# 12. Algorithms for constrained optimization

- penalty method
- augmented Lagrangian method
- ADMM
- distributed optimization via ADMM

## Penalized formulation

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & h_i(x) = 0, \quad i = 1, \dots, p \end{array}$$

**Penalized formulation**

$$\text{minimize} \quad f(x) + \rho P(h(x))$$

- $h(x) = (h_1(x), \dots, h_p(x))$
- $P : \mathbb{R}^p \to \mathbb{R}$ is the *penalty function*
- $\rho \in \mathbb{R}$ is the *penalty parameter*
- $\rho P(x)$ penalize constraints violation, *i.e.*, has large values for infeasible points

## Penalty function

**Penalty function:** the penalty function $P$ satisfies the following conditions:

1. $P$ is continuous
2. $P(h(x)) \geq 0$ for all $x \in \mathbb{R}^n$
3. $P(h(x)) = 0$ if and only if $x$ is feasible ($h(x) = 0$)

**Quadratic penalty function**

$$P(h(x)) = \|h(x)\|^2 = \sum_{i=1}^{p} \big(h_i(x)\big)^2$$

## Quadratic penalty formulation

$$\text{minimize} \quad f(x) + \rho \|h(x)\|^2$$

- a solution of the above problem might not feasible
- for large $\rho$ we expect to have small values $\big(h_i(x)\big)^2$
  *i.e.*, an approximate solution to the original problem
- solving the above for an increasing sequence of $\rho$ is called the *penalty method*

## Quadratic penalty method

```
**given** a starting point $x^{(0)}$, $\rho_0$, and a solution tolerance $\epsilon > 0$
**repeat for** $k = 0, 1, \ldots, K$
1. set $x^{(k+1)}$ to be the (approximate) minimizer of

       minimize    $f(x) + \rho_k \|h(x)\|^2$

   using an unconstrained optimization method with initial point $x^{(k)}$
2. update $\rho_{k+1} = 2\rho_k$
```

- terminate if $\|h(x)\|^2$ is small enough
- simple and easy to implement
- but has a major issue:
  - $\rho_k$ rapidly increases with iterations
  - solving penalty problem can be very slow or simply fail

## Inequality constraints

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_j(x) = 0, \quad j = 1, \ldots, p \end{array}$$

can be handled using the penalized formulation

$$\text{minimize} \quad f(x) + \rho \|h(x)\|^2 + \rho \|g^+(x)\|^2$$

- $g^+(x) = (g_1^+(x), \ldots, g_m^+(x))$ and

$$g_i^+(x) = \max\{0, g_i(x)\} = \begin{cases} 0 & \text{if } g_i(x) \leq 0 \\ g_i(x) & \text{if } g_i(x) > 0 \end{cases}$$

- there are other choices of penalty functions
- we just consider the simple quadratic penalization function

## Outline

- penalty method
- **augmented Lagrangian method**
- ADMM
- distributed optimization via ADMM

## Original problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & h(x) = 0 \end{array} \qquad (12.1)$$

- $f : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R}^p \to \mathbb{R}$
- Lagrangian: $L(x, \lambda) = f(x) + \lambda^T h(x)$ where $\lambda \in \mathbb{R}^p$
- problem is equivalent to (for any $\lambda$)

$$\begin{array}{ll} \text{minimize} & L(x, \lambda) = f(x) + \lambda^T h(x) \\ \text{subject to} & h(x) = 0 \end{array}$$

- if $x^\star$ is a solution and a regular point, then

$$\nabla_x L(x^\star, \lambda^\star) = 0 \quad \text{for some } \lambda^\star$$

## Augmented Lagrangian formulation

the **augmented Lagrangian** (AL) is

$$\begin{aligned} L_\rho(x,\lambda) &= L(x,\lambda) + \rho\|h(x)\|^2 \\ &= f(x) + \lambda^T h(x) + \rho\|h(x)\|^2 \end{aligned}$$

- this is the Lagrangian $L(x,\lambda)$ augmented with a quadratic penalty
- $\rho$ is a positive penalty parameter
- augmented Lagrangian is the Lagrangian of the equivalent problem

$$\begin{array}{ll} \text{minimize} & f(x) + \rho\|h(x)\|^2 \\ \text{subject to} & h(x) = 0 \end{array}$$

- solution of the original problem is also a solution of the AL formulation
- AL method minimizes $L_\rho(x,\lambda)$ for a sequence of values of $\lambda$ and $\rho$

## Lagrange multiplier update

- minimizer $\tilde{x}$ of augmented Lagrangian $L_\rho(x, \lambda)$ satisfies

$$\nabla f(\tilde{x}) + Dh(\tilde{x})^T(2\rho h(\tilde{x}) + \lambda) = 0$$

- if we define $\tilde{\lambda} = \lambda + 2\rho h(\tilde{x})$ this can be written as

$$\nabla f(\tilde{x}) + Dh(\tilde{x})^T\tilde{\lambda} = 0$$

- this is the first equation in the optimality conditions

$$\nabla f(\hat{x}) + Dh(\hat{x})^T\hat{\lambda} = 0, \quad h(\hat{x}) = 0$$

- shows that if $h(\tilde{x}) = 0$, then $\tilde{x}$ is optimal
- if $h(\tilde{x})$ is not small, suggests $\tilde{\lambda}$ is a good update for $\lambda$

## Augmented Lagrangian algorithm

**given** $x^{(0)}$, $\lambda^{(0)}$, $\rho^{(0)}$, and a solution tolerance $\epsilon > 0$

**repeat for** $k = 0, 1, \ldots$

1. set $x^{(k+1)}$ to be an (approximate) solution to

$$\text{minimize} \quad f(x) + (\lambda^{(k)})^T h(x) + \rho^{(k)} \|h(x)\|^2$$

   using any unconstrained optimization method with initial point $x^{(k)}$

2. update $\lambda^{(k)}$:

$$\lambda^{(k+1)} = \lambda^{(k)} + 2\rho^{(k)} h(x^{(k+1)})$$

3. set $\rho^{(k)}$ as constant or

$$\begin{cases} \rho^{(k)} & \text{if} \quad \|h(x^{(k+1)})\| < 0.25\|h(x^{(k)})\| \\ 2\rho^{(k)} & \text{otherwise} \end{cases}$$

- $\rho$ is increased only when needed, more slowly than in penalty method
- continues until $h(x^{(k)})$ and/or $\nabla L(x^{(k)}, \lambda^{(k)})$ are sufficiently small

## Example

consider applying the augmented Lagrangian method to the problem:

$$\begin{array}{ll} \text{minimize} & e^{3x_1} + e^{-4x_2} \\ \text{subject to} & x_1^2 + x_2^2 = 1 \end{array}$$

with $x^{(0)} = (-1, 1)$ and $\lambda^{(0)} = -1$, we set a constant penalty parameter $\rho_k = 10$

the augmented Lagrangian function is

$$L_\rho(x, \lambda) = e^{3x_1} + e^{-4x_2} + \lambda\left(x_1^2 + x_2^2 - 1\right) + (\rho/2)\left(x_1^2 + x_2^2 - 1\right)^2$$

for the inner minimization problems, we employ Newton's method:

$$\hat{x} \leftarrow \hat{x} + \nabla^2 L_\rho(\hat{x}, \lambda^{(k)})^{-1} \nabla L_\rho(\hat{x}, \lambda^{(k)})$$

the gradient and Hessian are:

$$\nabla L_\rho(x,\lambda) = \begin{bmatrix} 3e^{3x_1} + 2\lambda x_1 + 2\rho x_1(x_1^2 + x_2^2 - 1) \\ -4e^{-4x_2} + 2\lambda x_2 + 2\rho x_2(x_1^2 + x_2^2 - 1) \end{bmatrix}$$

and

$$\nabla^2 L_\rho(x,\lambda) = \begin{bmatrix} 9e^{3x_1} + 2\lambda + 2\rho(x_1^2 + x_2^2 - 1) + 4\rho x_1^2 & 4\rho x_1 x_2 \\ 4\rho x_1 x_2 & 16e^{-4x_2} + 2\lambda + 2\rho(x_1^2 + x_2^2 - 1) + 4\rho x_2^2 \end{bmatrix}$$

iteration starts from $\hat{x} = x^{(k)}$ and continues until $\|\nabla L_\rho(\hat{x}, \lambda^{(k)})\| < 10^{-4}$

the value $x^{(k+1)}$ is then set to $\hat{x}$ and the Lagrange multiplier is subsequently updated:

$$\lambda^{(k+1)} = \lambda^{(k)} + \rho\big((x_1^{(k+1)})^2 + (x_2^{(k+1)})^2 - 1\big)$$

## MATLAB code implementation

```
rho=10;
x=[-1;1];
lam=-1;
%% AL gradient and Hessian
g=@(x,lam)[3*exp(3*x(1))+2*lam*x(1)+2*rho*x(1)*(x(1)^2+x(2)^2-1);
-4*exp(-4*x(2))+2*lam*x(2)+2*rho*x(2)*(x(1)^2+x(2)^2-1)];
hess=@(x,lam)[9*exp(3*x(1))+2*lam+2*rho*(x(1)^2+x(2)^2-1)+4*rho*x(1)^2 4*rho*x(1)*x(2);
4*rho*x(1)*x(2)  16*exp(-4*x(2))+2*lam+2*rho*(x(1)^2+x(2)^2-1)+4*rho*x(2)^2];
%% AL method
for i=1:100
% Newton inner minimization
while (norm(g(x,lam)) >= 1e-4)
d = -hess(x,lam)\g(x,lam);
x = x+d;
end
% Lagrange update
lam=lam+rho*(x(1)^2+x(2)^2-1);
end
```

running for $100$ iterations, we get $x^\star = (-0.7483, 0.6633)$ and $\lambda^\star = 0.2123$

## Outline

- penalty method
- augmented Lagrangian method
- **ADMM**
- distributed optimization via ADMM

## ADMM problem form

the alternating direction method of multiplier (ADMM) solves problem of form:

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

- variables $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$
- $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$, and $c \in \mathbb{R}^p$
- augmented Lagrangian

$$L_\rho(x, z, \lambda) = f(x) + g(z) + \lambda^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|^2$$

## ADMM update

$$
\begin{aligned}
x^{(k+1)} &= \operatorname*{argmin}_x L_\rho(x, z^{(k)}, \lambda^{(k)}) \\
z^{(k+1)} &= \operatorname*{argmin}_z L_\rho(x^{(k+1)}, z, \lambda^{(k)}) \\
\lambda^{(k+1)} &= \lambda^{(k)} + \rho(Ax^{(k+1)} + Bz^{(k+1)} - c)
\end{aligned}
$$

- $\rho > 0$ is the ADMM penalty parameter
- $x$ and $z$ are updated in an alternating or sequential fashion
- this is different from AL method where $x$ and $z$ are minimized jointly

$$(x, z) = \operatorname*{argmin}_{x,z} L_\rho(x, z, \lambda^{(k)})$$

- separating the minimization over $x$ and $z$ allows to decompose large problems into smaller ones when $f$ or $g$ are separable

## ADMM scaled form

define the residual $r = Ax + Bz - c$ and $u = (1/\rho)\lambda$, then

$$\begin{aligned}\lambda^T r + (\rho/2)\|r\|^2 &= (\rho/2)\|r + (1/\rho)\lambda\|^2 - (1/2\rho)\|\lambda\|^2 \\ &= (\rho/2)\|r + u\|^2 - (\rho/2)\|u\|^2\end{aligned}$$

**ADMM scaled form**

$$\begin{aligned}x^{(k+1)} &= \underset{x}{\operatorname{argmin}} \left(f(x) + (\rho/2)\|Ax + Bz^{(k)} - c + u^{(k)}\|^2\right) \\ z^{(k+1)} &= \underset{z}{\operatorname{argmin}} \left(g(z) + (\rho/2)\|Ax^{(k+1)} + Bz - c + u^{(k)}\|^2\right) \\ u^{(k+1)} &= u^{(k)} + Ax^{(k+1)} + Bz^{(k+1)} - c\end{aligned}$$

## Example: quadratic programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^TQx + r^Tx \\ \text{subject to} & Cx = d \\ & x \geq 0 \end{array}$$

- $Q$ is positive semidefinite (reduces to an LP when $Q = 0$)
- we can express this problem in the ADMM form:

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & x - z = 0 \end{array}$$

  where

$$f(x) = (1/2)x^TQx + r^Tx, \quad \mathrm{dom}\, f = \{x \mid Cx = d\}$$

  is the original objective with restricted domain

- $g$ is the indicator function of the nonnegative orthant $\mathbb{R}^n_+$

the scaled form of ADMM consists of the iterations

$$\begin{aligned}
x^{(k+1)} &= \underset{x}{\operatorname{argmin}} \left(f(x) + (\rho/2)\|x - z^{(k)} + u^{(k)}\|^2\right) \\
z^{(k+1)} &= (x^{(k+1)} + u^{(k)})_+ \\
u^{(k+1)} &= u^{(k)} + x^{(k+1)} - z^{(k+1)}
\end{aligned}$$

the $x$-update is a constrained least squares problem with optimality conditions

$$\begin{bmatrix} Q + \rho I & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x^{(k+1)} \\ \nu \end{bmatrix} + \begin{bmatrix} r - \rho(z^{(k)} - u^{(k)}) \\ -d \end{bmatrix} = 0$$

## Norm-one regularized least squares

the **lasso** problem is the $\ell_1$ regularized least squares

$$\text{minimize} \quad (1/2)\|Ax-b\|^2 + \eta\|x\|_1$$

- $\eta > 0$ is a scalar regularization parameter
- in ADMM form, the lasso problem can be written as

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & x - z = 0 \end{array}$$

  where $f(x) = (1/2)\|Ax-b\|^2$ and $g(z) = \eta\|z\|_1$

the ADMM iteration is

$$
\begin{aligned}
x^{(k+1)} &= (A^TA + \rho I)^{-1}\big(A^Tb + \rho(z^{(k)} - u^{(k)})\big) \\
z^{(k+1)} &= S_{\eta/\rho}(x^{(k+1)} + u^{(k)}) \\
u^{(k+1)} &= u^{(k)} + x^{(k+1)} - z^{(k+1)}
\end{aligned}
$$

where $S$ is the soft thresholding operator defined element-wise as

$$
\begin{aligned}
S_\kappa(a) &= \begin{cases} a - \kappa & a > \kappa \\ 0 & |a| \le \kappa \\ a + \kappa & a < -\kappa \end{cases} \\
&= (a - \kappa)_+ - (-a - \kappa)_+
\end{aligned}
$$

## Outline

- penalty method
- augmented Lagrangian method
- ADMM
- **distributed optimization via ADMM**

## Consensus problem

$$\text{minimize} \quad f(x) = \sum_{i=1}^{N} f_i(x)$$

- variable $x \in \mathbb{R}^n$
- $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$ represents the $i$th component of the objective function
- $f_i$ is available only on machine processor $i$
- goal is to solve this problem with $f_i$ handled by processor $i$ only

## Example

many classification or regression problems can be formulated as:

$$\text{minimize} \quad \sum_{j=1}^{m} l(x;\xi_j)$$

- $l(x;\xi_j)$ represent the loss function for data $\xi_j$
- for large $m$, storing the data on a single machine may not be feasible
- the problem can be solved by distributing the data across multiple machines,

$$f_i(x) = \sum_{j \in \mathcal{J}_i} l(x;\xi_j),$$

  where $\mathcal{J}_i$ is the set of training data indices at machine $i$

## Equivalent formulation

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} f_i\left(x_i\right) \\ \text{subject to} & x_i - z = 0, \quad i = 1, \ldots, N \end{array}$$

- $x_i \in \mathbb{R}^n$ handled by processing unit $i$
- $z$ is a global variable $z$ handled by central processing unit called *central server*
- the constraints ensure that all local variables are equal
- objective is now separable in the variables $x_i$
- the augmented Lagrangian is

$$L_\rho(x_1, \ldots, x_N, z, \lambda) = \sum_{i=1}^{N} \left( f_i(x_i) + (\lambda_i)^T (x_i - z) + \frac{\rho}{2} \|x_i - z\|^2 \right)$$

## ADMM updates

$$
\begin{aligned}
x_i^{(k+1)} &= \operatorname*{argmin}_{x_i}\Big(f_i(x_i) + \lambda_i^{(k)T}(x_i - z^{(k)}) + \frac{\rho}{2}\|x_i - z^{(k)}\|^2\Big) \\
z^{(k+1)} &= \frac{1}{N}\sum_{i=1}^{N}(x_i^{(k+1)} + \frac{1}{\rho}\lambda_i^{(k)}) \\
\lambda_i^{(k+1)} &= \lambda_i^{(k)} + \rho(x_i^{(k+1)} - z^{(k+1)})
\end{aligned}
$$

- the first and last steps are updated independently by each machine $i$
- central server updates $z$ after it receives all $x_i$ and then send it back to machines

## Equivalent simpler update

using overline to denote the average of a vector, we can express the $z$-update as:

$$z^{(k+1)} = \bar{x}^{(k+1)} + \frac{1}{\rho}\bar{\lambda}^{(k)}$$

by taking the average of the $\lambda$-update, we get:

$$\bar{\lambda}^{(k+1)} = \bar{\lambda}^{(k)} + \rho(\bar{x}^{(k+1)} - z^{(k+1)})$$

substituting the first equation into the subsequent one, we obtain $\bar{\lambda}^{(k+1)} = 0$ for all $k$

hence $z^{(k)} = \bar{x}^{(k)}$ and ADMM can be reformulated as:

$$\begin{aligned}
x_i^{(k+1)} &= \underset{x_i}{\operatorname{argmin}}\big(f_i(x_i) + \lambda_i^{(k)\,T}(x_i - \bar{x}^{(k)}) + \frac{\rho}{2}\|x_i - \bar{x}^{(k)}\|^2\big) \\
\lambda_i^{(k+1)} &= \lambda_i^{(k)} + \rho(x_i^{(k+1)} - \bar{x}^{(k+1)})
\end{aligned}$$

## Regularized consensus problem

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} f_i\,(x_i) + g(z) \\ \text{subject to} & x_i - z = 0, \quad i = 1, \ldots, N, \end{array}$$

- objective term $g$ is a constraint or regularization (*e.g.*, $g(z) = \|z\|_1$)
- for this case, the ADMM method is:

$$x_i^{(k+1)} = \operatorname*{argmin}_{x_i}\Big(f_i(x_i) + \lambda_i^{(k)\,T}(x_i - z^{(k)}) + \frac{\rho}{2}\|x_i - z^{(k)}\|^2\Big)$$

$$z^{(k+1)} = \operatorname*{argmin}_{z}\Big(g(z) + \sum_{i=1}^{N}(-\lambda_i^{(k)\,T} z + \frac{\rho}{2}\|x_i^{(k+1)} - z\|^2)\Big)$$

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} + \rho(x_i^{(k+1)} - z^{(k+1)})$$

- collecting linear and quadratic terms, the $z$-update can be expressed as:

$$z^{(k+1)} = \operatorname*{argmin}_z \left(g(z) + \frac{N\rho}{2}\|z - \bar{x}^{(k+1)} - \frac{1}{\rho}\bar{\lambda}^{(k)}\|^2\right)$$

- when $g$ is nonzero, we don't typically get that $\bar{\lambda}^{(k)} = 0$
- hence $\lambda_i$ terms cannot be eliminated as in the non-regularized case
- using the above update form for $z$, ADMM is:

$$\begin{aligned} x_i^{(k+1)} &= \operatorname*{argmin}_{x_i} \big(f_i(x_i) + \lambda_i^{(k)T}(x_i - z^{(k)}) + \frac{\rho}{2}\|x_i - z^{(k)}\|^2\big) \\ z^{(k+1)} &= \operatorname*{argmin}_z \big(g(z) + \frac{N\rho}{2}\|z - \bar{x}^{(k+1)} - \frac{1}{\rho}\bar{\lambda}^{(k)}\|^2\big) \\ \lambda_i^{(k+1)} &= \lambda_i^{(k)} + \rho(x_i^{(k+1)} - z^{(k+1)}) \end{aligned}$$

## Examples

- for $g(z) = \eta\|z\|_1$, the $z$-update translates into a soft threshold operation:

$$z^{(k+1)} = S_{\eta/N\rho}(\bar{x}^{(k+1)} - \frac{1}{\rho}\bar{\lambda}^{(k)})$$

- considering $g$ as the indicator function of $\mathbb{R}^n_+$, then

$$z^{(k+1)} = (\bar{x}^{(k+1)} - \frac{1}{\rho}\bar{\lambda}^{(k)})_+$$

for this problem, the scaled variant of ADMM, exhibited below, is often more streamlined and manageable compared to its unscaled counterpart:

$$\begin{aligned}
x_i^{(k+1)} &= \underset{x_i}{\operatorname{argmin}}\big(f_i(x_i) + \frac{\rho}{2}\|x_i - z^{(k)} + u_i^{(k)}\|^2\big) \\
z^{(k+1)} &= \underset{z}{\operatorname{argmin}}\big(g(z) + \frac{N\rho}{2}\|z - \bar{x}^{(k+1)} - \bar{u}^{(k)}\|^2\big) \\
u_i^{(k+1)} &= u_i^{(k)} + x_i^{(k+1)} - z^{(k+1)}
\end{aligned}$$

## References and further readings

- I. Griva and S. G. Nash and A. Sofer. *Linear and Nonlinear Optimization.* SIAM, 2009.
- E. K.P. Chong, Wu-S. Lu, and S. H. Zak. *An Introduction to Optimization: With Applications to Machine Learning*. John Wiley & Sons, 2023. (chapter 14)
- S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers.* Foundations and Trends in Machine learning, 2011.