Modern Optimization Techniques

Lucas Rego Drumond

Information Systems and Machine Learning Lab (ISMLL)
Institute of Computer Science
University of Hildesheim, Germany

Distributed Optimization and Review

Outline

Outline

Problem set up

Given an equality constrained problem:

$$\begin{array}{rl} \text{minimize} & f_0(\mathbf{x}) \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \end{array}$$

The Lagrangian is given by:

$$L(\mathbf{x}, \nu) = f_0(\mathbf{x}) + \nu(A\mathbf{x} - \mathbf{b})$$

And the dual:

$$g(\nu) = \inf_{\mathbf{x}} L(\mathbf{x}, \nu)$$

We solve it by

1. $\nu^* := \arg\max_{\mathbf{x}} g(\nu)$
2. $\mathbf{x}^* := \arg\min_{\mathbf{x}} L(\mathbf{x}, \nu^*)$

Dual Ascent

We can apply the gradient method for maximizing the dual:

$$\nu^{t+1} = \nu^t + \mu^t \nabla g(\nu^t)$$

where, given that $\mathbf{x}' = \arg\min L(\mathbf{x}, \nu^t)$:

$$\nabla g(\nu^t) = A\mathbf{x}' + \mathbf{b}$$

Which gives us the following method for solving the dual:

$$\begin{aligned} \mathbf{x}^{t+1} &\leftarrow \arg\min L(\mathbf{x}, \nu^t) \\ \nu^{t+1} &\leftarrow \nu^t + \mu^t(A\mathbf{x}^{t+1} + \mathbf{b}) \end{aligned}$$

Dual Decomposition

Now suppose f_0 can be rewritten like this:

$$f_0(\mathbf{x}) = f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) + \ldots + f_N(\mathbf{x}_N), \qquad \mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$$

Partitioning $A = [A_1 \cdots A_N]$ so that $\sum_{i=1}^{n} A_i\mathbf{x}_i = A\mathbf{x}$, we can write the Lagrangian as:

$$\begin{aligned}
L(\mathbf{x}, \nu) &= f_0(\mathbf{x}) + \nu(A\mathbf{x} - \mathbf{b}) \\
L(\mathbf{x}, \nu) &= f_1(\mathbf{x}_1) + \ldots + f_N(\mathbf{x}_N) + \nu(A_1\mathbf{x}_1 + \ldots + A_n\mathbf{x}_n - \mathbf{b}) \\
L(\mathbf{x}, \nu) &= f_1(\mathbf{x}_1) + \nu^T A_1\mathbf{x}_1 + \ldots + f_N(\mathbf{x}_N) + \nu^T A_N\mathbf{x}_N - \nu^T\mathbf{b} \\
L(\mathbf{x}, \nu) &= L_1(\mathbf{x}_1, \nu) + \ldots + L_N(\mathbf{x}_N, \nu) - \nu^T\mathbf{b}
\end{aligned}$$

Where: $L_i(\mathbf{x}_i, \nu) = f_i(\mathbf{x}_i) + \nu^T A_i\mathbf{x}_i$

Dual Decomposition

The problem

$$\mathbf{x}^{t+1} := \arg\min_{\mathbf{x}} L(\mathbf{x}, \nu^t)$$

With the Lagrange function

$$L(\mathbf{x}, \nu) = \sum_{i=1}^{N} L_i(\mathbf{x}_i, \nu) - \nu^T \mathbf{b}$$

where $L_i(\mathbf{x}_i, \nu) = f_i(\mathbf{x}_i) + \nu^T A_i \mathbf{x}_i$

Can be solved by N minimization steps:

$$\mathbf{x}_i^{t+1} := \arg\min_{\mathbf{x}_i} L_i(\mathbf{x}_i, \nu^t)$$

carried out in parallel

Dual Decomposition

Dual decomposition method:

$$\mathbf{x}_i^{t+1} \leftarrow \arg\min_{\mathbf{x}_i} L_i(\mathbf{x}_i, \nu^t)$$

$$\nu^{t+1} \leftarrow \nu^t + \mu^t \left(\sum_{i=1}^{n} A_i \mathbf{x}_i^{t+1} + \mathbf{b} \right)$$

At each step:

- ν^t have to be broadcasted
- $\mathbf{x}_i^{t+1}$ are updated in parallel
- $A_i \mathbf{x}_i^{t+1}$ are gathered to compute the sum $\sum_{i=1}^{n} A_i \mathbf{x}_i^{t+1}$

Works if assumptions hold but often slow!!

Method of Multipliers

The method of multipliers uses the augmented lagrangian, $s > 0$:

$$L_s(\mathbf{x}, \nu) = f_0(\mathbf{x}) + \nu(A\mathbf{x} - \mathbf{b}) + (\frac{s}{2})||A\mathbf{x} - \mathbf{b}||_2^2$$

and solves the dual problem through the following steps:

$$\mathbf{x}^{t+1} \leftarrow \arg\min_{\mathbf{x}} L_s(\mathbf{x}, \nu^t)$$
$$\nu^{t+1} \leftarrow \nu^t + s\left(A\mathbf{x}^{t+1} + \mathbf{b}\right)$$

- Converges under more relaxed assumptions
- AugmentedLagrangian not separable because of the additional term (no parallelization)

Alternating Direction Method of Multipliers (ADMM)

ADMM assumes the problem can take the form:

$$\begin{array}{rl} \text{minimize} & f_0(\mathbf{x}) + h_0(\alpha) \\ \text{subject to} & A\mathbf{x} + B\alpha = \mathbf{c} \end{array}$$

which has the following augmented lagragian:

$$L_s(\mathbf{x}, \alpha, \nu) = f_0(\mathbf{x}) + h_0(\alpha) + \nu^T(A\mathbf{x} + B\alpha - \mathbf{c}) + (\frac{s}{2})||A\mathbf{x} + B\alpha - \mathbf{c}||_2^2$$

and solves the dual problem through the following steps:

$$\begin{aligned} \mathbf{x}^{t+1} &\leftarrow \arg\min_{\mathbf{x}} L_s(\mathbf{x}, \alpha^t, \nu^t) \\ \alpha^{t+1} &\leftarrow \arg\min_{\alpha} L_s(\mathbf{x}^{t+1}, \alpha, \nu^t) \\ \nu^{t+1} &\leftarrow \nu^t + s\left(A\mathbf{x}^{t+1} + B\alpha^{t+1} - \mathbf{c}\right) \end{aligned}$$

Alternating Direction Method of Multipliers (ADMM)

$$\begin{aligned}
\mathbf{x}^{t+1} &\leftarrow \arg\min_{\mathbf{x}} L_s(\mathbf{x}, \alpha^t, \nu^t) \\
\alpha^{t+1} &\leftarrow \arg\min_{\alpha} L_s(\mathbf{x}^{t+1}, \alpha, \nu^t) \\
\nu^{t+1} &\leftarrow \nu^t + s\left(A\mathbf{x}^{t+1} + B\alpha^{t+1} - \mathbf{c}\right)
\end{aligned}$$

At first ADMM seems very similar to the method of multipliers

- It reduces to the method of multipliers if $\mathbf{x}$ and α are optimized jointly
- if f_0 or h_0 are separable we can now perform the updates of $\mathbf{x}$ (or α) in parallel

ADMM: scaled form

We can rewrite the ADMM algorithm in a more convenient form.

From the augumented Lagrangian:

$$L_s(\mathbf{x}, \alpha, \nu) = f_0(\mathbf{x}) + h_0(\alpha) + \nu^T(A\mathbf{x} + B\alpha - \mathbf{c}) + (\frac{s}{2})||A\mathbf{x} + B\alpha - \mathbf{c}||_2^2$$

we can define: $\mathbf{r} = A\mathbf{x} + B\alpha - \mathbf{c}$ so that:

$$\begin{aligned} \nu^T\mathbf{r} + (\frac{s}{2})||\mathbf{r}||_2^2 &= \frac{s}{2}||\mathbf{r} - \frac{1}{s}\nu||_2^2 - \frac{1}{2s}||\nu||_2^2 \\ &= \frac{s}{2}||\mathbf{r} + \mathbf{u}||_2^2 - \frac{s}{2}||\mathbf{u}||_2^2 \end{aligned}$$

where $\mathbf{u} = \frac{1}{s}\nu$

ADMM: scaled form

From the augumented Lagrangian:

$$L_s(\mathbf{x}, \alpha, \nu) = f_0(\mathbf{x}) + h_0(\alpha) + \nu^T(A\mathbf{x} + B\alpha - \mathbf{c}) + (\frac{s}{2})||A\mathbf{x} + B\alpha - \mathbf{c}||_2^2$$

And $\mathbf{r} = A\mathbf{x} + B\alpha - \mathbf{c}$:

$$\nu^T\mathbf{r} + (\frac{s}{2})||\mathbf{r}||_2^2 = \frac{s}{2}||\mathbf{r} + \mathbf{u}||_2^2 - \frac{s}{2}||\mathbf{u}||_2^2$$

where $\mathbf{u} = \frac{1}{s}\nu$, we have:

$$\mathbf{x}^{t+1} \leftarrow \arg\min_{\mathbf{x}} f_0(\mathbf{x}) + \frac{s}{2}||A\mathbf{x} + B\alpha^t - \mathbf{c} + \mathbf{u}^t||_2^2$$
$$\alpha^{t+1} \leftarrow \arg\min_{\alpha} h_0(\alpha) + \frac{s}{2}||A\mathbf{x}^{t+1} + B\alpha - \mathbf{c} + \mathbf{u}^t||_2^2$$
$$\mathbf{u}^{t+1} \leftarrow \mathbf{u}^t + A\mathbf{x}^{t+1} + B\alpha^{t+1} - \mathbf{c}$$

Example: Machine Learning

The data is represented it as:

$$D_{m,n} = \begin{pmatrix} 1 & d_{1,1} & d_{1,2} & \dots & d_{1,n} \\ 1 & d_{2,1} & d_{2,2} & \dots & d_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & d_{m,1} & d_{m,2} & \dots & d_{m,n} \end{pmatrix} \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

We want to learn a model to predict y from X through parameters $\mathbf{x}$:

$$\hat{y}_i = \mathbf{x}^T \mathbf{d_i} = \mathbf{x}_0 1 + \mathbf{x}_1 d_{i,1} + \mathbf{x}_2 d_{i,2} + \ldots + \mathbf{x}_n d_{i,n}$$

Example: Machine Learning

Be $l : \mathbb{R}^m \to \mathbb{R}$ is a loss function and $r : \mathbb{R}^n \to \mathbb{R}$ is a regularization term the problem of learning a linear model can be written as:

$$\text{minimize } l(D\mathbf{x} - \mathbf{y}) + r(\mathbf{x})$$

Most losses l can be decomposed into losses on datapoints:

$$\text{minimize } \sum_{i=1}^{m} l_i(\mathbf{x}^T\mathbf{d_i} - y_i) + r(\mathbf{x})$$

Example: Ridge (Linear) Regression:

$$\text{minimize } ||D\mathbf{x} - \mathbf{y}||_2^2 + \lambda||\mathbf{x}||_2^2 =$$
$$\text{minimize } \sum_{i=1}^{m}(\mathbf{x}^T\mathbf{d_i} - y_i)^2 + \lambda\sum_{j=1}^{n}\mathbf{x}_j$$

Example: Machine Learning

Now we can rewrite our problem

$$\text{minimize } l(D\mathbf{x} - \mathbf{y}) + r(\mathbf{x})$$

as

$$\begin{aligned} \text{minimize} \quad & l(D\mathbf{x} - \mathbf{y}) + r(\alpha) \\ \text{subject to} \quad & \mathbf{x} - \alpha = 0 \end{aligned}$$

And solve it through ADMM:

$$\begin{aligned}
\mathbf{x}^{t+1} &\leftarrow \arg\min_{\mathbf{x}} l(D\mathbf{x} - \mathbf{y}) + {\nu^t}^T(\mathbf{x} - \alpha^t) + \frac{s}{2}||\mathbf{x} - \alpha^t||_2^2 \\
\alpha^{t+1} &\leftarrow \arg\min_{\alpha} r(\alpha) + {\nu^t}^T(\mathbf{x}^{t+1} - \alpha) + \frac{s}{2}||\mathbf{x}^{t+1} - \alpha||_2^2 \\
\nu^{t+1} &\leftarrow \nu^t + s\left(A\mathbf{x}^{t+1} + B\alpha^{t+1} - \mathbf{c}\right)
\end{aligned}$$

Example: Machine Learning

Given that the loss function is separable:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{M} l_i(\mathbf{x}^T \mathbf{d_i} - y_i) + r(\alpha) \\ \text{subject to} \quad & \mathbf{x} - \alpha = 0 \end{aligned}$$

We can rewrite the algorithm like:

$$\begin{aligned} \mathbf{x}_i^{t+1} &\leftarrow \arg\min_{\mathbf{x}_i} l(D_i \mathbf{x}_i - \mathbf{y}_i) + \frac{s}{2} ||\mathbf{x}_i - \alpha^t + \mathbf{u}_i^t||_2^2 \\ \alpha^{t+1} &\leftarrow \arg\min_{\alpha} r(\alpha) + \frac{Ms}{2} ||\alpha - \bar{\mathbf{x}}^{t+1} - \bar{\mathbf{u}}^t||_2^2 \\ \mathbf{u}_i^{t+1} &\leftarrow \mathbf{u}_i^t + \mathbf{x}_i^{t+1} - \alpha^{t+1} \end{aligned}$$

And solve for different $\mathbf{x}_i$ in parallel

Outline

Unconstrained Optimization Problems

An **unconstrained optimization problem** has the form:

$$\text{minimize} \quad f_0(\mathbf{x})$$

Where:

- $f_0 : \mathbb{R}^n \to \mathbb{R}$ is convex, twice differentiable
- An optimal $\mathbf{x}^*$ exists and $f(\mathbf{x}^*)$ is attained and finite

Descent Methods

The next point is generated using

- A step size μ
- A direction $\Delta \mathbf{x}$ such that

$$f_0(\mathbf{x}^t + \mu \Delta \mathbf{x}^{t-1}) < f_0(\mathbf{x}^{t-1})$$

```
1: procedure DescentMethod
   input: $f_0$
2:     Get initial point $\mathbf{x}$
3:     repeat
4:         Get Update Direction $\Delta \mathbf{x}$
5:         Get Step Size $\mu$
6:         $\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \mu \Delta \mathbf{x}^t$
7:     until convergence
8:     return $\mathbf{x}$, $f_0(\mathbf{x})$
9: end procedure
```

Methods For Unconstrained Optimization

- Gradient Descent:

$$\Delta \mathbf{x} = -\nabla f_0(\mathbf{x})$$

- Stochastic Gradient Descent:
  - If the function is if the form $f_0(\mathbf{x}) = \sum_{i=1}^{m} g(\mathbf{x}, i)$:
  -

$$\Delta_i \mathbf{x} = -\nabla g(\mathbf{x}, i)$$

- Coordinate Descent:

$$\mathbf{x}_k^{(t)} \leftarrow \arg\min_{\mathbf{x}_k} f_0(\mathbf{x}_1^{(t)}, \mathbf{x}_2^{(t)}, \ldots, \mathbf{x}_k, \ldots, \mathbf{x}_n^{(t-1)})$$

- Newton's Method:

$$\Delta \mathbf{x} = -\nabla^2 f_0(\mathbf{x})^{-1} \nabla f_0(\mathbf{x})$$

Choosing the step size

- The step size μ is a crucial parameter to be tuned
- Possible alternatives:
  - Fixed step size
  - Line Search
  - Bold-Driver
  - Adagrad

The Subgradient Method

Be f_0 a nondifferentiable and convex function $f_0 : \mathbb{R}^n \rightarrow \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^n$:

$$\text{minimize} \qquad f_0(\mathbf{x})$$

Be $\mathbf{g}^t$ **any** subgradient of f_0 at $\mathbf{x}^t$

1. Start with an initial solution $\mathbf{x}^{(0)}$
2. $t \leftarrow 0$
3. Repeat until convergence
   - 3.1 Find $\mathbf{x}^{t+1} = \mathbf{x}^t - \mu_t \mathbf{g}^t$
   - 3.2 $t \leftarrow t + 1$
4. Return $f_{0_\text{best}} = \min_{j=1,\dots,t} f_0(\mathbf{x}^j)$

The subgradient method is not a descent method!

Convex Constrained Optimization Problems

A constrained optimization problem:

$$\begin{array}{rll} \text{minimize} & f_0(\mathbf{x}) & \\ \text{subject to} & f_i(\mathbf{x}) \leq 0, & i = 1, \dots, m \\ & h_j(\mathbf{x}) = 0, & i = 1, \dots, p \end{array}$$

is convex iff:

- $f_0, \dots, f_m$ are convex
- $h_1, \dots, h_p$ are affine

$$\begin{array}{rll} \text{minimize} & f_0(\mathbf{x}) & \\ \text{subject to} & f_i(\mathbf{x}) \leq 0, & i = 1, \dots, m \\ & A\mathbf{x} = \mathbf{b} & \end{array}$$

Lagrangian

The **primal Lagrangian** of a constrained optimization problem is a function $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$:

$$L(\mathbf{x}, \lambda, \nu) = f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x})$$

Be $\mathcal{D}$ the domain of the problem, the **dual Lagrangian** of a constrained optimization problem is a function $g : \times\mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$:

$$\begin{aligned} g(\lambda, \nu) &= \inf_{\mathbf{x} \in \mathcal{D}} L(\mathbf{x}, \lambda, \nu) \\ &= \inf_{\mathbf{x} \in \mathcal{D}} \left(f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x}) \right) \end{aligned}$$

Karush-Kuhn-Tucker (KKT) Conditions

The following conditions are called the KKT conditions:

1. Primal feasibility: $f_i(\mathbf{x}) \leq 0$ and $h_j(\mathbf{x}) = 0$ for all i, j
2. Dual feasibility: $\lambda \succeq 0$
3. Complementary Slackness: $\lambda_i f_i(\mathbf{x}) = 0$ for all i
4. Stationarity: $\nabla f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \nabla f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i \nabla h_i(\mathbf{x}) = 0$

If strong duality holds and $\mathbf{x}, \lambda, \nu$ are optimal, then they **must** satisfy the KKT conditions

If $\mathbf{x}, \lambda, \nu$ satisfy the KKT conditions, then $\mathbf{x}$ is the primal solution and (λ, ν) is the dual solution

Solving ECP through the KKT Conditions

Given the following problem:

$$\begin{array}{rl} \text{minimize} & f_0(\mathbf{x}) \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \end{array}$$

The optimal solution $\mathbf{x}^*$ must fulfil the KKT Conditions:

- Primal feasibility: $h_j(\mathbf{x}^*) = 0$
- Stationarity: $\nabla f_0(\mathbf{x}^*) + \sum_{i=1}^{p} \nu_i \nabla h_i(\mathbf{x}^*) = 0$

$$\begin{bmatrix} P & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}^* \\ \nu^* \end{bmatrix} = \begin{bmatrix} -\mathbf{q} \\ \mathbf{b} \end{bmatrix}$$

Newton's method for Equality Constrained Problems

```
1: procedure NEWTONS METHOD
   input: $f_0$, initial feasible point $\mathbf{x} \in \operatorname{dom} f_0$ and $A\mathbf{x} = \mathbf{b}$
2:     repeat
3:         Get $\Delta$ by solving $\begin{bmatrix} \nabla^2 f_0(\mathbf{x}) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} -\nabla f_0(\mathbf{x}) \\ \mathbf{0} \end{bmatrix}$
4:         Get Step Size $\mu$
5:         $\mathbf{x} \leftarrow \mathbf{x} + \mu\Delta\mathbf{x}$
6:     until convergence
7:     return $\mathbf{x}$, $f_0(\mathbf{x})$
8: end procedure
```

What if we don't have a feasible $\mathbf{x}$ to start with?

The Interior Point Methods

```
1:  procedure BARRIER METHOD
    input: strictly feasible $\mathbf{x}^{(0)}$, $t^0 > 0$, step size $\mu > 1$, tolerance $\epsilon > 0$
2:      $t := t^0$
3:      $\mathbf{x} := \mathbf{x}^0$
4:      while $m/t < \epsilon$ do
            /* Centering Step */
5:          $\mathbf{x}^*(t) := \arg\min_{\mathbf{x}(t)} tf_0(\mathbf{x}(t)) + \phi(\mathbf{x}(t))$,
                                subject to $A\mathbf{x}(t) = \mathbf{b}$,
                                starting at $\mathbf{x}(t) = \mathbf{x}$
6:          $\mathbf{x} := \mathbf{x}^*(t)$
7:          $t := \mu t$
8:      end while
9:      return $\mathbf{x}$
10: end procedure
```

Cutting Plane Methods

```
1: procedure CUTTING PLANE METHOD
   input: Initial Polyhedron $\mathcal{P}_0 = \{\alpha | C\alpha \succeq \mathbf{d}\}$

2:     $t \leftarrow 0$
3:     while not converged do
4:         Get a point $\mathbf{x}^{t+1} \in \mathcal{P}_t$
5:         Query the oracle at $\mathbf{x}^{t+1}$
6:         if $\mathbf{x}^{t+1} \in \mathcal{B}$ then
7:             return $\mathbf{x}^{t+1}$
8:         end if
9:         $\mathcal{P}_{t+1} \leftarrow \mathcal{P}_t \cap \{\alpha | \mathbf{u}_{t+1}^T \alpha \leq v_{t+1}\}$
10:        if $\mathcal{P}_{t+1} = \emptyset$ then
11:            Quit
12:        end if
13:        $t \leftarrow t + 1$
14:    end while
```