IE510 Applied Nonlinear Programming

Lecture 4: Case Study: Logistic Regression

Ruoyu Sun

Feb 6, 2018

Survey Results and Course Plan

Thanks everyone for participating in the 4-week survey.

I'll first discuss the survey results and some of the suggestions.

In total 18 responses (till Monday morning). About 80% of all registered students.

Q1: The pace of the course is:

62%: just right; 33%: somewhat fast, 5%: somewhat slow.

Response: I'll try to be slightly slower during technical parts.

Survey Results and Course Plan

Q2: The difficulty level of the course:

40%: just right; 40%: somewhat difficult; 20%: too difficult.

In addition, a few students mentioned homeworks are too difficult.

Response: I'll try to make the course contents easier; and homeworks easier.

- I'll put difficult questions in bonus questions
- Class practice is a way to let me know whether you are following

Survey Results and Course Plan

Q3: What do you want to learn more?
Top 2 answers:

94%: specific optimization problems in machine learning and statistics (e.g. logistic regression, deep learning, EM, SVM)

67%: practical aspects (e.g. how to decide whether it converges, how to pick parameters, various subtle issues in practice)

Less popular:

22%: complicated or fancy algorithms (e.g. conjugate gradient, BFGS, SVRG, trust region)

28%: theoretical analysis (e.g. analysis of 2-dim quadratic case)

Response: I'll discuss logistic regression and some subtle practical issues today. Maybe more ML/stats applications later.

- Note: There is a trade-off. More applications mean fewer algorithms to cover.

Some Suggestions and Projects

- Most popular suggestions: more examples, less homework.
- Midterm time
  - Tentative date: 03/27 Tuesday (after spring break), in-class exam
- Discussion on possible project ideas
  - See the course website "Projects" page. I list some project ideas; and will add more later. You can use your own too.
  - Form 1: Find an application (e.g. Kaggle competition, CapsuleNet). Apply algorithms you learned to solve it.
    - Note: highlight the optimization part
  - Form 2: Extension of the homework. Study a simple problem, analyze various algorithms.
    - E.g. Does momentum work for 2-layer neural-nets? Study it thoroughly

Projects

- Form 3: Read a paper. Implement its algorithm for a specific problem.
  - e.g. apply SVRG to solve neural-nets or RNN
- Form 4: Read a paper with some theory. Write a reading report, and possibly experiments.
  - e.g. read “insufficiency of momentum in ML” ICLR18
- Form 5: Write a blog article like ”why momentum really works”, submit to Distill
  - e.g. why coordinate descent really works
- Form 6: up to you

Questions for Last Week

- **Q1**: Physical interpretation of momentum method?
- **Q2**: Your colleague implemented GD and found it takes one day to run the algorithm. He/she already tuned the stepsize for a while, but the improvement is little.

  You suggest using momentum. He/she asked why. What will you say?

Last Week: Convergence Rate of Momentum

- **Proposition 3.1**: Suppose f is a strongly convex quadratic function, and consider

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

Solving the problem by GD with momentum (heavy ball method) with stepsize α and momentum coefficient β.

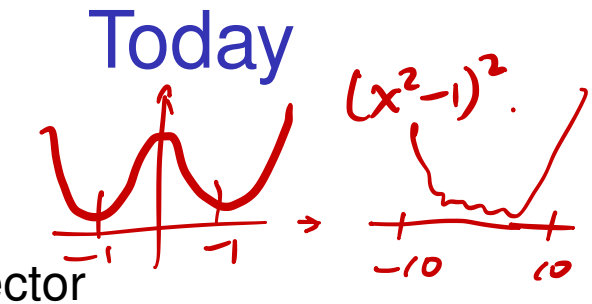
  - The iterates converges at a rate $1 - 1/\sqrt{\kappa}$ if

$$\alpha = \frac{1}{\lambda_{\max}}, \beta = (1 - \sqrt{\alpha\lambda_{\min}})^2,$$

  - The iterates converges at a rate $1 - 2/(\sqrt{\kappa}+1)$ if

$$\alpha = \left(\frac{2}{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}}\right)^2, \beta = (1 - \sqrt{\alpha\lambda_{\min}})^2,$$

- **Practical Choice**: Pick β slightly smaller than 1, and α as large as possible (still converging).



- Some common issues of homework 1:
  - Problem 6(i): figure should reflect "nonconvex"
  - Problem 6(ii). $Ax = \|x\|^2 x$ implies x is eigenvector

$Ax = \lambda x$

- **Today**: (Theory-Guided) Case study of logistic regression (LR), mostly using what you have learned
  - Note: examples may be (much) more complicated than theory
- After today's course, you will be able to
  - identify a few optimization issues of LR
  - explain the algorithm behavior of simple LR using existing theory
- More advanced goal: I wish you could
  - learn how to analyze an optimization problem step-by-step
  - apply this analysis framework to other complicated problems (e.g. in projects)

Outline

Review of Theories You Learned

- Optimality conditions: check local-min/global-min
  - Convexity is crucial
- When does GD "converge"? (every limit point is stationary)
  - Stepsize choice: $< 2/L$ with L-Lipschitz gradient
  - line search
- Convergence rate? $\sqrt{\kappa}$ for strongly convex problems

Apply Theory?

- We will see: it is nontrivial to apply these theories to even a simple problem
- From now on, image yourself being an data scientist/engineer/etc., trying to solve a problem.
  - Step 1: specify the problem
  - Step 2: write or modify code to solve it
  - Step 3: Observe algorithm behavior, good or bad? expected or not?
  - Step 4: If not good/expected, go to Step 2
- Keep in mind: as an optimizer, your goal is to understand all behavior (if possible)
  - The goal of a regular data scientist is probably "making it work"

Apply Theory?

- We will see: it is nontrivial to apply these theories to even a simple problem
- From now on, image yourself being an data scientist/engineer/etc., trying to solve a problem.
  - Step 1: specify the problem
  - Step 2: write or modify code to solve it
  - Step 3: Observe algorithm behavior, good or bad? expected or not?
  - Step 4: If not good/expected, go to Step 2
- Keep in mind: as an optimizer, your goal is to understand all behavior (if possible)
  - The goal of a regular data scientist is probably "making it work"

Apply Theory?

- We will see: it is nontrivial to apply these theories to even a simple problem
- From now on, image yourself being an data scientist/engineer/etc., trying to solve a problem.
  - Step 1: specify the problem
  - Step 2: write or modify code to solve it
  - Step 3: Observe algorithm behavior, good or bad? expected or not?
  - Step 4: If not good/expected, go to Step 2
- Keep in mind: as an optimizer, your goal is to understand all behavior (if possible)
  - The goal of a regular data scientist is probably “making it work”

Warm-up: 1-dim linear regression (basketball problem)

Before discussing logistic regression, we do a warm-up for linear regression.

- Set a baseline for the behavior
- Try to understand everything you observe

Problem setup: $\min_w (p - cw)^2$.

- Problem parameters: $p = 1, c = 5$.

Algorithm: $w^+ = w - \alpha \nabla f(w)$.

- Initialization: $w^0 = 5$. (or randn(1,1); let's do fixed first)
- Max-iteration: MaxIte = 50. (can change)

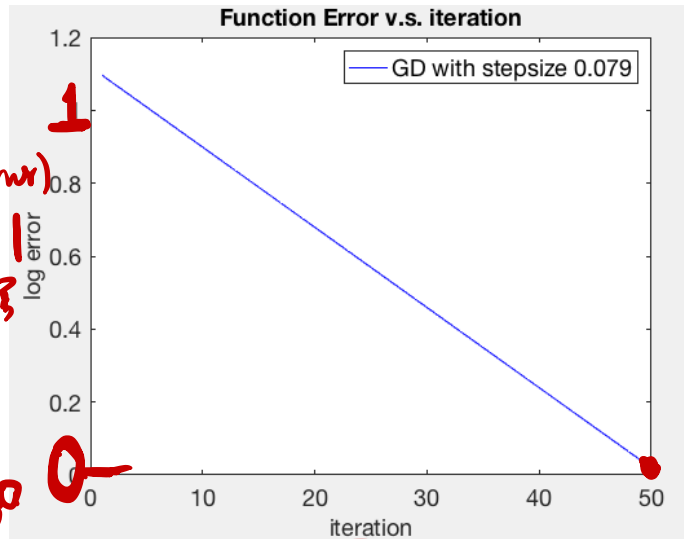
Tuning parameter: α.

1-dim regression: threshold

Start from

- $\alpha = 1$, diverge
- $\alpha = 0.1$, diverge
- $\alpha = 0.01$, converge.

You want to know the threshold, and find the threshold is 0.08.



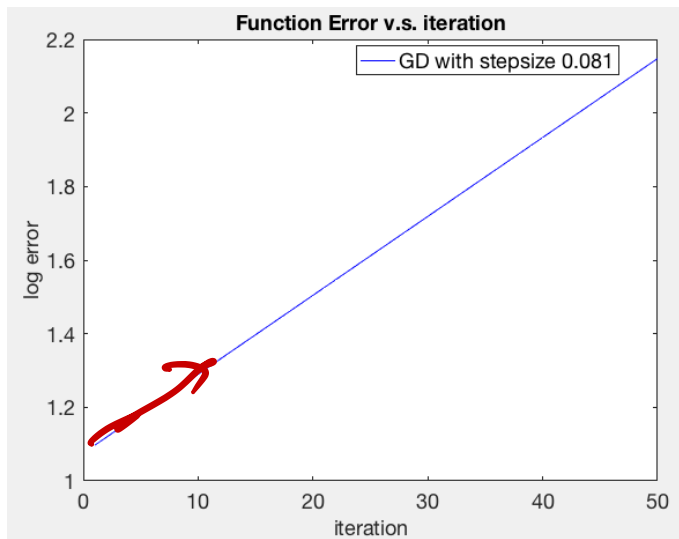
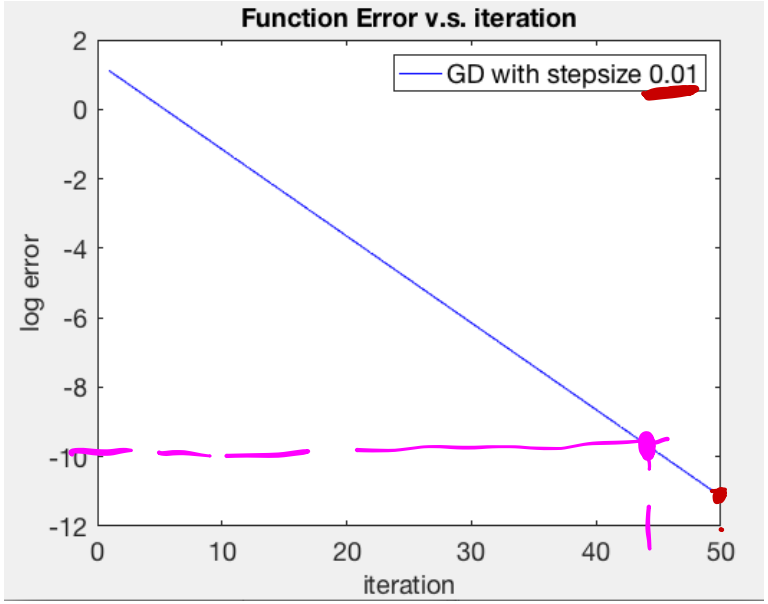


Figure: Left: stepsize 0.079; Right: stepsize 0.081

By theory, the threshold of $\alpha = 2/L = 2/c^2 = 0.08$.

1-dim regression: speed

You want to find α so it converges fast.



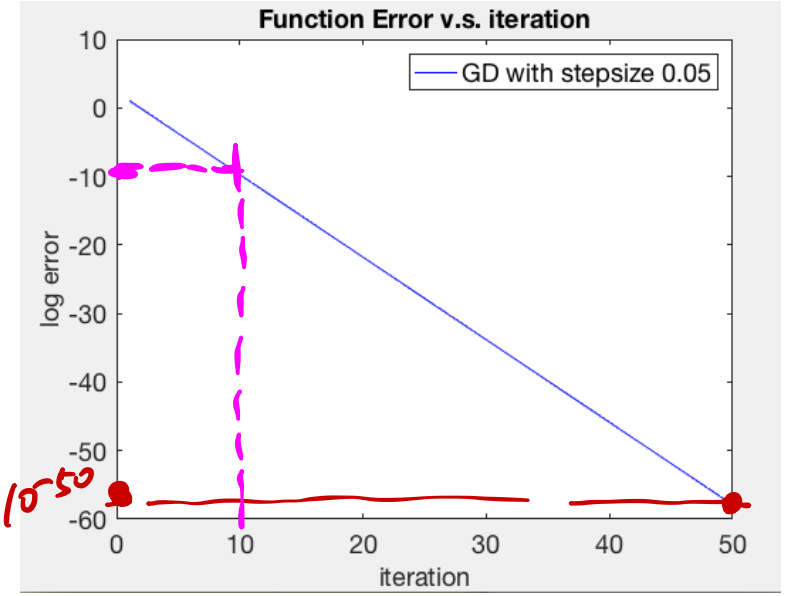


Figure: Left: stepsize 0.01; Right: stepsize 0.05

Stepsize affects the convergence speed a lot.

$\alpha = 0.01$, to achieve 10^{-10}, need 44 iterations.

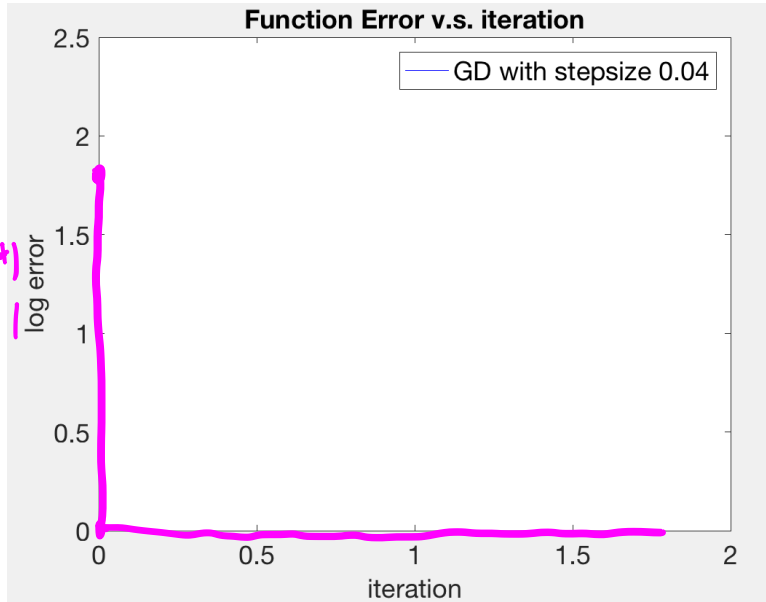
$\alpha = 0.05$, to achieve 10^{-10}, need 10 iterations.

What is the optimal stepsize?

1-dim regression: best stepsize

In theory, the best stepsize is $2/(L+\mu) = 1/L = 0.04$. (since in 1-dim, $L = \mu$).

Let's check $\alpha = 0.04 = 1/L$ and $\alpha = 0.08 = 2/L$.



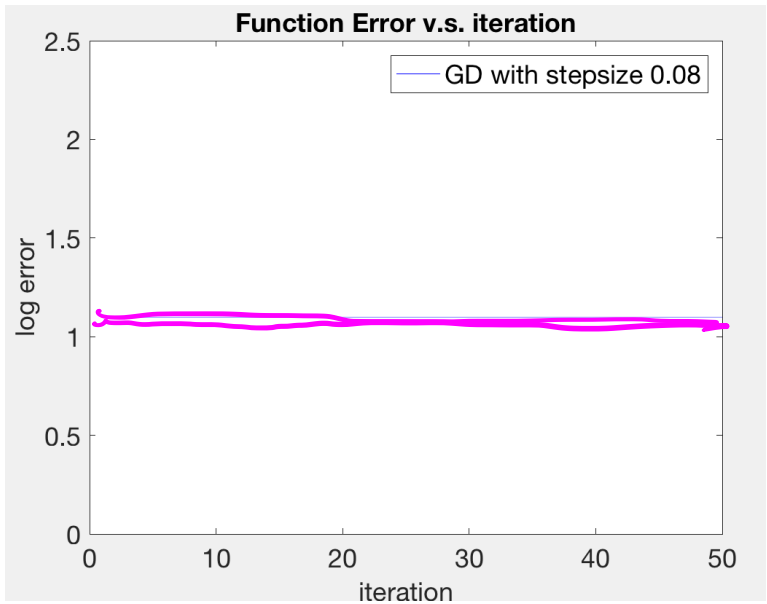


Figure: Left: stepsize 0.04: Right: stepsize 0.08

$\alpha = 0.04$ represents a general case: algorithm converges too fast.

Check numerical values: $\log_{10} f(x^k) = 1.069, -\text{Inf}, -\text{Inf}, \ldots$

1-dim regression: summary

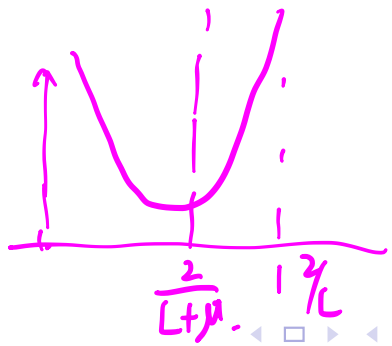
Summary of 1-dim linear regression: stepsize affects convergence speed.

For convergence, two phases:

- Phase 1: $\alpha > 0.08 = 2/L$. Diverge.
- Phase 2: $0 < \alpha < 0.08 = 2/L$. Converge.

For convergence speed, two phases: $(0, \frac{2}{L+\mu})$

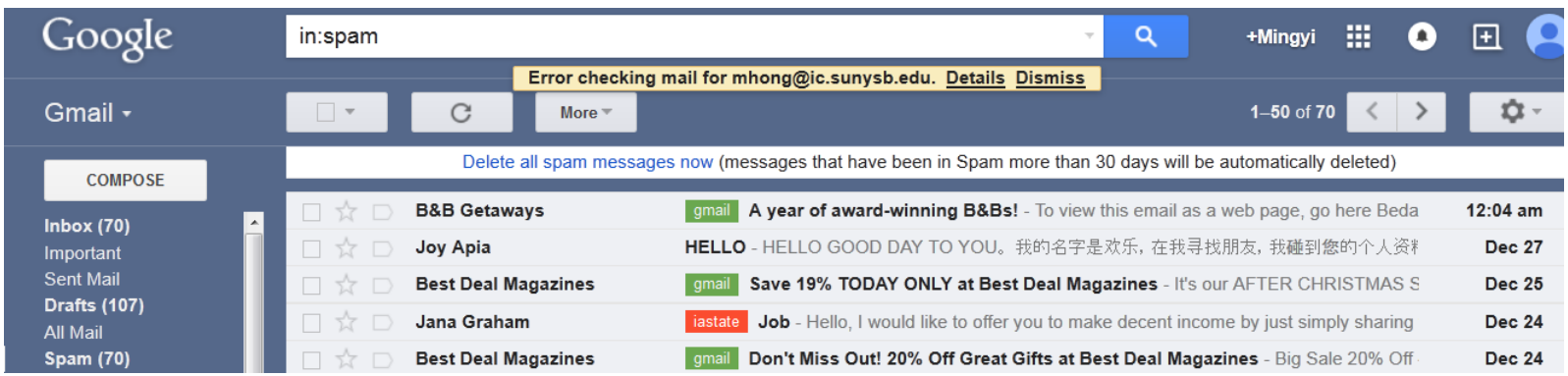
- Phase 1: $0 < \alpha < 0.04 = 2/(L+\mu)$ bigger stepsize implies faster
- Phase 2: $2/(L+\mu) = 0.04 < \alpha < 0.08 = 2/L$, bigger stepsize imples slower.



Classification Problem

- Classification: one of the key problems of machine learning
- Example: Spam email.
- Every email services have automatic spam detection mechanisms
- The basic task is the following
  1. Given an excerpt of an email, represented by a vector $\mathbf{x}_i$
  2. Ask the question whether it is a spam or not ($y_i = +1, -1$)

(1,0)



Example: Image Classification

- More complicated examples: image classification

ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

4

Linear Classification

Regression: $(x_i, y_i), i = 1, \ldots, n$.

- Find w such that dist$(w^T x_i, y_i)$ is small.
- Choose square loss function

$$\min_w \sum_i \text{dist}(w^T x_i, y_i) = \sum_i \|w^T x_i - y_i\|^2.$$

Classification: $(x_i, y_i), i = 1, \ldots, n$, where $y_i \in \{1, -1\}$.

- Find w such that dist$(w^T x_i, y_i)$ is small.
- First, choose 0-1 loss function

$$\max_w \sum_i \text{dist}(w^T x_i, y_i) = \sum_i \text{sign}(y_i w^T x_i).$$

- Second, choose a loss function $\log(1 + \exp(-y\hat{y}))$

$$\min_w \sum_i \text{dist}(w^T x_i, y_i) = \sum_i \log(1 + \exp(-y_i w^T x_i)).$$

Comparison of Two Loss Functions

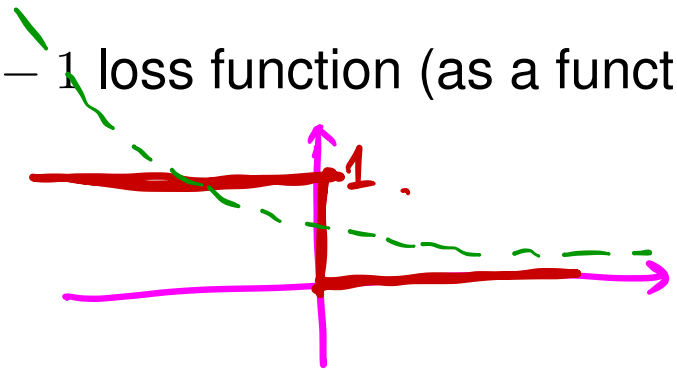
$y_i = 1$,

$w^T x_i = 0.5$, 1.5 $\rightarrow$ same

$-0.5 \rightarrow$ not same.

Problem: determine the "distance" between the prediction $\hat{y} = w^T x$ and true label $y \in \{1, -1\}$.

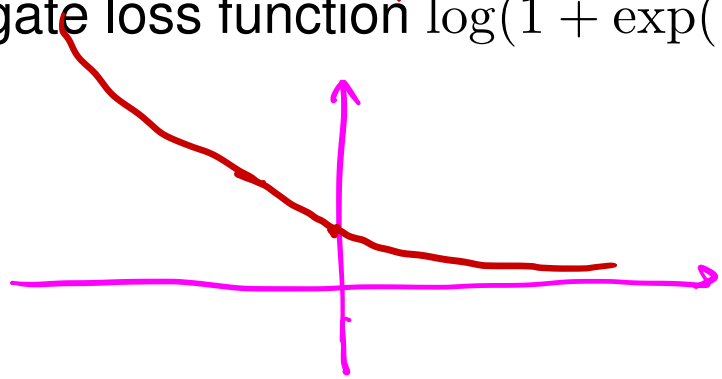
Ideal $0-1$ loss function (as a function of $y\hat{y}$):



$z = y\hat{y}$

"sign"$(z) = \begin{cases} 0, & \text{if } y\hat{y} \text{ same sign} \\ 1 & \text{differs} \end{cases}$

Surrogate loss function $\log(1 + \exp(-z))$ where $z = y\hat{y}$



$z \to \infty$, $h(z) = 0$?

$z \to -\infty$, $h(z) = \infty$.

$z = 0$, $h(z) = \log(2)$.

Side: Popularity of Logistic Regression

Google search ”logistic regression”: about 10 million results.

“There is a perception that LR is slow, unstable, and unsuitable for large learning or classification tasks” –Komarak’04 ”Logistic Regression for Data Mining and High- Dimensional Classification”

But It is still very popular nowadays:

- deep learning is built on LR
- very commonly used in IT companies like Facebook, Google, etc.

There are many tutorials on LR.

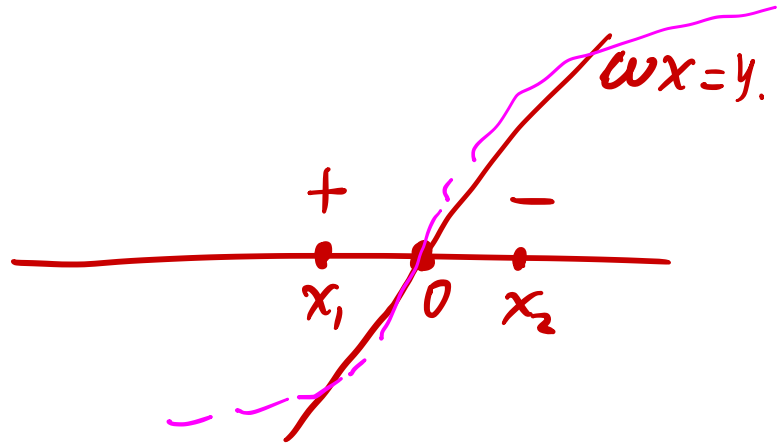
- One is Stanford CS231n course webpage
- Or you can check a detailed tutorial at http://www.dataschool.io/guide-to-logistic-regression/

We focus on optimization issues in this course.

Simplest problem: 1-dim case

- **Problem setup**:
  - Two data points in $\mathbb{R}$, x_1 and x_2.
  - Labels $y_1 = 1$, $y_2 = -1$.

  Graph:



- Remark: typically, one would use $wx_i + b$ to classify; but for simplicity, let's just use wx_i for now.

- **Objective function**

  - Find stationary pt directly. $f'(w)=0$.

$$f(w) = \frac{1}{2}\log(1+\exp(-wx_1)) + \frac{1}{2}\log(1+\exp(wx_2)).$$

Algorithm: $x^+ = x - \alpha\nabla f(x)$.

- Initialization: $w^0 = 5$. (or randn(1,1); let's do fixed first)
- Max-iteration: MaxIte = 20. (can change) 200.

Tuning parameter: α.

Preliminary Analysis

Before running the algorithm, let's do some simple calculations.

- First, is the problem convex?
- Obtain $f''(w) = x_1^2 \frac{e^{wx_1}}{(1+e^{wx_1})^2} + x_2^2 \frac{e^{wx_2}}{(1+e^{wx_2})^2} > 0.$
- Second, is the problem _____?

Preliminary Analysis

Before running the algorithm, let's do some simple calculations.

- First, is the problem convex?
- Obtain $f''(w) = x_1^2 \frac{e^{wx_1}}{(1+e^{wx_1})^2} + x_2^2 \frac{e^{wx_2}}{(1+e^{wx_2})^2} > 0.$
- Second, is the problem L-Lipschitz cts? $|f(x) - f(y)| \le L|x-y|$.

$$f(w) = \frac{1}{2}\log(1+\exp(-wx_1)) + \frac{1}{2}\log(1+\exp(wx_2)).$$ $x_1, x_2, w \in \mathbb{R}$.

Claim: Yes. If $x_1 = 1$, $x_2 = 2$, then $L < 5$.

First experiment

Data Setting 1: Set $x_1 = 1, x_2 = 2$;

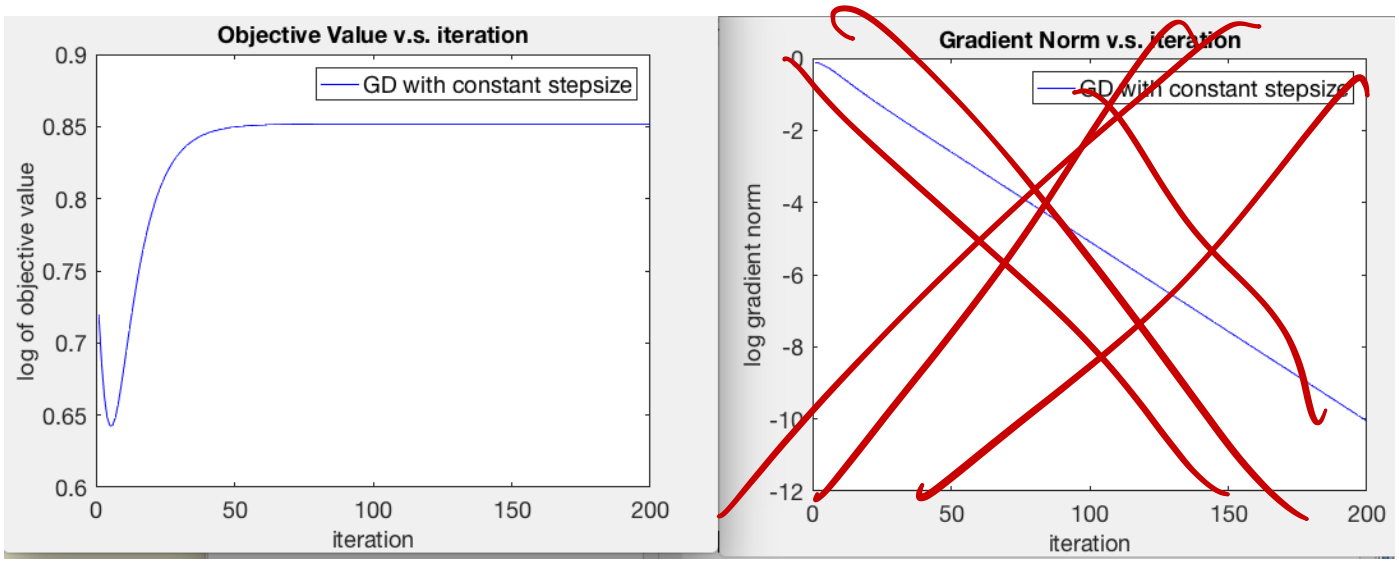


Figure: Function Value. $\alpha = 0.2$ for 1-dim logistic regression

- Why increasing?

  BUG! Used $\log(1 + \exp(y_i w^T x_i))$, missing a " $-$ ".

- Seriously? Why do you show a bug? I'll explain later.

First experiment

Data Setting 1: Set $x_1 = 1, x_2 = 2$.

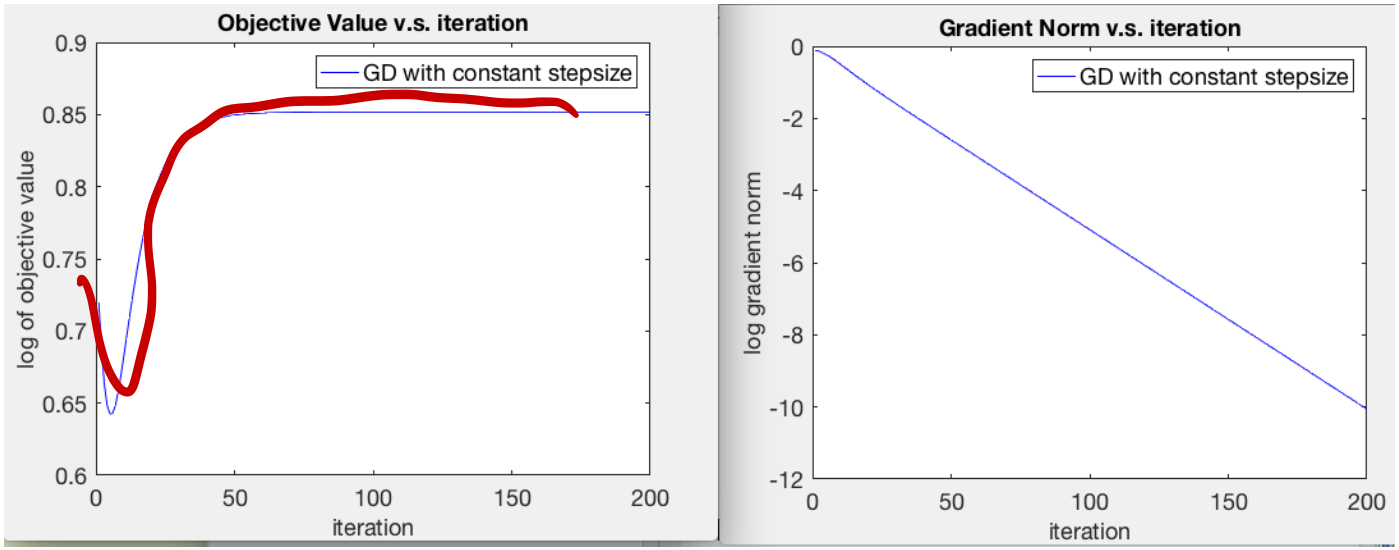


Figure: Function Value. $\alpha = 0.2$ for 1-dim logistic regression

- Why increasing?

  BUG! Used $\log(1 + \exp(y_i w^T x_i))$, missing a " $-$ ".

- Seriously? Why do you show a bug? I'll explain later.

First experiment

Data Setting 1: Set $x_1 = 1, x_2 = 2$.

Pick $\alpha = 0.2$. Figure of the algorithm for 200 iterations:

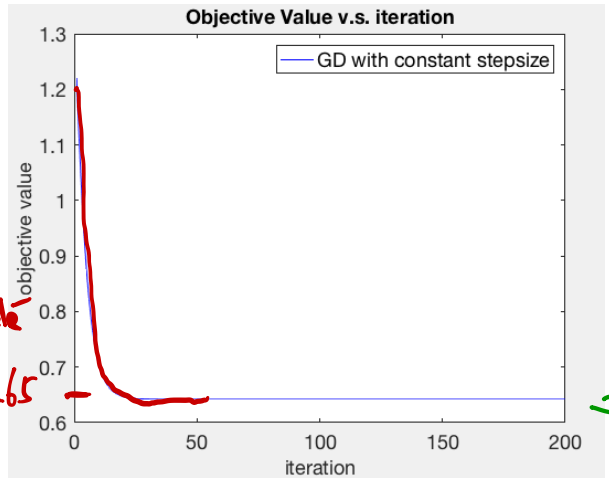


Figure: Function Value. $\alpha = 0.2$ for 1-dim logistic regression

Question: Has it already converged?

Issue: in theory, use $f(x^r) - f^*$ to measure error,

but in practice don't know

Performance Metric and Stopping Criteria

One choice: $f(x^r) - f(x^{r-1})$; last few iterations of $f(x^r)$:

$0.6420, 0.6420, 0.6420, \ldots$.

Drawbacks

- cannot see from figure;
- what meaning?

$f - f^* \to 0$

A better choice: gradient norm $\|\nabla f(x^r)\| \to 0$.

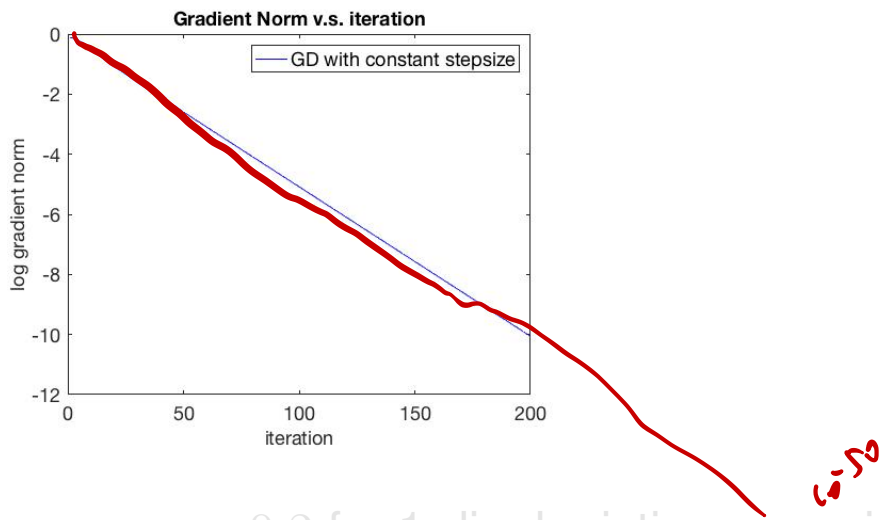


Figure: Gradient norm. $\alpha = 0.2$ for 1-dim logistic regression

Performance Metric and Stopping Criteria

One choice: $f(x^r) - f(x^{r-1})$; last few iterations of $f(x^r)$:

$0.6420, 0.6420, 0.6420, \ldots$

Drawbacks

- cannot see from figure;
- what meaning?

A better choice: gradient norm $\|\nabla f(x^r)\|$

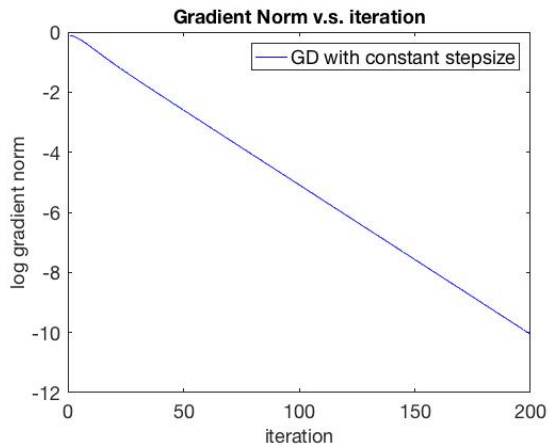


Figure: Gradient norm. $\alpha = 0.2$ for 1-dim logistic regression

Stopping Criteria

iterates. x^r v.s w^r.

Two metric: $\nabla f(x^r)$ and $|f(x^r) - f(x^{r+1})|$.

Can stop the algorithm when either of the two holds:

1. when $\|\nabla f(x^r)\| \leq \epsilon$
2. when $|f(x^r) - f(x^{r+1})| \leq \epsilon$
3. when $\|x^r - x^{r+1}\| \leq \epsilon$ for GD, $|x^{r+1} - x^r| = \alpha \|\nabla f(x^r)\|$.

Typical choice of error: $\epsilon = 10^{-3}$ (low precision), or $\epsilon = 10^{-10}$ (high precision)

Practice: relation between the three criteria? (practice).

End of 02/13 Tuesday Lecture.

02/15. Thursday.

Midterm time: Mar 27 Tues.

5+ people unavailable on Mar 13-15.

Homework 2: Extend 1 day.

e.g. theory stepsize is 10^{-5},
0.1, 0.01, 0.001, 0.0001,

Review Questions:

(i) For convex quadratics, what are the 2 phases?

(ii) How to measure performance?

error: $x - x^*$

$\|\nabla f(x^k)\|$

$\|x^k - x^{k-1}\|$. or $\|f(x^k) - f(x^{k-1})\|$

What else can these metric do?

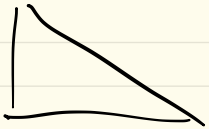
(iii) What're the two things to check before clicking "run"?

- Stopping criterion - initial points. - "Bugs". - Numerical error. | Has a solution or not. | * Convexity * Lipschitz const.

Summary Till Now (in this Lecture).

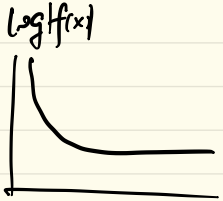
1) Linear regression - 1-dim.

Clear picture: 3 phases.



2) Logistic regression.

Data setting 1:



Be prepared for 2-minute thesis at the end.

Second experiment

Data Setting 2: Set $x_1 = 1, x_2 = -2.$

Pick $\alpha = 0.2$. Figure of the algorithm for 200 iterations (~~use~~ no log scale):

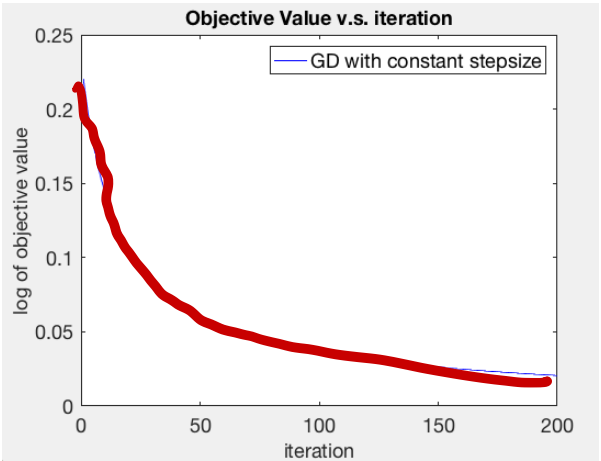


Figure: Function Value. $\alpha = 0.2$ for 1-dim logistic regression

Question: Has it "converged"?

Converged or Not?

Let's run 20k iterations. That should be enough, uhh?

Check log of function values, and gradient norms

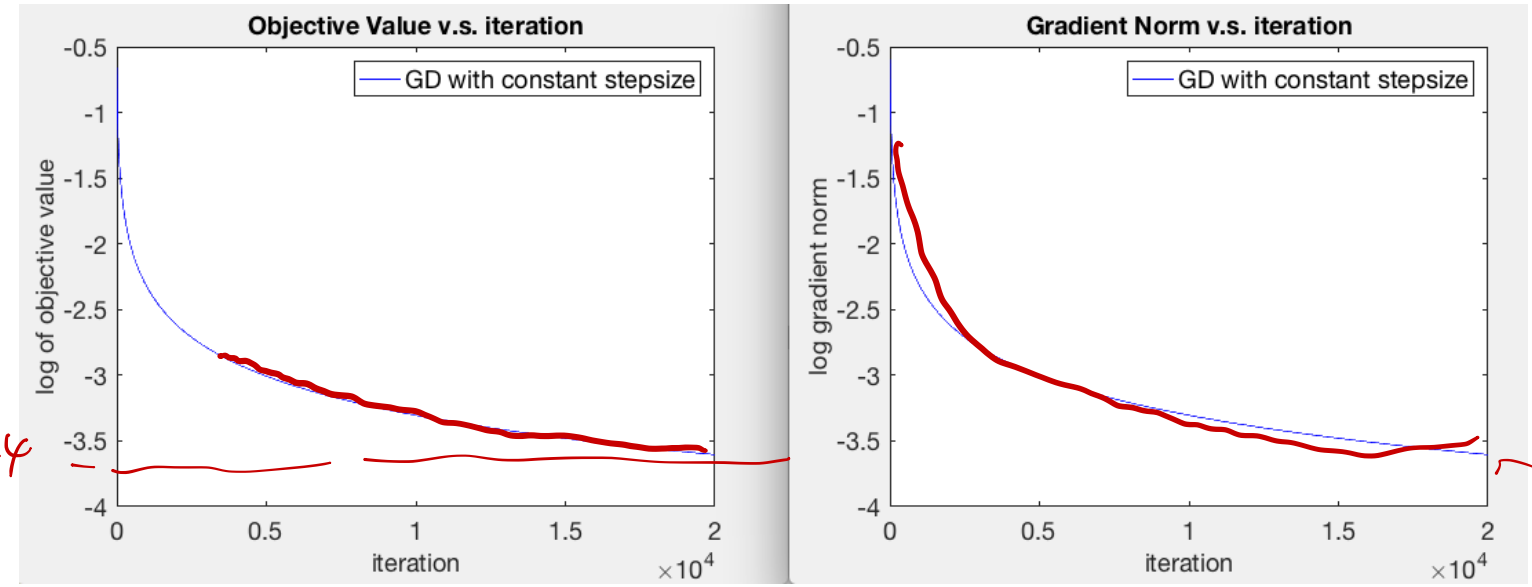


Figure: Left: Log of Function Value. Right: Gradient Norm. $20,000$ iterations

Again, has it "converged"? Or, should we count it as "converged"?

- Yes, since the loss function is small?
- No, since the gradient norm is still not small enough?

Discussion: Why This Happens

So strange... In all previous cases (linear regression, or Data Setting 1), either diverge or converge in the sense that "error" $< 10^{-10}$.

Re-examine the theory

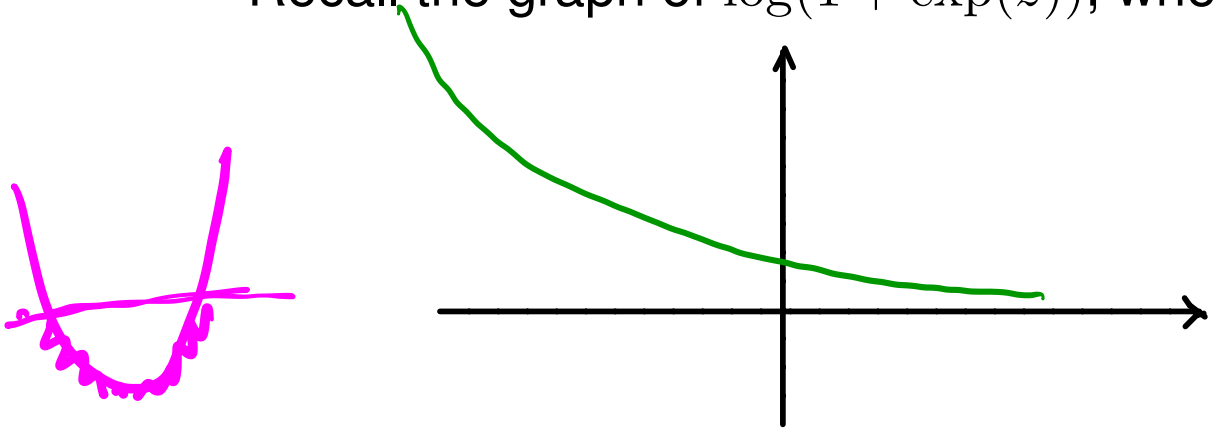
- If $\alpha < 2/L$, then every limit point is stationary point.
- If convex, then every stationary point is globally-optimal.

So, it should finally converge to a stationary point, which is globally optimal (right?). Maybe 20,000 iterations is not enough?

First Issue

First issue: Non-existence of global-min.

- Recall the graph of $\log(1+\exp(z))$, where $z = y\hat{y}$.



$0 = \min_z \log(1+\exp(z))$.

$\inf_z$

- **Observation**: Although strictly convex and lower bounded, it has no stationary point.

$\|\nabla f(w^k)\| \to 0, \ k \to \infty$.

- Iterates diverge: $\|w^k\|$ goes to ∞ to minimize $f(w)$

What about $f(w^k)$?

Vote: does $f(w^k)$ converge?

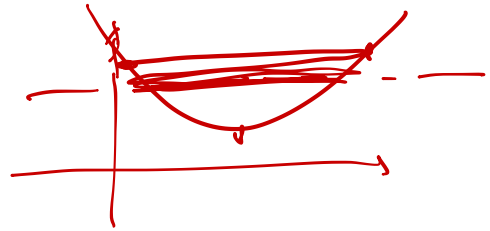
Question Does $f(w^k)$ converge ?

Yes, $f(w^k) \downarrow$ + lower bounded

$\Rightarrow f(w^k)$ converges

Does not imply $f(w^k) \to 0$.

Second Issue



- **Second Issue**: Convergence.
- No limit point... so we have nothing to say about "convergence"?
- But it seems that function values are converging to $0 = f^*$.
- **Claim:** Using GD with constant stepsize $\alpha < 2/L$ for this 1-diim LR problem, we have $f(w^r) \to f^* = \inf_w f(w)$.
  - Believe me, the proof is just 3 lines, as I've done in Lecture 2b. And it's almost the same as the proof of homework 2 problem 3(iv).

Second Issue

- **Second Issue**: Convergence.
- No limit point... so we have nothing to say about "convergence"?
- But it seems that function values are converging to $0 = f^*$.
- **Claim:** Using GD with constant stepsize $\alpha < 2/L$ for this 1-diim LR problem, we have $f(w^r) \to f^* = \inf_w f(w)$. $\therefore = 0$
  - Believe me, the proof is just 3 lines, as I've done in Lecture 2b. And it's almost the same as the proof of homework 2 problem 3(iv).

People claim GD converges to min value of LR:
cite Prop 1.1. in Bertsee's book.

Third Issue

- **Third Issue**: Convergence speed.
- Now we know $f(x^r)$ does converge to 0. But why is it so slow?
- First thought? Bug.

  Why do I insist on explaining this phenomenon?

  One reason: otherwise how do you know there is NO BUG?
- My story: Machine learning was a headache, when I first learned it
  - Often cannot make the algorithm work, and don't know why....
  - If you don't understand the algorithm, strange things can happen

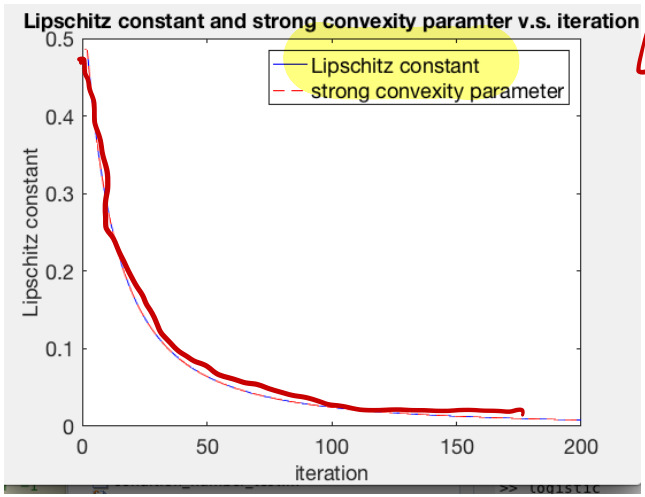
Third Issue: convergence speed

Prop: (strongly convex problems).
# of iterations $\leq K \log \frac{1}{\epsilon}$.

- Second thought: condition number too large.
- Compute condition number at every point, turn out to be... 1.
  - 1-dim problem, certainly at each point $\lambda_{\max} = \lambda_{\min}$
    $H = f''(w)$.
- What's wrong?

Third Issue: convergence speed

We compute $L(w^r)$ and $\mu(w^r)$ at all iterates:



Lipschitz $L(w)$, $L = \max_w L(w)$

$\mu(w)$, $\mu = \min_w \mu(w)$

Figure: L and μ of various iterates

Observation: $L(w^r) = \mu(w^r)$ is decreasing.

Definition of condition number: $\mu \leq f''(w) \leq L$ for all w, then $\kappa = \frac{L}{\mu}$.

E.g. $L(w^1) = 0.5$, $L(w^{200}) \approx 0.008$, then κ is at least $0.5/0.008 = 62.5$.

Non-strongly convex

Theoretically speaking, the condition number of f is $\frac{\max_x \lambda_{\max}(H(x))}{\min_x \lambda_{\min}(H(x))} \to \infty$.

So, the convergence rate result on κ does not apply here!

Then can we have convergence speed guarantee?

- for f with Lipschitz gradient, but no universal strong convexity parameter
- More general, for the following f:

$$0 \preceq \nabla f(w) \preceq LI.$$

Non-strongly-convex Case

Recall the following result in Lecture 2c.

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

$$\nabla^2 f(w) \leq LI, \ \forall w.$$

Consider

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

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

$$e^r = f(\mathbf{w}^r) - f(\hat{\mathbf{w}}) \leq \frac{2L}{r}\|\mathbf{w}^0 - \hat{\mathbf{w}}\|^2.$$

for any $\hat{\mathbf{w}}$.

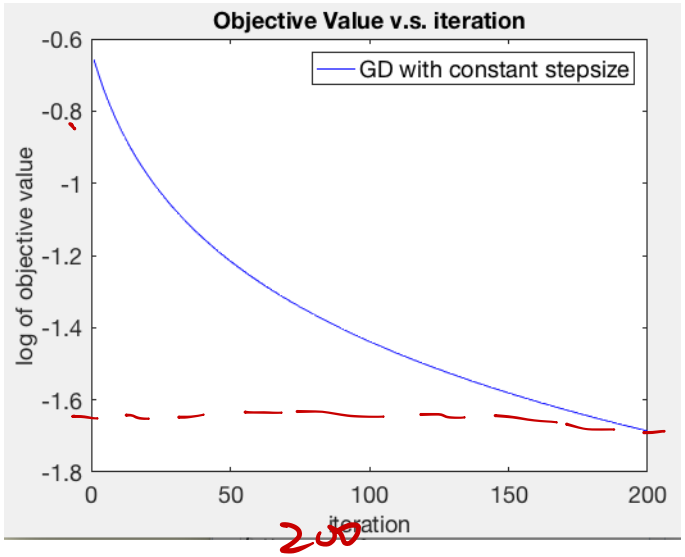
- **Remark:** This is called "sublinear rate" $O(1/r)$.

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

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

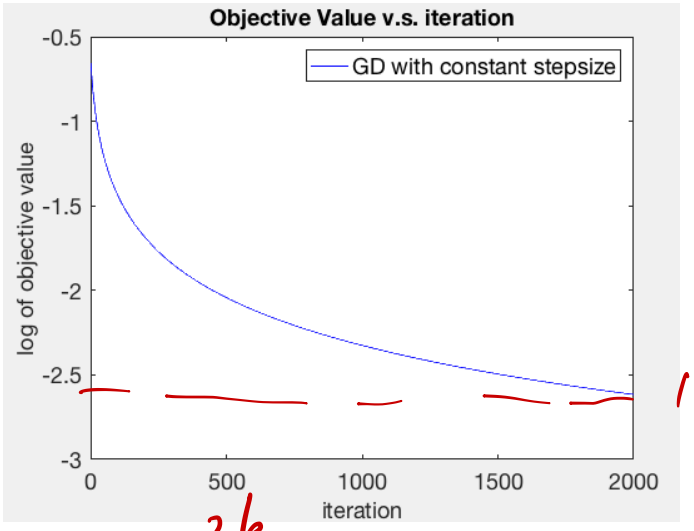
Verifying Sublinear Rate in Practice

Run for 200, 2k, 20k, 200k iterations.



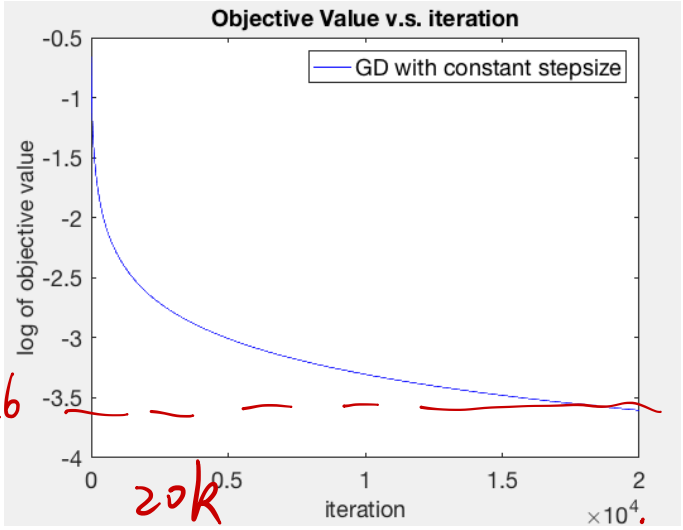
$10^{-1.7}$

200



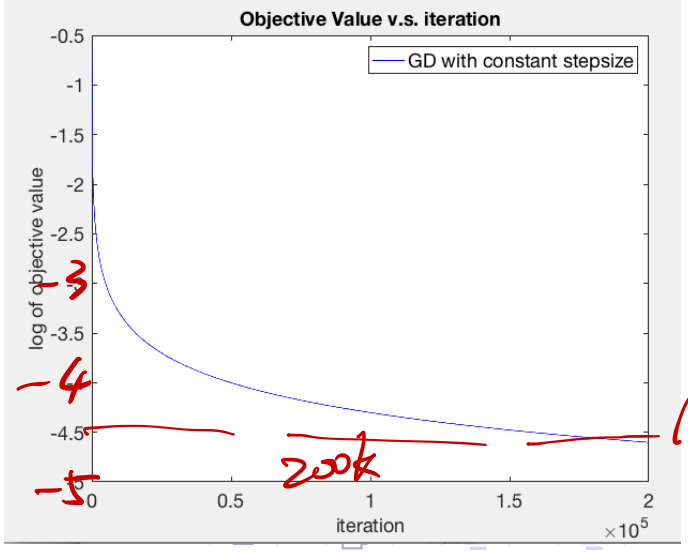
$10^{-2.6}$

2k



$10^{-3.6}$

20k



$10^{-4.6}$

200k

Verifying Sublinear Rate in Practice (cont'd)

- Collect the simulation results in a table:

# of Iterations	200	2k	20k	200k
error	$10^{-1.6}$	$10^{-2.6}$	$10^{-3.6}$	$10^{-4.6}$

Table: Error v.s. Iterations

- **Observation**: To reduce error to 1/10, need 10 times more iterations.
- **Theory**: error is $\frac{1}{r}C$, where $C = 2L\|\mathbf{w}^0 - \hat{\mathbf{w}}\|^2$.
- So the performance (almost) matches the theory! (finally)
- Yet another issue: what is $\hat{\mathbf{w}}$ in $\|\mathbf{w}^0 - \hat{\mathbf{w}}\|^2$?
  - Skip in class.
  - Not hard to resolve this issue... need a bit more work.

Why Two Data Settings Different

Why are Data Setting 1 $(1, 2)$ and Setting 2 $(1, -2)$ so different?

What if we use $(1, 5)$? or $(-0.3, -6)$?

like Setting 1 or Setting 2?

Two typical data settings:

- Separable case
- Non-separable case
- In separable case, w diverges, arrive at flat region!
- In non-separable case, w stays bounded, so essentially"strongly convex"!

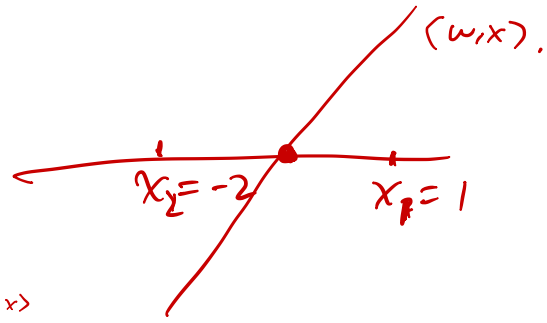
Question: Why care about iterates divergence? (objective small is enough)

- Because it may affect convergence rate a lot!

Why Two Data Settings Different

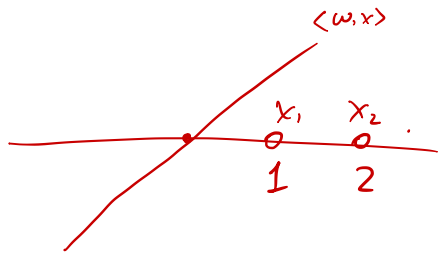
Why are Data Setting 1 $(1, 2)$ and Setting 2 $(1, -2)$ so different?

What if we use $(1, 5)$? or $(-0.3, -6)$?



Two typical data settings:

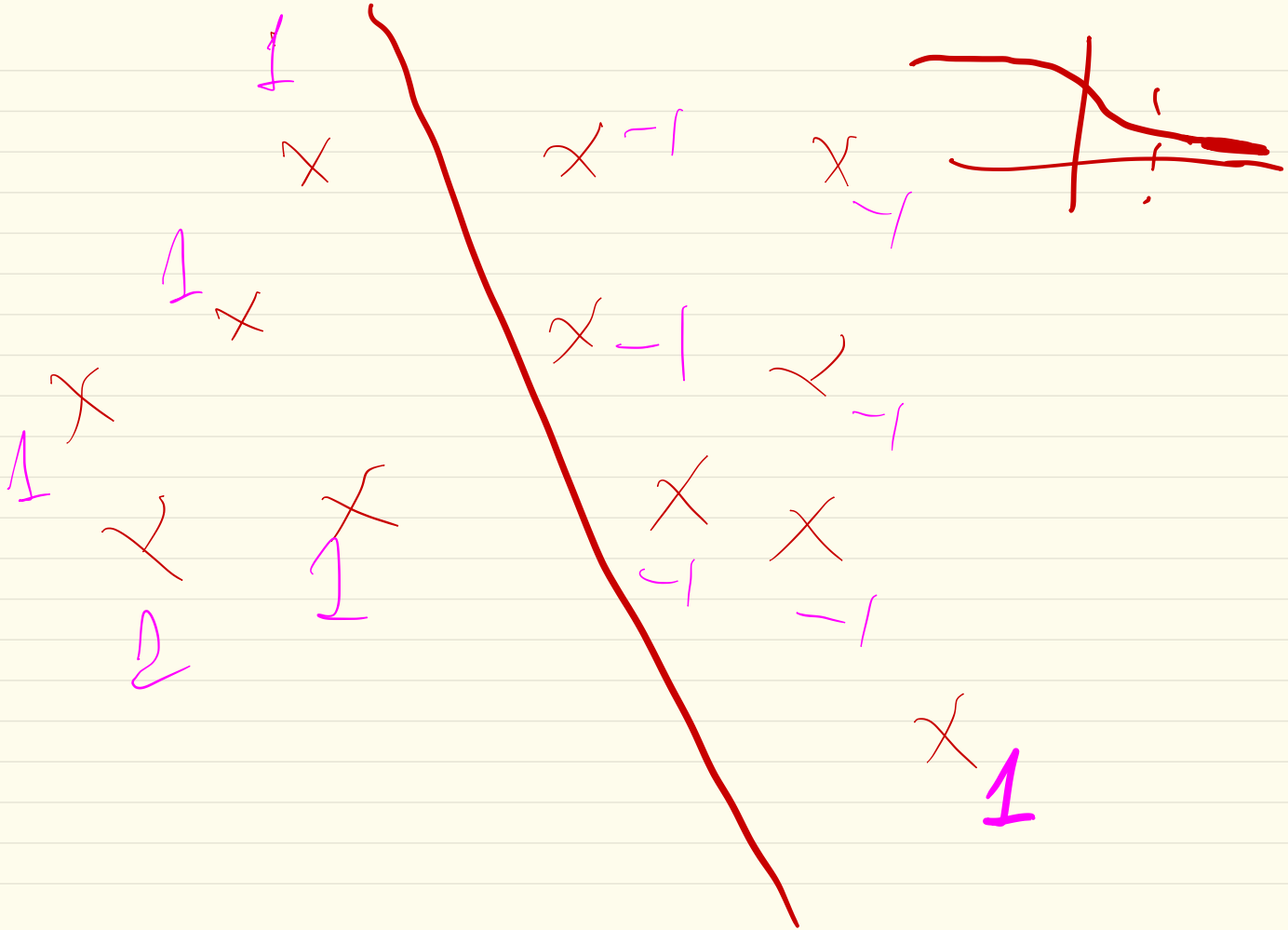
- Separable case
- Non-separable case



- In separable case, w diverges, arrive at flat region! $\inf_w f(w) = 0$.
- In non-separable case, w stays bounded, so essentially "strongly convex"!

Question: Why care about iterates divergence? (objective small is enough)

- Because it may affect convergence rate a lot!

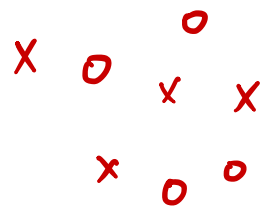


Summary of Results for 1-dim Logistic Regression

- Since $\log(1+e^{-z})$ is strictly convex, but NOT strongly convex, the worst-case convergence rate is sublinear, i.e., $O(1/r)$.
- Two convergence patterns:
  - Separable case: sublinear rate $O(1/r)$
  - Non-separable case: usually linear rate $O(\rho^r)$.

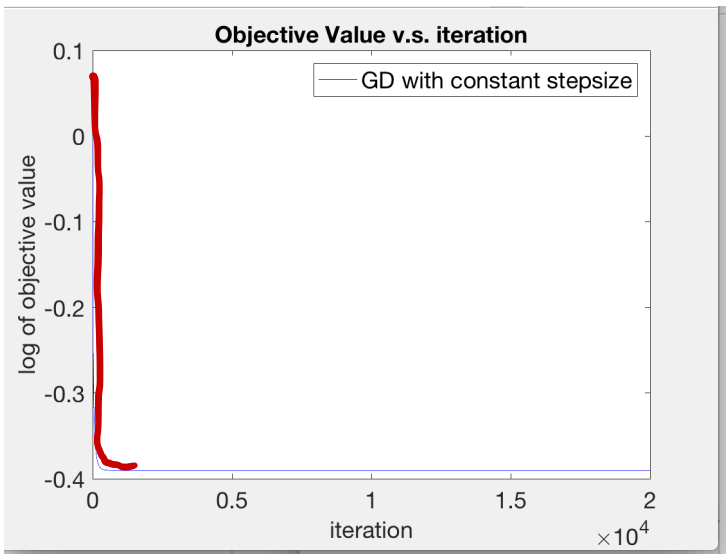
Extension: Higher-Dimensional Case?

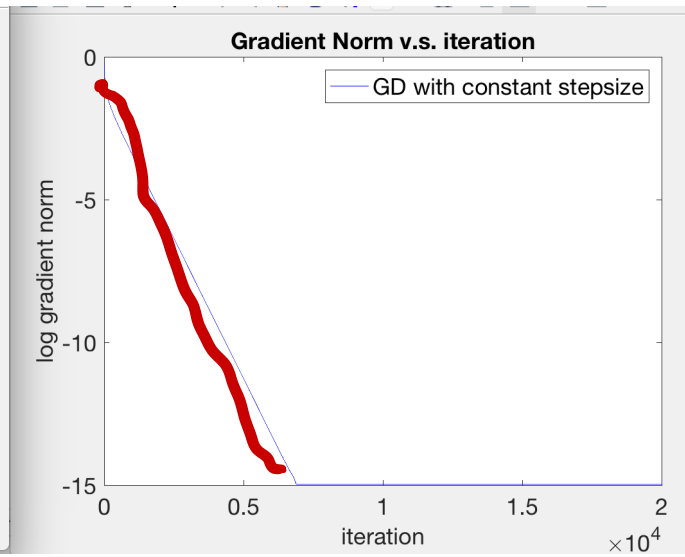
random labels: usually non-separable.



Data Setting 1: $n = 10, d = 5$.

- Random Ganssian data $x_1, \ldots, x_{10} \in \mathbb{R}^5$.
- Random labels y_i with prob. 1/2.





6000 iterations, $\|\nabla f(x)\| \simeq 10^{-15}$.

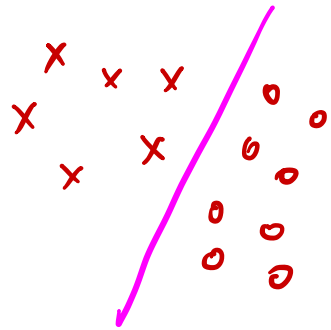
Explain?

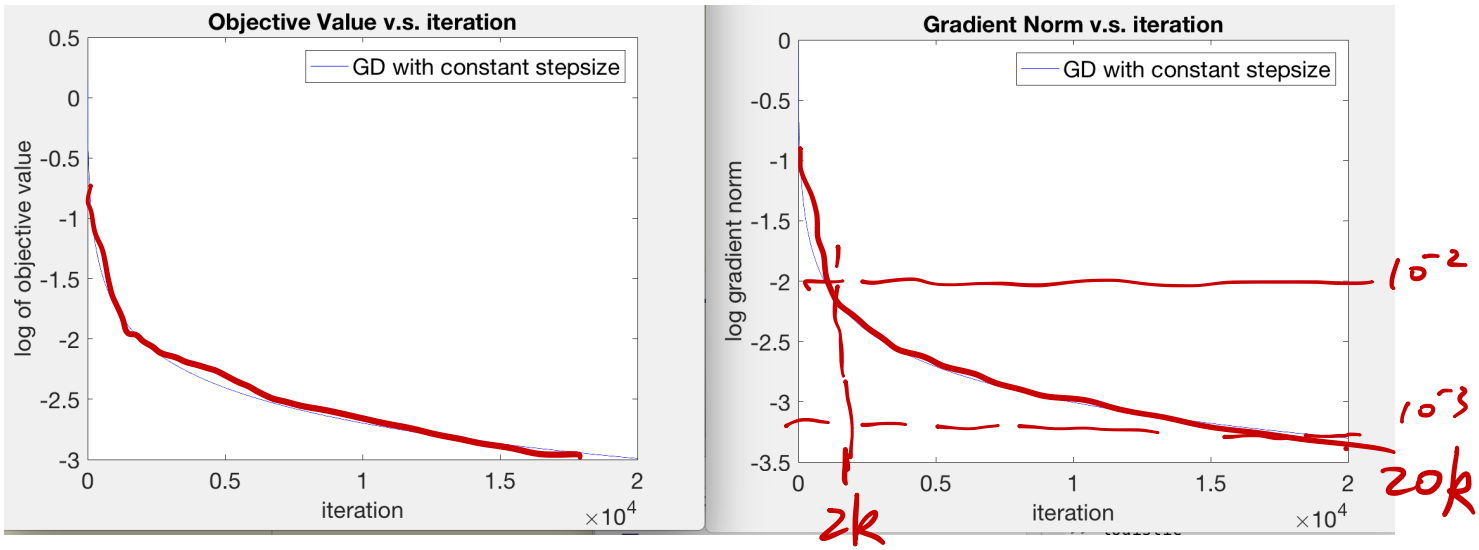
- This is a common case, but the general theory does not apply!

Extension: Higher-Dimensional Case?

Data Setting 2: $N = 10, d = 5$.

- Random Ganssian data $x_1, \ldots, x_{10} \in \mathbb{R}^5$.
- True separator $w^* = (1, 1, 1, 1, 1)$
- $y_i = \text{sign}(w' x_i), \forall i$.





Explain?

Applying Theory to Most Details

If you want, we could explain most details of the convergence behavior of LR.

e.g. in Data Setting 1, what determines the rate of convergence?

e.g. in Data Setting 2, it converges in $O(1/r)$ rate, what is the constant?

In higher-dimensional case, like $n, d = 1000$ or higher, the behavior may be more complicated.

- Again, if you are patient enough (and really understand homework 2 problem 2), you can explain the behavior

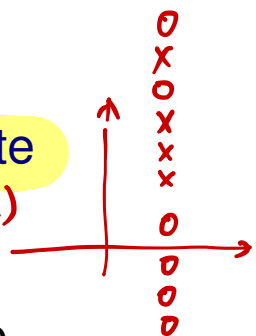
This confidence should NOT be extended to deep learning: we could NOT explain most behavior

Extension: High-Dimensional Case?

Separable case causes convergence issues – this was known for a long time.

- Allison 2008, "Convergence Failures in Logistic Regression."
- "A frequent problem in estimating logistic regression models is a failure of the ... algorithm to converge."
  - Here "failure of converge" means "failure of iterates to converge"
  - We know that the function values will converge
- Two possible data patterns: complete separation, or quasi-complete separation. The second is very common.

(∃ dummy variable)



- While complete separation is less common, combing LR with deep learning, it becomes more common

Resolve Issues in Separable Case

This was discussed in detail in [Allison 2008].

- Some statistical aspects need to be addressed

We discuss two possible solutions and the related issues.

Solution 1: Add regularizer, e.g., $\lambda\|\mathbf{w}\|^2$.

- Claim: If f is convex, then $f(\mathbf{w}) + \lambda\|\mathbf{w}\|^2$ is strongly convex.
- Issues: how to tune λ? Distort original solutions? Still slow....

Solution 2: Forget $f(\mathbf{w})$, only care about classification error.

- More precisely, $\sum_i |y_i - \text{sign}(w_i' x_i)|$; if zero, stop
- Issues: how to tune λ? distort original solutions? still slow in early stage...

Resolve Issues in Separable Case

This was discussed in detail in [Allison 2008].

- Some statistical aspects need to be addressed

We discuss two possible solutions and the related issues.

Solution 1: Add regularizer, e.g., $\lambda\|\mathbf{w}\|^2$.

- Claim: If f is convex, then $f(\mathbf{w}) + \lambda\|\mathbf{w}\|^2$ is strongly convex.
- Issues: how to tune λ? Distort original solutions? Still slow....

Solution 2: Forget $f(\mathbf{w})$, only care about classification error.

- More precisely, $\sum_i |y_i - \text{sign}(w_i' x_i)|$; if zero, stop
- Issues: how to tune λ? distort original solutions? still slow in early stage...

Resolve Issues in Separable Case

This was discussed in detail in [Allison 2008].

- Some statistical aspects need to be addressed

We discuss two possible solutions and the related issues.

Solution 1: Add regularizer, e.g., $\lambda\|\mathbf{w}\|^2$.

- Claim: If f is convex, then $f(\mathbf{w}) + \lambda\|\mathbf{w}\|^2$ is strongly convex.
- Issues: how to tune λ? Distort original solutions? Still slow....

Solution 2: Forget $f(\mathbf{w})$, only care about classification error.

- More precisely, $\sum_i |y_i - \text{sign}(w_i' x_i)|$; if zero, stop
- Issues: how to tune λ? distort original solutions? still slow in early stage...

Conclusion of This Lecture

Can you summarize yourself?

- Logistic regression problem may exhibit two convergence behavior: linear rate and sublinear rate
- When the data is good (separable), the problem is “hard”!
- Stopping criteria: $\|\nabla f(x^r)\|$ is a better metric for detecting convergence
- Sublinear rate: can happen even for 1-dim convex problems

Conclusion of This Lecture

Can you summarize yourself?

- Logistic regression problem may exhibit two convergence behavior: linear rate and sublinear rate
- When the data is good (separable), the problem is “hard”!
- Stopping criteria: $\|\nabla f(x^r)\|$ is a better metric for detecting convergence
- Sublinear rate: can happen even for 1-dim convex problems