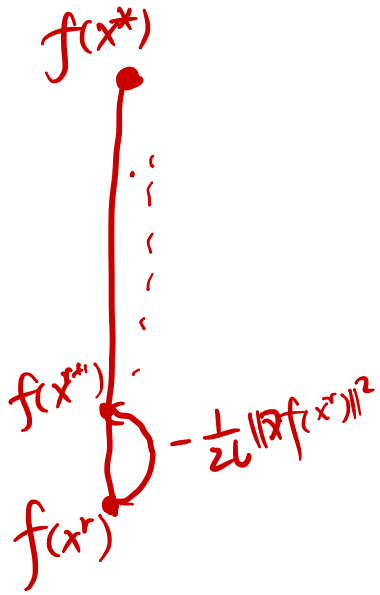
- Let's begin analysis of gradient descent for strongly convex problems
- **Goal** To see what convergence speed depends on
- Then $\alpha_r = \frac{1}{L}$ and from $f(x^{r+1}) \le f(x^r) + (-\alpha + \frac{1}{2}\alpha^2 L) \cdot \|\nabla f(x^r)\|^2$ (last lecture)

get $$f(\mathbf{x}^{r+1}) = f(\mathbf{x}^r + \alpha_r \nabla f(\mathbf{x}^r)) \le f(\mathbf{x}^r) - \frac{1}{2L}\|\nabla f(\mathbf{x}^r)\|^2$$

- This shows that the after one round of algorithm, the objective function achieves "sufficient descent"
- The amount of the descent can be measured by the size of the gradient

Graph

- Last page: each iteration reduces $-\frac{1}{2L}\|\nabla f(\mathbf{x}^r)\|^2$
- Want $f(\mathbf{x})$ gets close to $f* = f(\mathbf{x}^*)$.
- Graph:



E.g. A snail is climbing a wall.
The wall is 10 meters high.
climb 0.5 meters each day, $\frac{10}{0.5} = 20$ days.

Question: "height" $f(x^r) - f(x^*)$ v.s. $\|\nabla f(x^r)\|^2$?

GD for SC Function: Step 2

Goal: relate $f(x^*)-f(x^r)$ to $\|\nabla f(x^r)\|^2$.

- Using the strong convexity assumption, we have

$$f(\mathbf{x}^*) \geq f(\mathbf{x}^r) + \langle \nabla f(\mathbf{x}^r), \mathbf{x}^* - \mathbf{x}^r \rangle + \frac{\sigma}{2}\|\mathbf{x}^r - \mathbf{x}^*\|^2 \qquad (1)$$

$\triangleq h(x^*)$

$\Rightarrow f(x^*)-f(x^r) \geq \cdots \geq \|\nabla f(x^r)\|^2$.

- Minimizing the right hand side over $\mathbf{x}^*$, the optimal solution is

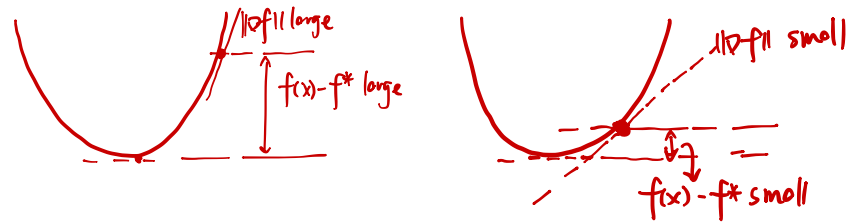
$\hat{x} = x^r - \frac{1}{\sigma}\nabla f(x^r) = \arg\min_x h(x)$,

$\Rightarrow \min_x h(x) = h(\hat{x}) = f(x^r) + \langle \nabla f(x^r), -\frac{1}{\sigma}\nabla f(x^r)\rangle + \frac{\sigma}{2}\cdot\frac{1}{\sigma^2}\|\nabla f(x^r)\|^2$

thus $f(\mathbf{x}^*) \geq f(\mathbf{x}^r) + \langle \nabla f(\mathbf{x}^r), \mathbf{x}^* - \mathbf{x}^r \rangle + \frac{\sigma}{2}\|\mathbf{x}^r - \mathbf{x}^*\|^2 \qquad (2)$

$\geq f(\mathbf{x}^r) - \frac{1}{2\sigma}\|\nabla f(x^r)\|^2 \qquad (3)$

- **Physical meaning**: if $\|\nabla f(x^r)\|$ is small, then $f(\mathbf{x}^r) - f^*$ is small
- Local error decides global error.



GD for SC Function: Combine 2 Steps

- Step 1 tells us how much descent we have at each step
- Step 2 tells us how close we are to the global min
- Combing the previous two steps, we have

$$\begin{aligned} f(\mathbf{x}^{r+1}) - f(x^*) &\overset{\text{Step 1}}{\leq} f(\mathbf{x}^r) - f(\mathbf{x}^*) - \frac{1}{2L}\|\nabla f(\mathbf{x}^r)\|^2 \\ &\overset{\text{Step 2}}{\leq} f(\mathbf{x}^r) - f(\mathbf{x}^*) - \frac{\sigma}{L}(f(\mathbf{x}^r) - f(\mathbf{x}^*)) \end{aligned}$$

- Rearranging terms, we have

$$e(\mathbf{x}^{r+1}) \leq (1 - \frac{\sigma}{L})e(\mathbf{x}^r) := \beta e(\mathbf{x}^r)$$

$e(\mathbf{x}^{r+1}) = f(\mathbf{x}^{r+1}) - f^*$

$e(\mathbf{x}^r) = f(\mathbf{x}^r) - f^*$

Summary of the Proof

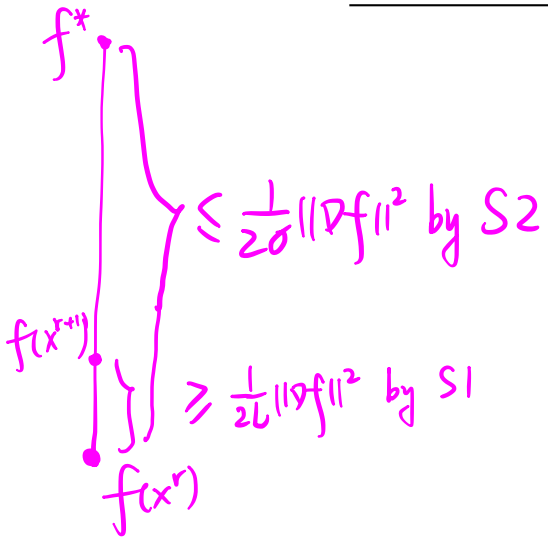
- **Summary of proof**: Two steps:
  1. S1: Sufficient Descent (the decrease achieved after each iteration):

$$f(\mathbf{x}^{r+1}) - f(\mathbf{x}^r) \leq -\frac{1}{2L}\|\nabla f(x^r)\|^2$$

$$\text{or } f(x^r) - f(x^{r+1}) \geqslant \frac{1}{2L}\|\nabla f(x^r)\|^2.$$

  2. S2: Estimating cost-to-go (how far away we are from the optimal):

$$f(\mathbf{x}^r) - f(\mathbf{x}^*) \leq \frac{1}{2\sigma}\|\nabla f(x^r)\|^2.$$



Result for Strongly Convex Function

Proposition 3b: Suppose f is a strongly convex function, and

$$\sigma I \preceq \nabla^2 f(x) \preceq L I,$$

Consider

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}).$$

Suppose GD with stepsize $1/L$ generates a sequence $\mathbf{x}^r$, then

$$\frac{e^r}{e^0} = \frac{f(\mathbf{x}^r) - f*}{f(\mathbf{x}^0) - f*} \leq \left(1 - \frac{1}{\kappa}\right)^r.$$

where the condition number $\kappa = \frac{L}{\sigma} = \frac{\text{max eigenvalues of all Hessians}}{\text{min eigenvalues of all Hessians}}$

- **Remark 1:** The assumption can be weakend to continuously differentiable function with L-Lipschitz gradient and is μ-strongly convex.
- Remark 2: Can prove iterates $\mathbf{x}^r$ converge with same speed.
- Remark 3: Best stepsize is $2/(L+\mu)$; see textbook or Nesterov's notes.

More General Cases (convex)

Proposition 3b: Suppose f is a convex function with Lipschitz gradient, i.e.,

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x-y\|, \quad \forall x, y \in \mathbb{R}^n.$$

Consider

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}).$$

Suppose GD with stepsize $1/L$ generates a sequence $\mathbf{x}^r$, then

$$e^r = f(\mathbf{x}^r) - f* \le \frac{2L}{r}\|\mathbf{x}^0 - \mathbf{x}^*\|^2.$$

- **Remark 1:** This is called "sublinear rate" $O(1/r)$.

  Linear rate: $1, 0.1, 0.01, \ldots$

  Sublinear rate: $1, 1/2, 1/3, \ldots, 1/10, \ldots, ...$

- **Remark 2:** For most convex problems you see, they achieve linear rate [Luo, Tseng'92]; maybe a homework problem

Nonconvex Problems?

- **First issue**:

  $f(\mathbf{x}^r) - f(\mathbf{x}^*)$ is not a good measure: it may not achieve zero.

- Typical measure: $\|\nabla f(\mathbf{x}^r)\| \to 0$.

- There are results on sublinear rate of GD, under Lipschitz gradient assumption.

- They are not relevant for many machine learning problems though...

Nonconvex Problems?

- **First issue**:

  $f(\mathbf{x}^r) - f(\mathbf{x}^*)$ is not a good measure: it may not achieve zero.

- Typical measure: $\|\nabla f(\mathbf{x}^r)\| \to 0$.

- There are results on sublinear rate of GD, under Lipschitz gradient assumption.

- They are not relevant for many machine learning problems though...

Conclusion of This Week

- How to select the constant stepsize for gradient descent (inverse proportional to the curvature)
- How to analyze gradient descent
- Practical performance of GD
- Analyze its convergence rate, and its relationship with the condition number and data preprocessing.