附录 A

高斯分布的性质

我们先总结常见的高斯分布的性质，它在本书的很多地方都会用到。

A.1 高斯分布

如果一个随机变量 x 服从高斯分布 $N(\mu, \sigma^2)$，那么它的概率密度函数为

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right). \tag{A.1}$$

其高维形式为

$$p(x) = \frac{1}{\sqrt{(2\pi)^N \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right). \tag{A.2}$$

A.2 高斯分布的运算

A.2.1 线性运算

设两个独立的高斯分布：

$$\boldsymbol{x} \sim N(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx}), \quad \boldsymbol{y} \sim N(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_{yy}),$$

那么，它们的和仍是高斯分布：

$$\boldsymbol{x}+\boldsymbol{y}\sim N(\boldsymbol{\mu}_x+\boldsymbol{\mu}_y,\boldsymbol{\Sigma}_{xx}+\boldsymbol{\Sigma}_{yy}). \tag{A.3}$$

如果以常数 a 乘以 $\boldsymbol{x}$，那么 $a\boldsymbol{x}$ 满足：

$$a\boldsymbol{x}\sim N(a\boldsymbol{\mu}_x,a^2\boldsymbol{\Sigma}_{xx}). \tag{A.4}$$

如果取 $\boldsymbol{y}=\boldsymbol{A}\boldsymbol{x}$，那么 $\boldsymbol{y}$ 满足：

$$\boldsymbol{y}\sim N(\boldsymbol{A}\boldsymbol{\mu}_x,\boldsymbol{A}\boldsymbol{\Sigma}_{xx}\boldsymbol{A}^{\mathrm{T}}). \tag{A.5}$$

A.2.2 乘积

设两个高斯分布的乘积满足 $p(\boldsymbol{x}\boldsymbol{y})=N(\boldsymbol{\mu},\boldsymbol{\Sigma})$，那么：

$$\begin{aligned}\boldsymbol{\Sigma}^{-1}&=\boldsymbol{\Sigma}_{xx}^{-1}+\boldsymbol{\Sigma}_{yy}^{-1}\\ \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}&=\boldsymbol{\Sigma}_{xx}^{-1}\boldsymbol{\mu}_x+\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\mu}_y.\end{aligned} \tag{A.6}$$

该公式可以推广到任意多个高斯分布之乘积。

A.2.3 复合运算

同样，考虑 $\boldsymbol{x}$ 和 $\boldsymbol{y}$，若其不独立，则其复合分布为

$$p(\boldsymbol{x},\boldsymbol{y})=N\left(\begin{bmatrix}\boldsymbol{\mu}_x\\ \boldsymbol{\mu}_y\end{bmatrix},\begin{bmatrix}\boldsymbol{\Sigma}_{xx}&\boldsymbol{\Sigma}_{xy}\\ \boldsymbol{\Sigma}_{yx}&\boldsymbol{\Sigma}_{yy}\end{bmatrix}\right). \tag{A.7}$$

由条件分布展开式 $p(\boldsymbol{x},\boldsymbol{y})=p(\boldsymbol{x}|\boldsymbol{y})p(\boldsymbol{y})$ 可以推出，条件概率 $p(\boldsymbol{x}|\boldsymbol{y})$ 满足：

$$p(\boldsymbol{x}|\boldsymbol{y})=N\left(\boldsymbol{\mu}_x+\boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}(\boldsymbol{y}-\boldsymbol{\mu}_y),\boldsymbol{\Sigma}_{xx}-\boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx}\right). \tag{A.8}$$

A.3 复合的例子

下面举一个和卡尔曼滤波器相关的例子。考虑随机变量 $\boldsymbol{x}\sim N(\boldsymbol{\mu}_x,\boldsymbol{\Sigma}_{xx})$，另一变量 $\boldsymbol{y}$ 满足：

$$\boldsymbol{y}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}+\boldsymbol{w} \tag{A.9}$$

其中 $\boldsymbol{A}, \boldsymbol{b}$ 为线性变量的系数矩阵和偏移量，$\boldsymbol{w}$ 为噪声项，为零均值的高斯分布：$\boldsymbol{w} \sim N(\boldsymbol{0}, \boldsymbol{R})$。我们来看 $\boldsymbol{y}$ 的分布。根据前面的介绍，可以推出：

$$p(\boldsymbol{y}) = N\left(\boldsymbol{A}\boldsymbol{\mu}_x + \boldsymbol{b}, \boldsymbol{R} + \boldsymbol{A}\boldsymbol{\Sigma}_{xx}\boldsymbol{A}^{\mathrm{T}}\right). \tag{A.10}$$

这为卡尔曼滤波器的预测部分提供了理论基础。

附录 B

矩阵求导

本节我们简单回顾有关矩阵求导方面的知识。

首先，标量函数的求导是显然的。假设一个函数 $f(x)$ 对 x 求导，那么将得到 $\frac{\mathrm{d}f}{\mathrm{d}x}$ 这样一个导数，显然它仍然是一个标量。下面我们分别讨论当 x 为向量或者 f 为向量函数的情况。

B.1 标量函数对向量求导

先考虑 $\boldsymbol{x}$ 为向量的情况。假设 $\boldsymbol{x} \in \mathbb{R}^m$，为列向量，那么：

$$\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}} = \left[\frac{\mathrm{d}f}{\mathrm{d}x_1}, \cdots, \frac{\mathrm{d}f}{\mathrm{d}x_m}\right]^{\mathrm{T}} \in \mathbb{R}^m. \tag{B.1}$$

这将得到一个 $m \times 1$ 的向量。有时，我们也写成对 $\boldsymbol{x}^{\mathrm{T}}$ 的求导：

$$\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \left[\frac{\mathrm{d}f}{\mathrm{d}x_1}, \cdots, \frac{\mathrm{d}f}{\mathrm{d}x_m}\right]. \tag{B.2}$$

这得到一个行向量。我们一般称 $\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}}$ 为梯度或者 Jacobian，但要注意，在不同的领域中，人们使用的习惯并不完全相同。

B.2　向量函数对向量求导

一个向量函数也可以对向量求导。考虑 $\boldsymbol{F}(\boldsymbol{x})$ 为一个向量函数：

$$\boldsymbol{F}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), \cdots, f_n(\boldsymbol{x})]^{\mathrm{T}},$$

其中每一个 f_k 都是一个自变量为向量，取值为标量的函数。考虑这样的函数对 $\boldsymbol{x}$ 求导时，通常的做法是写为

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}^{\mathrm{T}}} = \begin{bmatrix} \frac{\partial f_1}{\partial \boldsymbol{x}^{\mathrm{T}}} \\ \vdots \\ \frac{\partial f_n}{\partial \boldsymbol{x}^{\mathrm{T}}} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix} \in \mathbb{R}^{n \times m}, \tag{B.3}$$

也就是写成列函数对行向量求导的形式，这将得到一个 $n \times m$ 的雅可比矩阵。这种写法是规范的，典型的例子就是：

$$\frac{\partial \boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}^{\mathrm{T}}} = \boldsymbol{A}. \tag{B.4}$$

反之，一个行向量函数也可以对列向量求导，结果为之前的转置：

$$\frac{\partial \boldsymbol{F}^{\mathrm{T}}}{\partial \boldsymbol{x}} = \left(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}^{\mathrm{T}}}\right)^{\mathrm{T}}. \tag{B.5}$$

在本书中，我们习惯使用前者，即列向量对行向量求导。但是这种写法要求每次都对被求导变量加上转置符号，比较烦琐。在不引起歧义的情况下，我们将忽略分母上的转置符号，简单地记作：

$$\frac{\partial \boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{A}. \tag{B.6}$$

我们也可以定义矩阵函数对矩阵的求导，但这在本书中并没有用到，在此略过。

附录 C

ROS 入门

ROS 是机器人研究领域一个广为探讨的主题。为了避免使本书阅读门槛太高，我们没有在正文和例程中提到它。然而近年来，ROS 正逐步在各大高校的学生中间得到推广，渐渐为人们所熟知和接受，所以这里也简单介绍 ROS，希望能对读者有所帮助。

C.1 ROS 是什么

ROS（Robot Operating System）是 Willow Garage 公司于 2007 年发布的一个开源机器人操作系统，它为软件开发人员开发机器人应用程序提供了许多优秀的工具和库。同时，还有优秀的开发者不断地为它贡献代码。本质上，ROS 并不是一个真正意义上的操作系统，而更像是基于操作系统之上的一个软件包。它提供了众多在实际机器人中可能遇到的算法：导航、通信、路径规划，等等。

ROS 的版本代号是按照字母顺序编排的，并随着 Ubuntu 系统发布的更新而更新。通常，一个 ROS 版本会支持两到三个 Ubuntu 系统版本。ROS 从 Box Turtle 开始，截至本书写作时，已经更新到了 Kinetic Kame（ROS 的各个版本如图 C-1 所示）。同时，ROS 也已经彻底重构，推出了实时性更强的 2.0 版本。

ROS 支持很多操作系统，支持最完善的是 Ubuntu 及其衍生版本（Kubuntu、Linux Mint、Ubuntu GNOME 等），也支持其他 Linux 发布版本、Windows 等，不过没有那么完善。因此，推荐读者使用 Ubuntu 操作系统进行开发和研究。

ROS 支持目前被广泛使用的面向对象的编程语言 C++，以及脚本语言 Python。你可以选择自己喜欢的语言进行开发。

图 C-1　ROS 的各个版本

C.2　ROS 的特点

ROS 的设计初衷，就是使机器人开发能够像计算机开发一样，屏蔽底层硬件及其接口的不一致性，最终使得软件可以复用。

而软件复用也正是软件工程优美性最集中的体现之一，ROS 能够以统一消息格式使大家只需要关注算法层面的设计，而底层硬件的根本目的是接收各种各样的消息，如图像、数据等。各个硬件厂商将接收到的数据统一到 ROS 所规定的统一消息格式下，即可让用户方便地使用各种开源的机器人相关算法。

在第 14 讲中提到的常见的开源 SLAM 方案中，ORB-SLAM、ORB-SLAM2、LSD-SLAM、SVO、DVO、RTAB-MAP、RGBD-SLAM-V2、Hector SLAM、Gmapping、ROVIO 等均有 ROS 版本的开源代码，你可以很方便地在 ROS 中运行、调试和修改它们。

在调试 SLAM 程序时，数据的来源通常有 3 种：传感器、数据集，以及 bag 文件。若手头没有相应的传感器，通常就需要利用虚拟的数据运行 SLAM 程序。其中，最方便的方式当属利用 ROS 下的 bag 文件发布 topic，然后 SLAM 程序就可以监视 topic 发出的数据，就像使用真实的传感器采集数据一样。后面我们会简单介绍如何利用 bag 文件模拟真实的传感器数据。

C.3　如何快速上手 ROS

ROS 有完善的维基系统。首先，按照官网的介绍在机器上安装对应版本的 ROS：`http://wiki.ros.org/ROS/Installation`；然后，阅读 ROS 自带的教学程序即可。你会学习到 ROS 的基本概念、主题的发布和订阅，以及如何用 Python 和 C++ 控制它们。如果你觉得麻烦，则可以使用针对 ROS 定制的 Ubuntu 系统：`http://www.aicrobo.com/ubuntu_for_ros.html`。

除了基本知识，你还可以学着使用一些 ROS 的常用工具，例如：

1. rqt。rqt 是 ROS 下的一个软件框架，它以插件的方式提供了各种各样方便好用的 GUI（用户图形界面）。rqt 的功能非常强大，可以实时地查看 ROS 中流动的消息。
2. rosbag。rosbag 是 ROS 提供的一个非常好用的录制及播放 topic 数据的工具。当你想实际运行 SLAM 程序，但囿于手头没有实际的传感器时，可以考虑使用公开提供的 bag 文件进行图像或者数据的模拟，这种方式与使用一个真实的传感器在感觉上并无不同。rosbag 的使用方式请参考 ROS 的维基页面。此外，许多公开数据集也会提供 bag 格式的数据文件。
3. rviz。rviz 是 ROS 提供的可视化模块，你可以通过它实时地查看 ROS 中的图像、点云、地图、规划的路径，等等，从而更方便地调试程序。

我们相信，机器人的硬件层面和软件层面一定都会向着统一架构的方向前行，而 ROS 正是软件架构层面标准化的一个重要里程碑。其中，ROS 1.x 在之前被大量用于实验室的研究，或者公司产品 demo 的研发阶段，而 ROS 2 则解决了 ROS 实时性的问题，未来很有可能被直接用于实际产品的研发，为推进工业级机器人和服务机器人的应用做出重要的贡献。

本附录概述性地介绍了有关 ROS 的历史、优点，以及如何利用 ROS 中的一些可视化工具辅助 SLAM 程序开发等。我们希望读者系统地学习 ROS，并使用 ROS 开发自己的 SLAM 程序。