C477: Computational Optimisation
Constrained Optimisation – Algorithms

Panos Parpas
Department of Computing
Imperial College London

www.doc.ic.ac.uk/∼pp500
p.parpas@imperial.ac.uk

Outline

1. **Projected Methods**
   - Reminder: Descent Methods
   - Projected Gradient Methods
   - Projected Gradient with Linear Constraints
2. **Lagrangian Methods (Primal/Dual Methods)**
   - Lagrangian Methods with Equality Constraints
   - Lagrangian Methods with Inequality Constraints
3. **Penalty Methods**

Additional material:

- Chapter 22 in *An Introduction to Optimization*, Chong & Zhak, Third Edition.
- Chapter 12, 13 in *Linear and Nonlinear Programming*, Luenberger & Ye, Third Edition.

Outline

1. Projected Methods

- Reminder: Descent Methods
- Projected Gradient Methods
- Projected Gradient with Linear Constraints

2. Lagrangian Methods (Primal/Dual Methods)

- Lagrangian Methods with Equality Constraints
- Lagrangian Methods with Inequality Constraints

3. Penalty Methods

Problem Formulation

$$\begin{aligned} \min \ & f(\boldsymbol{x}) \\ \text{s.t. } & \boldsymbol{h}(\boldsymbol{x}) = 0 \\ & \boldsymbol{g}(\boldsymbol{x}) \leq 0 \end{aligned}$$

- Where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^p$, $m \leq n$ and $h : \mathbb{R}^n \to \mathbb{R}^m$
- As before $h_i(\boldsymbol{x}) = 0$, $i = 1, \ldots, m$ are equality constraints
- $g_i(\boldsymbol{x}) \leq 0$, $i = 1, \ldots, p$ are inequality constraints
- The feasible region is $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{h}(\boldsymbol{x}) = 0,\ \boldsymbol{g}(\boldsymbol{x}) \leq 0\}$.

Reminder: Descent Methods – Unconstrained

1. Given a point $\boldsymbol{x}_k$.
2. Transition to the next point,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$$

3. where $\alpha_k \in \arg\min f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k)$ (if an exact step-size strategy is used)

$d_k = -\boldsymbol{\nabla} f(\boldsymbol{x}_k)$ (steepest descent)
$d_k = -\boldsymbol{\nabla}^2 f(\boldsymbol{x}_k)^{-1} \nabla f(x_k)$ (Newton Raphson)

But what if x is required to stay within some feasible set Ω?

Reminder: Descent Methods – Unconstrained

1. Given a point $\boldsymbol{x}_k$.
2. Transition to the next point,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$$

3. where $\alpha_k \in \arg\min f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k)$ (if an exact step-size strategy is used)

$d_k = -\boldsymbol{\nabla} f(\boldsymbol{x}_k)$ (steepest descent)
$d_k = -\boldsymbol{\nabla}^2 f(\boldsymbol{x}_k)^{-1} \nabla f(x_k)$ (Newton Raphson)

But what if x is required to stay within some feasible set Ω?

Projection Methods

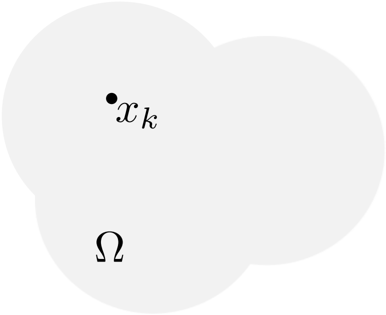
Basic idea: Project point back into feasible set.

$$\boldsymbol{x}_{k+1} = \begin{cases} \boldsymbol{x}_{k+1} & \text{if } \boldsymbol{x}_{k+1} \in \Omega \\ \Pi[\boldsymbol{x}_{k+1}] & \text{otherwise} \end{cases}$$

Projection Methods

Basic idea: Project point back into feasible set.

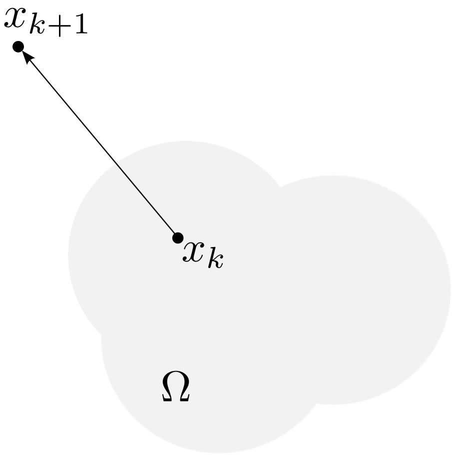
$$\boldsymbol{x}_{k+1} = \begin{cases} \boldsymbol{x}_{k+1} & \text{if } \boldsymbol{x}_{k+1} \in \Omega \\ \Pi[\boldsymbol{x}_{k+1}] & \text{otherwise} \end{cases}$$



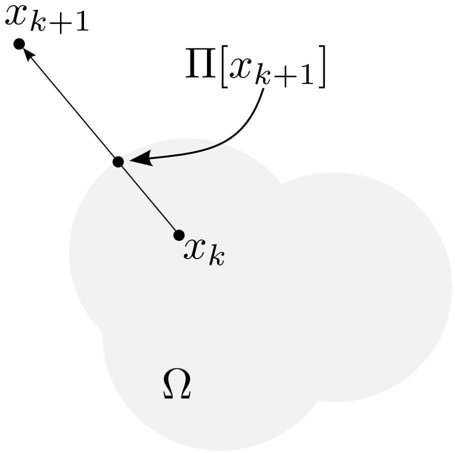
Projection Methods

Basic idea: Project point back into feasible set.

$$x_{k+1} = \begin{cases} x_{k+1} & \text{if } x_{k+1} \in \Omega \\ \Pi[x_{k+1}] & \text{otherwise} \end{cases}$$



Projection Methods



Basic idea: Project point back into feasible set.

$$\boldsymbol{x}_{k+1} = \begin{cases} \boldsymbol{x}_{k+1} & \text{if } \boldsymbol{x}_{k+1} \in \Omega \\ \Pi[\boldsymbol{x}_{k+1}] & \text{otherwise} \end{cases}$$

Projection Methods

Example: Box constraints

Suppose that the constraint set is, $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n \mid l_i \leq x_i \leq u_i\}$
Define $\boldsymbol{y} = \Pi[\boldsymbol{x}]$ as follows,

$$y_i = \begin{cases} u_i & \text{if } x_i > u_i \\ x_i & \text{if } l_i \leq x_i \leq u_i \\ l_i & \text{if } x_i < l_i \end{cases}$$

A concise way to write the above is $y_i = \min\{u_i, \max\{l_i, x_i\}\}$

The point $\Pi[\boldsymbol{x}]$ is called the projection of $\boldsymbol{x}$ into $\boldsymbol{y}$.
In general the projection operator is defined as,

$$\Pi[\boldsymbol{x}] = \arg\min_{z \in \Omega} \frac{1}{2} \|z - \boldsymbol{x}\|_2^2$$

Interpretation: $\Pi[\boldsymbol{x}]$ is the closest point in Ω to $\boldsymbol{x}$

Projection Methods

Example: Box constraints

Suppose that the constraint set is, $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n \mid l_i \leq x_i \leq u_i\}$
Define $\boldsymbol{y} = \Pi[\boldsymbol{x}]$ as follows,

$$y_i = \begin{cases} u_i & \text{if } x_i > u_i \\ x_i & \text{if } l_i \leq x_i \leq u_i \\ l_i & \text{if } x_i < l_i \end{cases}$$

A concise way to write the above is $y_i = \min\{u_i, \max\{l_i, x_i\}\}$

The point $\Pi[\boldsymbol{x}]$ is called the projection of $\boldsymbol{x}$ into $\boldsymbol{y}$.
<u>In general</u> the projection operator is defined as,

$$\Pi[\boldsymbol{x}] = \arg\min_{\boldsymbol{z} \in \Omega} \frac{1}{2} \|\boldsymbol{z} - \boldsymbol{x}\|_2^2$$

Interpretation: $\Pi[\boldsymbol{x}]$ is the closest point in Ω to $\boldsymbol{x}$

Practical Remarks on Projection Methods

$$\Pi[\boldsymbol{x}] = \arg\min_{z \in \Omega} \frac{1}{2} \|\boldsymbol{z} - \boldsymbol{x}\|_2^2$$

Projection problem can be as hard as the original problem

Suppose the original problem is:

$$\min \ \frac{1}{2} \|\boldsymbol{x}\|^2$$
$$\text{s.t.} \ \ \boldsymbol{x} \in \Omega.$$

If $\mathbf{0} \notin \Omega$, $\Pi[\mathbf{0}]$ is as difficult as the original problem.

Projection not always well defined

If Ω is convex then projection is well defined.
But for some Ω the $\arg\min$ may not be well defined.

Projected Gradient Methods

1. Given a point $\boldsymbol{x}_k$.
2. Transition to the next point,

$$\boldsymbol{x}_{k+1} = \Pi[\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)]$$

3. Where $\alpha_k \in \arg\min f(\Pi[\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)])$ (if an exact step-size strategy is used)

Example: Projected Gradient Methods

Consider the problem

$$\begin{aligned} &\min \ \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} \\ &\text{s.t.} \ \ \|\boldsymbol{x}\|_2^2 = 1 \end{aligned}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix. Suppose that a projected gradient method with fixed step size strategy is applied to this problem.

a. Derive a formula for the update equation for the algorithm (i.e. write an explicit formula for $\boldsymbol{x}_{k+1}$ as a function of $\boldsymbol{x}_k$, $\boldsymbol{Q}$, and fixed step size α). You may assume that the argument in the projection operator is never zero.

Example: Projected Gradient Methods

a. Derive a formula for the update equation for the algorithm (i.e. write an explicit formula for $\boldsymbol{x}_{k+1}$ as a function of $\boldsymbol{x}_k$, $\boldsymbol{Q}$, and fixed step size α). You may assume that the argument in the projection operator is never zero.

Example: Projected Gradient Methods

a. The projection problem is

$$\Pi^* = \min_{z} \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{z}\|_2^2$$
$$\text{s.t.} \|\boldsymbol{z}\|_2^2 = 1$$

Note that for any feasible $\boldsymbol{z}$ (i.e. $\|\boldsymbol{z}\|_2^2 = 1$).

$$\begin{aligned}
\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{z}\|_2^2 &= \frac{1}{2}\|\boldsymbol{z}\|_2^2 + \frac{1}{2}\|\boldsymbol{x}\|_2^2 - \boldsymbol{x}^\top \boldsymbol{z} \\
&\geq \frac{1}{2} + \frac{1}{2}\|\boldsymbol{x}\|_2^2 - \|\boldsymbol{x}\|_2\|\boldsymbol{z}\|_2 \text{ (from Cauchy-Schwarz inequality)} \\
&= \frac{1}{2} + \frac{1}{2}\|\boldsymbol{x}\|_2^2 - \|\boldsymbol{x}\|_2
\end{aligned}$$

Therefore $\Pi^* \geq \frac{1}{2} + \frac{1}{2}\|\boldsymbol{x}\|^2 - \|\boldsymbol{x}\|$. If we choose $\boldsymbol{z} = \boldsymbol{x}/\|\boldsymbol{x}\|$ the objective function is equal to $\frac{1}{2} + \frac{1}{2}\|\boldsymbol{x}\|^2 - \|\boldsymbol{x}\|$. Therefore $\boldsymbol{z} = \boldsymbol{x}/\|\boldsymbol{x}\|_2$ is the optimal solution of the projected problem. We now have,

$$\boldsymbol{x}_{k+1} = \beta_k(\boldsymbol{x}_k - \alpha \boldsymbol{Q}\boldsymbol{x}_k) = \beta_k(\boldsymbol{I} - \alpha \boldsymbol{Q})\boldsymbol{x}_k$$

where $\beta_k = 1/\|(\boldsymbol{I} - \alpha \boldsymbol{Q})\boldsymbol{x}_k\|$

Projected Gradient with Linear Constraints

$$\min \; f(\boldsymbol{x})$$
$$\text{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

Where

- $f : \mathbb{R}^n \to \mathbb{R}$
- $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $m < n$ and rank$(\boldsymbol{A}) = m$, $\boldsymbol{b} \in \mathbb{R}^m$.

Derivation of the projection matrix

Suppose that

- $\boldsymbol{x}_k$ is feasible i.e. $\boldsymbol{A}\boldsymbol{x}_k = \boldsymbol{b}$
- $\boldsymbol{d}_k$ is a descent but not a feasible direction.

Direction will be feasible if,

$$\boldsymbol{A}\boldsymbol{x}_{x+1} = \boldsymbol{A}(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) = \boldsymbol{b}$$
$$\boldsymbol{A}\boldsymbol{x}_k + \alpha_k A \boldsymbol{d}_k = \boldsymbol{b}$$

So if,

$$\boldsymbol{A}\boldsymbol{d}_k = \boldsymbol{0}$$

then $\boldsymbol{A}\boldsymbol{x}_{x+1} = \boldsymbol{b}$.

Derivation of the projection matrix

The projection problem is,

$$\begin{aligned} \min\ & \frac{1}{2}\|\boldsymbol{d}-\boldsymbol{d}_k\|_2^2 \\ \text{s.t. } & \boldsymbol{A}\boldsymbol{d} = 0 \end{aligned}$$

The projection operator is the matrix $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A}$.

Derivation of the projection matrix

The projection problem is,

$$\min \frac{1}{2}\|\boldsymbol{d}-\boldsymbol{d}_k\|_2^2$$
$$\text{s.t. } \boldsymbol{A}\boldsymbol{d}=0$$

The projection operator is the matrix $\boldsymbol{P}=\boldsymbol{I}-\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A}$.
To see this note that the first order conditions for this problem are,

$$\boldsymbol{d}-\boldsymbol{d}_k+\boldsymbol{A}^\top\boldsymbol{\lambda}=0$$

Therefore $\boldsymbol{\lambda}=(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A}\boldsymbol{d}_k$. Substituting this relationship back into the first order condition we obtain that the optimum solution is,

$$\boldsymbol{d}=(\boldsymbol{I}-\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A})\boldsymbol{d}_k$$

Properties of the projection matrix

Given a set of linear constraints,

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},$$

with $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $m < n$ and rank$(\boldsymbol{A}) = m$, $\boldsymbol{b} \in \mathbb{R}^m$. Then,

$$\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A}$$

is called the projection matrix.
Exercise: Show that the following statements are true for the projection matrix defined above.

1. $\boldsymbol{P}^\top\boldsymbol{P} = \boldsymbol{P}$
2. $\boldsymbol{P}^\top = \boldsymbol{P}$

Properties of the projection matrix

1. $\boldsymbol{P}^\top \boldsymbol{P} = \boldsymbol{P}$

$$\boldsymbol{P}^\top = (\boldsymbol{I} - \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A})^\top = \boldsymbol{I} - \boldsymbol{A}^\top((\boldsymbol{A}\boldsymbol{A}^\top)^{-1})^\top \boldsymbol{A}$$

(where we used the property that for two matrices $\boldsymbol{B}$ $\boldsymbol{C}$ the following holds: $(\boldsymbol{B}\boldsymbol{C})^\top = \boldsymbol{C}^\top \boldsymbol{B}^\top$). The result follows by direct calculation,

$$\begin{aligned}
\boldsymbol{P}^\top \boldsymbol{P} &= (\boldsymbol{I} - \boldsymbol{A}^\top((\boldsymbol{A}\boldsymbol{A}^\top)^{-1})^\top \boldsymbol{A})(\boldsymbol{I} - \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A}) \\
&= \boldsymbol{P} - \boldsymbol{A}^\top((\boldsymbol{A}\boldsymbol{A}^\top)^{-1})^\top \boldsymbol{A} + \boldsymbol{A}^\top((\boldsymbol{A}\boldsymbol{A}^\top)^{-1})^\top \boldsymbol{A}\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A} \\
&= \boldsymbol{P} - \boldsymbol{A}^\top((\boldsymbol{A}\boldsymbol{A}^\top)^{-1})^\top \boldsymbol{A} + \boldsymbol{A}^\top((\boldsymbol{A}\boldsymbol{A}^\top)^{-1})^\top \boldsymbol{A} \\
&= \boldsymbol{P}
\end{aligned}$$

Properties of the projection matrix

2. $\boldsymbol{P}^\top = \boldsymbol{P}$. Note that,

$$\begin{aligned}
\boldsymbol{P}^\top(\boldsymbol{I}-\boldsymbol{P}) &= (\boldsymbol{I}-\boldsymbol{A}^\top((\boldsymbol{A}\boldsymbol{A}^\top)^{-1})^\top\boldsymbol{A})(\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A}) \\
&= \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A} - \boldsymbol{A}^\top((\boldsymbol{A}\boldsymbol{A}^\top)^{-1})^\top\boldsymbol{A}\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A} \\
&= \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A} - \boldsymbol{A}^\top((\boldsymbol{A}\boldsymbol{A}^\top)^{-1})^\top\boldsymbol{A} \\
&= \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A} - \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A} = 0
\end{aligned}$$

Therefore using the property of the projection matrix in the previous slide we obtain $\boldsymbol{P}^\top = \boldsymbol{P}^\top\boldsymbol{P} = \boldsymbol{P}$
(In the derivation of 2. we used the property that for an invertible matrix $\boldsymbol{B}$ then $(\boldsymbol{B}^\top)^{-1} = (\boldsymbol{B}^{-1})^\top$).

Projected Gradient with Linear Constraints

General Iterative algorithm:

$$\boldsymbol{x}_{k+1} = \Pi[\boldsymbol{x}_k - \alpha_k \boldsymbol{\nabla} f(\boldsymbol{x}_k)]$$

If projection is on the set $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$ then,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{P} \boldsymbol{\nabla} f(\boldsymbol{x}_k)$$

where $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{A}^\top (\boldsymbol{A}\boldsymbol{A}^\top)^{-\mathbf{1}} \boldsymbol{A}$, and $\boldsymbol{x}_0$ was assumed to be in Ω.

Projected Gradient with Linear Constraints

Theorem (Feasibility)

In the projected gradient algorithm with linear constraints, if x_0 is feasible, then $Ax_k = b$, $k \geq 0$.

Proof.

Proof is by induction. Assume that $\boldsymbol{Ax_k = b}$ we show that $\boldsymbol{Ax_{k+1} = b}$. First note that,

$$\boldsymbol{AP\nabla f(x_k) = A(I - A^\top(AA^\top)^{-1}A)\nabla f(x_k) = (A - A)\nabla f(x_k)} = 0.$$

Therefore,

$$\boldsymbol{Ax_{k+1} = A(x_k - \alpha_k P\nabla f(x_k))) = Ax_k - \alpha_k AP\nabla f(x_k) = b},$$

as required. □

Projected Gradient and Descent Property

$$x_{k+1} = x_k - \alpha_k P \nabla f(x_k)$$

where $P = I - A^\top (AA^\top)^{-1} A$, and x_0 was assumed to be in Ω.
So far we know that if x_0 is feasible then all the iterates x_k will also be feasible.
But is this a descent algorithm?

Projected Gradient and Descent Property

Theorem

If $\{\boldsymbol{x}_k\}$ is the sequence of points generated by the projected gradient algorithm (with the exact step-size strategy). If $\boldsymbol{P}\nabla f(\boldsymbol{x_k}) \neq \mathbf{0}$ then $f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k)$.

Proof.

Projected Gradient and Descent Property

Theorem

If $\{\boldsymbol{x}_k\}$ is the sequence of points generated by the projected gradient algorithm (with the exact step-size strategy). If $\boldsymbol{P}\boldsymbol{\nabla} f(\boldsymbol{x_k}) \neq \boldsymbol{0}$ then $f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k)$.

Proof.

We first recall that,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{P}\boldsymbol{\nabla} f(\boldsymbol{x}_k),$$

Let $\phi_k(\alpha) = f(\boldsymbol{x}_k - \alpha \boldsymbol{P}\boldsymbol{\nabla} f(\boldsymbol{x}_k))$ and the step-size is chosen such that $\alpha_k \in \arg\min_{\alpha \geq 0} \phi_k(\alpha)$. We therefore have,

$$\phi_k(\alpha_k) \leq \phi_k(\alpha), \quad \forall \alpha \geq 0.$$

Using the chain rule we obtain,

$$\frac{d\phi_k}{d\alpha}(0) = -\boldsymbol{\nabla} f(\boldsymbol{x}_k)^\top \boldsymbol{P} \nabla \boldsymbol{f}(\boldsymbol{x}_k)$$

But since $\boldsymbol{P} = \boldsymbol{P}^\top \boldsymbol{P}$ we get,

$$\frac{d\phi_k}{d\alpha}(0) = -\boldsymbol{\nabla f}(\boldsymbol{x}_k)^\top \boldsymbol{P}^\top \boldsymbol{P}\boldsymbol{\nabla} f(\boldsymbol{x}_k) = -\|\boldsymbol{P}\boldsymbol{\nabla} f(\boldsymbol{x}_k)\|^2 < 0,$$

since $\boldsymbol{P}\boldsymbol{\nabla} f(\boldsymbol{x}_k) \neq 0$ by assumption. Thus there exists $\bar{\alpha} > 0$ such that $\phi_k(0) > \phi_k(\alpha)$ for all $\alpha \in (0, \bar{\alpha}]$. $\square$

Projected Gradient and Convergence

The convergence of the algorithm is based on the previous Theorem and the following result.

Theorem

Let $\boldsymbol{x}^*$ be a feasible point then $\boldsymbol{P}\boldsymbol{\nabla} f(\boldsymbol{x}^*) = 0$ if and only if $\boldsymbol{x}^*$ satisfies the Lagrange condition.

Proof.

Projected Gradient and Convergence

The convergence of the algorithm is based on the previous Theorem and the following result.

Theorem

Let $\boldsymbol{x}^*$ be a feasible point then $\boldsymbol{P}\boldsymbol{\nabla} f(\boldsymbol{x}^*) = 0$ if and only if $\boldsymbol{x}^*$ satisfies the Lagrange condition.

Proof.

We need to show that $\boldsymbol{P}\boldsymbol{\nabla} f(\boldsymbol{x}^*) = \mathbf{0}$ if and only if $\boldsymbol{\nabla} f(\boldsymbol{x}^*) = \boldsymbol{A}^\top \boldsymbol{\lambda}^*$ for some $\boldsymbol{\lambda}^* \in \mathbb{R}^m$.
If $\boldsymbol{P}\nabla f(\boldsymbol{x}^*) = 0$ then let $\boldsymbol{\lambda}^* = (\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A}\boldsymbol{\nabla} f(\boldsymbol{x}^*)$ we then have,

$$\mathbf{0} = \boldsymbol{P}\boldsymbol{\nabla} f(\boldsymbol{x}^*) = \boldsymbol{\nabla} f(\boldsymbol{x}*) - \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A}\boldsymbol{\nabla} f(\boldsymbol{x}^*) = \boldsymbol{\nabla} f(\boldsymbol{x}*) - \boldsymbol{A}^\top\boldsymbol{\lambda}^*$$

therefore $\boldsymbol{\nabla} f(\boldsymbol{x}^*) = \boldsymbol{A}^\top\boldsymbol{\lambda}^*$.
For the other direction, suppose that $\boldsymbol{\nabla} f(\boldsymbol{x}^*) = \boldsymbol{A}^\top\boldsymbol{\lambda}^*$ then,

$$\boldsymbol{P}\boldsymbol{\nabla} f(\boldsymbol{x}^*) = \boldsymbol{P}\boldsymbol{A}^\top\boldsymbol{\lambda}^* = (\boldsymbol{I} - \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{A})\boldsymbol{A}^\top\boldsymbol{\lambda}^* = \boldsymbol{A}^\top\boldsymbol{\lambda}^* - \boldsymbol{A}^\top\boldsymbol{\lambda}^* = \mathbf{0}.$$

$\square$

Summary Projected Gradient Methods

1. Fast and easy algorithm to implement.
2. All the algorithms (including Newton method) can be used in conjunction with projection.
3. Only possible when Ω is “simple" e.g. a box constraints or linear constraints

Outline

Lagrangian Algorithms

Equality Constrained Problem

$$\min \; f(\boldsymbol{x})$$
$$\text{s.t. } \boldsymbol{h}(\boldsymbol{x}) = 0$$

Where $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$ and $m \leq n$.

First Order Conditions

$$\begin{aligned}
&\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \boldsymbol{h}(\boldsymbol{x}) && \text{(Lagrangian)} \\
&\nabla \boldsymbol{f}(\boldsymbol{x}) + \nabla \boldsymbol{h}(\boldsymbol{x}) \boldsymbol{\lambda} = \mathbf{0} && \text{(First Order Conditions)} \\
&\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}
\end{aligned}$$

Lagrangian Algorithms

Lagrangian Algorithm

$$
\begin{aligned}
x_{k+1} &= x_k - \alpha_k(\nabla f(x_k) + \nabla h(x_k)\lambda_k) \\
\lambda_{k+1} &= \lambda_k + \beta_k h(x_k)
\end{aligned}
$$

- **Update equation for x** : same as applying the steepest descent method for minimising $L(x, \lambda)$ over x with no constraints
- **Update equation for λ** : same as applying the steepest descent method for maximising $L(x, \lambda)$ over λ
- Only gradients are used so method is called *first order method.*

The General Case

Equality Constrained Problem

$$
\begin{aligned}
\min\ & f(\boldsymbol{x}) \\
\text{s.t. } & \boldsymbol{h}(\boldsymbol{x}) = \mathbf{0} \\
& \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}
\end{aligned}
$$

First Order Conditions

$$
\begin{aligned}
&\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{\mu}^\top \boldsymbol{g}(\boldsymbol{x}) && \text{(Lagrangian)} \\
&\nabla \boldsymbol{f}(\boldsymbol{x}) + \nabla \boldsymbol{h}(\boldsymbol{x})\boldsymbol{\lambda} + \nabla \boldsymbol{g}(\boldsymbol{x})\boldsymbol{\mu} = \mathbf{0} && \text{(First Order Conditions)} \\
&\mu_i g_i(\boldsymbol{x}) = 0 \\
&\boldsymbol{\mu} \geq \mathbf{0} \\
&\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0} \\
&\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}
\end{aligned}
$$

Lagrangian Algorithm – Inequality constraints

Lagrangian Algorithm

$$
\begin{aligned}
\boldsymbol{x_{k+1}} &= \boldsymbol{x_k} - \alpha_k(\nabla \boldsymbol{f}(\boldsymbol{x_k}) + \nabla \boldsymbol{h}(\boldsymbol{x_k})\boldsymbol{\lambda_k} + \nabla \boldsymbol{g}(\boldsymbol{x_k})\boldsymbol{\mu_k}) \\
\boldsymbol{\lambda_{k+1}} &= \boldsymbol{\lambda}_k + \beta_k \boldsymbol{h}(\boldsymbol{x}_k) \\
\boldsymbol{\mu_{k+1}} &= P_+[\boldsymbol{\mu}_k + \gamma_k \boldsymbol{g}(\boldsymbol{x}_k)]
\end{aligned}
$$

- P_+ is the projection to the positive part of $\mathbb{R}^p$ applied component wise.
- **Update equation for** $\boldsymbol{x}$: same as applying the steepest descent method for minimising $\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ with no constraints
- **Update equation for** $\boldsymbol{\lambda}$: same as applying the steepest descent method for maximising $\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ over $\boldsymbol{\lambda}$
- **Update equation for** $\boldsymbol{\mu}$: same as applying the **projected** steepest descent method for maximising $\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ over $\boldsymbol{\mu}$
- Only gradients are used so method is called *first order method.*

Lagrangian Algorithm Theory

- Can be shown that method converges to a KKT point.
- Complementarity condition also satisfied.
- Rate of convergence is linear (since it is based on steepest descent method)
- No guarantees it will converge to the global minimum or that second order conditions will be satisfied.

Outline

1. **Projected Methods**
   - Reminder: Descent Methods
   - Projected Gradient Methods
   - Projected Gradient with Linear Constraints

2. **Lagrangian Methods (Primal/Dual Methods)**
   - Lagrangian Methods with Equality Constraints
   - Lagrangian Methods with Inequality Constraints

3. **Penalty Methods**

Penalty Methods

Basic Idea: Convert the constrained optimisation problem to an unconstrained problem

Original constrained problem:

$$\min_{x} f(\boldsymbol{x})$$

$$\boldsymbol{x} \in \Omega$$

Modified unconstrained problem:

$$\min_{x} f(\boldsymbol{x}) + \gamma P(\boldsymbol{x})$$

Where:

- γ is a positive scalar called the **penalty parameter.**
- $P : \mathbb{R}^n \to \mathbb{R}$ is called the penalty function. The aim of this function is to penalise points outside Ω

Penalty Methods

Basic Idea: Convert the constrained optimisation problem to an unconstrained problem
Original constrained problem:

$$\min_x f(\boldsymbol{x})$$
$$\boldsymbol{x} \in \Omega$$

Modified unconstrained problem:

$$\min_x f(\boldsymbol{x}) + \gamma P(\boldsymbol{x})$$

Where:

- γ is a positive scalar called the **penalty parameter.**
- $P : \mathbb{R}^n \to \mathbb{R}$ is called the penalty function. The aim of this function is to penalise points outside Ω

Penalty Methods

Basic Idea: Convert the constrained optimisation problem to an unconstrained problem

Original constrained problem:

$$\min_{x} f(\boldsymbol{x})$$

$$\boldsymbol{x} \in \Omega$$

Modified unconstrained problem:

$$\min_{x} f(\boldsymbol{x}) + \gamma P(\boldsymbol{x})$$

Where:

- γ is a positive scalar called the **penalty parameter.**
- $P : \mathbb{R}^n \rightarrow \mathbb{R}$ is called the penalty function. The aim of this function is to penalise points outside Ω

Penalty Methods

Original constrained problem:

$$\begin{aligned}&\min_x f(\boldsymbol{x})\\ &\text{s.t. } g_i(\boldsymbol{x}) \leq 0 \; i = 1, \ldots, p.\end{aligned}$$

Penalty Function:

$$P(\boldsymbol{x}) = \sum_{i=1}^{p} g_i^+(\boldsymbol{x})$$

where

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

The penalty function defined above is also called the absolute value penalty function since it is equal to $\sum_{i=1}^{p} |g_i(\boldsymbol{x})|$

Example

Suppose the feasible region is given by,

$$g_1(x) = x - 2 \leq 0.$$
$$g_2(x) = -(x+1)^3 \leq 0$$

The penalty function is defined as follows,

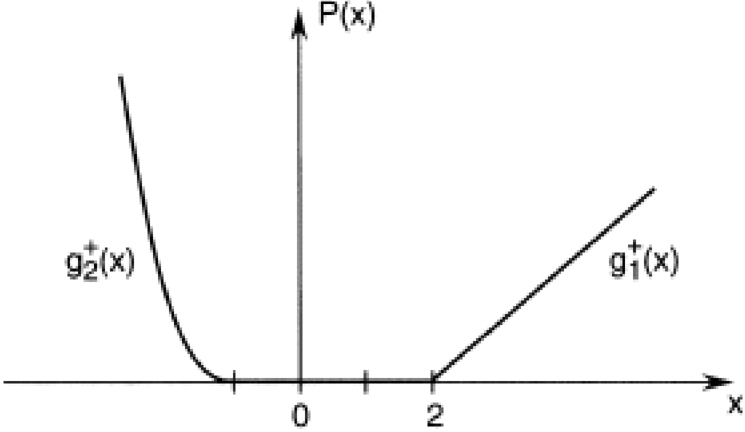
$$g_1^+(x) = \max\{0, g_1(x)\} = \begin{cases} 0 & \text{if } x \leq 2 \\ x - 2 & \text{otherwise} \end{cases}$$

$$g_2^+(x) = \max\{0, g_2(x)\} = \begin{cases} 0 & \text{if } x \geq -1 \\ -(x+1)^3 & \text{otherwise} \end{cases}$$

So,

$$P(x) = g_1^+(x) + g_2^+(x) = \begin{cases} x - 2 & \text{if } x > 2 \\ 0 & \text{if } -1 \leq x \leq 2 \\ -(x+1)^3 & \text{if } x < -1 \end{cases}$$

Example



Penalty Methods

The absolute value penalty function may not be differentiable everywhere (e.g. last example $P(x)$ is not differentiable at $x = 2$). Some differentiable & widely used alternatives are:

- The *Courant-Beltrami penalty function*

$$P(\boldsymbol{x}) = \sum_{i=1}^{p} (g_i^+(\boldsymbol{x}))^2$$

- *Logarithmic Barrier function*

$$P(\boldsymbol{x}) = -\sum_{i=1}^{p} \log(-g_i(\boldsymbol{x}))$$

- *Inverse Barrier function*

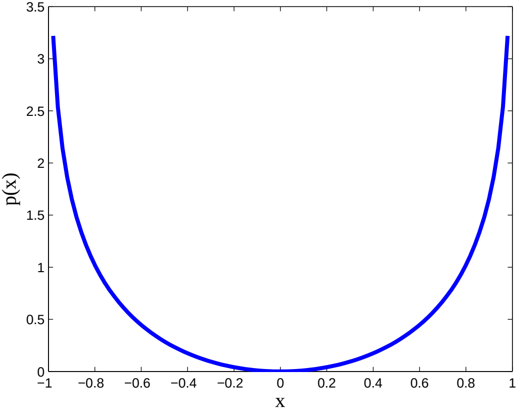
$$P(\boldsymbol{x}) = -\sum_{i=1}^{p} \frac{1}{g_i(\boldsymbol{x})}$$

For the two barrier functions the convention is to let the penalty parameter γ go to zero

Penalty Methods

The Logarithmic Barrier Function associated with the constraint,

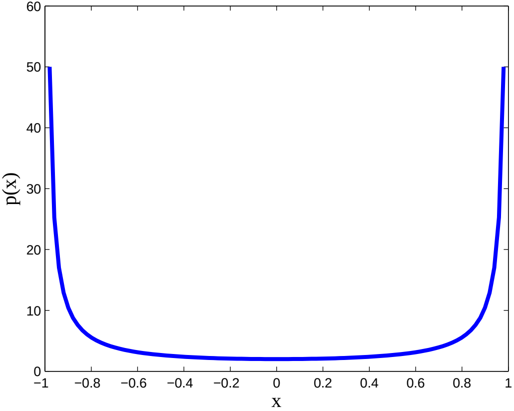
$$-1 \leq x \leq 1$$



Penalty Methods

The Inverse Barrier Function associated with the constraint,

$$-1 \leq x \leq 1$$



Penalty Methods Summary

- Penalty methods convert the problem into an unconstrained problem and use unconstrained algorithms (e.g. Steepest Descent, Newton Method etc..)
- Logarithmic Barrier Methods are very popular for solving convex optimisation problems (these are polynomial time algorithms)
- Convergence results exists that guarantee that these methods will converge to a KKT point as $\gamma \to \infty$ (or zero in the case of barrier penalty functions)
- Because of the penalty parameter problem becomes ill conditioned