About This Document

This document contains selected solutions to exercises from the textbook *Machine Learning Foundations, Volume 1: Supervised Learning*. The solutions are intended to help readers deepen their understanding of key concepts and techniques, and are written with clarity and pedagogical insight in mind.

This collection is not exhaustive and focuses on representative problems that illustrate important concepts or commonly encountered challenges. A full instructor solutions guide, covering all exercises in detail, is available for verified instructors.

Additional resources, including the full solutions guide, lecture slides, and supporting materials, can be found on the book's companion website:

https://www.roiyeho.com/ml-book

We hope this resource supports your learning experience.

Chapter 2

Supervised Machine Learning

2.1 (a), (b)

2.2 (c)

2.3 (a)

2.4 (b), (c)

2.5 (b), (c), (d)

2.6 (a)

2.7 (a), (d)

2.8 (a), (d)

2.9 (a), (c)

2.10 (a)

2.11 (a) Medical diagnosis and treatment recommendation. This problem can be framed as a supervised learning problem if historical patient data with known treatment outcomes are available. The inputs are patient symptoms, demographic information, and test results, and the output is a recommended treatment or diagnosis. This is typically a classification problem, though it may also be formulated as regression if predicting a continuous outcome such as risk or dosage.

(b) Customer-support chatbot. This is not a standard supervised learning problem in its full form, as chatbot behavior involves dialogue, context tracking, and decision making over time. While supervised learning may be used for subcomponents (e.g., intent classification or response ranking), modern chatbot systems often rely on large language models (LLMs), sometimes combined with retrieval-augmented generation or reinforcement learning, rather than a single supervised learning formulation.

(c) Flag offensive content in social media. This is a supervised learning problem when labeled examples of offensive and non-offensive content are available. The inputs are text, images, or videos posted by users, and the output is a binary or multi-class label indicating whether the content violates policy. This is a classification problem.

(d) Learn to play chess. This is not a supervised learning problem in the general case, as learning to play chess typically involves sequential decision making and long-term rewards. It is more naturally formulated as a reinforcement learning problem, though supervised learning may be used to imitate expert moves in an initial training phase (as in AlphaGo).

(e) Detect locations of ships in satellite images. This can be framed as a supervised learning problem if images are annotated with ship locations. The input is a satellite image, and the output is the presence and location of ships, often represented as bounding boxes or segmentation masks. This is a classification problem at the object level, combined with localization (or regression) for the coordinates.

(f) Movie recommendation system. This can be formulated as a supervised learning problem when past user–movie interactions and ratings are available. The inputs are user preferences and historical interactions, and the output is a predicted rating or relevance score for unseen movies, making it commonly a regression problem (or classification for like vs. dislike). In practice, recommendation systems also use other approaches such as collaborative filtering and matrix factorization, which are often considered weakly supervised or unsupervised since they exploit observed interaction patterns without explicit labels.

2.12 (a) By Bayes' theorem, the posterior probability of a tumor being malignant given its size x is:

$$P(\text{Malignant} \mid x) = \frac{p(x \mid \text{Malignant})\, P(\text{Malignant})}{p(x \mid \text{Malignant})\, P(\text{Malignant}) + p(x \mid \text{Benign}) P(\text{Benign})}.$$

Since the prior probabilities are equal ($P(\text{Malignant}) = P(\text{Benign}) = 0.5$), this simplifies to:

$$P(\text{Malignant} \mid x) = \frac{p(x \mid \text{Malignant})}{p(x \mid \text{Malignant}) + p(x \mid \text{Benign})}.$$

(b) The Bayes optimal classifier selects the class with the highest posterior probability. With equal priors, this is equivalent to:

$$\text{Predict Malignant if } p(x \mid \text{Malignant}) > p(x \mid \text{Benign}).$$

The decision boundary x^* is the value x at which the two densities are equal:

$$p(x^* \mid \text{Malignant}) = p(x^* \mid \text{Benign}).$$

Plugging in the Gaussian PDFs:

$$\frac{1}{\sqrt{2\pi} \cdot 8} \exp\left(-\frac{(x^* - 40)^2}{2 \cdot 8^2}\right) = \frac{1}{\sqrt{2\pi} \cdot 6} \exp\left(-\frac{(x^* - 20)^2}{2 \cdot 6^2}\right).$$

Cancel $\sqrt{2\pi}$ and take natural logs:

$$\begin{aligned} -\ln 8 - \frac{(x^* - 40)^2}{128} &= -\ln 6 - \frac{(x^* - 20)^2}{72} \\ \frac{(x^* - 20)^2}{72} - \frac{(x^* - 40)^2}{128} &= \ln \frac{4}{3}. \end{aligned}$$

Multiply by $\operatorname{lcm}(72, 128) = 1152$ and expand:

$$\begin{aligned} 16(x^* - 20)^2 - 9(x^* - 40)^2 &= 1152 \ln \frac{4}{3} \\ 7x^{*2} + 80x^* - 8000 &= 1152 \ln \frac{4}{3} \\ 7x^{*2} + 80x^* - 8000 - 1152 \ln \frac{4}{3} &= 0. \end{aligned}$$

By the quadratic formula,

$$x^* = \frac{-80 \pm \sqrt{230{,}400 + 32{,}256 \ln \frac{4}{3}}}{14}.$$

Thus the solutions are:

$$x_1^* \approx 29.255, \quad x_2^* \approx -40.683.$$

(c) The Bayes error E^* is the total probability of misclassification under the optimal decision rule:

$$E^* = \int_{-\infty}^{x^*} p(x \mid \text{Malignant})\, dx + \int_{x^*}^{\infty} p(x \mid \text{Benign})\, dx.$$

That is:

- The probability of predicting Benign when the tumor is actually Malignant (for $x < x^*$), plus
- The probability of predicting Malignant when the tumor is actually Benign (for $x > x^*$).

(d) The Bayes error represents the theoretical minimum classification error achievable under the given probabilistic model. It arises from the overlap between the two distributions and cannot be avoided, even by an ideal classifier with complete knowledge of the true data-generating process.

2.14 (a) For a single labeled data point (x, y), where $y \in \{0, 1\}$ and the success probability is p, the likelihood of observing y is given by the Bernoulli probability mass function:

$$\mathcal{L}(p) = p^y (1-p)^{1-y}.$$

This formula handles both cases:

- If $y = 1$, the expression becomes p.
- If $y = 0$, the expression becomes $1 - p$.

(b) For n independent data points $(x_1, y_1), \ldots, (x_n, y_n)$, where each has success probability p_i, the joint likelihood (under the independence assumption) is the product of the individual likelihoods:

$$\mathcal{L} = \prod_{i=1}^{n} p_i^{y_i} (1-p_i)^{1-y_i}.$$

(c) The log-likelihood is the natural logarithm of the likelihood function:

$$\begin{aligned} \ell = \log \mathcal{L} &= \log \left(\prod_{i=1}^{n} p_i^{y_i} (1-p_i)^{1-y_i} \right) \\ &= \sum_{i=1}^{n} \left[y_i \log p_i + (1-y_i) \log(1-p_i) \right]. \end{aligned}$$

(d) The negative log-likelihood (NLL) is simply the negative of the log-likelihood:

$$\text{NLL} = -\ell = -\sum_{i=1}^{n} \left[y_i \log p_i + (1 - y_i) \log(1 - p_i) \right].$$

This function is known as the log loss or binary cross-entropy loss.

2.16 (a) By Bayes' theorem, the posterior distribution of the model parameters $\boldsymbol{\theta}$ given the training data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ is:

$$p(\boldsymbol{\theta}|D) = \frac{p(D|\boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(D)},$$

where:

- $p(D|\boldsymbol{\theta})$ is the likelihood of the data given the parameters,
- $p(\boldsymbol{\theta})$ is the prior distribution over the parameters,
- $p(D) = \int p(D|\boldsymbol{\theta}) p(\boldsymbol{\theta})\, d\boldsymbol{\theta}$ is the marginal likelihood (normalizing constant).

(b) If the prior $p(\boldsymbol{\theta})$ is uniform, then it is constant with respect to $\boldsymbol{\theta}$ and does not affect the maximization. Maximizing the posterior becomes:

$$\boldsymbol{\theta}_{\text{Bayesian}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{\theta}|D) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \frac{p(D|\boldsymbol{\theta})\, c}{p(D)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(D|\boldsymbol{\theta}),$$

since $p(D)$ and the prior constant c do not depend on $\boldsymbol{\theta}$. This shows that the Bayesian estimator reduces to the maximum likelihood estimator (MLE) when the prior is uniform:

$$\boldsymbol{\theta}_{\text{Bayesian}} = \boldsymbol{\theta}_{\text{ML}}.$$

(c) In supervised learning, this result means that when no prior knowledge about the model parameters is available (i.e., assuming a uniform prior), Bayesian estimation and maximum likelihood estimation lead to the same parameter estimates.

2.17 (a) Typically, adding more training data helps the model generalize better and reduces overfitting by improving both bias and variance, although in some cases, with a model that is too simple, additional data might not help improve performance.

(b) Adding more features tends to decrease the bias of a model, as it allows the model to capture more complexity in the data. However, this can also lead to an increase in variance, especially if these features are not relevant or are noisy, as the model might start fitting to the noise in the data.

(c) Switching to a more complex model architecture, such as moving from a linear model to a non-linear one or using a deep neural network with more layers, typically decreases bias as it allows the model to capture more complex patterns in the data. However, this increased complexity often leads to a higher variance, as the model becomes more sensitive to fluctuations in the training data and might overfit, capturing noise instead of the underlying trend.

(d) Training a model for a longer time can lead to a decrease in bias if the additional training helps the model to better learn the underlying patterns. However, excessive training can also increase variance, especially in complex models, due to overfitting to the training data.

(e) Removing outliers mainly affects the variance of the model rather than the bias. It might help in reducing the variance as the model becomes less sensitive to extreme values that are not representative of the underlying data distribution.

(f) Reducing the strength of regularization typically decreases the bias of a model, as it allows more flexibility in fitting the training data. However, this can increase variance because the model may start to overfit.

2.19 Solution available at `MaximumLikelihoodEx.ipynb`.

2.20 Solution available at `BayesErrorEx.ipynb`.

2.21 Solution available at `GradientDescentEx.ipynb`.

Chapter 3

Introduction to Scikit-Learn

3.1 (d)

3.2 (a), (b)

3.3 (a), (b), (c)

3.4 (b)

3.5 (a)

3.6 (b), (d)

3.7 (b)

3.8 (b), (c)

3.9 (b), (c), (d)

3.10 (a)

Chapter 4

Linear Regression

4.1 (a)

4.2 (d)

4.3 (a), (d)

4.4 (a)

4.5 (b)

4.6 (b), (c)

4.7 (b), (c), (d)

4.8 (a), (c), (d)

4.9 (c), (d), (e)

4.10 (c)

4.11 (a) The linear regression model can be written as:

$$\hat{y} = w_0 + w_1 x.$$

The parameters w_0 (intercept) and w_1 (slope) are derived using the following formulas:

$$w_1 = \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i \right)^2},$$

$$w_0 = \frac{\sum_{i=1}^{n} y_i - w_1 \sum_{i=1}^{n} x_i}{n}.$$

Using the given data:

$$x = [1, 2, 4, 6, 8], \quad y = [52, 59, 67, 81, 90]$$

We compute the necessary summations:

$$\sum_i x_i = 1 + 2 + 4 + 6 + 8 = 21,$$
$$\sum_i y_i = 52 + 59 + 67 + 81 + 90 = 349,$$
$$\sum_i x_i^2 = 1^2 + 2^2 + 4^2 + 6^2 + 8^2 = 121,$$
$$\sum_i x_i y_i = 52 + 118 + 268 + 486 + 720 = 1644.$$

Substituting into the formula for w_1:

$$w_1 = \frac{5 \cdot 1644 - 21 \cdot 349}{5 \cdot 121 - 21^2} = \frac{891}{164} = 5.433.$$

Substitute $w_1 = 5.43$ into the formula for w_0:

$$w_0 = \frac{349 - 114.093}{5} = \frac{234.907}{5} = 46.981.$$

The final linear regression model is:

$$\hat{y} = 46.981 + 5.433x.$$

(b) The Mean Squared Error is given by:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Substituting $\hat{y}_i = 46.981 + 5.433x_i$, we calculate the predictions for each data point:

$$\hat{y}_1 = 46.981 + 5.433 \cdot 1 = 52.414,$$
$$\hat{y}_2 = 46.981 + 5.433 \cdot 2 = 57.847,$$
$$\hat{y}_3 = 46.981 + 5.433 \cdot 4 = 68.713,$$
$$\hat{y}_4 = 46.981 + 5.433 \cdot 6 = 79.579,$$
$$\hat{y}_5 = 46.981 + 5.433 \cdot 8 = 90.445.$$

Summing the squared errors:

$$(52-52.414)^2+(59-57.847)^2+(67-68.713)^2+(81-79.579)^2+(90-90.445)^2 = 6.653.$$

Therefore, the MSE is:

$$\text{MSE} = \frac{6.653}{5} = 1.331.$$

(c) Using the derived model $\hat{y} = 46.981 + 5.433x$:
For $x = 3$:

$$\hat{y} = 46.981 + 5.433 \cdot 3 = 46.981 + 16.299 = 63.280.$$

For $x = 5$:

$$\hat{y} = 46.981 + 5.433 \cdot 5 = 46.981 + 27.165 = 74.146.$$

4.13 We begin with the cost function for linear regression, which measures the mean squared error between the predicted values and the true outputs:

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=0}^{d} w_j x_{ij}\right)^2,$$

where $x_{i0} = 1$ for all i, and $\mathbf{w} \in \mathbb{R}^{d+1}$ is the weight vector (including the bias term).

To minimize this cost function, we take the partial derivative with respect to each weight w_k, for $k = 0, 1, \ldots, d$, and set it to zero:

$$\frac{\partial J(\mathbf{w})}{\partial w_k} = \frac{\partial}{\partial w_k}\left[\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=0}^{d} w_j x_{ij}\right)^2\right] = 0.$$

Applying the chain rule:

$$\frac{\partial J(\mathbf{w})}{\partial w_k} = \frac{1}{n}\sum_{i=1}^{n} 2\left(y_i - \sum_{j=0}^{d} w_j x_{ij}\right)\cdot(-x_{ik}) = 0.$$

Simplifying:

$$\sum_{i=1}^{n} x_{ik} \left(y_i - \sum_{j=0}^{d} w_j x_{ij} \right) = 0,$$

$$\sum_{i=1}^{n} x_{ik} y_i = \sum_{i=1}^{n} \sum_{j=0}^{d} w_j x_{ik} x_{ij}.$$

Switching the order of summation on the right-hand side:

$$\sum_{i=1}^{n} x_{ik} y_i = \sum_{j=0}^{d} w_j \sum_{i=1}^{n} x_{ik} x_{ij}.$$

This gives us one linear equation for each $k = 0, 1, \ldots, d$. We can now write this system in matrix form.

Define the design matrix $X \in \mathbb{R}^{n \times (d+1)}$, where each row is $\mathbf{x}_i^T$, the weight vector $\mathbf{w} \in \mathbb{R}^{d+1}$, and the output vector $\mathbf{y} \in \mathbb{R}^n$.

The (k, j)-th entry of the matrix $X^T X$ is:

$$[X^T X]_{kj} = \sum_{i=1}^{n} x_{ik} x_{ij},$$

and the k-th entry of the vector $X^T X \mathbf{w}$ is:

$$\left[X^T X \mathbf{w}\right]_k = \sum_{j=0}^{d} [X^T X]_{kj} w_j = \sum_{j=0}^{d} w_j \sum_{i=1}^{n} x_{ik} x_{ij}.$$

Similarly, the k-th entry of the vector $X^T \mathbf{y}$ is:

$$[X^T \mathbf{y}]_k = \sum_{i=1}^{n} x_{ik} y_i.$$

Therefore, the system

$$\sum_{i=1}^{n} x_{ik} y_i = \sum_{j=0}^{d} w_j \sum_{i=1}^{n} x_{ik} x_{ij}, \quad \text{for } k = 0, \ldots, d$$

can be written in matrix form as:

$$X^T \mathbf{y} = X^T X \mathbf{w},$$

which are the normal equations.

4.17 Duplicating every sample in the dataset D will not affect the learned weights of the linear regression model. To see why, recall that the objective in linear regression is to minimize the mean squared error (MSE), which is given by:

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(h(\mathbf{x}_i) - y_i)^2$$

Duplicating every sample effectively doubles the number of terms in the summation, but the cost function is divided by n, the number of observations. After duplication, the total number of observations becomes $2n$, and the cost function becomes:

$$J_{\text{new}}(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{2n}(h(\mathbf{x}_i) - y_i)^2$$

Since each sample is now duplicated, the summation simply repeats the same terms as before, and the factor of $\frac{1}{2n}$ cancels out the effect of duplication. As a result, the minimization of the cost function leads to the same weights as before.

Thus, the learned weights of the model will remain unchanged after duplicating the dataset.

4.18 We begin by expressing the least squares cost function in matrix form:

$$J(\mathbf{w}) = \frac{1}{n}(X\mathbf{w} - \mathbf{y})^T(X\mathbf{w} - \mathbf{y}).$$

Now, we need to compute the gradient of the cost function with respect to $\mathbf{w}$:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{2}{n}X^T(X\mathbf{w} - \mathbf{y}).$$

Next, we compute the Hessian of the cost function by differentiating the gradient with respect to $\mathbf{w}$:

$$H = \nabla_{\mathbf{w}}\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{2}{n}X^T X.$$

Since the Hessian is $\frac{2}{n}X^T X$, we need to check whether this matrix is positive semidefinite. A matrix is positive semidefinite if for any non-zero vector $\mathbf{z}$, the following condition holds:

$$\mathbf{z}^T X^T X\mathbf{z} \geq 0.$$

Notice that:

$$\mathbf{z}^T X^T X\mathbf{z} = (X\mathbf{z})^T(X\mathbf{z}) = \|X\mathbf{z}\|^2.$$

Since the square of a norm is always nonnegative, we conclude that $\mathbf{z}^T X^T X \mathbf{z} \geq 0$. Therefore, $X^T X$ is positive semidefinite. As a result, the Hessian is positive semidefinite, which proves that the least squares cost function is convex.

4.21 (a) The system of equations corresponding to a parabola that best fits these points has the following design matrix:

$$X = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \\ 1 & x_5 & x_5^2 \end{pmatrix} = \begin{pmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix}$$

parameters vector

$$\mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

and labels vector

$$\mathbf{y} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

We can calculate that:

$$X^T X = \begin{pmatrix} 5 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 34 \end{pmatrix}$$

and

$$X^T \mathbf{y} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Therefore, the least squares solution satisfies:

$$\begin{pmatrix} 5 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 34 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Solving this system of equations gives: $w_0 = \frac{17}{35}, w_1 = 0, w_2 = -\frac{1}{7}$, so that the best-fitting quadratic is

$$y = \frac{17}{35} - \frac{1}{7}x^2.$$

(b) Figure 4.1 shows the data points and the best fitting parabola.

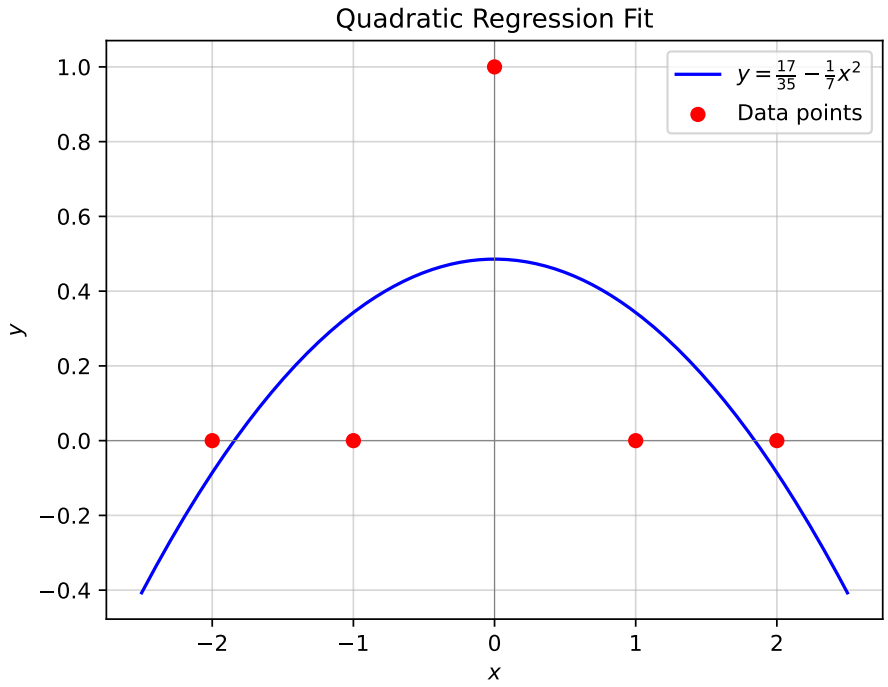


Figure 4.1: The best-fitting quadratic curve $y = \frac{17}{35} - \frac{1}{7}x^2$ obtained via least squares, shown alongside the five data points.

4.24 (a) We first write $J(\mathbf{w})$ in vector notation:

$$J(\mathbf{w}) = \sum_{i=1}^{n}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = (X\mathbf{w} - \mathbf{y})^T(X\mathbf{w} - \mathbf{y}) + \lambda\mathbf{w}^T\mathbf{w}$$

Next, we compute the derivative of $J(\mathbf{w})$ with respect to $\mathbf{w}$:

$$\begin{aligned}
\nabla_{\mathbf{w}} J(\mathbf{w}) &= \nabla_{\mathbf{w}} \left((X\mathbf{w} - \mathbf{y})^T (X\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w} \right) \\
&= \nabla_{\mathbf{w}} \left((X\mathbf{w})^T X\mathbf{w} - (X\mathbf{w})^T \mathbf{y} - \mathbf{y}^T (X\mathbf{w}) + \mathbf{y}^T \mathbf{y} + \lambda \mathbf{w}^T \mathbf{w} \right) \\
&= \nabla_{\mathbf{w}} \left(\mathbf{w}^T X^T X\mathbf{w} - \mathbf{y}^T (X\mathbf{w}) - \mathbf{y}^T (X\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w} \right) \\
&= \nabla_{\mathbf{w}} \left(\mathbf{w}^T (X^T X)\mathbf{w} - 2(X^T \mathbf{y})^T \mathbf{w} + \lambda \mathbf{w}^T \mathbf{w} \right) \\
&= 2X^T X\mathbf{w} - 2X^T \mathbf{y} + 2\lambda \mathbf{w}
\end{aligned}$$

To find the minimum of $J(\mathbf{w})$, we set its derivative to 0:

$$\begin{aligned}
2X^T X\mathbf{w} - 2X^T \mathbf{y} + 2\lambda \mathbf{w} &= 0 \\
X^T X\mathbf{w} + \lambda \mathbf{w} &= X^T \mathbf{y} \\
(X^T X + \lambda I)\mathbf{w} &= X^T \mathbf{y} \\
\mathbf{w} &= \left(X^T X + \lambda I \right)^{-1} X^T \mathbf{y}
\end{aligned}$$

(b) $\lambda = 0$: This is the same as ordinary linear regression, so the optimal w will be the same as in ordinary linear regression.

$\lambda \to +\infty$: Any non-zero value of weight w_j would cause λw_j^2 to be very large, which is not preferred by the objective, so the optimal w will have very small (or zero at $\lambda = +\infty$) weights, except for w_0. The value of w_0 will be the same as in ordinary linear regression.

$\lambda \to -\infty$: Since λw_j^2 is negative for $w_j \neq 0$, which is preferred when minimizing $J(w)$ (the ideal minimum value is $J(w) = -\infty$), the optimal value of w_j is $\pm\infty$ (such that λw_j^2 is as negative as possible). The value of w_0 will be the same as in ordinary linear regression. Note that this argument actually applies for *all* $\lambda < 0$, and since the optimal solution found in this case is not meaningful, we tend to use $\lambda \geq 0$ in practice.

(c) Note that we have been careful to ensure the index of the derivative (j) is

different from the index used in the summation of the regularizer (j').

$$\begin{aligned}
\frac{\partial}{\partial w_j} J(w) &= \frac{\partial}{\partial w_j} \left[\sum_{i=1}^{n} (h(\mathbf{x}_i) - y_i)^2 + \lambda \sum_{j'=0}^{d} w_{j'}^2 \right] \\
&= \sum_{i=1}^{n} \frac{\partial}{\partial w_j} (h(\mathbf{x}_i) - y_i)^2 + \lambda \sum_{j'=0}^{d} \frac{\partial}{\partial w_j} w_{j'}^2 \\
&= \sum_{i=1}^{n} 2(h(\mathbf{x}_i) - y_i) \frac{\partial}{\partial w_j} (h(\mathbf{x}_i) - y_i) + \lambda \sum_{j'=0}^{d} \begin{cases} 2w_{j'} & \text{if } j' = j \\ 0 & \text{if } j' \neq j \end{cases} \\
&= \sum_{i=1}^{n} 2(h(\mathbf{x}_i) - y_i) \frac{\partial}{\partial w_j} (w_0 + w_1 x_1^{(i)} + \ldots + w_j x_{ij} + \ldots + w_d x_i - y_i) + \lambda 2 w_j \\
&= \sum_{i=1}^{n} 2(h(\mathbf{x}_i) - y_i)(0 + 0 + \ldots + x_{ij} + \ldots + 0 - 0) + 2\lambda w_j \\
&= 2 \sum_{i=1}^{n} [(h(\mathbf{x}_i) - y_i) x_{ij}] + 2\lambda w_j
\end{aligned}$$

(d) The gradient descent update rule is:

$$w_j^{k+1} \leftarrow w_j^k - \alpha \frac{\partial}{\partial w_j} J(\mathbf{w}^k) = w_j^k - \alpha \left\{ 2 \sum_{i=1}^{n} [(h_{\mathbf{w}^k}(x_i) - y_i) x_{ij}] + 2\lambda w_j^k \right\}$$

Compared to the gradient descent rule for ordinary linear regression, we have an extra $\frac{1}{n}$ factor in front of the summation due to how the error function was defined; the impact of this is insignificant (because we can simply multiple the learning rate α by N, which is a known constant). More importantly, there is an extra $-2\alpha\lambda w_j$ term in the update rule. Since both $\alpha > 0$ and $\lambda > 0$, this term is *positive* when $w_j < 0$, and *negative* when $w_j > 0$. The term's effect on the next iterate of w_j is therefore to move it closer to 0 (increasing it when w_j is negative, decreasing it when w_j is positive).

This assumes that α is small, which is typically the case; otherwise, the weight may move past 0 and have a sign change. Another way to view this is to group together the w_j^k terms on the right hand side:

$$w_j^{k+1} \leftarrow (1 - 2\alpha\lambda)\, w_j^k - \alpha 2 \sum_{i=1}^{n} (h_{\mathbf{w}^k}(x_i) - y_i) x_{ij}$$

The second term is now the same as in ordinary linear regression (with an extra $\frac{1}{n}$ factor), and the first term isolates the effect of λ. Here, we see that, for $0 < \alpha < \frac{1}{2\lambda}$, the weight decreases via multiplication by $0 < (1-2\alpha\lambda) < 1$. In statistics, this effect is known as *weight shrinkage*.

(e) A common method is k-fold cross-validation. The data is split into k folds, and for each candidate λ, the model is trained on $k-1$ folds and evaluated on the remaining one. Averaging the validation errors gives an estimate of performance, and the λ with the lowest error is selected.

Another approach is to use a validation curve. The model is trained for a range of λ values (often on a log scale), and the validation error is plotted as a function of λ. The best λ is then chosen as the one that yields the lowest validation error or a good balance between accuracy and model simplicity.

Chapter 5

Logistic Regression

5.1 (b), (e)

5.2 (c)

5.3 (b), (d)

5.4 (d)

5.5 (c), (e)

5.6 (a), (c), (d), (e), (f)

5.7 (c)

5.8 (a), (b)

5.9 (a), (b), (e)

5.10 (b)

5.11 The logistic regression model calculates the log-odds as:

$$z = w_0 + w_1 \cdot \text{BMI} + w_2 \cdot \text{age}.$$

Substituting the given values:

$$z = -1.0 + 0.05 \cdot 35 + 0.03 \cdot 50 = 2.25.$$

Next, the probability of having diabetes is calculated using the sigmoid function:

$$P(\text{diabetes}) = \frac{1}{1+e^{-2.25}} \approx \frac{1}{1+0.1054} \approx 0.689.$$

Thus, the probability that a patient with a BMI of 35 and age of 50 has diabetes according to the model is approximately 68.9%.

5.12 (a) The logistic regression model predicts probabilities using the sigmoid function:

$$p(x) = \sigma(w_0 + w_1 x) = \frac{1}{1+e^{-(w_0+w_1 x)}}.$$

The log-likelihood function $L(w_0, w_1)$ is given by:

$$L(w_0, w_1) = \sum_{i=1}^{n} \left[y_i \log p_i + (1-y_i) \log(1-p_i) \right],$$

where $p_i = \sigma(w_0 + w_1 x_i)$ is the predicted probability for sample i.

(b) The gradients of the negative log-likelihood (i.e., the log loss) with respect to w_0 and w_1 are:

$$\frac{\partial L}{\partial w_0} = \frac{1}{n} \sum_{i=1}^{n} (p_i - y_i),$$

$$\frac{\partial L}{\partial w_1} = \frac{1}{n} \sum_{i=1}^{n} (p_i - y_i) x_i.$$

(c) We will perform 3 iterations of batch gradient descent to minimize the negative log-likelihood (NLL). The update rules are:

$$w_0 \leftarrow w_0 - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} (p_i - y_i),$$

$$w_1 \leftarrow w_1 - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} (p_i - y_i) x_i.$$

We initialize the weights as $w_0 = 0$ and $w_1 = 0$, and set the learning rate $\alpha = 0.5$.

Iteration 1:

- Since $w_0 = 0$ and $w_1 = 0$, the logits for all samples are zero and the predicted probabilities are:

$$p_1 = p_2 = p_3 = \sigma(0) = 0.5.$$

- Gradients:

$$\frac{\partial L}{\partial w_0} = \frac{1}{3}\left[(0.5-0) + (0.5-0) + (0.5-1)\right] \approx 0.1667$$
$$\frac{\partial L}{\partial w_1} = \frac{1}{3}\left[(0.5-0)\cdot 1 + (0.5-0)\cdot 2 + (0.5-1)\cdot 10\right] \approx -1.1667$$

- Update weights:

$$w_0 = 0 - 0.5 \cdot 0.1667 = -0.0833$$
$$w_1 = 0 - 0.5 \cdot (-1.1667) = 0.5833$$

Iteration 2:

- Logits and probabilities:

$$z_1 = -0.0833 + 0.5833 \cdot 1 = 0.5 \quad \Rightarrow p_1 = \sigma(0.5) \approx 0.6225$$
$$z_2 = -0.0833 + 0.5833 \cdot 2 = 1.0833 \quad \Rightarrow p_2 = \sigma(1.0833) \approx 0.7471$$
$$z_3 = -0.0833 + 0.5833 \cdot 10 = 5.7497 \quad \Rightarrow p_3 = \sigma(5.7497) \approx 0.9968$$

- Gradients:

$$\frac{\partial L}{\partial w_0} = \frac{1}{3}\left[(0.6225-0) + (0.7471-0) + (0.9968-1)\right] \approx 0.4555$$
$$\frac{\partial L}{\partial w_1} = \frac{1}{3}\left[(0.6225-0)\cdot 1 + (0.7471-0)\cdot 2 + (0.9968-1)\cdot 10\right] \approx 0.6949$$

- Update weights:

$$w_0 = -0.0833 - 0.5 \cdot 0.4555 = -0.3111$$
$$w_1 = 0.5833 - 0.5 \cdot 0.6949 = 0.2358$$

Iteration 3:

- Logits and probabilities:

$$z_1 = -0.3111 + 0.2358 \cdot 1 = -0.0753 \quad \Rightarrow p_1 = \sigma(-0.0753) \approx 0.4812$$
$$z_2 = -0.3111 + 0.2358 \cdot 2 = 0.1605 \quad \Rightarrow p_2 = \sigma(0.1605) \approx 0.5401$$
$$z_3 = -0.3111 + 0.2358 \cdot 10 = 2.0469 \quad \Rightarrow p_3 = \sigma(2.0469) \approx 0.8856$$

- Gradients:

$$\frac{\partial L}{\partial w_0} = \frac{1}{3}\left[(0.4812-0)+(0.5401-0)+(0.8856-1)\right] \approx 0.3023$$
$$\frac{\partial L}{\partial w_1} = \frac{1}{3}\left[(0.4812-0)\cdot 1+(0.5401-0)\cdot 2+(0.8856-1)\cdot 10\right] \approx 0.1393$$

- Update weights:

$$w_0 = -0.3111 - 0.5 \cdot 0.3023 = -0.4622$$
$$w_1 = 0.2358 - 0.5 \cdot 0.1393 = 0.1662$$

The final predicted probabilities are:

$$z_1 = -0.4622 + 0.1662 \cdot 1 = -0.2960 \quad \Rightarrow p_1 = \sigma(-0.2960) \approx 0.4265$$
$$z_2 = -0.4622 + 0.1662 \cdot 2 = -0.1298 \quad \Rightarrow p_2 = \sigma(-0.1298) \approx 0.4676$$
$$z_3 = -0.4622 + 0.1662 \cdot 10 = 1.1998 \quad \Rightarrow p_3 = \sigma(1.1998) \approx 0.7685$$

Using a threshold of 0.5 for classification, the predicted labels after 3 iterations are:

$$\hat{y}_1 = 0, \quad \hat{y}_2 = 0, \quad \hat{y}_3 = 1,$$

which match the true labels in the dataset. Thus, after just 3 iterations of batch gradient descent, the model correctly classifies all training examples.

5.15 (a) We begin with the vectorized gradient of the logistic regression cost function:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{n} X^T (\mathbf{p} - \mathbf{y}),$$

where $\mathbf{p} = \sigma(X\mathbf{w})$ is the vector of predicted probabilities, and $\sigma(z) = \frac{1}{1+e^{-z}}$ is applied elementwise.

To compute the Hessian, we differentiate the gradient with respect to $\mathbf{w}$:

$$H(\mathbf{w}) = \nabla^2_{\mathbf{w}} J(\mathbf{w}) = \frac{\partial}{\partial \mathbf{w}} \left(\frac{1}{n} X^T \mathbf{p} \right).$$

Since only $\mathbf{p}$ depends on $\mathbf{w}$, we apply the chain rule:

$$\frac{\partial \mathbf{p}}{\partial \mathbf{w}} = \frac{\partial \mathbf{p}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \mathbf{w}}, \quad \text{where } \mathbf{z} = X\mathbf{w}.$$

The Jacobian of $\mathbf{p}$ with respect to $\mathbf{z}$ is diagonal because each $p_i = \sigma(z_i)$ satisfies:

$$\frac{\partial p_i}{\partial z_i} = \sigma(z_i)(1 - \sigma(z_i)) = p_i(1 - p_i),$$

so:

$$\frac{\partial \mathbf{p}}{\partial \mathbf{z}} = \text{diag}(p_i(1 - p_i)) = R.$$

Since $\mathbf{z} = X\mathbf{w}$, we have $\frac{\partial \mathbf{z}}{\partial \mathbf{w}} = X$, and thus:

$$\frac{\partial \mathbf{p}}{\partial \mathbf{w}} = RX.$$

Substituting this into the Hessian expression gives:

$$H(\mathbf{w}) = \frac{1}{n} X^T R X,$$

where $R = \text{diag}(r_1, r_2, \ldots, r_n) \in \mathbb{R}^{n \times n}$ is a diagonal matrix, and each diagonal entry corresponds to the derivative of the sigmoid function for sample i:

$$r_i = p_i(1 - p_i) = \sigma(\mathbf{w}^T \mathbf{x}_i)(1 - \sigma(\mathbf{w}^T \mathbf{x}_i)).$$

(b) To show convexity, note that each $r_i = \sigma(z)(1 - \sigma(z)) \geq 0$. Thus, R is a diagonal matrix with non-negative entries.
Hence, for any vector $\mathbf{u} \in \mathbb{R}^d$:

$$\mathbf{u}^T H(\mathbf{w})\mathbf{u} = \mathbf{u}^T X^T R X \mathbf{u} = (X\mathbf{u})^T R (X\mathbf{u}) = \sum_{i=1}^{n} r_i (\mathbf{x}_i^T \mathbf{u})^2 \geq 0.$$

This proves that the Hessian matrix is positive semi-definite, and therefore, the logistic regression cost function $J(\mathbf{w})$ is convex.

(c) The Hessian matrix $H(\mathbf{w}) = X^T R X$ is invertible if and only if it is positive definite. This requires:

- Full rank data matrix X: Columns of X must be linearly independent.
- Positive diagonal entries in R: Entries $r_i = \sigma(\mathbf{w}^T \mathbf{x}_i)(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))$ must all be strictly positive, meaning the predicted probabilities p_i should not be exactly 0 or 1.

If any predicted probability becomes exactly 0 or 1, corresponding entries of R become zero, reducing the rank of $X^T R X$. In this case, the Hessian is singular and not invertible, causing issues in the Newton update step.

(d) Using the Hessian and gradient, the Newton update for logistic regression is given by:

$$\mathbf{w} \leftarrow \mathbf{w} - H(\mathbf{w})^{-1}\nabla_{\mathbf{w}}J(\mathbf{w}) = \mathbf{w} - (X^T RX)^{-1}X^T(\mathbf{p} - \mathbf{y}).$$

This update is iteratively applied until convergence to find the optimal logistic regression parameters

5.18 Precision is defined as the proportion of true positives (TP) among all predicted positives (TP + FP):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

Assume the positive class constitute $\alpha\%$ of the dataset, so the number of positive samples is

$$p = \alpha n,$$

where n is the total number of samples.

Then the number of negative samples is:

$$n_{\text{neg}} = (1 - \alpha)n.$$

Suppose the classifier predicts each sample as positive independently with probability β. Then:

- The expected number of true positives (correctly predicted positives) is:

$$\text{TP} = \beta \cdot p = \beta\alpha n.$$

- The expected number of false positives (negative samples predicted as positive) is:

$$\text{FP} = \beta \cdot n_{\text{neg}} = \beta(1 - \alpha)n.$$

Substituting into the definition of precision:

$$\text{Precision} = \frac{\beta\alpha n}{\beta\alpha n + \beta(1 - \alpha)n} = \frac{\alpha}{\alpha + (1 - \alpha)} = \alpha.$$

Thus, the expected precision of the classifier is α.

Recall is defined as the proportion of true positives among all actual positives:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

The expected number of false negatives (actual positives predicted as negative) is:

$$\text{FN} = (1 - \beta)p = (1 - \beta)\alpha n$$

Therefore,

$$\text{Recall} = \frac{\beta\alpha n}{\beta\alpha n + (1 - \beta)\alpha n} = \frac{\beta\alpha n}{\alpha n} = \beta.$$

Thus, the expected recall of the classifier is β.

5.19 The statement is false. Here is a counterexample demonstrating that increasing recall does not necessarily lead to a decrease in precision.

Consider the following confusion matrix for a binary classifier:

	Predicted Positive	Predicted Negative
Actual Positive	8	2
Actual Negative	2	8

We compute:

$$\text{Precision} = \frac{8}{8 + 2} = 0.8,$$
$$\text{Recall} = \frac{8}{8 + 2} = 0.8.$$

Now suppose the classifier improves and correctly classifies one additional positive instance. The new confusion matrix becomes:

	Predicted Positive	Predicted Negative
Actual Positive	9	1
Actual Negative	2	8

We now have:

$$\text{Precision} = \frac{9}{9 + 2} = 0.818,$$
$$\text{Recall} = \frac{9}{9 + 1} = 0.9.$$

Both precision and recall increased. This shows that it is possible for recall to increase without precision decreasing. While precision and recall often move in opposite directions (especially when varying a decision threshold), they are not inherently inversely related.

5.23 Consider the normal distribution used in ordinary least squares (OLS) linear regression. Its probability density function is

$$p(y|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right),$$

where $\mu = \mathbb{E}[y]$ is the mean of the distribution.

In linear regression, this mean is modeled using the identity link function, so that

$$\mu = \eta = \mathbf{w}^T\mathbf{x}.$$

We expand the exponent in the Gaussian density:

$$-\frac{(y-\mu)^2}{2\sigma^2} = -\frac{1}{2\sigma^2}\left(y^2 - 2y\mu + \mu^2\right) = -\frac{y^2}{2\sigma^2} + \frac{y\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2}.$$

Substituting back into the density gives:

$$\begin{aligned} p(y|\mu,\sigma^2) &= \exp\left(-\frac{1}{2}\log(2\pi\sigma^2) - \frac{y^2}{2\sigma^2} + \frac{y\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2}\right) \\ &= \exp\left(\frac{y\mu - \frac{1}{2}\mu^2}{\sigma^2} - \frac{y^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)\right). \end{aligned}$$

This expression is in the standard exponential-family form:

$$p(y|\mu,\sigma^2) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right),$$

with the following components:

$$\begin{aligned} \theta &= \mu, \\ b(\theta) &= \frac{1}{2}\theta^2, \\ a(\phi) &= \sigma^2, \\ c(y,\phi) &= -\frac{y^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2). \end{aligned}$$

This shows that OLS linear regression is a special case of a generalized linear model with a normal distribution and the identity link function.

Chapter 6

K-Nearest Neighbors

6.1 (c)

6.2 (b), (c)

6.3 (b)

6.4 (b)

6.5 (c), (d)

6.6 (b), (c), (d)

6.7 (c), (d)

6.8 (a), (b), (c)

6.9 (a), (d)

6.10 (c), (e)

6.11 (a) The Euclidean distance between two vectors $\mathbf{x} = (x_1, x_2, x_3)$ and $\mathbf{y} = (y_1, y_2, y_3)$ is given by:

$$d_{\text{Euclidean}}(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}.$$

The Euclidean distances from $\mathbf{x} = (0, 1, 2)$ to each point are shown in the following table:

The 3 nearest neighbors (smallest distances) are points:

- $i = 6$ with distance 1.41 and label $y = 1$.

i	x_1	x_2	x_3	y	Euclidean Distance to $(0,1,2)$
1	2	3	2	0	$\sqrt{(2-0)^2+(3-1)^2+(2-2)^2} = \sqrt{8} \approx 2.83$
2	3	-1	4	1	$\sqrt{(3-0)^2+(-1-1)^2+(4-2)^2} = \sqrt{17} \approx 4.12$
3	-2	1	0	0	$\sqrt{(-2-0)^2+(1-1)^2+(0-2)^2} = \sqrt{8} \approx 2.83$
4	0	3	-2	1	$\sqrt{(0-0)^2+(3-1)^2+(-2-2)^2} = \sqrt{20} \approx 4.47$
5	3	-2	2	0	$\sqrt{(3-0)^2+(-2-1)^2+(2-2)^2} = \sqrt{18} \approx 4.24$
6	-1	1	3	1	$\sqrt{(-1-0)^2+(1-1)^2+(3-2)^2} = \sqrt{2} \approx 1.41$
7	2	-3	0	0	$\sqrt{(2-0)^2+(-3-1)^2+(0-2)^2} = \sqrt{24} \approx 4.90$
8	-2	1	-1	1	$\sqrt{(-2-0)^2+(1-1)^2+(-1-2)^2} = \sqrt{13} \approx 3.61$
9	1	-2	2	0	$\sqrt{(1-0)^2+(-2-1)^2+(2-2)^2} = \sqrt{10} \approx 3.16$
10	0	2	-1	1	$\sqrt{(0-0)^2+(2-1)^2+(-1-2)^2} = \sqrt{10} \approx 3.16$

- $i = 1$ with distance 2.83 and label $y = 0$.
- $i = 3$ with distance 2.83 and label $y = 0$.

Majority Vote: Two of the three neighbors have label $y = 0$, so the new vector $\mathbf{x} = (0, 1, 2)$ is classified as $y = 0$.

(b) In distance-weighted voting, closer neighbors have a higher influence. The weight of each neighbor is given by:

$$\text{Weight of neighbor } i = \frac{1}{\text{distance to } i}.$$

The weights for the three nearest neighbors are:

- $i = 1$: $1/2.83 \approx 0.35$ for $y = 0$.
- $i = 3$: $1/2.83 \approx 0.35$ for $y = 0$.
- $i = 6$: $1/1.41 \approx 0.71$ for $y = 1$.

Weighted sum of labels:

- For $y = 0$: The weighted sum is $0.35 + 0.35 = 0.70$ (from points 1 and 3).
- For $y = 1$: The weighted sum is 0.71 (from point 6).

The weighted sums are close, but $y = 1$ has a slightly higher weight due to its proximity. Hence, using distance-weighted voting, the new vector $\mathbf{x} = (0, 1, 2)$ is classified as $y = 1$.

The label differs when using distance-weighted voting compared to the unweighted KNN method.

6.3 The expected squared prediction error of a model can be decomposed as:

$$\mathbb{E}\left[\left(\hat{f}(\mathbf{x}) - f(\mathbf{x})\right)^2\right] = \text{Bias}^2\left(\hat{f}(\mathbf{x})\right) + \text{Var}\left(\hat{f}(\mathbf{x})\right) + \varepsilon^2,$$

where $\hat{f}(\mathbf{x})$ is the model's prediction at input $\mathbf{x}$, $f(\mathbf{x})$ is the true underlying function, and ε^2 is the irreducible noise.

We now analyze how varying the number of neighbors k in the KNN algorithm affects the bias and variance components of the error.

- Small k: Low bias, high variance.

  When k is very small (e.g., $k = 1$), the model predicts the label of the nearest neighbor

  $$\hat{f}(\mathbf{x}) = y_{(1)},$$

  where $y_{(1)}$ is the output of the training point nearest to x. Since the prediction closely follows the training data, the model can capture fine-grained variations in $f(\mathbf{x})$, leading to low bias:

  $$\text{Bias}^2\left(\hat{f}(\mathbf{x})\right) \text{ is small.}$$

  However, because the prediction depends on a single (possibly noisy) data point, it is highly sensitive to the particular sample, yielding high variance:

  $$\text{Var}\left(\hat{f}(\mathbf{x})\right) \text{ is large.}$$

  In other words, small changes in the training data (e.g., replacing one point) can significantly alter the prediction.

- Large k: High bias, low variance.

  As k increases, the model predicts by averaging over the k-nearest training outputs:

  $$\hat{f}(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} y_{(i)}.$$

This averaging makes the model less flexible: it smooths out local variations and may fail to capture sharp changes in the true function. As a result, the bias increases:

$$\text{Bias}^2\left(\hat{f}(\mathbf{x})\right) \text{ is large.}$$

On the other hand, since the prediction depends on many points rather than one, the model is more robust to fluctuations in individual data samples. Thus, the variance decreases:

$$\text{Var}\left(\hat{f}(\mathbf{x})\right) \text{ is small.}$$

In this regime, the model may underfit the data, predicting similar outputs for inputs that should be treated differently.

6.5 We can build a minimal example that consists of only two points: $A(2, 4)$, which belongs to class 0, and $B(3, 3.1)$, which belongs to class 1. The query point is $Q(1, 1)$, which belongs to class 0, and we are using $k = 1$ (1-NN) for classification.

- Manhattan Distance: The Manhattan distance between $A(2, 4)$ and $Q(1, 1)$ is:

  $$d_{\text{Manhattan}} = |2 - 1| + |4 - 1| = 1 + 3 = 4$$

  The Manhattan distance between $B(3, 3.1)$ and $Q(1, 1)$ is:

  $$d_{\text{Manhattan}} = |3 - 1| + |3.1 - 1| = 2 + 2.1 = 4.1$$

  Since the Manhattan distance between A and Q is smaller than the distance between B and Q, the query point Q is correctly classified as class 0.

- Euclidean Distance: The Euclidean distance between $A(2, 4)$ and $Q(1, 1)$ is:

  $$d_{\text{Euclidean}} = \sqrt{(2 - 1)^2 + (4 - 1)^2} = \sqrt{1 + 9} = \sqrt{10} = 3.162$$

  The Euclidean distance between $B(3, 3.1)$ and $Q(1, 1)$ is:

  $$d_{\text{Euclidean}} = \sqrt{(3 - 1)^2 + (3.1 - 1)^2} = \sqrt{4 + 4.41} = \sqrt{8.41} = 2.9$$

  Since the Euclidean distance between B and Q is smaller than the distance between A and Q, the query point Q is misclassified as class 1.

6.6 The Minkowski distance between two points $A = (x_1, x_2, \ldots, x_n)$ and $B = (y_1, y_2, \ldots, y_n)$ in n-dimensional space is defined as

$$d_p(A, B) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}.$$

We want to show that as $p \to \infty$, the Minkowski distance converges to the Chebyshev distance, i.e.,

$$\lim_{p \to \infty} d_p(A, B) = \max_{1 \le i \le n} |x_i - y_i|.$$

Let $M = \max_{1 \le i \le n} |x_i - y_i|$, and let $k \in \{1, \ldots, n\}$ be such that $|x_k - y_k| = M$. We begin by bounding the Minkowski distance from above:

$$d_p(A, B) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \le (nM^p)^{1/p} = M \cdot n^{1/p}.$$

Since $\lim_{p \to \infty} n^{1/p} = 1$, we conclude that

$$\lim_{p \to \infty} d_p(A, B) \le M.$$

For the lower bound, we note that

$$\sum_{i=1}^{n} |x_i - y_i|^p \ge |x_k - y_k|^p = M^p,$$

so

$$d_p(A, B) \ge M.$$

Therefore,

$$M \le d_p(A, B) \le M \cdot n^{1/p}.$$

Since both bounds converge to M as $p \to \infty$, the squeeze theorem implies that

$$\lim_{p \to \infty} d_p(A, B) = M = \max_{1 \le i \le n} |x_i - y_i|.$$

6.8 The KD-tree is built by alternating splits between the x-axis and y-axis.

1. First split (root): The root node splits along the x-axis at $x = 5$, the (lower) median of the x-coordinates, dividing the dataset into two groups:
   - Left subtree: Points $(2, 3)$, $(4, 7)$, and $(5, 4)$, where $x \leq 5$.
   - Right subtree: Points $(7, 2)$, $(8, 1)$, and $(9, 6)$, where $x > 5$.
2. Second split (left subtree): The left subtree splits along the y-axis at $y = 4$, dividing the points as follows:
   - Left subtree: Points $(2, 3)$ and $(5, 4)$, where $y \leq 4$.
   - Right subtree: Point $(4, 7)$, where $y > 4$.
3. Third split (left subtree of the left subtree): The points $(2, 3)$ and $(5, 4)$ are separated along the x-axis at $x = 2$:
   - Left child: Point $(2, 3)$.
   - Right child: Point $(5, 4)$.
4. Fourth split (right subtree): The right subtree splits along the y-axis at $y = 2$:
   - Left child: Points $(8, 1)$ and $(7, 2)$, where $y \leq 2$.
   - Right child: Point $(9, 6)$, where $y > 2$.
5. Final split (left subtree of the right subtree): The points $(8, 1)$ and $(7, 2)$ are separated along the x-axis at $x = 7$:
   - Left child: Point $(7, 2)$.
   - Right child: Point $(8, 1)$.

The final KD-tree structure is shown in Figure 6.1.

We would like to find the nearest neighbor to the query point $Q(6, 5)$:

1. Root node check: The root node splits at $x = 5$. Since the x-coordinate of Q is 6, we move to the right subtree.
2. Right subtree split: The right subtree splits at $y = 2$. As the y-coordinate of Q is 5, we move to the right child node of this subtree.
3. Leaf node comparison: This node contains the point $(9, 6)$. We compute the Euclidean distance between $Q(6, 5)$ and $E(9, 6)$:

$$\sqrt{(9-6)^2 + (6-5)^2} = \sqrt{10} = 3.162.$$

4. Backtracking to previous splits:

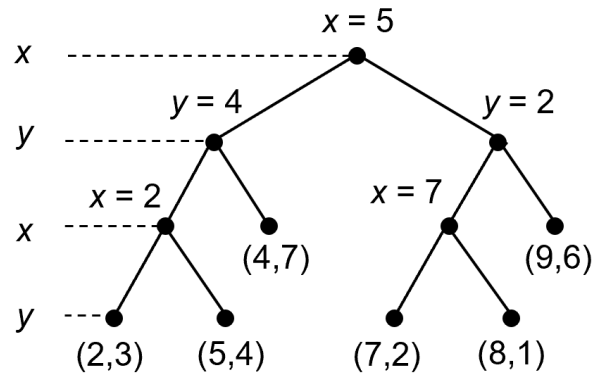


Figure 6.1: A KD-tree built for data points $(2,3)$, $(5,4)$, $(9,6)$, $(4,7)$, $(8,1)$, $(7,2)$.

(a) Backtrack at $y = 2$ split: The vertical distance between Q and the split is $|5-2| = 3$, which is less than 3.162. Therefore, we check the left child node, which splits at $x = 7$. Since the x-coordinate of Q is 6, we move to the left subtree. This node contains the point $(7,2)$. The distance between Q and $(7,2)$ is:

$$\sqrt{(7-6)^2 + (2-5)^2} = \sqrt{10} = 3.162.$$

Since the distance is equal to the current best, we may keep either. Here, we continue with $(9,6)$ as the current best.

(b) Backtrack at the root node's $x = 5$ split: The horizontal distance between Q and the split is $|6-5| = 1$, which is smaller than 3.162. Therefore, we check the left child node, which splits at $y = 4$. Since the y-coordinate of Q is 5, we move to the right subtree. This node contains the point $(4,7)$. The distance between Q and $(4,7)$ is:

$$\sqrt{(4-6)^2 + (7-5)^2} = \sqrt{8} = 2.828.$$

This distance is smaller than 3.162, so $(4,7)$ becomes the closest point so far.

(c) Backtrack at $y = 4$ split: The vertical distance between Q and the split is $|5-4| = 1$, which is less than 2.828. Therefore, we check the left child node, which splits at $x = 2$. Since the x-coordinate of Q is 6, we move to the right subtree. This node contains the point $(5,4)$. The distance between Q and $(5,4)$ is:

$$\sqrt{(5-6)^2 + (4-5)^2} = \sqrt{2} = 1.414.$$

This distance is smaller than 2.828, so $(5, 4)$ becomes the closest point. The search is now complete since there are no more possible backtracks.

Therefore, the nearest neighbor to $Q(6, 5)$ is $(5, 4)$, at a distance of approximately 1.414 units.

6.10 (a) Locality-Sensitive Hashing (LSH) is a randomized algorithmic technique designed to solve the approximate nearest neighbor (ANN) problem in high-dimensional spaces. The core idea is to use a family of hash functions with the property that the probability of collision is higher for points that are close than for points that are far apart. Formally, an LSH family satisfies:

$$\text{If } \|x - y\| \leq R, \text{ then } \Pr[h(x) = h(y)] \geq P_1,$$

$$\text{If } \|x - y\| \geq cR, \text{ then } \Pr[h(x) = h(y)] \leq P_2,$$

for some $c > 1$ and $P_1 > P_2$. LSH enables efficient ANN search by hashing both the dataset and the query into multiple hash tables, then only comparing the query to points in the same buckets across those tables. This significantly reduces the number of distance computations required.

(b) Approximate nearest neighbor algorithms provide a formal guarantee that the returned neighbor lies within a factor c of the true nearest neighbor distance. That is, if p^* is the true nearest neighbor of a query point q, and p' is the result returned by the algorithm, then:

$$\|q - p'\| \leq c \cdot \|q - p^*\|.$$

The main tradeoff is between accuracy and efficiency: smaller approximation factors c yield more accurate results but require more computation and/or memory (e.g., more hash tables), while allowing a larger c leads to faster query time at the cost of less precision.

(c) Exact nearest neighbor search in high-dimensional spaces requires computing distances to all n points in the dataset, leading to query time $\mathcal{O}(dn)$, where d is the dimensionality.

In contrast, the LSH scheme presented by Andoni and Indyk achieves sublinear query time. For c-approximate nearest neighbor, the expected query time is:

$$\mathcal{O}\left(d \cdot n^{\rho + o(1)}\right),$$

where $\rho = 1/c^2$ for the Euclidean norm. The space complexity is close to $\mathcal{O}(dn + n^{1+\rho+o(1)})$. Thus, for any fixed $c > 1$, LSH provides asymptotically faster queries than brute-force exact search.

(d) The curse of dimensionality refers to the exponential growth of the volume of the space with increasing dimension, which renders traditional exact search structures (like KD-trees) ineffective. LSH-based ANN methods mitigate this issue by accepting approximate answers and focusing the search on points that are more likely to be near the query. Instead of trying to partition the space exhaustively, LSH leverages randomness and probability to narrow the candidate set. This probabilistic approach avoids the exponential blow-up of search time with dimension and enables practical nearest neighbor search in very high-dimensional spaces (e.g., 100–1000 dimensions).

Chapter 7

Naive Bayes

7.1 (c)

7.2 (c)

7.3 (a), (d)

7.4 (c)

7.5 (b), (c)

7.6 (a), (b)

7.7 (a), (b), (d), (e)

7.8 (b), (d)

7.9 (a), (b), (e)

7.10 (a), (b), (c), (e)

7.11 The prior probabilities are:

$$P(\text{Recovery} = \text{Yes}) = \frac{5}{10} = 0.5, \quad P(\text{Recovery} = \text{No}) = \frac{5}{10} = 0.5$$

The likelihood of each feature given each class:

$$P(\text{Treatment} = \text{Surgery} \mid \text{Recovery} = \text{Yes}) = \frac{1}{5} = 0.2$$
$$P(\text{Treatment} = \text{Surgery} \mid \text{Recovery} = \text{No}) = \frac{2}{5} = 0.4$$
$$P(\text{Symptoms} = \text{Moderate} \mid \text{Recovery} = \text{Yes}) = \frac{2}{5} = 0.4$$
$$P(\text{Symptoms} = \text{Moderate} \mid \text{Recovery} = \text{No}) = \frac{2}{5} = 0.4$$
$$P(\text{Age} = \text{Young} \mid \text{Recovery} = \text{Yes}) = \frac{3}{5} = 0.6$$
$$P(\text{Age} = \text{Young} \mid \text{Recovery} = \text{No}) = \frac{1}{5} = 0.2$$

We calculate the posterior probabilities using Bayes' theorem:

$$\begin{aligned} P(\text{Recovery} = \text{Yes}|\mathbf{x}) &= \alpha P(\text{Yes}) \cdot P(\text{Surgery}|\text{Yes}) \cdot P(\text{Moderate}|\text{Yes}) \cdot P(\text{Young}|\text{Yes}) \\ &= \alpha \cdot 0.5 \cdot 0.2 \cdot 0.4 \cdot 0.6 \\ &= 0.024\alpha \\ P(\text{Recovery} = \text{No}|\mathbf{x}) &= \alpha P(\text{No}) \cdot P(\text{Surgery}|\text{No}) \cdot P(\text{Moderate}|\text{No}) \cdot P(\text{Young}|\text{No}) \\ &= 0.5 \cdot 0.4 \cdot 0.4 \cdot 0.2 \\ &= 0.016\alpha \end{aligned}$$

Therefore, the probability that the patient will recover within a week is:

$$P(\text{Recovery} = \text{Yes} \mid \mathbf{x}) = \frac{0.024}{0.024 + 0.016} = \frac{0.024}{0.04} = 0.6.$$

7.12 (a) To show that pairwise conditional independence does not imply mutual conditional independence, we provide a counterexample.

Let x_1, x_2, and x_3 be binary random variables taking values in $\{0, 1\}$, and let y also be a binary class label taking values in $\{0, 1\}$. Define the conditional distribution $P(x_1, x_2, x_3 \mid y = 0)$ as follows:

x_1	x_2	x_3	$P(x_1, x_2, x_3 \mid y = 0)$
0	0	0	1/4
0	1	1	1/4
1	0	1	1/4
1	1	0	1/4

To verify that the features are pairwise conditionally independent given $y = 0$, we explicitly compute the joint and marginal distributions for each pair of variables.

From the table, the marginal distributions are:

$$P(x_1 = 0 \mid y = 0) = P(x_2 = 0 \mid y = 0) = P(x_3 = 0 \mid y = 0) = \frac{1}{2}.$$

We now verify pairwise conditional independence for all three pairs:

Pair	Joint event	$P(x_i, x_j \mid y = 0)$	$P(x_i \mid y = 0)P(x_j \mid y = 0)$
(x_1, x_2)	$(0, 0)$	$1/4$	$1/2 \cdot 1/2 = 1/4$
(x_1, x_3)	$(0, 0)$	$1/4$	$1/2 \cdot 1/2 = 1/4$
(x_2, x_3)	$(0, 0)$	$1/4$	$1/2 \cdot 1/2 = 1/4$

Thus, for every pair (x_i, x_j),

$$P(x_i, x_j \mid y = 0) = P(x_i \mid y = 0)P(x_j \mid y = 0),$$

showing that the features are pairwise conditionally independent given the class label.

However, the full joint distribution does not factorize:

$$P(x_1, x_2, x_3 \mid y = 0) \neq P(x_1 \mid y = 0)P(x_2 \mid y = 0)P(x_3 \mid y = 0).$$

For example,

$$P(x_1 = 0 \mid y = 0)P(x_2 = 0 \mid y = 0)P(x_3 = 0 \mid y = 0) = \left(\frac{1}{2}\right)^3 = \frac{1}{8}.$$

But the joint probability is:

$$P(x_1 = 0, x_2 = 0, x_3 = 0 \mid y = 0) = \frac{1}{4} \neq \frac{1}{8}.$$

Hence, the features are not mutually conditionally independent given $y = 0$, despite being pairwise conditionally independent.

(b) Now assume that the joint conditional distribution factorizes:

$$P(\mathbf{x} \mid y) = P(x_1, x_2, \ldots, x_d \mid y) = \prod_{j=1}^{d} P(x_j \mid y).$$

To show that this implies pairwise conditional independence, fix any pair (x_i, x_j) and marginalize out the remaining variables:

$$\begin{aligned} P(x_i, x_j|y) &= \sum_{x_k,\, k\neq i,j} P(x_1, x_2, \ldots, x_d|y) \\ &= \sum_{x_k,\, k\neq i,j} \prod_{j=1}^{d} P(x_j|y) \\ &= P(x_i|y)P(x_j|y) \sum_{x_k,\, k\neq i,j} \prod_{k\neq i,j} P(x_k|y) \\ &= P(x_i|y)P(x_j|y) \prod_{k\neq i,j} \sum_{x_k} P(x_k|y). \end{aligned}$$

Since $\sum_{x_k} P(x_k|y) = 1$ for each k, we obtain:

$$P(x_i, x_j|y) = P(x_i|y)P(x_j|y),$$

which establishes pairwise conditional independence.

7.4 Let $\theta_{cj} = P(x_j = 1|y = c)$ be the probability of feature j being 1 given that the class is c. For a Bernoulli naive Bayes classifier, each feature x_j is binary (0 or 1) and follows a Bernoulli distribution conditioned on the class $y = c$. Let $\Theta = \{\theta_{cj}\}$ represent the set of parameters for these distributions.

Assume that we have n training samples $D = \{(\mathbf{x}_i, y_i\}_{i=1}^{n}$, which are independent and identically distributed (i.i.d.) samples from the distribution $P(\mathbf{x}_i, y_i)$. Then, the likelihood of the entire training dataset given the model parameters Θ is:

$$P(D|\Theta) = \prod_{i=1}^{n} P(\mathbf{x}_i, y_i|\Theta) = \prod_{i=1}^{n} P(\mathbf{x}_i|y_i, \Theta)P(y_i).$$

Since each feature x_j is conditionally independent given the class, we can express $P(\mathbf{x}_i|y_i, \Theta)$ as:

$$P(\mathbf{x}_i|y_i, \Theta) = \prod_{j=1}^{d} P(x_{ij}|y_i, \Theta) = \prod_{j=1}^{d} \theta_{y_i,j}^{x_{ij}} (1 - \theta_{y_i,j})^{1-x_{ij}}.$$

Taking the logarithm of the likelihood function gives us the log-likelihood:

$$\log P(D|\Theta) = \sum_{i=1}^{n} \sum_{j=1}^{d} [x_{ij} \log \theta_{y_i,j} + (1 - x_{ij}) \log(1 - \theta_{y_i,j})] + \sum_{i=1}^{n} \log P(y_i).$$

To find the maximum likelihood estimates, we differentiate the log-likelihood with respect to each parameter θ_{cj} and set the derivative to zero:

$$\frac{\partial}{\partial \theta_{cj}} \log P(D; \Theta) = \sum_{i=1}^{n} \frac{x_{ij} \mathbb{1}(y_i = c)}{\theta_{cj}} - \sum_{i=1}^{n} \frac{(1 - x_{ij}) \mathbb{1}(y_i = c)}{1 - \theta_{cj}} = 0,$$

where $\mathbb{1}(y_i = c)$ is an indicator function that is 1 if $y_i = c$ and 0 otherwise.

Rearranging the equation gives:

$$\frac{1}{\theta_{cj}} \sum_{i=1}^{n} x_{ij} \mathbb{1}(y_i = c) = \frac{1}{1 - \theta_{cj}} \sum_{i=1}^{n} (1 - x_{ij}) \mathbb{1}(y_i = c).$$

Cross-multiplying to isolate θ_{cj}:

$$\left(\sum_{i=1}^{n} x_{ij} \mathbb{1}(y_i = c) \right) (1 - \theta_{cj}) = \left(\sum_{i=1}^{n} (1 - x_{ij}) \mathbb{1}(y_i = c) \right) \theta_{cj}.$$

Expanding both sides:

$$\sum_{i=1}^{n} x_{ij} \mathbb{1}(y^{(i)} = c) - \theta_{cj} \sum_{i=1}^{n} x_{ij} \mathbb{1}(y_i = c) = \theta_{cj} \sum_{i=1}^{n} \mathbb{1}(y_i = c) - \theta_{cj} \sum_{i=1}^{n} x_{ij} \mathbb{1}(y_i = c).$$

The terms $\theta_{cj} \sum_{i=1}^{n} x_{ij} \mathbb{1}(y_i = c)$ cancel out on both sides, leading to

$$\sum_{i=1}^{n} x_{ij} \mathbb{1}(y_i = c) = \theta_{cj} \sum_{i=1}^{n} \mathbb{1}(y_i = c).$$

Finally, solving for θ_{cj} gives:

$$\theta_{cj} = \frac{\sum_{i=1}^{n} x_{ij} \mathbb{1}(y_i = c)}{\sum_{i=1}^{n} \mathbb{1}(y_i = c)}.$$

This result shows that the MLE estimate for $\theta_{cj} = P(x_j = 1 \mid y = c)$, is equal to the frequency of feature j being 1 among the samples of class c.

7.7 (a) The multinomial naive Bayes classifier calculates the posterior probability of each class as:

$$P(y = \text{spam} \mid w_1 = \text{send}, w_2 = \text{money}) > P(y = \text{ham} \mid w_1 = \text{send}, w_2 = \text{money}).$$

Using Bayes' theorem and the naive Bayes assumption, this becomes:

$$P(w_1 = \text{send} \mid Y = \text{spam}) \cdot P(w_2 = \text{money} \mid Y = \text{spam}) \cdot P(Y = \text{spam}) > P(w_1 = \text{send} \mid Y = \text{ham}) \cdot P(w_2 = \text{money} \mid Y = \text{ham}) \cdot P(Y = \text{ham}).$$

Substituting the word probabilities from Table 7.10:

$$\frac{1}{8} \cdot \frac{1}{6} \cdot P(Y = \text{spam}) > \frac{1}{12} \cdot \frac{1}{8} \cdot (1 - P(Y = \text{spam})).$$

Next, simplifying both sides gives:

$$\begin{aligned} \frac{1}{48} \cdot P(y = \text{spam}) &> \frac{1}{96} \cdot (1 - P(y = \text{spam})) \\ 96 \cdot P(y = \text{spam}) &> 48 \cdot (1 - P(y = \text{spam})) \\ 96 \cdot P(y = \text{spam}) &> 48 - 48 \cdot P(y = \text{spam}) \\ 144 \cdot P(y = \text{spam}) &> 48 \\ P(y = \text{spam}) &> \frac{48}{144} = \frac{1}{3}. \end{aligned}$$

Thus, the classifier will predict that the email is spam if:

$$P(y = \text{spam}) > \frac{1}{3}.$$

(b) i. There are 3 spam emails, and the total number of words in the spam emails is $7 + 13 + 6 = 26$. The word `claim` appears three times in the spam emails. Thus, the probability is:

$$P(\texttt{claim} \mid \text{spam}) = \frac{3}{26}$$

ii. The word `account` does not appear in the ham emails. Therefore, the probability is:

$$P(\texttt{account} \mid \text{ham}) = 0$$

(c) To compute the probabilities with Laplace smoothing, we use the formula:

$$P(w|y) = \frac{c(w,y) + \alpha}{N_y + \alpha V},$$

where $c(w, y)$ is the count of word w in emails of class y, N_y is the total number of words in emails of class y, α is the smoothing parameter, and V is the vocabulary size (number of distinct words).

i. $P(\texttt{claim} \mid \text{spam})$:

$$\begin{aligned} c(\texttt{claim}, \text{spam}) &= 3, \\ N_{\text{spam}} &= 26, \\ P(\texttt{claim} \mid \text{spam}) &= \frac{3 + \alpha}{26 + \alpha V}. \end{aligned}$$

ii. $P(\texttt{account} \mid \text{ham})$:

$$\begin{aligned} c(\texttt{account}, \text{ham}) &= 0, \\ N_{\text{ham}} &= 11, \\ P(\texttt{account} \mid \text{ham}) &= \frac{\alpha}{11 + \alpha V}. \end{aligned}$$

(d) i. In the new model, we need to estimate the probability of each word given both the label and the preceding word. For each possible preceding word w_{i-1} and label c, we need to calculate V conditional probabilities. Since there are V possible preceding words and two classes, the total number of conditional probabilities required is $2V^2$.

ii. Advantages of the new model:

- Modeling word dependencies can capture the context and sequential relationships between words, potentially leading to more accurate classification, especially in cases where the order of words is important.
- This approach can help to better differentiate between spam and ham by utilizing word sequences that are more common in one class than the other.

Disadvantages of the new model:

- The new model requires estimating $2V^2$ conditional probabilities, which significantly increases the number of parameters compared to the original $2V$. This can lead to overfitting, especially if the training dataset is small.

- Increased computational complexity in training and prediction due to the larger number of parameters.
- The model may struggle with rare word combinations or sequences that are not well-represented in the training data, potentially leading to inaccurate estimates for those cases.

7.9 We are given a new student with:

$$\text{Study Time} = 6.0, \quad \text{Participation} = \text{Moderate}, \quad \text{Previous Test} = 70.$$

From the dataset, the prior probabilities are:

$$P(\text{Pass}) = \frac{5}{10} = 0.5, \quad P(\text{Fail}) = \frac{5}{10} = 0.5.$$

We compute the likelihoods for each feature given the class.

- Study Time: We first compute the mean and standard deviation for each class.

$$\mu_{\text{Pass}} = \frac{10.2 + 7.8 + 9.0 + 12.3 + 8.2}{5} = 9.5,$$
$$\sigma_{\text{Pass}} = \sqrt{\frac{(10.2-9.5)^2 + (7.8-9.5)^2 + (9.0-9.5)^2 + (12.3-9.5)^2 + (8.2-9.5)^2}{5}} \approx 1.622,$$
$$\mu_{\text{Fail}} = \frac{3.5 + 2.0 + 4.5 + 5.5 + 1.5}{5} = 3.4,$$
$$\sigma_{\text{Fail}} = \sqrt{\frac{(3.5-3.4)^2 + (2.0-3.4)^2 + (4.5-3.4)^2 + (5.5-3.4)^2 + (1.5-3.4)^2}{5}} \approx 1.497.$$

Now we compute the likelihoods of the study time 6 using the normal PDF:

$$P(\text{Study Time} = 6.0 \mid \text{Pass}) = \frac{1}{\sqrt{2\pi} \cdot 1.622} \exp\left(-\frac{(6.0-9.5)^2}{2 \cdot 1.622^2}\right) \approx 0.024,$$
$$P(\text{Study Time} = 6.0 \mid \text{Fail}) = \frac{1}{\sqrt{2\pi} \cdot 1.497} \exp\left(-\frac{(6.0-3.4)^2}{2 \cdot 1.497^2}\right) \approx 0.059.$$

- Participation: The probabilities for Participation are calculated based on the training data:

$$P(\text{Participation} = \text{Moderate} \mid \text{Pass}) = \frac{2}{5} = 0.4,$$
$$P(\text{Participation} = \text{Moderate} \mid \text{Fail}) = \frac{1}{5} = 0.2.$$

- Previous Test: We first compute the mean and standard deviation for each class.

$$\mu_{\text{Pass}} = \frac{85+60+70+90+80}{5} = 77,$$
$$\sigma_{\text{Pass}} = \sqrt{\frac{(85-77)^2+(60-77)^2+(70-77)^2+(90-77)^2+(80-77)^2}{5}} \approx 10.77,$$
$$\mu_{\text{Fail}} = \frac{40+35+55+45+50}{5} = 45,$$
$$\sigma_{\text{Fail}} = \sqrt{\frac{(40-45)^2+(35-45)^2+(55-45)^2+(45-45)^2+(50-45)^2}{5}} \approx 7.071.$$

Now we compute the likelihoods of the test score 70 using the normal PDF:

$$P(\text{Previous Test} = 70 \mid \text{Pass}) = \frac{1}{\sqrt{2\pi}\cdot 10.77}\exp\left(-\frac{(70-77)^2}{2\cdot 10.77^2}\right) = 0.03,$$
$$P(\text{Previous Test} = 70 \mid \text{Fail}) = \frac{1}{\sqrt{2\pi}\cdot 7.071}\exp\left(-\frac{(70-45)^2}{2\cdot 7.071^2}\right) = 0.000109.$$

Now, we compute the posterior probabilities for Pass and Fail using the naive Bayes formula:

$$P(\text{Pass}|\mathbf{x}) \propto 0.5\cdot 0.024\cdot 0.4\cdot 0.03 = 0.000144,$$
$$P(\text{Fail}|\mathbf{x}) \propto 0.5\cdot 0.059\cdot 0.2\cdot 0.000109 = 0.000000642.$$

We normalize the posterior values:

$$Z = 0.000144 + 0.000000642 = 0.000144642$$
$$P(\text{Pass}|\mathbf{x}) = \frac{0.000144}{0.000144642} = 0.996,$$
$$P(\text{Fail}|\mathbf{x}) = \frac{0.000000642}{0.000144642} = 0.004.$$

Since $P(\text{Pass}|\mathbf{x}) > P(\text{Fail}|\mathbf{x})$, the model predicts that the student will pass the final exam.

7.13 To compute the probability that a vehicle requires major maintenance given that its age is less than 5 years ($A = 0$) and its mileage is categorized as medium

$(M = 1)$, we use the law of total probability and the independence assumptions derived from the network's structure:

$$\begin{aligned} P(MM = 1 \mid A = 0, M = 1) &= \sum_{MH=0}^{2} P(MM = 1 \mid A = 0, M = 1, MH) \\ &= \sum_{MH=0}^{2} P(MM = 1 \mid M = 1, MH) \cdot P(MH \mid A = 0). \end{aligned}$$

We first compute each term for $P(MM = 1 \mid M = 1, MH)$:

$$\begin{aligned} P(MM = 1 \mid M = 1, MH = 0) &= 0.3, \\ P(MM = 1 \mid M = 1, MH = 1) &= 0.6, \\ P(MM = 1 \mid M = 1, MH = 2) &= 0.8. \end{aligned}$$

Next, we compute $P(MH \mid A = 0)$ for each value of MH:

$$\begin{aligned} P(MH = 0 \mid A = 0) &= 0.1, \\ P(MH = 1 \mid A = 0) &= 0.2, \\ P(MH = 2 \mid A = 0) &= 1 - 0.1 - 0.2 = 0.7. \end{aligned}$$

Finally, we apply the law of total probability:

$$\begin{aligned} P(MM = 1 \mid A = 0, M = 1) &= P(MM = 1 \mid M = 1, MH = 0) \cdot P(MH = 0 \mid A = 0) \\ &\quad + P(MM = 1 \mid M = 1, MH = 1) \cdot P(MH = 1 \mid A = 0) \\ &\quad + P(MM = 1 \mid M = 1, MH = 2) \cdot P(MH = 2 \mid A = 0) \\ &= (0.3 \cdot 0.1) + (0.6 \cdot 0.2) + (0.8 \cdot 0.7) \\ &= 0.03 + 0.12 + 0.56 \\ &= 0.71. \end{aligned}$$

Thus, the probability that a vehicle requires major maintenance given that its age is less than 5 years and its mileage is medium is 0.71.

Chapter 8

Decision Trees

8.1 (c)

8.2 (b)

8.3 (c)

8.4 (a)

8.5 (c), (e)

8.6 (a), (d)

8.7 (a), (d)

8.8 (c)

8.9 (d)

8.10 (b), (d)

8.11 (a) The entropy for the IsEdible attribute:

$$H(\text{IsEdible}) = I(4,5) = -\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9} = 0.991$$

(b) We start from the root node. To find the attribute with the highest information gain, we calculate the information gain for each attribute:

- Information gain for IsRound:
  - IsRound = 1: Contains samples A, B, D, G, H with 3 Yes and 2 No.

  - IsRound = 0: Contains samples C, E, F, I with 1 Yes and 3 No.

$$\text{IG(IsRound)} = 0.991 - \frac{5}{9}I(3,2) - \frac{4}{9}I(1,3) = 0.091$$

- Information gain for IsColorful:
  - IsColorful = 1: Contains samples A, B, C, E, I with 3 Yes and 2 No.
  - IsColorful = 0: Contains samples D, F, G, H with 1 Yes and 3 No.

$$\text{IG(IsColorful)} = 0.991 - \frac{5}{9}I(3,2) - \frac{4}{9}I(1,3) = 0.091$$

- Information gain for HasSpikes:
  - HasSpikes = 1: Contains samples C, D, F, H, I with 1 Yes and 4 No.
  - HasSpikes = 0: Contains samples A, B, E, G with 3 Yes and 1 No.

$$\text{IG(HasSpikes)} = 0.991 - \frac{5}{9}I(1,4) - \frac{4}{9}I(3,1) = 0.229$$

- Information gain for IsSweet:
  - IsSweet = 1: Contains samples A, C, E, F, H, I with 2 Yes and 4 No.
  - IsSweet = 0: Contains samples B, D, G with 2 Yes and 1 No.

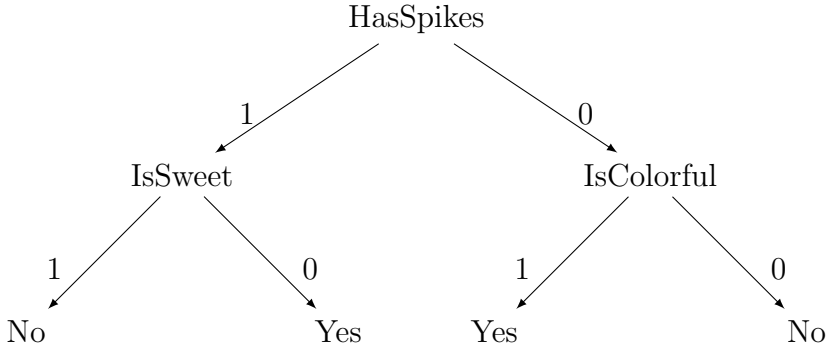
$$\text{IG(IsSweet)} = 0.991 - \frac{6}{9}I(2,4) - \frac{3}{9}I(2,1) = 0.073$$

The attribute with the highest information gain is HasSpikes (0.229), so it becomes the root of the decision tree.

(c) Constructing the decision tree:

- If HasSpikes = 1: In this subset, samples C, F, H, I are labeled No and sample D is labeled Yes. Sample D is the only one that has IsSweet = 0. Thus, fruits that have spikes and are sweet will be classified as not edible and fruits that have spikes and are not sweet will be classified as edible.
- If HasSpikes = 0: In this subset, samples A, B, E are labeled Yes and sample G is labeled No. Sample G is the only one that has IsColorful = 0. Thus, fruits that do not have spikes and are colorful will be classified as edible and fruits that do not have spikes and are not colorful will be classified as not edible.

The final decision tree is:



(d) Based on this decision tree, the classification of the new fruits is:

- Fruit J: HasSpikes = 0, IsColorful = 1. Thus, it will be classified as Edible = Yes.
- Fruit K: HasSpikes = 1, IsSweet = 1. Thus, it will be classified as Edible = No.
- Fruit L: HasSpikes = 1, IsSweet = 0. Thus, it will be classified as Edible = Yes.

8.13 Assume that we have k classes in the classification problem. Let $Q = (q_1, \ldots, q_k)$ represent the probability distribution of the classes at the parent node. The entropy of the parent node is defined as:

$$H(\text{Parent}) = -\sum_{j=1}^{k} q_j \log_2 q_j.$$

Now, suppose the dataset is split into m child nodes. Let C_i denote the i-th child node after the split, and let $P_i = (p_{i1}, \ldots, p_{ik})$ represent the probability distribution of the classes within this child node. The entropy of child node C_i is given by:

$$H(C_i) = -\sum_{j=1}^{k} p_{ij} \log_2 p_{ij}.$$

Let w_i denote the proportion of samples that are directed to C_i, with the constraint that $\sum_{i=1}^{m} w_i = 1$. The weighted average of the entropies of the child nodes is then:

$$H(\text{Children}) = -\sum_{i=1}^{m} w_i H(C_i) = -\sum_{i=1}^{m} w_i \sum_{j=1}^{k} p_{ij} \log_2 p_{ij}.$$

The information gain is the difference between the entropy of the parent node and the weighted average of the entropies of the child nodes:

$$
\begin{aligned}
\mathrm{IG} &= H(\text{Parent}) - H(\text{Children}) \\
&= -\sum_{j=1}^{k} q_j \log_2 q_j + \sum_{i=1}^{m} w_i \sum_{j=1}^{k} p_{ij} \log_2 p_{ij} \\
&= \sum_{i=1}^{m} w_i \sum_{j=1}^{k} p_{ij} \log_2 p_{ij} - \sum_{j=1}^{k} \log_2 q_j \left(\sum_{i=1}^{m} w_i p_{ij} \right) \\
&\qquad \text{(since } q_j = \sum_{i=1}^{m} w_i p_{ij} \text{ represents the total probability of class } j\text{)} \\
&= \sum_{\substack{1 \le i \le m \\ 1 \le j \le k}} w_i p_{ij} \log_2 p_{ij} - \sum_{\substack{1 \le i \le m \\ 1 \le j \le k}} w_i p_{ij} \log_2 q_j \\
&= \sum_{i,j} w_i p_{ij} \log_2 \left(\frac{p_{ij}}{q_j} \right) \\
&= -\sum_{i,j} w_i p_{ij} \log_2 \left(\frac{q_j}{p_{ij}} \right) \\
&\geq -\log_2 \left(\sum_{i,j} w_i p_{ij} \frac{q_j}{p_{ij}} \right) \\
&\qquad \text{(since } -\log_2 \text{ is convex and } \sum_{i,j} w_i p_{ij} = 1\text{, we can use Jensen's inequality)} \\
&= -\log_2 \left(\sum_{i,j} w_i q_j \right) \\
&= -\log_2 \left(\sum_{i=1}^{m} w_i \left(\sum_{j=1}^{k} q_j \right) \right) \\
&= -\log_2 \left(\sum_{i=1}^{m} w_i \right) \quad \text{(since } Q \text{ is a probability distribution)} \\
&= -\log_2(1) \quad \text{(since } \sum_{i=1}^{m} w_i = 1\text{)} \\
&= 0.
\end{aligned}
$$

This implies that any split in the decision tree yields a non-negative information gain, meaning we cannot make negative progress in classifying the training data.

8.15 (a) The misclassification rate considers only the proportion of samples in the majority class, ignoring the distribution of other class probabilities. As a result, it is less sensitive to changes in the underlying class distribution than impurity measures like entropy and Gini index, which take into account the entire class distribution.

(b) Consider a binary classification problem with two classes: positive and negative. Suppose we have a parent node P with 10 samples: $p = 7$ positive samples and $n = 3$ negative samples. The impurity calculations for the parent node are as follows:

$$G(P) = 1 - \left(\frac{p}{p+n}\right)^2 - \left(\frac{n}{p+n}\right)^2 = 1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2 = 0.42,$$
$$M(P) = 1 - \frac{p}{p+n} = 1 - \frac{7}{10} = 0.3.$$

Now suppose we split this node based on some attribute A, which results in two child nodes, v_1 and v_2, with the following distributions:

- Node v_1 has $p_1 = 3$ positive samples and $n_1 = 0$ negative samples.

$$G(v_1) = 1 - \left(\frac{p_1}{p_1+n_1}\right)^2 - \left(\frac{n_1}{p_1+n_1}\right)^2 = 1 - \left(\frac{3}{3}\right)^2 - \left(\frac{0}{3}\right)^2 = 0,$$
$$M(v_1) = 1 - \frac{p_1}{p_1+n_1} = 1 - 1 = 0.$$

- Node v_2 has $p_2 = 4$ positive samples and $n_2 = 3$ negative samples.

$$G(v_2) = 1 - \left(\frac{p_2}{p_2+n_2}\right)^2 - \left(\frac{n_2}{p_2+n_2}\right)^2 = 1 - \left(\frac{4}{7}\right)^2 - \left(\frac{3}{7}\right)^2 = 0.49,$$
$$M(v_2) = 1 - \frac{p_2}{p_2+n_2} = 1 - \frac{4}{7} = \frac{3}{7}.$$

The reduction in Gini index after the split is:

$$\begin{aligned}\Delta_G(A) &= G(P) - \left(\frac{p_1+n_1}{p+n} \cdot G(v_1) + \frac{p_2+n_2}{p+n} \cdot G(v_2)\right) \\ &= 0.42 - \frac{7}{10} \cdot 0.49 \\ &= 0.42 - 0.343 = 0.077.\end{aligned}$$

The change in misclassification rate is:

$$\begin{aligned}\Delta_M(A) &= M(P) - \left(\frac{p_1+n_1}{p+n}\cdot M(v_1) + \frac{p_2+n_2}{p+n}\cdot M(v_2)\right)\\ &= 0.3 - \frac{7}{10}\cdot\frac{3}{7}\\ &= 0.3 - 0.3 = 0.\end{aligned}$$

This example shows that the Gini index improves (indicating a purer split), while the misclassification rate gain is zero, suggesting no improvement. Thus, Gini index captures the benefit of the split more effectively than misclassification rate. This insensitivity makes misclassification rate a poor criterion during tree growth, although it is sometimes used for pruning.

8.19 (a) The first subtree T_t that is pruned is the one that minimizes the function

$$g(t) = \frac{R(t) - R(T_t)}{|\tilde{T}_t| - 1}.$$

where $R(t)$ is the error if T_t is pruned (i.e., t becomes a leaf), and $|\tilde{T}_t|$ is the number of leaves in the subtree T_t.

Minimizing $g(t)$ is equivalent to minimizing the ratio

$$\frac{\Delta R(T_t)}{\Delta|\tilde{T}_t|},$$

where $\Delta R(T_t) = R(t) - R(T_t)$ is the increase in training error, and $\Delta|\tilde{T}_t| = |\tilde{T}_t| - 1$ is the reduction in the number of leaves.

(b) Let $\alpha_1 < \alpha_2$ be two complexity parameters, and let $T(\alpha_1)$ and $T(\alpha_2)$ be their respective optimal pruned trees. Assume, for contradiction, that $T(\alpha_2)$ is not a subtree of $T(\alpha_1)$. Then there exists a subtree T_t that is pruned in $T(\alpha_1)$ but retained in $T(\alpha_2)$.

For any given α, the pruning algorithm removes the subtree that minimally increases the cost-complexity function:

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|,$$

where $R(T)$ is the training error and $|\tilde{T}|$ is the number of leaves in T.

Since T_t was pruned in $T(\alpha_1)$, it was the subtree with the smallest increase in $R_{\alpha_1}(T)$. At $\alpha_2 > \alpha_1$, the algorithm would again prioritize pruning T_t

because it would still have the minimal increase in cost-complexity. Thus, T_t should also be pruned at α_2, contradicting our assumption.

Therefore, $T(\alpha_2)$ must be a subtree of $T(\alpha_1)$.

(c) Let $\alpha_1 < \alpha_2$, and let $T(\alpha_1)$ and $T(\alpha_2)$ be the optimal pruned trees for complexity parameters α_1 and α_2, respectively. From the previous proof, we know that the sequence of pruned trees $T(\alpha)$ is nested, so $T(\alpha_2)$ is a subtree of $T(\alpha_1)$.

Since $T(\alpha_2)$ is obtained by further pruning $T(\alpha_1)$, it has fewer or the same number of leaves, which reduces the model's capacity to fit the data. Therefore, the training error can only increase or remain the same:

$$R(T(\alpha_1)) \leq R(T(\alpha_2)).$$

Thus, the training error of the pruned tree $T(\alpha)$ is monotonically non-decreasing as α increases.

Chapter 9

Ensemble Methods

9.1 (b), (c)

9.2 (c), (e)

9.3 (a), (c), (d)

9.4 (d), (e)

9.5 (b), (c)

9.6 (a), (c), (d)

9.7 (c), (d)

9.8 (c), (d)

9.9 (d), (e)

9.10 (a), (b), (c)

9.11 (a) For an ensemble with $n = 15$ base classifiers, each with an error rate of $\epsilon = 0.3$, the ensemble error rate is given by:

$$\epsilon_{\text{ensemble}} = \sum_{k=\lceil 15/2 \rceil}^{15} \binom{15}{k} (0.3)^k (0.7)^{15-k} \approx 0.05$$

Therefore, the ensemble error rate is significantly lower than the individual error rate of 0.3.

(b) To prove this, we rely on the properties of the binomial distribution Binomial(n, ϵ). The mean of this distribution is $\mu = n\epsilon$, and when $\epsilon < 0.5$, we have $\mu < n/2$. This implies that the probability mass of the distribution is more concentrated to the left of $n/2$, resulting in a right-skewed distribution with more weight on the side where fewer errors occur.

Since the ensemble makes an incorrect prediction only if more than half of its base classifiers are wrong, the ensemble's error corresponds to the upper tail probability of the binomial distribution. Because the distribution is skewed toward fewer errors when $\epsilon < 0.5$, this upper tail probability $\epsilon_{\text{ensemble}}$ is less than the individual error rate ϵ.

(c) By the central limit theorem, the binomial distribution can be approximated by a normal distribution for large n:

$$X \sim \text{Binomial}\,(n, \epsilon) \approx \mathcal{N}\left(n\epsilon, n\epsilon(1-\epsilon)\right),$$

where X is the random variable representing the number of errors made by the ensemble. We then have:

$$\epsilon_{\text{ensemble}} \approx P\left(X \geq \frac{n}{2}\right) = P\left(\frac{X-\mu}{\sigma} \geq \frac{n/2-\mu}{\sigma}\right),$$

where $\mu = n\epsilon$ and $\sigma = \sqrt{n\epsilon(1-\epsilon)}$.

Thus, we can express the ensemble error rate as:

$$\epsilon_{\text{ensemble}} \approx P\left(Z \geq \frac{\frac{n}{2} - n\epsilon}{\sqrt{n\epsilon(1-\epsilon)}}\right),$$

where Z follows a standard normal distribution $\mathcal{N}(0, 1)$.

(d) We can further simplify:

$$\epsilon_{\text{ensemble}} \approx P\left(Z \geq \sqrt{n}\frac{\frac{1}{2} - \epsilon}{\sqrt{\epsilon(1-\epsilon)}}\right).$$

Since $\epsilon < 0.5$, $\frac{1}{2} - \epsilon > 0$, making the argument inside $P(Z \geq \ldots)$ positive. As n increases, $\sqrt{n}$ increases, causing the argument inside $P(Z \geq \ldots)$ to grow larger. Since the probability $P(Z \geq \ldots)$ decreases as the argument increases, $\epsilon_{\text{ensemble}}$ is monotonically decreasing with n.

Furthermore, as $n \to \infty$, $\sqrt{n} \to \infty$, making the argument grow without bound. Consequently, $\epsilon_{\text{ensemble}} \to 0$.

9.14 Assume we have m independent base learners, each with a prediction $\hat{f}_i(x)$ for $i = 1, 2, \ldots, m$. The ensemble prediction $\hat{f}(x)$ is given by the average of the base learners:

$$\hat{f}(x) = \frac{1}{m} \sum_{i=1}^{m} \hat{f}_i(x).$$

The bias of the ensemble prediction $\hat{f}(x)$ is defined as:

$$\text{Bias}(\hat{f}(x)) = f(x) - \mathbb{E}[\hat{f}(x)].$$

Since $\hat{f}(x)$ is the average of the base learners' predictions, we have:

$$\mathbb{E}[\hat{f}(x)] = \mathbb{E}\left[\frac{1}{m} \sum_{i=1}^{m} \hat{f}_i(x)\right] = \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}[\hat{f}_i(x)].$$

If each base learner has the same bias, defined as $f(x) - \mathbb{E}[\hat{f}_i(x)]$, then we can write $\mathbb{E}[\hat{f}_i(x)] = c$ for some constant c. Therefore:

$$\mathbb{E}[\hat{f}(x)] = \frac{1}{m} \sum_{i=1}^{m} c = \frac{1}{m} \cdot m \cdot c = c.$$

We can then express the bias of the ensemble as:

$$\text{Bias}(\hat{f}(x)) = f(x) - \mathbb{E}[\hat{f}(x)] = f(x) - c$$

Since the bias of each individual base learner is:

$$\text{Bias}(\hat{f}_i(x)) = f(x) - c,$$

it follows that:

$$\text{Bias}(\hat{f}(x)) = \text{Bias}(\hat{f}_i(x)).$$

Thus, the bias of the ensemble prediction is the same as the bias of any individual base learner. Therefore, bagging does not increase the bias.

The variance of the ensemble prediction $\hat{f}(x)$ is given by:

$$\text{Var}(\hat{f}(x)) = \text{Var}\left(\frac{1}{m} \sum_{i=1}^{m} \hat{f}_i(x)\right).$$

Since the base learners are independent, we can write:

$$\text{Var}\left(\frac{1}{m}\sum_{i=1}^{m}\hat{f}_i(x)\right) = \frac{1}{m^2}\sum_{i=1}^{m}\text{Var}(\hat{f}_i(x)).$$

Let $\text{Var}(\hat{f}_i(x)) = \xi^2$ for each base learner. Then:

$$\text{Var}(\hat{f}(x)) = \frac{1}{m^2}\sum_{i=1}^{m}\xi^2 = \frac{\xi^2}{m}.$$

This result shows that the variance of the ensemble prediction decreases as m increases.

Combining both results, we have shown that bagging reduces the variance of the model without increasing the bias, which leads to an overall decrease in the expected squared error:

$$\mathbb{E}\left[(f(x) - \hat{f}(x))^2\right] = \text{Bias}^2(\hat{f}(x)) + \frac{\xi^2}{m} + \sigma^2.$$

9.16 (a) If $H(x_i) \neq y_i$, it follows that $y_i f(x_i) \leq 0$ because the sign of $f(x_i)$ disagrees with y_i. Consequently, $\exp(-f(x_i)y_i) \geq 1$. When $H(x_i) = y_i$, we have $\exp(-f(x_i)y_i) \geq 0$ by the definition of exponent.

Combining these observations, we have:

$$\mathbb{1}(H(x_i) \neq y_i) \leq \exp(-f(x_i)y_i) \quad \text{for all } i.$$

Summing over all the examples gives:

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}(H(x_i) \neq y_i) \leq \frac{1}{n}\sum_{i=1}^{n}\exp(-f(x_i)y_i).$$

(b) Expanding $D_{t+1}(i)$ recursively, we get:

$$\begin{aligned}
D_{t+1}(i) &= \frac{D_t(i)\exp(-\alpha_t h_t(x_i)y_i)}{Z_t} \\
&= \frac{D_{t-1}(i)\exp(-\alpha_t h_t(x_i)y_i)\exp(-\alpha_{t-1}h_{t-1}(x_i)y_i)}{Z_t Z_{t-1}} \\
&= \dots \\
&= \frac{D_1(i)\prod_{t=1}^{T}\exp(-\alpha_t h_t(x_i)y_i)}{\prod_{t=1}^{T} Z_t} \\
&= \frac{D_1(i)\exp\left(\sum_{t=1}^{T} -\alpha_t h_t(x_i)y_i\right)}{\prod_{t=1}^{T} Z_t} \\
&= \frac{D_1(i)\exp(-f(x_i)y_i)}{\prod_{t=1}^{T} Z_t} \\
&= \frac{\exp(-f(x_i)y_i)}{n\prod_{t=1}^{T} Z_t}.
\end{aligned}$$

Summing both sides over i and using the fact that $D_{t+1}(i)$ sums to 1 gives:

$$1 = \sum_{i=1}^{n} \frac{\exp(-f(x_i)y_i)}{n\prod_{t=1}^{T} Z_t}.$$

Thus, we have:

$$\prod_{t=1}^{T} Z_t = \frac{1}{n}\sum_{i=1}^{n}\exp(-f(x_i)y_i).$$

(c) By separating the sums over correctly classified and misclassified examples,

we have:

$$
\begin{aligned}
Z_t &= \sum_{i=1}^{n} D_t(i) \exp(-\alpha_t y_i h_t(x_i)) \\
&= \sum_{i:h_t(x_i)=y_i} D_t(i) \exp(-\alpha_t) + \sum_{i:h_t(x_i)\neq y_i} D_t(i) \exp(\alpha_t) \\
&= \sum_{i=1}^{n} D_t(i) \exp(-\alpha_t) \mathbb{1}(h_t(x_i) = y_i) + \sum_{i=1}^{n} D_t(i) \exp(\alpha_t) \mathbb{1}(h_t(x_i) \neq y_i) \\
&= \sum_{i=1}^{n} D_t(i) \exp(-\alpha_t)(1 - \mathbb{1}(h_t(x_i) \neq y_i)) + \sum_{i=1}^{n} D_t(i) \exp(\alpha_t) \mathbb{1}(h_t(x_i) \neq y_i) \\
&= \left(\sum_{i=1}^{n} D_t(i) - \sum_{i=1}^{n} D_t(i) \mathbb{1}(h_t(x_i) \neq y_i) \right) \exp(-\alpha_t) + \left(\sum_{i=1}^{n} D_t(i) \mathbb{1}(h_t(x_i) \neq y_i) \right) \exp(\alpha_t) \\
&= (1 - \epsilon_t) \exp(-\alpha_t) + \epsilon_t \exp(\alpha_t)
\end{aligned}
$$

Taking the derivative and setting it to zero:

$$
\frac{\partial Z_t}{\partial \alpha_t} = -(1 - \epsilon_t) \exp(-\alpha_t) + \epsilon_t \exp(\alpha_t) = 0.
$$

Solving for α_t:

$$
\begin{aligned}
\epsilon_t \exp(\alpha_t) &= \frac{1 - \epsilon_t}{\exp(\alpha_t)} \\
\left(\exp(\alpha_t)\right)^2 &= \frac{1 - \epsilon_t}{\epsilon_t} \\
\exp(\alpha_t) &= \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \\
\alpha_t^* &= \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}
\end{aligned}
$$

(d) We plug α_t^* into Z_t:

$$\begin{aligned}
Z_t &= (1-\epsilon_t)\exp\left(-\frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}\right) + \epsilon_t \exp\left(\frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}\right) \\
&= (1-\epsilon_t)\left(\exp\left(\log\frac{1-\epsilon_t}{\epsilon_t}\right)\right)^{-\frac{1}{2}} + \epsilon_t\left(\exp\left(\log\frac{1-\epsilon_t}{\epsilon_t}\right)\right)^{\frac{1}{2}} \\
&= (1-\epsilon_t)\left(\frac{1-\epsilon_t}{\epsilon_t}\right)^{-\frac{1}{2}} + \epsilon_t\left(\frac{1-\epsilon_t}{\epsilon_t}\right)^{\frac{1}{2}} \\
&= \frac{(1-\epsilon_t)\sqrt{\epsilon_t}}{\sqrt{1-\epsilon_t}} + \frac{\epsilon_t\sqrt{1-\epsilon_t}}{\sqrt{\epsilon_t}} \\
&= \sqrt{1-\epsilon_t}\sqrt{\epsilon_t} + \sqrt{\epsilon_t}\sqrt{1-\epsilon_t} \\
&= 2\sqrt{\epsilon_t(1-\epsilon_t)}.
\end{aligned}$$

(e) Let $\epsilon_t = \frac{1}{2} - \gamma_t$. Then,

$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)} = 2\sqrt{\left(\frac{1}{2}-\gamma_t\right)\left(\frac{1}{2}+\gamma_t\right)} = 2\sqrt{\left(\frac{1}{2}\right)^2 - \gamma_t^2} = \sqrt{1-4\gamma_t^2}.$$

Taking the natural logarithm on both sides, we have:

$$\log Z_t = \frac{1}{2}\log(1-4\gamma_t^2).$$

Using the inequality $\log(1-x) \leq -x$ for $0 < x \leq 1$, we get:

$$\log(1-4\gamma_t^2) \leq -4\gamma_t^2.$$

Thus,

$$\log Z_t \leq \frac{1}{2}(-4\gamma_t^2) = -2\gamma_t^2.$$

Exponentiating both sides yields:

$$Z_t \leq \exp(-2\gamma_t^2).$$

(f) From the first two parts of the question we know that the training error of the ensemble is bounded by the product of the normalization factors:

$$\epsilon_{\text{ensemble}} \leq \prod_{t=1}^{T} Z_t.$$

Using the previously established bound $Z_t \leq \exp(-2\gamma_t^2)$ and the assumption $\gamma_t \geq \gamma$, we get:

$$\prod_{t=1}^{T} Z_t \leq \prod_{t=1}^{T} \exp(-2\gamma^2) = \exp\left(-2T\gamma^2\right).$$

Therefore, we have:

$$\epsilon_{\text{ensemble}} \leq \exp(-2T\gamma^2).$$

(g) To achieve zero error on the training set, we need the error rate of the ensemble to be smaller than $1/n$, where n is the number of training samples. Thus, we need to find the number of boosting iterations T that satisfies the following condition:

$$\exp(-2T\gamma^2) < \frac{1}{n}.$$

Taking the natural logarithm of both sides, we have:

$$-2T\gamma^2 < \log\left(\frac{1}{n}\right),$$
$$T > \frac{\log n}{2\gamma^2}.$$

Thus, for $T = \lceil \frac{\log n}{2\gamma^2} \rceil + 1$, we are guaranteed to have an error rate less than $1/n$, effectively achieving zero error on the training set.

(h) Suppose there exists a weak classifier h_t with an error rate $\epsilon_t > 0.5$. Then, we can construct another weak classifier h_t' by flipping the predictions of h_t. The error rate of h_t' would be $1 - \epsilon_t < 0.5$.

9.21 (a) Consider a multi-class classification problem with K classes. Using the softmax function, the predicted probability that a sample $\mathbf{x}_i$ belongs to class k at the m-th boosting round is given by:

$$p_{imk} = P(y = k|\mathbf{x}_i) = \frac{\exp(F_{m,k}(\mathbf{x}_i))}{\sum_{j=1}^{K} \exp(F_{m,j}(\mathbf{x}_i))},$$

where $F_{m,k}(\mathbf{x}_i)$ is the cumulative output of the regression trees for class k up to the m-th boosting round.

(b) The cross-entropy loss function for a sample $\mathbf{x}_i$ at the m-th boosting round, assuming the true class is represented by the one-hot encoded vector $\mathbf{y}_i$, is given by:

$$L_{\text{CE}}(\mathbf{y}_i, \mathbf{p}_{im}) = -\sum_{k=1}^{K} y_{ik} \log p_{imk},$$

where y_{ik} is the k-th component of the one-hot encoded vector $\mathbf{y}_i$, and p_{imk} is the predicted probability for class k at the m-th boosting round.

(c) The derivative of the cross-entropy loss with respect to each logit z_j in the context of multinomial logistic regression is:

$$\frac{\partial L}{\partial z_j} = p_j - y_j.$$

In our case, the logits z_j are represented by $F_{m,k}(\mathbf{x}_i)$, which are the cumulative outputs of the regression trees for each class k. Therefore, the derivative of the cross-entropy loss with respect to $F_{m,k}(\mathbf{x}_i)$ becomes:

$$\frac{\partial L_{\text{CE}}}{\partial F_{m,k}(\mathbf{x}_i)} = p_{imk} - y_{ik}.$$

The pseudo-residuals for class k are defined as the negative of this derivative:

$$r_{imk} = -\frac{\partial L_{\text{CE}}}{\partial F_{m,k}(\mathbf{x}_i)} = y_{ik} - p_{imk}.$$

These pseudo-residuals r_{imk} represent the difference between the true label y_{ik} and the predicted probability p_{imk}. By fitting regression trees to these pseudo-residuals, the model iteratively corrects its errors, improving prediction accuracy over successive boosting rounds.

(d) From part (c), we know that the first derivative of the cross-entropy loss with respect to $F_{m,k}(\mathbf{x}_i)$ is:

$$\frac{\partial L_{\text{CE}}}{\partial F_{m,k}(\mathbf{x}_i)} = p_{imk} - y_{ik}.$$

To find the second derivative, we need to differentiate $p_{imk} - y_{ik}$ with respect to $F_{m,k}(\mathbf{x}_i)$. Since y_{ik} is a constant (it represents the true label and does not depend on $F_{m,k}(\mathbf{x}_i)$), its derivative is zero. Thus, we have:

$$\frac{\partial^2 L_{\text{CE}}}{\partial F_{m,k}(\mathbf{x}_i)^2} = \frac{\partial}{\partial F_{m,k}(\mathbf{x}_i)} \left(p_{imk}\right).$$

Recall that p_{imk} is given by the softmax function:

$$p_{imk} = \frac{\exp(F_{m,k}(\mathbf{x}_i))}{\sum_{j=1}^{K} \exp(F_{m,j}(\mathbf{x}_i))}.$$

We differentiate p_{imk} with respect to $F_{m,k}(\mathbf{x}_i)$ using the quotient rule. The quotient rule states that if $f(x) = \frac{u(x)}{v(x)}$, then:

$$\frac{d}{dx} f(x) = \frac{u'(x)v(x) - u(x)v'(x)}{v(x)^2}.$$

Applying this to p_{imk}:

- Let $u = \exp(F_{m,k}(\mathbf{x}_i))$ and $v = \sum_{j=1}^{K} \exp(F_{m,j}(\mathbf{x}_i))$.
- Then, $u' = \exp(F_{m,k}(\mathbf{x}_i))$ and $v' = \exp(F_{m,k}(\mathbf{x}_i))$.

Thus, we obtain:

$$\begin{aligned}
\frac{\partial p_{imk}}{\partial F_{m,k}(\mathbf{x}_i)} &= \frac{\exp(F_{m,k}(\mathbf{x}_i)) \left(\sum_{j=1}^{K} \exp(F_{m,j}(\mathbf{x}_i)) \right) - \exp(F_{m,k}(\mathbf{x}_i)) \exp(F_{m,k}(\mathbf{x}_i))}{\left(\sum_{j=1}^{K} \exp(F_{m,j}(\mathbf{x}_i)) \right)^2} \\
&= \frac{\exp(F_{m,k}(\mathbf{x}_i))}{\sum_{j=1}^{K} \exp(F_{m,j}(\mathbf{x}_i))} \left(1 - \frac{\exp(F_{m,k}(\mathbf{x}_i))}{\sum_{j=1}^{K} \exp(F_{m,j}(\mathbf{x}_i))} \right) \\
&= p_{imk} \left(1 - p_{imk} \right).
\end{aligned}$$

Therefore,

$$\frac{\partial^2 L_{\text{CE}}}{\partial F_{m,k}(\mathbf{x}_i)^2} = p_{imk} \left(1 - p_{imk} \right).$$

(e) Using the first and second derivatives, we can derive the formula for the leaf output values γ_{jmk}. The coefficient γ_{jmk} for all samples $\mathbf{x}_i$ in region R_{jmk} of the regression tree for class k is given by:

$$\gamma_{jmk} = \frac{\sum_{\mathbf{x}_i \in R_{jmk}} (y_{ik} - p_{imk})}{\sum_{\mathbf{x}_i \in R_{jmk}} p_{imk}(1 - p_{imk})}.$$

This coefficient minimizes the cross-entropy loss for the samples in region R_{jmk} and is used to update the model's predictions.

Chapter 10

Gradient Boosting Libraries

10.1 (a), (b), (c), (e)

10.2 (a), (d)

10.3 (b), (d)

10.4 (e), (f)

10.5 (b)

10.6 (c)

10.7 (a), (d)

10.8 (b), (c), (d)

10.9 (a), (b), (c)

10.10 (b), (d)

Chapter 11

Support Vector Machines

11.1 (b)

11.2 (c)

11.3 (a)

11.4 (a), (c), (d)

11.5 (b), (d), (e)

11.6 (a), (d)

11.7 (b)

11.8 (b), (d), (e)

11.9 (b)

11.10 (d), (e)

11.11 (a) Support vectors correspond to data points with $\alpha_i > 0$. From the table, the support vectors are:

$$\mathbf{x}_2 = (2, 2), \mathbf{x}_3 = (4, 4), \mathbf{x}_4 = (5, 1).$$

(b) The weight vector is computed as:

$$\begin{aligned}\mathbf{w} &= \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \\ &= 0.312 \cdot 1 \cdot (2, 2) + 0.187 \cdot (-1) \cdot (4, 4) + 0.125 \cdot (-1) \cdot (5, 1) \\ &= (-0.749, -0.249)\end{aligned}$$

To compute the bias, we can use the support vector $\mathbf{x}_2 = (2, 2)$:

$$b = y_2 - \mathbf{w}^T \mathbf{x}_2 = 1 - (-0.749 \cdot 2 - 0.249 \cdot 2) = 2.996.$$

(c) The equation of the separating line is:

$$\begin{aligned} w_1 x_1 + w_2 x_2 + b &= 0 \\ -0.749 x_1 - 0.249 x_2 + 2.996 &= 0 \\ x_2 &= \frac{-2.996 + 0.749 x_1}{-0.249} \\ x_2 &= 12.032 - 3.008 x_1 \end{aligned}$$

Figure 11.1 shows the separating line and the margins.

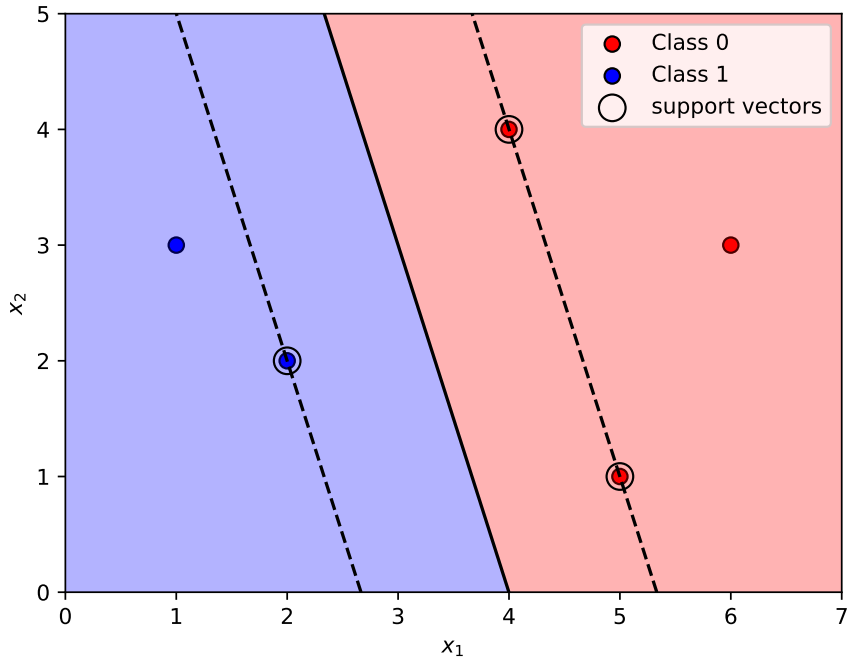


Figure 11.1: The separating line, margins, and support vectors.

(d) The margin width is given by:

$$\gamma = \frac{1}{\|w\|} = \frac{1}{\sqrt{(-0.749)^2 + (-0.249)^2}} = \frac{1}{0.789} = 1.267.$$

(e) The decision score of $\mathbf{x} = (3, 2)$ is:

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = -0.749 \cdot 3 - 0.249 \cdot 2 + 2.996 = 0.251.$$

Since $h(\mathbf{x}) > 0$, the predicted label is $+1$. The point is inside the margin because $0 < h(\mathbf{x}) < 1$.

11.13 (a) The primal optimization problem for the soft-margin SVM is:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i, \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n, \\ & \xi_i \geq 0, \quad i = 1, \ldots, n. \end{aligned}$$

(b) The Lagrangian for this optimization problem is given by:

$$\mathcal{L}(\mathbf{w}, b, \xi, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i\right] - \sum_{i=1}^{n}\mu_i\xi_i,$$

where:

- $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ are the Lagrange multipliers for the inequality constraints $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$, $\alpha_i \geq 0$.
- $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$ are the Lagrange multipliers for ξ_i, $\mu_i \geq 0$.

(c) At the optimal solution, the partial derivatives of $\mathcal{L}$ with respect to the primal variables $(\mathbf{w}, b, \xi_i)$ must be zero. Let's compute them:

- Derivative with respect to $\mathbf{w}$:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n}\alpha_i y_i\mathbf{x}_i = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i\mathbf{x}_i.$$

- Derivative with respect to b:

$$\frac{\partial\mathcal{L}}{\partial b} = -\sum_{i=1}^{n}\alpha_i y_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{n}\alpha_i y_i = 0.$$

- Derivative with respect to ξ_i:

$$\frac{\partial\mathcal{L}}{\partial\xi_i} = C - \alpha_i - \mu_i = 0 \quad \Rightarrow \quad \alpha_i + \mu_i = C.$$

Since $\mu_i \geq 0$, this implies $\alpha_i \leq C$.

(d) We start by substituting $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ into the Lagrangian to express the optimization problem in terms of the dual variables. The first step involves expanding the quadratic term $\|\mathbf{w}\|^2$:

$$\|\mathbf{w}\|^2 = \left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right)^T \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j.$$

Substituting this into the Lagrangian gives:

$$\begin{aligned}\mathcal{L} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + C\sum_{i=1}^{n} \xi_i \\ - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i\right) - \sum_{i=1}^{n} \mu_i \xi_i.\end{aligned}$$

Expand the constraint term $-\sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i\right)$:

$$-\sum_{i=1}^{n} \alpha_i y_i (\mathbf{w}^T \mathbf{x}_i) - \sum_{i=1}^{n} \alpha_i y_i b + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i \xi_i.$$

Using $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$, the term $\sum_{i=1}^{n} \alpha_i y_i (\mathbf{w}^T \mathbf{x}_i)$ becomes:

$$\sum_{i=1}^{n} \alpha_i y_i \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j.$$

Thus, the constraint term becomes:

$$-\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - b\sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i \xi_i.$$

After substitution, the Lagrangian becomes:

$$\mathcal{L} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + C\sum_{i=1}^{n} \xi_i - b\sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i \xi_i - \sum_{i=1}^{n} \mu_i \xi_i.$$

From the partial derivative of the Lagrangian with respect to b we have:

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

Thus, the term involving b vanishes.

Grouping the terms involving ξ_i:

$$C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}\mu_i\xi_i = \sum_{i=1}^{n}\xi_i(C-\alpha_i-\mu_i).$$

From the partial derivative of the Lagrangian with respect to ξ_i:

$$C-\alpha_i-\mu_i = 0 \quad \Rightarrow \quad \alpha_i+\mu_i = C.$$

Thus, the terms involving ξ_i vanish. After canceling the b and ξ_i terms, the Lagrangian simplifies to:

$$\mathcal{L} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i.$$

Hence, the dual optimization problem is:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j, \\ \text{subject to} \quad & 0 \le \alpha_i \le C, \quad i = 1,\ldots,n, \\ & \sum_{i=1}^{n}\alpha_i y_i = 0. \end{aligned}$$

11.5 (a) For large values of C, the penalty for misclassifying points is very high. As a result, the decision boundary will attempt to perfectly separate the data, if possible, potentially resulting in a more complex and less generalized boundary.

(b) For very small values of C, the classifier prioritizes maximizing the margin over correctly classifying every point. This could lead to a smoother decision boundary that tolerates misclassifications.

(c) A smaller C value (e.g., close to 0) may generalize better by focusing on margin maximization rather than perfect classification of every point, especially in noisy or overlapping regions.

(d) Adding a new point deep inside the correct classification region, far from the decision boundary, has little effect on the boundary. For instance, placing a red point (Class 1) inside the bottom-right red cluster will not alter the boundary.

(e) Adding a misclassified point forces the boundary to shift to reduce the error. For example, placing a blue point (Class 1) within the red region (Class 0) will significantly alter the boundary due to the high penalty on misclassifications.

11.16 (a) $k(\mathbf{x}, \mathbf{z}) = ck_1(\mathbf{x}, \mathbf{z})$ for $c > 0$.

Proof 1 (using Mercer's theorem):
Let K_1 be the kernel matrix for k_1. Since k_1 is a valid kernel, K_1 is positive semidefinite, i.e., $\mathbf{v}^T K_1 \mathbf{v} \geq 0$ for all $\mathbf{v} \in \mathbb{R}^n$.

The kernel matrix K for $k(\mathbf{x}, \mathbf{z} = c \cdot k_1(\mathbf{x}, \mathbf{z})$ is given by $K = c \cdot K_1$. Since $c > 0$, multiplying K_1 by c preserves positive semidefiniteness:

$$\mathbf{v}^T K \mathbf{v} = \mathbf{v}^T (c \cdot K_1)\mathbf{v} = c \cdot (\mathbf{v}^T K_1 \mathbf{v}) \geq 0.$$

Hence, $k(\mathbf{x}, \mathbf{z}) = c \cdot k_1(\mathbf{x}, \mathbf{z})$ is a valid kernel.

Proof 2 (using feature maps):
Let ϕ_1 be the feature map for kernel k_1, i.e., $k_1(\mathbf{x}, \mathbf{z}) = \phi_1(\mathbf{x})^T \phi_1(\mathbf{z})$. Then,

$$k(\mathbf{x}, \mathbf{z}) = c \cdot k_1(\mathbf{x}, \mathbf{z}) = (\sqrt{c}\phi_1(\mathbf{x}))^T (\sqrt{c}\phi_1(\mathbf{z})).$$

This shows that $k(\mathbf{x}, \mathbf{z})$ can be expressed as a dot product in the transformed feature space $\phi(\mathbf{x}) = \sqrt{c}\phi_1(\mathbf{x})$, which implies that $k(\mathbf{x}, \mathbf{z})$ is a valid kernel.

(b) $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$.

Proof 1 (using Mercer's theorem):
Let K_1 and K_2 be the kernel matrices corresponding to the kernels k_1 and k_2, respectively. Since k_1 and k_2 are valid kernels, K_1 and K_2 are positive semidefinite matrices. Their sum $K = K_1 + K_2$ is also positive semidefinite, because for any vector $\mathbf{v} \in \mathbb{R}^n$:

$$\mathbf{v}^T (K_1 + K_2)\mathbf{v} = \mathbf{v}^T K_1 \mathbf{v} + \mathbf{v}^T K_2 \mathbf{v} \geq 0,$$

as $\mathbf{v}^T K_1 \mathbf{v} \geq 0$ and $\mathbf{v}^T K_2 \mathbf{v} \geq 0$.
Thus, K is positive semidefinite, which implies that k is a valid kernel.

Proof 2 (using feature maps):
Let ϕ_1 and ϕ_2 be the feature maps for k_1 and k_2, respectively, such that:

$$k_1(\mathbf{x}, \mathbf{z}) = \phi_1(\mathbf{x})^T \phi_1(\mathbf{z}) \quad \text{and} \quad k_2(\mathbf{x}, \mathbf{z}) = \phi_2(\mathbf{x})^T \phi_2(\mathbf{z}).$$

Define the combined feature map:

$$\phi(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \end{bmatrix}.$$

The kernel $k(\mathbf{x}, \mathbf{z})$ can then be expressed as:

$$k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z}) = \phi_1(\mathbf{x})^T \phi_1(\mathbf{z}) + \phi_2(\mathbf{x})^T \phi_2(\mathbf{z}).$$

On the other hand, the dot product of the combined feature map is:

$$\phi(\mathbf{x})^T \phi(\mathbf{z}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \end{bmatrix}^T \begin{bmatrix} \phi_1(\mathbf{z}) \\ \phi_2(\mathbf{z}) \end{bmatrix} = \phi_1(\mathbf{x})^T \phi_1(\mathbf{z}) + \phi_2(\mathbf{x})^T \phi_2(\mathbf{z}).$$

This shows that:

$$k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z}),$$

which means that $k(\mathbf{x}, \mathbf{z})$ is a valid kernel, as it corresponds to a dot product in the transformed feature space $\phi(\mathbf{x})$.

(c) $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) \cdot k_2(\mathbf{x}, \mathbf{z})$

Proof 1 (using Mercer's theorem):
Let K_1 and K_2 be the kernel matrices corresponding to the kernels k_1 and k_2, respectively. Since k_1 and k_2 are valid kernels, both K_1 and K_2 are positive semidefinite matrices.

Now consider the element-wise product of K_1 and K_2, which gives the kernel matrix K corresponding to $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z})k_2(\mathbf{x}, \mathbf{z})$. By Schur product theorem, the element-wise product of two positive semidefinite matrices is also positive semidefinite.

Thus, $K = K_1 \circ K_2$ (element-wise product) is positive semidefinite. By Mercer's theorem, this implies that $k(\mathbf{x}, \mathbf{z})$ is a valid kernel.

Proof 2 (using feature maps):
Let ϕ_1 and ϕ_2 be the feature maps corresponding to the kernels k_1 and k_2, respectively, such that:

$$k_1(\mathbf{x}, \mathbf{z}) = \phi_1(\mathbf{x})^T \phi_1(\mathbf{z}) \quad \text{and} \quad k_2(\mathbf{x}, \mathbf{z}) = \phi_2(\mathbf{x})^T \phi_2(\mathbf{z}).$$

We construct a combined feature map using the Kronecker product, which captures all pairwise interactions between the elements of $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$.

For $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^n$, the Kronecker product is given as:

$$\mathbf{u} \otimes \mathbf{v} = \begin{bmatrix} u_1 v_1 \\ u_1 v_2 \\ \vdots \\ u_m v_n \end{bmatrix}.$$

This vectorization aligns with the concept of feature space mappings in kernel theory, as it produces a higher-dimensional vector that can be directly used for dot products in the transformed space.

We now show a key property of Kronecker product vectors: the dot product of two Kronecker product vectors corresponds to the product of the dot products of the original vectors:

$$\begin{aligned}(\mathbf{u} \otimes \mathbf{v})^T(\mathbf{u}' \otimes \mathbf{v}') &= \sum_{i=1}^{m}\sum_{j=1}^{n}(u_i v_j)(u_i' v_j') \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n}(u_i u_i')(v_j v_j') \\ &= (\mathbf{u}^T\mathbf{u}') \cdot (\mathbf{v}^T\mathbf{v}').\end{aligned}$$

Using this, we define the combined feature map as:

$$\phi(\mathbf{x}) = \phi_1(\mathbf{x}) \otimes \phi_2(\mathbf{x}).$$

The dot product in this transformed space is then:

$$\phi(\mathbf{x})^T\phi(\mathbf{z}) = (\phi_1(\mathbf{x}) \otimes \phi_2(\mathbf{x}))^T (\phi_1(\mathbf{z}) \otimes \phi_2(\mathbf{z})).$$

By the Kronecker product property shown above, this simplifies to:

$$\phi(\mathbf{x})^T\phi(\mathbf{z}) = (\phi_1(\mathbf{x})^T\phi_1(\mathbf{z})) \cdot (\phi_2(\mathbf{x})^T\phi_2(\mathbf{z})).$$

Thus, the combined kernel can be expressed as:

$$k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T\phi(\mathbf{z}).$$

This shows that $k(\mathbf{x}, \mathbf{z})$ is a valid kernel because it corresponds to a dot product in the transformed feature space $\phi(\mathbf{x})$.

11.18 (a) In the high-dimensional feature space $\mathbb{R}^p$, the gradient descent update rule becomes:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \sum_{i=1}^{n} \left(y_i - \mathbf{w}^T \phi(\mathbf{x}_i)\right) \phi(\mathbf{x}_i).$$

Computing $\phi(\mathbf{x})$ explicitly in a high-dimensional feature space can be computationally expensive, especially if p is very large or infinite (e.g., when using RBF kernels). Additionally, storing $\mathbf{w}$ in $\mathbb{R}^p$ and performing operations such as $\mathbf{w}^T \phi(\mathbf{x})$ require significant memory and computational resources.

(b) At initialization, we assume $\mathbf{w} = \mathbf{0}$, which satisfies:

$$\mathbf{w} = \sum_{j=1}^{n} \alpha_j \phi(\mathbf{x}_j),$$

where $\alpha_j = 0$ for all j.

Using induction, we show that this structure is preserved after each gradient descent update:

- Base case: At initialization, $\mathbf{w} = \mathbf{0}$, which can be written as:

$$\mathbf{w} = \sum_{j=1}^{n} \alpha_j \phi(\mathbf{x}_j),$$

where $\alpha_j = 0$ for all j.

- Inductive step: Assume that after t updates, $\mathbf{w} = \sum_{j=1}^{n} \alpha_j \phi(\mathbf{x}_j)$. For the $(t+1)$-th update, the gradient descent rule is:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \sum_{i=1}^{n} \left(y_i - \mathbf{w}^T \phi(\mathbf{x}_i)\right) \phi(\mathbf{x}_i).$$

Substituting $\mathbf{w} = \sum_{j=1}^{n} \alpha_j \phi(\mathbf{x}_j)$ into this rule:

$$\mathbf{w} \leftarrow \sum_{j=1}^{n} \alpha_j \phi(\mathbf{x}_j) + \eta \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{n} \alpha_j \phi(\mathbf{x}_j)^T \phi(\mathbf{x}_i) \right) \phi(\mathbf{x}_i).$$

The updated $\mathbf{w}$ remains a linear combination of $\phi(\mathbf{x}_j)$, with updated coefficients α_j.

By induction, $\mathbf{w} = \sum_{j=1}^{n} \alpha_j \phi(\mathbf{x}_j)$ holds at every step.

(c) Substituting $\mathbf{w} = \sum_{j=1}^{n} \alpha_j \phi(\mathbf{x}_j)$ into the gradient descent update rule:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \sum_{i=1}^{n} \left(y_i - \mathbf{w}^T \phi(\mathbf{x}_i)\right) \phi(\mathbf{x}_i).$$

Using the kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$, the update rule becomes:

$$\alpha_j \leftarrow \alpha_j + \eta \sum_{i=1}^{n} \left(y_i - \sum_{l=1}^{n} \alpha_l k(\mathbf{x}_l, \mathbf{x}_i) \right) k(\mathbf{x}_i, \mathbf{x}_j).$$

(d) Using $\mathbf{w} = \sum_{j=1}^{n} \alpha_j \phi(\mathbf{x}_j)$, the prediction for a new input $\mathbf{x}$ is:

$$\hat{y} = \mathbf{w}^T \phi(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j \phi(\mathbf{x}_j)^T \phi(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j k(\mathbf{x}_j, \mathbf{x}).$$

(e) Summary of kernelized gradient descent:

1. Initialize $\alpha_j = 0$ for $j = 1, \ldots, n$.
2. For each iteration:
   (a) Compute the predicted value for each training sample:

$$\hat{y}_i = \sum_{j=1}^{n} \alpha_j k(\mathbf{x}_j, \mathbf{x}_i).$$

   (b) Update α_j using the gradient descent rule:

$$\alpha_j \leftarrow \alpha_j + \eta \sum_{i=1}^{n} (y_i - \hat{y}_i)\, k(\mathbf{x}_i, \mathbf{x}_j).$$

3. To make predictions for a new input $\mathbf{x}$:

$$\hat{y} = \sum_{j=1}^{n} \alpha_j k(\mathbf{x}_j, \mathbf{x}).$$

11.20 (a) It is not possible to perfectly separate the positive and negative examples in the original feature space using a linear separator, as the data points alternate between positive and negative classes, resulting in two transitions.

(b) The transformed coordinates for the dataset using $\phi(x) = (x, x^2)$ are as follows:

$$\text{Positive samples: } (-3, 9), (2, 4), (3, 9)$$
$$\text{Negative samples: } (-1, 1), (0, 0), (1, 1).$$

A visualization of the transformed points is shown in Figure 11.2.

(c) Yes, the data points can be perfectly separated in the transformed $\mathbb{R}^2$ space using a linear separator, as the transformation $\phi(x)$ creates a clear separation.

(d) The kernel function corresponding to $\phi(x)$ is:

$$\begin{aligned} k(x, z) &= \phi(x) \cdot \phi(z) = (x, x^2) \cdot (z, z^2) \\ &= xz + x^2 z^2. \end{aligned}$$

(e) To maximize the margin, the hyperplane must lie equidistant between the closest points of opposite classes (the support vectors). Here, $\mathbf{u_1} = (1, 1)$ and $\mathbf{u_2} = (2, 4)$ are support vectors.

Thus, the hyperplane passes through the midpoint $(\frac{3}{2}, \frac{5}{2})$ of the line segment connecting $\mathbf{u_1}$ and $\mathbf{u_2}$ and is also perpendicular to this line segment.

The slope of the line connecting $\mathbf{u_1}$ and $\mathbf{u_2}$ is:

$$\frac{4-1}{2-1} = 3.$$

Since the hyperplane is perpendicular to this line, its slope is the negative reciprocal, which is -1/3. Using the $(\frac{3}{2}, \frac{5}{2})$ and the slope $-\frac{1}{3}$, we derive the equation of the hyperplane. Substituting into the point-slope form:

$$\frac{5}{2} = -\frac{1}{3} \cdot \frac{3}{2} + b \quad \Rightarrow \quad b = \frac{5}{2} + \frac{3}{6} = 3.$$

Therefore, the equation of the hyperplane is:

$$\frac{1}{3} z_1 + z_2 - 3 = 0,$$

where $w_1 = \frac{1}{3}$, $w_2 = 1$, and $b = -3$.

The margin is equal to half the Euclidean distance between the support vectors $\mathbf{u_1}$ and $\mathbf{u_2}$. The Euclidean distance between is:

$$\sqrt{(2-1)^2 + (4-1)^2} = \sqrt{10}.$$

Therefore, the margin is $\gamma = \dfrac{\sqrt{10}}{2}$.

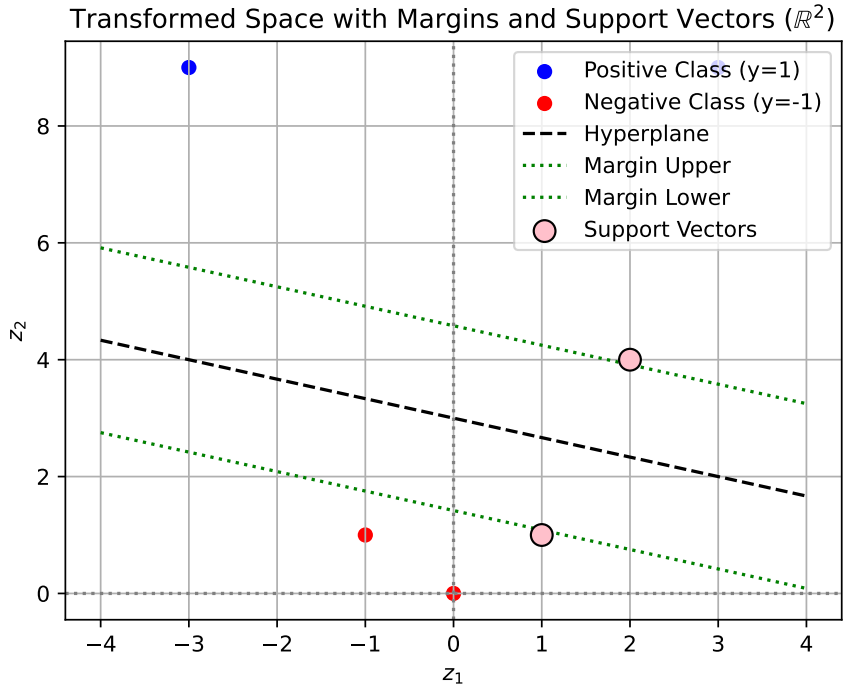


Figure 11.2: The transformed points in the feature space, the separating hyperplane and the support vectors.

(f) See Figure 11.2 for the hyperplane, margins, and support vectors.

(g) To transform the decision boundary back to the original feature space of x, we substitute the feature mapping $\mathbf{z} = \phi(x) = (x, x^2)$ into the hyperplane equation $z_2 = -\frac{1}{3}z_1 + 3$ from the transformed $\mathbb{R}^2$ space.

Thus, we substitute $z_1 = x$ and $z_2 = x^2$ into the hyperplane equation. This gives:

$$x^2 = -\frac{1}{3}x + 3.$$

To determine the decision boundary in the original feature space, we rearrange this equation into standard quadratic form:

$$3x^2 + x - 9 = 0$$

The roots of this equation are:

$$x = \frac{-1 \pm \sqrt{1^2 - 4(3)(-9)}}{6} = \frac{-1 \pm \sqrt{1 + 108}}{6} = \frac{-1 \pm \sqrt{109}}{6}.$$

The two roots are:

$$x_1 = \frac{-1 + \sqrt{109}}{6} = 1.573,$$

$$x_2 = \frac{-1 - \sqrt{109}}{6} = -1.906.$$

These roots represent the points in the original feature space where the decision boundary lies. The resulting decision boundary is visualized in Figure 11.3.

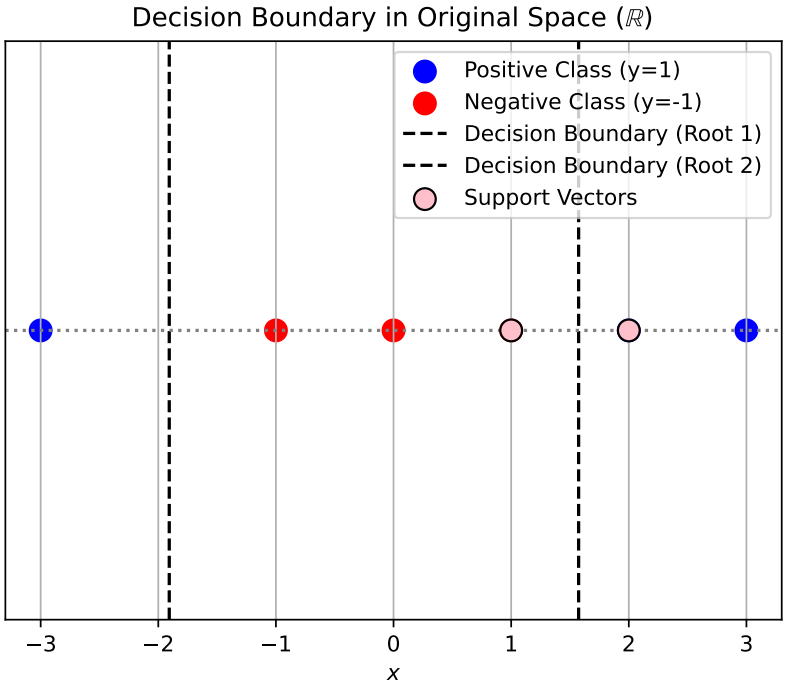


Figure 11.3: The separating hyperplane in the original space.

(h) To answer this problem, we need to solve the quadratic program on the support vectors $\mathbf{u_1}$ and $\mathbf{u_2}$. Since $\mathbf{u_1}$ and $\mathbf{u_2}$ are the only support vectors, only α_1 and α_2 are nonzero.

Due to the constraint on the quadratic program $\sum_{i=1}^{n} \alpha_i y_i = 0$, we also know that $\alpha_1 = \alpha_2 = \alpha$.

Thus, we can write the Lagrangian as follows:

$$\begin{aligned}
\mathcal{L}(\alpha) &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\
&= \alpha_1 + \alpha_2 - \frac{1}{2} \left(\alpha_1^2 \cdot (-1) \cdot (-1) \cdot \mathbf{u_1}^T \mathbf{u_1} + 2\alpha_1 \alpha_2 \cdot (-1) \cdot 1 \cdot \mathbf{u_1}^T \mathbf{u_2} + \alpha_2^2 \cdot 1 \cdot 1 \cdot \mathbf{u_2}^T \mathbf{u_2} \right) \\
&= \alpha_1 + \alpha_2 - \frac{1}{2} \alpha_1^2 \mathbf{u_1}^T \mathbf{u_1} + \alpha_1 \alpha_2 \mathbf{u_1}^T \mathbf{u_2} - \frac{1}{2} \alpha_2^2 \mathbf{u_2}^T \mathbf{u_2} \\
&= \alpha_1 + \alpha_2 - \alpha_1^2 + 6\alpha_1 \alpha_2 - 10\alpha_2^2 \\
&= 2\alpha - 5\alpha^2.
\end{aligned}$$

By taking the derivative and setting it equal to zero, we obtain:

$$2 - 10\alpha = 0 \quad \Rightarrow \quad \alpha = \frac{1}{5}.$$

Therefore, $\alpha_1 = \alpha_2 = \alpha = \frac{1}{5}$.

To compute b, we can compute $h(u_1)$ using the given equation:

$$\begin{aligned}
h(u_1) = \text{sign} \left(\sum_{i=1}^{\mathcal{S}} \alpha_i y_i k(u_1, u_i) + b \right) &= -1 \\
\frac{1}{5} \cdot (-1) \cdot (1,1) \cdot (1,1) + \frac{1}{5} \cdot 1 \cdot (1,1) \cdot (2,4) + b &= -1 \\
-\frac{2}{5} + \frac{6}{5} + b &= -1 \\
\frac{4}{5} + b &= -1 \\
b &= -\frac{9}{5}.
\end{aligned}$$

The final SVM decision function is:

$$h(x) = \text{sign} \left(\frac{1}{5} \cdot \left(y_1 k(x, \mathbf{u_1}) + y_2 k(x, \mathbf{u_2}) \right) - \frac{9}{5} \right).$$

Chapter 12

Summary and Additional Resources

12.1 (b), (c), (e)

12.2 (e)

12.3 (b), (c), (d)

12.4 (a)

12.5 (a), (b), (d), (e)

12.6 (c), (e)

12.7 (d)

12.8 (a), (b), (c), (f)

12.9 (d)

12.10 (b)

12.12 The models are:

1. Logistic regression
2. Decision tree
3. k-nearest neighbor with $k = 1$
4. k-nearest neighbor with $k = 10$
5. SVM with an RBF kernel
6. Gradient boosting

7. Gaussian Naive Bayes
8. SVM with a polynomial kernel

12.14 (a) k-nearest neighbors with $k = 1$: The training accuracy is not necessarily the same. The 1-nearest neighbor classifier assigns labels based on the single nearest neighbor. When we remove feature k, the dimensionality of the data reduces from d to $d - 1$. Although x_j is an affine function of x_k, removing one feature changes the distance metric. The pairwise distances between points may change, which can affect which neighbor is considered the closest. Therefore, the predicted labels can change, leading to potentially different training accuracy.

(b) Decision tree: The training accuracy will remain the same. Decision trees split the data by comparing feature values to a threshold, often chosen based on criteria like information gain. Since features x_j and x_k are perfectly correlated with the relationship:

$$x_{ik} = ax_{ij} + b, \quad \text{for some constants } a < 0 \text{ and } b,$$

each split that originally used x_k can be replaced with an equivalent split using x_j. Specifically, if a node split is defined by:

$$x_{ik} \leq t,$$

we can rewrite this condition as:

$$ax_{ij} + b \leq t \quad \Rightarrow \quad x_{ij} \leq \frac{t - b}{a}.$$

Thus, the decision tree can seamlessly use x_j instead of x_k, with an adjusted threshold $\frac{t-b}{a}$. Removing x_k does not change the structure of the tree or its predictions, ensuring that the training accuracy remains the same.

(c) Hard-margin SVM: The training accuracy will remain the same. In a hard-margin SVM, if the data is linearly separable, the solution is the maximum-margin hyperplane with no training errors. Removing the perfectly correlated feature x_k does not affect the linear separability of the data.

Hard-margin SVM finds the maximum-margin hyperplane defined by:

$$\mathbf{w}^T\mathbf{x} + b = 0.$$

Since features x_j and x_k are perfectly correlated, there exists a relationship:

$$x_{ik} = ax_{ij} + b, \quad \text{for some constants } a < 0 \text{ and } b.$$

Substitute this into the hyperplane equation:

$$w_j x_{ij} + w_k x_{ik} = w_j x_{ij} + w_k(a x_{ij} + b) = (w_j + a w_k) x_{ij} + w_k b.$$

After removing x_k, the new hyperplane in the reduced space is defined by a new weight $\tilde{w}_j = w_j + a w_k$ for x_j, and an adjusted bias term $b' = b + w_k b$. Although the equation changes, the decision boundary and maximum margin remain the same because the geometry and distances between points are preserved. Consequently, the training accuracy is unaffected.

(d) Soft-margin SVM with $C = 1$: The training accuracy is not necessarily the same. In a soft-margin SVM, the objective function balances two competing goals: maximizing the margin and minimizing the hinge loss (classification errors), controlled by the regularization parameter C. The optimization problem is:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i, \quad \text{subject to} \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

When features x_j and x_k are perfectly correlated:

$$x_{ik} = a x_{ij} + b,$$

the removal of x_k reduces the dimensionality of the feature space and constrains the optimization problem. Although the data remains linearly separable, the removal of x_k may force the model to find a new hyperplane with a different normal vector $\tilde{\mathbf{w}}$. Since the optimization seeks to minimize $\|\mathbf{w}\|$, having both correlated features may allow the model to find a separator with a smaller norm of $\|\mathbf{w}\|$ in the higher-dimensional space.

After removing x_k, the model may have to trade off accuracy for a smaller norm of $\|\tilde{\mathbf{w}}\|$, potentially leading to higher slack variables ξ_i. This can result in a different decision boundary and possibly a lower training accuracy.

12.15 (a) For this task, a gradient boosting model such as XGBoost or LightGBM is appropriate. These algorithms offer strong predictive performance, handle both numerical and categorical features, and are robust to class imbalance when appropriate techniques (e.g., class weighting, sampling) are applied. They are also relatively fast in making predictions, making them suitable for real-time detection. In addition, tree-based models offer some interpretability, which is important for auditability and customer explanations.

Justification:

- Domain suitability: Handles tabular financial data well.
- Predictive accuracy: State-of-the-art on many structured classification benchmarks.
- Data requirements: Scales well to large datasets (1 million rows).
- Computational efficiency: Efficient at training and inference time.
- Interpretability: Feature importance and SHAP values provide useful explanations.
- Robustness to noise and imbalance: Supports weighted loss functions and imbalance-aware techniques.

Data Preprocessing:

- Handle missing values using imputation (e.g., mean or median for numeric fields, most frequent for categorical).
- Encode categorical variables using target encoding or one-hot encoding, depending on cardinality and model support.
- Feature engineering: Create derived features such as transaction amount over balance, frequency patterns, country risk levels, etc.
- Apply oversampling (e.g., SMOTE) or class-weighting to address imbalance.

Evaluation and Testing:

- Use stratified train/validation/test splits to maintain class proportions.
- Metrics: Precision, recall, F1-score, and Area Under the Curve (AUC) are critical due to class imbalance.
- Analyze the confusion matrix to understand the cost of false positives vs. false negatives.
- Validate using time-based splits if transactions have timestamps, to avoid data leakage.

(b) A suitable approach is to use a hybrid model: a gradient boosting regressor (e.g., XGBoost) for structured data combined with a pre-trained language model (e.g., ClinicalBERT or BioBERT) for unstructured clinical notes.

Justification:

- Domain suitability: Structured hospital data and free-text notes require different representations.
- Predictive accuracy: Gradient boosting performs well on structured data, while BERT models excel at extracting semantic information from clinical text.

- Data requirements: Both models handle sparse and noisy inputs well.
- Computational efficiency: Prediction speed is not a constraint; training complexity is acceptable for offline models.
- Interpretability: Feature importance and SHAP values for structured features, and attention weights or concept extraction from BERT improve interpretability.
- Robustness: Ensemble of models increases generalization across units and populations.

Data Preprocessing:

- Impute missing structured data (e.g., median for lab results).
- Encode high-cardinality categorical variables (e.g., target or frequency encoding).
- Normalize skewed numerical features (e.g., log-transform lab values or stay durations).
- Extract features from unstructured notes using embeddings from ClinicalBERT or BioBERT.
- Optionally bin the target (e.g., short vs long stays) if switching to classification.
- Detect and cap or model outliers in length-of-stay to reduce distortion.

Evaluation and Testing:

- Use a stratified or grouped split by hospital unit to ensure generalization across settings.
- Use metrics such as R^2 score and Root Mean Squared Error (RMSE) to evaluate regression accuracy.
- Evaluate calibration and prediction intervals, especially for long-stay predictions.
- Consider segmenting performance by patient type (e.g., age group, diagnosis) to ensure fairness and reliability.

Appendix A

Linear Algebra

A.6 We want to prove the Cauchy–Schwarz inequality, which states that for any vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,

$$|\mathbf{x}^T\mathbf{y}| \leq \|\mathbf{x}\|_2\|\mathbf{y}\|_2.$$

The dot product of two vectors $\mathbf{x}$ and $\mathbf{y}$ is related to the angle θ between them by the identity:

$$\cos\theta = \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2}.$$

Rearranging, we can express their dot product as:

$$\mathbf{x}^T\mathbf{y} = \|\mathbf{x}\|_2\|\mathbf{y}\|_2\cos\theta.$$

Taking the absolute value on both sides:

$$|\mathbf{x}^T\mathbf{y}| = |\|\mathbf{x}\|_2\|\mathbf{y}\|_2|\cos\theta| = \|\mathbf{x}\|_2\|\mathbf{y}\|_2|\cos\theta|.$$

Since $-1 \leq \cos\theta \leq 1$, it follows that $|\cos\theta| \leq 1$. Substituting this into the equation above, we obtain:

$$|\mathbf{x}^T\mathbf{y}| \leq \|\mathbf{x}\|_2\|\mathbf{y}\|_2.$$

This completes the proof of the Cauchy-Schwarz inequality.

A.9 The ℓ_2-norm and ℓ_1-norm of $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ are given by:

$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2},$$

$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n|.$$

We first prove that $\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1$.

Expanding the square of the ℓ_1-norm:

$$\|\mathbf{x}\|_1^2 = (|x_1| + \cdots + |x_n|)^2 = \sum_{i=1}^{n} x_i^2 + 2 \sum_{1 \leq i < j \leq n} |x_i x_j| \geq \sum_{i=1}^{n} x_i^2 = \|\mathbf{x}\|_2^2.$$

Taking the square root on both sides gives:

$$\|\mathbf{x}\|_1 \geq \|\mathbf{x}\|_2,$$

or equivalently,

$$\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1.$$

Next, we prove that $\|\mathbf{x}\|_1 \leq \sqrt{n}\|\mathbf{x}\|_2$.

We express the ℓ_1-norm as a dot product:

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i| = \mathbf{u}^T \mathbf{v},$$

where we define the vectors:

$$\mathbf{u} = (|x_1|, |x_2|, \ldots, |x_n|), \quad \mathbf{v} = (1, 1, \ldots, 1).$$

Applying the Cauchy–Schwarz inequality:

$$|\mathbf{u}^T \mathbf{v}| \leq \|\mathbf{u}\|_2 \|\mathbf{v}\|_2.$$

We now compute the ℓ_2 norms of $\mathbf{u}$ and $\mathbf{v}$:

$$\|\mathbf{u}\|_2 = \sqrt{|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \|\mathbf{x}\|_2,$$
$$\|\mathbf{v}\|_2 = \sqrt{1^2 + 1^2 + \cdots + 1^2} = \sqrt{n}.$$

Substituting these values into the Cauchy–Schwarz inequality, we obtain:

$$\|\mathbf{x}\|_1 \leq \sqrt{n}\|\mathbf{x}\|_2.$$

A.13 To determine whether the vectors $(1, 2, 3)$, $(4, 5, 6)$, and $(7, 8, 9)$ are linearly independent, we check whether the only solution to the equation

$$\alpha(1, 2, 3) + \beta(4, 5, 6) + \gamma(7, 8, 9) = (0, 0, 0)$$

is $\alpha = \beta = \gamma = 0$.

This leads to the system of equations:

$$\begin{cases} \alpha + 4\beta + 7\gamma = 0, \\ 2\alpha + 5\beta + 8\gamma = 0, \\ 3\alpha + 6\beta + 9\gamma = 0. \end{cases}$$

Writing this system as an augmented matrix:

$$\left(\begin{array}{ccc|c} 1 & 4 & 7 & 0 \\ 2 & 5 & 8 & 0 \\ 3 & 6 & 9 & 0 \end{array}\right)$$

Performing row operations:

1. Subtract $2R_1$ from R_2 and $3R_1$ from R_3:

$$\left(\begin{array}{ccc|c} 1 & 4 & 7 & 0 \\ 0 & -3 & -6 & 0 \\ 0 & -6 & -12 & 0 \end{array}\right)$$

2. Divide R_2 by -3:

$$\left(\begin{array}{ccc|c} 1 & 4 & 7 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & -6 & -12 & 0 \end{array}\right)$$

3. Add $6R_2$ to R_3:

$$\left(\begin{array}{ccc|c} 1 & 4 & 7 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Since the last row is all zeros, the rank of the coefficient matrix is 2, which is less than 3. This means the vectors are linearly dependent.

A.19 To check if T is a linear transformation, we need to verify the following properties:

1. Additivity: For any vectors $(x_1, y_1, z_1), (x_2, y_2, z_2) \in \mathbb{R}^3$, we need to check if

$$T((x_1, y_1, z_1) + (x_2, y_2, z_2)) = T(x_1, y_1, z_1) + T(x_2, y_2, z_2).$$

Left-hand side:

$$T((x_1+x_2, y_1+y_2, z_1+z_2)) = (x_1+x_2+y_1+y_2, y_1+y_2+z_1+z_2, x_1+x_2-(z_1+z_2)).$$

Right-hand side:

$$\begin{aligned} T(x_1, y_1, z_1) + T(x_2, y_2, z_2) &= (x_1 + y_1, y_1 + z_1, x_1 - z_1) + (x_2 + y_2, y_2 + z_2, x_2 - z_2) \\ &= (x_1 + y_1 + x_2 + y_2, y_1 + z_1 + y_2 + z_2, x_1 - z_1 + x_2 - z_2) \\ &= (x_1 + x_2 + y_1 + y_2, y_1 + y_2 + z_1 + z_2, x_1 + x_2 - (z_1 + z_2)). \end{aligned}$$

Since the left-hand side and right-hand side are equal, the additivity property holds.

2. Scalar multiplication: For any scalar $c \in \mathbb{R}$ and vector $(x, y, z) \in \mathbb{R}^3$, we need to check if

$$T(c(x, y, z)) = c\,T(x, y, z).$$

Left-hand side:

$$T(cx, cy, cz) = (cx + cy, cy + cz, cx - cz).$$

Right-hand side:

$$cT(x, y, z) = c(x + y, y + z, x - z) = (c(x + y), c(y + z), c(x - z)).$$

Since both sides are equal, the scalar multiplication property holds.

Since both properties are satisfied, T is a linear transformation.

A.24 The statement is incorrect. Counterexample: consider the matrix

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

We compute A^2:

$$A^2 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

Since $A^2 = A$, the matrix A is idempotent. However, A is neither the identity matrix nor the zero matrix. Therefore, the statement is false.

A.35 Let A be an $n \times n$ lower-triangular matrix, meaning that all elements above the main diagonal are zero:

$$A = \begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ a_{21} & a_{22} & 0 & \cdots & 0 \\ a_{31} & a_{32} & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}.$$

Since A is invertible, there exists a matrix $B = A^{-1}$ such that:

$$AB = BA = I.$$

Let's denote B as:

$$B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & \cdots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \cdots & b_{2n} \\ b_{31} & b_{32} & b_{33} & \cdots & b_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & b_{n3} & \cdots & b_{nn} \end{pmatrix}.$$

Our goal is to show that all elements above the main diagonal of B (i.e., b_{ij}) for $i < j$) are zero.

Since $AB = I$, the entries of the identity matrix satisfy:

$$(AB)_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} = \delta_{ij}, \quad \text{where } \delta_{ij} \text{ is the Kronecker delta.}$$

For $i < j$ (above the main diagonal), the left-hand side expands as:

$$\sum_{k=1}^{n} a_{ik} b_{kj} = 0.$$

Since A is lower-triangular, we know that $a_{ik} = 0$ for all $k > i$, meaning that the sum only includes terms where $k \leq i$:

$$a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{ii} b_{ij} = 0.$$

Since A is invertible, its diagonal elements are nonzero, i.e., $a_{ii} \neq 0$. The equation above expresses b_{ij} in terms of previously determined entries b_{kj} for $k < i$.

By induction:

- For $i = 1$, the only term in the sum is $a_{11}b_{1j} = 0$, which implies $b_{1j} = 0$.
- For $i = 2$, the sum involves b_{2j} and previously determined $b_{1j} = 0$, so we get $b_{2j} = 0$.
- Repeating this process, we conclude $b_{ij} = 0$ for all $i < j$, meaning all entries above the diagonal in B are zero.

Since we have shown that A^{-1} has no nonzero entries above the diagonal, it follows that A^{-1} is also lower-triangular.

A.40 The range space of A is the span of its column vectors:

$$\mathbf{a}_1 = \begin{pmatrix} 1 \\ 4 \end{pmatrix}, \quad \mathbf{a}_2 = \begin{pmatrix} 2 \\ 5 \end{pmatrix}, \quad \mathbf{a}_3 = \begin{pmatrix} 3 \\ 6 \end{pmatrix}.$$

Since A is a 2×3 matrix, its rank is $\text{rank}(A) \leq 2$, thus the three column vectors must be linearly dependent. To find the dependency, suppose there exist scalars c_1, c_2, c_3, not all zero, such that:

$$c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + c_3\mathbf{a}_3 = \mathbf{0}.$$

Substituting the column vectors:

$$c_1 \begin{pmatrix} 1 \\ 4 \end{pmatrix} + c_2 \begin{pmatrix} 2 \\ 5 \end{pmatrix} + c_3 \begin{pmatrix} 3 \\ 6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

This results in the system:

$$\begin{aligned} c_1 + 2c_2 + 3c_3 &= 0, \\ 4c_1 + 5c_2 + 6c_3 &= 0. \end{aligned}$$

To solve, we multiply the first equation by 4:

$$4c_1 + 8c_2 + 12c_3 = 0.$$

We then subtract from the second equation:

$$\begin{aligned} (4c_1 + 5c_2 + 6c_3) - (4c_1 + 8c_2 + 12c_3) &= 0, \\ -3c_2 - 6c_3 &= 0, \\ c_2 &= -2c_3. \end{aligned}$$

Substituting into the first equation:

$$\begin{aligned} c_1 + 2 \cdot (-2c_3) + 3c_3 = 0, \\ c_1 - c_3 = 0, \\ c_1 = c_3. \end{aligned}$$

Thus, we get the linear dependence relation:

$$\begin{aligned} c_3\mathbf{a}_1 - 2c_3\mathbf{a}_2 + c_3\mathbf{a}_3 = 0, \\ \mathbf{a}_1 - 2\mathbf{a}_2 + \mathbf{a}_3 = 0 \\ \mathbf{a}_3 = -\mathbf{a}_1 + 2\mathbf{a}_2. \end{aligned}$$

Since $\mathbf{a}_3$ is a linear combination of $\mathbf{a}_1$ and $\mathbf{a}_2$, the rank of A is 2, and the range space is spanned by the first two columns of A:

$$\text{range}(A) = \text{span}\left\{ \begin{pmatrix} 1 \\ 4 \end{pmatrix}, \begin{pmatrix} 2 \\ 5 \end{pmatrix} \right\}.$$

The null space consists of all solutions to the homogeneous system:

$$A\mathbf{x} = 0.$$

That is:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

This gives the system:

$$\begin{aligned} x_1 + 2x_2 + 3x_3 = 0, \\ 4x_1 + 5x_2 + 6x_3 = 0. \end{aligned}$$

Since this is the same system of equations obtained earlier when determining the linear dependence among the columns of A, its solution is:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = x_3 \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}.$$

Thus, the null space of A is:

$$\text{null}(A) = \text{span}\left\{\begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}\right\}.$$

The nullity (dimension of the null space) of A is 1, as its null space is spanned by a single vector. In addition, we previously determined that the rank of A is 2. Since A has $n = 3$ columns, the rank-nullity theorem is verified:

$$\text{rank}(A) + \text{nullity}(A) = 2 + 1 = 3.$$

A.49 Let A be an $n \times n$ symmetric matrix. Since A is symmetric, it is guaranteed to have n real eigenvalues (not necessarily distinct).

For each eigenvalue λ_i, let its geometric multiplicity be k_i. Since all eigenvalues are real, the corresponding eigenspace is a subspace of $\mathbb{R}^n$ with dimension k_i, meaning there are k_i linearly independent eigenvectors in $\mathbb{R}^n$ associated with λ_i.

Applying the Gram-Schmidt process to the eigenvectors of each eigenspace, we obtain an orthonormal basis for every eigenspace. Repeating this for all eigenvalues, we obtain a set of n orthonormal eigenvectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$. These eigenvectors form a complete orthonormal basis for $\mathbb{R}^n$, as any two eigenvectors corresponding to distinct eigenvalues are orthogonal (as shown in the previous exercise).

We arrange these orthonormal eigenvectors as the columns of a matrix Q:

$$Q = \begin{pmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \\ | & | & & | \end{pmatrix}.$$

Since the columns of Q are orthonormal, Q is an orthogonal matrix, which implies that $Q^T Q = I$.

Next, we define the diagonal matrix Λ containing the eigenvalues:

$$\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n).$$

Since $A\mathbf{v}_i = \lambda_i \mathbf{v}_i$ for each eigenvector $\mathbf{v}_i$, we can express this in matrix form:

$$AQ = Q\Lambda.$$

Multiplying both sides by Q^T on the right, we obtain:

$$A = Q\Lambda Q^T.$$

Thus, every symmetric matrix is orthogonally diagonalizable, proving the spectral theorem.

A.54 The matrix is positive semidefinite if for all vectors $\mathbf{x} \in \mathbb{R}^3$, we have:

$$\mathbf{x}^T A\mathbf{x} \geq 0.$$

For an arbitrary vector $\mathbf{x} = (x_1, x_2, x_3)^T$, we compute:

$$\mathbf{x}^T A\mathbf{x} = (x_1, x_2, x_3) \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

Expanding the quadratic form:

$$\begin{aligned} \mathbf{x}^T A\mathbf{x} &= x_1(x_1 + x_3) + x_2(x_2 + x_3) + x_3(x_1 + x_2 + 2x_3) \\ &= x_1^2 + 2x_1x_3 + x_2^2 + 2x_2x_3 + 2x_3^2 \\ &= (x_1 + x_3)^2 + (x_2 + x_3)^2 \geq 0. \end{aligned}$$

Since this expression is always nonnegative for all x_1, x_2, x_3, the matrix A is positive semidefinite.

Appendix B

Calculus

B.4 We prove the Fundamental Theorem of Polynomial Interpolation in two parts: existence and uniqueness.

For existence, we construct the interpolating polynomial using the Lagrange interpolation formula:

$$P(x) = \sum_{i=0}^{n} y_i L_i(x),$$

where the Lagrange basis polynomials are defined as

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

Each basis polynomial $L_i(x)$ satisfies the property $L_i(x_j) = \delta_{ij}$ (the Kronecker delta), ensuring that $P(x_i) = y_i$ for all $i = 0, \ldots, n$. Thus, the polynomial $P(x)$ of degree at most n that interpolates the given points exists.

To prove uniqueness, suppose there exist two polynomials, $P_1(x)$ and $P_2(x)$, of degree at most n that satisfy the interpolation conditions:

$$P_1(x_i) = P_2(x_i) = y_i, \quad \text{for } i = 0, 1, \ldots, n.$$

Define the polynomial $Q(x) = P_1(x) - P_2(x)$. Then, $Q(x)$ is a polynomial of degree at most n and satisfies

$$Q(x_i) = P_1(x_i) - P_2(x_i) = 0, \quad \text{for } i = 0, 1, \ldots, n.$$

By the Root Theorem, since $Q(x)$ has $n+1$ distinct roots and is a polynomial of degree at most n, it must be the zero polynomial, i.e., $Q(x) \equiv 0$. This implies that $P_1(x) = P_2(x)$ for all x, proving uniqueness.

Thus, we have shown that there exists a unique polynomial $P(x)$ of degree at most n that interpolates the given points.

B.11 We want to prove that for any real number $k > 0$, the exponential function e^x grows faster than the power function x^k, meaning:

$$\lim_{x\to\infty} \frac{x^k}{e^x} = 0.$$

Since both $x^k \to \infty$ and $e^x \to \infty$ as $x \to \infty$, we can apply L'Hôpital's Rule repeatedly, differentiating x^k a total of k times:

$$\begin{aligned}\lim_{x\to\infty} \frac{x^k}{e^x} &= \lim_{x\to\infty} \frac{kx^{k-1}}{e^x} \\ &= \lim_{x\to\infty} \frac{k(k-1)x^{k-2}}{e^x} \\ &\vdots \\ &= \lim_{x\to\infty} \frac{k!}{e^x}.\end{aligned}$$

Since e^x still grows exponentially while $k!$ is a constant, we conclude:

$$\lim_{x\to\infty} \frac{k!}{e^x} = 0.$$

Thus, we have proved that:

$$\lim_{x\to\infty} \frac{x^k}{e^x} = 0.$$

B.18 (a) To prove that $f(x) = |x|$ is Lipschitz continuous on $\mathbb{R}$, we need to show that there exists a constant L such that for all $x_1, x_2 \in \mathbb{R}$,

$$||x_1| - |x_2|| \leq L|x_1 - x_2|.$$

From the triangle inequality, we have for all $x_1, x_2 \in \mathbb{R}$:

$$||x_1| - |x_2|| \leq |x_1 - x_2|.$$

Thus, $f(x) = |x|$ is Lipschitz continuous with constant $L = 1$.

(b) To prove that $f(x) = \sin x$ is Lipschitz continuous on $\mathbb{R}$, we need to show that there exists a constant L such that for all $x_1, x_2 \in \mathbb{R}$,

$$|\sin x_1 - \sin x_2| \leq L|x_1 - x_2|.$$

Since $\sin x$ is differentiable on $\mathbb{R}$, it satisfies the conditions of the Mean Value Theorem. Thus, for any $x_1, x_2 \in \mathbb{R}$ with $x_1 \neq x_2$, there exists some c in the interval (x_1, x_2) such that:

$$\frac{\sin x_1 - \sin x_2}{x_1 - x_2} = \cos c.$$

Since $|\cos c| \leq 1$ for all $c \in \mathbb{R}$, it follows that:

$$|\sin x_1 - \sin x_2| = |\cos c||x_1 - x_2| \leq |x_1 - x_2|. \tag{B.1}$$

This shows that $f(x) = \sin x$ is Lipschitz continuous with $L = 1$.

(c) To prove that $f(x) = \sqrt{1 + x^2}$ is Lipschitz continuous, we must show that there exists a constant L such that for all $x_1, x_2 \in \mathbb{R}$,

$$|f(x_1) - f(x_2)| \leq L|x_1 - x_2|.$$

Since $f(x)$ is differentiable on $\mathbb{R}$, we apply the Mean Value Theorem. For any $x_1, x_2 \in \mathbb{R}$ with $x_1 \neq x_2$, there exists some c in the interval (x_1, x_2) such that:

$$\frac{f(x_1) - f(x_2)}{x_1 - x_2} = f'(c).$$

The derivative of $f(x)$ is:

$$f'(x) = \frac{x}{\sqrt{1 + x^2}}.$$

Taking absolute values, we obtain:

$$|f'(x)| = \left| \frac{x}{\sqrt{1 + x^2}} \right|.$$

To find an upper bound, we rewrite the expression as:

$$\left| \frac{x}{\sqrt{1 + x^2}} \right| = \left| \frac{x}{\sqrt{x^2(1 + 1/x^2)}} \right| = \frac{|x|}{|x|\sqrt{1 + 1/x^2}} = \frac{1}{\sqrt{1 + 1/x^2}}.$$

Since $\sqrt{1+1/x^2} \geq 1$ for all $x \neq 0$, we conclude that:

$$|f'(x)| \leq 1 \quad \text{for all } x \in \mathbb{R}.$$

Thus, applying the Mean Value Theorem, we obtain:

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|.$$

This proves that $f(x) = \sqrt{1+x^2}$ is Lipschitz continuous with Lipschitz constant $L = 1$.

B.32 (a) The function is well-defined for all $x \in \mathbb{R}$ since the denominator $1 + x^2$ is always positive and the numerator $x^2 e^{-x}$ is defined everywhere.
Thus, the domain of $f(x)$ is:

$$\text{dom}(f) = (-\infty, \infty).$$

Since $f(x)$ is given as a quotient of continuous functions and the denominator is never being zero, it follows that $f(x)$ is continuous for all $x \in \mathbb{R}$.

(b) To find the critical points, we compute the derivative using the quotient rule:

$$\begin{aligned} f'(x) &= \frac{(2xe^{-x} - x^2e^{-x})(1+x^2) - (x^2e^{-x})(2x)}{(1+x^2)^2} \\ &= \frac{e^{-x}\left[(2x - x^2)(1+x^2) - 2x^3\right]}{(1+x^2)^2} \\ &= \frac{e^{-x}(2x - x^2 - x^4)}{(1+x^2)^2}. \end{aligned}$$

Since e^{-x} and the denominator are always positive, the critical points occur when:

$$2x - x^2 - x^4 = 0.$$

Factoring out x:

$$x(2 - x - x^3) = 0.$$

This gives one obvious critical point at $x = 0$. For the additional critical points, we solve:

$$x^3 + x - 2 = 0.$$

Using the Rational Root Theorem, we test possible rational roots $\pm 1, \pm 2$. Substituting $x = 1$:

$$1^3 + 1 - 2 = 0.$$

Thus, $x = 1$ is a root, and we factor $(x - 1)$ out using synthetic division:

$$\begin{array}{c|cccc} 1 & 1 & 0 & 1 & -2 \\ & & 1 & 1 & 2 \\ \hline & 1 & 1 & 2 & 0 \end{array}$$

The quotient is $x^2 + x + 2$, which has no real roots since its discriminant is negative $(1^2 - 4 \cdot 2 = -7)$.

Thus, the only real critical points are:

$$x = 0, \quad x = 1.$$

(c) We classify the critical points using the first derivative test (computing the second derivative is algebraically complicated in this case).

Since e^{-x} and $(1 + x^2)^2$ are always positive, the sign of $f'(x)$ depends only on:

$$2x - x^2 - x^4.$$

We first analyze $f'(x)$ around $x = 0$:

- For $x < 0$, we choose $x = -1$:

$$2 \cdot (-1) - (-1)^2 - (-1)^4 = -4.$$

  Since this is negative, we have $f'(x) < 0$ for $x < 0$.
- For $x > 0$, we choose $x = 0.5$:

$$2 \cdot 0.5 - (0.5)^2 - (0.5)^4 = 0.6875.$$

  Since this is positive, we have $f'(x) > 0$ for $x > 0$.

Since $f'(x)$ changes from negative to positive at $x = 0$, the function has a local minimum at $x = 0$.

We now analyze $f'(x)$ around $x = 1$:

- For $x < 1$, since $0.5 < 1$ and we already computed $f'(0.5) = 0.6875 > 0$, we conclude that $f'(x) > 0$ for $x < 1$.

- For $x > 1$, we choose $x = 1.5$:

$$2 \cdot 1.5 - (1.5)^2 - (1.5)^4 = -4.3125.$$

  Since this is negative, we have $f'(x) < 0$ for $x > 1$.

Since $f'(x)$ changes from positive to negative at $x = 1$, the function has a local maximum at $x = 1$.

(d) Asymptotic behavior:

- As $x \to \infty$, $e^{-x} \to 0$, and the fraction vanishes:

$$\lim_{x \to \infty} f(x) = 0.$$

  This implies that the function has a horizontal asymptote at $y = 0$.

- As $x \to -\infty$, the exponential $e^{-x} \to \infty$, and since $x^2 e^{-x}$ grows faster than $1 + x^2$, we conclude:

$$\lim_{x \to -\infty} f(x) \to \infty.$$

  This shows that the function grows without bound as $x \to -\infty$, meaning it does not have a horizontal asymptote in that direction.

(e) The function has no global maximum since $f(x) \to \infty$ as $x \to -\infty$.
The function has a global minimum at $x = 0$, where

$$f(0) = \frac{0^2 e^0}{1 + 0^2} = 0.$$

Since $f(x) \to 0$ as $x \to \infty$, and the function never attains a lower value than 0, the global minimum is achieved at $x = 0$ with $f(0) = 0$.

(f) The graph of the function is shown in Figure B.1.

B.38 We prove the Fundamental Theorem of Calculus (Part 1). Let $f(x)$ be a continuous function on the interval $[a, b]$, and define the function:

$$F(x) = \int_a^x f(t)\, dt.$$

We want to show that $F(x)$ is differentiable at every $x \in (a, b)$ and that its derivative satisfies:

$$F'(x) = f(x).$$

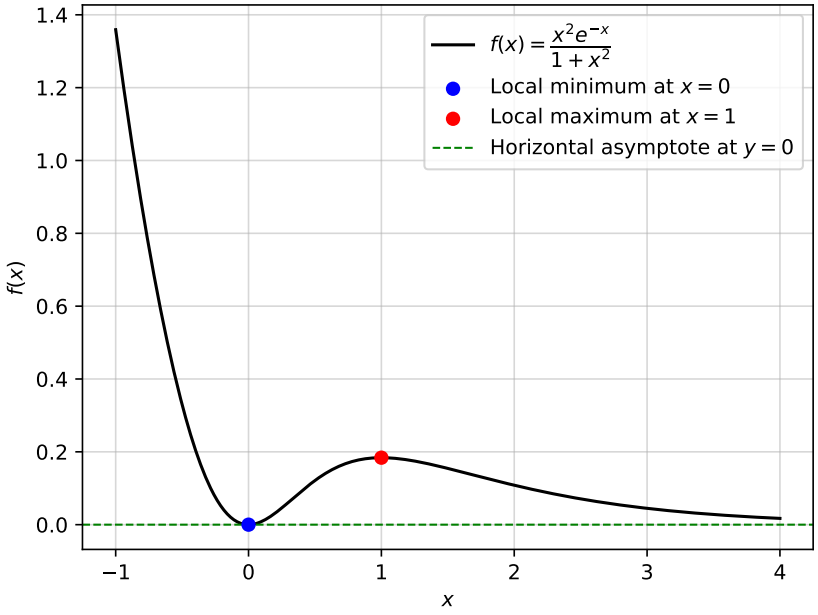


Figure B.1: Plot of the function $f(x) = \dfrac{x^2 e^{-x}}{1+x^2}$

By the definition of the derivative:

$$F'(x) = \lim_{h \to 0} \frac{F(x+h) - F(x)}{h}.$$

Substituting the definition of $F(x)$:

$$F(x+h) = \int_a^{x+h} f(t)\,dt, \quad F(x) = \int_a^x f(t)\,dt.$$

Thus, the difference $F(x+h) - F(x)$ is:

$$F(x+h) - F(x) = \int_a^{x+h} f(t)\,dt - \int_a^x f(t)\,dt.$$

Using the property of definite integrals:

$$\int_a^{x+h} f(t)\,dt - \int_a^x f(t)\,dt = \int_x^{x+h} f(t)\,dt.$$

Therefore,

$$F'(x) = \lim_{h \to 0} \frac{1}{h} \int_x^{x+h} f(t)\, dt.$$

Since $f(t)$ is continuous on $[x, x+h]$, we can apply the Mean Value Theorem for Integrals, which guarantees that there exists a point $c \in (x, x+h)$ such that:

$$\int_x^{x+h} f(t)\, dt = f(c)(x + h - x) = f(c)h.$$

Substituting this into our expression for $F'(x)$:

$$F'(x) = \lim_{h \to 0} \frac{f(c)h}{h} = \lim_{h \to 0} f(c).$$

Since $f(x)$ is continuous on $[a, b]$, and c is chosen in $(x, x+h)$, as $h \to 0$, we have $c \to x$. By the continuity of f at x:

$$\lim_{h \to 0} f(c) = f(x).$$

Thus,

$$F'(x) = f(x).$$

□

B.44 (a) The function f depends on two terms: $\phi(x, z)$ and $2xy$. Let's define $g(x, y) = 2xy$, such that:

$$f(x, y, z) = \cos(\phi(x, z) + g(x, y)).$$

We can write $\phi(x, z)$ explicitly by substituting the given expressions for the intermediate variables:

$$\phi(x, z) = u(x, v) = e^x + v^3 = e^x + (x^2 + \tan z)^3.$$

In order to differentiate f, we apply the multivariable chain rule for each input variable:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial \phi}\frac{\partial \phi}{\partial x} + \frac{\partial f}{\partial g}\frac{\partial g}{\partial x},$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial \phi}\frac{\partial \phi}{\partial y} + \frac{\partial f}{\partial g}\frac{\partial g}{\partial y},$$

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial \phi}\frac{\partial \phi}{\partial z}.$$

From the definition of f:

$$\frac{\partial f}{\partial \phi} = -\sin(\phi + g), \quad \frac{\partial f}{\partial g} = -\sin(\phi + g).$$

Applying the chain rule to $\phi(x, z) = u(x, v(x, z))$:

$$\frac{\partial \phi}{\partial x} = \frac{\partial u}{\partial x} + \frac{\partial u}{\partial v}\frac{\partial v}{\partial x}, \quad \frac{\partial \phi}{\partial z} = \frac{\partial u}{\partial v}\frac{\partial v}{\partial z}.$$

Compute the partial derivatives of $u(x, v) = e^x + v^3$:

$$\frac{\partial u}{\partial x} = e^x, \quad \frac{\partial u}{\partial v} = 3v^2 = 3(x^2 + \tan z)^2.$$

Compute the partial derivatives of $v(x, z) = x^2 + \tan z$:

$$\frac{\partial v}{\partial x} = 2x, \quad \frac{\partial v}{\partial z} = \sec^2 z.$$

Compute the partial derivatives of $g(x, y)$:

$$\frac{\partial g}{\partial x} = 2y, \quad \frac{\partial g}{\partial y} = 2x.$$

Compute the final partial derivatives:

$$\begin{aligned}
\frac{\partial f}{\partial x} &= -\sin(\phi + g)\left(e^x + 6x(x^2 + \tan z)^2 + 2y\right) \\
&= -\sin\left(e^x + (x^2 + \tan z)^3 + 2xy\right)\left(e^x + 6x(x^2 + \tan z)^2 + 2y\right), \\
\frac{\partial f}{\partial y} &= -\sin(\phi + g)\,(2x) \\
&= -2x\sin\left(e^x + (x^2 + \tan z)^3 + 2xy\right), \\
\frac{\partial f}{\partial z} &= -\sin(\phi + g)\left(3(x^2 + \tan z)^2 \sec^2 z\right) \\
&= -3\sin\left(e^x + (x^2 + \tan z)^3 + 2xy\right)(x^2 + \tan z)^2 \sec^2 z.
\end{aligned}$$

(b) First, substitute:

$$\phi(x, z) = e^x + (x^2 + \tan z)^3, \quad g(x, y) = 2xy.$$

Thus:

$$f(x, y, z) = \cos(e^x + (x^2 + \tan z)^3 + 2xy).$$

Now, differentiate directly:

$$\begin{aligned}
\frac{\partial f}{\partial x} &= -\sin(e^x + (x^2 + \tan z)^3 + 2xy)\left(e^x + 6x(x^2 + \tan z)^2 + 2y\right), \\
\frac{\partial f}{\partial y} &= -2x \sin(e^x + (x^2 + \tan z)^3 + 2xy), \\
\frac{\partial f}{\partial z} &= -3 \sin(e^x + (x^2 + \tan z)^3 + 2xy)(x^2 + \tan z)^2 \sec^2 z.
\end{aligned}$$

Since the results match the chain rule computation, our calculations are verified.

B.53 We analyze the function:

$$f(x, y) = x^3 - 3x + y^2 - 3y.$$

(a) The function is a polynomial and is therefore defined for all real numbers:

$$\mathrm{dom}(f) = \mathbb{R}^2.$$

Since $x^3 - 3x$ can take arbitrarily large positive and negative values, the function is unbounded. Therefore, its range is:

$$\mathrm{Im}(f) = \mathbb{R}.$$

(b) To find the critical points, we compute the gradient $\nabla f(x, y) = (f_x, f_y)$ and set it to zero.

The first-order partial derivatives are:

$$\begin{aligned}
f_x &= \frac{\partial}{\partial x}(x^3 - 3x + y^2 - 3y) = 3x^2 - 3, \\
f_y &= \frac{\partial}{\partial y}(x^3 - 3x + y^2 - 3y) = 2y - 3.
\end{aligned}$$

Setting these equal to zero:

$$3x^2 - 3 = 0, \quad 2y - 3 = 0.$$

Solving for x:

$$3x^2 - 3 = 0 \quad \Rightarrow \quad x^2 = 1 \quad \Rightarrow \quad x = \pm 1.$$

Solving for y:

$$2y - 3 = 0 \quad \Rightarrow \quad y = \frac{3}{2}.$$

Thus, the function has two critical points:

$$(1, \frac{3}{2}), \quad (-1, \frac{3}{2}).$$

To classify the critical points, we compute the Hessian matrix is:

$$H_f(x, y) = \begin{pmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{pmatrix} = \begin{pmatrix} 6x & 0 \\ 0 & 2 \end{pmatrix}.$$

To classify the critical points, we apply Sylvester's criterion:

- At $(1, \frac{3}{2})$:

$$H_f(1, \frac{3}{2}) = \begin{pmatrix} 6 & 0 \\ 0 & 2 \end{pmatrix}.$$

$$\det(H_f) = 12 > 0.$$

  Since $f_{xx} = 6 > 0$ and $\det(H_f) > 0$, the Hessian is positive definite, indicating that $(1, \frac{3}{2})$ is a local minimum.
- At $(-1, \frac{3}{2})$:

$$H_f(-1, \frac{3}{2}) = \begin{pmatrix} -6 & 0 \\ 0 & 2 \end{pmatrix}.$$

$$\det(H_f) = -12 < 0.$$

  Since $\det(H_f) < 0$, the Hessian is indefinite, indicating that $(-1, \frac{3}{2})$ is a saddle point.

(c) Since the function contains a cubic term x^3, its behavior for large $|x|$ is dominated by this term. Specifically, as $x \to \pm\infty$, the x^3 term grows faster than any other term, leading to:

$$\lim_{x \to \pm\infty} f(x, y) = \pm\infty.$$

Along the y-axis (i.e., when x is fixed or small compared to y), the dominant term is y^2, which grows quadratically as $|y| \to \infty$.

Therefore, the function does not approach a finite limit in any specific direction instead exhibits unbounded growth, either cubically in x or quadratically in y, depending on which variable is larger in magnitude.

(d) The function grows unbounded as $x \to \pm\infty$ due to the x^3 term, meaning there is no global maximum or minimum.

(e) To identify inflection points, we analyze the second-order and third-order derivatives, looking for regions where concavity changes.

A point (x_0, y_0) is an inflection point if the second derivative changes sign across that point.

The second-order derivative in the x-direction is:

$$f_{xx} = 6x.$$

- For $x < 0$, $f_{xx} = 6x < 0$ (concave down).
- For $x > 0$, $f_{xx} = 6x > 0$ (concave up).
- At $x = 0$, $f_{xx} = 0$, indicating a possible inflection point.

The second-order derivative in the x-direction is:

$$f_{yy} = 2.$$

Since f_{yy} is always positive, the function is always concave up in the y-direction. Therefore, no inflection points exist in the y-direction.

To confirm a concavity change at $x = 0$, we compute the third derivative:

$$f_{xxx} = \frac{\partial}{\partial x}(6x) = 6.$$

Since $f_{xxx} \neq 0$, we conclude that concavity changes at $x = 0$, confirming an inflection point.

Therefore, the function has an infinite set of inflection points along the vertical line $x = 0$:

$$\{(0, y) \mid y \in \mathbb{R}\}.$$

Along this line, the function transitions from concave down to concave up in the x-direction.

(f) The graph of the function is shown in Figure B.2.

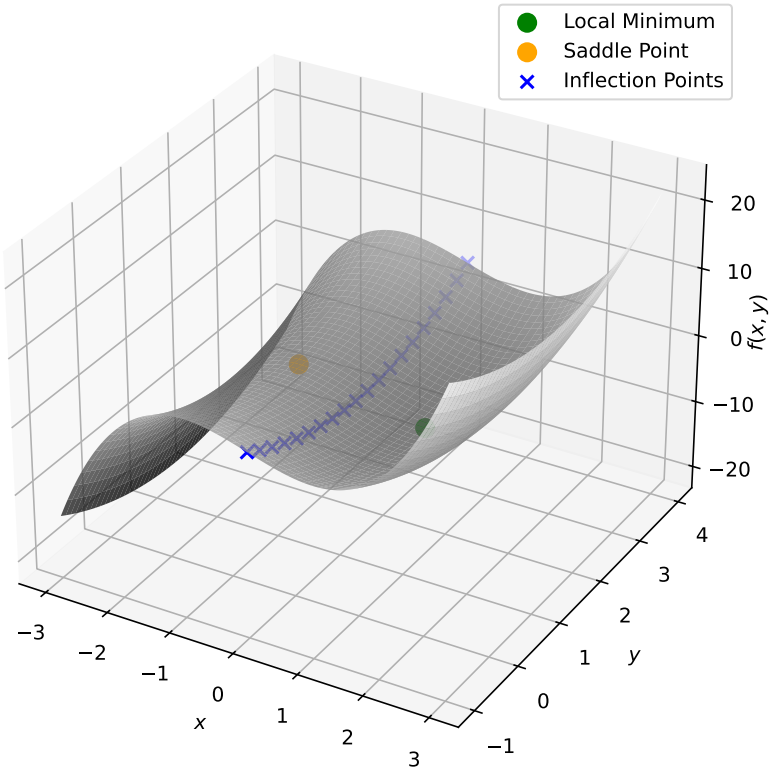


Figure B.2: Plot of the function $f(x, y) = x^3 - 3x + y^2 - 3y$

B.55 The function $f\colon \mathbb{R}^n \to \mathbb{R}^m$ is given by:

$$f(\mathbf{x}) = A\mathbf{x}.$$

Let $\mathbf{a}_1^T, \ldots, \mathbf{a}_m^T$ be the rows of A, so that we can express $A\mathbf{x}$ as:

$$A\mathbf{x} = \begin{pmatrix} \mathbf{a}_1^T\mathbf{x} \\ \vdots \\ \mathbf{a}_m^T\mathbf{x} \end{pmatrix}.$$

The Jacobian matrix is computed as:

$$J_{\mathbf{f}} = \nabla_{\mathbf{x}}(A\mathbf{x}) = \begin{pmatrix} \dfrac{\partial \mathbf{a}_1^T \mathbf{x}}{\partial \mathbf{x}} \\ \vdots \\ \dfrac{\partial \mathbf{a}_m^T \mathbf{x}}{\partial \mathbf{x}} \end{pmatrix}.$$

Since each row $\mathbf{a}_i^T \mathbf{x}$ is a linear function of $\mathbf{x}$, its derivative is simply $\mathbf{a}_i^T$, so we obtain:

$$J_{\mathbf{f}} = \begin{pmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_m^T \end{pmatrix} = A.$$

B.59 Define:

$$I = \int_{-\infty}^{\infty} e^{-x^2}\, dx.$$

Since the function e^{-x^2} has no elementary antiderivative, we compute I^2 as a double integral:

$$I^2 = \left(\int_{-\infty}^{\infty} e^{-x^2}\, dx \right) \left(\int_{-\infty}^{\infty} e^{-y^2}\, dy \right).$$

Writing this as a double integral:

$$I^2 = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-(x^2+y^2)}\, dx\, dy.$$

We convert to polar coordinates using:

$$x = r\cos\theta, \quad y = r\sin\theta.$$

The Jacobian determinant gives:

$$dx\, dy = r\, dr\, d\theta.$$

Rewriting the integral in polar form:

$$I^2 = \int_0^{2\pi} \int_0^{\infty} e^{-r^2} r\, dr\, d\theta.$$

Using the substitution $u = r^2$, so that $du = 2r\,dr$:

$$\int_0^\infty e^{-r^2} r\,dr = \frac{1}{2}\int_0^\infty e^{-u}\,du.$$

Now, evaluating the standard integral:

$$\int_0^\infty e^{-u}\,du = \left[-e^{-u}\right]_0^\infty = -(0-1) = 1.$$

Thus, we obtain:

$$\int_0^\infty e^{-r^2} r\,dr = \frac{1}{2}.$$

Now, integrating over θ:

$$I^2 = \frac{1}{2}\int_0^{2\pi} d\theta = \frac{1}{2}\cdot 2\pi = \pi.$$

Taking the square root:

$$\int_{-\infty}^\infty e^{-x^2}\,dx = \sqrt{\pi}.$$

□

B.60 (a) Since $f(\mathbf{x})$ is a quadratic function of $\mathbf{g}(\mathbf{x})$ and A is symmetric, we use the differentiation rule:

$$\nabla_{\mathbf{g}} f = 2A\mathbf{g}(\mathbf{x}).$$

Substituting the values of A and $\mathbf{g}(\mathbf{x})$:

$$\nabla_{\mathbf{g}} f = 2\begin{pmatrix}3 & 1\\ 1 & 2\end{pmatrix}\begin{pmatrix}x_1^2 + x_2\\ \sin x_1 + x_2^2\end{pmatrix}.$$

Computing the matrix-vector product:

$$\nabla_{\mathbf{g}} f = 2\begin{pmatrix}3(x_1^2 + x_2) + 1(\sin x_1 + x_2^2)\\ 1(x_1^2 + x_2) + 2(\sin x_1 + x_2^2)\end{pmatrix} = \begin{pmatrix}6x_1^2 + 6x_2 + 2\sin x_1 + 2x_2^2\\ 2x_1^2 + 2x_2 + 4\sin x_1 + 4x_2^2\end{pmatrix}.$$

The Jacobian matrix of $\mathbf{g}(\mathbf{x})$ is:

$$J_{\mathbf{g}}(\mathbf{x}) = \begin{pmatrix}\frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2}\\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2}\end{pmatrix}.$$

Computing the partial derivatives:

- For $g_1(x) = x_1^2 + x_2$:

$$\frac{\partial g_1}{\partial x_1} = 2x_1, \quad \frac{\partial g_1}{\partial x_2} = 1.$$

- For $g_2(x) = \sin x_1 + x_2^2$:

$$\frac{\partial g_2}{\partial x_1} = \cos x_1, \quad \frac{\partial g_2}{\partial x_2} = 2x_2.$$

Thus, the Jacobian matrix is:

$$J_{\mathbf{g}}(\mathbf{x}) = \begin{pmatrix} 2x_1 & 1 \\ \cos x_1 & 2x_2 \end{pmatrix}.$$

Using the Jacobian chain rule:

$$\nabla_{\mathbf{x}} f = J_{\mathbf{g}}^T \nabla_{\mathbf{g}} f.$$

Substituting our values:

$$\nabla_{\mathbf{x}} f = \begin{pmatrix} 2x_1 & \cos x_1 \\ 1 & 2x_2 \end{pmatrix} \cdot \begin{pmatrix} 6x_1^2 + 6x_2 + 2\sin x_1 + 2x_2^2 \\ 2x_1^2 + 2x_2 + 4\sin x_1 + 4x_2^2 \end{pmatrix}.$$

Thus, the gradient $\nabla_{\mathbf{x}} f$ is:

$$\begin{pmatrix} 12x_1^3 + 12x_1x_2 + 4x_1\sin x_1 + 4x_1x_2^2 + 2x_1^2\cos x_1 + 2x_2\cos x_1 + 4\sin x_1\cos x_1 + 4x_2^2\cos x_1 \\ 6x_1^2 + 6x_2 + 2\sin x_1 + 2x_2^2 + 4x_1^2x_2 + 4x_2^2 + 8x_2\sin x_1 + 8x_2^3 \end{pmatrix}.$$

(b) Expanding the function explicitly:

$$f(\mathbf{x}) = 3(x_1^2 + x_2)^2 + 2(x_1^2 + x_2)(\sin x_1 + x_2^2) + 2(\sin x_1 + x_2^2)^2.$$

Computing $\dfrac{\partial f}{\partial x_1}$:

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= 12x_1(x_1^2 + x_2) + 4x_1(\sin x_1 + x_2^2) + 2(x_1^2 + x_2)\cos x_1 + 4(\sin x_1 + x_2^2)\cos x_1 \\ &= 12x_1^3 + 12x_1x_2 + 4x_1\sin x_1 + 4x_1x_2^2 + 2x_1^2\cos x_1 + 2x_2\cos x_1 + 4\sin x_1\cos x_1 + 4x_2^2\cos x_1. \end{aligned}$$

Computing $\dfrac{\partial f}{\partial x_2}$:

$$\frac{\partial f}{\partial x_2} = 6x_1^2 + 6x_2 + 2\sin x_1 + 2x_2^2 + 4x_1^2x_2 + 4x_2^2 + 8x_2\sin x_1 + 8x_2^3.$$

This matches the result from matrix calculus, confirming correctness.

B.63 (a) Expanding $f(\mathbf{x})$ as a sum:

$$f(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i A_{ij} x_j.$$

Differentiating with respect to x_k gives:

$$\frac{\partial f}{\partial x_k} = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial}{\partial x_k}(x_i A_{ij} x_j).$$

Each term in the sum $x_i A_{ij} x_j$ contributes based on whether $i = k$, $j = k$, or both:

- When $i = k$, but $j \neq k$:

$$\frac{\partial}{\partial x_k}(x_k A_{kj} x_j) = A_{kj} x_j.$$

- When $j = k$, but $i \neq k$:

$$\frac{\partial}{\partial x_k}(x_i A_{ik} x_k) = A_{ik} x_i.$$

- When $i = k$ and $j = k$:

$$\frac{\partial}{\partial x_k}(x_k A_{kk} x_k) = \frac{\partial}{\partial x_k}(A_{kk} x_k^2) = 2A_{kk} x_k.$$

Combining all the contributions, we obtain:

$$\frac{\partial f}{\partial x_k} = \sum_{j=1}^{n} A_{kj} x_j + \sum_{i=1}^{n} A_{ik} x_i.$$

where the case $i = k, j = k$ is automatically included because it contributes $2A_{kk}x_k$, which appears in both sums.

Now, we express the partial derivatives in terms of matrix-vector products. We note that the two summations above correspond to the k-th entries of the matrix-vector products:

$$(A\mathbf{x})_k = \sum_{j=1}^{n} A_{kj} x_j, \quad (A^T\mathbf{x})_k = \sum_{i=1}^{n} A_{ik} x_i.$$

Thus, we can rewrite the partial derivative as:

$$\frac{\partial f}{\partial x_k} = (A\mathbf{x})_k + (A^T\mathbf{x})_k.$$

Since this holds for all k, we conclude that the gradient vector is:

$$\nabla_{\mathbf{x}} f = (A + A^T)\mathbf{x}.$$

(b) We begin by defining an auxiliary variable $\mathbf{y} = A\mathbf{x}$.
Rewriting the function in terms of $\mathbf{y}$:

$$f(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} = \mathbf{x}^T\mathbf{y}.$$

Thus, we can express f as:

$$f(\mathbf{x}) = \phi(\mathbf{x}, \mathbf{y}),$$

where $\phi(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$ and $\mathbf{y} = A\mathbf{x}$.
Since f depends on $\mathbf{x}$ both directly and indirectly via $\mathbf{y}$, the gradient of f with respect to $\mathbf{x}$ is given by the gradient chain rule:

$$\frac{\partial f}{\partial \mathbf{x}} = \underbrace{\frac{\partial \phi}{\partial \mathbf{x}}}_{(1)} + \underbrace{\frac{\partial \phi}{\partial \mathbf{y}}}_{(2)} \underbrace{\frac{\partial \mathbf{y}}{\partial \mathbf{x}}}_{(3)}. \tag{B.2}$$

We now compute each term separately:

- First term: $\frac{\partial \phi}{\partial \mathbf{x}}$.
  Using the gradient of a linear function,

  $$\nabla_{\mathbf{x}}(\mathbf{u}^T\mathbf{x}) = \mathbf{u},$$

  we obtain:

  $$\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{y}) = \mathbf{y}^T.$$

  Here, the gradient is expressed as a row vector, consistent with the convention that $\frac{\partial f}{\partial \mathbf{x}}$ is a row vector in matrix calculus.
- Second term: $\frac{\partial \phi}{\partial \mathbf{y}}$.
  Differentiating $\mathbf{x}^T\mathbf{y}$ with respect to $\mathbf{y}$, treating $\mathbf{x}$ as constant, yields:

  $$\frac{\partial}{\partial \mathbf{y}}(\mathbf{x}^T\mathbf{y}) = \mathbf{x}^T.$$

- Third term: $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$.
  Since $\mathbf{y} = A\mathbf{x}$, its Jacobian is simply:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = A.$$

Substituting these results back into the chain rule:

$$\frac{\partial f}{\partial \mathbf{x}} = \mathbf{y}^T + \mathbf{x}^T A = \mathbf{x}^T A + (A\mathbf{x})^T.$$

Since the gradient $\nabla_{\mathbf{x}} f$ is conventionally a column vector, we take the transpose:

$$\nabla_{\mathbf{x}} f = \left(\frac{\partial f}{\partial \mathbf{x}}\right)^T = A\mathbf{x} + A^T\mathbf{x}.$$

Thus, the final result is:

$$\nabla_{\mathbf{x}} f = (A + A^T)\mathbf{x}.$$

B.63 We differentiate the squared loss function:

$$L(\mathbf{x}) = \frac{1}{2}\|A\mathbf{x} - \mathbf{b}\|^2.$$

Recalling that the squared norm of a vector $\mathbf{z}$ is given by $\|\mathbf{z}\|^2 = \mathbf{z}^T\mathbf{z}$, we rewrite $L(\mathbf{x})$ as:

$$L(\mathbf{x}) = \frac{1}{2}(A\mathbf{x} - \mathbf{b})^T(A\mathbf{x} - \mathbf{b}).$$

Define $\mathbf{y} = A\mathbf{x} - \mathbf{b}$, so that:

$$L(\mathbf{x}) = \frac{1}{2}\mathbf{y}^T\mathbf{y}.$$

Applying the gradient chain rule:

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \mathbf{x}}.$$

For the first term, we use the gradient rule for the squared norm:

$$\frac{\partial L}{\partial \mathbf{y}} = \frac{\partial}{\partial \mathbf{y}}\left(\frac{1}{2}\mathbf{y}^T\mathbf{y}\right) = \mathbf{y}^T.$$

For the second term, we differentiate $\mathbf{y} = A\mathbf{x} - \mathbf{b}$ with respect to $\mathbf{x}$. Using the linearity of differentiation and applying the Linear Transformation Rule (see Section B.9.3), we obtain:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial (A\mathbf{x} - \mathbf{b})}{\partial \mathbf{x}} = \frac{\partial A\mathbf{x}}{\partial \mathbf{x}} - \frac{\partial \mathbf{b}}{\partial \mathbf{x}} = A - 0 = A.$$

Substituting these results:

$$\frac{\partial L}{\partial \mathbf{x}} = \mathbf{y}^T A = (A\mathbf{x} - \mathbf{b})^T A.$$

Since the gradient is conventionally written as a column vector, we take the transpose:

$$\nabla_{\mathbf{x}} L = A^T (A\mathbf{x} - \mathbf{b}).$$

B.72 We begin with the well-known Maclaurin series for e^x:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

To obtain the Maclaurin series for e^{-2x^2}, we substitute x with $-2x^2$:

$$e^{-2x^2} = \sum_{n=0}^{\infty} \frac{(-2x^2)^n}{n!} = \sum_{n=0}^{\infty} \frac{(-2)^n x^{2n}}{n!}.$$

Now, multiplying both sides by x^3, we obtain:

$$f(x) = x^3 e^{-2x^2} = x^3 \sum_{n=0}^{\infty} \frac{(-2)^n x^{2n}}{n!} = \sum_{n=0}^{\infty} \frac{(-2)^n}{n!} x^{2n+3}.$$

Extracting the first four terms explicitly:

$$\begin{aligned} f(x) &= \frac{(-2)^0}{0!} x^3 + \frac{(-2)^1}{1!} x^5 + \frac{(-2)^2}{2!} x^7 + \frac{(-2)^3}{3!} x^9 + \dots \\ &= x^3 - 2x^5 + \frac{4}{2} x^7 - \frac{8}{6} x^9 + \dots \\ &= x^3 - 2x^5 + 2x^7 - \frac{4}{3} x^9 + \dots \end{aligned}$$

B.74 We begin with the well-known Maclaurin series for e^x:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

Multiplying both sides by $(x+1)$:

$$\begin{aligned}
(x+1)e^x &= (x+1)\sum_{n=0}^{\infty} \frac{x^n}{n!} \\
&= x\sum_{n=0}^{\infty} \frac{x^n}{n!} + \sum_{n=0}^{\infty} \frac{x^n}{n!} \\
&= \sum_{n=0}^{\infty} \frac{x^{n+1}}{n!} + \sum_{n=0}^{\infty} \frac{x^n}{n!} \\
&= \sum_{n=1}^{\infty} \frac{x^n}{(n-1)!} + \left(1 + \sum_{n=1}^{\infty} \frac{x^n}{n!}\right) \\
&= 1 + \sum_{n=1}^{\infty} \left(\frac{1}{(n-1)!} + \frac{1}{n!}\right) x^n \\
&= 1 + \sum_{n=1}^{\infty} \frac{n+1}{n!} x^n \\
&= \sum_{n=0}^{\infty} \frac{n+1}{n!} x^n.
\end{aligned}$$

B.80 (a) We derive the Maclaurin series for $f(x) = \frac{1}{1+x}$ by rewriting it as a geometric series:

$$f(x) = \frac{1}{1+x} = \frac{1}{1-(-x)}.$$

This is the geometric series expansion of $\frac{1}{1-r}$ with $r = -x$, which converges for $|x| < 1$:

$$\frac{1}{1+x} = \sum_{n=0}^{\infty} (-1)^n x^n.$$

(b) The third-degree Taylor polynomial is:

$$P_3(x) = 1 - x + x^2 - x^3.$$

Evaluate at $x = 0.2$:

$$P_3(0.2) = 1 - 0.2 + 0.2^2 - 0.2^3 = 0.832.$$

(c) We compute the actual value of the function at $x = 0.2$:

$$f(0.2) = \frac{1}{1+0.2} = \approx 0.8333.$$

Thus, the actual approximation error is:

$$\text{Error} = |f(0.2) - P_3(0.2)| = |0.8333 - 0.832| = 0.0013.$$

(d) To estimate the error, we use the remainder formula:

$$R_3(x) = \frac{f^{(4)}(\xi)}{4!}x^4 \quad \text{for some } \xi \in (0, x).$$

Since

$$f^{(4)}(x) = (-1)^4 \cdot 4! \cdot (1+x)^{-5} = 4! \cdot (1+x)^{-5},$$

we obtain

$$|R_3(0.2)| = \left| \frac{(0.2)^4}{(1+\xi)^5} \right|, \quad \text{for some } \xi \in (0, 0.2).$$

To bound the error, we observe that for $\xi \in (0, 0.2)$, the quantity $(1+\xi)^5$ is minimized when $\xi = 0$. Therefore,

$$|R_3(0.2)| \leq \frac{(0.2)^4}{(1+0)^5} = 0.0016.$$

This shows that the third-degree Taylor polynomial approximates $f(0.2)$ with an error less than 0.0016, which is consistent with the actual approximation error of 0.0013 computed in part (c).

(e) We want the smallest n such that:

$$|R_n(0.2)| = |x^{n+1}| < 10^{-4}.$$

So we solve:

$$(0.2)^{n+1} < 10^{-4}.$$

Trying different values of n:

$$(0.2)^5 = 3.2 \times 10^{-4}, \quad (0.2)^6 = 6.4 \times 10^{-5}.$$

So $n + 1 = 6 \Rightarrow n = 5$.

Therefore, at least 6 terms (up to degree 5) are needed to ensure the error is below 10^{-4} at $x = 0.2$.

B.83 We prove that the function $f(x) = e^x$ is strictly convex using three different methods:

(a) Using the definition of convexity.
A function f is strictly convex if for any two points x_1, x_2 and for any $\lambda \in (0, 1)$, we have:

$$f(\lambda x_1 + (1 - \lambda)x_2) < \lambda f(x_1) + (1 - \lambda)f(x_2).$$

Let's apply this to $f(x) = e^x$. The left side of the inequality is:

$$f(\lambda x_1 + (1 - \lambda)x_2) = e^{\lambda x_1 + (1-\lambda)x_2} = e^{\lambda x_1} \cdot e^{(1-\lambda)x_2}.$$

Now, let's look at the right side of the inequality:

$$\lambda f(x_1) + (1 - \lambda)f(x_2) = \lambda e^{x_1} + (1 - \lambda)e^{x_2}.$$

To prove strict convexity, we need to show:

$$e^{\lambda x_1} \cdot e^{(1-\lambda)x_2} < \lambda e^{x_1} + (1 - \lambda)e^{x_2}.$$

We apply the weighted arithmetic mean-geometric mean (AM-GM) inequality, which states that for positive real numbers a, b and weights $\alpha, \beta > 0$ that sum to 1, we have:

$$a^\alpha b^\beta \leq \alpha a + \beta b,$$

with equality if and only if $a = b$.
Applying this to our function, we set:

$$a = e^{x_1}, \quad b = e^{x_2}, \quad \alpha = \lambda, \quad \beta = 1 - \lambda.$$

These choices satisfy the conditions of AM-GM: exponentials are always positive, and the weights sum to 1 since $\lambda + (1 - \lambda) = 1$.
Thus, the AM-GM inequality gives:

$$e^{\lambda x_1} \cdot e^{(1-\lambda)x_2} \leq \lambda e^{x_1} + (1 - \lambda)e^{x_2}.$$

Since $x_1 \neq x_2$, we have $e^{x_1} \neq e^{x_2}$, so the inequality is strict:

$$e^{\lambda x_1 + (1-\lambda)x_2} < \lambda e^{x_1} + (1 - \lambda)e^{x_2}.$$

Therefore, $f(x) = e^x$ is strictly convex.

(b) Using the first-order condition of convexity.
A differentiable function $f(x)$ is strictly convex if and only if, for all $x_1, x_2 \in \mathbb{R}$ with $x_1 \neq x_2$:

$$f(x_2) > f(x_1) + f'(x_1)(x_2 - x_1).$$

The derivative of $f(x) = e^x$ is the same as the function itself $f'(x) = e^x$.
Applying the first-order condition at x_1, we need to show:

$$e^{x_2} > e^{x_1} + e^{x_1}(x_2 - x_1).$$

Define $t = x_2 - x_1$, so that $x_2 = x_1 + t$. Substituting this into the inequality:

$$e^{x_1+t} > e^{x_1} + e^{x_1}t.$$

Factoring e^{x_1} out of the left-hand side:

$$e^{x_1}e^t > e^{x_1} + e^{x_1}t.$$

Dividing both sides by e^{x_1} (which is always positive):

$$e^t > 1 + t.$$

The inequality:

$$e^t \geq 1 + t, \quad \forall t \in \mathbb{R},$$

is a well-known tangent line approximation of e^x at $x = 0$. This inequality holds for all t, and is strict for $t \neq 0$, meaning:

$$e^t > 1 + t, \quad \forall t \neq 0.$$

Since $t = x_2 - x_1 \neq 0$, we conclude:

$$e^{x_2} > e^{x_1} + e^{x_1}(x_2 - x_1).$$

This confirms that the function $f(x) = e^x$ satisfies the strict first-order convexity condition, proving that e^x is strictly convex.

(c) Using the second-order condition of convexity.
A twice-differentiable function $f(x)$ is strictly convex if:

$$f''(x) > 0, \quad \text{for all } x.$$

For $f(x) = e^x$, we compute:

$$f'(x) = e^x, \quad f''(x) = e^x.$$

Since $e^x > 0$ for all x, the function is strictly convex.

B.86 Assume for contradiction that x^* is a local minimum but not a global minimum. Then, there exists a point x' such that:

$$f(x') < f(x^*).$$

Since x^* is a local minimum, there exists $\epsilon > 0$ such that for all y satisfying $\|y - x^*\| < \epsilon$, we have:

$$f(y) \geq f(x^*).$$

Next, consider the point:

$$x_\lambda = (1 - \lambda)x^* + \lambda x',$$

where we define λ as:

$$\lambda = \frac{\epsilon}{\|x^* - x'\|}.$$

Choosing ϵ such that $\epsilon < \|x^* - x'\|$ ensures that $\lambda \in (0, 1)$, which guarantees that x_λ is a convex combination of x^* and x'. Applying the convexity of f, we obtain:

$$f(x_\lambda) \leq (1 - \lambda)f(x^*) + \lambda f(x').$$

Since we assumed $f(x') < f(x^*)$, it follows that:

$$f(x_\lambda) < (1 - \lambda)f(x^*) + \lambda f(x^*) = f(x^*).$$

However, x_λ is within the ϵ-neighborhood of x^*, since:

$$\|x_\lambda - x^*\| = \lambda\|x^* - x'\| = \epsilon.$$

This contradicts the assumption that x^* is a local minimum, since x_λ is within the ϵ-neighborhood of x^* but has a strictly lower function value.

Thus, our assumption must be false, meaning that no such x' can exist. Therefore, x^* is a global minimum.

B.91 We aim to prove the Power Mean Inequality:

$$\left(\frac{x_1^p + x_2^p + \cdots + x_n^p}{n}\right)^{\frac{1}{p}} \geq \left(\frac{x_1^q + x_2^q + \cdots + x_n^q}{n}\right)^{\frac{1}{q}}$$

for any positive numbers $x_1, x_2, \ldots, x_n$ and for exponents $p > q > 0$.

Consider the function:

$$f(x) = x^r, \quad \text{where } x > 0 \text{ and } r = \frac{p}{q} > 1.$$

To verify convexity, we compute its second derivative:

$$f''(x) = r(r-1)x^{r-2}.$$

Since $r > 1$ and $x > 0$, we have $f''(x) > 0$, which confirms that $f(x)$ is convex on $(0, \infty)$.

Jensen's inequality states that for a convex function f, we have:

$$f\left(\frac{x_1 + x_2 + \cdots + x_n}{n}\right) \leq \frac{f(x_1) + f(x_2) + \cdots + f(x_n)}{n}.$$

Applying this to $f(x) = x^r$ with the values x_i^q, we obtain:

$$\left(\frac{x_1^q + x_2^q + \cdots + x_n^q}{n}\right)^r \leq \frac{(x_1^q)^r + (x_2^q)^r + \cdots + (x_n^q)^r}{n}.$$

Since $p = qr$, we can rewrite the right-hand side as:

$$\left(\frac{x_1^q + x_2^q + \cdots + x_n^q}{n}\right)^r \leq \frac{x_1^p + x_2^p + \cdots + x_n^p}{n}.$$

Taking the p-th root on both sides gives:

$$\left(\frac{x_1^q + x_2^q + \cdots + x_n^q}{n}\right)^{\frac{r}{p}} \leq \left(\frac{x_1^p + x_2^p + \cdots + x_n^p}{n}\right)^{\frac{1}{p}}.$$

Since $\frac{r}{p} = \frac{1}{q}$, we rewrite it as:

$$\left(\frac{x_1^p + x_2^p + \cdots + x_n^p}{n}\right)^{\frac{1}{p}} \geq \left(\frac{x_1^q + x_2^q + \cdots + x_n^q}{n}\right)^{\frac{1}{q}}.$$

Thus, the Power Mean Inequality is proved. □

B.92 (a) $f(x) = |x| + x^2$ at $x = 0$.

The function is defined as:

$$f(x) = \begin{cases} x + x^2, & x \geq 0, \\ -x + x^2, & x < 0. \end{cases}$$

This function is continuous everywhere but is not differentiable at $x = 0$ because the left and right derivatives do not agree.

By definition, the subdifferential of f at $x = 0$ consists of all values g satisfying:

$$f(y) \geq f(0) + g(y - 0), \quad \forall y \in \mathbb{R}.$$

Since $f(0) = 0$, this simplifies to:

$$|y| + y^2 \geq gy, \quad \forall y \in \mathbb{R}.$$

To determine the values of g that satisfy this inequality for all x, we consider two cases:

- For $y > 0$: We substitute $f(y) = y + y^2$:

$$\begin{aligned} y + y^2 &\geq gy, \\ y + y^2 - gy &\geq 0, \\ y(1 + y - g) &\geq 0. \end{aligned}$$

  Since $y > 0$, this inequality holds if and only if:

$$1 + y - g \geq 0, \quad \forall y > 0,$$

  or equivalently:

$$g \leq 1 + y, \quad \forall y > 0.$$

  Thus, for the inequality to hold for all $y > 0$, we require:

$$g \leq 1.$$

- For $y < 0$: We substitute $f(y) = -y + y^2$:

$$\begin{aligned} -y + y^2 &\geq gy, \\ -y + y^2 - gy &\geq 0, \\ y(-1 + y - g) &\geq 0. \end{aligned}$$

Since $y < 0$, this inequality holds if and only if:

$$-1 + y - g \leq 0, \quad \forall y < 0,$$

or equivalently:

$$g \geq y - 1, \quad \forall y < 0.$$

Thus, for the inequality to hold for all $y < 0$, we require:

$$g \geq -1.$$

Combining both cases:

$$-1 \leq g \leq 1.$$

Thus, the subdifferential of $f(x)$ at $x = 0$ is:

$$\partial f(0) = [-1, 1].$$

(b) $f(x) = \max(x, 0)$ at $x = 0$.
The function is defined as:

$$f(x) = \begin{cases} x, & x \geq 0, \\ 0, & x < 0. \end{cases}$$

This function is piecewise linear and is not differentiable at $x = 0$ because the left and right derivatives do not agree.
By definition, the subdifferential of f at $x = 0$ consists of all values g satisfying:

$$f(x) \geq f(0) + g(x - 0), \quad \forall x \in \mathbb{R}.$$

Since $f(0) = 0$, this simplifies to:

$$\max(x, 0) \geq gx, \quad \forall x \in \mathbb{R}.$$

To determine the values of g that satisfy this inequality for all x, we consider two cases:

- For $x > 0$, we have $f(x) = x$, so the inequality becomes:

$$x \geq gx.$$

Dividing both sides by $x > 0$, we obtain:

$$1 \geq g.$$

- For $x < 0$, we have $f(x) = 0$, so the inequality becomes:

$$0 \geq gx.$$

  Dividing both sides by $x < 0$ reverses the inequality:

$$g \geq 0.$$

Since g must satisfy both conditions, it follows that:

$$0 \leq g \leq 1.$$

Thus, the subdifferential of $f(x)$ at $x = 0$ is:

$$\partial f(0) = [0, 1].$$

Appendix C

Probability Theory

C.2 (a) To find $P(A \cap B)$, we use the inclusion-exclusion formula:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B).$$

Thus,

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) = \frac{1}{3} + \frac{1}{4} - \frac{1}{2} = \frac{1}{12}.$$

(b) The events A, B, and C are not disjoint, since $P(A \cap B) = \frac{1}{12}$, and therefore $A \cap B \neq \emptyset$.

(c) We start by using the set-theoretic identity:

$$C - (A \cup B) = (C \cup (A \cup B)) - (A \cup B).$$

This identity holds because removing $A \cup B$ from C yields the same result as removing $A \cup B$ from the larger set $C \cup (A \cup B)$; the additional elements in the union are already being removed by the set difference.

Now, since we are given that $A \cup B \cup C = \Omega$, it follows that:

$$C - (A \cup B) = \Omega - (A \cup B) = (A \cup B)^c.$$

Thus,

$$P(C - (A \cup B)) = P((A \cup B)^c) = 1 - P(A \cup B) = \frac{1}{2}.$$

(d) We use the set-theoretic identity:

$$(A \cup B) \cap C = (A \cap C) \cup (B \cap C).$$

Applying the inclusion-exclusion formula:

$$P((A \cap C) \cup (B \cap C)) = P(A \cap C) + P(B \cap C) - P((A \cap C) \cap (B \cap C)).$$

Since $(A \cap C) \cap (B \cap C) = A \cap B \cap C$, we obtain:

$$P((A \cup B) \cap C) = P(A \cap C) + P(B \cap C) - P(A \cap B \cap C).$$

Plugging the given values:

$$P((A \cup B) \cap C) = \frac{1}{10} + \frac{1}{20} - 0 = \frac{3}{20}.$$

C.7 Let X be a random variable such that $a \leq X \leq b$. Define a new random variable

$$Y = X - \frac{a+b}{2}.$$

Since $X \in [a, b]$, we have

$$Y \in \left[a - \frac{a+b}{2},\ b - \frac{a+b}{2}\right] = \left[-\frac{b-a}{2},\ \frac{b-a}{2}\right],$$

so

$$Y^2 \leq \left(\frac{b-a}{2}\right)^2.$$

Therefore,

$$\mathrm{Var}(X) = \mathrm{Var}(Y) = \mathbb{E}[Y^2] - (\mathbb{E}[Y])^2 \leq \mathbb{E}[Y^2] \leq \left(\frac{b-a}{2}\right)^2 = \frac{(b-a)^2}{4}.$$

C.10 Let X be the number of matches between your chosen 6 numbers and the 6 winning numbers. Then X follows a hypergeometric distribution:

$$X \sim \text{Hypergeometric}(N = 49, K = 6, n = 6),$$

where:

- $N = 49$ is the total number of items,
- $K = 6$ is the number of "successes" in the population (the 6 winning numbers),
- $n = 6$ is the number of draws (your chosen numbers).

(a) The probability of getting exactly 4 matches is given by the hypergeometric PMF:

$$P(X = 4) = \frac{\binom{6}{4}\binom{43}{2}}{\binom{49}{6}} = \frac{15 \cdot 903}{13{,}983{,}816} \approx 0.000969.$$

(b) The probability of getting at least 3 matches is:

$$\begin{aligned} P(X \geq 3) &= P(X = 3) + P(X = 4) + P(X = 5) + P(X = 6) \\ &= \frac{\binom{6}{3}\binom{43}{3}}{\binom{49}{6}} + \frac{\binom{6}{4}\binom{43}{2}}{\binom{49}{6}} + \frac{\binom{6}{5}\binom{43}{1}}{\binom{49}{6}} + \frac{\binom{6}{6}\binom{43}{0}}{\binom{49}{6}} \\ &= 0.01765 + 0.000969 + 0.0000185 + 0.0000000715 \approx 0.01864. \end{aligned}$$

(c) The expected value of a hypergeometric distribution is:

$$\mathbb{E}[X] = n \cdot \frac{K}{N} = 6 \cdot \frac{6}{49} \approx 0.735.$$

(d) The variance of a hypergeometric distribution is:

$$\mathrm{Var}(X) = n \cdot \frac{K}{N} \cdot \frac{N-K}{N} \cdot \frac{N-n}{N-1} = 6 \cdot \frac{6}{49} \cdot \frac{43}{49} \cdot \frac{43}{48} \approx 0.578.$$

Thus, the standard deviation is:

$$\sigma_X = \sqrt{\mathrm{Var}(X)} \approx \sqrt{0.578} \approx 0.76.$$

C.12 (a) To find the constant c, we use the fact that the total area under the PDF must equal 1:

$$\int_{-\infty}^{\infty} f_X(x)\,dx = 1.$$

Computing the integral:

$$\int_{-\infty}^{\infty} f_X(x)\,dx = \int_{-1}^{1} c(1-x^2)\,dx$$
$$= c\left[x - \frac{x^3}{3}\right]_{-1}^{1} = c\left(1 - \frac{1}{3} - \left(-1 + \frac{1}{3}\right)\right) = c \cdot \frac{4}{3}.$$

Thus, we must have:

$$c \cdot \frac{4}{3} = 1 \;\Rightarrow\; c = \frac{3}{4}.$$

(b) To compute $\mathbb{E}[X]$, we evaluate:

$$\mathbb{E}[X] = \int_{-1}^{1} x f_X(x)\,dx = \int_{-1}^{1} x \cdot \frac{3}{4}(1-x^2)\,dx = \frac{3}{4}\int_{-1}^{1}(x - x^3)\,dx.$$

Since both x and x^3 are odd functions over a symmetric interval:

$$\int_{-1}^{1} x\,dx = 0, \quad \int_{-1}^{1} x^3\,dx = 0 \quad \mathbb{E}[X] = 0.$$

Next, we compute $\mathbb{E}[X^2]$:

$$\mathbb{E}[X^2] = \int_{-1}^{1} x^2 f_X(x)\,dx = \int_{-1}^{1} x^2 \cdot \frac{3}{4}(1-x^2)\,dx = \frac{3}{4}\int_{-1}^{1}(x^2 - x^4)\,dx$$
$$= \frac{3}{4}\left(\int_{-1}^{1} x^2\,dx - \int_{-1}^{1} x^4\,dx\right) = \frac{3}{4}\left(2\int_{0}^{1} x^2\,dx - 2\int_{0}^{1} x^4\,dx\right)$$
$$= \frac{3}{4}\left(2 \cdot \frac{1}{3} - 2 \cdot \frac{1}{5}\right) = \frac{3}{4} \cdot \frac{4}{15} = \frac{1}{5}.$$

Thus, the variance is:

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 = \frac{1}{5} - 0^2 = \frac{1}{5}.$$

(c) To compute $P(X \geq 0.5)$, we integrate the PDF:

$$P(X \geq 0.5) = \int_{0.5}^{1} \frac{3}{4}(1-x^2)\,dx = \frac{3}{4}\int_{0.5}^{1}(1-x^2)\,dx = \frac{3}{4}\left[x - \frac{x^3}{3}\right]_{0.5}^{1}$$
$$= \frac{3}{4}\left(\left(1 - \frac{1}{3}\right) - \left(0.5 - \frac{(0.5)^3}{3}\right)\right) = \frac{5}{32} = 0.15625.$$

C.14 (a) To find $P(X > 4)$, we standardize the normal variable $X \sim \mathcal{N}(1, 9)$. Note that the standard deviation is $\sqrt{9} = 3$, so:

$$P(X > 4) = P\left(\frac{X-1}{3} > \frac{4-1}{3}\right) = P(Z > 1),$$

where $Z \sim \mathcal{N}(0, 1)$. Using standard normal tables:

$$P(Z > 1) = 1 - \Phi(1) \approx 1 - 0.8413 = 0.1587.$$

(b) To find $P(-4 < Y < 2)$, we first determine the distribution of $Y = 5 - 3X$. Since $X \sim \mathcal{N}(1, 9)$, we know:

$$Y \sim \mathcal{N}(5 - 3 \cdot 1, 9 \cdot 3^2) = \mathcal{N}(2, 81).$$

Standardizing:

$$P(-4 < Y < 2) = P\left(\frac{-4-2}{9} < Z < \frac{2-2}{9}\right) = P\left(-\frac{2}{3} < Z < 0\right),$$

where $Z \sim \mathcal{N}(0, 1)$. From standard normal tables:

$$P\left(-\frac{2}{3} < Z < 0\right) = P\left(0 < Z < \frac{2}{3}\right) = \Phi\left(\frac{2}{3}\right) - \Phi(0) \approx 0.7475 - 0.5 = 0.2475.$$

C.18 Let X be the amount of time (in hours) until the first email arrives. We are told that emails arrive according to a Poisson process with rate λ, meaning the number of arrivals in time t is Poisson distributed:

$$P(Y = k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}, \quad k = 0, 1, 2, \ldots$$

To find the distribution of X, we compute the probability that no emails arrive in the interval $[0, t]$. That is:

$$P(X > t) = P(\text{no emails arrive in } [0, t]) = P(Y = 0) = e^{-\lambda t}.$$

Therefore, the cumulative distribution function (CDF) of X is:

$$F_X(t) = P(X \leq t) = 1 - P(X > t) = 1 - e^{-\lambda t}, \quad t \geq 0.$$

This is the CDF of an exponential distribution with rate parameter λ, so:

$$X \sim \text{Exponential}(\lambda).$$

C.22 Let us define the following events:

- S: the email is spam
- F: the email is flagged as spam

From the problem statement:

- $P(S) = 0.2$, $P(S^c) = 0.8$
- $P(F|S) = 0.9$
- $P(F|S^c) = 0.1$

(a) We are asked for $P(F, S^c)$:

$$P(F, S^c) = P(S^c) \cdot P(F|S^c) = 0.8 \cdot 0.1 = 0.08.$$

(b) Using Bayes' theorem, we compute:

$$P(S|F) = \frac{P(F|S) \cdot P(S)}{P(F)}.$$

First compute $P(F)$ using the law of total probability:

$$P(F) = P(F|S)P(S) + P(F|S^c)P(S^c) = 0.9 \cdot 0.2 + 0.1 \cdot 0.8 = 0.26.$$

Then:

$$P(S|F) = \frac{0.9 \cdot 0.2}{0.26} \approx 0.692.$$

(c) The probability that a flagged email is legitimate is the complement:

$$P(S^c|F) = 1 - P(S|F) = 1 - 0.692 = 0.308.$$

C.26 (a) To compute the conditional PDF $f_{X|Y}(x|y)$, we first need the marginal density of Y:

$$\begin{aligned} f_Y(y) &= \int_0^2 f_{X,Y}(x, y)\, dx = \int_0^2 \frac{1}{15}(x + y)\, dx \\ &= \frac{1}{15}\left[\frac{x^2}{2} + yx\right]_0^2 = \frac{1}{15}\left(\frac{4}{2} + 2y\right) = \frac{2(1 + y)}{15}. \end{aligned}$$

Now, we can compute the conditional PDF:

$$f_{X|Y}(x|y) = \frac{f_{X,Y}(x,y)}{f_Y(y)} = \frac{\frac{1}{15}(x+y)}{\frac{2(1+y)}{15}} = \frac{x+y}{2(1+y)}.$$

Thus, the conditional PDF is:

$$f_{X|Y}(x|y) = \frac{x+y}{2(1+y)}, \quad \text{for } 0 \le x \le 2,\ 0 \le y \le 3.$$

(b) To compute the conditional probability:

$$P(X < 1 \mid Y = y) = \int_0^1 f_{X|Y}(x|y)\,dx = \int_0^1 \frac{x+y}{2(1+y)}\,dx$$
$$= \frac{1}{2(1+y)}\left[\frac{x^2}{2} + yx\right]_0^1 = \frac{1}{2(1+y)}\left(\frac{1}{2} + y\right) = \frac{1+2y}{4(1+y)}.$$

Therefore:

$$P(X < 1 \mid Y = y) = \frac{1+2y}{4(1+y)}.$$

C.31 (a) There are $2^3 = 8$ possible outcomes of the coin flips for the three people. Enumerating all configurations and counting how many people receive a gift in each case:

Configuration	Gifted	X
HHH	none	0
HHT	person 3	1
HTH	person 2	1
HTT	person 1	1
THH	person 1	1
THT	person 2	1
TTH	person 3	1
TTT	none	0

So the PMF of X is:

$$P(X = 0) = \frac{2}{8} = 0.25, \quad P(X = 1) = \frac{6}{8} = 0.75.$$

(b) The expectation X is:

$$\mathbb{E}[X] = 0 \cdot \frac{2}{8} + 1 \cdot \frac{6}{8} = \frac{6}{8} = 0.75.$$

The second moment is:

$$\mathbb{E}[X^2] = 0^2 \cdot \frac{2}{8} + 1^2 \cdot \frac{6}{8} = \frac{6}{8}.$$

Therefore, the variance is:

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 = \frac{6}{8} - \left(\frac{6}{8}\right)^2 = \frac{12}{64} = 0.1875.$$

(c) The random variables X_1, X_2, X_3 are not independent. Each X_i depends on the outcomes of person i and their two neighbors. Since the table is circular, the outcomes involved in X_1, X_2, X_3 overlap. For example, X_1 and X_2 both depend on person 2's outcome. As a result, knowledge of X_1 gives information about X_2, and so on.

(d) From the table of outcomes, we see that each person receives a gift in exactly 2 out of the 8 possible configurations. Therefore,

$$\mathbb{E}[X_1] = \mathbb{E}[X_2] = \mathbb{E}[X_3] = \frac{2}{8} = 0.25.$$

To compute $\mathrm{Cov}(X_1, X_2)$, we use the formula:

$$\mathrm{Cov}(X_1, X_2) = \mathbb{E}[X_1 X_2] - \mathbb{E}[X_1]\,\mathbb{E}[X_2].$$

From our previous analysis, we know that in every configuration where someone receives a gift, only one person does. Therefore, it is never the case that both $X_1 = 1$ and $X_2 = 1$ in the same configuration, which implies:

$$\mathbb{E}[X_1 X_2] = 0.$$

Thus,

$$\mathrm{Cov}(X_1, X_2) = 0 - 0.25 \cdot 0.25 = -0.0625.$$

By symmetry:

$$\mathrm{Cov}(X_i, X_j) = -0.0625 \quad \text{for all } i \neq j.$$

This confirms that the gift indicators are negatively correlated: if one person receives a gift, it becomes less likely for their neighbors to do so.

(e) We now extend the setting to $n \geq 3$.

- The expected value $\mathbb{E}[X_i]$ is the probability that person i receives a gift—that is, the probability that their coin flip differs from both immediate neighbors. This occurs only when both neighbors flip the same value (either HH or TT), and person i flips the opposite value.
  The probability that the neighbors agree is:

$$P(\text{neighbors agree}) = P(\text{HH}) + P(\text{TT}) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}.$$

  Given that the neighbors agree, person i has a $\frac{1}{2}$ chance of flipping the opposite result.
  Therefore, by the multiplication rule:

$$P(X_i = 1) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}.$$

  Thus, the expected sum of the indicator variables is:

$$\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] = n \cdot \frac{1}{4} = \frac{n}{4}.$$

- Each X_i depends on the outcomes of person i and their two immediate neighbors, so two variables X_i and X_j are dependent if they share one or more of the same coin flips.
  In general, $\text{Cov}(X_i, X_j)$ is nonzero only when i and j are at most two positions apart (modulo n), that is, when $|i - j| \leq 2 \mod n$. This creates a *banded dependency structure*, where most covariances are zero, but some local dependencies remain.
  Consequently, computing each nonzero term requires careful case analysis, making it difficult to derive a closed-form expression for the total variance in the general case:

$$\text{Var}(X) = \sum_{i=1}^{n} \text{Var}(X_i) + 2 \sum_{i<j} \text{Cov}(X_i, X_j).$$

C.36 We begin by computing the moment-generating function (MGF) of a Poisson random variable $X \sim \text{Poisson}(\lambda)$:

$$M_X(t) = \mathbb{E}[e^{tX}] = \sum_{k=0}^{\infty} e^{tk} \cdot P(X = k).$$

Substituting the PMF of the Poisson distribution:

$$P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!},$$

we get:

$$\begin{aligned} M_X(t) &= \sum_{k=0}^{\infty} e^{tk} \cdot \frac{e^{-\lambda}\lambda^k}{k!} \\ &= e^{-\lambda} \sum_{k=0}^{\infty} \frac{(\lambda e^t)^k}{k!} \\ &= e^{-\lambda} \cdot \exp(\lambda e^t) \quad \text{(by the Taylor series for } e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}\text{)} \\ &= \exp(\lambda(e^t - 1)). \end{aligned}$$

Thus, the MGF of a Poisson(λ) random variable is:

$$M_X(t) = \exp(\lambda(e^t - 1)).$$

Now let $X \sim \text{Poisson}(\lambda_1)$ and $Y \sim \text{Poisson}(\lambda_2)$ be independent. Using the result above, their MGFs are:

$$M_X(t) = \exp(\lambda_1(e^t - 1)), \quad M_Y(t) = \exp(\lambda_2(e^t - 1)).$$

Since the MGF of a sum of independent random variables is the product of their MGFs, we have:

$$M_Z(t) = M_X(t) \cdot M_Y(t) = \exp((\lambda_1 + \lambda_2)(e^t - 1)).$$

This is the MGF of a Poisson($\lambda_1 + \lambda_2$) distribution. Hence, $Z \sim \text{Poisson}(\lambda_1 + \lambda_2)$.

C.40 Expanding the outer product and applying linearity of expectation:

$$\begin{aligned} \mathbb{E}[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^T] &= \mathbb{E}[\mathbf{X}\mathbf{X}^T - \mathbf{X}\boldsymbol{\mu}^T - \boldsymbol{\mu}\mathbf{X}^T + \boldsymbol{\mu}\boldsymbol{\mu}^T] \\ &= \mathbb{E}[\mathbf{X}\mathbf{X}^T] - \mathbb{E}[\mathbf{X}]\boldsymbol{\mu}^T - \boldsymbol{\mu}\,\mathbb{E}[\mathbf{X}^T] + \boldsymbol{\mu}\boldsymbol{\mu}^T \\ &= \mathbb{E}[\mathbf{X}\mathbf{X}^T] - \boldsymbol{\mu}\boldsymbol{\mu}^T. \end{aligned}$$

C.44 (a) To show that $\mathbf{X} + \mathbf{Y}$ is multivariate normal, we use the definition: a random vector $\mathbf{Z}$ is multivariate normal if for every vector $\mathbf{a} \in \mathbb{R}^n$, the linear combination $\mathbf{a}^T\mathbf{Z}$ is a univariate normal random variable.

Let $\mathbf{Z} = \mathbf{X} + \mathbf{Y}$. For any $\mathbf{a} \in \mathbb{R}^n$, we have:

$$\mathbf{a}^T\mathbf{Z} = \mathbf{a}^T(\mathbf{X} + \mathbf{Y}) = \mathbf{a}^T\mathbf{X} + \mathbf{a}^T\mathbf{Y}.$$

Since $\mathbf{X} \sim \mathcal{N}(\mu_X, \Sigma_X)$ and $\mathbf{Y} \sim \mathcal{N}(\mu_Y, \Sigma_Y)$, it follows that both $\mathbf{a}^T\mathbf{X}$ and $\mathbf{a}^T\mathbf{Y}$ are univariate normal random variables.

Moreover, since $\mathbf{X}$ and $\mathbf{Y}$ are independent random vectors, any measurable functions of them — in particular, the linear combinations $\mathbf{a}^T\mathbf{X}$ and $\mathbf{a}^T\mathbf{Y}$ — are also independent.

Hence, $\mathbf{a}^T\mathbf{Z}$ is a sum of independent univariate normal random variables, and is therefore itself normally distributed.

Since this holds for every $\mathbf{a} \in \mathbb{R}^n$, we conclude that $\mathbf{Z} = \mathbf{X}+\mathbf{Y}$ is multivariate normal.

(b) By linearity of expectation:

$$\mathbb{E}[\mathbf{X} + \mathbf{Y}] = \mathbb{E}[\mathbf{X}] + \mathbb{E}[\mathbf{Y}] = \mu_X + \mu_Y.$$

Since $\mathbf{X}$ and $\mathbf{Y}$ are independent, the covariance of their sum satisfies:

$$\mathrm{Cov}[\mathbf{X} + \mathbf{Y}] = \mathrm{Cov}[\mathbf{X}] + \mathrm{Cov}[\mathbf{Y}] = \Sigma_X + \Sigma_Y.$$

Therefore,

$$\mathbf{X} + \mathbf{Y} \sim \mathcal{N}(\mu_X + \mu_Y, \Sigma_X + \Sigma_Y).$$

C.57 We are given that $X_1, \ldots, X_n$ are independent random variables, each satisfying $X_i \in [0, 1]$ and $\mathbb{E}[X_i] = \mu = 0.5$. The sample average is:

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

We are asked to bound the probability $P(\overline{X}_n \geq 0.8)$ for $n = 100$.

(a) Since $\overline{X}_n \geq 0$, we apply Markov's inequality:

$$P(\overline{X}_n \geq 0.8) \leq \frac{\mathbb{E}[\overline{X}_n]}{0.8} = \frac{0.5}{0.8} = 0.625.$$

(b) Since $X_i \in [0,1]$, we have $\text{Var}(X_i) \leq \frac{1}{4}$. Then:

$$\text{Var}(\overline{X}_n) = \frac{1}{n^2}\sum_{i=1}^{n}\text{Var}(X_i) \leq \frac{n \cdot \frac{1}{4}}{n^2} = \frac{1}{4n}.$$

For $n = 100$, this gives $\text{Var}(\overline{X}_n) \leq \frac{1}{400}$. Chebyshev's inequality yields:

$$P(\overline{X}_n \geq 0.8) = P(\overline{X}_n - 0.5 \geq 0.3) \leq P(|\overline{X}_n - 0.5| \geq 0.3) \leq \frac{1/400}{0.3^2} = \approx 0.0278.$$

(c) Hoeffding's inequality for bounded independent variables $X_i \in [0,1]$ gives:

$$P(\overline{X}_n \geq \mu + t) \leq \exp(-2nt^2).$$

Substituting $n = 100$, $t = 0.3$, we get:

$$P(\overline{X}_n \geq 0.8) \leq \exp(-2 \cdot 100 \cdot 0.09) = e^{-18} \approx 1.52 \times 10^{-8}.$$

(d) Since the X_i are independent and bounded in $[0,1]$, we can apply the Chernoff bound to the sample average $\overline{X}_n$ for any mean $\mu = \mathbb{E}[X_i]$. In our case, $\mu = 0.5$, and we are interested in bounding the probability that $\overline{X}_n$ exceeds this mean by a factor of δ:

$$P(\overline{X}_n \geq (1+\delta)\mu) \leq \exp\left(-\frac{\delta^2 \mu n}{3}\right),$$

where $\delta = \dfrac{0.8 - 0.5}{0.5} = 0.6$.
Substituting into the bound gives:

$$P(\overline{X}_n \geq 0.8) \leq \exp\left(-\frac{0.6^2 \cdot 0.5 \cdot 100}{3}\right) = e^{-6} \approx 0.00248.$$

(e) The four bounds are:

- Markov: 0.625
- Chebyshev: ≈ 0.0278
- Chernoff: ≈ 0.00248
- Hoeffding: $\approx 1.52 \times 10^{-8}$

Hoeffding gives the tightest bound because it fully exploits the independence and boundedness of the variables. Chernoff bounds are typically tighter than Hoeffding's for sums of Bernoulli or sub-Gaussian variables, and they can be extended to a broader class of distributions using moment-generating functions. Chebyshev is looser but uses variance, while Markov is the weakest, relying only on the mean.

C.62 Let X_i be the number of IV bags needed by the i-th patient. Then $X_i \in \{0, 1, 2\}$, and the distribution is given by:

$$P(X_i = 0) = \frac{1}{5}, \quad P(X_i = 1) = \frac{1}{2}, \quad P(X_i = 2) = \frac{3}{10}.$$

The expected value of X_i is:

$$\mathbb{E}[X_i] = 0 \cdot \frac{1}{5} + 1 \cdot \frac{1}{2} + 2 \cdot \frac{3}{10} = \frac{11}{10} = 1.1.$$

The second moment is:

$$\mathbb{E}[X_i^2] = 0^2 \cdot \frac{1}{5} + 1^2 \cdot \frac{1}{2} + 2^2 \cdot \frac{3}{10} = \frac{17}{10} = 1.7.$$

The variance is:

$$\mathrm{Var}(X_i) = \mathbb{E}[X_i^2] - (\mathbb{E}[X_i])^2 = 1.7 - (1.1)^2 = 0.49.$$

Let $T = \sum_{i=1}^{100} X_i$ be the total number of IV bags needed. Then by the linearity of expectation and independence:

$$\mathbb{E}[T] = 100 \cdot 1.1 = 110, \quad \mathrm{Var}(T) = 100 \cdot 0.49 = 49.$$

By the Central Limit Theorem, for large n, the distribution of T is approximately normal:

$$T \approx \mathcal{N}(110, 49).$$

Let t be the number of IV bags prepared. We want:

$$P(T \leq t) \geq 0.95.$$

Standardize:

$$P\left(\frac{T - 110}{\sqrt{49}} \leq \frac{t - 110}{7}\right) \geq 0.95.$$

Let $z = \dfrac{t - 110}{7}$. Then $\Phi(z) \geq 0.95 \Rightarrow z \approx 1.645$. Thus,

$$\frac{t - 110}{7} \approx 1.645 \Rightarrow t \approx 110 + 1.645 \cdot 7 = 121.515.$$

Therefore, the hospital should prepare at least 122 IV bags to ensure a 95% probability of meeting demand.

Appendix D

Statistics

D.4 (a) We begin by expanding the product inside C_n:

$$(X_i - \overline{X})(Y_i - \overline{Y}) = X_iY_i - X_i\overline{Y} - \overline{X}Y_i + \overline{X}\,\overline{Y}.$$

Summing over i and dividing by n gives

$$\begin{aligned} C_n &= \frac{1}{n}\sum_{i=1}^{n} X_iY_i - \overline{Y}\frac{1}{n}\sum_{i=1}^{n} X_i - \overline{X}\frac{1}{n}\sum_{i=1}^{n} Y_i + \overline{XY} \\ &= \frac{1}{n}\sum_{i=1}^{n} X_iY_i - \overline{X}\,\overline{Y}. \end{aligned}$$

Taking expectations on both sides and using linearity of expectation, we obtain

$$\mathbb{E}[C_n] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} X_iY_i\right] - \mathbb{E}[\overline{X}\,\overline{Y}].$$

Since the pairs (X_i, Y_i) are i.i.d., each term X_iY_i has the same distribution as X_1Y_1, and therefore

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} X_iY_i\right] = \mathbb{E}[X_1Y_1].$$

By definition,

$$\mathrm{Cov}(X_1, Y_1) = \mathbb{E}[X_1Y_1] - \mathbb{E}[X_1]\,\mathbb{E}[Y_1],$$

and since $\mathbb{E}[X_1] = \mu_X$ and $\mathbb{E}[Y_1] = \mu_Y$, it follows that

$$\mathbb{E}[X_1 Y_1] = \sigma_{XY} + \mu_X \mu_Y.$$

It remains to compute $\mathbb{E}[\overline{XY}]$:

$$\mathbb{E}[\overline{XY}] = \mathbb{E}\left[\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} X_i Y_j\right] = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbb{E}[X_i Y_j].$$

For $i = j$, $\mathbb{E}[X_i Y_i] = \sigma_{XY} + \mu_X\mu_Y$, while for $i \neq j$, independence implies $\mathbb{E}[X_i Y_j] = \mu_X \mu_Y$. Hence,

$$\mathbb{E}[\overline{XY}] = \frac{1}{n^2}\left[n(\sigma_{XY} + \mu_X\mu_Y) + n(n-1)\mu_X\mu_Y\right] = \frac{1}{n}\sigma_{XY} + \mu_X\mu_Y.$$

Substituting back,

$$\mathbb{E}[C_n] = (\sigma_{XY} + \mu_X\mu_Y) - \left(\frac{1}{n}\sigma_{XY} + \mu_X\mu_Y\right) = \frac{n-1}{n}\sigma_{XY}.$$

Thus, C_n is a biased estimator of σ_{XY}.

(b) The corrected estimator is

$$s_{XY} = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y}) = \frac{n}{n-1}\,C_n.$$

Taking expectations and using the result from part (a),

$$\mathbb{E}[s_{XY}] = \frac{n}{n-1}\,\mathbb{E}[C_n] = \frac{n}{n-1}\cdot\frac{n-1}{n}\sigma_{XY} = \sigma_{XY}.$$

Therefore, s_{XY} is an unbiased estimator of the population covariance.

D.9 (a) Let $X_1, X_2, \ldots, X_8$ denote the observed counts. Assume $X_i \sim \text{Poisson}(\lambda)$ i.i.d., with PMF

$$f(x_i;\lambda) = \frac{e^{-\lambda}\lambda^{x_i}}{x_i!}.$$

The likelihood function is:

$$L(\lambda) = \prod_{i=1}^{8}\frac{e^{-\lambda}\lambda^{x_i}}{x_i!} = e^{-8\lambda}\lambda^{\sum_{i=1}^{8}x_i}\prod_{i=1}^{8}\frac{1}{x_i!}.$$

(b) The log-likelihood function is:

$$\ell(\lambda) = \log L(\lambda) = -8\lambda + \left(\sum_{i=1}^{8} x_i\right) \log \lambda - \sum_{i=1}^{8} \log(x_i!).$$

Taking the derivative with respect to λ and setting it to zero:

$$\frac{d\ell}{d\lambda} = -8 + \frac{\sum x_i}{\lambda} = 0 \quad \Rightarrow \quad \hat{\lambda}_{\text{MLE}} = \frac{1}{8} \sum_{i=1}^{8} x_i.$$

(c) The observed data are: $3, 2, 4, 1, 2, 3, 0, 4$. Thus,

$$\hat{\lambda}_{\text{MLE}} = \frac{3+2+4+1+2+3+0+4}{8} = \frac{19}{8} = 2.375.$$

(d) Since λ represents the expected number of emails per hour, the predicted number of emails in the next hour is:

$$\mathbb{E}[X_{\text{next}}] \approx \hat{\lambda}_{\text{MLE}} = 2.375.$$

D.12 (a) Let us define the observed counts of commuting choices under each weather condition:

- For non-rainy days:

$$n_{0,\text{bus}} = 6,\ n_{0,\text{train}} = 8,\ n_{0,\text{bike}} = 4.$$

- For rainy days:

$$n_{1,\text{bus}} = 7,\ n_{1,\text{train}} = 4,\ n_{1,\text{bike}} = 1.$$

The commuting choices are assumed independent across days and follow a categorical distribution conditional on the weather. Since each weather condition is modeled separately, the joint likelihood is the product of two independent multinomial likelihoods — one for non-rainy days and one for rainy days:

$$L(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1) = \frac{18!}{6!8!4!}\, \theta_{0,\text{bus}}^6 \theta_{0,\text{train}}^8 \theta_{0,\text{bike}}^4 \cdot \frac{12!}{7!4!1!}\, \theta_{1,\text{bus}}^7 \theta_{1,\text{train}}^4 \theta_{1,\text{bike}}^1.$$

Taking logarithms and ignoring constants not involving the parameters:

$$\begin{aligned}\ell(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1) = {} & 6 \log \theta_{0,\text{bus}} + 8 \log \theta_{0,\text{train}} + 4 \log \theta_{0,\text{bike}} \\ & + 7 \log \theta_{1,\text{bus}} + 4 \log \theta_{1,\text{train}} + 1 \log \theta_{1,\text{bike}},\end{aligned}$$

subject to the constraints:

$$\theta_{0,\text{bus}} + \theta_{0,\text{train}} + \theta_{0,\text{bike}} = 1, \quad \theta_{1,\text{bus}} + \theta_{1,\text{train}} + \theta_{1,\text{bike}} = 1.$$

(b) Since the log-likelihood is separable, we can maximize each part independently. For a multinomial distribution, the MLE for each category is simply the observed relative frequency. Applying this to the non-rainy days:

$$\hat{\theta}_{0,\text{bus}} = \frac{6}{18} = 0.333, \quad \hat{\theta}_{0,\text{train}} = \frac{8}{18} = 0.444, \quad \hat{\theta}_{0,\text{bike}} = \frac{4}{18} = 0.222.$$

Similarly, for the rainy days:

$$\hat{\theta}_{1,\text{bus}} = \frac{7}{12} = 0.583, \quad \hat{\theta}_{1,\text{train}} = \frac{4}{12} = 0.333, \quad \hat{\theta}_{1,\text{bike}} = \frac{1}{12} = 0.083.$$

(c) On non-rainy days, train is the most common commuting mode (44.4%), followed by bus (33.3%) and bike (22.2%). On rainy days, bus usage increases substantially (to 58.3%) while bike usage drops sharply (to 8.3%). This reflects intuitive commuting behavior: people tend to avoid biking in the rain and shift toward bus transport.

D.15 Since the population variance is unknown and the sample size is small ($n = 16$), we construct the confidence interval for the population mean using the t-distribution:

$$\overline{x} \pm t_{n-1,\,\alpha/2} \cdot \frac{s}{\sqrt{n}}.$$

We are given:

$$\overline{x} = 12.4, \quad s = 3.2, \quad n = 16, \quad \alpha = 0.05 \Rightarrow \alpha/2 = 0.025.$$

From a standard t-distribution table, the critical value for 95% confidence with $n - 1 = 15$ degrees of freedom is:

$$t_{15,\,0.025} \approx 2.131.$$

The margin of error is:

$$2.131 \cdot \frac{3.2}{\sqrt{16}} = 1.705.$$

Thus, the confidence interval is:

$$12.4 \pm 1.705 \quad \Rightarrow \quad [10.695,\ 14.105].$$

D.18 Since the sample size is large, we can invoke the CLT and use the normal distribution to construct the confidence interval:

$$\overline{x} \pm z_{\alpha/2} \cdot \frac{\sigma}{\sqrt{n}}.$$

We are given:

$$\overline{x} = 47.8, \quad \sigma^2 = 36 \Rightarrow \sigma = 6, \quad n = 100, \quad \alpha = 0.05 \Rightarrow \alpha/2 = 0.025.$$

The critical value from the standard normal distribution is:

$$z_{0.025} \approx 1.96.$$

The margin of error is:

$$1.96 \cdot \frac{6}{\sqrt{100}} = 1.176.$$

Thus, the confidence interval is:

$$47.8 \pm 1.176 \quad \Rightarrow \quad [46.624,\ 48.976].$$

D.20 (a) We are testing whether the machine is underfilling, so we formulate the hypotheses as:

$$H_0 \colon \mu = 500 \qquad \text{vs.} \qquad H_1 \colon \mu < 500.$$

(b) Since the population standard deviation is known, we use a one-sample z-test. The test statistic is:

$$z_{\text{obs}} = \frac{\overline{x} - \mu_0}{\sigma/\sqrt{n}} = \frac{498.33 - 500}{5/\sqrt{36}} \approx -2.004.$$

(c) The p-value is the probability of observing such an extreme value (or more extreme) under the null hypothesis:

$$p = P(Z < -2.004) \approx 0.0225.$$

(d) Since $p = 0.0225 > \alpha = 0.01$, we do not reject H_0. At the 1% significance level, there is not enough evidence to conclude that the machine is underfilling the boxes.

D.26 (a) Let $D_i = \text{New Technique}_i - \text{Traditional}_i$ be the difference in test scores for student i. We test:

$$H_0 \colon \mu_D = 0 \qquad \text{vs.} \qquad H_1 \colon \mu_D > 0$$

That is, we test whether the new technique improves scores on average.

(b) To perform a paired t-test, we first calculate the differences in scores:

$$D = \{7, 4, -2, 7, 5, 8, 4, 4, -1, 5\}.$$

We compute the sample mean and standard deviation of the differences:

$$\overline{D} = \frac{7 + 4 + (-2) + 7 + 5 + 8 + 4 + 4 + (-1) + 5}{10} = 4.1,$$

$$s_D = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(D_i - \overline{D})^2} = \sqrt{\frac{1}{9}\left[(7 - 4.1)^2 + \cdots + (5 - 4.1)^2\right]} \approx 3.281.$$

The test statistic is:

$$t_{\text{obs}} = \frac{\overline{D}}{s_D/\sqrt{n}} = \frac{4.1}{3.281/\sqrt{10}} \approx 3.953.$$

Degrees of freedom: $n - 1 = 9$.

For a one-sided t-test with $\alpha = 0.05$ and 9 degrees of freedom, the critical value is:

$$t_{0.05,9} \approx 1.833.$$

Since $t_{\text{obs}} = 3.953 > 1.833$, we reject H_0. There is significant evidence that the new technique improves scores.

(c) The absolute differences and ranks are:

Student	D_i	$\lvert D_i \rvert$	Rank	Signed Rank
1	7	7	8.5	8.5
2	4	4	4	4
3	-2	2	2	-2
4	7	7	8.5	8.5
5	5	5	6.5	6.5
6	8	8	10	10
7	4	4	4	4
8	4	4	4	4
9	-1	1	1	-1
10	5	5	6.5	6.5

The sum of the positive signed ranks is:

$$W^+ = 8.5 + 4 + 8.5 + 6.5 + 10 + 4 + 4 + 6.5 = 52.$$

We now look up a Wilcoxon signed-rank table for a one-sided test with $n = 10$ non-zero differences and significance level $\alpha = 0.05$. The table provides the minimum rank sum required to reject H_0 for small differences:

$$W_{\min} = 8.$$

The total rank sum is:

$$T = \frac{n(n+1)}{2} = \frac{10 \cdot 11}{2} = 55.$$

Thus, for a right-tailed test, the rejection region is:

$$W^+ \geq T - W_{\min} + 1 = 55 - 8 + 1 = 48.$$

Since $W^+ = 52 > 48$, we reject H_0. There is statistically significant evidence at the 5% level that the new study technique improves test scores.

(d) Both tests lead to rejecting the null hypothesis, suggesting a statistically significant improvement in scores with the new technique. The paired t-test assumes the differences are normally distributed, while the Wilcoxon test is nonparametric and does not require this assumption.

In this case, the differences appear reasonably symmetric and show no extreme outliers, so the normality assumption for the paired t-test seems appropriate.

(e) If one student had an extremely large difference (e.g., 30 points), the paired t-test would be heavily influenced, potentially exaggerating the statistical significance. In contrast, the Wilcoxon signed-rank test, which is based on the ranks of the differences, would be much less affected. This makes the Wilcoxon test more robust to outliers.

D.29 (a) We perform a one-way ANOVA to test whether the mean accuracy differs significantly across algorithms using the following steps:

1. Compute the group means and grand mean:

$$\overline{X}_{\text{DT}} = \frac{83.2 + 82.3 + 86.1 + 84.2 + 86.4}{5} = 84.44,$$
$$\overline{X}_{\text{SVM}} = \frac{86.4 + 88.2 + 87.5 + 87.3 + 89.1}{5} = 87.7,$$
$$\overline{X}_{\text{KNN}} = \frac{85.6 + 84.7 + 85.8 + 85.1 + 87.2}{5} = 85.68,$$
$$\overline{X} = \frac{(5 \cdot 84.44) + (5 \cdot 87.7) + (5 \cdot 85.68)}{15} = 85.94.$$

2. Compute the between-group sum of squares (SSB):

$$\text{SSB} = 5\left[(84.44 - 85.94)^2 + (87.7 - 85.94)^2 + (85.68 - 85.94)^2\right] = 27.08.$$

3. Compute the within-group sum of squares (SSW) for each algorithm by summing the squared deviations from the group mean:

$$\text{SSW}_{\text{DT}} = \sum_{i=1}^{5}(x_i - 84.44)^2 = 12.774,$$
$$\text{SSW}_{\text{SVM}} = \sum_{i=1}^{5}(x_i - 87.7)^2 = 4.1,$$
$$\text{SSW}_{\text{KNN}} = \sum_{i=1}^{5}(x_i - 85.68)^2 = 3.626.$$

   Total within-group variation:

$$\text{Total SSW} = 12.774 + 4.1 + 3.626 = 20.5.$$

4. Degrees of freedom:

$$\text{df}_{\text{between}} = k - 1 = 2, \qquad \text{df}_{\text{within}} = N - k = 15 - 3 = 12.$$

5. Mean squares:

$$\text{MSB} = \frac{\text{SSB}}{\text{df}_{\text{between}}} = \frac{27.08}{2} = 13.54, \qquad \text{MSW} = \frac{\text{SSW}}{\text{df}_{\text{within}}} = \frac{20.5}{12} = 1.708.$$

6. F-statistic:

$$F = \frac{\text{MSB}}{\text{MSW}} = \frac{13.54}{1.708} \approx 7.93.$$

Using an F-distribution with $(2, 12)$ degrees of freedom, we find a p-value of approximately 0.007. Since $p < 0.05$, we reject the null hypothesis and conclude that at least one algorithm differs significantly in average accuracy.

(b) 1. First, we assign ranks within each row (dataset). Ties are handled by averaging ranks. The rankings are shown in Table D.1.

Dataset	**DT**	**SVM**	**KNN**	**Rank DT**	**Rank SVM**	**Rank KNN**
1	83.2	86.4	85.6	1	3	2
2	82.3	88.2	84.7	1	3	2
3	86.1	87.5	85.8	2	3	1
4	84.2	87.3	85.1	1	3	2
5	86.4	89.1	87.2	1	3	2

Table D.1: Ranking of three classification algorithms used as input for the Friedman test.

2. Compute the sum of ranks for each algorithm:

$$\begin{aligned} R_{\text{DT}} &= 1 + 1 + 2 + 1 + 1 = 6 \\ R_{\text{SVM}} &= 3 + 3 + 3 + 3 + 3 = 15 \\ R_{\text{KNN}} &= 2 + 2 + 1 + 2 + 2 = 9 \end{aligned}$$

3. The Friedman test statistic is given by:

$$Q = \frac{12}{nk(k+1)} \sum_{j=1}^{k} R_j^2 - 3n(k+1),$$

where $n = 5$ (datasets) and $k = 3$ (algorithms).
Substituting:

$$Q = \frac{12}{5 \cdot 3 \cdot 4}(6^2 + 15^2 + 9^2) - 3 \cdot 5 \cdot 4 = 8.4$$

4. Under the null hypothesis (no difference in algorithms), Q approximately follows a chi-squared distribution with $k - 1 = 2$ degrees of freedom.
5. Using a chi-squared table or Python, the corresponding p-value is:

$$p = P(\chi_2^2 \geq 8.4) \approx 0.015.$$

Since $p < 0.05$, we reject the null hypothesis. There is statistically significant evidence that at least one algorithm performs differently from the others.

(c) The Nemenyi test compares the average ranks of all algorithm pairs. The average ranks based on the Friedman test are:

$$\overline{R}_{\text{DT}} = 1.2, \quad \overline{R}_{\text{KNN}} = 1.8, \quad \overline{R}_{\text{SVM}} = 3.0.$$

The standard error (SE) for the difference in ranks is:

$$\text{SE} = \sqrt{\frac{k(k+1)}{6n}} = \sqrt{\frac{3 \cdot 4}{6 \cdot 5}} = \sqrt{0.4} \approx 0.632.$$

We now compute the observed test statistic for each pair:

$$q_{\text{obs}}(\text{DT}, \text{KNN}) = \frac{|\bar{R}_{\text{DT}} - \bar{R}_{\text{KNN}}|}{\text{SE}} = \frac{|1.2 - 1.8|}{0.632} \approx 0.949,$$
$$q_{\text{obs}}(\text{DT}, \text{SVM}) = \frac{|\bar{R}_{\text{DT}} - \bar{R}_{\text{SVM}}|}{\text{SE}} = \frac{|1.2 - 3.0|}{0.632} \approx 2.846,$$
$$q_{\text{obs}}(\text{KNN}, \text{SVM}) = \frac{|\bar{R}_{\text{KNN}} - \bar{R}_{\text{SVM}}|}{\text{SE}} = \frac{|1.8 - 3.0|}{0.632} \approx 1.899.$$

The critical value from the studentized range distribution for $k = 3$ groups at significance level $\alpha = 0.05$, scaled for pairwise comparison, is:

$$q_{0.05,3,\infty}/\sqrt{2} \approx \frac{3.314}{\sqrt{2}} \approx 2.344.$$

We conclude:

- $q_{\text{obs}}(\text{DT}, \text{KNN}) = 0.949 < 2.344 \Rightarrow$ DT and KNN do not differ significantly.
- $q_{\text{obs}}(\text{DT}, \text{SVM}) = 2.846 > 2.344 \Rightarrow$ DT and SVM differ significantly.
- $q_{\text{obs}}(\text{KNN}, \text{SVM}) = 1.899 < 2.344 \Rightarrow$ SVM and KNN do not differ significantly.

(d) The Nemenyi test identifies a statistically significant difference between DT and SVM. No other pairwise differences are significant at the 5% level.

D.31 (a) The likelihood is:[1]

$$f_{Y|X}(y|x) = \frac{1}{x}, \quad \text{for } y \le x \le 1.$$

Since $1/x$ is a decreasing function of x, it is maximized at the lower endpoint of the interval, i.e., at $x = y$. Thus,

$$\hat{x}_{\text{MLE}} = y.$$

(b) Using Bayes' rule, the unnormalized posterior is:

$$f_{X|Y}(x|y) \propto f_{Y|X}(y|x) f_X(x) = \frac{1}{x} \cdot 3x^2 = 3x, \quad \text{for } y \le x \le 1.$$

To normalize, we compute the integral:

$$\int_y^1 3x\, dx = 3\left[\frac{x^2}{2}\right]_y^1 = \frac{3}{2}(1 - y^2).$$

So the posterior density is:

$$f_{X|Y}(x|y) = \frac{3x}{\frac{3}{2}(1-y^2)} = \frac{2x}{1-y^2}, \quad \text{for } y \le x \le 1.$$

(c) To find the MAP estimate, we maximize the posterior $f_{X|Y}(x|y)$, which is an increasing function of x on the interval $[y, 1]$. Thus, the maximum occurs at:

$$\hat{x}_{\text{MAP}} = 1.$$

(d) The MMSE estimate is:

$$\hat{x}_{\text{MMSE}} = \int_y^1 x f_{X|Y}(x|y)\, dx = \int_y^1 x \cdot \frac{2x}{1-y^2} dx = \frac{2}{1-y^2} \int_y^1 x^2 dx$$
$$= \frac{2}{1-y^2} \cdot \left[\frac{x^3}{3}\right]_y^1 = \frac{2}{1-y^2} \cdot \left(\frac{1}{3} - \frac{y^3}{3}\right) = \frac{2(1-y^3)}{3(1-y^2)}.$$

[1]Note: Once we observe $Y = y$, we only need to write the likelihood for that specific value of y. There is no need to explicitly state the condition $0 \le y \le 1$ inside the likelihood formula, because any value of y outside that range would make the likelihood zero and thus correspond to impossible data.

(e) The interpretation of the three estimates is:

- The MLE is purely data-driven and returns the lowest value of x that could explain the observation y, which is $x = y$.
- The MAP incorporates prior information and chooses the most likely x value under the posterior. Since the prior favors larger x, the posterior peaks at the upper end, giving $\hat{x}_{\text{MAP}} = 1$.
- The MMSE computes the average of all values of x weighted by their posterior probability. It lies between the MLE and MAP and reflects the overall shape of the posterior rather than just its peak.

D.33 (a) Starting from Bayes' rule,

$$p(\lambda|x) \propto p(x|\lambda)\, p(\lambda),$$

we substitute the Poisson likelihood and the Gamma prior:

$$\begin{aligned} p(\lambda|x) &\propto \left(\frac{\lambda^x e^{-\lambda}}{x!}\right)\left(\frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}\right) \\ &\propto \lambda^x e^{-\lambda} \cdot \lambda^{\alpha-1}e^{-\beta\lambda} \\ &= \lambda^{\alpha+x-1}e^{-(\beta+1)\lambda}. \end{aligned}$$

This is the kernel of a Gamma distribution with updated parameters $\alpha' = \alpha + x$ and $\beta' = \beta + 1$.

Therefore,

$$\lambda|x \sim \text{Gamma}(\alpha + x,\, \beta + 1).$$

(b) The observed count x adds directly to the shape parameter, contributing additional "pseudo-counts" of events. The number of observations (in this case, one) increments the rate parameter, which slightly increases the posterior precision.

As a result, the posterior mean

$$\mathbb{E}[\lambda|x] = \frac{\alpha + x}{\beta + 1}$$

combines the prior information with the new data: large values of x pull the posterior mean upward, while a larger β represents a stronger prior that moderates this effect.

Appendix E

Optimization

E.3 (a) Let $x_0 = (a+b)/2$ be the midpoint of the initial interval $[a, b]$. Since f is continuous and $f(a)f(b) < 0$, the root x^* lies in $[a, b]$, and the initial error is bounded by:

$$|x_0 - x^*| \leq \frac{b-a}{2}.$$

At each step of the bisection method, the interval length is halved. Therefore, after k iterations, the length of the interval is:

$$L_k = \frac{b-a}{2^k},$$

and the midpoint approximation x_k lies at the center of the interval.

The root still lies within the interval, so the error is at most half the interval length:

$$|x_k - x^*| \leq \frac{L_k}{2} = \frac{b-a}{2^{k+1}}.$$

(b) From part (a), we have the error bound:

$$e_k = |x_k - x^*| \leq \frac{b-a}{2^{k+1}}.$$

Similarly,

$$e_{k+1} \leq \frac{b-a}{2^{k+2}}.$$

Taking the ratio, we get:

$$\frac{e_{k+1}}{e_k} \leq \frac{\frac{b-a}{2^{k+2}}}{\frac{b-a}{2^{k+1}}} = \frac{1}{2}.$$

Therefore,

$$\lim_{k\to\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|} \leq \frac{1}{2}.$$

Hence, the bisection method converges linearly, with an asymptotic convergence rate bounded above by $\mu = 1/2$.

E.8 (a) We have

$$f(x) = (x^2 - 1)^2.$$

By the chain rule,

$$f'(x) = 2(x^2 - 1) \cdot 2x = 4x(x^2 - 1).$$

At $x_0 = 1.5$,

$$x_0^2 - 1 = 1.5^2 - 1 = 2.25 - 1 = 1.25,$$

so

$$f'(1.5) = 4 \cdot 1.5 \cdot 1.25 = 7.5.$$

The steepest descent direction is

$$p_0 = -f'(x_0) = -7.5.$$

The initial function value is

$$f(x_0) = f(1.5) = (1.5^2 - 1)^2 = 1.25^2 = 1.5625.$$

(b) The Armijo (sufficient decrease) condition at x_0 with step size α is

$$f(x_0 + \alpha p_0) \;\leq\; f(x_0) + c_1 \alpha f'(x_0) p_0.$$

Here, $c_1 = 10^{-4}$ and

$$f'(x_0)p_0 = 7.5 \cdot (-7.5) = -56.25,$$

so the right-hand side (RHS) becomes

$$f(x_0) + c_1 \alpha f'(x_0) p_0 = 1.5625 + 10^{-4}\alpha(-56.25) = 1.5625 - 0.005625\alpha.$$

(c) We now test candidate step sizes

$$\alpha = 1,\ 0.5,\ 0.25,\ 0.125, \ldots$$

by backtracking with reduction factor $\beta = 0.5$.
For each α, we compute

$$x_0 + \alpha p_0, \quad \text{LHS} = f(x_0 + \alpha p_0), \quad \text{RHS} = 1.5625 - 0.005625\,\alpha.$$

Using $p_0 = -7.5$ and $x_0 = 1.5$:

1. For $\alpha = 1$:
$$x_0 + \alpha p_0 = 1.5 - 7.5 = -6,$$
$$f(-6) = (36 - 1)^2 = 35^2 = 1225,$$
$$\text{RHS} = 1.5625 - 0.005625 \cdot 1 \approx 1.5569.$$
Since $1225 \not\leq 1.5569$, the Armijo condition fails.
2. For $\alpha = 0.5$:
$$x_0 + \alpha p_0 = 1.5 - 3.75 = -2.25,$$
$$f(-2.25) = \left((2.25)^2 - 1\right)^2 = (5.0625 - 1)^2 = 4.0625^2 \approx 16.5039,$$
$$\text{RHS} = 1.5625 - 0.005625 \cdot 0.5 \approx 1.5597.$$
Again $16.5039 \not\leq 1.5597$, so the condition fails.
3. For $\alpha = 0.25$:
$$x_0 + \alpha p_0 = 1.5 - 1.875 = -0.375,$$
$$f(-0.375) = \left((-0.375)^2 - 1\right)^2 = (0.140625 - 1)^2 \approx 0.7385$$
$$\text{RHS} = 1.5625 - 0.005625 \cdot 0.25 \approx 1.5611.$$
This time, $0.7385 \leq 1.5611$, so the Armijo condition is satisfied.

We can summarize the trials in a table (values rounded):

Trial	α	$f(x_0 + \alpha p_0)$	$f(x_0) + c_1 \alpha f'(x_0) p_0$	Armijo?
1	1	1225.0000	1.5569	✗
2	0.5	16.5039	1.5597	✗
3	0.25	0.7385	1.5611	✓

(d) The first step size that satisfies the Armijo condition is $\alpha = 0.25$, so the accepted update is

$$x_1 = x_0 + \alpha p_0 = 1.5 + 0.25 \cdot (-7.5) = -0.375.$$

The new function value is

$$f(x_1) = f(-0.375) \approx 0.7385,$$

which is smaller than $f(x_0) = 1.5625$, and x_1 is closer to the minimizer at $x = -1$.

(e) We now perform two more iterations, each time starting with $\alpha_0 = 1$ and using Armijo backtracking with the same parameters.

- Second iteration (from $x_1 = -0.375$):
  Compute the gradient at x_1:

  $$f'(-0.375) = 4(-0.375)\left((-0.375)^2 - 1\right) \approx 1.2891,$$

  so the steepest descent direction is

  $$p_1 = -f'(-0.375) \approx -1.2891.$$

  Applying backtracking (details omitted here), the first step size that satisfies the Armijo condition is

  $$\alpha_1 = 0.5.$$

  The updated iterate is

  $$x_2 = x_1 + \alpha_1 p_1 = -0.375 + 0.5 \cdot (-1.2891) \approx -1.0195.$$

  The corresponding function value is

  $$f(x_2) \approx 0.0016.$$

- Third iteration (from $x_2 = -1.0195$):
  Compute the gradient at x_2:

  $$f'(-1.0195) = 4(-1.0195)\left((-1.0195)^2 - 1\right) \approx -0.1607.$$

  The steepest descent direction is therefore

  $$p_2 = -f'(x_2) \approx 0.1607.$$

Using backtracking, the first step size that satisfies the Armijo condition is

$$\alpha_2 = 0.125.$$

The updated iterate is

$$x_3 = x_2 + \alpha_2 p_2 = -1.0195 + 0.125 \cdot 0.1607 \approx -0.9994.$$

The corresponding function value is

$$f(x_3) = \left(x_3^2 - 1\right)^2 \approx 1.4 \cdot 10^{-6}.$$

These iterations show that the sequence $\{x_k\}$ converges toward the minimizer at $x = -1$, and that the Armijo backtracking line search adaptively chooses smaller step sizes as we approach the minimum.

E.10 (a) To analyze the difference $f(\mathbf{x}+\mathbf{d}) - f(\mathbf{x})$, we define a scalar-valued function along the line segment between $\mathbf{x}$ and $\mathbf{x} + \mathbf{d}$:

$$g(t) = f(\mathbf{x} + t\mathbf{d}), \quad \text{for } t \in [0, 1].$$

Then we have:

$$f(\mathbf{x} + \mathbf{d}) = g(1), \quad f(\mathbf{x}) = g(0),$$

so that:

$$f(\mathbf{x} + \mathbf{d}) - f(\mathbf{x}) = g(1) - g(0).$$

By the Fundamental Theorem of Calculus:

$$g(1) - g(0) = \int_0^1 g'(t)\, dt.$$

Using the chain rule, we compute:

$$g'(t) = \frac{d}{dt} f(\mathbf{x} + t\mathbf{d}) = \nabla f(\mathbf{x} + t\mathbf{d})^T \mathbf{d}.$$

Substituting into the integral gives:

$$\begin{aligned} f(\mathbf{x} + \mathbf{d}) - f(\mathbf{x}) &= \int_0^1 \nabla f(\mathbf{x} + t\mathbf{d})^T \mathbf{d}\, dt \\ &= \nabla f(\mathbf{x})^T \mathbf{d} + \int_0^1 \left(\nabla f(\mathbf{x} + t\mathbf{d}) - \nabla f(\mathbf{x})\right)^T \mathbf{d}\, dt. \end{aligned}$$

To bound the second term, we begin by applying the Cauchy–Schwarz inequality:

$$(\nabla f(\mathbf{x}+t\mathbf{d}) - \nabla f(\mathbf{x}))^T \mathbf{d} \leq \|\nabla f(\mathbf{x}+t\mathbf{d}) - \nabla f(\mathbf{x})\| \cdot \|\mathbf{d}\|.$$

Now, using the L-Lipschitz continuity of the gradient:

$$\|\nabla f(\mathbf{x}+t\mathbf{d}) - \nabla f(\mathbf{x})\| \leq Lt\|\mathbf{d}\|.$$

Substituting this into the previous inequality gives:

$$(\nabla f(\mathbf{x}+t\mathbf{d}) - \nabla f(\mathbf{x}))^T \mathbf{d} \leq Lt\|\mathbf{d}\|^2.$$

Integrating this bound:

$$\int_0^1 (\nabla f(\mathbf{x}+t\mathbf{d}) - \nabla f(\mathbf{x}))^T \mathbf{d}\, dt \leq \int_0^1 Lt\|\mathbf{d}\|^2\, dt = \frac{L}{2}\|\mathbf{d}\|^2.$$

Hence, we conclude:

$$f(\mathbf{x}+\mathbf{d}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^T\mathbf{d} + \frac{L}{2}\|\mathbf{d}\|^2.$$

(b) Let $\mathbf{d} = -\alpha\nabla f(\mathbf{x}_t)$ and apply the previous result with $\mathbf{x} = \mathbf{x}_t$:

$$\begin{aligned} f(\mathbf{x}_{t+1}) &= f(\mathbf{x}_t + \mathbf{d}) \\ &\leq f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^T(-\alpha\nabla f(\mathbf{x}_t)) + \frac{L}{2}\| - \alpha\nabla f(\mathbf{x}_t)\|^2 \\ &= f(\mathbf{x}_t) - \alpha\|\nabla f(\mathbf{x}_t)\|^2 + \frac{L\alpha^2}{2}\|\nabla f(\mathbf{x}_t)\|^2 \\ &= f(\mathbf{x}_t) - \left(\alpha - \frac{L\alpha^2}{2}\right)\|\nabla f(\mathbf{x}_t)\|^2. \end{aligned}$$

(c) If $\alpha \leq 1/L$, then we have:

$$\alpha - \frac{L\alpha^2}{2} \geq \alpha - \frac{\alpha}{2} = \frac{\alpha}{2}.$$

Therefore,

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \frac{\alpha}{2}\|\nabla f(\mathbf{x}_t)\|^2.$$

E.12 (a) We are given the objective function:

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}),$$

and the stochastic gradient:

$$g(\mathbf{x};\xi) = \nabla_{\mathbf{x}} f_\xi(\mathbf{x}),$$

where $\xi \in \{1,\dots,n\}$ is a random index sampled uniformly.

To show that $g(\mathbf{x};\xi)$ is an unbiased estimator of $\nabla f(\mathbf{x})$, we compute the expectation:

$$\mathbb{E}_\xi\left[g(\mathbf{x};\xi)\right] = \mathbb{E}_\xi\left[\nabla_{\mathbf{x}} f_\xi(\mathbf{x})\right] = \sum_{i=1}^{n} P(\xi = i)\cdot \nabla_{\mathbf{x}} f_i(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\mathbf{x}} f_i(\mathbf{x})$$
$$= \nabla_{\mathbf{x}}\left(\frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x})\right) = \nabla f(\mathbf{x}).$$

Hence, $\mathbb{E}_\xi[g(\mathbf{x};\xi)] = \nabla f(\mathbf{x})$, which proves the stochastic gradient is an unbiased estimator of the full gradient.

(b) Choosing ξ uniformly ensures that each sample contributes equally to the expected gradient. If some samples are selected with higher probability than others (i.e., non-uniform sampling), the expectation becomes:

$$\mathbb{E}_\xi[g(\mathbf{x};\xi)] = \sum_{i=1}^{n} p_i \cdot \nabla f_i(\mathbf{x}),$$

where $p_i = P(\xi = i)$ and $\sum_{i=1}^{n} p_i = 1$. Unless all $p_i = 1/n$, this weighted sum will not equal $\nabla f(\mathbf{x})$, and the stochastic gradient will be a biased estimator of the full gradient.

Such bias may lead to incorrect or unstable convergence behavior in gradient-based optimization algorithms.

E.15 Let us denote the variables from the first formulation as $\mathbf{v}_k^{(1)}, \alpha_1, \beta_1$ and from the second formulation as $\mathbf{v}_k^{(2)}, \alpha_2, \beta_2$. Our goal is to find a relationship between these sets of parameters such that both updates yield the same $\mathbf{x}_k$ sequence.

From the first formulation:

$$\mathbf{v}_{k+1}^{(1)} = \beta_1 \mathbf{v}_k^{(1)} + \nabla f(\mathbf{x}_k), \tag{1.v}$$
$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_1 \mathbf{v}_{k+1}^{(1)}. \tag{1.x}$$

From the second formulation:

$$\mathbf{v}_{k+1}^{(2)} = \beta_2 \mathbf{v}_k^{(2)} + \alpha_2 \nabla f(\mathbf{x}_k), \tag{2.v}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{v}_{k+1}^{(2)}. \tag{2.x}$$

To match equations (1.x) and (2.x), we define:

$$\mathbf{v}_k^{(2)} = \alpha_1 \mathbf{v}_k^{(1)}.$$

Then from equation (1.v), we get:

$$\mathbf{v}_{k+1}^{(2)} = \alpha_1 \mathbf{v}_{k+1}^{(1)} = \alpha_1 \left(\beta_1 \mathbf{v}_k^{(1)} + \nabla f(\mathbf{x}_k) \right) = \beta_1 \alpha_1 \mathbf{v}_k^{(1)} + \alpha_1 \nabla f(\mathbf{x}_k) = \beta_1 \mathbf{v}_k^{(2)} + \alpha_1 \nabla f(\mathbf{x}_k).$$

This has the same form as equation (2.v) with:

$$\beta_2 = \beta_1, \quad \alpha_2 = \alpha_1.$$

Thus, under the change of variables $\mathbf{v}_k^{(2)} = \alpha_1 \mathbf{v}_k^{(1)}$, the two formulations are equivalent, provided we scale the velocity vector accordingly and set:

$$\beta_2 = \beta_1, \quad \alpha_2 = \alpha_1, \quad \mathbf{v}_k^{(2)} = \alpha_1 \mathbf{v}_k^{(1)}.$$

Conversely, given formulation (2), if we define $\mathbf{v}_k^{(1)} = \mathbf{v}_k^{(2)}/\alpha_2$, the original formulation (1) is recovered. Therefore, the two formulations are mathematically equivalent up to a rescaling of the velocity vector and consistent parameter mapping.

E.18 We are given the quadratic function

$$f(x, y) = 3x^2 + 4xy + 5y^2.$$

(a) Since f is a convex quadratic function, its global minimizer occurs where the gradient vanishes:

$$\nabla f(x, y) = \begin{pmatrix} 6x + 4y \\ 4x + 10y \end{pmatrix} = \mathbf{0}.$$

Solving the system:

$$6x + 4y = 0 \quad \Rightarrow \quad 3x + 2y = 0 \tag{1}$$

$$4x + 10y = 0 \quad \Rightarrow \quad 2x + 5y = 0 \tag{2}$$

From (1) $y = -\frac{3}{2}x$. Substituting into (2):

$$2x + 5\left(-\frac{3}{2}x\right) = 2x - \frac{15}{2}x = -\frac{11}{2}x = 0 \quad \Rightarrow \quad x = 0,\ y = 0.$$

So the global minimizer is $(x^*, y^*) = (0, 0)$, and the minimum function value is $f(0, 0) = 0$.

(b) We perform two full cycles (4 steps) of coordinate descent starting from $(1, -1)$. We will alternate minimizing over x and y, with exact minimization in each coordinate.

1. Minimize over x with $y = -1$.

$$f(x, -1) = 3x^2 - 4x + 5$$

Setting the derivative to zero,

$$\frac{d}{dx}f(x, -1) = 6x - 4 = 0 \quad \Rightarrow \quad x^{(1)} = \frac{2}{3}.$$

The updated iterate is

$$(x^{(1)}, y^{(0)}) = \left(\frac{2}{3}, -1\right)$$

with objective value

$$f\left(\frac{2}{3}, -1\right) = 3\left(\frac{2}{3}\right)^2 + 4 \cdot \left(\frac{2}{3}\right)(-1) + 5 = \frac{11}{3} \approx 3.6667.$$

2. Minimize over y with $x = 2/3$.

$$f\left(\frac{2}{3}, y\right) = 3\left(\frac{2}{3}\right)^2 + \frac{8}{3}y + 5y^2$$

Setting the derivative to zero,

$$\frac{d}{dy}f\left(\frac{2}{3}, y\right) = \frac{8}{3} + 10y = 0 \quad \Rightarrow \quad y^{(1)} = -\frac{4}{15}.$$

The updated iterate is

$$(x^{(1)}, y^{(1)}) = \left(\frac{2}{3}, -\frac{4}{15}\right)$$

with objective value

$$f\left(\frac{2}{3},-\frac{4}{15}\right)=3\cdot\left(\frac{2}{3}\right)^2+4\cdot\frac{2}{3}\cdot\left(-\frac{4}{15}\right)+5\left(-\frac{4}{15}\right)^2=\frac{44}{45}\approx 0.9778.$$

3. Minimize over x with $y=-4/15$.

$$f\left(x,-\frac{4}{15}\right)=3x^2-\frac{16}{15}x+5\cdot\left(\frac{4}{15}\right)^2$$

Setting the derivative to zero,

$$\frac{d}{dx}f\left(x,-\frac{4}{15}\right)=6x-\frac{16}{15}=0\quad\Rightarrow\quad x^{(2)}=\frac{8}{45}.$$

The updated iterate is

$$(x^{(2)},y^{(1)})=\left(\frac{8}{45},-\frac{4}{15}\right)$$

and the corresponding objective value is

$$f\left(\frac{8}{45},-\frac{4}{15}\right)=3\cdot\left(\frac{8}{45}\right)^2+4\cdot\frac{8}{45}\cdot\left(-\frac{4}{15}\right)+5\cdot\left(\frac{4}{15}\right)^2=\frac{176}{675}\approx 0.2607.$$

4. Minimize over y with $x=8/45$.

$$f\left(\frac{8}{45},y\right)=3\cdot\left(\frac{8}{45}\right)^2+\frac{32}{45}y+5y^2$$

Setting the derivative to zero,

$$\frac{d}{dy}f\left(\frac{8}{45},y\right)=\frac{32}{45}+10y=0\quad\Rightarrow\quad y^{(2)}=-\frac{16}{225}.$$

The updated iterate is

$$(x^{(2)},y^{(2)})=\left(\frac{8}{45},-\frac{16}{225}\right)$$

and the corresponding objective value is

$$f\left(\frac{8}{45},-\frac{16}{225}\right)=3\cdot\left(\frac{8}{45}\right)^2+4\cdot\frac{8}{45}\cdot\left(-\frac{16}{225}\right)+5\cdot\left(\frac{16}{225}\right)^2=\frac{704}{10125}\approx 0.0695.$$

(c) The exact minimum value is $f(0,0) = 0$. After four steps of coordinate descent, the function value is approximately 0.0695. This is already quite close to the true minimum, showing that coordinate descent has made significant progress in just two cycles.

E.22 We are asked to minimize the quadratic function

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x}$$

along the direction $\mathbf{p}_k$, starting from the current iterate $\mathbf{x}_k$. Define the one-dimensional function

$$g(\alpha) = f(\mathbf{x}_k + \alpha\mathbf{p}_k).$$

Substituting into f, we get

$$\begin{aligned} g(\alpha) &= \frac{1}{2}(\mathbf{x}_k + \alpha\mathbf{p}_k)^T A(\mathbf{x}_k + \alpha\mathbf{p}_k) - \mathbf{b}^T(\mathbf{x}_k + \alpha\mathbf{p}_k) \\ &= \frac{1}{2}\mathbf{x}_k^T A\mathbf{x}_k + \alpha\mathbf{x}_k^T A\mathbf{p}_k + \frac{1}{2}\alpha^2\mathbf{p}_k^T A\mathbf{p}_k - \mathbf{b}^T\mathbf{x}_k - \alpha\mathbf{b}^T\mathbf{p}_k. \end{aligned}$$

Now take the derivative with respect to α:

$$g'(\alpha) = \mathbf{x}_k^T A\mathbf{p}_k + \alpha\mathbf{p}_k^T A\mathbf{p}_k - \mathbf{b}^T\mathbf{p}_k.$$

Set $g'(\alpha) = 0$ to find the minimizer:

$$\begin{aligned} \mathbf{x}_k^T A\mathbf{p}_k + \alpha\mathbf{p}_k^T A\mathbf{p}_k - \mathbf{b}^T\mathbf{p}_k = 0, \\ \alpha = \frac{\mathbf{b}^T\mathbf{p}_k - \mathbf{x}_k^T A\mathbf{p}_k}{\mathbf{p}_k^T A\mathbf{p}_k}. \end{aligned}$$

Using the residual $\mathbf{r}_k = A\mathbf{x}_k - \mathbf{b}$ and the fact that A is symmetric, we get:

$$\mathbf{b}^T\mathbf{p}_k - \mathbf{x}_k^T A\mathbf{p}_k = -(\mathbf{x}_k^T A - \mathbf{b}^T)\mathbf{p}_k = -(A\mathbf{x}_k - \mathbf{b})^T\mathbf{p}_k = -\mathbf{r}_k^T\mathbf{p}_k.$$

Thus, the step length is:

$$\alpha_k = -\frac{\mathbf{r}_k^T\mathbf{p}_k}{\mathbf{p}_k^T A\mathbf{p}_k}.$$

□

E.28 We are given the function

$$f(x) = x^4 - 3x^2 + 2,$$

and we are asked to find its minimum using Newton's method starting from $x_0 = 1$.

We begin by computing the first and second derivatives:

$$f'(x) = 4x^3 - 6x, \quad f''(x) = 12x^2 - 6.$$

The Newton iteration formula is:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}.$$

We now compute five iterations starting from $x_0 = 1$:

1. $x_0 = 1$

$$f'(1) = -2, \quad f''(1) = 6, \quad x_1 = 1 - \frac{-2}{6} = 1 + \frac{1}{3} = 1.3333.$$

2. $x_1 = 1.3333$

$$f'(1.3333) = 4 \cdot 1.3333^3 - 6 \cdot 1.3333 \approx 1.4810,$$
$$f''(1.3333) = 12 \cdot 1.3333^2 - 6 \approx 15.3323,$$
$$x_2 = 1.3333 - \frac{1.4810}{15.3323} \approx 1.2367.$$

3. $x_2 = 1.2367$

$$f'(1.2367) = 4 \cdot 1.2367^3 - 6 \cdot 1.2367 \approx 0.1456,$$
$$f''(1.2367) = 12 \cdot 1.2367^2 - 6 \approx 12.3531,$$
$$x_3 = 1.2367 - \frac{0.1456}{12.3531} \approx 1.2249.$$

4. $x_3 = 1.2249$

$$f'(1.2249) = 4 \cdot 1.2249^3 - 6 \cdot 1.2249 \approx 0.0019,$$
$$f''(1.2249) = 12 \cdot 1.2249^2 - 6 \approx 12.0046,$$
$$x_4 = 1.2249 - \frac{0.0019}{12.0046} \approx 1.2247.$$

5. $x_4 = 1.2247$

$$f'(1.2247) = 4 \cdot 1.2247^3 - 6 \cdot 1.2247 \approx -0.0005,$$

$$f''(1.2247) = 12 \cdot 1.2247^2 - 6 \approx 11.9987,$$

$$x_5 = 1.2247 - \frac{-0.0005}{11.9987} \approx 1.2247.$$

To find the exact minimizer, we solve $f'(x) = 0$:

$$4x^3 - 6x = 2x(2x^2 - 3) = 0 \;\Rightarrow\; x = 0, \pm\sqrt{\frac{3}{2}} \approx \pm 1.2247.$$

To determine which is a minimum, we evaluate $f''(x)$:

$$f''(0) = -6 < 0,\; f''(\pm\sqrt{3/2}) = 12 \cdot \frac{3}{2} - 6 = 12 > 0,$$

so both $x = \pm\sqrt{3/2}$ are local minima.

Since we started at $x_0 = 1 > 0$, Newton's method converged to the positive local minimizer:

$$x^* = \sqrt{\frac{3}{2}} \approx 1.2247,$$

which agrees with the final iterate $x_5 \approx 1.2247$.

E.30 (a) To derive the second-order approximation of f around a point $\mathbf{a} \in \mathbb{R}^n$, we start with the multivariate Taylor expansion:

$$f(\mathbf{x}) = f(\mathbf{a}) + \nabla f(\mathbf{a})^T(\mathbf{x} - \mathbf{a}) + \frac{1}{2}(\mathbf{x} - \mathbf{a})^T H_f(\mathbf{a})(\mathbf{x} - \mathbf{a}) + \mathcal{O}(\|\mathbf{x} - \mathbf{a}\|^3).$$

Here:

- $\nabla f(\mathbf{a})$ is the gradient of f evaluated at $\mathbf{a}$, a column vector in $\mathbb{R}^n$.
- $H_f(\mathbf{a})$ is the Hessian matrix of second partial derivatives of f at $\mathbf{a}$.

Truncating the series after the quadratic term, we obtain the second-order approximation:

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

(b) Let $\mathbf{d} = \mathbf{x} - \mathbf{a}$. Then the second-order approximation becomes:

$$q(\mathbf{d}) = f(\mathbf{a}) + \nabla f(\mathbf{a})^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T H_f(\mathbf{a})\mathbf{d}.$$

To minimize $q(\mathbf{d})$, we compute its gradient with respect to $\mathbf{d}$. Note the following:

- $\nabla_{\mathbf{d}} \nabla f(\mathbf{a})^T \mathbf{d} = \nabla f(\mathbf{a})$, since this is a linear term in $\mathbf{d}$.
- $\nabla_{\mathbf{d}} \frac{1}{2}\mathbf{d}^T H_f(\mathbf{a})\mathbf{d} = H_f(\mathbf{a})\mathbf{d}$, since this is a standard quadratic form and the Hessian is symmetric.

Therefore,

$$\nabla_{\mathbf{d}} q(\mathbf{d}) = \nabla f(\mathbf{a}) + H_f(\mathbf{a})\mathbf{d}.$$

(c) To find the stationary point, we set this gradient to zero:

$$\nabla f(\mathbf{a}) + H_f(\mathbf{a})\mathbf{d} = \mathbf{0}.$$

Solving for $\mathbf{d}$ gives:

$$H_f(\mathbf{a})\mathbf{d} = -\nabla f(\mathbf{a}) \quad \Rightarrow \quad \mathbf{d} = -H_f(\mathbf{a})^{-1} \nabla f(\mathbf{a}).$$

Substituting back $\mathbf{x} = \mathbf{a} + \mathbf{d}$, the next iterate becomes:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - H_f(\mathbf{x}_k)^{-1} \nabla f(\mathbf{x}_k),$$

which is the Newton update rule for multivariate optimization.

E.36 (a) Write a Python function that implements Newton's method to find a minimum of a multivariable function.

```
def newton_minimize(f, grad_f, hess_f, x0, tol=1e-6, max_iter=100):
    """
    Newton's method for unconstrained minimization of a
       multivariable function.

    Parameters:
    f       : Function to minimize
    grad_f  : Gradient function (returns vector)
    hess_f  : Hessian function (returns square matrix)
    x0      : Initial guess (1D array)
    tol     : Stopping tolerance on gradient norm
```

```
    max_iter: Maximum number of iterations

    Returns:
    x       : Approximate minimizer
    """
    x = np.array(x0)

    for i in range(max_iter):
        grad = grad_f(x)
        hess = hess_f(x)

        if np.linalg.norm(grad) < tol:
            return x

        try:
            delta = np.linalg.solve(hess, grad)
        except np.linalg.LinAlgError:
            raise ValueError(f"Hessian is singular at iteration
                {i}, x = {x}")

        x = x - delta

    print("Newton's method did not converge within the maximum
        number of iterations.")
    return x
```

(b) Test your implementation on the function

$$f(x_1, x_2) = (x_1 - 1)^2 + 2(x_2 + 2)^2$$

which has a minimum at $(1, -2)$.

```
def f(x):
    return (x[0] - 1)**2 + 2 * (x[1] + 2)**2

def grad_f(x):
    return np.array([
        2 * (x[0] - 1),
        4 * (x[1] + 2)
    ])
```

```
def hess_f(x):
    return np.array([
        [2, 0],
        [0, 4]
    ])

x0 = np.array([0, 0])
minimizer = newton_minimize(f, grad_f, hess_f, x0)
print("Approximate minimizer:", minimizer)
```

```
Approximate minimizer: [ 1. -2.]
```

(c) Compare your result with the output of `scipy.optimize.minimize` using the `'Newton-CG'` method.

```
from scipy.optimize import minimize

res = minimize(f, x0, method='Newton-CG', jac=grad_f, hess=hess_f)
print("SciPy minimizer:", res.x)
```

```
SciPy minimizer: [ 1. -2.]
```

E.38 (a) We first expand the product:

$$\begin{aligned}&\left(I-\rho_k \mathbf{s}_k \mathbf{y}_k^T\right) H_k \left(I-\rho_k \mathbf{y}_k \mathbf{s}_k^T\right)\\ &= \left(H_k-\rho_k \mathbf{s}_k \mathbf{y}_k^T H_k\right)\left(I-\rho_k \mathbf{y}_k \mathbf{s}_k^T\right)\\ &= H_k-\rho_k H_k \mathbf{y}_k \mathbf{s}_k^T-\rho_k \mathbf{s}_k \mathbf{y}_k^T H_k+\rho_k^2 \mathbf{s}_k \mathbf{y}_k^T H_k \mathbf{y}_k \mathbf{s}_k^T.\end{aligned}$$

Therefore, using the BFGS formula,

$$H_{k+1}-H_k=-\rho_k \mathbf{s}_k \mathbf{y}_k^T H_k-\rho_k H_k \mathbf{y}_k \mathbf{s}_k^T+\rho_k^2 \mathbf{s}_k\left(\mathbf{y}_k^T H_k \mathbf{y}_k\right) \mathbf{s}_k^T+\rho_k \mathbf{s}_k \mathbf{s}_k^T.$$

(b) We introduce the notation

$$\mathbf{a}=\mathbf{s}_k, \quad \mathbf{b}=H_k \mathbf{y}_k, \quad t=\mathbf{y}_k^T H_k \mathbf{y}_k, \quad \rho=\rho_k.$$

Then we can rewrite each term in the expression for $H_{k+1}-H_k$ as

$$\mathbf{s}_k \mathbf{y}_k^T H_k=\mathbf{a} \mathbf{b}^T, \quad H_k \mathbf{y}_k \mathbf{s}_k^T=\mathbf{b} \mathbf{a}^T, \quad \mathbf{s}_k t \mathbf{s}_k^T=t\, \mathbf{a} \mathbf{a}^T, \quad \mathbf{s}_k \mathbf{s}_k^T=\mathbf{a} \mathbf{a}^T.$$

Substituting into the expression obtained in part (1) gives

$$\begin{aligned}H_{k+1}-H_k &= -\rho \mathbf{a} \mathbf{b}^T-\rho \mathbf{b} \mathbf{a}^T+\rho^2 t \mathbf{a} \mathbf{a}^T+\rho \mathbf{a} \mathbf{a}^T\\ &= \rho(1+\rho t) \mathbf{a} \mathbf{a}^T-\rho\left(\mathbf{a} \mathbf{b}^T+\mathbf{b} \mathbf{a}^T\right).\end{aligned}$$

We now look for scalars $\alpha, \beta \in \mathbb{R}$ such that

$$\rho(1+\rho t)\mathbf{a}\mathbf{a}^T - \rho\left(\mathbf{a}\mathbf{b}^T + \mathbf{b}\mathbf{a}^T\right) = 2\alpha\mathbf{a}\mathbf{a}^T + \beta\left(\mathbf{a}\mathbf{b}^T + \mathbf{b}\mathbf{a}^T\right).$$

Matching coefficients gives:

$$2\alpha = \rho(1+\rho t), \quad \beta = -\rho.$$

Thus, we can choose

$$\alpha = \frac{\rho(1+\rho t)}{2}, \quad \beta = -\rho.$$

We define:

$$\mathbf{v} = \mathbf{a}, \quad \mathbf{u} = \alpha\mathbf{a} + \beta\mathbf{b}.$$

Then:

$$\begin{aligned}\mathbf{u}\mathbf{v}^T + \mathbf{v}\mathbf{u}^T &= (\alpha\mathbf{a} + \beta\mathbf{b})\mathbf{a}^T + \mathbf{a}(\alpha\mathbf{a} + \beta\mathbf{b})^T \\ &= \alpha\mathbf{a}\mathbf{a}^T + \beta\mathbf{b}\mathbf{a}^T + \alpha\mathbf{a}\mathbf{a}^T + \beta\mathbf{a}\mathbf{b}^T \\ &= 2\alpha\mathbf{a}\mathbf{a}^T + \beta\left(\mathbf{a}\mathbf{b}^T + \mathbf{b}\mathbf{a}^T\right),\end{aligned}$$

which matches the expression for $H_{k+1} - H_k$. Therefore,

$$H_{k+1} - H_k = \mathbf{u}\mathbf{v}^T + \mathbf{v}\mathbf{u}^T.$$

(c) Since $\mathbf{u}\mathbf{v}^T$ and $\mathbf{v}\mathbf{u}^T$ are both rank-one matrices (or zero, in degenerate cases), their sum has rank at most two. Thus $H_{k+1} - H_k$ is a sum of two rank-one matrices and therefore

$$\operatorname{rank}(H_{k+1} - H_k) \leq 2.$$

This shows that the BFGS formula defines a rank-two update.

E.48 (a) The Lagrangian is:

$$\mathcal{L}(x, y, z, \lambda_1, \lambda_2) = x^2 + y^2 + z^2 + \lambda_1(x + y + z - 3) + \lambda_2(x - y - 1)$$

(b) We set the gradient of the Lagrangian to zero:

$$\frac{\partial \mathcal{L}}{\partial x} = 2x + \lambda_1 + \lambda_2 = 0 \tag{1}$$
$$\frac{\partial \mathcal{L}}{\partial y} = 2y + \lambda_1 - \lambda_2 = 0 \tag{2}$$
$$\frac{\partial \mathcal{L}}{\partial z} = 2z + \lambda_1 = 0 \tag{3}$$
$$\frac{\partial \mathcal{L}}{\partial \lambda_1} = x + y + z - 3 = 0 \tag{4}$$
$$\frac{\partial \mathcal{L}}{\partial \lambda_2} = x - y - 1 = 0 \tag{5}$$

(c) Solving the system:
From (5):

$$x = y + 1 \tag{6}$$

Plug (6) into (1):

$$2(y+1) + \lambda_1 + \lambda_2 = 0 \;\Rightarrow\; 2y + 2 + \lambda_1 + \lambda_2 = 0 \tag{1'}$$

Subtract (2) from (1'):

$$(2y + 2 + \lambda_1 + \lambda_2) - (2y + \lambda_1 - \lambda_2) = 0 \;\Rightarrow\; \lambda_2 = -1$$

Plug into (2):

$$2y + \lambda_1 - (-1) = 0 \;\Rightarrow\; \lambda_1 = -2y - 1$$

Plug into (3):

$$2z + \lambda_1 = 0 \;\Rightarrow\; 2z = -\lambda_1 = 2y + 1 \;\Rightarrow\; z = y + \frac{1}{2}$$

Now use constraint (4):

$$x + y + z = 3$$

Using (6) and $z = y + \frac{1}{2}$:

$$(y+1) + y + \left(y + \frac{1}{2}\right) = 3 \;\Rightarrow\; 3y + \frac{3}{2} = 3 \;\Rightarrow\; y = \frac{1}{2}$$

$$z = y + \frac{1}{2} = 1$$
$$x = y + 1 = \frac{3}{2}$$

Thus, the final solution is:

$$(x, y, z) = \left(\frac{3}{2}, \frac{1}{2}, 1\right)$$

(d) The objective function $f(x, y, z) = x^2 + y^2 + z^2$ is strictly convex, and the constraints are affine. Hence, this is a convex optimization problem with a unique global minimizer, and the KKT conditions are sufficient.

(e) Geometrically, we are looking for the point on the intersection of the two planes

$$x + y + z = 3, \qquad x - y = 1$$

that is closest to the origin. The feasible region is the intersection of two planes in $\mathbb{R}^3$, which forms a line. Minimizing $f(x, y, z) = x^2 + y^2 + z^2$ is equivalent to projecting the origin orthogonally onto this line.

The gradient of the objective is

$$\nabla f(x, y, z) = (2x, 2y, 2z),$$

and the gradients of the constraints (i.e., the normals to the two planes) are

$$\nabla g_1 = (1, 1, 1), \qquad \nabla g_2 = (1, -1, 0).$$

At the optimal point, the vector from the origin to the solution lies in the feasible set and must be orthogonal to it, i.e., it must be orthogonal to the direction of the line defined by the intersection. Equivalently, the gradient of the objective must lie in the span of the normals to the constraints. This is exactly what the Lagrange multipliers condition enforces:

$$\nabla f(x, y, z) + \lambda_1 \nabla g_1(x, y, z) + \lambda_2 \nabla g_2(x, y, z) = 0$$

Therefore, the point

$$(x, y, z) = \left(\frac{3}{2}, \frac{1}{2}, 1\right)$$

obtained from solving the Lagrange system is the unique point on the feasible set that is closest to the origin. This confirms that the geometric and analytic views are fully consistent.

E.50 First, we redefine the constraints as inequalities less than or equal to zero:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^2} \quad & f(\mathbf{x}) = x_1^2 + x_2^2 \\ \text{subject to} \quad & g_1(\mathbf{x}) = 1 - x_1 - x_2 \leq 0 \\ & g_2(\mathbf{x}) = -x_1 \leq 0 \\ & g_3(\mathbf{x}) = -x_2 \leq 0 \end{aligned}$$

Next, we introduce Lagrange multipliers for each inequality constraint, and define the Lagrangian:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) = f(\mathbf{x}) + \alpha_1 g_1(\mathbf{x}) + \alpha_2 g_2(\mathbf{x}) + \alpha_3 g_3(\mathbf{x})$$

Substituting the functions:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) = x_1^2 + x_2^2 + \alpha_1(1 - x_1 - x_2) + \alpha_2(-x_1) + \alpha_3(-x_2)$$

The objective function in this case is convex with linear constraints, thus it has a unique global minimum, and the KKT conditions are both necessary and sufficient to find this minimum.

The KKT conditions are as follows:

- Stationarity: We take the partial derivatives of $\mathcal{L}$ with respect to x_1 and x_2 and set them to zero:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial x_1} &= 2x_1 - \alpha_1 - \alpha_2 = 0 \\ \frac{\partial \mathcal{L}}{\partial x_2} &= 2x_2 - \alpha_1 - \alpha_3 = 0 \end{aligned}$$

- Primal feasibility: The constraints of the primal problem must be satisfied:

$$\begin{aligned} 1 - x_1 - x_2 &\leq 0 \\ -x_1 &\leq 0 \Rightarrow x_1 \geq 0 \\ -x_2 &\leq 0 \Rightarrow x_2 \geq 0 \end{aligned}$$

- Dual feasibility: The lagrange multipliers associated with the inequality constraints must be nonnegative:

$$\alpha_1 \geq 0, \quad \alpha_2 \geq 0, \quad \alpha_3 \geq 0$$

- Complementary slackness: For each constraint $g_i(\mathbf{x})$, the product $\alpha_i g_i(\mathbf{x})$ must be zero:

$$\alpha_1(x_1 + x_2 - 1) = 0$$
$$\alpha_2 x_1 = 0$$
$$\alpha_3 x_2 = 0$$

Begin with the constraint(s) violated by the unconstrained minimizer

Based on these conditions, we will explore three potential cases to identify feasible solutions that satisfy the KKT conditions.

Case 1: Suppose $x_1 > 0$ and $x_2 > 0$:

- From the second complementary condition $\alpha_2 x_1 = 0$, we get $\alpha_2 = 0$ (as $x_1 > 0$).
- From the third complementary condition $\alpha_3 x_2 = 0$, we get $\alpha_3 = 0$ (as $x_2 > 0$).
- The stationarity conditions become: $2x_1 - \alpha_1 = 0$ and $2x_2 - \alpha_1 = 0$, which implies $x_1 = x_2$.
- From the first complementary condition, we have $\alpha_1(x_1 + x_2 - 1) = 0$. Given $x_1, x_2 > 0$, α_1 must be nonzero, thus $x_1 + x_2 = 1$.
- The only feasible solution under these conditions is $x_1 = x_2 = 0.5$, which satisfies all the KKT conditions.

Case 2: Suppose $x_1 = 0$ and $x_2 > 0$:

- From the third complementary condition $\alpha_3 x_2 = 0$, we get $\alpha_3 = 0$ (as $x_2 > 0$).
- The stationarity condition for x_2 becomes $2x_2 - \alpha_1 = 0$, which implies $\alpha_1 = 2x_2$.
- From the first complementary condition $\alpha_1(x_1 + x_2 - 1) = \alpha_1(x_2 - 1) = 0$. Since $\alpha_1 = 2x_2 > 0$, we must have $x_2 = 1$ and $\alpha_1 = 2$.
- The stationarity condition for x_1 yields $2x_1 - \alpha_1 - \alpha_2 = -\alpha_1 - \alpha 2 = 0$. With $\alpha_1 = 2$, this implies $\alpha_2 = -2$, which is not possible since $\alpha_2 \geq 0$.

Analogously, the case $x_1 > 0$ and $x_2 = 0$ yields no feasible solutions, and the case $x_1 = x_2 = 0$ is immediately ruled out as it violates the primary constraint $x_1 + x_2 \geq 1$.

Thus, the only feasible solution that satisfies all the KKT conditions, which is also the global minimum (due to the convexity of the function) is $x_1 = x_2 = 0.5$. The optimal value of the objective function is:

$$f(0.5, 0.5) = 0.5^2 + 0.5^2 = 0.25$$

E.53 (a) The function $f(x) = \frac{1}{2}x^2$ is convex because it is a quadratic function with positive second derivative $f''(x) = 1 > 0$. The constraint $x \geq 1$ can be written as $-x + 1 \leq 0$, which is an affine function, and hence convex. Therefore, this is a convex optimization problem.

(b) The unconstrained minimum of $f(x)$ occurs at $x = 0$, but this point is not feasible. The feasible region is $x \geq 1$, so the minimum must occur at the boundary:

$$x^* = 1, \quad f(x^*) = \frac{1}{2}(1)^2 = \frac{1}{2}.$$

(c) We rewrite the constraint in the form $g(x) = 1 - x \leq 0$, and define the Lagrangian:

$$\mathcal{L}(x, \alpha) = \frac{1}{2}x^2 + \alpha(1 - x),$$

where $\alpha \geq 0$ is the Lagrange multiplier.

(d) To derive the dual function, we minimize the Lagrangian over $x \in \mathbb{R}$:

$$\theta(\alpha) = \inf_{x \in \mathbb{R}} \left[\frac{1}{2}x^2 + \alpha(1 - x) \right].$$

To compute the minimum, we take the derivative with respect to x and set it to zero:

$$\frac{d}{dx}\mathcal{L}(x, \alpha) = x - \alpha = 0 \quad \Rightarrow \quad x = \alpha.$$

Substituting $x = \alpha$ into the Lagrangian yields:

$$\theta(\alpha) = \frac{1}{2}\alpha^2 + \alpha(1 - \alpha) = \alpha - \frac{1}{2}\alpha^2.$$

(e) The dual problem is:

$$\sup_{\alpha \geq 0} \theta(\alpha) = \alpha - \frac{1}{2}\alpha^2.$$

This is a concave quadratic function. To find its maximum, we differentiate:

$$\frac{d\theta}{d\alpha} = 1 - \alpha = 0 \quad \Rightarrow \quad \alpha^* = 1.$$

Substituting into the dual function gives:

$$\theta(\alpha^*) = 1 - \frac{1}{2} = \frac{1}{2}.$$

(f) The primal optimal value is $f(x^*) = 1/2$, and the dual optimal value is $\theta(\alpha^*) = 1/2$. Since these values are equal, strong duality holds.

E.56 (a) We first introduce slack variables $s_1, s_2 \geq 0$:

$$\begin{aligned} \max_{x_1,x_2} \quad & 3x_1 + 5x_2, \\ \text{subject to} \quad & x_1 + 2x_2 + s_1 = 8, \\ & 3x_1 + 2x_2 + s_2 = 12, \\ & x_1, x_2, s_1, s_2 \geq 0. \end{aligned}$$

The initial simplex tableau is:

	x_1	x_2	s_1	s_2	RHS
s_1	1	2	1	0	8
s_2	3	2	0	1	12
f	-3	-5	0	0	0

The most negative coefficient in the objective row is -5, so x_2 enters. Compute ratios:

$$\frac{8}{2} = 4, \quad \frac{12}{2} = 6,$$

so s_1 leaves.

We now pivot on the entry 2 in row 1, column x_2:

- Normalize row 1:

$$R_1 \leftarrow 0.5R_1 = (0.5, 1, 0.5, 0 \mid 4).$$

- Eliminate x_2 from row 2:

$$R_2 \leftarrow R_2 - 2R_1 = (2, 0, -1, 1 \mid 4).$$

- Update objective row:

$$f \leftarrow f + 5R_1 = (-0.5, 0, 2.5, 0 \mid 20).$$

The new tableau is:

	x_1	x_2	s_1	s_2	RHS
x_2	0.5	1	0.5	0	4
s_2	2	0	-1	1	4
f	-0.5	0	2.5	0	20

The only negative coefficient in the objective row is -0.5, so x_1 enters.
Compute ratios:

$$\frac{4}{0.5} = 8, \quad \frac{4}{2} = 2,$$

so s_2 leaves.
We pivot on the entry 2 in row 2, column x_1:

- Normalize row 2:

$$R_2 \leftarrow 0.5R_2 = (1, 0, -0.5, 0.5 \mid 2).$$

- Eliminate x_1 from row 1:

$$R_1 \leftarrow R_1 - 0.5R_2 = (0, 1, 0.75, -0.25 \mid 3).$$

- Update objective row:

$$f \leftarrow f + 0.5R_2 = (0, 0, 2.25, 0.25 \mid 21).$$

The final tableau is:

	x_1	x_2	s_1	s_2	RHS
x_2	0	1	0.75	-0.25	3
x_1	1	0	-0.5	0.5	2
f	0	0	2.25	0.25	21

Thus, the optimal solution is:

$$x_1^* = 2,\ x_2^* = 3,\ f^* = 21.$$

(b) The following code uses the `scipy.optimize.linprog` function to solve the same LP. Since `linprog` expects a minimization problem by default, we negate the objective function coefficients to convert our original maximization problem into an equivalent minimization problem.

```python
from scipy.optimize import linprog

# Objective function coefficients (maximize 3x1 + 5x2 -> minimize
    -3x1 - 5x2)
c = [-3, -5]

# Inequality constraints
A = [[1, 2],
     [3, 2]]
b = [8, 12]

# Variable bounds
bounds = [(0, None), (0, None)]

# Solve the LP
res = linprog(c, A_ub=A, b_ub=b, bounds=bounds)
res
```

```
        message: Optimization terminated successfully. (HiGHS Status 7: Optimal)
        success: True
         status: 0
            fun: -21.0
              x: [ 2.000e+00  3.000e+00]
            nit: 2
          lower:  residual: [ 2.000e+00  3.000e+00]
                 marginals: [ 0.000e+00  0.000e+00]
          upper:  residual: [       inf        inf]
                 marginals: [ 0.000e+00  0.000e+00]
          eqlin:  residual: []
                 marginals: []
        ineqlin:  residual: [ 0.000e+00  0.000e+00]
                 marginals: [-2.250e+00 -2.500e-01]
 mip_node_count: 0
 mip_dual_bound: 0.0
        mip_gap: 0.0
```

The solver returns the optimal solution $x_1 = 2$, $x_2 = 3$, with an optimal objective value of $f^* = 21$, which matches the result obtained by the simplex method in part (a).

E.61 (a) Write a Python function that implements the interior-point method using a logarithmic barrier and adaptive barrier parameter reduction.

```
import numpy as np
import matplotlib.pyplot as plt

from scipy.optimize import minimize

def interior_point_method(f, constraints, x0, mu_init=1.0,
    mu_min=1e-6, mu_factor=0.1, verbose=False):
    """
    Interior-point method using dynamic barrier parameter
        reduction.

    Parameters:
    f           : Objective function f(x)
    constraints : List of inequality functions g_i(x) <= 0
    x0          : Initial strictly feasible point
    mu_init     : Initial barrier parameter (default 1.0)
    mu_min      : Stopping threshold for mu (default 1e-6)
    mu_factor   : Multiplicative reduction factor for mu (default
        0.1)

    Returns:
    x_final     : Final solution x*
    f_val       : Final objective value
    trajectory  : List of iterates across outer iterations
    """
    x = x0
    trajectory = [x]
    mu = mu_init

    while mu >= mu_min:
        def barrier_obj(x_init):
            if any(g(x_init) >= 0 for g in constraints):
                return np.inf
            penalty = -mu * sum(np.log(-g(x_init)) for g in
                constraints)
            return f(x_init) + penalty

        res = minimize(barrier_obj, x)
        x = res.x
        trajectory.append(x)
```

```
        if verbose:
            print(f"mu = {mu:.1e} -> x = {x}, f = {f(x):.6f}")

        mu *= mu_factor

    return x, f(x), np.array(trajectory)
```

(b) Test your implementation on the following constrained optimization problem:

$$\begin{aligned} \min_{x>0,\, y>0} \quad & (x-2)^2 + (y-1)^2 \\ \text{subject to} \quad & x + y \leq 3. \end{aligned}$$

starting from the point $\mathbf{x} = (0.5, 0.5)$.

```
def f(x):
    return (x[0] - 2)**2 + (x[1] - 1)**2

constraints = [
    lambda x: -x[0],
    lambda x: -x[1],
    lambda x: x[0] + x[1] - 3
]

mu_values = [1.0, 0.1, 0.01]
x0 = np.array([0.5, 0.5]) # strictly feasible

x_star, f_star, trajectory = interior_point_method(
    f, constraints, x0=np.array([0.5, 0.5]), mu_init=1.0,
        mu_min=1e-6,
    mu_factor=0.1, verbose=True
)
```

```
mu = 1.0e+00 -> x = [1.53995381 0.82288575], f = 0.243012
mu = 1.0e-01 -> x = [1.84654068 0.8765076 ], f = 0.038800
mu = 1.0e-02 -> x = [1.95057383 0.95325606], f = 0.004628
mu = 1.0e-03 -> x = [1.98424887 0.98450483], f = 0.000488
mu = 1.0e-04 -> x = [1.9950101 0.9950279], f = 0.000050
mu = 1.0e-05 -> x = [1.99841978 0.99842179], f = 0.000005
mu = 1.0e-06 -> x = [1.99949887 0.99949917], f = 0.000001
```

(c) Compare your result with the solution obtained using `scipy.optimize.minimize` with built-in constraint handling.

```
from scipy.optimize import Bounds

scipy_res = minimize(
    f,
    x0=[0.5, 0.5],
    bounds=Bounds([0, 0], [np.inf, np.inf]),
    constraints=[{'type': 'ineq', 'fun': lambda x: 3 - x[0] - x[1]}]
)

print("\nscipy.optimize.minimize solution:", scipy_res.x)
print("Function value:", scipy_res.fun)
```

```
scipy.optimize.minimize solution: [1.99999999 0.99999999]
Function value: 2.4980319713623524e-16
```

(d) Plot the optimization trajectory over the level sets of f, including the feasible region defined by the constraints.

```
def plot_trajectory(f, trajectory):
    x_vals = np.linspace(0.01, 3, 300)
    y_vals = np.linspace(0.01, 3, 300)
    X, Y = np.meshgrid(x_vals, y_vals)
    Z = f(np.array([X, Y]))

    feasible_mask = (X > 0) & (Y > 0) & (X + Y <= 3)

    plt.contour(X, Y, Z, levels=30, cmap='gray')
    plt.contourf(X, Y, feasible_mask, levels=[0.5, 1.5],
        colors=['#dddddd'], alpha=0.7)

    traj = np.array(trajectory)
    plt.plot(traj[:, 0], traj[:, 1], 'ro-', label='Interior-point
        path')
    plt.scatter([traj[0, 0]], [traj[0, 1]], color='black',
        label='Start', zorder=5)
    plt.scatter([x_star[0]], [x_star[1]], color='blue',
        label='Final point', zorder=5)

    plt.title('Interior-Point Optimization Trajectory')
```

```
    plt.xlabel('$x$')
    plt.ylabel('$y$')
    plt.xlim(0, 3)
    plt.ylim(0, 3)
    plt.grid(alpha=0.5)
    plt.legend()

plot_trajectory(f, trajectory)
```

The resulting plot is shown in Figure E.1.

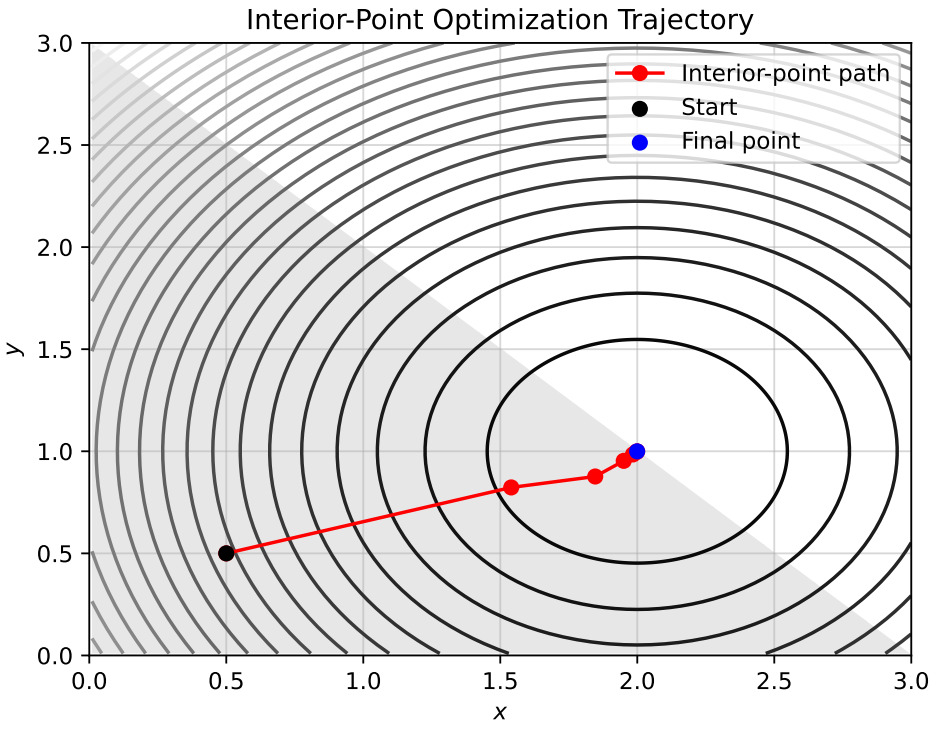


Figure E.1: Optimization trajectory of the interior-point method applied to the constrained problem $\min(x-2)^2+(y-1)^2$ subject to $x>0$, $y>0$, and $x+y\leq 3$. The background shows contour lines of the objective function, while the shaded region indicates the feasible set defined by the constraints. The red markers trace the iterates produced by the interior-point method, starting from a strictly feasible initial point and converging to the constrained optimum on the boundary of the feasible region.

Appendix F

Python Libraries

F.4

```
def create_checkerboard(n, m):
    board = np.zeros((n, m), dtype=int)
    board[::2, 1::2] = 1 # rows 0,2,4,... -> set columns 1,3,5,...
    board[1::2, ::2] = 1 # rows 1,3,5,... -> set columns 0,2,4,...
    return board

create_checkerboard(5, 6)
```

```
array([[0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1]])
```

F.13

```
def multiplication_table(n):
    x = np.arange(n)              # Shape: (n,)
    y = np.arange(n).reshape(-1, 1) # Shape: (n, 1)
    return y * x                  # Broadcasting to shape (n, n)

multiplication_table(10)
```

```
array([[ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18],
       [ 0,  3,  6,  9, 12, 15, 18, 21, 24, 27],
       [ 0,  4,  8, 12, 16, 20, 24, 28, 32, 36],
       [ 0,  5, 10, 15, 20, 25, 30, 35, 40, 45],
```

```
        [ 0,  6, 12, 18, 24, 30, 36, 42, 48, 54],
        [ 0,  7, 14, 21, 28, 35, 42, 49, 56, 63],
        [ 0,  8, 16, 24, 32, 40, 48, 56, 64, 72],
        [ 0,  9, 18, 27, 36, 45, 54, 63, 72, 81]])
```

F.16

```
def find_peaks(x):
    # Check all elements from index 1 to len(x)-2
    peaks = x[1:-1][(x[1:-1] >= x[:-2]) & (x[1:-1] >= x[2:])]
    return peaks

# Example usage
x = np.array([5, 8, 3, 1, 10, 9, 12])
print(find_peaks(x))
```

```
[ 8 10]
```

F.19

```
np.random.seed(0)

# Generate the array and print it
arr = np.random.randint(1, 101, size=(10, 10))
print("Array:\n", arr)

# Minimum and maximum
print("Min:", np.min(arr))
print("Max:", np.max(arr))

# Average of even-indexed rows (0, 2, 4, 6, 8)
even_row_means = np.mean(arr[::2], axis=1)
print("Average of even-indexed rows:", even_row_means)

# Standard deviation of odd-indexed columns (1, 3, 5, 7, 9)
odd_col_stds = np.std(arr[:, 1::2], axis=0)
print("Standard deviation of odd-indexed columns:\n", odd_col_stds)

# Number of rows with at least one number < 5
rows_with_small_values = np.any(arr < 5, axis=1)
print("Rows with value < 5:", np.count_nonzero(rows_with_small_values))

# Columns with at least one number > 90
cols_with_large_values = np.any(arr > 90, axis=0)
```

```
print("Column indices with value > 90:",
    np.where(cols_with_large_values)[0])

# Numbers that appear more than twice
unique, counts = np.unique(arr, return_counts=True)
more_than_twice = unique[counts > 2]
print("Numbers that appear more than twice:", more_than_twice)

# Number of rows with duplicate values
rows_with_duplicates = [len(np.unique(row)) < len(row) for row in arr]
print("Rows with duplicates:", np.count_nonzero(rows_with_duplicates))
```

```
Array:
 [[ 45  48  65  68  68  10  84  22  37  88]
 [ 71  89  89  13  59  66  40  88  47  89]
 [ 82  38  26  78  73  10  21  81  70  80]
 [ 48  65  83 100  89  50  30  20  20  15]
 [ 40  33  66  10  58  33  32  75  24  36]
 [ 76  56  29  35   1   1  37  54   6  39]
 [ 18  80   5  43  59  32   2  66  42  58]
 [ 36  12  47  83  92   1  15 100  54  13]
 [ 43  85  76  69   7  69  48   4  77  53]
 [ 79  16  21 100  59  24  80  14  86  49]]
Min: 1
Max: 100
Average of even-indexed rows: [53.5 55.9 40.7 40.5 53.1]
Standard deviation of odd-indexed columns:
 [26.25947448 31.33831521 23.97164992 32.92476272 26.0959767 ]
Rows with value < 5: 4
Column indices with value > 90: [3 4 7]
Numbers that appear more than twice: [  1  10  48  59  66  80  89 100]
Rows with duplicates: 6
```

We now compute the probability that a specific number appears at least three times in the matrix using the binomial distribution:

```
from scipy.stats import binom

# Parameters:
n = 100       # total number of elements in the matrix
p = 1 / 100   # probability of a specific number (uniform from 1 to 100)
```

```
# Probability that a specific number appears at least 3 times:
prob_at_least_3 = 1 - binom.cdf(2, n, p)
print("Probability that a specific number appears at least 3 times:",
    prob_at_least_3)

# Expected number of such numbers in the matrix:
expected_count = 100 * prob_at_least_3
print("Expected number of numbers that appear at least 3 times:",
    expected_count)
```

```
Probability that a specific number appears at least 3 times: 0.07937320225218047
Expected number of numbers that appear at least 3 times: 7.937320225218047 5
```

F.25

```
def find_saddle_points(A):
    # Row-wise max and column-wise min masks
    row_max_mask = A == A.max(axis=1, keepdims=True)
    col_min_mask = A == A.min(axis=0, keepdims=True)

    # Saddle points: max in row AND min in column
    saddle_mask = row_max_mask & col_min_mask

    # Use np.where and zip to get (i, j) index tuples
    i, j = np.where(saddle_mask)
    if len(i) == 0:
        return None

    return (int(i[0]), int(j[0]))

# Example
A = np.array([
    [3, 8, 4, 7],
    [5, 9, 1, 6],
    [1, 3, 0, 2],
    [6, 4, 2, 8]
])

find_saddle_points(A)

# Example
A = np.array([
```

```
    [3, 8, 4, 7],
    [5, 9, 1, 6],
    [1, 3, 0, 2],
    [6, 4, 2, 8]
])

find_saddle_points(A)
```

```
(2, 1)
```

F.28

```
def is_magic_square(A):
    # Check that A is square
    if A.ndim != 2 or A.shape[0] != A.shape[1]:
        return False

    n = A.shape[0]

    # Flatten and check that A contains exactly the numbers 1 to n^2
    flattened = A.flatten()
    if set(flattened) != set(range(1, n*n + 1)):
        return False

    # Compute the target sum (e.g., sum of the first row)
    target = np.sum(A[0])

    # Check all row sums
    if not np.all(A.sum(axis=1) == target):
        return False

    # Check all column sums
    if not np.all(A.sum(axis=0) == target):
        return False

    # Check main diagonal
    if np.sum(np.diag(A)) != target:
        return False

    # Check secondary diagonal
    if np.sum(np.diag(np.fliplr(A))) != target:
        return False
```

```
    return True

A = np.array([
    [2, 16, 13, 3],
    [11, 5, 8, 10],
    [7, 9, 12, 6],
    [14, 4, 1, 15]
])

is_magic_square(A)
```

```
True
```

F.36

```
import numpy as np
import matplotlib.pyplot as plt

# Years and transistor counts (in millions)
years = np.array([
    1971, 1972, 1974, 1978, 1982, 1985, 1989, 1993, 1997, 2000,
    2002, 2006, 2008, 2012, 2015, 2021, 2023
])

transistors_million = np.array([
    0.0023, 0.0035, 0.0060, 0.029, 0.134, 0.275, 1.18, 3.10, 7.50, 42.0,
    220.0, 291.0, 731.0, 1400.0, 1750.0, 6000.0, 46000.0
])

# Convert to raw counts
transistors = transistors_million * 1e6

# Moore's Law prediction
y0, n0 = years[0], transistors[0]
x_pred = np.arange(y0, years[-1] + 1)
log_n_pred = np.log10(n0) + (x_pred - y0) / 2 * np.log10(2)

# Plotting
plt.plot(years, np.log10(transistors), '*', markersize=12, color='r',
        markeredgecolor='r', label='Observed (log_10)')
plt.plot(x_pred, log_n_pred, 'k--', linewidth=2, label="Moore's Law
    prediction")
```

```
plt.xlabel('Year', fontsize=12)
plt.ylabel(r'$\log_{10}(\mathrm{Transistor~count})$', fontsize=12)
plt.title("Moore's Law: Transistor Growth Over Time", fontsize=14)
plt.legend()
plt.grid(alpha=0.5)
```

The resulting plot is shown in Figure F.1.

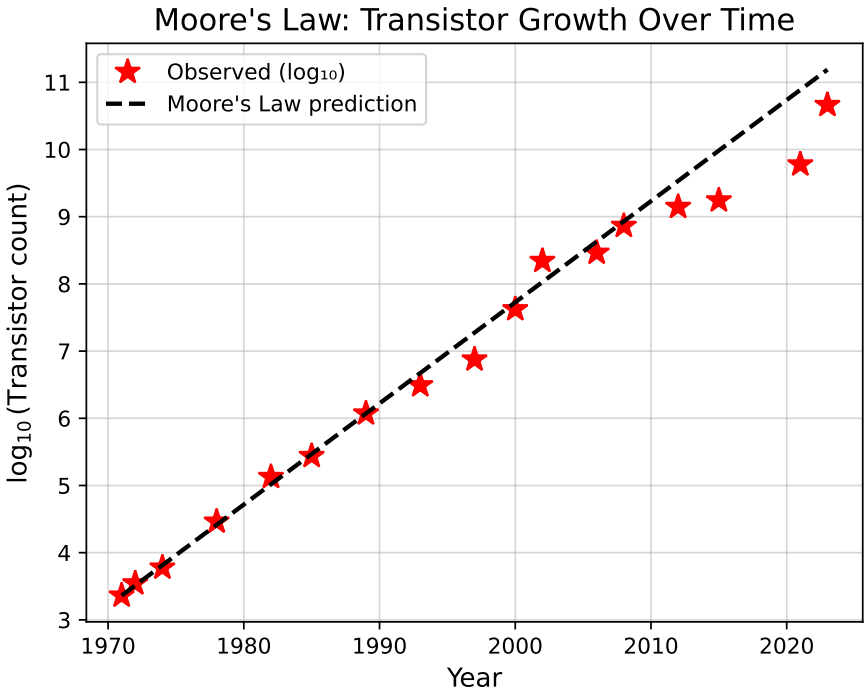


Figure F.1: Moore's law predictions vs. actual number of processors

F.41

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml

# Load MNIST dataset
X, y = fetch_openml('mnist_784', return_X_y=True, as_frame=False)

X = X[:50]
```

```
y = y[:50]

images = X.reshape(-1, 28, 28)

# Plot in a 5x10 grid
fig, axes = plt.subplots(5, 10, figsize=(12, 6))

for i, ax in enumerate(axes.flat):
    ax.imshow(images[i], cmap='gray')
    ax.axis('off')
    ax.text(1, 4, str(y[i]), color='red', fontsize=10, weight='bold')

fig.suptitle("First 50 MNIST Digit Images", fontsize=16)
plt.tight_layout()
```

The resulting plot is shown in Figure F.2.

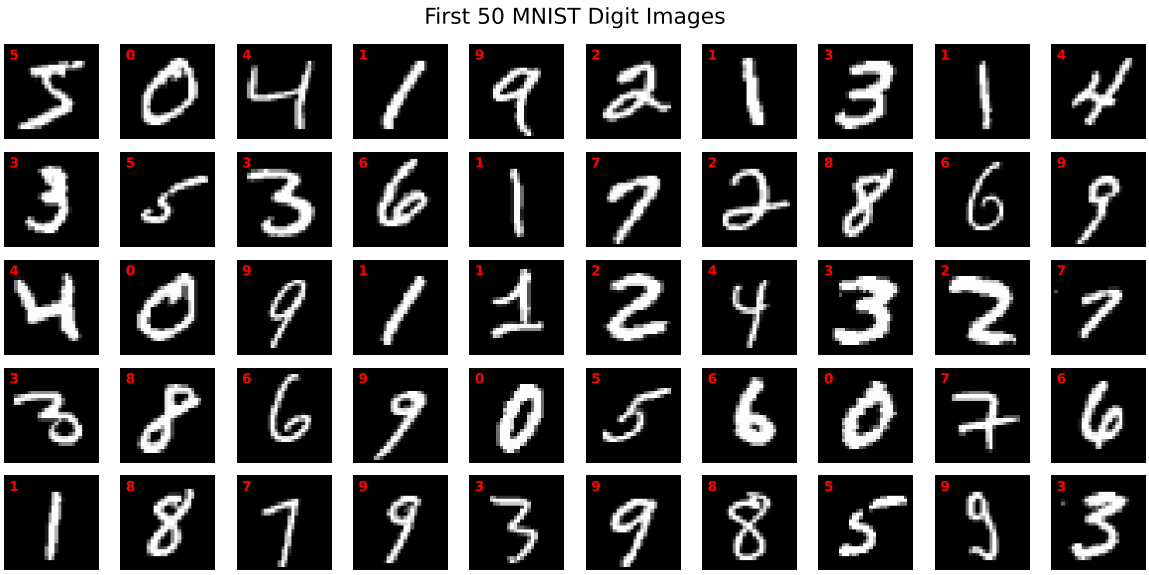


Figure F.2: The first 50 digit images from the MNIST dataset, arranged in a 5×10 grid.

F.43

```
import numpy as np
import matplotlib.pyplot as plt
```

```
# Create grid
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)

# Compute the function values
Z = X**2 - Y**2

# Create the 3D figure and axis
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

# Plot the surface
surf = ax.plot_surface(X, Y, Z, cmap='coolwarm', edgecolor='none')

# Label axes and add title
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
ax.set_zlabel('$f(x, y)$', labelpad=-2)
ax.set_title("3D Surface Plot of $f(x, y) = x^2 - y^2$")
```

The resulting plot is shown in Figure F.3.

F.45 (a) Select all the players who play for the Northeastern Huskies. Display only their name, class, and position.

```
df[df['Team'] == 'Northeastern Huskies'][['Name', 'Class',
    'Position']]
```

```
            Name  Class Position
2   Liam Koelsch    Fr.   Center
6  Xander Alarie  R-Fr.  Forward
```

(b) Show all players that play as centers or are listed as seniors.

```
df[(df['Position'] == 'Center') | (df['Class'] == 'Sr.')]
```

```
             Name                  Team       Position Class Height
0     Jordan Pope       Texas Longhorns          Guard   Sr.    6-1
1   Langston Love     Georgetown Hoyas  Guard/Forward   Sr.    6-5
2    Liam Koelsch  Northeastern Huskies         Center   Fr.    6-9
```

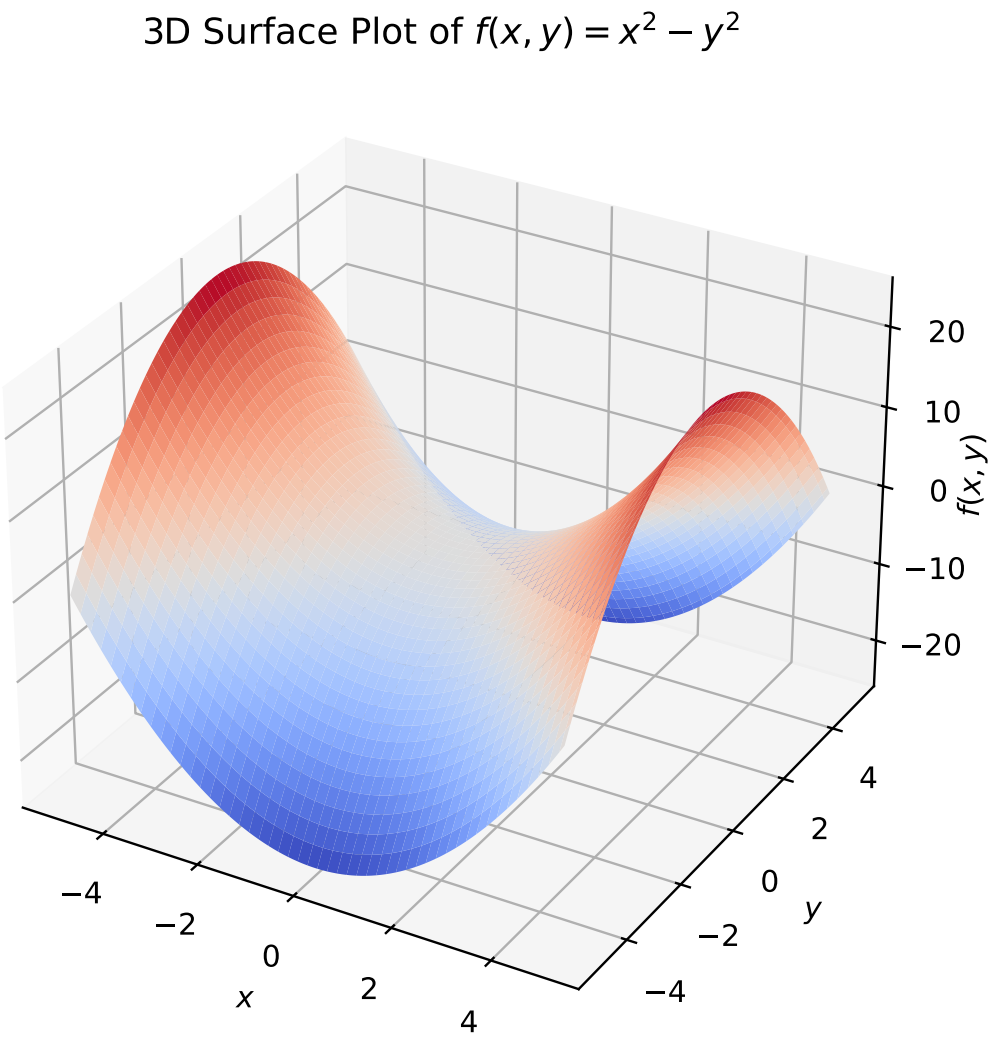


Figure F.3: Three-dimensional surface plot of the function $f(x, y) = x^2 - y^2$

```
4     Favour Aire   Coppin State Eagles         Center   Sr.   6-11
5  Michael Miller    Seattle U Redhawks         Center   Sr.    6-8
7    Trace Salton      Marist Red Foxes  Guard/Forward   Sr.    6-5
```

(c) Count how many players come from teams whose names include the word "Texas".

```
df[df['Team'].str.contains('Texas')].shape[0]
```

```
2
```

(d) Show the number of players at each position.

```
df['Position'].value_counts()
```

```
Position
Center           3
Guard/Forward    2
Forward          2
Guard            1
Name: count, dtype: int64
```

(e) List all players whose height is at least 6–8 (80 inches).

```
def parse_height(h):
    # Convert height to inches
    feet, inches = map(int, h.split('-'))
    return feet * 12 + inches
df['Height_inch'] = df['Height'].apply(parse_height)

df[df['Height_inch'] >= 80]
```

	Name	Team	Position	Class	Height	Height_inch
2	Liam Koelsch	Northeastern Huskies	Center	Fr.	6-9	81
3	Noah Pagotto	East Texas A&M	Forward	Jr.	6-8	80
4	Favour Aire	Coppin State Eagles	Center	Sr.	6-11	83
5	Michael Miller	Seattle U Redhawks	Center	Sr.	6-8	80
6	Xander Alarie	Northeastern Huskies	Forward	R-Fr.	6-8	80

(f) Show the name and height of the tallest player in the table.

```
df.loc[df['Height_inch'].idxmax(), ['Name', 'Height']]
```

```
Name      Favour Aire
Height           6-11
Name: 4, dtype: object
```

(g) Find the team with the lowest average player height.

```
df.groupby('Team')['Height_inch'].mean().idxmin()
```

```
'Texas Longhorns'
```

(h) Print the average age of the three tallest players (Assume: Fr. = 18, R-Fr. = 19, Jr. = 20, Sr. = 21).

```
# Map class to approximate age
class_age = {'Fr.': 18, 'R-Fr.': 19, 'Jr.': 20, 'Sr.': 21}
df['Age'] = df['Class'].map(class_age)

df.nlargest(3, 'Height_inch')['Age'].mean()
```

```
19.666666666666668
```

F.48 Before starting, remove the dollar sign from the `item_price` column and convert it to a numeric type.

```
df['item_price'] = df['item_price'].str.replace('$', '').astype(float)
```

(a) How many times was a `'Chicken Bowl'` ordered (in total)?

```
df[df['item_name'] == 'Chicken Bowl']['quantity'].sum()
```

```
761
```

(b) Which item was ordered the highest number of times in a single order? For that item, show its name and quantity.

```
df.loc[df['quantity'].idxmax()][['item_name', 'quantity']]
```

```
item_name    Chips and Fresh Tomato Salsa
quantity                                15
Name: 3598, dtype: object
```

(c) Which items have a unit price greater than $10.00? Compute the unit price of each item as `item_price` divided by `quantity`, and display the unique item names whose unit price exceeds 10.

```
df['unit_price'] = df['item_price'] / df['quantity']
df[df['unit_price'] > 10]['item_name'].unique()
```

```
array(['Chicken Bowl', 'Steak Burrito', 'Chicken Burrito',
       'Barbacoa Bowl', 'Veggie Burrito', 'Veggie Bowl',
       'Chicken Soft Tacos', 'Steak Bowl', 'Carnitas Burrito',
       'Carnitas Bowl', 'Barbacoa Burrito', 'Chicken Salad Bowl',
       'Barbacoa Crispy Tacos', 'Veggie Salad Bowl', 'Chicken Salad',
       'Steak Salad Bowl', 'Chicken Crispy Tacos', 'Veggie Soft Tacos',
       'Barbacoa Soft Tacos', 'Carnitas Crispy Tacos',
       'Carnitas Salad Bowl', 'Barbacoa Salad Bowl', 'Steak Soft Tacos',
       'Carnitas Soft Tacos', 'Steak Crispy Tacos'], dtype=object)
```

(d) Which items appear in at least 100 different orders (i.e., in 100 or more unique `order_id` values)?

```
df.groupby('item_name').filter(lambda g: g['order_id'].nunique()
    >= 200)['item_name'].unique()
```

```
array(['Chicken Bowl', 'Steak Burrito', 'Chips and Guacamole',
       'Chicken Burrito', 'Chips', 'Canned Soft Drink'], dtype=object)
```

(e) Display a bar plot showing the total quantity ordered for each item. Label the axes appropriately.

```
import matplotlib.pyplot as plt

item_counts = df.groupby('item_name')['quantity'].sum()

plt.figure(figsize=(8, 4))
item_counts.plot(kind='bar')
plt.title('Total Quantity Ordered per Item')
plt.xlabel('Item Name')
plt.ylabel('Total Quantity')
```

The resulting plot is shown in Figure F.4.

(f) (*) Identify the top 5 items most frequently ordered together with a `'Chicken Bowl'`. Display their names and how many times they appeared in the same order as `'Chicken Bowl'`.

```
# Find all order_ids that include a Chicken Bowl
orders_with_cb = df[df['item_name'] == 'Chicken
    Bowl']['order_id'].unique()

# Filter to only those orders
co_orders = df[df['order_id'].isin(orders_with_cb)]

# Remove Chicken Bowl itself from the result
co_items = co_orders[co_orders['item_name'] != 'Chicken Bowl']

# Count co-occurring items
co_item_counts = co_items['item_name'].value_counts().nlargest(5)
print(co_item_counts)
```

```
item_name
Chips and Guacamole    153
```

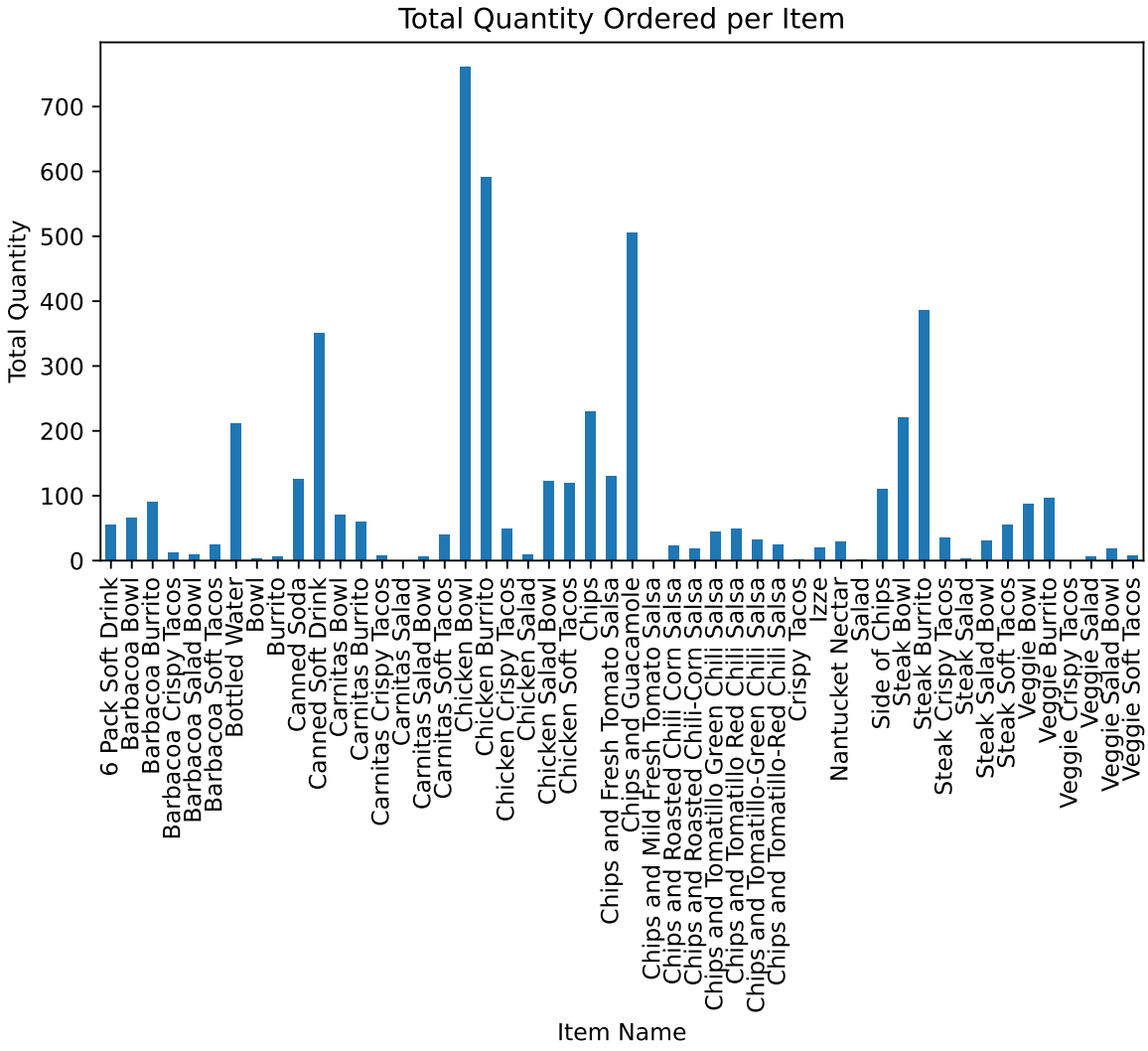


Figure F.4: Bar plot showing the total quantity ordered for each menu item in the dataset.

```
Chips                  124
Canned Soft Drink      119
Chicken Burrito         89
Bottled Water           72
Name: count, dtype: int64
```

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
import time

np.random.seed(42)

# Define the objective function
def f(x):
    return np.sin(x[0]) + x[0]**2 / 10

# Plot the function over [-10, 10]
x_vals = np.linspace(-10, 10, 400)
y_vals = [f([x]) for x in x_vals]

plt.figure(figsize=(8, 5))
plt.plot(x_vals, y_vals, label=r'$f(x) = \sin(x) + \frac{x^2}{10}$')

# Initial guess
x0 = [0.0]

# Methods to compare
methods = ['BFGS', 'Nelder-Mead', 'CG']
results = []

# Perform optimization and timing
for method in methods:
    start = time.time()
    res = minimize(f, x0, method=method)
    duration = time.time() - start
    results.append((method, res.fun, res.x[0], duration))
    plt.plot(res.x[0], res.fun, 'o', label=f'{method} min')

plt.xlabel('$x$')
plt.ylabel('$f(x)$')
plt.title("Optimization of $f(x) = \sin(x) + x^2 / 10$")
plt.grid(alpha=0.5)
plt.legend()

# Display results
print("Method comparison:")
```

```
for method, value, x, duration in results:
    print(f"{method:12} -> Minimum at x = {x:.5f}, f(x) = {value:.5f},
        time = {duration:.5f} sec")
```

```
Method comparison:
BFGS         -> Minimum at x = -1.30644, f(x) = -0.79458, time = 0.00100 sec
Nelder-Mead  -> Minimum at x = -1.30644, f(x) = -0.79458, time = 0.00196 sec
CG           -> Minimum at x = -1.30644, f(x) = -0.79458, time = 0.00098 sec
```

The resulting plot is shown in Figure F.5.

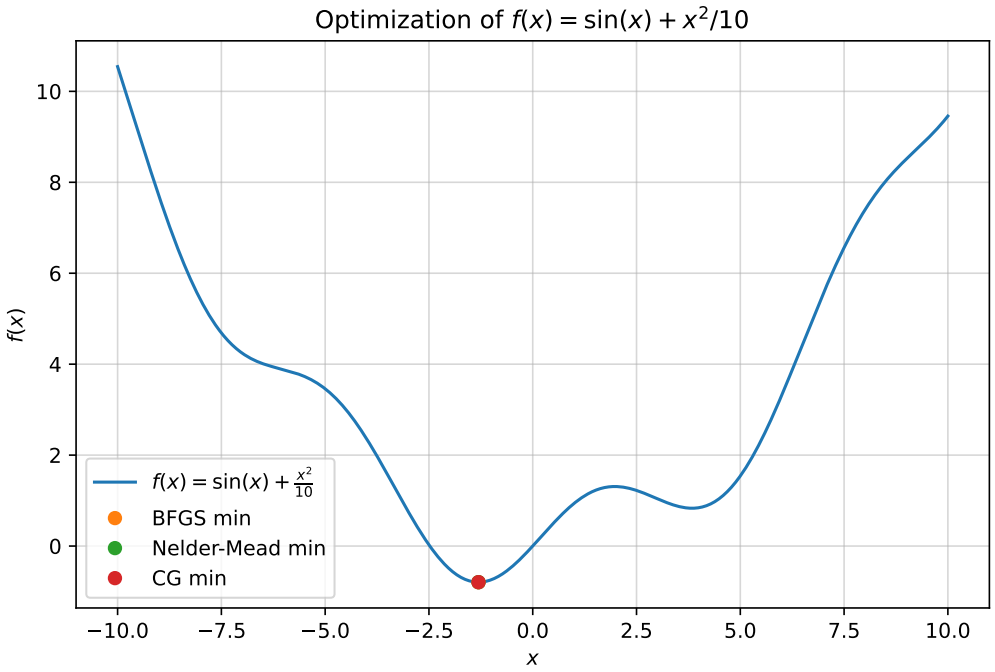


Figure F.5: Comparison of optimization methods applied to the nonconvex function $f(x) = \sin x + x^2/10$. The curve shows the objective function over the interval $[-10, 10]$, while the markers indicate the minima found by the BFGS, Nelder–Mead, and Conjugate Gradient (CG) methods starting from the same initial point. All methods converge to the same local minimum near $x \approx -1.31$.

F.57

```
import numpy as np
import matplotlib.pyplot as plt
```

```
from scipy.stats import ttest_1samp, chisquare, chi2
from numpy.linalg import inv

np.random.seed(42)

# Set parameters
mu = np.array([1, 2])
Sigma = np.array([[2, 0.5],
                  [0.5, 1]])
n_samples = 1000

# 1. Generate multivariate normal data
samples = np.random.multivariate_normal(mu, Sigma, size=n_samples)

# 2. Empirical mean and covariance
emp_mean = np.mean(samples, axis=0)
emp_cov = np.cov(samples, rowvar=False)

print("Empirical mean:", emp_mean)
print("Empirical covariance:\n", emp_cov)

# 3. t-tests for each coordinate
print("\nOne-sample t-tests (true means are [1, 2]):")
for i in range(2):
    t_stat, p_val = ttest_1samp(samples[:, i], popmean=mu[i])
    print(f"Dimension {i+1}: t = {t_stat:.4f}, p = {p_val:.4f}")

# 4. Compute squared Mahalanobis distances
Sigma_inv = inv(Sigma)
mahal_sq = np.array([
    (x - mu) @ Sigma_inv @ (x - mu).T
    for x in samples
])

# 5. Histogram of the distances
# Define histogram bins with overflow bin for values > 10
bins = np.array([0, 1, 2, 3, 4, 5, 6, 8, 10, np.inf])
counts, bin_edges = np.histogram(mahal_sq, bins=bins)

# Plot histogram and overlay expected chi-squared PDF
x_vals = np.linspace(0, 10, 300)
```

```
pdf_vals = chi2.pdf(x_vals, df=2)

plt.hist(mahal_sq, bins=bins[:-1], density=True, alpha=0.6,
    label="Empirical (binned)")
plt.plot(x_vals, pdf_vals, 'r-', label=r'$\chi^2_2$ PDF')
plt.xlabel("Squared Mahalanobis distance")
plt.ylabel("Density")
plt.title("Squared Mahalanobis Distances vs. $\chi^2_2$ Distribution")
plt.legend()
plt.grid(alpha=0.5)

# 6. Perform chi-squared goodness-of-fit test
# Expected frequencies under chi-squared(2) distribution
expected_probs = chi2.cdf(bin_edges[1:], df=2) -
    chi2.cdf(bin_edges[:-1], df=2)
expected_counts = n_samples * expected_probs

chi2_stat, chi2_pval = chisquare(counts, f_exp=expected_counts)
print("\nChi-squared goodness-of-fit test:")
print(f"Chi-squared statistic = {chi2_stat:.4f}, p = {chi2_pval:.4f}")
```

```
Empirical mean: [1.07713984 1.98091213]
Empirical covariance:
 [[1.92011568 0.49554576]
 [0.49554576 1.03737005]]

One-sample t-tests (true means are [1, 2]):
Dimension 1: t = 1.7604, p = 0.0786
Dimension 2: t = -0.5926, p = 0.5536

Chi-squared goodness-of-fit test:
Chi-squared statistic = 10.1516, p = 0.2545
```

The resulting plot is shown in Figure F.6.

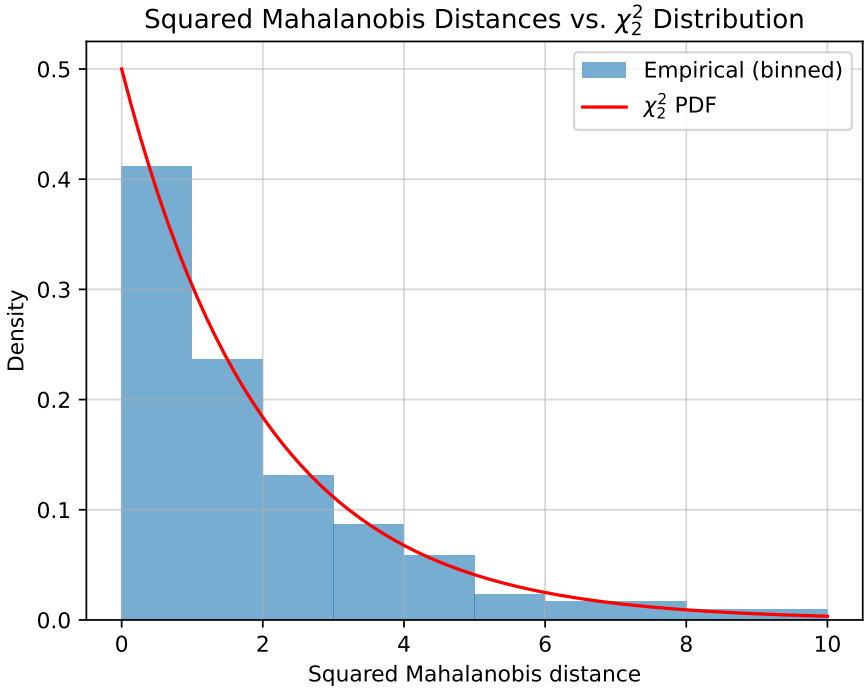


Figure F.6: Histogram of the squared Mahalanobis distances computed from a 2D Gaussian sample. The red curve shows the theoretical χ^2_2 probability density function, illustrating the agreement between the empirical distribution and the expected chi-squared distribution.