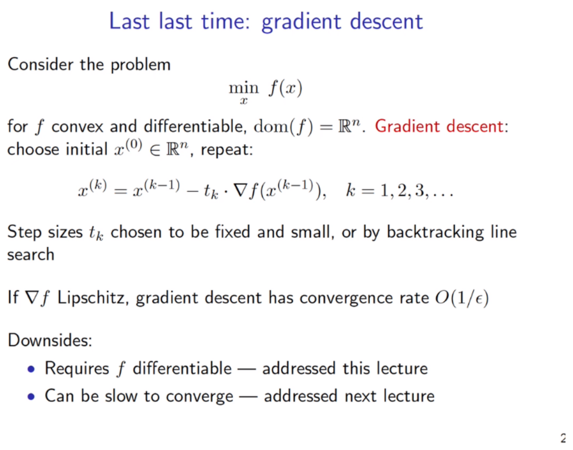
Subgradient method context

- a second algorithm to solve convex programs
  - the first being gradient descent
- similar flavour to gradient descent
- but more general



Last last time: gradient descent

Consider the problem

$$\min_x \ f(x)$$

for f convex and differentiable, $\mathrm{dom}(f) = \mathbb{R}^n$. Gradient descent: choose initial $x^{(0)} \in \mathbb{R}^n$, repeat:

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

Step sizes t_k chosen to be fixed and small, or by backtracking line search

If ∇f Lipschitz, gradient descent has convergence rate $O(1/\epsilon)$

Downsides:

- Requires f differentiable — addressed this lecture
- Can be slow to converge — addressed next lecture

Subgradient method

Consider f convex, dom(f) = Rn, but not necessarily differentiable

We want $min_x \ f(x)$

Method

1. initialize $x^{(0)}$
2. repeat $x^{(k)} = x^{(k-1)} - t_k \ . \ g^{(k-1)}, \ k = 1, \ 2, \ 3, \ ...$

where $g^{(k-1)} \in \partial f\left(x^{(k-1)}\right)$, any subgradient of f at $x^{(k-1)}$

Note that this is not necessarily a descent method, so we keep track of best iterate $x^{(k)}_{best}$ among $x^{(0)}, \ ..., \ x^{(k)}$ so far.

Question: Why do we know that gradient descent is a descent method?

If ∇f is Lipschitz with constant L

then $f(y) \leqslant f(x) + \nabla f(x)^T (y - x) + \dfrac{L}{2} ||y - x||_2^2$

- using Taylor series approximation
- and using the fact that the largest eigenvalue of the Hessian is L

use $y = x^{+} = x - t \nabla f(x)$

and take $t \leqslant \dfrac{1}{L} \implies L \leqslant \dfrac{1}{t}$

then $f\left(x^{+}\right) \leqslant f(x) + \nabla f(x)^{T}(-t\nabla f(x)) + \frac{L}{2}||\nabla f(x)||_2^2$

$\implies f\left(x^{+}\right) \leqslant f(x) - t||\nabla f(x)||_2^2 + \frac{1}{2t}||\nabla f(x)||_2^2$

$\implies f\left(x^{+}\right) \leqslant f(x) - ||\nabla f(x)||_2^2 . \left(t + \frac{1}{2t}\right)$

as $\left(t + \frac{1}{2t}\right) > 0\ for\ all\ t > 0$, this means that we descend the gradient every iteration.

For subgradient analysis, we don't have such strong assumptions about the function. That is we don't know if the gradient is Lipschitz continuous or anything, so can't provide an upper bound on the curvature of the function.

Outline for today

- how to choose step sizes
- convergence analysis
- intersection of sets
- project subgradient method (useful for constraints)

A. Step sizes

One immediate diff vs GD is how to choose step sizes.

1. Fixed step sizes

2. Diminising step sizes

- want to step sizes to go towards zero but not too fast
- square summable to something finite, but in itself sums to something infinite
- $\sum t_k^2 < inf,\ but \sum t_k = inf$
- eg: $t_k = 1/k$ (we know $\sum \frac{1}{k^2} = \frac{\pi^2}{6}$)

There is no adaptive step sizing here - no backtracking. We know the step size a priori for every step.

B. Convergence analysis

Assume that the function itself is Lipschitz with constant G.

- We don't know if the function has a gradient, so can't make the assumption about the gradient, as we do for gradient descent analysis

This implies that for all $g \in \partial f(x),\ ||g||_2 \leqslant G.$

Why?

Recall from definition of subgradient that $f(y) \geqslant f(x) + g^{T}(y-x)$ for all $g \in \partial f(x)$

$\implies g^T(y-x) \leqslant f(y) - f(x)$

We know also from definition of Lipschitz that for f() Lipschitz with constant G,

$|f(x) - f(y)| \leqslant G|x - y|$

so, $||g||_2 \leqslant G$

- don't know exactly how

Interpretation: the subgradients cannot be get too large in norm. Some notion of regularity on the function.

Two theorems

- weaker than what we can say about GD: unbounded number of iterations
  - unlike with GD, we can't say where we will be after a certain number of steps k

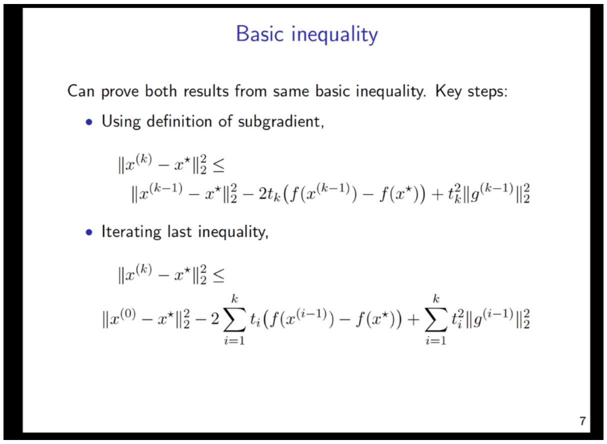
Theorem 1: For a fixed step size t, the subgradient method satisfies

$$\lim_{k \to \infty} f\left(x_{best}^{(k)}\right) \leqslant f^* + G^2 t/2$$

Theorem 2: For diminishing step sizes, subgradient method satisfies

$$\lim_{k \to \infty} f\left(x_{best}^{(k)}\right) = f^*$$

- Note that neither of these two tells us about the **rate** of convergence - just tell us what to expect in the limit



Basic inequality

Can prove both results from same basic inequality. Key steps:

- Using definition of subgradient,

$$\|x^{(k)} - x^\star\|_2^2 \le \|x^{(k-1)} - x^\star\|_2^2 - 2t_k\big(f(x^{(k-1)}) - f(x^\star)\big) + t_k^2\|g^{(k-1)}\|_2^2$$

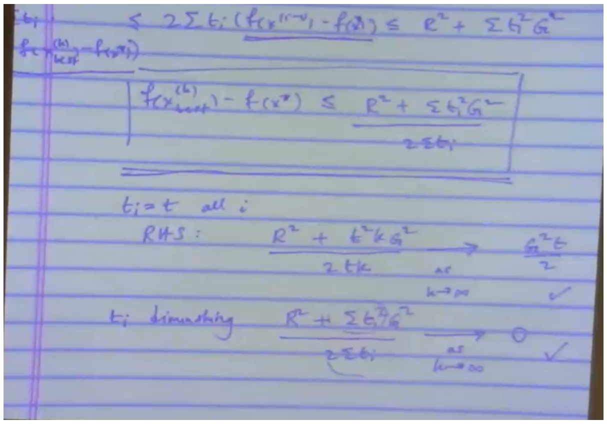
- Iterating last inequality,

$$\|x^{(k)} - x^\star\|_2^2 \le \|x^{(0)} - x^\star\|_2^2 - 2\sum_{i=1}^{k} t_i\big(f(x^{(i-1)}) - f(x^\star)\big) + \sum_{i=1}^{k} t_i^2\|g^{(i-1)}\|_2^2$$

Form of proof

- start with assumptions you have eg Lipschitz
- write current iteration as function of past iteration(s)
- iterate over inequality you get for step k over all k
  - dont have to sum over the inequalities for all k - just repeat the bound logic for k-1 in terms of k-2, k-2 in terms of k-3, ..., all the way till k=1

(see lecture for proofs, very nice)



$\Sigma t_i (f(x^{(k)}_{best}) - f(x^*)) \le 2\Sigma t_i (f(x^{(i-1)}) - f(x^*)) \le R^2 + \Sigma t_i^2 G^2$

$$\boxed{f(x^{(k)}_{best}) - f(x^*) \le \frac{R^2 + \Sigma t_i^2 G^2}{2\Sigma t_i}}$$

$t_i = t$ all i

RHS: $\frac{R^2 + t^2 k G^2}{2tk} \xrightarrow[k\to\infty]{\text{as}} \frac{G^2 t}{2}$ ✓

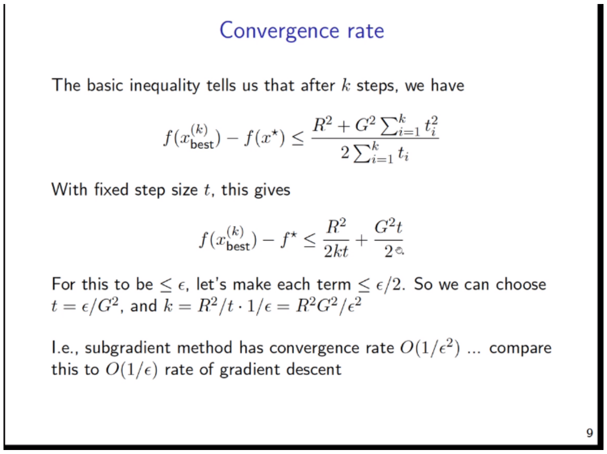
t_i diminishing $\frac{R^2 + \Sigma t_i^2 G^2}{2\Sigma t_i} \xrightarrow[k\to\infty]{\text{as}} 0$ ✓

Q: Why ever use fixed step size?

Two applications of the subgradient method

1. classical - this is a general method
2. stochastic - if the subgradient is the sum of a bunch of things just take one of those things?
   - used in ML where you dont care about global optimality
   - often use fixed step sizes

C. Convergence rate



Convergence rate

The basic inequality tells us that after k steps, we have

$$f(x^{(k)}_{\text{best}}) - f(x^\star) \le \frac{R^2 + G^2 \sum_{i=1}^k t_i^2}{2\sum_{i=1}^k t_i}$$

With fixed step size t, this gives

$$f(x^{(k)}_{\text{best}}) - f^\star \le \frac{R^2}{2kt} + \frac{G^2 t}{2}$$

For this to be $\le \epsilon$, let's make each term $\le \epsilon/2$. So we can choose $t = \epsilon/G^2$, and $k = R^2/t \cdot 1/\epsilon = R^2G^2/\epsilon^2$

I.e., subgradient method has convergence rate $O(1/\epsilon^2)$ ... compare this to $O(1/\epsilon)$ rate of gradient descent

9

Example: if eps=0.01 = 10^-2

Then with GD, O(1/eps), the number of steps = 100 iterations

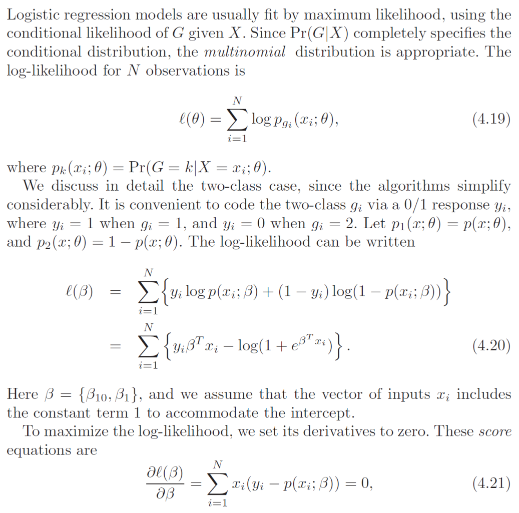
With SGM, O(1/eps^2), the number of steps = 10000 iterations

Example: regularized logistic regression

Given $(x_i, y_i) \in R^p \times \{0, 1\}$ $for\ i = 1, ..., n$ the logistic regression loss (2 class) is

$$f(B) = \sum_{i=1}^{n} \left(\log\left(1 + \exp\left(x_i^T B\right)\right) - y_i x_i^T B \right)$$

Working through this myself

- 

Logistic regression models are usually fit by maximum likelihood, using the conditional likelihood of G given X. Since $\Pr(G|X)$ completely specifies the conditional distribution, the *multinomial* distribution is appropriate. The log-likelihood for N observations is

$$\ell(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta), \tag{4.19}$$

where $p_k(x_i; \theta) = \Pr(G = k|X = x_i; \theta)$.

We discuss in detail the two-class case, since the algorithms simplify considerably. It is convenient to code the two-class g_i via a $0/1$ response y_i, where $y_i = 1$ when $g_i = 1$, and $y_i = 0$ when $g_i = 2$. Let $p_1(x; \theta) = p(x; \theta)$, and $p_2(x; \theta) = 1 - p(x; \theta)$. The log-likelihood can be written

$$\begin{aligned} \ell(\beta) &= \sum_{i=1}^{N} \Big\{ y_i \log p(x_i; \beta) + (1 - y_i) \log(1 - p(x_i; \beta)) \Big\} \\ &= \sum_{i=1}^{N} \Big\{ y_i \beta^T x_i - \log(1 + e^{\beta^T x_i}) \Big\}. \end{aligned} \tag{4.20}$$

Here $\beta = \{\beta_{10}, \beta_1\}$, and we assume that the vector of inputs x_i includes the constant term 1 to accommodate the intercept.

To maximize the log-likelihood, we set its derivatives to zero. These *score* equations are

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i (y_i - p(x_i; \beta)) = 0, \tag{4.21}$$

- from ESLII page 120

Is this convex?

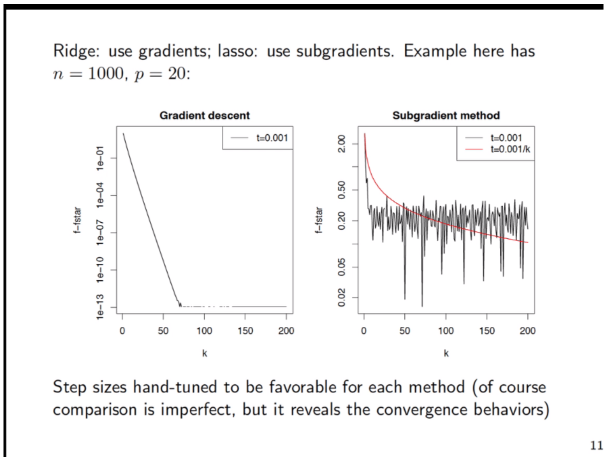
Yes. $\sum \log\left(1 + \exp\left(x_i^T B\right)\right)$ is a log-sum-exp function which we know is convex from Lecture 3

the rest is affine

It has $\nabla f(B) = \sum_{i=1}^{n} (y_i - p_i(B)) x_i$ where $p_i(B) = \dfrac{\exp\left(x_i^T B\right)}{1 + \exp\left(x_i^T B\right)}$

Consider the regularized problem $min_B\ f(B) + \lambda P(B)$

- for ridge regularized logistic, $P(B) = ||B||_2^2$ use GD as it is smooth
- for lasso regularized logistic, $P(B) = ||B||_1$ use SG as it is non smooth
  - observe that convergence is far slower and thr improvement on criterion value is far worse in the latter case!
  - The ridge criterion function is strongly convex for the sub level set when we say the criterion value must be less than what it used to be



Ridge: use gradients; lasso: use subgradients. Example here has $n = 1000$, $p = 20$:

Step sizes hand-tuned to be favorable for each method (of course comparison is imperfect, but it reveals the convergence behaviors)

11

Clarke subgradient - vectors that observe subgradient rules local to a region but not across the whole domain

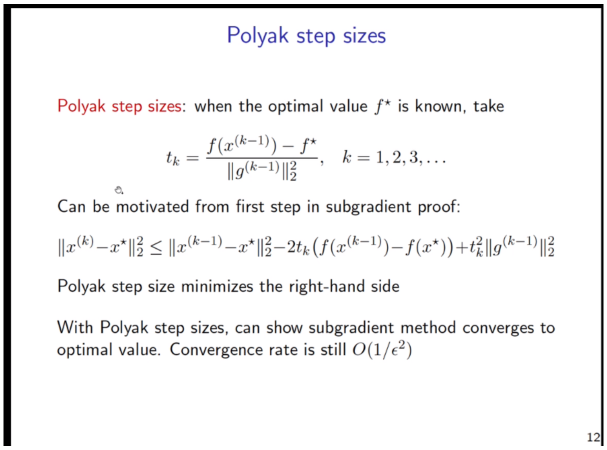
- for non convex functions

D. Polyak step sizes

If you knew the min criterion value $(f(x^*),\ not\ x^*\ itself)$ you could calculate the best step size for every iteration

$$t_k \ = \frac{f(x^{(k-1)}) - f^*}{||g^{(k-1)}||_2^2}$$

= criterion function / squared L2 norm of the subgradient you used in the update



Polyak step sizes

Polyak step sizes: when the optimal value $f^\star$ is known, take

$$t_k = \frac{f(x^{(k-1)}) - f^\star}{\|g^{(k-1)}\|_2^2}, \quad k = 1, 2, 3, \ldots$$

Can be motivated from first step in subgradient proof:

$$\|x^{(k)} - x^\star\|_2^2 \le \|x^{(k-1)} - x^\star\|_2^2 - 2t_k\big(f(x^{(k-1)}) - f(x^\star)\big) + t_k^2\|g^{(k-1)}\|_2^2$$

Polyak step size minimizes the right-hand side

With Polyak step sizes, can show subgradient method converges to optimal value. Convergence rate is still $O(1/\epsilon^2)$

12

Example: intersection of sets

Problem: suppose we want to find $x^* \ \in C_1\ \cap\ ...\ \cap C_m$ i.e. find a point in the intersection of these closed convex sets.

First, define

$$f_i = dist(x,\ C_i),\ i = 1,\ ...,\ m$$

$f(x) \; = \; \max_i f_i(x)$
Then solve $\min f(x) \; = \; \min \max_i f_i(x)$ - find the point x^* that minimizes the max distance to any set

Note that if $f\left(x^*\right)$ = max distance to any set = 0, then that means x is in the intersection of all the sets

- So **we know what the optimal criterion value is!**
- This makes it feasible for **Polyak's step size solver**

Is this convex?
Yes.

Is f(x) differentiable?
No. The max function is not differentiable so we cant assume a gradient. Therefore we will need to use a subgradient method to solve this problem.

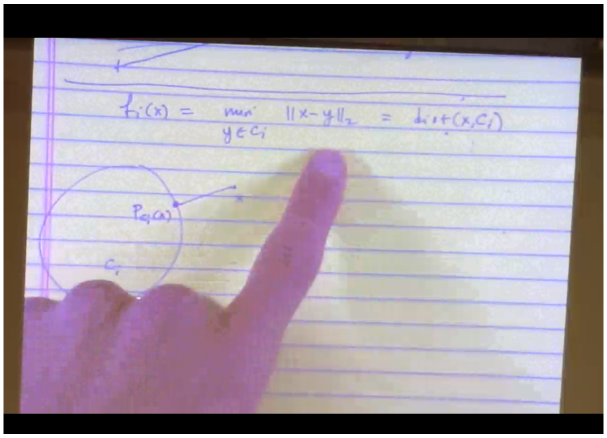
Note that $f_i(x)$ is indeed differentiable - we know the gradient (see point 2). But f(x), which is the max across these functions, is not.

So we want to find some x that does this!

- Here, we can use Polyak step sizes to aid the computation

1. Recall the distance function $f_i(x) \; = \; dist(x, \; C_i) \; = \; min_{y \in C_i} ||y - x||_2$
   - the y that fulfills this is hte projection of x onto the set C
2. We compute its gradient $(g =)\; \nabla dist(x, \; C) \; = \; \frac{x \; - \; P_C(x)}{||x - P_C(x)||_2}$ = normalized distance vector

where $P_C(x)$ is the projection of x onto C.

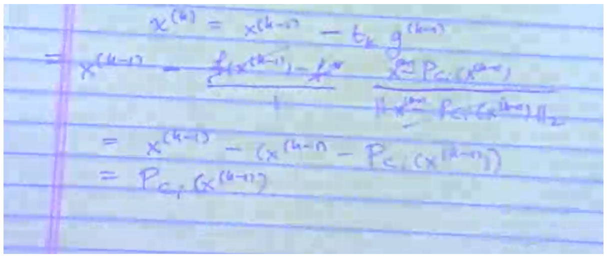


$$f_i(x) = \min_{y \in C_i} \|x - y\|_2 = dist(x, C_i)$$

3. Also recall subgradient rule: if $f(x) = max\, f_i(x)$ then $\partial f(x) = conv(\cup_{f_i(x) = f(x)} \partial f_i(x))$ But in our case, we know that $f_i(x)$ is differentiable - in fact we know its gradient (point 2) - so each subdifferential $\partial f_i(x)$ is a singleton, i.e. it just contains the gradient

4. This means that the subdifferential $\partial f(x)$ = {subgradients of all the functions that share the max distance} = {gradients of the functions that share the max distance}

- So just pick any gradient for any these - it would be a valid subgradient for $f(x)$
- At every step k
  - Pick any C_i furthest from x
  - use the gradient of its distance function as a subgradient for $f(x)$ = $g^{(k-1)}$
  - apply the subgradient method update rule: $x^{(k)} = x^{(k-1)} - t_k . g^{(k-1)}$
  -

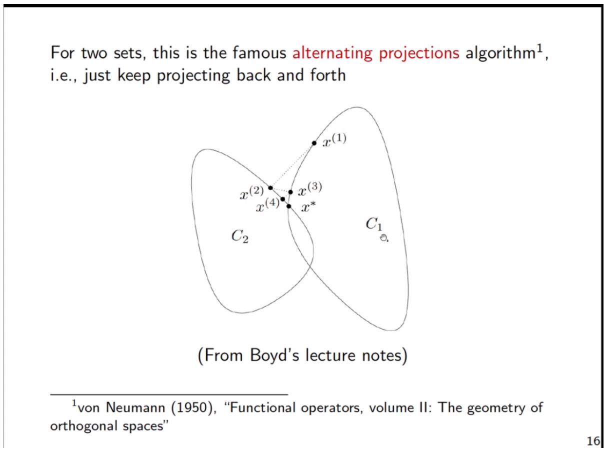


$$x^{(k)} = x^{(k-1)} - t_k\, g^{(k-1)}$$
$$= x^{(k-1)} - \frac{f(x^{(k-1)}) - f^*}{1} \frac{x^{(k-1)} - P_{C_i}(x^{(k-1)})}{\|x^{(k-1)} - P_{C_i}(x^{(k-1)})\|_2}$$
$$= x^{(k-1)} - (x^{(k-1)} - P_{C_i}(x^{(k-1)}))$$
$$= P_{C_i}(x^{(k-1)})$$

  - Use Polyak's rule for t_k. But as the subgradient is normalized, the L2 norm is 1!
  - Also notice that since we are picking C_i that is furthest away, the criterion value = the norm of the dist function!
    - * this is true by definition

5. Everything cancels out and you get $x^{(k)} = P_{C_i}\left(x^{(k-1)}\right)$ - i.e. at every step, project onto the furthest set!

This is the update and it is very intuitive!



For two sets, this is the famous alternating projections algorithm[1], i.e., just keep projecting back and forth

(From Boyd's lecture notes)

[1] von Neumann (1950), "Functional operators, volume II: The geometry of orthogonal spaces"

16

- We also know from this approach to the algorithm that the the convergence is

the alternating projections algorithm is $O\left(1/\epsilon^2\right)$. This is not obvious geometrically!

E. Projected subgradient method

- for constraints - you may start outside the feasible region - just project onto the constraint set and continue
  $min\ f(x)\ subject\ to\ x\ \in C$
- the projection is a subroutine so don't want to pick a projection problem that is in itself a hard QP