# Learning Machine Learning

tommyod

November 11, 2018

**Abstract**

This document contains some notes and solutions to the book "Pattern Classification" by Duda et al.

## Contents

# 1 Notes from "Pattern Classification"

## 1.2 Bayesian Decision Theory

- Bayes theorem is

$$P(\omega_j \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \omega_j) \times P(\omega_j)}{p(\boldsymbol{x})} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}.$$

The Bayes decision rule is to choose the state of nature $\omega_m$ such that

$$\omega_m = \arg\max_j P(\omega_j \mid \boldsymbol{x}).$$

- Loss functions (or risk functions) with losses other than zero-one are possible. In general, we choose the action $\lambda$ to minimize the risk $R(\lambda \mid \boldsymbol{x})$.
- The multivariate normal density (the Gaussian) is given by

$$p(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left[ -\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right].$$

It is often analytically tractable, and closed form discriminant functions exist.
- If features $\boldsymbol{y}$ are missing, we integrate them out (marginalize) using the sum rule

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{y}) \, d\boldsymbol{y} = \int p(\boldsymbol{x} \mid \boldsymbol{y}) p(\boldsymbol{y}) \, d\boldsymbol{y}.$$

- In Bayesian belief networks, influences are represented by a directed network. If $B$ is dependent on $A$, we add a directed edge $A \to B$ to the network.

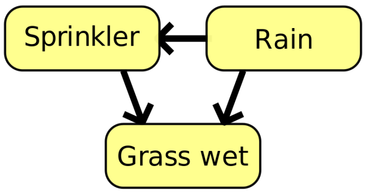

Figure 1: A Bayesian belief network. The source is Wikipedia.

## 1.3 Maximum-likelihood and Bayesian parameter estimation

- The *maximum likelihood* of a distribution $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ is given by $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D} \mid \boldsymbol{\theta})$, assuming i.i.d. data points and maximizing the log-likelihood, we have

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \ln p(\mathcal{D} \mid \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \ln \prod_{i=1}^{n} p(\boldsymbol{x}_i \mid \boldsymbol{\theta}).$$

Analytical solutions exist for the Gaussian, but in general a maximum likelihood estimate may be biased, in the sense that $\mathbb{E}_x[\hat{\boldsymbol{\theta}}] = \int \hat{\boldsymbol{\theta}} p(\boldsymbol{x}) \, d\boldsymbol{x} \neq \boldsymbol{\theta}$.

- In the Bayesian framework, the parameter $\boldsymbol{\theta}$ is expressed by a probability density function $p(\boldsymbol{\theta})$. This is called the *prior* distribution of $\boldsymbol{\theta}$, which is updated when new data is observed. The result is called the *posterior* distribution, given by

$$p(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} = \frac{p(\mathcal{D} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathcal{D} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})\, d\boldsymbol{\theta}}.$$

  The estimate of $\boldsymbol{x}$ then becomes

$$p(\boldsymbol{x} \mid \mathcal{D}) = \int p(\boldsymbol{x}, \boldsymbol{\theta} \mid \mathcal{D})\, d\boldsymbol{\theta} = \int p(\boldsymbol{x} \mid \boldsymbol{\theta}, \mathcal{D})p(\boldsymbol{\theta} \mid \mathcal{D})\, d\boldsymbol{\theta} = \int p(\boldsymbol{x} \mid \boldsymbol{\theta})p(\boldsymbol{\theta} \mid \mathcal{D})\, d\boldsymbol{\theta},$$

  which may be interpreted as a weighted average of models $p(\boldsymbol{x} \mid \boldsymbol{\theta})$, where $p(\boldsymbol{\theta} \mid \mathcal{D})$ is the weight associated with the model.
- The Bayesian framework is analytically tractable when using Gaussians. For instance, we can compute $p(\boldsymbol{\mu} \mid \mathcal{D})$ if we assume $p(\boldsymbol{\mu}) \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. The distribution $p(\boldsymbol{\mu})$ is called the *conjugate prior* and $p(\boldsymbol{\mu} \mid \mathcal{D})$ is a *reproducing density*, since the normal prior transforms to a normal posterior (with different parameters) when new data is observed.
- In summary the Bayesian framework allows us to incorporate prior information, but the maximum-likelihood approach is simpler. Maximum likelihood gives us an estimate $\hat{\boldsymbol{\theta}}$, but the Bayesian framework gives us $p(\boldsymbol{\theta} \mid \mathcal{D})$—the full distribution.
- Principal Component Analysis (PCA) yields components useful for *representation*. The covariance matrix is diagonalized, and low-variance directions in the hyperellipsoid are eliminated. The computation is performed using the SVD.
- Discriminant Analysis (DA) projects to a lower dimensional subspace with optimal *discrimination* (and not representation).
- Expectation Maximization (EM) is an iterative algorithm for finding the maximum-likelihood when data is missing (or latent).

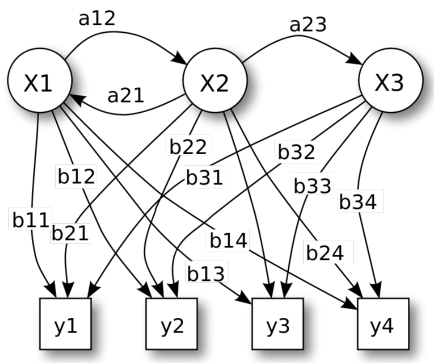

Figure 2: A hidden Markov model. The source is Wikipedia.

- A discrete, first order, hidden Markov model consists of a transition matrix $\boldsymbol{A}$ and an emission matrix $\boldsymbol{B}$. The probability of transition from state $i$ to state $j$ is given by $a_{ij}$, and the probability that state $i$ emits signal $j$ is given by $b_{ij}$. Three fundamental problems related to Markov models are:

  - The evaluation problem - probability that $\boldsymbol{V}^T$ was emitted, given $\boldsymbol{A}$ and $\boldsymbol{B}$.
  - The decoding problem - determine most likely sequence of hidden states $\boldsymbol{\omega}^T$, given emitted $\boldsymbol{V}^T$, $\boldsymbol{A}$ and $\boldsymbol{B}$.
  - The learning problem - determine $\boldsymbol{A}$ and $\boldsymbol{B}$ given training observations of $\boldsymbol{V}^T$ and a coarse model.

## 1.4 Nonparametric techniques

- Two conceptually different approaches to nonparametric pattern recognition are:
  - Estimation of densities $p(\boldsymbol{x} \mid w_j)$, called the *generative* approach.
  - Estimation of $P(w_j \mid \boldsymbol{x})$, called the *discriminative* approach.
- Parzen-windows (kernel density estimation) is a generative method. It places a *kernel function* $\phi : \mathbb{R}_+ \to \mathbb{R}_+$ on every data point $\boldsymbol{x}_i$ to create a density estimate
$$p_n(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \phi \left( \frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|_p}{h_n} \right),$$
where $\|\cdot\|_p : \mathbb{R}^d \to \mathbb{R}_+$ is the $p$-norm (which induces the so-called *Minkowski metric*) and $h_n \geq 0$ is the bandwidth.
- $k$-nearest neighbors is a discriminative method. It uses information about the $k$ nearest neighbors of a point $\boldsymbol{x}$ to compute $P(w_j \mid \boldsymbol{x})$. This automatically uses more of the surrounding space when data is sparse, and less of the surrounding space when data is dense. The $k$-nearest neighbor estimate is given by
$$P(w_j \mid \boldsymbol{x}) = \frac{\#\text{ samples labeled } w_j}{k}.$$
- The *nearest neighbor method* uses $k = 1$. It can be shown that the error rate $P$ of the nearest neighbor method is never more than twice the Bayes error rate $P^*$ in the limit of infinite data. More precisely, we have $P^* \leq P \leq P^*(2 - \frac{c}{c-1} P^*)$.
- In some applications, careful thought must be put into metrics. Examples include periodic data on $\mathbb{R}/\mathbb{Z}$ and image data where the metric should be invariant to small shifts and rotations. One method to alleviate the problems of using the 2-norm as a metric on images is to introduce the *tangent distance*. For an image $\boldsymbol{x}'$, the tangent vector of a transformation $\mathcal{F}$ (such as rotation by an angle $\alpha_i$) is given by
$$\boldsymbol{TV}_i = \mathcal{F}(\boldsymbol{x}'; \alpha_i) - \boldsymbol{x}'.$$
If several transformations are available, their linear combination may be computed. For each test point $\boldsymbol{x}$, we search the tangent space for the linear combination minimizing the metric. This gives a metric $D(\boldsymbol{x}, \boldsymbol{x}')$ which is more invariant to transformations such as small rotations and translations, compared to the 2-norm.
- *Reduced Coloumb energy networks* use ideas from both Parzen windows and $k$-nearest neighbors. It adjusts the size of the window so that it is less than some maximal radius, while not touching any observation of a different class. This creates "basins of attraction" for classification.

## 1.5 Linear discriminant functions

- Linear discriminant functions split the feature space in two with a hyperplane. The equation for a hyperplane is given by

$$g(\boldsymbol{x}) = \boldsymbol{\omega}^T \boldsymbol{x} + \omega_0 = \boldsymbol{a}^T \boldsymbol{y} = \begin{pmatrix} \omega_0 & \boldsymbol{\omega} \end{pmatrix} \begin{pmatrix} 1 \\ \boldsymbol{x} \end{pmatrix},$$

where $\omega_0$ is the bias. The form $\boldsymbol{a}^T\boldsymbol{y}$ is called the *augmented* form.

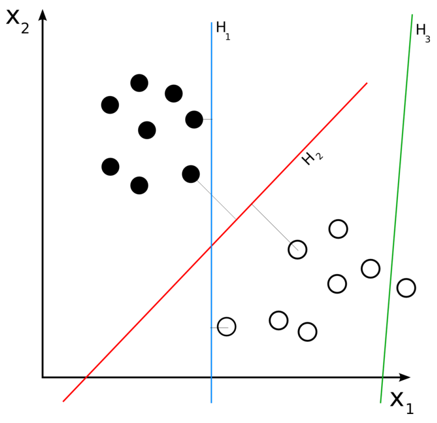

Figure 3: Linear discriminant functions. The source is Wikipedia.

- A linear machine assigns a point $\boldsymbol{x}$ to $\omega_i$ if

$$g_i(\boldsymbol{x}) \geq g_j(\boldsymbol{x})$$

for every other class $j$. This leaves no ambiguous regions in the feature space.
- By introducing mappings $\boldsymbol{y} = h(\boldsymbol{x})$ to higher- or lower-dimensional spaces, non-linearities in the original $\boldsymbol{x}$-space may be captured by linear classifiers working in $\boldsymbol{y}$-space. An example is $\boldsymbol{y} = h(\boldsymbol{x}) = \exp(-\boldsymbol{x}^T\boldsymbol{x})$ if data from one class is centered around the origin. Another example is transforming periodic data with period $P$ from $0 \leq x < P$ to $\boldsymbol{y}$ by use of the functions

$$y_1 = \cos(2\pi x/P) \qquad y_2 = \sin(2\pi x/P).$$

- Several algorithms may be used to minimize an error function $J(\boldsymbol{a})$. Two popular choices are *gradient descent* and *Newton descent*.
  - Gradient descent moves in the direction of the negative gradient. It is often controlled by a step length parameter $\eta(k)$, which may decrease as the iterations $k$ increase. The update rule is given by

$$\boldsymbol{a} \leftarrow \boldsymbol{a} - \eta(k)\nabla J(\boldsymbol{a}).$$

  - Newton descent also moves in the direction of the negative gradient, but the optimal step length is computed by linearizing the function $\nabla J(\boldsymbol{a})$ (or, equivalently, a second order approximation of $\boldsymbol{a}$). The update rule is given by

$$\boldsymbol{a} \leftarrow \boldsymbol{a} - \boldsymbol{H}^{-1}\nabla J(\boldsymbol{a}).$$

- Criterion functions for linearly separable data sets include:
  - The Perceptron function $\sum_{y \in \mathcal{Y}} (-\boldsymbol{a}^T \boldsymbol{y})$, which is not smooth.
  - The squared error with margin, given by $\sum_{y \in \mathcal{Y}} (\boldsymbol{a}^T \boldsymbol{y} - b)^2 / \|\boldsymbol{y}\|^a$.
- The *Mean Squared Error* (MSE) approach may be used, but it is not guaranteed to yield a separating hyperplane—even if one exists.
  - The MSE solution is found analytically by the pseudoinverse $\boldsymbol{A}^\dagger = \left(\boldsymbol{A}^T \boldsymbol{A}\right)^{-1} \boldsymbol{A}^T$. The pseudoinverse should never be used explicitly because it's numerically wasteful and unstable. It represents the analytical solution to the problem

    $$\min_x \boldsymbol{e}^T \boldsymbol{e} = \min_x (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})^T (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}),$$

    which is solved by $\boldsymbol{x} = \boldsymbol{A}^\dagger \boldsymbol{b}$.
  - The MSE approach is related to Fisher's linear discriminant for an appropriate choice of margin vector $\boldsymbol{b}$.
  - LMS may be computed using matrix procedures (never use the pseudoinverse directly) or by the gradient descent algorithm.
  - Ho-Kashyap procedures yield a separating hyperplane if one exists.
- *Linear programming* (LP) may also be used to find a separating hyperplane. Several reductions are possible by introduction of *artificial variables.*
  - Minimizing the Perceptron criterion function may be formulated as LP, and the result is decent even if a separating hyperplane does not exist.
- *Support Vector Machines* (SVM) find the minimum margin hyperplane. This is a quadratic programming (QP) problem, and the dual problem is easier to solve than the primal problem.

## 1.6 Multilayer Neural Networks

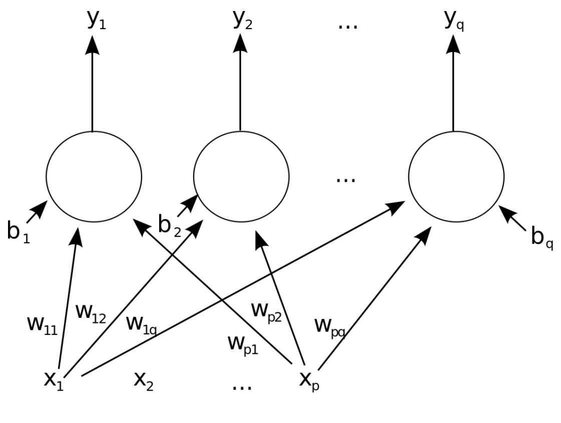

Figure 4: A three-layer neural network with bias. The source is Wikipedia.

- The feedforward operation on a $d - n_H - c$ three-layer neural network is defined by the following equation for the output

$$z_k = f\left(\sum_{j=1}^{n_H} w_{kj} f\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0}\right).$$

- The *Kolmogorov-Arnold representation theorem* implies that *any* continuous function from input to output may be expressed by a three layer $d - n_H - c$ neural network with sufficiently many hidden units.

- Backpropagation learns the weights $\boldsymbol{w}$ by the gradient descent equation $\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \eta \nabla J(\boldsymbol{w}_n)$. The gradient, or derivative, is found by repeated application of the chain rule of calculus. Several protocols are available for backpropagation: stochastic, batch and on-line.
  - Backpropagation may be thought of as feature mapping. While the inputs $x_i$ are not necessarily linearly separable, the outputs $y_j$ of the hidden units become linearly separable as the weights are learned. The final linear discriminant works on this data instead of the $x_i$.
- Practical tips for improving learning in neural networks include: standardizing features, adding noise and data augmentations, initializing weights to random values in the range $-1/\sqrt{d} < w_{ji} < 1/\sqrt{d}$, using momentum in the gradient descent algorithm, adding weight decay (equivalent to regularization) while learning, training with hints (output units which are subsequently removed) and experimenting with various error functions.
- Seconds order methods for learning the weights inluce
  - **Newtons method** – uses $\boldsymbol{H}$ in addition to $\nabla J(\boldsymbol{w})$.
  - **Quickprop** – two evaluations of $\nabla J(\boldsymbol{w})$ to approximate a quadratic.
  - **Conjugate gradient descent** – uses conjugate directions, which consists of a series of line searches. A given search direction does not spoil the result of the previous line searches. This is equivalent to a "smart momentum."
- Other networks include:
  - Convolutional Neural Networks (CNNs) – translation invariant, has achieved great success on image data.
  - Recurrent Neural Networks (RNNs) – the output of the previous prediction is fed into the subsequent prediction. This simulates "memory", and RNNs have been successful on time series data.
  - Cascade correlation – a technique where the topology is altered by adding more units until the performance is sufficiently good.

## 1.7 Stochastic methods

- Stochastic methods are used to search for optimal solutions when techniques such as gradient descent are not viable. For instance if the model is very complex, has a discrete nature where gradients are not suitable, or if there are time constraints.
- *Simulated annealing* is an optimization technique. To minimize the error $E(\boldsymbol{s})$, where $\boldsymbol{s} \in [-1, 1]^n$, we change a random entry of $\boldsymbol{s}$.
  - If the change produces a better result, then keep the new $\boldsymbol{s}$.
  - If the change does not produce a better result, we still might keep the change.

  The probability of keeping a change which increases the error $E(\boldsymbol{s})$ is a function of the temperature $T$, which typically decreases exponentially as the algorithm progresses. Initially simulated annealing is a *random search*, and as the temperature progresses it becomes a *greedy search*.

- *Deterministic simulated annealing* replaces discrete $s_i$ with analog (continuous) $s_i$. This forces the other magnets $s_k$ $(k \neq i)$ to determine $s_i$

$$s_i = f(T, \ell_i) = \tanh\left(\frac{\ell_i}{T}\right).$$

  As $T \to 0$, the $\tanh(\cdot)$ sigmoid function converges to a step function.
- *Boltzmann networks* (or Boltzmann machines) employ simulated annealing in a network to make predictions. First, weights $w_{ij}$ are learned so that inputs $s_j \in \alpha^i$ lead to correct outputs $s_k \in \alpha^o$ during classification. In the classification phase, the inputs $\alpha^i$ are *clamped* (fixed), and simulated annealing produces outputs $\alpha^o$. If the weights $w_{ij}$ are learned correctly, then the algorithm will produce good classifications.
  - Boltzmann networks are able to perform *pattern completion*.

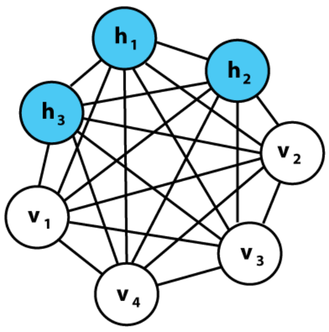

Figure 5: A non-restricted Boltzmann machine. The source is Wikipedia.

- *Evolutionary methods* take a population of classifiers through many generations. In each generation, new classifiers (offspring) are produced from the previous generation. The best classifiers are subject to (1) replication, (2) crossover and (3) mutation to produce offspring. The classifiers may be encoded as 2-bit chromosomes of length $L$. The bits represent some property of the classifier.
- Genetic programming is the process of modifying formulas such as

$$[(-x_1) + x_2] \,/\, [(\ln x_3) - x_2]$$

  by evolutionary methods, mutating variables and operators and performing crossovers.

## 1.8 Nonmetric methods

- A *decision tree* typically splits the feature space along the axes if the data is numeric, and works well for non-metric (categorical, or nominal) data as well. To implement a decision tree, one must consider
  - The number of splits made per node (typically 2, since a higher branching factor $B$ may be reduced to $B = 2$ anyway).

  - How to choose an attribute to split on – often solved using *information gain*.
  - When a node should be declared a leaf node.
  - How to handle missing data.
- To decide which attribute to split on, an *impurity function* is defined for a node $N$ consisting of training samples from various categories $\omega_1, \ldots, \omega_C$. The impurity function should be 0 when all samples in node $N$ are from the same category $\omega_j$, and peak when the samples are uniformly drawn from the $C$ categories.

  Two examples of impurity functions include the *entropy impurity* and the *Gini impurity*, which are respectively defined as

$$i(N) = -\sum_{j=1}^{C} P(\omega_j) \ln P(\omega_j) \qquad i(N) = \sum_{j=1}^{C} P(\omega_j) \sum_{k \neq j} P(\omega_k) = \sum_{j=1}^{C} P(\omega_j) \left[1 - P(\omega_j)\right].$$

  A split is chosen so that it maximizes the decrease in impurity, i.e.

$$\Delta i(N) = i(N) - \left[P_L \, i(N_L) + (1 - P_L) \, i(N_R)\right].$$

  The equation says that the change in impurity equals the original impurity at node $N$ minus the weighted average of the impurity of the left and right child node.
- Other considerations in decision trees include:
  - Pruning – simplifying the tree after training (bypassing the horizon effect).
  - Penalizing complexity – regularization of the tree structure.
  - Missing attributes – for instance using (1) surrogate splits or (2) sending a training sample down every path and then performing a weighted average.
- Four *string problems* in pattern classification are:
  - **Matching**: naive matching is slow, the *Boyer-Moore string matching algorithm* is much more efficient. It operates by increasing the shift $s$ of $\boldsymbol{x}$ using two heuristics in parallel: the *bad-character heuristic* and the *good-suffix* heuristic.
  - **Edit distance**: a way to compare the "distance" between strings by counting the number of insertions, deletions and substitutions required to transform $\boldsymbol{x}$ to $\boldsymbol{y}$. If all costs are equal, then $D(\boldsymbol{x}, \boldsymbol{y})$ is a metric. A dynamic programing algorithm is used to compute edit distance.
  - **Matching with errors** is the same as matching, but using for instance the edit distance to find approximate matches. The problem is to find a shift $s$ that minimizes the edit distance.
  - **Matching with the "don't care"-symbol $\emptyset$**: same as matching, but the $\emptyset$-symbol matches any character in the alphabet $\mathcal{A}$.
- A grammar $G = (\mathcal{A}, \mathcal{I}, \mathcal{S}, \mathcal{P})$ consists of symbols $\mathcal{A}$, variables $\mathcal{I}$, a root symbol $\mathcal{S}$ and productions $\mathcal{P}$. Concrete examples include English sentences and pronunciation of numbers. There are several types of grammars, and they constitute a hierarchy

$$\text{Type}\,3 \subset \text{Type}\,2 \subset \text{Type}\,1 \subset \text{Type}\,0.$$

  The types are respectively called regular, context free, context sensitive and free.
  - A central question is whether a string $\boldsymbol{x}$ is in the language $\mathcal{L}$ generated by the grammar $G$, i.e. whether $\boldsymbol{x} \in \mathcal{L}(G)$. This can be answered using bottom-up parsing, which employs the product rules $\mathcal{P}$ backwards.

## 1.9 Algorithm-independent machine learning

- asdsdf

# 2 Solutions to “Pattern Classification”

This section contains solutions to problems in “Pattern Classification” by Duda et al. There are approximately 8 solved problems from each chapter.

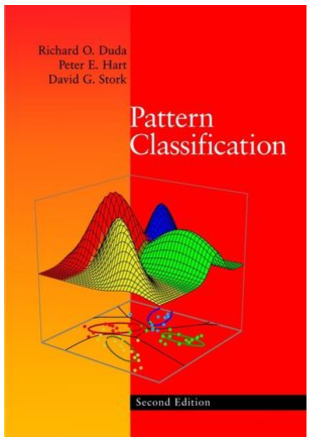

Figure 6: The front cover of [Duda et al., 2000].

## 2.2 Bayesian Decision Theory

### Problem 2.6

a) We want the probability of choosing action $\alpha_2$ to be smaller than, or equal to, $E_1$, given that the true state of nature is $\omega_1$. Let's assume that $\mu_1 < \mu_2$ and that the decision threshold is $x^*$, so we decide $\alpha_2$ if $x > x^*$. We then have

$$\begin{aligned} P(\alpha_2 \mid \omega_1) &\leq E_1 \\ p(x > x^* \mid \omega_1) &\leq E_1 \\ \left[1 - \int_0^{x^*} p(x \mid \omega_1)\, dx\right] &\leq E_1 \end{aligned}$$

We let $\Phi : \mathbb{R} \to [0,1]$ denote the cumulative Gaussian distribution, and $\Phi^{-1} : [0,1] \to \mathbb{R}$ it's inverse function. Making use of $\Phi$ we write

$$\begin{aligned} 1 - \Phi\left(\frac{x^* - \mu_1}{\sigma_1}\right) &\leq E_1 \\ x^* &\geq \mu_1 + \sigma_1 \Phi^{-1}(1 - E_1). \end{aligned}$$

If the desired error is close to zero, then $x^*$ goes to positive infinity. If the desired error is close to one, then $x^*$ goes to negative infinity.

b) The error rate for classifying $\omega_2$ as $\omega_1$ is

$$P(\alpha_1 \mid \omega_2) = p(x \leq x^* \mid \omega_2) = \int_0^{x^*} p(x \mid \omega_2)\, dx = \Phi\left(\frac{x^* - \mu_2}{\sigma_2}\right).$$

Making use of $x^*$ from the previous problem, we obtain

$$\Phi\left(\frac{\mu_1 + \sigma_1 \Phi^{-1}(1 - E_1) - \mu_2}{\sigma_2}\right) = \Phi\left(\frac{\mu_1 - \mu_2}{\sigma_2} + \frac{\sigma_1}{\sigma_2}\Phi^{-1}(1 - E_1)\right).$$

c) The overall error rate becomes

$$\begin{aligned} P(\text{error}) &= P(\alpha_1, \omega_2) + P(\alpha_2, \omega_1) \\ &= P(\alpha_1 \mid \omega_2)P(\omega_2) + P(\alpha_2 \mid \omega_1)P(\omega_1) \\ &= \frac{1}{2}\left[P(\alpha_1 \mid \omega_2) + P(\alpha_2 \mid \omega_1)\right] \\ &= \frac{1}{2}\left[E_1 + \Phi\left(\frac{\mu_1 - \mu_2}{\sigma_2} + \frac{\sigma_1}{\sigma_2}\Phi^{-1}(1 - E_1)\right)\right]. \end{aligned}$$

In the last equality we used the results from the previous problems.

d) We substitute the given values into the equations, and obtain $x^* \approx 0.6449$. The total error rate is $P(\text{error}) \approx 0.2056$.

e) The Bayes error rate, as a function of $x^*$, is given by

$$\begin{aligned} P(\text{error}) &= P(\alpha_2 \mid \omega_1)P(\omega_1) + P(\alpha_1 \mid \omega_2)P(\omega_2) \\ &= \frac{1}{2}\left[p(x > x^* \mid \omega_1) + p(x < x^* \mid \omega_2)\right] \\ &= \frac{1}{2}\left[\left(1 - \Phi\left(\frac{x^* - \mu_1}{\sigma_1}\right)\right) + \Phi\left(\frac{x^* - \mu_2}{\sigma_2}\right)\right] \end{aligned}$$

The Bayes error rate is depicted in Figure 7.

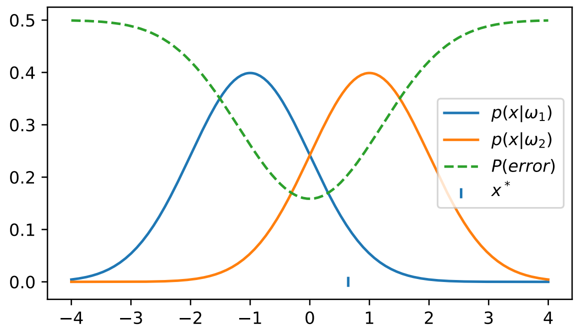

Figure 7: Graf accompanying problem 2.6.

## Problem 2.12

a) The key observation is that the maximal value $P(\omega_{\max} \mid \boldsymbol{x})$ is greater than, or equal to, the average. Therefore we obtain

$$P(\omega_{\max} \mid \boldsymbol{x}) \geq \frac{1}{c}\sum_{i=1}^{c} P(\omega_i \mid \boldsymbol{x}) = \frac{1}{c},$$

where the last equality is due to probabilities summing to unity.

b) The minimum error rate is achieved by choosing $\omega_{\max}$, the most likely state of nature. The average probability of error over the data space is therefore the probability that $\omega_{\max}$ is *not* the true state of nature for a given $\boldsymbol{x}$, that is:

$$P(\text{error}) = \mathbb{E}_x \left[1 - P(\omega_{\max} \mid \boldsymbol{x})\right] = 1 - \int P(\omega_{\max} \mid \boldsymbol{x}) p(\boldsymbol{x})\, d\boldsymbol{x}.$$

c) We see that

$$P(\text{error}) = 1 - \int P(\omega_{\max} \mid \boldsymbol{x}) p(\boldsymbol{x})\, d\boldsymbol{x} \leq 1 - \int \frac{1}{c} p(\boldsymbol{x})\, d\boldsymbol{x} = 1 - \frac{1}{c} = \frac{c-1}{c},$$

where we used $\int p(\boldsymbol{x})\, d\boldsymbol{x} = 1$.

d) A situation where $P(\text{error}) = (c-1)/c$ arises when $P(\omega_i) = 1/c$ for every $i$. Then the maximum value is equal to the average value, and the inequality in problem a) becomes an equality.

**Problem 2.19**

a) The entropy is given by $\mathrm{H}\left[p(x)\right] = -\int p(x) \ln p(x)\, dx$. The optimization problem gives the synthetic function

$$H_s = -\int p(x) \ln p(x)\, dx + \sum_{k=1}^{q} \lambda_k \left( \int b_k(x) p(x)\, dx - a_k \right),$$

and since a probability density function has $\int p(x)\, dx = 1$ we add an additional constraint for $k = 0$ with $b_0(x) = 1$ and $a_k = 1$. Collecting terms we obtain

$$\begin{aligned} H_s &= -\int p(x) \ln p(x)\, dx + \sum_{k=0}^{q} \lambda_k \int b_k(x) p(x)\, dx - \sum_{k=0}^{q} \lambda_k a_k \\ &= -\int p(x) \left[ \ln p(x) - \sum_{k=0}^{q} \lambda_k b_k(x) \right] dx - \sum_{k=0}^{q} \lambda_k a_k, \end{aligned}$$

which is what we were asked to show.

b) Differentiating the equation above with respect to $p(x)$ and equating it to zero we obtain

$$-\int \left( 1 \left[ \ln p(x) - \sum_{k=0}^{q} \lambda_k b_k(x) \right] + p(x) \left[ \frac{1}{p(x)} \right] \right) dx = 0.$$

This integral is zero if the integrand is zero for every $x$, so we require that

$$\ln p(x) - \sum_{k=0}^{q} \lambda_k b_k(x) + 1 = 0,$$

and solving this equation for $p(x)$ gives the desired answer.

**Problem 2.21**

We are asked to compute the entropy of the (1) Gaussian distribution, (2) triangle distribution and (3) uniform distribution. Every p.d.f has $\mu = 0$ and standard deviation $\sigma$, and we must write every p.d.f parameterized using $\sigma$.

**Gaussian** We use the definition $\mathrm{H}\left[p(x)\right] = -\int p(x) \ln p(x)\, dx$ to compute

$$\mathrm{H}\left[p(x)\right] = -\int \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\frac{x^2}{\sigma^2}\right) \left[\ln\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \frac{1}{2}\frac{x^2}{\sigma^2}\right] dx.$$

Let us denote $K = \frac{1}{\sqrt{2\pi}\sigma}$ to simplify notation. We obtain

$$-\int K \exp\left(-\frac{1}{2}\frac{x^2}{\sigma^2}\right)\left[\ln K - \frac{1}{2}\frac{x^2}{\sigma^2}\right] dx =$$
$$-K\ln K \int \exp\left(-\frac{1}{2}\frac{x^2}{\sigma^2}\right) dx + K\int \frac{1}{2}\frac{x^2}{\sigma^2}\exp\left(-\frac{1}{2}\frac{x^2}{\sigma^2}\right) dx$$

The first term is simply negative $\ln K$, since it's the normal distribution with an additional factor $-\ln K$. The second term is not as straightforward. We change variables to $y = x/\left(\sqrt{2}\sigma\right)$, and write it as

$$K\int y^2 \exp\left(-y^2\right)\sqrt{2}\sigma\, dy,$$

which can be solved by using the following observation (from integration by parts)

$$\int 1e^{-x^2}\, dx = \underbrace{xe^{-x^2}\Big|}_{0 \text{ at } \pm\infty} - \int x(-2x)e^{-x^2}\, dx.$$

Using the above equation in reverse, we integrate as follows:

$$K\sqrt{2}\sigma\int y^2\exp\left(-y^2\right) dy = K\sqrt{2}\sigma\frac{1}{2}\int \exp\left(-y^2\right) dy = K\sqrt{2}\sigma\frac{1}{2}\sqrt{\pi} = \frac{1}{2}$$

To recap, the first integral evaluated to $-\ln K$, and the second evaluated to $\frac{1}{2}$. The entropy of the Gaussian is therefore $1/2 + \ln\sqrt{2\pi}\sigma$.

**Triangle** The triangle distribution may be written in the form

$$f(x) = \begin{cases} h - \frac{hx}{b} & \text{if } |x| < b \\ 0 & \text{if } |x| \geq b, \end{cases}$$

where $h$ is the height and $b$ is the width to the left of, and to the right of, $x = 0$.

Since the integral must evaluate to unity, we impose $hb = 1$ and obtain $f(x; b) = \frac{1}{b}\left(1 - \frac{x}{b}\right)$. We wish to parameterize the triangle distribution using the standard deviation $\sigma$ instead of width $b$. We can use $\mathrm{var}(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$ to find the variance, since in this case

$\mathbb{E}(X)^2 = \mu^2 = 0$ since the function is centered on $x = 0$. Computing $\mathbb{E}(X^2)$ yields $b^2/6$, so $b^2 = 6\sigma^2$. The revised triangle distribution then becomes

$$f(x;\sigma) = \begin{cases} \frac{1}{\sqrt{6}\sigma}\left(1 - \frac{x}{\sqrt{6}\sigma}\right) & \text{if } |x| < \sqrt{6}\sigma \\ 0 & \text{if } |x| \geq \sqrt{6}\sigma. \end{cases}$$

We set $k = \frac{1}{\sqrt{6}\sigma}$ to ease notation. Due to symmetry, we compute the entropy as

$$\mathrm{H}\left[f(x;\sigma)\right] = -2\int_0^{\sqrt{6}\sigma} k\left(1 - kx\right)\ln\left(k\left(1 - kx\right)\right)\, dx.$$

Changing variables to $y = 1 - kx$ we obtain

$$\begin{aligned} &-2\int_{x=0}^{x=\sqrt{6}\sigma} ky\left(\ln k + \ln y\right)\, dx = -2\int_{y=1}^{y=0} ky\left(\ln k + \ln y\right)\left(\frac{1}{-k}\right)\, dy \\ &-2\int_0^1 y\left(\ln k + \ln y\right) dy = -2\int_0^1 y\ln k\, dy - 2\int_0^1 y\ln y\, dy = -2\left(\ln k - \frac{1}{4}\right), \end{aligned}$$

where the last integral can be evaluated using integration by parts. The entropy of the triangle distribution turns out to be $1/2 + \ln\sqrt{6}\sigma$.

**Uniform** Using the same logic as with the triangle distribution to normalize a uniform distribution, and then parameterizing by $\sigma$, we obtain

$$u(x;\sigma) = \left\{\begin{array}{ll} \frac{1}{2b} & \text{if } |x| < b \\ 0 & \text{if } |x| \geq b \end{array}\right\} = \begin{cases} \frac{1}{2\sqrt{3}\sigma} & \text{if } |x| < \sqrt{3}\sigma \\ 0 & \text{if } |x| \geq \sqrt{3}\sigma. \end{cases}$$

Computing the entropy is easier than in the case of the Gaussian and the triangle distribution, we simply evaluate the integral as

$$\mathrm{H}\left[p(x)\right] = 2\int_0^{\sqrt{3}\sigma} \frac{1}{2\sqrt{3}\sigma}\ln\frac{1}{2\sqrt{3}\sigma}\, dx = \ln 2\sqrt{3}\sigma.$$

.

Let's briefly compare the results of our computations as follows:

$$\begin{aligned} H_{\text{Gaussian}}(\sigma) &= 1/2 + \ln\sqrt{2\pi}\sigma = \frac{1}{2} + \ln\sqrt{2\pi} + \ln\sigma \approx 1.4189 + \ln\sigma \\ H_{\text{Triangle}}(\sigma) &= 1/2 + \ln\sqrt{6}\sigma = \frac{1}{2} + \ln\sqrt{6} + \ln\sigma \approx 1.3959 + \ln\sigma \\ H_{\text{Uniform}}(\sigma) &= \ln 2\sqrt{3}\sigma = 0 + \ln 2\sqrt{3} + \ln\sigma \approx 1.2425 + \ln\sigma \end{aligned}$$

This verifies that out of the three distributions, the Gaussian has the maximal entropy. This was expected, since the Gaussian maximizes the entropy over *any* continuous p.d.f. having a prescribed mean and variance.

**Problem 2.23**

a) To solve this problem, we need to find the inverse matrix, the determinant, and $\boldsymbol{w} = \boldsymbol{x} - \boldsymbol{\mu}$.

$$\boldsymbol{\Sigma}^{-1} = \frac{1}{21}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & -2 \\ 0 & -2 & 5 \end{pmatrix} \quad \det \boldsymbol{\Sigma} = 21 \quad \boldsymbol{w} = \boldsymbol{x} - \boldsymbol{\mu} = \begin{pmatrix} -0.5 \\ -2 \\ -1 \end{pmatrix}$$

The number of dimension $d$ is 3. The solution is

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{3}{2}} 21^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\boldsymbol{w}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{w}\right) = \frac{1}{(2\pi)^{\frac{3}{2}} 21^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\frac{1}{21}\frac{69}{4}\right).$$

b) The eigenvalues of $\boldsymbol{\Sigma}$ are $\lambda_1 = 3$, $\lambda_1 = 7$ and $\lambda_1 = 21$. The corresponding eigenvectors are $\boldsymbol{v}_1 = (0, 1, -1)^T/\sqrt{2}$, $\boldsymbol{v}_2 = (0, 1, 1)^T/\sqrt{2}$ and $\boldsymbol{v}_3 = (1, 0, 0)^T$. The whitening transformation is therefore given by

$$\boldsymbol{A}_w = \boldsymbol{\Phi}\boldsymbol{\Lambda}^{-1/2} = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 & 0 & \sqrt{2} \\ 1 & 1 & 0 \\ -1 & 1 & 0 \end{pmatrix}\begin{pmatrix} -\sqrt{3} & 0 & 0 \\ 0 & -\sqrt{7} & 0 \\ 0 & 0 & -\sqrt{21} \end{pmatrix}.$$

The rest of the numerical computations are skipped.

c) Skipped.

d) Skipped.

e) We are going to examine if the p.d.f is unchanged when vectors are transformed with $\boldsymbol{T}^T\boldsymbol{x}$ and matrices with $\boldsymbol{T}^T\boldsymbol{\Sigma}\boldsymbol{T}$. Let's consider the term $(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})$ in the exponent first. Substituting $\boldsymbol{x} \mapsto \boldsymbol{T}^T\boldsymbol{x}$, $\boldsymbol{\mu} \mapsto \boldsymbol{T}^T\boldsymbol{\mu}$ and $\boldsymbol{\Sigma} \mapsto \boldsymbol{T}^T\boldsymbol{\Sigma}\boldsymbol{T}$, we observe that

$$\begin{gathered}\left(\boldsymbol{T}^T\boldsymbol{x} - \boldsymbol{T}^T\boldsymbol{\mu}\right)^T \left(\boldsymbol{T}^T\boldsymbol{\Sigma}\boldsymbol{T}\right)^{-1} \left(\boldsymbol{T}^T\boldsymbol{x} - \boldsymbol{T}^T\boldsymbol{\mu}\right) \\ \left(\boldsymbol{T}^T(\boldsymbol{x} - \boldsymbol{\mu})\right)^T \left(\boldsymbol{T}^T\boldsymbol{\Sigma}\boldsymbol{T}\right)^{-1} \boldsymbol{T}^T(\boldsymbol{x} - \boldsymbol{\mu}) \\ (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{T} \left(\boldsymbol{T}^T\boldsymbol{\Sigma}\boldsymbol{T}\right)^{-1} \boldsymbol{T}^T(\boldsymbol{x} - \boldsymbol{\mu}) \\ (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{T}\boldsymbol{T}^{-1}\boldsymbol{\Sigma}^{-1}\boldsymbol{T}^{-T}\boldsymbol{T}^T(\boldsymbol{x} - \boldsymbol{\mu}) \\ (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}),\end{gathered}$$

where we have used $(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T\boldsymbol{A}^T$ and $(\boldsymbol{A}\boldsymbol{B})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$, which are basic facts from linear algebra. The density remains proportional when applying a linear transformation, but not unscaled, since the proportionality term $|\boldsymbol{\Sigma}|^{1/2}$ becomes $\left|\boldsymbol{T}^T\boldsymbol{\Sigma}\boldsymbol{T}\right|^{1/2} = \left|\boldsymbol{T}^T\right|^{1/2} |\boldsymbol{\Sigma}|^{1/2} |\boldsymbol{T}|^{1/2} = |\boldsymbol{T}|\, |\boldsymbol{\Sigma}|^{1/2}$.

f) Here we use the eigendecomposition of a symmetric matrix. We assume that $\boldsymbol{\Sigma}$ is positive definite such that every eigenvalue is positive. We write $\boldsymbol{\Sigma} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^T$ and apply the whitening transformation.

$$\boldsymbol{A}_{\boldsymbol{w}}^T\boldsymbol{\Sigma}\boldsymbol{A}_{\boldsymbol{w}} = \boldsymbol{A}_{\boldsymbol{w}}^T\boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^T\boldsymbol{A}_w = \left(\boldsymbol{\Phi}\boldsymbol{\Lambda}^{-1/2}\right)^T \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^T \left(\boldsymbol{\Phi}\boldsymbol{\Lambda}^{-1/2}\right)$$

The matrix $\mathbf{\Phi}$ is orthogonal, so it's transpose is the inverse. Using this fact and proceeding, we obtain

$$\left(\mathbf{\Phi}\mathbf{\Lambda}^{-1/2}\right)^T \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T \left(\mathbf{\Phi}\mathbf{\Lambda}^{1/2}\right) = \left(\mathbf{\Lambda}^{-1/2}\right)^T \mathbf{\Lambda}\mathbf{\Lambda}^{-1/2} = \mathbf{\Lambda}^{-1/2}\mathbf{\Lambda}\mathbf{\Lambda}^{-1/2} = \boldsymbol{I},$$

so the covariance is proportional to the identity matrix, as we were tasked to show. The normalization constant becomes 1 , since the proportionality term becomes $|\boldsymbol{T}|\,|\mathbf{\Sigma}|^{1/2}$ under the transformation, and

$$|\boldsymbol{T}|\,|\mathbf{\Sigma}|^{1/2} = \left|\mathbf{\Phi}\mathbf{\Lambda}^{-1/2}\right|\,|\mathbf{\Sigma}|^{1/2} = \left|\mathbf{\Phi}\mathbf{\Lambda}^{-1/2}\right|\,\left|\mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T\right|^{1/2} = |\boldsymbol{I}| = 1.$$

## Problem 2.28

a) We prove that if $p(x_i - \mu_i, x_j - \mu_j) = p(x_i - \mu_i)p(x_j - \mu_j)$, then $\sigma_{ij} = \mathbb{E}\left[(x_i - \mu_i)(x_j - \mu_j)\right] = 0$. With words: we prove that statistical independence implies zero covariance.

$$\begin{gathered}
\mathbb{E}\left[(x_i - \mu_i)(x_j - \mu_j)\right] = \\
\iint p(x_i - \mu_i, x_j - \mu_j)(x_i - \mu_i)(x_j - \mu_j)\, dx_j dx_i = \\
\iint p(x_i - \mu_i)p(x_j - \mu_j)(x_i - \mu_i)(x_j - \mu_j)\, dx_j dx_i \\
\int p(x_i - \mu_i)(x_i - \mu_i)\left(\int p(x_j - \mu_j)(x_j - \mu_j)\, dx_j\right) dx_i
\end{gathered}$$

If the term in the parenthesis is identically zero, then $\sigma_{ij} = 0$. This is indeed true, since we find that

$$\int p(x_j - \mu_j)(x_j - \mu_j)\, dx_j = \mathbb{E}\left[(x_j - \mu_j)\right] = \mathbb{E}\left[x_j\right] - \mathbb{E}\left[\mu_j\right] = \mu_j - \mu_j = 0.$$

b) We wish to prove the converse of a) in the Gaussian case. To achieve this, we must show that $\sigma_{ij} = 0$ when $p(x_i - \mu_i, x_j - \mu_j) = p(x_i - \mu_i)p(x_j - \mu_j)$. Let's simplify the notation to $x$ and $y$ instead of $x_i$ and $x_j$. If $\sigma_{xy} = 0$, then the covariance matrix is a diagonal matrix $\boldsymbol{D} = \operatorname{diag}(\sigma_x^2, \sigma_y^2)$. We write the probability $p(x_i - \mu_i, x_j - \mu_j)$ as $p(x, y)$, where the means $\mu_x$ and $\mu_y$ are both zero. We write

$$\begin{aligned}
p(x, y) &= \frac{1}{(2\pi)^{2/2}\sigma_x\sigma_y}\exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{D}^{-1}\boldsymbol{x}\right) = \frac{1}{(2\pi)^{2/2}\sigma_x\sigma_y}\exp\left(-\frac{1}{2}\left(x^2/\sigma_x^2 + y^2/\sigma_y^2\right)\right) \\
&= \frac{1}{(2\pi)^{1/2}\sigma_x}\exp\left(-\frac{1}{2}\left(x^2/\sigma_x^2\right)\right)\frac{1}{(2\pi)^{1/2}\sigma_y}\exp\left(-\frac{1}{2}\left(y^2/\sigma_y^2\right)\right) = p(x)p(y).
\end{aligned}$$

This proves that when $\sigma_{xy} = 0$, the covariance matrix is diagonal, and the Gaussian factors into products and we have statistical independence.

c) This problem asks us to find a counterexample of the above, i.e. an example showing that $\sigma_{xy} \not\Rightarrow p(x, y) = p(x)p(y)$. The probability density function

$$p(x, y) = K\frac{1}{1 + x^2 + y^2}, \quad K^{-1} = \iint_{\mathbb{R}} \frac{1}{1 + x^2 + y^2}\, dxdy$$

achieves this. The covariance is zero, since $\sigma_{xy} = \mathbb{E}\left[(x-0)(y-0)\right] = \iint_{\mathbb{R}} \frac{xy}{1+x^2+y^2}\, dxdy = \iint_{\mathbb{R}} I(x,y)\, dxdy$ is zero because the integrand $I(x,y)$ is an odd function.

On the other hand, $p(x,y)$ does *not* factor into $p(x)p(y)$. We have proved that $\sigma_{xy} \not\Rightarrow p(x,y) = p(x)p(y)$ by finding a counterexample.

## Problem 2.31

a) We'll assume that $\mu_1 < \mu_2$. Since $\sigma_1 = \sigma_2 = \sigma$, the minimum probability of error is achieved by setting the decision threshold to $x^* = (\mu_1 + \mu_2)/2$. To follow the derivation below, it helps to draw the real line and two Gaussians. The probability of error is then

$$\begin{aligned}
P_e &= P(x \in R_2, \omega_1) + P(x \in R_1, \omega_2) \\
&= P(x \in R_2 \mid \omega_1)P(\omega_1) + P(x \in R_1 \mid \omega_2)P(\omega_2) \\
&= \int_{R_2} p(x \mid \omega_1)P(\omega_1)\, dx + \int_{R_1} p(x \mid \omega_2)P(\omega_2)\, dx \\
&= \frac{1}{2}\left(\int_{x^*}^{\infty} p(x \mid \omega_1)\, dx + \int_0^{x^*} p(x \mid \omega_2)\, dx\right) = \int_{x=(\mu_1+\mu_2)/2}^{\infty} p(x \mid \omega_1)\, dx \\
&= \int_{x=(\mu_1+\mu_2)/2}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\frac{(x-\mu_1)^2}{\sigma^2}\right) dx.
\end{aligned}$$

Changing variables to $u = (x-\mu_1)/\sigma$ and using $dx = \sigma\, du$ yields

$$P_e = \int_{u=a}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-u^2/\sigma^2\right)\, du,$$

where $a = (x-\mu_1)/\sigma = ((\mu_1+\mu_2)/2 - \mu_1)/\sigma = (\mu_2 - \mu_1)/2\sigma$, as required.

b) Using the inequality stated in the problem, it remains to show that

$$\lim_{a\to\infty} f(a) = \lim_{a\to\infty} \frac{1}{\sqrt{2\pi}a} \exp\left(-a^2/\sigma^2\right) = 0.$$

This holds if the derivative is negative as $a \to \infty$, since then the function decreases as $a \to \infty$. The derivative of $f(a)$ is

$$f'(x) = -\exp\left(-a^2/2\right)\left(1 - \frac{1}{a^2}\right),$$

which is negative as long as $|a| \geq 1$. Alternatively, we see that both factors in $f(a)$ go to zero as $a \to \infty$.

## Problem 2.43

a) $p_{ij}$ is the probability that the $i$'th entry in the vector $\boldsymbol{x}$ equals 1, given a state of nature $\omega_j$.

b) We decide $\omega_j$ if $P(\omega_j \mid \boldsymbol{x})$ is greater than $P(\omega_k \mid \boldsymbol{x})$ for every $k \neq j$.

$$P(\omega_j \mid \boldsymbol{x}) \propto p(\boldsymbol{x} \mid \omega_j) P(\omega_j)$$

We use the fact that $p(\boldsymbol{x} \mid \omega_j) = \prod_{i=1}^{d} p(x_i \mid \omega_j)$, which follows from the fact that the entries are statistically independent. Furthermore, we see that

$$p(x_i \mid \omega_j) = \begin{Bmatrix} p_{ij} & \text{if } x_i = 1 \\ 1 - p_{ij} & \text{if } x_i = 0 \end{Bmatrix} = p_{ij}^{x_i} \left(1 - p_{ij}\right)^{1-x_i}.$$

Now we take logarithms and obtain

$$\begin{aligned} &\ln\left(\prod_{i=1}^{d} p(x_i \mid \omega_j) P(\omega_j)\right) = \sum_{i=1}^{d} \ln p(x_i \mid \omega_j) + \ln P(\omega_j) \\ &= \sum_{i=1}^{d} \ln p_{ij}^{x_i} \left(1 - p_{ij}\right)^{1-x_i} + \ln P(\omega_j) \\ &= \sum_{i=1}^{d} x_i \ln p_{ij} + (1 - x_i) \ln(1 - p_{ij}) + \ln P(\omega_j), \end{aligned}$$

which is easily arranged to correspond with the expression in the problem statement. In summary we choose the class $\omega_j$ if the probability of that class given the data point exceeds the probability of every other class.

## 2.3 Maximum-likelihood and Bayesian parameter estimation

### Problem 3.2

a) The maximum likelihood estimate for $\theta$ is $\max_\theta p(x \mid \theta) = \max_\theta \prod_{i=1}^n p(x_i \mid \theta)$. The probability of a single sample $p(x_i \mid \theta)$ is given by the expression

$$p(x_i \mid \theta) = \begin{cases} 1/\theta & \text{if } 0 \leq x_i \leq \theta \\ 0 & \text{if } x_i > \theta. \end{cases}$$

Clearly the product $\prod_{i=1}^n p(x_i \mid \theta)$ is zero if any $x_i$ is larger than $\theta$. Therefore $\theta$ must be larger than, or equal to, $\max_k x_k$ for the likelihood to be non-zero.

On the other hand, the product equals $1/\theta^n$, and taking logarithms we obtain $-n \ln \theta$. This function is maximized when $\theta$ is as small as possible.

The conclusion is that $\theta$ must be $\geq \max_k x_k$ to avoid the likelihood being zero, and also as small as possible to maximize the likelihood. Therefore the maximum likelihood is given by $\hat{\theta} = \max_k x_k = \max \mathcal{D}$.

b) Skipping this plot. The explanation of why the other points are not needed is given in part a) of the problem.

### Problem 3.4

The maximum likelihood estimate is

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = \prod_{k=1}^n p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \prod_{k=1}^n \prod_{i=1}^d \theta_i^{x_{ik}} (1-\theta_i)^{(1-x_{ik})}.$$

The log likelihood $\ell(\boldsymbol{\theta})$ is $\ln p(\mathcal{D} \mid \boldsymbol{\theta})$, which is explicitly given by

$$\ell(\boldsymbol{\theta}) = \sum_{k=1}^n \sum_{i=1}^d x_{ik} \ln \theta_i + (1 - x_{ik}) \ln (1 - \theta_i).$$

Differentiating $\ell(\boldsymbol{\theta})$ with respect to $\theta_i$, every term in the sum $\sum_{i=1}^d$ vanishes except the $i$'th. We perform the differentiation and equate the result to zero, yielding

$$\frac{d\ell(\boldsymbol{\theta})}{\theta_i} = \sum_{k=1}^n \left[ \frac{x_{ik}}{\theta_i} + \frac{x_{ik} - 1}{1 - \theta_i} \right] = \sum_{k=1}^n [x_{ik} - \theta_i] = 0.$$

Solving this for $\theta_i$ yields $\theta_i = n^{-1} \sum_{k=1}^n x_{ik}$, or in vector notation, $\boldsymbol{\theta} = n^{-1} \sum_{k=1}^n \boldsymbol{x}_k$. This is what the problem asked us to show.

**Problem 3.13**

a) Familiarity with summation notation helps when solving this problem. The matrix-vector product $\boldsymbol{A}\boldsymbol{a}$ may be written as $\sum_j A_{ij}a_j$. The sum is typically taken over repeated indices, but we will explicitly typeset the summation index.

Let's write the outer product as $\boldsymbol{a}\boldsymbol{b}^T = a_i \oplus b_j$, the trace as $\operatorname{tr}(\boldsymbol{A}) = \sum_i A_{ii}$ and

$$\operatorname{tr}(\boldsymbol{a}\boldsymbol{b}^T) = \sum_{i=j} a_i \oplus b_j = \sum_i a_i b_i.$$

In other words, the effect of $\sum_{i=j}$ on a summand is to replace $i$ by $j$, or vice versa.

In summation notation, $\boldsymbol{a}^T\boldsymbol{A}\boldsymbol{a} = \sum_i \sum_j A_{ij}a_j a_i$. Using the definitions above, and recalling that $\boldsymbol{A}\boldsymbol{a}$ is just a vector with value $\sum_j A_{ij}a_j$ in the $i$'th index, we see that

$$\operatorname{tr}(\boldsymbol{A}\boldsymbol{a}\boldsymbol{a}^T) = \sum_{i=k}\left(\sum_j A_{ij}a_j\right) \oplus a_k = \sum_i \sum_j A_{ij}a_j a_i.$$

b) The likelihood is given by the expression

$$\begin{aligned} p(\mathcal{D} \mid \theta) &= \prod_{k=1}^n \frac{|\boldsymbol{\Sigma}^{-1}|^{1/2}}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_k - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_k - \boldsymbol{\mu})\right) \\ &= \frac{|\boldsymbol{\Sigma}^{-1}|^{n/2}}{(2\pi)^{nd/2}} \exp\left(-\frac{1}{2}\sum_{k=1}^n (\boldsymbol{x}_k - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_k - \boldsymbol{\mu})\right). \end{aligned}$$

Applying $\boldsymbol{a}^T\boldsymbol{A}\boldsymbol{a} = \operatorname{tr}(\boldsymbol{A}\boldsymbol{a}\boldsymbol{a}^T)$ from problem a) yields

$$p(\mathcal{D} \mid \theta) = \frac{|\boldsymbol{\Sigma}^{-1}|^{n/2}}{(2\pi)^{nd/2}} \exp\left(-\frac{1}{2}\sum_{k=1}^n \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_k - \boldsymbol{\mu})(\boldsymbol{x}_k - \boldsymbol{\mu})^T\right)\right),$$

and by using $\operatorname{tr}(\boldsymbol{A}+\boldsymbol{B}) = \operatorname{tr}(\boldsymbol{A}) + \operatorname{tr}(\boldsymbol{B})$ on the exponent we complete the problem.

c) To solve this problem, we make use of two proofs from linear algebra.

(i) The determinant is the product of the eigenvalues, i.e. $\det \boldsymbol{A} = \prod_{i=1}^d \lambda_i$. This can be proved by taking the determinant of the eigenvalue decomposition of $\boldsymbol{A}$, i.e. $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, since the determinant of an orthogonal matrix is unity and $\boldsymbol{\Lambda}$ has eigenvalues on the diagonal, the result is that $\det \boldsymbol{A} = |\boldsymbol{A}| = \prod_{i=1}^d \lambda_i$.

(ii) The trace is the sum of the eigenvalues, i.e. $\operatorname{tr} \boldsymbol{A} = \sum_{i=1}^d \lambda_i$. The proof for this involves characteristic polynomials, but will not be sketched here.

The term involving $\boldsymbol{\Sigma}^{-1}$ is transformed in the following way: if $\boldsymbol{A} = \boldsymbol{\Sigma}^{-1}\hat{\boldsymbol{\Sigma}}$, then $|\boldsymbol{A}| = |\boldsymbol{\Sigma}^{-1}|\,|\hat{\boldsymbol{\Sigma}}|$. We then write

$$\left|\boldsymbol{\Sigma}^{-1}\right| = \frac{|\boldsymbol{A}|}{|\hat{\boldsymbol{\Sigma}}|} = \frac{\prod_{i=1}^d \lambda_i}{|\hat{\boldsymbol{\Sigma}}|}.$$

To transform the exponent, we write

$$\operatorname{tr}\left(\sum_{k=1}^{n} \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}_k - \boldsymbol{\mu}\right)\left(\boldsymbol{x}_k - \boldsymbol{\mu}\right)^T\right) = \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1} n \hat{\boldsymbol{\Sigma}}\right) = n \operatorname{tr}\left(\boldsymbol{A}\right) = n \sum_{i=1}^{d} \lambda_i.$$

Substituting these transformation into the previous equation yields the required expression, which is

$$p(\mathcal{D} \mid \theta) = \frac{\left(\prod_{i=1}^{d} \lambda_i\right)^{n/2}}{(2\pi)^{nd/2} |\hat{\boldsymbol{\Sigma}}|^{n/2}} \exp\left(-\frac{n}{2} \sum_{i=1}^{d} \lambda_i\right). \tag{1}$$

d) Taking logarithms of equation (1) above gives us the log-likelihood function

$$\ln p(\mathcal{D} \mid \theta) = \frac{n}{2}\left[\sum_{i=1}^{d} \ln \lambda_i - d \ln 2\pi - \ln |\hat{\boldsymbol{\Sigma}}|\right] - \frac{n}{2} \sum_{i=1}^{d} \lambda_i,$$

and differentiating it with respect to the eigenvalue $\lambda_j$ yields

$$\frac{\partial \ln p(\mathcal{D} \mid \theta)}{\partial \lambda_j} = \frac{n}{2}\left(1 - \lambda_j^2\right) = 0.$$

The Hessian matrix (second derivative) is a diagonal matrix with $-2\lambda_j$ on the $(j,j)$'th entry, so it is negative definite (i.e. $\boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} < 0$ for every $\boldsymbol{x}$) when all $\lambda_j$'s are positive. When the Hessian is negative definite, then the solution is a maximum. Therefore we take $\lambda_j = 1$ as the solution for every $j$ (and not $\lambda_j = -1$, which is not a maximal point).

If every eigenvalues of $\boldsymbol{A}$ is 1, then $\boldsymbol{A} = \boldsymbol{I}$. Recall that $\boldsymbol{A} = \boldsymbol{\Sigma}^{-1}\hat{\boldsymbol{\Sigma}}$, so when we substitute the identity for $\boldsymbol{A}$ we obtain $\boldsymbol{\Sigma}^{-1}\hat{\boldsymbol{\Sigma}} = \boldsymbol{I}$. The likelihood is maximized when we take $\hat{\boldsymbol{\Sigma}}$ to be $n^{-1}\sum_{k=1}^{n}\left(\boldsymbol{x}_k - \boldsymbol{\mu}\right)\left(\boldsymbol{x}_k - \boldsymbol{\mu}\right)^T$, and the proof is complete.

## Problem 3.15

a) Starting with equation (31) from the book, we get rid of $\sigma_n^2$ by substituting the expression given in equation (32) to obtain

$$\mu_n = \underbrace{\left[\frac{1}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}}\right]}_{\sigma_n^2} \frac{n}{\sigma^2}\hat{\mu}_n + \underbrace{\left[\frac{1}{\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}}\right]}_{\sigma_n^2} \frac{\mu_0}{\sigma_0^2}.$$

We cancel terms, use the dogmatism $n_0 = \sigma^2/\sigma_0^2$, and realize that $\mu_0 = n_0^{-1}\sum_{k=-n_0+1}^{0} x_k$.

$$\begin{aligned}
\mu_n &= \left[\frac{1}{1+\frac{\sigma^2}{n\sigma_0^2}}\right]\hat{\mu}_n + \left[\frac{1}{1+\frac{n\sigma_0^2}{\sigma^2}}\right]\mu_0 \\
&= \left[\frac{1}{1+\frac{n_0}{n}}\right]\frac{1}{n}\sum_{k=1}^{n} x_k + \left[\frac{1}{1+\frac{n}{n_0}}\right]\frac{1}{n_0}\sum_{k=-n_0+1}^{0} x_k \\
&= \frac{1}{n+n_0}\sum_{k=1}^{n} x_k + \frac{1}{n+n_0}\sum_{k=-n_0+1}^{0} x_k = \frac{1}{n+n_0}\sum_{k=-n_0+1}^{n} x_k.
\end{aligned}$$

(a) One interpretation of $\mu_n$ is that it's a weighted average of the real samples and the fictitious samples, since

$$\mu_n = \frac{n}{n+n_0}\left[\frac{1}{n}\sum_{k=0}^{n} n\right] + \frac{n_0}{n+n_0}\left[\frac{1}{n_0}\sum_{k=-n_0+1}^{0} x_k\right] = \frac{1}{n+n_0}\sum_{k=-n_0+1}^{n} x_k$$

An interpretation of $\sigma_n^2$ is more straightforward if we consider it's inverse, the *precision* $\sigma_n^{-2}$. The precision can be written as $\sigma_n^{-2} = n\sigma^{-2} + n_0\sigma^{-2}$, so the number of fictitious samples provides an initial estimate for the precision, and as more real data points are added the precision is increased linearly.

**Problem 3.16**

a) The trick is to multiply both terms by $\boldsymbol{I}$ at an opportune moment, expanding the identity and using $(\boldsymbol{AB})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$. We prove the theorem by writing

$$\begin{aligned}
\left(\boldsymbol{A}^{-1}+\boldsymbol{B}^{-1}\right)^{-1} &= \left(\boldsymbol{A}^{-1}\boldsymbol{I}+\boldsymbol{I}\boldsymbol{B}^{-1}\right)^{-1} = \left(\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{B}^{-1}+\boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{B}^{-1}\right)^{-1} \\
&= \left(\boldsymbol{A}^{-1}\left(\boldsymbol{B}+\boldsymbol{A}\right)\boldsymbol{B}^{-1}\right)^{-1} = \boldsymbol{B}\left(\boldsymbol{A}+\boldsymbol{B}\right)^{-1}\boldsymbol{A}.
\end{aligned}$$

Replacing the second equality by $\boldsymbol{I}\boldsymbol{A}^{-1}+\boldsymbol{B}^{-1}\boldsymbol{I} = \boldsymbol{B}^{-1}\boldsymbol{B}\boldsymbol{A}^{-1}+\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{A}^{-1}$ would yield the second quality in the theorem as stated in the problem.

b) The matrices must be square, since we use $\boldsymbol{I} = \boldsymbol{A}^{-1}\boldsymbol{A}$ to prove the first equality, and $\boldsymbol{I} = \boldsymbol{A}^{-1}\boldsymbol{A}$ to prove the second. The same logic applies to $\boldsymbol{B}$. In other words, we require that $\boldsymbol{A}$ has a left-inverse and a right-inverse, and it must therefore be square.

An alternative approach is to consider the second equality in the theorem directly. Some algebra reveals that this equality requires $\boldsymbol{A}$ to have a left and right inverse. It must therefore be square, against since no non-square matrix has a left-inverse and a right-inverse.

c) Our staring point is the equations

$$\boldsymbol{\Sigma}_n^{-1} = n\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}_0^{-1} \quad \text{and} \quad \boldsymbol{\Sigma}_n^{-1}\boldsymbol{\mu}_n = n\boldsymbol{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_n + \boldsymbol{\Sigma}_0^{-1} + \boldsymbol{\mu}_0,$$

and we wish to solve these equations with respect to $\boldsymbol{\Sigma}_n$ and $\boldsymbol{\mu}_n$. In other words, we want the functional dependence to be $\boldsymbol{\Sigma}_n = f\left(\hat{\boldsymbol{\mu}}_n, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_0\right)$ and $\boldsymbol{\mu}_n = \left(\hat{\boldsymbol{\mu}}_n, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_0\right)$.

We start with the covariance matrix. To solve for $\boldsymbol{\Sigma}_n$, we write

$$\boldsymbol{\Sigma}_n = \left(\boldsymbol{\Sigma}_n^{-1}\right)^{-1} = \left(n\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}_0^{-1}\right)^{-1} = \boldsymbol{\Sigma}_0 \left(\boldsymbol{\Sigma}_0 + \frac{1}{n}\boldsymbol{\Sigma}\right)^{-1} \frac{1}{n}\boldsymbol{\Sigma},$$

where the last equality comes from using

$$\left(\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1}\right)^{-1} = \boldsymbol{B}^{-1}\left(\boldsymbol{B} + \boldsymbol{A}\right)^{-1}\boldsymbol{A}^{-1}.$$

To solve for the mean $\boldsymbol{\mu}_n$, we write

$$\begin{aligned}
\boldsymbol{\mu}_n &= \boldsymbol{\Sigma}_n n \boldsymbol{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_n + \boldsymbol{\Sigma}_n\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0 \\
&= \left[\boldsymbol{\Sigma}_0\left(\boldsymbol{\Sigma}_0 + \frac{1}{n}\boldsymbol{\Sigma}\right)^{-1}\frac{1}{n}\boldsymbol{\Sigma}\right] n\boldsymbol{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_n + \left[\frac{1}{n}\boldsymbol{\Sigma}\left(\frac{1}{n}\boldsymbol{\Sigma} + \boldsymbol{\Sigma}_0\right)^{-1}\boldsymbol{\Sigma}_0\right]\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0 \\
&= \boldsymbol{\Sigma}_0\left(\boldsymbol{\Sigma}_0 + \frac{1}{n}\boldsymbol{\Sigma}\right)^{-1}\hat{\boldsymbol{\mu}}_n + \frac{1}{n}\boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}_0 + \frac{1}{n}\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\mu}_0
\end{aligned}$$

Where we made use of both equalities given in the theorem from subproblem a).

### Problem 3.24

We are tasked to find the maximum likelihood estimate of $\theta$ in the Rayleigh distribution, which is given by

$$p(x \mid \theta) = \begin{cases} 2\theta x \exp\left(-\theta x^2\right) & \text{if } x \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

The log-likelihood function is given by

$$\ell(\theta) = \ln\left(p(\mathcal{D} \mid \theta)\right) = \sum_{k=1}^{n} \ln\left(2\theta x \exp\left(-\theta x^2\right)\right) = \sum_{k=1}^{n} -\theta x_i^2 + \ln 2\theta x_i.$$

Differentiating the log-likelihood with respect to $\theta$ yields

$$\ell'(\theta) = \sum_{k=1}^{n}\left(-x_i^2 + \frac{2x_i}{2\theta x_i}\right) = \sum_{k=1}^{n}\left(-x_i^2 + \frac{1}{\theta}\right) = \frac{n}{\theta} - \sum_{k=1}^{n} x_i^2 = 0,$$

and solving this equation for $\theta$ reveals the desired answer.

### Problem 3.32

a) A simple $\Theta(n^2)$ algorithm is given by

```
f = 0
for i=0 to n-1 do:
    f = f + a[i] * x**i
end
```

Assuming that the computation of $x^i$ requires $i-1$ flops, the complexity for iteration $i$ is $\Theta(i)$, and the full complexity becomes $1+2+3+\cdots+(n-1)=\Theta(n^2)$.

b) The waste in the algorithm from problem a) above stems from having to compute powers of $x$ many times. If we know $x^{k-1}$, then $x^k = x^{k-1}x$, and there is no need to compute $x^k$ as $\underbrace{x \cdot x \cdot \ldots \cdot x}_{k \text{ times}}$. Define $S_k := \sum_{i=k}^{n-1} a_i x^{i-k}$, then

$$f(x) = S_0 = a_0 + xS_1 = a_0 + x\left(a_1 + xS_2\right) = a_0 + x\left(a_1 + x\left(a_2 + xS_3\right)\right).$$

Clearly $S_k = a_k + xS_{k+1}$, looping backwards we can use the following algorithm

```
f = a_[n-1]
for i=n-2 to 0 do:
    f = a[i] + x * f
end
```

which requires a constant number flops in each iteration. The computational complexity is therefore $\Theta(n)$. A specific example when $n=4$ is

$$a_0x^0 + a_1x^1 + a_2x^2 + a_3x^3 = a_0 + x(a_1 + x(a_2 + x(a_3 + x))),$$

and the algorithm evaluates this using the right-hand side, from the inside and out.

## Problem 3.35

a) The complexity of computing $\hat{\boldsymbol{\mu}}_n$ is $O(nd)$, where $n$ is the number of data points and $d$ is the dimension. Adding $n$ vectors of length $d$ requires $d(n-1)$ floating point operations (flops), diving through by $n$ requires another $d$ flops. The resulting complexity is therefore $d(n-1)+d = O(nd)$.

The complexity of computing $\boldsymbol{C}_n$ is $O(nd^2)$. For each term in the sum, we compute the subtraction by $d$ flops, then the outer product using $d^2$ flops. This is done of each of the $n$ terms, so we need $n(d+d^2)$ flops to compute the terms. We then add the terms, requiring $d^2(n-1)$ flops. Diving requires $d^2$ flops. In total, the cost is $n(d+d^2) + d^2(n-1) + d^2 \sim 2nd^2 = O(nd^2)$ flops.

b) We will show that these equations are indeed correct by two different methods.

The recursive definition of $\hat{\boldsymbol{\mu}}_n$ may be proved by induction. It's clearly valid for $n=0$, since then it reduces to $\hat{\boldsymbol{\mu}}_0 = \boldsymbol{x}_1$. The inductive step is

$$\begin{aligned}\hat{\boldsymbol{\mu}}_{n+1} &= \hat{\boldsymbol{\mu}}_n + \frac{1}{n+1}\left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{\mu}}_n\right) = \left(1 - \frac{1}{n+1}\right)\hat{\boldsymbol{\mu}}_n + \frac{1}{n+1}\boldsymbol{x}_{n+1} \\ &= \left(\frac{n}{n+1}\right)\hat{\boldsymbol{\mu}}_n + \frac{1}{n+1}\boldsymbol{x}_{n+1} = \frac{1}{n+1}\sum_{k=1}^{n}\boldsymbol{x}_k + \frac{1}{n+1}\boldsymbol{x}_{n+1} = \frac{1}{n+1}\sum_{k=1}^{n+1}\boldsymbol{x}_k,\end{aligned}$$

and this proves the recursive relation for $\hat{\boldsymbol{\mu}}_{n+1}$.

We will prove that the recursive equation for $\boldsymbol{C}_{n+1}$ is correct by deriving it from the definition. This is a somewhat tedious computation, and the strategy will be to write $\boldsymbol{C}_{n+1}$ as a function of known terms, i.e. $\boldsymbol{C}_n$, $\hat{\boldsymbol{\mu}}_n$, $\hat{\boldsymbol{\mu}}_{n+1}$ and $\boldsymbol{x}_{n+1}$. We staring by writing the definition of $\boldsymbol{C}_{n+1}$, which is

$$\boldsymbol{C}_{n+1} = \frac{1}{n} \sum_{k=1}^{n+1} \left(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_{n+1}\right) \left(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_{n+1}\right)^T .$$

We then subtract and add $\hat{\boldsymbol{\mu}}_n$ to write the outer product as

$$\left((\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_n) - (\hat{\boldsymbol{\mu}}_{n+1} - \hat{\boldsymbol{\mu}}_n)\right) \left((\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_n) - (\hat{\boldsymbol{\mu}}_{n+1} - \hat{\boldsymbol{\mu}}_n)\right)^T . \tag{2}$$

The last term in each product may be written as

$$\hat{\boldsymbol{\mu}}_{n+1} - \hat{\boldsymbol{\mu}}_n = \frac{1}{n+1} \left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{\mu}}_n\right) .$$

Equation (2) has the same functional form as

$$(\boldsymbol{a} - \boldsymbol{b})(\boldsymbol{a} - \boldsymbol{b})^T = \boldsymbol{a}\boldsymbol{a}^T - \boldsymbol{a}\boldsymbol{b}^T - \boldsymbol{b}\boldsymbol{a}^T + \boldsymbol{b}\boldsymbol{b}^T,$$

and we expand it as such to obtain the following sum

$$\begin{aligned} \boldsymbol{C}_{n+1} = \frac{1}{n} \sum_{k=1}^{n+1} & \left(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_n\right) \left(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_n\right)^T + \left(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_n\right) \frac{1}{n+1} \left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{\mu}}_n\right)^T \\ & + \frac{1}{n+1} \left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{\mu}}_n\right) \left(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_n\right) + \frac{1}{n+1} \left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{\mu}}_n\right) \frac{1}{n+1} \left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{\mu}}_n\right)^T , \end{aligned} \tag{3}$$

and we will study each of the four preceding terms in order.

The **first term** in Equation (3) may be written as

$$\frac{1}{n} \sum_{k=1}^{n+1} \left(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_n\right) \left(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_n\right)^T = \frac{n-1}{n} \boldsymbol{C}_n + \frac{1}{n} \left(\boldsymbol{x}_{n+1} - \boldsymbol{\mu}_n\right) \left(\boldsymbol{x}_{n+1} - \boldsymbol{\mu}_n\right)^T ,$$

where we stripped out the last term in the sum and used the definition of $\boldsymbol{C}_n$.

The **second term** in Equation (3) may be written as

$$\frac{1}{n(n+1)} \left[\sum_{k=1}^{n+1} \left(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_n\right)\right] \left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{\mu}}_n\right)^T = \frac{1}{n(n+1)} \left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{\mu}}_n\right) \left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{\mu}}_n\right)^T ,$$

since $\sum_{k=1}^{n+1} \left(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_n\right) = \left(\sum_{k=1}^{n} \boldsymbol{x}_k\right) + \boldsymbol{x}_{n+1} - (n+1)\hat{\boldsymbol{\mu}}_n = n\hat{\boldsymbol{\mu}}_n + \boldsymbol{x}_{n+1} - n\hat{\boldsymbol{\mu}}_n - \hat{\boldsymbol{\mu}}_n$.

The **third term** in Equation (3) may also be written as

$$\frac{1}{n(n+1)} \left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{\mu}}_n\right) \left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{\mu}}_n\right)^T ,$$

where we used the same logic as in computations related to the second term.

The **fourth term** in Equation (3) may be written as

$$\frac{1}{n(n+1)}\left(\boldsymbol{x}_{n+1}-\hat{\boldsymbol{\mu}}_n\right)\left(\boldsymbol{x}_{n+1}-\hat{\boldsymbol{\mu}}_n\right)^T,$$

since the entire term is constant with respect to the index $k$. We multiplied the last term with $(n+1)$ to get it out of the sum, canceled part of the fraction, and applied the $n^{-1}$ fraction from outside the sum.

Let's return Equation (3) again, and write $\boldsymbol{w}_k = (\boldsymbol{x}_{n+1}-\hat{\boldsymbol{\mu}}_n)$ to ease notation. Using our findings from above, the sum becomes

$$\begin{aligned}\boldsymbol{C}_{n+1} =& \frac{n-1}{n}\boldsymbol{C}_n + \frac{1}{n}\boldsymbol{w}\boldsymbol{w}^T - \frac{1}{n(n+1)}\boldsymbol{w}\boldsymbol{w}^T \\ &- \frac{1}{n(n+1)}\boldsymbol{w}\boldsymbol{w}^T + \frac{1}{n(n+1)}\boldsymbol{w}\boldsymbol{w}^T,\end{aligned}$$

and since $1/n - 1/(n(n+1)) = 1/(n+1)$ we obtain the desired result

$$\boldsymbol{C}_{n+1} = \frac{n-1}{n}\boldsymbol{C}_n + \frac{1}{n+1}\boldsymbol{w}\boldsymbol{w}^T.$$

This concludes our derivation of the recursive formulas.

c) The calculation of $\hat{\boldsymbol{\mu}}_{n+1}$ is dominated by vector addition and subtraction, and the complexity is $O(d)$. The calculation of $\boldsymbol{C}_{n+1}$ is dominated by an outer product, which has $O(d^2)$ complexity, and a matrix addition, which is $O(d^2)$ too. The overall complexity is for the iterative sample covariance matrix $\boldsymbol{C}_{n+1}$ is therefore $O(d^2)$.

d) When data arrives in a stream (on-line learning), the recursive methods are clearly superior to application of the naive formulas. Consider data iteratively entering a learning system. If we compute $\hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\mu}}_2, \ldots, \hat{\boldsymbol{\mu}}_n$ naively, the overall complexity is

$$1d + 2d + \cdots + nd = (1+2+\cdots+n)\,d = O(n^2 d).$$

If we compute the sequence using the iterative equation, the complexity resolves to

$$1d + 1d + \cdots + 1d = (1+1+\cdots+1)\,d = O(nd).$$

Using the same logic, naive computation of $\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_n$ will have complexity $O(n^2d^2)$, while using the recursive formulas result in $O(nd^2)$ complexity. If data arrives sequentially and predictions are to be made before all of the data has arrived, then clearly the recursive formulas are superior. They also require less intermediate storage. It would be interesting to know the numerical stability of both approaches.

**Problem 3.36**

a) We'll prove the Sherman-Morrison-Woodbury matrix identity by demonstrating that it reduces to $I = I$. Recall that $1 + \boldsymbol{y}^T A^{-1}\boldsymbol{x}$ is a scalar.

$$\begin{aligned}
\left(A + \boldsymbol{x}\boldsymbol{y}^T\right)^{-1} &= A^{-1} - \frac{A^{-1}\boldsymbol{x}\boldsymbol{y}^T A^{-1}}{1 + \boldsymbol{y}^T A^{-1}\boldsymbol{x}} \\
\left(A + \boldsymbol{x}\boldsymbol{y}^T\right)\left(A + \boldsymbol{x}\boldsymbol{y}^T\right)^{-1} &= \left(A + \boldsymbol{x}\boldsymbol{y}^T\right)A^{-1} - \left(A + \boldsymbol{x}\boldsymbol{y}^T\right)\frac{A^{-1}\boldsymbol{x}\boldsymbol{y}^T A^{-1}}{1 + \boldsymbol{y}^T A^{-1}\boldsymbol{x}} \\
I &= I + \boldsymbol{x}\boldsymbol{y}^T A^{-1} - \frac{\boldsymbol{x}\boldsymbol{y}^T A^{-1} + \boldsymbol{x}\boldsymbol{y}^T A^{-1}\boldsymbol{x}\boldsymbol{y}^T A^{-1}}{1 + \boldsymbol{y}^T A^{-1}\boldsymbol{x}} \\
I &= I + \boldsymbol{x}\boldsymbol{y}^T A^{-1} - \frac{\boldsymbol{x}\left(1 + \boldsymbol{y}^T A^{-1}\boldsymbol{x}\right)\boldsymbol{y}^T A^{-1}}{1 + \boldsymbol{y}^T A^{-1}\boldsymbol{x}} \\
I &= I + \boldsymbol{x}\boldsymbol{y}^T A^{-1} - \boldsymbol{x}\boldsymbol{y}^T A^{-1}
\end{aligned}$$

b) Recall the recursive equation for the sample covariance matrix, where we define $a$, $b$ and $\boldsymbol{w}$ to ease notation in the following derivation.

$$\boldsymbol{C}_{n+1} = \underbrace{\frac{n-1}{n}}_{a}\boldsymbol{C}_n + \underbrace{\frac{1}{n+1}}_{b}\underbrace{\left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{\mu}}_n\right)}_{\boldsymbol{w}}\underbrace{\left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{\mu}}_n\right)^T}_{\boldsymbol{w}^T} = a\boldsymbol{C}_n + b\boldsymbol{w}\boldsymbol{w}^T$$

The inverse is given by

$$\boldsymbol{C}_{n+1}^{-1} = \left(a\boldsymbol{C}_n + b\boldsymbol{w}\boldsymbol{w}^T\right)^{-1},$$

and we now apply the Sherman-Morrison-Woodbury matrix identity to obtain

$$\begin{aligned}
\boldsymbol{C}_{n+1}^{-1} = \left(a\boldsymbol{C}_n + b\boldsymbol{w}\boldsymbol{w}^T\right)^{-1} &= \left(a\boldsymbol{C}_n\right)^{-1} - \frac{\left(a\boldsymbol{C}_n\right)^{-1} b\boldsymbol{w}\boldsymbol{w}^T\left(a\boldsymbol{C}_n\right)^{-1}}{1 + \boldsymbol{w}^T\left(a\boldsymbol{C}_n\right)^{-1} b\boldsymbol{w}} \\
&= \frac{1}{a}\left(\boldsymbol{C}_n^{-1} - \frac{b}{a}\left(\frac{\boldsymbol{C}_n^{-1}\boldsymbol{w}\boldsymbol{w}^T\boldsymbol{C}_n^{-1}}{1 + \frac{b}{a}\boldsymbol{w}^T\boldsymbol{C}_n^{-1}\boldsymbol{w}}\right)\right) \\
&= \frac{1}{a}\left(\boldsymbol{C}_n^{-1} - \frac{\boldsymbol{C}_n^{-1}\boldsymbol{w}\boldsymbol{w}^T\boldsymbol{C}_n^{-1}}{\frac{a}{b} + \boldsymbol{w}^T\boldsymbol{C}_n^{-1}\boldsymbol{w}}\right).
\end{aligned}$$

Using the fact that $a^{-1} = n/(n-1)$ and $a/b = (n^2 - 1)/n$, we have shown that the inverse of $\boldsymbol{C}_{n+1}$ is indeed given by

$$\boldsymbol{C}_{n+1}^{-1} = \frac{n}{n-1}\left(\boldsymbol{C}_n^{-1} - \frac{\boldsymbol{C}_n^{-1}\boldsymbol{w}\boldsymbol{w}^T\boldsymbol{C}_n^{-1}}{\frac{n^2-1}{n} + \boldsymbol{w}^T\boldsymbol{C}_n^{-1}\boldsymbol{w}}\right).$$

c) The computational complexity is $O(d^2)$. First $\boldsymbol{w}$ is computed using $O(d)$ flops. In the numerator, the matrix-vector product $\boldsymbol{x} = \boldsymbol{C}_n^{-1}\boldsymbol{w}$ is computed in $O(d^2)$ time, and so is $\boldsymbol{y}^T = \boldsymbol{w}^T\boldsymbol{C}_n^{-1}$. Then the outer product $\boldsymbol{x}\boldsymbol{y}^T = \left(\boldsymbol{C}_n^{-1}\boldsymbol{w}\right)\left(\boldsymbol{w}^T\boldsymbol{C}_n^{-1}\right)$ may be computed in $O(d^2)$ time too.

The denominator is also computed in $O(d^2)$ time, and so is the matrix subtraction. Therefore the overall computational complexity is $O(d^2)$. Notice that if $\boldsymbol{w}\boldsymbol{w}^T$ is computed first in the denominator, the computational complexity would be $O(d^3)$, since matrix-matrix products are $O(d^3)$.

d) See the answer to problem 35 d).

## Problem 3.38

a) The criterion function is given by

$$J_1(\boldsymbol{w}) = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} = \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \left( \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2 \right) \boldsymbol{w}},$$

since the numerator is given by

$$\begin{aligned} (\mu_1 - \mu_2)^2 &= \left( \boldsymbol{w}^T \boldsymbol{\mu}_1 - \boldsymbol{w}^T \boldsymbol{\mu}_2 \right) \left( \boldsymbol{w}^T \boldsymbol{\mu}_1 - \boldsymbol{w}^T \boldsymbol{\mu}_2 \right)^T \\ &= \boldsymbol{w}^T \left( \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 \right) \left( \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 \right)^T \boldsymbol{w} = \boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w} \end{aligned}$$

and the denominator is given by

$$\sigma_1^2 + \sigma_1^2 = \boldsymbol{w}^T \left( \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2 \right) \boldsymbol{w}.$$

Compare this with equation (96) in the book. By the same logic used there, the solution is given by equation (106) from the book, which is

$$\boldsymbol{w} = \left( \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2 \right)^{-1} \left( \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 \right) = \left( \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \right) \left( \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 \right).$$

b) Simply substitute $\sigma_i^2 \to P(\omega_i)\sigma_i^2$ into problem a).

c) Recall that $\tilde{s}_i^2 = \sum_{y \in \mathcal{Y}_i} (y_i - \tilde{m}_i)^2$. This expression is $n_i$ times the population variance, so $\tilde{s}_i^2 \approx n_i \sigma_i^2$. Equation (96) in the book therefore has $n_1\sigma_1^2 + n_2\sigma_2^2$ in the denominator, $J_1(\boldsymbol{w})$ from subproblem a) had $\sigma_1^2 + \sigma_2^2$ in the denominator and the denominator of $J_2(\boldsymbol{w})$ is

$$P(\omega_1)\sigma_1^2 + P(\omega_2)\sigma_2^2 = \frac{n_1}{n}\sigma_1^2 + \frac{n_2}{n}\sigma_2^2 = \frac{1}{n}\left( n_1\sigma_1^2 + n_2\sigma_2^2 \right).$$

The denominator in equation (96) in the book is proportional to $J_2(\boldsymbol{w})$, so they are the most similar. Their optimization results in the same value of $\boldsymbol{w}$, since constants make no difference when optimizing $J(\boldsymbol{w})$.

## Problem 3.31

a) We start by expanding terms

$$\begin{aligned} J_1 &= \frac{1}{n_1 n_2} \sum_i \sum_j \left( y_i^2 - 2y_i y_j + y_j^2 \right) \\ &= \frac{1}{n_1 n_2} \left( n_2 \sum_i y_i^2 - 2 \left( \sum_i y_i \right) \left( \sum_j y_j \right) + n_1 \sum_j y_j^2 \right). \end{aligned}$$

From $\operatorname{var}(Y) = \mathbb{E}(Y^2) - \mathbb{E}(Y)^2$ we note that

$$\sum_i y_i^2 = \sum_i (y_i - m_i)^2 + \frac{1}{n_i}\left(\sum_i y_i\right)^2 = s_1^2 + \frac{1}{n_i}\left(\sum_i y_i\right)^2.$$

Now we denote $k_i = \sum_i y_i$ and $k_j = \sum_j y_j$, then we substitute the equation above into the equation for $J_1$ to obtain

$$\begin{aligned} J_1 &= \frac{1}{n_1 n_2}\left[n_2\left(s_1^2 + \frac{1}{n_1}k_i^2\right) - 2k_i k_j + n_1\left(s_2^2 + \frac{1}{n_2}k_j^2\right)\right] \\ &= \frac{s_1^2}{n_1} + \frac{s_2^2}{n_2} + \frac{k_i^2}{n_1^2} - 2\frac{k_i k_j}{n_1 n_2} + \frac{k_j^2}{n_2^2} \\ &= \frac{s_1^2}{n_1} + \frac{s_2^2}{n_2} + m_1^2 - 2m_1 m_2 + m_2^2 = \frac{s_1^2}{n_1} + \frac{s_2^2}{n_2} + (m_1 - m_2)^2. \end{aligned}$$

b) Not sure about this one. TODO
c) Not sure about this one. TODO

## Problem 3.40

a) We have $\tilde{\boldsymbol{S}}_W = \boldsymbol{W}^T \boldsymbol{S}_W \boldsymbol{W}$, using the fact that $\boldsymbol{W}$ contains eigenvector columns with $\boldsymbol{e}_i^T \boldsymbol{S}_W \boldsymbol{e}_j^T = \delta_{ij}$, we observe that

$$\begin{aligned} \boldsymbol{W}^T \boldsymbol{S}_W \boldsymbol{W} &= \begin{pmatrix} \boldsymbol{e}_1^T \\ \vdots \\ \boldsymbol{e}_n^T \end{pmatrix} \begin{pmatrix} \boldsymbol{S}_W \boldsymbol{e}_1 & \dots & \boldsymbol{S}_W \boldsymbol{e}_n \end{pmatrix} = \begin{pmatrix} \boldsymbol{e}_1^T \\ \vdots \\ \boldsymbol{e}_n^T \end{pmatrix} \boldsymbol{S}_W \begin{pmatrix} \boldsymbol{e}_1 & \dots & \boldsymbol{e}_n \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{e}_1^T \boldsymbol{S}_W \boldsymbol{e}_1 & \boldsymbol{e}_1^T \boldsymbol{S}_W \boldsymbol{e}_2 & \dots \\ \boldsymbol{e}_2^T \boldsymbol{S}_W \boldsymbol{e}_1 & \boldsymbol{e}_2^T \boldsymbol{S}_W \boldsymbol{e}_2 & \vdots \\ \vdots & \dots & \boldsymbol{e}_n^T \boldsymbol{S}_W \boldsymbol{e}_n \end{pmatrix} = \delta_{ij} = \boldsymbol{I}. \end{aligned}$$

The same exact procedure may be applied to $\tilde{\boldsymbol{S}}_B$ to show that it's a diagonal matrix with eigenvalues, $\boldsymbol{\Lambda}$. We will not devote space to this computation.

b) Applying the result from a), we see that the value of $J$ is

$$J = \frac{\left|\tilde{\boldsymbol{S}}_B\right|}{\left|\tilde{\boldsymbol{S}}_W\right|} = \frac{\left|\boldsymbol{W}^T \boldsymbol{S}_B \boldsymbol{W}\right|}{\left|\boldsymbol{W}^T \boldsymbol{S}_W \boldsymbol{W}\right|} = \frac{|\boldsymbol{\Lambda}|}{|\boldsymbol{I}|} = \prod_{i=1}^n \lambda_i.$$

c) The relation is $\boldsymbol{y} = \boldsymbol{W}^T \boldsymbol{x}$, and the relation after scaling and rotating becomes $\boldsymbol{y}' = \boldsymbol{Q}\boldsymbol{D}\boldsymbol{W}^T \boldsymbol{x}$. We replace $\boldsymbol{W}^T$ by $\boldsymbol{Q}\boldsymbol{D}\boldsymbol{W}^T$ and obtain

$$J' = \frac{\left|\left(\boldsymbol{Q}\boldsymbol{D}\boldsymbol{W}^T\right) \boldsymbol{S}_B \left(\boldsymbol{Q}\boldsymbol{D}\boldsymbol{W}^T\right)^T\right|}{\left|\left(\boldsymbol{Q}\boldsymbol{D}\boldsymbol{W}^T\right) \boldsymbol{S}_W \left(\boldsymbol{Q}\boldsymbol{D}\boldsymbol{W}^T\right)^T\right|} = \frac{|\boldsymbol{Q}|\,|\boldsymbol{D}|\,|\boldsymbol{\Lambda}|\,|\boldsymbol{D}|\,|\boldsymbol{Q}^{-1}|}{|\boldsymbol{Q}|\,|\boldsymbol{D}|\,|\boldsymbol{I}|\,|\boldsymbol{D}|\,|\boldsymbol{Q}^{-1}|} = \frac{|\boldsymbol{\Lambda}|}{|\boldsymbol{I}|} = J,$$

where we used $\boldsymbol{W}^T \boldsymbol{S}_W \boldsymbol{W} = \boldsymbol{I}$ and $\boldsymbol{W}^T \boldsymbol{S}_B \boldsymbol{W} = \boldsymbol{\Lambda}$, as well as $\boldsymbol{Q}^T = \boldsymbol{Q}^{-1}$. Clearly $\boldsymbol{W}^T \to \boldsymbol{Q}\boldsymbol{D}\boldsymbol{W}^T$ leaves $J$ unchanged, so it's invariant to this transformation.

## 2.4 Nonparametrix techniques

### Problem 4.2

a) We'll prove $\bar{p}_n(x) \sim \mathcal{N}(\mu, \sigma^2 + h_n^2)$ by direct computation. An alternative approach would be to use the Fourier convolution theorem, which states that convolution becomes pointwise-multiplication in the Fourier basis.

   We start with
$$\bar{p}_n(x) = \int \frac{1}{h_n}\phi\left(\frac{x-v}{h_n}\right) p(v)\, dv,$$
   and if $\phi(x) \sim \mathcal{N}(0,1)$ then it is easy to see that $\frac{1}{h_n}\phi\left(\frac{x-v}{h_n}\right) \sim \mathcal{N}(v, h_n^2)$. We'll expand the exponents, write it as a quadratic function of $v$, integrate out the $v$-variable and with algebra turn the result into a well-known form. The integral is
$$\frac{1}{\sqrt{2\pi}\sigma}\frac{1}{\sqrt{2\pi}h_n}\int \exp\left[-\frac{1}{2}\left(\left(\frac{x-v}{h_n}\right)^2 + \left(\frac{v-\mu}{\sigma}\right)^2\right)\right] dv, \tag{4}$$
   and we wish to integrate out the $v$. To do so, we write the exponent as a function of $v$. Defining $\beta = x^2\sigma^2 + h_n^2\mu^2$, the exponent may be written as
$$\frac{\sigma^2\left(x^2 - 2xv + v^2\right) + h_n^2\left(v^2 - 2v\mu + \mu^2\right)}{(h_n\sigma)^2} = \frac{v^2(\sigma^2 + h_n^2) - 2v(x\sigma^2 + \mu h_n^2) + \beta}{(h_n\sigma)^2}.$$
   Now we introduce $y = v\sqrt{\sigma^2 + h_n^2}$, since we are aiming to write the exponent as $(y-a)^2/b^2 + c$ for some $y$. Defining $\alpha$ and completing the square, the right hand side of the equation above may be written as
$$\frac{y^2 - 2v\overbrace{\left(\frac{x\sigma^2 + \mu h_n^2}{\sqrt{\sigma^2 + h_n^2}}\right)}^{\alpha} + \beta}{(h_n\sigma)^2} = \frac{y^2 - 2v\alpha + \beta}{(h_n\sigma)^2} = \frac{(y-\alpha)^2 - \alpha^2 + \beta}{(h_n\sigma)^2}.$$
   We return to the integral in (4), use $dv = (\sigma^2 + h_n^2)^{-1/2}dy$ and write
$$\frac{1}{\sqrt{2\pi}\sigma}\frac{1}{\sqrt{2\pi}h_n}\frac{1}{\sqrt{\sigma^2 + h_n^2}}\exp\left[-\frac{1}{2}\left(\frac{-\alpha^2 + \beta}{(h_n\sigma)^2}\right)\right]\underbrace{\int \exp\left[-\frac{1}{2}\left(\frac{y-\alpha}{h_n\sigma}\right)^2\right] dy}_{=\sqrt{2\pi}h_n\sigma},$$
   where the integral is evaluated easily by virtue of being $\mathcal{N}(\alpha, h_n^2\sigma^2)$. We have
$$\bar{p}_n(x) = \frac{1}{\sqrt{2\pi}\sqrt{\sigma^2 + h_n^2}}\exp\left[-\frac{1}{2}\left(\frac{\beta - \alpha^2}{(h_n\sigma)^2}\right)\right]$$

and all that remains is to clean up the exponent. We write

$$\begin{aligned}\frac{-\alpha^2+\beta}{(h_n\sigma)^2} &= \frac{1}{(h_n\sigma)^2}\left[\frac{(\sigma^2+h_n^2)(x^2\sigma^2+h_n^2\mu^2)}{\sigma^2+h_n^2} - \frac{\left(x\sigma^2+\mu h_n^2\right)^2}{\sigma^2+h_n^2}\right] \\ &= \frac{1}{\sigma^2+h_n^2}\left(x^2+2\mu x+\mu^2\right) = \frac{(x-\mu)^2}{\sigma^2+h_n^2},\end{aligned}$$

where some tedious algebra was omitted. Finally the integral becomes

$$\bar{p}_n(x) = \frac{1}{\sqrt{2\pi}\sqrt{\sigma^2+h_n^2}}\exp\left[-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2+h_n^2}\right]$$

which is clearly $\mathcal{N}(\mu, \sigma^2+h_n^2)$, and this is what we were asked to show.

**Problem 4.3**

a) The mean of the Parzen-window estimate is given by

$$\bar{p}(x) = \frac{1}{V_n}\int p(v)\phi\left(\frac{x-v}{h_n}\right)\,dv,$$

where $p(v) = U(0,a)$. Two observations about when this integral is zero are needed.

(i) $p(v) = U(0,a)$ is zero outside of $0 < v < a$.

(ii) $\phi\left(\frac{x-v}{h_n}\right)$ is zero when $x - v > 0 \Leftrightarrow v < x$.

Let's consider the integral in every case.

**When** $x < 0$, $v$ must be 0 too since $v < x$, and the integral is zero.

**When** $0 < x < a$, $v$ can range from 0 to $x$. We obtain

$$\begin{aligned}\bar{p}(x) &= \frac{1}{V_n}\int p(v)\phi\left(\frac{x-v}{h_n}\right)dv = \frac{1}{h_n}\int_0^x \frac{1}{a}\exp\left(\frac{v-x}{h_n}\right)dv \\ &= \frac{1}{ah_n}h_n\exp\left(\frac{v-x}{h_n}\right)\Bigg|_{v=0}^{v=x} = \frac{1}{a}\left(1-\exp\left(\frac{-x}{h_n}\right)\right).\end{aligned}$$

**When** $x > a$, $v$ is not affected by $x$ and ranges from 0 to $a$. We obtain

$$\begin{aligned}\bar{p}(x) &= \frac{1}{V_n}\int p(v)\phi\left(\frac{x-v}{h_n}\right)dv = \frac{1}{h_n}\int_a^a \frac{1}{a}\exp\left(\frac{v-x}{h_n}\right)dv \\ &= \frac{1}{ah_n}h_n\exp\left(\frac{v-x}{h_n}\right)\Bigg|_{v=0}^{v=a} = \frac{1}{a}\left(\exp\left(\frac{a-x}{h_n}\right)-\exp\left(\frac{-x}{h_n}\right)\right) \\ &= \frac{1}{a}\left(\exp\left(\frac{a}{h_n}\right)-1\right)\exp\left(\frac{-x}{h_n}\right).\end{aligned}$$

b) The plot is found in figure 8.

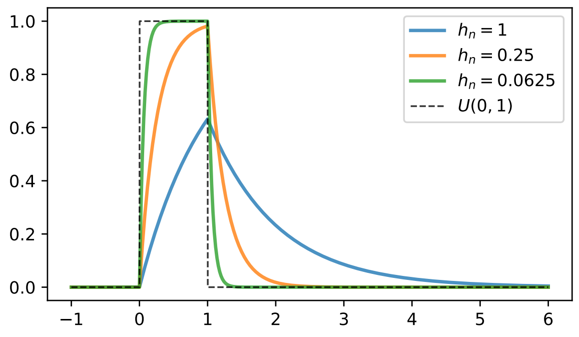

Figure 8: Plot of $\bar{p}(x)$ versus $x$ for $a = 1$ and $h = 1$, $1/4$ and $1/16$.

c) The bias is $\mathbb{E}(p(x) - \hat{p}(x)) = p(x) - \bar{p}(x)$, and we obtain the relative bias (in percentage) by diving with $p(x)$ so that

$$\text{bias}(x) = \frac{|p(x) - \bar{p}(x)|}{p(x)} = \frac{\frac{1}{a} - \bar{p}(x)}{\frac{1}{a}} = 1 - ap(x) = e^{-x/h_n}.$$

The bias is decreasing on $0 < x < a$, so if we want the bias to be less than 0.01 on 99% of the interval, it amounts to requiring that

$$\text{bias}\left(\frac{a}{100}\right) = 0.01 \quad \Leftrightarrow \quad \exp\left(-\frac{a}{100h_n}\right) = 0.01.$$

Solving this for $h_n$ yields $h_n = a/(100 \ln 100)$.

d) When $a = 0$, $h_n$ becomes $h_n = 1/(100 \ln 100) \approx 0.0022$. See figure 9 for a plot. Although it is hard to tell exactly from the plot, observe that when $x = 0.01$, the relative bias is indeed close to 0.01 so that $\bar{p}(0.01) = 0.99$.

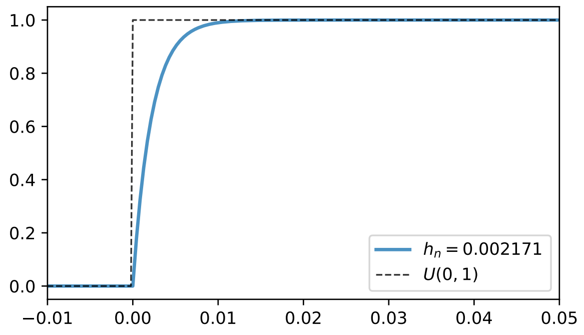

Figure 9: Plot accompanying problem 4.3d).

## Problem 4.8

a) TODO

**Problem 4.17**

We assume that $p(\omega_i) = 1/c$ and $p(\boldsymbol{x} \mid \omega_i) = p(\boldsymbol{x})$. In this case, for any point $\boldsymbol{x}$, any guess is as good as any other, and the Bayes error rate is clearly

$$P^* = \frac{c-1}{c}.$$

To prove that the bound

$$P \leq P^* \left( 2 - \frac{c}{c-1} P^* \right) \tag{5}$$

is achieved, we calculate the error rate $P$. In the following calculation, we use the fact that $p(\boldsymbol{x} \mid \omega_i) = p(\boldsymbol{x})$ and $\int p(\boldsymbol{x})\, d\boldsymbol{x} = 1$. We observe that

$$\begin{aligned}
P &= \int \left[ 1 - \sum_{i=1}^{c} P^2\left(\omega_i \mid \boldsymbol{x}\right) \right] p(\boldsymbol{x})\, d\boldsymbol{x} \\
&= \int \left[ 1 - \sum_{i=1}^{c} \left( \frac{p(\boldsymbol{x} \mid \omega_i) P(\omega_i)}{p(\boldsymbol{x})} \right)^2 \right] p(\boldsymbol{x})\, d\boldsymbol{x} \\
&= \int \left[ 1 - \sum_{i=1}^{c} P^2(\omega_i) \right] p(\boldsymbol{x})\, d\boldsymbol{x} \\
&= \left[ 1 - \sum_{i=1}^{c} \frac{1}{c^2} \right] \int p(\boldsymbol{x})\, d\boldsymbol{x} \\
&= 1 - c\frac{1}{c^2} = \frac{c-1}{c}.
\end{aligned}$$

In other words, $P^*$ and $P$ are equal. When we substitute $P^*$ and $P$ into equation (5), we see that the bound is achieved since

$$P \leq P^* \left( 2 - \frac{c}{c-1} P^* \right) \quad \Rightarrow \quad \frac{c-1}{c} \leq \frac{c-1}{c} \left( 2 - \frac{c}{c-1} \frac{c-1}{c} \right) = \frac{c-1}{c},$$

which completes the proof.

**Problem 4.27**

a) TODO
b) TODO

Table 1: Ranked order of combinations of words for problem 4.27.

| Word 1 | Word 2 | $D_{\text{Tantimoto}}$ |
|---|---|---|
| `pots` | `stop` | 0.0 |
| `pattern` | `elementary` | 0.444 |
| `pattern` | `pat` | 0.5 |
| `taxonomy` | `elementary` | 0.5 |
| `pat` | `pots` | 0.6 |
| `pat` | `stop` | 0.6 |
| `pattern` | `taxonomy` | 0.7 |
| `pattern` | `pots` | 0.75 |
| `pattern` | `stop` | 0.75 |
| `pat` | `taxonomy` | 0.75 |
| `pat` | `elementary` | 0.778 |
| `pots` | `taxonomy` | 0.778 |
| `stop` | `taxonomy` | 0.778 |
| `pots` | `elementary` | 0.909 |
| `stop` | `elementary` | 0.909 |

## 2.5 Linear discriminant functions

### Problem 5.4

a) We wish to solve the problem

$$\begin{aligned}\underset{\boldsymbol{x}}{\text{minimize}} \; & \|\boldsymbol{x}-\boldsymbol{x}_a\|^2 = \boldsymbol{x}_a^T\boldsymbol{x}_a - 2\boldsymbol{x}^T\boldsymbol{x}_a + \boldsymbol{x}^T\boldsymbol{x} \\ \text{subject to } \; & g(\boldsymbol{x}) = \boldsymbol{\omega}^T\boldsymbol{x} + \omega_0 = 0.\end{aligned}$$

To accomplish this, we start by constructing the Lagrange function

$$L(\boldsymbol{x},\lambda) = \boldsymbol{x}_a^T\boldsymbol{x}_a - 2\boldsymbol{x}^T\boldsymbol{x}_a + \boldsymbol{x}^T\boldsymbol{x} - \lambda(\boldsymbol{\omega}^T\boldsymbol{x} + \omega_0 - 0),$$

which we differentiate with respect to $\boldsymbol{x}$ and $\lambda$ to obtain:

$$\begin{aligned}L_{\boldsymbol{x}} &= -2\boldsymbol{x}_a + 2\boldsymbol{x} - \lambda\boldsymbol{\omega} = 0 \\ L_\lambda &= \boldsymbol{\omega}^T\boldsymbol{x} + \omega_0 = 0\end{aligned}$$

We wish to solve these equations for $\boldsymbol{x}$.

Solving the first equation for $\boldsymbol{x}$ yields $\boldsymbol{x} = \lambda\boldsymbol{\omega}/2 + \boldsymbol{x}_a$. This is all well and good, but we need $\lambda$ too. If we left-multiply by $\boldsymbol{\omega}^T$ and compare with the second equation, we observe that

$$\boldsymbol{\omega}^T\boldsymbol{x} = -\omega_0 \quad \text{and} \quad \boldsymbol{\omega}^T\boldsymbol{x} = \frac{\|\boldsymbol{\omega}\|^2\lambda}{2} + \boldsymbol{\omega}^T\boldsymbol{x}_a.$$

This implies that

$$-\omega_0 = \frac{\|\boldsymbol{\omega}\|^2\lambda}{2} + \boldsymbol{\omega}^T\boldsymbol{x}_a \quad \Leftrightarrow \quad \lambda = -\frac{2}{\|\boldsymbol{\omega}\|^2}\left(\omega_0 + \boldsymbol{\omega}^T\boldsymbol{x}_a\right),$$

and substituting this into $\boldsymbol{x} = \lambda\boldsymbol{\omega}/2 + \boldsymbol{x}_a$ yields the optimal answer

$$\boldsymbol{x}^* = -\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|^2}\left(\boldsymbol{\omega}^T\boldsymbol{x}_a + \omega_0\right) + \boldsymbol{x}_a = -\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|^2}g(\boldsymbol{x}_a) + \boldsymbol{x}_a. \tag{6}$$

Inserting this into $\|\boldsymbol{x}^* - \boldsymbol{x}_a\|$ yields

$$\|\boldsymbol{x}^* - \boldsymbol{x}_a\| = \left\|\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|^2}g(\boldsymbol{x}_a)\right\| = \frac{|g(\boldsymbol{x}_a)|}{\|\boldsymbol{\omega}\|}.$$

b) The projection onto the plane is the minimizer $\boldsymbol{x}^*$ from equation (6) in the previous problem, so we immediately see that

$$\boldsymbol{x}^* = -\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|^2}\left(\boldsymbol{\omega}^T\boldsymbol{x}_a + \omega_0\right) + \boldsymbol{x}_a = -\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|^2}g(\boldsymbol{x}_a) + \boldsymbol{x}_a = \boldsymbol{x}_a - \frac{g(\boldsymbol{x}_a)}{\|\boldsymbol{\omega}\|^2}\boldsymbol{\omega},$$

which is what we were required to show.

**Problem 5.13**

We wish to choose $\eta(k)$ to minimize the quadratic function

$$J\left(\boldsymbol{a}(k+1)\right) \simeq J\left(\boldsymbol{a}(k)\right) - \eta(k)\left\|\nabla \boldsymbol{J}\right\|^2 + \frac{1}{2}\eta^2(k)\nabla \boldsymbol{J}^T \boldsymbol{H} \nabla \boldsymbol{J}.$$

Differentiating and setting equal to zero yields

$$\frac{\partial J\left(\boldsymbol{a}(k+1)\right)}{\partial \eta(k)} = -\left\|\nabla \boldsymbol{J}\right\|^2 + \eta(k)\nabla \boldsymbol{J}^T \boldsymbol{H} \nabla \boldsymbol{J} = 0,$$

and solving this for $\eta(k)$ yields the desired answer

$$\eta(k) = \frac{\left\|\nabla \boldsymbol{J}\right\|^2}{\nabla \boldsymbol{J}^T \boldsymbol{H} \nabla \boldsymbol{J}}.$$

**Problem 5.15**

If $\alpha > \beta^2/2\gamma$, then $-2\alpha\gamma + \beta^2 < 0$ and

$$\left\|\boldsymbol{a}(k+1) - \alpha\hat{\boldsymbol{a}}\right\|^2 \leq \left\|\boldsymbol{a}(k) - \alpha\hat{\boldsymbol{a}}\right\|^2 - 2\alpha\gamma + \beta^2$$

represents error which decreases at each step. After $k$ corrections we obtain

$$\left\|\boldsymbol{a}(k+1) - \alpha\hat{\boldsymbol{a}}\right\|^2 \leq \left\|\boldsymbol{a}(1) - \alpha\hat{\boldsymbol{a}}\right\|^2 + k(-2\alpha\gamma + \beta^2),$$

and the error is zero when $\left\|\boldsymbol{a}(1) - \alpha\hat{\boldsymbol{a}}\right\|^2 + k_0(-2\alpha\gamma + \beta^2) = 0$, which implies that

$$k_0 = \frac{\left\|\boldsymbol{a}(1) - \alpha\hat{\boldsymbol{a}}\right\|^2}{2\alpha\gamma - \beta^2}.$$

This is what we were asked to show.

**Problem 5.21**

To ease the notation, let us write $\boldsymbol{m} := \boldsymbol{m}_1 - \boldsymbol{m}_2$[1]. Staring with equation (53) from the book, we left-multiply by the bracketed term to isolate $\boldsymbol{w}$ as

$$\boldsymbol{w} = \left[\frac{1}{n}\boldsymbol{S}_W + \frac{n_1 n_2}{n^2}\boldsymbol{m}\boldsymbol{m}^T\right]^{-1}\boldsymbol{m}. \tag{7}$$

Recall from problem 3.36 that the Sherman-Morrison-Woodbury matrix identity is

$$\left(\boldsymbol{A} + \boldsymbol{x}\boldsymbol{y}^T\right)^{-1} = \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{x}\boldsymbol{y}^t\boldsymbol{A}^{-1}}{1 + \boldsymbol{y}^T\boldsymbol{A}^{-1}\boldsymbol{x}}.$$

[1]This definition of $\boldsymbol{m}$ is not the same as the one used in the book, where $\boldsymbol{m}$ is the grand mean.

We now apply the identity to the bracketed term in equation (7). In doing so, we identify

$$\boldsymbol{A} \cong \frac{1}{n}\boldsymbol{S}_W \qquad \text{and} \qquad \boldsymbol{x}\boldsymbol{y}^T \cong \frac{n_1 n_2}{n^2}\boldsymbol{m}\boldsymbol{m}^T.$$

Applying the matrix identity, we obtain

$$\begin{aligned}\boldsymbol{w} &= \left[ n\boldsymbol{S}_W^{-1} - \frac{n^2\left(\frac{n_1n_2}{n^2}\right)\boldsymbol{S}_W^{-1}\boldsymbol{m}\boldsymbol{m}^T\boldsymbol{S}_W^{-1}}{1 + n\left(\frac{n_1n_2}{n^2}\right)\boldsymbol{m}^T\boldsymbol{S}_W^{-1}\boldsymbol{m}} \right]\boldsymbol{m} \\ &= n\boldsymbol{S}_W^{-1}\boldsymbol{m} - \frac{n\left(\frac{n_1n_2}{n}\right)\boldsymbol{S}_W^{-1}\boldsymbol{m}\boldsymbol{m}^T\boldsymbol{S}_W^{-1}\boldsymbol{m}}{1 + \left(\frac{n_1n_2}{n}\right)\boldsymbol{m}^T\boldsymbol{S}_W^{-1}\boldsymbol{m}}.\end{aligned}$$

To simplify the notation and remind ourselves that some of these quantities are mere scalars, let us denote $a := n_1n_2/n$ and $b := \boldsymbol{m}^T\boldsymbol{S}_W^{-1}\boldsymbol{m}$. We simplify and factor our $n\boldsymbol{S}_W^{-1}\boldsymbol{m}$ to obtain

$$\boldsymbol{w} = n\boldsymbol{S}_W^{-1}\boldsymbol{m} - \frac{na\boldsymbol{S}_W^{-1}\boldsymbol{m}b}{1+ab} = n\boldsymbol{S}_W^{-1}\boldsymbol{m}\left[1 - \frac{ab}{1+ab}\right] = n\boldsymbol{S}_W^{-1}\boldsymbol{m}\left[1+ab\right]^{-1}.$$

Recalling now that $a := n_1n_2/n$ and $b := \boldsymbol{m}^T\boldsymbol{S}_W^{-1}\boldsymbol{m}$, we have accomplished the goal. The result is that

$$\boldsymbol{w} = n\boldsymbol{S}_W^{-1}\boldsymbol{m}\alpha = n\boldsymbol{S}_W^{-1}\boldsymbol{m}\big[1 + \underbrace{\left(\frac{n_1n_2}{n}\right)}_{a}\underbrace{\boldsymbol{m}^T\boldsymbol{S}_W^{-1}\boldsymbol{m}}_{b}\big]^{-1},$$

which shows that $\alpha$ is indeed given by the quantity in the problem statement.

**Problem 5.25**

a) We supply a proof by induction: we first show the base case, and then the inductive step.

   **The base case** is verified by checking that the relation holds for $\eta^{-1}(2) = \eta^{-1}(1) + y_1^2$. This is true, since it implies

   $$\eta(2) = \frac{1}{\eta^{-1}(1) + y_1^2} = \frac{\eta(1)}{1 + \eta(1)y_1^2} = \frac{\eta(1)}{1 + \eta(1)\sum_{i=1}^{1} y_i^2},$$

   which is clearly the given formula for $k = 2$.

   In **the inductive step** we assume that the relation holds for $\eta(k)$, and show that this implies that it holds for $\eta(k+1)$ too. The required algebra is

   $$\begin{aligned}\eta^{-1}(k+1) &= \eta^{-1}(k) + y_k^2 \\ &= \left(\frac{\eta(1)}{1 + \eta(1)\sum_{i=1}^{k-1} y_i^2}\right)^{-1} + y_k^2 \\ &= \frac{1 + \eta(1)\sum_{i=1}^{k-1} y_i^2}{\eta(1)} + \frac{\eta(1)}{\eta(1)}y_k^2 \\ &= \frac{1 + \eta(1)\sum_{i=1}^{k} y_i^2}{\eta(1)}.\end{aligned}$$

Inverting this shows that $\eta(k+1) = \eta(1)/\left(1+\eta(1)\sum_{i=1}^{k} y_i^2\right)$, as required.

b) To show why the sequence of coefficients will satisfy the sums, we will first bound the terms, and the convert the problems to integrals.

If $0 < a \le y_i^2 \le b < \infty$ for every $i$, then the sum is bounded by

$$a(k-1) \le \sum_{i=1}^{k-1} y_i^2 \le b(k-1),$$

and this in turn implies the expression $\eta(k)$ that may be bounded by

$$\frac{\eta(1)}{1+\eta(1)b(k-1)} \le \eta(k) \le \frac{\eta(1)}{1+\eta(1)a(k-1)}.$$

To show that $\sum \eta(k) \to \infty$, we note that $\sum \eta(k) \simeq \lim_{\alpha\to\infty} \int_{x=1}^{x=\alpha} \eta(x)\,dx$. We observe that the integral

$$\lim_{\alpha\to\infty} \int_{x=1}^{x=\alpha} \frac{\eta(1)}{1+\eta(1)(x-1)b}\,dx = \lim_{\alpha\to\infty} \frac{1}{b} \ln|u| \Big|_{u=1}^{u=1+\eta(1)(\alpha-1)b}$$

diverges for any value of $b$, where we used the substitution $u = 1+\eta(1)(x-1)b$. Since $b$ represents the maximal value of the terms $\eta(k)$, any other value of $y_i^2$ will diverge too, and the sum $\sum \eta(k)$ diverges to infinity.

To show that $\sum \eta^2(k) \to L < \infty$, we again use $\sum \eta^2(k) \simeq \lim_{\alpha\to\infty} \int_{x=1}^{x=\alpha} \eta^2(x)\,dx$. Now the integral converges, since for any value of $a$ the integral

$$\lim_{\alpha\to\infty} \int_{x=1}^{x=\alpha} \frac{\eta^2(1)}{(1+\eta(1)(x-1)b)^2}\,dx = \lim_{\alpha\to\infty} \frac{\eta(1)}{bu} \Big|_{u=1+\eta(1)(\alpha-1)a}^{u=1} \le \frac{\eta(1)}{b}$$

diverges, and $a$ represents the maximal bound on $\eta(k)$. The sum $\sum \eta(k)$ diverges to infinity, for any $0 < a \le y_i^2 \le b < \infty$.

## Problem 5.27

a) The data points are plotted in figure 10 on page 40, and as seen they are not linearly separable.

b) From equation (95) we observe that the optimal choice of $\eta$ is given by

$$\eta(k) = \frac{\left\|\boldsymbol{Y}^T |\boldsymbol{e}(k)|\right\|^2}{\left\|\boldsymbol{Y}\boldsymbol{Y}^T |\boldsymbol{e}(k)|\right\|^2} = \frac{\boldsymbol{e}^T \boldsymbol{Y}\boldsymbol{Y}^T \boldsymbol{e}}{\boldsymbol{e}^T \boldsymbol{Y}\boldsymbol{Y}^T \boldsymbol{Y}\boldsymbol{Y}^T \boldsymbol{e}}.$$

This value varies from loop to loop, depending on $\boldsymbol{e}$. For this specific data set, the value of $\boldsymbol{Y}$ and $\boldsymbol{Y}\boldsymbol{Y}^T$ are given in equation (b)).

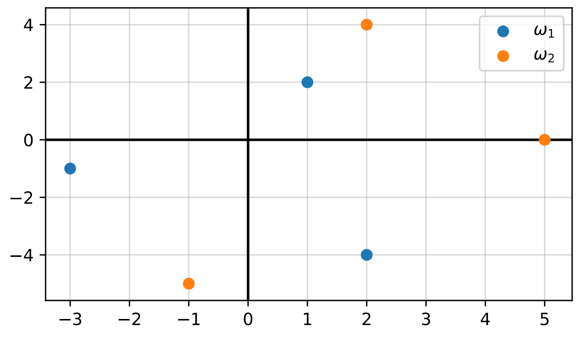

Figure 10: Graf accompanying problem 5.27.

$$
\boldsymbol{Y} = \begin{pmatrix} 1.0 & 1.0 & 2.0 \\ 1.0 & 2.0 & -4.0 \\ 1.0 & -3.0 & -1.0 \\ -1.0 & -2.0 & -4.0 \\ -1.0 & 1.0 & 5.0 \\ -1.0 & -5.0 & 0.0 \end{pmatrix}
$$

$$
\boldsymbol{Y}\boldsymbol{Y}^T = \begin{pmatrix} 6.0 & -5.0 & -4.0 & -11.0 & 10.0 & -6.0 \\ -5.0 & 21.0 & -1.0 & 11.0 & -19.0 & -11.0 \\ -4.0 & -1.0 & 11.0 & 9.0 & -9.0 & 14.0 \\ -11.0 & 11.0 & 9.0 & 21.0 & -21.0 & 11.0 \\ 10.0 & -19.0 & -9.0 & -21.0 & 27.0 & -4.0 \\ -6.0 & -11.0 & 14.0 & 11.0 & -4.0 & 26.0 \end{pmatrix}
$$

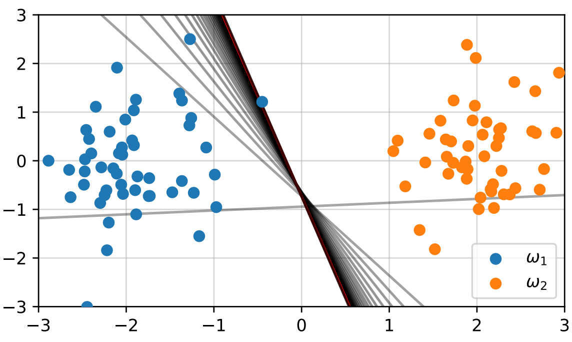

Figure 11: Convergence of the Ho Kashyap algorithm.

**Problem 5.29**

To show that there always exists a mapping to a higher dimensional space which leaves points from two classes linearly separable, we will explicitly provide such a mapping. The mapping would be very inefficient in practice, but provides a constructive proof.

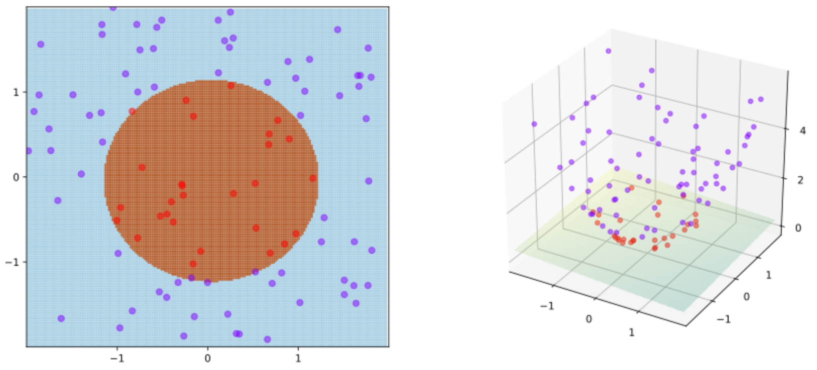

Figure 12: Using a kernel to raise points in a new dimension. Source: `https://commons.wikimedia.org/wiki/File:Kernel_trick_idea.svg`

Observe first that to apply a linear classifier to points where data from one class is close to the origin, a function such as $y = \phi(\boldsymbol{x}) = \exp(-\boldsymbol{x}^T\boldsymbol{x})$ may be used. This leave the data linearly separable in the new space, as illustrated in figure 12. Below we will extend this idea by introducing many distinct mappings $\phi(\boldsymbol{x})$.

Consider now points $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ in $\mathbb{R}^d$. Assume that some points $S \subseteq \mathcal{D}$ belong to $\omega_1$, and that we know exactly which points. If we know which points belong to $\omega_1$, then a kernel density estimate (Parzen window) such as

$$y = \text{Parzen}(S) = \frac{1}{|S|}\sum_{\boldsymbol{x}_i \in S} \frac{1}{h}\phi\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right)$$

with a sufficiently small value of the bandwidth $h$ will raise these points in the new $y$ feature space. When $h \ll 1$, neighboring points not belonging to $\omega_1$ will be unaffected. A plane such as $y = 0.1$ will then perfectly separate the points.

Since we do not know the true subset $S \subseteq \mathcal{D}$, we can apply this transformation on *every* subset of $\mathcal{D}$ (the *power set* of $\mathcal{D}$). There are $2^{|S|}$ such subsets. We use

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x} & \text{Parzen}(S_1) & \text{Parzen}(S_1) & \ldots & \text{Parzen}(S_{2^{|S|}}) \end{pmatrix}$$

to map $\boldsymbol{x} \in \mathbb{R}^d$ to a $d + 2^{|S|}$ space. In other words: if $\{\boldsymbol{x_1}\}$ is “raised” in one new feature dimension, $\{\boldsymbol{x_1}, \boldsymbol{x_2}\}$ in another, $\{\boldsymbol{x_1}, \boldsymbol{x_3}\}$ in yet another, etc. for *every* combination of points, then in some dimension in the feature space the points are linearly separable.

## Problem 5.32

a) The plot is show in figure 13. By inspection the weight vector is

$$\boldsymbol{a} = (a_0, a_1, a_2) = (-1.5, 1, 1).$$

This corresponds to the line $y = 1.5 - x$. The optimal margin is the distance from the line $y = 1.5 - x$ to a point, say $(1, 1)$, and this distance is $\sqrt{2}/4 \approx 0.3536$.

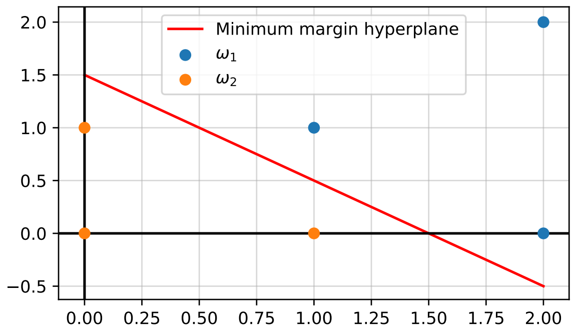

Figure 13: Plot related to problem 5.32a).

b) From inspection there are four support vectors, they are

$$\begin{matrix} (0,1)^T & (1,0)^T \\ (1,1)^T & (2,0)^T. \end{matrix}$$

c) This laborious computation is omitted.

## Problem 5.33

a) The optimization problem

$$L(\boldsymbol{a}, \boldsymbol{\alpha}) = \frac{1}{2} \|\boldsymbol{a}\|^2 - \sum_{k=1}^{n} \alpha_k \left[ z_k \boldsymbol{a}^T \boldsymbol{y}_k - 1 \right] \tag{8}$$

has a solution which is a saddle point since we wish to maximize with respect to $\boldsymbol{\alpha}$ and minimize with respect to $\boldsymbol{a}$.

b) Here I believe there to be a slight error in the text, which might lead to confusion. In the initial pages of the chapter, the distance from a point to a hyperplane is given by $r = (\boldsymbol{\omega}^T \boldsymbol{x} + \omega_0) / \|\boldsymbol{\omega}\|$, and this is indeed correct.

In the context of SVMs, however, the distance is said to be

$$\frac{g(y)}{\|\boldsymbol{a}\|} = \frac{\boldsymbol{a}^T \boldsymbol{y}}{\|\boldsymbol{a}\|} = \frac{(\omega_0, \boldsymbol{a}')(1, \boldsymbol{y}')^T}{\|\boldsymbol{a}\|} = \frac{\boldsymbol{a}'^T \boldsymbol{y} + \omega_0}{\|\boldsymbol{a}\|},$$

which is different because $\boldsymbol{a}$ is in the denominator, not $\boldsymbol{a}'$. Amending equation (8) with this information, the $\frac{1}{2}\|\boldsymbol{a}\|^2$ should be replaced by $\frac{1}{2}\|\boldsymbol{a}'\|^2$, i.e. dropping $\boldsymbol{a}_0 = \omega_0$. If we do this and differentiate with respect to the first component of $\boldsymbol{a}$, we obtain

$$\frac{\partial L(\boldsymbol{a}, \boldsymbol{\alpha})}{\partial \boldsymbol{a}_0} = \sum_k \alpha_k^* z_k \boldsymbol{y}_0 = 0,$$

which gives the desired result since $\boldsymbol{y}_0 = 1$ due to the augmentation of the vector.

c) To prove this, we differentiate with respect to $\boldsymbol{a}$ and obtain

$$\frac{\partial L(\boldsymbol{a}, \boldsymbol{\alpha})}{\partial \boldsymbol{a}} = \boldsymbol{a}^* - \sum_k \alpha_k^* z_k \boldsymbol{y}_k = 0.$$

d) If the Lagrange multiplier (or *undetermined multiplier*) $\alpha_k^*$ is zero, then it's said to be *inactive*. At the optimum, the constraint is not used. The optimum of $L(\boldsymbol{a}, \boldsymbol{\alpha})$ is the same with or without this constraint.

If the Lagrange multiplier $\alpha_k^*$ is non-zero, then it's said to be *active*. The constrained solution is different from the unconstrained solution, and the optimum lies on the boundary of the constraint. Since $\alpha_k^* z_k \boldsymbol{y}_k \geq 1$ in the feasible region, the optimal solution is on the boundary if $\alpha_k^* z_k \boldsymbol{y}_k = 1$, but then $\alpha_k^*$ is non-zero since the constraint is active.

In conclusion, either $\alpha_k^* z_k \boldsymbol{y}_k = 1$ if the constraint is active, or $\alpha_k^* = 0$ if the constraint is inactive. This is one of the Karush–Kuhn–Tucker (KKT) conditions, and it may be expressed as

$$\alpha_k^* \left[\alpha_k^* z_k \boldsymbol{y}_k - 1\right] = 0 \qquad k = 1, \ldots, n.$$

e) Simply multiply the brackets in equation (8) from subproblem a).

$$L(\boldsymbol{a}, \boldsymbol{\alpha}) = \frac{1}{2} \|\boldsymbol{a}\|^2 - \sum_{k=1}^n \alpha_k \left[z_k \boldsymbol{a}^T \boldsymbol{y}_k - 1\right] = \frac{1}{2} \|\boldsymbol{a}\|^2 - \sum_{k=1}^n \alpha_k z_k \boldsymbol{a}^T \boldsymbol{y}_k + \sum_{k=1}^n \alpha_k$$

f) Using $\boldsymbol{a}^* = \sum_j \alpha_j^* z_j \boldsymbol{y}_j$ we observe that

$$\begin{aligned} L(\boldsymbol{\alpha}) &= \frac{1}{2} \|\boldsymbol{a}^*\|^2 - \sum_{k=1}^n \alpha_k z_k \boldsymbol{a}^{*T} \boldsymbol{y}_k + \sum_{k=1}^n \alpha_k \\ &= \frac{1}{2} \left(\sum_j \alpha_j^* z_j \boldsymbol{y}_j^T\right) \left(\sum_k \alpha_k^* z_k \boldsymbol{y}_k\right) - \sum_{k=1}^n \alpha_k z_k \left(\sum_j \alpha_j^* z_j \boldsymbol{y}_j^T\right) \boldsymbol{y}_k + \sum_{k=1}^n \alpha_k \end{aligned}$$

and since the first and second terms are equal, we obtain

$$-\frac{1}{2} \left(\sum_j \alpha_j^* z_j \boldsymbol{y}_j^T\right) \left(\sum_k \alpha_k^* z_k \boldsymbol{y}_k\right) + \sum_{k=1}^n \alpha_k = -\frac{1}{2} \sum_j \sum_k \alpha_j^* \alpha_k^* z_j z_k \boldsymbol{y}_j^T \boldsymbol{y}_k + \sum_{k=1}^n \alpha_k$$

as desired. We have formulated the problem as a maximization over $L(\boldsymbol{\alpha})$.

## 2.6 Multilayer Neural Networks

### Problem 6.5

The backpropagation rule for input-to-hidden weights is given by equation (21), which is

$$\Delta w_{ji} = \eta \delta_j x_i = \eta \left[ \sum_{k=1}^{c} w_{kj} \delta_k \right] f'(\text{net}_j) x_i.$$

Notice that if $z_k = t_k$ for every value of $k$, there is nothing to learn since the error is zero. In this case, $\delta_k = 0$ and as a result $\Delta w_{ji} = 0$. The learning rate is proportional to the error $(t_k - z_k)$, but it's also weighted by $w_{kj}$. The higher this weight is, the more an error contributes and the further it is adjusted in the backpropagation rule.

### Problem 6.8

a) In this problem it helps to make a drawing, or study Figure 6.4 or Figure 6.2.
   (i) The bias is connected to $n_H + c$ weights.
   (ii) Every one of the $d$ inputs is connected to every one of the $n_H$ hidden units, for a total of $dn_H$ weights.
   (iii) Every hidden unit $n_H$ is connected to every one of the $c$ ouputs, for $n_H c$ weights.

   The total number of weights is therefore given by

$$n_H + c + dn_H + n_H c = n_H(1 + d + c) + c.$$

b) Consider the equation

$$z_k = f \left[ \sum_{j=1}^{n_H} w_{kj} \, f \left( \sum_{i=1}^{d} w_{ji} x_i + w_{j0} \right) + w_{k0} \right]$$

   for a neural network output. If the sign is flipped on every weight going into a hidden unit, and the weights leading out from that same unit are also flipped, then the net result is no change if the activation function obeys $f(-x) = -f(x)$. In other words, if the $w_{ji} \mapsto -w_{ji}$ and $w_{j0} \mapsto -w_{j0}$ in the equation above, then

$$f \left( \sum_{i=1}^{d} -w_{ji} x_i - w_{j0} \right) = -f \left( \sum_{i=1}^{d} w_{ji} x_i + w_{j0} \right)$$

   and a mapping $w_{kj} \mapsto -w_{kj}$ in the outer sum will cancel the sign flip. This only applies to odd functions where $f(-x) = -f(x)$.

c) If there are $n_H$ hidden units, there are $n_H!$ ways to permute them. For each of these permutations, every associated weight might have it's sign flipped or not, for a total of $2^{n_H}$ possibilities with respect to sign flipping. The total count is therefore $n_H! 2^{n_H}$ exchange symmetries (not $n_H 2^{n_H}$ as [Duda et al., 2000] claims). This result is verified in [Bishop, 2011] on page 232, so we assume that [Duda et al., 2000] contains a typo here.

**Problem 6.10**

a) The rules of differentiation in calculus shows that when $f(x) = 1/(1+e^{ax})$, then the derivative may be expressed in terms of the function as

$$f'(x) = -ae^{ax}f^2(x).$$

b) We'll study $f(x) = a(1-e^{-2bx})/(1+e^{-2bx})$, and it pays off to ease notation somewhat. Letting $g(x) := e^{-2bx}$, we have $g'(x) = -2bg(x)$ and we expedite notation by writing

$$f(x) = a\frac{1-e^{-2bx}}{1+e^{-2bx}} := a\frac{1-g(x)}{1+g(x)} = a(1-g(x))(1+g(x))^{-1}.$$

Differentiating this and using the chain rule and product rule of calculus, we obtain

$$\begin{aligned} f'(x) &= \frac{-ag'(x)(1+g(x)) - ag'(x)(1-g(x))}{(1+g(x))^2} \\ &= \frac{4abg(x)}{(1+g(x))^2} = \underbrace{a\frac{1-g(x)}{1+g(x)}}_{f'(x)} \frac{4bg(x)}{(1-g(x))(1+g(x))}. \end{aligned}$$

Substituting back $g(x) := e^{-2bx}$, we see that the derivative may be expressed in terms of the function as

$$f'(x) = f(x)\frac{4be^{-2bx}}{1-e^{-4bx}}.$$

**Problem 6.16**

a) Without loss of generality we assume that $J(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{H}\boldsymbol{w}$, since by a transformation of the type $\boldsymbol{w} = \boldsymbol{w'} - \boldsymbol{c}$ we could transform a general, non-centered quadratic equation to this form. This is analogous to scaling $f(x) = ax^2 + bx + x$ in order to formulate it as $f(x') = a'x'^2$.

To study convergence, we examine the learning rule, which may be written as

$$\begin{aligned} \boldsymbol{w}_{n+1} &= \boldsymbol{w}_n - \eta\nabla J(\boldsymbol{w}_n) \\ &= \boldsymbol{w}_n - \eta\boldsymbol{H}\boldsymbol{w}_n \\ &= (\boldsymbol{I} - \eta\boldsymbol{H})\,\boldsymbol{w}_n, \end{aligned}$$

and observe that $\boldsymbol{w}_n = (\boldsymbol{I} - \eta\boldsymbol{H})^n\,\boldsymbol{w}_0$. Convergence is ensured if $(\boldsymbol{I} - \eta\boldsymbol{H})^n$ goes to zero as $n \to \infty$. Informally, repeated application of the matrix should bring the vector to zero. Problems relate to "repeated application of a matrix to a vector" is controlled by eigenvectors and eigenvalues.

We write $(\boldsymbol{I} - \eta\boldsymbol{H}) = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$ to express the symmetric, positive definite matrix in it's eigenvalue decomposition. Then we define $\boldsymbol{w}'_k = \boldsymbol{Q}\boldsymbol{w}_k = \sum_{i=1}^d \alpha_i\boldsymbol{v}_i$ to express $\boldsymbol{w}_k$ in the *eigenvalue basis* of $\boldsymbol{H}$, where the $\boldsymbol{v}_i$ are the orthonormal eigenvectors. This transformation is a rotation of the space, since $\boldsymbol{Q}$ is orthonormal.

The expression on the left below is for the vector, and the expression on the right captures the behavior of a single eigenvector component $\boldsymbol{v}_i$.

$$\left(\sum_{i=1}^{d} \alpha_i \boldsymbol{v}_i\right)_{n+1} = \boldsymbol{Q}\boldsymbol{\Lambda}\left(\sum_{i=1}^{d} \alpha_i \boldsymbol{v}_i\right)_{n} \quad \Leftrightarrow \quad (\alpha_i \boldsymbol{v}_i)_{n+1} = \boldsymbol{Q}\lambda_i(\alpha_i \boldsymbol{v}_i)_{n+1}$$

The matrix $\boldsymbol{Q}$ is orthogonal, so it merely rotates or reflects a vector, but $\|\boldsymbol{v}_i\| = \|\boldsymbol{Q}\boldsymbol{v}_i\|$ so the magnitude is unchanged. Therefore we ignore it when considering convergence, and see that learning is ensured if $|\lambda_i| < 1$ for every $i = 1, 2, \ldots, d$. Since every eigenvalue is positive, the maximal eigenvalue controls convergence.

The only thing left to do is to relate the eigenvalues of $(\boldsymbol{I} - \eta\boldsymbol{H})$ to the eigenvalues of $\boldsymbol{H}$. If a matrix is multiplied by a scalar, then so are it's eigenvalues. The effect of subtraction by the identity matrix is not as obvious, but if we write out the equation for the characteristic polynomials we observe that

$$\begin{aligned} \text{eig. vals of } (\boldsymbol{I} - \eta\boldsymbol{H}) &\Leftrightarrow \text{ zeros of } \det(\boldsymbol{I} - \eta\boldsymbol{H} - \lambda\boldsymbol{I}) = \det(\boldsymbol{I}(1-\lambda) - \eta\boldsymbol{H}) \\ \text{eig. vals of } \eta\boldsymbol{H} &\Leftrightarrow \text{ zeros of } \det(\boldsymbol{I}\lambda' - \eta\boldsymbol{H}), \end{aligned}$$

and comparing we observe that $\lambda' = \lambda - 1$. We're really after the eigenvalues of of $\boldsymbol{H}$, which we'll now denote by $\lambda_{\boldsymbol{H}} = \lambda'/\eta$. From this, we have convergence if

$$\begin{aligned} -1 <& \lambda_{\max} < 1 \\ -1 <& 1 - \eta\lambda_{\boldsymbol{H}\max} < 1 \\ \eta\lambda_{\boldsymbol{H}\max} &< 2, \end{aligned}$$

which is the same as requiring that $\eta < \lambda_{\boldsymbol{H}\max}/2$.

b) The best learning rate is achieved if we manage to choose $\eta$ such that the maximal eigenvalue (in absolute value) of $(\boldsymbol{I} - \eta\boldsymbol{H})$ is close to zero. The optimal rate is given by[2]

$$\operatorname{rate}(\eta) = \max_i |1 - \eta\lambda_i| = \max(|1 - \eta\lambda_1|, |1 - \eta\lambda_d|)$$

where $\lambda_1$ is the smallest eigenvalue of $\boldsymbol{H}$ and $\lambda_d$ is the largest. Convergence is fastest when these arguments are equal in absolute value, so we require that

$$(1 - \eta\lambda_1) = -(1 - \eta\lambda_d)$$

and solve for $\eta$ to obtain $\eta^* = 2/(\lambda_1 + \lambda_d)$. The optimal learning rate is obtained by substituting this $\eta^*$ into rate$(\cdot)$. We consider the positive value $(1 - \eta^*\lambda_1)$, which becomes

$$\operatorname{rate}(\eta^*) = (1 - \eta^*\lambda_1) = \frac{\lambda_d - \lambda_1}{\lambda_d + \lambda_1} = \frac{\lambda_d/\lambda_1 - 1}{\lambda_d/\lambda_1 + 1},$$

and recognize that the learning rate is indeed dependent on the ratio of the largest to the smallest eigenvalue of $\boldsymbol{H}$.

c) An informal argument is as follows: Consider unstandardized data which is not highly correlated (i.e. the covariance matrix is nearly diagonal), but where data is

[2]See `https://distill.pub/2017/momentum/` for details.

from different dimensions and is on vastly different scales (i.e. the diagonal of the covariance matrix has entries on different scales). Assume that this data is used to train a neural network, and that the weights are initialized to sensible, small, random values. Changing one weight value $w_1$ might then potentially change the error drastically compared to changing another weight $w_2$, and $J(\boldsymbol{w}) \approx \frac{1}{2}\boldsymbol{w}^T \boldsymbol{H} \boldsymbol{w}$ will have diagonals on vastly different scales. The eigenvalues are $\lambda_1 \approx \min \operatorname{diag} \boldsymbol{H}$ and $\lambda_d \approx \max \operatorname{diag} \boldsymbol{H}$, so learning will be slow since $\operatorname{rate}(\eta^*)$ will be close to 1.

Standardizing reduces these problems. The data is on the same scale, so the diagonals entries of $\boldsymbol{H}$ will likely be on the same scale and learning will be faster.

d) Standardizing the data by subtracting means and diving by standard deviations in every dimension individually transforms the data to the same scale. The data might still be highly correlated. The whitening transform consists of scaling *and* rotating the data. The rotation diagonalizes the covariance matrix, and the scaling makes the spectrum of eigenvalues uniform, so that the variances are all unity.

**Problem 6.21**

a) The softmax function is given by equation (30) in the book, which is

$$z_k = f(\mathrm{net}_k) = \frac{e^{\mathrm{net}_k}}{\sum_{m=1}^{c} e^{\mathrm{net}_m}} \propto e^{\mathrm{net}_k}.$$

The learning rule when the sum squared error function is used becomes dependent on $\frac{\partial J}{\partial \mathrm{net}_k}$ and $\frac{\partial \mathrm{net}_k}{\partial w_{kj}}$, see equation (13) in the book. The only change is that now $f(\mathrm{net}_k)$ is not a sigmoid, but the softmax function. To ease notation, we define $k := \mathrm{net}_k$, and differentiate.

$$\frac{\partial f(k)}{\partial k} = e^k \left[ \sum_{m=1}^{c} e^m \right]^{-1} - e^k \left[ \sum_{m=1}^{c} e^m \right]^{-2} e^k = \frac{e^k \left[ \sum_{\substack{m=1 \\ m \neq k}}^{c} e^m \right]}{\left[ \sum_{m=1}^{c} e^m \right]^2}$$

The update rules becomes a simple modification of equation (17) from the book:

$$\Delta w_{kj} = \eta \delta_k y_j = \eta (t_k - z_k) \frac{e^{\mathrm{net}_k} \left[ \sum_{\substack{m=1 \\ m \neq k}}^{c} e^{\mathrm{net}_m} \right]}{\left[ \sum_{m=1}^{c} e^{\mathrm{net}_m} \right]^2} y_j.$$

Extending the result to $\Delta w_{ji}$ is straightforward, since $\Delta w_{ji}$ a function of $\delta_k$ above.

b) When employing cross entropy, the learning rule is identical to the sum squared error above, save for $\frac{\partial J}{\partial z_k}$, which now becomes

$$\frac{\partial J(\boldsymbol{w})}{\partial z_k} = \frac{\partial}{\partial z_k} \left[ \sum_{k=1}^{c} t_k \ln \frac{t_k}{z_k} \right] = t_k \frac{z_k}{t_k} \frac{\partial}{\partial z_k} \left( t_k z_k^{-1} \right) = -\frac{t_k}{z_k}.$$

Everything else is exactly equal to the solution to a) above.

**Problem 6.24**

We compare equation (7) for the 3-layer neural network with equation (32) for general additive models (GAM). The equations are, respectively, given by

$$z_k = f\left[\sum_{j=1}^{n_H} w_{kj}\, f\left(\sum_{i=1}^{d} w_{ji}x_i + w_{j0}\right) + w_{k0}\right] \quad \text{and}$$
$$z = f\left[\sum_{j=1}^{d} w_{kj} f_i\left(\boldsymbol{x}_i\right) + w_{k0}\right].$$

The functions $f_i(\boldsymbol{x}_i)$ in GAM may in general be multivariate. According to Wikipedia "The GAM model class is quite broad, given that *smooth function* is a rather broad category. For example, a covariate $\boldsymbol{x}_i$ may be multivariate and the corresponding $f_i$ a smooth function of several variables, or $f_i$ might be the function mapping the level of a factor to the value of a random effect."

Comparing the equations, we observe that

$$\sum_{j=1}^{d} w_{kj} f_i\left(\boldsymbol{x}_i\right) \cong \sum_{j=1}^{n_H} w_{kj}\, f\left(\sum_{i=1}^{d} w_{ji}x_i + w_{j0}\right).$$

In the typical case, the $f$ on the right hand side a sigmoid function. The right hand side is a weighted sum of sigmoids, where the input to the sigmoids are again a linear function of the neural network inputs. The number of terms in the sum has no significance in the left-hand sum, since a function $f_i(\boldsymbol{x})$ could be defined as $f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$. The *Kolmogorov-Arnold representation theorem* guarantees that both of these constructions can, in theory, approximate any continuous function.

**Problem 6.39**

a) If we write out the sums, we obtain

$$f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{K}\boldsymbol{x} = \sum_i\sum_j x_i K_{ij} x_j.$$

Of course, the function $f(\cdot)$ above is a mapping from a vector $\boldsymbol{x} \in \mathbb{R}^d$ to a real number $\mathbb{R}$. In this setting, the derivative is the gradient, i.e. $\nabla f(\boldsymbol{x}) = f'(\boldsymbol{x})$. To find the gradient using explicit components, we write out the sums as

$$\frac{d}{dx_k}\left(\boldsymbol{x}^T\boldsymbol{K}\boldsymbol{x}\right) =$$
$$\frac{d}{dx_k}\left(\sum_i x_i \sum_j K_{ij}x_j\right) =$$
$$\frac{d}{dx_k}\left(x_1\left(\sum_j K_{1j}x_j\right) + x_2\left(\sum_j K_{2j}x_j\right) + \cdots + x_k\left(\sum_j K_{kj}x_j\right) + \dots\right). \tag{9}$$

For every term $i \neq k$ , the derivative is $x_i K_{ik}$ since every other term in the sums vanish. On the $k$'th term, we apply the product rule of differentiation to obtain

$$\frac{d}{dx_k}\left(x_k\left(\sum_j K_{kj}x_j\right)\right) = \sum_j K_{kj}x_j + x_k\left(K_{kk}\right).$$

Applying these results to the non-$k$'th terms and the $k$'th term respectively, equation (9) may be written in a more readable form as

$$\begin{aligned}\frac{d}{dx_k}\left(\boldsymbol{x}^T\boldsymbol{K}\boldsymbol{x}\right) &= x_1K_{1k} + x_2K_{2k} + x_3K_{3k} + \cdots + \left(\sum_j K_{kj}x_j + x_kK_{kk}\right) + \ldots \\ &= \sum_i x_iK_{ik} + \sum_j K_{kj}x_j = \boldsymbol{K}^T\boldsymbol{x} + \boldsymbol{K}\boldsymbol{x} = \left(\boldsymbol{K}^T + \boldsymbol{K}\right)\boldsymbol{x},\end{aligned}$$

where the last equality follows from

$$\begin{aligned}\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} \quad &\Leftrightarrow \quad b_i = \sum_j A_{ij}x_j \\ \boldsymbol{b} = \boldsymbol{A}^T\boldsymbol{x} \quad &\Leftrightarrow \quad b_i = \sum_j A_{ji}x_j.\end{aligned}$$

b) Here's an approach which does not require the use of indices, and is therefore more expedient. Let $f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}$, where $\boldsymbol{H}$ is symmetric. We have

$$f(\boldsymbol{x}+\delta\boldsymbol{x}) - f(\boldsymbol{x}) = (\delta\boldsymbol{x})^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{H}(\delta\boldsymbol{x}) + O\left(\|\delta\boldsymbol{x}\|^2\right),$$

and when $\boldsymbol{H}$ is symmetric this becomes

$$f(\boldsymbol{x}+\delta\boldsymbol{x}) - f(\boldsymbol{x}) = 2(\delta\boldsymbol{x})\boldsymbol{H}\boldsymbol{x}^T + O\left(\|\delta\boldsymbol{x}\|^2\right).$$

Observe that $2\boldsymbol{H}\boldsymbol{x}$ is the first-order approximation to $f(\boldsymbol{x}+\delta\boldsymbol{x})$, i.e. the derivative.

## Problem 6.42

The weight decay rule of equation (38) in the book does not exactly lead to equation (39). The rule multiplies the weight found by gradient descent by the factor $(1-\epsilon)$, such that

$$\boldsymbol{w}_{n+1} = \left(\boldsymbol{w}_n - \eta\nabla J(\boldsymbol{w}_n)\right)(1-\epsilon).$$

After some algebra, we observe that this is equivalent to

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \eta\underbrace{\left((1-\epsilon)\nabla J(\boldsymbol{w}_n) + \frac{\epsilon}{\eta}\boldsymbol{w}_n\right)}_{\nabla J_{ef}(\boldsymbol{w}_n)}.$$

Since $\nabla\boldsymbol{w}^T\boldsymbol{w} = 2\boldsymbol{w}$, we see that

$$J_{ef}(\boldsymbol{w}_n) = (1-\epsilon)J(\boldsymbol{w}_n) + \epsilon\frac{1}{2\eta}\boldsymbol{w}_n^T\boldsymbol{w}_n.$$

This is a weighted average of the ordinary error function $J(\boldsymbol{w}_n)$ and the regularization term $\frac{1}{2\eta}\boldsymbol{w}_n^T\boldsymbol{w}_n$, weighted by $(1-\epsilon)$ and $\epsilon$ respectively. We can easily verify that

i) When $\epsilon = 0$, everything reduces to gradient descent with no weight decay.
ii) When $\epsilon = 1$, the weight vector $\boldsymbol{w}_{n+1}$ is forever stuck at $\mathbf{0}$.

This also shows that equation (39) in the book is wrong with respect to equation (38), since if $\epsilon = 1$ in the book, equation (38) would always set $\boldsymbol{w}_{n+1} = \mathbf{0}$, while optimizing equation (39) would not—it would amount to optimizing $J(\boldsymbol{w})$ with a regularization term.

## 2.7 Stochastic methods

### Problem 7.4

Every magnet can be turned up or down, so there are $2^N$ possible configurations. However, flipping the sign of a state vector $\boldsymbol{s}$ does not alter $E$. For instance $\boldsymbol{s} = (1, 1, -1)$ results in the same energy as $\boldsymbol{s} = (-1, -1, 1)$. Therefore, only half of the $2^N$ states correspond to (possibly) unique energy levels, i.e. $2^{N-1}$ unique energy levels.

The total time required for exhaustive search will therefore be $T(N) = 2^{N-1} \cdot 10^{-8}$ seconds. Searching the space for $N = 100$ units would take $6.338 \cdot 10^{21}$ seconds. Searching the space for $N = 1000$ units would take $5.357 \cdot 10^{292}$ seconds. Both of these numbers are huge. Searching the space would take millions and millions of years.

### Problem 7.5

a) The formula is $E \propto \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} s_i s_j$, which is naively computed using $n^2$ flops. Since we have symmetry and $w_{ij} = w_{ji}$, we can write the sum as $\sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} s_i s_j$. The number of flops is then

$$(n-1) + (n-2) + \cdots + 2 + 1 = \frac{n(n-1)}{2}.$$

Furthermore, there are $2^{n-1}$ possibly unique energy energy levels. So the total time is given by

$$T(n) = 2^{n-2} n(n-1) \cdot 10^{-10}.$$

b) See figure 14 for a plot.

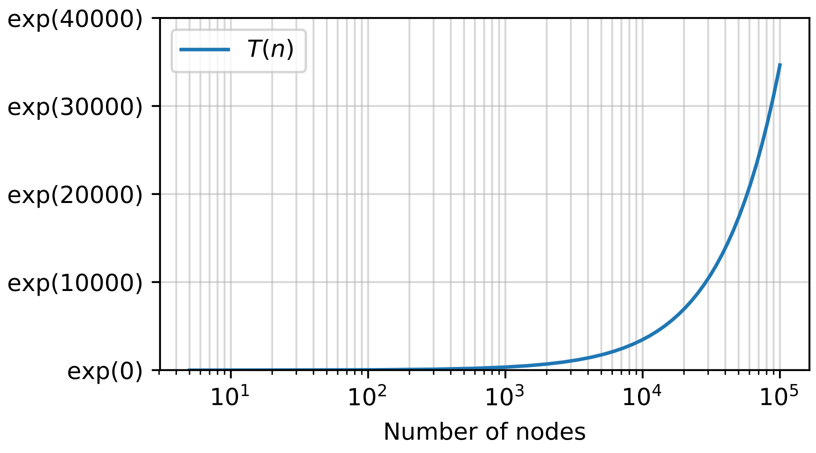

Figure 14: Solution to problem 7.5b).

c) To find the value of $n$ which may be computed in a day, we find $n$ such that

$$T(n) = 3600 \cdot 24 \quad \text{or, alternatively} \quad \ln T(n) = \ln(3600 \cdot 24).$$

Working in log-space avoids numerical overflow, since $T(n)$ grows very large very quickly. The answers are:

$$\begin{aligned} \text{A day:} \quad & \ln T(n) = \ln(3600 \cdot 24) && \Rightarrow n = 40 \\ \text{A year:} \quad & \ln T(n) = \ln(3600 \cdot 24 \cdot 365) && \Rightarrow n = 48 \\ \text{A century:} \quad & \ln T(n) = \ln(3600 \cdot 24 \cdot 365 \cdot 100) && \Rightarrow n = 55 \end{aligned}$$

## Problem 7.6

We wish to show that at high temperature, every configuration is equally likely. The probability that a system is in a configuration of energy $E_\gamma$ is given by equation (3) in chapter 7 the book, i.e. $P(\gamma) \propto \exp(-E_\gamma/T)$. As $T$ goes to infinity, we have

$$\lim_{T\to\infty} P(\gamma) = \lim_{T\to\infty} \left(e^{-1/T}\right)^{E_\gamma} = 1^{E_\gamma} = 1,$$

and so the probability of every state $\gamma$ is equally likely. The normalization constant $Z(T)$ makes it a true probability summing to unity.

## Computer exercise 7.2

A simple implementation of Algorithm 1 in from chapter 7 is found in the Python file `simulated_annealing.py`. The answers to sub-problem a) and b) are shown in Figures 15 and 16 respectively. A plot showing the average reduction in energy over 500 runs in show in 17, and it's interesting to see that the average reduction appears to be exponential.

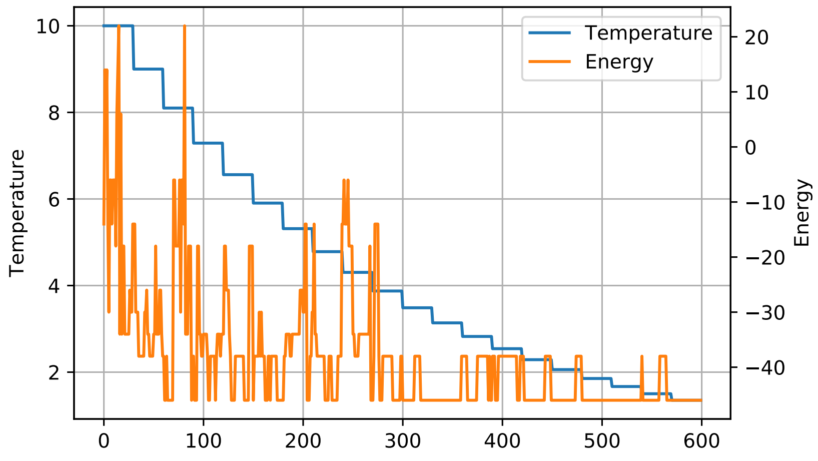

Figure 15: Simulated annealing with $T(1) = 10$ and $c = 0.9$. A total of 20 main iterations (with fixed temperature), and $5n$ sub iterations (with the same temperature) gives $100n = 600$ iterations.

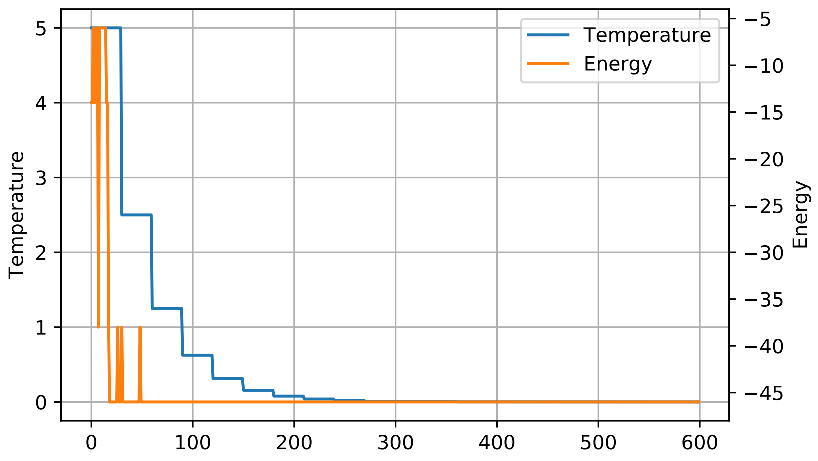

Figure 16: Simulated annealing with $T(1) = 5$ and $c = 0.5$. A total of 20 main iterations (with fixed temperature), and $5n$ sub iterations (with the same temperature) gives $100n = 600$ iterations.

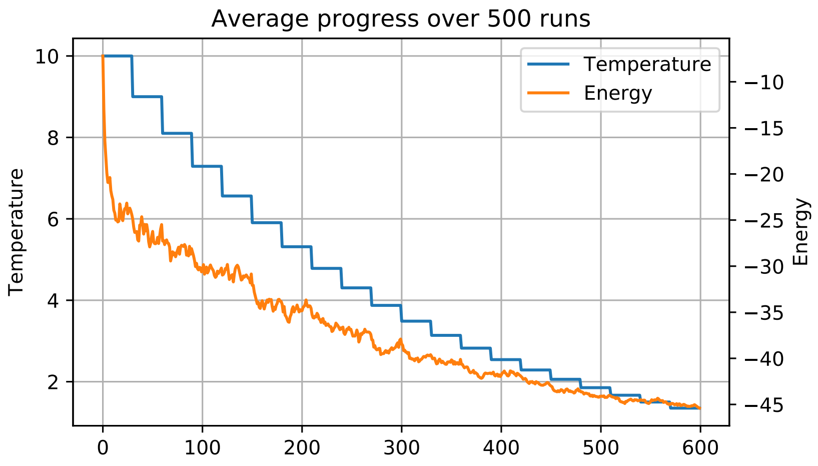

Figure 17: Simulated annealing over 500 runs with $T(1) = 10$ and $c = 0.9$. A total of 20 main iterations (with fixed temperature), and $5n$ sub iterations (with the same temperature) gives $100n = 600$ iterations.

## 2.8 Nonmetric methods

### Problem 8.10

a) One of the defining properties of an impurity function is that it should be zero when only $\omega_1$ is present, or when only $\omega_2$ is present. For a polynomial to incorporate this information, it must have two zero points, and therefore it must at least be quadratic.

b) The simplest quadratic form of $P(\omega_1)$ obeying the boundary conditions is

$$i(P(\omega_1)) \propto P(\omega_1)\left[1 - P(\omega_1)\right] = P(\omega_1)P(\omega_2).$$

c) Let $X$ be a Bernoulli variable, with $X \in \{0,1\} = \{\omega_2, \omega_1\} = C$ and $P(X=1) = P(\omega_1)$. In the given problem, we do not know the true probabilities—but they may be estimated from the fractions. The variance of $X$ is given by

$$\begin{aligned}
\operatorname{var}[X] &= \sum_{w_j \in C} P(\omega_j)\left[\omega_j - \mu\right]^2 \\
&= P(\omega_1)\left[\omega_1 - \mu\right]^2 + P(\omega_2)\left[\omega_2 - \mu\right]^2 \\
&= P(\omega_1)\left[1 - \mu\right]^2 + \left(1 - P(\omega_1)\right)\left[0 - \mu\right]^2 \\
&= P(\omega_1)\left[1 - P(\omega_1)\right]^2 + \left(1 - P(\omega_1)\right)P(\omega_1)^2 \\
&= \left[1 - P(\omega_1)\right]\left[P(\omega_1)\left(1 - P(\omega_1)\right) + P(\omega_1)^2\right] \\
&= \left(1 - P(\omega_1)\right)P(\omega_1).
\end{aligned}$$

In other words, sample variance is proportional to the impurity estimate defined in subproblem b). If the variance is high, the data is impure.

### Problem 8.14

We generalize the problem of a single missing attribute in a single training point to several missing attributes, and to several deficient training points.

In this problem, a pattern $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ with $d$ attributes can have between 0 and $d-1$ missing attributes. Furthermore, the data set $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ may have many training points with missing attributes. The data might look something like what is presented in Table 2.

High level Python-like pseudocode is given below. The principal difference between the code below and the code for the same problem without missing attributes, is that there might now be $(n-1-m_i)$ possible splits, where $m_i$ are the number of data points missing from attribute $i$. This is in contrast to the original algorithm, where there are in general $(n-1)$ possible splits for every attribute $i$.

```
for attribute=1 to d do:
  attribute_data = data[attribute, :]
  num_missing = count_missing(attribute_data)
```

Table 2: Example of what data for problem 8.14 might look like.

| $x_1$ | $x_2$ | $\dots$ | $x_d$ |
|---|---|---|---|
| 0 | $\emptyset$ | $\dots$ | 1 |
| $\emptyset$ | $\emptyset$ | $\dots$ | 1 |
| 1 | $\emptyset$ | $\dots$ | $\emptyset$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| 0 | $\emptyset$ | $\dots$ | 1 |

```
best_split = None
maximal_impurity_gain = -inf

for possible_split=1 to (n - 1 - num_missing) do:
  impurity_gain = compute_gain(attribute_data, possible_split)

  if impurity_gain > maximal_impurity_gain:
        best_split = (attribute, possible_split)
        maximal_impurity_gain = impurity_gain

split_along_maximal_impurity_gain(best_split)
```

## Problem 8.16

a) The following tree was constructed by hand. Recall that YES is to the left, and NO is to the right. The notation $\omega_{ij}$ signifies a pattern $j$ from category $i$.

$$\begin{tikzcd}
 & x_2 = 0 \arrow[dl, "\text{yes}"'] \arrow[dr, "\text{no}"] & & \\
\{\omega_{12}, \omega_{13}, \omega_{14}\} & & x_3 = 0 \arrow[dl, "\text{yes}"'] \arrow[dr, "\text{no}"] & \\
 & \{\omega_{11}, \omega_{21}\} & & \{\omega_{22}, \omega_{23}, \omega_{24}\}
\end{tikzcd}$$

The leftmost leaf node assigns to $\omega_1$ with probability 1, the middle leaf node assigns to either $\omega_1$ or $\omega_2$ with probability 0.5, and the rightmost node assigns to $\omega_2$ with probability 1.

b) The resulting tree would be identical to the one given in subproblem a), but the assignment from the middle node should be $\omega_1$ with probability 2/3.

**Problem 8.18**

We will consider every alignment of $\boldsymbol{x}$ and *text*. For each of these alignments, there will between 1 and $m$ comparisons before termination, depending on whether matches are found. There is a probability $d^{-1}$ of a match for each comparison. We will see that expected number of comparisons may be expressed as a *arithmetico-geometric* sum, i.e. a sum of terms which are the product of an arithmetic and geometric sum.

We start out by observing that there are $(n-m+1)$ alignments of $\boldsymbol{x}$ and *text*. For every alignment, $m$ characters must be compared at most. Let $1 \leq \ell \leq m$ be the number of comparisons made before the loop terminates. The loop terminates when a comparison yields no match. Fixating $\boldsymbol{x}$ and considering every character of *text* to be randomly drawn from an alphabet $\mathcal{A}$ of $d$ characters, there is a probability of $d^{-1}$ of a match in each comparison.

Let us now study $P(\ell)$, the probability of $\ell$ comparisons before termination of the loop. For instance $P(\ell) = 1 - 1/d$, since it represents the probability of one comparison before exiting the loop, and this happens if there is no match on the first comparison. More generally, we observe that

$$\begin{aligned}
P(\ell = 1) &= 1 - \frac{1}{d} && \text{(no match on the first)} \\
P(\ell = 2) &= \left(1 - \frac{1}{d}\right)\frac{1}{d} && \text{(match, then no match)} \\
P(\ell = 2) &= \left(1 - \frac{1}{d}\right)\frac{1}{d^2} && \text{(match, match, then no match)} \\
\vdots &= \vdots \\
P(\ell = m) &= \left(1 - \frac{1}{d}\right)\frac{1}{d^{m-1}} + \frac{1}{d^m} && \text{(no match on final, or match on final)}
\end{aligned}$$

We now take the expected value of $\ell$ to obtain the expected number of comparisons, and introduce a variable $r = 1/d$ for notational convenience.

$$\begin{aligned}
\mathbb{E}[\ell] = \sum_{k=1}^{m} P(\ell = k)\, k &= \left(1 - \frac{1}{d}\right)\left(1 + \frac{2}{d} + \frac{3}{d^2} + \cdots + \frac{m}{d^{m-1}}\right) + \frac{m}{d^m} \\
&= (1 - r)\left(1 + 2r + 3r^2 + \cdots + mr^{m-1}\right) + mr^m
\end{aligned}$$

The sum in the second parenthesis in the first term is a arithmetico-geometric series.

Using the summation formula[3] we obtain

$$\begin{aligned}\mathbb{E}\left[\ell\right] = \sum_{k=1}^{m} P(\ell = k)\, k &= (1-r)\left(1 + 2r + 3r^2 + \cdots + mr^{m-1}\right) + mr^m \\ &= (1-r)\left(\frac{1-(1-m)r^m}{1-r} + \frac{r(1-r^m)}{(1-r)^2}\right) + mr^m \\ &= \frac{(1-r) - (1-r)(1+m)r^m + r(1-r^m) + (1-r)mr^m}{1-r}.\end{aligned}$$

After some algebraic operations on this final fraction, we obtain the desired result

$$\mathbb{E}\left[\ell\right] = \sum_{k=1}^{m} P(\ell = k)\, k = \frac{1-r^m}{1-r} = \frac{1-d^{-m}}{1-d^{-1}}.$$

The problem is solved, since the expected number of comparisons is

$$\text{alignments} \times \text{comparisons} = (n-m+1) \times \frac{1-d^{-m}}{1-d^{-1}}.$$

The inequality $(1-d^{-m})/(1-d^{-1}) \leq 2$ presented in the problem description stems from the fact that the fraction is the sum of a geometric series

$$\frac{1-d^{-m}}{1-d^{-1}} = 1 + \frac{1}{d} + \frac{1}{d^2} + \cdots + \frac{1}{d^{m-1}}.$$

When $d = 2$ above, the sum equals 2. Clearly, when $d > 2$ the terms become smaller, and the sum must therefore be $< 2$. Therefore, as long as $d \geq 2$, the sum is $\leq 2$. This proves the inequality when $d \geq 2$, which is a reasonable assumption considering the problem.

## Problem 8.22

**A naive algorithm** which loops through the alphabet (of size $d$) for every letter in $\boldsymbol{x}$ (of size $m$) would give an $O(dm)$ algorithm. This is not very efficient, and faster algorithms may be constructed easily.

We can construct a **better algorithm** with runtime $O(d+m)$ as follows.

```
1. Construct an empty lookup table for every character in the alphabet
2. For every character and position in x, overwrite the lookup table
```

Below is actual Python code implementing the pseudocode above.

```
def last_occurence(alphabet, word):
  """
  Returns a mapping F[char] -> last occurrence in word.
  """
  mapping = {i:-1 for i in alphabet}  # This is O(d)
```

[3]See Wikipedia for the summation formula of a arithmetico-geometric series.

```
  for index, char in enumerate(word):  # This is O(m)
    mapping[char] = index

  return mapping
```

It's possible to make the algorithm closer to $O(\min(m, d))$, if we know the values of $m$ and $d$ before the algorithm starts. If $d \gg m$ or $m \gg d$, such an algorithm can avoid looping over the longest string, but we will not pursue this any further in this problem.

a) The time complexity of the algorithm is $O(m + d)$.
b) The space complexity is $O(d)$ in the algorithm above. We can avoid storing the empty matches (for instance by using the `collections.defaultdict` data structure in Python) and get the storage down to $O(\text{unique\_chars}(m))$.
c) Running the algorithm above requires $d+m$ operations, so that $\boldsymbol{x}$ = "`bonbon`" yields $26 + 6$ operations. Since $m < d$ here, we can use a `collections.defaultdict` data structure in Python and bring it down to $m = 6$ operations.

Below is a print showing that `collections.defaultdict` indeed leads to faster runtime in practice. The `collections.defaultdict` data structure is a dictionary (hash-table) which calls a *default factory function* is a value of a given key is not found.

```
%timeit last_occurence(alphabet, word)
  2.32 microseconds +/- 42 ns per loop
%timeit last_occurence_defaultdict(alphabet, word)
  1.17 microseconds +/- 3.17 ns per loop
```

## Problem 8.26

a) The properties of a metric $D(\boldsymbol{a}, \boldsymbol{b})$ are (i) Non-negativity; (ii) Reflexivity; (iii) Symmetry; and (iv) Triangle inequality. These properties are listed on page 187 in [Duda et al., 2000], which corresponds to section 4.6.1. In order, we see that
   (i) Non-negativity is always obeyed, since every cost is positive.
   (ii) Reflexivity is always obeyed, since if $\boldsymbol{a} = \boldsymbol{b}$ no operation is performed and the total cost of transformation is zero.
   (iii) Symmetry is not always obeyed. A counterexample follows.
   (iv) The triangle inequality is not always obeyed. A counterexample follows.
b) We prove counterexamples for symmetry and the triangle inequality.

   **Counterexample for symmetry** Let $\boldsymbol{a}$ = `dog` and $\boldsymbol{b}$ = `dogs`. Define the distance $D(\boldsymbol{a}, \boldsymbol{b})$ as the cost of transforming $\boldsymbol{a}$ to $\boldsymbol{b}$. Then $D(\boldsymbol{a}, \boldsymbol{b})$ is the distance of a single insertion to transform `dog` $\mapsto$ `dogs`, while $D(\boldsymbol{b}, \boldsymbol{a})$ is the distance of a single deletion `dogs` $\mapsto$ `dog`. If the cost of insertion and deletion differ, then $D(\boldsymbol{a}, \boldsymbol{b}) \neq D(\boldsymbol{b}, \boldsymbol{a})$. This provides a counterexample of the symmetry property of a metric when the costs of operations may be different.

**Counterexample for the triangle inequality** Let $\epsilon$ be the empty string. Consider the transformation shown in the diagram below. If the cost of substitution is much greater than the cost of deletion and insertion, then $D(\mathtt{a}, \epsilon) + D(\epsilon, \mathtt{b}) \ll D(\mathtt{a}, \mathtt{b})$. This provides a counterexample of the triangle inequality of a metric.

$$\begin{tikzcd}
\mathtt{a} \arrow[rr, "\text{substitution}"] \arrow[dr, "\text{deletion}"'] & & \mathtt{b} \\
& \epsilon \arrow[ur, "\text{insertion}"'] &
\end{tikzcd}$$

## Problem 8.31

a) A grammar generating $\mathcal{L}(G) = \{\mathtt{a}^n\mathtt{b} \mid n \geq 1\}$ is, for instance, given by

$$\mathcal{A} = \{a, b\} \qquad \mathcal{S} = S \qquad \mathcal{I} = \{A\}$$
$$\mathcal{P} = \{S \to Ab, A \to a, A \to Aa\}.$$

b) Below is a derivation tree showing how $ab$ and $aaaaab$ are generated.

$$\begin{tikzcd}
S \arrow[r] & Ab \arrow[r] \arrow[d] & Aab \arrow[r] \arrow[d] & Aaab \arrow[r] \arrow[d] & Aaaab \arrow[r] \arrow[d] & Aaaaab \arrow[r] \arrow[d] & \ldots \\
& ab & aab & aaab & aaaab & aaaaab &
\end{tikzcd}$$

## Problem 8.32

a) Since **Type** 3 $\subset$ **Type** 2 $\subset$ **Type** 1, we check if it's a **Type** 3 (regular) grammar first. It's not a regular grammar, since the rule $B \to aBa$ is not of the form $\alpha \to z\beta$ or of the form $\alpha \to z$. It is, however, a **Type** 2 (context-free) grammar since every rule is of the form $I \to \gamma$. In other words, every rewrite rule is from an intermediate symbol $I$ to a string $\gamma$ made up of intermediate of terminal symbols.

b) To see that the grammar generates the language $\mathcal{L}(G) = \{\mathtt{ca}^n\mathtt{cba}^n\mathtt{b} \mid n \geq 1\}$, we draw a derivation tree with intermediate states in the first row, and final states in the second row. The rule $B \to cb$ is highlighted in boldface.

$$\begin{tikzcd}
S \arrow[r] & cAb \arrow[r] & caBab \arrow[r] \arrow[d] & caaBaab \arrow[r] \arrow[d] & caaaBaaab \arrow[r] \arrow[d] & \ldots \\
& & ca\mathbf{cb}ab & caa\mathbf{cb}aab & caaa\mathbf{cb}aaab &
\end{tikzcd}$$

Clearly moving down from $caBab$ produces $\{\mathtt{ca}^n\mathtt{cba}^n\mathtt{b} \mid n = 1\}$, and moving $\ell$ to the right from $caBab$, and then down, produces $\{\mathtt{ca}^n\mathtt{cba}^n\mathtt{b} \mid n = \ell + 1\}$.

c) See the solution to subproblem b) above for derivation trees.

## 2.9 Algorithm-independent machine learning

### Problem 9.1

a) sdf

**Problem X.YY**

a) sdf

**Problem 6.26**

a) sdf

**Problem 6.31**

a) sdf

**Problem 6.37**

a) sdf

# 3 Notes from “Pattern Recognition and Machine Learning”

## 3.1 test

## 4 Solutions to "Pattern Recognition and Machine Learning"

This section contains solutions to problems in "Pattern Recognition and Machine Learning" by Christopher M. Bishop. There are approximately TODO solved problems from each chapter.

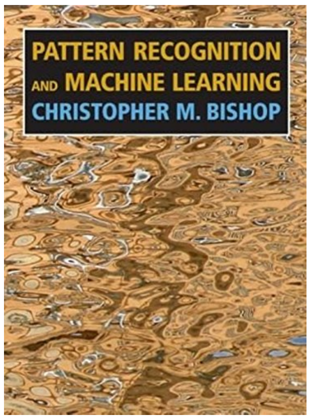

Figure 18: The front cover of [Bishop, 2011].

## References

[Bishop, 2011] Bishop, C. M. (2011). *Pattern Recognition and Machine Learning*. Springer, New York.

[Duda et al., 2000] Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification*. Wiley-Interscience, New York, 2 edition edition.