SVAN 2016 Mini Course: Stochastic Convex Optimization Methods in Machine Learning

Mark Schmidt

University of British Columbia, May 2016

www.cs.ubc.ca/~schmidtm/SVAN16

Some images from this lecture are taken from Google Image Search.

Last Time: Convex Functions

- Last time we discussed convex functions:
  - All local minima are global minima (and no saddle points).
  - Three definitions of convex functions (depending on differentiability):

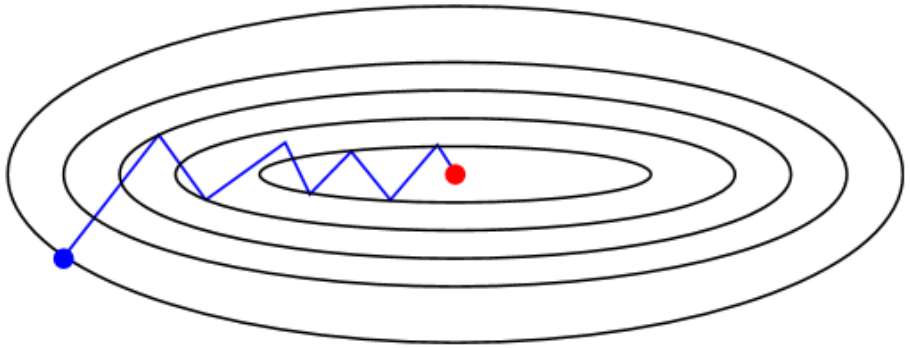
1. $f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$ for all x and y, and $0 \leq \theta \leq 1$.
2. Once-differentiable and $f(y) \geq f(x) + \nabla f(x)^T (y - x)$ for all x and y.
3. Twice-differentiable and $\nabla^2 f(x) \succeq 0$ for all x (symmetric positive semi-definite)

  - We discussed ways to show functions are convex:
    - Show one of the above holds.
    - Use operations that preserve convexity.
      - Non-negative sum, composition with affine function, maximum.

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



- Cost of algorithm scales linearly with number of variables 'd':
  - E.g., 't' iterations costs O(ndt) for least squares, logistic regression, etc.
    - Note that the input size is O(nd).
  - For $t < d$, faster than $O(nd^2 + d^3)$ of least squares and Newton's method.

    Faster in high-dimensions for small 't'.

Last Time: Convergence Rate of Gradient Descent

- We asked "how many iterations 't' before we have an accuracy ε?"
- We assumed strong-convexity and strong-smoothness:

$$\mu I \preceq \nabla^2 f(x) \preceq LI \quad \text{for all } x \text{ and } 0 < \mu \leq L < \infty.$$

(I: identity matrix)

($A \succeq B$ means that $y^T A y - y^T B y \geq 0$ for all y)

So $LI \succeq \nabla^2 f(x)$ means that $y^T(LI)y - y^T\nabla^2 y \geq 0$ or $L\|y\|^2 \geq y^T\nabla^2 f(x)y$ for all y.

- By using multivariate 2nd-order Taylor expansion,

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

for some z for any x and y,

we showed linear convergence rate which implies $t = O(\log(1/\varepsilon))$.

Last Time: Gradient Descent Theory and Practice

- We discussed further properties of gradient descent:
  - "Strong-smoothness" weakened to "gradient is L-Lipschitz continuous".
    - And only along the line segments between x^t and x^{t+1}.
  - No need to know 'L':
    - Adaptive step-size, Armijo line-search, or exact step-size.
  - "Strong-convexity" is implied if we have $f(x) + \lambda||x||^2$ and 'f' is convex.
    - If 'f' is not convex, convergence rate only holds near solution.
- We overviewed methods with better performance:
  - Nesterov's accelerated-gradient method.
  - Approximations to Newton's method.

How Hard is Optimization?

- Consider a generic optimization problem:

$$\underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \; f(x)$$

- Assume that a solution 'x^*' exists.
- Assume a "black-box" optimization algorithm:
  - At step 't', algorithms chooses parameters x^t and receives $f(x^t)$.
- How many steps does it take before we find ε-optimal solution?

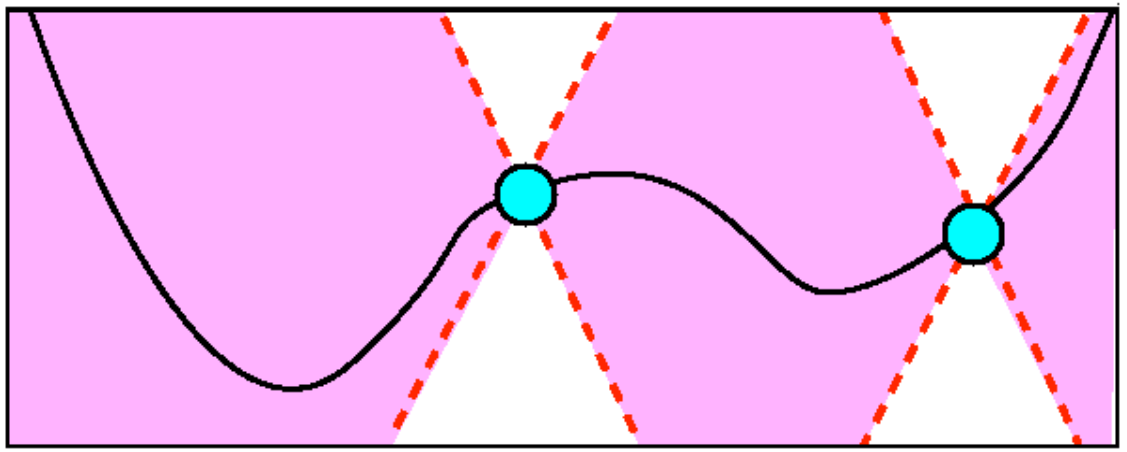
$$f(x^t) - f(x^*) \leq \varepsilon$$

- General function: impossible!

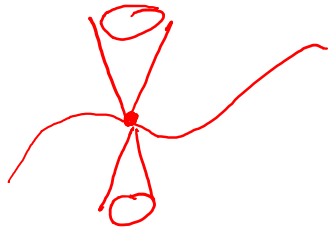
How Hard is Optimization?

- We need to make some assumptions about the function
- Typically, we assume function or gradient can't change too quickly.
  - E.g., function 'f' is Lipschitz-continuous:

$$|f(x) - f(y)| \leq L\|x - y\| \quad \text{for some 'L' and all 'x' and 'y'}$$



In two-dimensions, Lipschitz rules out cones:



  - Over $[0,1]^d$, now it's possible to solve the problem in $O(1/\varepsilon^d)$:
    - Exponential in dimensionality, but a small assumption made a bit difference.

Continuous Optimization Zoo

Assumptions	Algorithm	Rate
f is L-Lipschitz, x is bounded	Grid-search	$O(1/\epsilon^d)$
f is convex but non-smooth	Sub-gradient	$O(1/\epsilon^2)$
Smooth approximation to non-smooth f (f is convex)	Gradient	$O(1/\epsilon^2)$
	Nesterov	$O(1/\epsilon)$
∇f is L-Lipschitz, f is convex	Gradient	$O(1/\epsilon)$
	Nesterov	$O(1/\sqrt{\epsilon})$
f is strongly-convex but non-smooth	Sub-gradient	$O(1/\epsilon)$
∇f is L-Lipschitz, f is μ-strongly convex	Gradient	$O(\log(\frac{1}{\epsilon}))$
	Nesterov	$O(\log(\frac{1}{\epsilon}))$
∇f is L-Lipschitz, f is μ-strongly convex, $\nabla^2 f$ is M-Lipschitz	Quasi-Newton	$o(\log(\frac{1}{\epsilon}))$

sublinear

linear

superlinear

convexity

better algorithm

smoothness

strong-convexity

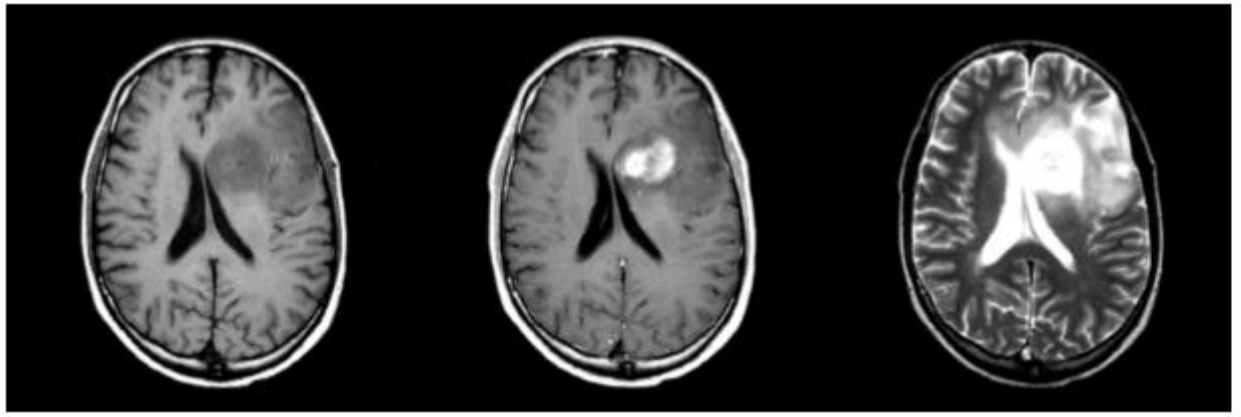
strong convexity

approximating 2nd derivatives, but cost is $O(d^2)$.

(pause)

Motivation: Automatic Brain Tumor Segmentation

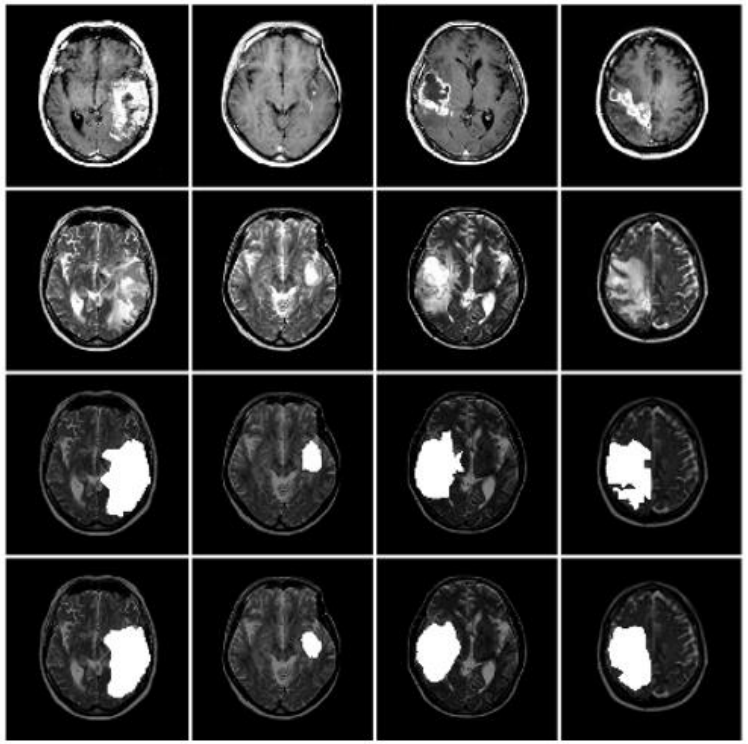
- Task: identifying tumours in multi-modal MRI data.

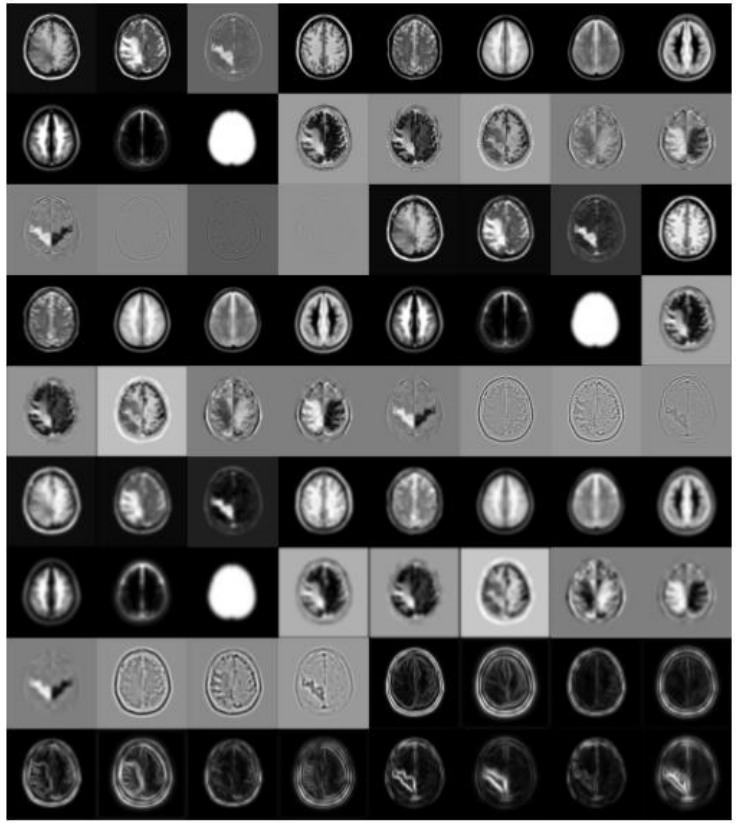


- Applications:
  - image-guided surgery.
  - radiation target planning.
  - quantifying treatment response
  - discovering growth patterns.

Motivation: Automatic Brain Tumor 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).



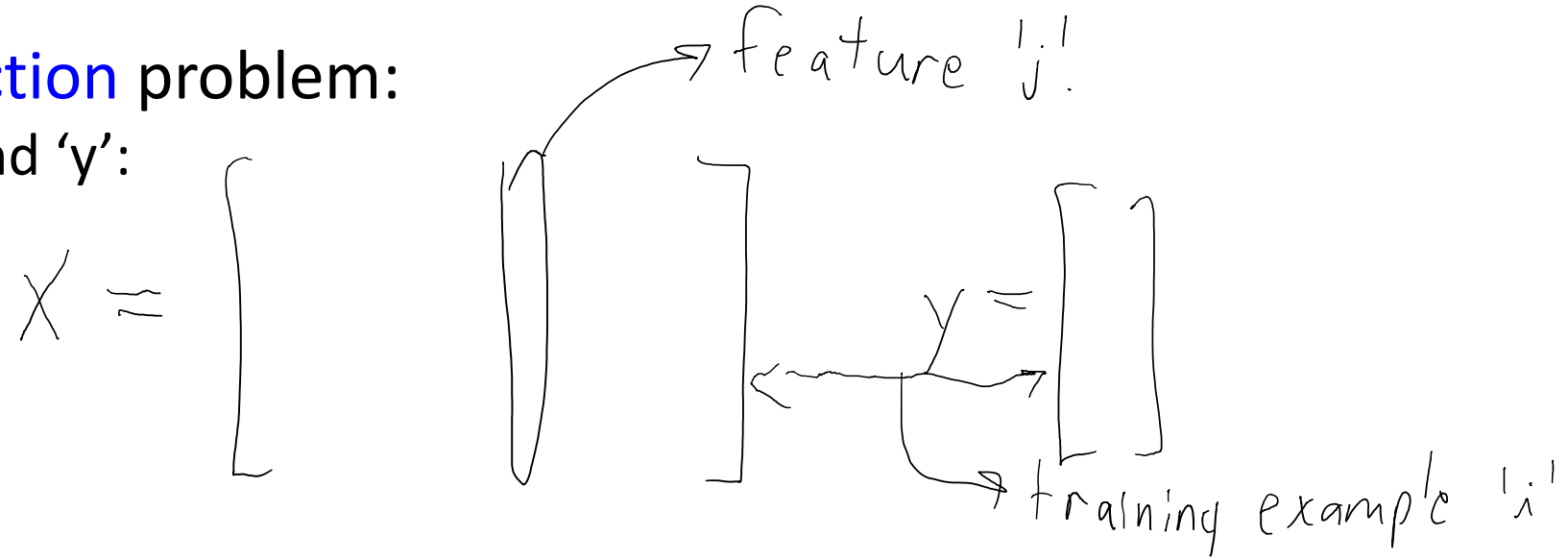


Motivation: Automatic Brain Tumor Segmentation

- Logistic regression was 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, stagewise,…
    - In this application, these are too slow.

Feature Selection

- General feature selection problem:
  - Given our usual 'X' and 'y':



  - We think some features/columns of 'X' are irrelevant for predicting 'y'.
- We want to fit a model that uses the 'best' set of features.
  - Special case: choosing 'best' basis from a set of possible bases.
- One of most important problems in ML/statistics, but very very messy.
  - Can be difficult to define what 'relevant' means.
  - For now, a feature is 'relevant' if it helps predict y_i from x_i.

L1-Regularization

- Popular approach to feature selection is L1-regularization:

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \ \frac{1}{2}\|Xw - y\|^2 + \lambda \|w\|_1 \qquad \|w\|_1 = \sum_{j=1}^{d} |w_j|$$

- Written above for squared loss, but can be used for any loss.
- Advantages:
  - Fast: can apply to large datasets, just minimizing convex function.
  - Reduces overfitting because it simultaneously regularizes.
- Disadvantage:
  - Prone to false positives, particularly if you pick λ by cross-validation.
  - Not unique: there may be infinite solutions.

L1-Regularization

- Key property of L1-regularization: if λ is large, solution w^* is sparse:
  - w^* has many values that are exactly zero.
- What this has to do with feature selection:

$$\hat{y}_i = w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3} + w_4 x_{i4} + w_5 x_{i5}$$

- If w = [0 0 3 0 -2], then:

$$\hat{y}_i = 0 x_{i1} + 0 x_{i2} + 3 x_{i3} + 0 x_{i4} + (-2) x_{i5}$$
$$= 3 x_{i3} - 2 x_{i5}$$ (features {1,2,4} are ignored)

- Why does L1-regularization give sparsity but not L2-regularization?

Why not just threshold ‘w’?

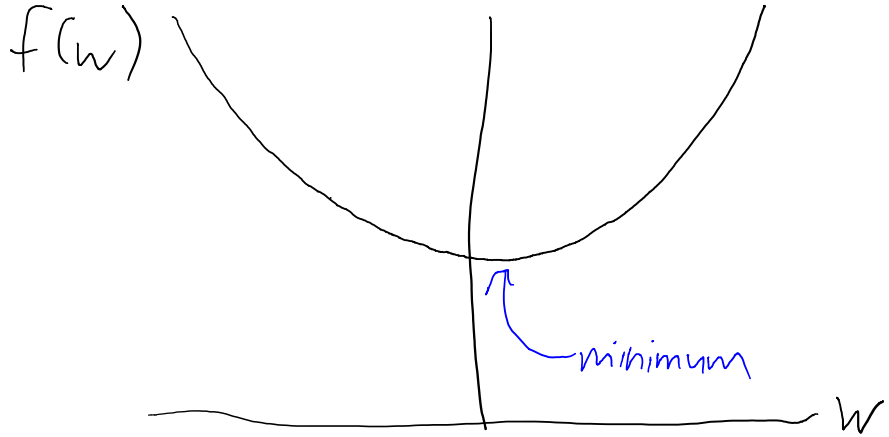
- Why not just compute least squares ‘w’ and threshold?
  - You can show some nice properties of this, but it does some silly things:
    - Let feature 1 be an irrelevant feature, and assume feature 2 is a copy of feature 1.
    - Without regularization, could have $w_1 = -w_2$ with both values arbitrarily large.
- Why not just compute L2-regularized ‘w’ and threshold?
  - Fixes the above problem, but still does weird things:
    - Let feature 1 be irrelevant and feature 2 be relevant.
    - Assume feature 3 is also relevant, and features 4:d are copies of feature 3.
    - For ‘d’ large enough, L2-regularization prefers irrelevant feature ‘1’ or relevant 3:d. (L1-regularization should pick at least one among 3:d for any ‘d’.)
- (I’m not saying L1-regularization doesn’t do weird things, too.)
- If features are orthogonal, thresholding and L1 are equivalent.
  - But feature selection is not interesting in this case.

Sparsity and Least Squares

- Consider 1D least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - wx_i)^2$$

- This is a convex 1D quadratic function of 'w' (i.e., a parabola):



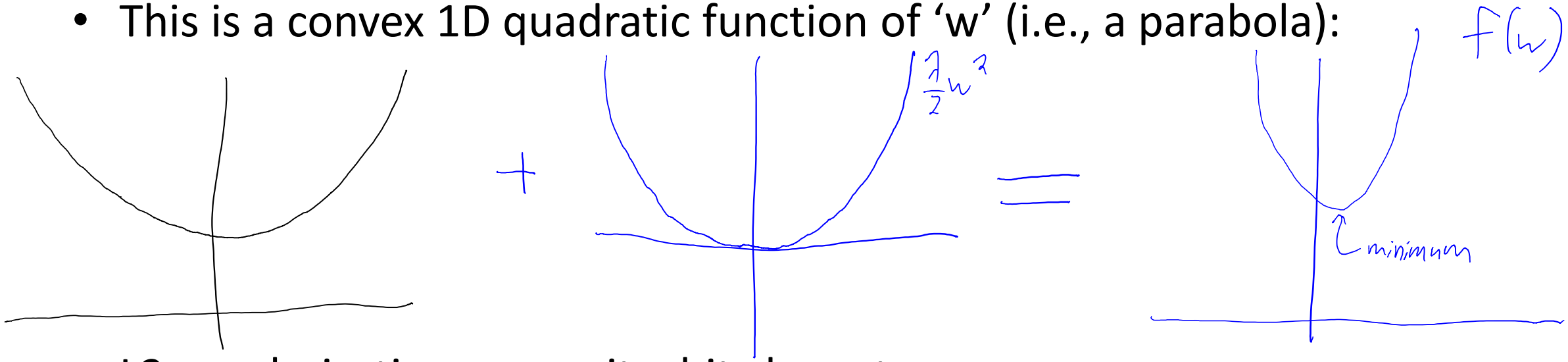
- This variable does not look relevant (minimum is close to 0).
  - If it's really irrelevant, minimum will move to 0 as 'n' goes to infinity.
  - But for finite 'n', minimum of parabola is unlikely to be exactly zero.

Sparsity and L2-Regularization

- Consider 1D L2-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - wx_i)^2 + \frac{\lambda}{2}w^2$$

- This is a convex 1D quadratic function of 'w' (i.e., a parabola):



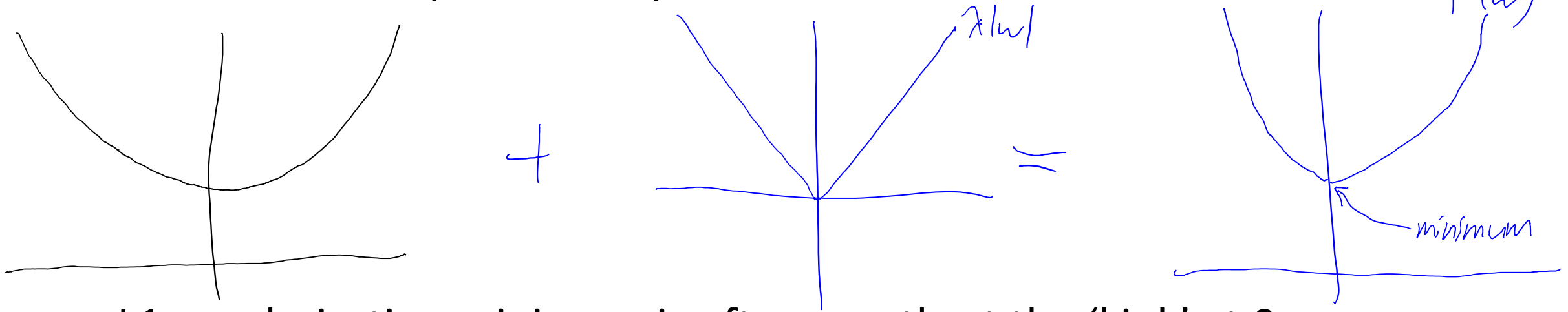
- L2-regularization moves it a bit closer to zero.
  - But there is nothing special about being 'exactly' zero.
  - Unless cost is flat at zero, L2-regularization always sets 'w_j' non-zero.

Sparsity and L1-Regularization

- Consider 1D L1-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - wx_i)^2 + \lambda|w| = \begin{cases} \frac{1}{2}\sum_{i=1}^{n}(y_i - wx_i)^2 + \lambda w, & w \geq 0 \\ \frac{1}{2}\sum_{i=1}^{n}(y_i - wx_i)^2 - \lambda w, & w < 0 \end{cases}$$

- This is a convex piecewise-quadratic function of 'w' with 'kink' at 0:

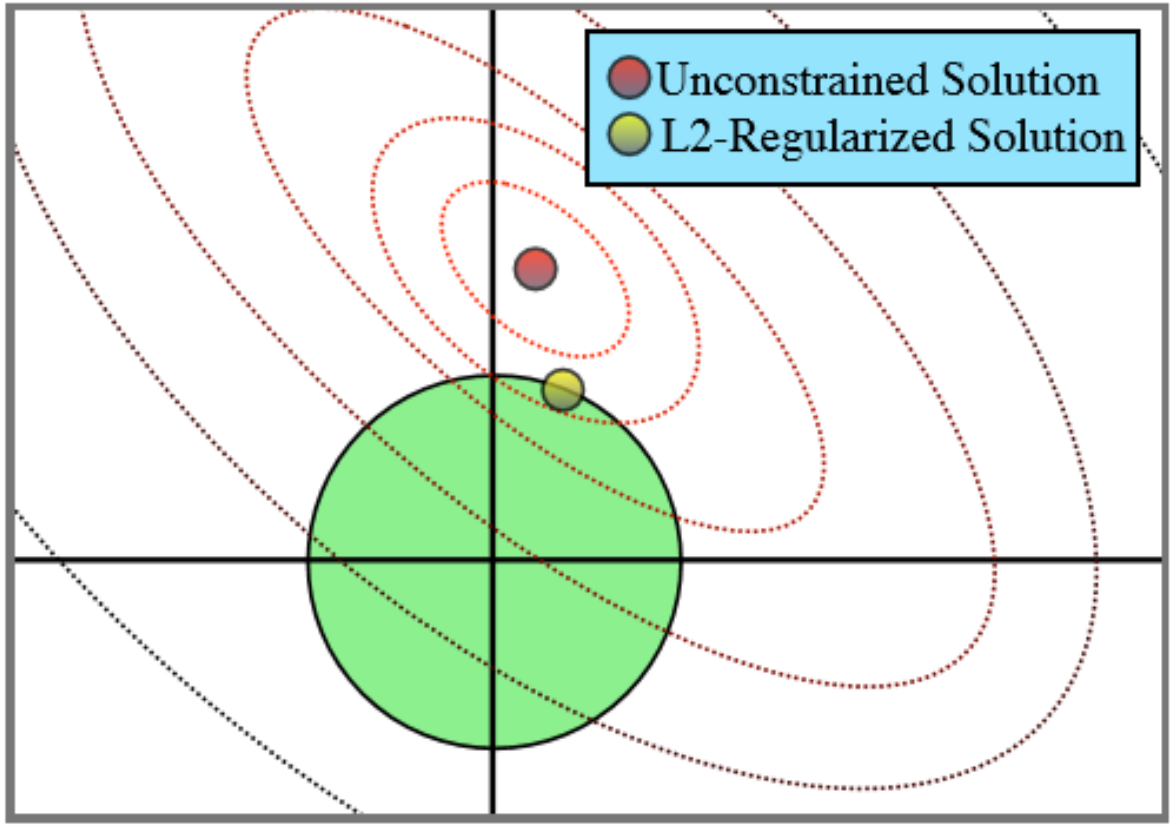


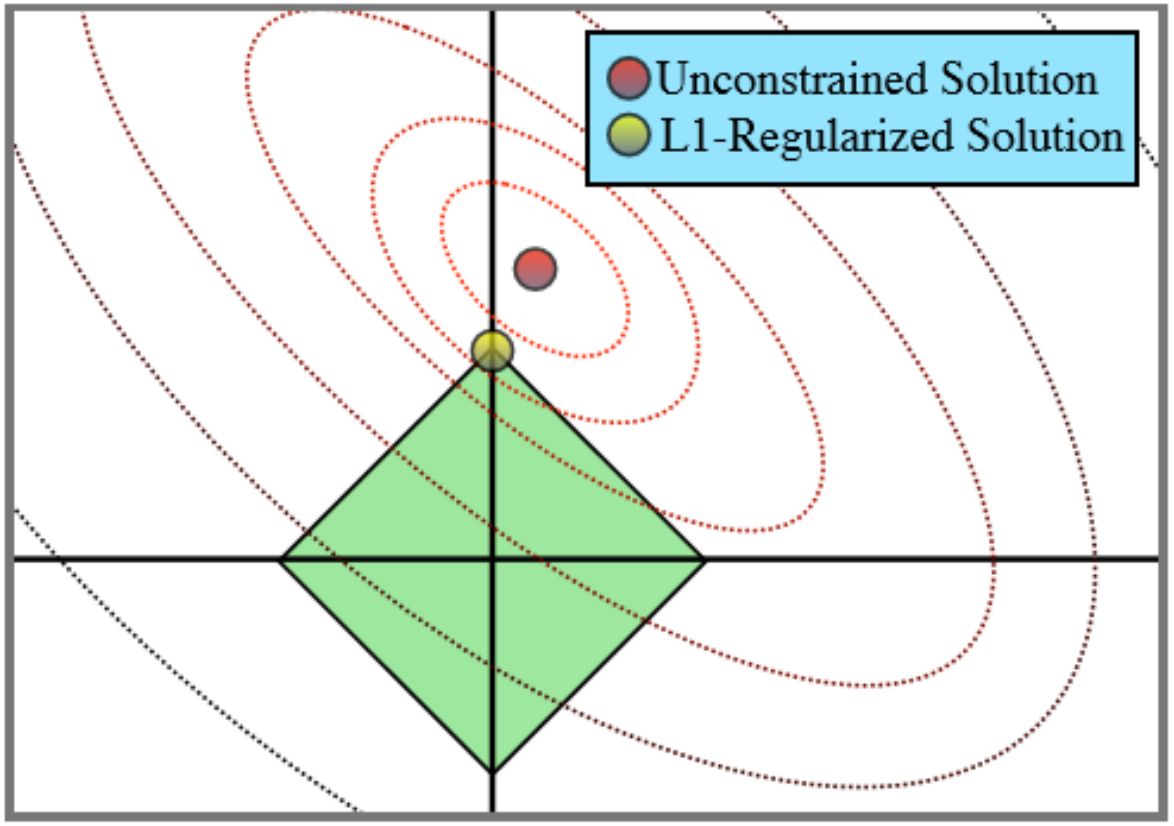
- L1-regularization minimum is often exactly at the 'kink' at 0:
  - It sets the feature to exactly 0, removing it from the model.
  - Big λ means kink is 'steep'. Small λ makes 0 unlikely to be minimum.

Where does sparsity come from?

- Another view on sparsity of L2- vs. L1-regularization:

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \ \frac{1}{2}\|Xw - y\|^2 + \lambda \|w\|_p \iff \underset{w \in \mathbb{R}^d, r \in \mathbb{R}}{\operatorname{argmin}} \ \frac{1}{2}\|Xw - y\|^2 + \lambda r \quad \text{subject to } r \geq \|w\|_p$$





L1-Regularization: Discussion

- "Sample complexity" [Ng, 2004]:
  - L2-regularization: you can learn with linear number of irrelevant features.
  - L1-regularization: you can learn with exponential number of irrelevant.
- "Elastic net":
  - Use both L2-regularization and L1-regularization.
  - Makes problem strongly-convex, so it has a unique solution.
- "Bolasso":
  - Run L1-regularization on boostrap samples.
  - Take features that are non-zero in all samples: fewer false positives.
- Non-convex regularizers:
  - Less sensitive to false positives, but solving optimization is NP-hard.

for example, $\sum_{j=1}^{d} \sqrt{w_j}$ will give higher-sparsity.

Solving L1-Regularization Problems

- How can we minimize non-smooth L1-regularized objectives?

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

  - And let's assume X^TX is positive-definite, or we add L2-regularization.
    - Either conditions makes it strongly-convex.
- Use our trick to formulate as a quadratic program?
  - $O(d^2)$ or worse.
- Formulate as non-smooth convex optimization?
  - Sub-linear $O(1/\varepsilon)$ convergence rate.
- Make a smooth approximation to L1-norm?
  - Destroys sparsity.

Solving L1-Regularization Problems

- Key insight: this is not a general non-smooth convex function.
  - We can use structure to get large-scale O(log(1/ε)) methods.
- We can write it as:

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \; g(x) + h(x) \quad \text{where } 'g' \text{ is smooth and } 'h' \text{ is "simple"}$$

  - This lets us apply proximal-gradient methods (next lecture).
- We can also write it as:

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \; g(x) + \sum_{j=1}^{d} h_j(x_j) \quad \text{where } 'g' \text{ is smooth.}$$

"separable"

  - This lets us apply coordinate optimization methods (this lecture)

Coordinate Optimization

- We want to optimize a differentiable function:

$$\operatorname{argmin}_{x \in \mathbb{R}^d} f(x)$$

- Coordinate optimization:
  - At each iteration 't', we update one variable 'j_t':

$$x^{t+1} = x^t + \alpha_t e_{j_t} \quad \text{where} \quad e_j = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \leftarrow \text{position } 'j'$$

"just change variable j_t"

- How do we pick the variable 'j_t' to update?
  - Classic choices: cyclic, random, and greedy.
- How do we update the variable we chose?
  - Classic choices: constant step-size, line-search, exact optimization.

Coordinate Optimization

- This is an obvious, old, and widely-used algorithm.
- But until ~2010, we had no theory about when to use it.
  - For some applications it works great, for some applications it's terrible.
- Key insight in ~2010:
  - If you can do 'd' coordinate updates for the cost of one gradient update, then randomized coordinate optimization is faster than gradient descent.
  - Applies to random or greedy selection and 1/L or exact updates.
- When is this true?
  - Simplest case is separable function, $\Sigma_i f_i(x_i)$, like L2- or L1-regularization.
  - There are two more complicated classes…

Problems Suitable for Coordinate Descent

- Coordinate update is n times faster than gradient update for:

$$\underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \; f(x) = g(Ax)$$

  - Where 'g' is smooth/cheap but bottleneck is multiplication by 'A'.
  - For example, least squares and logistic regression.
  - Key idea: can track the product Ax^t after single-coordinate updates,

$$Ax^{t+1} = A(x^t + \alpha_t e_{j_t}) = Ax^t + \alpha_t Ae_{j_t}$$

old value

$O(n)$ because e_{j_t} has one non-zero.

  - And since 'g' is cheap you get gradient for random coordinate by:

$$\nabla f(x) = A^T \nabla g(Aw) \qquad \nabla_j f(x) = \nabla g(Aw)^T a_j$$

column of A: cost is $O(n)$. → Compare to gradient which costs $O(nd)$

- The other class where coordinate update is 'n' times faster:

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{d} \sum_{j=1}^{d} g_{ij}(x_i, x_j)$$ (e.g., graph-based semi-supervised learning)

Analysis of Coordinate Optimization

- For gradient descent we assume gradient is Lipschitz continuous:

$$\|\nabla f(x) - \nabla f(y)\| \leq L_f \|x - y\| \qquad \nabla^2 f(x) \preceq L_f I$$

- For coordinate optimization we assume coordinate-wise L-Lipschitz:

$$|\nabla_j f(x + \alpha e_j) - \nabla_j f(x)| \leq L|\alpha| \qquad \nabla^2_{ii} f(x) \leq L$$

- Note that neither of these is stronger:
  - If gradient is L_f-Lipschitz, then its coordinate-wise L_f-Lipschitz, so $L \leq L_f$.
  - If coordinate-wise L-Lipschitz, then gradient is dL-Lipschitz, so $L_f \leq dL$.
- Gradient descent requires $O((L_f/\mu)\log(1/\varepsilon))$ iterations.
- Coordinate optimization requires $O(d(L/\mu)\log(1/\varepsilon))$ iterations.
  - This is slower because $L_f \leq dL$.
  - But this faster is if iterations are 'd' times cheaper, because $L \leq L_f$.

Coordinate Optimization Progress Bound

- First let's assume a step-size of 1/L:

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

Coordinate-wise Lipschitz implies $f(y) \leq f(x) + \nabla_j f(x)(y-x) + \frac{L}{2}\|y-x\|^2$ for any j and all x and y that differ in one variable

Use $x = x^t$ and $y = x^{t+1}$ to get

so $y - x = \frac{\nabla_{j_t} f(x^t)}{L} e_{j_t}$

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

This also holds for exact optimization, since $\min_y \{f(y)\} \leq f(x^{t+1})$

This holds for any choice of j_t.

Random Selection Rule

- Our bound for any coordinate: $f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}|\nabla_{j_t} f(x^t)|^2$
- Let's consider random selection of each 'j' with probability 1/d:

$$\begin{aligned}
E[f(x^{t+1})] &\leq E\left[f(x^t) - \tfrac{1}{2L}|\nabla_{j_t} f(x^t)|^2\right] && \text{(expectation with respect to } j_t\text{)} \\
&= E[f(x^t)] - \tfrac{1}{2L} E\left[|\nabla_{j_t} f(x^t)|^2\right] && \text{(expectation is linear)} \\
&= f(x^t) - \tfrac{1}{2L}\sum_{j=1}^{d} p(j)|\nabla_j f(x^t)|^2 && \text{(definition of expectation)} \\
&= f(x^t) - \tfrac{1}{2L}\sum_{j=1}^{d} \tfrac{1}{d}|\nabla_{j_t} f(x^t)|^2 && \text{(using } p(j) = \tfrac{1}{d}\text{)} \\
&= f(x^t) - \tfrac{1}{2Ld}\sum_{j=1}^{d}|\nabla_{j_t} f(x^t)|^2 \\
&= f(x^t) - \tfrac{1}{2Ld}\|\nabla f(x^t)\|^2 && \text{(} \|v\|^2 = \sum_{j=1}^{d}|v_i|^2\text{)}
\end{aligned}$$

Analysis of Coordinate Optimization

- If 'f' is μ-strongly-convex, then we get a linear convergence rate:

$$\begin{aligned} E[f(x^{t+1}) - f(x^*)] &\leq f(x^t) - f(x^*) - \frac{1}{2Ld}\|\nabla f(x^t)\|^2 \\ &\leq f(x^t) - f(x^*) - \frac{\mu}{Ld}[f(x^t) - f(x^*)] \\ &= \left(1 - \frac{\mu}{Ld}\right)[f(x^t) - f(x^*)] \end{aligned}$$

(subtract $f(x^*)$ from both sides)

($-\|\nabla f(x^t)\|^2 \leq 2\mu(f(x^t) - f(x^0))$ from strong-convexity)

Analysis of Coordinate Optimization

- If 'f' is μ-strongly-convex, then we get a linear convergence rate:

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

(subtract $f(x^*)$ from both sides)

$$\leq f(x^t) - f(x^*) - \frac{\mu}{Ld}[f(x^t) - f(x^*)]$$

($-\|\nabla f(x^t)\|^2 \leq 2\mu(f(x^t) - f(x^0))$ from strong-convexity)

$$= \left(1 - \frac{\mu}{Ld}\right)[f(x^t) - f(x^*)]$$

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

(expectation with respect to j_{t-1})

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

iterated expectation ($\mathbb{E}_Y[\mathbb{E}_{X|Y}[X|Y]] = \mathbb{E}[X]$)

Analysis of Coordinate Optimization

- If 'f' is μ-strongly-convex, then we get a linear convergence rate:

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

(subtract $f(x^*)$ from both sides)

$$\leq f(x^t) - f(x^*) - \frac{\mu}{L_d}[f(x^t) - f(x^*)]$$

($-\|\nabla f(x^t)\|^2 \leq 2\mu(f(x^t) - f(x^0))$ from strong-convexity)

$$= (1 - \frac{\mu}{L_d})[f(x^t) - f(x^*)]$$

$$\mathbb{E}[\mathbb{E}[f(x^{t+1}) - f(x^*)]] = \mathbb{E}[(1 - \frac{\mu}{L_d})[f(x^t) - f(x^*)]]$$

(expectation with respect to j_{t-1})

iterated expectation ($\mathbb{E}_Y[\mathbb{E}_{X|Y}[X|Y]] = \mathbb{E}[X]$)

$$\mathbb{E}[f(x^{t+1}) - f(x^*)] = (1 - \frac{\mu}{L_d})\mathbb{E}[f(x^t) - f(x^*)]$$

apply recursively

$$\leq (1 - \frac{\mu}{L_d})^2[f(x^{t-1}) - f(x^*)]$$

$$\vdots$$

Finally giving $\mathbb{E}[f(x^k) - f(x^*)] \leq (1 - \frac{\mu}{L_d})^t[f(x^0) - f(x^*)]$

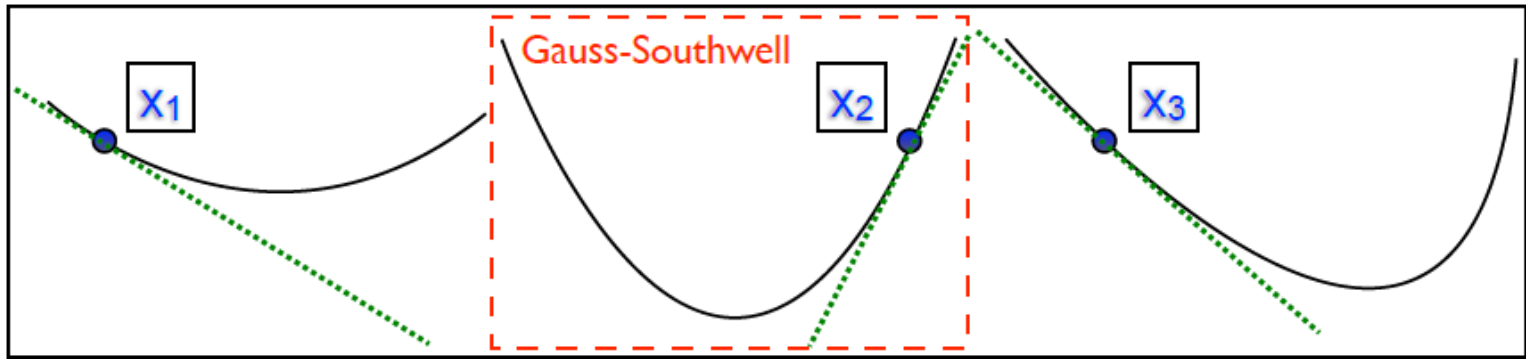
This implies we need $O(d\frac{L}{\mu}\log(\frac{1}{\epsilon}))$ iterations until $\mathbb{E}[f(x^k) - f(x^*)] \leq \epsilon$

Gauss-Southwell Selection Rule

- Our bound for any coordinate: $f(x^{t+1}) \leq f(x^t) - \frac{1}{2L}|\nabla_{j_t} f(x^t)|^2$
- The "best" coordinate to update is:

$$j_t \in \operatorname{argmax}_j \{|\nabla_j f(x^t)|\}$$

  - Called the 'Gauss-Southwell' or greedy rule.



  - You can derive a convergence rate by using that $|\nabla_{j_t} f(x^t)|^2 = \|\nabla f(x^t)\|_\infty^2$
  - Typically, this can't be implemented 'd' times faster than gradient method.
    - But some sparsity structures allow us to track the gradient.

Lipschitz Sampling and Gauss-Southwell-Lipschitz

- You can go faster than random with an L_j for each coordinate:

$$|\nabla_j f(x+\alpha e_j) - \nabla_j f(x)| \leq L_j |\alpha|$$

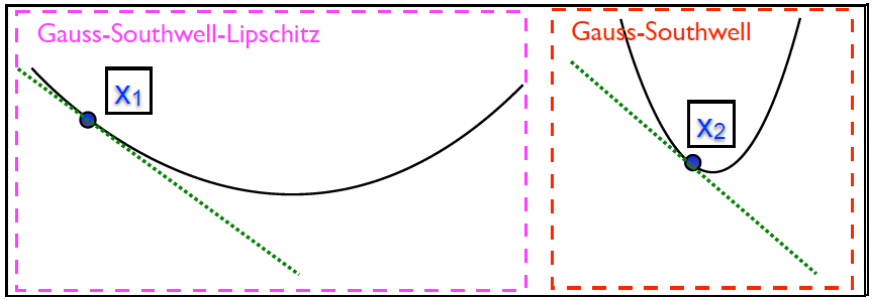
- If you sample j_t proportional to L_j, you can get a rate of:

$$\mathbb{E}[f(x^t) - f(x^*)] \leq \left(1 - \frac{\mu}{\bar{L} d}\right)^t [f(x^0) - f(x^*)] \quad \text{where } \bar{L} = \frac{1}{d}\sum_{j=1}^{d} L_j$$

  - Depends on average L_j instead of maximum L_j.

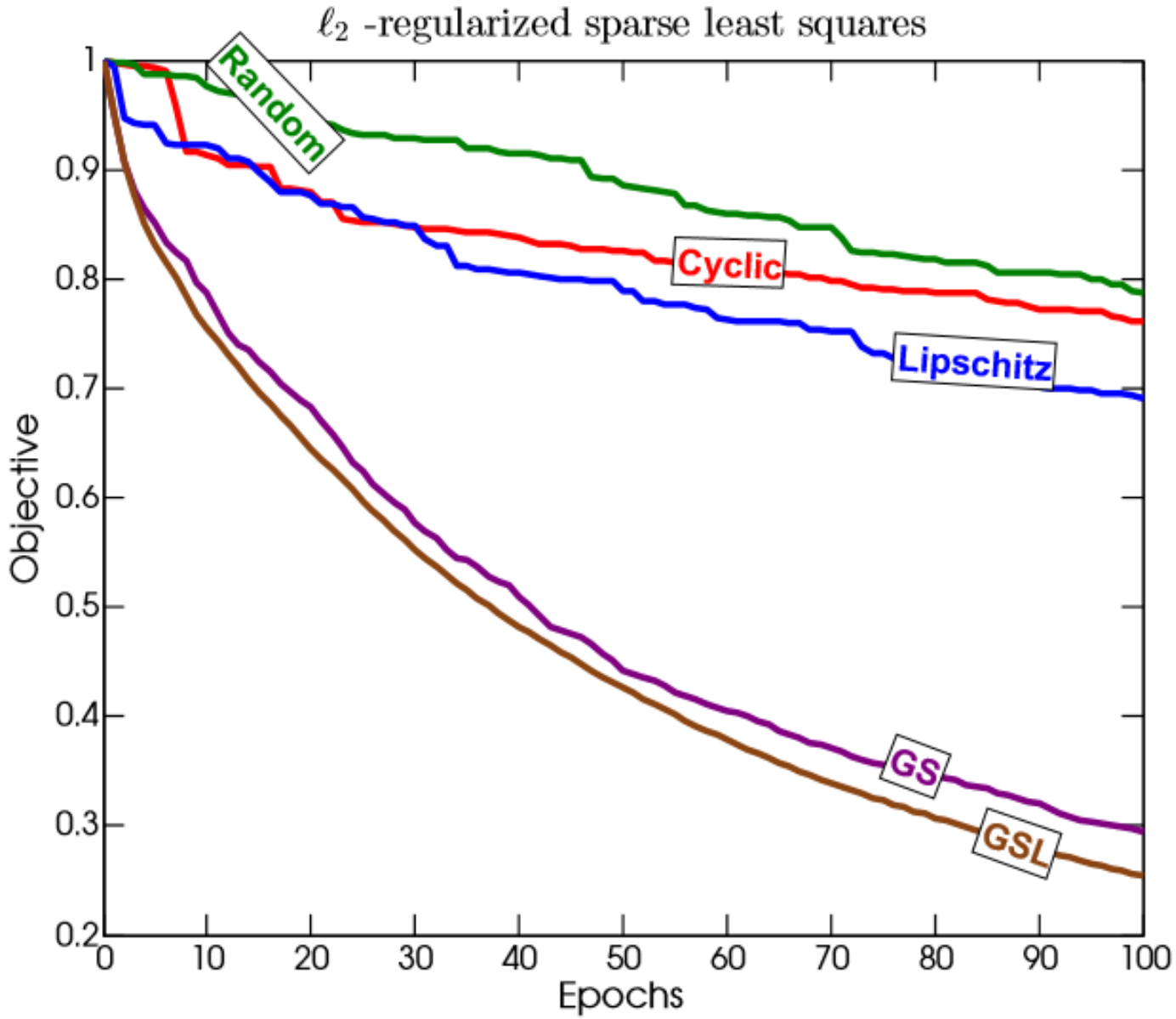
- The Gauss-Southwell-Lipschitz rule:

$$j_t \in \operatorname{argmax}_j \left\{ \frac{|\nabla_j f(x^t)|^2}{L_j} \right\}$$



  - Even faster, and optimal for quadratic functions.

Comparison of Coordinate Selection Rules



Coordinate Optimization for Non-Smooth Objectives

- Consider an optimization problem of the form:

$$\operatorname*{argmin}_{x \in \mathbb{R}^d} \; f(x) + \sum_{i=1}^{d} g_i(x_i)$$

(f: smooth; g_i: "separable")

- Assume:
  - 'f' is coordinate-wise L-Lipschitz continuous and μ-strongly convex.
  - 'h_i' are general convex functions (could be non-smooth).
  - You do exact coordinate optimization.
- For example, L1-regularized least squares:

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} \; \frac{1}{2}\|Xw - y\|^2 + \lambda \sum_{j=1}^{d} |x_i|$$

($f(x)$; $g_j(x_j) = \lambda |x_j|$)

- Linear convergence rate still holds (proof more complicated):

$$E[f(x^t) - f(x^*)] \leq \left(1 - \frac{\mu}{Ld}\right)^t [f(x^0) - f(x^*)]$$

- We can solve these non-smooth problems much faster than O(1/ε).

Summary

- Convex optimization zoo: rate of convergence for different methods.
- Feature selection: choosing set of relevant variables.
- L1-regularization: feature selection as convex optimization.
- Coordinate optimization: when updating single variable is fast.
- Coordinate optimization convergence rate analysis.
- Group L1-regularization encourages sparsity in variable groups.
- Structured sparsity encourages other patterns in variables.

- Next time: how do we encourage more complicated sparsity patterns?