

CPSC 540: Machine Learning

Optimization Zoo, Coordinate Optimization

Mark Schmidt

University of British Columbia

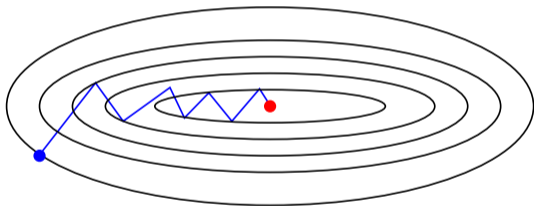
Winter 2017

Admin

- **Auditting/registration forms:**
 - Add/drop deadline is tomorrow.
 - We can discuss/sign the remaining ones at the end class.
- **Assignment 1:**
 - Due tonight.
 - 1 late day to hand it in Wednesday.
 - 2 late days to hand it in next Monday.
- **Assignment 2:**
 - Out soon.
 - Due February 6 (3 weeks).

Last Time: Gradient Descent

- Gradient descent:
 - Iterative algorithm for finding stationary point of differentiable function.
 - For convex functions it finds a global minimum.



Start with x^0 , apply

$$x^{t+1} = x^t - \alpha_t \nabla f(x^t),$$

for step-size α_t .

- Cost of algorithm scales linearly with number of variables d .
 - Costs $O(ndt)$ for t iterations for least squares and logistic regression.
 - For $t < d$, faster than $O(nd^2 + d^3)$ of normal equations or Newton's method.

Last Time: Convergence Rate of Gradient Descent

- We introduced the **iteration complexity** of an algorithm for a problem class:
 - “How many iterations t before we guarantee an accuracy ϵ ”?
- We assumed **strong-convexity** and **strong-smoothness**,

$$\mu I \preceq \nabla^2 f(x) \preceq LI$$

for all x and $0 < \mu \leq L < \infty$

(eigenvalues bounded between positive constants everywhere)

- By using multivariate second-order **Taylor expansion**,

$$f(y) = f(x) + \nabla^T f(x)(y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(z)(y - x),$$

we showed gradient descent has **linear convergence rate** which implies $t = \log(1/\epsilon)$.

Last Time: Gradient Descent Practical Issues

- Optimization has both a theoretical and practical component:
 - We can use theory to justify/guide our choice of method.
 - But bounds tend to be loose, so we need to experiment.
- In practice you don't need L and probably don't want it:
 - Adaptive step-size, Armijo line-search.
- We overviewed methods with better performance:
 - Nesterov's accelerated gradient method (better in theory, sometimes in practice)
 - Approximations to Newton's method that only need $O(d)$ -cost operations.
 - Diagonal scaling, Barzilai-Borwein, Hessian-free, limited-memory quasi-Newton.
 - Typically these are the best methods in practice for large d .

(Take Michael Friedlander's classes for more details.)

Differentiability Classes and Lipschitz Continuity

- We'll find it convenient to define **differentiability classes**:
 - \mathcal{C}^0 is the set of continuous functions.
 - \mathcal{C}^1 is the set of once-differentiable functions.
 - \mathcal{C}^2 is the set of twice-differentiable functions.
- Our $(1 - \mu/L)$ convergence rate holds for \mathcal{C}^2 functions.
- But it can be generalized to \mathcal{C}^1 functions.
- To show this, we'll say that a function is **Lipschitz-continuous** if

$$\|g(x) - g(y)\| \leq L\|x - y\|,$$

for some **constant L for all x and y** (g can have multiple inputs and outputs).

- It means that g can't change too quickly.

Convergence Rate for \mathcal{C}^1 Functions

- Consider functions f with a **Lipschitz-continuous gradient**,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

- For \mathcal{C}^2 functions, this is **equivalent to strong-smoothness**.
- For \mathcal{C}^1 functions, this implies the **descent lemma** (bonus slide),

$$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2}\|y - x\|^2.$$

- This is the **upper bound we used in our convergence proof**.
 - We can weaken **strong smoothness** to **Lipschitz-continuous gradient**.

Convergence Rate for \mathcal{C}^1 Functions

- We can define **strong-convexity** for \mathcal{C}^0 (not necessarily differentiable) functions
 - Function f is strongly-convex if the function

$$x \mapsto f(x) - \frac{\mu}{2}\|x\|^2,$$

is convex for some $\mu > 0$.

- For \mathcal{C}^2 functions, this is equivalent to our previous definition.
- For \mathcal{C}^1 functions, this implies that (bonus slide)

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\mu}{2}\|y - x\|^2.$$

- This is the lower-bound used in our proof (we **don't need twice-differentiability**).

Strong-Convexity and Regularization

- We can define **strong-convexity** for \mathcal{C}^0 (not necessarily differentiable) functions
 - Function f is strongly-convex if the **function**

$$x \mapsto f(x) - \frac{\mu}{2}\|x\|^2,$$

is convex for some $\mu > 0$.

- Useful implication:
 - If f is **convex**, then $f + \frac{\lambda}{2}\|x\|^2$ is **strongly-convex** with $\mu \geq \lambda$.
- So **adding L2-regularization to a convex function makes it strongly-convex**.
 - We converge faster and **guarantee that a unique solution exists**.

Outline

- 1 How hard is optimization?
- 2 Weaker Assumptions for Linear Convergence
- 3 L1-Regularization and Coordinate Optimization

First-Order Oracle Model of Computation

- Should we be happy with an algorithm that takes $O(\log(1/\epsilon))$ iterations?
 - Is it possible that algorithms *exist* that solve the problem faster?
- To answer this question, need a **class of functions**.
 - For example, **strongly-convex** with **Lipschitz-continuous gradient**.
- We also need a **model of computation**: what operations are allowed?
- We will typically use a **first-order oracle** model of computation:
 - On iteration t , algorithm choose an x^t and receives $f(x^t)$ and $\nabla f(x^t)$.
 - To choose x^t , algorithm **can do anything** that doesn't involve f .
- Common variation is **zero-order oracle** where algorithm only receives $f(x^t)$.

Complexity of Minimizing Real-Valued Functions

- Consider minimizing **real-valued** functions over the unit hyper-cube,

$$\min_{x \in [0,1]^d} f(x).$$

- You can use **any algorithm** you want.

(simulated annealing, gradient descent + random restarts, genetic algorithms, Bayesian optimization, ...)

- How many zero-order oracle calls t before we can guarantee $f(x^t) - f(x^*) \leq \epsilon$?

- **Impossible!**

- Given any algorithm, we can construct an f where $f(x^t) - f(x^*) > \epsilon$ **forever**.

- Make $f(x) = 0$ except at x^* where $f(x) = -\epsilon - 2^{\text{whatever}}$.

(the x^* is algorithm-specific)

- To say anything in oracle model we **need assumptions on f** .

Complexity of Minimizing Lipschitz-Continuous Functions

- One of the simplest assumptions is that f is **Lipschitz-continuous**,

$$|f(x) - f(y)| \leq L\|x - y\|.$$

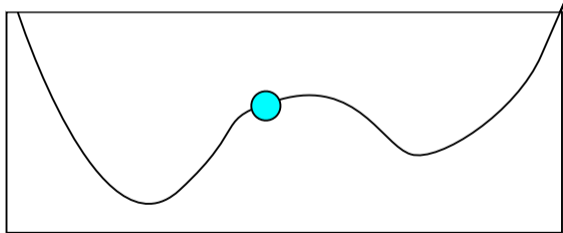
- Function can't change arbitrarily fast as you change x .

Complexity of Minimizing Lipschitz-Continuous Functions

- One of the simplest assumptions is that f is **Lipschitz-continuous**,

$$|f(x) - f(y)| \leq L\|x - y\|.$$

- Function can't change arbitrarily fast as you change x .

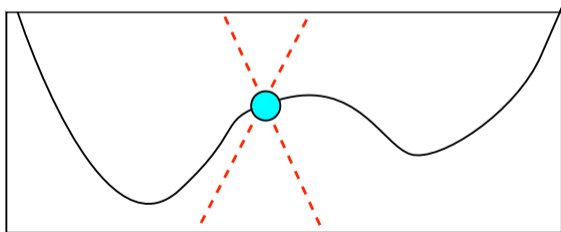


Complexity of Minimizing Lipschitz-Continuous Functions

- One of the simplest assumptions is that f is **Lipschitz-continuous**,

$$|f(x) - f(y)| \leq L\|x - y\|.$$

- Function can't change arbitrarily fast as you change x .

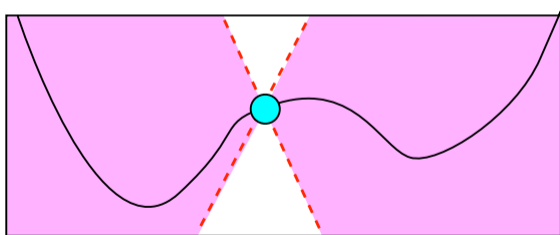


Complexity of Minimizing Lipschitz-Continuous Functions

- One of the simplest assumptions is that f is **Lipschitz-continuous**,

$$|f(x) - f(y)| \leq L\|x - y\|.$$

- Function can't change arbitrarily fast as you change x .

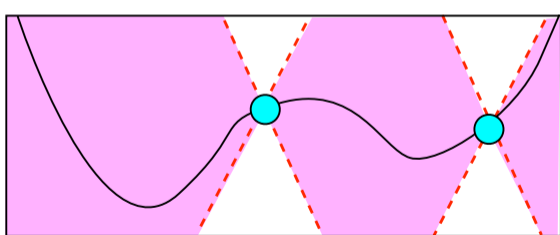


Complexity of Minimizing Lipschitz-Continuous Functions

- One of the simplest assumptions is that f is **Lipschitz-continuous**,

$$|f(x) - f(y)| \leq L\|x - y\|.$$

- Function can't change arbitrarily fast as you change x .



Complexity of Minimizing Lipschitz-Continuous Functions

- One of the simplest assumptions is that f is **Lipschitz-continuous**,

$$|f(x) - f(y)| \leq L\|x - y\|.$$

- Function can't change arbitrarily fast as you change x .
- Under only this assumption, **any algorithm requires at least $\Omega(1/\epsilon^d)$ iterations**.
- An **optimal $O(1/\epsilon^d)$ worst-case rate** is achieved by a grid-based search method.
- You can also achieve **optimal rate in expectation** by **random guesses**.
 - Lipschitz-continuity implies there is a ball of ϵ -optimal solutions around x^* .
 - The radius of the ball is $\Omega(\epsilon)$ so its area is $\Omega(\epsilon^d)$.
 - If we succeed with probability $\Omega(\epsilon^d)$, we expect to need $O(1/\epsilon^d)$ trials.
(mean of geometric random variable)

Complexity of Minimizing Convex Functions

- Life gets better if we assume **convexity**.
 - We'll consider **first-order oracles** and rates with no dependence on d .
- Subgradient methods (next week) can minimize convex functions in $O(1/\epsilon^2)$.
 - This is optimal in dimension-independent setting.
- If the **gradient is Lipschitz continuous**, gradient descent requires $O(1/\epsilon)$.
 - With Nesterov's algorithm, this improves to $O(1/\sqrt{\epsilon})$ which is optimal.
 - Here we don't yet have strong-convexity.
- What about the CPSC 340 approach of **smoothing** non-smooth functions?
 - Gradient descent **still requires $O(1/\epsilon^2)$** in terms of solving original problem.
 - Nesterov improves to $O(1/\epsilon)$ in terms of original problem.

Complexity of Minimizing Strongly-Convex Functions

- For **strongly-convex** functions:
 - Sub-gradient methods achieve optimal rate of $O(1/\epsilon)$.
 - If ∇f is **Lipschitz continuous**, we've shown that gradient descent has $O(\log(1/\epsilon))$.
- Nesterov's algorithms improves this from $O(\frac{L}{\mu} \log(1/\epsilon))$ to $O(\sqrt{\frac{L}{\mu}} \log(1/\epsilon))$.
 - It's close to optimal.

Sublinear vs. Linear vs. Superlinear Convergence

- Having $t = O(1/\epsilon^2)$ means that $\epsilon = O(1/\sqrt{t})$ after t iterations.
- Having $t = O(1/\epsilon)$ means that $\epsilon = O(1/t)$ after t iterations.
- Having $t = O(1/\sqrt{\epsilon})$ means that $\epsilon = O(1/t^2)$ after t iterations.
- These are **sublinear** rates, **time increases exponentially** in digits of accuracy.

- Having $t = O(\log(1/\epsilon))$ means that $\epsilon = O(\rho^t)$ after t iterations.
- This is called a **linear** rate, **time increases polynomially** in digits of accuracy.

- **Superlinear** means we have $\epsilon = O(\prod_{k=1}^t \rho_k)$ where ρ_k converges to 0.
 - An example is **quadratic convergence** where $t = O(\log \log(1/\epsilon))$.

Sublinear vs. Linear vs. Superlinear Convergence

- Iteration complexity measures number of iterations.
- Total time complexity will be iteration complexity times cost of iterations.
 - Usually, cost of iterations is simpler to compute.
- For logistic regression:
 - Iteration complexity for gradient descent is $O(\log(1/\epsilon))$ iterations.
 - Iteration cost for gradient descent is $O(nd)$ time.
 - So total time is $O(nd \log(1/\epsilon))$.
- Note the different use of sublinear/superlinear in the two settings:
 - Sublinear convergence is bad and superlinear convergence is good.
 - Sublinear cost is good and superlinear cost is bad.
 - We ideally want superlinear convergence with sublinear cost.

Superlinear Convergence in Practice?

- You get **local superlinear convergence** if:
 - Gradient is Lipschitz-continuous and f is strongly-convex.
 - Function is in \mathcal{C}^2 and Hessian is **Lipschitz continuous**.
 - Oracle is second-order and method **asymptotically uses Newton's direction**.
- But the **practical Newton-like methods** don't achieve this:
 - Diagonal scaling, Barzilai-Borwein, and L-BFGS don't converge to Newton.
 - Hessian-free uses conjugate gradient which isn't superlinear in high-dimensions.
- Full quasi-Newton methods achieve this, but require $\Omega(d^2)$ memory/time.

Outline

- 1 How hard is optimization?
- 2 Weaker Assumptions for Linear Convergence**
- 3 L1-Regularization and Coordinate Optimization

Linear Convergence with Alternate Step-Sizes

- Linear convergence is possibly the best we can hope for in large-scale settings.
- Can we weaken our previous assumptions and still get linear convergence?
- We used constant $\alpha_t = 1/L$.
 - Proof will work for any constant $\alpha_t < 2/L$, so we just need small enough step-size.
 - Proof works if you find the optimal step-size,
 - Value of f with optimal step-size cannot be greater than with $1/L$.
 - You can compute this for quadratics: just minimizing a 1D quadratic.
 - Proof can be modified to work adaptive L or backtracking
 - Rate is slightly uglier/slower.

Linear Convergence with Non-Lipschitz Gradient

- Instead of **Lipschitz-continuous gradient** we really only need that

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

for all x^t and some L .


- For example, in covariance matrix estimation (later in course) ∇f is not Lipschitz.
 - For differentiable functions, this is a very **weak assumption**.
- Instead of **strong-convexity** we really only need for some μ that

$$f(x^*) \geq f(x) - \frac{1}{2\mu} \|\nabla f(x^t)\|^2.$$

- This is called the **Polyak-Łojasiewicz (PL) inequality**.
 - Some non-convex functions satisfy this: $x^2 + 3 \sin^2(x)$ has $\mu > 1/32$.

Linear Convergence without Strong-Convexity

- The **least squares** problem is convex but **not strongly convex**.
 - We could add a regularizer to make it strongly-convex.
 - But if we really want the MLE, are we stuck with sub-linear rates?
- Many conditions give linear rates that are weaker than strong-convexity:
 - 1963: Polyak-Łojasiewicz (PL).
 - 1993: Error bounds.
 - 2000: Quadratic growth.
 - 2013-2015: essential strong-convexity, weak strong convexity, restricted secant inequality, restricted strong convexity, optimal strong convexity, semi-strong convexity.
- Least squares satisfies all of the above (bonus slide for PL).
- Do we need to study any of the newer ones?
 - No! All of the above imply PL except for QG.
 - But with only QG gradient descent may not find optimal solution.

Boris Polyak boris@ipu.ru [via](#) cs.ubc.ca
to hamedkarim, jnutini, schmidtm 

 9/2/16



Dear colleagues,

it was a pleasure to read your paper **Linear Convergence of Gradient and Proximal-Gradient Methods Under the Polyak-Lojasiewicz Condition.**

Actually it was a surprise for me why the simple condition from my old publication attracted so little attention, and now

your work exhibits its importance.

You cite my paper in Russian, find attached its English translation. I also attach our paper with Nesterov where some extensions of the condition (and some its applications) can be found.

Best regards,
Boris Polyak

Linear Convergence for “Locally-Nice” Functions

- For linear convergence it's sufficient to have

$$L[f(x^{t+1}) - f(x^t)] \geq \frac{1}{2} \|\nabla f(x^t)\|^2 \geq \mu[f(x^t) - f^*],$$

for all x^t for some L and μ with $L \geq \mu > 0$.

(technically, we could even get rid of the connection to the gradient)

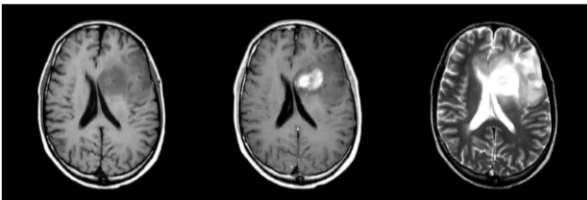
- Notice that this **only needs to hold for all x^t** , not for all possible x .
 - We could get linear rate for “nasty” function if the iterations stay in a “nice” region.
 - We can get lucky and converge faster than the global L/μ would suggest.
- Arguments like this give linear rates for some non-convex problems like PCA.

Outline

- 1 How hard is optimization?
- 2 Weaker Assumptions for Linear Convergence
- 3 L1-Regularization and Coordinate Optimization**

Motivation: Automatic Brain Tumour Segmentation

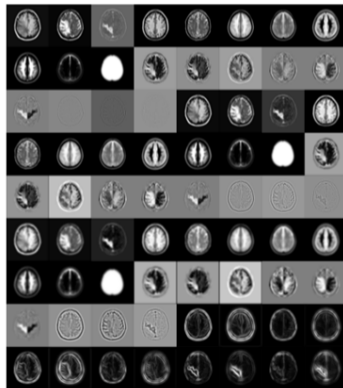
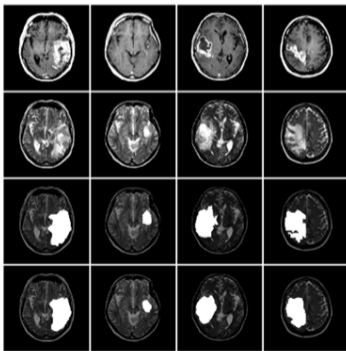
- Task: identifying tumours in multi-modal MRI data.



- Applications:
 - Image-guided surgery.
 - Radiation target planning.
 - Quantifying treatment response.
 - Discovering growth patterns.

Motivation: Automatic Brain Tumour Segmentation

- Formulate as **supervised learning**:
 - Pixel-level classifier that predicts “tumour” or “non-tumour”.
 - Features: convolutions, expected values (in aligned template), and symmetry.
 - All at multiple scales.



Motivation: Automatic Brain Tumour Segmentation

- Logistic regression was among the most effective, with the right features.
- But if you used all features, it **overfit**.
 - We needed **feature selection**.
- Classical approach:
 - Define some score: AIC, BIC, cross-validation error, etc.
 - Search for features that optimize score:
 - Usually NP-hard, so we use greedy: forward selection, backward selection,...
 - In this application, these are too slow: one image is 8 million training examples.

Feature Selection

- General **feature selection** problem:
 - Given our usual X and y , we'll use x_j to represent column j :

$$X = \begin{bmatrix} | & | & \dots & | \\ x_1 & x_2 & \dots & x_d \\ | & | & \dots & | \end{bmatrix}, \quad y = \begin{bmatrix} | \\ y \\ | \end{bmatrix}.$$

- We think **some features/columns x_j are irrelevant** for predicting y .
- We want to fit a model that uses the “best” set of features.
- **One of most important problems in ML/statistics, but very very messy.**
 - In 340 we saw how difficult it is to define what “relevant” means.
 - For now, a feature is “relevant” if it helps predict y^i from x^i .

L1-Regularization

- A popular approach to feature selection we saw in 340 is **L1-regularization**:

$$w^* \in \operatorname{argmin}_{w \in \mathbb{R}^d} g(w) + \lambda \|w\|_1.$$

- Advantages:
 - **Fast**: can apply to large datasets, just minimizing one convex function.
 - **Reduces overfitting** because it simultaneously regularizes.
 - Sample complexity is **logarithmic in number of irrelevant features**.
 - Compared to **linear dependence** for L2-regularization.
- Disadvantages:
 - **Prone to false positives**, particularly if you pick λ by cross-validation.
 - **Not unique**: there may be infinite solutions.
- There exist many extensions:
 - “Elastic net” adds L2-regularization to make solution unique.
 - “Bolasso” uses bootstrapping to reduce false positives.
 - Non-convex regularizers reduce false positives but are NP-hard.

L1-Regularization

- Key property of **L1-regularization**: if λ is large, **solution w^* is sparse**:
 - w^* has many values that are exactly zero.
- How this performs feature selection in linear models:

$$\hat{y}^i = w^T x^i = w_1 x_1^i + w_2 x_2^i + w_3 x_3^i + w_4 x_4^i + w_5 x_5^i.$$

- If $w = [0 \ 0 \ 3 \ 0 \ -2]$ then:

$$\begin{aligned}\hat{y}^i &= 0x_1^i + 0x_2^i + 3x_3^i + 0x_4^i + (-2)x_5^i \\ &= 3x_3^i - 2x_5^i.\end{aligned}$$

- **Features $\{1, 2, 4\}$ are not used in making predictions.**

L1-Regularization vs. L2-Regularization

- Why does L1-regularization give sparsity but L2-regularization does not?
 - Don't they both shrink variables towards 0?
- Consider a 2-variable problem whose MLE is given by

$$w = \begin{bmatrix} 100 \\ 0.01 \end{bmatrix}.$$

which has $\|w\|^2 = 10000.0001$.

- And MAP estimates with one variable “shrunk” by 0.01:

$$w^1 = \begin{bmatrix} 100 \\ 0 \end{bmatrix}, \quad w^2 = \begin{bmatrix} 99.99 \\ 0.01 \end{bmatrix}.$$

- L2-regularization **strongly prefers** w^2 .

$$\|w^1\|^2 = 10000, \quad \|w^2\|^2 = 99.99^2 + 0.01^2 = 9998.0001 + 0.0001 = 9998.0002,$$

- It focuses on **decreasing largest variables and making magnitudes similar**.

L1-Regularization vs. L2-Regularization

- Why does L1-regularization give sparsity but L2-regularization does not?
 - Don't they both shrink variables towards 0?
- Consider a 2-variable problem whose MLE is given by

$$w = \begin{bmatrix} 100 \\ 0.01 \end{bmatrix}.$$

which has $\|w\|^2 = 10000.0001$.

- And MAP estimates with one variable “shrunk” by 0.01:

$$w^1 = \begin{bmatrix} 100 \\ 0 \end{bmatrix}, \quad w^2 = \begin{bmatrix} 99.99 \\ 0.01 \end{bmatrix}.$$

- L1-regularization has no preference:

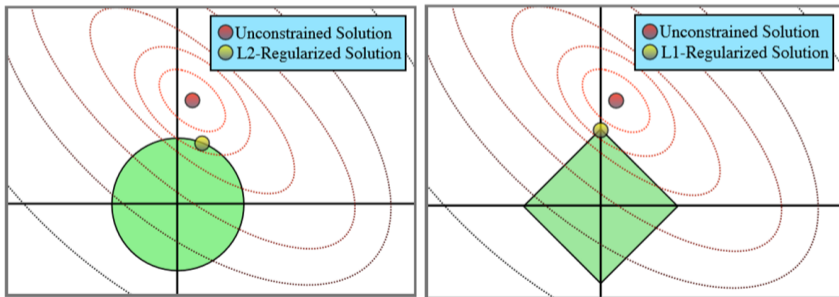
$$\|w^1\|_1 = 100, \quad \|w^2\|_1 = 99.99 + 0.01 = 100.$$

- So it will focus on **decreasing both variables** until the small one is zero.

L1-Regularization vs. L2-Regularization

- Another view on sparsity of L2- vs. L1-regularization using our constraint trick:

$$\operatorname{argmin}_{w \in \mathbb{R}^d} f(w) + \lambda \|w\|_p \Leftrightarrow \operatorname{argmin}_{w \in \mathbb{R}^d, \tau \in \mathbb{R}} f(w) + \lambda \tau \text{ with } \tau \geq \|w\|_p.$$



- Notice that L2-regularization has a rotational invariance.
 - This actually makes it **more sensitive to irrelevant features**.

Solving L1-Regularization Problems

- How can we minimize **non-smooth** L1-regularized objectives?
 - For example, the LASSO problem is

$$\operatorname{argmin}_{w \in \mathbb{R}^d} \frac{1}{2} \|Xw - y\|^2 + \lambda \|w\|_1.$$

and let's assume $X^T X$ is positive-definite, or we add L2-regularization.

- Either condition makes it strongly-convex.
- Use our trick to formulate as a quadratic program?
 - $O(d^2)$ or worse.
- Make a smooth approximation to the L1-norm?
 - **Destroys sparsity.**
- Formulate as a non-smooth convex optimization?
 - **Sub-linear** $O(1/\epsilon)$ convergence rate.

The Key to Faster Methods

- How can we beat the $O(1/\epsilon)$ lower bound in the first-order oracle model?
 - Make extra assumptions about the function/algorithm f .
- For L1-regularized least squares, we'll use that the objective has the form

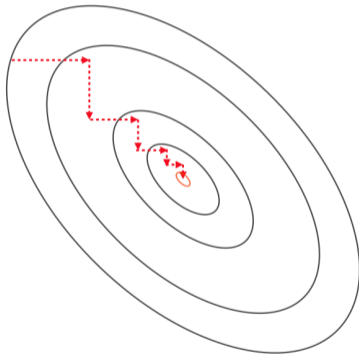
$$\operatorname{argmin}_{x \in \mathbb{R}^d} g(x) + \underbrace{\sum_{j=1}^d h_j(x_j)}_{\text{"separable"}}$$

that it's the **sum of a smooth function and a separable function**.

- The “non-smoothness” independently affects each coordinate.
- We can obtain $O(\log(1/\epsilon))$ rates for strongly-convex functions of this form.

Coordinate Optimization

- An algorithm that can achieve this property is **coordinate optimization**:
 - At each iteration we **only update one variable**.



- It's **easy to deal with the positive/negative cases for one variable**.

Coordinate Optimization

- In **coordinate optimization** we select one variable j_t and update this variable,

$$x_{j_t}^{t+1} = x_{j_t}^t + \gamma_t,$$

for some scalar γ_t .

- We can write this as an update to all variables using

$$x^{t+1} = x^t + \gamma_t e_{j_t},$$

where e_{j_t} has a zero in every position except j_t .

$$e_3^T = [0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0]$$

- Classic choices for **picking the variable j_t** to update:
 - **Cyclic** (in order), **random**, and **greedy**.
- Classic choices for **choosing the step-size γ_t** :
 - **Gradient descent** (smooth functions), **line-search**, **exact optimization**.

Coordinate Optimization

- This is an obvious, old, and widely-used algorithm.
- But until 2010, we had little theory about *when* to use it.
 - For some applications it's great, for other it's terrible.
- 2010: randomized coordinate optimization is faster than gradient descent if:
 - We can do d coordinate updates for the cost of one gradient descent step.
- This applies to random or greedy coordinate selection.
- So when can we do coordinate updates d -times faster?

Problems Suitable for Coordinate Optimization

- Coordinate update is d times faster than gradient update for
 - ① Separable functions,

$$f(x) = \sum_{j=1}^d h_j(x_j).$$

- We have $\nabla_j f(x) = h'_j(x_j)$ which costs $O(1)$, while $\nabla f(x)$ costs $O(d)$.
- The h_j can be minimized independently, so coordinate optimization is obvious choice.

Problems Suitable for Coordinate Optimization

- Coordinate update is d times faster than gradient update for
 - Composition of smooth function with affine map,

$$f(x) = g(Ax + b),$$

for a matrix A , a vector b , and smooth function g with cost of $O(n)$.

(includes least squares and logistic regression)

- The partial derivatives have the form

$$\nabla_j f(x) = \nabla a_j^T g(Ax + b).$$

- If we have Ax , this costs $O(n)$ instead of $O(nd)$ for the full gradient.
- We can track the product Ax^t as we go with $O(n)$ cost,

$$Ax^{t+1} = A(x^t + \gamma_t e_{j_t}) = \underbrace{Ax^t}_{\text{old value}} + \gamma_t \underbrace{Ae_{j_t}}_{O(n)},$$

Problems Suitable for Coordinate Optimization

- Coordinate update is d times faster than gradient update for
 - ③ Functions depending on smooth functions of variable pairs,

$$f(x) = \sum_{i=1}^d \sum_{j=1}^d g_{ij}(x_i, x_j).$$

- Cost of gradient is $O(d^2)$ but cost of partial derivative is $O(d)$.
(each variable only appears in d out of the d^2 functions g_{ij})
- Includes quadratics
- Includes graph-based semi-supervised learning from 340.
- Includes many variational inference methods (later in course).
- Note that it's still d times faster if we combine all three types:

$$f(x) = g(Ax) + \sum_{j=1}^d h_j(x_j) + \sum_{i=1}^d \sum_{j=1}^d g_{ij}(x_i, x_j),$$

which includes L1-regularized least squares.

Convergence Rate of Coordinate Optimization

- To analyze coordinate optimization, let's first assume each $\nabla_j f$ is L -Lipshitz,

$$|\nabla_j f(x + \gamma e_j) - \nabla_j f(x)| \leq L|\gamma|,$$

which for \mathcal{C}^2 functions is equivalent to $\nabla_{ii}^2 f(x) \leq L$ for all i .

(diagonals of Hessian are bounded)

- This is not a stronger assumption: it's implied by the gradient being Lipschitz.
- We'll also assume that we use the update

$$x^{t+1} = x^t - \frac{1}{L} \nabla_{j_t} f(x^t) e_{j_t},$$

which is a gradient step along the coordinate with a step-size of $1/L$.

Convergence Rate of Coordinate Optimization

- The **coordinate-wise** L -Lipschitz assumption implies that

$$f(y) \leq f(x) + \nabla_j f(x)(y - x)_j + \frac{L}{2}(y - x)_j^2,$$

for any x and y that only differ in coordinate j .

- If we plug in our previous analysis of gradient descent we get

$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L} |\nabla_{j_t} f(x^t)|^2,$$

a progress bound based on **only updating coordinate j_t** .

- Bound also holds for exact update, uglier-but-similar bounds hold for line-searches.

Convergence Rate of Randomized Coordinate Optimization

- Our bound for updating coordinate j_t is

$$f(x^{t+1}) \leq f(x^t) - \frac{1}{2L} |\nabla_{j_t} f(x^t)|^2,$$

but we need to pick j_t giving sufficient progress.

- Let's consider expected progress with random selection, $p(j_t) = 1/d$,

$$\mathbb{E}[f(x^{t+1})] \leq \mathbb{E} \left[f(x^t) - \frac{1}{2L} |\nabla_{j_t} f(x^t)|^2 \right] \quad (\text{expectation wrt } j_t)$$

$$= \sum_{j=1}^d \frac{1}{d} \left[f(x^t) - \frac{1}{2L} |\nabla_j f(x^t)|^2 \right] \quad (p(j) = 1/d)$$

$$= f(x^t) - \frac{1}{2dL} \sum_{j=1}^d |\nabla_j f(x^t)|^2$$

$$= f(x^t) - \frac{1}{2dL} \|\nabla f(x^t)\|^2.$$

Convergence Rate of Randomized Coordinate Optimization

- Our guaranteed progress bound for randomized coordinate optimization,

$$\mathbb{E}[f(x^{t+1})] \leq f(x^t) - \frac{1}{2Ld} \|\nabla f(x^t)\|^2.$$

- If we subtract $f(x^*)$ and use **strong-convexity** or PL,

$$\mathbb{E}[f(x^{t+1})] - f(x^*) \leq \left(1 - \frac{\mu}{dL}\right) [f(x^t) - f(x^*)].$$

- Which we'll show implies a linear convergence rate.

Summary

- **Optimization zoo** for minimizing continuous functions.
- **Weaker assumptions** for gradient descent:
 - Lipschitz-continuity, PL inequality, practical step-sizes.
- **L1-regularization**: feature selection as convex optimization.
- **Coordinate optimization**: for problems with fast single-variable updates.

- Next time: going beyond L1-regularization to “group sparsity”.

Bonus Slide: Descent Lemma

- Let ∇f be L -Lipschitz continuous, and define $g(\alpha) = f(x + \alpha z)$ for a scalar α .

$$f(y) = f(x) + \int_0^1 \nabla f(x + \alpha(y - x))^T (y - x) d\alpha \quad (\text{fund. thm. calc.})$$

$$= f(x) + \nabla f(x)^T (y - x) + \int_0^1 (\nabla f(x + \alpha(y - x)) - \nabla f(x))^T (y - x) d\alpha$$

$$\text{(CS ineq.)} \leq f(x) + \nabla f(x)^T (y - x) + \int_0^1 \|\nabla f(x + \alpha(y - x)) - \nabla f(x)\| \|y - x\| d\alpha$$

$$\text{(Lipschitz)} \leq f(x) + \nabla f(x)^T (y - x) + \int_0^1 L \|x + \alpha(y - x) - x\| \|y - x\| d\alpha$$

$$= f(x) + \nabla f(x)^T (y - x) + \int_0^1 L\alpha \|y - x\|^2 d\alpha$$

$$= f(x) + \nabla f(x)^T (y - x) + \frac{L}{2} \|y - x\|^2.$$

Bonus Slide: Strong-Convexity Lemma

- If $g(x) = f(x) - \frac{\mu}{2}\|x\|^2$ is convex then from C^1 definition of convexity

$$g(y) \geq g(x) + \nabla g(x)^T (y - x)$$

or that

$$f(y) - \frac{\mu}{2}\|y\|^2 \geq f(x) - \frac{\mu}{2}\|x\|^2 + (\nabla f(x) - \mu x)^T (y - x),$$

which gives

$$\begin{aligned} f(y) &\geq f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2}\|y\|^2 - \mu x^T y + \frac{\mu}{2}\|x\|^2 \\ &= f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2}\|y - x\|^2. \end{aligned}$$

Bonus Slide: PL Inequality for Least Squares

- Least squares can be written as $f(x) = g(Ax)$ for a σ -strongly-convex g and matrix A , we'll show that the PL inequality is satisfied for this type of function.
- The function is minimized at some $f(y^*)$ with $y^* = Ax$ for some x , let's use $\mathcal{X}^* = \{x | Ax = y^*\}$ as the set of minimizers. We'll use x_p as the "projection" (defined next lecture) of x onto \mathcal{X}^* .

$$\begin{aligned}
 f^* = f(x_p) &\geq f(x) + \langle \nabla f(x), x_p - x \rangle + \frac{\sigma}{2} \|A(x_p - x)\|^2 \\
 &\geq f(x) + \langle \nabla f(x), x_p - x \rangle + \frac{\sigma\theta(A)}{2} \|x_p - x\|^2 \\
 &\geq f(x) + \min_y \left[\langle \nabla f(x), y - x \rangle + \frac{\sigma\theta(A)}{2} \|y - x\|^2 \right] \\
 &= f(x) - \frac{1}{2\theta(A)\sigma} \|\nabla f(x)\|^2.
 \end{aligned}$$

- The first line uses strong-convexity of g , the second line uses the "Hoffman bound" which relies on \mathcal{X}^* being a polyhedral set defined in this particular way to give a constant $\theta(A)$ depending on A that holds for all x (in this case it's the smallest non-zero singular value of A), and the third line uses that x_p is a particular y in the min.