STAT 542: Statistical Learning

Introduction to Numerical Optimization

Ruoqing Zhu, Ph.D. <rqzhu@illinois.edu>

Course Website: https://teazrq.github.io/stat542/

February 6, 2022

Department of Statistics
University of Illinois at Urbana-Champaign

Numerical Optimization

- This lecture gives a very brief introduction to some numerical optimization approaches, while most of them are for convex optimizations
- The goal is to have sufficient knowledge to deal with specific problems such as Lasso, SVM, etc.
- Reference:
  Boyd, Stephen, and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- Many of the figures in this lecture are taken from online sources. I may not include the reference for all of them. I want to thank all of them!

Overview

Convex Optimization

- The problem: minimizing a convex function in a convex set

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{minimize}} \quad & f(\boldsymbol{\beta}) \\ \text{subject to} \quad & g_i(\boldsymbol{\beta}) \leq 0, \quad i = 1, \ldots, m \\ & \mathbf{A}\boldsymbol{\beta} = b \end{aligned}$$

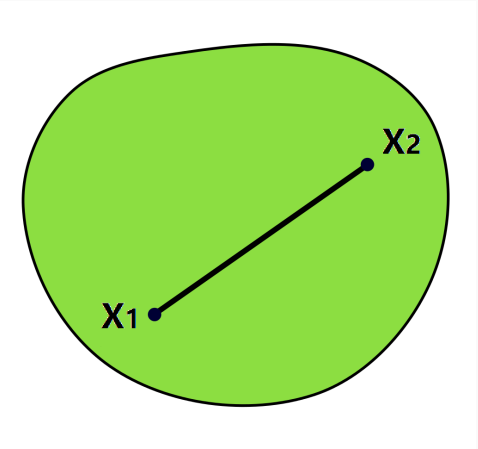
- Examples:
  - Linear regression: minimize $\frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$, subject to none.
  - Ridge regression: minimize $\frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$, subject to $\sum_{j=1}^{p} \beta_j^2 < s$
  - First principal component: maximize $\boldsymbol{\beta}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\boldsymbol{\beta}$, subject to $\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\beta} = 1$
- To be consistent with the notation in the literature, we will use $\mathbf{x}$ as the argument instead of using $\boldsymbol{\beta}$ in the objective function f.

Convex Optimization

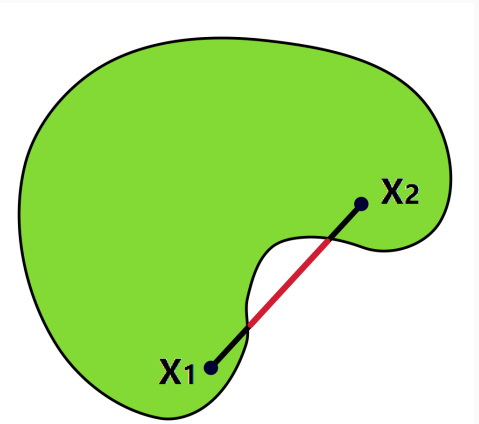
- What is a convex set $C \in \mathbb{R}^p$?

$$\mathbf{x}_1, \mathbf{x}_2 \in C \quad \Longrightarrow \quad \alpha \mathbf{x}_1 + (1-\alpha)\mathbf{x}_2 \in C, \quad \forall\ 0 \leq \alpha \leq 1.$$

- Visual:



convex set



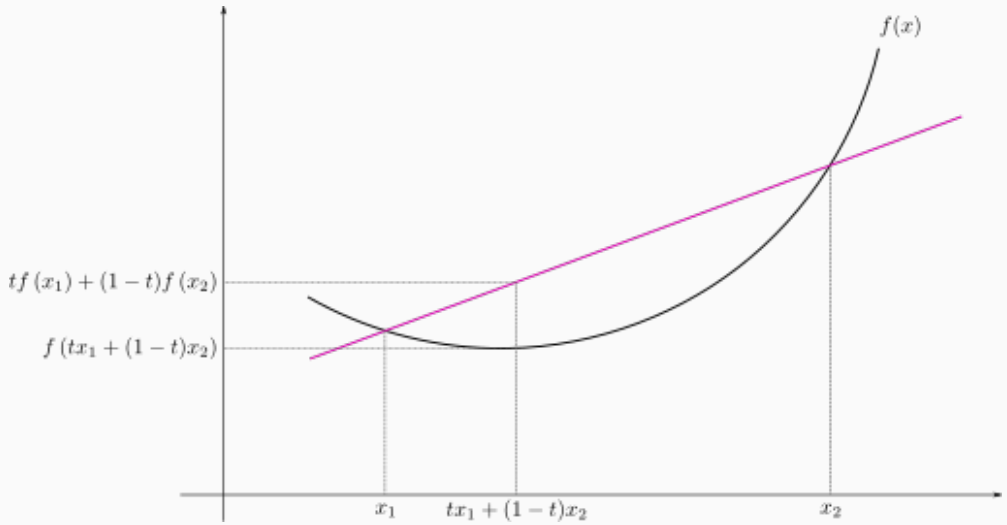
noncovex set

Convex Optimization

- What is a convex function $f : \mathbb{R}^p \to \mathbb{R}$?

$$f\big(\alpha \mathbf{x}_1 + (1-\alpha)\mathbf{x}_2\big) \leq \alpha f(\mathbf{x}_1) + (1-\alpha) f(\mathbf{x}_2) \quad \forall\ 0 \leq \alpha \leq 1.$$

- Visual:



  - Famous result: Jensen's inequality

Convex functions

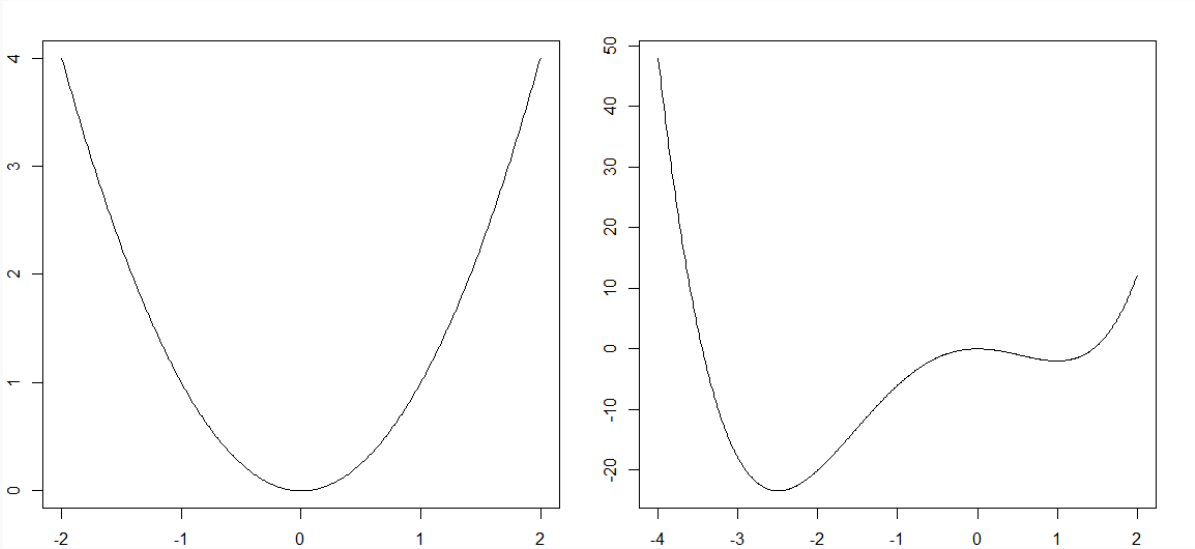
- Examples of convex functions:
  - $\exp(x)$, $-\log(x)$, etc.
  - Affine: $a^{\mathsf{T}}\mathbf{x} + b$ is both convex and concave
  - Quadratic: $\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x} + b^{\mathsf{T}}\mathbf{x} + c$, if $\mathbf{A}$ is positive semidefinite.
  - All norms: ℓ_p
- A function is strictly convex if we can remove the equal sign:

$$f\big(\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2\big) < \alpha f(\mathbf{x}_1) + (1-\alpha) f(\mathbf{x}_2) \quad \forall\ 0 < \alpha < 1.$$

- f is convex $\iff$ $-f$ is concave

Global and Local Minimizers

- In many examples, we want to find a global minimizer of an objective function. This can usually be achieved for convex optimization problems.
- A global minimizer x^* satisfies $f(x^*) \leq f(x)$ for any x in the feasible set.
- As contrast, a local minimizer may exist for non-convex problems, and our algorithms may get "trapped" at a local minimum.



Properties of Convex functions

- First-order property: If f is differentiable with convex domain, then f is convex iff

$$f(\mathbf{x}) \geq f(\mathbf{x}^*) + \bigtriangledown f(\mathbf{x}^*)^\mathsf{T}(\mathbf{x} - \mathbf{x}^*)$$

- A stationary point $\mathbf{x}^*$ is a point where $\bigtriangledown f(\mathbf{x}^*) = \mathbf{0}$
- Combining the two (convex objective function and a stationary point), we have

$$f(\mathbf{x}) \geq f(\mathbf{x}^*)$$

- This means that $\mathbf{x}^*$ is a global minimizer. It may not be unique, but is as good as any other solution.
- Hence in convex optimization problems, we just need to find a stationary point. Example: solving $\boldsymbol{\beta}$ in a linear regression

Properties of Convex functions

- A stationary point in a noncovex problem is not necessarily a minimizer. It could be a maximizer or neither (e.g., a saddle point)
- Hence, we often consider the second-order property. If f is twice differentiable with convex domain, then f is convex iff $\forall\ \mathbf{x}$

$$\bigtriangledown^2 f(\mathbf{x}) = \left(\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} \right) = \mathbf{H}(\mathbf{x}) \succeq 0$$

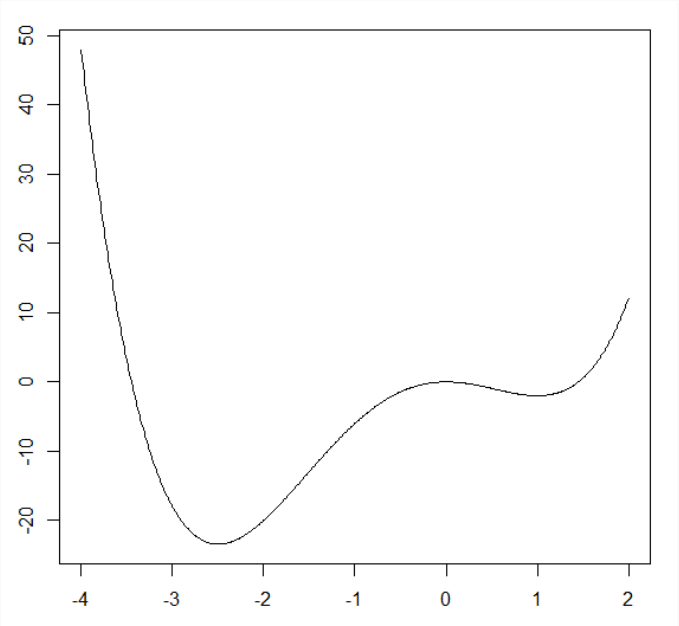
- Even when f is not convex, we may still use this property at a local point $\mathbf{x}^*$, instead of globally for all points, to show that $\mathbf{x}^*$ is a local minimizer.

Example

- Consider a function $f(x) = x^4 + 2x^3 - 5x^2$
- To obtain stationary points,

$$\bigtriangledown f(x) = 4x^3 + 6x^2 - 10x = x(2x + 5)(2x - 2) \doteq 0.$$

Then, there are three stationary points: -2.5, 0, and 1.

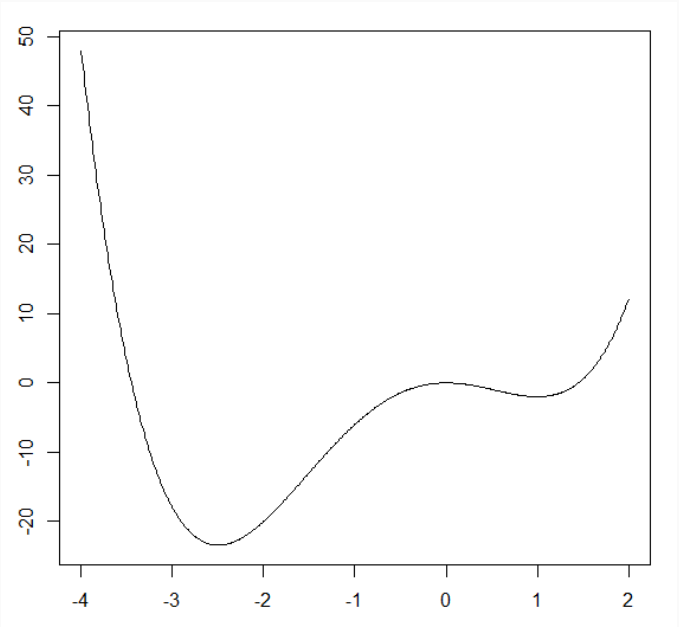


Example

- We can easily check the second order property:

$$\bigtriangledown^2 f(x) = 12x^2 + 12x - 10$$

- $f''(-2.5) = 35$, $f''(0) = -10$, and $f''(1) = 14$.
- Hence, -2.5 and 1 are both local minimizers.



Solving Optimization problems

- In most situations, we do not have a simple objective function, and it is impossible to give an analytic solution.

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad f(\boldsymbol{\beta})$$

- Hence, we often use a descent algorithm: start with a candidate point $\mathbf{x}^{(0)}$, then recursively finding $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, ..., such that

$$f(\mathbf{x}^{(0)}) > f(\mathbf{x}^{(1)}) > f(\mathbf{x}^{(2)}) > \dots$$

- Such methods usually involve finding a direction to move x such that $f(x)$ can decrease.
- We shall discuss two major types: gradient methods and coordinate methods

Solving Optimization problems

- Gradient methods: use quadratic approximation to solve for point such that $\bigtriangledown f(\mathbf{x}) = 0$.
  - Newton's method: use both $\bigtriangledown f(\mathbf{x})$ and $\mathbf{H}(\mathbf{x})$
  - quasi-Newton: use $\bigtriangledown f(\mathbf{x})$ and approximate $\mathbf{H}(\mathbf{x})$
  - Gradient descent: use $\bigtriangledown f(\mathbf{x})$ only
- Coordinate descent: improve the objective function one dimension at a time.
  - Example: solving the Lasso solution

Gradient Methods

Local Quadratic Approximation

- Suppose at a point $\mathbf{x}$, we want to move to a better point $\mathbf{x}^*$.
- Consider the Taylor expansion near $\mathbf{x}$:

$$\begin{aligned} & f(\mathbf{x}^*) \\ \approx & f(\mathbf{x}) + \bigtriangledown f(\mathbf{x})^\mathsf{T}(\mathbf{x}^* - \mathbf{x}) + \frac{1}{2}(\mathbf{x}^* - \mathbf{x})^\mathsf{T}\mathbf{H}(\mathbf{x})(\mathbf{x}^* - \mathbf{x}) \end{aligned}$$

- Our goal is to find $\mathbf{x}^*$ such that $\bigtriangledown f(\mathbf{x}^*) = 0$. By taking derivative with respect to $\mathbf{x}^*$, we have

$$0 = \bigtriangledown f(\mathbf{x}^*) = 0 + \bigtriangledown f(\mathbf{x}) + (\mathbf{x}^* - \mathbf{x})^\mathsf{T}\mathbf{H}(\mathbf{x}).$$

Hence, we should move to the new point

$$\mathbf{x}^* = \mathbf{x} - \mathbf{H}(\mathbf{x})^{-1} \bigtriangledown f(\mathbf{x}).$$

Newton-Raphson

- When we have explicit formula of the Hessian, we can simply compute them the current point $\mathbf{x}^{(k)}$ and follow the updating scheme

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{H}(\mathbf{x}^{(k)})^{-1} \bigtriangledown f(\mathbf{x}^{(k)})$$

  or, sometimes for numerical stability,

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \delta\, \mathbf{H}(\mathbf{x}^{(k)})^{-1} \bigtriangledown f(\mathbf{x}^{(k)})$$

- When the objective function is a quadratic function (e.g., linear regression, ridge regression), we only need to perform this once to reach the optimizer.
- In general, when $\mathbf{x}^{(k+1)}$ is not too far away from $\mathbf{x}^{(k)}$, the quadratic approximation is fairly accurate.

Numerical Approximation

- Sometimes computing $\mathbf{H}$ or even the gradient $\bigtriangledown f(\mathbf{x})$ is difficult.
- If the dimension of the problem is not very large, we may approximate them numerically (element-wise).
- By definition, a finite-difference approximation of derivatives is

$$\frac{\partial f}{\partial x_j} \approx \frac{f(\mathbf{x} + \delta \mathbf{e}_j) - f(\mathbf{x})}{\delta},$$

- Second order derivatives can be done by

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \approx \frac{f(\mathbf{x} + \delta_i \mathbf{e}_i + \delta_j \mathbf{e}_j) - f(\mathbf{x} + \delta_i \mathbf{e}_i) - f(\mathbf{x} + \delta_j \mathbf{e}_j) + f(\mathbf{x})}{\delta_i \delta_j}.$$

- Here, $\mathbf{e}_j$ is a vector with the jth element 1 and 0 everywhere else.

Quasi-Newton Methods

- However, such approximations (especially for $\mathbf{H}$) can be extremely slow!
- In addition, when the dimension of the problem is large, $\mathbf{H}$ is a huge matrix.
  - Numerically approximating $\mathbf{H}$ takes $(p + p^2)/2$ function calls
  - Inverting $\mathbf{H}$ is at least $\mathcal{O}\left(p^2 \log(p)\right)$ time complex
- Quasi-Newton methods avoid such calculations using rank-one updates of $\mathbf{H}^{(-1)}$ based on the Sherman–Morrison–Woodbury formula

Quasi-Newton Methods

- Intuition of quasi-Newton method
  - Start with $\mathbf{H}(\mathbf{x})^{-1} = \mathbf{I}$ at $\mathbf{x}^{(0)}$
  - We shall update $\mathbf{x}^{(0)} \to \mathbf{x}^{(1)} \to \mathbf{x}^{(2)} \to \cdots$
  - Along this path, we need to compute $\bigtriangledown f(\mathbf{x}^{(0)}) \to \bigtriangledown f(\mathbf{x}^{(1)}) \to \cdots$
  - If we treat the function $f(\mathbf{x}^{(k)})$ locally as a quadratic function,

$$\bigtriangledown f(\mathbf{x}^{(k+1)}) - \bigtriangledown f(\mathbf{x}^{(k)}) \approx \mathbf{H}(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})$$

  - This poses an additional information (rank-one condition) on $\mathbf{H}$, and can be used to update $\mathbf{H}^{(-1)}$
- BFGS is a famous example. Implemented by the `optim()` function in R

Implementation

- If you have a smooth objective function, usually the log-likelihood $L(\mathbf{y}, \mathbf{X}, \boldsymbol{\beta})$, and we want to solve the parameters $\boldsymbol{\beta}$.
- You can utilize the `optim()` function in R

```
> L <- function(b, X, Y) ...
> bhat = optim(rep(0, P), L, X = X, Y = Y, method = "BFGS")
```

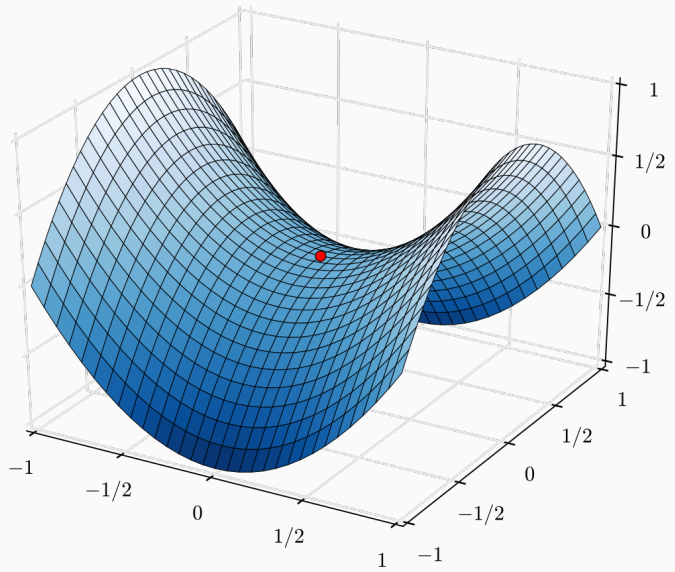
- Sometimes, using a different initial value may be better.

BFGS

Broyden, Fletcher, Goldfarb, Shanno

Saddle Point

- At a local minimizer, the Hessian matrix $\mathbf{H}$ will be positive semi-definite
- At a local maximizer, the Hessian matrix $\mathbf{H}$ will be negative semi-definite
- If a Hessian obtains both positive and negative eigen-values, then we are at a saddle point, which is a more difficult issue.



Gradient descent

- In some other cases, for computational convince, we may just use an identity matrix $\frac{1}{\delta}\mathbf{I}$ as $\mathbf{H}$.
- This is the gradient descent and update is

$$\mathbf{x}^* = \mathbf{x} - \delta \bigtriangledown f(\mathbf{x})$$

- However, we have to choose δ, the step size:
  - A step size too large may not even converge at all.
  - How about we just fix δ to be a small value, say 10^{-5}.
  - A step size too small will take many iterations to converge.
  - Sometimes we may use line search

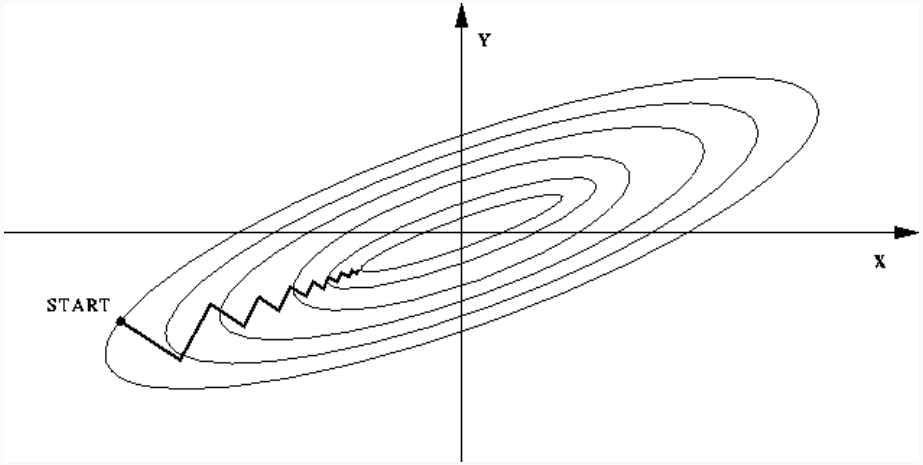
Gradient descent

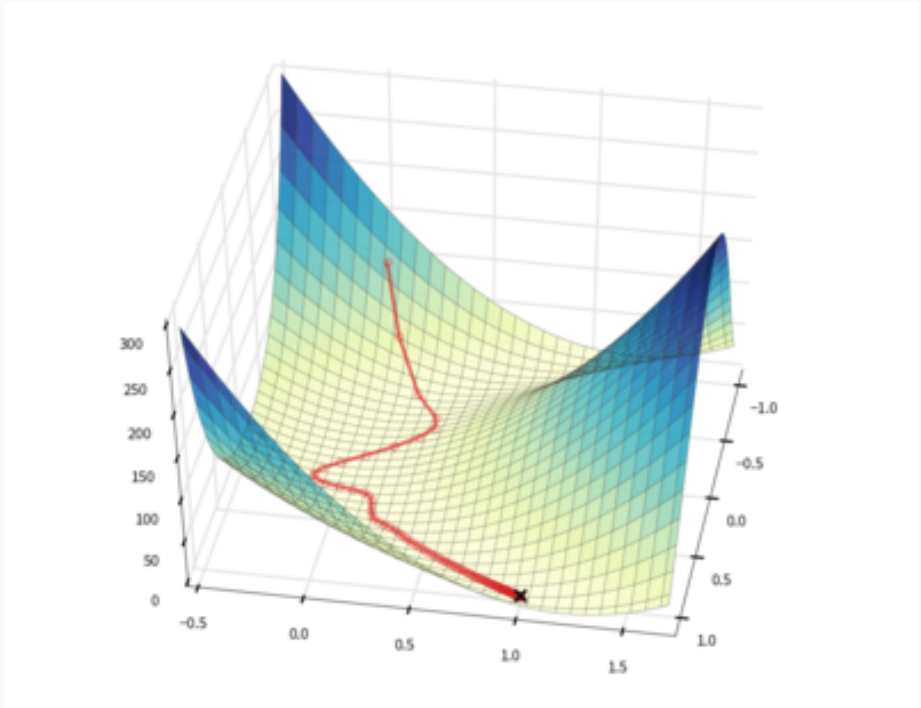
- Line search is a common approach for choosing the step size δ
- Once we have $\bigtriangledown f(\mathbf{x})$, search for δ by

$$\delta = \arg\min_{\delta} f(\mathbf{x}^{(k)} - \delta \bigtriangledown f(\mathbf{x}^{(k)}))$$

  - This is a 1-dimensional problem once we know $\bigtriangledown f(\mathbf{x})$
  - It guarantees the descent property, hence, less risky than fixed step size
  - It requires additional calls of the function evaluation, hence could be slow if functional calls are expensive. In that case, a constant step size may be more beneficial.

Gradient descent





Coordinate Descent

Coordinate Descent

- Coordinate descent means updating one coordinate at a time.
- The Gauss-Seidel style coordinate descent algorithm at the kth (grand) iteration:

$$\begin{aligned}
\beta_1^{(k+1)} &= \arg\min_{\beta_1} \quad f(\beta_1, \beta_2^{(k)}, \ldots, \beta_p^{(k)}) \\
\beta_2^{(k+1)} &= \arg\min_{\beta_2} \quad f(\beta_1^{(k+1)}, \beta_2, \ldots, \beta_p^{(k)}) \\
&\ldots \\
\beta_p^{(k+1)} &= \arg\min_{\beta_p} \quad f(\beta_1^{(k+1)}, \beta_2^{(k+1)}, \ldots, \beta_p)
\end{aligned}$$

- After we complete this loop, all β_j are updated to their new values, and we proceed to the next step.

Coordinate Descent

- The Jacobi style algorithm (can be parallelized) at the kth (grand) iteration:

$$\begin{aligned}
\beta_1^{(k+1)} &= \arg\min_{\beta_1} \quad f(\beta_1, \beta_2^{(k)}, \ldots, \beta_p^{(k)}) \\
\beta_2^{(k+1)} &= \arg\min_{\beta_2} \quad f(\beta_1^{(k)}, \beta_2, \ldots, \beta_p^{(k)}) \\
&\ldots \\
\beta_p^{(k+1)} &= \arg\min_{\beta_p} \quad f(\beta_1^{(k)}, \beta_2^{(k)}, \ldots, \beta_p)
\end{aligned}$$

- After we complete this loop, update all β_j to their new values, and start over.
- Jacobi style algorithm can be computed in a parallel fashion, while Gauss-Seidel style (more popular) can only be done sequentially.

Coordinate Descent

- Why is coordinate descent preferred over gradient descent under some cases?
  - The objective function may not be differentiable (this is very common)
  - Analytic solution may exist for one-dimensional problems
- Example: Lasso is decomposable

$$f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$$

with differentiable g and non-differentiable (but convex) h.

$$\frac{1}{2n}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^{\mathsf{T}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{i=1}^{p} |\beta_i|$$

Note: we switch back to the "$\boldsymbol{\beta}$" notation for parameters.

Coordinate Descent for Lasso

- For coordinate descent, fix all parameters expect the jth entry.
- Let $\boldsymbol{\beta}_{(-j)}$ denote the vector by removing the jth entry, and $\mathbf{X}_{(-j)}$ be the corresponding design matrix by removing the jth column from $\mathbf{X}$.
- The coordinate descent problem at the kth iteration is to find

$$\begin{aligned} &\arg\min_{\beta_j} \frac{1}{2n}\|\mathbf{y} - X_j\beta_j - \mathbf{X}_{(-j)}\boldsymbol{\beta}_{(-j)}^{(k)}\|_2^2 + \lambda\sum_{i=1}^{p}|\beta_i| \\ =&\arg\min_{\beta_j} \frac{1}{2n}\|\mathbf{y} - X_j\beta_j - \mathbf{X}_{(-j)}\boldsymbol{\beta}_{(-j)}^{(k)}\|_2^2 + \lambda|\beta_j| \end{aligned}$$

- This is a one-dimensional Lasso problem we have analyzed before!

Coordinate Descent for Lasso

- First, we know the optimizer to the differentiable part

$$\frac{1}{n}\|\mathbf{y} - X_j\beta_j - \mathbf{X}_{(-j)}\boldsymbol{\beta}_{(-j)}^{(k)}\|_2^2$$

- Lets define the vector of temporary "outcome"

$$\mathbf{r} = \mathbf{y} - \mathbf{X}_{(-j)}\boldsymbol{\beta}_{(-j)}^{(k)}$$

  which is also the residual after removing the effect of all variables except j.

- We need to optimize

$$\frac{1}{n}\|\mathbf{r} - X_j\beta_j\|_2^2 + \lambda|\beta_j|$$

Coordinate Descent for Lasso

- We know the OLS solution to the squared error loss part is

$$\beta_j^{\text{OLS}} = \frac{\mathbf{X}_j^\top \mathbf{r}}{\mathbf{X}_j^\top \mathbf{X}_j}$$

- And adding the penalty is simply performing a soft-thresholding to the OLS solution (see the penalized regression lecture note page 34 and 35)
- However, be careful that the scale $\mathbf{X}_j^\top \mathbf{X}_j$ will play a role here in the shrinkage (if we do not assume $\mathbf{X}_j^\top \mathbf{X}_j = 1$ or n)
- An easier solution is to pre-scale all covaraites: $\mathbf{X}_j^\top \mathbf{X}_j = n$

Coordinate Descent for Lasso

- Further improving the efficiency in the coordinate descent algorithm for Lasso
- Calculating the residual $\mathbf{r} = \mathbf{y} - \mathbf{X}_{(-j)}\boldsymbol{\beta}^{(k)}_{(-j)}$ can be very costly since it involves multiplying using a $n \times (p-1)$ matrix
- Instead, since we only update one β_j at a time, the residual $\mathbf{r}$ at the next iteration can be obtained with

$$\mathbf{r}^{\text{new}} = \underbrace{\mathbf{r} - \mathbf{X}_{(j)}\boldsymbol{\beta}^{(k+1)}_{(j)}}_{\text{residual of current model}} + \underbrace{\mathbf{X}_{(j+1)}\boldsymbol{\beta}^{(k)}_{(j+1)}}_{\text{add } j+1 \text{ back}}$$

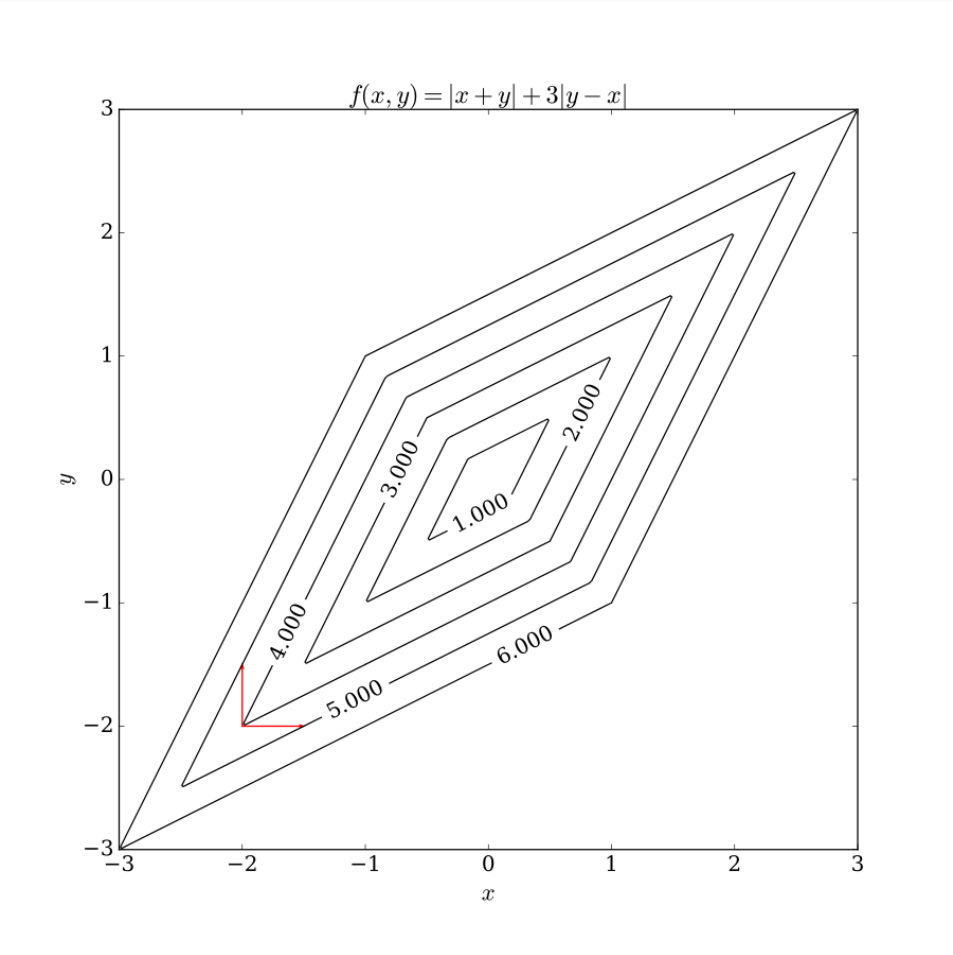
Complexity

- Which is faster? Coordinate descent or gradient descent? Let's compare them using a linear regression model:
- Gradient descent
  - $\mathbf{X}\boldsymbol{\beta}$ is $(n \times p) \times (p \times 1)$, i.e. $\mathcal{O}(np)$
  - The gradient $-\mathbf{X}^{\mathsf{T}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$ also cost $\mathcal{O}(np)$
  - Updating $\boldsymbol{\beta} = \boldsymbol{\beta} + \delta\, \mathbf{X}^{\mathsf{T}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$ costs very little
  - Hence overall $\mathcal{O}(np)$ flops
- Coordinate descent
  - Calculating $\mathbf{r}$ with our efficient algorithm coats $2n$ flops
  - $X_j^{\mathsf{T}}\mathbf{r}$ cost n flops
  - $X_j^{\mathsf{T}}X_j$ cost none if we pre-scale all $X_j^{\mathsf{T}}X_j = n$
  - Loop overall p variables will multiple the above by p. Hence the overall cost is $\mathcal{O}(np)$ flops.

Complexity

- However, using coordinate descent for some particular problems can be very beneficial (e.g. Lasso) since each parameter updates to its minimizer fully, while gradient descent only move toward the correct direction with a small step.
- When is coordinate descent useful/better?
  - If updating each coordinate is cheap, and maybe the solution is explicit (our lasso problem is an example).
  - When the one-dimensional problem does not have simple analytic solution, we may still update the one-dimensional problem in a gradient descent fashion: $\beta_j^{k+1} = \beta_j^{(k)} - \delta \frac{\partial f(\beta^{(k)})}{\partial \beta_j^k}$
- Will coordinate descent fail?

Coordinate descent may fail



Coordinate descent may fail

Stochastic Gradient Descent

Stochastic Gradient Descent

- SGD: stochastic approximation of gradient descent optimization
- When the objective function has a form of sum (e.g. sum of squared errors)

$$L(\beta) = \frac{1}{n} \sum_{i=1}^{n} l(x_i, y_i, \beta)$$

- In gradient descent we calculate the gradient of this function by taking derivative w.r.t β
- However, this requires $\mathcal{O}(n)$ complexity, since we use all n subjects. This can be slow for large datasets.

Stochastic gradient descent

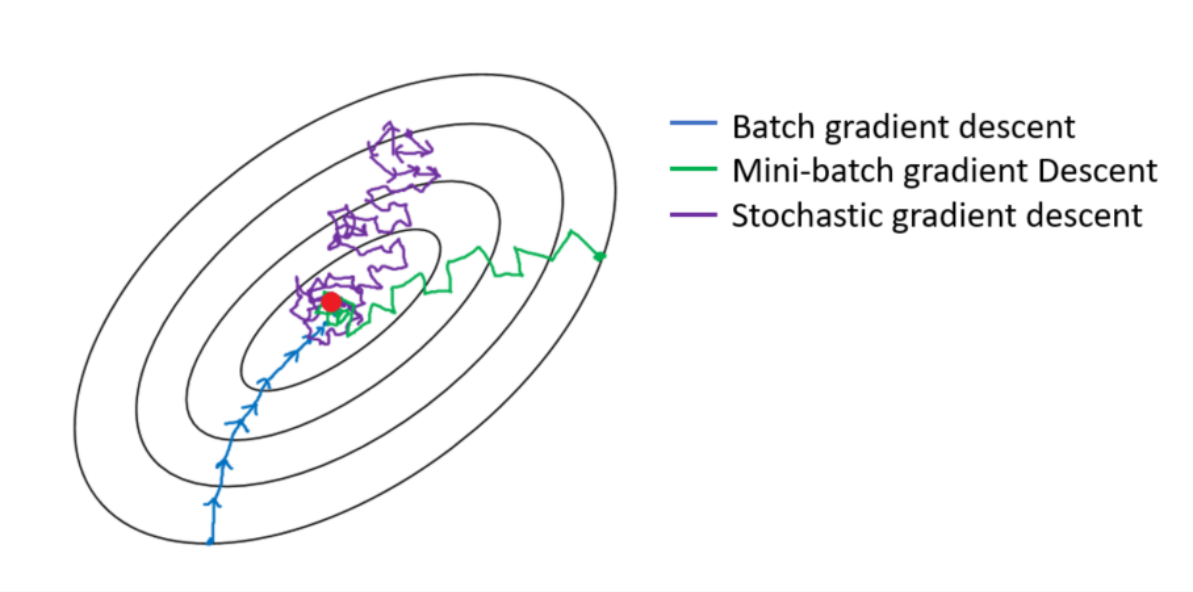
- How about we just calculate the gradient w.r.t the loss of one subject? In this case, we update β using

$$\beta^{\text{new}} = \beta^{\text{old}} - \delta \nabla l(x_i, y_i, \beta)$$

  and iterate with $i = 1, 2, \ldots, n$.

- This turns out to be very fast and eventually coverage to the same target
- Note: tuning the step size is δ extremely crucial and trickly in this case.
- A variant of this algorithm is the "mini-batch" SGD, which creates small groups of subjects, and calculate the gradient based on each group, and then loop over all groups

Stochastic gradient descent



by Midhilesh elavazhagan

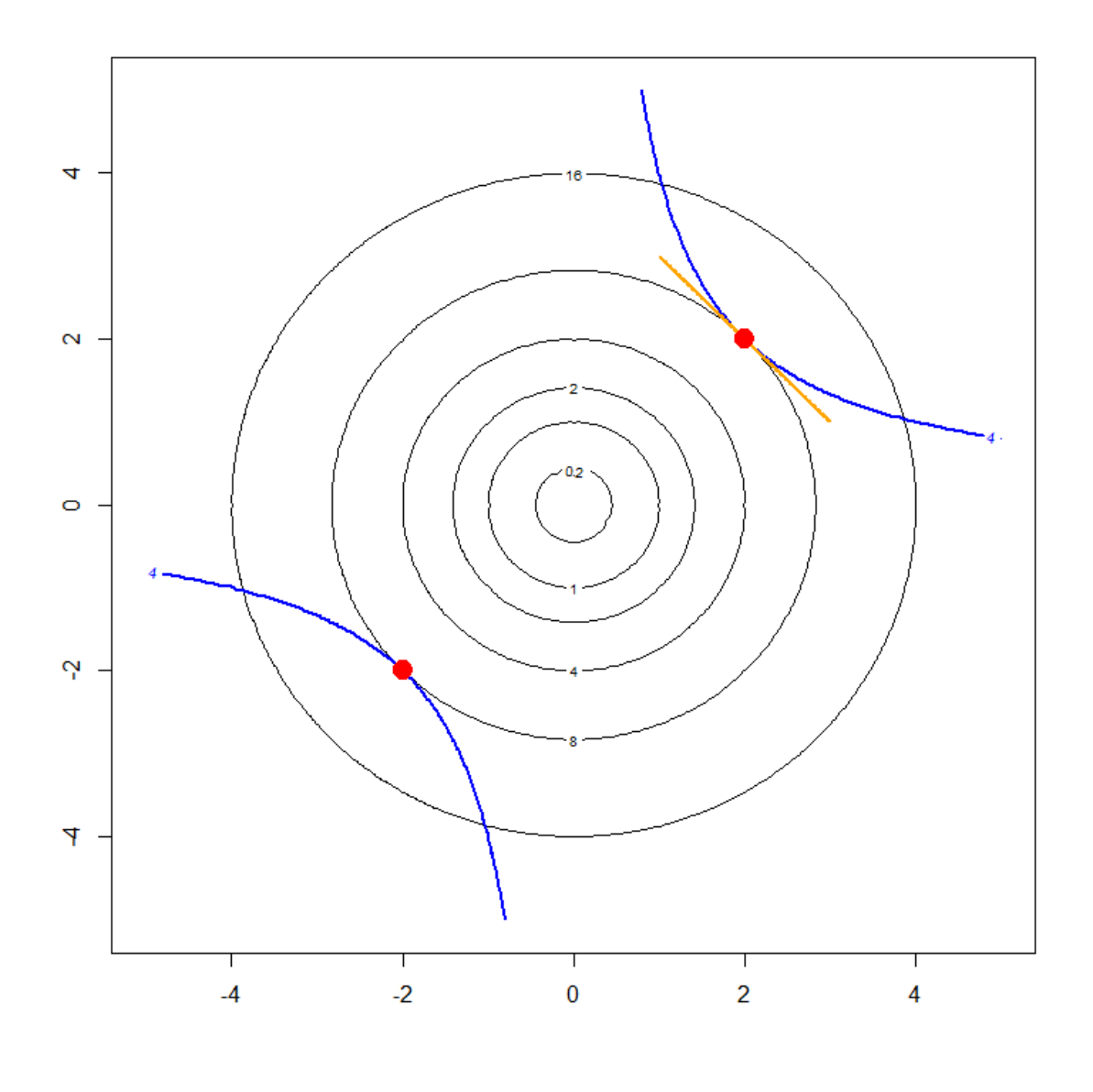
Lagrangian Multiplier

Constrained Optimizations

- For the most part in this lecture, we have not discussed constrained optimization.
- However, we have already see and will encounter more such problems
  - Lasso and Ridge can both be seen as setting the constrain in the form of $\|\boldsymbol{\beta}\|_q \leq s$
  - Support vector machine will be introduced later
  - Toy example:

$$\begin{aligned} \text{minimize} \quad & f(x, y) = x^2 + y^2 \\ \text{subj. to} \quad & g(x, y) = xy - 4 = 0 \end{aligned}$$

Constrained Optimizations



Constrained Optimizations

- As the level curve grows, it touches the constrain curve at certain point.
- This implies that the tangent line of the level curve must coincide with the tangent line of the constraint
- Hence their gradients will be a multiple of each other:

$$\bigtriangledown f = \lambda \bigtriangledown g$$

$$\Longrightarrow \begin{cases} 2x = \lambda y & \text{by taking derivative w.r.t. } x \\ 2y = \lambda x & \text{by taking derivative w.r.t. } y \\ xy - 4 = 0 & \text{the constraint itself} \end{cases}$$

Constrained Optimizations

- The first two equations leads to $x = y = 0$ or $\lambda = \pm 2$, while $x = y = 0$ is not feasible.
- $\lambda = \pm 2$ leads to $x = y = 2$ or $x = y = -2$, both are feasible.
- Noticing that the equation $\bigtriangledown f = \lambda \bigtriangledown g$ is simply the derivative of the Lagrangian function

$$\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda g(x, y)$$

the constrained problem is essentially solving for stationary point (with respect to x, y, and λ) of the Lagrangian.