ECS289: Scalable Machine Learning

Cho-Jui Hsieh
UC Davis

Oct 1, 2015

Outline

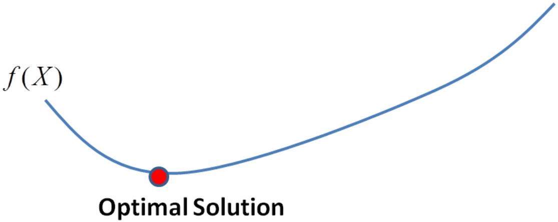
- Convex vs Nonconvex Functions
- Coordinate Descent
- Gradient Descent
- Newton's method
- Stochastic Gradient Descent

Numerical Optimization

- Numerical Optimization:

$$\min_X f(X)$$

- Can be applied to computer science, economics, control engineering, operating research, ...
- **Machine Learning: find a model that minimizes the prediction error.**



Properties of the Function

- Smooth function: a function has continuous derivative.

  Example: ridge regression

$$\operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{2}\|X\boldsymbol{w} - \mathbf{y}\|^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

- Non-smooth function: Lasso, primal SVM

$$\text{Lasso:}\quad \operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{2}\|X\boldsymbol{w} - \mathbf{y}\|^2 + \lambda\|\boldsymbol{w}\|_1$$

$$\text{SVM:}\quad \operatorname*{argmin}_{\boldsymbol{w}} \sum_{i=1}^{n} \max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

Convex Functions

- A function is convex if:

$$\forall \boldsymbol{x}_1, \boldsymbol{x}_2, \forall t \in [0,1], f(t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_2) \leq tf(\boldsymbol{x}_1) + (1-t)f(\boldsymbol{x}_2)$$

- No local optimum (why?)

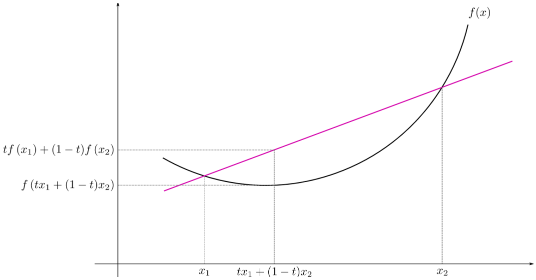


Figure from Wikipedia

Convex Functions

- If $f(\boldsymbol{x})$ is twice differentiable, then

  f is convex if and only if $\nabla^2 f(\boldsymbol{x}) \succeq 0$

- Optimal solution may not be unique:

  has a set of optimal solutions $\mathcal{S}$

- Gradient: capture the first order change of f:

$$f(\boldsymbol{x} + \alpha \boldsymbol{d}) = f(\boldsymbol{x}) + \alpha \nabla f(\boldsymbol{x})^T \boldsymbol{d} + O(\alpha^2)$$

- If f is differentiable, we have the following optimality condition:

$$\boldsymbol{x}^* \in \mathcal{S} \text{ if and only if } \nabla f(\boldsymbol{x}) = 0$$

Strongly Convex Functions

- f is strongly convex if there exists a $m > 0$ such that

$$f(\mathbf{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\mathbf{y} - \boldsymbol{x}) + \frac{m}{2}\|\mathbf{y} - \boldsymbol{x}\|_2^2$$

- A strongly convex function has a unique global optimum $\boldsymbol{x}^*$ (why?)
- If f is twice differentiable, then

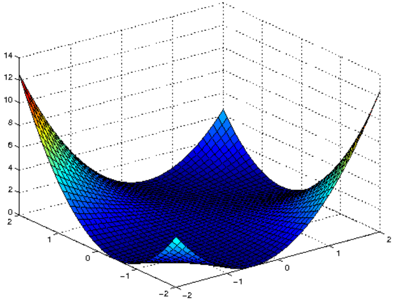
$$f \text{ is strongly convex if and only if } \nabla^2 f(\boldsymbol{x}) \succ mI > 0 \text{ for all } \boldsymbol{x}$$

- Gradient descent, coordinate descent will converge linearly (will see later)

Nonconvex Functions

- If f is nonconvex, most algorithms can only converge to **stationary points**
- $\bar{\boldsymbol{x}}$ is a stationary point if and only if $\nabla f(\bar{\boldsymbol{x}}) = 0$
- Three types of stationary points:

  (1) Global optimum (2) Local optimum (3) Saddle point
- Example: matrix completion, neural network, ...
- Example: $f(x, y) = \frac{1}{2}(xy - a)^2$



Coordinate Descent

Coordinate Descent

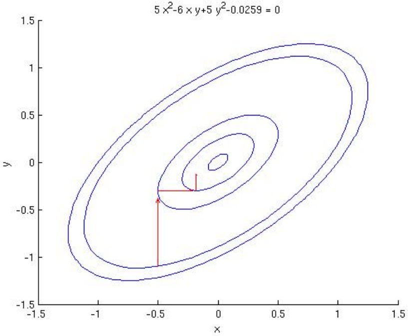
- Update one variable at a time
- Coordinate Descent: repeatedly perform the following loop

  Step 1: pick an index i

  Step 2: compute a step size δ^* by (approximately) minimizing

$$\operatorname*{argmin}_{\delta} f(\boldsymbol{x} + \delta \boldsymbol{e}_i)$$

  Step 3: $x_i \leftarrow x_i + \delta^*$



Coordinate Descent (update sequence)

- Three types of updating order:
- Cyclic: update sequence

$$\underbrace{x_1, x_2, \ldots, x_n}_{\text{1st outer iteration}}, \underbrace{x_1, x_2, \ldots, x_n}_{\text{2nd outer iteration}}, \ldots$$

  - A more general setting: update each variable at least once within every T steps
  - Randomly permute the sequence for each outer iteration (faster convergence in practice)

Coordinate Descent (update sequence)

- Three types of updating order:
- Cyclic: update sequence

$$\underbrace{x_1, x_2, \ldots, x_n}_{\text{1st outer iteration}}, \underbrace{x_1, x_2, \ldots, x_n}_{\text{2nd outer iteration}}, \ldots$$

  - A more general setting: update each variable at least once within every T steps
  - Randomly permute the sequence for each outer iteration (faster convergence in practice)
- Random: each time pick a random coordinate to update
  - Typical way: sample from uniform distribution
  - Sample from uniform distribution vs sample from biased distribution

    P. Zhao and T. Zhang, Stochastic Optimization with Importance Sampling for Regularized Loss Minimization. In ICML 2015

    D. Csiba, Z. Qu and P. Richtarik, Stochastic Dual Coordinate Ascent with Adaptive Probabilities. In ICML 2015

Greedy Coordinate Descent

- Greedy: choose the most "important" coordinate to update
- How to measure the importance?
  - By first derivative: $|\nabla_i f(\boldsymbol{x})|$
  - By first and second derivative: $|\nabla_i f(\boldsymbol{x}) / \nabla_{ii}^2 f(\boldsymbol{x})|$
  - By maximum reduction of objective function

$$i^* = \operatorname*{argmax}_{i=1,\ldots,n} \left(f(\boldsymbol{x}) - \min_{\delta} f(\boldsymbol{x} + \delta \boldsymbol{e}_i) \right)$$

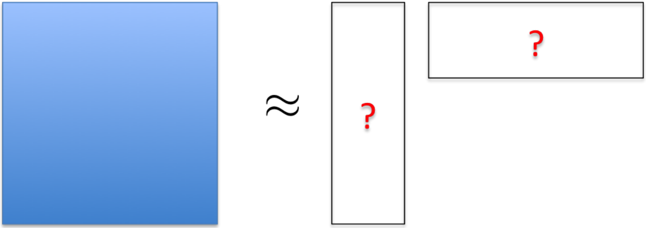
- Need to consider the time complexity for variable selection
- Useful for kernel SVM (see lecture 6)

Extension: block coordinate descent

- Variables are divided into blocks $\{\mathcal{X}_1, \ldots, \mathcal{X}_p\}$, where each $\mathcal{X}_i$ is a subset of variables and

$$\mathcal{X}_1 \cup \mathcal{X}_2, \ldots, \mathcal{X}_p = \{1, \ldots, n\}, \quad \mathcal{X}_i \cap \mathcal{X}_j = \varphi, \quad \forall i, j$$

- Each time update a $\mathcal{X}_i$ by (approximately) solving the subproblem within the block
- Example: alternating minimization for matrix completion (2 blocks). (See lecture 7)

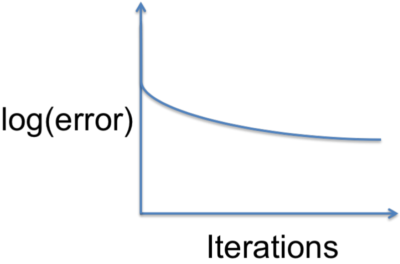


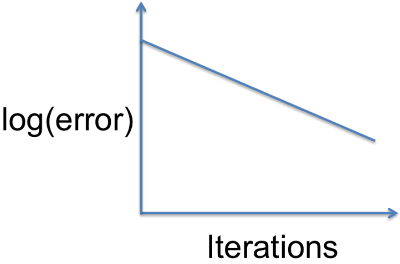
Coordinate Descent (convergence)

- Converge to an optimum if $f(\cdot)$ is convex and smooth
- Has a linear convergence rate if $f(\cdot)$ is strongly convex
- Linear convergence: error $f(\boldsymbol{x}^t) - f(\boldsymbol{x}^*)$ decays as

$$\beta, \beta^2, \beta^3, \ldots$$

  for some $\beta < 1$.
- Local linear convergence: an algorithm converges linearly after $\|\boldsymbol{x} - \boldsymbol{x}^*\| \leq K$ for some $K > 0$





Coordinate Descent (nonconvex)

- Block coordinate descent with 2 blocks:

  converges to stationary points

- With > 2 blocks:

  converges to stationary points if each subproblem has a unique minimizer.

Coordinate Descent: other names

- Alternating minimization (matrix completion)
- Iterative scaling (for log-linear models)
- Decomposition method (for kernel SVM)
- Gauss Seidel (for linear system when the matrix is positive definite)
- ...

Gradient Descent

Gradient Descent

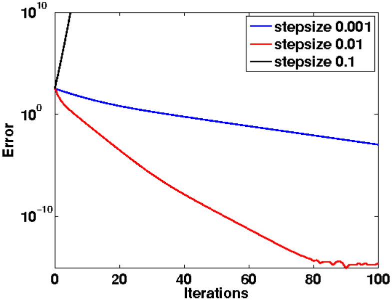
- Gradient descent algorithm: repeatedly conduct the following update:

$$\boldsymbol{x}^{t+1} \leftarrow \boldsymbol{x}^{t} - \alpha \nabla f(\boldsymbol{x}^{t})$$

  where $\alpha > 0$ is the step size
- It is a fixed point iteration method:

$$\boldsymbol{x} - \alpha \nabla f(\boldsymbol{x}) = \boldsymbol{x} \text{ if and only if } \boldsymbol{x} \text{ is an optimal solution}$$

- Step size too large $\Rightarrow$ diverge; too small $\Rightarrow$ slow convergence



Gradient Descent: successive approximation

- At each iteration, form an approximation of $f(\cdot)$:

$$\begin{aligned} f(\boldsymbol{x}^t + \boldsymbol{d}) \approx \tilde{f}_{\boldsymbol{x}^t}(\boldsymbol{d}) :=& f(\boldsymbol{x}^t) + \nabla f(\boldsymbol{x}^t)^T \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T (\frac{1}{\alpha} I)\boldsymbol{d} \\ =& f(\boldsymbol{x}^t) + \nabla f(\boldsymbol{x}^t)^T \boldsymbol{d} + \frac{1}{2\alpha}\boldsymbol{d}^T \boldsymbol{d} \end{aligned}$$

- Update solution by $\boldsymbol{x}^{t+1} \leftarrow \boldsymbol{x}^t + \text{argmin}_{\boldsymbol{d}}\, \tilde{f}_{\boldsymbol{x}^t}(\boldsymbol{d})$
- $\boldsymbol{d}^* = -\alpha \nabla f(\boldsymbol{x}^t)$ is the minimizer of $\text{argmin}_{\boldsymbol{d}}\, \tilde{f}_{\boldsymbol{x}^t}(\boldsymbol{d})$
- $\boldsymbol{d}^*$ may not decrease the original objective function f

Gradient Descent: successive approximation

- However, the function value will decrease if

  Condition 1: $\tilde{f}_{\boldsymbol{x}}(\boldsymbol{d}) \geq f(\boldsymbol{x}+\boldsymbol{d})$ for all $\boldsymbol{d}$

  Condition 2: $\tilde{f}_{\boldsymbol{x}}(\mathbf{0}) = f(\boldsymbol{x})$

- Why?

$$\begin{aligned} f(\boldsymbol{x}^t + \boldsymbol{d}^*) &\leq \tilde{f}_{\boldsymbol{x}^t}(\boldsymbol{d}^*) \\ &\leq \tilde{f}_{\boldsymbol{x}^t}(\mathbf{0}) \\ &= f(\boldsymbol{x}^t) \end{aligned}$$

- Condition 2 is satisfied by construction of $\tilde{f}_{\boldsymbol{x}^t}$
- Condition 1 is satisfied if $\frac{1}{\alpha} I \succeq \nabla^2 f(\boldsymbol{x})$ for all $\boldsymbol{x}$ (why?)

Gradient Descent: step size

- A function has L-Lipchitz continuous gradient if

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\mathbf{y})\| \leq L\|\boldsymbol{x} - \mathbf{y}\| \quad \forall \boldsymbol{x}, \mathbf{y}$$

- If f is twice differentiable, this implies

$$\nabla^2 f(\boldsymbol{x}) \leq LI \quad \forall \boldsymbol{x}$$

- In this case, Condition 2 is staisfied if $\alpha \leq \frac{1}{L}$
- **Theorem:** gradient descent converges if $\alpha \leq \frac{1}{L}$
- **Theorem:** gradient descent converges linearly with $\alpha \leq \frac{1}{L}$ if f is strongly convex

Gradient Descent

- In practice, we do not know L......
- Step size α too large: the algorithm diverges
- Step size α too small: the algorithm converges very slowly

Gradient Descent: line search

- $\boldsymbol{d}^*$ is a "descent direction" if and only if $(\boldsymbol{d}^*)^T \nabla f(\boldsymbol{x}) < 0$
- Armijo rule bakctracking line search:
  Try $\alpha = 1, \frac{1}{2}, \frac{1}{4}, \ldots$ until it staisfies

$$f(\boldsymbol{x} + \alpha \boldsymbol{d}^*) \leq f(\boldsymbol{x}) + \gamma \alpha (\boldsymbol{d}^*)^T \nabla f(\boldsymbol{x})$$

  where $0 < \gamma < 1$

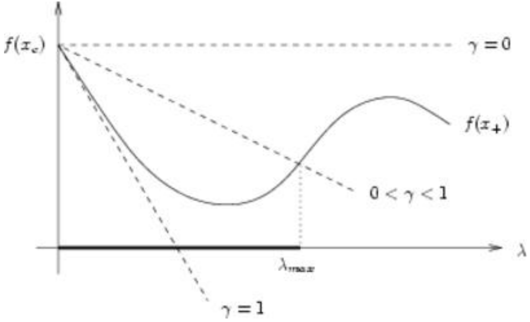


Figure from http://ool.sourceforge.net/ool-ref.html

Gradient Descent: line search

- Gradient descent with line search:
  - Converges to optimal solutions if f is smooth
  - Converges linearly if f is strongly convex
- However, each iteration requires evaluating f several times
- Several other step-size selection approaches

  (an ongoing research topic, especially for stochastic gradient descent)

Gradient Descent: applying to ridge regression

Input: $X \in \mathbb{R}^{N \times d}$, $\mathbf{y} \in \mathbb{R}^N$, initial $\boldsymbol{w}^{(0)}$

Output: Solution $\boldsymbol{w}^* := \text{argmin}_{\boldsymbol{w}} \frac{1}{2}\|X\boldsymbol{w} - \mathbf{y}\|^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$

$t = 0$

while not converged **do**

  Compute the gradient

$$\boldsymbol{g} = X^T(X\boldsymbol{w} - \mathbf{y}) + \lambda \boldsymbol{w}$$

  Choose step size α^t

  Update $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha^t \boldsymbol{g}$

  $t \leftarrow t + 1$

end while

Time complexity: $O(\text{nnz}(X))$ per iteration

Proximal Gradient Descent

- How can we apply gradient descent to solve the Lasso problem?

$$\operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{2}\|X\boldsymbol{w} - \mathbf{y}\|^2 + \lambda \underbrace{\|\boldsymbol{w}\|_1}_{non-differentiable}$$

- General composite function minimization:

$$\operatorname*{argmin}_{\boldsymbol{x}} f(\boldsymbol{x}) := \{g(\boldsymbol{x}) + h(\boldsymbol{x})\}$$

  where g is smooth and convex, h is convex but may be non-differentiable
- Usually assume h is simple (for computational efficiency)

Proximal Gradient Descent: successive approximation

- At each iteration, form an approximation of $f(\cdot)$:

$$
\begin{aligned}
f(\boldsymbol{x}^t + \boldsymbol{d}) \approx \tilde{f}_{\boldsymbol{x}^t}(\boldsymbol{d}) :=& g(\boldsymbol{x}^t) + \nabla g(\boldsymbol{x}^t)^T \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T(\frac{1}{\alpha} I)\boldsymbol{d} + h(\boldsymbol{x}^t + \boldsymbol{d}) \\
=& g(\boldsymbol{x}^t) + \nabla g(\boldsymbol{x}^t)^T \boldsymbol{d} + \frac{1}{2\alpha}\boldsymbol{d}^T\boldsymbol{d} + h(\boldsymbol{x}^t + \boldsymbol{d})
\end{aligned}
$$

- Update solution by $\boldsymbol{x}^{t+1} \leftarrow \boldsymbol{x}^t + \mathrm{argmin}_{\boldsymbol{d}}\, \tilde{f}_{\boldsymbol{x}^t}(\boldsymbol{d})$
- This is called "proximal" gradient descent
- Sometimes $\boldsymbol{d}^* = \mathrm{argmin}_{\boldsymbol{d}}\, \tilde{f}_{\boldsymbol{x}^t}(\boldsymbol{d})$ has a closed form solution

Proximal Gradient Descent: ℓ_1-regularization (*)

- The subproblem:

$$\begin{aligned}
\boldsymbol{x}^{t+1} =& \boldsymbol{x}^t + \operatorname*{argmin}_{\boldsymbol{d}} \nabla g(\boldsymbol{x}^t)^T \boldsymbol{d} + \frac{1}{2\alpha}\boldsymbol{d}^T\boldsymbol{d} + \lambda\|\boldsymbol{x}^t + \boldsymbol{d}\|_1 \\
=& \operatorname*{argmin}_{\boldsymbol{u}} \frac{1}{2}\|\boldsymbol{u} - (\boldsymbol{x}^t - \alpha\nabla g(\boldsymbol{x}^t))\|^2 + \lambda\alpha\|\boldsymbol{u}\|_1 \\
=& \mathcal{S}(\boldsymbol{x}^t - \alpha\nabla g(\boldsymbol{x}^t), \alpha\lambda),
\end{aligned}$$

where $\mathcal{S}$ is the soft-thresholding operator defined by

$$\mathcal{S}(a, z) = \begin{cases} a - z & \text{if } a > z \\ a + z & \text{if } a < -z \\ 0 & \text{if } a \in [-z, z] \end{cases}$$

Proximal Gradient: soft-thresholding

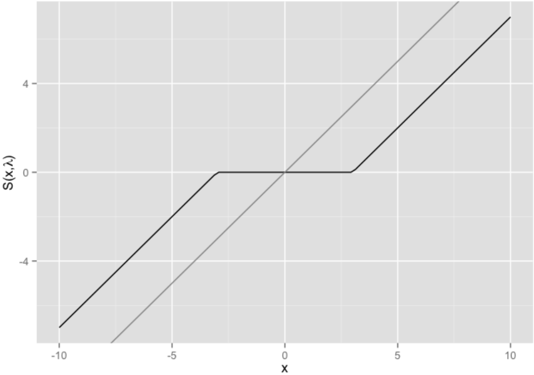


Figure from http://jocelynchi.com/soft-thresholding-operator-and-the-lasso-solution/

Proximal Gradient Descent for Lasso

Input: $X \in \mathbb{R}^{N\times d}$, $\mathbf{y} \in \mathbb{R}^N$, initial $\boldsymbol{w}^{(0)}$
Output: Solution $\boldsymbol{w}^* := \text{argmin}_{\boldsymbol{w}} \frac{1}{2}\|X\boldsymbol{w} - \mathbf{y}\|^2 + \lambda\|\boldsymbol{w}\|_1$

$t = 0$
while not converged **do**
  Compute the gradient

$$\boldsymbol{g} = X^T(X\boldsymbol{w} - \mathbf{y})$$

  Choose step size α^t
  Update $\boldsymbol{w} \leftarrow \mathcal{S}(\boldsymbol{w} - \alpha^t\boldsymbol{g}, \alpha^t\lambda)$
  $t \leftarrow t + 1$
end while

Time complexity: $O(\text{nnz}(X))$ per iteration

Newton's Method

Newton's Method

- Iteratively conduct the following updates:

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - \alpha \nabla^2 f(\boldsymbol{x})^{-1} \nabla f(\boldsymbol{x})$$

  where α is the step size

- If $\alpha = 1$: converges quadratically when $\boldsymbol{x}^t$ is close enough to $\boldsymbol{x}^*$:

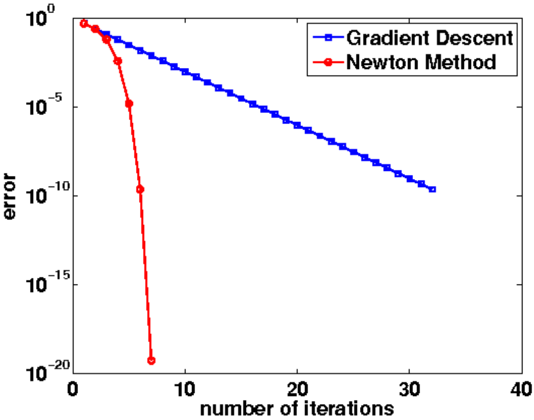
$$\|\boldsymbol{x}^{t+1} - \boldsymbol{x}^*\| \leq K \|\boldsymbol{x}^t - \boldsymbol{x}^*\|^2$$

  for some constant K. This means the error $f(\boldsymbol{x}^t) - f(\boldsymbol{x}^*)$ decays quadratically:

$$\beta, \beta^2, \beta^4, \beta^8, \beta^{16}, \dots$$

- Only need few iterations to converge in this "quadratic convergence region"

Newton's Method



However, Newton's update rule is more expensive than gradient descent/coordinate descent

Newton's Method

- Need to compute $\nabla^2 f(\boldsymbol{x})^{-1}\nabla f(\boldsymbol{x})$
- Closed form solution: $O(d^3)$ for solving a d dimensional linear system
- Usually solved by another iterative solver:

  gradient descent

  coordinate descent

  conjugate gradient method

  ...
- Useful for the cases where the quadratic subproblem can be solved more efficiently than the original problem
- Examples: primal L2-SVM/logistic regression, ℓ_1-regularized logistic regression, ...

Newton's Method(*)

- At each iteration, form an approximation of $f(\cdot)$:

$$f(\boldsymbol{x}^t + \boldsymbol{d}) \approx \tilde{f}_{\boldsymbol{x}^t}(\boldsymbol{d}) := f(\boldsymbol{x}^t) + \nabla f(\boldsymbol{x}^t)^T \boldsymbol{d} + \frac{1}{2\alpha} \boldsymbol{d}^T \nabla^2 f(\boldsymbol{x}) \boldsymbol{d}$$

- Update solution by $\boldsymbol{x}^{t+1} \leftarrow \boldsymbol{x}^t + \text{argmin}_{\boldsymbol{d}} \tilde{f}_{\boldsymbol{x}^t}(\boldsymbol{d})$
- When $\boldsymbol{x}$ is far away from $\boldsymbol{x}^*$, needs line search to gaurantee convergence
- Assume $LI \succeq \nabla^2 f(\boldsymbol{x}) \succeq mI$ for all $\boldsymbol{x}$, then $\alpha \leq \frac{m}{L}$ gaurantee the objective function value deacrease because

$$\frac{L}{m} \nabla^2 f(\boldsymbol{x}) \succeq \nabla^2 f(\mathbf{y}) \quad \forall \boldsymbol{x}, \mathbf{y}$$

- In practice, we often just use line search.

Proximal Newton (*)

- What if $f(\boldsymbol{x}) = g(\boldsymbol{x}) + h(\boldsymbol{x})$ and $h(\boldsymbol{x})$ is non-smooth ($h(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$)?
- At each iteration, form an approximation of $f(\cdot)$:

$$f(\boldsymbol{x}^t + \boldsymbol{d}) \approx \tilde{f}_{\boldsymbol{x}^t}(\boldsymbol{d}) := g(\boldsymbol{x}^t) + \nabla g(\boldsymbol{x}^t)^T \boldsymbol{d} + \frac{\alpha}{2} \boldsymbol{d}^T \nabla^2 g(\boldsymbol{x}) \boldsymbol{d} + h(\boldsymbol{x} + \boldsymbol{d})$$

- Update solution by $\boldsymbol{x}^{t+1} \leftarrow \boldsymbol{x}^t + \operatorname{argmin}_{\boldsymbol{d}} \tilde{f}_{\boldsymbol{x}^t}(\boldsymbol{d})$
- Need another iterative solver for solving the subproblem

Stochastic Gradient

Stochastic Gradient: Motivation

- Widely used for machine learning problems (with large number of samples)
- Given training samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, we usually want to solve the following empirical risk minimization (ERM) problem:

$$\operatorname*{argmin}_{\boldsymbol{w}} \sum_{i=1}^{n} \ell_i(\boldsymbol{x}_i),$$

  where each $\ell_i(\cdot)$ is the loss function
- Minimize the summation of individual loss on each sample

Stochastic Gradient

- Assume the objective function can be written as

$$f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})$$

- Stochastic gradient method:

  Iterative conducts the following updates

  1. Choose an index i (uniform) randomly
  2. $\boldsymbol{x}^{t+1} \leftarrow \boldsymbol{x}^t - \eta^t \nabla f_i(\boldsymbol{x}^t)$

- $\eta^t > 0$ is the step size

Stochastic Gradient

- Assume the objective function can be written as

$$f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})$$

- Stochastic gradient method:

  Iterative conducts the following updates
  1. Choose an index i (uniform) randomly
  2. $\boldsymbol{x}^{t+1} \leftarrow \boldsymbol{x}^t - \eta^t \nabla f_i(\boldsymbol{x}^t)$

- $\eta^t > 0$ is the step size
- Why does SG work?

$$E_i[\nabla f_i(\boldsymbol{x})] = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}) = \nabla f(\boldsymbol{x})$$

Stochastic Gradient

- Assume the objective function can be written as

$$f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})$$

- Stochastic gradient method:

  Iterative conducts the following updates
  1. Choose an index i (uniform) randomly
  2. $\boldsymbol{x}^{t+1} \leftarrow \boldsymbol{x}^t - \eta^t \nabla f_i(\boldsymbol{x}^t)$

- $\eta^t > 0$ is the step size
- Why does SG work?

$$E_i[\nabla f_i(\boldsymbol{x})] = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}) = \nabla f(\boldsymbol{x})$$

- Is it a fixed point method? No if $\eta > 0$ because $\boldsymbol{x}^* - \eta \nabla_i f(\boldsymbol{x}^*) \neq \boldsymbol{x}^*$
- Is it a descent method? No, because $f(\boldsymbol{x}^{t+1}) \not< f(\boldsymbol{x}^t)$

Stochastic Gradient

- Step size η has to decay to 0
  (e.g., $\eta^t = Ct^{-a}$ for some constant a, C)
- Many variants proposed recently (SVRG, SAGA, ...)
- Widely used in online setting

Stochastic Gradient: applying to ridge regression

- Objective function:

$$\operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2 + \lambda\|\boldsymbol{w}\|^2$$

- How to write as $\operatorname{argmin}_{\boldsymbol{w}} \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w})$?
- How to decompose into n components?

Stochastic Gradient: applying to ridge regression

- Objective function:

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2 + \lambda \|\boldsymbol{w}\|^2$$

- How to write as $\operatorname{argmin}_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} f_i(\boldsymbol{w})$?
- First approach: $f_i(\boldsymbol{w}) = (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2 + \lambda \|\boldsymbol{w}\|^2$
- Update rule:

$$\begin{aligned} \boldsymbol{w}^{t+1} &\leftarrow \boldsymbol{w}^t - 2\eta^t (\boldsymbol{w}^T \boldsymbol{x}_i - y_i) \boldsymbol{x}_i - 2\eta^t \lambda \boldsymbol{w} \\ &= (1 - 2\eta^t \lambda) \boldsymbol{w} - 2\eta^t (\boldsymbol{w}^T \boldsymbol{x}_i - y_i) \boldsymbol{x}_i \end{aligned}$$

Stochastic Gradient: applying to ridge regression

- Objective function:

$$\operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2 + \lambda \|\boldsymbol{w}\|^2$$

- How to write as $\operatorname{argmin}_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} f_i(\boldsymbol{w})$?
- First approach: $f_i(\boldsymbol{w}) = (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2 + \lambda \|\boldsymbol{w}\|^2$
- Update rule:

$$\begin{aligned} \boldsymbol{w}^{t+1} &\leftarrow \boldsymbol{w}^t - 2\eta^t (\boldsymbol{w}^T \boldsymbol{x}_i - y_i) \boldsymbol{x}_i - 2\eta^t \lambda \boldsymbol{w} \\ &= (1 - 2\eta^t \lambda) \boldsymbol{w} - 2\eta^t (\boldsymbol{w}^T \boldsymbol{x}_i - y_i) \boldsymbol{x}_i \end{aligned}$$

- Need $O(d)$ complexity per iteration even if data is sparse

Stochastic Gradient: applying to ridge regression

- Objective function:

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2 + \lambda \|\boldsymbol{w}\|^2$$

- How to write as $\operatorname{argmin}_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} f_i(\boldsymbol{w})$?
- First approach: $f_i(\boldsymbol{w}) = (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2 + \lambda \|\boldsymbol{w}\|^2$
- Update rule:

$$\begin{aligned} \boldsymbol{w}^{t+1} &\leftarrow \boldsymbol{w}^t - 2\eta^t (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)\boldsymbol{x}_i - 2\eta^t \lambda \boldsymbol{w} \\ &= (1 - 2\eta^t \lambda)\boldsymbol{w} - 2\eta^t (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)\boldsymbol{x}_i \end{aligned}$$

- Need $O(d)$ complexity per iteration even if data is sparse
- Solution: store $\boldsymbol{w} = s\boldsymbol{v}$ where s is a scalar

Stochastic Gradient: applying to ridge regression

- Objective function:

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2 + \lambda\|\boldsymbol{w}\|^2$$

- Second approach:

  define $\Omega_i = \{j \mid X_{ij} \neq 0\}$ for $i = 1, \ldots, n$

  define $n_j = |\{i \mid X_{ij} \neq 0\}|$ for $j = 1, \ldots, d$

  define $f_i(\boldsymbol{w}) = (\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2 + \sum_{j\in\Omega_i} \frac{\lambda n}{n_j} w_j^2$

Stochastic Gradient: applying to ridge regression

- Objective function:

$$\operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2 + \lambda\|\boldsymbol{w}\|^2$$

- Second approach:

  define $\Omega_i = \{j \mid X_{ij} \neq 0\}$ for $i = 1, \ldots, n$

  define $n_j = |\{i \mid X_{ij} \neq 0\}|$ for $j = 1, \ldots, d$

  define $f_i(\boldsymbol{w}) = (\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2 + \sum_{j\in\Omega_i} \frac{\lambda n}{n_j} w_j^2$
- Update rule when selecting index i:

$$w_j^{t+1} \leftarrow w_j^t - 2\eta^t(\boldsymbol{x}_i^T\boldsymbol{w}^t - y_i)X_{ij} - \frac{2\eta^t\lambda n}{n_j}w_j^t, \quad \forall j \in \Omega_i$$

- Solution: update can be done in $O(|\Omega_i|)$ operations

Coming up

- Next class: Parallel Optimization Methods

Questions?