IB Optimization

Ishan Nath, Easter 2022

Based on Lectures by Prof. Richard Weber

June 21, 2022

Contents

1 Convexity

1.1 Generic optimization problem

All the problems we are looking to solve are of the form

$$\text{minimise } f(x) \text{ such that } g(x) = b, \text{ for all } x \in X.$$

Note maximising f is equivalent to minimising $-f$. Here f is our **objective function**, and $x \in \mathbb{R}^n$ is the **decision variable(s)**. Our subset $X \subseteq \mathbb{R}^n$ is the **regional constraints** (for example $x \geq 0$), and $g : \mathbb{R}^n \to \mathbb{R}^m$ is the **functional constraints**. We can also define the **feasible set**

$$X(b) = \{x \in X \mid g(x) = b\}.$$

The problem is **feasible** if $X(b)$ is non-empty, and **bounded** if f is bounded on $X(b)$. x^* is an **optimal solution** if it minimises f over $X(b)$.

A condition of the form $g(x) \leq b$ can be turned into an equality constraint by using a **slack variable** z, and have functional constraint

$$g(x) + z = b, \quad z \geq 0.$$

1.2 Gradient Descent

Consider a problem whose only constraint is minimising $f(x)$ over $x \in \mathbb{R}^n$. An intuitive idea is to start at some point x_0 and make a sequence of small steps, each downhill. To see which direction to take these steps, we Taylor expand:

$$f(x_0 + tu) = f(x_0) + t\nabla f(x_0)^T u + O(t^2).$$

If we say u is a unit vector, i.e. $u^T u = 1$, then $\nabla f(x_0)^T u$ is minimized by

$$u = -\frac{\nabla f(x_0)}{\|\nabla f(x_0)\|}.$$

From this, we can define the gradient descent algorithm:

1) Start with an initial x_0.
2) Pick a step size t.
3) For $k = 0, 1, \ldots$, define $x_{k+1} = x_k - t\nabla f(x_k)$.

This will work if t is small: for example, if $t > 1$, then consider $f(x) = x^2$. We have $x_{k+1} = x_k - 2tx_k = (1 - 2t)x_k$, which grows in size.

1.3 Convexity

Definition 1.1. A set $S \in \mathbb{R}^n$ is **convex** if, for all $x, y \in S$ and $0 \leq \lambda \leq 1$, the point $\lambda x + (1-\lambda)y \in S$.

Definition 1.2. $f : S \to \mathbb{R}$ is a convex function if

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y),$$

for all $x, y \in S$.

Definition 1.3. f is strictly convex if it is convex and equality never holds, and is strongly convex if there exists $\alpha > 0$ such that

$$f(x) - \frac{\alpha}{2}\|x\|^2 \text{ is convex.}$$

Moreover f is concave if $-f$ is convex.

For example, $f(x) = x^2$ is convex, and $f(x) = \log x$ is concave.

An equivalent definition is given by the **epigraph**:

$$\text{epi}(f) = \{(x, t) \mid f(x) \leq t\}.$$

Then f is convex if and only if $\text{epi}(f)$ is a convex set.

Theorem 1.1 (Supporting hyperplane theorem)**.** *Let $f : S \to \mathbb{R}$, with S a convex set. Then f is convex if and only if for every $x \in S$, there exists $\lambda(x) \in \mathbb{R}^n$ such that for all y,*

$$f(y) \geq f(x) + \lambda(x)^T(y - x).$$

If f is differentiable, then $\lambda(x) = \nabla f(x)$.

This theorem essentially says a function is convex if and only if at every point, there exists a plane tangent to the curve at that point, such that the curve lies on one side of the plane.

Proof: First, suppose a supporting hyperplane exists for every point. Consider $y, z \in S$, and an interior point $x = py + qz$, where $p + q = 1$ and $0 < p < 1$. Then,

$$pf(y) + qf(z) \geq p[f(x) + \lambda(x)^T(y - x)] + q[f(x) + \lambda(x)^T(z - x)] = f(x).$$

We will prove a supporting hyperplane exists for differentiable f. The condition that f is convex is equivalent to

$$\frac{f(x+p(y-x))-f(x)}{p} \leq f(y)-f(x).$$

Letting $p \to 0$, we get that

$$\nabla f(x)^T(y-x) \leq f(y)-f(x).$$

Corollary 1.1. *If f is convex and differentiable, x^* is feasible, and $\nabla f(x^*)=0$, then x^* is the global minimum of f.*

This follows from letting $x = x^*$ in the supporting hyperplane theorem.

Theorem 1.2. *Let $f : S \to \mathbb{R}$, with S a convex set. Then,*

(a) *f is convex $\implies$ every local minimum is a global minimum.*

(b) *f is strictly convex $\implies$ the global minimum is unique.*

(c) *f is continuous and strongly convex, and $S \subseteq \mathbb{R}^n$ is closed $\implies$ there exists an optimal solution to the problem of minimising f over S.*

Proof:

(a) Let x^* be a local minimum, and y be any other point. Then if $z = (1-\lambda)x^* + \lambda y$, we have $f(x^*) \leq f(z)$ for sufficiently small λ. However, we know by convexity that

$$f(z) \leq (1-\lambda)f(x^*) + \lambda f(y),$$

$$\implies f(x^*) \leq f(y).$$

(b) Suppose x, y are both global minima. Then

$$f\left(\frac{1}{2}x + \frac{1}{2}y\right) < \frac{1}{2}f(x) + \frac{1}{2}f(y).$$

This contradicts the fact x and y are both minima.

(c) Since f is strongly convex, $f(x) - \alpha\|x\|^2/2$ is convex. Using the supporting hyperplane theorem,

$$f(x) - \alpha\|x\|^2/2 \geq f(0) + \lambda(0)^T x,$$

where $\lambda = \lambda(0)$. By Cauchy-Schwartz, $\lambda^T x \geq -\|\lambda\|\|x\|$, so if $\|x\| > R = 2\|\lambda\|/\alpha$,

$$f(x) \geq f(0) - \|\lambda\|\|x\| + \alpha\|x\|^2/2 > f(0).$$

Thus we may restrict to a bounded region, but a continuous function attains a minimum on a compact set.

Theorem 1.3 (Lower bound on the gradient)**.** *Suppose $f : S \to \mathbb{R}$ is differentiable and strongly convex with constant $\alpha > 0$. Then for any $x, y \in S$,*

$$\|\nabla f(x)\|^2 \geq 2\alpha(f(x) - f(y)).$$

In particular, if $y = x^*$ is the minimizer of f, then

$$f(x^*) \geq f(x) - \frac{1}{2\alpha}\|\nabla f(x)\|^2.$$

Proof: Again, we will apply the SHT on $f(x) - \alpha\|x\|^2/2$. Then we get that

$$\begin{aligned} & f(y) - \alpha\|y\|^2 \geq f(x) - \alpha\|x\|^2 + (\nabla f(x) - \alpha x)^T(y - x) \\ \iff\ & f(y) - f(x) \geq \nabla f(x)^T(y - x) + \alpha\|y - x\|^2/2 \\ & = \frac{\alpha}{2}\left\|(y - x) + \frac{1}{\alpha}\nabla f(x)\right\|^2 - \frac{1}{2\alpha}\|\nabla f(x)\|^2 \\ & \geq -\frac{1}{2\alpha}\|\nabla f(x)\|^2. \end{aligned}$$

2 Gradient Descent and Newton's method

2.1 Second-order conditions

Definition 2.1. A $n \times n$ symmetric matrix A is said to be **non-negative definite** if $x^T Ax \geq 0$ for all x.

The **Hessian**, $\nabla^2 f(x)$, is an $n \times n$ matrix such that

$$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

Theorem 2.1. *If $\nabla^2 f(x)$ is non-negative definite for all x then f is convex.*

Proof: For $x, y \in S$, by multivariate Taylor's theorem, we can write

$$f(y) = f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}(y - x)^T \nabla^2(\xi)(y - x),$$

for some $\xi = px + (1 - p)y$, $p \in (0, 1)$. Since $\nabla^2 f$ is non-negative definite, we get that

$$f(y) \geq f(x) + \nabla f(x)^T (y - x).$$

We are then done by SHT.

In fact, the converse of theorem 2.1 also holds.

Recall f is strongly convex if there exists $\alpha > 0$ such that $f(x) - \alpha\|x\|^2/2$ is convex. If f is differentiable, this is equivalent to

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\alpha}{2}\|y - x\|^2,$$

and if f is twice differentiable, this is equivalent to $\nabla^2 f(x) - \alpha I$ is non-negative definite. We write this as $\nabla^2 f(x) - \alpha I \succeq 0$, or

$$\alpha I \preceq \nabla^2 f(x).$$

Definition 2.2. For any symmetric matrices A and B, we write $A \preceq B$ to mean $B - A$ is non-negative definite.

2.2 Convergence of gradient descent

Recall that in gradient descent, we set

$$x_{k+1} = x_k - t\nabla f(x_k).$$

Here t is called the **learning rate** in machine learning.

Definition 2.3. f is said to be β-smooth if

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\|.$$

Theorem 2.2. *If f is β-smooth, then*

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{\beta}{2}\|x - y\|^2.$$

Proof:

$$\begin{aligned} f(y) - f(x) - \nabla f(x)^T (y - x) &= \int_0^1 [\nabla f(x + t(y - x)) - \nabla f(x)]^T (y - x)\, dt \\ &\leq \int_0^1 \|\nabla f(x + t(y - x)) - \nabla f(x)\| \cdot \|y - x\|\, dt \\ &\leq \beta \|x - y\|^2 \int_0^1 t\, dt = \frac{\beta}{2}\|x - y\|^2. \end{aligned}$$

In theorem 2.2, the right hand side is minimized when

$$y = x - \frac{1}{\beta}\nabla f(x).$$

This suggest $t = 1/\beta$ is a good learning rate.

Theorem 2.3. *Suppose f is twice differentiable and there are positive constants α and β such that*

$$\alpha I \preceq \nabla^2 f(x) \preceq \beta I$$

for all x. Then applying the gradient descent algorithm with $t = 1/\beta$, we have

$$f(x_k) - f(x^*) \leq \left(1 - \frac{\alpha}{\beta}\right)^k (f(x_0) - f(x^*)).$$

Proof: We take $x_{k+1} = x_k - \nabla f(x_k)/\beta$. From theorem 1.3, we get that

$$\|\nabla f(x_k)\|^2 \geq 2\alpha(f(x_k) - f(x^*)).$$

Using Taylor's theorem, we get that

$$\begin{aligned} f(x_{k+1}) - f(x_k) &= \nabla f(x_k)^T(x_{k+1} - x_k) + \frac{1}{2}(x_{k+1} - x_k)^T \nabla^2 f(\xi)(x_{k+1} - x_k) \\ &\leq \nabla f(x_k)^T(x_{k+1} - x_k) + \frac{\beta}{2}\|x_{k+1} - x_k\|^2 \\ &= -\frac{1}{2\beta}\|\nabla f(x_k)\|^2 \leq -\frac{\alpha}{\beta}(f(x_k) - f(x^*)), \\ \implies f(x_{k+1}) - f(x^*) &\leq \left(1 - \frac{\alpha}{\beta}\right)(f(x_k) - f(x^*)). \end{aligned}$$

The result then follows by induction.

In fact, the theorem holds without the assumption of a second derivative. The ratio β/α is called the **condition number**.

Example 2.1. Let $f(x) = (x_1^2 + 100x_2^2)/2$. Then

$$\nabla^2 f(x) = \begin{pmatrix} 1 & 0 \\ 0 & 100 \end{pmatrix}.$$

Therefore $\alpha = 1$, $\beta = 100$, so the condition number is 100. Applying gradient descent,

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \frac{1}{100}\begin{pmatrix} x_1 \\ 100x_2 \end{pmatrix} = \begin{pmatrix} 0.99x_1 \\ 0 \end{pmatrix}.$$

Here, (x_n) goes to 0 very slowly. We can change this by setting $y = 10x_2$, then letting $f(x, y) = (x_1^2 + y^2)/2$, gradient descent gets to the minimum in one step.

2.3 Newton's method

From Taylor expansion, we have

$$f(x) \approx f(x_0) + \nabla f(x_0)^T(x - x_0) + \frac{1}{2}(x - x_0)^T \nabla^2 f(x_0)(x - x_0).$$

The right side is minimised at

$$x = x_0 - (\nabla^2 f(x_0))^{-1} \nabla f(x_0).$$

We can define a new algorithm, **Newton's method**, as follows:

1) Start with a guess x_0.

2) For $k = 0, 1, \ldots$, define $x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k)$.

Definition 2.4. Suppose A is $n \times n$. Let $\|A\|$ be the smallest a such that $\|Az\| \leq a\|z\|$ for all $z \in \mathbb{R}^n$. If A is non-negative definite, then $\|A\|$ is the largest eigenvalue of A.

Theorem 2.4. *Suppose f is twice differentiable and there are constant $\alpha, L > 0$ such that*

$$\alpha I \preceq \nabla^2 f(x),$$
$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\|,$$

for all $x, y \in \mathbb{R}^n$. Then with Newton's method,

$$f(x_k) - f(x^*) \leq \frac{2\alpha^3}{L^2} \left(\frac{L}{2\alpha^2} \|\nabla f(x_0)\| \right)^{2^{k+1}}.$$

This is much faster than gradient descent due to the power of 2^{k+1} present. Note that in order to work, we need $\|\nabla f(x_0)\|$ to be less than $2\alpha^2/L$, which we can do by running gradient descent for a small amount of steps.

Example 2.2. Take the same example $f(x) = (x_1^2 + 100x_2^2)/2$. Then

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & 100 \end{pmatrix}^{-1} \begin{pmatrix} x_1 \\ 100x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

So Newton's method converges in one step.

2.4 Neural networks

Consider the task of machine learning. Start off with an input $x[0]$, and expected output y, and construct a neural network as follows. We compute $y[0]$ as a linear function of the components of $x[0]$, so that

$$y_i[0] = w_i[0]^T x[0] - b_i[0],$$

where the w_i are weights and $b_i[0]$ are a bias. We can then define $x_i[1] = \phi(y_i[0])$, and inductively define

$$y_i[1] = w_i[1]^T x_[1] - b_i[1],$$

and so on. If ϕ is linear, then each of the $y[i]$ are simply a linear combination of $x[0]$. However, with some other function, such as the sigmoid function, we can get some interesting results. Stopping at $y[1]$, we get that

$$y_1[1](x[0]) = \sum_{j=1}^{d[1]} w_j[1]\phi(w_j[0]^T x[0] - b_j[0]).$$

We can then define the **cost**

$$\sum_{i=1}^{m} (y_i - y_1[1](x_i))^2,$$

which we seek to minimise. This can be done by the techniques we have seen before.

3 Lagrangian Methods

3.1 The Lagrangian Sufficiency Theorem

Remember our problem is to solve the following problem:

$$\text{minimise } f(x) \text{ such that } g(x) = b, \text{ for all } x \in X, \tag{P}$$

for $X \subseteq \mathbb{R}^n$, $b \in \mathbb{R}^m$. We will look at Lagrange's method at solving such problems.

Definition 3.1. The **Lagrangian** is defined as

$$L(x, \lambda) = f(x) + \lambda^T (b - g(x)),$$

with $\lambda \in \mathbb{R}^m$.

Theorem 3.1 (Lagrangian sufficiency theorem)**.** *Let x^* be feasible. Suppose there exists $\lambda^* \in \mathbb{R}^m$ such that*

$$L(x^*, \lambda^*) \leq L(x, \lambda^*) \text{ for all } x \in X.$$

Then x^* is optimal for (P).

Proof: For any feasible x and any λ, we have

$$L(x, \lambda) = f(x) + \lambda^T (b - g(x)) = f(x),$$

since being feasible implies $g(x) = b$. This implies that

$$f(x^*) = L(x^*, \lambda^*) \leq L(x, \lambda^*) = f(x)$$

for all feasible x, as required.

Remark.

1. There is no guarantee we can find such a λ^*.
2. At first sight, we have a method of testing if x^* is optimal (given a λ^*). But how do we find a λ^*?

3.2 Using LST

There is some strategy to solving problems with LST:

1) Minimise $L(x, \lambda)$ subject to $x \in X$, and identify those λ such that the minimum is greater than $-\infty$. Define
$$\Lambda = \{\lambda \mid \min_{x \in X} L(x, \lambda) > -\infty\}.$$

2) For $\lambda \in \Lambda$, the minimum will occur at some $x(\lambda)$.

3) Vary λ until it reaches a value such that $g(x(\lambda)) = b$.

3.3 Examples of LST

We will see how LST can be used in the following examples:

Example 3.1. Minimise $x_1^2 + 3x_2^2$, such that $2x_1 + 3x_2 = b$, for all $x \in \mathbb{R}^2$.

Proof: To start with, notice the Lagrangian is $L = x_1^2 + 3x_2^2 + \lambda(b - 2x_1 - 3x_2)$. Then,
$$\frac{\partial L}{\partial x_1} = 2x_1 - 2\lambda = 0, \quad \frac{\partial L}{\partial x_2} = 6x_2 - 3\lambda = 0.$$
Thus we must have $x_1 = \lambda$, $x_2 = \lambda/2$, and moreover
$$2x_1 + 3x_2 = 2\lambda + 3\frac{\lambda}{2} = b.$$
Therefore, we can set $\lambda^* = 2b/7$, which gives
$$(x_1, x_2) = \left(\frac{2b}{7}, \frac{b}{7}\right) \quad \phi(b) = \frac{b^2}{7}.$$

Example 3.2. Minimise $\dfrac{1}{1 + x_1} + \dfrac{2}{2 + x_2}$, such that $x_1 + x_2 \leq b$, for $x_1, x_2 \geq 0$.

Proof: We will write $x_1 + x_2 + x_3 = b$, for $x_3 \geq 0$ (a slack variable). Then
$$\begin{aligned} L(x, \lambda) &= \frac{1}{1 + x_1} + \frac{1}{2 + x_2} + \lambda(b - x_1 - x_2 - x_3) \\ &= \left(\frac{1}{1 + x_1} - \lambda x_1\right) + \left(\frac{1}{2 + x_2} - \lambda x_2\right) - \lambda x_3 + \lambda b. \end{aligned}$$
If $\lambda > 0$, then this is minimized when $x_3 \to \infty$, which is spurious. Similarly, if $\lambda = 0$, this is minimized as x_1 and x_2 go to infinity. Thus $\lambda < 0$, so the function is minimized when $x_3 = 0$.

Now we are interested in minimising the function

$$\frac{1}{a+x} - \lambda x, \quad x \geq 0.$$

This has optimal solution

$$x(\lambda) = \left(-a + \sqrt{\frac{-1}{\lambda}}\right)^+,$$

where $c^+ = \max(0, c)$. Therefore, we can find that

$$x_1(\lambda) + x_2(\lambda) = \left(-1 + \sqrt{\frac{-1}{\lambda}}\right)^+ + \left(-2 + \sqrt{\frac{-1}{\lambda}}\right)^+ = b.$$

The function on the right is continuous, and goes from 0 to ∞ as λ goes from $-\infty$ to 0. So by the intermediate value theorem, there exists λ^* such that $x_1(\lambda^*) + x_2(\lambda^*) = b$, for all positive b. We get that

$$x_1(\lambda^*) + x_2(\lambda^*) = \begin{cases} 0 & \lambda \leq -1, \\ -1 + 1/\sqrt{-\lambda} & \lambda \in [-1, -1/4], \\ -3 + 2/\sqrt{-\lambda} & \lambda \in [-1/4, 0). \end{cases}$$

If we solve, we get that

$$x^* = \begin{cases} (b, 0) & b \leq 1, \\ \frac{1}{2}(b+1, b-1) & b \geq 1. \end{cases}$$

3.4 Inequality constraints and complementary slackness

We return to a variation of our problem (P);

$$\text{minimise } f(x) \text{ such that } g(x) \leq b, \text{ for all } x \in \mathbb{R}^n. \tag{P'}$$

Writing $g(x) + z = b$, for some $z \geq 0, z \in \mathbb{R}^m$, then

$$L(x, \lambda) = f(x) + \lambda^T(b - g(x) - z).$$

Think about the term $-\lambda^T z$:

- If $\lambda_i > 0$ for some i, then by letting $z_i \to \infty$, $L \to -\infty$, a contradiction.

- If $\lambda_i = 0$, then $\lambda_i z_i = 0$.
- If $\lambda_i < 0$, then $z_i = 0$, so $\lambda_i z_i = 0$.

Therefore, $\lambda^T z^* = 0$. This is called **complementary slackness** of λ^* and z^*.

3.5 Failure of Lagrangian methods

Consider the following example:

Example 3.3. Minimise $f(x)$ such that $x = b$, for $x \geq 0$.

Clearly the minimum is given by $f(b)$. What if we use LST? The Lagrangian is

$$L(x, \lambda) = f(x) + \lambda(b - x).$$

If we use $f(x) = \sqrt{x}$, then we get

$$L(x, \lambda) = \sqrt{x} + \lambda(b - x).$$

This has minimum either at $x = 0$ or $x \to \infty$, so here Lagrangian methods don't work.

We can try the same problem, but with $f(x) = x^2$. Then

$$L(x, \lambda) = x^2 + \lambda(b - x).$$

This has a minimum at $x(\lambda) = \lambda/2$, so we can take $\lambda = 2b$, which works.

To see why this work, take the **value function**

$$\phi(b) = \min_{\substack{x \in X \\ g(x) = b}} f(x).$$

Then Lagrangian methods did not work for $\phi(b) = \sqrt{b}$, but not for $\phi(b) = b^2$. The key reason why is because $\phi(b) = b^2$ is convex, whereas $\phi(b) = \sqrt{b}$ is not.

3.6 Large deviations

Consider rolling a dice n times. The expected sum to see is $3.5n$, but suppose we have instead rolled them in such a way that the sum is $5n$. This could have been done by rolling all 5's, but this seems unlikely. So what is the most likely distribution of rolls?

The problem is then to maximise the multinomial

$$\frac{n!}{n_1! \cdots n_6!} \left(\frac{1}{6}\right)^n = f(x),$$

subject to the constraints

$$\sum_{i=1}^{6} in_i = 5n, \quad \sum_{i=1}^{6} n_i = n.$$

Using Stirling's approximation on the objective function, we can reduce it to finding the minimum of

$$\prod p_i^{p_i+1/2n},$$

where $p_i = n_i/n$ is the proportion of rolls of i. If we take the log of this, this is equivalent to minimising the sum

$$\sum_{i=1}^{6} p_i \log p_i.$$

Then we wish to minimise the Lagrangian

$$L(f, \lambda) = \sum_{i=1}^{6} p_i \log p_i + \lambda \left(5 - \sum_{i=1}^{6} ip_i\right) + \mu \left(1 - \sum_{i=1}^{6} p_i\right).$$

Taking the partial derivatives,

$$\frac{\partial L}{\partial p_i} = \log p_i + 1 - \lambda i - \mu = 0 \implies p_i = e^{(\mu-1)+\lambda i}.$$

From here, we may find satisfactory λ, μ such that the p_i satisfy the functional constraints.

4 The Lagrangian Dual

4.1 Lagrangian necessity

Recall the value function

$$\phi(b) = \inf_{\substack{x \in X \\ g(x)=b}} f(x).$$

We have seen examples where Lagrangian methods have and have not worked. The following theorem explains why.

Theorem 4.1 (Lagrangian necessity)**.** *If the value function ϕ is convex and finite, then there exists λ such that*

$$\phi(b) = \inf_{x \in X} L(x, \lambda).$$

Furthermore, if ϕ is differentiable, then $\lambda = \nabla\phi(b)$.

Proof: If ϕ is convex, by SHT there exists λ such that $\phi(c) \geq \phi(b) + \lambda^T(c-b)$, for all c. Thus,

$$\begin{aligned}
\phi(b) &= \inf_c \{\phi(c) + \lambda^T(b-c)\} \\
&= \inf_c \inf_{\substack{x \in X \\ g(x)=c}} \{f(x) + \lambda^T(b-c)\} \\
&= \inf_c \inf_{\substack{x \in X \\ g(x)=c}} \{f(x) + \lambda^T(b-g(x))\} \\
&= \inf_{x \in X} L(x, \lambda).
\end{aligned}$$

If $\phi(b)$ is differentiable, then $\lambda = \nabla\phi(b)$.

Therefore, we know Lagrangian methods work if ϕ is convex. Thus it would be nice to know when ϕ is convex. Thankfully, we have the following theorem:

Theorem 4.2 (Sufficiency conditions for convexity of the value function)**.** *Suppose*

1. *X is convex.*
2. *The objective function f is convex.*
3. *The functional constraint is of the form $g(x) \leq b$.*
4. *g_i is convex for all $1 \leq j \leq m$.*

Then ϕ is convex.

Proof: We prove that, given b_1, b_2 and $\lambda \in (0,1)$, that

$$\phi(\lambda b_1 + (1-\lambda)b_2) \leq \lambda\phi(b_1) + (1-\lambda)\phi(b_2).$$

Assume $\phi(b_1) = f(x_1)$, $\phi(b_2) = f(x_2)$. Then consider $x = \lambda x_1 + (1-\lambda)x_2$.

- Since X is convex, $x \in X$.
- Since the objective function f is convex, $f(x) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$.
- Since the g_i are convex,

$$g(x) \leq \lambda g(x_1) + (1-\lambda)g(x_2) \leq \lambda b_1 + (1-\lambda)b_2 = b.$$

Thus,

$$\phi(\lambda b_1 + (1-\lambda)b_2) \leq f(x) \leq \lambda\phi(b_1) + (1-\lambda)\phi(b_2),$$

as required.

4.2 Shadow prices

Suppose ϕ is differentiable: then $\nabla\phi(b) = \lambda$. Now consider a constraint of the form

$$g(x) \leq b \iff g(x) + z = b.$$

Suppose we increase b: how much can we decrease ϕ by? We get that

$$\phi(b+\varepsilon) - \phi(b) = \nabla\phi(b) \cdot \varepsilon + \mathcal{O}(\varepsilon^2) \approx \lambda^T\varepsilon.$$

By complementary slackness, since $z \geq 0$, we must have $\lambda \leq 0$. Therefore increasing b_i by ε_i allows the minimal value to decrease by $\lambda_i\varepsilon_i$.

For this reason, we call the Lagrange multipliers are called **shadow prices**.

4.3 The Lagrangian dual problem

Recall that

$$\Lambda = \{\lambda \mid \min_{x\in X} L(x,\lambda) > -\infty\}.$$

For $\lambda \in \Lambda$, we define

$$L(\lambda) = \min_{x\in X} L(x,\lambda).$$

We can also write

$$X_b = \{x \mid g(x) = b, x \in X\}.$$

Theorem 4.3 (Weak duality theorem)**.** *For any feasible $x \in X_b$ and $\lambda \in \Lambda$,*

$$f(x) \geq L(\lambda).$$

Proof: For $x \in X_b$, $\lambda \in \Lambda$,

$$f(x) = L(x, \lambda) \geq \min_{x \in X_b} L(x, \lambda) \geq \min_{x \in X} L(x, \lambda) = L(\lambda).$$

As a corollary, $\phi(b) \geq L(\lambda)$. Therefore, we can make progress on minimizing $\phi(b)$ by maximizing $L(\lambda)$. This is another problem

$$\text{maximize } L(\lambda) \text{ subject to } \lambda \in \Lambda. \tag{D}$$

This is known as the **Lagrangian dual problem**, and the original problem is the **primal problem**. By the weak duality theorem, the optimal value of the dual is no more than the optimal value of the primal. Moreover, if they are equal, we call it **strong duality**.

There is also an economic interpretation hiding in here: consider an agent producing goods in quantities $(x_1, \ldots, x_n)$, which they sell for $f(x_1, \ldots, x_n)$. Initially they have m resources, say $(b_1, \ldots, b_m)$, and to produce the goods, they require $g(x_1, \ldots, x_n)$ resources.

If $g_i(x) > b_i$, then they have a **shortfall**, and if $g_i(x) < b_i$, they have a **surplus**. Hence we can consider a profit function of

$$f(x) + \sum_{i=1}^{m} \lambda_i (b_i - g_i(x)),$$

where the price of resource i is λ_i. In fact, this resource is simply the Lagrangian

$$f(x) + \lambda^T (b - g(x)).$$

In a competitive market, λ will be such that the profit is minimized. Hence the market is solving the problem

$$\min_{\lambda} \max_{x} \{f(x) + \lambda^T (b - g(x))\}.$$

Typically, f is concave, since as supply increase, demand will likely decrease. Similarly g is concave, so this problem can be solved by Lagrangian methods.

Example 4.1. Recall example (3.1): minimizing $x_1^2 + 3x_2^2$, such that $2x_1 + 3x_2 = b$. In this case L is minimized at $x = \lambda(1, 1/2)$. This gives

$$L(\lambda) = L(x^*, \lambda) = \lambda^2 + \frac{3}{4}\lambda^2 + \lambda\left(b - 2\lambda - 3\frac{\lambda}{2}\right) = \lambda b - \frac{7}{4}b^2.$$

This is minimized at $\lambda = 2b/7$, and $\phi(b) = b^2/7 = L(\lambda^*)$, so we have strong duality.

4.4 Barrier methods

Consider our slack problem (P′), on all of $\mathbb{R}^n$. We can define a new problem

$$\text{minimize } \left[f(x) - \varepsilon \sum_{j=1}^{m} \log(b_j - g_j(x))\right] \text{ subject to } x \in \mathbb{R}^n. \qquad (\mathrm{P}_\varepsilon)$$

Since $\log x \to -\infty$ as $x \to 0$, this ensure that the minimum of P_ε is away from the boundary, and thus it is a stationary point.

Theorem 4.4. *Suppose x^* and x_ε are optimal for P' and P_ε, respectively. Then*

$$0 \le f(x_\varepsilon) - f(x^*) \le m\varepsilon,$$

where m is the number of constraints.

Proof: The optimum of P_ε occurs at a stationary point, thus where

$$\nabla f(x_\varepsilon) + \varepsilon \sum_{j=1}^{m} \frac{\nabla g_j(x_\varepsilon)}{b_j - g_j(x_\varepsilon)} = 0.$$

Let $\bar{\lambda}$ be such that

$$\bar{\lambda}_i = \frac{-\varepsilon}{b_i - g_i(x_\varepsilon)}.$$

Then,

$$\nabla\left[f(x) + \bar{\lambda}^T(b - g(x))\right] = \nabla f(x) - \bar{\lambda}^T \nabla g(x) = 0,$$

at $x = x_\varepsilon$. Therefore,

$$f(x^*) \ge L(\bar{\lambda}) \ge \inf_x \left[f(x) + \bar{\lambda}^T(b - g(x))\right] = f(x_\varepsilon) + \bar{\lambda}^T(b - g(x_\varepsilon)) = f(x_\varepsilon) - m\varepsilon.$$

5 Linear Programming

5.1 Extreme points and optimality

We look at a special case of our problem (P′): consider when f and g_i are all linear. Then we have a **linear programming problem**. These tend to be much easier to solve than the general problem.

Definition 5.1. We say x is an **extreme point** of a convex set S if whenever $x = \lambda y + (1 - \lambda z)$, for $y, z \in S$ and $\lambda \in (0, 1)$, then $y = z = x$.

Extreme points are useful in linear programming, due to the following theorem:

Theorem 5.1 (Fundamental Theorem of Linear Progamming)**.** *If a linear programming problem is feasible and bounded, then it has an optimum at an extreme point of the feasible set.*

From this, we can define a very rudimentary algorithm to solve linear programming problems:

1) Find all the vertices of the feasible set.

2) Pick the best one.

However, this is computationally expensive: if $X \subset \mathbb{R}^n$, and there are m constraints, then we have $\binom{n+m}{m}$ vertices.

5.2 Basic solutions

Consider the following problem:

$$\begin{aligned} \text{maximize } & x_1 + x_2, \\ \text{subject to } & x_1 + 2x_2 + x_1 = 6, \\ & x_1 - x_2 + z_2 = 3, \\ & x_1, x_2, z_1, z_2 \geq 0. \end{aligned}$$

We can write the constraints as

$$\begin{pmatrix} 1 & 2 & 1 & 0 \\ 1 & -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} 6 \\ 3 \end{pmatrix}.$$

This is of the form $Ax + z = b$, where $x, z \geq 0$.

To find a point on the simplex, we find the values when x_1, x_4 are non-zero and x_2, x_3 are both 0. Then we get

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 6 \\ 1 \end{pmatrix}.$$

Thus $x_1 = 6, z_2 = -3$. However this is not feasible as $z_2 < 0$. We can do this again, considering two variables to be 0 and finding the value of the others. We generalise this approach as follows:

Definition 5.2.

- The **support** of a vector x is the set of indices for which x is non-zero:

$$S(x) = \{i \mid x_i \neq 0\}.$$

- A **basic solution** to $Ax = b$, where A is $m \times n$ with $m \leq n$, is one for which $|S(x)| \leq m$.
- Set B is called the **basis** if

$$\begin{cases} x_i = 0 & i \notin B, \\ x_i = 1 & i \in B. \end{cases}$$

  x_i is called **basic** if $x_i \in B$, and **non-basic** if $x_i \notin B$.
- A basic solution is **non-degenerate** if it contains exactly $n - m$ vector components which are 0, so $|S(x)| = m$.

In this course, we deal with only non-degenerate basic solutions. This is equivalent to matrix A having rank m.

For a basis B, let A_B denote the $m \times m$ matrix whose columns are those with indices in B. Then by the above assumption, A_B is invertible, so there exists x_B such that

$$A_B x_B = b \iff x_B = A_B^{-1} b.$$

Theorem 5.2. *A vector is a basic feasible solution of $Ax = b$ if and only if it is an extreme point of the set $X(b) = \{x \mid Ax = b, x \geq 0\}$.*

Proof: Suppose first that x is not a basic feasible solution, so $|S(x)| > m$. Then there exists y such that $S(y) \subseteq S(x)$ and $Ay = 0$. But then, for small enough ε, x is the midpoint of $x + \varepsilon y$ and $x - \varepsilon y$, so x is not an extreme point.

Now suppose x is not an extreme point. Then $x = \delta y + (1 - \delta z)$, for some $y, z \in X(b)$, with $y \neq z$. But since

$$Az = b = Ax = A(\delta y + (1 - \delta)z) = Az + \delta A(y - z),$$

this implies $A(y-z) = 0$, so since $S(y-z) \subseteq S(x)$, we must have $|S(x)| > m$, so x is not a basic feasible solution.

Theorem 5.3. *If a linear program is feasible and bounded, then it has an optimal solution that is a basic feasible solution.*

Proof: Suppose x is optimal but is not a basic feasible solution. Then there exists y such that $S(y) \subseteq S(x)$ and $Ay = 0$. Consider $x(\varepsilon) = x + \varepsilon y$. For small ε, this is feasible, and $Ax(\varepsilon) = b$.

Now say $c^T y \geq 0$, by perhaps changing y to $-y$. Then

$$c^T x(\varepsilon) = c^T x + \varepsilon c^T y \geq c^T x.$$

Increase ε until some $i \in S(x)$ becomes 0. Thus we have found $x(\varepsilon)$ such that $|S(x(\varepsilon))| < |S(x)|$.

This proves the fundamental theorem of linear programming. With this, we can find another algorithm:

1) Find all basic solutions.

2) Test to see which are feasible.

3) Choose the best solution.

Again, this is slow, as there are $\binom{n+m}{m}$ basic solutions.

5.3 Preview of the Simplex Method

The simplex algorithm is as follows:

1) Start with a basic feasible solution.

2) Test if it is optimal.

3) If it is not, then move to a better basic feasible solution are repeat.

To see what this means, write a solution as $x_B + x_N$, where B is a basis, and N

are the other elements. Then we get

$$\begin{aligned} A_B x_B + A_N x_N &= b, \\ x_B &= A_B^{-1} b - A_B^{-1} A_N x_N \geq 0. \end{aligned}$$

Then, the objective function is

$$c^T x = c_B^T x_B + c_N^T x_N = c_B^T A_B^{-1} b + (c_N^T - c_B^T A_B^{-1} A_N) x_N.$$

Hence, if $c_N^T - c_B^T A_B^{-1} A_N$ has every component less than or equal to 0, then we are at the optimum.

On the other hand, if the i'th component is positive, for $i \in N$, then by increasing x_i, $c^T x$ will increase. However to maintain $Ax = b$, other variables will need to change, and in particular we need $x_B \geq 0$.

Therefore we increase x_i until some x_j, for $j \in B$, becomes 0. Then we can repeat our algorithm, as we have a new basic feasible solution.

6 The Simplex Method

6.1 The simplex algorithm

We write down a simplex tableau as

x_1	x_2	z_1	z_2	
1	2	1	0	6
1	−1	0	1	3
1	1	0	0	0

We can also write this as

x_B	x_N	
$A_B^{-1}A_B = I$	$A_B^{-1}A_N$	$A_B^{-1}b$
$c_B^T - c_B^T A_B^{-1} A_B = 0$	$c_N^T - c_B^T A_B^{-1} A_N$	$-c_B^T A_B^{-1} b$

We also index the table by a_{ij}, where we zero-index, with a_{i0} being the rightmost column and a_{0j} being the bottom row.

The simplex algorithm is as follows:

1. **Choose a pivot column.** Pick an element at the bottom for which $a_{0j} > 0$. This is the same as picking a j such that $c_N^T - c_B^T A_B^{-1} A_n$ is positive. If no such j exists, then we are at an optimum.
2. **Find the pivot row.** Suppose $a_{ij} > 0$. As x_j increases, x_i will have to decrease. We can only increase x_j by an amount

$$\min\left(\frac{a_{i0}}{a_{ij}} \mid a_{ij>0}\right).$$

   If i minimizes a_{i0}/a_{ij}, then i is the pivot row. If the problem is non-degenerate, then the pivot row is unique. The pivot is the intersection of the pivot row and column.
3. **Perform a pivot.** To perform a pivot, we do the following:
   (a) multiply the pivot row by $1/a_{ij}$,
   (b) add $-a_{kj}/a_{ij}$ times the pivot row to each row $k \neq i$, including the bottom row.

Remark.

1. At every step, we have two things in mind:
   (a) a particular choice of basis,
   (b) a convenient way of writing the problem.
2. In general, there is no way to efficiently choose pivot column. However in practice the running time is about linear in m and n.
3. The tableau contains redundant information, such as the identity matrix and the zeroes in the bottom row.

7 The Dual Linear Program

7.1 The dual problem for LP

Recall our problem (P′). In linear programming, our problem is

$$\text{maximize } c^T x \text{ such that } Ax \leq b, x \geq 0. \tag{LP}$$

The Lagrangian is then

$$L(x, z, \lambda) = c^T x - \lambda^T(Ax + z - b) = (c^T - \lambda^T A)x - \lambda^T z + \lambda^T b.$$

Then

$$\Lambda = \{\lambda \mid \sup_{z,x\geq 0} L(x, z, \lambda) < \infty\} = \{\lambda \mid \lambda \geq 0, c^T - \lambda^T A \leq 0\}.$$

Hence

$$L(\lambda) = \sup_{z,x\geq 0} L(x, z, \lambda) = \lambda^T b.$$

This has an optimum when $(c^T - \lambda^T A)x = 0$ and $\lambda^T z = 0$. Hence the dual problem, which is to minimize $L(\lambda)$, for $\lambda \in \Lambda$, becomes

$$\text{minimize } \lambda^T b \text{ such that } \lambda \geq 0, \lambda^T A \geq c^T.$$

But this is the same as

$$\text{maximize } (-b)^T \lambda \text{ such that } (-A)^T \lambda \leq -c, \lambda \geq 0. \tag{LD}$$

This implies the following:

Lemma 7.1. *In linear programming, the dual of the dual is the primal.*

We also get the weak duality theorem:

Theorem 7.1 (Weak duality theorem for LP)**.** *If x is feasible for LP and λ is feasible for LD, then $c^T x \leq \lambda^T b$.*

Proof: Note that $L(x, z, \lambda) = c^T x - \lambda^T(Ax + z - b)$. Since x, z, λ are feasible,

$$c^T x = L(x, z, \lambda) = (c^T = \lambda^T A)x - \lambda^T z + \lambda^T b \leq \lambda^T b.$$

7.2 Conditions for Optimality

Theorem 7.2 (Sufficient conditions for optimality in LP)**.** *If x^*, z^* is feasible for LP and λ^* is feasible for LD, and $(c^T - \lambda^{*T} A)x^* = \lambda^{*T} z^* = 0$, then x^* and λ^* are optimal for P and D. Furthermore, $c^T x^* = \lambda^{*T} b$.*

Proof: We have

$$c^T x^* = L(x^*, z^*, \lambda^*) = (c^T - \lambda^{*T} A)x^* - \lambda^{*T} z^* + \lambda^{*T} b = \lambda^{*T} b.$$

But for all x feasible for P, $c^T x \leq \lambda^{*T} b$, so x^* is optimal.

Theorem 7.3 (Strong duality in LP)**.** *If both LP and LD are feasible, then there exists x, λ satisfying the sufficiency conditions.*

This follows since LP and LD can be solved by Lagrangian methods.

7.3 The utility of primal dual theory

The usefulness of the duals are as follows:

1. LD may be easier to solve than LP. For example, if LP has three variables and two constraints, LD has two variables and three constraints.
2. Having found solutions to LD we can find solutions to LP by using complementary slackness.
3. Sometimes it is best to solve both LP and LD simultaneously.
4. We seek LP and LD feasibility and complementary slackness.

7.4 Primal-dual relationships

We can look at the lists of basic solutions for LP and LD, and observe the following:

1. For each basic solution for LP, there is a corresponding basic solution for LD. Moreover, each pair has the same value of the objective function, and satisfies complementary slackness.
2. There is only one pair that is feasible for both LP and LD, and that solution has same optimum.
3. For any x feasible for LP and λ feasible for LD, we have $c^T x \leq b^T \lambda$, with equality if and only if x, λ are optima for LP and LD.
4. LP and LD are related thus:
   - If LP has a finite optimum, then LD has a finite optimum.
   - If LP is feasible, then LD is bounded.
   - If LP is infeasible, then LD is infeasible of unbounded.

8 Shadow prices

8.1 Dual problem and the final tableau

Consider the initial tableau

$$\begin{array}{cc|c} A_N & A_N & b \\ \hline c_N^T & c_B^T & 0 \end{array}$$

Suppose the bottom row of the final tableau comes from subtracting λ_i times the i'th row from the initial bottom row. Then we get $c_N^T - \lambda^T A_N \leq 0$ and $c_B^T - \lambda^T A_B \leq 0$. However, the final part is $-\lambda^T$, so we can read off λ from the final tableau.

8.2 Shadow prices and sensitivity analysis

We will look at how the solution changes for small changes in b or c. Consider an initial tableau

1	2	1	0	6
1	-1	0	1	3
1	1	0	0	0

and a final tableau

0	1	$\frac{1}{3}$	$-\frac{1}{3}$	1
1	0	$\frac{1}{3}$	$\frac{2}{3}$	4
0	0	$-\frac{2}{3}$	$-\frac{1}{3}$	-5

If we change the value of b from $(6, 3)$ to $(6 + \varepsilon_1, 3 + \varepsilon_2)$ then since we know that the bottom row $R_0' = R_0 - \frac{2}{3}R_1 - \frac{1}{3}R_2$, we get that the final value will be

$$5 + \frac{2}{3}\varepsilon_1 + \frac{1}{3}\varepsilon_2 = 5 + \lambda^T \varepsilon.$$

Moreover since $R_1' = \frac{1}{3}R_1 - \frac{1}{3}R_2$ and $R_2' = \frac{1}{3}R_1 + \frac{2}{3}R_2$, we get the final tableau of

0	1	$\frac{1}{3}$	$-\frac{1}{3}$	$1 + \frac{1}{3}\varepsilon_1 - \frac{1}{3}\varepsilon_2$
1	0	$\frac{1}{3}$	$\frac{2}{3}$	$4 + \frac{1}{3}\varepsilon_1 + \frac{2}{3}\varepsilon_2$
0	0	$-\frac{2}{3}$	$-\frac{1}{3}$	$-5 - \frac{2}{3}\varepsilon_1 - \frac{1}{3}\varepsilon_2$

Hence the maximum value is indeed $5 + \frac{2}{3}\varepsilon_1 + \frac{1}{3}\varepsilon_2$, provided that $1 + \frac{1}{3}\varepsilon_1 - \frac{1}{3}\varepsilon_2$ and $4 + \frac{1}{3}\varepsilon_1 + \frac{2}{3}\varepsilon_2$ are greater than or equal to 0.

Similarly, if we want to change the value of c, we can do the same thing on the dual tableau. However we can also do this more directly. If our initial tableau was

1	2	1	0	6
1	-1	0	1	3
$1+\delta_1$	$1+\delta_2$	0	0	0

then doing the same process leads us to

0	1	$\frac{1}{3}$	$-\frac{1}{3}$	1
1	0	$\frac{1}{3}$	$\frac{2}{3}$	4
δ_1	δ_2	$-\frac{2}{3}$	$-\frac{1}{3}$	-5

Therefore we need to subtract δ_1 times the second row and δ_2 times the first row, to give a final tableau of

0	1	$\frac{1}{3}$	$-\frac{1}{3}$	1
1	0	$\frac{1}{3}$	$\frac{2}{3}$	4
0	0	$-\frac{2}{3}-\frac{1}{3}\delta_1-\frac{1}{3}\delta_2$	$-\frac{1}{3}-\frac{2}{3}\delta_1+\frac{1}{3}\delta_2$	$-5-4\delta_1-\delta_2$

For this to be an optimum, we require $-\frac{2}{3}-\frac{1}{3}\delta_1-\frac{1}{3}\delta_2$ and $-\frac{1}{3}-\frac{2}{3}\delta_1+\frac{1}{3}\delta_2$ to be non-positive, and then the optimum will be

$$5+4\delta_1+\delta_2.$$

8.3 Shadow prices and the diet problem

Let a_{ij} be the amount of vitamin i in one unit of food stuff j. Then our problem is to

$$\text{minimize } P=\sum_{j=1}^{n} x_j c_j \text{ such that } \sum_{j=1}^{n} a_{ij}x_j \geq b_i \text{ for all } i, x_j \geq 0.$$

Suppose that a vitamin company sells a vitamin based diet. If they choose price p_i, then their problem is to

$$\text{maximize } \sum_{i=1}^{n} b_i p_i \text{ such that } \sum_{i=1}^{n} a_{ij}p_i \leq c_j \text{ for all } j, p_i \geq 0.$$

This is the dual to our problem.

9 Two Person Zero-Sum Games

9.1 Games with a saddle point

A **zero-sum** game is one in which one player wins and the other loses. The players play simultaneously, and both have a **payoff matrix** A, which is $m \times n$. Then a_{ij} is the amount that the first player wins and the second player loses, if the first player picks i and the second player picks j. Consider the payoff matrix

$$\begin{bmatrix} -5 & 3 & 1 & 20 \\ 5 & 5 & 4 & 6 \\ -4 & 6 & 0 & -5 \end{bmatrix}.$$

We look at what the players will play, given what the other player has played.

- If 2 plays the first column, then 1 plays the second row, gaining 5.
- If 2 plays the second column, then 1 can gain 6.
- If 2 plays the third column, then 1 can gain 4.
- If 2 plays the fourth column, then 1 can gain 20.

We can do a similar thing, looking at how 2 will play.

- If 1 plays the first row, then 2 will lose -5.
- If 1 plays the second row, then 2 will lose 4.
- If 1 plays the third row, then 2 will lose -5.

Notice that here, the minimum column maximum is the maximal row minimum, and this point is $(2, 3)$. Hence we say $(2, 3)$ is a **saddle point**.

Definition 9.1. A saddle point of a payoff matrix A is a pair of strategies (i^*, j^*) such that

$$a_{i^*j^*} = \min_j \max_i a_{ij} = \max_i \min_j a_{ij}.$$

Remark.

1. Each player maximizes his minimum gain.
2. It does not matter if a player announces his move in advance.
3. We always have
$$\max_i \min_j a_{ij} \leq \min_j \max_i a_{ij}.$$

9.2 Game without a saddle-point

We consider the game Morra. Two players displays either one or two fingers, and guess how many fingers the opponent will show. If one person wins and one person loses, the gain is the total number of fingers shown. Thus the payoff matrix looks as follows:

$$\begin{array}{|cccc|}\hline 0 & 2 & -3 & 0 \\ -2 & 0 & 0 & 3 \\ 3 & 0 & 0 & -4 \\ 0 & -3 & 4 & 0 \\ \hline\end{array}$$

Here, there is no saddle point, as

$$-2 = \max_i \min_j a_{ij} < \min_j \max_i a_{ij} = 2.$$

9.3 Determination of an optimal strategy

Instead of playing a fixed value, we look at what happens when the players play each value with a fixed probability. Say player 1 plays row i with probability p_i, and player 2 plays column j with probability q_j.

Now if player 2 plays j, then player 1 will gain

$$\sum_{i=1}^{n} a_{ij} p_i.$$

Therefore, player 1 attempts to

$$\text{maximize } \left\{ \min_j \sum_i a_{ij} p_i \right\} \text{ such that } \sum_i p_i = 1. p_i \geq 0.$$

This is equivalent to the problem of

$$\text{maximize } v \text{ such that } \sum_i a_{ij} p_i \geq v \text{ for all } j, \sum_i p_i = 1, p_i \geq 0.$$

If e is the all ones vector, then we can write this as

$$\text{maximize } v \text{ such that } p^T A \geq e^T v, e^T p = 1, p \geq 0. \tag{GP}$$

Similarly, the problem for player 2 is

$$\text{minimize } v \text{ such that } Aq \leq ev, e^T q = 1, q \geq 0. \tag{GD}$$

Note that these are dual problems.

Theorem 9.1. *Suppose $p \in \mathbb{R}^m$, $q \in \mathbb{R}^n$, and $v \in \mathbb{R}$, such that*

(a) $p \geq 0$, $e^T p = 1$, $p^T A \geq v e^T$,

(b) $q \geq 0$, $e^T q = 1$, $Aq \leq ve$,

(c) $v = p^T A q$.

*Then p is optimal for GP and q is optimal for GD, with common optimum, which we call the **value of the game**, v.*

Proof: Player 1 can guarantee to get

$$\min_q p^T A q \geq \min_q (ve^T) q = v,$$

and similarly player 2 pays out no more than

$$\max_p p^T A q \leq \max_p p^T (ve) = v.$$

Thus the optimal p, q give a saddle point in mixed strategies.

Remark.

1. If a game has a saddle point in pure strategies, then the values of p and q in theorem 9.1 is $p_{i^*} = q_{j^*} = 1$, and the other p_i, q_j are equal to 0.
2. For two finger Morra, the optimal strategy is

   $$p = q = \left(0, \frac{4}{7}, \frac{3}{7}, 0\right).$$

   In fact, there is another solution

   $$p = q = \left(0, \frac{3}{5}, \frac{2}{5}, 0\right),$$

   and any strategy on the line between these two strategies works.
3. To solve using the simplex algorithm, we do the following:
   (a) Add a constant k to each a_{ij} to make the game positive.
   (b) We maximize v such that $p^T A \geq e^T v$, subject to $e^T p = 1$. Letting $x = p/v$, then this becomes

   $$\text{minimize } e^T x \text{ such that } x^T A \geq e^T, x \geq 0.$$

The dual is

$$\text{maximize } e^T y \text{ such that } Ay \leq e, y \geq 0.$$

We can then solve this using the simplex algorithm.

10 Maximal Flow in a Network

10.1 Max-flow/min-cut theory

A network is a connected graph of vertices and edges. Here we look at a directed network, where the edge between vertex i and j has capacity c_{ij}.

Our problem is to maximise v, the flow into and out of the network, such that $0 \leq x_{ij} \leq c_{ij}$, where x_{ij} is the flow along edge (i, j), and the flow out of a node i equals the flow into the node i, apart from the start and end node. Thus,

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = \begin{cases} v & i = 1, \\ 0 & i = 2, \ldots, n-1, \\ -v & i = n. \end{cases}$$

Note we can turn an undirected edge into two directed edges.

Definition 10.1. A **cut** $(S, \bar{S})$ is a partition of the node set N into two disjoint subsets, such that $1 \in S$, $n \in \bar{S}$.

Definition 10.2. The **capacity** of a cut is

$$C(S, \bar{S}) = \sum_{\substack{i \in S \\ j \in \bar{S}}} c_{ij}.$$

Note that the maximal flow is less than or equal to $C(S, \bar{S})$, over all possible cuts.

Theorem 10.1 (Max flow/min cut Theorem)**.** *The maximal flow value through the network is equal to the minimal cut capacity.*

Proof: We sum the feasibility constraints over $i \in S$:

$$\begin{aligned} v &= \sum_{i \in S, j \in N} x_{ij} - \sum_{j \in N, i \in S} x_{ji} \\ &= \sum_{i \in S, j \in \bar{S}} x_{ij} - \sum_{j \in \bar{S}, i \in S} x_{ji} \\ &\leq C(S, \bar{S}). \end{aligned}$$

Therefore, any flow v is less than or equal to any cut capacity $C(S, \bar{S})$.

Now let f be a flow and define $S \subset N$ recursively:

(1) $1 \in S$.

(2) If $i \in S$ and $x_{ij} < c_{ij}$, then $j \in S$.

(3) If $i \in S$ and $x_{ji} > 0$, then $j \in S$.

We keep doing this until no more nodes can be added to S.

Now if $n \in S$, then we can increase the flow along some path. Eventually we find $n \notin S$. Then

$$\begin{aligned} v &= \sum_{i \in S, j \in \bar{S}} x_{ij} - \sum_{j \in \bar{S}, i \in S} x_{ji} \\ &= \sum_{i \in S, j \in \bar{S}} c_{ij} - 0 = C(S, \bar{S}). \end{aligned}$$

Corollary 10.1. *If we can reach a flow with value equal to a cut capacity then we have found the maximum flow.*

10.2 Ford-Fulkerson algorithm

The Ford-Fulkerson algorithm to find maximum flow is as follows:

1) Start with a feasible flow, for example $x_{ij} = 0 = v$.

2) Consider S recursively, as above.

3) If $n \in S$, there is a path from 1 to n along which flow can be increased by

$$\varepsilon = \min_{(i,j)} \max\left(x_{ji}, c_{ij} - x_{ij}\right) > 0.$$

We construct this flow, then recursively apply this algorithm again.

4) Eventually $n \notin S$, then we are done.

Provided all capacities are rational, the algorithm will terminate in a finite number of steps.

10.3 Hall's matching theorem

Consider a bipartite graph between sets of vertices L and R, where $|L| = |R|$. We wish to find when there is a perfect matching between L and R.

Theorem 10.2 (Hall's theorem)**.** *Consider a bipartite graph $G = (L \cup R, E)$ with $|L| = |R|$. It has a perfect matching if and only if $|N(X)| \geq |X|$ for every $X \subseteq L$.*

Here we define

$$N(X) = \{j \mid j \in R, (i,j) \in E \text{ for some } i \in X\},$$

i.e. $N(X)$ is the set of neighbours of X.

Proof: Clearly if a perfect matching exists, then $|N(X)| \geq X$ for every subset X.

Conversely, take a network on the bipartite graph $(L \cup R)$. We assign a start node S and end node T, and connect S with every node in L with capacity 1, and similarly connect T with every node in R with capacity 1. Then we give every edge in E a capacity of ∞. Apply the Ford-Fulkerson algorithm to this network.

At the end, we have unit flows along some edges, and no flow along some others. Then consider the edges in S and $\bar{S}$. If X is the edges in L, which are in S, then the set of edges in R which are in S has cardinality $|N(X)| \geq |X|$. Then since $T \in \bar{S}$, the edges connecting $N(X)$ and T must be cut. Hence at least $|L| - |X| + |X|$ edges must be cut, as required.

11 Minimum Cost Circulation Problems

11.1 Minimum cost circulation

Consider a closed network. For each arc (i, j), assign a flow x_{ij}, such that $c_{ij}^- \leq x_{ij} \leq c_{ij}^+$. Then we want to minimize the cost

$$\sum_{ij} d_{ij} x_{ij},$$

subject to the constraint that the flow is a circulation, meaning

$$\sum_j x_{ij} - \sum_j x_{ji} = 0.$$

for each i. Note this is equivalent to the maximum flow problem: take our network, give cost 0 to each edge, and add an edge of cost -1 between T and S.

11.2 Sufficient conditions for minimal cost circulation

Let X be constraints on x_{ij}

$$X = \{x_{ij} \mid c_{ij}^- \leq x_{ij} \leq c_{ij}^+\}.$$

Then the Lagrangian is

$$\begin{aligned} L(x, \lambda) &= \sum_{ij} d_{ij} x_{ij} + \sum_i \lambda_i \left(\sum_j x_{ji} - \sum_j x_{ij} \right) \\ &= \sum_{ij} (d_{ij} - \lambda_i + \lambda_j) x_{ij}. \end{aligned}$$

Hence, to minimize, we get

$$x_{ij} = \begin{cases} c_{ij}^- & \text{if } d_{ij} - \lambda_i + \lambda_j > 0, \\ c_{ij}^+ & \text{if } d_{ij} - \lambda_i + \lambda_j < 0. \end{cases}$$

Therefore if $c_{ij}^- < x_{ij} < c_{ij}^+$, then $d_{ij} - \lambda_i + \lambda_j = 0$. In this case, $\lambda_i - \lambda_j$ is called the **tension**.

11.3 The transportation problem

Consider supplies at node $1, \ldots, n$, in amounts $s_1, \ldots, s_n$, and demand at nodes $1, \ldots, m$, in amount $d_1, \ldots, d_m$. Then we want to minimise

$$\sum_{ij} d_{ij} x_{ij},$$

where d_{ij} is the cost and x_{ij} is the amount shipped, subject to constraints

$$\sum_j x_{ij} = s_i, quad \sum_i x_{ij} = d_j,$$

and $x_{ij} \geq 0$. Note that we can change the transportation problem into a minimum cost circulation problem, and it can be shown the converse is true as well.

12 Transportation and Assignment Problems

12.1 The transportation algorithm

First using Lagrangian methods on the transportation problem, the Lagrangian is

$$\begin{aligned} L(x, \lambda, \mu) &= \sum_{ij} x_{ij} d_{ij} + \sum_i \lambda_i \left(s_i - \sum_j x_{ij} \right) + \sum_j \mu_j \left(d_j - \sum_i x_{ij} \right) \\ &= \sum_{ij} (d_{ij} - \lambda_i - \mu_j) x_{ij}. \end{aligned}$$

Note that whenever $x_{ij} > 0$, we must have $d_{ij} = \lambda_i + \mu_j$. Moreover everywhere we must have $d_{ij} \geq \lambda_i + \mu_j$. Note that λ_i, μ_j is determined up to an additive constant, so we can let $\lambda_1 = 0$.

The transportation algorithm is as follows:

1) Arrange the problem into a grid.

2) We use the Northwest method: greedily work from Northwest, supplying as much as we can from one node to another. If we run out of demand, we move right, and if we run out of supply, we move down. This gives a basic feasible solution, where the number of non-zero entries is one less than the sum of the number of rows and columns.

3) For each positive entry, we can figure out what the value of λ_i and μ_j is, by using $d_{ij} = \lambda_i + \mu_j$.

4) Everywhere else, we check that $d_{ij} \geq \lambda_i + \mu_j$. If this is true, we are done.

5) Otherwise, find a violating cell, and add ε to it. To maintain the supply and demand conditions, add and subtract ε from the other cells which form a cycle with the violating cell. Then we repeat this algorithm.

Remark. The route around which you need to adjust may be more complicated, depending on which cell we choose to change.

Index