EE5904/ME5404

Neural Networks

Part II

Support Vector Machines

and

Reinforcement Learning

Peter C. Y. Chen PhD

Associate Professor
Department of Mechanical Engineering
College of Design and Engineering
National University of Singapore
Email: mpechenp@nus.edu.sg
https://sites.google.com/view/peter-chen/home

Contents

Preface

These notes are supplementary to the slides used in the lectures. They represent an additional means for explaining various abstract concepts, with problems and their solutions presented in detail to illustrate these concepts. They can be studied (as preparation) prior to the lectures on specific topics. In this context, these notes serve to facilitate independent learning by the students.

Chapters 1 to 4 are essential material for the topic of support vector machines, while Chapter 5 provides detailed solutions to a number of problems about support vector machines. Chapters 6, 7 and 8 cover the basic concepts and techniques of reinforcement learning, with a number of examples illustrating the detailed steps of key algorithms.

These notes are to be used in conjunction with the lectures slides, which contain many diagrams and illustrations to provide intuitive visual representation and interpretation of the concepts. The slides aim to enable the students to grasp the high-level concepts and solution approaches quickly, while these supplementary notes and problems aim to facilitate the study of the course material in greater detail, with reference to the structure laid out in the slides.

Peter C. Y. Chen

February 2023

Part A

Support Vector Machines

Chapter 1

Preliminaries

1.1 Basic definitions

$\mathbb{R}$ denotes the set of real numbers. $\mathbb{R}^n$, or $\mathbb{R}^{n\times 1}$, denotes an n-dimensional space of real numbers. Small boldface letters (e.g., $\mathbf{x}$) usually denote vectors. Capital boldface letters (e.g., $\mathbf{A}$) usually denote matrices. The transpose of a vector $\mathbf{x}$ is denoted by $\mathbf{x}^T$.

When we define a function, we can also express it as a "mapping" by specifying its domain and range. For example, a function f that takes as input two integers s and a from two sets of integers S and A, and produces as output an integer s' that is also in the set S, can be written as

$$f(s,a) = s', \quad s, s' \in S, \quad a \in A$$

and, in terms of its mapping, can be specified as

$$f : S \times A \to S$$

Definition 1.1 (Argument of the maximum): *Suppose that a given function $f(x)$ has the maximum M. The set of values of x for which $f(x)$ attains M is expressed as*

$$\operatorname*{argmax}_x (f(x))$$

Example 1.1 *For the function $f(x) = \sin x$,*

$$\operatorname*{argmax}_x (\sin x) = \{360^\circ k + 90^\circ\}, \qquad k = 0, 1, 2, \ldots$$

□

Definition 1.2 (Signum function): *The signum function (also called the sign function), denoted by* sgn, *is defined as*

$$\operatorname{sgn}[u] = \begin{cases} +1 & \text{for } u \geq 0 \\ -1 & \text{for } u < 0 \end{cases}$$

□

Definition 1.3 (Gradient): *The gradient of a function* $f(\mathbf{w})$, *with respect to* $\mathbf{w} \in \mathbb{R}^{n\times 1}$, *is defined as*

$$\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}} = \begin{bmatrix} \dfrac{\partial f(\mathbf{w})}{\partial w_1} \\ \vdots \\ \dfrac{\partial f(\mathbf{w})}{\partial w_n} \end{bmatrix}$$

1.2 Vector and matrix operations

Definition 1.4 (Inner product): *The inner product of two vectors* $\mathbf{x} = [x_1 \dots x_n]^T \in \mathbb{R}^{n\times 1}$ *and* $\mathbf{y} = [y_1 \dots y_n]^T \in \mathbb{R}^{n\times 1}$, *denoted by* $\langle \mathbf{x} \cdot \mathbf{y} \rangle$, *is defined as*

$$\langle \mathbf{x} \cdot \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n = \mathbf{x}^T \mathbf{y}$$

□

Example 1.2 *Consider the following two vectors:*

$$\begin{aligned} \mathbf{u} &= [\ 1 \quad 1 \quad -1 \quad -1\ ]^T \\ \mathbf{v} &= [\ -1 \quad 1 \quad 1 \quad -1\ ]^T \end{aligned}$$

Their inner product is

$$\mathbf{u} \cdot \mathbf{v} = [\ 1 \quad 1 \quad -1 \quad -1\ ]^T \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix} = -1+1-1+1 = 0$$

Definition 1.5 (Outer product): *The outer product of two vectors* $\mathbf{u} = [u_1 \dots u_n]^T \in \mathbb{R}^{n\times 1}$ *and* $\mathbf{v} = [v_1 \dots v_n]^T \in \mathbb{R}^{n\times 1}$ *is defined as*

$$\mathbf{u}\mathbf{v}^T = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix} = \begin{bmatrix} u_1 v_1 & \cdots & u_1 v_n \\ \vdots & \ddots & \vdots \\ u_n v_1 & \cdots & u_n v_n \end{bmatrix}$$

□

Example 1.3 *Consider the two vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *in Example 1.2 above. Their outer product is:*

$$\mathbf{u}\mathbf{v}^T = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Definition 1.6 (Norm of a vector): *The Euclidean norm (also called the* l_2*-norm) of a vector* $\mathbf{x} \in \mathbb{R}^{n\times 1}$*, denoted by* $\|\mathbf{x}\|$*, is defined as*

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \sqrt{\langle \mathbf{x} \cdot \mathbf{x} \rangle} \equiv \sqrt{\mathbf{x}^T \mathbf{x}}$$

□

Definition 1.7 (Orthogonality): *Two vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *are said to be orthogonal if their inner product is zero, i.e.,*

$$\langle \mathbf{u} \cdot \mathbf{v} \rangle = \mathbf{u}^T \mathbf{v} = 0$$

□

For vectors $\mathbf{u}, \mathbf{v} \in R^{n\times 1}$ and a matrix $\mathbf{A} \in R^{n\times n}$, we have the following rules.

$$\begin{aligned} \mathbf{u}^T \mathbf{v} &= \mathbf{v}^T \mathbf{u} \\ \frac{\partial \left(\mathbf{u}^T \mathbf{v} \right)}{\partial \mathbf{v}} &= \mathbf{u} \\ \frac{\partial \left(\mathbf{u}^T \mathbf{A} \mathbf{u} \right)}{\partial \mathbf{u}} &= \left(\mathbf{A} + \mathbf{A}^T \right) \mathbf{u} \end{aligned}$$

Let $\mathbf{I}$ denote the identity matrix. For the special case of $\mathbf{A} = \mathbf{I}$, we have

$$\frac{\partial \left(\mathbf{u}^T \mathbf{A} \mathbf{u} \right)}{\partial \mathbf{u}} = \left(\mathbf{A} + \mathbf{A}^T \right) \mathbf{u} = \left(\mathbf{I} + \mathbf{I}^T \right) \mathbf{x} = 2\,\mathbf{u}$$

1.3 Eigenvalues and eigenvectors

Given a matrix $\mathbf{A}$ and a vector $\mathbf{x}$, the expression $\mathbf{y}=\mathbf{A}\mathbf{x}$ can be interpreted as $\mathbf{A}$ operating on $\mathbf{x}$ to produce an output vector $\mathbf{y}$. Such an operation involves $\mathbf{x}$ being scaled and/or rotated. For instance, given $\mathbf{A}=\begin{bmatrix} 0 & 1/2 \\ -1 & 1/2 \end{bmatrix}$ and $\mathbf{x}=\begin{bmatrix} 2 \\ 2 \end{bmatrix}$, we have

$$\mathbf{y}=\mathbf{A}\mathbf{x}=\begin{bmatrix} 0 & 1/2 \\ -1 & 1/2 \end{bmatrix}\begin{bmatrix} 2 \\ 2 \end{bmatrix}=\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Geometrically, we can interpret $\mathbf{y}$ as the result of $\mathbf{x}$ being rotated (by $-90°$) and scaled down in length (by a factor of $1/2$) by $\mathbf{A}$.

If we limit the operation of $\mathbf{A}$ on $\mathbf{x}$ to just scaling, then we can express this scaling (only) operation as $\mathbf{A}\mathbf{x}=\lambda\mathbf{x}$, where λ is the scaling factor.

Definition 1.8 (Eigenvectors and eigenvalues): *Consider the equation $\mathbf{A}\mathbf{x}=\lambda\mathbf{x}$. The non-zero vectors $\mathbf{x}$ are called the eigenvectors of $\mathbf{A}$, and the corresponding values of λ are called the eigenvalues of $\mathbf{A}$.* □

We can re-write $\mathbf{A}\mathbf{x}=\lambda\mathbf{x}$ above as $\lambda\mathbf{x}-\mathbf{A}\mathbf{x}=0$ or $(\lambda I-\mathbf{A})\mathbf{x}=0$. Existence of non-trivial solutions for $(\lambda I-\mathbf{A})\mathbf{x}=0$ requires that the determinant of $(\lambda I-\mathbf{A})$ be zero, i.e., $|\lambda I-\mathbf{A}|=0$, from which we can determine the eigenvalues of $\mathbf{A}$.

Example 1.4 *Find the eigenvalues and eigenvectors of the matrix*

$$\mathbf{A}=\begin{bmatrix} 3 & -2 \\ -1 & 4 \end{bmatrix}$$

Solution: For the given matrix, we have

$$|\lambda I-\mathbf{A}|=\begin{vmatrix} \lambda-3 & 2 \\ 1 & \lambda-4 \end{vmatrix}=(\lambda-3)(\lambda-4)-2=0$$

Solving for λ yields two values: $\lambda_1=5$ and $\lambda_2=2$ □

Once the eigenvalues are found, they can be substituted back into the equation $\mathbf{A}\mathbf{x}=\lambda\mathbf{x}$ to determine the eigenvectors. Consider the case of $\lambda_1=5$. Now

$$\mathbf{A}\mathbf{x}=\lambda\mathbf{x} \quad \Rightarrow \quad \begin{bmatrix} 3 & -2 \\ -1 & 4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}=5\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \Rightarrow \quad x_1=-x_2$$

Simply setting $x_2 = 1$ yields the eigenvector associated with λ_1 as

$$\mathbf{x}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Similarly, the eigenvector associated with λ_2 can be found as

$$\mathbf{x}_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

1.4 Random variables and probabilities

Definition 1.9 (Expected value, or expectation, of a random variable): *Suppose a random variable X can take value x_1 with probability p_1, value x_2 with probability p_2, and so on, up to value x_k with probability p_k. Then the expectation of X is defined as*

$$\mathbb{E}[X] = x_1 p_1 + x_2 p_2 + \cdots x_k p_k$$

$\mathbb{E}$ is also called the expectation operator. □

The expectation operator is linear with the following properties:

$$\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y] \tag{1.1}$$
$$\mathbb{E}[X+c] = \mathbb{E}[X] + c \tag{1.2}$$
$$\mathbb{E}[cX] = c\mathbb{E}[X] \tag{1.3}$$

where c is a real constant. □

Chapter 2

Method of Lagrange Multipliers

2.1 Constrained optimization problem

Solving a constrained optimization problem is about finding some optimal value of a given function, subject to certain constraint(s) on the values that the variables of the function may be allowed to take on.

Definition 2.1 *The basic class of constrained optimization problems involves only equality constraints. Such a problem is usually expressed in the form*

$$\begin{aligned} &\textit{Minimize:} \quad f(\mathbf{w}) \\ &\textit{Subject to:} \quad h_i(\mathbf{w}) = 0, \quad i = 1, \ldots, m \end{aligned}$$

where $\mathbf{w}$ *is a vector in* $\mathbb{R}^{n\times 1}$. *The function* $f(\cdot)$ *is called the* objective function *and the functions* $h_i(\cdot) = 0$ *are called the* constraints. *The optimal value of the function* $f(\cdot)$ *is called the* value of the optimization problem. *The value of* $\mathbf{w}$ *corresponding to the optimal* $f(\mathbf{w})$ *is called the* optimal solution, *and is denoted by* $\mathbf{w}_\circ$. □

If a certain objective function is to be maximized instead, such a maximization problem can be expressed as a minimization problem by the transformation

$$\max_{\mathbf{w}} f(\mathbf{w}) = -\min_{\mathbf{w}} [-f(\mathbf{w})]$$

The problem is then to minimize $-f(\mathbf{w})$ subject to the given constraints.

Example 2.1 *Consider a box whose surface area is fixed at some value* c. *Formulate the optimization problem for maximizing the volume of the box.*

Solution: The surface area of the box is: $2(wu + uv + vw) = c$. The volume of the box is: wuv. Maximizing the objective function wuv is equivalent to minimizing $-wuv$. Thus, the optimization problem can be stated as

$$\begin{aligned} \text{Minimize:} \quad & f(u,v,w) = -uvw \\ \text{Subject to:} \quad & h(u,v,w) = wu + uv + vw - c/2 = 0 \end{aligned}$$

□

The method of Lagrange multipliers offers a convenient way of formulating constrained optimization problems that makes them simpler to solve, and provides insight about the characteristics of the solutions (if they exist).

2.2 Lagrangian function and the Lagrange Theorem

The method of Lagrange multiplier is based on the following property concerning the gradient of the objective function and that of the constraints at the optimal solution. The property can stated as follows.

For the constrained optimization problem with the objective function $f(\mathbf{w})$ and equality constraints $h_i(\mathbf{w}), i = 1, \ldots, m$, at the optimal solution $\mathbf{w}_\circ$ the gradient of $f(\mathbf{w}_\circ)$ is parallel to the gradient of the constraint $h_i(\mathbf{w}_\circ)$; that is, there exist constants β_i such that

$$\left[\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}}\right]_{\mathbf{w}_\circ} + \sum_{i=1}^{m} \left[\beta_i \left(\frac{\partial h_i(\mathbf{w})}{\partial \mathbf{w}}\right)\right]_{\mathbf{w}_\circ} = \mathbf{0}$$

□

A detailed discussion on the proof for this proposition can be found in the book by Fletcher [5]. A convenient way to re-state the above property is to introduce an entity called the *Lagrangian function* (or simply, the Lagrangian).

Definition 2.2 (Lagrangian function and Lagrange multipliers): *Given an optimization problem with the objective function $f(\mathbf{w})$ and equality constraints $h_i(\mathbf{w}), i = 1, \ldots, m$, the Lagrangian function is defined as*

$$L(\mathbf{w}, \beta) = f(\mathbf{w}) + \sum_{i=1}^{m} \beta_i h_i(\mathbf{w})$$

where the coefficients β_i are the elements of the vector β, and are called the Lagrange multipliers. □

Using the Lagrangian function, the relationship between the gradient of $f(\mathbf{w})$ and the gradients of the constraints $h_i(\mathbf{w}) = 0$ can be expressed in a compact form as

$$\left.\frac{\partial L(\mathbf{w}, \beta)}{\partial \mathbf{w}}\right|_{\mathbf{w}_\circ} = \mathbf{0}$$

The following theorem of Lagrange characterizes the solution to the optimization problem with equality constraints.

Theorem 2.1 (Lagrange's Theorem): *For an optimization problem with objective function* $f(\mathbf{w})$ *and equality constraints* $h_i(\mathbf{w}) = 0, i = 1, \ldots, m$, *the necessary conditions for the existence of an optimal solution* $\mathbf{w}_\circ$ *are*

$$\begin{aligned} \frac{\partial L(\mathbf{w}, \beta)}{\partial \mathbf{w}} &= \mathbf{0} \\ \frac{\partial L(\mathbf{w}, \beta)}{\partial \beta_i} &= 0 \end{aligned}$$

□

Note that the second condition recovers the equality constraints $h_i(\cdot) = 0$. For simple problems, we can find the solutions manually using these conditions.

Example 2.2 *Consider a box whose surface area is fixed at some value* $c \neq 0$. *Find the dimensions* u, v, *and* w *that maximize the volume of the box.*

Solution: From Example 2.1, the optimization problem can be stated as:

$$\begin{aligned} \text{Minimize:} \quad & f(u,v,w) = -uvw \\ \text{Subject to:} \quad & h(u,v,w) = wu + uv + vw - c/2 = 0 \end{aligned}$$

The Lagrangian function is

$$\begin{aligned} L(u,v,w) &= f(u,v,w) + \beta\, h(u,v,w) \\ &= -uvw + \beta\left(wu + uv + vw - \frac{c}{2}\right) \end{aligned}$$

Applying the two conditions as stated in Lagrange's Theorem yields:

$$\begin{aligned} \frac{\partial L}{\partial u} = -vw + \beta(v+w) &= 0 \\ \frac{\partial L}{\partial v} = -wu + \beta(w+u) &= 0 \\ \frac{\partial L}{\partial w} = -uv + \beta(u+v) &= 0 \\ wu + uv + vw &= \frac{c}{2} \end{aligned}$$

The solution is

$$u = v = w = \sqrt{\frac{c}{6}}$$

and the maximum volume is

$$\max(wuv) = -\min(-wuv) = -\left(-\left(\frac{c}{6}\right)^{\frac{3}{2}}\right) = \left(\frac{c}{6}\right)^{\frac{3}{2}}$$

□

2.3 Primal problem

Lagrange's Theorem can be extended to deal with optimization problems involving both equality and inequality constraints. For inequality constraints, the standard form is $q(\cdot) \leq 0$. For constraints initially expressed as a "$\geq$" relationship, it is possible to transform it into the standard form by multiplying both sides by -1.

Definition 2.3 (Primal problem): *The standard form of a constrained problem (also called the primal problem), with m equality constrains and k inequality constraints, can be expressed as*

$$\begin{aligned} &\textit{Minimize:} && f(\mathbf{w}) \\ &\textit{Subject to:} && q_i(\mathbf{w}) \leq 0, \quad i = 1, \ldots, k \\ & && h_i(\mathbf{w}) = 0, \quad i = 1, \ldots, m \end{aligned}$$

□

The Lagrangian function for the primal problem (also called the *generalized Lagrangian function*) is defined as

$$L(\mathbf{w}, \alpha, \beta) = f(\mathbf{w}) + \sum_{i=1}^{k} \alpha_i q_i(\mathbf{w}) + \sum_{i=1}^{m} \beta_i h_i(\mathbf{w})$$

where α_i and β_i are the elements of the vectors $\boldsymbol{\alpha} \in \mathbb{R}^k$ and $\boldsymbol{\beta} \in \mathbb{R}^m$.

There are various techniques and algorithms for solving the primal problem. A good source of information on these techniques and algorithms is the book by Fletcher [5]. A detailed technical study of these techniques and algorithms is beyond the scope of this module. It is more relevant for us to understand the *characteristics* of the solution.

2.4 Karush-Kuhn-Tucker conditions and dual problem

The key results concerning the characteristics of a solution to the constrained optimization problem in the context of Lagrange multipliers are expressed in the Karush-Kuhn-Tucker conditions.

Theorem 2.2 (Kuhn-Tucker): *Consider the primal problem with the Lagrangian function as defined earlier. The sufficient and necessary condition for a point* $\mathbf{w} = \mathbf{w}_\circ$ *to be an optimal solution is the existence of* $\boldsymbol{\alpha} = \boldsymbol{\alpha}_\circ$ *and* $\boldsymbol{\beta} = \boldsymbol{\beta}_\circ$ *such that*

$$\begin{aligned}
\frac{\partial L(\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \mathbf{w}} &= \mathbf{0} \\
\frac{\partial L(\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \beta_i} &= 0 \\
\alpha_i q_i(\mathbf{w}) &= 0, \ i = 1, \ldots, k \\
q_i(\mathbf{w}) &\leq 0, \ i = 1, \ldots, k \\
\alpha_i &\geq 0, \ i = 1, \ldots, k
\end{aligned}$$

□

The conditions in the Kuhn-Tucker Theorem are also referred to as the *Karush-Kuhn-Tucker (or KKT) conditions.*

Remark 2.1 We note that the Kuhn-Tucker Theorem requires the optimization problem to be convex [5]. This requirement is to ensure the existence of an optimal solution. For our purpose in this module, we simply consider the problems of interest to be convex without giving any formal proof. □

The KKT conditions enable us to reformulate the primal problem into an alternative form called the *dual problem.* The main advantage of reformulating a primal problem into the corresponding dual problem is that the dual problem is usually less complex and thus easier to solve. Once we solve the dual problem, we also obtain the solution to the primal problem (and vice versa). The dual problem can be stated as follows.

Definition 2.4 (Dual problem): *The dual problem of constrained optimization can in general be expressed as*

$$\begin{aligned}
\textit{Maximize:} \quad & L(\mathbf{w}, \alpha, \beta) \\
\textit{Subject to:} \quad & \frac{\partial L(\mathbf{w}, \alpha, \beta)}{\partial \mathbf{w}} = 0 \\
& \alpha \geq 0
\end{aligned}$$

□

Note that the *value of the optimization problem* (see Definition 2.1) for the primal problem is $f(\mathbf{w}_\circ)$ while that for the dual problem is $L(\mathbf{w}_\circ, \alpha_\circ, \beta_\circ)$, with the former to be minimized and the latter maximized. The reason that the primal and dual problems are equivalent is due to the fact that, at the respective optimal solutions, $f(\mathbf{w}_\circ) = L(\mathbf{w}_\circ, \alpha_\circ, \beta_\circ)$. We will not discuss here the proof of this fact or the general procedure for formulating the dual problem based the primal problem. An in-depth discussion on such a procedure is available in various textbooks, such as on the one by Fletcher [5]. We will just illustrate, through a simple example, the process of reformulating a given primal problem into the corresponding dual problem.

Example 2.3 *Consider a set of data* $\{(\mathbf{x}_i, d_i)\}$*, where* $\mathbf{x}_i \in \mathbb{R}^{n\times 1}$ *and* d_i *is a binary label* $(+1$ *or* $-1)$ *for* $\mathbf{x}_i$*, with* $i = 1, 2, \ldots, N$*. Let* $\mathbf{w} \in \mathbb{R}^{n\times 1}$ *and* $b \in \mathbb{R}$*. Derive the dual problem for the following primal problem.*

$$\begin{aligned} \textit{Minimizing :} \quad & f(\mathbf{w}) = \tfrac{1}{2}\mathbf{w}^T\mathbf{w} \\ \textit{Subject to :} \quad & d_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1, \quad i = 1, \ldots, N \end{aligned}$$

Solution: We first write the constraints in the standard form so that

$$d_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1$$

becomes

$$q_i = -d_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) + 1 \leq 0$$

Now the Lagrangian function is

$$\begin{aligned} L(\mathbf{w}, b, \alpha) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i \left(d_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1\right) \\ &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i d_i \mathbf{w}^T\mathbf{x}_i - b\sum_{i=1}^{N} \alpha_i d_i + \sum_{i=1}^{N} \alpha_i \end{aligned}$$

The KKT conditions are

$$\begin{aligned} &\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{0} \\ &\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = 0 \quad (b \text{ is a variable that affects the value of the Lagrangian}) \\ &d_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 \\ &\alpha_i\left(d_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1\right) = 0 \\ &\alpha_i \geq 0 \end{aligned}$$

Now

$$\begin{aligned}
\frac{\partial L(\mathbf{w},b,\alpha)}{\partial \mathbf{w}} &= \frac{1}{2}\frac{\partial\left(\mathbf{w}^T\mathbf{w}\right)}{\partial \mathbf{w}} - \sum_{i=1}^{N}\alpha_i d_i\left(\frac{\partial\left(\mathbf{w}^T\mathbf{x}_i\right)}{\partial \mathbf{w}}\right) \\
&= \frac{1}{2}(2\mathbf{w}) - \sum_{i=1}^{N}\alpha_i d_i\left(\frac{\partial\left(\mathbf{x}_i^T\mathbf{w}\right)}{\partial \mathbf{w}}\right) \quad \left(\text{Since } \mathbf{u}^T\mathbf{v} = \mathbf{v}^T\mathbf{u}\right) \\
&= \mathbf{w} - \sum_{i=1}^{N}\alpha_i d_i\mathbf{x}_i \quad \left(\text{Since } \frac{\partial\left(\mathbf{u}^T\mathbf{v}\right)}{\partial \mathbf{v}} = \mathbf{u}\right) \\
&= \mathbf{0}
\end{aligned}$$

Therefore,

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i\, d_i\,\mathbf{x}_i$$

With respect to the variable b, we have

$$\frac{\partial L(\mathbf{w},b,\alpha)}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{N}\alpha_i\, d_i = 0$$

So

$$\frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}\left(\sum_{i=1}^{N}\alpha_i\, d_i\,\mathbf{x}_i^T\right)\left(\sum_{j=1}^{N}\alpha_j\, d_j\,\mathbf{x}_j\right) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\,\alpha_j\, d_i\, d_j\,\mathbf{x}_i^T\mathbf{x}_j$$

and

$$\sum_{i=1}^{N}\alpha_i\, d_i\,\mathbf{w}^T\mathbf{x}_i = \sum_{i=1}^{N}\alpha_i\, d_i\left(\sum_{j=1}^{N}\alpha_j\, d_j\,\mathbf{x}_j^T\right)\mathbf{x}_i = \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\,\alpha_j\, d_i\, d_j\,\mathbf{x}_i^T\mathbf{x}_j$$

Substituting the above expressions into the Lagrangian function yields

$$L(\mathbf{w},b,\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\,\alpha_j\, d_i\, d_j\,\mathbf{x}_i^T\mathbf{x}_j$$

Since the only unknown in the above expression for $L(\mathbf{w},b,\alpha)$ is α_i, we use the symbol $Q(\alpha)$ to represent this expression. Hence, the dual problem can be written as

$$\begin{aligned}
\text{Maximize:} \quad & Q(\alpha) \equiv L(\mathbf{w},b,\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\,\alpha_j\, d_i\, d_j\,\mathbf{x}_i^T\mathbf{x}_j \\
\text{Subject to:} \quad & \sum_{i=1}^{N}\alpha_i\, d_i = 0 \\
& \alpha_i \geq 0
\end{aligned}$$

□

We note that in the example above, the dual problem only involves the Lagrange multipliers α_i as the only unknowns (since $\mathbf{x}_i$ and d_i are just values in the given data set). This is the main reason that the dual problem is usually less complex to solve than the primal problem. In fact, when we apply the method of Lagrange multiplier to the construction of support vector machines, solving the dual problem (instead of the primal problem) becomes an even more desirable alternative because the values of many (if not most) α_i turn out to be zero.

Another benefit of expressing the primal problem in the dual form is that the objective function $Q(\alpha)$ in the dual problem contains a useful mathematical expression called a *kernel*. In the above example, the expression $\mathbf{x}_i^T\mathbf{x}_j$ is in fact a type of kernel called a *linear kernel*. The next chapter discusses the purpose and construction of kernels.

Chapter 3

Feature Space and Kernels

3.1 Input space and feature space

When we attempt to classify some given set of data, we can represent the data in various ways. For instance, a point can be represented in a two-dimensional space by the coordinates (x, y); it can also be represented in a three-dimensional space by the coordinates (x, y, z), with $z = 0$ for all points.

When we build learning machines that process input data to generate some decision as the output of the machines, it is sometimes convenient to transform the input data from its original space into another space and perform the data processing in the latter. Usually, the space in which the data point is originally represented is referred to as the *input space*, while the space into which the original data are transformed is referred to as the *feature space*. In general, a mapping from the input space of dimension n to a feature space of dimension l can be expressed as

$$\mathbf{x} = [x_1, \ldots, x_n]^T \;\mapsto\; \varphi(\mathbf{x}) = [\varphi_1(\mathbf{x}), \ldots, \varphi_l(\mathbf{x})]^T$$

Example 3.1 *Consider Newton's law of gravitation*

$$f(m_1, m_2, r) = C\left(\frac{m_1 m_2}{r^2}\right)$$

For a simple illustration, assume $r = 1$. Map the function $f(m_1, m_2)$ into a new function $g(u, v)$ using the nonlinear mapping

$$\mathbf{x} = [m_1, m_2]^T \mapsto \varphi(\mathbf{x}) = [\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x})]^T = [\ln m_1, \ln m_2]^T = [u, v]^T$$

Solution: From the given mapping, we have

$$u \;=\; \varphi_1(\mathbf{x}) = \ln m_1$$

$$v = \varphi_2(\mathbf{x}) = \ln m_2$$

To map the function $f(m_1, m_2)$ into $g(u, v)$, we apply the natural log on f, i.e.,

$$\begin{aligned} \ln f(m_1, m_2) &= \ln\left(C\left(\frac{m_1 m_2}{r^2}\right)\right) \\ &= \ln C + \ln m_1 + \ln m_2 - 2\ln 1 \\ &= u + v + c \qquad (\text{where } c = \ln C) \\ &= g(u, v) \end{aligned}$$

We note that $g(u, v)$ is linear with respect to the new coordinates (u, v). □

The above example demonstrates that a nonlinear function in the input space may be transformed into a linear function in the feature space through a suitable nonlinear mapping. In this particular example, the dimension of the feature space happens to be the same as that of the input space. In practice, it is common to find that the dimension of the feature space is much greater than that of the input space. The rationale for having a higher dimensionality for the feature space comes from the Cover Theorem [3], which states that the probability of the transformed data points being linearly separable (in the feature space) increases when the data points are nonlinearly mapped from a lower-dimension input space to a higher dimensional feature space.

3.2 Kernels and their characterization

Definition 3.1 (Kernel): *A kernel is a function K such that for two vectors $\mathbf{x}$ and $\mathbf{y}$ in the input space,*

$$K(\mathbf{x}, \mathbf{y}) = \langle \varphi(\mathbf{x}) \cdot \varphi(\mathbf{y}) \rangle = \varphi^T(\mathbf{x}) \varphi(\mathbf{y})$$

where φ is a mapping from the input space to (an inner product) feature space. □

A property of a kernel is that it is symmetric, i.e., $K(\mathbf{x}, \mathbf{y}) = K(\mathbf{y}, \mathbf{x})$. If φ is the identity mapping, i.e., $\varphi(\mathbf{x}) = \mathbf{x}$, then the kernel

$$K(\mathbf{x}, \mathbf{y}) = \langle \varphi(\mathbf{x}) \cdot \varphi(\mathbf{y}) \rangle = \mathbf{x}^T \mathbf{y}$$

is called a *linear kernel*.

Example 3.2 *Consider the mapping*

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \\ x_2 \end{bmatrix} \mapsto \varphi(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{bmatrix}$$

(i) Determine the kernel $K(\mathbf{x},\mathbf{y})$.

(ii) Calculate the value of the kernel if $\mathbf{x} = [1\ 2]^T$ *and* $\mathbf{y} = [3\ 4]^T$.

Solution: (i) The kernel defined by this mapping is

$$\begin{aligned} K(\mathbf{x},\mathbf{y}) &= \varphi^T(\mathbf{x})\varphi(\mathbf{y}) \\ &= \begin{bmatrix} 1 & x_1^2 & \sqrt{2}x_1x_2 & x_2^2 & \sqrt{2}x_1 & \sqrt{2}x_2 \end{bmatrix} \begin{bmatrix} 1 \\ y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \\ \sqrt{2}y_1 \\ \sqrt{2}y_2 \end{bmatrix} \\ &= 1 + x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 + 2x_1y_1 + 2x_2y_2 \end{aligned}$$

(ii) With $\mathbf{x} = [1\ 2]^T$ and $\mathbf{y} = [3\ 4]^T$, the value of the kernel is

$$\begin{aligned} K(\mathbf{x},\mathbf{y}) &= 1 + x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 + 2x_1y_1 + 2x_2y_2 \\ &= 1 + 1^2\cdot 3^2 + 2\cdot 1\cdot 2\cdot 3\cdot 4 + 2^2\cdot 4^2 + 2\cdot 1\cdot 3 + 2\cdot 2\cdot 4 \\ &= 1 + 9 + 48 + 64 + 6 + 16 = 144 \end{aligned}$$

□

If we do not know the mapping φ explicitly, we can still find a kernel for a given set of data by first selecting a candidate expression for the kernel and then checking whether this expression is a suitable kernel. This approach relies on the so-called Mercer's condition, which utilizes a matrix (built using the kernel and the given data)

called the Gram matrix.

Definition 3.2 (Gram matrix): *Given a set of data points* $\{\mathbf{x}_i\}, i=1,\ldots,N$, *and a kernel* K, *the Gram matrix, denoted by* $\mathbf{K}$, *is defined as*

$$\mathbf{K} = \begin{bmatrix} K(\mathbf{x}_1,\mathbf{x}_1) & \ldots & K(\mathbf{x}_1,\mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_N,\mathbf{x}_1) & \ldots & K(\mathbf{x}_N,\mathbf{x}_N) \end{bmatrix} \in R^{N\times N}$$

□

The result stated in the following theorem is often referred to as *Mercer's condition*. It characterizes a property that a function must have in order for it to be considered as an admissible kernel.

Theorem 3.1 (Mercer's condition[1]): *Let* $K(\mathbf{x},\mathbf{y})$ *be a symmetric function on the input space. Then* $K(\mathbf{x},\mathbf{y})$ *is an admissible kernel if and only if its Gram matrix* $\mathbf{K}$ *is positive semi-definite; that is,* $\mathbf{K}$ *has non-negative eigenvalues.* □

Example 3.3 *Consider the set of data in the table below and the kernel* $K(\mathbf{x}_i,\mathbf{x}_j) = (1+\mathbf{x}_i^T\mathbf{x}_j)^2$. *Determine the Gram matrix.*

i	1	2	3	4
$\mathbf{x}_i$	$[-1,-1]^T$	$[-1,+1]^T$	$[+1,-1]^T$	$[+1,+1]^T$

Solution:

$$\begin{aligned}
K(\mathbf{x}_1,\mathbf{x}_1) &= \left(1+\mathbf{x}_1^T\mathbf{x}_1\right)^2 = \left(1+\begin{bmatrix} -1 & -1 \end{bmatrix}\begin{bmatrix} -1 \\ -1 \end{bmatrix}\right)^2 = 9 \\
K(\mathbf{x}_1,\mathbf{x}_2) &= \left(1+\mathbf{x}_1^T\mathbf{x}_2\right)^2 = \left(1+\begin{bmatrix} -1 & -1 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \end{bmatrix}\right)^2 = 1 = K(\mathbf{x}_2,\mathbf{x}_1) \\
K(\mathbf{x}_1,\mathbf{x}_3) &= \left(1+\mathbf{x}_1^T\mathbf{x}_3\right)^2 = \left(1+\begin{bmatrix} -1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right)^2 = 1 = K(\mathbf{x}_3,\mathbf{x}_1) \\
K(\mathbf{x}_1,\mathbf{x}_4) &= \left(1+\mathbf{x}_1^T\mathbf{x}_4\right)^2 = \left(1+\begin{bmatrix} -1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right)^2 = 1 = K(\mathbf{x}_4,\mathbf{x}_1)
\end{aligned}$$

[1] A detailed exposition on the proof of this theorem can be found in [4].

$$
\begin{aligned}
K(\mathbf{x}_2,\mathbf{x}_2) &= \left(1+\mathbf{x}_2^T\mathbf{x}_2\right)^2 = \left(1+\begin{bmatrix} -1 & 1 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \end{bmatrix}\right)^2 = 9 \\
K(\mathbf{x}_2,\mathbf{x}_3) &= \left(1+\mathbf{x}_2^T\mathbf{x}_3\right)^2 = \left(1+\begin{bmatrix} -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right)^2 = 1 = K(\mathbf{x}_3,\mathbf{x}_2) \\
K(\mathbf{x}_2,\mathbf{x}_4) &= \left(1+\mathbf{x}_2^T\mathbf{x}_4\right)^2 = \left(1+\begin{bmatrix} -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right)^2 = 1 = K(\mathbf{x}_4,\mathbf{x}_2) \\
K(\mathbf{x}_3,\mathbf{x}_3) &= \left(1+\mathbf{x}_3^T\mathbf{x}_3\right)^2 = \left(1+\begin{bmatrix} 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right)^2 = 9 \\
K(\mathbf{x}_3,\mathbf{x}_4) &= \left(1+\mathbf{x}_3^T\mathbf{x}_4\right)^2 = \left(1+\begin{bmatrix} 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right)^2 = 1 = K(\mathbf{x}_4,\mathbf{x}_3) \\
K(\mathbf{x}_4,\mathbf{x}_4) &= \left(1+\mathbf{x}_4^T\mathbf{x}_4\right)^2 = \left(1+\begin{bmatrix} 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right)^2 = 9
\end{aligned}
$$

Therefore, the Gram matrix is

$$
\mathbf{K} = \begin{bmatrix} K(\mathbf{x}_1,\mathbf{x}_1) & \dots & K(\mathbf{x}_1,\mathbf{x}_4) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_4,\mathbf{x}_1) & \dots & K(\mathbf{x}_4,\mathbf{x}_4) \end{bmatrix} = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}
$$

□

Example 3.4 *Consider the set of data in the table below and the kernel candidate* $K(\mathbf{x}_i,\mathbf{x}_j) = \tanh(\mathbf{x}_i^T\mathbf{x}_j - 1)$.

i	$\mathbf{x}_i$
1	$[-1,-1]^T$
2	$[-1,+1]^T$
3	$[+1,-1]^T$
4	$[+1,+1]^T$

(i) Set up the equation that can be solved to obtain the eigenvalues of the Gram matrix associated with this kernel candidate.

(ii) Determine whether the kernel candidate satisfies Mercer's condition. (Note: You might need to use some software, such as MATLAB, to find the eigenvalues.)

Solution: (i) We first set up the Gram matrix.

$$
\begin{aligned}
K(\mathbf{x}_1,\mathbf{x}_1) &= \tanh\left(\mathbf{x}_1^T\mathbf{x}_1-1\right)=\tanh\left(\begin{bmatrix}-1 & -1\end{bmatrix}\begin{bmatrix}-1\\-1\end{bmatrix}-1\right)=0.7616\\
K(\mathbf{x}_1,\mathbf{x}_2) &= \tanh\left(\mathbf{x}_1^T\mathbf{x}_2-1\right)=\tanh\left(\begin{bmatrix}-1 & -1\end{bmatrix}\begin{bmatrix}-1\\1\end{bmatrix}-1\right)=-0.7616\\
K(\mathbf{x}_1,\mathbf{x}_3) &= \tanh\left(\mathbf{x}_1^T\mathbf{x}_3-1\right)=\tanh\left(\begin{bmatrix}-1 & -1\end{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix}-1\right)=-0.7616\\
K(\mathbf{x}_1,\mathbf{x}_4) &= \tanh\left(\mathbf{x}_1^T\mathbf{x}_4-1\right)=\tanh\left(\begin{bmatrix}-1 & -1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}-1\right)=-0.9951\\
K(\mathbf{x}_2,\mathbf{x}_2) &= \tanh\left(\mathbf{x}_2^T\mathbf{x}_2-1\right)=\tanh\left(\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}-1\\1\end{bmatrix}-1\right)=0.7616\\
K(\mathbf{x}_2,\mathbf{x}_3) &= \tanh\left(\mathbf{x}_2^T\mathbf{x}_3-1\right)=\tanh\left(\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix}-1\right)=-0.9951\\
K(\mathbf{x}_2,\mathbf{x}_4) &= \tanh\left(\mathbf{x}_2^T\mathbf{x}_4-1\right)=\tanh\left(\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}-1\right)=-0.7616
\end{aligned}
$$

$$
\begin{aligned}
K(\mathbf{x}_3,\mathbf{x}_3) &= \tanh\left(\mathbf{x}_3^T\mathbf{x}_3-1\right)=\tanh\left(\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix}-1\right)=0.7616\\
K(\mathbf{x}_3,\mathbf{x}_4) &= \tanh\left(\mathbf{x}_3^T\mathbf{x}_4-1\right)=\tanh\left(\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}-1\right)=-0.7616\\
K(\mathbf{x}_4,\mathbf{x}_4) &= \tanh\left(\mathbf{x}_4^T\mathbf{x}_4-1\right)=\tanh\left(\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}-1\right)=0.7616
\end{aligned}
$$

The Gram matrix is

$$\mathbf{K} = \begin{bmatrix} K(\mathbf{x}_1,\mathbf{x}_1) & \dots & K(\mathbf{x}_1,\mathbf{x}_4) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_4,\mathbf{x}_1) & \dots & K(\mathbf{x}_4,\mathbf{x}_4) \end{bmatrix}$$

$$= \begin{bmatrix} 0.7616 & -0.7616 & -0.7616 & -0.9951 \\ -0.7616 & 0.7616 & -0.9951 & -0.7616 \\ -0.7616 & -0.9951 & 0.7616 & -0.7616 \\ -0.9951 & -0.7616 & -0.7616 & 0.7616 \end{bmatrix}$$

The eigenvalues of $\mathbf{K}$ are determined by solving the equation

$$|\lambda\mathbf{I}-\mathbf{K}| = \left| \lambda \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.7616 & -0.7616 & -0.7616 & -0.9951 \\ -0.7616 & 0.7616 & -0.9951 & -0.7616 \\ -0.7616 & -0.9951 & 0.7616 & -0.7616 \\ -0.9951 & -0.7616 & -0.7616 & 0.7616 \end{bmatrix} \right|$$

$$= \begin{vmatrix} \lambda-0.7616 & 0.7616 & 0.7616 & 0.9951 \\ 0.7616 & \lambda-0.7616 & 0.9951 & 0.7616 \\ 0.7616 & 0.9951 & \lambda-0.7616 & 0.7616 \\ 0.9951 & 0.7616 & 0.7616 & \lambda-0.7616 \end{vmatrix} = 0$$

(ii) The eigenvalues of the Gram matrix $\mathbf{K}$ are found to be: $\lambda_1 = -1.7567 < 0$, $\lambda_2 = 1.7567$, $\lambda_3 = 1.7567$, and $\lambda_4 = 1.2897$. Since one of the eigenvalues is negative, this kernel candidate is not admissible. □

In the context of the theory of support vector machines, the ideal outcome of transforming a set of data points $\{\mathbf{x}_i\}$ from the input space into a feature space via the mapping φ is that the resulting set $\{\varphi(\mathbf{x}_i)\}$ is linearly separable in the feature space[2]. The transformation φ is then used in the construction of a kernel K. To achieve this φ needs to be properly chosen. Unfortunately, currently there is no systematic way of determining a suitable φ for a given data set.

To circumvent this difficulty, we can choose a candidate expression for K first then

[2]If the transformed data are still not linearly separable in the feature space, then the approach of *soft margin* (see Problem 5.2 in Chapter 5) provides a possible solution approach in the design of a workable SVM.

use Mercer's condition to check whether such an expression is an admissible kernel. This (indirect) way of finding an admissible kernel is referred to in the support vector machine literature as the *kernel trick*.

Chapter 4

Margins and Optimal Hyperplane

4.1 Training data and classification

Definition 4.1 (Example): *An example consists of an n-dimensional vector* $\mathbf{x} = [x_1, \ldots, x_n]^T \in R^{n\times 1}$ *together with a label* $d \in \{-1, +1\}$*, and is written as* $(\mathbf{x}, d)$. □

Definition 4.2 (Training set): *A training set is a set of* N *examples, i.e.,*

$$S = \left\{ (\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), \ldots, (\mathbf{x}_N, d_N) \right\}.$$

□

A trivial training set is one in which all examples have the same label, i.e., $d_i = d_j, i \neq j$, with $i, j = 1, \ldots, N$.

To classify a training set S correctly is to find a hyperplane defined by the equation

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0 \quad \text{(Note: } \mathbf{x} \text{ here is just a variable.)}$$

such that, for any example $(\mathbf{x}_i, d_i)$ in a training set S, we have

$$\operatorname{sgn}\left[\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b\right] = d_i \quad \text{or} \quad \operatorname{sgn}\left[d_i \left(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b\right)\right] = 1 \tag{4.1}$$

We often use $(\mathbf{w}, b)$ to denote a hyperplane.

4.2 Functional margin and geometric margin

Definition 4.3 (Functional margin of an example): *The functional margin of an example*

$(\mathbf{x}_i, d_i)$, denoted by γ_i^f, is defined with respect to a hyperplane $(\mathbf{w}, b)$ as

$$\gamma_i^f = d_i\Big(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b\Big)$$

□

Definition 4.4 (Geometric margin of an example): *The geometric margin of an example $(\mathbf{x}_i, d_i)$, denoted by γ_i^g, is defined with respect to a hyperplane $(\mathbf{w}, b)$ as*

$$\gamma_i^g = d_i \left(\left\langle \frac{1}{\|\mathbf{w}\|} \mathbf{w} \cdot \mathbf{x}_i \right\rangle + \frac{1}{\|\mathbf{w}\|} b \right)$$

□

In other words, the geometric margin of an example is the functional margin of that example with a normalized weight vector.

Example 4.1 *Show that the geometric margin of an example $(\mathbf{x}_i, d_i)$ with respect to a given hyperplane $(\mathbf{w}, b)$ is the Euclidean distance from the point defined by $\mathbf{x}_i$ to the hyperplane $(\mathbf{w}, b)$.*

Solution: We first show that $\frac{\mathbf{w}}{\|\mathbf{w}\|}$ is a unit vector perpendicular to the hyperplane $(\mathbf{w}, b)$. Consider two points $\mathbf{y}_1$ and $\mathbf{y}_2$ on the hyperplane $(\mathbf{w}, b)$. By definition, we have

$$\begin{aligned} \langle \mathbf{w} \cdot \mathbf{y}_1 \rangle + b &= \mathbf{w}^T \mathbf{y}_1 + b = 0 \\ \langle \mathbf{w} \cdot \mathbf{y}_2 \rangle + b &= \mathbf{w}^T \mathbf{y}_2 + b = 0 \end{aligned}$$

The vector from $\mathbf{y}_1$ to $\mathbf{y}_2$, denoted by $\mathbf{z}$, is $\mathbf{z} = \mathbf{y}_2 - \mathbf{y}_1$ (which is on the hyperplane). Now

$$\begin{aligned} \langle \mathbf{w} \cdot \mathbf{z} \rangle &= \langle \mathbf{w} \cdot (\mathbf{y}_2 - \mathbf{y}_1) \rangle = \langle \mathbf{w} \cdot \mathbf{y}_2 \rangle - \langle \mathbf{w} \cdot \mathbf{y}_1 \rangle \\ &= \mathbf{w}^T \mathbf{y}_2 - \mathbf{w}^T \mathbf{y}_1 = -b - (-b) \\ &= 0 \end{aligned}$$

Hence, $\mathbf{w}$ is perpendicular to the hyperplane $(\mathbf{w}, b)$.

Suppose that we draw a line from $\mathbf{x}_i$ perpendicular to the hyperplane so that it intersects the hyperplane at the point $\mathbf{y}_p$. Then we can express $\mathbf{x}_i$ as

$$\mathbf{x}_i = \mathbf{y}_p + r_i \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \right)$$

where $|r_i|$ is the distance from the point $\mathbf{x}_i$ to the hyperplane. Since $\mathbf{y}_p$ is on the hyperplane, we have $\mathbf{w}^T\mathbf{y}_p + b = 0$, or, $\mathbf{w}^T\mathbf{y}_p = -b$. Now

$$\begin{aligned}\langle \mathbf{w}\cdot\mathbf{x}_i\rangle + b &= \mathbf{w}^T\mathbf{x}_i + b = \mathbf{w}^T\left(\mathbf{y}_p + r_i\left(\frac{\mathbf{w}}{\|\mathbf{w}\|}\right)\right) + b \\ &= \mathbf{w}^T\mathbf{y}_p + r_i\left(\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|}\right) + b = -b + r_i\left(\frac{\|\mathbf{w}\|^2}{\|\mathbf{w}\|}\right) + b \\ &= r_i\|\mathbf{w}\|\end{aligned}$$

So

$$r_i = \frac{\langle \mathbf{w}\cdot\mathbf{x}_i\rangle + b}{\|\mathbf{w}\|} = \left\langle \frac{1}{\|\mathbf{w}\|}\mathbf{w}\cdot\mathbf{x}_i \right\rangle + \frac{b}{\|\mathbf{w}\|}$$

From the definition of the geometric margin of $(\mathbf{x}_i, d_i)$, we have $\gamma_i^g = d_i r_i$. Since $d_i = \pm 1$, and d_i and r_i will have the same sign (for a correctly classified example), we can conclude that $\gamma_i^g = |r_i|$, which is the Euclidean distance from $\mathbf{x}_i$ to the hyperplane $(\mathbf{w}, b)$. □

By definition, we have

$$\gamma_i^g = d_i\left(\left\langle \frac{1}{\|\mathbf{w}\|}\mathbf{w}\cdot\mathbf{x}_i \right\rangle + \frac{1}{\|\mathbf{w}\|}b\right) = \frac{1}{\|\mathbf{w}\|}\Big(d_i\left(\langle \mathbf{w}\cdot\mathbf{x}_i\rangle + b\right)\Big) = \frac{\gamma_i^f}{\|\mathbf{w}\|}$$

It is clear that the functional margin and the geometric margin are equal when the weight vector is a unit vector.

Definition 4.5 (Functional margin of a training set): *For a training set S with N training examples, the functional margin of S with respect to a given hyperplane is the minimum of all the functional margins of the individual examples, i.e.,*

$$\gamma^f = \min_{1\le i\le N}\left\{\gamma_i^f\right\}.$$

□

Definition 4.6 (Geometric margin of a training set): *For a training set S with N training examples, the geometric margin of S with respect to a given hyperplane is the minimum of all the geometric margins of the individual examples, i.e.,*

$$\gamma^g = \min_{1\le i\le N}\left\{\gamma_i^g\right\}.$$

□

For a given training set S, there may exist many possible hyperplanes that correctly

classify S. Each hyperplane defines both a functional margin and a geometric margin for the set S.

Definition 4.7 (Margin of a training set): *The margin of a training set is the maximum geometric margin over all hyperplanes.* □

Definition 4.8 (Optimal hyperplane): *The optimal hyperplane of a training set is the hyperplane associated with the margin of that training set.* □

The equation representing a hyperplane is invariant when scaled by an arbitrary real constant. For instance, the hyperplane described by the equation

$$\langle \mathbf{w}_1 \cdot \mathbf{x} \rangle + b_1 = 0$$

can also be described by

$$\langle c\mathbf{w}_1 \cdot \mathbf{x} \rangle + cb_1 = 0$$

where c is a scalar. Let $g(\mathbf{x}) = \langle c\mathbf{w}_1 \cdot \mathbf{x} \rangle + cb_1$. For an example $(\mathbf{x}_i, d_i)$, the functional margin is

$$\gamma_i^f = d_i\, g(\mathbf{x}_i) = d_i \Big(\langle c\mathbf{w}_1 \cdot \mathbf{x}_i \rangle + cb_1 \Big)$$

where d_i (with a value of either $+1$ or -1) is the label for the vector $\mathbf{x}_i$. If the hyperplane correctly classifies a training set S, then $(\langle c\mathbf{w}_1 \cdot \mathbf{x}_i \rangle + cb_1)$ has the same sign as d_i. For any given $\mathbf{w}_1$ and b_1, we can always find some value for c such that

$$d_i \Big(\langle c\mathbf{w}_1 \cdot \mathbf{x}_i \rangle + cb_1 \Big) = 1\,.$$

In other words, we can always find some value for c such that

$$g(\mathbf{x}_i) = \langle c\mathbf{w}_1 \cdot \mathbf{x}_i \rangle + cb_1 = \begin{cases} +1 & \text{if } \mathbf{x}_i \text{ has the label } d_i = +1 \\ -1 & \text{if } \mathbf{x}_i \text{ has the label } d_i = -1 \end{cases}$$

Now let $\mathbf{w}_2 = c\mathbf{w}_1$ and $b_2 = cb_1$. Then we have

$$g(\mathbf{x}_i) = \langle \mathbf{w}_2 \cdot \mathbf{x}_i \rangle + b_2 = \pm 1$$

For a general expression, we can drop the subscript 2 and simply write

$$g(\mathbf{x}_i) = \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b = \pm 1$$

Thus,

$$d_i\, g(\mathbf{x}_i) = d_i \Big(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \Big) = 1$$

This expression implies that, for any given example $(\mathbf{x}_i, d_i)$ and a hyperplane $(\mathbf{w}, b)$, we can always scale $\mathbf{w}$ and b to give the example a functional margin of 1.

Remark 4.1 In the construction of support vector machines, we deliberately make the functional margin of a training set S to be 1, and refer to the vector $\mathbf{x}_i$ that gives such a functional margin as a *support vector*. Specifically, for a support vector $\mathbf{x}_s$ with the label d_s, the functional margin of $(\mathbf{x}_s, d_s)$, with respect to the optimal hyperplane $(\mathbf{w}^T, b_\circ)$, is

$$\gamma_s^f = d_s g(\mathbf{x}_s) = d_s\left(\mathbf{w}_\circ^T \mathbf{x}_s + b_\circ\right) = 1$$

Since, by definition, a support vector represents an example that is nearest to the optimal hyperplane (i.e., at the edge of the margin of separation), the functional margin of the support vector is the minimum among the function margins of all the examples in the training set S, i.e.,

$$\gamma_s^f = \min\left\{\gamma_i^f\right\}, \text{ for all } (\mathbf{x}_i, d_i) \in S$$

Consequently, for an example $\mathbf{x}_i$ in the training set that is *not* a support vector, we have

$$\gamma_i^f = d_i g(\mathbf{x}_i) = d_i\left(\mathbf{w}_\circ^T \mathbf{x}_i + b_\circ\right) > 1$$

Therefore, the values of $\mathbf{w}_\circ$ and $b_\circ$ that define the optimal hyperplane must satisfy the constraint

$$\gamma_i^f = d_i g(\mathbf{x}_i) = d_i\left(\mathbf{w}_\circ^T \mathbf{x}_i + b_\circ\right) \geq 1, \text{ for all } (\mathbf{x}_i, d_i) \in S$$

□

Example 4.2 *Consider a training set S with only one example: $(\mathbf{x}_1, d_1) = ([1\ 4]^T, +1)$, and a hyperplane defined by $(\mathbf{w}, b) = ([5\ 3]^T, 6)$.*

(i) Determine the functional margin and the geometric margin of the example with respect to the given hyperplane.

(ii) Determine the weight vector that results in the example having a functional margin of 1 with respect to the given hyperplane. Calculate the geometric margin of the example associated with this new weight vector. Comment on the result of the geometric margin calculation.

(iii) Find a hyperplane that results in the example having a geometric margin of $\frac{1}{\sqrt{2}}$ and a functional margin of 1.

Solution: The information we have are

$$\mathbf{w} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}, \quad \mathbf{x}_1 = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \quad d_1 = +1, \quad b = 6$$

(i) Now

$$g(\mathbf{x}_1) = \mathbf{w}^T \mathbf{x}_1 + b = \begin{bmatrix} 5 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 6 = 23$$

Hence the functional margin and the geometric margin of the example are

$$\begin{aligned} \gamma_1^f &= d_1 \, g(\mathbf{x}_1) = 1 \times 23 = 23 \\ \gamma_1^g &= \frac{\gamma_1^f}{\|\mathbf{w}\|} = \frac{23}{\sqrt{5^2 + 3^2}} = \frac{23}{\sqrt{34}} \end{aligned}$$

(ii) Since the hyperplane is invariant when $\mathbf{w}$ and b are scaled by a constant c, we can find a value for c such that the functional margin of the example becomes 1, i.e.,

$$d_1 \left(c\mathbf{w}^T \mathbf{x}_1 + cb \right) = 1 \left(c \begin{bmatrix} 5 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 6c \right) = 23c = 1 \Rightarrow c = \frac{1}{23}$$

Thus the new weight vector is

$$\mathbf{w} = \frac{1}{23} \begin{bmatrix} 5 \\ 3 \end{bmatrix} = \begin{bmatrix} \frac{5}{23} \\ \\ \frac{3}{23} \end{bmatrix}$$

and the geometric margin of the example is

$$\gamma_1^g = \frac{\gamma_1^f}{\|\mathbf{w}\|} = \frac{1}{\sqrt{\left(\frac{5}{23}\right)^2 + \left(\frac{3}{23}\right)^2}} = \frac{1}{\frac{\sqrt{34}}{23}} = \frac{23}{\sqrt{34}}$$

We note that the geometric margin does not change under the scaling operation. This is not surprising because this margin is the Euclidean distance from the point defined by $\mathbf{x}_1$ to the given hyperplane, while the hyperplane and the vector $\mathbf{x}_1$ remain the same after the scaling.

(iii) Let $\mathbf{w} = [w_1\ w_2]^T$. To have $\gamma_1^f = 1$ and $\gamma_1^g = \frac{1}{\sqrt{2}}$ requires that

$$\begin{cases} g(\mathbf{x}_1) = \mathbf{w}^T\mathbf{x}_1 + b = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 6 = 1 \quad \Rightarrow \quad w_1 + 4w_2 = -5 \\ \gamma_1^g = \dfrac{\gamma_1^f}{\|\mathbf{w}\|} = \dfrac{1}{\sqrt{w_1^2 + w_2^2}} = \dfrac{1}{\sqrt{2}} \quad \Rightarrow \quad w_1^2 + w_2^2 = 2 \end{cases}$$

Solving for w_1 and w_2 yields $\mathbf{w} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$ and $\mathbf{w} = \begin{bmatrix} -7/17 \\ -23/17 \end{bmatrix}$. These correspond to two hyperplanes that result in a new geometric margin of $\frac{1}{\sqrt{2}}$ for the same example. □

4.3 Margin and performance of classifiers

Why is it important to find the margin of a training set? The answer is that the margin of a training set directly affects the performance of a classifier. For the particular case of a Perceptron, the margin of a training set influences the upper bound on the number of mistakes made by the Perceptron learning algorithm before convergence. For a more general class of classifiers called support vector machines, the margin of a training set affects the probability of the support vector machine (constructed based on this given training set) making a mistake in classifying an example.

We first look at the case of a Perceptron.

Theorem 4.1 *Let $S = \{(\mathbf{x}_i, d_i)\}$, $i = 1, \ldots, N$, be a non-trivial training set, and let $R = \max_{1 \le i \le N} \|\mathbf{x}_i\|$. Consider a Perceptron with weights $\mathbf{w}$, bias b, and the following update rules*

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \eta\, d_i \mathbf{x}_i \quad \text{and} \quad b_{k+1} = b_k + \eta\, d_i R^2$$

and suppose that there exist a vector $\mathbf{w}_{\text{opt}}$ (with $\|\mathbf{w}_{\text{opt}}\| = 1$) and a scalar b_{opt} such that

$$d_i \left(\langle \mathbf{w}_{\text{opt}} \cdot \mathbf{x}_i \rangle + b_{\text{opt}} \right) \ge \gamma_{\text{opt}}, \ i = 1, \ldots, N$$

where γ_{opt} is the geometric margin of S with respect to the hyperplane defined by $(\mathbf{w}_{\text{opt}}, b_{\text{opt}})$. Then the number of mistakes made by the on-line Perceptron learning algorithm on S is at most $(2R/\gamma_{\text{opt}})^2$. □

The proof for this theorem can be found in the book by Cristianini and Shawe-Taylor [4]. It is not necessary for us to examine the detailed steps of the proof here.

The key point for us to emphasize is the fact that γ_{opt} affects the performance of the Perceptron learning algorithm. The larger the value of γ_{opt} of a linearly-separable training set, the fewer the mistakes the Perceptron learning algorithm makes when finding a solution to the classification problem.

In the framework of Vapnik-Chervonenkis theory [8], the relationship between the margin of a training set and the probability of a classifier making a misclassification is expressed in the following theorem.

Theorem 4.2 (Theorem 4.18 in [4]): *Consider thresholding real-valued linear functions $\mathcal{L}$ with unit weight vectors on an inner product space X and fix $\gamma \in \mathbb{R}^+$. For any probability distribution $\mathcal{D}$ on $X \times \{-1, 1\}$ with support in a ball of radius R around the origin, with probability $1-\delta$ over l random examples S, any hypothesis $f \in \mathcal{L}$ that has margin $m_s(f) \geq \gamma$ on S has error no more than*

$$err_{\mathcal{D}}(f) \leq \varepsilon(l, \mathcal{L}, \delta, \gamma) = \frac{2}{l}\left(\frac{64R^2}{\gamma^2}\log\frac{el\gamma}{4R}\log\frac{128lR^2}{\gamma^2} + \log\frac{4}{\delta}\right)$$

provided $l > 2/\varepsilon$ and $64R^2/\gamma^2 < l$. □

The detailed context of this theorem (which is beyond the scope of this module) can be found [4]. What is important for us to appreciate here is the fact that the probability of misclassifying an example (represented by the quantity $\text{err}_D(f)$ in the theorem) is inversely proportional to the margin γ of the training set. The larger the margin γ, the lower the probability of a misclassification. This fact naturally leads to the development of a class of classifiers called support vector machines. The main objective in constructing a support vector machine is to find, for a given training set S, a hyperplane that maximizes the geometric margin of the set S. Such a hyperplane is called the *optimal hyperplane*. In other words, to construct a support vector machine with respect to a given S is to find the optimal hyperplane for S.

More details on how to construct a support vector machine are presented in the lecture slides.

Chapter 5

SVM Problems with Solutions

Problem 5.1 *Suppose that a support vector machine with the linear kernel* $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$ *is to be constructed using the training set below.*

i	$\mathbf{x}_i$	d_i
1	$[\ 1 \quad -1\ ]^T$	-1
2	$[\ 2 \quad 1\ ]^T$	1
3	$[\ 3 \quad 1\ ]^T$	1

(i) Determine the Lagrangian function L.

(ii) Show the explicit form of the dual problem.

Solution: (i) The Lagrangian function is

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i \left(d_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) - 1 \right)$$

From the given data, we have $N = 3$, and $\mathbf{w} = [w_1, w_2]^T$. Now

$$\begin{aligned} L(\mathbf{w}, b, \alpha) \;=\; & \frac{1}{2} \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \\ & - \alpha_1(-1) \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \alpha_1(-1)b + \alpha_1 \\ & - \alpha_2(1) \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} - \alpha_2(1)b + \alpha_2 \end{aligned}$$

$$
\begin{aligned}
& -\alpha_3(1)\begin{bmatrix} w_1 & w_2 \end{bmatrix}\begin{bmatrix} 3 \\ 1 \end{bmatrix} - \alpha_3(1)b + \alpha_3 \\
= \; & \frac{1}{2}\left(w_1^2 + w_2^2\right) + \alpha_1\left(w_1 - w_2 + 1\right) - \alpha_2\left(2w_1 + w_2 - 1\right) \\
& -\alpha_3\left(3w_1 + w_2 - 1\right) + \left(\alpha_1 - \alpha_2 - \alpha_3\right)b
\end{aligned}
$$

(ii) The objective function Q (to be maximized) of the dual problem is determined as follows.

$$
\begin{aligned}
Q(\alpha) &= \sum_{i=1}^{3}\alpha_i - \frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{3}\alpha_i d_i \alpha_j d_j \mathbf{x}_i^T\mathbf{x}_j \\
&= \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}\alpha_1 d_1 \alpha_1 d_1 \mathbf{x}_1^T\mathbf{x}_1 - \frac{1}{2}\alpha_1 d_1 \alpha_2 d_2 \mathbf{x}_1^T\mathbf{x}_2 - \frac{1}{2}\alpha_1 d_1 \alpha_3 d_3 \mathbf{x}_1^T\mathbf{x}_3 \\
&\quad -\frac{1}{2}\alpha_2 d_2 \alpha_1 d_1 \mathbf{x}_2^T\mathbf{x}_1 - \frac{1}{2}\alpha_2 d_2 \alpha_2 d_2 \mathbf{x}_2^T\mathbf{x}_2 - \frac{1}{2}\alpha_2 d_2 \alpha_3 d_3 \mathbf{x}_2^T\mathbf{x}_3 \\
&\quad -\frac{1}{2}\alpha_3 d_3 \alpha_1 d_1 \mathbf{x}_3^T\mathbf{x}_1 - \frac{1}{2}\alpha_3 d_3 \alpha_2 d_2 \mathbf{x}_3^T\mathbf{x}_2 - \frac{1}{2}\alpha_3 d_3 \alpha_3 d_3 \mathbf{x}_3^T\mathbf{x}_3 \\
&= \alpha_1 + \alpha_2 + \alpha_3 \\
&\quad -\frac{1}{2}\alpha_1(-1)\alpha_1(-1)\begin{bmatrix} 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} - \frac{1}{2}\alpha_1(-1)\alpha_2(1)\begin{bmatrix} 1 & -1 \end{bmatrix}\begin{bmatrix} 2 \\ 1 \end{bmatrix} \\
&\quad -\frac{1}{2}\alpha_1(-1)\alpha_3(1)\begin{bmatrix} 1 & -1 \end{bmatrix}\begin{bmatrix} 3 \\ 1 \end{bmatrix} - \frac{1}{2}\alpha_2(1)\alpha_1(-1)\begin{bmatrix} 2 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} \\
&\quad -\frac{1}{2}\alpha_2(1)\alpha_2(1)\begin{bmatrix} 2 & 1 \end{bmatrix}\begin{bmatrix} 2 \\ 1 \end{bmatrix} - \frac{1}{2}\alpha_2(1)\alpha_3(1)\begin{bmatrix} 2 & 1 \end{bmatrix}\begin{bmatrix} 3 \\ 1 \end{bmatrix} \\
&\quad -\frac{1}{2}\alpha_3(1)\alpha_1(-1)\begin{bmatrix} 3 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} - \frac{1}{2}\alpha_3(1)\alpha_2(1)\begin{bmatrix} 3 & 1 \end{bmatrix}\begin{bmatrix} 2 \\ 1 \end{bmatrix} \\
&\quad -\frac{1}{2}\alpha_3(1)\alpha_3(1)\begin{bmatrix} 3 & 1 \end{bmatrix}\begin{bmatrix} 3 \\ 1 \end{bmatrix} \\
&= \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}\left(2\alpha_1^2 + 5\alpha_2^2 + 10\alpha_3^2 - 2\alpha_1\alpha_2 - 4\alpha_1\alpha_3 + 14\alpha_2\alpha_3\right)
\end{aligned}
$$

The constraints are:

$$
\begin{aligned}
& \sum_{i=1}^{3}\alpha_i d_i = \alpha_1(-1) + \alpha_2(1) + \alpha_3(1) = -\alpha_1 + \alpha_2 + \alpha_3 = 0 \\
& \alpha_1 \geq 0, \;\; \alpha_2 \geq 0, \;\; \alpha_3 \geq 0
\end{aligned}
$$

□

Problem 5.2 *Formulate the dual problem for the case of support vector machines with soft margin.*

Solution: The primal problem is

$$\begin{aligned} \text{Minimize:} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \\ \text{Subject to:} \quad & d_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1 + \xi_i \geq 0 \\ & \xi_i \geq 0 \end{aligned}$$

The Lagrangian function is

$$\begin{aligned} & L(\mathbf{w}, b, \xi, \alpha, \beta) \\ = \; & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left(d_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1 + \xi_i\right) - \sum_{i=1}^{N}\beta_i\xi_i \\ = \; & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i d_i \mathbf{w}^T\mathbf{x}_i - b\sum_{i=1}^{N}\alpha_i d_i + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i\xi_i - \sum_{i=1}^{N}\beta_i\xi_i \end{aligned}$$

where α_i and β_i are the Lagrange multipliers. The KKT conditions are

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{N}\alpha_i d_i \mathbf{x}_i = \mathbf{0} \\ \frac{\partial L}{\partial b} &= -\sum_{i=1}^{N}\alpha_i d_i = 0 \\ \frac{\partial L}{\partial \xi_i} &= C - \alpha_i - \beta_i = 0 \\ d_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1 + \xi_i &\geq 0 \qquad (5.1) \\ \alpha_i\left(d_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1 + \xi_i\right) &= 0 \\ \beta_i\xi_i &= 0 \qquad (5.2) \\ \alpha_i &\geq 0 \\ \beta_i &\geq 0 \qquad (5.3) \end{aligned}$$

Therefore,

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i d_i \mathbf{x}_i \qquad (5.4)$$

and consequently,

$$\begin{aligned} \mathbf{w}^T\mathbf{w} &= \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i d_i \alpha_j d_j \mathbf{x}_i^T\mathbf{x}_j \\ \sum_{i=1}^{N}\alpha_i d_i \mathbf{w}^T\mathbf{x}_i &= \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i d_i \alpha_j d_j \mathbf{x}_i^T\mathbf{x}_j \end{aligned}$$

Moreover, from the KKT conditions we have

$$C = \alpha_i + \beta_i \tag{5.5}$$

Hence,

$$\begin{aligned} & L(\mathbf{w},b,\xi,\alpha,\beta) \\ = \;& \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i d_i \mathbf{w}^T\mathbf{x}_i - b\sum_{i=1}^{N}\alpha_i d_i + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i\xi_i - \sum_{i=1}^{N}\beta_i\xi_i \\ = \;& -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i d_i \alpha_j d_j \mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{N}(\alpha_i+\beta_i)\xi_i + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i\xi_i - \sum_{i=1}^{N}\beta_i\xi_i \\ = \;& -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i d_i \alpha_j d_j \mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{N}\alpha_i \end{aligned}$$

Note that this Lagrangian function is same as the that for the case of linearly separable patterns. The effect of the error penalty $\sum_{i=1}^{N}\xi_i$ on the optimization problem is to set an upper bound for the Lagrange multiplier α_i. This can be seen by the fact that, from the KKT conditions, $\alpha_i = C - \beta_i$ with $\beta_i \geq 0$; that is, $0 \leq \alpha_i \leq C$. Consequently, we have

$$Q(\alpha) \equiv L(\mathbf{w},b,\xi,\alpha,\beta) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i d_i \alpha_j d_j \mathbf{x}_i^T\mathbf{x}_j$$

and the dual problem can be expressed as

$$\begin{aligned} \text{Find}: \quad & \alpha_i \\ \text{Maximize}: \quad & Q(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j \\ \text{Subject to}: \quad & \sum_{i=1}^{N}\alpha_i d_i = 0 \\ & 0 \leq \alpha_i \leq C \end{aligned}$$

□

Remark 5.1 Recall that any example in a training data set must satisfy the following two (among other) KKT conditions, i.e., Equations (5.1) and (5.2):

$$d_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)-1+\xi_i\geq 0 \quad \text{and} \quad \beta_i\xi_i=0$$

Moreover, from Equations (5.5) and (5.3), we have

$$C=\alpha_i+\beta_i \quad \text{and} \quad \beta_i\geq 0$$

If an example $\mathbf{x}_i$ is a *support vector* (i.e., with $\alpha_i>0$), then from Equation (5.1) we have

$$d_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)-1+\xi_i=0 \quad \text{or} \quad d_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)=1-\xi_i \tag{5.6}$$

In the soft-margin approach, α_i has a upper limit C. Thus, for a support vector $\mathbf{x}_i$, we have $0<\alpha_i\leq C$. Here we have two cases: (i) $\alpha_i<C$ and (ii) $\alpha_i=C$.

For the first case, we have $\beta_i=C-\alpha_i>0$. To satisfy the condition that $\beta_i\xi_i=0$, we must have $\xi_i=0$. This means that the support vector $\mathbf{x}_i$ (with $\alpha_i<C$) is located on the border of the margin of separation.

For the second case (i.e., $\alpha_i=C$), we have $\beta_i=C-\alpha_i=0$. Thus $\beta_i\xi_i=0$ can still be satisfied by having $\xi_i\neq 0$. Consequently, for this case the support vector $\mathbf{x}_i$ is located inside the margin of separation. Recall, from Equation (4.1), that an example $\mathbf{x}_i$ is classified correction if

$$\text{sgn}\left[d_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)\right]=1$$

Now for the case where $\alpha_i=C$ we have, from Equation (5.6),

$$\text{sgn}\left[d_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)\right]=\text{sgn}\left[1-\xi_i\right]$$

Hence , if a support vector located inside the margin of separation is to be classified correctly, we must have $sgn[1-\xi_i]=1$, which is possible only when $\xi_i\leq 1$. In other words, the support vector $\mathbf{x}_i$ (with $\alpha_i=C$) will be classified incorrectly if $\xi_i>1$, i.e., $\mathbf{x}_i$ is located (inside the margin of separation but) at the "wrong" side of the hyperplane. □

Problem 5.3 *Consider the training data set shown in Table 5.1 below.*

(a) Suggest a kernel for constructing a support vector machine to classify this set of training data. (Justify your suggestion.)

(b) Write down the dual problem in an explicit form for the given training data set, using the kernel that you have suggested in part (a) above.

Table 5.1: Training data set.

i	1	2	3
$\mathbf{x}_i$	$\begin{bmatrix}4 & 1\end{bmatrix}^T$	$\begin{bmatrix}4 & -1\end{bmatrix}^T$	$\begin{bmatrix}2 & 0\end{bmatrix}^T$
d_i	$+1$	$+1$	-1

(c) Assume that the values of the Lagrange multipliers for the dual problem in part (b) above are $\alpha_1 = 0.25$, $\alpha_2 = 0.25$, and $\alpha_3 = 0.5$. Determine the functional margin and the geometric margin of the training set.

(d) Suppose that a new example, $\mathbf{x}_4 = [4.5\ 0]^T$ with the label $d_4 = 1$, is added to the training data set shown in Table 5.1. Given the transformation as specified below, determine (i) the new training data set in the feature space, and (ii) the Gramm matrix.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \rightarrow \quad \varphi(\mathbf{x}) = \begin{bmatrix} x_1 + x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix}$$

Solution: (a) Plotting the three data points (as shown in Figure 5.1 below) reveals that they are linearly separable. Therefore, a linear kernel can be used.

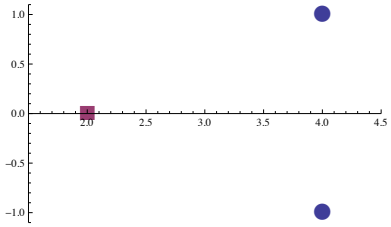


Figure 5.1: Training data.

(b) From the given training examples, we have

$$\begin{aligned} Q(\alpha) &= a1 + a2 + a3 - \frac{1}{2}\Big(a_1a_1d_1d_1\mathbf{x}_1^T\mathbf{x}_1 + a_2a_2d_2d_2\mathbf{x}_2^T\mathbf{x}_2 + a_3a_3d_3d_3\mathbf{x}_3^T\mathbf{x}_3 \\ &+ a_1a_2d_1d_2\mathbf{x}_1^T\mathbf{x}_2 + a_1a_3d_1d_3\mathbf{x}_1^T\mathbf{x}_3 + a_2a_1d_2d_1\mathbf{x}_2^T\mathbf{x}_1 + a_2a_3d_2d_3\mathbf{x}_2^T\mathbf{x}_3 \\ &+ a_3a_1d_3d_1\mathbf{x}_3^T\mathbf{x}_1 + a_3a_2d_3d_2\mathbf{x}_3^T\mathbf{x}_2\Big) \end{aligned}$$

The explicit form of the problem is:

$$\begin{aligned} \text{Find}: &\quad \alpha_1, \alpha_2, \alpha_3 \\ \text{Maximize}: &\quad Q(\alpha) \text{ as obtained above} \\ \text{Subject to}: &\quad \alpha_1 + \alpha_2 - \alpha_3 = 0 \\ &\quad \alpha_i \geq 0 \end{aligned}$$

(c) Since the data set is linearly separable, we can first determine $\mathbf{w}$ and b, and then calculate the functional and geometric margins. Now,

$$\begin{aligned}
\mathbf{w}_\circ &= \sum_{i=1}^{3} \alpha_i d_i \mathbf{x}_i \\
&= 0.25 \cdot 1 \cdot \begin{bmatrix} 4 \\ 1 \end{bmatrix} + 0.25 \cdot 1 \cdot \begin{bmatrix} 4 \\ -1 \end{bmatrix} + 0.5 \cdot (-1) \cdot \begin{bmatrix} 2 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\end{aligned}$$

To find b, we use the first example (which is also a support vector). So

$$b_\circ = \frac{1}{d_1} - \mathbf{w}_\circ^T \mathbf{x}_1 = \frac{1}{1} - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 1 \end{bmatrix} = -3$$

The functional margins of the examples are

$$\begin{aligned}
\gamma_1^f &= d_1 \left(\mathbf{w}_\circ^T \mathbf{x}_1 + b_\circ \right) = 1 \cdot \left(\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 1 \end{bmatrix} - 3 \right) = 1 \\
\gamma_2^f &= d_2 \left(\mathbf{w}_\circ^T \mathbf{x}_2 + b_\circ \right) = 1 \cdot \left(\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ -1 \end{bmatrix} - 3 \right) = 1 \\
\gamma_3^f &= d_3 \left(\mathbf{w}_\circ^T \mathbf{x}_3 + b_\circ \right) = -1 \cdot \left(\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \end{bmatrix} - 3 \right) = 1
\end{aligned}$$

Thus, the functional margin of the training set is

$$\gamma^f = \min \left\{ \gamma_1^f, \gamma_1^f, \gamma_1^f \right\} = 1$$

and the geometric margin of the training set is

$$\gamma^g = \frac{\gamma^f}{\|\mathbf{w}_\circ\|} = \frac{1}{\sqrt{1^1 + 0^2}} = 1$$

(d)-(i) With the nonlinear transformation

$$\varphi(\mathbf{x}) = \begin{bmatrix} x_1 + x_2^2 \\ \sqrt{2} x_1 x_2 \end{bmatrix}$$

the training data set in the feature space becomes

$$\begin{aligned}
\varphi(\mathbf{x}_1) &= \begin{bmatrix} x_1+x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix} = \begin{bmatrix} 4+1^2 \\ \sqrt{2}\cdot 4\cdot 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 4\sqrt{2} \end{bmatrix} \\
\varphi(\mathbf{x}_2) &= \begin{bmatrix} x_1+x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix} = \begin{bmatrix} 4+(-1)^2 \\ \sqrt{2}\cdot 4\cdot (-1) \end{bmatrix} = \begin{bmatrix} 5 \\ -4\sqrt{2} \end{bmatrix} \\
\varphi(\mathbf{x}_3) &= \begin{bmatrix} x_1+x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix} = \begin{bmatrix} 2+0^2 \\ \sqrt{2}\cdot 2\cdot 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \\
\varphi(\mathbf{x}_4) &= \begin{bmatrix} x_1+x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix} = \begin{bmatrix} 4.5+0^2 \\ \sqrt{2}\cdot 4.5\cdot 0 \end{bmatrix} = \begin{bmatrix} 4.5 \\ 0 \end{bmatrix}
\end{aligned}$$

(d)-(ii) For the given transformation, the kernel is

$$K(\mathbf{x}_i,\mathbf{x}_j) = \varphi^T(\mathbf{x}_i)\varphi(\mathbf{x}_j)$$

Hence,

$$\begin{aligned}
K(\mathbf{x}_1,\mathbf{x}_1) &= \varphi^T(\mathbf{x}_1)\varphi(\mathbf{x}_1) = \begin{bmatrix} 5 & 4\sqrt{2} \end{bmatrix}\begin{bmatrix} 5 \\ 4\sqrt{2} \end{bmatrix} = 57 \\
K(\mathbf{x}_1,\mathbf{x}_2) &= K(\mathbf{x}_2,\mathbf{x}_1) = \begin{bmatrix} 5 & 4\sqrt{2} \end{bmatrix}\begin{bmatrix} 5 \\ -4\sqrt{2} \end{bmatrix} = -7 \\
K(\mathbf{x}_1,\mathbf{x}_3) &= K(\mathbf{x}_3,\mathbf{x}_1) = \begin{bmatrix} 5 & 4\sqrt{2} \end{bmatrix}\begin{bmatrix} 2 \\ 0 \end{bmatrix} = 10 \\
K(\mathbf{x}_1,\mathbf{x}_4) &= K(\mathbf{x}_4,\mathbf{x}_1) = \begin{bmatrix} 5 & 4\sqrt{2} \end{bmatrix}\begin{bmatrix} 4.5 \\ 0 \end{bmatrix} = 22.5 \\
K(\mathbf{x}_2,\mathbf{x}_2) &= \begin{bmatrix} 5 & -4\sqrt{2} \end{bmatrix}\begin{bmatrix} 5 \\ -4\sqrt{2} \end{bmatrix} = 57 \\
K(\mathbf{x}_2,\mathbf{x}_3) &= K(\mathbf{x}_3,\mathbf{x}_2) = \begin{bmatrix} 5 & -4\sqrt{2} \end{bmatrix}\begin{bmatrix} 2 \\ 0 \end{bmatrix} = 10 \\
K(\mathbf{x}_2,\mathbf{x}_4) &= K(\mathbf{x}_4,\mathbf{x}_2) = \begin{bmatrix} 5 & -4\sqrt{2} \end{bmatrix}\begin{bmatrix} 4.5 \\ 0 \end{bmatrix} = 22.5 \\
K(\mathbf{x}_3,\mathbf{x}_3) &= \begin{bmatrix} 2 & 0 \end{bmatrix}\begin{bmatrix} 2 \\ 0 \end{bmatrix} = 4 \\
K(\mathbf{x}_3,\mathbf{x}_4) &= K(\mathbf{x}_4,\mathbf{x}_3) = \begin{bmatrix} 2 & 0 \end{bmatrix}\begin{bmatrix} 4.5 \\ 0 \end{bmatrix} = 9
\end{aligned}$$

$$K(\mathbf{x}_4, \mathbf{x}_4) = \begin{bmatrix} 4.5 & 0 \end{bmatrix} \begin{bmatrix} 4.5 \\ 0 \end{bmatrix} = 20.25$$

and the Gram matrix is

$$\mathbf{K} = \begin{bmatrix} 57 & -7 & 10 & 22.5 \\ -7 & 57 & 10 & 22.5 \\ 10 & 10 & 4 & 9 \\ 22.5 & 22.5 & 9 & 20.25 \end{bmatrix}$$

□

More numerical examples are provided in the lecture slides.

Part B

Reinforcement Learning

Chapter 6

Markov Decision Processes

6.1 Elements of a Markov decision process

A Markov decision process is a model of a dynamical system[1] comprising of the following elements, namely, *state*, *action*, *transition function*, and *reward*.

- *A set of states* [2]:
$$S = \{s_1, s_2, \ldots, s_{|S|}\} \tag{6.1}$$
- *A set of actions*:
$$A = \{a_1, a_2, \ldots, a_{|A|}\} \tag{6.2}$$
- *A transition function*: A transition function can be deterministic or non-deterministic. A deterministic transition function is defined as
$$\bar{f} : S \times A \to S \tag{6.3}$$
In other words, a deterministic transition function $\bar{f}$ specifies the new state s' of the system after taking action a at state s, i.e.,
$$\bar{f}(s, a) = s' \tag{6.4}$$
A non-deterministic transition function, on the other hand, specifies the probability of the system reaching the new state s' after taking action a at state s, i.e.,
$$f : S \times A \times S \to [0, 1] \tag{6.5}$$

[1] Such a system must exhibit the so-called Markov property, which can be intuitively summarized as "given the present, the future does not depend on the past". A formal definition can be found in many textbook on stochastic processes. Most of physical systems can be considered as having the Markov property. An example of a non-Markov process is that of ion movement across cell membrane [11].

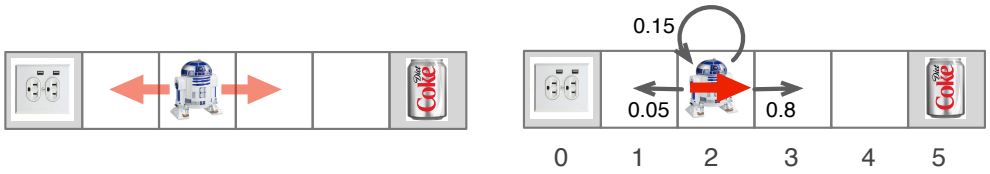
[2] The notation $|S|$ represents the number of elements in the set S.

That is,

$$f\left(s,a,s'\right)=P\left(s_{t+1}=s'|\,s_t=s,a_t=a\right)\equiv P_{ss'}^{a} \tag{6.6}$$

Remark 6.1 Note that $\bar{f}(s,a)$ and $f\left(s,a,s'\right)$ are equivalent if $P_{ss'}^{a}=1.0$. **In this module, we will mostly deal with on systems with non-deterministic transitions.** However, systems with deterministic transitions (such as that in Example 6.3) will sometimes be used in these notes as simple examples for illustration of concepts and principles. □

Example 6.1 Consider the mobile robot designed for a cleaning task as shown in Figure 6.1(a). The robot's behavior consisting of picking up a can and recharging its battery. Suppose that the robot can only move to the left (defined as the action $a=-1$) or right ($a=1$) in a horizontal grid, with each square in the grid represents one position of the robot. In this case, the robot is the agent with the state set $S=\{0,1,2,3,4,5\}$ and the action set $A=\{-1,1\}$.



(a) Robot for cleaning task. (b) Probabilities of transitions.

Figure 6.1: A simple cleaning task.

Suppose that due to uncertainties in the robot's operation (such as slippery floor or error in the robot's position-tracking system, etc.), the robot's action may become non-deterministic. For example, for the case as shown in Figure 6.1(b), the transition function is defined as

$$\begin{cases} f(2,1,3)=0.8 \\ f(2,1,2)=0.15 \\ f(2,1,1)=0.05 \end{cases} \tag{6.7}$$

□

- *Reward r and return R*:

  When the agent takes an action a to make a transition from state s to another state s', it receives a numerical reward r. This reward is received one time-step later, with the agent in the new state s'. Hence, we can express the reward given to the agent for taking action a_t at state s_t at time step t and reaching the new state s_{t+1} as

$$r_{t+1}=\rho\left(s_t,a_t,s_{t+1}\right) \tag{6.8}$$

where $\rho : S \times A \times S \to \mathbb{R}$ is called the reward function. If the transition function is non-deterministic, then the reward for taking action a at state s is characterized by the *expected value* of r_{t+1} over all possible new states, i.e.,

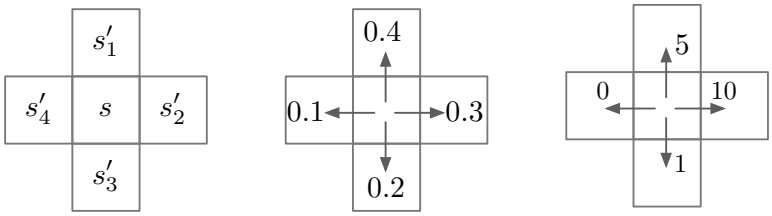
$$\mathbb{E}\Big[r_{t+1}\Big| s_t = s\Big] = \sum_{s'} \Big(P^a_{ss'} r_{t+1}\Big| s_{t+1} = s'\Big) = \sum_{s'} P^a_{ss'}\, \rho(s,a,s') \tag{6.9}$$

where the summation over s' accounts for all the possible next states, and $P^a_{ss'}$ is the probability of the agent reaching a specific new state s' after taking action a at state s, as defined in Equation (6.6).

Remark 6.2 The condition $s_{t+1} = s'$ is explicitly shown in Equation (6.9) to emphasize the fact that the definition of r_{t+1} as given in Equation (6.8) involves a new state s_{t+1}. In the sequel, this condition is sometimes omitted (for the sake of readability) in the expression of the expected value of r_{t+1}. □

The example below illustrates the calculation of such an expected value.

Example 6.2 Consider the situation in which an agent at state s takes action a in a stochastic setting, as illustrated in Figure 6.2. Because of the non-deterministic transition function $f(s,a,s')$, there are four possibilities for the new state s', namely, s'_1, s'_2, s'_3, and s'_4.



(a) States. (b) Transition probabilities. (c) Rewards.

Figure 6.2: Taking a at s with a non-deterministic transition function.

The reward for reaching the four different new states are (as given in the figure):

$$\begin{aligned} r_{t+1}|_{s_{t+1}=s'_1} = \rho\left(s,a,s'_1\right) &= 5 \\ r_{t+1}|_{s_{t+1}=s'_2} = \rho\left(s,a,s'_2\right) &= 10 \\ r_{t+1}|_{s_{t+1}=s'_3} = \rho\left(s,a,s'_3\right) &= 1 \\ r_{t+1}|_{s_{t+1}=s'_4} = \rho\left(s,a,s'_4\right) &= 0 \end{aligned} \tag{6.10}$$

and the transition functions are:

$$f\left(s,a,s'_1\right) = P\left(s_{t+1} = s'_1 | \, s_t = s, a_t = a\right) \equiv P^a_{ss'_1} = 0.4$$

$$\begin{aligned} f\left(s,a,s_2'\right) &= P\left(s_{t+1}=s_2'|\, s_t=s, a_t=a\right) \equiv P^a_{ss_2'} = 0.3 \\ f\left(s,a,s_3'\right) &= P\left(s_{t+1}=s_3'|\, s_t=s, a_t=a\right) \equiv P^a_{ss_3'} = 0.2 \\ f\left(s,a,s_4'\right) &= P\left(s_{t+1}=s_4'|\, s_t=s, a_t=a\right) \equiv P^a_{ss_4'} = 0.1 \end{aligned} \tag{6.11}$$

Thus, the reward for taking action a at state s at time step t is

$$\begin{aligned} \mathbb{E}\left[r_{t+1}\right] &= \sum_{s'}\left(P^a_{ss'}\, r_{t+1}\right) = \sum_{s'} P^a_{ss'}\,\rho\left(s,a,s'\right) \\ &= P^a_{ss_1'}\,\rho\left(s,a,s_1'\right) + P^a_{ss_2'}\,\rho\left(s,a,s_2'\right) \\ &\quad + P^a_{ss_3'}\,\rho\left(s,a,s_3'\right) + P^a_{ss_4'}\,\rho\left(s,a,s_4'\right) \\ &= 0.4\times 5 + 0.3\times 10 + 0.2\times 1 + 0.1\times 0 \\ &= 5.2 \end{aligned} \tag{6.12}$$

□

Remark 6.3 It is important to note, as illustrated in Example 6.2, the difference between (i) the (expected) reward for taking action a at s, and (ii) the reward for taking a at s *and reaching a specific new state*. In that example, for case (i) the reward is 5.2, while for case (ii) the specific value of the reward depends on which new state is reached, as indicated in Equation (6.10). □

Consider the scenario where the agent starts from an initial state and reaches some state s after moving t steps. We now ask the question: What is the total reward we can obtain *from this point onward* (i.e., from state s) if the agent continues to make transitions? We call this total reward the *return* (denoted by R_t) with respect to the time step t. This return is defined as:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{6.13}$$

where k is the index of the time steps taken after time-step t, with $k=0$ for the first time step after t, and γ (with $0 \le \gamma \le 1$) is called the *discount rate*. R_t thus determines the present value of future rewards. A reward received k steps in the future is discounted by a factor of γ^{k-1}. A small value for γ indicates that the agent focuses more on the immediate rewards from the next few steps while heavily discounts the rewards from future steps. Conversely, a large γ forces the agent to take into account future rewards more strongly, i.e., the agent becomes more far-sighted.

Example 6.3 Consider the cleaning task described earlier in Example 6.1, with the state set $S = \{0,1,2,3,4,5\}$ and the action set $A = \{-1,1\}$. Suppose that

1. all transitions are deterministic as defined below:

$$\bar{f}(s,a) = \begin{cases} s+a & \text{if } 1 \le s \le 4 \\ s & \text{if } s = 0 \text{ or } s = 5 \end{cases}$$

2. the reward for the robot to move from state 4 to state 5 (to pick up the can) is 5 and from state 1 to state 0 (to recharge its battery) is 1, as illustrated in Figure 6.3, and

3. the rewards for all other transitions are 0.

Calculate the return (with a discount rate of $\gamma = 0.5$) if the robot starts at state 2 and reaches state 5 without any change of direction of its motion.

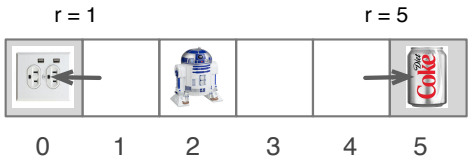


Figure 6.3: Rewards in the simple cleaning task.

Solution: From items 2 and 3 given above, we can express the reward function as

$$\rho(s,a,s') = \begin{cases} 5 & \text{if } s \neq 5 \text{ and } s' = 5 \\ 1 & \text{if } s \neq 0 \text{ and } s' = 0 \\ 0 & \text{otherwise} \end{cases}$$

To reach state 5 from state 2 without any change of direction, the robot will make three transitions, i.e., $\bar{f}(2,1) = 3$, $\bar{f}(3,1) = 4$ and $\bar{f}(4,1) = 5$. The specified return therefore is

$$\begin{aligned} R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \\ &= \rho(2,1,3) + 0.5 \cdot \rho(3,1,4) + 0.5^2 \cdot \rho(4,1,5) \\ &= 0 + 0 + 0.25 \cdot 5 \\ &= 1.25 \end{aligned}$$

□

6.2 Control of Markov decision processes

To control a Markov decision process is to impose a decision-making mechanism on the process. This decision-making mechanism is generally referred to as a *policy*.

A policy specifies which action to be taken by the system at a given state. Policies can be deterministic or non-deterministic. A deterministic policy specifies a particular action to be taken at a particular state, i.e.,

$$\pi : S \to A; \;\; \pi(s) = a, \; s \in S, \; a \in A \tag{6.14}$$

while a non-deterministic policy specifies the probability of an action a being taken at state s at time t, i.e.,

$$\pi : S \times A \to [0,1]; \;\; \pi(s,a) = P\left(a_t = a \middle| s_t = s\right), \; s \in S, a \in A \tag{6.15}$$

In this module, we only consider deterministic policies.

Example 6.4 In Example 6.1, suppose that the behavior of the robot is restricted to the following specifications:

1. In state 1, the robot must take the action $a = -1$.
2. In state 0 or 5, the robot can take either action.
3. In any of the remaining states, the robot must take the action $a = 1$.

Then the policy that governs the behavior of the robot is:

$$\begin{cases} \pi(1) = -1 & \\ \pi(s) = \pm 1 & \text{if} \quad s = 0, 5 \\ \pi(s) = 1 & \text{if} \quad s = 2, 3, 4 \end{cases}$$

□

The problem of optimal control of a Markov decision process is to find a policy that yields the maximum reward when the agent performs a given task. Such a policy is call an *optimal policy*. To determine an optimal policy requires the concept of Q-function, also called *action-value function*.

6.3 Q-function

The Q-function measures the "worth" of taking a specific action a at a particular state s under a given policy π. This "worth" is represented by a value given by the function $Q^\pi(s,a)$, with $Q^\pi : S \times A \to \mathbb{R}$, which is defined as the *expected return* from taking action a at state s at time step t, and thereafter following policy π, i.e.,

$$Q^\pi(s,a) = \mathbb{E}^\pi\left[R_t \middle| s_t = s\right] \tag{6.16}$$

Example 6.5 Determine the value of the Q-function for the policy as given in Example 6.3, with the assumption that the robot stops after reaching state 5.

Solution: The policy is $\pi(2)=\pi(3)=\pi(4)=1$. Since all transitions in this example are deterministic, the Q-function degenerates to just $Q^\pi(2,1)=\mathbb{E}[R_t|\,s_t=2]=(R_t|s_t=2)$, i.e., there is no need to consider the expectation operator $\mathbb{E}$. Therefore, the value for $Q^\pi(2,1)$ is the same as the value for R_t calculated in Example 6.3. □

Remark 6.4 Determining the value for the Q-function when the state transitions are non-deterministic is more complicated, as is shown later in Example 6.6. □

Since the Q-function measures the "worth" of a state-action pair (s,a) in terms of the rewards received when the agent is executing a task, an optimal policy is then defined as a policy that maximizes (with respect to a given task) the values of the Q-function over *all* possible (s,a) pairs. Solving a reinforcement learning problems is essentially about finding an optimal policy for executing a given task. To do this requires the *Bellman Equation*.

6.4 The Bellman Equation

The Bellman Equation provides an iterative method for determining the values of the Q-function for a given policy. This equation specifies a direct relationship between the value of the Q-function at state s_t with action a_t (i.e., $Q(s_t,a_t)$) and the value of the Q-function at state s_{t+1} with action a_{t+1} (i.e., $Q(s_{t+1},a_{t+1})$).

From the definition of the Q-function given in Equation (6.16), we have

$$\begin{aligned}
Q^\pi(s,a) &= \mathbb{E}^\pi\Big[R_t\Big|\,s_t=s\Big] \\
&= \mathbb{E}^\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\Big|\,s_t=s\right] \qquad (6.17)\\
&= \mathbb{E}^\pi\left[r_{t+1}+\gamma r_{t+2}+\gamma^2 r_{t+3}+\cdots\Big|\,s_t=s\right] \\
&= \mathbb{E}^\pi\left[\left(r_{t+1}+\gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}\right)\Big|\,s_t=s\right] \quad \big(\text{Isolating } r_{t+1}\big) \\
&= \mathbb{E}^\pi\left[r_{t+1}\right]+\mathbb{E}^\pi\left[\gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}\Big|\,s_t=s\right] \qquad (6.18)
\end{aligned}$$

Note that Equation (6.18) results from the linearity of the expectation operator $\mathbb{E}$, i.e., $\mathbb{E}[X+Y]=\mathbb{E}[X]+\mathbb{E}[Y]$, as stated in Equation (1.1).

We note that the condition in the second expectation operation in Equation (6.18)

is $s_t = s$, but the rewards are calculated starting from s_{t+1} since the first term (i.e., when $k = 0$) of $\sum_{k=0}^{\infty} \gamma^k r_{t+k+2}$ is r_{t+2}. Figure 6.4 illustrates this situation.

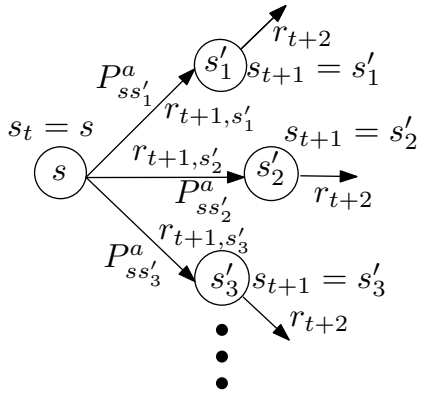


Figure 6.4: Non-deterministic transitions.

Referring to the state transition structure as illustrated in Figure 6.4 and the definition of *return* given by Equation (6.13), the probabilities of the agent (after taking action a at state $s_t = s$) ending up in state $s_{t+1} = s'$ is $P^a_{ss'}$, while the return from executing policy π starting from <u>one</u> of the possible new state $s_{t+1} = s'$ is

$$R_{t+1} = \sum_{k=0}^{\infty} \left[\gamma^k r_{t+k+2} \middle| s_{t+1} = s' \right], \quad \text{where } s' \in \{s'_1, s'_2, s'_3, \ldots\} \tag{6.19}$$

Thus, we take into consideration of the state transition probability $P^a_{ss'}$ to express $\sum_{k=0}^{\infty} \left[\gamma^k r_{t+k+2} \middle| s_{t+1} = s \right]$ (i..e., the summation term inside the expectation operator $\mathbb{E}^{\pi}$ in the second term of Equation (6.18) in terms of s' and $P^a_{ss'}$ as

$$\begin{aligned} & \sum_{k=0}^{\infty} \left[\gamma^k r_{t+k+2} \middle| s_t = s \right] \\ = \; & P^a_{ss'_1} \sum_{k=0}^{\infty} \left[\gamma^k r_{t+k+2} \middle| s_{t+1} = s'_1 \right] + P^a_{ss'_2} \sum_{k=0}^{\infty} \left[\gamma^k r_{t+k+2} \middle| s_{t+1} = s'_2 \right] + \\ & P^a_{ss'_3} \sum_{k=0}^{\infty} \left[\gamma^k r_{t+k+2} \middle| s_{t+1} = s'_3 \right] + \cdots \\ = \; & \sum_{s'} P^a_{ss'} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \middle| s_{t+1} = s' \right] \end{aligned} \tag{6.20}$$

Now substituting Equation (6.20) into the second term in the last line of Equation

(6.18) yields

$$\begin{aligned}
& \mathbb{E}^{\pi}\left[\gamma\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+2}\,\middle|\,s_t=s\right] \\
= \; & \mathbb{E}^{\pi}\left[\gamma\sum_{s'}P_{ss'}^{a}\left[\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+2}\,\middle|\,s_{t+1}=s'\right]\right] \\
= \; & \sum_{s'}P_{ss'}^{a}\gamma\underbrace{\mathbb{E}^{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+2}\,\middle|\,s_{t+1}=s'\right]}_{\text{This is } Q^{\pi}\left(s',a'\right) \text{ by definition}} \quad \text{(Note: } P_{ss'}^{a} \text{ and } \gamma \text{ are independent of } \pi\text{)} \\
= \; & \sum_{s'}P_{ss'}^{a}\gamma Q^{\pi}\left(s',a'\right) \qquad (6.21)
\end{aligned}$$

Recall from Equation (6.9) that $E\left[r_{t+1}\right]=\sum_{s'}P_{ss'}^{a}\rho(s,a,s')$, where the values of the reward function ρ are the rewards r_{t+1,s_1}, r_{t+1,s_2}, etc., as indicated in Figure 6.4. Hence, Equation (6.18) can be written as

$$\begin{aligned}
Q^{\pi}(s,a) &= \mathbb{E}^{\pi}\left[r_{t+1}\right]+\mathbb{E}^{\pi}\left[\gamma\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+2}\,\middle|\,s_t=s\right] \\
&= \sum_{s'}P_{ss'}^{a}\rho(s,a,s')+\sum_{s'}P_{ss'}^{a}\gamma Q^{\pi}\left(s',a'\right) \\
&= \sum_{s'}P_{ss'}^{a}\left(\rho(s,a,s')+\gamma Q^{\pi}\left(s',a'\right)\right) \qquad (6.22)
\end{aligned}$$

Equation (6.22) is called the Bellman Equation for the Q-function. It states that, under a given policy π, the value of taking action $a_t=a$ at state $s_t=s$ must equal to the expected reward of transitioning into the new state $s_{t+1}=s'$, i.e., $\rho(s,a,s')$, plus the discounted expected value of taking action $a_{t+1}=a'=\pi(s')$ at state s'. This **recursive relationship** between $Q^{\pi}(s,a)$ and $Q^{\pi}(s',a')$ is very important, because it serves as the foundation of the method of Dynamic Programming for the determination of optimal policies, as discussed in Chapter 7.

Example 6.6 Consider the cleaning task described earlier in Example 6.1 with the following reward function:

$$\rho(s,a,s')=\begin{cases}5 & \text{if } s\neq 5 \text{ and } s'=5\\ 1 & \text{if } s\neq 0 \text{ and } s'=0\\ 0 & \text{otherwise}\end{cases} \qquad (6.23)$$

Assuming that the robot stops after reaching state 0 or 5, determine the value of

$Q^{\pi}(2,1)$ for the policy as given in Example 6.4 with the transition probabilities associated with the given policy as shown in Figure 6.5.

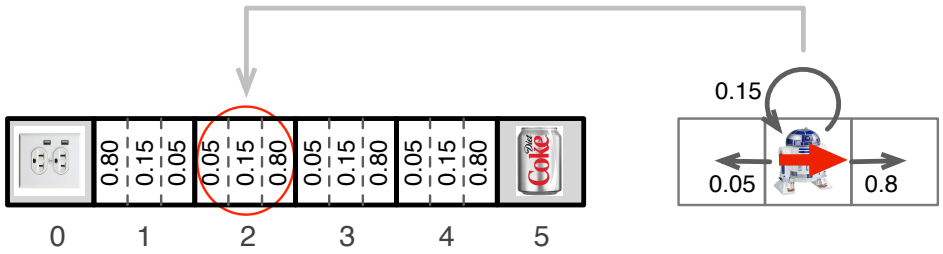


Figure 6.5: Transition probabilities for the policy as given in Example 6.4. The three numbers in each square indicate the probabilities of the robot, when taking the action prescribed by the policy, reaching the state to its left, remaining at the same state, and reaching the state to its right, respectively.

Solution: Recall from Example (6.3) that the policy is

$$\begin{cases} \pi(1) = -1 \\ \pi(s) = 1 \quad \text{if} \quad s = 2,3,4 \\ \pi(s) = \pm 1 \quad \text{if} \quad s = 0,5 \end{cases}$$

From Figure 6.5, we have the following state-transition probabilities:

$$\begin{aligned} f(1,-1,0) &= 0.80 \equiv P_{10}^{-1} & f(2,1,1) &= 0.05 \equiv P_{21}^{1} \\ f(1,-1,1) &= 0.15 \equiv P_{11}^{-1} & f(2,1,2) &= 0.15 \equiv P_{22}^{1} \\ f(1,-1,2) &= 0.05 \equiv P_{12}^{-1} & f(2,1,3) &= 0.80 \equiv P_{23}^{1} \end{aligned}$$

$$\begin{aligned} f(3,1,2) &= 0.05 \equiv P_{32}^{1} & f(4,1,3) &= 0.05 \equiv P_{43}^{1} \\ f(3,1,3) &= 0.15 \equiv P_{33}^{1} & f(4,1,4) &= 0.15 \equiv P_{44}^{1} \\ f(3,1,4) &= 0.80 \equiv P_{34}^{1} & f(4,1,5) &= 0.80 \equiv P_{45}^{1} \end{aligned}$$

We can interpret the assumption that the robot will stop once it reaches state 0 or 5 to mean that the probability of the robot remaining at either of these two states is 1, i.e.,

$$f(0,\pm 1,0) = f(5,\pm 1,5) = 1$$

Also from the reward function as given in Equation (6.23), we have

$$\begin{aligned} \rho(0,1,0) &= 0 & \rho(0,-1,0) &= 0 & \rho(1,1,2) &= 0 \\ \rho(1,-1,0) &= 1 & \rho(2,-1,1) &= 0 & \rho(2,1,2) &= 0 \\ \rho(3,-1,2) &= 0 & \rho(3,1,4) &= 0 & \rho(4,1,5) &= 5 \\ \rho(4,-1,3) &= 0 & \rho(5,1,5) &= 0 & \rho(5,-1,5) &= 0 \end{aligned}$$

Applying the Bellman Equation yields

$$\begin{aligned}
Q^{\pi}(2,1) &= \sum_{s'} P_{ss'}^{a}\left(\rho(s,a,s')+\gamma Q^{\pi}\left(s',a'\right)\right) \\
&= \sum_{j=1}^{3} P_{2j}^{1}\left(\rho(2,1,j)+\gamma Q^{\pi}\left(j,\pi(j)\right)\right) \\
&= P_{21}^{1}\left(\rho(2,1,1)+\gamma Q^{\pi}\left(1,-1\right)\right) \text{ (Note: } j=1 \text{ and } \pi(1)=-1) \\
&+ P_{22}^{1}\left(\rho(2,1,2)+\gamma Q^{\pi}\left(2,1\right)\right) \text{ (Note: } j=2 \text{ and } \pi(2)=1) \\
&+ P_{23}^{1}\left(\rho(2,1,3)+\gamma Q^{\pi}\left(3,1\right)\right) \text{ (Note: } j=3 \text{ and } \pi(3)=1) \\
&= 0.05\cdot\left(0+0.5Q^{\pi}\left(1,-1\right)\right) \\
&+ 0.15\cdot\left(0+0.5Q^{\pi}\left(2,1\right)\right) \\
&+ 0.80\cdot\left(0+0.5Q^{\pi}\left(3,1\right)\right) \\
&= 0.025Q^{\pi}(1,-1)+0.075Q^{\pi}(2,1)+0.4Q^{\pi}(3,1) \qquad (6.24)
\end{aligned}$$

$$\begin{aligned}
Q^{\pi}(1,-1) &= \sum_{s'} P_{ss'}^{a}\left(\rho(s,a,s')+\gamma Q^{\pi}\left(s',a'\right)\right) \\
&= \sum_{j=0}^{2} P_{1j}^{-1}\left(\rho(1,-1,j)+\gamma Q^{\pi}\left(j,\pi(j)\right)\right) \\
&= P_{10}^{-1}\left(\rho(1,-1,0)+\gamma Q^{\pi}\left(0,\pm 1\right)\right) \quad \text{(Note: } j=0 \text{ and } Q^{\pi}(0,\pm 1)=0) \\
&+ P_{11}^{-1}\left(\rho(1,-1,1)+\gamma Q^{\pi}\left(1,-1\right)\right) \quad \text{(Note: } j=1 \text{ and } \pi(1)=-1) \\
&+ P_{12}^{-1}\left(\rho(1,-1,2)+\gamma Q^{\pi}\left(2,1\right)\right) \quad \text{(Note: } j=2 \text{ and } \pi(2)=1) \\
&= 0.8\cdot 1+0.15\cdot\left(0+0.5Q^{\pi}(1,-1)\right)+0.05\cdot\left(0+0.5Q^{\pi}(2,1)\right) \\
&= 0.8+0.075Q^{\pi}(1,-1)+0.025Q^{\pi}(2,1) \qquad (6.25)
\end{aligned}$$

$$\begin{aligned}
Q^{\pi}(3,1) &= \sum_{s'} P_{ss'}^{a}\left(\rho(s,a,s')+\gamma Q^{\pi}\left(s',a'\right)\right) \\
&= \sum_{j=2}^{4} P_{3j}^{1}\left(\rho(3,1,j)+\gamma Q^{\pi}\left(j,\pi(j)\right)\right) \\
&= P_{32}^{1}\left(\rho(3,1,2)+\gamma Q^{\pi}\left(2,1\right)\right) \\
&+ P_{33}^{1}\left(\rho(3,1,3)+\gamma Q^{\pi}\left(3,1\right)\right)
\end{aligned}$$

$$
\begin{aligned}
&+ \quad P_{34}^{1}\Big(\rho(3,1,4)+\gamma Q^{\pi}(4,1)\Big) \\
&= \quad 0.05\cdot\Big(0+0.5Q^{\pi}(2,1)\Big) \\
&+ \quad 0.15\cdot\Big(0+0.5Q^{\pi}(3,1)\Big) \\
&+ \quad 0.8\cdot\Big(0+0.5Q^{\pi}(4,1)\Big) \\
&= \quad 0.025Q^{\pi}(2,1)+0.075Q^{\pi}(3,1)+0.4Q^{\pi}(4,1) \qquad (6.26)
\end{aligned}
$$

$$
\begin{aligned}
Q^{\pi}(4,1) &= \sum_{s'} P_{ss'}^{a}\Big(\rho(s,a,s')+\gamma Q^{\pi}\left(s',a'\right)\Big) \\
&= \sum_{j=3}^{5} P_{4j}^{1}\Big(\rho(4,1,j)+\gamma Q^{\pi}\left(j,\pi(j)\right)\Big) \\
&= P_{43}^{1}\Big(\rho(4,1,3)+\gamma Q^{\pi}\left(3,1\right)\Big) \\
&+ P_{44}^{1}\Big(\rho(4,1,4)+\gamma Q^{\pi}\left(4,1\right)\Big) \\
&+ P_{45}^{1}\Big(\rho(4,1,5)+\gamma Q^{\pi}\left(5,\pm 1\right)\Big) \quad \text{(Note: } Q^{\pi}(5,\pm 1)=0\text{)} \\
&= 0.05\cdot\Big(0+0.5Q^{\pi}(3,1)\Big)+0.15\cdot\Big(0+0.5Q^{\pi}(4,1)\Big)+0.8\cdot 5 \\
&= 0.025Q^{\pi}(3,1)+0.075Q^{\pi}(4,1)+4 \qquad (6.27)
\end{aligned}
$$

Note that the values for $Q^{\pi}(0,\pm 1)$ and $Q^{\pi}(5,\pm 1)$ are 0 because the rewards $\rho(0,\pm 1,0)$ and $\rho(5,\pm 1,5)$ are 0 and the probability of transitioning out of these states by taking the actions $a=\pm 1$ are also 0.

Equations (6.24), (6.25), (6.26), and (6.27) are four linear equations containing four unknowns, namely, $Q^{\pi}(1,-1)$, $Q^{\pi}(2,1)$, $Q^{\pi}(3,1)$, and $Q^{\pi}(4,1)$. They can be solved simultaneously to obtain $Q^{\pi}(2,1)$. With the transition probabilities as stated earlier, the four linear equations can be written as:

$$
\begin{aligned}
Q^{\pi}(2,1) &= 0.025Q^{\pi}(1,-1)+0.075Q^{\pi}(2,1)+0.4Q^{\pi}(3,1) \\
Q^{\pi}(1,-1) &= 0.8+0.075Q^{\pi}(1,-1)+0.025Q^{\pi}(2,1) \\
Q^{\pi}(3,1) &= 0.025Q^{\pi}(2,1)+0.075Q^{\pi}(3,1)+0.4Q^{\pi}(4,1) \\
Q^{\pi}(4,1) &= 0.025Q^{\pi}(3,1)+0.075Q^{\pi}(4,1)+4 \qquad (6.28)
\end{aligned}
$$

Solving the set of equations in (6.28) yields,

$$
Q^{\pi}(1,-1) \quad = \quad 0.888
$$

$$\begin{aligned} Q^{\pi}(2,1) &= 0.852 \\ Q^{\pi}(3,1) &= 1.915 \\ Q^{\pi}(4,1) &= 4.376 \end{aligned} \tag{6.29}$$

Hence, the answer is: $Q^{\pi}(2,1) = 0.852$. □

We can use the Q-function to establish a partial order[3] over a set of policies. For example, for two policies π_1 and π_2, we say that $\pi_1 \geq \pi_2$ (i.e., π_1 is "better" than π_2) if and only if $Q^{\pi_1}(s,a) \geq Q^{\pi_2}(s,a)$ for all $s \in S$ and $a \in A$. For a given Markov decision process, the number of possible policies is finite. The policy whose Q-function values are greater or equal to that of all other policies (for all possible state-action pairs) is called an *optimal policy*, and this Q-function is called the *optimal Q-function*, denoted by Q^*, i.e.,

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a), \text{ for } s \in S \text{ and } a \in A \tag{6.30}$$

It is known that (i) the value of $Q^*(s,a)$ is unique, and (ii) it is possible that more than one policy can achieve the same optimal values of the Q-function [7].

Solution approaches for reinforcement learning problems can be classified into two types. The first is for determining an optimal policy under the condition that a model of the system is available; that is, all the state-transition probabilities $f(s,a,s')$ and the reward function $\rho(s,a,s')$ are known *a priori*. This type of solution approach is discussed in Chapter 7.

The second type of solution approach concerns the situation where the states of a Markov model is defined but all other information must be "learned" while the agent tries to find an optimal policy. This type of solution approach (called model-free reinforcement learning) is discussed in Chapter 8.

[3] A relation $\leq$ is called a *partial order* on a set X if it has: (1) reflexivity: $a \leq a$ for all $a \in X$; (2) antisymmetry: $a \leq b$ and $b \leq a$ implies $a = b$; and (3) transitivity: $a \leq b$ and $b \leq c$ implies $a \leq c$. An intuitive way to interpret a partial order is that it is a ranking of a set of entities in a certain way.

Chapter 7

Dynamic Programming

Example 6.6 demonstrates that exact calculation of the values of the Q-function (for a given policy) by using the Bellman Equation involves solving a set of linear equations, one for each state under the policy. For practical reinforcement learning problems, this solution approach may not be feasible for the following two main reasons:

1. The dynamics of the system as described by the state-transition probabilities may not be known *a priori*.

2. The computational resources required for solving the Bellman Equation may be prohibitively substantial, because this approach is akin to an exhaustive search through the entire state set. For example, in the game of backgammon [9], the number of states is in the order of 10^{20}.

There is an iterative algorithm that can be used to find such values more efficiently. This algorithm has the added advantages of not only enabling the determination of an optimal policy for a Markov decision process, but also serving as the basis for a technique (called *Q-learning*, discussed in Chapter 8) for finding an optimal policy when the state-transition probabilities are not known *a priori*.

Richard Bellman called this iterative approach *Dynamics Programming* when he pioneered it in the 1950s [1]. The origination of this name appears to have an interesting historical context, as described in the following excerpt from Bellman's autobiography [2]:

"An interesting question is, Where did the name, dynamic programming, come from? The 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word, research. I'm not using the term lightly; I'm using it precisely. His face would suffuse, he would turn red, and he would get violent if people used the term, research, in his presence. You can imagine

how he felt, then, about the term, mathematical. The RAND Corporation was employed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation. What title, what name, could I choose? In the first place I was interested in planning, in decision making, in thinking. But planning, is not a good word for various reasons. I decided therefore to use the word, "programming". I wanted to get across the idea that this was dynamic, this was multistage, this was time-varying — I thought, let's kill two birds with one stone. Let's take a word that has an absolutely precise meaning, namely dynamic, in the classical physical sense. It also has a very interesting property as an adjective, and that is it's impossible to use the word, dynamic, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. Its impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities." (p. 159). □

Solving the shortest-path problem by the approach as discussed in Example 7.3 can be computationally intensive if the number of cities involved is large. The approach of Dynamic Programming offers an efficient alternative. There are two main algorithms for doing this. One is called *value iteration*; the other *policy iteration*. In this module, we will focus on value iteration.

7.1 Value iteration

The central idea of value iteration is to turn the Bellman Equation (6.22) into an update rule in the evaluation of the Q-function, such that, by repeated application of this rule, the iterative process will converge to yield the optimal value Q^*, from which an optimal policy π^* can be directly identified.

7.1.1 Iterative algorithm

Table 7.1 states this algorithm, while the detailed calculations are illustrated in Example 7.1 (for the case of deterministic state-transitions) and Example 7.2 (for the case of non-deterministic state-transitions).

In practice, the algorithm can be stopped once the change in the values of the Q-function between two consecutive iterations is below a pre-defined threshold.

Example 7.1 Consider the cleaning task described earlier in Example 6.1, with the following deterministic state transition function $\bar{f}$ and reward function ρ:

$$\bar{f}(s,a) = \begin{cases} s+a & \text{if } 1 \leq s \leq 4 \\ s & \text{if } s = 0 \text{ or } s = 5 \end{cases} \tag{7.1}$$

Table 7.1: Algorithm for value iteration.

Input:	state-transition probability f reward function ρ discount factor γ
	Initialize Q-function, e.g., $Q_0 \leftarrow 0$
	Repeat for each l
	$Q \leftarrow Q_l$
	For every (s,a) do
	$Q(s,a) \leftarrow \sum_{s'} P_{ss'}^{a} \left(\rho(s,a,s') + \gamma \max_{a'} Q\left(s',a'\right) \right)$
	End for-loop
	$Q_{l+1} \leftarrow Q$
	Until $Q_{l+1} = Q_l$
Output:	$Q^* = Q_l$

$$\rho(s,a,s') = \begin{cases} 5 & \text{if } s \neq 5 \text{ and } s' = 5 \\ 1 & \text{if } s \neq 0 \text{ and } s' = 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

Let $\gamma = 0.5$. Calculate the optimal values of the Q-function using the iterative algorithm.

Solution: For deterministic state transitions, the update rule in the iterative algorithm degenerates to

$$Q(s,a) = \rho(s,a,s') + \gamma \max_{a'} Q\left(\bar{f}(s,a),a'\right) \tag{7.3}$$

The iterative algorithm proceeds as follows. First, the values for the Q-function are initialized to be identically 0 for all states and actions, i.e.,

$$Q_0(s,a) = 0 \quad \text{for} \quad s = 0,1,2,3,4,5 \ \text{ and } \ a = \pm 1 \tag{7.4}$$

The detailed steps of calculation for the first two iterations are shown below.

Iteration 1

$$\begin{aligned} Q(0,1) &= \rho(0,1,0) + 0.5 \max_{a'} Q\left(\bar{f}(0,1),a'\right) \quad \left(\text{Note: } \bar{f}(0,1) = 0\right) \\ &= \rho(0,1,0) + 0.5 \max \underbrace{\left[Q\left(0,-1\right), Q\left(0,1\right)\right]}_{\text{for } a' = \pm 1 \text{ at state } 0} \end{aligned}$$

$$\begin{aligned} &= 0+0.5\cdot\max\left[0,0\right] \quad \left(\text{Note: } \max[\ldots] \text{ means the maximum in } [\ ]\right) \\ &= 0 \end{aligned} \tag{7.5}$$

$$\begin{aligned} Q(0,-1) &= \rho(0,-1,0)+0.5\max_{a'}Q\left(\bar{f}(0,-1),a'\right) \quad \left(\text{Note: } \bar{f}(0,-1)=0\right) \\ &= \rho(0,-1,0)+0.5\max\left[Q\left(0,-1\right),Q\left(0,1\right)\right] \\ &= 0+0.5\max\left[0,0\right] \\ &= 0 \end{aligned} \tag{7.6}$$

$$\begin{aligned} Q(1,1) &= \rho(1,1,2)+0.5\max_{a'}Q\left(\bar{f}(1,1),a'\right) \quad \left(\text{Note: } \bar{f}(1,1)=2\right) \\ &= \rho(1,1,2)+0.5\max\underbrace{\left[Q\left(2,-1\right),Q\left(2,1\right)\right]}_{\text{for } a'=\pm 1 \text{ at state } 2} \\ &= 0+0.5\cdot\max\left[0,0\right] \\ &= 0 \end{aligned} \tag{7.7}$$

$$\begin{aligned} Q(1,-1) &= \rho(1,-1,0)+0.5\max_{a'}Q\left(\bar{f}(1,-1),a'\right) \quad \left(\text{Note: } \bar{f}(1,-1)=0\right) \\ &= \rho(1,-1,0)+0.5\max\left[Q\left(0,-1\right),Q\left(0,1\right)\right] \\ &= 1+0.5\cdot\max\left[0,0\right] \\ &= 1 \end{aligned} \tag{7.8}$$

$$\begin{aligned} Q(2,1) &= \rho(2,1,3)+0.5\max_{a'}Q\left(\bar{f}(2,1),a'\right) \quad \left(\text{Note: } \bar{f}(2,1)=3\right) \\ &= \rho(2,1,3)+0.5\max\left[Q\left(3,-1\right),Q\left(3,1\right)\right] \\ &= 0+0.5\cdot\max\left[0,0\right] \\ &= 0 \end{aligned} \tag{7.9}$$

$$\begin{aligned} Q(2,-1) &= \rho(2,-1,1)+0.5\max_{a'}Q\left(\bar{f}(2,-1),a'\right) \quad \left(\text{Note: } \bar{f}(2,-1)=1\right) \\ &= \rho(2,-1,1)+0.5\max\left[Q\left(1,-1\right),Q\left(1,1\right)\right] \\ &= 0+0.5\cdot\max\left[1,0\right] \\ &= 0.5 \end{aligned} \tag{7.10}$$

Note that in Equation (7.10) above we use the updated value of $\mathbf{Q(1,-1)=1}$

as calculated in Equation (7.8), and <u>not</u> $\mathbf{Q(1,-1)=0}$ as initially assigned in Equation (7.4) before the start of the iteration. This is an asynchronous implementation of Dynamic Programming.

$$\begin{aligned} Q(3,1) &= \rho(3,1,4)+0.5\max_{a'}Q\left(\bar{f}(3,1),a'\right) \quad \left(\text{Note: } \bar{f}(3,1)=4\right) \\ &= \rho(3,1,4)+0.5\max\left[Q(4,-1),Q(4,1)\right] \\ &= 0+0.5\cdot\max\left[0,0\right] \\ &= 0 \end{aligned} \tag{7.11}$$

$$\begin{aligned} Q(3,-1) &= \rho(3,-1,2)+0.5\max_{a'}Q\left(\bar{f}(3,-1),a'\right) \quad \left(\text{Note: } \bar{f}(3,-1)=2\right) \\ &= \rho(3,-1,2)+0.5\max\left[Q(2,-1),Q(2,1)\right] \\ &= 0+0.5\cdot\max\left[0.5,0\right] \\ &= 0.25 \end{aligned} \tag{7.12}$$

$$\begin{aligned} Q(4,1) &= \rho(4,1,5)+0.5\max_{a'}Q\left(\bar{f}(4,1),a'\right) \quad \left(\text{Note: } \bar{f}(4,1)=5\right) \\ &= \rho(4,1,5)+0.5\max\left[Q(5,-1),Q(5,1)\right] \\ &= 5+0.5\cdot\max\left[0,0\right] \\ &= 5 \end{aligned} \tag{7.13}$$

$$\begin{aligned} Q(4,-1) &= \rho(4,-1,3)+0.5\max_{a'}Q\left(\bar{f}(4,-1),a'\right) \quad \left(\text{Note: } \bar{f}(4,-1)=3\right) \\ &= \rho(4,-1,3)+0.5\max\left[Q(3,-1),Q(3,1)\right] \\ &= 0+0.5\cdot\max\left[0.25,0\right] \\ &= 0.125 \end{aligned} \tag{7.14}$$

$$\begin{aligned} Q(5,1) &= \rho(5,1,5)+0.5\max_{a'}Q\left(\bar{f}(5,1),a'\right) \quad \left(\text{Note: } \bar{f}(5,1)=5\right) \\ &= \rho(5,1,5)+0.5\max\left[Q(5,-1),Q(5,1)\right] \\ &= 0+0.5\cdot\max\left[0,0\right] \\ &= 0 \end{aligned} \tag{7.15}$$

$$\begin{aligned} Q(5,-1) &= \rho(5,-1,5)+0.5\max_{a'} Q\left(\bar{f}(5,-1),a'\right) \quad \left(\text{Note: } \bar{f}(5,-1)=5\right) \\ &= \rho(5,-1,5)+0.5\max\left[Q(5,-1),Q(5,1)\right] \\ &= 0+0.5\max\left[0,0\right] \\ &= 0 \end{aligned} \tag{7.16}$$

Iteration 2

$$\begin{aligned} Q(0,1) &= \rho(0,1,0)+0.5\max_{a'} Q\left(\bar{f}(0,1),a'\right) \\ &= \rho(0,1,0)+0.5\max\left[Q(0,-1),Q(0,1)\right] \\ &= 0+0.5\cdot\max\left[0,0\right] \\ &= 0 \end{aligned} \tag{7.17}$$

$$\begin{aligned} Q(0,-1) &= \rho(0,-1,0)+0.5\max_{a'} Q\left(\bar{f}(0,-1),a'\right) \\ &= \rho(0,-1,0)+0.5\max\left[Q(0,-1),Q(0,1)\right] \\ &= 0+0.5\max\left[0,0\right] \\ &= 0 \end{aligned} \tag{7.18}$$

$$\begin{aligned} Q(1,1) &= \rho(1,1,2)+0.5\max_{a'} Q\left(\bar{f}(1,1),a'\right) \\ &= \rho(1,1,2)+0.5\max\left[Q(2,-1),Q(2,1)\right] \\ &= 0+0.5\cdot\max\left[0.5,0\right] \\ &= 0.25 \end{aligned} \tag{7.19}$$

$$\begin{aligned} Q(1,-1) &= \rho(1,-1,0)+0.5\max_{a'} Q\left(\bar{f}(1,-1),a'\right) \\ &= \rho(1,-1,0)+0.5\max\left[Q(0,-1),Q(0,1)\right] \\ &= 1+0.5\cdot\max\left[0,0\right] \\ &= 1 \end{aligned} \tag{7.20}$$

$$\begin{aligned} Q(2,1) &= \rho(2,1,3)+0.5\max_{a'} Q\left(\bar{f}(2,1),a'\right) \\ &= \rho(2,1,3)+0.5\max\left[Q(3,-1),Q(3,1)\right] \\ &= 0+0.5\cdot\max\left[0.25,0\right] \\ &= 0.125 \end{aligned} \tag{7.21}$$

$$\begin{aligned} Q(2,-1) &= \rho(2,-1,1)+0.5\max_{a'} Q\left(\bar{f}(2,-1),a'\right) \\ &= \rho(2,-1,1)+0.5\max\left[Q(1,-1),Q(1,1)\right] \\ &= 0+0.5\cdot\max\left[1,0.25\right] \\ &= 0.5 \end{aligned} \tag{7.22}$$

$$\begin{aligned} Q(3,1) &= \rho(3,1,4)+0.5\max_{a'} Q\left(\bar{f}(3,1),a'\right) \\ &= \rho(3,1,4)+0.5\max\left[Q(4,-1),Q(4,1)\right] \\ &= 0+0.5\cdot\max\left[0.125,5\right] \\ &= 2.5 \end{aligned} \tag{7.23}$$

$$\begin{aligned} Q(3,-1) &= \rho(3,-1,2)+0.5\max_{a'} Q\left(\bar{f}(3,-1),a'\right) \\ &= \rho(3,-1)+0.5\max\left[Q(2,-1),Q(2,1)\right] \\ &= 0+0.5\cdot\max\left[0.5,0.125\right] \\ &= 0.25 \end{aligned} \tag{7.24}$$

$$\begin{aligned} Q(4,1) &= \rho(4,1,5)+0.5\max_{a'} Q\left(\bar{f}(4,1),a'\right) \\ &= \rho(4,1,5)+0.5\max\left[Q(5,-1),Q(5,1)\right] \\ &= 5+0.5\cdot\max\left[0,0\right] \\ &= 5 \end{aligned} \tag{7.25}$$

$$\begin{aligned} Q(4,-1) &= \rho(4,-1,3)+0.5\max_{a'} Q\left(\bar{f}(4,-1),a'\right) \\ &= \rho(4,-1,3)+0.5\max\left[Q(3,-1),Q(3,1)\right] \\ &= 0+0.5\cdot\max\left[0.25,2.5\right] \\ &= 1.25 \end{aligned} \tag{7.26}$$

$$\begin{aligned} Q(5,1) &= \rho(5,1,5)+0.5\max_{a'} Q\left(\bar{f}(5,1),a'\right) \\ &= \rho(5,1)+0.5\max\left[Q(5,-1),Q(5,1)\right] \\ &= 0+0.5\cdot\max\left[0,0\right] \\ &= 0 \end{aligned} \tag{7.27}$$

$$\begin{aligned} Q(5,-1) &= \rho(5,-1,5)+0.5\max_{a'} Q\left(\bar{f}(5,-1),a'\right) \\ &= \rho(5,-1)+0.5\max\left[Q(5,-1),Q(5,1)\right] \\ &= 0+0.5\max\left[0,0\right] \\ &= 0 \end{aligned} \tag{7.28}$$

The iterations are carried out until the values of the Q-function converge to the optimal values, which are shown in Table 7.2. The optimal values for the Q-function are achieved after five iterations, i.e., $Q^* = Q_5$. The first row of Table 7.2 lists all six states (i.e., positions in the grid) of the robot. The five rows below the row of initial values (i.e., Q_0) show the values of the Q-function for five iterations, with the two numbers (separated by a vertical bar "|") in each state column indicate the values of the Q-function when taking action $a=-1$ and $a=1$ at that state. For instance, the two numbers for Q_2 under state 1 are 1.000 and 0.250. This means that $Q_2(1,-1)=1.000$ and $Q_2(1,1)=0.250$.

Table 7.2: Values of Q-function for robot with deterministic state transitions.

State	0	1	2	3	4	5
Q_0	0.000|0.000	0.000|0.000	0.000|0.000	0.000|0.000	0.000|0.000	0.000|0.000
Q_1	0.000|0.000	1.000|0.000	0.500|0.000	0.250|0.000	0.125|5.000	0.000|0.000
Q_2	0.000|0.000	1.000|0.250	0.500|0.125	0.250|0.250	1.250|5.000	0.000|0.000
Q_3	0.000|0.000	1.000|0.250	0.500|1.250	0.625|2.500	1.250|5.000	0.000|0.000
Q_4	0.000|0.000	1.000|0.625	0.500|1.250	0.625|2.500	1.250|5.000	0.000|0.000
Q_5	0.000|0.000	1.000|0.625	0.500|1.250	0.625|2.500	1.250|5.000	0.000|0.000

□

Example 7.2 Consider the problem described in Example 7.1. Suppose that the behavior of the robot is now stochastic with the state transition probabilities as given in Table 7.3 below. Calculate the optimal values of the Q-function using the iterative algorithm.

Solution: The solution process is similar to that shown in Example 7.1. The difference is that now when updating the values for the Q-function we need to take into account the stochastic state transitions by using the general form of the Bellman Equation:

$$Q(s,a) = \sum_{s'} P^a_{ss'}\left(\rho(s,a,s')+\gamma\max_{a'} Q\left(s',a'\right)\right) \tag{7.29}$$

where the values for $P^a_{ss'} \equiv f(s,a,s')$ are as given in Table 7.3. The initial values are set to identically 0, i.e., $Q_0(s,a)=0$ for $s=0,1,2,3,4,5$ and $a=\pm 1$.

Table 7.3: State-transition probabilities.

(s,a)	$f(s,a,0)$	$f(s,a,1)$	$f(s,a,2)$	$f(s,a,3)$	$f(s,a,4)$	$f(s,a,5)$
$(0,-1)$	1	0	0	0	0	0
$(1,-1)$	0.8	0.15	0.05	0	0	0
$(2,-1)$	0	0.8	0.15	0.05	0	0
$(3,-1)$	0	0	0.8	0.15	0.05	0
$(4,-1)$	0	0	0	0.8	0.15	0.05
$(5,-1)$	0	0	0	0	0	1
$(0,1)$	1	0	0	0	0	0
$(1,1)$	0.05	0.15	0.8	0	0	0
$(2,1)$	0	0.05	0.15	0.8	0	0
$(3,1)$	0	0	0.05	0.15	0.8	0
$(4,1)$	0	0	0	0.05	0.15	0.8
$(5,1)$	0	0	0	0	0	1

Iteration 1:

$$\begin{aligned}
Q(0,-1) &= \sum_{s'} P_{0s'}^{-1}\left(\rho(0,-1,s')+\gamma\max_{a'} Q\left(s',a'\right)\right) \\
&= \sum_{j=0}^{0} P_{0j}^{-1}\left(\rho(0,-1,j)+\gamma\max_{a'} Q\left(j,a'\right)\right) \\
&= P_{00}^{-1}\left(\rho(0,-1,0)+\gamma\max_{a'} Q\left(0,a'\right)\right) \text{ (Since 0 is the only } s') \\
&= 1\cdot\left(0+0.5\max\left[Q(0,-1),Q(0,1)\right]\right) \\
&= 0.5\max\left[0,0\right] \\
&= 0 \qquad (7.30)
\end{aligned}$$

$$\begin{aligned}
Q(0,1) &= \sum_{s'} P_{0s'}^{1}\left(\rho(0,1,s')+\gamma\max_{a'} Q\left(s',a'\right)\right) \\
&= \sum_{j=0}^{0} P_{0j}^{1}\left(\rho(0,1,j)+\gamma\max_{a'} Q\left(j,a'\right)\right) \\
&= P_{00}^{1}\left(\rho(0,1,0)+\gamma\max_{a'} Q\left(0,a'\right)\right) \text{ (Since 0 is the only } s') \\
&= 1\cdot\left(0+0.5\max\left[Q(0,-1),Q(0,1)\right]\right) \\
&= 0.5\max\left[0,0\right] \\
&= 0 \qquad (7.31)
\end{aligned}$$

$$Q(1,-1) = \sum_{s'} P_{1s'}^{-1}\left(\rho(1,-1,s')+\gamma\max_{a'} Q\left(s',a'\right)\right)$$

$$
\begin{aligned}
&= \sum_{j=0}^{2} P_{1j}^{-1}\left(\rho(1,-1,j)+\gamma\max_{a'} Q\left(j,a'\right)\right)\\
&= P_{10}^{-1}\left(\rho(1,-1,0)+\gamma\max_{a'} Q\left(0,a'\right)\right)\ \ \left(\text{Note: } j=0\right)\\
&+ P_{11}^{-1}\left(\rho(1,-1,1)+\gamma\max_{a'} Q\left(1,a'\right)\right)\ \ \left(\text{Note: } j=1\right)\\
&+ P_{12}^{-1}\left(\rho(1,-1,2)+\gamma\max_{a'} Q\left(2,a'\right)\right)\ \ \left(\text{Note: } j=2\right)\\
&= 0.8\cdot\left(1+0.5\cdot\max\left[Q(0,-1),Q(0,1)\right]\right)\ \ \left(\text{Note: } \rho(1,-1,0)=1\ \right)\\
&+ 0.15\cdot\left(0+0.5\cdot\max\left[Q(1,-1),Q(1,1)\right]\right)\\
&+ 0.05\cdot\left(0+0.5\cdot\max\left[Q(2,-1),Q(2,1)\right]\right)\\
&= 0.8\cdot\left(1+0.5\cdot\max\left[0,0\right]\right)+0.15\cdot 0.5\cdot\max\left[0,0\right]+0.05\cdot 0.5\cdot\max\left[0,0\right]\\
&= 0.8 \qquad (7.32)
\end{aligned}
$$

$$
\begin{aligned}
Q(1,1) &= \sum_{s'} P_{1s'}^{1}\left(\rho(1,1,s')+\gamma\max_{a'} Q\left(s',a'\right)\right)\\
&= \sum_{j=0}^{2} P_{1j}^{1}\left(\rho(1,1,j)+\gamma\max_{a'} Q\left(j,a'\right)\right)\\
&= P_{10}^{1}\left(\rho(1,1,0)+\gamma\max_{a'} Q\left(0,a'\right)\right)\ \ \left(\text{Note: } j=0\right)\\
&+ P_{11}^{1}\left(\rho(1,1,1)+\gamma\max_{a'} Q\left(1,a'\right)\right)\ \ \left(\text{Note: } j=1\right)\\
&+ P_{12}^{1}\left(\rho(1,1,2)+\gamma\max_{a'} Q\left(2,a'\right)\right)\ \ \left(\text{Note: } j=2\right)\\
&= 0.05\cdot\left(1+0.5\cdot\max\left[Q(0,-1),Q(0,1)\right]\right)\ \ \left(\text{Note: } \rho(1,1,0)=1\right)\\
&+ 0.15\cdot\left(0+0.5\cdot\max\left[Q(1,-1),Q(1,1)\right]\right)\ \ \left(\text{Note: } Q(1,-1)=0.8\right)\\
&+ 0.8\cdot\left(0+0.5\cdot\max\left[Q(2,-1),Q(2,1)\right]\right)\\
&= 0.05\left(1+0.5\cdot\max\left[0,0\right]\right)+0.15\cdot 0.5\cdot\max\left[0.8,0\right]+0.8\cdot 0.5\cdot\max\left[0,0\right]\\
&= 0.11 \qquad (7.33)
\end{aligned}
$$

Note that in Equation (7.33) above, we use the $\mathbf{Q(1,-1)}$ value just updated from Equation (7.32), i.e., $\mathbf{Q(1,-1)=0.8}$, instead of $\mathbf{Q(1,-1)=0}$ as initially assigned.

$$
Q(2,-1) = \sum_{s'} P_{2s'}^{-1}\left(\rho(2,-1,s')+\gamma\max_{a'} Q\left(s',a'\right)\right)
$$

$$
\begin{aligned}
&= \sum_{j=1}^{3} P_{2j}^{-1}\left(\rho(2,-1,j)+\gamma\max_{a'} Q\left(j,a'\right)\right)\\
&= P_{21}^{-1}\left(\rho(2,-1,1)+\gamma\max_{a'} Q\left(1,a'\right)\right)\quad\left(\text{Note: } j=1\right)\\
&+ P_{22}^{-1}\left(\rho(2,-1,2)+\gamma\max_{a'} Q\left(2,a'\right)\right)\quad\left(\text{Note: } j=2\right)\\
&+ P_{23}^{-1}\left(\rho(2,-1,3)+\gamma\max_{a'} Q\left(3,a'\right)\right)\quad\left(\text{Note: } j=3\right)\\
&= 0.8\cdot\left(0+0.5\cdot\max\left[Q(1,-1),Q(1,1)\right]\right)\\
&+ 0.15\cdot\left(0+0.5\cdot\max\left[Q(2,-1),Q(2,1)\right]\right)\\
&+ 0.05\cdot\left(0+0.5\cdot\max\left[Q(3,-1),Q(3,1)\right]\right)\\
&= 0.8\cdot 0.5\cdot\max\left[0.8,0.11\right]+0+0\\
&= 0.32
\end{aligned}
\tag{7.34}
$$

$$
\begin{aligned}
Q(2,1) &= \sum_{s'} P_{2s'}^{1}\left(\rho(2,1,s')+\gamma\max_{a'} Q\left(s',a'\right)\right)\\
&= \sum_{j=1}^{3} P_{2j}^{1}\left(\rho(2,1,j)+\gamma\max_{a'} Q\left(j,a'\right)\right)\\
&= P_{21}^{1}\left(\rho(2,1,1)+\gamma\max_{a'} Q\left(1,a'\right)\right)\\
&+ P_{22}^{1}\left(\rho(2,1,2)+\gamma\max_{a'} Q\left(2,a'\right)\right)\\
&+ P_{23}^{1}\left(\rho(2,1,3)+\gamma\max_{a'} Q\left(3,a'\right)\right)\\
&= 0.05\cdot\left(0+0.5\cdot\max\left[Q(1,-1),Q(1,1)\right]\right)\\
&+ 0.15\cdot\left(0+0.5\cdot\max\left[Q(2,-1),Q(2,1)\right]\right)\\
&+ 0.8\cdot\left(0+0.5\cdot\max\left[Q(3,-1),Q(3,1)\right]\right)\\
&= 0.05\cdot 0.5\cdot\max\left[0.8,0.11\right]+0.15\cdot 0.5\cdot\max\left[0.32,0\right]+0\\
&= 0.044
\end{aligned}
\tag{7.35}
$$

$$
\begin{aligned}
Q(3,-1) &= \sum_{s'} P_{3s'}^{-1}\left(\rho(3,-1,s')+\gamma\max_{a'} Q\left(s',a'\right)\right)\\
&= \sum_{j=2}^{4} P_{3j}^{-1}\left(\rho(3,-1,j)+\gamma\max_{a'} Q\left(j,a'\right)\right)
\end{aligned}
$$

$$
\begin{aligned}
&= P_{32}^{-1}\left(\rho(3,-1,2)+\gamma\max_{a'}Q\left(2,a'\right)\right)\\
&+ P_{33}^{-1}\left(\rho(3,-1,3)+\gamma\max_{a'}Q\left(3,a'\right)\right)\\
&+ P_{34}^{-1}\left(\rho(3,-1,4)+\gamma\max_{a'}Q\left(4,a'\right)\right)\\
&= 0.8\cdot\left(0+0.5\cdot\max\left[Q(2,-1),Q(2,1)\right]\right)\\
&+ 0.15\cdot\left(0+0.5\cdot\max\left[Q(3,-1),Q(3,1)\right]\right)\\
&+ 0.05\cdot\left(0+0.5\cdot\max\left[Q(4,-1),Q(4,1)\right]\right)\\
&= 0.8\cdot 0.5\cdot\max\left[0.32,0.044\right]+0+0\\
&= 0.128
\end{aligned}
\tag{7.36}
$$

$$
\begin{aligned}
Q(3,1) &= \sum_{s'}P_{3s'}^{1}\left(\rho(3,1,s')+\gamma\max_{a'}Q\left(s',a'\right)\right)\\
&= \sum_{j=2}^{4}P_{3j}^{1}\left(\rho(3,1,j)+\gamma\max_{a'}Q\left(j,a'\right)\right)\\
&= P_{32}^{1}\left(\rho(3,1,2)+\gamma\max_{a'}Q\left(2,a'\right)\right)\\
&+ P_{33}^{1}\left(\rho(3,1,3)+\gamma\max_{a'}Q\left(3,a'\right)\right)\\
&+ P_{34}^{1}\left(\rho(3,1,4)+\gamma\max_{a'}Q\left(4,a'\right)\right)\\
&= 0.05\cdot\left(0+0.5\cdot\max\left[Q(2,-1),Q(2,1)\right]\right)\\
&+ 0.15\cdot\left(0+0.5\cdot\max\left[Q(3,-1),Q(3,1)\right]\right)\\
&+ 0.8\cdot\left(0+0.5\cdot\max\left[Q(4,-1),Q(4,1)\right]\right)\\
&= 0.05\cdot 0.5\cdot\max\left[0.32,0.044\right]+0.15\cdot 0.5\cdot\max\left[0.128,0\right]+0\\
&= 0.018
\end{aligned}
\tag{7.37}
$$

$$
\begin{aligned}
Q(4,-1) &= \sum_{s'}P_{4s'}^{-1}\left(\rho(4,-1,s')+\gamma\max_{a'}Q\left(s',a'\right)\right)\\
&= \sum_{j=3}^{5}P_{4j}^{-1}\left(\rho(4,-1,j)+\gamma\max_{a'}Q\left(j,a'\right)\right)\\
&= P_{43}^{-1}\left(\rho(4,-1,3)+\gamma\max_{a'}Q\left(3,a'\right)\right)\\
&+ P_{44}^{-1}\left(\rho(4,-1,4)+\gamma\max_{a'}Q\left(4,a'\right)\right)
\end{aligned}
$$

$$
\begin{aligned}
&+ \quad P_{45}^{-1}\left(\rho(4,-1,5)+\gamma\max_{a'}Q\left(5,a'\right)\right)\\
&= \quad 0.8\cdot\left(0+0.5\cdot\max\left[Q(3,-1),Q(3,1)\right]\right)\\
&+ \quad 0.15\cdot\left(0+0.5\cdot\max\left[Q(4,-1),Q(4,1)\right]\right)\\
&+ \quad 0.05\cdot\left(5+0.5\cdot\max\left[Q(5,-1),Q(5,1)\right]\right)\\
&= \quad 0.8\cdot 0.5\cdot\max\left[0.128,0.018\right]+0+0.05\cdot 5\\
&= \quad 0.301
\end{aligned}
\tag{7.38}
$$

$$
\begin{aligned}
Q(4,1) &= \sum_{s'}P_{4s'}^{1}\left(\rho(4,1,s')+\gamma\max_{a'}Q\left(s',a'\right)\right)\\
&= \sum_{j=3}^{5}P_{4j}^{1}\left(\rho(4,1,j)+\gamma\max_{a'}Q\left(j,a'\right)\right)\\
&= P_{43}^{1}\left(\rho(3,1,3)+\gamma\max_{a'}Q\left(3,a'\right)\right)\\
&+ P_{44}^{1}\left(\rho(4,1,4)+\gamma\max_{a'}Q\left(4,a'\right)\right)\\
&+ P_{45}^{1}\left(\rho(4,1,5)+\gamma\max_{a'}Q\left(5,a'\right)\right)\\
&= 0.05\cdot\left(0+0.5\cdot\max\left[Q(3,-1),Q(3,1)\right]\right)\\
&+ 0.15\cdot\left(0+0.5\cdot\max\left[Q(4,-1),Q(4,1)\right]\right)\\
&+ 0.8\cdot\left(5+0.5\cdot\max\left[Q(5,-1),Q(5,1)\right]\right)\\
&= 0.05\cdot 0.5\cdot\max\left[0.128,0.018\right]+0.15\cdot 0.5\cdot\max\left[0.301,0\right]+0.8\cdot 5\\
&= 4.026
\end{aligned}
\tag{7.39}
$$

$$
\begin{aligned}
Q(5,-1) &= \sum_{s'}P_{5s'}^{-1}\left(\rho(5,-1,s')+\gamma\max_{a'}Q\left(s',a'\right)\right)\\
&= \sum_{j=5}^{5}P_{5j}^{-1}\left(\rho(5,-1,j)+\gamma\max_{a'}Q\left(j,a'\right)\right)\\
&= P_{55}^{-1}\left(\rho(5,-1,5)+\gamma\max_{a'}Q\left(5,a'\right)\right)\\
&= P_{55}^{-1}\left(\rho(5,-1,j)+0.5\max\left[Q(5,-1),Q(5,1)\right]\right)\\
&= 1\cdot\left(0+0.5\max\left[0,0\right]\right)\\
&= 0
\end{aligned}
\tag{7.40}
$$

$$\begin{aligned} Q(5,1) &= \sum_{s'} P_{5s'}^{1}\left(\rho(5,1,s')+\gamma\max_{a'} Q\left(s',a'\right)\right) \\ &= \sum_{j=5}^{5} P_{5j}^{1}\left(\rho(5,1,j)+\gamma\max_{a'} Q\left(j,a'\right)\right) \\ &= P_{55}^{1}\left(\rho(5,1,5)+\gamma\max_{a'} Q\left(5,a'\right)\right) \\ &= P_{55}^{1}\left(\rho(5,1,j)+0.5\max\left[Q(5,-1),Q(5,1)\right]\right) \\ &= 1\cdot\left(0+0.5\max\left[0,0\right]\right) \\ &= 0 \end{aligned} \tag{7.41}$$

Such iteration repeats until the values of the Q-function converge to the optimal values after 22 iterations (i.e., $Q^* = Q_{22}$), which are shown in Table 7.4.

Table 7.4: Values of Q-function for non-deterministic state transitions.

State	0	1	2	3	4	5
Q_0	0.000|0.000	0.000|0.000	0.000|0.000	0.000|0.000	0.000|0.000	0.000|0.000
Q_1	0.000|0.000	0.800|0.110	0.320|0.044	0.128|0.018	0.301|4.026	0.000|0.000
Q_2	0.000|0.000	8.868|0.243	0.374|0.101	0.260|1.639	1.208|4.343	0.000|0.000
Q_3	0.000|0.000	0.874|0.265	0.419|0.709	0.515|1.878	1.327|4.373	0.000|0.000
Q_4	0.000|0.000	0.883|0.400	0.453|0.826	0.581|1.911	1.342|4.376	0.000|0.000
...	...	...	...	...	...	...
Q_{12}	0.000|0.000	0.888|0.458	0.467|0.852	0.594|1.915	1.344|4.376	0.000|0.000
...	...	...	...	...	...	...
Q_{22}	0.000|0.000	0.888|0.458	0.467|0.852	0.594|1.915	1.344|4.376	0.000|0.000

□

7.1.2 Convergence

Proving the convergence of the iterative algorithm given in Table 7.1 involves two steps. We will first show that values of the Q-function are bounded from one iteration to the next. We will then show that the difference between a current value and the optimal value of the Q-function decreases as the number of iterations increases, as is illustrated in Figure 7.1.

Step 1: Suppose that at iteration k the value of the Q-function is $Q_k(s,a)$ for some state s and action a. If $Q_k(s,a)$ is not the optimal value, i.e.,$Q_k(s,a) \neq Q^*(s,a)$, then we can write the maximum of the difference between $Q_k(s,a)$ and $Q^*(s,a)$ as

$$\delta_k = \max_{s,a} |Q^*(s,a) - Q_k(s,a)| \tag{7.42}$$

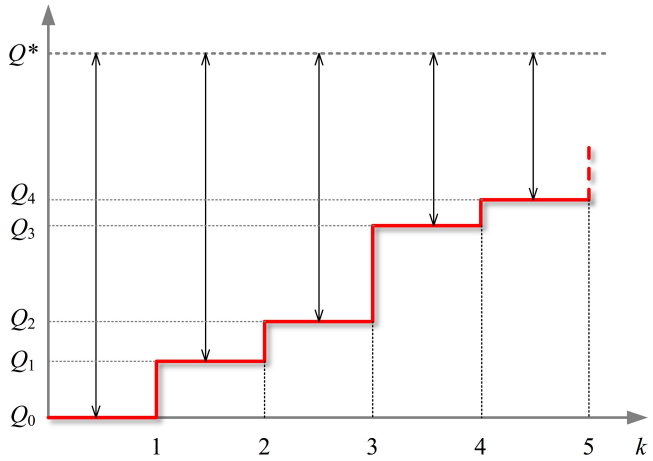


Figure 7.1: Convergence of value iteration algorithm.

Removing the maximum operator yields

$$|Q^*(s,a) - Q_k(s,a)| \leq \delta_k$$

From this inequality, we have

$$\begin{aligned}
& -\delta_k \leq Q^*(s,a) - Q_k(s,a) \leq \delta_k \\
\Rightarrow\ & -Q^*(s,a) - \delta_k \leq -Q_k(s,a) \leq -Q^*(s,a) + \delta_k \\
\Rightarrow\ & Q^*(s,a) + \delta_k \geq Q_k(s,a) \geq Q^*(s,a) - \delta_k \\
\Rightarrow\ & Q^*(s,a) - \delta_k \leq Q_k(s,a) \leq Q^*(s,a) + \delta_k
\end{aligned} \tag{7.43}$$

Now consider the iteration $(k+1)$. The update rule is:

$$Q_{k+1}(s,a) = \sum_{s'} P^a_{ss'} \left(\rho(s,a,s') + \gamma \max_{a'} Q_k\left(s',a'\right) \right) \tag{7.44}$$

Since Equation (7.43) holds for any given state-action pair (s',a'), we have

$$Q_k(s',a') \leq Q^*(s',a') + \delta_k$$

So

$$\begin{aligned}
Q_{k+1}(s,a) &= \sum_{s'} P^a_{ss'} \left(\rho(s,a,s') + \gamma \max_{a'} Q_k\left(s',a'\right) \right) \\
&\leq \sum_{s'} P^a_{ss'} \left(\rho(s,a,s') + \gamma \left(\max_{a'} Q^*\left(s',a'\right) + \delta_k \right) \right) \\
&= \sum_{s'} P^a_{ss'} \left(\rho(s,a,s') + \gamma \max_{a'} Q^*\left(s',a'\right) + \gamma\delta_k \right)
\end{aligned}$$

$$
\begin{aligned}
&= \underbrace{\sum_{s'} P_{ss'}^{a}\left(\rho(s,a,s')+\gamma \max_{a'} Q^{*}\left(s',a'\right)\right)}_{\text{This is } Q^{*}(s,a)} + \gamma\delta_k \\
&= Q^{*}(s,a)+\gamma\delta_k \qquad (7.45)
\end{aligned}
$$

that is,

$$Q_{k+1}(s,a) \leq Q^{*}(s,a)+\gamma\delta_k$$

Note that, as was mentioned earlier, $Q^{*}(s,a)$ is finite and unique. This means that the value of the Q-function in the new iteration $(k+1)$ is bounded if the value of δ_k is bounded. Since initially Q_0 is set to zero, and δ_k is the maximum absolute difference between Q and Q^{*} in each iteration, Q_{k+1} is also bounded.

Step 2: We will next show that the difference between a current value and the optimal value of the Q-function decreases as the number of iterations increases.

From Equation (7.43), we have

$$Q_k(s',a') \geq Q^{*}(s',a')-\delta_k$$

So

$$
\begin{aligned}
Q_{k+1}(s,a) &= \sum_{s'} P_{ss'}^{a}\left(\rho(s,a,s')+\gamma \max_{a'} Q_k\left(s',a'\right)\right) \\
&\geq \sum_{s'} P_{ss'}^{a}\left(\rho(s,a,s')+\gamma \max_{a'}\left(Q^{*}\left(s',a'\right)-\delta_k\right)\right) \\
&= \underbrace{\sum_{s'} P_{ss'}^{a}\left(\rho(s,a,s')+\gamma \max_{a'} Q^{*}\left(s',a'\right)\right)}_{\text{This is } Q^{*}(s,a)} - \gamma\delta_k \\
&= Q^{*}(s,a)-\gamma\delta_k \qquad (7.46)
\end{aligned}
$$

that is,

$$Q^{*}(s,a)-\gamma\delta_k \leq Q_{k+1}(s,a)$$

or

$$Q^{*}(s,a)-Q_{k+1}(s,a) \leq \gamma\delta_k$$

Therefore,

$$\delta_{k+1} \equiv \max_{s,a}|Q^{*}(s,a)-Q_{k+1}(s,a)| \leq \max_{s,a}|\gamma\,\delta_k| = \gamma\,\delta_k$$

i.e.,

$$\delta_{k+1} \leq \gamma\,\delta_k$$

Since $0 \leq \gamma < 1$, the error measure δ_k converges to 0 as k goes to infinity. In practice,

the iteration can be stopped when δ_k drops below a prescribed tolerance.

We have shown that the value iteration algorithm is guaranteed to converge to Q^*. Once Q^* is obtained, an optimal policy can be directly identified from Q^* based on the Principle of Optimality.

7.1.3 Bellman's Principle of Optimality

Principle of Optimality [1]: *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

The following example serves as an intuitive illustration of the Principle of Optimality.

Example 7.3 Suppose that we would like to find the shortest path for travelling from New York to Los Angeles based on the information as shown in Figure 7.2, where the number inside a circle indicates a state while an arrow pointing from one city to another represents an action. The number associated with each arrow specifies the distance (in miles) between the two cities. This number can be considered as a (negative) reward for taking the action. We will refer to this negative reward as *cost* to be minimized in the subsequent analysis.

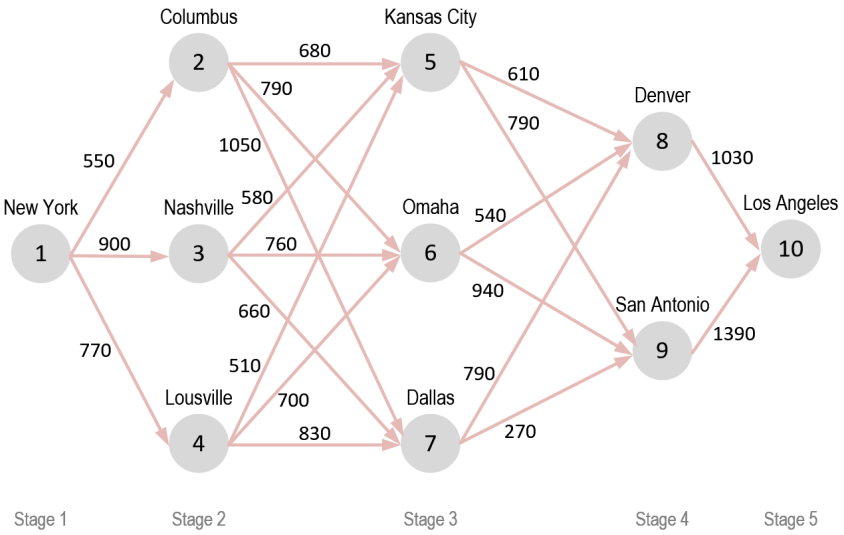


Figure 7.2: Paths from New York to Los Angeles.

The locations of the cities under consideration naturally form a structure (consisting of five stages) for determining the shortest path. Stage 1 and Stag 5 have one city each (i.e., the starting point and the destination), while each of the intermediate stages has multiple cities. We can solve this shortest-path problem by starting from the destination (i.e., Los Angeles) and work backwards one stage at a time, while

identifying the shortest distance between two adjacent stages, to reach the starting city (i.e., New York).

So we start with Stage 5, Los Angeles, which can be reached directly from the two cities in Stage 4 with a distance of 1030 (from Denver) and 1390 (from San Antonio). Since there is only one path from each of these two cities leading to Los Angeles, each path is optimal for each city by default. These optimal paths are represented in Figure 7.3 by the bold-number labels on the arrows from these cities to Los Angeles (i.e., **1030** for Denver and **1390** for San Antonio), while the cost (which is the same as the distance) of going from each of these two cities to Los Angeles is indicated by the grey number just beneath the circled number representing the city.

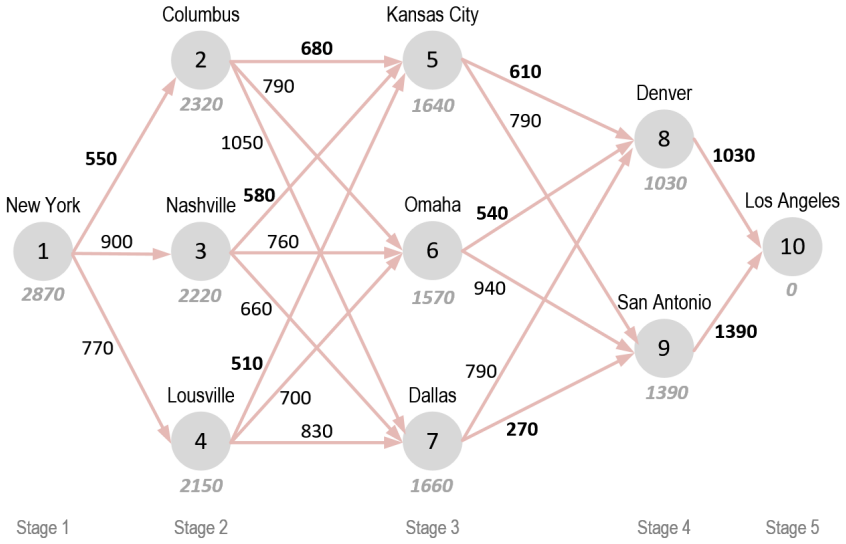


Figure 7.3: Paths from New York to Los Angeles.

We apply the same reasoning to determine the optimal path and cost for travelling from each city in Stage 3 to Los Angeles via one of the cities in Stage 4. Stage 3 has three cities, namely, Kansas City, Omaha, and Dallas. Consider Kansas City. It can reach Denver (at a cost of 610) and San Antonio (cost 790), with the total cost of travelling from Kansas City to Los Angeles via Denver being $610 + 1030 = 1640$ and that via San Antonio being $790 + 1390 = 2180$. Between these two, the optimal choice (i.e., action) is to go via Denver. This optimal action is represented by the bold-number (i.e., **610**) as the label of the arrow pointing from ⑤ to ⑧, and the total cost associated with taking this optimal action in order to eventually reach Los Angeles is represented by the grey number 1640 (i.e., the smaller of 1640 and 2180 as calculated earlier) beneath ⑤. We have thus determined that, if we were to start our journey from Kansas City with the destination being Los Angeles, the optimal action to take at Kansas City is to go to Denver.

Repeating the same procedure above on the other two cities in Stage 3 yields the

optimal total cost of 1570 for Omaha (with the optimal action to be take at Ohama being going to Denver) and 1660 for Dallas (with the optimal action to be taken at Dallas being going to San Antonio). Continuing in the same fashion all the way back to Stage 1, we obtain the optimal total cost and the optimal action (identified by the bold-number labels on the relevant arrows) for each of the remaining cities as shown in Figure 7.3. Note that at this point we already know that the minimum total cost for travelling from New York to Los Angeles is 2870 (i.e., the number beneath ①), but we have not yet figured out the path that will result in this minimum cost.

We are now ready to determine the optimal (i.e., shortest) path for travelling from New York to Los Angeles, based on the information we have obtained so far as shown Figure 7.3. Starting from New York, we see that the optimal action is to go to Columbus (as indicated by the bold-number label **550**). From Columbus, the optimal action is to go to Kansas City (**680**). From Kansas City, the optimal action is to go to Denver (**610**), from where we finally reach Los Angeles (**1130**). Hence, the total (minimum) cost for travelling from New York to Los Angeles is: $550 + 680 + 610 + 1030 = 2870$. This optimal path is illustrated by the dark circles in Figure 7.4, and the optimal policy π^* for travelling from New York to Los Angeles can be expressed as:

$$\pi^*(1) = \text{'go to Columbus'} \qquad \pi^*(2) = \text{'go to Kansas City'}$$
$$\pi^*(5) = \text{'go to Denver'} \qquad \pi^*(8) = \text{'go to Los Angeles'}$$

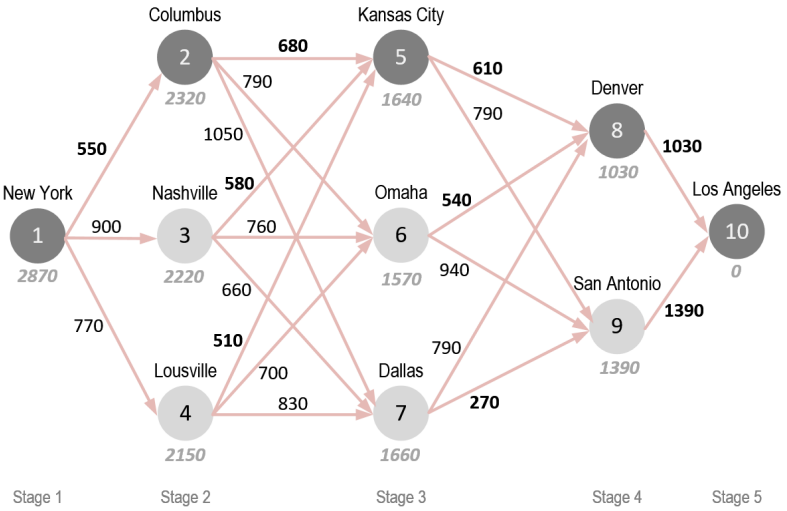


Figure 7.4: Paths from New York to Los Angeles.

We can now give an intuitive interpretation of the Principle of Optimality in the context of this optimal policy. Take any state under the optimal policy, say ② (i.e., Columbus), and treat it as the "initial" state from which subsequent actions are chosen.

From Columbus there are six paths leading to Los Angeles:

②-⑤-⑧-⑩	②-⑤-⑨-⑩
②-⑥-⑧-⑩	②-⑥-⑨-⑩
②-⑦-⑧-⑩	②-⑦-⑨-⑩

These paths span Stages 2 to 5. Referring to Figure 7.4, we see that, as stipulated by the Principle of Optimality, the optimal policy for the sub-problem involving only Stages 2 to 5 is part of the optimal policy for the overall problem (i.e., involving all stages). Similarly, the optimal policy for the sub-problem involving only Stages 3 to 5 is also part of the optimal policy for the overall problem, and so on. In other words, if among all the cities in Stage 4 Denver is nearest to Los Angeles, then any path starting from the "optimal city" chosen in Stage 3 (in this case it happens to be Kansas City) must go through Denver. Similarly, if among all the cities in Stage 3 Kansas City is nearest to Los Angeles, then any path starting from the "optimal city" chosen in Stage 2 (in this case it happens to be Columbus) must go through Kansas City. □

7.1.4 Optimal policy

By definition, the optimal value of the Q-function, i.e., $Q^*(s,a)$, satisfies the Bellman Equation (6.22), i.e.,

$$Q^*(s,a) = \sum_{s'} P^a_{ss'} \left(\rho(s,a,s') + \gamma \max_{a'} Q^* \left(s', a' \right) \right) \tag{7.47}$$

Equation (7.47) states that the optimal value of the Q-function for the current state s and an action a is the same as the sum of

1. the reward in the current state s for the chosen action a, and
2. the (discounted) value of the Q-function for the "best" action available in the successor state s'.

Hence, we can determine an optimal policy π^* recursively by picking (at state s') the action that yields the maximum value for Q^*. This is conceptualized in Bellman's Principle of Optimality discussed above.

Recall from Equation (6.30) that the policy whose Q-function values are optimal (i.e., greater or equal to that of all other policies for all possible state-action pairs) is an optimal policy, i.e.,

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a), \quad \text{for all } s \in S \text{ and } a \in A \tag{7.48}$$

Any policy that selects at each state s an action a that yields the largest value for the Q-function is optimal. Such a policy is called a *greedy policy*. Thus, once the optimal values of the Q-function are determined, an optimal policy can be directly identified from the values of $Q^*(s,a)$ by picking the action associated with the maximum value of the Q-function at a given state, that is,

$$\pi^*(s) \in \arg\max_a Q^*(s,a) \tag{7.49}$$

where $\arg\max_a$ is the argument of the maximum; it gives all values of a for which the given function $Q^*(s,a)$ attains its maximum value. When multiple choices of such actions are available, any one of these actions can be chosen randomly.

Example 7.4 Determine an optimal policy π^* for the robot in (i) Example 7.1, and (ii) Example 7.2.

Solution: (i) We can construct an optimal policy by picking the action that yields the maximum Q^* at each state, i.e., a greedy policy. For Example 7.1, the optimal values are $Q^* = Q_5$ as shown in Table 7.5.

Table 7.5: Optimal values for Q-function in Example 7.1.

State	0	1	2	3	4	5
Q_5	0.000|0.000	1.000|0.625	0.500|1.250	0.625|2.500	1.250|5.000	0.000|0.000
π^*		-1	1	1	1	

From the table, we see that at state 1, the optimal values for the Q-function for the two actions are 1.000 (for $a=-1$) and 0.625 (for $a=1$). So we have $\pi^*(1)=-1$. Applying the same reasoning on other states yields the following optimal policy (which is illustrated in Figure 7.5): $\pi^*(1)=-1$, $\pi^*(2)=1$, $\pi^*(3)=1$, and $\pi^*(4)=1$.

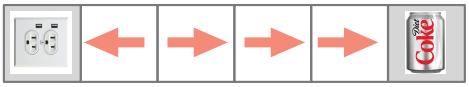


Figure 7.5: Optimal policy.

(ii) The optimal values of the Q-function for Example 7.2 are as shown in Table 7.6. From the table, an optimal policy is: $\pi^*(1)=-1$, $\pi^*(2)=1$, $\pi^*(3)=1$, and $\pi^*(4)=1$, which happens to be identical to that for Example 7.1.

Note that for both cases, no action is selected for either state 0 or 5, since the reinforcement learning task is considered terminated at those states.

□

Table 7.6: Optimal values for Q-function in Example 7.2.

State	0	1	2	3	4	5
Q_{22}	0.000|0.000	0.888|0.458	0.467|0.852	0.594|1.915	1.344|4.376	0.000|0.000
π^*		-1	1	1	1	

Chapter 8

Q-Learning

Chapter 7 shows that we can use Dynamic Programming to find an optimal policy by first determining the optimal values for the Q-function and then obtaining an optimal policy directly from these optimal values. The implicit assumption there is that the state transition probabilities are known *a priori*. In practice, such detailed information about the system is rarely (if at all) readily available. A difficulty arises when we try to execute a task *without having any knowledge about state transitions.*

The reinforcement learning technique called Q-learning enables us to overcome this difficulty. In Q-learning, the optimal values for the Q-function are determined by the iterative process of model-free value iteration.

8.1 Model-free value iteration

The Q-learning technique performs value iteration without a state-transition model of the system being available. It uses solely the reward received from the *observed* state transitions. The iteration uses the update rule:

$$\begin{aligned} & Q_{k+1}(s_k, a_k) \\ = \; & Q_k(s_k, a_k) + \alpha_k \left(\underbrace{r_{k+1} + \gamma \max_{a'} Q_k\left(s_{k+1}, a'\right)}_{\text{Estimate of } Q^*(s_k, a_k)} - Q_k(s_k, a_k) \right) \end{aligned} \tag{8.1}$$

where $\alpha_k \in (0, 1]$ is the learning rate. The expression inside the brackets in Equation (8.1) represents the difference between the updated estimate of $Q^*(s_k, a_k)$ and the current estimate $Q_k(s_k, a_k)$. This difference is called the *temporal difference.*

It has been proved that model-free value iteration converges to Q^* as the number of iterations k approaches infinity, provided that the following two conditions are satisfied

[12][10][6]:

1. $\sum_{k=0}^{\infty} \alpha_k^2 < \infty \quad \text{and} \quad \sum_{k=0}^{\infty} \alpha_k \to \infty.$
2. All the state-action pairs (s, a) are (asymptotically) visited infinitely often.

The first condition can be met by setting $\alpha_k = 1/k$, with the requirement that the initial value of α (i.e., α_0) is set to some finite value. Note that in the implementation of Q-learning, the number of iterations k is also the number of state transitions.

The second condition can be satisfied if we make sure that the agent is prescribed with a non-zero probability of selecting any available action in every state that it visits. This is called *exploration*.

8.2 Exploration and exploitation

In reinforcement learning, exploration is studied in conjunction with the concept of exploitation.

In Q-learning, the update rule, i.e., Equation (8.1), calculates the new value for the Q-function based on a greedy policy of selection the next action a' at state s_{k+1}, as indicated by the term $\max_{a'} Q_k(s_{k+1}, a')$. This way of utilizing what is already "learned" about the best action to take at a certain state is called *exploitation*.

Exploration is an essential characteristic of learning. In general, if we do not try new things (i.e., explore), then we will never develop new capabilities and skills. However, if we always explore and do not utilize what we have learned (i.e., exploitation), then we will most likely not be able to attain any significant achievement. The necessity for balancing this trade-off between exploitation and exploration is succinctly captured in Q-learning. One simple way to balance this trade-off is the so-called *ϵ-greedy exploration*.

In ϵ-greedy exploration, the agent at state s_k selects the action according to the following rule:

$$a_k = \begin{cases} a \in \arg\max_{\hat{a}} Q_k(s_k, \hat{a}) & \text{with probability } 1-\epsilon_k \\ \text{action uniformly randomly selected} & \\ \text{from all other actions available at } s_k & \text{with probability } \epsilon_k \end{cases} \tag{8.2}$$

In other words, the agent chooses exploitation with the probability $(1-\epsilon_k)$ and chooses exploration with the probability ϵ_k. In implementation, the probability ϵ is kept small so that the agent will focus on executing (in the best way it could) the task at hand

most of the time, and occasionally will wander off to explore by taking an action that it is not necessarily the "best" based on what it knows at moment[1]. This is illustrated in Figure 8.1. If there are multiple actions associated with the same $Q^*(s_k, \hat{a})$ value, then any one of those can be randomly chosen.

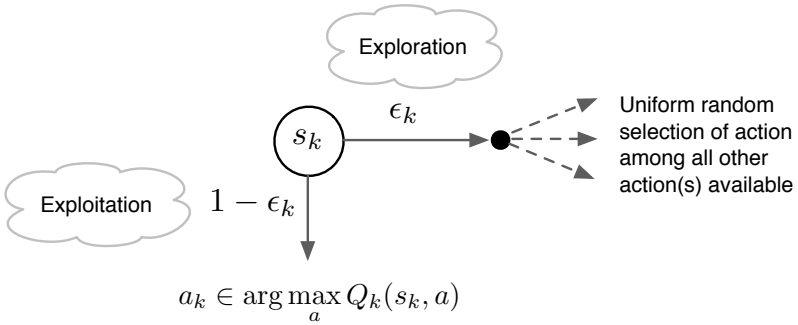


Figure 8.1: Exploration and exploitation.

The algorithm for implementing Q-learning with ϵ-greedy exploration is shown in Table 8.1.

Table 8.1: Algorithm for Q-learning with ϵ-greedy exploration.

Input: Discount factor γ; exploration probability ϵ_k; learning rate α_k

- Initialize Q-function, e.g., $Q_0 \leftarrow 0$
- Determine the initial state s_0
- For time step k, select action a_k according to:

$$a_k = \begin{cases} a \in \arg\max_{\hat{a}} Q_k(s_k, \hat{a}) & \text{with probability } 1-\epsilon_k \\ \text{an action uniformly randomly selected} \\ \text{from all actions available at state } s_k & \text{with probability } \epsilon_k \end{cases}$$

- Apply action a_k, receive reward r_{k+1}, then observe next state s_{k+1}
- Update Q-function with:

$$Q_{k+1}(s_k, a_k) = Q_k(s_k, a_k) + \alpha_k \left(r_{k+1} + \gamma \max_{a'} Q_k\left(s_{k+1}, a'\right) - Q_k(s_k, a_k) \right)$$

- Set $k = k+1$, and repeat the for-loop for the next time step

Since exploration can be viewed as an activity that may slow down the execution of a task, it is best not to allow it to occur indefinitely. That is, as time goes on, we would like to reduce the extent of exploration and have the agent focus on executing the task at hand (assuming that it has explored and learned enough to do the job

[1]Another strategy for exploration is to use the Boltzmann distribution to set the probability of the agent taking action a at state s_k as: $P(a|s_k) = \exp(Q_k(s_k,a)/\tau_k) / \sum_{\hat{a}} \exp(Q_k(s_k,\hat{a})/\tau_k)$, where the "temperature" $\tau_k \geq 0$ controls the randomness of the exploration. The two extreme cases of $\tau_k \to 0$ and $\tau_k \to \infty$ correspond to greedy exploration and uniformly random action selection, respectively.

satisfactorily). One way to achieve this is to have $\epsilon_k \to 0$ as $k \to \infty$. This can be done by simply setting $\epsilon_k = 1/k$.

Remark 8.1 Unlike the cases of dynamic programming and value iteration, Q-learning does not require any prior knowledge about the (deterministic or non-deterministic) transition function. All it requires is the implicit assumption that the agent is able to ascertain the new state it is in, after making a transition by taking a certain action. For example, consider the situation as shown in Figure 8.2.

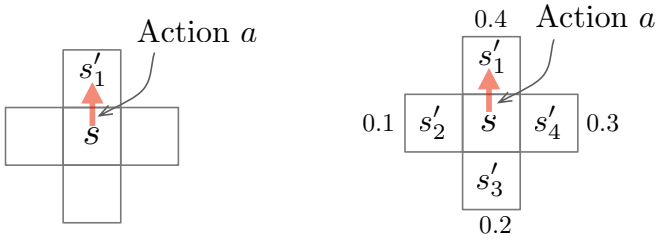


Figure 8.2: (a) Deterministic transition. (b) Non-deterministic transition.

Suppose that at state s the agent (after executing the ϵ-greedy exploration rule) decides to take the action a. If the Q-learning were to follow a deterministic transition function, then the agent is required to enter s'_1 as its new state, as shown in Figure 8.2(a). If the Q-learning were to utilize a non-deterministic transition function, then it would need the transition probabilities as shown in Figure 8.2(b). The fact is that, for Q-learning, when the agent takes action a it does not know in which state it may eventually end up; all it knows is that the new state may be any one among s, s'_1, s'_2, s'_3, and s'_4. It is after the agent has executed action a and then observes its new state to be, say, s'_2 that we can say that, for this particular transition, $f(s,a) = s'_2$. Once again, the implicit assumption is that the agent has some way of ascertain its new state after executing a chosen action. In summary, Q-learning permits actions that may fail to bring the agent to the "expected" new state. In such cases, the updated Q values will automatically account for such eventuality and so no distributions or expected values are needed. □

8.3 Implementation

Theoretically, the Q-learning process may continue until convergence of the values of the Q-function. In practice, a Q-learning process may consist of many trials, each terminating when a goal state is reached. When a trial is terminated, the values of the Q-function and the policy thus obtained can be saved as output of the trial and be re-used for exploitation in the next trial.

Example 8.1 Suppose that a robot is to learn an optimal policy on a 3×3 grid for reaching the goal state (the gray square) as shown in Figure 8.3. At a state, the robot can take one of four actions to move up ($a = 1$), right ($a = 2$), down ($a = 3$), or left ($a = 4$) into the corresponding adjacent state. If no such adjacent state exists, then the robot's state remains unchanged after the execution of the action. The reward function ρ is given as follows:

$$\rho(s,a,s') = \begin{cases} 1 & \text{if } s \neq 8 \text{ and } s' = 8 \\ 0 & \text{otherwise} \end{cases} \tag{8.3}$$

Let $\gamma = 0.9$ and $\alpha_k = 1/k$ with an initial value of $\alpha_0 = 1$.

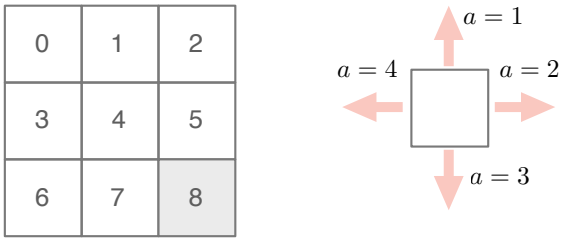


Figure 8.3: Reinforcement learning on a 3×3 grid.

The learning process will consist of a series of trials. In a trial the robot will start at an initial state and make transitions according to the algorithm given in Table 8.1 until it reaches the goal state ($s = 8$), upon which the trial ends.

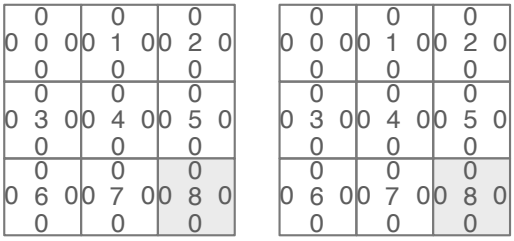
In this example, we will focus on the calculation of the values for the Q-function. So we will *not* actually simulate the probabilistic selection of an action at each state during a trial. Instead we will simply assume a sequence of actions for a trial (as if they were observed to have occurred) in order to carry out the calculation of the values for the Q-function .

For record keeping, we define a function $N(s,a)$ which counts the accumulative number of times an action a has been taken at state s. Note that $N(s,a)$ is accumulative and so it is not reset to 0 after a trial. For graphical illustration of the learning process, we will use two types of diagrams: one showing the values of $N(s,a)$ at the end of a trial; the other showing the values of the Q-function at the end of a trial. Figure 8.4 shows the diagrams for $N(s,a)$ and Q-function before learning starts.

Trial 1: The (assumed) action sequence for Trial 1 is shown in Figure 8.5. Note that $\alpha_0 = 1$.

From the update rule given in Equation (8.1), we have

$$Q_1(0,2) = Q_0(0,2) + \alpha_0\Big(\rho(0,2,1) + \gamma \max_{a'} Q_0\left(1,a'\right) - Q_0(0,2)\Big)$$



(a) $N(s,a)$ (b) Q-function

Figure 8.4: Initial values of $N(s,a)$ and Q-function. Each value at an edge of a state corresponds to the action in the outward direction perpendicular to that edge.

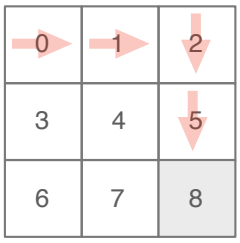


Figure 8.5: Action sequences in Trial 1.

$$\begin{aligned} &= 0+1\cdot(0+0.9\cdot 0-0) \quad \big(\text{Note: } \alpha_0=1\big) \\ &= 0 \end{aligned} \tag{8.4}$$

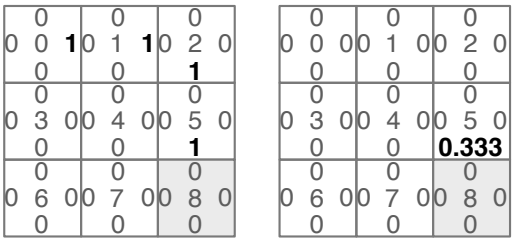
$$\begin{aligned} Q_2(1,2) &= Q_1(1,2)+\alpha_1\Big(\rho(1,2,2)+\gamma\max_{a'}Q_1\left(2,a'\right)-Q_1(1,2)\Big) \\ &= 0+\frac{1}{1}\Big(0+0.9\cdot 0-0\Big) \quad \big(\text{Note: } k=1;\ \alpha_k=1/k\big) \\ &= 0 \end{aligned} \tag{8.5}$$

$$\begin{aligned} Q_3(2,3) &= Q_2(2,3)+\alpha_2\Big(\rho(2,3,5)+\gamma\max_{a'}Q_2\left(5,a'\right)-Q_2(2,3)\Big) \\ &= 0+\frac{1}{2}\Big(0+0.9\cdot 0-0\Big) \\ &= 0 \end{aligned} \tag{8.6}$$

$$\begin{aligned} Q_4(5,3) &= Q_3(5,3)+\alpha_3\Big(\rho(5,3,8)+\gamma\max_{a'}Q_3\left(8,a'\right)-Q_3(5,3)\Big) \\ &= 0+\frac{1}{3}\Big(1+0.9\cdot 0-0\Big) \\ &= 0.333 \end{aligned} \tag{8.7}$$

The trial ends when the robot enters the goal state $s=8$. The values of $N(s,a)$

and the Q-function at the end of the trial are shown in Figure 8.6.



(a) $N(s,a)$ (b) Q-function

Figure 8.6: Values of $N(s,a)$ and Q-function at the end of Trial 1.

Trial 2: The (assumed) action sequence for Trial 2 is shown in Figure 8.7. The initial learning rate is reset to $\alpha_0 = 1$ and the initial values of the Q-function are those obtained from Trial 1 as shown in Figure 8.6(b).

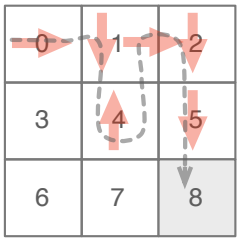


Figure 8.7: Action sequences in Trial 2.

$$\begin{aligned} Q_1(0,2) &= Q_0(0,2)+\alpha_0\Big(\rho(0,2,1)+\gamma\max_{a'}Q_0\left(1,a'\right)-Q_0(0,2)\Big) \\ &= 0+1\cdot(0+0.9\cdot 0-0) \\ &= 0 \end{aligned} \tag{8.8}$$

$$\begin{aligned} Q_2(1,3) &= Q_1(1,3)+\alpha_1\Big(\rho(1,3,4)+\gamma\max_{a'}Q_1\left(4,a'\right)-Q_1(1,3)\Big) \\ &= 0+\frac{1}{1}\Big(0+0.9\cdot 0-0\Big) \\ &= 0 \end{aligned} \tag{8.9}$$

$$\begin{aligned} Q_3(4,1) &= Q_2(4,1)+\alpha_2\Big(\rho(4,1,1)+\gamma\max_{a'}Q_2\left(1,a'\right)-Q_2(4,1)\Big) \\ &= 0+\frac{1}{2}\Big(0+0.9\cdot 0-0\Big) \\ &= 0 \end{aligned} \tag{8.10}$$

$$\begin{aligned} Q_4(1,2) &= Q_3(1,2)+\alpha_3\Big(\rho(1,2,2)+\gamma\max_{a'}Q_3\left(2,a'\right)-Q_3(1,2)\Big) \\ &= 0+\frac{1}{3}\Big(0+0.9\cdot 0-0\Big) \\ &= 0 \end{aligned} \tag{8.11}$$

$$\begin{aligned} Q_5(2,3) &= Q_4(2,3)+\alpha_4\Big(\rho(2,3,5)+\gamma\max_{a'}Q_4\left(5,a'\right)-Q_4(2,3)\Big) \\ &= 0+\frac{1}{4}\Big(0+0.9\max\left[0,0,0.333,0\right]-0\Big) \\ &= \frac{1}{4}\cdot 0.9\cdot 0.333 \\ &= 0.075 \end{aligned} \tag{8.12}$$

$$\begin{aligned} Q_6(5,3) &= Q_5(5,3)+\alpha_5\Big(\rho(5,3,8)+\gamma\max_{a'}Q_5\left(8,a'\right)-Q_5(5,3)\Big) \\ &= 0.333+\frac{1}{5}\Big(1+0.9\cdot 0-0.333\Big) \\ &= 0.466 \end{aligned} \tag{8.13}$$

The values of $N(s,a)$ and the Q-function at the end of the trial are shown in Figure 8.8.

0 0 0 **2** 0	0 0 1 **2** **1**	0 0 2 0 **2**
0 0 3 0 0	**1** 0 4 0 0	0 0 5 0 **2**
0 0 6 0 0	0 0 7 0 0	0 0 8 0 0

(a) $N(s,a)$

0 0 0 0 0	0 0 1 0 0	0 0 2 0 **0.075**
0 0 3 0 0	0 0 4 0 0	0 0 5 0 **0.466**
0 0 6 0 0	0 0 7 0 0	0 0 8 0 0

(b) Q-function

Figure 8.8: Values of $N(s,a)$ and Q-function at the end of Trial 2.

The above process repeats until the values of the Q-function converge to the optimal values, upon which an optimal policy can be obtained.

□

Bibliography

[1] Bellman, R.E., *Dynamic Programming*. Princeton University Press, 1957.

[2] Bellman, R.E., *Eye of the Hurricane: An Autobiography*. World Scientific Pub. Co. Inc., 1984.

[3] Cover, T. M., Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14(3), 326-334, 1965.

[4] Cristianini, N., and Shawe-Taylor, J., *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[5] Fletcher, R., *Practical Methods of Optimization*. John Wiley and Sons, 1987.

[6] Jaakkola, T., Jordan, M. I., and Singh, S. P., On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185-1201, 1994.

[7] Sutton, R. S. and Barto, A. G., *Reinforcement Learning: An Introduction*. The MIT Press, 2nd ed., 2018.

[8] Vapnik, V. N., *Statistical Learning Theory*. Wiley-Interscience, 1998.

[9] Tesauro, G. J., Practical issues in temporal difference learning. *Machine Learning*, 8:257-277, 1992.

[10] Tsitsiklis, J. N., Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16(1):185-202, 1994.

[11] Van Kampen, N. G., Remarks on non-Markov processes. *Brazilian Journal of Physics*, vol. 28, no. 2, 90-96, 1998.

[12] Watkins, C. J. C. H. and Dayan, P., Q-learning. *Machine Learning*, 8:279-292, 1992.