STA141C: Big Data & High Performance Statistical Computing
Lecture 8: Optimization

Cho-Jui Hsieh
UC Davis

May 9, 2017
Optimization
Numerical Optimization

- Numerical Optimization:
  \[ \min_X f(X) \]

- Can be applied to many areas.

- **Machine Learning**: find a model that *minimizes* the prediction error.

![Diagram of function optimization](image)

\[ f(X) \]

Optimal Solution
Properties of the Function

- Smooth function: functions with continuous derivative.
  Example: ridge regression

  \[
  \text{argmin} \quad \frac{1}{2} \|Xw - y\|^2 + \frac{\lambda}{2} \|w\|^2
  \]

- Non-smooth function: Lasso, primal SVM

  Lasso: \(\text{argmin} \quad \frac{1}{2} \|Xw - y\|^2 + \lambda \|w\|_1\)

  SVM: \(\text{argmin} \quad \sum_{i=1}^{n} \max(0, 1 - y_i w^T x_i) + \frac{\lambda}{2} \|w\|^2\)
A function is convex if:

\[ \forall x_1, x_2, \forall t \in [0, 1], f(tx_1 + (1 - t)x_2) \leq tf(x_1) + (1 - t)f(x_2) \]

No local optimum (why?)
Convex Functions

- If \( f(x) \) is twice differentiable, then
  \[
  f \text{ is convex if and only if } \nabla^2 f(x) \succeq 0
  \]
- Optimal solution may not be unique:
  has a set of optimal solutions \( S \)
- Gradient: capture the first order change of \( f \):
  \[
  f(x + \alpha d) = f(x) + \alpha \nabla f(x)^T d + O(\alpha^2)
  \]
- If \( f \) is convex and differentiable, we have the following optimality condition:
  \[
  x^* \in S \text{ if and only if } \nabla f(x) = 0
  \]
Nonconvex Functions

- If $f$ is nonconvex, most algorithms can only converge to **stationary points**
- $\bar{x}$ is a stationary point if and only if $\nabla f(\bar{x}) = 0$
- 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 $\delta^*$ by (approximately) minimizing
    \[
    \arg\min_{\delta} f(x + \delta e_i)
    \]
  Step 3: $x_i \leftarrow x_i + \delta^*$
Coordinate Descent (update sequence)

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

\[
\underbrace{x_1, x_2, \ldots, x_n}, \quad \underbrace{x_1, x_2, \ldots, x_n}, \ldots
\]

  1st outer iteration  2nd outer iteration

- Randomly permute the sequence for each outer iteration (faster convergence in practice)

  Some interesting theoretical analysis for this recently

Coordinate Descent (update sequence)

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

\[
\underbrace{x_1, x_2, \ldots, x_n}, \quad \underbrace{x_1, x_2, \ldots, x_n}, \ldots
\]

1st outer iteration 2nd outer iteration

- Randomly permute the sequence for each outer iteration (faster convergence in practice)

  Some interesting theoretical analysis for this recently


- 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: $|\nabla_i f(x)|$
  - By first and second derivative: $|\nabla_i f(x)/\nabla_{ii}^2 f(x)|$
  - By maximum reduction of objective function

$$i^* = \arg\max_{i=1,...,n} \left( f(x) - \min_{\delta} f(x + \delta e_i) \right)$$

- 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

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

- 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 optimal if $f(\cdot)$ is convex and smooth
- Has a linear convergence rate if $f(\cdot)$ is strongly convex
- Linear convergence: error $f(x_t) - f(x^*)$ decays as
  $$\beta, \beta^2, \beta^3, \ldots$$
  for some $\beta < 1$.
- Local linear convergence: an algorithm converges linearly after
  $\|x - x^*\| \leq K$ for some $K > 0$
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:
  \[ x^{t+1} \leftarrow x^t - \alpha \nabla f(x^t) \]
  where \( \alpha > 0 \) is the step size
- It is a fixed point iteration method:
  \[ x - \alpha \nabla f(x) = x \text{ if and only if } x \text{ is an optimal solution} \]
- Step size too large \( \Rightarrow \) diverge; too small \( \Rightarrow \) slow convergence
Gradient Descent: step size

- A twice differentiable function has $L$-Lipchitz continuous gradient if and only if
  \[ \nabla^2 f(x) \leq L \quad \forall x \]

- In this case, Condition 2 is satisfied if $\alpha < \frac{1}{L}$

**Theorem:** gradient descent converges if $\alpha < \frac{1}{L}$

**Theorem:** gradient descent converges linearly with $\alpha < \frac{1}{L}$ if $f$ is strongly convex
Gradient Descent

- In practice, we do not know $L$.
- Step size $\alpha$ too large: the algorithm diverges
- Step size $\alpha$ too small: the algorithm converges very slowly
**Gradient Descent: line search**

- $d^*$ is a “descent direction” if and only if $(d^*)^T \nabla f(x) < 0$
- Armijo rule backtracking line search:
  
  Try $\alpha = 1, \frac{1}{2}, \frac{1}{4}, \ldots$ until it satisfies
  
  $$f(x + \alpha d^*) \leq f(x) + \gamma \alpha (d^*)^T \nabla f(x)$$

  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
Gradient Descent: applying to ridge regression

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

Output: Solution $w^* := \arg\min_w \frac{1}{2} \|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$

1: $t = 0$
2: while not converged do
3: Compute the gradient
4: $g = X^T(Xw - y) + \lambda w$
5: Choose step size $\alpha^t$
6: Update $w \leftarrow w - \alpha^t g$
7: $t \leftarrow t + 1$
8: end while

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

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

\[
\arg\min_w \frac{1}{2} \|Xw - y\|^2 + \lambda \|w\|_1 \tag{non-differentiable}
\]

- General composite function minimization:

\[
\arg\min_x f(x) := \{g(x) + h(x)\}
\]

where \(g\) is smooth and convex, \(h\) is convex but may be non-differentiable

- Usually assume \(h\) is simple (for computational efficiency)
At each iteration, form an approximation of $f(\cdot)$:

$$f(x^t + d) \approx \tilde{f}_{x^t}(d) := g(x^t) + \nabla g(x^t)^T d + \frac{1}{2} d^T \left( \frac{1}{\alpha} I \right) d + h(x^t + d)$$

$$= g(x^t) + \nabla g(x^t)^T d + \frac{1}{2\alpha} d^T d + h(x^t + d)$$

Update solution by $x^{t+1} \leftarrow x^t + \text{argmin}_d \tilde{f}_{x^t}(d)$

This is called “proximal” gradient descent

Usually $d^* = \text{argmin}_d \tilde{f}_{x^t}(d)$ has a closed form solution
The subproblem:

\[ x^{t+1} = x^t + \arg\min_d \nabla g(x^t)^T d + \frac{1}{2\alpha} d^T d + \lambda \| x^t + d \|_1 \]

\[ = \arg\min_u \frac{1}{2} \| u - (x^t - \alpha \nabla g(x^t)) \|_2^2 + \lambda \alpha \| u \|_1 \]

\[ = \mathcal{S}(x^t - \alpha \nabla g(x^t), \alpha \lambda), \]

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:** \( X \in \mathbb{R}^{N \times d}, y \in \mathbb{R}^{N}, \) initial \( w^{(0)} \)

**Output:** Solution \( w^* := \arg\min_w \frac{1}{2} \|Xw - y\|^2 + \lambda \|w\|_1 \)

1. \( t = 0 \)
2. while not converged do
3.   Compute the gradient
4.     \[ g = X^T(Xw - y) \]
5.   Choose step size \( \alpha^t \)
6.   Update \( w \leftarrow \text{s}(w - \alpha^t g, \alpha^t \lambda) \)
7.  \( t \leftarrow t + 1 \)
8. end while

Time complexity: \( O(\text{nnz}(X)) \) per iteration
Newton’s Method
Newton’s Method

- Iteratively conduct the following updates:
  \[
  x \leftarrow x - \alpha \nabla^2 f(x)^{-1} \nabla f(x)
  \]

  where \( \alpha \) is the step size

- If \( \alpha = 1 \): converges quadratically when \( x^t \) is close enough to \( x^* \):
  \[
  \|x^{t+1} - x^*\| \leq K \|x^t - x^*\|^2
  \]
  for some constant \( K \). This means the error \( f(x^t) - f(x^*) \) decays quadratically:
  \[
  \beta, \beta^2, \beta^4, \beta^8, \beta^{16}, \ldots
  \]

- 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 $\nabla^2 f(x)^{-1} \nabla f(x)$
- 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
  - ...
- Examples: primal L2-SVM/logistic regression, $\ell_1$-regularized logistic regression, ...
Stochastic Gradient Method
Stochastic Gradient Method: Motivation

- Widely used for machine learning problems (with large number of samples)
- Given training samples $x_1, \ldots, x_n$, we usually want to solve the following empirical risk minimization (ERM) problem:

$$\arg\min_w \sum_{i=1}^{n} \ell_i(x_i),$$

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

- Assume the objective function can be written as
  \[ f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \]

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

- \( \eta^t > 0 \) is the step size

Why does SG work?

\[ \mathbb{E}_i [\nabla f_i(x)] = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x) = \nabla f(x) \]

Is it a fixed point method? No if \( \eta > 0 \) because

\[ x^* - \eta \nabla f_i(x^*) \neq x^* \]

Is it a descent method? No, because

\[ f(x^{t+1}) \neq f(x^t) \]
Stochastic Gradient Method

- Assume the objective function can be written as

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

- Stochastic gradient method:

  Iterative conducts the following updates

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

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

\[
E_i[\nabla f_i(x)] = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x) = \nabla f(x)
\]
Stochastic Gradient Method

- Assume the objective function can be written as
  \[ f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \]

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

- \( \eta^t > 0 \) is the step size
- Why does SG work?
  \[ E_i[\nabla f_i(x)] = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x) = \nabla f(x) \]

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

- Step size $\eta$ has to decay to 0
  
  \[ \eta^t = C t^{-a} \text{ for some constant } a, C \]

  SGD converges sub-linearly

- Many variants proposed recently
  
  - SVRG, SAGA (2013, 2014): variance reduction
  - AdaGrad (2011): adaptive learning rate
  - RMSProp (2012): estimate learning rate by a running average of gradient.
  - Adam (2015): adaptive moment estimation

- Widely used in machine learning
AdaGrad: adaptive step size for each parameter

Update rule at the $t$-th iteration:

- Compute $\mathbf{g} = \nabla f(\mathbf{x})$
- Estimate the second moment: $G_{ii} = G_{ii} + g_i^2$ for all $i$
- Parameter update: $x_i \leftarrow x_i - \frac{\eta}{\sqrt{G_{ii}}} g_i$ for all $i$

Proposed for convex optimization in:

“Adaptive subgradient methods for online learning and stochastic optimization” (JMLR 2011)

“Adaptive Bound Optimization for Online Convex Optimization” (COLT 2010)
AdaGrad: adaptive step size for each parameter

Update rule at the $t$-th iteration:

- Compute $\mathbf{g} = \nabla f(x)$
- Estimate the second moment: $G_{ii} = G_{ii} + g_i^2$ for all $i$
- Parameter update: $x_i \leftarrow x_i - \frac{\eta}{\sqrt{G_{ii}}} g_i$ for all $i$
Stochastic Gradient: applying to ridge regression

- **Objective function:**

\[
\arg\min_w \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda \|w\|^2
\]

- How to write as \(\arg\min_w \frac{1}{n} \sum_{i=1}^{n} f_i(w)\)?
- How to decompose into \(n\) components?
Stochastic Gradient: applying to ridge regression

- Objective function:

\[
\argmin_w \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda \|w\|^2
\]

- How to write as \(\argmin_w \frac{1}{n} \sum_{i=1}^{n} f_i(w)\)?

- First approach: \(f_i(w) = (w^T x_i - y_i)^2 + \lambda \|w\|^2\)

- Update rule:

\[
w^{t+1} \leftarrow w^t - 2\eta^t (w^T x_i - y_i)x_i - 2\eta^t \lambda w
\]

\[
= (1 - 2\eta^t \lambda)w - 2\eta^t (w^T x_i - y_i)x_i
\]
Stochastic Gradient: applying to ridge regression

- Objective function:

$$\arg\min_{w} \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda \|w\|^2$$

- How to write as $\arg\min_{w} \frac{1}{n} \sum_{i=1}^{n} f_i(w)$?

- First approach: $f_i(w) = (w^T x_i - y_i)^2 + \lambda \|w\|^2$

- Update rule:

$$w^{t+1} \leftarrow w^t - 2\eta^t (w^T x_i - y_i)x_i - 2\eta^t \lambda w$$

$$= (1 - 2\eta^t \lambda)w - 2\eta^t (w^T x_i - y_i)x_i$$
Coming up

- Other classification models

Questions?