Notes and Solutions for: Pattern Recognition by Sergios Theodoridis and Konstantinos Koutroumbas.

John L. Weatherwax*

October 17, 2015

*wax@alum.mit.edu

Introduction

Here you'll find some notes that I wrote up as I worked through this excellent book. I've worked hard to make these notes as good as I can, but I have no illusions that they are perfect. If you feel that that there is a better way to accomplish or explain an exercise or derivation presented in these notes; or that one or more of the explanations is unclear, incomplete, or misleading, please tell me. If you find an error of any kind – technical, grammatical, typographical, whatever – please tell me that, too. I'll gladly add to the acknowledgments in later printings the name of the first person to bring each problem to my attention.

Acknowledgments

Special thanks to (most recent comments are listed first): Ronald Santos, Karonny F., and Mohammad Heshajin for helping improve these notes and solutions.

All comments (no matter how small) are much appreciated. In fact, if you find these notes useful I would appreciate a contribution in the form of a solution to a problem that I did not work, a mathematical derivation of a statement or comment made in the book that was unclear, a piece of code that implements one of the algorithms discussed, or a correction to a typo (spelling, grammar, etc) about these notes. Sort of a "take a penny, leave a penny" type of approach. Remember: pay it forward.

Classifiers Based on Bayes Decision Theory

Notes on the text

Minimizing the average risk

The symbol r_k is the expected risk associated with *observing* an object from class k. This risk is divided up into parts that depend on what we then *do* when an object from class k with feature vector x is observed. Now we only observe the feature vector x and not the true class label k. Since we must still perform an action when we observe x let λ_{ki} represent the *loss* associated with the event that the object is truly from class k and we decided that it is from class i. Define r_k as the *expected* loss when an object of type k is presented to us. Then

$$\begin{aligned} r_k &= \sum_{i=1}^{M} \lambda_{ki} P(\text{we classify this object as a member of class } i) \\ &= \sum_{i=1}^{M} \lambda_{ki} \int_{R_i} p(x|\omega_k) dx \,, \end{aligned}$$

which is the books equation 2.14. Thus the *total* risk r is the expected value of the class dependent risks r_k taking into account how likely each class is or

$$\begin{aligned} r &= \sum_{k=1}^{M} r_k P(\omega_k) \\ &= \sum_{k=1}^{M} \sum_{i=1}^{M} \lambda_{ki} \int_{R_i} p(x|\omega_k) P(\omega_k) dx \\ &= \sum_{i=1}^{M} \int_{R_i} \left(\sum_{k=1}^{M} \lambda_{ki} p(x|\omega_k) P(\omega_k) \right) dx \,. \end{aligned} \tag{1}$$

The decision rule that leads to the smallest total risk is obtained by selecting R_i to be the region of feature space in which the integrand above is as small as possible. That is, R_i should be defined as the values of x such that for that value of i we have

$$\sum_{k=1}^{M} \lambda_{ki} p(x|\omega_k) P(\omega_k) < \sum_{k=1}^{M} \lambda_{kj} p(x|\omega_k) P(\omega_k) \quad \forall j \,.$$

In words the index i, when put in the sum above gives the *smallest* value when compared to all other possible choices. For these values of x we should select class ω_i as our classification decision.

Bayesian classification with normal distributions

When the covariance matrices for two classes are the same and diagonal i.e. $\Sigma_i = \Sigma_j = \sigma^2 I$ then the discrimination functions $g_{ij}(x)$ are given by

$$g_{ij}(x) = w^T(x - x_0) = (\mu_i - \mu_j)^T(x - x_0)\,, \tag{2}$$

since the vector w is $w = \mu_i - \mu_j$ in this case. Note that the point x_0 is *on* the decision hyperplane i.e. satisfies $g_{ij}(x) = 0$ since $g_{ij}(x_0) = w^T(x_0 - x_0) = 0$. Let x be another point on the decision hyperplane, then $x - x_0$ is a vector in the decision hyperplane. Since x is a point on the decision hyperplane it also must satisfy $g_{ij}(x) = 0$ from the functional form for $g_{ij}(\cdot)$ and the definition of w is this means that

$$w^T(x - x_0) = (\mu_i - \mu_j)^T(x - x_0) = 0\,.$$

This is the statement that the line connecting μ_i and μ_j is orthogonal to the decision hyperplane. In the same way, when the covariance matrices of each class are not diagonal but are nevertheless the $\Sigma_i = \Sigma_j = \Sigma$ the same logic that we used above states that the decision hyperplane is again orthogonal to the vector w which in this case is $\Sigma^{-1}(\mu_i - \mu_j)$.

The magnitude of $P(\omega_i)$ relative to $P(\omega_j)$ influences how close the decision hyperplane is to the respective class means μ_i or μ_j, in the sense that the class with the larger a priori probability will have a "larger" region of $\mathbb{R}^l$ assigned to it for classification. For example, if $P(\omega_i) < P(\omega_j)$ then $\ln\left(\frac{P(\omega_i)}{P(\omega_j)}\right) < 0$ so the point x_0 which in the case $\Sigma_i = \Sigma_j = \Sigma$ is given by

$$x_0 = \frac{1}{2}(\mu_i + \mu_j) - \ln\left(\frac{P(\omega_i)}{P(\omega_j)}\right)\frac{\mu_i - \mu_j}{||\mu_i - \mu_j||^2_{\Sigma^{-1}}}\,, \tag{3}$$

we can write as

$$x_0 = \frac{1}{2}(\mu_i + \mu_j) + \alpha(\mu_i - \mu_j)\,,$$

with the value of $\alpha > 0$. Since $\mu_i - \mu_j$ is a vector from μ_j to μ_i the expression for x_0 above starts at the midpoint $\frac{1}{2}(\mu_i + \mu_j)$ and moves closer to μ_i. Meaning that the amount of $\mathbb{R}^l$ assigned to class ω_j is "larger" than the amount assigned to class ω_i. This is expected since the prior probability of class ω_j is larger than that of ω_i.

Notes on Example 2.2

To see the final lengths of the principal axes we start with the transformed equation of constant Mahalanobis distance of $d_m = \sqrt{2.952}$ or

$$\frac{(x_1' - \mu_{11}')^2}{\lambda_1} + \frac{(x_2' - \mu_{12}')^2}{\lambda_2} = (\sqrt{2.952})^2 = 2.952\,.$$

Since we want the principal axis about $(0, 0)$ we have $\mu_{11}' = \mu_{12}' = 0$ and λ_1 and λ_2 are the eigenvalues given by solving $|\Sigma - \lambda I| = 0$. In this case, we get $\lambda_1 = 1$ (in direction v_1) and $\lambda_2 = 2$ (in direction v_2). Then the above becomes in "standard form" for a conic section

$$\frac{(x_1')^2}{2.952\lambda_1} + \frac{(x_2')^2}{2.952\lambda_2} = 1\,.$$

From this expression we can read off the lengths of the principle axis

$$\begin{aligned} 2\sqrt{2.952\lambda_1} &= 2\sqrt{2.952} = 3.43627 \\ 2\sqrt{2.952\lambda_2} &= 2\sqrt{2.952(2)} = 4.85962\,. \end{aligned}$$

Maximum A Posteriori (MAP) Estimation: Example 2.4

We will derive the MAP estimate of the population mean μ when given N samples x_k distributed as $p(x|\mu)$ and a normal prior on μ i.e. $N(\mu_0, \sigma_\mu^2 I)$. Then the estimate of the population mean μ given the sample $X \equiv \{x_k\}_{k=1}^N$ is proportional to

$$p(\mu|X) \propto p(\mu)p(X|\mu) = p(\mu)\prod_{k=1}^{N} p(x_k|\mu)\,.$$

Note that we have written $p(\mu)$ on the outside of the product terms since it should only appear *once* and not N times as might be inferred by had we written the product as $\prod_{k=1}^N p(\mu)p(x_k|\mu)$. To find the value of μ that maximized this we take begin by taking the natural log of the expression above, taking the μ derivative and setting the resulting expression equal to zero. We find the natural log of the above given by

$$\ln(p(\mu)) + \sum_{k=1}^{N} \ln(p(x_k|\mu)) = -\frac{1}{2}\frac{||\mu - \mu_0||^2}{\sigma_\mu^2} - \frac{1}{2}\sum_{k=1}^{N}(x_k - \mu)^T\Sigma^{-1}(x_k - \mu)\,.$$

Then taking the derivative with respect to μ, setting the result equal to zero, and calling that solution $\hat{\mu}$ gives

$$-\frac{1}{\sigma_\mu^2}(\hat{\mu} - \mu_0) + \frac{1}{\sigma^2}\sum_{k=1}^{N}(x_k - \hat{\mu}) = 0\,,$$

were we have assumed that the density $p(x|\mu)$ is $N(\mu, \Sigma)$ with $\Sigma = \sigma^2 I$. When we solve for $\hat{\mu}$ in the above we get

$$\hat{\mu} = \frac{\frac{1}{\sigma_\mu^2}\mu_0 + \frac{1}{\sigma^2}\sum_{k=1}^{N} x_k}{\frac{N}{\sigma^2} + \frac{1}{\sigma_\mu^2}} = \frac{\mu_0 + \frac{\sigma_\mu^2}{\sigma^2}\sum_{k=1}^{N} x_k}{1 + \frac{\sigma_\mu^2}{\sigma^2}N} \tag{4}$$

Maximum Entropy Estimation

As another method to determine distribution parameters we seek to maximize the entropy H or

$$H = -\int_{\mathcal{X}} p(x)\ln(p(x))dx\,. \tag{5}$$

This is equivalent to minimizing its negative or $\int_{\mathcal{X}} p(x)\ln(p(x))dx$. To incorporate the constraint that the density must integrate to one, we form the entropy Lagrangian

$$H_L = \int_{x_1}^{x_2} p(x)\ln(p(x))dx - \lambda\left(\int_{x_1}^{x_2} p(x)dx - 1\right)\,,$$

where we have assumed that our density is non-zero only over $[x_1, x_2]$. The negative of the above is equivalent to

$$-H_L = -\int_{x_1}^{x_2} p(x)(\ln(p(x)) - \lambda)dx - \lambda\,.$$

Taking the $p(x)$ derivative and setting it equal to zero

$$\begin{aligned}\frac{\partial(-H_L)}{\partial p} &= -\int_{x_1}^{x_2}[(\ln(p) - \lambda) + p\left(\frac{1}{p}\right)]dx \\ &= -\int_{x_1}^{x_2}[\ln(p) - \lambda + 1]dx = 0\,.\end{aligned}$$

Solving for the integral of $\ln(p(x))$ we get

$$\int_{x_1}^{x_2} \ln(p(x))dx = (\lambda - 1)(x_2 - x_1)\,.$$

Take the x_2 derivative of this expression and we find x

$$\ln(p(x_2)) = \lambda - 1 \quad \Rightarrow \quad p(x_2) = e^{\lambda-1},\,.$$

To find the value of λ we put this expression into our constraint of $\int_{x_1}^{x_2} p(x)dx = 1$ to get

$$e^{\lambda-1}(x_2 - x_1) = 1\,,$$

or $\lambda - 1 = \ln\left(\frac{1}{x_2 - x_1}\right)$, thus

$$p(x) = \exp\left\{\ln\left(\frac{1}{x_2 - x_1}\right)\right\} = \frac{1}{x_2 - x_1}\,,$$

a *uniform* distribution.

Problem Solutions

Problem 2.1 (the Bayes' rule minimized the probability of error)

Following the hint in the book, the probability of correct classification P_c is given by

$$P_c = \sum_{i=1}^{M} P(x \in \mathcal{R}_i, \omega_i)\,,$$

since in order to be correct when $x \in \mathcal{R}_i$ the sample that generated x must come from the class ω_i. Now this joint probability is given by

$$\begin{aligned}P(x \in \mathcal{R}_i, \omega_i) &= P(x \in \mathcal{R}_i|\omega_i)P(\omega_i) \\ &= \left(\int_{\mathcal{R}_i} p(x|\omega_i)dx\right) P(\omega_i)\,.\end{aligned}$$

So the expression for P_c then becomes

$$\begin{aligned} P_c &= \sum_{i=1}^{M} \left(\int_{\mathcal{R}_i} p(x|\omega_i) dx \right) P(\omega_i) \\ &= \sum_{i=1}^{M} \left(\int_{\mathcal{R}_i} p(x|\omega_i) P(\omega_i) dx \right) . \end{aligned} \tag{6}$$

Since this is a sum of M different terms to maximize P_c we will define $\mathcal{R}_i$ to be the region of x where

$$p(x|\omega_i)P(\omega_i) > p(x|\omega_j)P(\omega_j) \quad \forall j \neq i \,. \tag{7}$$

If we do this, then since in $\mathcal{R}_i$ from Equation 7 the expression $\int_{\mathcal{R}_i} p(x|\omega_i)P(\omega_i)dx$ will be as large as possible. As Equation 6 is the sum of such terms we will have also maximized P_c. Now dividing both sides of Equation 7 and using Bayes' rule we have

$$P(\omega_i|x) > P(\omega_j|x) \quad \forall j \neq i \,,$$

as the multi-class decision boundary, what we were to show.

Problem 2.2 (finding the decision boundary)

Using the books notation where λ_{ki} is the loss associated with us deciding an object is from class i when it in fact is from class k we need to compare the expressions given by Equation 1. Since this is a two class problem $M = 2$ and we need to compare

$$\begin{aligned} l_1 &= \lambda_{11} p(x|\omega_1) P(\omega_1) + \lambda_{21} p(x|\omega_2) P(\omega_2) \\ l_2 &= \lambda_{12} p(x|\omega_1) P(\omega_1) + \lambda_{22} p(x|\omega_2) P(\omega_2) \,. \end{aligned}$$

Under zero-one loss these become

$$\begin{aligned} l_1 &= \lambda_{21} p(x|\omega_2) P(\omega_2) \\ l_2 &= \lambda_{12} p(x|\omega_1) P(\omega_1) \,. \end{aligned}$$

When $l_1 < l_2$ we will classify x as from class ω_1 and from class ω_2 otherwise. The decision boundary will be the point x_0 where $l_1 = l_2$. This later equation (if we solve for the likelihood ratio) is

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} = \frac{\lambda_{21} P(\omega_2)}{\lambda_{12} P(\omega_1)} \,. \tag{8}$$

If we assume that $p(x|\omega_1) \sim \mathcal{N}(0, \sigma^2)$ and $p(x|\omega_2) \sim \mathcal{N}(1, \sigma^2)$ then

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} = \frac{e^{-\frac{1}{2}\frac{x^2}{\sigma^2}}}{e^{-\frac{1}{2}\frac{(x-1)^2}{\sigma^2}}} = \exp\left\{ -\frac{1}{2}\frac{1}{\sigma^2}(2x-1) \right\} .$$

Setting this equal to $\frac{\lambda_{21} P(\omega_2)}{\lambda_{12} P(\omega_1)}$ and solving for x gives

$$x = \frac{1}{2} - \sigma^2 \ln\left(\frac{\lambda_{21} P(\omega_2)}{\lambda_{12} P(\omega_1)} \right) ,$$

as we were to show.

Problem 2.3 (an expression for the average risk)

We are told to define the errors $\varepsilon_{1,2}$ as the class conditional error or

$$\begin{aligned}\varepsilon_1 &= P(x \in \mathcal{R}_2|\omega_1) = \int_{\mathcal{R}_2} p(x|\omega_1)dx \\ \varepsilon_2 &= P(x \in \mathcal{R}_1|\omega_2) = \int_{\mathcal{R}_1} p(x|\omega_2)dx\,.\end{aligned}$$

Using these definitions we can manipulate the average risk as

$$\begin{aligned}r &= P(\omega_1)\left(\lambda_{11}\int_{\mathcal{R}_1} p(x|\omega_1)dx + \lambda_{12}\int_{\mathcal{R}_2} p(x|\omega_1)dx\right) \\ &+ P(\omega_2)\left(\lambda_{21}\int_{\mathcal{R}_1} p(x|\omega_2)dx + \lambda_{22}\int_{\mathcal{R}_2} p(x|\omega_2)dx\right) \\ &= P(\omega_1)\left(\lambda_{11}\left(1-\int_{\mathcal{R}_2} p(x|\omega_1)dx\right) + \lambda_{12}\int_{\mathcal{R}_2} p(x|\omega_1)dx\right) \\ &+ P(\omega_2)\left(\lambda_{21}\int_{\mathcal{R}_1} p(x|\omega_2)dx + \lambda_{22}\left(1-\int_{\mathcal{R}_1} p(x|\omega_2)dx\right)\right) \\ &= \lambda_{11}P(\omega_1) - \lambda_{11}\varepsilon_1 P(\omega_1) + \lambda_{12}\varepsilon_1 P(\omega_1) + \lambda_{21}\varepsilon_2 P(\omega_2) + \lambda_{22}P(\omega_1) - \lambda_{22}\varepsilon_2 P(\omega_2) \\ &= \lambda_{11}P(\omega_1) + \lambda_{22}P(\omega_2) + (\lambda_{12}-\lambda_{11})\varepsilon_1 P(\omega_1) + (\lambda_{12}-\lambda_{22})\varepsilon_2 P(\omega_2)\,,\end{aligned}$$

resulting in the desired expression.

Problem 2.4 (bounding the probability of error)

We desire to show that

$$P_e \leq 1 - \frac{1}{M}\,.$$

To do this recall that since $\sum_{i=1}^{M} P(\omega_i|x) = 1$ at least one $P(\omega_i|x)$ must be larger than $\frac{1}{M}$ otherwise the sum $\sum_{i=1}^{M} P(\omega_i|x)$ would have to be less than one. Now let $P(\omega_{i^*}|x)$ be the Bayes' classification decision. That is

$$P(\omega_{i^*}|x) = \max_i P(\omega_i|x)\,.$$

From the above discussion $P(\omega_{i^*}|x) \geq \frac{1}{M}$. From this we see that

$$P_e = 1 - \max_i P(\omega_i|x) \leq 1 - \frac{1}{M} = \frac{M-1}{M}\,,$$

the desired expression.

Problem 2.5 (classification with Gaussians of the same mean)

Since this is a two-class problem we can use the results from the book. We compute l_{12} the likelihood ratio

$$l_{12} = \frac{p(x|\omega_1)}{p(x|\omega_2)} = \frac{\sigma_2^2}{\sigma_1^2} \exp\left\{ -\frac{x^2}{2}\left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2}\right)\right\} . \tag{9}$$

and compare this to the threshold t defined by

$$t \equiv \frac{P(\omega_2)}{P(\omega_1)}\left(\frac{\lambda_{21}-\lambda_{22}}{\lambda_{12}-\lambda_{11}}\right) .$$

Then if $l_{12} > t$, then we classify x as a member of ω_1. In the same way if $l_{12} < t$ then we classify x as a member of ω_2. The decision point x_0 where we switch classification is given by $l_{12}(x_0) = t$ or

$$\exp\left\{ -\frac{x_0^2}{2}\left(\frac{\sigma_2^2-\sigma_1^2}{\sigma_1^2\sigma_2^2}\right)\right\} = \frac{\sigma_1^2}{\sigma_2^2} t .$$

Solving for x_0 we get

$$x_0 = \pm \frac{2\sigma_1^2\sigma_2^2}{(\sigma_2^2-\sigma_1^2)} \ln\left(\frac{\sigma_1^2 P(\omega_2)}{\sigma_2^2 P(\omega_1)}\left(\frac{\lambda_{21}-\lambda_{22}}{\lambda_{12}-\lambda_{11}}\right)\right) .$$

For specific values of the parameters in this problem: σ_i^2, $P(\omega_i)$, and λ_{ij} the two values for x_0 above can be evaluated. These two values x_0 differ only in their sign and have the same magnitude. For these given values of x_0 we see that if $|x| \leq |x_0|$ one class is selected as the classification decision while if $|x| > |x_0|$ the other class is selected. The class selected depends on the relative magnitude of the parameters σ_i^2, $P(\omega_i)$, and λ_{ij} and seems difficult to determine a priori. To determine the class once we are given a fixed specification of these numbers we can evaluate l_{12} in Equation 9 for a specific value of x such that $|x| \leq |x_0|$ (say $x = 0$) to determine if $l_{12} < t$ or not. If so the region of x's given by $|x| \leq |x_0|$ will be classified as members of class ω_1, while the region of x's where $|x| > |x_0|$ would be classified as members of ω_2.

Problem 2.6 (the Neyman-Pearson classification criterion)

Warning: I was not able to solve this problem. Below are a few notes I made while attempting it. If anyone sees a way to proceed please contact me.

For this problem we want to *fix* the value of the error probability for class one at a particular value say $\varepsilon_1 = \varepsilon$ and then we want to minimize the probability of error we make when classifying the *other* class. Now recall the definitions of the error probability ε_i

$$\varepsilon_1 = \int_{\mathcal{R}_2} p(x|\omega_1)P(\omega_1)dx ,$$

and

$$\varepsilon_2 = \int_{\mathcal{R}_1} p(x|\omega_2)P(\omega_2)dx .$$

We want to find a region $\mathcal{R}_1$ (equivalently a region $\mathcal{R}_2$) such that ε_2 is minimized under the constraint that $\varepsilon_1 = \varepsilon$. We can do this using the method of Lagrange multipliers. To use this method first form the Lagrangian q defined as

$$q = \theta(\varepsilon_1 - \varepsilon) + \varepsilon_2 \,.$$

To determine the classification decision region boundary x_0 that minimizes this Lagrangian let $\mathcal{R}_1 = (-\infty, x_0)$ and $\mathcal{R}_2 = (x_0, +\infty)$ to get for q

$$q = \theta \left(P(\omega_1) \int_{x_0}^{+\infty} p(x|\omega_1)dx - \varepsilon \right) + P(\omega_2) \int_{-\infty}^{x_0} p(x|\omega_2)dx \,.$$

To minimized q with respect to x_0 requires taking $\frac{\partial}{\partial x_0}$ of the above expression and setting this derivative equal to zero. This gives

$$\frac{\partial q}{\partial x_0} = -\theta P(\omega_1)p(x_0|\omega_1) + P(\omega_2)p(x_0|\omega_2) = 0 \,.$$

Solving for θ in the above expression we see that it is given by

$$\theta = \frac{P(\omega_2)p(x_0|\omega_2)}{P(\omega_1)p(x_0|\omega_1)} = \frac{P(\omega_2|x_0)}{P(\omega_1|x_0)} \,.$$

Problem 2.7

Part (a): To classify a feature vector x we form the three discriminants

$$\begin{aligned} g_i(x) &= w_i^T x + w_{i0} \quad \text{for} \quad i = 1, 2, 3 \\ w_i &= \Sigma^{-1}\mu_i \\ w_{i0} &= \ln(P(\omega_i)) - \frac{1}{2}\mu_i^T \Sigma^{-1} \mu_i \,, \end{aligned} \tag{10}$$

and classify to the class i that has the largest value of $g_i(x)$. If the three classes are equiprobable then $P(\omega_i) = \frac{1}{3}$ and we are given values for the means and the covariance matrix thus we can compute the three values $g_i(x)$ for $i = 1, 2, 3$. In the MATLAB file `chap_2_prob_7.m` we do this calculation and find that the largest index corresponds to the *second* class.

Part (b): To plot curves of equal Mahalanobis distance from the given vector we want to plot contours of the function

$$d(x, y) = \sqrt{\begin{bmatrix} x - 2.1 \\ y - 1.9 \end{bmatrix}^T \Sigma \begin{bmatrix} x - 2.1 \\ y - 1.9 \end{bmatrix}} \,.$$

We could do this with the `contour` function in MATLAB.

Problem 2.8

The linear discriminant function for a two class classification problem has its individual discriminants given by $g_i(x)$ via Equation 10 and then in the two class case we define

$$g_{12}(x) \equiv g_1(x) - g_2(x)\,.$$

We can classify a given x with this function by evaluating the sign of $g_{12}(x)$. If $g_{12}(x) > 0$ then we classify the sample as from class "one" while if $g_{12}(x) < 0$ we classify the sample as from class "two". Using the above formulas for w_i and w_{i0} in the definition of $g_i(x)$ we can show that we can write $g_{12}(x)$ as

$$g_{12}(x) = w^T(x - x_0)\,,$$

where

$$\begin{aligned} w &= \Sigma^{-1}(\mu_1 - \mu_2) \\ x_0 &= \frac{1}{2}(\mu_1 + \mu_2) - \ln\left(\frac{P(\omega_1)}{P(\omega_2)}\right)\frac{\mu_1 - \mu_2}{||\mu_1 - \mu_2||^2_{\Sigma^{-1}}} \\ ||x||_{\Sigma^{-1}} &= (x^T\Sigma^{-1}x)^{1/2}\,. \end{aligned}$$

In the MATLAB file `chap_2_prob_8.m` we implement this procedure for the numbers given in the problem. For concreteness we take $P(\omega_1) = 0.3$ so $P(\omega_2) = 0.7$. When we execute that script we find that our linear decision surface is the evaluation of the following

$$\begin{aligned} g_{12}(x_1, x_2, x_3) = w^T(x - x_0) &= \begin{bmatrix} 0 & -2.5 & -2.5 \end{bmatrix} \begin{bmatrix} x_1 + 0.4195 \\ x_2 + 0.4195 \\ x_3 + 0.4195 \end{bmatrix} \\ &= -2.5(x_2 + 0.4195) - 2.5(x_3 + 0.4195) \\ &= -2.5(x_2 + x_3) - 2.0975\,. \end{aligned}$$

Problem 2.9 (deriving an expression for P_B)

If $P(\omega_1) = P(\omega_2) = \frac{1}{2}$, then the zero-one likelihood ratio given by Equation 8 when $\lambda_{12} = \lambda_{21} = 1$, becomes

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} = 1\,.$$

Taking logarithms of both sides of this expression gives

$$\ln(p(x|\omega_1)) - \ln(p(x|\omega_2)) = 0\,.$$

If we define this *scalar* difference as u we see that when both class condition densities are multi-dimensional normal with the same covariance Σ (but with different means μ_i) that u

becomes

$$\begin{aligned} u &\equiv \ln\left(\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left\{-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right\}\right) \\ &- \ln\left(\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left\{-\frac{1}{2}(x-\mu_2)^T\Sigma^{-1}(x-\mu_2)\right\}\right) \\ &= -\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)+\frac{1}{2}(x-\mu_2)^T\Sigma^{-1}(x-\mu_2) \\ &= -\frac{1}{2}\left[x^T\Sigma^{-1}x-2\mu_1^T\Sigma^{-1}x+\mu_1^T\Sigma^{-1}\mu_1-x^T\Sigma^{-1}x+2\mu_2^T\Sigma^{-1}x-\mu_2^T\Sigma^{-1}\mu_2\right] \\ &= (\mu_1-\mu_2)^T\Sigma^{-1}x-\frac{1}{2}\mu_1\Sigma^{-1}\mu_1+\frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2\,. \end{aligned}$$

Now if x is a sample from class ω_1 then from the class density assumptions and the above expression (it is linear in x) we see that the variable u will be a Gaussian random variable with a mean m_1 given by

$$\begin{aligned} m_1 &= (\mu_1-\mu_2)\Sigma^{-1}\mu_1-\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1+\frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 \\ &= \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1-\mu_2\Sigma^{-1}\mu_1+\frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 \\ &= \frac{1}{2}(\mu_1-\mu_2)^T\Sigma^{-1}(\mu_1-\mu_2)\,, \end{aligned} \tag{11}$$

and a variance given by

$$(\mu_1-\mu_2)^T\Sigma^{-1}\Sigma\Sigma^{-1}(\mu_1-\mu_2)=(\mu_1-\mu_2)^T\Sigma^{-1}(\mu_1-\mu_2)\,. \tag{12}$$

In the same way, if x is from class ω_2 then u will be Gaussian with a mean m_2 given by

$$\begin{aligned} m_2 &= (\mu_1-\mu_2)\Sigma^{-1}\mu_2-\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1+\frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 \\ &= \mu_1^T\Sigma^{-1}\mu_2-\frac{1}{2}\mu_1\Sigma^{-1}\mu_1-\frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 \\ &= -\frac{1}{2}(\mu_1-\mu_2)^T\Sigma^{-1}(\mu_1-\mu_2)=-m_1\,, \end{aligned} \tag{13}$$

and the same variance as in the case when x is from ω_1 and given by Equation 12. Note that in terms of the Mahalanobis distance d_m between the means μ_i defined as

$$d_m^2 \equiv (\mu_1-\mu_2)^T\Sigma^{-1}(\mu_1-\mu_2)\,,$$

we have $m_1=\frac{1}{2}d_m^2=-m_2$. Also note that the variance of $p(u|\omega_i)$ given in Equation 12 in terms of the Mahalanobis distance is d_m^2 and so the common standard deviation would be d_m. Thus an expression for the Bayesian error probability, P_B, for this classification problem, can be computed based on the *one-dimensional* random variable u rather than the multidimensional random variable x. Since u is a scalar expression for which we know its probability density we can compute the probability of error for a classification procedure on x by using u's density. As just computed the densities $p(u|\omega_1)$ and $p(u|\omega_2)$ are Gaussian with symmetric means ($m_1=-m_2$) and equal variances so the the optimal Bayes classification

threshold value (where classification decisions change) is 0. This makes it easy to calculate P_B the optimal Bayes error estimate using

$$P_B = \int_{-\infty}^{0} p(u|\omega_1)P(\omega_1)du + \int_{0}^{\infty} p(u|\omega_2)P(\omega_2)du\,.$$

Since $P(\omega_1) = P(\omega_1) = \frac{1}{2}$, and the integrals of the conditional densities are equal (by symmetry) we can evaluate just one. Specifically, from the discussion above we have

$$p(u|\omega_1) = \frac{1}{\sqrt{2\pi}d_m}\exp\left\{-\frac{1}{2}\left(\frac{u-\frac{1}{2}d_m^2}{d_m}\right)\right\}\,,$$

and we find

$$P_B = \frac{1}{\sqrt{2\pi}}\frac{1}{d_m}\int_{-\infty}^{0} e^{-\frac{1}{2}\left(\frac{u-\frac{1}{2}d_m^2}{d_m}\right)}du\,.$$

Let $z = \frac{u-\frac{1}{2}d_m^2}{d_m}$ so $dz = \frac{1}{d_m}du$ and P_B becomes

$$P_B = \frac{1}{\sqrt{2\pi}}\frac{d_m}{d_m}\int_{-\infty}^{-\frac{1}{2}d_m} e^{-\frac{z^2}{2}}dz = \frac{1}{\sqrt{2\pi}}\int_{\frac{1}{2}d_m}^{+\infty} e^{-\frac{z^2}{2}}dz\,, \tag{14}$$

the result we were to show. Using the cumulative distribution function for the standard normal, denoted here in the notation of the Matlab function, normcdf, defined as

$$\text{normcdf}(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{x} e^{-\frac{1}{2}\frac{(t-\mu)^2}{\sigma^2}}dt\,, \tag{15}$$

we can write the above expression for P_B as

$$P_B = 1 - \text{normcdf}\left(\frac{1}{2}d_m; 0, 1\right)\,, \tag{16}$$

which is easier to implement in Matlab.

Problem 2.10-11 (Mahalanobis distances and classification)

Taking the logarithm of the likelihood ratio l_{12} gives a decision rule to state $x \in \omega_1$ (and $x \in \omega_2$ otherwise) if

$$\ln(p(x|\omega_1)) - \ln(p(x|\omega_2)) > \ln(\theta)\,. \tag{17}$$

If our conditional densities, $p(x|\omega_i)$, are given by a multidimensional normal densities then they have functional forms given by

$$p(x|\omega_i) = \mathcal{N}(x;\mu_i,\Sigma_i) \equiv \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}}\exp\left\{-\frac{1}{2}(x-\mu_i)^t\Sigma_i^{-1}(x-\mu_i)\right\}\,. \tag{18}$$

Taking the logarithm of this expression as required by Equation 17 we find

$$\begin{aligned}\ln(p(x|\omega_i)) &= -\frac{1}{2}(x-\mu_i)^t\Sigma_i^{-1}(x-\mu_i) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma_i|)\\ &= -\frac{1}{2}d_m(\mu_i, x|\Sigma_i)^2 - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma_i|)\,,\end{aligned}$$

where we have introduced the Mahalanobis distance d_m in the above. Our decision rule given by 17 in the case when $p(x|\omega_i)$ is a multidimensional Gaussian is thus given by

$$-\frac{1}{2}d_m(\mu_1, x|\Sigma_1)^2 - \frac{1}{2}\ln(|\Sigma_1|) + \frac{1}{2}d_m(\mu_2, x|\Sigma_2)^2 + \frac{1}{2}\ln(|\Sigma_2|) > \ln(\theta)\,.$$

or

$$d_m(\mu_1, x|\Sigma_1)^2 - d_m(\mu_2, x|\Sigma_2)^2 + \ln\left(\frac{|\Sigma_1|}{|\Sigma_2|}\right) < -2\ln(\theta)\,. \tag{19}$$

We will now consider some specializations of these expressions for various possible values of Σ_i. If our decision boundaries are given by Equation 19, but with equal covariances, then we have that $\ln\left(\frac{|\Sigma_1|}{|\Sigma_2|}\right) = 0$ and for decision purposes the left-hand-side of Equation 19 becomes

$$\begin{aligned} d_m(\mu_1, x|\Sigma_1)^2 - d_m(\mu_2, x|\Sigma_2)^2 &= \left(x^T\Sigma^{-1}x - 2x^T\Sigma^{-1}\mu_1 + \mu_1^T\Sigma^{-1}\mu_1\right) \\ &- \left(x^T\Sigma^{-1}x - 2x^T\Sigma^{-1}\mu_2 + \mu_2^T\Sigma^{-1}\mu_2\right) \\ &= -2x^T\Sigma^{-1}(\mu_1 - \mu_2) + \mu_1^T\Sigma^{-1}\mu_1 - \mu_2\Sigma^{-1}\mu_2\,. \end{aligned}$$

Using this in Equation 19 we get

$$(\mu_1 - \mu_2)^T\Sigma^{-1}x > \ln(\theta) + \frac{1}{2}||\mu_1||^2_{\Sigma^{-1}} - \frac{1}{2}||\mu_2||^2_{\Sigma^{-1}}\,, \tag{20}$$

for the decision boundary. I believe the book is missing the square on the Mahalanobis norm of μ_i.

Problem 2.12 (an example classifier design)

Part (a): The optimal two class classifier that minimizes the probability of error is given by classifying a point x depending on the value of the likelihood ratio

$$l_{12} = \frac{p(x|\omega_1)}{p(x|\omega_2)}\,.$$

When the class conditional probability densities, $p(x|\omega_i)$, are multivariate Gaussian with each feature dimension having the same variance $\sigma_1^2 = \sigma_2^2 = \sigma^2$ the above likelihood ratio l_{12} becomes

$$l_{12} = \exp\{-\frac{1}{2\sigma^2}((x-\mu_1)^T(x-\mu_1) - (x-\mu_2)^T(x-\mu_2))\}\,.$$

The value of l_{12} is compared to an expression involving the priori probabilities $P(\omega_i)$, and loss expressions, λ_{ki}, where

- λ_{ki} is the loss associated with classifying an object from the *true* class k as an object from class i.

The classification decision is then made according to

$$x \in \omega_1 \quad \text{if} \quad l_{12} = \frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{P(\omega_2)}{P(\omega_1)}\left(\frac{\lambda_{21} - \lambda_{22}}{\lambda_{12} - \lambda_{11}}\right)\,, \tag{21}$$

and $x \in \omega_2$ otherwise. To minimize the error probability requires we take $\lambda_{ki} = 1 - \delta_{ki}$ where δ_{ki} is the Kronecker delta. In this case, and when the priori probabilities are equal the constant on the right-hand-side of Equation 21 evaluates to 1. Thus our classification rules is to select class 1 if $l_{12} > 1$ and otherwise select class 2. This rule is equivalent to select $x \in \omega_1$ if $p(x|\omega_1) > p(x|\omega_2)$. Using the above expression the decision rule $l_{12} > 1$ simplifies as follows

$$\begin{aligned}(x-\mu_1)^T(x-\mu_1) - (x-\mu_2)^T(x-\mu_2) &< 0 \quad \text{or} \\ -2(\mu_1-\mu_2)^T x &< \mu_2^T\mu_2 - \mu_1^T\mu_1 \quad \text{or} \\ (\mu_1-\mu_2)^T x &> \frac{\mu_1^T\mu_1 - \mu_2^T\mu_2}{2}\,.\end{aligned}$$

This is equivalent to Equation 20 when we take $\theta = 1$ and equal features covariances.

Part (b): In this case, from the given loss matrix, Λ, we see that $\lambda_{11} = 0$, $\lambda_{12} = 1$, $\lambda_{21} = 0.5$, $\lambda_{22} = 0$ and the right-hand-side of Equation 21 becomes $\frac{1}{2}$. Then the requirement on the likelihood ratio is then $l_{12} > \frac{1}{2}$, which when we take the logarithm of both sides becomes

$$-\frac{1}{2\sigma^2}[(x-\mu_1)^T(x-\mu_1) - (x-\mu_2)^T(x-\mu_2)] > -\ln(2)\,,$$

which simplifies in exactly the same was as before to

$$(\mu_1-\mu_2)^T x > \frac{\mu_1^T\mu_1 - \mu_2^T\mu_2}{2} + \sigma^2\ln(2)\,.$$

Experiments at generating and classifying 10000 random feature vectors from each class using the previous expressions and then estimating the classification error probability can be found in the Matlab script `chap_2_prob_12.m`. For Part (a) we can also use the results of Problem 2.9 on Page 12 namely Equation 16 to *exactly* compute the error probability P_B and compare it to our empirically computed error probability. When we run that script we get the following results

```
empirical P_e=   0.215950; analytic Bayes P_e=   0.214598
```

showing that the empirical results are quite close to the theoretical. For Part (b) we can compute the empirical loss associated with using this classifier in the following way. Let L_{12} be the *number* of samples from the first class that are misclassified as belonging to the second class, L_{21} be the number of samples from the second class that are misclassified as belonging to the first class, and N be the total number of samples we classified. Then an empirical estimate of the expected loss $\hat{r}$ given by

$$\hat{r} = \frac{L_{12} + 0.5L_{21}}{N}\,.$$

Problem 2.13 (more classifier design)

Note that since for this problem since the functional form of the class conditional densities, $p(x|\omega_i)$, have changed, to construct the decision rule in terms of x as we did for Problem 2.12,

we would need to again simplify the likelihood ratio expression Equation 21. If all we care about is the classification of a point x we can skip these algebraic transformations and simply compute $l_{12} = \frac{p(x|\omega_1)}{p(x|\sigma_2)}$, directly and then compare this to the simplified right-hand-sides of Equation 21, which for Part (a) is 1 and for Part (b) is $\frac{1}{2}$. Again for Part (a) we can exactly compute the Bayes error rate using Equation 16. This procedure is implemented in Matlab script `chap_2_prob_13.m`. When we run that script we get

```
empirical P_e=   0.374650; analytic Bayes P_e=   0.373949
```

again showing that the empirical results are quite close to the theoretical.

Problem 2.14 (orthogonality of the decision hyperplane)

We want to show that the decision hyperplane at the point x_0 is tangent to the constant Mahalanobis distance hyperellipsoid $d_m(x, \mu_i) = c_i$. If we can show that the vector v defined by

$$v \equiv \left. \frac{\partial d_m(x, \mu_i)}{\partial x} \right|_{x=x_0} .$$

is orthogonal to all vectors in the decision hyperplane, then since v is normal to the surfaces $d_m(x, \mu_i) = c_i$, we will have a tangent decision hyperplane. The Mahalanobis distance between μ_i and a point x is given by

$$d_m(\mu_i, x) = \left((x - \mu_i)^T \Sigma^{-1} (x - \mu_i)\right)^{1/2} .$$

The gradient of $d_m(\mu_i, x)$ considered as a function of x is given by

$$\begin{aligned} \frac{\partial d_m(\mu_i, x)}{\partial x} &= \frac{1}{2} \left((x - \mu_i)^T \Sigma^{-1} (x - \mu_i)\right)^{-1/2} \frac{\partial}{\partial x} \left((x - \mu_i)^T \Sigma^{-1} (x - \mu_i)\right) \\ &= \frac{1}{2} (d_m^2)^{-1/2} \left(2\Sigma^{-1}(x - \mu_i)\right) \\ &= \frac{1}{d_m(\mu_i, x)} \Sigma^{-1}(x - \mu_i) . \end{aligned}$$

Consider this expression evaluated at $x = x_0$, we would have to evaluate $x_0 - \mu_i$. From the expression for x_0 this is

$$x_0 - \mu_i = \frac{1}{2}(-\mu_i + \mu_j) - \ln\left(\frac{P(\omega_i)}{P(\omega_j)}\right) \frac{\mu_i - \mu_j}{||\mu_i - \mu_j||_{\Sigma^{-1}}} ,$$

which is a vector proportional to $\mu_i - \mu_j$. Thus we see that for a point x on the decision hyperplane we have that

$$\begin{aligned} v^T(x - x_0) &= \frac{1}{d_m(\mu_i, x_0)} \left(\Sigma^{-1}(x_0 - \mu_i)\right)^T (x - x_0) \\ &\propto \left(\Sigma^{-1}(\mu_i - \mu_j)\right)^T (x - x_0) = 0 , \end{aligned}$$

since for the case of multidimensional Gaussians with equal non-diagonal covariance matrices, Σ, the points x on the decision hyperplanes are given by

$$g_{ij}(x) = \left(\Sigma^{-1}(\mu_i - \mu_j)\right)^T (x - x_0) = 0\,.$$

The decision hyperplane is also tangent to the surfaces $d_m(x, \mu_j) = c_j$. Since to show that we would then need to evaluate $\frac{1}{d_m(\mu_j,x)}\Sigma^{-1}(x - \mu_j)$ at $x = x_0$ and we would again find that $x_0 - \mu_j$ is again proportional to $\mu_i - \mu_j$.

Problem 2.15 (bounding the probability of error)

We are told to assume that

$$\begin{aligned} p(x|\omega_1) &\sim \mathcal{N}(\mu, \sigma^2) \\ p(x|\omega_2) &\sim \mathcal{U}(a,b) = \begin{cases} \frac{1}{b-a} & a < x < b \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

The minimum probability of error criterion is to classify a sample with feature x as a member of the class ω_1 if

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{P(\omega_2)}{P(\omega_1)}\,,$$

and classify x as from the class ω_2 otherwise. Since $p(x|\omega_2)$ is zero for some values of x we should write this expression as

$$p(x|\omega_1) > \frac{P(\omega_2)}{P(\omega_1)} p(x|\omega_2)\,.$$

Note that from the above if $x \notin [a, b]$ since $p(x|\omega_2) = 0$ the above expression will be true and we would classify that point as from class ω_1. It remains to determine how we would classify any points $x \in [a, b]$ as. Since $p(x|\omega_2)$ is a uniform distribution the above inequality is given by

$$p(x|\omega_1) > \frac{P(\omega_2)}{P(\omega_1)} \frac{1}{b-a}\,. \tag{22}$$

Given that the density $p(x|\omega_1)$ is a Gaussian we can find any values x_0 such that the above inequality is an equality. That is x_0 must satisfy

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x_0-\mu)^2}{\sigma^2}} = \frac{P(\omega_2)}{P(\omega_1)} \frac{1}{b-a}\,.$$

When we solve for x_0 we find

$$x_0 = \mu \pm \sigma \sqrt{-2 \ln \left(\frac{\sqrt{2\pi}\sigma}{b-a} \frac{P(\omega_2)}{P(\omega_1)} \right)}\,. \tag{23}$$

There are at most two real solutions for x_0 in the above expression which we denote by x_0^- and x_0^+, where $x_0^- < x_0^+$. Depending on the different possible values of x_0^- and x_0^+ we

will evaluate the error probability P_e. If neither x_0^- and x_0^+ are real then the classification decision regions in this case become

$$\begin{aligned}\mathcal{R}_1 &= (-\infty, a) \cup (b, +\infty) \\ \mathcal{R}_2 &= (a, b)\,.\end{aligned}$$

Then P_e is given by

$$\begin{aligned}P_e &= P(\omega_1)\int_{\mathcal{R}_2} p(x|\omega_1)dx + P(\omega_2)\int_{\mathcal{R}_1} p(x|\omega_2)dx \\ &= P(\omega_1)\int_a^b p(x|\omega_1)dx < \int_a^b p(x|\omega_1)dx\,, \qquad (24)\end{aligned}$$

since $\int_{\mathcal{R}_1} p(x|\omega_2)dx = 0$ for the $\mathcal{R}_1$ region.

Next consider the case where $x_0^\pm$ are both real. From the expression for x_0 given by Equation 23 this means that

$$-\ln\left(\frac{\sqrt{2\pi}\sigma}{b-a}\frac{P(\omega_2)}{P(\omega_1)}\right) > 0\,,$$

or

$$\frac{1}{\sqrt{2\pi}\sigma} > \frac{P(\omega_2)}{P(\omega_1)}\frac{1}{b-a}\,.$$

What we notice about this last expression is that it is exactly Equation 22 evaluated at $x = \mu$. Since $\mu \in (x_0^-, x_0^+)$ this means that all points in the range (x_0^-, x_0^+) are classified as belonging to the first class. Thus we have shown that in the case when $x_0^\pm$ are both real we have

$$\begin{aligned}\mathcal{R}_1 &= (-\infty, a) \cup (b, +\infty) \cup (x_0^-, x_0^+) \\ \mathcal{R}_2 &= (a, b) \setminus (x_0^-, x_0^+)\,.\end{aligned}$$

With these regions P_e is given by

$$\begin{aligned}P_e &= P(\omega_1)\int_{\mathcal{R}_2} p(x|\omega_1)dx + P(\omega_2)\int_{\mathcal{R}_1} p(x|\omega_2)dx \\ &= P(\omega_1)\int_{(a,b)\setminus(x_0^-,x_0^+)} p(x|\omega_1)dx + P(\omega_2)\int_{(a,b)\cap(x_0^-,x_0^+)} p(x|\omega_2)dx\,.\end{aligned}$$

Now for any x between x_0^- and x_0^+ we have argued using Equation 22 that $P(\omega_1)p(x|\omega_1) > P(\omega_2)p(x|\omega_2)$ and thus we can bound the second term above as

$$P(\omega_2)\int_{(a,b)\cap(x_0^-,x_0^+)} p(x|\omega_2)dx \le P(\omega_1)\int_{(a,b)\cap(x_0^-,x_0^+)} p(x|\omega_1)dx\,.$$

Thus the above expression for P_e is bounded as

$$\begin{aligned}P_e &\le P(\omega_1)\int_{(a,b)\setminus(x_0^-,x_0^+)} p(x|\omega_1)dx + P(\omega_1)\int_{(a,b)\cap(x_0^-,x_0^+)} p(x|\omega_1)dx \\ &= P(\omega_1)\int_a^b p(x|\omega_1)dx < \int_a^b p(x|\omega_1)dx\,.\end{aligned}$$

This last expression is exactly like the bound presented for P_e in Equation 24. The next step is to simplify the expression $\int_a^b p(x|\omega_1)dx$. Since $p(x|\omega_1)$ is $\mathcal{N}(\mu, \sigma^2)$ to evaluate this integral we make the change of variable from x to z defined as $z = \frac{x-\mu}{\sigma}$ to get

$$\begin{aligned}
\int_a^b p(x|\omega_1)dx &= \int_{\frac{a-\mu}{\sigma}}^{\frac{b-\mu}{\sigma}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} dz \\
&= \int_{-\infty}^{\frac{b-\mu}{\sigma}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} dz - \int_{-\infty}^{\frac{a-\mu}{\sigma}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} dz \\
&= G\left(\frac{b-\mu}{\sigma}\right) - G\left(\frac{a-\mu}{\sigma}\right),
\end{aligned}$$

the expression we were to show.

Problem 2.16 (the mean of the expression $\frac{\partial \ln(p(x;\theta))}{\partial \theta}$)

We compute this directly

$$\begin{aligned}
E\left[\frac{\partial \ln(p(x;\theta))}{\partial \theta}\right] &= E\left[\frac{1}{p(x;\theta)}\frac{\partial p(x;\theta)}{\partial \theta}\right] \\
&= \int \left(\frac{1}{p(x;\theta)}\frac{\partial p(x;\theta)}{\partial \theta}\right) p(x;\theta)dx = \int \frac{\partial p(x;\theta)}{\partial \theta} dx \\
&= \frac{\partial}{\partial \theta}\int p(x;\theta)dx = \frac{\partial}{\partial \theta} 1 = 0,
\end{aligned}$$

as claimed.

Problem 2.17 (the probability of flipping heads)

We have a likelihood (probability) for the N flips given by

$$P(X;q) = \prod_{i=1}^{N} q^{x_i}(1-q)^{1-x_i}.$$

The loglikelihood of this expression is then

$$L(q) \equiv \ln(P(X;q)) = \sum_{i=1}^{N} (x_i \ln(q) + (1-x_i)\ln(1-q)).$$

To find the ML estimate for q, we will maximize $\ln(P(X;q))$ as a function of q. To do this we compute

$$\frac{dL}{dq} = \sum_{i=1}^{N} \left(\frac{x_i}{q} - \frac{1-x_i}{1-q}\right) = 0.$$

When we solve for q in the above we find

$$q = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

Problem 2.18 (the Cramer-Rao bound)

When we consider the ML estimate of the mean μ of a Gaussian multidimensional random variable with known covariance matrix Σ we are lead to consider the loglikelihood $L(\mu)$ given by

$$L(\mu) = -\frac{N}{2}\ln((2\pi)^l|\Sigma|) - \frac{1}{2}\sum_{k=1}^{N}(x_k-\mu)^T\Sigma^{-1}(x_k-\mu)\,, \tag{25}$$

with $l = 1$ and $\Sigma = \sigma^2$ we have the loglikelihood, $L(\mu)$, given by

$$L(\mu) = -\frac{N}{2}\ln(2\pi\sigma^2) - \frac{1}{2}\sum_{k=1}^{N}\frac{(x_k-\mu)^2}{\sigma^2}\,.$$

To discuss the Cramer-Rao lower bound we begin with the assumption that we have a density for our samples x that is parametrized via the elements of a vector θ such that $x \sim p(x;\theta)$. We then consider any unbiased estimator of these density parameters, θ, denoted as $\hat{\theta}$. If the estimator is unbiased, the Cramer-Rao lower bound then gives us a lower bound on the variance of this estimator. The specific lower bound on the variance of $\hat{\theta}$ is given by forming the loglikelihood

$$L(\theta) \equiv \ln\left(\prod_{k=1}^{N} p(x_i;\theta)\right)\,,$$

and then taking the expectation over partial derivatives of this expression. Namely, construct the **Fisher Matrix** which has i,jth element given by

$$J_{ij} \equiv -E\left[\frac{\partial^2 L(\theta)}{\partial\theta_i\partial\theta_j}\right]\,.$$

Then the variance of the $\hat{\theta}_i$ must be larger than the value of J_{ii}^{-1}. As an equation this statement is that

$$E[(\hat{\theta}_i-\theta_i)^2] \geq J_{ii}^{-1}\,.$$

If we happen to find an estimator $\hat{\theta}_i$ where the above inequality is satisfied as an equality then our estimator is said to be *efficient*.

From the above discussion we see that we need to evaluate derivatives of L with respect to the parameters of the density. We find the first two derivatives of this expression with respect to μ are given by

$$\begin{aligned}\frac{\partial L}{\partial\mu} &= \frac{1}{\sigma^2}\sum_{k=1}^{N}(x_k-\mu)\\ \frac{\partial^2 L}{\partial\mu^2} &= \frac{1}{\sigma^2}\sum_{k=1}^{N}-1 = -\frac{N}{\sigma^2}\,.\end{aligned}$$

Thus we get

$$-E\left[\frac{\partial^2 L}{\partial\mu^2}\right] = \frac{N}{\sigma^2} = \frac{1}{\frac{\sigma^2}{N}} = \frac{1}{\mathrm{var}(\hat{\mu}_{\mathrm{ML}})}\,.$$

Since we have shown that $-E\left[\frac{\partial^2 L}{\partial \mu^2}\right] = \hat{\mu}_{\text{ML}}^{-1}$ the ML estimator for the mean $\hat{\mu}_{\text{ML}}$ is efficient.

Now consider the case where the unknown parameter of our density is the variance σ^2 then we have

$$\begin{aligned}\frac{\partial L}{\partial \sigma^2} &= -\frac{N}{2}\frac{1}{\sigma^2} + \frac{1}{2}\sum_{k=1}^{N}\frac{(x_k-\mu)^2}{(\sigma^2)^2} \\ \frac{\partial^2 L}{\partial (\sigma^2)^2} &= \frac{N}{2}\frac{1}{(\sigma^2)^2} - \sum_{k=1}^{N}\frac{(x_k-\mu)^2}{(\sigma^2)^3}\,.\end{aligned}$$

Recall that $E[(x_k-\mu)^2] = \sigma^2$ since $x_k \sim N(\mu, \sigma^2)$ and we can evaluate the expectation of the second derivative of L with respect to σ^2 as

$$E\left[\frac{\partial^2 L}{\partial (\sigma^2)^2}\right] = \frac{N}{2}\frac{1}{(\sigma^2)^2} - \sum_{k=1}^{N}\frac{\sigma^2}{(\sigma^2)^3} = -\frac{N}{2}\frac{1}{(\sigma^2)^2}\,.$$

Thus

$$-E\left[\frac{\partial^2 L}{\partial (\sigma^2)^2}\right] = \frac{N}{2}\frac{1}{(\sigma^2)^2} = \frac{1}{\frac{(\sigma^2)^2}{N/2}}\,.$$

To consider whether the ML estimator of the variance is efficient (i.e. satisfied the Cramer-Rao lower bound) we need to determine if the above expression is equal to

$$\frac{1}{\text{var}(\hat{\sigma^2}_{\text{ML}})}\,.$$

Then from Problem 2.19 below the ML estimator of σ^2 in the scalar Gaussian case is given by

$$\hat{\sigma^2}_{\text{ML}} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu}_{\text{ML}})^2\,,$$

where $\hat{\mu}_{\text{ML}}$ is the ML estimate of the mean or $\frac{1}{N}\sum_{k=1}^{N} x_k$. Then it can be shown [1] that $\hat{\sigma^2}_{\text{ML}}$ has a chi-squared distribution with $N-1$ degrees of freedom, in the sense that

$$\hat{\sigma^2}_{\text{ML}} \sim \frac{\sigma^2}{N}\chi^2_{N-1}$$

Thus since the expectation and variance of a chi-squared distribution with $N-1$ degrees of freedom is $N-1$ and $2(N-1)$ respectively we have that

$$\begin{aligned}E[\hat{\sigma^2}_{\text{ML}}] &= \frac{\sigma^2}{N}(N-1) = \frac{N-1}{N}\sigma^2 \\ \text{var}(\hat{\sigma^2}_{\text{ML}}) &= \frac{\sigma^4}{N^2}2(N-1) = \frac{2(N-1)}{N^2}\sigma^4\,.\end{aligned}$$

From the given expression for $\text{var}(\hat{\sigma^2}_{\text{ML}})$ we see that

$$\frac{1}{\text{var}(\hat{\sigma^2}_{\text{ML}})} \neq -E\left[\frac{\partial^2 L}{\partial (\sigma^2)^2}\right]\,,$$

and thus the ML estimate of σ^2 is *not* efficient.

[1] http:en.wikipedia.orgwikiNormal_distribution

Problem 2.19 (ML with the multidimensional Gaussian)

We recall that the probability density function for the multidimensional Gaussian is given by

$$p(x|\mu,\Sigma) = \frac{1}{(2\pi)^{l/2}|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(x-\mu)^t\Sigma^{-1}(x-\mu)\}\,.$$

So the loglikelihood L when given N samples from the above distribution is given by

$$L = \ln(\prod_{k=1}^{N} p(x_k|\mu,\Sigma)) = \sum_{k=1}^{N} \ln(p(x_k|\mu,\Sigma))\,.$$

The logarithm of the probability density function for a multidimensional Gaussian is given by

$$\ln(p(x_k|\mu,\Sigma)) = -\frac{1}{2}(x-\mu)^t\Sigma^{-1}(x-\mu) - \frac{1}{2}\ln((2\pi)^d|\Sigma|)\,,$$

so that the above becomes

$$L = -\frac{1}{2}\sum_{k=1}^{N}(x_k-\mu)^t\Sigma^{-1}(x_k-\mu) - \frac{N}{2}\ln((2\pi)^d|\Sigma|)\,.$$

To evaluate the maximum likelihood estimates for μ and Σ we would notionally take the derivative with respect to these two parameters, set the resulting expression equal equal to zero, and solve for the parameters. To do this requires some "vector" derivatives. Remembering that

$$\frac{\partial}{\partial \mathbf{x}}\mathbf{x}^t M\mathbf{x} = (M+M^t)\mathbf{x}\,,$$

we see that the derivative of L with respect to μ is given by

$$\frac{\partial L}{\partial \mu} = \sum_{k=1}^{N}\Sigma^{-1}(x_k-\mu) = 0\,,$$

since Σ^{-1} is a symmetric matrix. On multiplying by the covariance matrix Σ on both sides of the above we have

$$\sum_{k=1}^{N} x_k - \mu\cdot N = 0 \quad \text{or} \quad \mu = \frac{1}{N}\sum_{k=1}^{N} x_k\,.$$

So the maximum likelihood estimate of μ is just the sample mean as claimed. To evaluate the maximum likelihood estimate for Σ, we begin by instead computing the maximum likelihood estimate for Σ^{-1}. In terms of $M \equiv \Sigma^{-1}$ we have L given by

$$L(M) = -\frac{1}{2}\sum_{k=1}^{N}(x_k-\mu)^t M(x-\mu) - \frac{N}{2}\ln((2\pi)^d) + \frac{N}{2}\ln(|M|)\,.$$

To evaluate the derivative of the above it is helpful to recall two identities regarding matrix derivatives. The first involves the logarithm of the determinant and is

$$\frac{\partial \ln(|\mathbf{M}|)}{\partial M} = (\mathbf{M}^{-1})^t = (\mathbf{M}^t)^{-1} = \mathbf{M}^{-1}\,,$$

since M and M^{-1} are both symmetric. The second involves the matrix derivative of scalar form $\mathbf{x}^t\mathbf{M}\mathbf{x}$. We recall that

$$\frac{\partial}{\partial \mathbf{M}}(a^t\mathbf{M}b) = ab^t\,.$$

Using these two identities we have that the $\mathbf{M}$ derivative of L is given by

$$\frac{\partial L}{\partial M} = -\frac{1}{2}\sum_{k=1}^{N}(x_k-\mu)(x_k-\mu)^t + \frac{N}{2}\mathbf{M}^{-1}\,.$$

When we set this equal to zero and then solve for $\mathbf{M}^{-1}$ we get

$$\mathbf{M}^{-1} = \frac{1}{N}\sum_{k=1}^{N}(x_k-\mu)(x_k-\mu)^t\,.$$

Since $\mathbf{M}^{-1} = \Sigma$ and we evaluate μ at the maximum likelihood solution given above, the previous expression is what we were to show.

Problem 2.20 (the ML estimate in an Erlang distribution)

The likelihood function for θ given the N measurements x_i from the Erlang distribution is

$$P(X;\theta) = \prod_{i=1}^{N}\theta^2 x_i e^{-\theta x_i}u(x_i)\,.$$

Since $u(x_i) = 1$ for all i this factor is not explicitly needed. From this expression the loglikelihood of X is given by

$$\begin{aligned} L(\theta) &\equiv \ln(P(X;\theta)) = \sum_{i=1}^{N}(2\ln(\theta)+\ln(x_i)-\theta x_i) \\ &= 2\ln(\theta)N + \sum_{i=1}^{N}\ln(x_i) - \theta\sum_{i=1}^{N}x_i\,. \end{aligned}$$

To find this maximum of this expression we take the derivative with respect to θ, set the expression to zero, and solve for θ. We find

$$\frac{dL}{d\theta} = 0\,,$$

means

$$\frac{2N}{\theta} - \sum_{i=1}^{N}x_i = 0\,.$$

Which has a solution for θ given by

$$\theta = \frac{2N}{\sum_{i=1}^{N}x_i}\,,$$

as we were to show.

Problem 2.21 (the ML estimate occurs at a maximum)

When we consider the ML estimate of the mean μ of a Gaussian multidimensional random variable with known covariance matrix Σ we are lead to consider the loglikelihood $L(\mu)$ given by Equation 25. We have shown that the first derivative of $L(\mu)$ with respect to μ is given by

$$\frac{\partial L(\mu)}{\partial \mu} = \Sigma^{-1} \sum_{k=1}^{N} (x_k - \mu) = \Sigma^{-1} \sum_{k=1}^{N} x_k - N\Sigma^{-1}\mu \,.$$

The second derivative of $L(\mu)$ with respect to μ is then

$$\frac{\partial^2 L(\mu)}{\partial \mu^2} = -N\Sigma^{-1} \,.$$

Since the matrix Σ is positive definite, we have that Σ^{-1} is positive definite, and so $-N\Sigma^{-1}$ is negative definite. That the matrix second derivative is negative definite is the condition for the solution to $\frac{\partial L(\mu)}{\partial \mu} = 0$ to be a maximum of the objective function $L(\mu)$.

Problem 2.22 ($p(x|X)$ is normal with a given mean and covariance)

We assume that once we know the mean μ the sample x is drawn from $p(x|\mu) \sim \mathcal{N}(\mu, \sigma^2)$ and that the true mean μ is random itself and given by a random draw from another normal distribution $p(\mu) \sim \mathcal{N}(\mu_0, \sigma_0^2)$. Then using Bayes' rule we can compute what the a posteriori distribution of μ is *after* having observed the data set X as

$$p(\mu|X) = \frac{p(X|\mu)p(\mu)}{p(X)} \,.$$

As a function of μ the expression $p(X)$ is a constant. Assuming independence of the data samples x_i in X we conclude

$$p(X|\mu) = \prod_{k=1}^{N} p(x_k|\mu) \,.$$

Thus combining all of these expressions we see that

$$\begin{aligned}
p(\mu|X) &= \alpha p(\mu) \prod_{k=1}^{N} p(x_k|\mu) \\
&= \alpha \left(\frac{1}{\sqrt{2\pi}\sigma_0} \exp\left\{ -\frac{1}{2}\frac{(\mu-\mu_0)^2}{\sigma_0^2} \right\} \right) \prod_{k=1}^{N} \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{1}{2}\frac{(x_k-\mu)^2}{\sigma^2} \right\} \right) \\
&= \alpha' \exp\left[-\frac{1}{2} \left(\sum_{k=1}^{N} \left(\frac{\mu^2}{\sigma^2} - \frac{2x_k\mu}{\sigma^2} + \frac{x_k^2}{\sigma^2} \right) + \frac{\mu^2 - 2\mu\mu_0 + \mu_0^2}{\sigma_0^2} \right) \right] \\
&= \alpha' \exp\left[-\frac{1}{2} \left(\frac{\mu^2 N}{\sigma^2} - \frac{2\mu}{\sigma^2} \sum_{k=1}^{N} x_k + \frac{1}{\sigma^2} \sum_{k=1}^{N} x_k^2 + \frac{\mu^2}{\sigma_0^2} - 2\frac{\mu\mu_0}{\sigma_0^2} + \frac{\mu_0^2}{\sigma_0^2} \right) \right] \\
&= \alpha'' \exp\left[-\frac{1}{2} \left[\left(\frac{N}{\sigma^2} + \frac{1}{\sigma_0^2} \right) \mu^2 - 2 \left(\frac{1}{\sigma^2} \sum_{k=1}^{N} x_k + \frac{\mu_0}{\sigma_0^2} \right) \mu \right] \right] \,. \qquad (26)
\end{aligned}$$

Note that we have written $p(\mu)$ on the outside of the product terms since it should only appear *once* and not N times as might be inferred by had we written the product as $\prod_{k=1}^{N} p(\mu)p(x_k|\mu)$. From Equation 26 we see that the density $p(\mu|X)$ is *Gaussian*, due to the quadratic expression for μ in the argument of the exponential To find the values of the mean and variance of this Gaussian define σ_N^2 to be such that

$$\frac{1}{\sigma_N^2} = \frac{N}{\sigma^2} + \frac{1}{\sigma_0^2} \Rightarrow \sigma_N^2 = \frac{1}{\frac{N}{\sigma^2} + \frac{1}{\sigma_0^2}} = \frac{\sigma^2\sigma_0^2}{N\sigma_0^2 + \sigma^2}\,, \tag{27}$$

Now defining $\bar{x}$ as $\frac{1}{N}\sum_{k=1}^{N} x_k$ and take μ_N as the value that makes the ratio μ_N/σ_N^2 equal to the coefficient of μ in the argument of the exponential above or

$$\frac{\mu_N}{\sigma_N^2} = \frac{N\bar{x}}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}\,,$$

or solving for μ_N we get

$$\begin{aligned}
\mu_N &= \left(\frac{\sigma^2\sigma_0^2}{N\sigma_0^2 + \sigma^2}\right)\left(\frac{N\bar{x}}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}\right) = \frac{N\bar{x}\sigma_0^2 + \mu_0\sigma^2}{N\sigma_0^2 + \sigma^2} \\
&= \left(\frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2}\right)\bar{x} + \frac{\sigma^2}{N\sigma_0^2 + \sigma^2}\mu_0\,.
\end{aligned}$$

Once we have defined these variables we see that $p(\mu|X)$ is given by

$$\begin{aligned}
p(\mu|X) &= \alpha'' \exp\left\{-\frac{1}{2}\left(\frac{\mu^2}{\sigma_N^2} - \frac{2\mu_N\mu}{\sigma_N^2}\right)\right\} \\
&= \alpha''' \exp\left\{-\frac{1}{2}\left(\frac{\mu^2 - 2\mu_N\mu + \mu_N^2}{\sigma_N^2}\right)\right\} \\
&= \alpha''' \exp\left\{-\frac{1}{2}\frac{(\mu - \mu_N)^2}{\sigma_N^2}\right\}\,,
\end{aligned}$$

showing that $p(\mu|X)$ is a Gaussian with mean μ_N and variance σ_N^2. Next we are asked to compute $p(x|X)$ where using the above expression for $p(\mu|X)$ we find

$$\begin{aligned}
p(x|X) &= \int p(x,\mu|X)d\mu \\
&= \int p(x|\mu,X)p(\mu|X)d\mu = \int p(x|\mu)p(\mu|X)d\mu \\
&= \int_{-\infty}^{\infty}\left(\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma_N}e^{-\frac{1}{2}\frac{(x-\mu_N)^2}{\sigma_N^2}}\right)d\mu \\
&= \frac{1}{2\pi\sigma\sigma_N}\int_{-\infty}^{\infty}\exp\left\{-\frac{1}{2}\left[\frac{x^2 - 2x\mu + \mu^2}{\sigma^2} + \frac{\mu^2 - 2\mu\mu_N + \mu_N^2}{\sigma_N^2}\right]\right\}d\mu\,.
\end{aligned}$$

Grouping terms in the argument of the exponent gives the expression

$$-\frac{1}{2}\left[\left(\frac{1}{\sigma_N^2} + \frac{1}{\sigma^2}\right)\mu^2 - 2\left(\frac{x}{\sigma^2} + \frac{\mu_N}{\sigma_N^2}\right)\mu + \left(\frac{x^2}{\sigma^2} + \frac{\mu_N^2}{\sigma_N^2}\right)\right]\,.$$

In `prob_2_22_integration.nb` we integrate the above to get

$$\begin{aligned} p(x|X) &= \frac{1}{2\pi\sigma\sigma_N}\left(\sqrt{2\pi}\frac{1}{\sqrt{\frac{1}{\sigma^2}+\frac{1}{\sigma_N^2}}}e^{-\frac{1}{2}\frac{(x-\mu_N)^2}{\sigma^2+\sigma_N^2}}\right) \\ &= \frac{1}{\sqrt{2\pi}\sqrt{\sigma^2+\sigma_N^2}}e^{-\frac{1}{2}\frac{(x-\mu_N)^2}{\sigma^2+\sigma_N^2}}\,, \end{aligned}$$

or a Gaussian distribution with mean μ_N and variance $\sigma^2+\sigma_N^2$.

Problem 2.23 (the MAP estimate of μ for a Rayleigh distribution)

The likelihood of observing all N measurements x_i from a normal $\mathcal{N}(\mu,\sigma^2)$, where μ is from a Rayleigh distribution probability density function is given by

$$l(\mu) = \left(\prod_{i=1}^{N} p(x_i|\mu,\sigma^2)\right) p(\mu)\,,$$

where $p(\mu)$ is the Rayleigh density $p(\mu) = \frac{\mu\exp(-\mu^2/(2\sigma_\mu^2))}{\sigma_\mu^2}$. Then the loglikelihood becomes

$$\begin{aligned} L(\mu) &= \ln(l(\mu)) = \ln(p(\mu)) + \sum_{i=1}^{N}\ln(p(x_i|\mu,\sigma^2)) \\ &= \ln(\mu) - \frac{\mu^2}{2\sigma_\mu^2} - \ln(\sigma_\mu^2) + \sum_{i=1}^{N}\left[\ln(\frac{1}{\sqrt{2\pi}\sigma}) - \frac{(x_i-\mu)^2}{2\sigma^2}\right] \\ &= \ln(\mu) - \frac{\mu^2}{2\sigma_\mu^2} - \ln(\sigma_\mu^2) + N\ln(\frac{1}{\sqrt{2\pi}\sigma}) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2\,. \end{aligned}$$

Now to maximize $L(\mu)$ take the derivative with respect to μ, set the result equal to zero and solve for μ. We find

$$\frac{dL}{d\mu} = \frac{1}{\mu} - \frac{\mu}{\sigma_\mu^2} + \frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i-\mu) = 0\,.$$

or

$$\left(\frac{1}{\sigma_\mu^2}+\frac{N}{\sigma^2}\right)\mu^2 - \left(\frac{1}{\sigma^2}\right)\left(\frac{1}{N}\sum_{i=1}^{N}x_i\right)\mu - 1 = 0\,.$$

Defining the coefficient of μ^2 to be R and the coefficient of μ to be Z we have using the quadratic equation that the ML solution for μ is given by

$$\mu = \frac{Z\pm\sqrt{Z^2+4R}}{2R} = \frac{Z}{2R}\left(1\pm\sqrt{1+\frac{4R}{Z}}\right)\,,$$

the desired result since we must take the positive sign in the above expression since $\mu > 0$.

Problem 2.24 (ML with the lognormal distribution)

For the lognormal distribution the density is given by

$$p(x) = \frac{1}{\sigma x\sqrt{2\pi}} \exp\left(-\frac{(\ln(x)-\theta)^2}{2\sigma^2}\right) \quad x > 0\,, \tag{28}$$

so the loglikelihood of observing N specific samples x_i is given by

$$\begin{aligned} L(\theta) &= \sum_{i=1}^{N} \ln(p(x_i)) \\ &= -\sum_{i=1}^{N} \left(\ln(\sigma x_i\sqrt{2\pi}) + \frac{(\ln(x_i)-\theta)^2}{2\sigma^2}\right)\,. \end{aligned}$$

Then $\frac{dL}{d\theta} = 0$ is given by

$$\frac{dL}{d\theta} = -\sum_{i=1}^{N} \frac{(\ln(x_i)-\theta)}{\sigma^2} = 0\,.$$

Solving for θ we get

$$\theta = \frac{1}{N}\sum_{i=1}^{N} \ln(x_i)\,.$$

Problem 2.25 (maximum entropy estimation of $p(x)$ with known mean and variance)

We want to maximize $-\int p(x)\ln(p(x))dx$ subject to the following constraints

$$\begin{aligned} \int p(x)dx &= 1 \\ \int xp(x)dx &= \mu \\ \int (x-\mu)^2 p(x)dx &= \sigma^2\,. \end{aligned}$$

This is the same as minimizing the negative of the above entropy expression. Adding the constraints to form the Lagrangian, our constrained minimization problem becomes

$$\begin{aligned} H_L &= \int p(x)\ln(p(x))dx - \lambda_1\left(\int p(x)dx - 1\right) \\ &- \lambda_2\left(\int xp(x)dx - \mu\right) - \lambda_3\left(\int (x-\mu)^2 p(x)dx - \sigma^2\right)\,. \end{aligned}$$

Taking the p derivative of this expression and setting the result equal to zero gives

$$\begin{aligned} \frac{\partial H_L}{\partial p} &= \int \ln(p(x))dx + \int dx - \lambda_1\int dx \\ &- \lambda_2\int xdx - \lambda_3\int (x-\mu)^2 dx = 0\,. \end{aligned}$$

We next solve for the expression, $\int \ln(p(x))dx$, and then take the derivative of the resulting expression to get

$$\ln(p(x)) = -(1 - \lambda_1 - \lambda_2 x - \lambda_3(x-\mu)^2)\,.$$

Thus $p(x)$ is given by

$$p(x) = e^{-(1-\lambda_1-\lambda_2 x-\lambda_3(x-\mu)^2)}\,.$$

We now need to evaluate the Lagrangian multipliers λ_1, λ_2, and λ_3. To do this we use the three known constraints which we write in terms of the known functional form for our density $p(x)$ as

$$\begin{aligned}
\int e^{-(1-\lambda_1-\lambda_2 x-\lambda_3(x-\mu)^2)}dx &= 1 \\
\int x e^{-(1-\lambda_1-\lambda_2 x-\lambda_3(x-\mu)^2)}dx &= \mu \\
\int (x-\mu)^2 e^{-(1-\lambda_1-\lambda_2 x-\lambda_3(x-\mu)^2)}dx &= \sigma^2\,.
\end{aligned}$$

We next perform the integrations on the left-hand-side of the above expressions in the Mathematica file `prob_2_25.nb`. We then solve the resulting three equations for λ_1, λ_2, and λ_3. When we do that we find

$$\begin{aligned}
\lambda_1 &= \frac{1}{4}\ln\left(\frac{e^4}{4\pi^2\sigma^4}\right) = 1 - \frac{1}{2}\ln(2\pi\sigma^2) \\
\lambda_2 &= 0 \\
\lambda_3 &= -\frac{1}{2\sigma^2}\,.
\end{aligned}$$

Thus with these values for λ the form of $p(x)$ is

$$\begin{aligned}
p(x) &= e^{-(1-\lambda_1-\lambda_3(x-\mu)^2)} = e^{-\frac{1}{2}\ln(2\pi\sigma^2)} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \\
&= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\,,
\end{aligned}$$

or a Gaussian distribution, $\mathcal{N}(\mu, \sigma^2)$, as claimed.

Problem 2.26 (the derivation the EM algorithm)

Continuing the discussion in the book on the EM algorithm we will present the derivation of the algorithm in the case where the density of our samples x_k is given by a multidimensional Gaussian. That is we assume that

$$p(x_k|j;\theta) = N(x_k|\mu_j, C_j)\,.$$

Then following the book, for the E-step of the EM algorithm we take the expectation over the unobserved sample cluster labels which we denote as $P(j_k|x_k;\Theta(t))$ as

$$\begin{aligned}
Q(\Theta;\Theta(t)) &= \sum_{k=1}^{N}\sum_{j_k=1}^{J} P(j_k|x_k;\Theta(t))\ln(p(x_k|j_k,\theta)P_{j_k}) \\
&= \sum_{k=1}^{N}\sum_{j=1}^{J} P(j|x_k;\Theta(t))\ln(p(x_k|j,\theta)P_j)\,. \qquad (29)
\end{aligned}$$

From the assumed form for $p(x_k|j;\theta)$, its natural logarithm is given by

$$\ln(p(x_k|j;\theta)) = -\frac{1}{2}\ln((2\pi)^l|C_j|) - \frac{1}{2}(x_k-\mu_j)^T C_j^{-1}(x_k-\mu_j)\,.$$

With this expression and dropping constants that won't change the value of the subsequent maximization, for $Q(\Theta;\Theta(t))$ we get the following expression

$$\sum_{k=1}^{N}\sum_{j=1}^{J} P(j|x_k;\Theta(t))\left[-\frac{1}{2}\ln(|C_j|) - \frac{1}{2}(x_k-\mu_j)^T C_j^{-1}(x_k-\mu_j) + \ln(P_j)\right].$$

For the M-step we maximize over the parameters μ_j, C_j and P_j the above expression, subject to the constraint that $\sum_j P_j = 1$. The classical way to solve this maximization is using the method of Lagrange multipliers. In that method we would extend $Q(\Theta;\Theta(t))$, creating a new objective function $Q'(\Theta;\Theta(t))$, to include a Lagrange multiplier (denoted by λ) to enforce the constraint that $\sum_j P_j = 1$ as

$$\begin{aligned} Q'(\Theta;\Theta(t)) &= \sum_{k=1}^{N}\sum_{j=1}^{J}\left[P(j|x_k;\Theta(t))\ln(N(x_k|\mu_j,C_j)) + P(j|x_j;\Theta(t))\ln(P_j)\right] \\ &- \lambda\left(\sum_{j=1}^{J} P_j - 1\right). \end{aligned} \tag{30}$$

We then proceed to maximize this expression by taking derivatives with respect to the variables μ_j, C_j, and P_j, setting the resulting expressions equal to zero, and solving the resulting equations for them. We begin by taking $\frac{\partial}{\partial\mu_j}$ of $Q'(\Theta;\Theta(t))$. We find

$$\frac{\partial}{\partial\mu_j}Q'(\Theta;\Theta(t)) = \sum_{k=1}^{N} P(j|x_k;\Theta(t))\left(\frac{1}{N(x_k|\mu_j,C_j)}\right)\left(\frac{\partial}{\partial\mu_j}N(x_k|\mu_j,C_j)\right).$$

The derivative required in the above is given by

$$\begin{aligned} \frac{\partial}{\partial\mu_j}N(x_k|\mu_j,C_j) &= N(x_k|\mu_j,C_j)\frac{\partial}{\partial\mu_j}\left(-\frac{1}{2}(x_k-\mu_j)^T C_j^{-1}(x_k-\mu_j)\right) \\ &= \frac{1}{2}N(x_k|\mu_j,C_j)\left(C_j^{-1}+C_j^{-T}\right)(x_k-\mu_j) \\ &= N(x_k|\mu_j,C_j)C_j^{-1}(x_k-\mu_j)\,. \end{aligned} \tag{31}$$

Thus

$$\frac{\partial}{\partial\mu_j}Q'(\Theta;\Theta(t)) = \sum_{k=1}^{N} P(j|x_k;\Theta(t))C_j^{-1}(x_k-\mu_j)\,. \tag{32}$$

Setting this expression equal to zero and solving for μ_j we have

$$\mu_j = \frac{\sum_{k=1}^{N} P(j|x_k;\Theta(t))x_k}{\sum_{k=1}^{N} P(j|x_k;\Theta(t))}\,. \tag{33}$$

Next we take the derivative of $Q'(\Theta;\Theta(t))$ with respect to C_j. Which we will evaluate using the chain rule transforming the derivative with respect to C_j into one with respect to C_j^{-1}. We have

$$\frac{\partial}{\partial C_j}Q'(\Theta;\Theta(t)) = \frac{\partial}{\partial {C_j}^{-1}}Q'(\Theta;\Theta(t))\,\frac{\partial {C_j}^{-1}}{\partial C_j}\,.$$

Thus if $\frac{\partial}{\partial C_j^{-1}} Q'(\Theta;\Theta(t)) = 0$, we have that $\frac{\partial}{\partial C_j} Q'(\Theta;\Theta(t)) = 0$ also. From this we can look for zeros of the derivative by looking for values of C_j where the derivative of the *inverse* of C_j vanishes. Taking the derivative of $Q'(\Theta;\Theta(t))$ with respect to ${C_j}^{-1}$ we find

$$\begin{aligned}\frac{\partial}{\partial {C_j}^{-1}} Q'(\Theta;\Theta(t)) &= \sum_{k=1}^{N} P(j|x_k;\Theta(t)) \frac{\partial}{\partial {C_j}^{-1}} \ln(N(x_k|\mu_j, C_j)) \\ &= \sum_{k=1}^{N} P(j|x_k;\Theta(t)) \left(\frac{1}{N(x_k|\mu_j, C_j)}\right) \frac{\partial}{\partial {C_j}^{-1}} N(x_k|\mu_j, C_j)\,.\end{aligned}$$

From which we see that as a sub problem we need to compute $\frac{\partial}{\partial {C_j}^{-1}} N(x_k|\mu_j, C_j)$, which we now do

$$\begin{aligned}\frac{\partial}{\partial {C_j}^{-1}} N(x_k|\mu_j, C_j) &= \frac{\partial}{\partial {C_j}^{-1}} \left(\frac{1}{(2\pi)^l |C_j|^{1/2}} \exp\left\{-\frac{1}{2}(x_k-\mu_j)^T {C_j}^{-1}(x_k-\mu_j)\right\}\right) \\ &= \frac{1}{(2\pi)^l} \frac{\partial}{\partial {C_j}^{-1}} \left(\frac{1}{|C_j|^{1/2}}\right) \exp\left\{-\frac{1}{2}(x_k-\mu_j)^T {C_j}^{-1}(x_k-\mu_j)\right\} \\ &+ \frac{1}{(2\pi)^l} \frac{1}{|C_j|^{1/2}} \frac{\partial}{\partial {C_j}^{-1}} \exp\left\{-\frac{1}{2}(x_k-\mu_j)^T {C_j}^{-1}(x_k-\mu_j)\right\}\,,\end{aligned}$$

using the product rule. To evaluate the first derivative in the above we note that

$$\begin{aligned}\frac{\partial}{\partial {C_j}^{-1}} \left(\frac{1}{|C_j|^{1/2}}\right) &= \frac{\partial}{\partial {C_j}^{-1}} |{C_j}^{-1}|^{1/2} \\ &= \frac{1}{2} |{C_j}^{-1}|^{-1/2} \frac{\partial}{\partial {C_j}^{-1}} |{C_j}^{-1}|\,,\end{aligned}$$

but using the following matrix derivative of a determinant identity

$$\frac{\partial}{\partial \mathbf{X}} |\mathbf{AXB}| = |\mathbf{AXB}|(\mathbf{X}^{-1})^T = |\mathbf{AXB}|(\mathbf{X}^T)^{-1}\,, \tag{34}$$

with $A = B = I$ we have $\frac{\partial}{\partial X}|X| = |X|(X^{-1})^T$ and the derivative $\frac{\partial}{\partial {C_j}^{-1}} \left(\frac{1}{|C_j|^{1/2}}\right)$ becomes

$$\begin{aligned}\frac{\partial}{\partial {C_j}^{-1}} \left(\frac{1}{|C_j|^{1/2}}\right) &= \frac{1}{2} |{C_j}^{-1}|^{-1/2} |{C_j}^{-1}| {C_j}^T \\ &= \frac{1}{2} \frac{1}{|C_j|^{1/2}} C_j\,.\end{aligned}$$

Next using the matrix derivative of an inner product is given by

$$\frac{\partial}{\partial \mathbf{X}} (\mathbf{a}^T \mathbf{X} \mathbf{b}) = \mathbf{a}\mathbf{b}^T\,, \tag{35}$$

we have the derivative of the inner product expression

$$\frac{\partial}{\partial {C_j}^{-1}} \left\{-\frac{1}{2}(x_k-\mu_j)^T {C_j}^{-1}(x_k-\mu_j)\right\} = -\frac{1}{2}(x_k-\mu_j)(x_k-\mu_j)^T\,.$$

Putting everything together we find that

$$\begin{aligned}\frac{\partial}{\partial {C_j}^{-1}} N(x_k|\mu_j, C_j) &= \frac{1}{2}\frac{1}{(2\pi)^l}\frac{1}{|C_j|^{1/2}}\exp\left\{-\frac{1}{2}(x_k-\mu_j)^T {C_j}^{-1}(x_k-\mu_j)\right\} C_j \\ &- \frac{1}{2}N(x_k|\mu_j, C_j)(x_k-\mu_j)^T(x_k-\mu_j) \\ &= \frac{1}{2}N(x_k|\mu_j, C_j)\left(C_j - (x_k-\mu_j)(x_k-\mu_j)^T\right) . \end{aligned} \tag{36}$$

So combining these subproblems we finally find

$$\frac{\partial}{\partial {C_j}^{-1}} \ln(N(x_k|\mu_j, C_j)) = \frac{1}{2}\left(C_j - (x_k-\mu_j)(x_k-\mu_j)^T\right) . \tag{37}$$

Using this in the expression for $\frac{\partial}{\partial {C_j}^{-1}} Q'(\Theta;\Theta(t)) = 0$, we find the equation

$$\sum_{k=1}^{N} P(j|x_k;\Theta(t))C_j - \sum_{k=1}^{N} P(j|x_k;\Theta(t))(x_k-\mu_j)(x_k-\mu_j)^T = 0 .$$

Which when we solve for C_j we find

$$C_j = \frac{\sum_{k=1}^{N} P(j|x_k;\Theta(t))(x_k-\mu_j)(x_k-\mu_j)^T}{\sum_{k=1}^{N} P(j|x_k;\Theta(t))} . \tag{38}$$

Warning: The above has μ_j meaning the old value of μ_j rather than $\mu_j(t+1)$ the newly computed value via Equation 33. I'm a bit unclear hear as to whether or not this matters, is a typo, or something else. If anyone has any information on this please contact me. Chapter 14 of this book also discusses the expectation maximization algorithm and has an equivalent formulation to the one above. In situations like this if we replace μ_j with $\mu_j(t+1)$ we get faster convergence.

To complete a full maximization of $Q'(\Theta;\Theta(t))$ with we still need to determine P_j the priori probabilities of the k-th cluster. Setting $\frac{\partial Q'(\Theta;\Theta(t))}{\partial P_j} = 0$ gives

$$\sum_{k=1}^{N} \frac{P(j|x_k;\Theta(t))}{P_j} - \lambda = 0 ,$$

or

$$\lambda P_j = \sum_{k=1}^{N} P(j|x_k;\Theta(t)) .$$

Summing this equation over j for $j = 1$ to J since $\sum_{j=1}^{J} P_j = 1$ we have

$$\lambda = \sum_{j=1}^{J}\sum_{j=1}^{N} P(j|x_k;\Theta(t)) .$$

This can be simplified by observing that

$$\lambda = \sum_{j=1}^{J}\sum_{k=1}^{N} P(j|x_k;\Theta(t)) = \sum_{k=1}^{N}\sum_{j=1}^{J} P(j|x_k;\Theta(t)) = \sum_{k=1}^{N} 1 = N .$$

Where we have used the fact that $P(j|x_k;\Theta(t))$ is a probability that the sample x_k is from cluster j. Since there are $1, 2, \ldots, J$ clusters summing this probability gives one. Thus

$$P_j = \frac{1}{N}\sum_{k=1}^{N} P(j|x_k;\Theta(t))\,. \tag{39}$$

Combining this expression with Equations 33 and 38 gives the EM algorithm.

Problem 2.27 (consistency of the counting density)

We are told that k is distributed as a binomial random variable with parameters (P, N). This means that the probability we observe the value of k samples in our interval of length h after N trials is given by

$$p(k) = \left(\begin{array}{c} N \\ k \end{array} \right) P^k (1-P)^{N-k} \quad \text{for} \quad 0 \leq k \leq N\,.$$

We desire to estimate the probability of success, P, from the measurement k via the ratio $\frac{k}{N}$. Lets compute the expectation and variance of this estimate of P. The expectation is given by

$$E\left[\frac{k}{N}|P\right] = \frac{1}{N}E[k|P]\,.$$

Now since k is drawn from a binomial random variable with parameters (N, P), the expectation of k is PN, from which we see that the above equals P (showing that our estimator of P is unbiased). To study the conditional variance of our error (defined as $e = \hat{P} - P$) consider

$$\begin{aligned} \sigma_e^2(P) &= E[(e - E[e])^2|P] \\ &= E[\left(\frac{1}{N}k - P\right)^2|P] = \frac{1}{N^2}E[(k-NP)^2|P] \\ &= \frac{1}{N^2}(NP(1-P)) = \frac{P(1-P)}{N}\,. \end{aligned} \tag{40}$$

In the above we have used the result that the variance of a binomial random variable with parameters (N, P) is $NP(1-P)$. Thus we have shown that the estimator $\frac{k}{N}$ is asymptotically consistent.

Problem 2.28 (experiments with the EM algorithm)

See the MATLAB file `chap_2_prob_28.m` for an implementation of this problem. When that code is run we get the following output

```
mu_true =   1.000000;   3.000000;   2.000000;
```

```
mu_j    =   1.016347;   2.993095;   2.040666;

s2_true =   0.100000;   0.100000;   0.200000;
sigma2_j=   0.096464;   0.106047;   0.256857;

p_true  =   0.250000;   0.500000;   0.250000;
P_j     =   0.239808;   0.499241;   0.260951;
```

Note that when we run the EM algorithm, and it converges to the true solution, the actual ordering of the elements in the estimated vectors holding the estimated values of μ, σ^2, and P_j does not have to match the ordering of the "truth". Thus in the above output we explicitly permute the order of the estimated results to make the estimates line up with the true values.

Problem 2.30 (nearest neighbors classification)

See the MATLAB file `chap_2_prob_30.m` for an implementation of this problem. When that code is run and we classify the test samples using the nearest neighbor classifier and for various numbers of nearest neighbors we get the following probability of error output

```
P_e  1NN=   0.360000;
P_e  2NN=   0.320000;
P_e  3NN=   0.300000;
P_e  4NN=   0.320000;
P_e  5NN=   0.310000;
P_e  6NN=   0.330000;
P_e  7NN=   0.310000;
P_e  8NN=   0.300000;
P_e  9NN=   0.260000;
P_e 10NN=   0.320000;
P_e 11NN=   0.310000;
P_B=   0.214598
```

It can be shown that

$$P_{3NN} \approx P_B + 3P_B^2 \,, \tag{41}$$

and thus since $P_B \approx 0.21$ we see that $P_{3NN} \approx 0.35275$ using the above formula. This value is in the same ballpark as the empirical value obtained above.

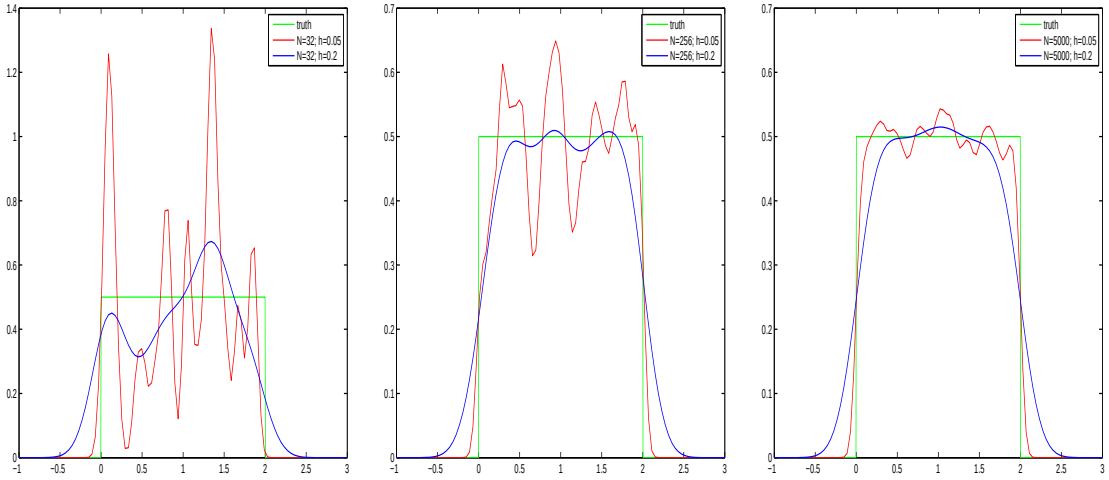

Figure 1: **Left:** Parzen window probability density estimation with $N = 32$ points. **Center:** Parzen window probability density estimation with $N = 256$ points. **Right:** Parzen window probability density estimation with $N = 5000$ points.

Problem 2.31 (Parzen window density estimation)

Parzen window density estimation with $\mathcal{N}(0, 1)$ for a kernel function, ϕ, means that we take $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ and then estimate our density $p(x)$ for l dimensional features vectors using

$$\hat{p}(x) = \frac{1}{h^l} \left(\frac{1}{N} \sum_{i=1}^{N} \phi \left(\frac{x_i - x}{h} \right) \right) . \tag{42}$$

In this problem l, the dimension of the feature vectors, is 1. See the MATLAB file `chap_2_prob_31.m` for an implementation of this problem. When that code is run we get the plots shown in Figure 1. Each plot is the density estimate based on N points where N is 32, 256, or 5000 and for h given by 0.05 and 0.2.

Problem 2.32 (k nearest neighbor density estimation)

For k-nearest neighbor density estimation we approximate the true density $p(x)$ using samples via

$$\hat{p}(x) = \frac{k}{NV(x)} , \tag{43}$$

where in this expression k and N are fixed and $V(x)$ is the volume of a sphere around the point x such that it contains the k nearest neighbors. For 1-dimensional densities this "volume" is a length. Thus the procedure we implement when given a point x where we want to evaluate the empirical density is to

1. Find the k nearest neighbors around the point x.

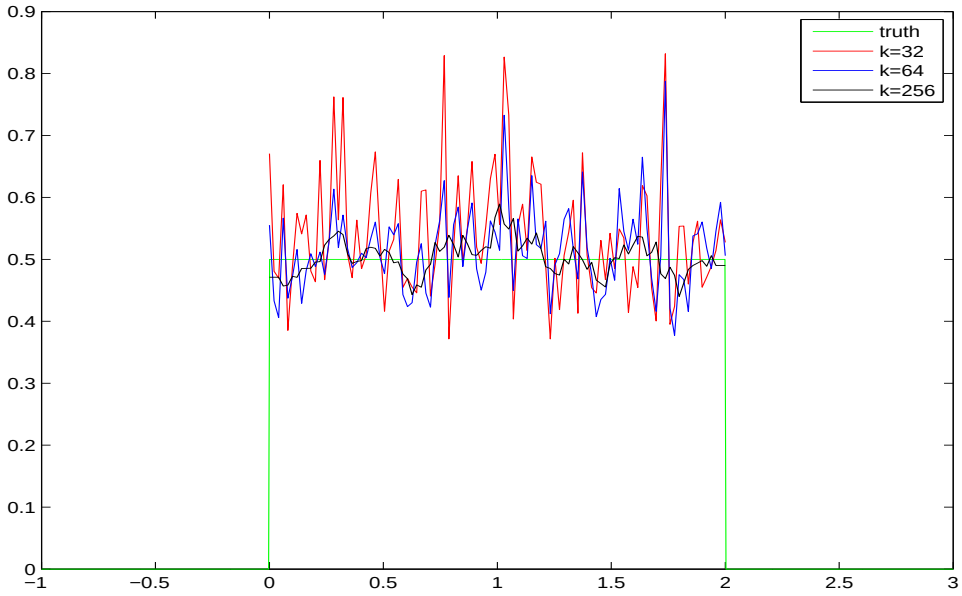

Figure 2: k nearest neighbor density estimation with $k = 32$, 64, and 256 points.

2. Compute $V(x)$ as the length between the largest and smallest points in the above set.

See the MATLAB file `chap_2_prob_32.m` for an implementation of this problem. When that code is run we get the plots shown in Figure 2. Note that these density estimates seems very noisy and this noise decreases as we take more and more neighbors.

Linear Classifiers

Notes on the text

Notes on Linear Discriminant Functions: the derivation of d

In two-dimensions, our weight vector and feature vectors have two components so $\mathbf{w}^T = (w_1, w_2)$ and our discriminant function $g(\mathbf{x})$ is given by

$$g(\mathbf{x}) = g(x_1, x_2) = w_1 x_1 + w_2 x_2 + w_0 \,.$$

Then $g(\mathbf{x}) = 0$ is a line that will intersect the x_1 axis at

$$x_2 = 0 \Rightarrow w_1 x_1 + w_0 = 0 \quad \text{or} \quad x_1 = -\frac{w_0}{w_1} \,,$$

and the x_2 axis at

$$x_1 = 0 \Rightarrow w_2 x_2 + w_0 = 0 \quad \text{or} \quad x_2 = -\frac{w_0}{w_2} \,.$$

Plotting a line that goes though the points $(0, -\frac{w_0}{w_2})$ and $(-\frac{w_0}{w_1}, 0)$ and assuming $w_0 < 0$ gives Figure 3.1 presented in the book. We now want to derive the expressions for d and z given in this section. To do that lets denote P be the point on the decision line that is closest to the origin and then d be the distance from the origin to this point P. Since P is the closest point to $(0, 0)$ the vector from $(0, 0)$ to P must be orthogonal to the decision line i.e. parallel to the vector $\mathbf{w}^T = (w_1, w_2)$. Thus we seek to determine the value of d such that the point that is d away from $(0, 0)$ and in the direction of of w i.e.

$$d\hat{\mathbf{w}} = d\left(\frac{w_1}{\sqrt{w_1^2 + w_2^2}}, \frac{w_2}{\sqrt{w_1^2 + w_2^2}} \right) \,,$$

is on the discriminant line $g(x_1, x_2) = 0$. When we put in the above two components for this point into $g(x_1, x_2) = 0$ we get that

$$\frac{dw_1^2}{\sqrt{w_1^2 + w_2^2}} + \frac{dw_2^2}{\sqrt{w_1^2 + w_2^2}} + w_0 = 0 \,.$$

When we solve for d in the above expression we get

$$d = -\frac{w_0}{\sqrt{w_1^2 + w_2^2}} \,.$$

Since we are assuming that $w_0 < 0$ we see that d can also be written as

$$d = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}} \,. \tag{44}$$

We can also obtain this formula using similar triangles. If we note that the "big" triangle with vertices's $(0, 0)$, $\left(-\frac{w_0}{w_1}, 0\right)$, $\left(-\frac{w_0}{w_2}, 0\right)$ is similar to the "small" triangle with vertices's P,

$\left(-\frac{w_0}{w_1}, 0\right)$, and $(0,0)$ in that they are both right triangles and have a common acute angle (with vertex at $\left(-\frac{w_0}{w_1}, 0\right)$). The ratio of the hypotenuse to leg of the triangle opposite the common acute angle must be equal in both triangle or

$$\frac{-\frac{w_0}{w_1}}{d} = \frac{\sqrt{\left(0+\frac{w_0}{w_1}\right)^2 + \left(0+\frac{w_0}{w_2}\right)^2}}{-\frac{w_0}{w_2}} .$$

When we solve this expression for d we get

$$d = \frac{w_0^2}{w_1 w_2} \frac{1}{\sqrt{\left(\frac{w_0}{w_1}\right)^2 + \left(\frac{w_0}{w_2}\right)^2}} = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}} ,$$

the same expression as before.

Notes on Linear Discriminant Functions: the derivation of z

We now seek to derive the expression for z, the distance from any point not on the decision surface to the decision surface. Let $\mathbf{x}$ be a point *not* on the decision hyperplane and then define z to be the distance from $\mathbf{x}$ to the closest point on the decision line. In the earlier part of these notes we derived the expression for z when the point not on the decision line was the point $(0,0)$. Using this it might be easier to compute the value of z if we first translate the decision line so that it passes thought the origin. To do that we subtract $d\hat{w}$ from every point in the $\mathbb{R}^2$ plane, where

$$d = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}} \quad \text{and} \quad \hat{w} = \left(\frac{w_1}{\sqrt{w_1^2 + w_2^2}}, \frac{w_2}{\sqrt{w_1^1 + w_2^2}} \right) .$$

Then the new coordinates, $\tilde{x}$, are then given by

$$\tilde{x} = x - d\hat{w} .$$

In the translated space where our decision surface passes though the origin the points $\tilde{x}$ can be decomposed into a component in the direction of $\hat{w}$ and in a direction perpendicular to $\hat{w}$ which we denote $\hat{w}_\perp$. The vector $\hat{w}_\perp$ has components that are related to tho of $\hat{w}$ as

$$\hat{w}_\perp = \left(-\frac{w_2}{\sqrt{w_1^2 + w_2^2}}, \frac{w_1}{\sqrt{w_1^1 + w_2^2}} \right) .$$

Then given these two vectors we can decompose $\tilde{x}$ as

$$\tilde{x} = (\hat{w}^T \tilde{x})\hat{w} + (\hat{w}_\perp^T \tilde{x})\hat{w}_\perp .$$

Then the distance from a point $\tilde{x}^*$ to the line $w^T\tilde{x} = 0$ or z will be the absolute value of the coefficient of the vector $\hat{w}$ above and (assuming it is positive meaning that $\tilde{x}^*$ is "to the right" of the decision line) we find

$$z = \hat{w}^T \tilde{x}^* .$$

In terms of the original variables we have z given by

$$\begin{aligned} z &= \hat{w}^T(x^* - d\hat{w}) = \hat{w}^T x^* - d \\ &= \frac{w_1 x_1^* + w_2 x_2^*}{\sqrt{w_1^2 + w_2^2}} - \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}} \,. \end{aligned}$$

If $w_0 < 0$ then $|w_0| = -w_0$ and the above becomes

$$z = \frac{w_1 x_1^* + w_2 x_2^* + w_0}{\sqrt{w_1^2 + w_2^2}} = \frac{g(x)}{\sqrt{w_1^2 + w_2^2}} \,.$$

Which is the expression we wanted to show.

Notes on the Perceptron Algorithm

If $x \in \omega_1$ and x is misclassified by the perceptron algorithm then $w^T x < 0$ so if we take $\delta_x = -1$ then the product $\delta_x(w^T x) > 0$. If $x \in \omega_2$ and is misclassified then the product $w^T x > 0$ so if we take $\delta_x = +1$ then $\delta_x(w^T x) > 0$. In all cases, where a sample x is misclassified, each term in the perceptron cost function

$$J(w) = \sum_{x \in Y} \delta_x w^T x \,, \tag{45}$$

is positive.

Notes on the Proof of the Perceptron Algorithm Convergence

When we begin with the gradient decent algorithm applied to the perceptron cost function $J(w)$ given by Equation 45. This algorithm is

$$w(t+1) = w(t) - \rho_t \sum_{x \in Y} \delta_x x \,,$$

where Y is the set samples misclassified by the current weight vector $w(t)$. Since the set Y depends on the current weight vector $w(t)$ we could indicate this with the notation $Y(t)$ if desired. We would like to now show that this algorithm converges to a vector that is parallel to the optimal vector w^*. A vector parallel to w^* is any vector of the form αw^* and this is what we subtract from $w(t+1)$ above to get

$$w(t+1) - \alpha w^* = w(t) - \alpha w^* - \rho_t \sum_{x \in Y} \delta_x x \,.$$

Then squaring both sides of the above we get

$$||w(t+1) - \alpha w^*||^2 = ||w(t) - \alpha w^*||^2 + \rho_t^2 || \sum_{x \in Y} \delta_x x||^2 - 2\rho_t \sum_{x \in Y} \delta_x (w(t) - \alpha w^*)^T x \,.$$

Since the vector $w(t)$ is not optimal it will misclassify some sample points. Thus the negative of the perceptron cost function $J(w)$ or $-\sum_{x\in Y}\delta_x w(t)^T x$ is a negative number. This gives the upper bound on $||w(t+1)-\alpha w^*||^2$ of

$$||w(t+1)-\alpha w^*||^2 \leq ||w(t)-\alpha w^*||^2 + \rho_t^2||\sum_{x\in Y}\delta_x x||^2 + 2\rho_t\alpha\sum_{x\in Y}\delta_x w^{*T}x\,.$$

Now since the expression only $||\sum_{x\in Y}\delta_x x||^2$ depends on the training data and not on the algorithm used to compute $w(t)$ we can introduce β^2 such that

$$\beta^2 \equiv \max_{Y'}||\sum_{x\in Y'}\delta_x x||^2\,,$$

thus β^2 is largest possible value for the given sum. At this point we have

$$||w(t+1)-\alpha w^*||^2 \leq ||w(t)-\alpha w^*||^2 + \rho_t^2\beta^2 + 2\rho_t\alpha\sum_{x\in Y}\delta_x w^{*T}x\,.$$

Since w^* is a solution vector from how x is classified and the definition of δ_x as discussed on Page 39 we know that $\sum_{x\in Y}\delta_x w^{*T}x < 0$, since each term in the sum is negative. As with the introduction of β^2 the above sum is over training points so we can introduce γ as

$$\gamma \equiv \max_{Y'}\sum_{x\in Y'}\delta_x w^{*T}x < 0\,.$$

For any fixed set Y we then have

$$\sum_{x\in Y}\delta_x w^{*T}x < \gamma < 0\,,$$

by the definition of γ. Now since $\gamma < 0$ we can write $\gamma = -|\gamma|$ and thus have

$$\sum_{x\in Y}\delta_x w^{*T}x < \gamma = -|\gamma|\,,$$

Using this result we thus are able to bound $||w(t+1)-\alpha w^*||^2$ as

$$||w(t+1)-\alpha w^*||^2 \leq ||w(t)-\alpha w^*||^2 + \rho_t^2\beta^2 - 2\rho_t\alpha|\gamma|\,. \tag{46}$$

Up to this point we have been studying the distance between $w(t)$ and an arbitrary vector αw^* that is parallel to w^*. Lets now consider how the distance between $w(t)$ and αw^* behaves when $\alpha = \frac{\beta^2}{2|\gamma|}$. Using that value in the above expression we find that

$$\rho_t^2\beta^2 - 2\rho_t\alpha|\gamma| = \rho_t^2\beta^2 - \rho_t\beta^2 = \beta^2(\rho_t^2-\rho_t)\,,$$

and the bound above become

$$||w(t+1)-\alpha w^*||^2 \leq ||w(t)-\alpha w^*||^2 + \beta^2(\rho_t^2-\rho_t)\,.$$

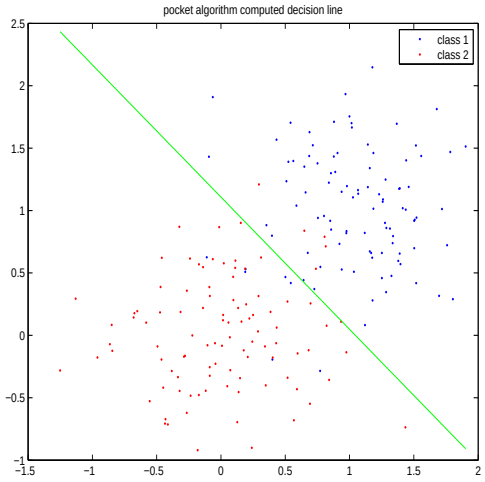

Figure 3: The decision surface produced when running the script `pocket_algorithm.m`. Note that this data is not linearly separable.

Writing out these expressions for $t = 0, 1, 2, \ldots$ gives

$$\begin{aligned}
||w(1) - \alpha w^*||^2 &\leq ||w(0) - \alpha w^*||^2 + \beta^2(\rho_0^2 - \rho_0) \\
||w(2) - \alpha w^*||^2 &\leq ||w(1) - \alpha w^*||^2 + \beta^2(\rho_1^2 - \rho_1) \\
&\leq ||w(0) - \alpha w^*||^2 + \beta^2(\rho_0^2 - \rho_0) + \beta^2(\rho_1^2 - \rho_1) \\
&= ||w(0) - \alpha w^*||^2 + \beta^2 \left(\sum_{k=0}^{1} \rho_k^2 - \sum_{k=0}^{1} \rho_k \right) \\
||w(3) - \alpha w^*||^2 &\leq ||w(2) - \alpha w^*||^2 + \beta^2(\rho_2^2 - \rho_2) \\
&\leq ||w(0) - \alpha w^*||^2 + \beta^2 \left(\sum_{k=0}^{1} \rho_k^2 - \sum_{k=0}^{1} \rho_k \right) + \beta^2(\rho_2^2 - \rho_2) \\
&\leq ||w(0) - \alpha w^*||^2 + \beta^2 \left(\sum_{k=0}^{2} \rho_k^2 - \sum_{k=0}^{2} \rho_k \right) \\
&\vdots \\
||w(t+1) - \alpha w^*||^2 &\leq ||w(0) - \alpha w^*||^2 + \beta^2 \left(\sum_{k=0}^{t} \rho_k^2 - \sum_{k=0}^{t} \rho_k \right) . \qquad (47)
\end{aligned}$$

Notes on the Pocket algorithm

See the MATLAB script `pocket_algorithm.m` for code that implements the Pocket Algorithm. An example decision surface when this script is run is given in Figure 3.

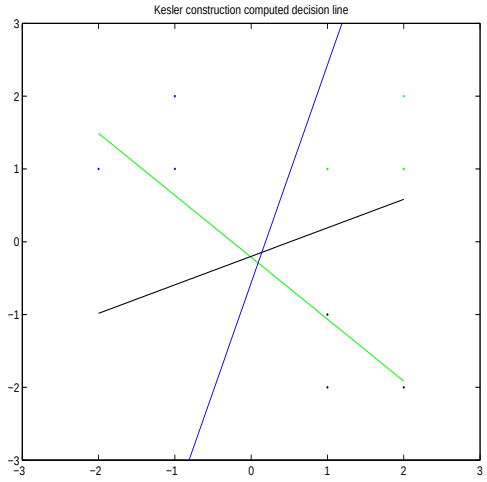

Figure 4: The three decision lines produced when running the script `dup_example_3_2.m`.

Duplication of Example 3.2 (Kesler's construction)

See the MATLAB script `dup_example_3_2.m` for code that duplicates the numerical results from this example. When this code is run it produces a plot like that shown in Figure 4. The previous command also verifies that $w_j^T x_i > w_k^T x_i$, when $x_i \in \omega_j$ by computing these inner products. We find

```
class=   1; w1^T x=  11.4449; w2^T x=  -4.5025; w3^T x=  -2.6150
class=   1; w1^T x=  21.7109; w2^T x=  -7.8885; w3^T x=  -6.2647
class=   1; w1^T x=  16.1638; w2^T x=  -2.3232; w3^T x=  -8.0956
class=   2; w1^T x=   0.3508; w2^T x=   6.6281; w3^T x=  -6.2770
class=   2; w1^T x=  -5.1963; w2^T x=  12.1934; w3^T x=  -8.1079
class=   2; w1^T x=  -0.4773; w2^T x=  14.3727; w3^T x= -13.5885
class=   3; w1^T x=   2.0070; w2^T x=  -8.8611; w3^T x=   8.3461
class=   3; w1^T x=   7.5540; w2^T x= -14.4264; w3^T x=  10.1771
class=   3; w1^T x=  -2.7120; w2^T x= -11.0404; w3^T x=  13.8267
```

verifying that indeed we are correctly classifying all of the training data.

Notes on Mean Square Error Estimation

Consider the objective function $J(w)$ given by

$$J(w) = E[|y - x^T w|^2] \,,$$

where the expectation in the above is taken with respect to the *joint* density (X, Y). To evaluate this we will use the idea of iterated expectation where

$$E[X] = E[E[X|Y]] \,.$$

Since the Y variable is discrete $Y = \pm 1$ and once Y is chosen the x density is conditional on the value of y we have that $J(w)$ is given by

$$\begin{aligned} J(w) &= E[|y - x^T w|^2] \\ &= E\left[|y - x^T w|^2 | Y = -1\right] P(Y = -1) + E\left[|y - x^T w|^2 | Y = +1\right] P(Y = +1) \\ &= E\left[|-1 - x^T w|^2 | Y = -1\right] P(\omega_1) + E\left[|1 - x^T w|^2 | Y = +1\right] P(\omega_2) \\ &= P(\omega_1) \int (1 + x^T w)^2 p(x|\omega_1) dx + P(\omega_2) \int (1 - x^T w)^2 p(x|\omega_2) dx \,. \end{aligned} \tag{48}$$

Notes on the Multiclass Generalization

The text states "lets define $\mathbf{y} = [y_1, \ldots, y_M]$ for a given vector $\mathbf{x}$" but does not explicitly say how to define it. If we assume that the sample $\mathbf{x} \in \omega_j$ then the vector $\mathbf{y}$ should be all zeros except for a single one in the jth position. This procedure of encoding the class label as a position in a vector is known as *position encoding*.

Duplication of Example 3.3

See the MATLAB script `dup_example_3_3.m` for code that duplicates the numerical results from this example.

Notes on support vector machines in the linearly separable case

The book discusses the motivation for the support vector machine optimization problem in the case when the points are linearly separable. This problem is to compute the parameters, $\mathbf{w}$ and w_0 of the decision hyperplane $\mathbf{w}^T x + w_0 = 0$ such that

$$\text{minimize} \quad J(\mathbf{w}) \equiv \frac{1}{2}||\mathbf{w}||^2 \tag{49}$$

$$\text{subject to} \quad y_i(\mathbf{w}^T x_i + w_0) \geq 1 \quad \text{for} \quad i = 1, 2, \cdots, N \,. \tag{50}$$

This is a minimization problem with inequality constraints. The necessary conditions for its minimum are given by the Karush-Kuhn-Tucker (KKT) conditions. To introduce these we need to form the Lagrangian $\mathcal{L}$ given by

$$\mathcal{L}(w, w_0, \lambda) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \lambda_i [y_i(\mathbf{w}^T x_i + w_0) - 1] \,. \tag{51}$$

Then the KKT conditions for the above problem are:

- $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0$ and $\frac{\partial \mathcal{L}}{\partial w_0} = 0$.

- $\lambda_i \geq 0$ for $i = 1, 2, \cdots, N$
- $\lambda_i(y_i(\mathbf{w}^T x_i + w_0) - 1) = 0$ for $i = 1, 2, \cdots, N$.

We now need to use these conditions to *find* the optimum. One way to do this that might work for small problems is to assume some of the constraints are active i.e.

$$y_i(\mathbf{w}^T x_i + w_0) - 1 = 0\,,$$

for some set of i's. This is equivalent to fixing/determining the support vectors. Then we solve the resulting equations for the remaining Lagrange multipliers λ_i and *verify* that they are in fact *non-negative*.

We can also solve this optimization problem if we recognize that $J(\mathbf{w})$ is a strictly convex function and apply the Wolfe Dual representation to this problem. This states that the above minimization problem is equivalent to the following *maximization* problem

$$\begin{aligned} \text{maximize} \quad & \mathcal{L}(\mathbf{w}, w_0, \lambda) && (52)\\ \text{subject to} \quad & \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial w_0} = 0 && (53)\\ \text{and} \quad & \lambda_i \geq 0 \quad \text{for} \quad i = 1, 2, \cdots, N\,. && (54) \end{aligned}$$

To use this representation we take the form for $\mathcal{L}(\mathbf{w}, w_0, \lambda)$ given above and find that $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0$ gives

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \lambda_i y_i x_i = 0\,, \tag{55}$$

while $\frac{\partial \mathcal{L}}{\partial w_0} = 0$ is

$$\frac{\partial \mathcal{L}}{\partial w_0} = -\sum_{i=1}^{N} \lambda_i y_i = 0\,. \tag{56}$$

Putting these two constraints given by Equation 55 and 56 into Equation 51 we find

$$\begin{aligned} \mathcal{L} &= \frac{1}{2}\left(\sum_{i=1}^{N} \lambda_i y_i x_i\right)^T \left(\sum_{i=1}^{N} \lambda_i y_i x_i\right) - \sum_{i=1}^{N} \lambda_i \left[y_i \sum_{j=1}^{N} \lambda_j y_j x_j^T x_i + y_i w_0 - 1\right] \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j - \sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_j^T x_i - w_0 \sum_{i=1}^{N} \lambda_i y_i + \sum_{i=1}^{N} \lambda_i \\ &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j + \sum_{i=1}^{N} \lambda_i\,, \end{aligned} \tag{57}$$

for $\mathcal{L}$. We want to maximize the above expression under the two constraints on λ_i that $\sum_{i=1}^{N} \lambda_i y_i = 0$ (Equation 56) with $\lambda_i \geq 0$ (Equation 54). This can be done using quadratic programming and is an easier problem because the complicated inequality constraints from Equation 50 have now been replaced with equality constraints via Equation 56. Once we have computed λ_i with a quadratic programming algorithm we then compute $\mathbf{w}$ using $\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i x_i$. The computation of w_0 is done by *averaging* the complementary slackness conditions or

$$\lambda_i[y_i(\mathbf{w}^T x_i + w_0) - 1] = 0 \quad \text{for} \quad i = 1, 2, \cdots N\,.$$

Notes on support vector machines in the non-separable case

When the data points are non-separable the optimization problem in the separable case changes to compute the parameters, $\mathbf{w}$ and w_0 of the decision hyperplane $\mathbf{w}^T x + w_0 = 0$ such that

$$\begin{aligned} \text{minimize} \quad J(\mathbf{w}, w_0, \xi) &\equiv \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i && (58) \\ \text{subject to} \quad y_i(\mathbf{w}^T x_i + w_0) &\geq 1 - \xi_i \quad \text{for} \quad i = 1, 2, \cdots, N && (59) \\ \text{and} \quad \xi_i &\geq 0 \quad \text{for} \quad i = 1, 2, \cdots, N\,. && (60) \end{aligned}$$

This is again a minimization problem with inequality constraints. We form the Lagrangian $\mathcal{L}$ given by

$$\begin{aligned} \mathcal{L}(w, w_0, \xi; \lambda, \mu) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \\ &- \sum_{i=1}^{N}\mu_i\xi_i - \sum_{i=1}^{N}\lambda_i[y_i(\mathbf{w}^T x_i + w_0) - 1 + \xi_i]\,. \end{aligned}$$

The the necessary KKT conditions for this Lagrangian are that $\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = 0$ or

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N}\lambda_i y_i x_i = 0\,, \tag{61}$$

while $\frac{\partial\mathcal{L}}{\partial w_0} = 0$ is

$$\frac{\partial\mathcal{L}}{\partial w_0} = -\sum_{i=1}^{N}\lambda_i y_i = 0\,, \tag{62}$$

while $\frac{\partial\mathcal{L}}{\partial\xi_i} = 0$ is

$$\frac{\partial\mathcal{L}}{\partial\xi_i} = C - \mu_i - \lambda_i = 0\,, \tag{63}$$

for $i = 1, 2, \cdots, N$. Plus the conditions that the Lagrange multipliers are non-negative $\lambda_i \geq 0$, $\mu_i \geq 0$ for $i = 1, 2, \cdots, N$ and the complementary slackness conditions:

$$\begin{aligned} \lambda_i(y_i(\mathbf{w}^T w_i + w_0) - 1 - \xi_i) &= 0 \\ \mu_i\xi_i &= 0 \quad \text{for} \quad i = 1, 2, \cdots, N\,. \end{aligned}$$

To solve this we can again apply the Wolfe dual representation which states that the above minimization problem is equivalent to the maximization problem

$$\begin{aligned} &\text{maximize} && \mathcal{L}(\mathbf{w}, w_0, \xi; \lambda, \mu) && (64) \\ &\text{subject to} && \frac{\partial\mathcal{L}}{\partial\mathbf{w}} = 0\,, \quad \frac{\partial\mathcal{L}}{\partial w_0} = 0 \quad \text{and} \quad \frac{\partial\mathcal{L}}{\partial\xi} = 0 \quad \text{or} \\ & && \mathbf{w} = \sum_{i=1}^{N}\lambda_i y_i x_i\,, \quad \sum_{i=1}^{N}\lambda_i y_i = 0 \quad \text{and} \quad C - \mu_i - \lambda_i = 0\,, \\ &\text{and} && \lambda_i \geq 0 \quad \text{and} \quad \mu_i \geq 0 \quad \text{for} \quad i = 1, 2, \cdots, N\,. && (65) \end{aligned}$$

Using these constraints into the definition of $\mathcal{L}$ we find

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\left(\sum_{i=1}^{N}\lambda_i y_i x_i\right)^T\left(\sum_{i=1}^{N}\lambda_i y_i x_i\right) + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}(C-\lambda_i)\xi_i \\ &- \sum_{i=1}^{N}\lambda_i y_i\left(\sum_{j=1}^{N}\lambda_j y_j x_j^T\right)x_i - w_0\sum_{i=1}^{N}\lambda_i y_i + \sum_{i=1}^{N}\lambda_i - \sum_{i=1}^{N}\lambda_i\xi_i \\ &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j x_i^T x_j + \sum_{i=1}^{N}\lambda_i\,, \end{aligned} \tag{66}$$

which is another quadratic programming problem to be maximized over the vectors of Lagrangian multipliers λ_i and μ_i. Since the elements μ_i don't appear in the Wolfe dual objective function above the maximization takes place over the variable λ_i alone just as in the separable case above. What changes however is to recognize that the constraints $C-\mu_i-\lambda_i=0$, or $\mu_i = C-\lambda_i$ means that $\mu_i \geq 0$ implies $C-\lambda_i \geq 0$ or the λ_i constraint of $\lambda_i \leq C$.

Problem Solutions

Problem 3.1 (the perceptron cost function is continuous)

That the perceptron objective function

$$J(w) = \sum_{x\in Y}\delta_x w^T x\,,$$

where Y is the set of points misclassified by the weight vector w, is continuous as a function of w can be argued as follows. If changing w does not change the set Y of misclassified samples then we can write $J(w)$ as

$$J(w) = \sum_{x\in Y}\delta_x x^T w = \left(\sum_{x\in Y}\delta_x x\right)^T w = \alpha^T w\,,$$

or the product of a fixed vector, defined here to be α and the weight vector w. This is a continuous function. If changing w causes a point x go in or out of the misclassified set, Y, then around the value of w that causes this change $J(w)$ will change by $\pm\delta_x(w^T x)$. The point to note is that for a point x to go from correctly classified to incorrectly classified means that $w^T x$ must pass thought zero, since for one sign $w^T x$ will classify the point correctly while for the other sign it will not. The fact that this additional term $\pm\delta_x(w^T x)$ is continuous as a function of w imply that the full objective function $J(w)$ is continuous.

Problem 3.2 (if $\rho_k = \rho$ the perceptron converges)

From the notes on Page 39, namely Equation 46 when ρ_k does not depend on k we get

$$||w(t+1) - \alpha w^*||^2 \leq ||w(t) - \alpha w^*||^2 + \rho^2\beta^2 - 2\rho\alpha|\gamma|\,. \tag{67}$$

Lets observe how the distance between $w(t)$ and αw^* changes when $\alpha = \frac{\beta^2}{|\gamma|}$ (note this is a factor of two larger than the convergence result studied on Page 39). For this value of α we find that

$$||w(t+1) - \alpha w^*||^2 \leq ||w(t) - \alpha w^*||^2 + \beta^2(\rho^2 - 2\rho)\,.$$

Iterating this as before gives

$$||w(t) - \alpha w^*||^2 \leq ||w(0) - \alpha w^*||^2 + \beta^2(\rho^2 - 2\rho)t\,.$$

Thus if $\rho^2 - 2\rho < 0$ then as t increases in magnitude eventually the right-hand-side of the above becomes negative or

$$||w(0) - \alpha w^*||^2 + \beta^2(\rho^2 - 2\rho)t < 0\,. \tag{68}$$

The inequality $\rho^2 - 2\rho < 0$ requires that $\rho(\rho - 2) < 0$ or that $0 < \rho < 2$. The step k_0 where this above inequality Equation 68 first happens and we are guaranteed convergence when

$$t \geq k_0 \equiv \frac{||w(0) - \alpha w^*||^2}{\beta^2\rho(2-\rho)}\,. \tag{69}$$

Problem 3.3 (the reward and punishment form of the perceptron)

The form of the *reward and punishment form of the perceptron* is to start with an initial guess $w(0)$ at the separating hyperplane vector w^* and some ordering of the input data vectors, and then iterate

$$\begin{aligned}
w(t+1) &= w(t) + \rho x_{(t)} \quad \text{if} \quad x_{(t)} \in \omega_1 \quad \text{and} \quad w^T(t)x_{(t)} \leq 0 \\
w(t+1) &= w(t) - \rho x_{(t)} \quad \text{if} \quad x_{(t)} \in \omega_2 \quad \text{and} \quad w^T(t)x_{(t)} \geq 0 \\
w(t+1) &= w(t) \quad \text{otherwise}\,,
\end{aligned} \tag{70}$$

where $x_{(t)}$ is the data sample to consider on the tth step. Now consider how this algorithm might approach a multiple of the optimal vector or αw^*. Since only the first two equations above are relevant by subtracting αw^* from both sides and squaring we get

$$\begin{aligned}
||w(t+1) - \alpha w^*||^2 &= ||w(t) - \alpha w^*||^2 + \rho^2||x_{(t)}||^2 + 2\rho x_{(t)}^T(w(t) - \alpha w^*) \\
&\quad \text{if} \quad x_{(t)} \in \omega_1 \quad \text{and} \quad w^T(t)x_{(t)} \leq 0 \\
||w(t+1) - \alpha w^*||^2 &= ||w(t) - \alpha w^*||^2 - \rho^2||x_{(t)}||^2 - 2\rho x_{(t)}^T(w(t) - \alpha w^*) \\
&\quad \text{if} \quad x_{(t)} \in \omega_2 \quad \text{and} \quad w^T(t)x_{(t)} \geq 0\,.
\end{aligned}$$

Let $\beta^2 \equiv \max ||x_{(t)}||^2$, then the equation above become inequalities

$$\begin{aligned}
||w(t+1) - \alpha w^*||^2 &\leq ||w(t) - \alpha w^*||^2 + \rho^2\beta^2 + 2\rho x_{(t)}^T(w(t) - \alpha w^*) \\
&\quad \text{if} \quad x_{(t)} \in \omega_1 \quad \text{and} \quad w^T(t)x_{(t)} \leq 0 \\
||w(t+1) - \alpha w^*||^2 &\leq ||w(t) - \alpha w^*||^2 - \rho^2\beta^2 - 2\rho x_{(t)}^T(w(t) - \alpha w^*) \\
&\quad \text{if} \quad x_{(t)} \in \omega_2 \quad \text{and} \quad w^T(t)x_{(t)} \geq 0\,.
\end{aligned}$$

In the first equation, since w^* is a true solution, and $x_{(t)} \in \omega_1$ we have that both terms

$$2\rho x_{(t)}^T w(t) \quad \text{and} \quad -2\alpha\rho x_{(t)}^T w^*\,,$$

are negative. In the second equation, again since w^* is a true solution, and $x_{(t)} \in \omega_2$ both terms

$$-2\rho x_{(t)}^T w(t) \quad \text{and} \quad 2\alpha\rho x_{(t)}^T w^*\,,$$

are negative. Since the terms with $w(t)$ depend on the past iterates (and the starting conditions of the algorithm) we will drop $2\rho x_{(t)}^T w(t)$ from the first and $-2\rho x_{(t)}^T w(t)$ from the second. That means that we now have

$$\begin{aligned} ||w(t+1) - \alpha w^*||^2 &\leq ||w(t) - \alpha w^*||^2 + \rho^2\beta^2 - 2\rho\alpha x_{(t)}^T w^* \\ &\quad \text{if} \quad x_{(t)} \in \omega_1 \quad \text{and} \quad w^T(t)x_{(t)} \leq 0 \\ ||w(t+1) - \alpha w^*||^2 &\leq ||w(t) - \alpha w^*||^2 - \rho^2\beta^2 + 2\rho\alpha x_{(t)}^T w^* \\ &\quad \text{if} \quad x_{(t)} \in \omega_2 \quad \text{and} \quad w^T(t)x_{(t)} \geq 0\,. \end{aligned}$$

From the above discussion we know $-x_{(t)}^T w^*$ and $x_{(t)}^T w^*$ are negative so lets take the largest possible values for $-x_{(t)}^T w^*$ and $x_{(t)}^T w^*$ via

$$\begin{aligned} \gamma_1 &= \max_{x\in\omega_1}(-x_{(t)}^T w^*) < 0 \\ \gamma_2 &= \max_{x\in\omega_2}(x_{(t)}^T w^*) < 0 \\ \gamma &= \max(\gamma_1, \gamma_2) < 0\,. \end{aligned}$$

Then with the parameter γ both update lines collapse to

$$||w(t+1) - \alpha w^*||^2 \leq ||w(t) - \alpha w^*||^2 + \rho^2\beta^2 + 2\rho\alpha\gamma\,.$$

Since $\gamma < 0$ we can write it as $\gamma = -|\gamma|$ and we get

$$||w(t+1) - \alpha w^*||^2 \leq ||w(t) - \alpha w^*||^2 + \rho^2\beta^2 - 2\rho\alpha|\gamma|\,.$$

This is the same as Equation 67 and the same arguments as above show that $w(t) \to \alpha w^*$ as $t \to \infty$ proving the convergence of the reward and punishment form of the perceptron algorithm. In fact the arguments above show that we converge in a finite number of steps i.e. when $t \geq k_0$ where k_0 is given by Equation 69.

Problem 3.4 (iterating the perceptron algorithm by hand)

This problem is worked in the MATLAB `chap_3_prob_4.m`, and when it is run produces the plot shown in Figure 5. While the reward and punishment form of the perceptron is processing the weight $w(t)$ iterates through the following points

```
w_i=[       0.0,        0.0,        0.0]
w_i=[       0.0,        0.0,        1.0]
w_i=[      -1.0,        0.0,        0.0]
w_i=[      -1.0,        0.0,        1.0]
w_i=[      -2.0,        0.0,        0.0]
w_i=[      -2.0,        0.0,        1.0]
```

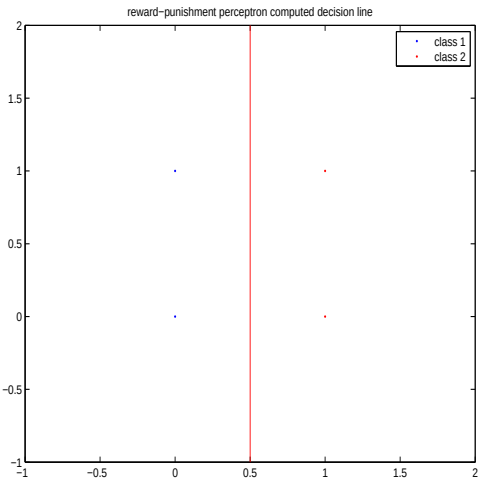

Figure 5: The decision line produced when running the script `chap_3_prob_4.m`.

The classification rule is then

$$w^T x > 0 \quad \Rightarrow \quad x \in \omega_1 \,,$$

and $x \in \omega_2$ otherwise. For the final decision line where $w = \begin{bmatrix} -2 & 0 & 1 \end{bmatrix}$ this corresponds to a classification rule where

$$x_1 < \frac{1}{2} \quad \Rightarrow \quad x \in \omega_1 \,.$$

Problem 3.5 (implementing the perceptron)

This problem is worked in the MATLAB `chap_3_prob_5.m`. When this script is run we get the result shown in Figure 6 (left).

Problem 3.6 (implementing the LMS algorithm)

This problem is worked in the MATLAB `chap_3_prob_6.m`. When this script is run we get the result shown in Figure 6 (center).

Problem 3.7 (Kesler's construction)

Consider applying the reward and punishment form of the perceptron to the data set obtained by using Kesler's construction. In that case we order the Kesler's expanded data samples $\tilde{x}_{(t)}$ and present them one at a time to the following algorithm

$$\begin{aligned} w(t+1) &= w(t) + \rho \tilde{x}_{(t)} \quad \text{if } \tilde{x}_{(t)} \text{ is misclassified by } w(t) \text{ i.e. } w^T(t)\tilde{x}_{(t)} \leq 0 \\ w(t+1) &= w(t) \quad \text{otherwise} \,. \end{aligned}$$

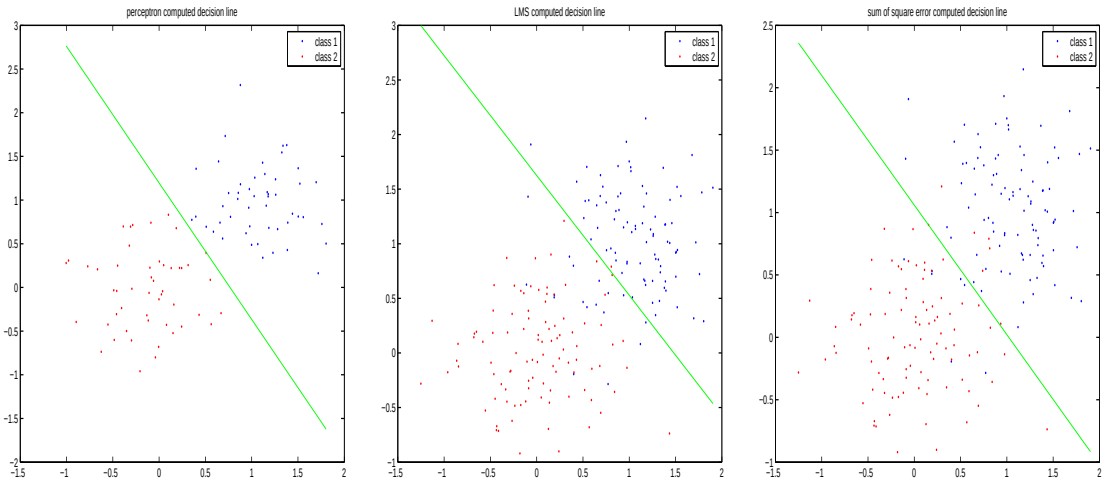

Figure 6: Data from two two-dimensional normals with mean vectors $\mu_1 = [1,1]^T$, $\mu_2 = [0,0]^T$, and $\sigma_1^2 = \sigma_2^2 = 0.2$. **Left:** The decision region produced by the perceptron algorithm. **Center:** The decision region produced by the LMS algorithm. **Right:** The decision region produced by the sum of square error criterion. Visually, all three algorithms produce similar results.

Since we are only trying to enforce $w^T(t)\tilde{x}_{(t)} > 0$ and $\tilde{x}_{(t)}$ is the Kesler's extended vector where the unexpanded vector $x_{(t)}$ is an element of ω_i. Since $w(t)$ in Kesler's construction is defined as $w = [w_1^T, w_2^T, \cdots, w_M^T]^T$ and since $\tilde{x}_{(t)}$ has a $+x_{(t)}$ at the block spot i and a $-x_{(t)}$ at the block spot j and zeros elsewhere the equation

$$w(t+1) = w(t) + \rho\tilde{x}_{(t)}\,,$$

will be executed/triggered if $w^T(t)\tilde{x}_{(t)} \leq 0$ or when

$$w_i^T x_{(t)} - w_j^T x_{(j)} \leq 0\,,$$

or

$$w_i^T x_{(t)} \leq w_j^T w_{(t)}\,,$$

and will update the block components of $w(t)$ as

$$\begin{aligned} w_i(t+1) &= w_i(t) + \rho x_{(t)} \\ w_j(t+1) &= w_j(t) - \rho x_{(t)} \\ w_k(t+1) &= w_k(t) \quad \forall k \neq i, j\,, \end{aligned}$$

as we were to show.

Problem 3.8 (the sum of square error optimal weight tends to MSE)

Recall that the objective functions for sum of square errors (SSE) and the mean square error (MSE) criteria are

$$\begin{aligned} J_{\text{SSE}}(w) &= \sum_{i=1}^{N}(y_i - x_i^T w)^2 \\ J_{\text{MSE}}(w) &= E[(y - x^T w)^2]\,, \end{aligned}$$

respectively. Since the objective function $J_{\text{SSE}}(w)$ is a multiple of a sample based estimate of the expectation we expect that as long as the sample estimate converge to the population values we expect the SSE weight should converge to the MSE weight. Since we can write the solution vector w under the SSE criterion (by taking the derivative of the above expression with respect to w) as

$$\sum_{i=1}^{N} x_i(y_i - x_i^T \hat{w}) = 0\,,$$

by dividing by N we get that this is equivalent to

$$\frac{1}{N}\sum_{i=1}^{N} x_i y_i - \left(\frac{1}{N}\sum_{i=1}^{N} x_i x_i^T\right)\hat{w} = 0\,.$$

which as we take the limit $N \to \infty$ and assuming sample convergence to the population values becomes

$$E[\mathbf{x}y] - E[\mathbf{x}\mathbf{x}^T]\hat{w} = 0\,,$$

which is the equation that the MSE solution $\hat{w}_{\text{MSE}}$ must satisfy.

Problem 3.9 (the sum of square error classifier)

This problem is worked in the MATLAB `chap_3_prob_9.m`. When this script is run we get the result shown in Figure 6 (right).

Problem 3.10 (the multiclass sum of square error criterion)

If we have a M classes problem then the sum of error squares estimation would be formalized by introducing the vector $\mathbf{y}_i = [y_1, y_2, \ldots, y_M]^T$, where y_j would take the value one, if the ith sample, x_i, was a member of class ω_j and would be zero otherwise. Then we introduce M vector weights w_j in the matrix W defined as $W = [w_1, w_2, \ldots, w_M]$. Thus the jth column of W is the vector weights w_j. To specify these weights we seek to minimize the square error

over all of our training samples or

$$\begin{aligned} J(W) &= \sum_{i=1}^{N} ||\mathbf{y}_i - W^T \mathbf{x}_i||^2 \\ &= \sum_{i=1}^{N} \sum_{j=1}^{M} (\mathbf{y}_{ij} - \mathbf{w}_j{}^T \mathbf{x}_i)^2 \\ &= \sum_{j=1}^{M} \left(\sum_{i=1}^{N} (\mathbf{y}_{ij} - \mathbf{w}_j{}^T \mathbf{x}_i)^2 \right) . \end{aligned}$$

Since this is the sum of M terms we can minimize the entire expression by picking w_j for each $j = 1, 2, \ldots, M$ to minimize the inner summation. Thus we are doing M parallel one dimensional problems. The jth problem find the value of w_j by associating to the target, y_i, a value of one if $x_i \in \omega_j$ and zero otherwise.

Problem 3.11 (jointly Gaussian random variables)

For this problem we are told that the joint distribution of the pair (X, Y) is given by

$$\begin{aligned} p_{X,Y}(x,y) &= \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\alpha^2}} \\ &\times \exp\left\{ -\frac{1}{2\alpha(1-\alpha)} \left[\left(\frac{x-\mu_x}{\sigma_x} \right)^2 + \left(\frac{y-\mu_y}{\sigma_y} \right)^2 - \frac{2\alpha(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} \right] \right\} . \end{aligned}$$

From the expression for $P_{X,Y}(x, y)$ it can be shown [10], that the conditional distribution $p\{X|Y\}$ is given by

$$p\{X|Y\} = \mathcal{N}\left(\mu_x + \alpha\frac{\sigma_x}{\sigma_y}(y - \mu_y), \sigma_x^2(1-\alpha^2) \right) .$$

The book asks for the distribution of $P\{Y|X\}$ which can be obtained from the above by exchanging x and y in the expression above.

Problem 3.12 (sum of error square criterion is the same as ML)

If we have M classes, the statement that the classifier outputs vary around the corresponding desired response values d_k^i, according to a Gaussian distribution with a known variance, means that we would expect that $g(x_i; w_k) \sim \mathcal{N}(d_k^i, \sigma^2)$ for $k = 1, 2, \cdots M$. In that case if we group all of the M response for a given sample x_i into a target vector like

$$\mathbf{y}_i \equiv \begin{bmatrix} d_1^i \\ d_2^i \\ \vdots \\ d_M^i \end{bmatrix} .$$

The distribution of $\mathbf{y}_i$ is a multivariate Gaussian with a mean given by $\mathbf{y}_i$, and a covariance matrix given by $\sigma^2 I$. Thus the distribution of $\mathbf{y}_i$ is given by

$$\begin{bmatrix} g(x_i; w_1) \\ g(x_i; w_2) \\ \vdots \\ g(x_i; w_M) \end{bmatrix} \sim \mathcal{N}(\mathbf{y}_i, \sigma^2 I) .$$

Thus we expect from how $\mathbf{y}_i$ is defined that

$$\begin{bmatrix} g(x_i; w_1) \\ g(x_i; w_2) \\ \vdots \\ g(x_i; w_M) \end{bmatrix} - \begin{bmatrix} d_1^i \\ d_2^i \\ \vdots \\ d_M^i \end{bmatrix} \sim \mathrm{N}(\mathbf{0}, \sigma^2 I) .$$

Thus the loglikelihood L of N samples x_i would then be given by

$$\begin{aligned} L(\{x_i\}) &= \ln\left(\prod_{i=1}^{N} \frac{1}{(2\pi)^l \sigma^{2l}} \exp\left\{ -\frac{1}{2\sigma^2} \begin{bmatrix} g(x_i; w_1) - d_1^i \\ g(x_i; w_2) - d_2^i \\ \vdots \\ g(x_i; w_M) - d_M^i \end{bmatrix}^T \begin{bmatrix} g(x_i; w_1) - d_1^i \\ g(x_i; w_2) - d_2^i \\ \vdots \\ g(x_i; w_M) - d_M^i \end{bmatrix} \right\} \right) \\ &= \ln\left(\prod_{i=1}^{N} \frac{1}{(2\pi)^l \sigma^{2l}} \exp\left\{ -\frac{1}{2\sigma^2} \left[\sum_{k=1}^{M} (g(x_i; w_k) - d_i^k)^2 \right] \right\} \right) \\ &= \ln\left(\frac{1}{(2\pi)^{lN} \sigma^{2lN}} \exp\left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{N} \sum_{k=1}^{M} (g(x_i; w_k) - d_i^k)^2 \right\} \right) . \end{aligned}$$

Since we will want to maximize the loglikelihood as a function of the vectors w_k we can drop all things that don't depend on w and minimize the negative of what remains. We then need to minimize as a function of w_k for $k = 1, 2, \cdots, M$ the following expression

$$L'(\{x_i\}) = \sum_{i=1}^{N} \sum_{k=1}^{M} (g(x_i; w_k) - d_i^k)^2 .$$

This is exactly the sum of squares error criterion in the multiclass case.

Problem 3.13 (approximating $P(\omega_1|x) - P(\omega_2|x)$ in a MSE sense)

The MSE training procedure (in the two class case) computes the decision surface $f(x; w)$ weights w via

$$\begin{aligned} \hat{w} &= \mathrm{argmin}_w E\left[(f(x; w) - y)^2 \right] \\ &= \mathrm{argmin}_w \int_{-\infty}^{\infty} \sum_{j=1}^{2} (f(x; w) - y)^2 P(x, \omega_j) dx . \end{aligned}$$

Using $p(x,\omega_j) = P(\omega_j|x)p(x)$ and if we code each classes response as $y=+1$ when $x \in \omega_1$ and $y=-1$ when $x \in \omega_2$ then the argument inside the integration in the above minimization becomes

$$\begin{aligned}
& (f(x;w)-1)^2 P(\omega_1|x) + (f(x;w)+1)^2 P(\omega_2|x) \\
= & (f(x;w)^2 - 2f(x;w)+1)P(\omega_1|x) + (f(x;w)^2 + 2f(x;w)+1)P(\omega_2|x) \\
= & f(x;w)^2 - 2f(x;w)[P(\omega_1|x) - P(\omega_2|x)] + 1 \\
= & f(x;w)^2 - 2f(x;w)[P(\omega_1|x) - P(\omega_2|x)] \\
+ & [P(\omega_1|x) - P(\omega_2|x)]^2 - [P(\omega_1|x) - P(\omega_2|x)]^2 + 1 \\
= & [f(x;w) - (P(\omega_1|x) - P(\omega_2|x))]^2 - [P(\omega_1|x) - P(\omega_2|x)]^2 + 1 \,,
\end{aligned}$$

where we have used the fact that $P(\omega_1|x) + P(\omega_2|x) = 1$. Note that the last two terms above namely $-[P(\omega_1|x) - P(\omega_2|x)]^2 + 1$ do not depend on the the vector w and thus don't change the results of the minimization. Thus we are left with

$$\begin{aligned}
\hat{w} &= \text{argmin}_w \int_{-\infty}^{\infty} (f(x;w) - \{P(\omega_1|x) - P(\omega_2|x)\})^2 p(x) dx \\
&= \text{argmin}_w E\left[(f(x;w) - \{P(\omega_1|x) - P(\omega_2|x)\})^2\right] \,.
\end{aligned}$$

Thus we are picking w to have $f(x;w)$ approximate the decision boundary $P(\omega_1|x) - P(\omega_2|x)$ optimally in the mean square error sense.

Problem 3.14 (how the Bayesian and MSE classifier differ)

The Bayesian classifier in the equal class covariance case is given by Equation 20. Thus we need to compute the MSE hyperplane we first extend each feature vector by adding the value of 1 to get new vectors $\mathbf{v}$ defined by

$$\mathbf{v} = [\mathbf{x}^T, 1]^T \,.$$

The vector $\mathbf{v}$ is now of dimension $l+1$. Then in this augmented space the optimal MSE linear classifier is given by computing $\hat{w}^T v$. Then if $\hat{w}^T v > 0$ we declare $v \in \omega_1$ and otherwise declare $v \in \omega_2$.

$$\hat{w} = R_v^{-1} E[vy] \,. \tag{71}$$

where $\hat{w}$ is the augmented vector $\begin{bmatrix} w \\ w_0 \end{bmatrix}$ so that it includes an intercept w_0 and

$$R_x = E[vv^T] = \begin{bmatrix} E[v_1 v_1] & \cdots & E[v_1 v_{l+1}] \\ E[v_2 v_1] & \cdots & E[v_2 v_{l+1}] \\ \vdots & & \vdots \\ E[v_{l+1} v_1] & \cdots & E[v_{l+1} v_{l+1}] \end{bmatrix} \,. \tag{72}$$

We can now evaluate several of these expectations. We have that $\mathbf{v}\mathbf{v}^T$ in terms of the original $\mathbf{x}$ is given by

$$\mathbf{v}\mathbf{v}^T = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}\mathbf{x}^T & \mathbf{x} \\ \mathbf{x}^T & 1 \end{bmatrix} \,.$$

Taking the expectation of this gives

$$\begin{bmatrix} R & E[x] \\ E[x]^T & 1 \end{bmatrix}.$$

Next we compute the expectation of $\mathbf{v}y = \begin{bmatrix} x \\ 1 \end{bmatrix} y$ or $E[\mathbf{v}y] = \begin{bmatrix} E[xy] \\ E[y] \end{bmatrix}$. Since the response is $y = +1$ when $x \in \omega_1$ and $y = -1$ when $x \in \omega_2$ we can compute

$$E[y] = 1P(\omega_1) - 1P(\omega_2) = 0\,,$$

when $P(\omega_1) = P(\omega_2) = \frac{1}{2}$. Next we find $E[xy]$ under the same condition as

$$\begin{aligned} E[xy] &= E[x|y=+1]P(\omega_1) - E[x|y=-1]P(\omega_2) \\ &= \mu_1 P(\omega_1) - \mu_2 P(\omega_2) = \mu_1 - \mu_2\,. \end{aligned}$$

For completeness (and to be used later) we now compute $E[x]$ and find

$$E[x] = E[x|x \in \omega_1]P(\omega_1) + E[x|x \in \omega_2]P(\omega_2) = \frac{1}{2}(\mu_1 + \mu_2)\,.$$

From these expressions we see that we need to solve $R_v \hat{w} = E[vy]$ or

$$\begin{bmatrix} R & E[x] \\ E[x]^T & 1 \end{bmatrix} \begin{bmatrix} w \\ w_0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(\mu_1 - \mu_2) \\ 0 \end{bmatrix},$$

for $\hat{w}$. From the second of these equations we can solve for w_0 as

$$E[x]^T w + w_0 = 0 \Rightarrow w_0 = -E[x]^T w\,, \tag{73}$$

which we can put in the first of the equations above to get

$$(R - E[x]E[x]^T)w = \frac{1}{2}(\mu_1 - \mu_2)\,.$$

To further evaluate this note that by expanding the quadratic and distributing the expectation we can show that

$$\begin{aligned} \Sigma &\equiv E[(x - E[x])(x - E[x])^T] \\ &= R - E[x]E[x]^T \end{aligned} \tag{74}$$

Thus from Equation 74 we see that w is given by

$$w = \frac{1}{2}\Sigma^{-1}(\mu_1 - \mu_2)\,.$$

When we put that expression into Equation 73 and use what we have computed for $E[x]$ we get

$$w_0 = -\frac{1}{4}(\mu_1 + \mu_2)^T \Sigma^{-1}(\mu_1 - \mu_2)\,.$$

Thus the MSE decision line is to classify a point x as a member of ω_1 if $x^T w + w_0 > 0$ or

$$\frac{1}{2}x^T\Sigma^{-1}(\mu_1 - \mu_2) - \frac{1}{4}(\mu_1 + \mu_2)^T\Sigma^{-1}(\mu_1 - \mu_2) > 0\,,$$

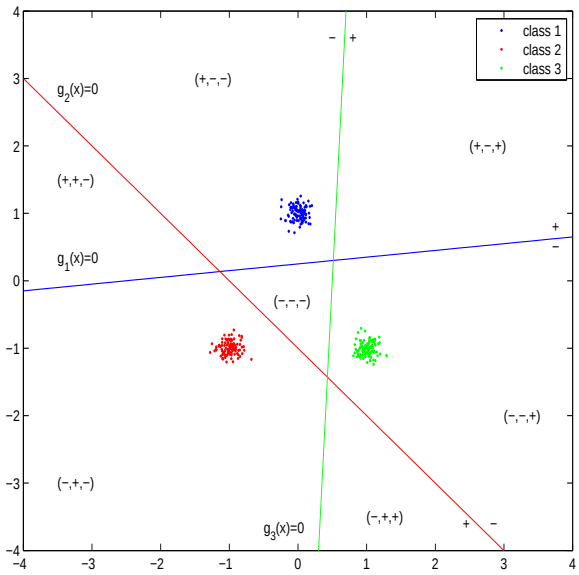

Figure 7: The decision surface produced when running the script `chap_3_prob_15.m`.

or

$$\left(x - \frac{1}{2}(\mu_1 + \mu_2)\right)^T \Sigma^{-1}(\mu_1 - \mu_2) > 0\,.$$

or writing it like Equation 20 we have

$$(\mu_1 - \mu_2)^T \Sigma^{-1} x > \frac{1}{2}(\mu_1 - \mu_2)\,.$$

Note that this equation is only different from Equation 20 with regard to the right-hand-side threshold.

Problem 3.15 (the design of M hyperplanes)

An example like one requested is produced via running the MATLAB script `chap_3_prob_15.m`. When this script is run we get the result shown in Figure 7. Note that no data exists in the region where the three discriminant functions are negative which is denoted $(-,-,-)$. Also regions with discriminant signs like $(+,+,-)$ exist where more than one discriminant function is positive.

Problem 3.16 (using the KKT conditions)

To use the Karush-Kuhn-Tucker (KKT) conditions for the given data points in Example 3.4 we assume that the points are linearly separable and use the problem formulation given in

on Page 43. Thus we start from the formulation

$$\begin{aligned} \text{minimize } J(\mathbf{w}) &= \frac{1}{2}||\mathbf{w}||^2 \quad \text{subject to} \\ y_i(\mathbf{w}^T x_i + w_0) &\geq 1 \quad \text{for} \quad i = 1, 2, \cdots, N\,. \end{aligned}$$

The specific Lagrangian for this problem is

$$\begin{aligned} \mathcal{L}(\mathbf{w}, w_0, \lambda) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \lambda_i(y_i(\mathbf{w}^T x_i + w_0) - 1) \\ &= \frac{1}{2}(w_1^2 + w_2^2) \\ &- \lambda_1(w_1 + w_2 + w_0 - 1) - \lambda_2(w_1 - w_2 + w_0 - 1) \\ &- \lambda_3(w_1 - w_2 - w_0 - 1) - \lambda_4(w_1 + w_2 - w_0 - 1)\,, \end{aligned}$$

since $y_i = +1$ when $x_i \in \omega_1$ and $y_i = -1$ when $x_i \in \omega_2$. The necessary Karush-Kuhn-Tucker conditions are then given by

$$\frac{\partial \mathcal{L}}{\partial w_1} = 0 \Rightarrow w_1 - \lambda_1 - \lambda_2 - \lambda_3 - \lambda_4 = 0 \tag{75}$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = 0 \Rightarrow w_2 - \lambda_1 + \lambda_2 + \lambda_3 - \lambda_4 = 0 \tag{76}$$

$$\frac{\partial \mathcal{L}}{\partial w_0} = 0 \Rightarrow -\lambda_1 - \lambda_2 + \lambda_3 + \lambda_4 = 0\,. \tag{77}$$

The complementary slackness conditions for the four points are given by

$$\begin{aligned} \lambda_1(w_1 + w_2 + w_0 - 1) &= 0 \\ \lambda_2(w_1 - w_2 + w_0 - 1) &= 0 \\ \lambda_3(-(-w_1 + w_2 + w_0) - 1) &= 0 \\ \lambda_4(-(-w_1 - w_2 + w_0) - 1) &= 0\,, \end{aligned}$$

with $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0$. If we want to consider searching only for lines that pass thought the origin we take $w_0 = 0$ and the equations above simplify to

$$\begin{aligned} w_1 &= \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 \\ w_2 &= \lambda_1 - \lambda_2 - \lambda_3 + \lambda_4 \\ & -\lambda_1 - \lambda_2 + \lambda_3 + \lambda_4 = 0 \\ & \lambda_1(w_1 + w_2 - 1) = 0 \\ & \lambda_2(w_1 - w_2 - 1) = 0 \\ & \lambda_3(w_1 - w_2 - 1) = 0 \\ & \lambda_4(w_1 + w_2 - 1) = 0\,. \end{aligned}$$

Put the expressions just derived for w_1 and w_2 into the complementary slackness conditions and by changing the ordering of the equations we get

$$\begin{aligned} \lambda_1(2\lambda_1 + 2\lambda_4 - 1) &= 0 \\ \lambda_4(2\lambda_1 + 2\lambda_4 - 1) &= 0 \\ \lambda_2(2\lambda_2 + 2\lambda_3 - 1) &= 0 \\ \lambda_3(2\lambda_2 + 2\lambda_3 - 1) &= 0 \\ -\lambda_1 - \lambda_2 + \lambda_3 + \lambda_4 &= 0\,. \end{aligned} \tag{78}$$

These equations have multiple solutions, but the first two will hold true if $2\lambda_1 + 2\lambda_4 - 1 = 0$ and the second two will be true if $2\lambda_2 + 2\lambda_3 - 1 = 0$. Lets specify that these constraints are active. That is we will assume that

$$\begin{aligned} 2\lambda_1 + 2\lambda_4 - 1 &= 0 \Rightarrow \lambda_1 = \frac{1}{2} - \lambda_4 \\ 2\lambda_2 + 2\lambda_3 - 1 &= 0 \Rightarrow \lambda_2 = \frac{1}{2} - \lambda_3 \,. \end{aligned} \tag{79}$$

If we put these two expressions into Equation 78 we get

$$-\frac{1}{2} + \lambda_4 - \frac{1}{2} + \lambda_3 + \lambda_3 + \lambda_4 = 0 \,,$$

or

$$2\lambda_4 + 2\lambda_3 - 1 = 0 \,,$$

which again has multiple solutions. If we pick $\lambda_4 \geq 0$ arbitrary then solving for λ_3 we have $\lambda_3 = \frac{1}{2} - \lambda_4$. Using Equation 79 for λ_2 in terms of λ_3 we have

$$\lambda_2 = \frac{1}{2} - \left(\frac{1}{2}\lambda_4\right) = \lambda_4 \,.$$

Thus we have shown all values of λ_i can be written in terms of λ_4 as

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} - \lambda_4 \\ \lambda_4 \\ \frac{1}{2} - \lambda_4 \\ \lambda_4 \end{bmatrix} .$$

Using this our weight vector $\mathbf{w}$ is

$$\begin{aligned} \mathbf{w} &= \sum_{i=1}^{4} \lambda_i y_i x_i \\ &= \left(\frac{1}{2} - \lambda_4\right)(+1)\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \lambda_4(+1)\begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ &+ \left(\frac{1}{2} - \lambda_4\right)(-1)\begin{bmatrix} -1 \\ 1 \end{bmatrix} + \lambda_4(-1)\begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} , \end{aligned}$$

the same solution quoted in the text.

Nonlinear Classifiers

Notes on the text

Notes on the XOR Problem

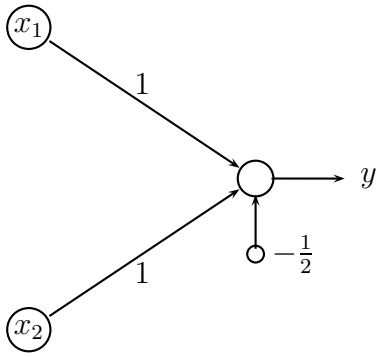

Figure 8: The perceptron that implements the OR gate.

The perceptron that implements the OR gate is shown in Figure 8. From the given diagram we see that this perceptron gives an output value of 1 when

$$x_1 + x_2 - \frac{1}{2} > 0\,.$$

This expression is false for the point $(0,0)$ and is true for the points $(1,0)$, $(0,1)$, and $(1,1)$.

Algorithms based on exact classification of the training set

We documents some notes on Meza's *tiling algorithm* for building a two-class neural network that exactly classifies the given input data points. In this algorithm we will be *growing* a network that depends on the training data rather than starting with a fixed network and then determining the parameters of the fixed network. The book does a very nice job explaining the general procedure and these are just some notes I wrote up going into more detail on the simple example given. The example points and decision lines for this section are duplicated in Figure 9. For this procedure we begin by first dividing the initial data set into two regions using one of the linear algorithms discussed in the previous chapter. For example, the pocket algorithm or a SVM like algorithm could be used to define the initial splitting of the data. This is denoted as the decision line #1 in Figure 9. After this line is specified, we determine that the training points are not classified with 100% certainty. We define the set X^+ to be the set of points on the "plus" side of line #1. In this set there is one misclassified vector B_1. We define the set X^- as the set of points on the "minus" side of line #1. The set X^- has the misclassified vector A_1. For any set $X^\pm$ that has misclassified vectors we then recursively apply the previous algorithm to these sets. In the example above, we would divide the set X^+

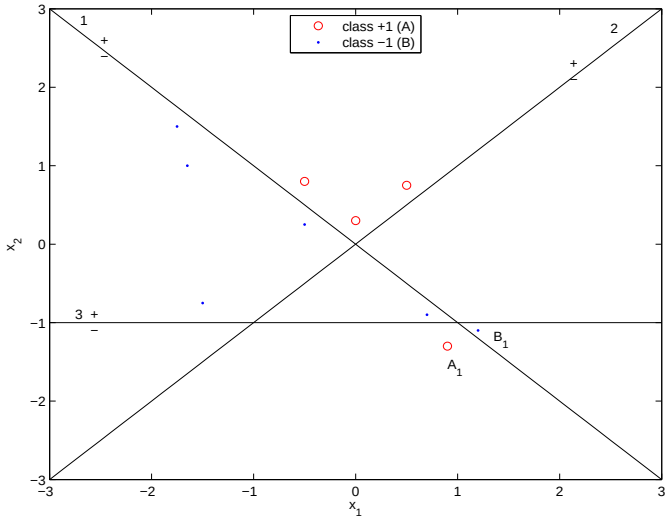

Figure 9: The sample data points and decision regions used to explain the Meza's tiling procedure for generating a network that correctly classifies the given input points.

again using an algorithm that produces a separating hyperplane (like the pocket algorithm) and produce the decision line #2 in Figure 9. For the set X^- we do the same thing and produce the separating hyperplane denoted by the decision line #3 in Figure 9. The sets on the "plus" and "minus" side of X^+ are then denoted as X^{++} and X^{+-}. The same sets for the set X^- are denoted as X^{-+} and X^{--}. If any of these four new sets has misclassified vectors it will be further reduced using a separating hyperplane. This procedure is guaranteed to finish since each separating hyperplane we introduce is processing a smaller and smaller set of data. Each of these separating hyperplanes is pictorially represented as a node in the input layer horizontally located next to the *master* unit (or the node representing the first data split). As each hyperplane introduced is used to split misclassified data into groups with correct classification, no two data points, in different classes, can end up mapped to the same point after the first layer. Meza's algorithm next operates recursively on the set of points computed from the first layer or X_1 where X_1 is given by

$$X_1 = \{y : y = f_1(x), \quad \text{for} \quad x \in X\}\,,$$

and f_1 is the mapping implemented in the first layer. Meza has shown that this recursive procedure converges, while the book argues that since no two data points are mapped to the same vertex of a hypercube correct classification may be possible with at most three layers.

Notes on backpropagation algorithm: the antisymmetric logistic

For the *logistic function*, $f(x)$, given by

$$f(x) = \frac{2}{1+\exp(-ax)} - 1\,. \tag{80}$$

We can show that this is equivalent to a hyperbolic tangent function with

$$\begin{aligned} f(x) &= \frac{2}{1+\exp(-ax)} - \frac{1+\exp(-ax)}{1+\exp(-ax)} = \frac{1-\exp(-ax)}{1+\exp(-ax)} \\ &= \frac{e^{\frac{ax}{2}} - e^{-\frac{ax}{2}}}{e^{\frac{ax}{2}} + e^{-\frac{ax}{2}}} = \frac{\sinh(\frac{ax}{2})}{\cosh(\frac{ax}{2})} = \tanh\left(\frac{ax}{2}\right) . \end{aligned}$$

Notes on backpropagation algorithm: the gradients

This section are some simple notes that I took as I worked through the derivation of the backpropagation algorithm. Much of the discussion is of the same general form as presented in the book, but these notes helped me understand this material so I wrote them up so that they might help others.

We will assume that we have a L layer network where L is given to us and fixed and we want to learn the values of the parameters that will give us the smallest mean square error (MSE) over all of the training samples. The input layer is denoted as the layer 0 and the output layer is denoted as layer L. The notation k_r, for $r = 0, 1, 2, \cdots L$, will denote the number of nodes in the rth layer so for example, we will have k_0 nodes in the first and input layer, and k_L nodes in the Lth or output layer. To denote the weights that will go *into* the summation at the j node in the rth layer from the nodes in the $r-1$th layer we will use the notation

$$\mathbf{w}_j^r = \left[\begin{array}{cccc} w_{j0}^r , & w_{j1}^r , & \cdots , & w_{jk_{r-1}}^r \end{array} \right]^T . \tag{81}$$

Where $r = 1, 2, \cdots, L$. The value of w_{0j}^r is the constant bias input used by the jth node in the rth layer. We will update these vectors $\mathbf{w}_j^r$ iteratively to minimize the MSE using a gradient decent scheme. Thus we will use the weight update equation

$$\mathbf{w}_j^r(\text{new}) = \mathbf{w}_j^r(\text{old}) + \Delta\mathbf{w}_j^r , \tag{82}$$

where

$$\Delta\mathbf{w}_j^r = -\mu \frac{\partial J}{\partial \mathbf{w}_j^r} , \tag{83}$$

and J is the cost function we seek to minimize. For the application here the cost function J we will use will be a sum of individual errors over all of the sample points in the training set of

$$J \equiv \sum_{i=1}^{N} \mathcal{E}(i) . \tag{84}$$

Here $\mathcal{E}(i)$ is the *sample error* we assign to the approximate output of the Lth output layer denoted $\hat{y}_m(i)$. We let $e_m(i)$ denote the error in the mth component of the output our network makes when attempting to classify the sample $x(i)$ or

$$e_m(i) \equiv y_m(i) - \hat{y}_m(i) \quad \text{for} \quad m = 1, 2, \cdots, k_L . \tag{85}$$

Using this definition the sample error $\mathcal{E}(i)$ is defined as just the mean square error of the networks mth component output due to the ith sample or $\hat{y}_m(i)$ against the true expected

result or

$$\mathcal{E}(i) \equiv \frac{1}{2}\sum_{m=1}^{k_L} e_m(i)^2 = \frac{1}{2}\sum_{m=1}^{k_L}(y_m(i) - \hat{y}_m(i))^2 \,. \tag{86}$$

At this point we are almost ready to derive the backpropagation algorithm but we will need a few more definitions. In general, a neural network works by multiplying internal node outputs by weights, summing these products, and then passing this sum through the nonlinear activation function $f(\cdot)$. We can imagine that for every input-output pairing (x_i, y_i) for $i = 1, 2, \cdots, N$ we have a value for all of the variables just introduced. Thus our notation needs to depend on the sample index i. We do this by including this index on a variable, say X, using functional notation as in $X(i)$. To represent the other variables we will let $y_k^{r-1}(i)$ represent the *output* the kth neuron in the $r-1$th layer (for $r = 2, 3, \cdots, L, L+1$). In this notation, when $r = 2$ then we have $y_k^1(i)$ which is the output of the first hidden layer and when $r = L+1$ we have $y_k^L(i)$ which is the output from the last (or output) layer. Since there are k_{r-1} nodes in the $r-1$ layer we have $k = 1, 2, \cdots, k_{r-1}$. As introduced above, the scalar w_{jk}^r is the weight *leading into* the jth neuron of the rth layer from the $k$the neuron in the $r-1$ layer. Since the rth layer has k_r nodes we have $j = 1, 2, \cdots, k_r$. Note that we assume that after sufficient training the weights will converge and there is no i index in their notational specification. As the input to the activation function for node j in layer r we need to multiply these weights with the neuron outputs from layer $r-1$ and sum. We denote this result $v_j^r(i)$. Thus we have now defined

$$v_j^r(i) = \sum_{k=1}^{k_{r-1}} w_{jk}^r y_k^{r-1}(i) + w_{j0}^r = \sum_{k=0}^{k_{r-1}} w_{jk}^r y_k^{r-1}(i) \,, \tag{87}$$

where we take $y_0^{r-1} \equiv 1$ to make the incorporation of a constant bias weight w_{j0}^r transparent.

With these definitions we are now ready to derive the backpropagation algorithm for learning neural network weights $\mathbf{w}_j^r$. Since we assume an initial random assignment of the neural network weights we can assume that we know values for all of the variables introduced thus far for every sample. We seek to use derivative information to change these initial values for $\mathbf{w}_j^r$ into weights that make the global cost function J smaller. The backpropagation procedure starts with the weights that feed into the last L layer, namely $\mathbf{w}_j^L$, and works backwards updating the weights between each hidden layer until it reaches the weights at the first hidden layer $\mathbf{w}_j^1$. Once all of the weights are updated we pass every sample (x_i, y_i) thought the modified network (in the forward direction) and are ready to do another backpropagation pass.

To use the gradient decent algorithm we will need the derivatives of the sample error function $\mathcal{E}(i)$ with respect to $\mathbf{w}_j^r$, the weights that go into the j neuron in the rth layer. Converting this derivative using the chain rule into the derivative with respect to the output $v_j^r(i)$ we have

$$\frac{\partial \mathcal{E}(i)}{\partial \mathbf{w}_j^r} = \frac{\partial \mathcal{E}(i)}{\partial v_j^r(i)} \frac{\partial v_j^r(i)}{\partial \mathbf{w}_j^r} \,. \tag{88}$$

Because of the linear nature in which $v_j^r(i)$ and $\mathbf{w}_j^r$ are related via Equation 87 this second

derivative is easily computed

$$\frac{\partial v_j^r(i)}{\partial \mathbf{w}_j^r} = \begin{bmatrix} \frac{\partial v_j^r(i)}{\partial w_{j0}^r} \\ \frac{\partial v_j^r(i)}{\partial w_{j1}^r} \\ \vdots \\ \frac{\partial v_j^r(i)}{\partial w_{jk_{r-1}}^r} \end{bmatrix} = \begin{bmatrix} y_0^{r-1}(i) \\ y_1^{r-1}(i) \\ \vdots \\ y_{k_r-1}^{r-1}(i) \end{bmatrix} = \begin{bmatrix} 1 \\ y_1^{r-1}(i) \\ \vdots \\ y_{k_r-1}^{r-1}(i) \end{bmatrix} \equiv \mathbf{y}^{r-1}(i)\,. \tag{89}$$

In the above notation $\mathbf{y}^{r-1}(i)$ is the vector of *all* outputs from neurons in the $r-1$st layer of the network. Notice that this value is *the same* for all nodes j in the rth layer. Lets define the remaining derivative above or $\frac{\partial \mathcal{E}(i)}{\partial v_j^r(i)}$ as

$$\delta_j^r(i) \equiv \frac{\partial \mathcal{E}(i)}{\partial v_j^r(i)}\,, \tag{90}$$

for every j in the rth layer or $j = 1, 2, \cdots k_r$. Using these results we have $\Delta \mathbf{w}_j^r$ given by.

$$\Delta \mathbf{w}_j^r = -\mu \frac{\partial J}{\partial \mathbf{w}_j^r} = -\mu \sum_{i=1}^{N} \frac{\partial \mathcal{E}(i)}{\partial \mathbf{w}_j^r} = -\mu \sum_{i=1}^{N} \delta_j^r(i) \mathbf{y}^{r-1}(i)\,. \tag{91}$$

It is these $\delta_j^r(i)$ we will develop a recursive relationship for and the above expression will enable us to compute the derivatives needed. Recall that we define $\mathcal{E}(i)$ the "error" in the ith sample output as in Equation 86 here expressed in terms of $v_m^L(i)$ as

$$\mathcal{E}(i) \equiv \frac{1}{2} \sum_{m=1}^{k_r} (y_m(i) - \hat{y}_m(i))^2 = \frac{1}{2} \sum_{m=1}^{k_r} (y_m(i) - f(v_m^L(i)))^2\,. \tag{92}$$

The output layer: In this case $r = L$ and we are at the last layer (the output layer) and we want to evaluate $\delta_j^L(i) = \frac{\partial \mathcal{E}(i)}{\partial v_j^L(i)}$ when $\mathcal{E}(i)$ is given by Equation 92. From that expression we see that the v_j^L derivative selects only the jth element from the sum and we get

$$\begin{aligned} \frac{\partial \mathcal{E}(i)}{\partial v_j^L(i)} &= -\frac{1}{2}(2)(y_j(i) - f(v_j^L(i)))f'(v_j^L(i)) \\ &= -e_j(i) f'(v_j^L(i))\,. \end{aligned} \tag{93}$$

where we have defined the error $e_j(i)$ as in Equation 85.

The hidden layers: In this case $r < L$ and the influence of v_j^{r-1} on $\mathcal{E}(i)$ comes indirectly through its influence on v_j^r. Thus using the chain rule to introduce this variable we have

$$\frac{\partial \mathcal{E}(i)}{\partial v_j^{r-1}(i)} = \sum_{k=1}^{k_r} \frac{\partial \mathcal{E}(i)}{\partial v_k^r(i)} \frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)}\,. \tag{94}$$

On using the definition of $\delta_j^r(i)$ given by Equation 90 in both side of this expression we have

$$\delta_j^{r-1}(i) = \sum_{k=1}^{k_r} \delta_k^r(i) \frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)}\,. \tag{95}$$

We stop here to note that this is a expression for the previous layer's δ_j^{r-1} showing how to compute it given values of the current layer's δ_j^r. To fully evaluate that we need to compute $\frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)}$. Using Equation 87 we find

$$\frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} = \frac{\partial}{\partial v_j^{r-1}(i)} \left[\sum_{m=0}^{k_{r-1}} w_{km}^r y_m^{r-1}(i) \right] = \frac{\partial}{\partial v_j^{r-1}(i)} \left[\sum_{m=0}^{k_{r-1}} w_{km}^r f(v_m^{r-1}(i)) \right] .$$

This derivative again selects the $m = j$th term in the above sum and we find

$$\frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} = w_{kj}^r f'(v_j^{r-1}(i)) . \tag{96}$$

Thus the recursive propagation of $\delta_j^r(i)$ the then given by using Equation 95 with the above derivative where we find

$$\begin{aligned} \delta_j^{r-1}(i) &= \sum_{k=1}^{k_r} \delta_k^r(i) w_{kj}^r f'(v_j^{r-1}(i)) = f'(v_j^{r-1}(i)) \sum_{k=1}^{k_r} \delta_k^r(i) w_{kj}^r & (97) \\ &= \hat{e}_j^{r-1} f'(v_j^{r-1}(i)) , & (98) \end{aligned}$$

with $\hat{e}_j^{r-1}$ defined by

$$\hat{e}_j^{r-1} = \sum_{k=1}^{k_r} \delta_k^r(i) w_{kj}^r . \tag{99}$$

When our activation function $f(x)$ is the *sigmoid function* defined by

$$f(x) = \frac{1}{1 + e^{-ax}} , \tag{100}$$

we find its derivative given by

$$\begin{aligned} f'(x) &= \frac{-(-a)}{(1 + e^{-ax})^2} e^{-ax} = f(x)(a) \frac{e^{-ax}}{1 + e^{-ax}} = a f(x) \left[\frac{1 + e^{-ax} - 1}{1 + e^{-ax}} \right] \\ &= a f(x)(1 - f(x)) . & (101) \end{aligned}$$

With all of these pieces we are ready to specify the *backpropagation algorithm.*

The backpropagation algorithm

- Initialization: We randomly initialize all weights in our network w_{km}^r for all layers $r = 1, 2, \cdots L$ and for all internal nodes where the index k selects a node from the layer $r - 1$ thus $k = 1, 2, \cdots k_{r-1}$ and the index m selects a node from the layer r and thus $m = 1, 2, \cdots k_r$.

- Forward propagation: Once the weights are assigned values for each sample (x_i, y_i) for $i = 1, 2, \cdots N$ we can evaluate using forward propagation the variables $v_j^r(i)$ using Equation 87 and then $y_j^r(i)$ via $f(v_j^r(i))$. We can also evaluate the individual errors $\mathcal{E}(i)$ using Equation 86 and the total error J using Equation 84

- Then starting at the last layer where $r = L$ for each sample $i = 1, 2, \cdots N$ and for each neuron $j = 1, 2, \cdots k_L$ first compute δ_j^L using Equation 93. Then working backwards compute δ_j^{r-1} using Equation 98 and 99. Since we know δ_j^L we can do this for $r = L, L-1, \cdots, 2$ and for each neuron in layer $r-1$ so $j = 1, 2, \cdots k_{r-1}$.
- Once we have δ_j^r computed we can now update the weights $\mathbf{w}_j^r$ using Equation 82 and 83 with the explicit for form $\Delta\mathbf{w}_j^r$ given by Equation 91. Once we have updated our weight vectors we are ready to apply the same procedure again, i.e. issuing a forward propagation followed by a backwards propagation sweep.

Notes on variations on the backpropagation algorithm

Consider the backpropagation weight update equation with the *momentum factor* $\alpha\mathbf{w}_j^r(t-1)$ given by

$$\Delta\mathbf{w}_j^r(t) = \alpha\Delta\mathbf{w}_j^r(t-1) - \mu\mathbf{g}(t)\,. \tag{102}$$

By writing out this recurrence relationship for $t = T, T-1, T-2, \cdots$ as

$$\begin{aligned}
\Delta\mathbf{w}_j^r(T) &= \alpha\Delta\mathbf{w}_j^r(T-1) - \mu\mathbf{g}(T) \\
&= \alpha\left[\alpha\Delta\mathbf{w}_j^r(T-2) - \mu\mathbf{g}(T-1)\right] - \mu\mathbf{g}(T) \\
&= \alpha^2\Delta\mathbf{w}_j^r(T-2) - \mu\left[\alpha\mathbf{g}(T-1) + \mathbf{g}(T)\right] \\
&= \alpha^2\left[\alpha\Delta\mathbf{w}_j^r(T-3) - \mu\mathbf{g}(T-2)\right] - \mu\left[\alpha\mathbf{g}(T-1) + \mathbf{g}(T)\right] \\
&= \alpha^3\Delta\mathbf{w}_j^r(T-3) - \mu\left[\alpha^2\mathbf{g}(T-2) + \alpha\mathbf{g}(T-1) + \mathbf{g}(T)\right] \\
&\vdots \\
&= \alpha^T\Delta\mathbf{w}_j^r(0) - \mu\sum_{t=0}^{T-1}\alpha^t\mathbf{g}(T-t)\,.
\end{aligned} \tag{103}$$

As we require $\alpha < 1$ then $\alpha^T\Delta\mathbf{w}_j^r(0) \to 0$ as $T \to +\infty$. If we assume that our gradient vector is a constant across time or $\mathbf{g}(T-t) \approx \mathbf{g}$ then we see that $\Delta\mathbf{w}_j^r(T)$ can be approximated as

$$\Delta\mathbf{w}_j^r(T) \approx -\mu\left[1 + \alpha + \alpha^2 + \alpha^3 + \cdots\right]\mathbf{g} = -\mu\left(\frac{1}{1-\alpha}\right)\mathbf{g}\,.$$

Problem Solutions

Problem 4.1 (a simple multilayer perceptron)

The given points for this problem are plotted in the Figure 10 and is generated with the MATLAB script `chap_4_prob_1.m`.. From that figure we see that these points are a scattering of points around the XOR pattern, which we know are not linearly separable, these points are also *not* be linearly separable. We can however separate the two classes in the same way

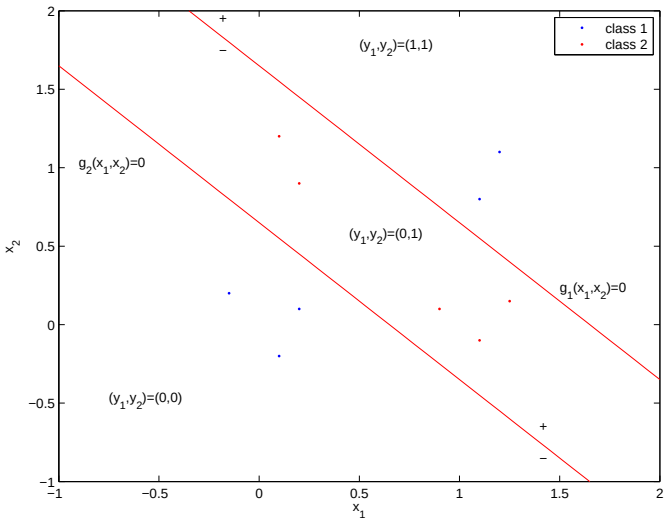

Figure 10: The data points from Problem 4.1 and the two linear discriminant functions $g_1(x_1, x_2) = 0$ and $g_2(x_1, x_2) = 0$, that when combined, separate the two classes.

we did for the XOR pattern. If we introduce two discriminate lines $g_1(x_1, x_2)$ and $g_2(x_1, x_2)$ given by

$$g_1(x_1, x_2) = x_1 + x_2 - 1.65 = 0 \quad \text{and} \quad g_2(x_1, x_2) = x_1 + x_2 - 0.65 = 0\,.$$

Next we introduce threshold variables, y_i, that are mappings of the values taken by g_i when evaluated at a particular pair (x_1, x_2). For example, $y_i = 0$ if $g_i < 0$ and $y_i = 1$ if $g_i \geq 0$. Then we see that the entire (x_1, x_2) space has been mapped to one of three (y_1, y_2) points: $(0, 0)$, $(0, 1)$, and $(1, 1)$ depending on where the point (x_1, x_2) falls relative to the two lines $g_1 = 0$ and $g_2 = 0$. Under the (y_1, y_2) mapping just discussed, only the point $(0, 1)$ is associated with the second class, while the other two points, $(0, 0)$ and $(1, 1)$, are associated with the first class. Given the output (y_1, y_2) our task is now to design a linear hyperplane that will separate the point $(0, 1)$ from the two points $(0, 0)$ and $(1, 1)$. A line that does this is

$$y_2 - y_1 - \frac{1}{2} = 0\,.$$

Problem 4.2 (using a neural net to classify the XOR problem)

See the `R` script `chap_4_prob_2.R` where this problem is worked. When that script is run it produces the plots shown in Figure 12. In that figure we see that after very few iterations the neural network is able to classify the training data almost perfectly. The error on the testing data set decreases initially and then increases as the net overfits and learns information that is not generalizable. The degree to which this is over fitting takes place does not really appear to be that great however.

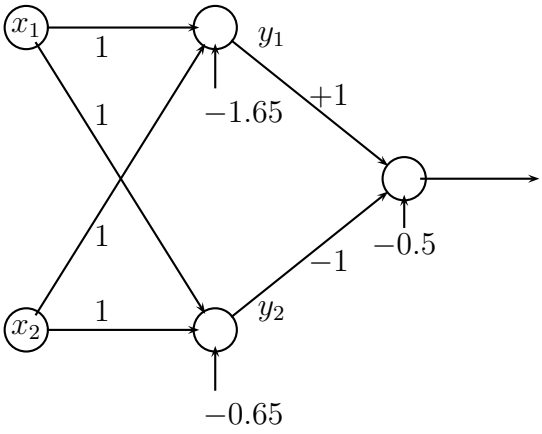

Figure 11: The network suggested by Problem 4.1.

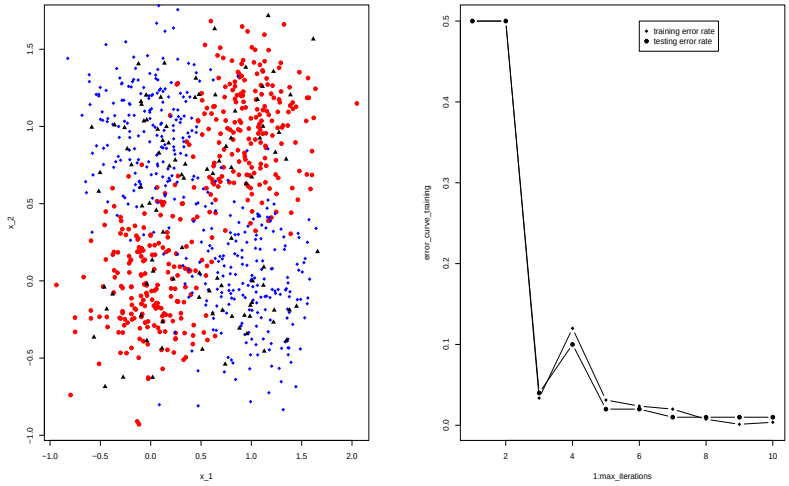

Figure 12: **Left:** Classification of the XOR problem. **Right:** The training and testing error rates as a function of iteration number.

Problem 4.3 (multilayer perceptron based on cube vertexes)

Part (a): The given decision regions for this problem are drawn in Figure 13, which is generated with the MATLAB script `chap_4_prob_3.m`. The vertex to which each region is mapped is specified using the notation $(\pm, \pm, \pm)$, where a minus is mapped to 0 and a plus is mapped to a 1. From the discussion in the book a two layer neural network can classify unions of polyhedrons but not unions of unions. Thus if consider class ω_1 to be composed of the points in the regions $(-,-,-)$, $(+,-,-)$, $(-,+,-)$, and $(+,+,-)$. While the class ω_2 is composed of points taken from the other polyhedrons. Then the ω_1 polyhedrons map to the four points on the (y_1, y_2, y_3) hypercube $(0,0,0)$, $(1,0,0)$, $(0,1,0)$, and $(1,1,0)$, while the other polyhedrons map to the upper four points of this hypercube. These two sets of points in the mapped (y_1, y_2, y_3) space can be separated easily with the hyperplane $y_3 = \frac{1}{2}$. Thus we can implement the desired classifier in this case using the two-layer neural network

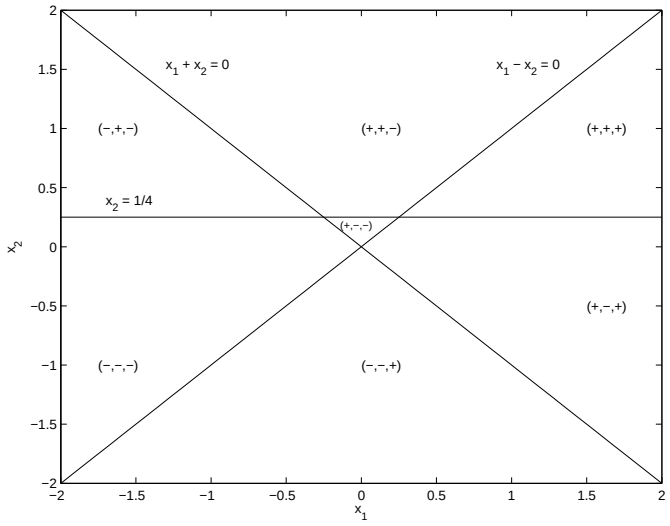

Figure 13: The given decision regions specified for Problem 4.3.

shown in Figure 14.

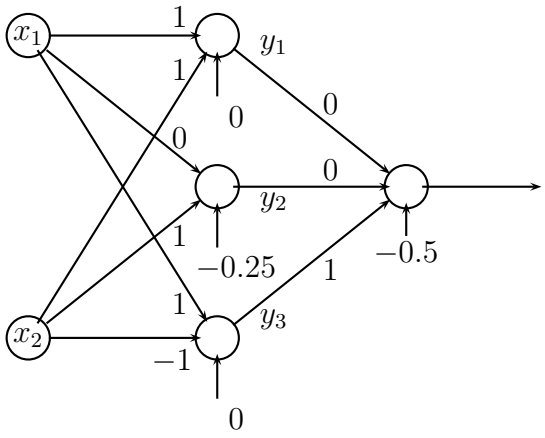

Figure 14: The two layer network for Problem 4.3.

Part (b): To require a three node network it is sufficient to have the mapped classes in the (y_1, y_2, y_3) space mapped to the XOR problem on the unit hypercube. Thus if we pick the points in the polyhedrons $(-,-,-)$ and $(+,+,+)$ to be members of class ω_1 and the points in the other polyhedrons to be from class ω_2 we will require a three layer network to perform classification. In that case we can use an additional layer (the second hidden layer) to further perform the classification. The resulting neural network is given in Figure 15. In that figure we have denoted the output of the two neurons in the second hidden layer as z_1 and z_2. To determine the weights to put on the neurons that feed from the first hidden layer into the second hidden layer in Figure 15 since in the (y_1, y_2, y_3) space this is the XOR problem and we can solve it in the same way that we did in the earlier part of this chapter. That is we will create two planes, one that "cuts off" the vertex $(0, 0, 0)$ from the other

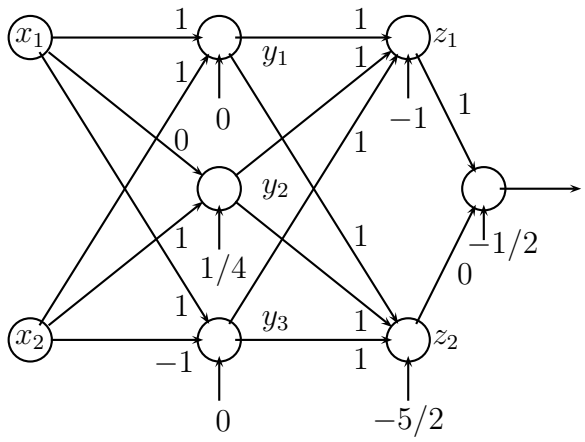

Figure 15: The three layer network for Problem 4.3.

vertexes of the hypercube and the second plane that "cuts off" the node $(1,1,1)$ from the other vertexes of the hypercube. The points in between these two planes will belong to one class and the points outside of these two planes will belong to the other class. For the first plane and the one that cuts off the vertex $(0,0,0)$ of the many possible one plane that does this is the one that passes through the three points

$$(1/2,0,0)\,,\quad (0,1/2,0)\,,\quad (0,0,1/2)\,.$$

While for the second plane and the one that cuts off the vertex $(1,1,1)$ of the many possible planes that do this one plane that works is the one that passes through the three points

$$(1,1,1/2)\,,\quad (1/2,1,1)\,,\quad (1,1/2,1)\,.$$

We thus need to be able obtain the equation for a plane in three space that passes through three points. This is discussed in [9] where it is shown that the equation of a plane

$$c_1x + c_2y + c_3z + c_4 = 0\,,$$

that must pass thought the three points (x_1,y_1,z_1), (x_2,y_2,z_2), and (x_3,y_3,z_3) is given by evaluating

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = 0\,.$$

As an example of this, for the first plane we need to evaluate

$$\begin{vmatrix} x & y & z & 1 \\ 1/2 & 0 & 0 & 1 \\ 1/2 & 0 & 0 & 1 \\ 1/2 & 0 & 0 & 1 \end{vmatrix} = 0\,,$$

or

$$x\begin{vmatrix} 0 & 0 & 1 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 1 \end{vmatrix} - \frac{1}{2}\begin{vmatrix} y & z & 1 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 1 \end{vmatrix} = 0\,,$$

or finally

$$x + y + z - 1 = 0\,.$$

The same procedure for the second plane gives

$$x + y + z - \frac{5}{2} = 0\,.$$

Thus with the above planes we have computed the weights feeding into the second hidden layer. To finish the problem we recognized that with the first plane, only the single point $(y_1, y_2, y_3) = (0, 0, 0)$ is mapped to the value of 0 while all other points are mapped to 1. With the second plane, only the single point $(y_1, y_2, y_3) = (1, 1, 1)$ is mapped to the value of 1 while all other points are mapped to 0. Thus when we threshold on the sign of the values of the two discriminant mappings above we see map the points $(0, 0, 0) \to (0, 0)$, $(1, 1, 1) \to (0, 1)$, and all other (y_1, y_2, y_3) points are mapped to $(1, 0)$. To finish our classification we need to find a hyperplane that splits the two points $(0, 0)$ and $(0, 1)$ from $(1, 0)$. Such a discriminant surface is $z_1 = \frac{1}{2}$, where we assume the second hidden layer maps the points (y_1, y_2, y_3) to the point (z_1, z_2). This final discrimination surface is also represented in Figure 15.

Problem 4.4 (separating the points x_1 and x_2 with a hyperplane)

First recall that the difference vector $x_1 - x_2$ is a vector from the vector x_2 and pointing to the vector x_1, since if we add the vector x_2 to this difference vector we get x_1 i.e.

$$x_2 + (x_1 - x_2) = x_1\,.$$

The midpoint between the two points x_1 and x_2 is the vector $\frac{1}{2}(x_1 + x_2)$. Thus this problem asks to find the plane with a normal vector proportional to $x_1 - x_2$ and passing through the point $x_0 \equiv \frac{1}{2}(x_1 + x_2)$. This means that if we take x to be a vector in the hyperplane then the vector $x - x_0$ must be *orthogonal* to $x_1 - x_2$ or have a zero dot product

$$(x_1 - x_2)^T (x - x_0) = 0\,.$$

Using the definition for x_0 we have this expression is equal to

$$(x_1 - x_2)^T x - \frac{1}{2}(x_1 - x_2)^T (x_1 + x_2) = 0\,,$$

or

$$(x_1 - x_2)^T x - \frac{1}{2}(x_1^T x_1 - x_2^T x_2) = (x_1 - x_2)^T x - \frac{1}{2}||x_1||^2 + \frac{1}{2}||x_2||^2 = 0\,.$$

It remains to show that x_1 is on the positive side of the hyperplane. To show this consider the above expression evaluated at $x = x_1$. We find

$$\begin{aligned}
(x_1 - x_2)^T x - \frac{1}{2}||x_1||^2 + \frac{1}{2}||x_2||^2 &= ||x_1||^2 - x_2^T x_1 - \frac{1}{2}||x_1||^2 + \frac{1}{2}||x_2||^2 \\
&= \frac{1}{2}||x_1||^2 - x_2^T x_1 + \frac{1}{2}||x_2||^2 \\
&= \frac{1}{2}(||x_1||^2 - 2x_2^T x_1 + ||x_2||^2) \\
&= \frac{1}{2}(x_1 - x_2)^T (x_1 - x_2) = \frac{1}{2}||x_1 - x_2||^2\,,
\end{aligned}$$

which is positive showing that x_1 is on the positive side of the above hyperplane.

Problem 4.6 (backpropagation with cross-entropy)

The cross entropy 4.33 is

$$J = -\sum_{i=1}^{N}\sum_{k=1}^{k_L} y_k(i)\ln\left(\frac{\hat{y}_k(i)}{y_k(i)}\right) .$$

Thus we see that $\mathcal{E}(i)$ in this case is given by

$$\mathcal{E}(i) = -\sum_{k=1}^{k_L} y_k(i)\ln\left(\frac{\hat{y}_k(i)}{y_k(i)}\right) .$$

Thus we can evaluate $\delta_j^L(i)$ as

$$\delta_j^L(i) \equiv \frac{\partial \mathcal{E}(i)}{\partial v_j^L(i)} = \frac{\partial}{\partial v_j^L(i)}\left[-\sum_{k=1}^{k_L} y_k(i)\ln\left(\frac{f(v_k^L)}{y_k(i)}\right)\right] .$$

This derivative will select the $k = j$th element out of the sum and gives

$$\delta_j^L(i) = -y_j(i)\frac{\partial}{\partial v_j^L(i)}\left(\ln\left(\frac{f(v_j^L)}{y_k(i)}\right)\right) = -y_j(i)\frac{f'(v_j^L)}{f(v_j^L)} .$$

If the activation function $f(\cdot)$ is the sigmoid function Equation 100 then its derivative is given in Equation 101 where we have $f'(v_j^L) = -af(v_j^L)(1 - f(v_j^L))$ and the above becomes

$$\delta_j^L(i) = ay_j(i)(1 - f(y_j^L)) = ay_j(i)(1 - \hat{y}_j(i)) .$$

Problem 4.7 (backpropagation with softmax)

The softmax activation function has its output $\hat{y}_k$ given by

$$\hat{y}_k \equiv \frac{\exp(v_k^L)}{\sum_{k'}\exp(v_{k'}^L)} . \tag{104}$$

Note that this expression depends on v_j^L in both the numerator and the denominator. Using the result from the previous exercise we find

$$\begin{aligned}
\delta_j^L(i) &\equiv \frac{\partial \mathcal{E}(i)}{\partial v_j^L(i)} \\
&= \frac{\partial}{\partial v_j^L(i)}\left(-\sum_{k=1}^{k_L} y_k(i)\ln\left(\frac{\hat{y}_k}{y_k(i)}\right)\right) \\
&= -\frac{\partial}{\partial v_j^L(i)}\left(y_j(i)\ln\left(\frac{\hat{y}_j}{y_j(i)}\right)\right) - \frac{\partial}{\partial v_j^L(i)}\left(\sum_{k=1;k\neq j}^{k_L} y_k(i)\ln\left(\frac{\hat{y}_k}{y_k(i)}\right)\right) \\
&= -\frac{y_j(i)}{\hat{y}_j}\frac{\partial \hat{y}_j}{\partial v_j^L(i)} - \sum_{k=1;k\neq j}^{k_L}\frac{y_k(i)}{\hat{y}_k}\frac{\partial \hat{y}_k}{\partial v_j^L(i)} .
\end{aligned}$$

To evaluate this we first consider the first term or $\frac{\partial \hat{y}_j}{\partial v_j^L(i)}$ where we find

$$\begin{aligned}\frac{\partial \hat{y}_j}{\partial v_j^L(i)} &= \frac{\partial}{\partial v_j^L(i)}\left(\frac{\exp(v_j^L)}{\sum_{k'}\exp(v_{k'}^L)}\right) \\ &= \frac{\exp(v_j^L)}{\sum_{k'}\exp(v_{k'}^L)} - \frac{\exp(v_j^L)\exp(v_j^L)}{\left(\sum_{k'}\exp(v_{k'}^L)\right)^2} = \hat{y}_j - \hat{y}_j^2 \,.\end{aligned}$$

While for the second term we get (note that $j \neq k$)

$$\frac{\partial \hat{y}_k}{\partial v_j^L(i)} = \frac{\partial}{\partial v_j^L}\left(\frac{\exp(v_k^L)}{\sum_{k'}\exp(v_{k'}^L)}\right) = -\frac{\exp(v_k^L)\exp(v_j^L)}{\left(\sum_{k'}\exp(v_{k'}^L)\right)^2} = -\hat{y}_k\hat{y}_j \,.$$

Thus we find

$$\begin{aligned}\delta_j^L &= -\frac{y_j(i)}{\hat{y}_j}(\hat{y}_j - \hat{y}_j^2) - \sum_{k=1;k\neq j}^{k_L}\frac{y_k(i)}{\hat{y}_k}(-\hat{y}_k\hat{y}_j) \\ &= -y_j(i)(1-\hat{y}_j) + \hat{y}_j\sum_{k=1;k\neq j}^{k_L} y_k(i) \,.\end{aligned}$$

Since $\hat{y}_k(i)$ and $y_k(i)$ are probabilities of class membership we have

$$\sum_{k=1}^{k_L} y_k(i) = 1 \,,$$

and thus $\sum_{k=1;k\neq j}^{k_L} y_k(i) = 1 - y_j(i)$. Using this we find for $\delta_j^L(i)$ that

$$\delta_j^L(i) = -y_j(i) + y_j(i)\hat{y}_j + \hat{y}_j(1-y_j) = \hat{y}_j - y_j(i) \,,$$

the expression we were to show.

Problem 4.9 (the maximum number of polyhedral regions)

The books equation 4.37 is

$$M = \sum_{m=0}^{l}\left(\begin{array}{c} k \\ m \end{array}\right) \quad \text{with} \quad \left(\begin{array}{c} k \\ m \end{array}\right) = 0 \quad \text{if} \quad m > k \,. \tag{105}$$

where M is the maximum number of polyhedral regions possible for a neural network with one hidden layer containing k neurons and an input feature dimension of l. Assuming that $l \geq k$ then

$$M = \sum_{m=0}^{l}\left(\begin{array}{c} k \\ m \end{array}\right) = \sum_{m=0}^{k}\left(\begin{array}{c} k \\ m \end{array}\right) = 2^k \,,$$

where we have used the fact that $\left(\begin{array}{c} k \\ m \end{array}\right) = 0$ to drop all terms in the sum when $m = k+1, k+2, \cdots, l$ if there are any. That the sum of the binomial coefficients sums to 2^k follows from expanding $(1+1)^k$ using the binomial theorem.

Problem 4.12 (an approximation of $\frac{\partial^2 J}{\partial w_{kj}^r \partial w_{k'j'}^{r'}}$)

For J given by

$$J = \frac{1}{2}\sum_{i=1}^{N}\sum_{m=1}^{k_L}(\hat{y}_m(i) - y_m(i))^2 .$$

We will evaluate the second derivatives of this expression. First we take the w_{kj}^r derivative of J directly and find

$$\frac{\partial J}{\partial w_{kj}^r} = \sum_{i=1}^{N}\sum_{m=1}^{k_L}(\hat{y}_m(i) - y_m(i))\frac{\partial \hat{y}_m(i)}{\partial w_{kj}^r} .$$

Next we take the $w_{k'j'}^{r'}$ derivative of this expression. We find

$$\begin{aligned}\frac{\partial^2 J}{\partial w_{kj}^r \partial w_{k'j'}^{r'}} &= \sum_{i=1}^{N}\sum_{m=1}^{k_L}\frac{\partial \hat{y}_m(i)}{\partial w_{k'j'}^{r'}}\frac{\partial \hat{y}_m(i)}{\partial w_{kj}^r} \\ &+ \sum_{i=1}^{N}\sum_{m=1}^{k_L}(\hat{y}_m(i) - y_m(i))\frac{\partial^2 \hat{y}_m(i)}{\partial w_{kj}^r \partial w_{k'j'}^{r'}} .\end{aligned}$$

If we are near a minimum of the objective function we can assume that $\hat{y}_m(i) - y_m(i) \approx 0$ and can thus approximate the above derivative as

$$\frac{\partial^2 J}{\partial w_{kj}^r \partial w_{k'j'}^{r'}} = \sum_{i=1}^{N}\sum_{m=1}^{k_L}\frac{\partial \hat{y}_m(i)}{\partial w_{kj}^r}\frac{\partial \hat{y}_m(i)}{\partial w_{k'j'}^{r'}} ,$$

showing that we can approximate the second derivative by products of the first order ones. Recall that the variable w_{kj}^r represent the weight from neuron k in layer $r-1$ to the neuron j in layer r. We would expect that the effect of changes in w_{kj}^r on the output $\hat{y}_m(i)$ would be propagated though the variables $v_j^r(i)$. From the chain rule we have

$$\frac{\partial \hat{y}_m(i)}{\partial w_{kj}^r} = \frac{\partial \hat{y}_m(i)}{\partial v_j^r(i)}\frac{\partial v_j^r}{\partial w_{kj}^r} .$$

Using Equation 89 we see that $\frac{\partial v_j^r}{\partial w_{kj}^r} = y_k^{r-1}$ and thus we get

$$\frac{\partial \hat{y}_m(i)}{\partial w_{kj}^r} = \frac{\partial \hat{y}_m(i)}{\partial v_j^r(i)} y_k^{r-1} ,$$

which if we define $\frac{\partial \hat{y}_m(i)}{\partial v_j^r(i)} \equiv \hat{\delta}_{jm}^r$ is the expression we wanted to show.

Problem 4.13 (approximating the Hessian)

We will assume that the weight notation for this problem is the same as in the book where by the expression w_{jk}^r is the weight *from* the neuron k in the $r-1$st layer *into* the neuron j in the r-th layer.

- Using Equation 88 and Equation 89 from the chain rule we have (dropping the i dependence)

$$\frac{\partial^2 \mathcal{E}}{(\partial w_{jk}^r)^2} = \frac{\partial v_j^r}{\partial w_{jk}^r} \frac{\partial}{\partial v_j^r} \left(\frac{\partial \mathcal{E}}{\partial v_j^r} y_k^{r-1} \right) .$$

Since the input y_k^{r-1} is independent of v_j^r we get

$$\frac{\partial^2 \mathcal{E}}{(\partial w_{jk}^r)^2} = (y_k^{r-1})^2 \frac{\partial^2 \mathcal{E}}{(\partial v_j^r)^2} . \tag{106}$$

- Using Equation 93 we get

$$\begin{aligned} \frac{\partial^2 \mathcal{E}}{(\partial v_j^L)^2} &= -f''(v_j^L) e_j - \left[\frac{\partial}{\partial v_j^L} (y_j - f(v_j^L)) \right] f'(v_j^L) \\ &= -f''(v_j^L) e_j + f'(v_j^L)^2 . \end{aligned}$$

- Now to evaluate $\frac{\partial^2 \mathcal{E}}{(\partial v_j^{r-1})^2}$ we note that from Equation 97 and Equation 90 we have

$$\frac{\partial \mathcal{E}}{\partial v_j^{r-1}} = \left[\sum_{k=1}^{k_r} \delta_k^r w_{kj}^r \right] f'(v_j^r) ,$$

Thus the v_j^{r-1} derivative of this is given by

$$\frac{\partial^2 \mathcal{E}}{(\partial v_j^{r-1})^2} = f'(v_j^{r-1}) \sum_{k=1}^{k_r} w_{kj}^r \frac{\partial \delta_k^r}{\partial v_j^{r-1}} + f''(v_j^{r-1}) \left[\sum_{k=1}^{k_r} \delta_k^r w_{kj}^r \right] .$$

Thus we need to evaluate $\frac{\partial \delta_k^r}{\partial v_j^{r-1}}$. To do this we will use the definition of δ_k^r given by Equation 90, an expression like Equation 94 and subsequent developments following that equation, namely

$$\begin{aligned} \frac{\partial}{\partial v_j^{r-1}} \left(\frac{\partial \mathcal{E}}{\partial v_k^r} \right) &= \sum_{k'=1}^{k_r} \frac{\partial}{\partial v_{k'}^r} \left(\frac{\partial \mathcal{E}}{\partial v_k^r} \right) \frac{\partial v_{k'}^r}{\partial v_j^{r-1}} \\ &= f'(v_j^{r-1}) \sum_{k'=1}^{k_r} w_{k'j}^r \frac{\partial^2 \mathcal{E}}{\partial v_{k'}^r \partial v_k^r} . \end{aligned}$$

Dropping all off-diagonal terms in the summation above we keep only the $k' = k$ element and find

$$\frac{\partial \delta_k^r}{\partial v_j^{r-1}} = f'(v_j^{r-1}) w_{kj}^r \frac{\partial^2 \mathcal{E}}{(\partial v_k^r)^2} .$$

Using this we finally get for $\frac{\partial^2 \mathcal{E}}{(\partial v_j^{r-1})^2}$ the following

$$\frac{\partial^2 \mathcal{E}}{(\partial v_j^{r-1})^2} = (f'(v_j^{r-1}))^2 \sum_{k=1}^{k_r} (w_{kj}^r)^2 \frac{\partial^2 \mathcal{E}}{(\partial v_k^r)^2} + f''(v_j^{r-1}) \left[\sum_{k=1}^{k_r} \delta_k^r w_{kj}^r \right] .$$

Note that this expression is *different* than that given in the book in that the first term in the book has a summation with an argument of $\frac{\partial^2 \mathcal{E}}{(\partial v_j^r)^2}$ (note the j subscript) rather than $\frac{\partial^2 \mathcal{E}}{(\partial v_k^r)^2}$ (with a k subscript). Since the first of these two expressions is independent of k it could be taken out of the summation making me think the book has a typo in its equation. Please contact me if anyone sees any errors in this derivation.

Problem 4.15 (a different activation function)

When one changes the activation function in the backpropagation algorithm what changes is the function we use to evaluate any expression with $f(\cdot)$ or $f'(\cdot)$, for example in Equations 93 and 98. One of the nice things about the backpropagation algorithm is that calls to the activation function f and its derivative f' can simply be viewed as algorithmic "subroutines" that can be replaced and modified if needed. For the suggested hyperbolic tangent function $f(x)$ given by

$$f(x) = c\tanh(bx)\,, \tag{107}$$

we have its derivative given by

$$f'(x) = c\,b\,\mathrm{sech}^2(bx)\,.$$

From the identity $\cosh^2(x) - \sinh^2(x) = 1$, by dividing by $\cosh^2(x)$ we can conclude that $\mathrm{sech}^2(x) = 1 - \tanh^2(x)$ and thus

$$\begin{aligned} f'(x) &= cb(1 - \tanh^2(bx)) \\ &= b\left(1 - \frac{f(x)^2}{c}\right)\,. \end{aligned} \tag{108}$$

These two functions then need to be implemented to use this activation function.

Problem 4.16 (an iteration dependent learning parameter μ)

A Taylor expansion of $\frac{1}{1+\frac{t}{t_0}}$ or

$$\frac{1}{1+\frac{t}{t_0}} \approx 1 - \frac{t}{t_0} + \frac{t^2}{t_0^2} + \cdots\,.$$

Thus when $t \ll t_0$ the fraction $\frac{1}{1+\frac{t}{t_0}} \approx 1$ to leading order and thus $\mu \approx \mu_0$. On the other hand when $t \gg t_0$ we have that $1 + \frac{t}{t_0} \approx \frac{t}{t_0}$ and the fraction above is given by

$$\frac{1}{1+\frac{t}{t_0}} \approx \frac{1}{\frac{t}{t_0}} = \frac{t_0}{t}\,.$$

Thus in this stage of the iterations $\mu(t)$ decreases inversely in proportion to t.

Problem 4.17 (using a neural net as a function approximation)

This problem is worked in the `R` script `chap_4_prob_17.R`. When that script is run it produces a plot like that shown in Figure 16. The neural network with two hidden nodes was created using the `nnet` command from the `nnet` package. We see that the neural network in this case does a very good job approximating the true function.

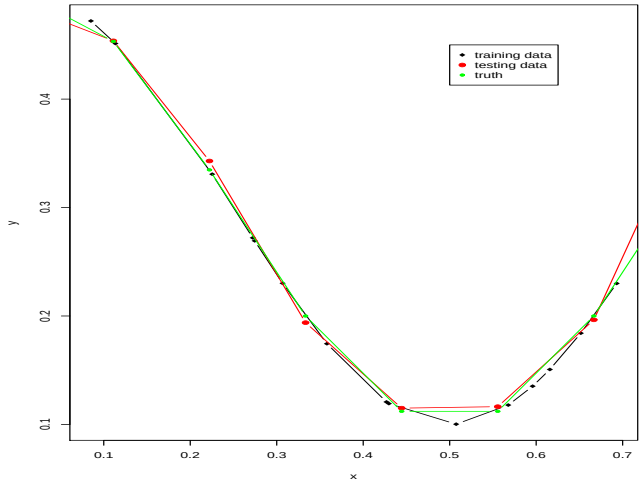

Figure 16: The function to fit and its neural network approximation for Problem 4.17.

Problem 4.19 (when $N = 2(l+1)$ the number of dichotomies is 2^{N-1})

We have

$$O(N, l) = 2 \sum_{i=0}^{l} \left(\begin{array}{c} N-1 \\ i \end{array} \right) , \tag{109}$$

where N is the number of points embedded in a space of dimension l and $O(N, l)$ is the number of groupings that can be formed by hyperplanes in $\mathbb{R}^l$ to separate the points into two classes. If $N = 2(l+1)$ then

$$O(2(l+1), l) = 2 \sum_{i=0}^{l} \left(\begin{array}{c} 2l+1 \\ i \end{array} \right) .$$

Given the identity

$$\left(\begin{array}{c} 2n+1 \\ n-i+1 \end{array} \right) = \left(\begin{array}{c} 2n+1 \\ n+i \end{array} \right) , \tag{110}$$

by taking $i = n+1, n, n-1, \cdots, 1$ we get the following equivalences

$$\begin{aligned}
\begin{pmatrix} 2n+1 \\ 0 \end{pmatrix} &= \begin{pmatrix} 2n+1 \\ 2n+1 \end{pmatrix} \\
\begin{pmatrix} 2n+1 \\ 1 \end{pmatrix} &= \begin{pmatrix} 2n+1 \\ 2n \end{pmatrix} \\
\begin{pmatrix} 2n+1 \\ 2 \end{pmatrix} &= \begin{pmatrix} 2n+1 \\ 2n-1 \end{pmatrix} \\
&\vdots \\
\begin{pmatrix} 2n+1 \\ n-1 \end{pmatrix} &= \begin{pmatrix} 2n+1 \\ n+2 \end{pmatrix} \\
\begin{pmatrix} 2n+1 \\ n \end{pmatrix} &= \begin{pmatrix} 2n+1 \\ n+1 \end{pmatrix}
\end{aligned}$$

Now write $O(2(l+1), l)$ as

$$\sum_{i=0}^{l} \begin{pmatrix} 2l+1 \\ i \end{pmatrix} + \sum_{i=0}^{l} \begin{pmatrix} 2l+1 \\ i \end{pmatrix},$$

or two sums of the same thing. Next note that using the above identities we can write the second sum as

$$\begin{aligned}
\sum_{i=0}^{l} \begin{pmatrix} 2l+1 \\ i \end{pmatrix} &= \begin{pmatrix} 2l+1 \\ 0 \end{pmatrix} + \begin{pmatrix} 2l+1 \\ 1 \end{pmatrix} + \cdots + \begin{pmatrix} 2l+1 \\ l-1 \end{pmatrix} + \begin{pmatrix} 2l+1 \\ l \end{pmatrix} \\
&= \begin{pmatrix} 2l+1 \\ 2l+1 \end{pmatrix} + \begin{pmatrix} 2l+1 \\ 2l \end{pmatrix} + \cdots + \begin{pmatrix} 2l+1 \\ l+2 \end{pmatrix} + \begin{pmatrix} 2l+1 \\ l+1 \end{pmatrix} \\
&= \sum_{i=l+1}^{2l+1} \begin{pmatrix} 2l+1 \\ i \end{pmatrix}.
\end{aligned}$$

Thus using this expression we have that

$$O(2(l+1), l) = \sum_{i=0}^{l} \begin{pmatrix} 2l+1 \\ i \end{pmatrix} + \sum_{i=l+1}^{2l+1} \begin{pmatrix} 2l+1 \\ i \end{pmatrix} = \sum_{i=0}^{2l+1} \begin{pmatrix} 2l+1 \\ i \end{pmatrix} = 2^{2l+1}.$$

Since $2l+1 = N-1$ we have that $O(2(l+1), l) = 2^{N-1}$ as we were to show.

Problem 4.22 (the kernel trick)

From the given mapping $\phi(x)$ we have that

$$\begin{aligned}
y_i^T y_j &= \phi(x_i)^T \phi(x_j) \\
&= \frac{1}{2} + \cos(x_i)\cos(x_j) + \cos(2x_i)\cos(2x_j) + \cdots + \cos(kx_i)\cos(kx_j) \\
&+ \sin(x_i)\sin(x_j) + \sin(2x_i)\sin(2x_j) + \cdots + \sin(kx_i)\sin(kx_j).
\end{aligned}$$

Since $\cos(\alpha)\cos(\beta)+\sin(\alpha)\sin(\beta) = \cos(\alpha-\beta)$ we can match cosigns with sines in the above expression and simplify a bit to get

$$y_i^T y_j = \frac{1}{2} + \cos(x_i - x_j) + \cos(2(x_i - x_j)) + \cdots + \cos(k(x_i - x_j)) \,.$$

To evaluate this sum we note that by writing the cosigns above in terms of their exponential representation and using the geometric series we can show that

$$1 + 2\cos(\alpha) + 2\cos(2\alpha) + 2\cos(3\alpha) + \cdots + 2\cos(n\alpha) = \frac{\sin\left(\left(n+\frac{1}{2}\right)\alpha\right)}{\sin\left(\frac{x}{2}\right)} \,. \tag{111}$$

Thus using this we can show that $y_i^T y_j$ is given by

$$\frac{1}{2}\frac{\sin\left(\left(k+\frac{1}{2}\right)(x_i - x_j)\right)}{\sin\left(\frac{x}{2}\right)} \,,$$

as we were to show.

Feature Selection

Notes on the text

Notes on Data Normalization

First recall that for small y we have $e^{-y} \approx 1 - y + \frac{1}{2}y^2 + \cdots$, thus

$$\hat{x}_{ik} = \frac{1}{1+e^{-y}} \approx \frac{1}{1+1-y+\frac{1}{2}y^2} = \frac{1}{2}\left(\frac{1}{1-\frac{y}{2}+\frac{y^2}{4}+\cdots}\right).$$

Next recall that for small v we have $\frac{1}{1-v} = \sum_{k=0}^{\infty} v^k$ thus we get

$$\begin{aligned}\frac{1}{1-\frac{y}{2}+\frac{y^2}{4}+\cdots} &\approx 1 + \left(\frac{y}{2} - \frac{y^2}{4} + \cdots\right) + \left(\frac{y}{2} - \frac{y^2}{4} + \cdots\right)^2 + \cdots \\ &= 1 + \frac{y}{2} - \frac{y^2}{4} - \frac{y^2}{4} - \frac{y^3}{4} + \cdots = 1 + \frac{y}{2}\cdots .\end{aligned}$$

which is a linear function of y as claimed.

Notes on the Unknown Variance Case

Consider the expectation

$$\begin{aligned}E[(x_i - \mu + \mu - \bar{x})^2] &= E[(x_i-\mu)^2 + 2(x_i-\mu)(\mu-\bar{x}) + (\mu-\bar{x})^2] \\ &= \sigma^2 + 2E[(x_i-\mu)(\mu-\bar{x})] + \frac{\sigma^2}{2}.\end{aligned}$$

We can evaluate the inner expectation using

$$\begin{aligned}E[(x_i-\mu)(\mu-\bar{x})] &= E[(x_i-\mu)\left(\frac{1}{N}\sum_{i'=1}^{N}\mu - \frac{1}{N}\sum_{i'=1}^{N}x_{i'}\right)] \\ &= -\frac{1}{N}\sum_{i'=1}^{N}E[(x_i-\mu)(x_{i'}-\mu)] = -\frac{1}{N}E[(x_i-\mu)^2],\end{aligned}$$

since by independence $E[(x_i-\mu)(x_j-\mu)] = 0$ if $i \neq j$. Since $E[(x_i-\mu)^2] = \sigma^2$ we get

$$\begin{aligned}E[\hat{\sigma}^2] &= \frac{1}{N-1}\sum_{i=1}^{N}\left(\sigma^2 - 2\left(\frac{\sigma^2}{N}\right) + \frac{\sigma^2}{N}\right) \\ &= \frac{N}{N-1}\left(1-\frac{1}{N}\right)\sigma^2 = \sigma^2.\end{aligned}$$

Notes on Example 5.3

See the `R` script `chap_5_example_5_3.R` for code that duplicates the results from this example.

Notes on the derivation of the divergence for Gaussian distributions

When the conditional densities are Gaussian we have

$$\begin{aligned} p(x|\omega_i) &\sim N(\mu_i, \Sigma_i) \\ p(x|\omega_j) &\sim N(\mu_j, \Sigma_j)\,. \end{aligned}$$

Then to compute the divergence d_{ij} given by

$$d_{ij} = \int_{-\infty}^{\infty} (p(x|\omega_i) - p(x|\omega_j)) \ln\left(\frac{p(x|\omega_i)}{p(x|\omega_j)}\right) dx\,, \tag{112}$$

we first need to compute the log term $\ln\left(\frac{p(x|\omega_i)}{p(x|\omega_j)}\right)$, where we find

$$\ln\left(\frac{p(x|\omega_i)}{p(x|\omega_j)}\right) = -\frac{1}{2}\left[(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i) - (x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j)\right] + \frac{1}{2}\ln\left(\frac{|\Sigma_j|}{|\Sigma_i|}\right)\,.$$

When we expand the quadratics above we get

$$\begin{aligned} \ln\left(\frac{p(x|\omega_i)}{p(x|\omega_j)}\right) &= -\frac{1}{2}\left[x^T\Sigma_i^{-1}x - x^T\Sigma_j^{-1}x - 2\mu_i^T\Sigma_i^{-1}x + 2\mu_j^T\Sigma_j^{-1}x\right] \\ &\quad - \frac{1}{2}\left[\mu_i^T\Sigma_i^{-1}\mu_i - \mu_j^T\Sigma_j^{-1}\mu_j\right] + \frac{1}{2}\ln\left(\frac{|\Sigma_j|}{|\Sigma_i|}\right)\,. \end{aligned}$$

Only the first four terms depend on x while the remaining terms are independent of x and can be represented by a constant C. Because the densities $p(x|\cdot)$ are normalized we note that

$$\int_{-\infty}^{\infty} (p(x|\omega_i) - p(x|\omega_j))C dx = C(1-1) = 0\,,$$

and these terms do not affect the divergence. Thus we only need to worry about how to integrate the first four terms. To do these lets first consider the integral of these terms against $p(x|\omega_i)$ (integrating against $p(x|\omega_j)$ will be similar). To do these integral we will use Equation 295 from Appendix A to evaluate the integral of the terms like $x^T\Sigma^{-1}x$, against $p(x|\omega_i)$. When we do that we find the integral of the log ratio term expressed above is given by (multiplied by $-1/2$)

$$\begin{aligned} -2\int_{-\infty}^{\infty} \ln\left(\frac{p(x|\omega_i)}{p(x|\omega_j)}\right) p(x|\omega_i)dx &= \mu_i^T\Sigma_i^{-1}\mu_i + \mathrm{trace}(\Sigma_i\Sigma_i^{-1}) - \mu_i^T\Sigma_j^{-1}\mu_i - \mathrm{trace}(\Sigma_i\Sigma_j^{-1}) \\ &\quad - 2\mu_i^T\Sigma_i^{-1}\mu_i + 2\mu_j^T\Sigma_j^{-1}\mu_i \\ &= -\mu_i^T\Sigma_i^{-1}\mu_i - \mu_i^T\Sigma_j^{-1}\mu_i + 2\mu_j^T\Sigma_j^{-1}\mu_i \\ &\quad + \mathrm{trace}(I) - \mathrm{trace}(\Sigma_i\Sigma_j^{-1})\,. \end{aligned}$$

In the same way the integral of the log ratio term against $p(x|\omega_j)$ is given by

$$\begin{aligned} 2\int_{-\infty}^{\infty} \ln\left(\frac{p(x|\omega_i)}{p(x|\omega_j)}\right) p(x|\omega_j)dx &= -\mu_j^T\Sigma_i^{-1}\mu_j - \text{trace}(\Sigma_j\Sigma_i^{-1}) + \mu_j^T\Sigma_j^{-1}\mu_j + \text{trace}(\Sigma_j\Sigma_j^{-1}) \\ &+ 2\mu_i^T\Sigma_i^{-1}\mu_j - 2\mu_j^T\Sigma_j^{-1}\mu_j \\ &= -\mu_j^T\Sigma_j^{-1}\mu_j - \mu_j^T\Sigma_i^{-1}\mu_j + 2\mu_i^T\Sigma_i^{-1}\mu_j \\ &+ \text{trace}(I) - \text{trace}(\Sigma_j\Sigma_i^{-1})\,. \end{aligned}$$

If we take -1 of the first and second expression and add them together we get two types of terms. Terms involving the trace operation and terms that don't depend on the trace. The trace terms add to give

$$\begin{aligned} \text{trace terms} &= -\text{trace}(I) + \text{trace}(\Sigma_i\Sigma_j^{-1}) - \text{trace}(I) + \text{trace}(\Sigma_j\Sigma_i^{-1}) \\ &= -2\text{trace}(I) + \text{trace}(\Sigma_i\Sigma_j^{-1}) + \text{trace}(\Sigma_j\Sigma_i^{-1})\,. \end{aligned}$$

The non-trace terms add together to give

$$\begin{aligned} \text{non-trace terms} &= \mu_i^T\Sigma_i^{-1}\mu_i + \mu_i^T\Sigma_j^{-1}\mu_i - 2\mu_j^T\Sigma_j^{-1}\mu_i \\ &+ \mu_j^T\Sigma_j^{-1}\mu_j + \mu_j^T\Sigma_i^{-1}\mu_j - 2\mu_i^T\Sigma_i^{-1}\mu_j \\ &= \mu_i^T(\Sigma_i^{-1}\mu_i + \Sigma_j^{-1}\mu_i - 2\Sigma_j^{-1}\mu_j - 2\Sigma_i^{-1}\mu_j) + \mu_j^T(\Sigma_i^{-1}\mu_j + \Sigma_j^{-1}\mu_j) \\ &= \mu_i^T((\Sigma_i^{-1} + \Sigma_j^{-1})\mu_i - 2(\Sigma_i^{-1} + \Sigma_j^{-1})\mu_j) + \mu_j^T(\Sigma_i^{-1} + \Sigma_j^{-1})\mu_j \\ &= \mu_i^T(\Sigma_i^{-1} + \Sigma_j^{-1})(\mu_i - \mu_j) - \mu_i^T(\Sigma_i^{-1} + \Sigma_j^{-1})\mu_j + \mu_j^T(\Sigma_i^{-1} + \Sigma_j^{-1})\mu_j \\ &= \mu_i^T(\Sigma_i^{-1} + \Sigma_j^{-1})(\mu_i - \mu_j) - (\mu_i^T - \mu_j^T)(\Sigma_i^{-1} + \Sigma_j^{-1})\mu_j \\ &= (\mu_i - \mu_j)^T(\Sigma_i^{-1} + \Sigma_j^{-1})\mu_i - (\mu_i - \mu_j)^T(\Sigma_i^{-1} + \Sigma_j^{-1})\mu_j \\ &= (\mu_i - \mu_j)(\Sigma_i^{-1} + \Sigma_j^{-1})(\mu_i - \mu_j)\,. \end{aligned}$$

In total when we divide by 2 and add together the trace and the non-trace expressions we get

$$d_{ij} = \frac{1}{2}(\mu_i - \mu_j)(\Sigma_i^{-1} + \Sigma_j^{-1})(\mu_i - \mu_j) + \frac{1}{2}\text{trace}(\Sigma_i\Sigma_j^{-1} + \Sigma_j\Sigma_i^{-1} - 2I)\,, \tag{113}$$

for the expression for the divergence between two Gaussians.

Notes on Example 5.4

From the expression for B derived in the book

$$B = \frac{l}{2}\log\left(\frac{\sigma_1^2 + \sigma_2^2}{2\sigma_1\sigma_2}\right)\,,$$

if we consider $r = \frac{\sigma_2}{\sigma_1}$ and put in $\sigma_2 = r\sigma_1$ in to the above we get

$$B = \frac{l}{2}\log\left(\frac{1 + r^2}{2r}\right) \to +\infty\,,$$

as $r \to 0$. Thus

$$P_e \leq \sqrt{P(\omega_1)P(\omega_2)}e^{-B} \to 0\,.$$

Notes on scatter matrices

Recall the definitions of the **within-class scatter matrix**, S_w, given by

$$S_w = \sum_{i=1}^{M} P_i E[(x-\mu_i)(x-\mu_i)^T | x \in \omega_i] \,, \tag{114}$$

and the definition of the **between-class scatter matrix**, S_b, given by

$$S_b = \sum_{i=1}^{M} P_i (\mu_i - \mu_0)(\mu_i - \mu_0)^T \,, \tag{115}$$

The variable μ_0 is the "global mean vector" or mean over all features independent of class. We can show that this is equivalent to the expression given in the book as follows

$$\mu_0 = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{1}{N}\sum_{i=1}^{M}\sum_{j=1}^{n_i} x_j = \frac{1}{N}\sum_{i=1}^{M} n_i \mu_i = \sum_{i=1}^{M} P_i \mu_i \,, \tag{116}$$

Where in the second sum above we mean to sum only over those features x that are members of class i. Here M is the number of classes. Note that *means* either μ_i (the class specific means) and μ_0 (the global mean) are linear functions of the raw features x. Thus if we consider a linear transformation of x such as

$$y = A^T x \,,$$

then x "means" denoted by μ_x transform into y "means" in the expected manner

$$\mu_y = A^T \mu_x \,.$$

From the definitions of the x based scatter matrices S_{wx} and S_{bx} given above and how the x based μ's change under a linear transformation we see that

$$S_{yw} = A^T S_{xw} A \quad \text{and} \quad S_{yb} = A^T S_{xb} A \,. \tag{117}$$

We will use these two results when we pick the transformation A^T so that the transformed vectors y are optimal in some way.

If we consider a single feature (a scalar) in the two class case where was assume with equal class probabilities we have for S_w and S_b the following expressions

$$\begin{aligned} S_w &= \frac{1}{2}(S_1 + S_2) = \frac{1}{2}(\sigma_1^2 + \sigma_2^2) \\ S_b &= \frac{1}{2}((\mu_1 - \mu_0)^2 + (\mu_2 - \mu_0)^2) \,. \end{aligned}$$

Since $\mu_0 = \frac{1}{2}(\mu_1 + \mu_2)$ we find the differences needed in the expression for S_b given by

$$\mu_1 - \mu_0 = \frac{1}{2}(\mu_1 - \mu_2) \quad \text{and} \quad \mu_2 - \mu_0 = \frac{1}{2}(\mu_2 - \mu_1) \,,$$

and we have $S_b = (\mu_1 - \mu_2)^2$. Thus

$$J_2' = \frac{|S_b|}{|S_w|} \propto \frac{(\mu_1 - \mu_2)^2}{(\sigma_1^2 + \sigma_2^2)} .$$

This later expression is known as *Fisher's discriminant ration* or FDR. The book give a multidimensional generalization of

$$FDR_{\text{md}} = \sum_{i=1}^{M} \sum_{j \neq i}^{M} \frac{(\mu_i - \mu_j)^2}{\sigma_i^2 + \sigma_j^2} ,$$

but I would think that one would want to incorporate the class priors as was done in the multiclass generalization of the divergence via

$$FDR_{\text{md}} = \sum_{i=1}^{M} \sum_{j \neq i}^{M} \frac{(\mu_i - \mu_j)^2}{\sigma_i^2 + \sigma_j^2} P(\omega_i) P(\omega_j) .$$

Notes on sequential backwards selection (SBS)

Here we derive the number of times we evaluate the class separability metric when using sequential backwards selection to find a suboptimal collection of l features. We start with the initial m features, and begin by evaluating the class separability measure $J(\cdot)$ using the full m dimensional feature vector. This results in 1 evaluation of J. We then sequentially remove each of the m features from the full feature set and evaluate J on using each reduced vector. This requires m evaluations of J. We select the set of features of size $m-1$ that gives the largest value of J. Using this selection of variables we continue this process by sequentially removing each variable to obtain a set of $m-1$ vectors each of dimension $m-2$ and evaluate J on each. This requires $m-1$ evaluations. Thus now we have performed

$$1 + m + m - 1 ,$$

evaluations of J to select the vector of size $m-2$. If we continue this pattern one more step we will do

$$1 + m + m - 1 + m - 2 ,$$

evaluations of J to select the optimal set of features of size $m-3$. Generalizing this we need

$$1 + m + m - 1 + m - 2 + \cdots + l + 1 ,$$

evaluations of J to select the optimal set of features of size l. This can be simplified as

$$\begin{aligned} 1 + \sum_{k=l+1}^{m} k &= 1 + \sum_{k=1}^{m} k - \sum_{k=1}^{l} k \\ &= 1 + \frac{1}{2} m(m+1) - \frac{1}{2} l(l+1) . \end{aligned}$$

A simple python implementation of this search procedure is given in `backwards_selection.py` and `backwards_selection_run_best_subsets.py`.

Notes on sequential forwards selection (SFS)

In the same way as in sequential backwards selection we start by performing m evaluations of J to pick the best single feature. After this feature is selected we then need to evaluate J, $m-1$ times to pick the set of two features that is best. After the best two features are picked we need to evaluate J $m-2$ time to pick the best set of three features. This process continues until we have selected l features. Thus we have in total

$$\begin{aligned}\sum_{k=m-(l-1)}^{m} k &= \sum_{k=1}^{m} k - \sum_{k=1}^{m-l} \\ &= \frac{1}{2}m(m+1) - \frac{1}{2}(m-l)(m-l+1) \\ &= lm - \frac{1}{2}l(l-1)\,,\end{aligned}$$

when we simplify. A simple python implementation of this search procedure is given in `forward_selection.py` and `forward_selection_run_best_subsets.py`.

Notes on optimal feature generation

For J_3 defined as $\text{trace}(S_w^{-1}S_m)$ when we perform a linear transformation of the raw input feature vectors x as $y = A^T x$, the scatter matrices transform as given by Equation 117 or $S_{yw} = A^T S_{xw} A$ and $S_{ym} = A^T S_{xm} A$ the objective J_3 as a function of A becomes

$$J_3(A) = \text{trace}((A^T S_{xw} A)^{-1}(A^T S_{xm} A))\,. \tag{118}$$

We would like to pick the value of A such that when we map the input features y under A^T the value of $J_3(A)$ is maximal. Taking the derivative of the above and using the results on Page 99 we get that the equation $\frac{\partial J_3(A)}{\partial A} = 0$ imply (when we postmultiply by $A^T S_{xw} A$)

$$S_{xw}A(A^T S_{xw}A)^{-1}(A^T S_{xb}A) = S_{xb}A\,.$$

But because the *transformed* scatter matrices S_{yw} and S_{yb} are given by $A^T S_{xw} A$ and $A^T S_{xb} A$ respectively by using these expressions and premultipling the above by S_{xw}^{-1}, we can write the above expression as

$$AS_{yw}^{-1}S_{yb} = S_{xw}^{-1}S_{xb}A\,. \tag{119}$$

Note that this expression has scatter matrices in terms of y on the left and x on the right. When written in this form, this expression "hides" the A matrices in the definition of S_{yw} and S_{yb}. Since we don't know A we can't directly compute the matrices S_{yw} and S_{yb}. Assuming for the moment that we could compute these two matrices, since they are both symmetric we can find an *invertible* matrix B such that diagonalizes both S_{yw} and S_{yb}. This means that there is an invertible linear transformation B such that

$$B^T S_{yw} B = I \quad \text{and} \quad B^T S_{yb} B = D\,,$$

where D is a diagonal matrix[2]. This means that in terms of B and D we can write S_{yw}^{-1} and S_{yb} as

$$S_{yw}^{-1} = (B^{-T}B^{-1})^{-1} = BB^T \quad \text{and} \quad S_{yb} = B^{-T}DB^{-1}\,.$$

If we use these expressions after we postmultiply Equation 119 by B we find

$$\begin{aligned}(S_{xw}^{-1}S_{xb})AB &= AS_{yw}^{-1}S_{yb}B = ABB^TB^{-T}DB^{-1}B \\ &= ABD\,.\end{aligned}$$

If we let $C \equiv AB$ we have

$$S_{xw}^{-1}S_{xb}C = CD\,. \tag{120}$$

This is an eigenvalue problem where columns of C are the eigenvectors of $S_{xw}^{-1}S_{xb}$ and D is a diagonal matrix with the eigenvectors on the diagonal.

To complete this discussion we now need to decide *which* of the eigenvectors of $S_{xw}^{-1}S_{xb}$ we are going to select as the columns of C. In an M class problem the rank of S_{xb} is at most $M-1$. Thus the rank of the product matrix $S_{xw}^{-1}S_{xb}$ is at most $M-1$. Thus we can have at most $M-1$ non-zero eigenvalues and thus there can be at most $M-1$ associated eigenvectors.

Question: These eigenvalues are positive, but I currently don't see an argument why that needs to be so. If anyone knows of one please let me know.

Since we are asked to take the m original features from the vector x and optimally (with respect to J_3) linearly transform them into a smaller set l features the largest l can be is $M-1$.

- If $l = M-1$ then we should take C to have columns represented by *all* of the non-zero eigenvectors of $S_{xw}^{-1}S_{xb}$. This will have the same maximal value for J_3 in that in the original space of x J_3 has the value

  $$J_{3,x} = \text{trace}(S_{xw}^{-1}S_{xb}) = \lambda_1 + \lambda_2 + \cdots + \lambda_{M-1}\,,$$

  since a matrix trace is equivalent to the sum of that matrices eigenvalues. While after performing the C^T transformation on x or $\hat{y} = C^Tx$ we have J_3 given by Equation 118 or

  $$J_{3,\hat{y}} = \text{trace}((C^TS_{xw}C)^{-1}(C^TS_{xb}C))\,.$$

  To evaluate this expression recall that C is the matrix in Equation 120 or $S_{xb}C = S_{xw}CD$. If we premultiply this by C^T we get

  $$C^TS_{xb}C = C^TS_{xw}CD\,,$$

  so

  $$(C^TS_{xw}C)^{-1}(C^TS_{xb}C) = D\,.$$

  Thus

  $$\text{trace}((C^TS_{xw}C)^{-1}(C^TS_{xb}C)) = \text{trace}(D) = \lambda_1 + \lambda_2 + \cdots + \lambda_{M-1}\,,$$

  the same as $J_{3,x}$ obtained earlier.

- If $l < M-1$ then we take the l eigenvectors associated with the l largest eigenvalues of $S_{xw}^{-1}S_{xb}$.

[2]Note that B is *not* necessarily orthogonal, all we know is that it is invertible.

Problem Solutions

Problem 5.1 ($\frac{(2N-2)s_z^2}{\sigma^2}$ is a chi-squared random variable)

To solve this problem we will use several results from Appendix A.11 (chi-squared distribution). To begin recall that when given N draws, $\{x_i\}_{i=1}^N$, from a Gaussian random variable with variance σ^2 and sample mean $\bar{x}$ the expression

$$\frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i-\bar{x})^2\,,$$

is given by a chi-squared distribution with $N-1$ degrees of freedom. Next recall that if χ_1^2 and χ_2^2 are independent random variables from chi-squared distributions with N_1 and N_2 degrees of freedom then

$$\chi^2=\chi_1^2+\chi_2^2\,,$$

is a random variable from a chi-squared distribution with N_1+N_2 degrees of freedom. Thus when we consider the expression $\frac{(2N-2)s_z^2}{\sigma^2}$ or

$$\sum_{i=1}^{N}\frac{(x_i-\bar{x})^2}{\sigma^2}+\sum_{i=1}^{N}\frac{(y_i-\bar{y})^2}{\sigma^2}\,,$$

we have the sum of two independent chi-squared random variables each of degree $N-1$. Thus this expression is another chi-squared random variable with $2N-2$ degrees of freedom.

Problem 5.2 (q has a t-distribution)

Using the same arguments as in problem 5.1 above we first note that

$$\frac{(N_1+N_2-2)s_z^2}{\sigma^2}\,,$$

is given by a χ^2 random variable with N_1+N_2-2 degrees of freedom. Next, if we consider the random variable $\bar{x}-\bar{y}-\mu_1+\mu_2$ we know that it is Gaussian with a zero mean and a variance given by

$$\begin{aligned}\mathrm{Var}[\bar{x}-\bar{y}-\mu_1+\mu_2] &= \mathrm{Var}[((\bar{x}-\mu_1)-(\bar{y}-\mu_2))]\\ &= \mathrm{Var}[\bar{x}-\mu_1]-2\mathrm{Cov}[(\bar{x}-\mu_1),(\bar{y}-\mu_2)]+\mathrm{Var}[\bar{y}-\mu_2]\\ &= \mathrm{Var}[\bar{x}-\mu_1]+\mathrm{Var}[\bar{y}-\mu_2]=\frac{\sigma^2}{N_1}+\frac{\sigma^2}{N_2}\,,\end{aligned}$$

since we are assuming that $\mathrm{Cov}[(\bar{x}-\mu_1),(\bar{y}-\mu_2)]=0$. Thus the random variable

$$\frac{\bar{x}-\bar{y}-\mu_1+\mu_2}{\sigma\sqrt{\frac{1}{N_1}+\frac{1}{N_2}}}\,,$$

is a standard normal. Then using the results from Appendix A.12 (t-distribution) where it is stated that the ratio of a standard normal random variable over a scaled χ^2 random variable with n degrees of freedom is a t-distributed random variable of degree n we have, forming a ratio of the required form, that

$$\frac{\frac{\bar{x}-\bar{y}-\mu_1+\mu_2}{\sigma\sqrt{\frac{1}{N_1}+\frac{1}{N_2}}}}{\sqrt{\frac{(N_1+N_2-2)s_z^2}{\sigma^2}\left(\frac{1}{N_1+N_2-2}\right)}},$$

is a t distributed random variable with N_1+N_2-2 degrees of freedom. The above expression simplifies to

$$\frac{\bar{x}-\bar{y}-\mu_1+\mu_2}{s_z\sqrt{\frac{1}{N_1}+\frac{1}{N_2}}}.$$

Which shows that the desired expression is a t distributed random variable with N_1+N_2-2 degrees of freedom.

Problem 5.3 (A is orthonormal)

The given matrix A has components $A(i,j)$ that can be represented as

$$\begin{aligned} A(1,j) &= \frac{1}{\sqrt{n}} \quad 1 \le j \le n \\ A(i,i) &= \frac{i-1}{\sqrt{i(i-1)}} \quad i \ge 2 \\ A(i,j) &= -\frac{1}{\sqrt{i(i-1)}} \quad i \ge 2 \quad \text{and} \quad 1 \le j \le i-1\,. \end{aligned}$$

Then the (p,q) element of the product AA^T is given by

$$(AA^T)(p,q) = \sum_{k=1}^{n} A(p,k)A^T(k,q) = \sum_{k=1}^{n} A(p,k)A(q,k)\,.$$

We will evaluate this expression for all possible values of (p,q) and show that in all cases this matrix product equals the identity matrix. Since the first row above seems different than the general case we start there. If $p=1$ then we have

$$(AA^T)(1,q) = \frac{1}{\sqrt{n}}\sum_{k=1}^{n} A(q,k)\,.$$

If we then take $q=1$ we get

$$(AA^T)(1,1) = \frac{1}{\sqrt{n}}\sum_{k=1}^{n} A(1,k) = \frac{1}{n}\sum_{k=1}^{n} 1 = 1\,.$$

If $q > 1$ then we get

$$\begin{aligned}(AA^T)(1,q) &= \frac{1}{\sqrt{n}}\sum_{k=1}^{n} A(q,k) = \frac{1}{\sqrt{n}}\left[A(q,q) + \sum_{k=1}^{q-1} A(q,k)\right] \\ &= \frac{1}{\sqrt{n}}\left[\frac{q-1}{\sqrt{q(q-1)}} - \frac{1}{\sqrt{q(q-1)}}\sum_{k=1}^{q-1} 1\right] = 0\,.\end{aligned}$$

Now assume that $p > 1$. Then we have for $(AA^T)(p,q)$ the following

$$\begin{aligned}(AA^T)(p,q) &= \sum_{k=1}^{n} A(p,k)A(q,k) = A(p,p)A(q,p) + \sum_{k=1}^{p-1} A(p,k)A(q,k) \\ &= \frac{p-1}{\sqrt{p(p-1)}}A(q,p) - \frac{1}{\sqrt{p(p-1)}}\sum_{k=1}^{p-1} A(q,k)\,. \qquad (121)\end{aligned}$$

To evaluate this lets first assume that $q < p$ then $A(q,k) = 0$ if $k > q$ and then Equation 121 gives

$$\begin{aligned}(AA^T)(p,q) &= 0 - \frac{1}{\sqrt{p(p-1)}}\sum_{k=1}^{q} A(q,k) \\ &= -\frac{1}{\sqrt{p(p-1)}}\left[A(q,q) + \sum_{k=1}^{q} A(q,k)\right] \\ &= -\frac{1}{\sqrt{p(p-1)}}\left[\frac{q-1}{\sqrt{q(q-1)}} - \sum_{k=1}^{q-1}\frac{1}{\sqrt{q(q-1)}}\right] = 0\,.\end{aligned}$$

If $q > p$ then Equation 121 gives

$$\frac{p-1}{\sqrt{p(p-1)}}\left(\frac{1}{\sqrt{q(q-1)}}\right) - \frac{1}{\sqrt{p(p-1)}}\sum_{k=1}^{p-1}\frac{1}{\sqrt{q(q-1)}} = 0\,.$$

Finally, if $p = q$ then Equation 121 gives

$$\begin{aligned}(AA^T)(p,p) &= \frac{p-1}{\sqrt{p(p-1)}}\left(\frac{1}{\sqrt{q(q-1)}}\right) - \frac{1}{\sqrt{p(p-1)}}\sum_{k=1}^{p-1}\frac{1}{\sqrt{q(q-1)}} \\ &= \frac{1}{\sqrt{p(p-1)}\sqrt{q(q-1)}}\left[(p-1)(q-1) + (p-1)\right] \\ &= \frac{(p-1)q}{\sqrt{p(p-1)}\sqrt{q(q-1)}} = 1\,,\end{aligned}$$

when we convert all q's into p's. Thus we have shown that $AA^T = I$ and A is an orthogonal matrix.

Problem 5.4 (linear combinations of Gaussian random variables)

Recall [2] that the characteristic function for multidimensional Gaussian random vector x with mean μ and covariance Σ is given by

$$\zeta_X(t) = E[e^{it^T X}] = \exp\left(it^T\mu - \frac{1}{2}t^T\Sigma t \right) . \tag{122}$$

If our random vector y is a linear combination of the elements of the vector x then $y = Ax$ and the characteristic function for y is given by

$$\begin{aligned} \zeta_Y(t) &= E[e^{it^T Y}] = E[e^{it^T AX}] = E[e^{i(A^T t)^T x}] = \zeta_X(A^T t) \\ &= \exp\left(it^T A\mu - \frac{1}{2}t^T A\Sigma A^T t \right) , \end{aligned}$$

which is the same as the characteristic function of a multidimensional Gaussian random vector that has a mean vector of $A\mu$ and covariance matrix of $A\Sigma A^T$ as we were to show. If x_i are mutually independent with *identical* variances say σ^2 then Σ is a multiple of the identity matrix, say $\Sigma = \sigma^2 I$. In that case $A\Sigma A^T = \sigma^2 AA^T$. In that case if A is orthogonal the covariance matrix for y_i is $\sigma^2 I$ and these transformed variables are also mutually independent.

Problem 5.5 (the ambiguity function)

We define the *ambiguity function* as

$$A = -\sum_{i=1}^{M}\sum_{j=1}^{K} P(\Delta_j)P(\omega_i|\Delta_j)\log_M(P(\omega_i|\Delta_j)) . \tag{123}$$

If the distribution of features over each class is completely overlapping, then $P(\Delta_j|\omega_i)$ is independent of ω_i. That is $P(\Delta_j|\omega_i) = P(\Delta_j)$. In this case, then using Bayes' rule we have

$$P(\omega_i|\Delta_j) = \frac{P(\Delta_j|\omega_i)P(\omega_i)}{P(\Delta_j)} = P(\omega_i) .$$

The ambiguity function in this case then becomes

$$A = -\sum_{i=1}^{M}\sum_{j=1}^{K} P(\Delta_j)P(\omega_i)\log_M(P(\omega_i)) = -\sum_{i=1}^{M} P(\omega_i)\log_M(P(\omega_i)) .$$

If we further assume that each class is equally likely then $P(\omega_i) = \frac{1}{M}$, so $\log_M\left(\frac{1}{M}\right) = -1$ and we find A becomes

$$A = \frac{1}{M}\sum_{i=1}^{M} 1 = 1 .$$

If the distribution of features are perfectly separated, then $P(\omega_i|\Delta_j) = 0$ if class i does not have any "overlap" with the region Δ_j, otherwise $P(\omega_i|\Delta_j) = 1$, since in that case only class

ω_i is present. To evaluate A we break the inner sum of j into regions where class i has feature overlap and does not have

$$A = -\sum_{i=1}^{M} \left\{ \sum_{j\text{: class } i \text{ overlaps } \Delta_j} + \sum_{j\text{: class } i \text{ does not overlap } \Delta_j} \right\}$$

In the first sum, since $P(\omega_i|\Delta_j) = 1$ each term is zero and the entire sum vanishes. In the second sum, when $P(\omega_i|\Delta_j) = 0$ by a limiting argument one can show that

$$P(\omega_i|\Delta_j) \log_M(P(\omega_i|\Delta_j)) = 0\,,$$

and thus the entire sum also vanishes. Thus we have shown that $A = 0$.

Problem 5.7 (the divergence increase for Gaussian densities)

To begin this problem, we are told that when we consider the generation of original feature vectors $\mathbf{x}$ of dimension m, that the two classes i and j have the same covariance matrix Σ. If we then add an additional feature x_{m+1} so that we desire to consider the covariances of vectors defined as $\begin{bmatrix} \mathbf{x} \\ x_{m+1} \end{bmatrix}$, we will assume that these larger vectors also have equal covariance matrices when considered from class i and j. In this case that covariance matrix will be take of the form

$$\hat{\Sigma} = \begin{bmatrix} \Sigma & r \\ r^T & \sigma^2 \end{bmatrix}.$$

When two classes have equal covariances the trace terms in the divergence d_{ij} given by Equation 113 vanish and d_{ij} simplifies to

$$d_{ij} = (\hat{\boldsymbol{\mu}}_i - \hat{\boldsymbol{\mu}}_j)\hat{\Sigma}^{-1}(\hat{\boldsymbol{\mu}}_i - \hat{\boldsymbol{\mu}}_j)\,. \tag{124}$$

Here $\hat{\boldsymbol{\mu}}_i$ and $\hat{\boldsymbol{\mu}}_j$ are the mean vectors for the larger vector with the scalar x_{m+1} appended to the original $\mathbf{x}$, for example

$$\hat{\boldsymbol{\mu}}_i = \begin{bmatrix} \boldsymbol{\mu}_i \\ \mu_i \end{bmatrix},$$

where μ_i is the mean of x_{m+1} under class i. The same notation will be used for class j. Thus to find a recursive relationship for d_{ij} we need a way of decomposing the inner product defined above.

Since we are to assume that the covariance matrix, $\hat{\Sigma}$, for the vector $\begin{bmatrix} \mathbf{x} \\ x_{m+1} \end{bmatrix}$ is given in block form as

$$\hat{\Sigma} \equiv \begin{bmatrix} \Sigma & r \\ r^T & \sigma^2 \end{bmatrix}.$$

Where Σ is a $m \times m$ matrix and r is a $m \times 1$ column vector. Thus to further simplify the divergence we need to derive an expression for $\hat{\Sigma}^{-1}$. To compute this inverse we will multiply $\hat{\Sigma}$ on the left by a block matrix with some variable entries which we hope we can find suitable

values for and thus derive the block inverse. As an example of this lets multiply $\hat{\Sigma}$ on the left by the block matrix $\begin{bmatrix} \Sigma^{-1} & 0 \\ b' & d \end{bmatrix}$, where b is a $m \times 1$ dimensional vector and d is a scalar. Currently, the values of these two variables are unknown. When we multiply by this matrix we desire to find b and d such that

$$\begin{bmatrix} \Sigma^{-1} & 0 \\ b' & d \end{bmatrix} \begin{bmatrix} \Sigma & r \\ r^T & \sigma^2 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix} . \tag{125}$$

Equating the block multiplication result on the left to the components of the block matrix on the right gives

$$b'\Sigma + dr^T = 0 .$$

for the $(2,1)$ component. This later equation can be solved for b by taking transposes and inverting Σ as

$$b = -\Sigma^{-1} r d .$$

If we take $d = 1$ and b given by the solution above, the product on the left-hand-side given by Equation 125 does not becomes the identity but is given by

$$\begin{bmatrix} \Sigma^{-1} & 0 \\ -r^T\Sigma^{-1} & 1 \end{bmatrix} \begin{bmatrix} \Sigma & r \\ r^T & \sigma^2 \end{bmatrix} = \begin{bmatrix} I & \Sigma^{-1}r \\ 0 & \sigma^2 - r^T\Sigma^{-1}r \end{bmatrix} . \tag{126}$$

Note what we have just done is the "forward solve" step in Gaussian elimination. Taking the inverse of both sides of this later equation we find

$$\begin{bmatrix} \Sigma & r \\ r^T & \sigma^2 \end{bmatrix}^{-1} \begin{bmatrix} \Sigma^{-1} & 0 \\ -r^T\Sigma^{-1} & 1 \end{bmatrix}^{-1} = \begin{bmatrix} I & \Sigma^{-1}r \\ 0 & \sigma^2 - r^T\Sigma^{-1}r \end{bmatrix}^{-1} .$$

or

$$\begin{bmatrix} \Sigma & r \\ r^T & \sigma^2 \end{bmatrix}^{-1} = \begin{bmatrix} I & \Sigma^{-1}r \\ 0 & \sigma^2 - r^T\Sigma^{-1}r \end{bmatrix}^{-1} \begin{bmatrix} \Sigma^{-1} & 0 \\ -r^T\Sigma^{-1} & 1 \end{bmatrix} .$$

Thus it remains to find the inverse of the block matrix $\begin{bmatrix} I & \Sigma^{-1}r \\ 0 & \sigma^2 - r^T\Sigma^{-1}r \end{bmatrix}$. This inverse is the well known "backwards solve" in Gaussian elimination. Note that this inverse is given by

$$\begin{bmatrix} I & \Sigma^{-1}r \\ 0 & \frac{1}{\alpha} \end{bmatrix}^{-1} = \begin{bmatrix} I & -\alpha\Sigma^{-1}r \\ 0 & \alpha \end{bmatrix} ,$$

where we have made the definition of the scalar α such that $\frac{1}{\alpha} \equiv \sigma^2 - r^T\Sigma^{-1}r$. Using this result we have that

$$\begin{aligned} \begin{bmatrix} \Sigma & r \\ r^T & \sigma^2 \end{bmatrix}^{-1} &= \begin{bmatrix} I & -\alpha\Sigma^{-1}r \\ 0 & \alpha \end{bmatrix} \begin{bmatrix} \Sigma^{-1} & 0 \\ -r^T\Sigma^{-1} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \Sigma^{-1} + \alpha\Sigma^{-1}rr^T\Sigma^{-1} & -\alpha\Sigma^{-1}r \\ -\alpha r^T\Sigma^{-1} & \alpha \end{bmatrix} . \end{aligned} \tag{127}$$

Using this expression one of the required product in the evaluation of Equation 124 is given

by

$$\begin{bmatrix} \Sigma & r \\ r^T & \sigma^2 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \\ \mu_i - \mu_j \end{bmatrix} = \begin{bmatrix} \Sigma^{-1} + \alpha\Sigma^{-1}rr^T\Sigma^{-1} & -\alpha\Sigma^{-1}r \\ -\alpha r^T\Sigma^{-1} & \alpha \end{bmatrix} \begin{bmatrix} \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \\ \mu_i - \mu_j \end{bmatrix}$$
$$= \begin{bmatrix} (\Sigma^{-1} + \alpha\Sigma^{-1}rr^T\Sigma^{-1})(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) - \alpha\Sigma^{-1}r(\mu_i - \mu_j) \\ -\alpha r^T\Sigma^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) + \alpha(\mu_i - \mu_j) \end{bmatrix}$$
$$= \begin{bmatrix} d + \alpha\Sigma^{-1}rr^Td - \alpha\Sigma^{-1}r(\mu_i - \mu_j) \\ -\alpha r^Td + \alpha(\mu_i - \mu_j) \end{bmatrix}.$$

Where since the product $\Sigma^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$ appears a great number of times we defined it to be d, so $d \equiv \Sigma^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$. Computing the product needed to produce the full quadratic term in d_{ij} we get

$$\begin{aligned} (\boldsymbol{\mu}_i^T - \boldsymbol{\mu}_j^T, \mu_i - \mu_j) \begin{bmatrix} d + \alpha\Sigma^{-1}rr^Td - \alpha\Sigma^{-1}r(\mu_i - \mu_j) \\ -\alpha r^Td + \alpha(\mu_i - \mu_j) \end{bmatrix} &= (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^Td \\ &+ \alpha(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T\Sigma^{-1}rr^Td \\ &- \alpha(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T\Sigma^{-1}r(\mu_i - \mu_j) \\ &- \alpha(\mu_i - \mu_j)r^Td \\ &+ \alpha(\mu_i - \mu_j)^2. \end{aligned}$$

Taking the transpose of either term we see that the third and fourth scalar products in the above expressions are equal. Combining these we get

$$(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^Td + \alpha d^Trr^Td + \alpha(\mu_i - \mu_j)^2 - 2\alpha d^T r(\mu_i - \mu_j).$$

Completing the square of the expression with respect to $\mu_i - \mu_j$ we have this expression given by

$$\alpha\left[(\mu_i - \mu_j) - d^Tr\right]^2 - \alpha(d^Tr)^2 + \alpha d^Trr^Td + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^Td = \alpha\left[(\mu_i - \mu_j) - d^Tr\right]^2 + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^Td.$$

Thus using this and the definition of d and α we see that d_{ij} is given by

$$\begin{aligned} d_{ij}(x_1, x_2, \cdots, x_m, x_{m+1}) &= (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T\Sigma^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) + \frac{\left[\mu_i - \mu_j - d^Tr\right]^2}{\sigma^2 - r^T\Sigma^{-1}r} \\ &= d_{ij}(x_1, x_2, \cdots, x_m) + \frac{\left[\mu_i - \mu_j - (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T\Sigma^{-1}r\right]^2}{\sigma^2 - r^T\Sigma^{-1}r}. \end{aligned}$$

If the new feature is uncorrelated with the original ones then we have the vector r equal zero and the second expression follows from this one.

Problem 5.8 (the divergence sums for statistically independent features)

Consider the divergence d_{ij} defined by

$$d_{ij} = \int_{\mathbf{x}} (p(\mathbf{x}|\omega_i) - p(\mathbf{x}|\omega_j)) \ln\left(\frac{p(\mathbf{x}|\omega_i)}{p(\mathbf{x}|\omega_j)}\right) d\mathbf{x}. \tag{128}$$

Then if the features are statistically independent in each class we have

$$p(\mathbf{x}|\omega_i) = \prod_{k=1}^{l} p(x_k|\omega_i)\,.$$

Thus the logarithmic term above becomes

$$\begin{aligned}\ln\left(\frac{p(\mathbf{x}|\omega_i)}{p(\mathbf{x}|\omega_j)}\right)d\mathbf{x} &= \ln\left(\prod_{k=1}^{l}\frac{p(x_k|\omega_i)}{p(x_k|\omega_j)}\right) \\ &= \sum_{k=1}^{l}\ln\left(\frac{p(x_k|\omega_i)}{p(x_k|\omega_j)}\right)\,.\end{aligned}$$

Then we get for d_{ij} is

$$d_{ij} = \sum_{k=1}^{l}\int_{\mathbf{x}}(p(\mathbf{x}|\omega_i) - p(\mathbf{x}|\omega_j))\ln\left(\frac{p(x_k|\omega_i)}{p(x_k|\omega_j)}\right)d\mathbf{x}\,.$$

Since the logarithmic term only depends on x_k (and not the other k's) we can integrate out them by performing the $\int_{\mathbf{x}}$ integration for all variables but x_k. This then gives

$$d_{ij} = \sum_{k=1}^{l}\int_{x_k}(p(x_k|\omega_i) - p(x_k|\omega_j))\ln\left(\frac{p(x_k|\omega_i)}{p(x_k|\omega_j)}\right)dx_k\,,$$

which is the sum of l scalar divergences each one over a different variable.

Problem 5.9 (deriving the Chernoff bound)

The books equation 5.17 is given by

$$P_e \le P(\omega_1)^s P(\omega_2)^{1-s}\int p(x|\omega_1)^s p(x|\omega_2)^{1-s}dx \quad \text{for} \quad 0 \le s \le 1\,. \tag{129}$$

When the densities $p(x|\omega_i)$ for $i = 1, 2$ are d-dimensional multidimensional Gaussians then

$$p(x|\omega_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}}\exp\left\{-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\right\}\,, \tag{130}$$

so the product in the integrand in Equation 129 is given by

$$\begin{aligned}p(x|\omega_1)^s p(x|\omega_2)^{1-s} &= \frac{1}{(2\pi)^{\frac{ds}{2}}(2\pi)^{\frac{d(1-s)}{2}}|\Sigma_1|^{\frac{s}{2}}|\Sigma_2|^{\frac{1-s}{2}}} \\ &\times \exp\left\{-\frac{s}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - \frac{(1-s)}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right\}\,.\end{aligned}$$

Expanding the terms in the exponential we find (ignoring for now the factor $-\frac{1}{2}$)

$$sx^T\Sigma_1^{-1}x - 2sx^T\Sigma_1^{-1}\mu_1 + s\mu_1^T\Sigma_1^{-1}\mu_1 + (1-s)x^T\Sigma_2^{-1}x - 2(1-s)x^T\Sigma_2^{-1}\mu_2 + (1-s)\mu_2\Sigma_2^{-1}\mu_2\,.$$

Grouping the quadratic, linear, and constant terms we find

$$x^T(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})x - 2x^T(s\Sigma_1^{-1}\mu_1 + (1-s)\Sigma_2^{-1}\mu_2) + s\mu_1^T\Sigma_1^{-1}\mu_1 + (1-s)\mu_2^T\Sigma_2^{-1}\mu_2\,.$$

Using this expression the product we are considering then becomes

$$\begin{aligned} p(x|\omega_1)^s p(x|\omega_2)^{1-s} &= \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_1|^{\frac{s}{2}}|\Sigma_2|^{\frac{1-s}{2}}} \exp\left\{-\frac{1}{2}\left(s\mu_1^T\Sigma_1^{-1}\mu_1 + (1-s)\mu_2^T\Sigma_2^{-1}\mu_2\right)\right\} \qquad (131) \\ &\times \exp\left\{-\frac{1}{2}\left(x^T(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})x - 2x^T(s\Sigma_1^{-1}\mu_1 + (1-s)\Sigma_2^{-1}\mu_2)\right)\right\}. \end{aligned}$$

Thus we want to integrate this expression over all possible x values. The trick to evaluating an integral like this is to convert it into an integral that we know how to integrate. Since this involves the integral of a Gaussian like kernel we might be able to evaluate this integral by converting exactly it into the integral of a Gaussian. Then since it is known that the integral over all space of a Gaussians is one we may have evaluated indirectly the integral we are interested in. To begin this process we first consider what the argument of the *exponential* (without the $-1/2$) of a Gaussian with mean θ and covariance A would look like

$$(x-\theta)^T A^{-1}(x-\theta) = x^T A^{-1}x - 2x^T A^{-1}\theta + \theta^T A^{-1}\theta\,. \qquad (132)$$

Using this expression to match the arguments of the quadratic and linear terms in the exponent in Equation 131 would indicate that

$$\begin{aligned} A^{-1} &= s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1} \quad \text{and} \\ A^{-1}\theta &= s\Sigma_1^{-1}\mu_1 + (1-s)\Sigma_2^{-1}\mu_2\,. \end{aligned}$$

Thus the Gaussian with a mean value θ and covariance A given by

$$\begin{aligned} A &= (s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1} \qquad (133) \\ \theta &= A(s\Sigma_1^{-1}\mu_1 + (1-s)\Sigma_2^{-1}\mu_2) \\ &= (s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1}(s\Sigma_1^{-1}\mu_1 + (1-s)\Sigma_2^{-1}\mu_2)\,, \qquad (134) \end{aligned}$$

would evaluate to having exactly the *same* exponential terms (modulo the expression $\theta^T A^{-1}\theta$). The point of this is that with the definitions of A and θ we can write

$$x^T(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})x - 2x^T(s\Sigma_1^{-1}\mu_1 + (1-s)\Sigma_2^{-1}\mu_2) = (x-\theta)^T A^{-1}(x-\theta) - \theta^T A^{-1}\theta\,,$$

so that the integral we are attempting to evaluate can be written as

$$\begin{aligned} \int p(x|\omega_1)^s p(x|\omega_2)^{1-s}dx &= \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_1|^{\frac{s}{2}}|\Sigma_2|^{\frac{1-s}{2}}} \\ &\times \exp\left\{-\frac{1}{2}\left(s\mu_1^T\Sigma_1^{-1}\mu_1 + (1-s)\mu_2^T\Sigma_2^{-1}\mu_2\right)\right\}\exp\left\{\frac{1}{2}\theta^T A^{-1}\theta\right\} \\ &\times \int \exp\left\{-\frac{1}{2}(x-\theta)^T A^{-1}(x-\theta)\right\}dx\,. \end{aligned}$$

In effect what we are doing is "completing the square" of the argument in the exponential. Since we know that multidimensional Gaussians integrate to one, this final integral becomes

$$\int \exp\left\{-\frac{1}{2}(x-\theta)^T A^{-1}(x-\theta)\right\}dx = (2\pi)^{d/2}|A|^{1/2}\,. \qquad (135)$$

In addition, the argument in the exponential in front of the (now evaluated) integral is given by

$$s\mu_1^T\Sigma_1^{-1}\mu_1 + (1-s)\mu_2^T\Sigma_2^{-1}\mu_2 - \theta^T A^{-1}\theta\,. \tag{136}$$

When we put in the definition of A and θ given by Equations 133 and 134 we have that $\theta^T A^{-1}\theta$ is equivalent to three (somewhat complicated) terms

$$\begin{aligned}
\theta^T A^{-1}\theta &= (s\mu_1^T\Sigma_1^{-1} + (1-s)\mu_2^T\Sigma_2^{-1})(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1}(s\Sigma_1^{-1}\mu_1 + (1-s)\Sigma_2^{-1}\mu_2) \\
&= s^2\mu_1^T\Sigma_1^{-1}(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1}\Sigma_1^{-1}\mu_1 \\
&+ 2s(1-s)\mu_1^T\Sigma_1^{-1}(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1}\Sigma_2^{-1}\mu_2 \\
&= (1-s)^2\mu_2^T\Sigma_2^{-1}(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1}\Sigma_2^{-1}\mu_2\,.
\end{aligned}$$

Given that we still have to add the terms $s\mu_1^T\Sigma_1^{-1}\mu_1 + (1-s)\mu_2^T\Sigma_2^{-1}\mu_2$ to the negative of this expression we now stop and look at what our end result should look like in hopes of helping motivate the transformations to take next. Since we might want to try and factor this into an expression like $(\mu_1-\mu_2)^T B(\mu_1-\mu_2)$ by expanding this we see that we should try to get the expression above into a three term form that looks like

$$\mu_1^T B\mu_1 - 2\mu_1^T B\mu_2 + \mu_2^T B\mu_2\,, \tag{137}$$

for some matrix B. Thus lets add $s\mu_1^T\Sigma_1^{-1}\mu_1 + (1-s)\mu_2^T\Sigma_2^{-1}\mu_2$ to the negative of $\theta^T A^{-1}\theta$ and write the result in the three term form suggested by Equation 137 above. We find that Equation 136 then becomes when factored in this way

$$\begin{aligned}
& s\mu_1^T\left[\Sigma_1^{-1} - s\Sigma_1^{-1}(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1}\Sigma_1^{-1}\right]\mu_1 && (138)\\
- \;& 2s(1-s)\mu_1^T\left[\Sigma_1^{-1}(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1}\Sigma_2^{-1}\right]\mu_2 && (139)\\
+ \;& (1-s)\mu_2^T\left[\Sigma_2^{-1} - (1-s)\Sigma_2^{-1}(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1}\Sigma_2^{-1}\right]\mu_2\,. && (140)
\end{aligned}$$

We now use the *inverse of inverse matrix sums lemma* (IIMSL) given by

$$(A^{-1} + B^{-1})^{-1} = A(A+B)^{-1}B = B(A+B)^{-1}A\,, \tag{141}$$

to write the matrix products in the middle term of the above expression as

$$\Sigma_1^{-1}(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1}\Sigma_2^{-1} = ((1-s)\Sigma_1 + s\Sigma_2)^{-1}\,. \tag{142}$$

Recognizing this matrix as one that looks familiar and that we would like to turn the others into lets now "hope" that the others can be transformed into a form that looks like that. To further see if this is possible, and to motivate the transformations done next, consider how the *desired* expression would look like expanded as in Equation 137. We have without the factor of $-\frac{1}{2}s(1-s)$ the following

$$\begin{aligned}
(\mu_1-\mu_2)^T((1-s)\Sigma_1 + s\Sigma_2)^{-1}(\mu_1-\mu_2) &= \mu_1^T((1-s)\Sigma_1 + s\Sigma_2)^{-1}\mu_1 && (143)\\
&- 2\mu_1^T((1-s)\Sigma_1 + s\Sigma_2)^{-1}\mu_2 && (144)\\
&+ \mu_2^T((1-s)\Sigma_1 + s\Sigma_2)^{-1}\mu_2\,. && (145)
\end{aligned}$$

Since as just shown the middle terms match as desired, looking at the terms Equation 138 and 143, to have the desired equality we want to show if we can prove

$$s\left[\Sigma_1^{-1} - s\Sigma_1^{-1}(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1}\Sigma_1^{-1}\right] = s(1-s)((1-s)\Sigma_1 + s\Sigma_2)^{-1}\,, \tag{146}$$

and

$$(1-s)\left[\Sigma_2^{-1} - (1-s)\Sigma_2^{-1}(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1}\Sigma_2^{-1}\right] = s(1-s)((1-s)\Sigma_1 + s\Sigma_2)^{-1}\,, \quad (147)$$

the similar expression for the terms Equation 140 and 145. To show that in fact this matrix difference is correct we will use another matrix identity lemma. This time we will use the *Woodbury* identity which can be written as

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}\,. \quad (148)$$

If we specialize this identity by taking C and V to both be identity matrices we obtain

$$\begin{aligned}(A+U)^{-1} &= A^{-1} - A^{-1}U(I + A^{-1}U)^{-1}A^{-1} \\ &= A^{-1} - A^{-1}(U^{-1} + A^{-1})^{-1}A^{-1}\,.\end{aligned}$$

Using this last expression with $A = s\Sigma_1^{-1}$ and $U = (1-s)\Sigma_2^{-1}$ we can derive that

$$\begin{aligned}(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1} &= \frac{1}{s}\Sigma_1 - \frac{1}{s}\Sigma_1\left(\frac{1}{1-s}\Sigma_2 + \frac{1}{s}\Sigma_1\right)^{-1}\frac{1}{s}\Sigma_1 \\ &= \frac{1}{s}\Sigma_1 - \frac{(1-s)}{s}\Sigma_1\left((1-s)\Sigma_1 + s\Sigma_2\right)^{-1}\Sigma_1\,.\end{aligned}$$

Multiplying this last expression by $s\Sigma_1^{-1}$ on the left and Σ_1^{-1} on the right to get

$$s\Sigma_1^{-1}(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1}\Sigma_1^{-1} = \Sigma_1^{-1} - (1-s)((1-s)\Sigma_1 + s\Sigma_2)^{-1}\,.$$

This last expression gives that

$$\Sigma_1^{-1} - s\Sigma_1^{-1}(s\Sigma_1^{-1} + (1-s)\Sigma_2^{-1})^{-1}\Sigma_1^{-1} = (1-s)((1-s)\Sigma_1 + s\Sigma_2)^{-1}\,,$$

which is equivalent to the desired Equation 146. Using exactly the same steps one can prove Equation 147. In summary then we have shown that

$$\begin{aligned}\int p(x|\omega_1)^s p(x|\omega_2)^{1-s}dx &= \frac{|A|^{1/2}}{|\Sigma_1|^{\frac{s}{2}}|\Sigma_2|^{\frac{1-s}{2}}} \\ &\times \exp\left\{-\frac{1}{2}s(1-s)(\mu_1-\mu_2)^T((1-s)\Sigma_1 + s\Sigma_2)^{-1}(\mu_1-\mu_2)\right\}.\end{aligned}$$

It remains to evaluate the coefficient $\frac{|A|^{1/2}}{|\Sigma_1|^{\frac{s}{2}}|\Sigma_2|^{\frac{1-s}{2}}}$. Taking the determinant of both sides of Equation 142 and solving for the expression A defined in Equation 133 we find

$$|A| = \frac{|\Sigma_1||\Sigma_2|}{|(1-s)\Sigma_1 + s\Sigma_2|}\,. \quad (149)$$

When we put this into what we have found for $\int p(x|\omega_1)^s p(x|\omega_2)^{1-s}dx$ we obtain

$$\begin{aligned}\int p(x|\omega_1)^s p(x|\omega_2)^{1-s}dx &= \frac{|\Sigma_1|^{\frac{1-s}{2}}|\Sigma_2|^{\frac{s}{2}}}{|(1-s)\Sigma_1 + s\Sigma_2|^{\frac{1}{2}}} \\ &\times \exp\left\{-\frac{1}{2}s(1-s)(\mu_1-\mu_2)^T((1-s)\Sigma_1 + s\Sigma_2)^{-1}(\mu_1-\mu_2)\right\}.\end{aligned}$$

If we define the above expression equal to $e^{-b(s)}$ we see that $b(s)$ is given by

$$\begin{aligned} b(s) &= \frac{1}{2}s(1-s)(\mu_1-\mu_2)^T((1-s)\Sigma_1+s\Sigma_2)^{-1}(\mu_1-\mu_2) \\ &+ \frac{1}{2}\ln\left\{\frac{|(1-s)\Sigma_1+s\Sigma_2|}{|\Sigma_1|^{1-s}|\Sigma_2|^s}\right\}. \end{aligned} \tag{150}$$

When this is combined with Equation 129 we have finally proved the Chernoff inequality. If we now consider the case when $\Sigma_1 = \Sigma_2 = \Sigma$ we have

$$b(s) = \frac{s(1-s)}{2}(\mu_1-\mu_2)^T\Sigma^{-1}(\mu_1-\mu_2)\,.$$

Then as $\frac{1}{2}(\mu_1-\mu_2)^T\Sigma^{-1}(\mu_1-\mu_2)$ is a scalar multiplier of the function $s(1-s)$, its value does not change the location of the extrema of $b(s)$. To find the extrema of $b(s)$ we take the first derivative, set the result equal to zero and solve for s. We find

$$b'(s) = 1-s-s = 0 \Rightarrow s = \frac{1}{2}\,.$$

the second derivative of the function $b(s)$ is given by $b''(s) = -2$. Since this is negative $s = \frac{1}{2}$ is a maximum of $b(s)$ or a minimum of $e^{-b(s)}$.

Problem 5.10 (the mixture scatter matrix is the sum of S_w and S_b)

Consider evaluating the expectation in the definition of S_m by conditioning on each class

$$S_m = E[(x-\mu_0)(x-\mu_0)^T] = \sum_{i=1}^{M} E[(x-\mu_0)(x-\mu_0)^T|x\in\omega_i]P_i\,.$$

where $P_i = P(x\in\omega_i)$. Then write $x-\mu_0 = x-\mu_i+\mu_i-\mu_0$ and expand the inner product above as

$$(x-\mu_0)(x-\mu_0)^T = (x-\mu_i)(x-\mu_i)^T + 2(x-\mu_i)(\mu_i-\mu_0)^T + (\mu_i-\mu_0)(\mu_i-\mu_0)^T\,.$$

Then taking the conditional expectation of the above expression with respect to ω_i since $E[x-\mu_i|x\in\omega_i] = 0$ the middle term in above vanishes. The last term does not depend on x and is therefore a constant with respect to the expectation and we get for S_m

$$S_m = \sum_{i=1}^{M} P_i E[(x-\mu_i)(x-\mu_i)^T|x\in\omega_i] + \sum_{i=1}^{M} P_i(\mu_i-\mu_0)(\mu_i-\mu_0)^T\,,$$

which when we recall the definitions of S_w and S_b given by

$$S_w = \sum_{i=1}^{M} P_i E[(x-\mu_i)(x-\mu_i)^T|x\in\omega_i] \tag{151}$$

$$S_b = \sum_{i=1}^{M} P_i(\mu_i-\mu_0)(\mu_i-\mu_0)^T\,, \tag{152}$$

we recognize as expressing S_m as $S_m = S_w + S_b$.

Problem 5.11 (bounds on the cross-correlation coefficient)

When we take the vectors $\mathbf{x}$ and $\mathbf{y}$ as

$$\mathbf{x} = \begin{bmatrix} x_{1i} \\ x_{2i} \\ \vdots \\ x_{Ni} \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} y_{1i} \\ y_{2i} \\ \vdots \\ y_{Ni} \end{bmatrix},$$

the Schwartz's inequality $|x^T y| \leq ||x||||y||$ show that $|\rho_{ij}| \leq 1$ where ρ_{ij} is defined by

$$\rho_{ij} = \frac{\sum_{n=1}^{N} x_{ni} y_{nj}}{\sqrt{\sum_{n=1}^{N} {x_{ni}}^2 \sum_{n=1}^{N} {y_{nj}}^2}}.$$

Problem 5.12 (the divergence of a two class problem)

The divergence between two Gaussians is given by Equation 113. If we assume that the Gaussians have the same covariance then $\Sigma_1 = \Sigma_2 = \Sigma$ and the divergence becomes

$$\begin{aligned} d_{12} &= (\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2) = \text{trace}((\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)) \\ &= \text{trace}(\Sigma^{-1} (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T). \end{aligned}$$

When the classes are equiprobable $P_1 = P_2 = \frac{1}{2}$. Then the within class scatter matrix Equation 114 becomes

$$S_w = \frac{1}{2}\Sigma_1 + \frac{1}{2}\Sigma_2 = \Sigma.$$

Now lets compute S_b using Equation 115. We have

$$S_b = \sum_{i=1}^{M} P_i (\mu_i - \mu_0)(\mu_i - \mu_0)^T = \frac{1}{2}\left[(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T + (\mu_2 - \mu_0)(\mu_2 - \mu_0)^T\right].$$

Since $\mu_0 = \sum_{i=1}^{M} P_i \mu_i = \frac{1}{2}(\mu_1 + \mu_2)$ when we compute the needed differences to compute S_b we calculate

$$\begin{aligned} S_b &= \frac{1}{2}\left[\frac{1}{4}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T + \frac{1}{4}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T\right] \\ &= \frac{1}{4}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T. \end{aligned}$$

Thus if we consider the expression $\text{trace}(S_w^{-1} S_b)$ we see that it equals in this case

$$\text{trace}(S_w^{-1} S_b) = \frac{1}{4}\text{trace}(\Sigma^{-1}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T).$$

We see that this is *proportional* to the expression d_{12} derived above.

Problem 5.13 (the number of combinations in backwards elimination)

See the notes on Page 83 where we derive this expression.

Problem 5.14 (the derivative of a trace)

We want to evaluate

$$\frac{\partial}{\partial A}\text{trace}\{(A^T S_1 A)^{-1}(A^T S_2 A)\}$$

The algebraic procedure for computing derivatives like $\frac{\partial}{\partial A}\text{trace}\{F(A)\}$ where $F(\cdot)$ is a matrix function of a matrix argument is discussed in [3]. The basic procedure is the following. We consider the matrix derivative as several scalar derivative (one derivative for each component a_{kl} of A). We pass the derivative of a_{kl} though the trace operation and take the scalar derivative of various matrix expressions i.e. $\frac{\partial F(A)}{\partial a_{kl}}$. Taking these derivatives is easier if we introduce the matrix $V(k,l)$ which is a matrix of all zeros except for a single one at the location (k,l). This is a helpful matrix to have since

$$\frac{\partial}{\partial a_{kl}}A = V(k,l)\,.$$

Once we have computed the derivative of the argument of the trace $F(A)$ with respect to a_{kl} we need to write it in the form

$$\frac{\partial F(A)}{\partial a_{kl}} = \sum_i g_i(A)V(k,l)h_i(A)\,.$$

We can then take the trace of the above expression and use the permutability of matrices in the argument of the trace to write

$$\begin{aligned}\text{trace}\left\{\frac{\partial F(A)}{\partial a_{kl}}\right\} &= \text{trace}\left\{\sum_i g_i(A)V(k,l)h_i(A)\right\} \\ &= \sum_i \text{trace}\left\{g_i(A)V(k,l)h_i(A)\right\} \\ &= \sum_i \text{trace}\left\{h_i(A)g_i(A)V(k,l)\right\}\,. \end{aligned} \tag{153}$$

Finally we use the property of the trace to conclude that for any $n \times n$ matrix M

$$MV(k,l) = \begin{bmatrix} & \begin{bmatrix} m_{1k} \\ m_{2k} \\ \vdots \\ m_{n-1,k} \\ m_{nk} \end{bmatrix} & \\ \mathbf{0} & & \mathbf{0} \end{bmatrix}\,,$$

or a matrix with the kth column of M in the lth column. Since the only nonzero column is the lth, to take the trace of this matrix, we need to find what the element of the lth row in

that column is. From the above we see that this element is m_{lk}. Thus we have just argued that

$$\text{trace}\left\{MV(k,l)\right\} = M(l,k)\,.$$

When we reassemble all elements, from this result, to compute the full matrix derivative of $\text{trace}\{MA\}$ we see that

$$\frac{\partial}{\partial A}\text{trace}\left\{MA\right\} = M^T\,.$$

Back to Equation 153 we can use the above to get the full matrix derivative

$$\frac{\partial}{\partial A}\text{trace}\{F(A)\} = \sum_i (h_i(A)g_i(A))^T\,. \tag{154}$$

For this problem we now implement this procedure.

To begin we evaluate the a_{kl} derivative of $(A^TS_1A)^{-1}(A^TS_2A)$. From the product rule we have

$$\frac{\partial}{\partial a_{kl}}\left[(A^TS_1A)^{-1}(A^TS_2A)\right] = \left[\frac{\partial}{\partial a_{kl}}(A^TS_1A)^{-1}\right](A^TS_2A) + (A^TS_1A)^{-1}\left[\frac{\partial}{\partial a_{kl}}(A^TS_2A)\right]\,.$$

To evaluate the a_{kl} derivative of $(A^TS_1A)^{-1}$ recall that if $F(A) = G^{-1}(A)$ then

$$\frac{\partial F(A)}{\partial a_{kl}} = -G^{-1}(A)\frac{\partial G(A)}{\partial a_{kl}}G^{-1}(A)\,. \tag{155}$$

Thus we get

$$\frac{\partial (A^TS_1A)^{-1}}{\partial a_{kl}} = -(A^TS_1A)^{-1}\frac{\partial (A^TS_1A)}{\partial a_{kl}}(A^TS_1A)^{-1}\,.$$

Thus we need to evaluate the derivative of A^TS_1A (a similar needed derivative is of A^TS_2A). We get

$$\frac{\partial (A^TS_1A)}{\partial a_{kl}} = V^T(k,l)S_1A + A^TS_1V(k,l)\,.$$

Combining these results we get

$$\begin{aligned}\frac{\partial}{\partial a_{kl}}(A^TS_1A)^{-1}(A^TS_2A) = &-(A^TS_1A)^{-1}\left[V^T(k,l)S_1A + A^TS_1V(k,l)\right](A^TS_1A)^{-1}(A^TS_2A)\\ &+ (A^TS_1A)^{-1}\left[V^T(k,l)S_2A + A^TS_2V(k,l)\right]\,.\end{aligned}$$

Then for each term (there are four of them) once we take the trace we can write each one as $g_i(A)V(k,l)h_i(A)$ for functions $g_i(\cdot)$ and $h_i(\cdot)$ for $i = 1,2,3,4$ by using

$$\text{trace}(A^T) = \text{trace}(A)\,,$$

if needed. We will need to use that identity for the first and third terms. We get

$$\begin{aligned}
g_1(A) &= -(A^TS_2A)(A^TS_1A)^{-1}A^TS_1\,, \quad \text{and} \quad h_1(A) = (A^TS_1A)^{-1}\\
g_2(A) &= -(A^TS_1A)^{-1}A^TS_1\,, \quad \text{and} \quad h_2(A) = (A^TS_1A)^{-1}(A^TS_2A)\\
g_3(A) &= A^TS_2\,, \quad \text{and} \quad h_3(A) = (A^TS_1A)^{-1}\\
g_4(A) &= (A^TS_1A)^{-1}A^TS_2\,, \quad \text{and} \quad h_4(A) = I\,.
\end{aligned}$$

Once we have done this we use Equation 154 (but without the transpose yet) to get

$$\begin{aligned}\left(\frac{\partial}{\partial A}\text{trace}\left\{(A^TS_1A)^{-1}(A^TS_2A)\right\}\right)^T &= -(A^TS_1A)^{-1}(A^TS_2A)(A^TS_1A)^{-1}A^TS_1 \\ &\quad -(A^TS_1A)^{-1}(A^TS_2A)(A^TS_1A)^{-1}A^TS_1 \\ &\quad +(A^TS_1A)^{-1}A^TS_2 \\ &\quad +(A^TS_1A)^{-1}A^TS_2\,.\end{aligned}$$

Thus taking the transpose of both sides we finally find

$$\frac{\partial}{\partial A}\text{trace}\left\{(A^TS_1A)^{-1}(A^TS_2A)\right\} = -2S_1A(A^TS_1A)^{-1}(A^TS_2A)(A^TS_1A)^{-1}+2S_2A(A^TS_1A)^{-1}\,,$$

as we were to show.

Problem 5.17 (the eigenstructure for $S_w^{-1}S_b$ in a two class problem)

In a two class problem $M = 2$, $P_1 + P_2 = 1$, and we have S_b given by

$$S_b = \sum_{i=1}^{M} P_i(\mu_i - \mu_0)(\mu_i - \mu_0)^T = P_1(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T + P_2(\mu_2 - \mu_0)(\mu_2 - \mu_0)^T\,.$$

Since $\mu_0 = \sum_{i=1}^{M} P_i\mu_i = P_1\mu_1 + P_2\mu_2$ we have that

$$\begin{aligned}\mu_1 - \mu_0 &= \mu_1 - P_1\mu_1 - P_2\mu_2 = (1-P_1)\mu_1 - P_2\mu_2 = P_2(\mu_1 - \mu_2) \\ \mu_2 - \mu_0 &= -P_1\mu_1 + (1-P_2)\mu_2 = P_1(\mu_2 - \mu_1)\,.\end{aligned}$$

Using these we see that S_b is given by

$$\begin{aligned}S_b &= P_1P_2^2(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T + P_2P_1^2(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \\ &= P_1P_2(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T\,.\end{aligned}$$

Thus the matrix $S_w^{-1}S_b$ is

$$P_1P_2S_w^{-1}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T\,.$$

Since the matrix $(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$ is rank one the matrix $S_w^{-1}S_b$ is rank one, and thus we have one non-zero eigenvalue (and its corresponding eigenvector). Consider the vector $v_1 \equiv S_w^{-1}(\mu_1 - \mu_2)$ and observe that

$$\begin{aligned}S_w^{-1}S_bv_1 &= P_1P_2S_w^{-1}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^TS_w^{-1}(\mu_1 - \mu_2) \\ &= (P_1P_2(\mu_1 - \mu_2)^TS_w^{-1}(\mu_1 - \mu_2))S_w^{-1}(\mu_1 - \mu_2) \\ &= \lambda_1v_1\,,\end{aligned}$$

where we take $\lambda_1 = P_1P_2(\mu_1 - \mu_2)^TS_w^{-1}(\mu_1 - \mu_2)$. Thus v_1 is an eigenvector of $S_w^{-1}S_b$ and λ_1 is its corresponding eigenvalue.

Problem 5.18 (orthogonality of the eigenvectors of $S_1^{-1}S_2$)

Since S_1 and S_2 can be simultaneously diagonalized, there exists and invertible matrix B such that

$$B^T S_1 B = I \quad \text{and} \quad B^T S_2 B = D\,,$$

where D is a diagonal matrix. Since B is invertible we can solve for S_1 and S_2 in terms of B and D as

$$S_1 = B^{-T}B^{-1} \quad \text{and} \quad S_2 = B^{-T}DB^{-1}\,.$$

Using these consider the product

$$S_1^{-1}S_2 = (BB^T)(B^{-T}DB^{-1}) = BDB^{-1}\,.$$

Let v_i be an eigenvector of $S_1^{-1}S_2$ with eigenvalue λ_i. Then by the definition of an eigenvector we have

$$S_1^{-1}S_2 v_i = \lambda_i v_i\,,$$

or from the expression for $S_1^{-1}S_2$ in terms of B and D

$$BDB^{-1}v_i = \lambda_i v_i\,.$$

This gives two expressions for v_i

$$\begin{aligned} v_i &= \frac{1}{\lambda_i} BDB^{-1}v_i \\ v_i &= \lambda_i BD^{-1}B^{-1}v_i\,. \end{aligned}$$

Now consider $v_i^T S_1 v_j$, using the first of these we will replace v_i with $\frac{1}{\lambda_i}BDB^{-1}v_i$, S_1 with $B^{-T}B^{-1}$, and using the second expression above v_j with $\lambda_j BD^{-1}B^{-1}v_j$ to get

$$\begin{aligned} v_i^T S_1 v_j &= \frac{\lambda_j}{\lambda_i} v_i^T B^{-T}DB^T B^{-T}B^{-1}BD^{-1}B^{-1}v_j \\ &= \frac{\lambda_j}{\lambda_i} v_i^T B^{-T}B^{-1}v_j = \frac{\lambda_j}{\lambda_i} v_i^T S_1 v_j\,. \end{aligned}$$

Thus

$$\left(1 - \frac{\lambda_j}{\lambda_i}\right) v_i^T S_1 v_j = 0\,.$$

So if $i \neq j$ (where we assume that $\lambda_i \neq \lambda_j$) then the last equation shows that $v_i S_1 v_j = 0$ as we were to show.

Feature Generation I: Linear Transforms

Notes on the text

Notes on basis vectors and images

We define a *separable* transformation of X to be one where the transform Y is given by

$$Y = U^H X V \,. \tag{156}$$

A separable transformation can be thought of as two successive transformations, one over the columns of X and another over the rows of the matrix product $U^H X$. To see this first define the product $U^H X$ as Z and write U^H as $\begin{bmatrix} u_0^H \\ u_1^H \\ \vdots \\ u_{N-1}^H \end{bmatrix}$ so that considered as a block matrix product, the expression $Z = U^H X$ is the product of a $N \times 1$ block matrix times a 1×1 block matrix or

$$\begin{bmatrix} u_0^H \\ u_1^H \\ \vdots \\ u_{N-1}^H \end{bmatrix} X = \begin{bmatrix} u_0^H X \\ u_1^H X \\ \vdots \\ u_{N-1}^H X \end{bmatrix} .$$

This result has N rows where each one is of the form $u_i^H X$ or the inner product transform of the N columns of X. Thus the transformation $U^H X$ is a transformation over the rows of X. Now consider the product $(U^H X)V$, which we can write as $ZV = (V^H Z^H)^H$. Notice that $V^H Z^H$ is the same "type" of transformation as we just discussed i.e. inner product transforms of the columns of Z^H. Equivalently inner product transforms of the rows of $Z = U^H X$, proving the statement made above. Note that some of the calculations for Example 6.1 are performed in the MATLAB script `chap_6_example_6_1.m`.

Notes on independent component analysis (ICA)

From the derivation in the book we have that at a stationary point

$$\frac{\partial J(W)}{\partial W} W^T = E[I - \phi(y) y^T] = 0 \,. \tag{157}$$

If we postmultiply by W and use $W^T W = I$ we get

$$\frac{\partial J(W)}{\partial W} = E[I - \phi(y) y^T] W = 0 \,.$$

The expression $\frac{\partial J(W)}{\partial W} = E[I - \phi(y) y^T] W$ is called the *natural gradient*.

Notes on the discrete Fourier transform

We define the scalar W_N as an Nth root of unity or

$$W_N \equiv \exp\left(-j\frac{2\pi}{N}\right) , \tag{158}$$

with $j \equiv \sqrt{-1}$ and the matrix W^H is given by

$$\frac{1}{\sqrt{N}}\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & W_N & W_N^2 & W_N^3 & \cdots & W_N^{N-2} & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & W_N^6 & \cdots & W_N^{2(N-2)} & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 1 & W_N^{N-2} & W_N^{2(N-2)} & W_N^{3(N-2)} & \cdots & W_N^{(N-2)(N-2)} & W_N^{(N-1)(N-2)} \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & W_N^{3(N-1)} & \cdots & W_N^{(N-2)(N-1)} & W_N^{(N-1)(N-1)} \end{bmatrix} . \tag{159}$$

From this we see that the (i,j)th component of W^H is given by

$$(W^H)(i,j) = \frac{1}{\sqrt{N}} W_N^{ij} . \tag{160}$$

Using W^H above and the fact that $W_N^* = W_N^{-1}$ the matrix W is given by

$$\frac{1}{\sqrt{N}}\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & W_N^{-1} & W_N^{-2} & W_N^{-3} & \cdots & W_N^{-(N-2)} & W_N^{-(N-1)} \\ 1 & W_N^{-2} & W_N^{-4} & W_N^{-6} & \cdots & W_N^{-2(N-2)} & W_N^{-2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 1 & W_N^{-(N-2)} & W_N^{-2(N-2)} & W_N^{-3(N-2)} & \cdots & W_N^{-(N-2)(N-2)} & W_N^{-(N-1)(N-2)} \\ 1 & W_N^{-(N-1)} & W_N^{-2(N-1)} & W_N^{-3(N-1)} & \cdots & W_N^{-(N-2)(N-1)} & W_N^{-(N-1)(N-1)} \end{bmatrix} . \tag{161}$$

From this we see that the (i,j)th component of W is given by

$$W(i,j) = \frac{1}{\sqrt{N}} W_N^{-ij} . \tag{162}$$

These expressions will be used in the problems and derivations below.

Notes on the two-dimensional Fourier transform

The two-dimensional discrete Fourier transform is defined as

$$Y(k,l) = \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} X(m,n) W_N^{km} W_N^{ln} . \tag{163}$$

Recalling via Equation 160 that the (k,m) element of W^H is $\frac{1}{\sqrt{N}} W_N^{km}$ in terms of the elements of W^H this sum is

$$Y(k,l) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} X(m,n)(W^H)(k,m)(W^H)(l,n) .$$

Using Equation 177 to convert this double sum into a matrix product we have we have $Y = W^H X (W^H)^T$ but since W^H is symmetric we have

$$Y = W^H X W^H \,. \tag{164}$$

Notes on the Haar transform

For the Haar transform, given the index n, we have 2^n basis functions index by k where $k = 0, 1, 2, \cdots 2^n - 1$ and denoted by $h_k(z)$. Given a value of the index k in the range just specified we can convert this index k uniquely into two other nonnegative integers p and q. The integer p (for power) is the largest natural number such that $2^p \leq k$ and then $q - 1$ is the "remainder". Thus let $q - 1$ be given by

$$q - 1 = k - 2^p \,.$$

Thus with these two definitions of p and q we have written the index k as

$$k = 2^p + q - 1 \,. \tag{165}$$

This definition works for $p \neq 0$, where if $p = 0$ then $q = 0$ or 1. For example, if we take $n = 3$ then there are 8 basis functions $k = 0, 1, 2, \cdots, 7$ and we have the mapping described above from k into (q, p) given by

$$\begin{aligned}
k &= 0 \Rightarrow p = 0 \quad \text{and} \quad q = 0 \\
k &= 1 \Rightarrow p = 0 \quad \text{and} \quad q = 1 \\
k &= 2 \Rightarrow p = 1 \quad \text{and} \quad q = 2 - 2^1 + 1 = 1 \\
k &= 3 \Rightarrow p = 1 \quad \text{and} \quad q = 3 - 2^1 + 1 = 2 \\
k &= 4 \Rightarrow p = 2 \quad \text{and} \quad q = 4 - 2^2 + 1 = 1 \\
k &= 5 \Rightarrow p = 2 \quad \text{and} \quad q = 5 - 2^2 + 1 = 2 \\
k &= 6 \Rightarrow p = 2 \quad \text{and} \quad q = 6 - 2^2 + 1 = 3 \\
k &= 7 \Rightarrow p = 2 \quad \text{and} \quad q = 7 - 2^2 + 1 = 4 \,.
\end{aligned}$$

The reason for introducing the indexes (p, q) is that it is easier to write the expression for the basis functions $h_k(z)$ in terms of the numbers p and q. Given the above equivalence we can convert sums over k (the number of basis functions) into a double sum over p and q as

$$\sum_{k=0}^{2^n-1} h_k(z) \equiv h_{p=0,q=0}(z) + h_{p=0,q=1}(z) + \sum_{p=1}^{n-1} \sum_{q=1}^{2^p} h_{pq}(z) \,, \tag{166}$$

since the range of p is $0 \leq p \leq n - 1$ and q is between $1 \leq q \leq 2^p$. Note that due to the slightly different conditions that happen when $k = 0$ and $k = 1$ in Equation 165, we have represented these terms on their own and outside of the general summation notation.

Notes on the two-band discrete time wavelet transform (DTWT)

Most of the notes in this section are verifications of the expressions given in the book. While there are no real comments with this section these notes provide more details in that they explicitly express some of the intermediate expressions which make verification of the proposed expressions easier. We begin with the two-band filter equations, when our two filters have impulse response functions given by $h_0(k)$ and $h_1(k)$. In that case we have

$$y_0(k) = \sum_l x(l)\, h_0(n-l)|_{n=2k} \tag{167}$$

$$y_1(k) = \sum_l x(l)\, h_1(n-l)|_{n=2k}\ . \tag{168}$$

When $k=0$ for $y_0(k)$ then $n=0$ and the first equation above gives

$$\begin{aligned} y_0(0) &= \sum_l x(l)h_0(0-l) \\ &= \cdots + x(-3)h_0(3) + x(-2)h_0(2) + x(-1)h_0(1) \\ &+ x(0)h_0(0) \\ &+ x(1)h_0(-1) + x(2)h_0(-2) + x(3)h_0(-3) + \cdots . \end{aligned}$$

When $k=1$ for $y_0(k)$ then $n=2$ and we get

$$\begin{aligned} y_0(1) &= \sum_l x(l)h_0(2-l) \\ &= \cdots + x(-3)h_0(5) + x(-2)h_0(4) + x(-1)h_0(3) \\ &+ x(0)h_0(2) \\ &+ x(1)h_0(1) + x(2)h_0(0) + x(3)h_0(2) + \cdots . \end{aligned}$$

The same type of expressions will hold for $y_1(k)$ but with h_0 replace with h_1. When we list these equations in a matrix form we get

$$\begin{bmatrix} \vdots \\ y_0(-2) \\ y_1(-2) \\ y_0(-1) \\ y_1(-1) \\ y_0(0) \\ y_1(0) \\ y_0(1) \\ y_1(1) \\ y_0(2) \\ y_1(2) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & h_0(-2) & h_0(-3) & h_0(-4) & h_0(-5) & h_0(-6) & \cdots \\ \cdots & h_1(-2) & h_1(-3) & h_1(-4) & h_1(-5) & h_1(-6) & \cdots \\ \cdots & h_0(0) & h_0(-1) & h_0(-2) & h_0(-3) & h_0(-4) & \cdots \\ \cdots & h_1(0) & h_1(-1) & h_1(-2) & h_1(-3) & h_1(-4) & \cdots \\ \cdots & h_0(2) & h_0(1) & h_0(0) & h_0(-1) & h_0(-2) & \cdots \\ \cdots & h_1(2) & h_1(1) & h_1(0) & h_1(-1) & h_1(-2) & \cdots \\ \cdots & h_0(4) & h_0(3) & h_0(2) & h_0(1) & h_0(0) & \cdots \\ \cdots & h_1(4) & h_1(3) & h_1(2) & h_1(1) & h_1(0) & \cdots \\ \cdots & h_0(6) & h_0(5) & h_0(4) & h_0(3) & h_0(2) & \cdots \\ \cdots & h_1(6) & h_1(5) & h_1(4) & h_1(3) & h_1(2) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \vdots \\ x(-2) \\ x(-1) \\ x(0) \\ x(+1) \\ x(+2) \\ \vdots \end{bmatrix} .$$

I explicitly included more terms than in the book so that the pattern of the elements is as clear as possible. In practice, while we can tolerate a non causal impulse filters ($h_0(k)$ and

$h_1(k)$ nonzero for negative k) but since we would like the matrix above to be of finite extent we require that h_0 and h_1 have only a finite number of nonzero terms. As a matrix equation we can write this as

$$y = T_i x \,,$$

where T_i is the mapping "into" the wavelet domain. Once we have constructed the two outputs $y_0(k)$ and $y_1(k)$ we seek another pair of filters of a special form that act as an inverse to the above mapping, in that they can synthesis the original signal, x, from the output pair y_0 and y_1. In sort of the same way we split x into y_0 and y_1 we will process y_0 and y_1 independently and then combined them to get x. The two functions that we combine are

$$x_0(n) = \sum_k y_0(k) g_0(n-2k) \tag{169}$$

$$x_1(n) = \sum_k y_1(k) g_1(n-2k) \,, \tag{170}$$

and the combination of x_0 and x_1 gives x

$$x(n) = x_0(n) + x_1(n) = \sum_k y_0(k) g_0(n-2k) + y_1(k) g_1(n-2k) \,.$$

To derive the matrix representation of this mapping we again write out the above equation for a couple of values of n to get a feel for the coefficients that result. For $n = 0$ we have

$$\begin{aligned} x(0) &= \cdots + y_0(-2)g_0(4) + y_1(-2)g_1(4) + y_0(-1)g_0(2) + y_1(-1)g_1(2) \\ &+ y_0(0)g_0(0) + y_1(0)g_1(0) \\ &+ y_0(1)g_0(-2) + y_1(1)g_1(-2) + y_0(2)g_0(-4) + y_1(2)g_1(-4) + \cdots \,. \end{aligned}$$

For $n = 1$ we have

$$\begin{aligned} x(1) &= \cdots + y_0(-2)g_0(5) + y_1(-2)g_1(5) + y_0(-1)g_0(3) + y_1(-1)g_1(3) \\ &+ y_0(0)g_0(1) + y_1(0)g_1(1) \\ &+ y_0(1)g_0(-1) + y_1(1)g_1(-1) + y_0(2)g_0(-3) + y_1(2)g_1(-3) + \cdots \,. \end{aligned}$$

Thus as a matrix we have the mapping from y to x in terms of values of g_0 and g_1 in great detail as

$$\begin{bmatrix} \vdots \\ x(-2) \\ x(-1) \\ x(0) \\ x(+1) \\ x(+2) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & g_0(2) & g_1(2) & g_0(0) & g_1(0) & g_0(-2) & g_1(-2) & g_0(-4) & g_1(-4) & g_0(-6) & g_1(-6) & \cdots \\ \cdots & g_0(3) & g_1(3) & g_0(1) & g_1(1) & g_0(-1) & g_1(-1) & g_0(-3) & g_1(-3) & g_0(-5) & g_1(-5) & \cdots \\ \cdots & g_0(4) & g_1(4) & g_0(2) & g_1(2) & g_0(0) & g_1(0) & g_0(-2) & g_1(-2) & g_0(-4) & g_1(-4) & \cdots \\ \cdots & g_0(5) & g_1(5) & g_0(3) & g_1(3) & g_0(1) & g_1(1) & g_0(-1) & g_1(-1) & g_0(-3) & g_1(-3) & \cdots \\ \cdots & g_0(6) & g_1(6) & g_0(4) & g_1(4) & g_0(2) & g_1(2) & g_0(0) & g_1(0) & g_0(-2) & g_1(-2) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \vdots \\ y_0(-2) \\ y_1(-2) \\ y_0(-1) \\ y_1(-1) \\ y_0(0) \\ y_1(0) \\ y_0(1) \\ y_1(1) \\ y_0(2) \\ y_1(2) \\ \vdots \end{bmatrix} .$$

As a matrix equation we can write this as

$$x = T_o y \,,$$

where T_o is the mapping "out" of the wavelet domain. For good reconstructive properties i.e. to be able to take the inverse transform of the direct transform and get the original signal back again we must have

$$T_i T_o = T_o T_i = I \,.$$

This creates the biorthogonality conditions that h_i and g_i must satisfy.

Problem Solutions

Problem 6.1 (some properties of the transformation $Y = U^H X V$)

Part (a): Given $X = UYV^H$, lets write the matrix U in "block" form as columns as $U = \begin{bmatrix} u_0 & u_1 & \cdots & u_{N-1} \end{bmatrix}$, the matrix V^H in block form as rows

$$V^H = \begin{bmatrix} v_0^H \\ v_1^H \\ \vdots \\ v_{N-1}^H \end{bmatrix},$$

and the matrix Y as the $N \times N$ matrix with components $Y(i,j)$. Then viewing the product UYV^H in block form as a $1 \times N$ matrix (the block matrix U) times a $N \times N$ matrix (the block matrix Y) times a $N \times 1$ matrix (the block matrix V^H) we get for X the product

$$\begin{bmatrix} u_0 & u_1 & \cdots & u_{N-1} \end{bmatrix} \begin{bmatrix} Y(0,0) & Y(0,1) & \cdots & Y(0,N-1) \\ Y(1,0) & Y(1,1) & \cdots & Y(1,N-1) \\ \vdots & \vdots & \vdots & \vdots \\ Y(N-1,0) & Y(N-1,1) & \cdots & Y(N-1,N-1) \end{bmatrix} \begin{bmatrix} v_0^H \\ v_1^H \\ \vdots \\ v_{N-1}^H \end{bmatrix}.$$

Using block matrix multiplication we have that the product of the two right most matrices is given by

$$\begin{bmatrix} Y(0,0)v_0^H + Y(0,1)v_1^H + \cdots + Y(0,N-1)v_{N-1}^H \\ Y(1,0)v_0^H + Y(1,1)v_1^H + \cdots + Y(1,N-1)v_{N-1}^H \\ \vdots \\ Y(N-1,0)v_0^H + Y(N-1,1)v_1^H + \cdots + Y(N-1,N-1)v_{N-1}^H \end{bmatrix}.$$

Or in summation notation

$$\begin{bmatrix} \sum_{j=0}^{N-1} Y(0,j)v_j^H \\ \sum_{j=0}^{N-1} Y(1,j)v_j^H \\ \vdots \\ \sum_{j=0}^{N-1} Y(N-1,j)v_j^H \end{bmatrix}.$$

Thus then X equals $\begin{bmatrix} u_0 & u_1 & \cdots & u_{N-1} \end{bmatrix}$ times this result or

$$\begin{aligned} X &= \sum_{j=0}^{N-1} Y(0,j)u_0 v_j^H + \sum_{j=0}^{N-1} Y(1,j)u_1 v_j^H + \cdots + \sum_{j=0}^{N-1} Y(N-1,j)u_{N-1} v_j^H \\ &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} Y(i,j)u_i v_j^H \,, \end{aligned}$$

as we were to show.

Part (b): To compute the value of $\langle \mathcal{A}_{ij}, X \rangle$ we first recall the definition of the matrix inner product $\langle \cdot, \cdot \rangle$ of

$$\langle A, B \rangle \equiv \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} A^*(m,n)B(m,n)\,, \tag{171}$$

and the definition of $\mathcal{A}_{ij}$ of $\mathcal{A}_{ij} = u_i v_j^H$. Then using the rank-one decomposition of X of

$$X = \sum_{i'=0}^{N-1}\sum_{j'=0}^{N-1} Y(i',j')u_{i'} v_{j'}^H \,, \tag{172}$$

Equation 171 then requires us to compute the (m,n)th component of the matrix $\mathcal{A}_{ij}$ and of $u_{i'} v_{j'}^H$ since $X(m,n)$ is obtained by summing such elements via Equation 172. Consider the (m,n)th component of $u_i v_j^H$. Recall u_i is the ith column of U and as v_j^H is the jth row of V^H we see that v_j is the jth column of V. Then the product $u_i v_j^H$ looks like

$$\begin{aligned} u_i v_j^H &= \begin{bmatrix} U(0,i) \\ U(1,i) \\ \vdots \\ U(N-1,i) \end{bmatrix} \begin{bmatrix} V(0,j)^* & V(1,j)^* & \cdots & V(N-1,j)^* \end{bmatrix} \\ &= \begin{bmatrix} U(0,i)V(0,j)^* & U(0,i)V(1,j)^* & \cdots & U(0,i)V(N-1,j)^* \\ U(1,i)V(0,j)^* & U(1,i)V(1,j)^* & \cdots & U(1,i)V(N-1,j)^* \\ \vdots & \vdots & \cdots & \vdots \\ U(N-1,i)V(0,j)^* & U(N-1,i)V(1,j)^* & \cdots & U(N-1,i)V(N-1,j)^* \end{bmatrix}. \end{aligned}$$

Thus the (m,n) element of the matrix $u_i v_j^H$ is

$$U(m,i)V(n,j)^*\,, \tag{173}$$

and the conjugate of the (m,n)th element of the matrix $u_i v_j^H$ is

$$U(m,i)^* V(n,j)\,. \tag{174}$$

Using these results we find

$$\begin{aligned}\langle A_{ij}, X\rangle &= \langle u_i v_j^H, \sum_{i'=0}^{N-1}\sum_{j'=0}^{N-1} Y(i',j')u_{i'}v_{j'}^H\rangle \\ &= \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} U(m,i)^* V(n,j)\left(\sum_{i'=0}^{N-1}\sum_{j'=0}^{N-1} Y(i',j')U(m,i')V(n,j')^*\right) \\ &= \sum_{i'=0}^{N-1}\sum_{j'=0}^{N-1} Y(i',j')\sum_{m=0}^{N-1} U(m,i)^*U(m,i')\sum_{n=0}^{N-1} V(n,j)V(n,j')^* .\end{aligned}$$

To evaluate these sums recall that U is a unitary matrices and thus $UU^H = I$ and $U^HU = I$. If we consider the (i,i')th element of the product $U^HU = I$ we get

$$\sum_{n=0}^{N-1}(U^H)(i,n)U(n,i') = I(i,i') ,$$

or

$$\sum_{n=0}^{N-1} U(n,i)^*U(n,i') = I(i,i') , \tag{175}$$

where $I(i,i')$ is the Kronecker delta symbol, i.e. $I(i,i') = 1$ if $i = i'$ and is 0 otherwise. Since V is also a Hermitian matrix a similar result hold for sums of components of V. Sums like this appear twice in the above expression for $\langle \mathcal{A}_{ij}, X\rangle$ and we have the following

$$\langle A_{ij}, X\rangle = \sum_{i'=0}^{N-1}\sum_{j'=0}^{N-1} Y(i',j')I(i,i')I(j,j') = Y(i,j) ,$$

as we were to show.

Problem 6.2 (separable transforms)

We first recall that to compute the lexicographic row ordered vector x the rows of the matrix X are ordered sequentially in a column vector. Thus if we let $X(i,:)$ be a row vector from X then the lexicographically ordered row vector x is given by

$$x = \begin{bmatrix} X(0,:)^T \\ X(1,:)^T \\ \vdots \\ X(N-1,:)^T \end{bmatrix} .$$

Next recall that if A is a $m \times n$ matrix and B is a $p \times q$ matrix the Kronecker outer product of two matrices A and B denoted as $A \otimes B$ is defined as the $mn \times pq$ matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & a_{13}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & a_{23}B & \cdots & a_{2n}B \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{m1}B & a_{m2}B & a_{m3}B & \cdots & a_{mn}B \end{bmatrix} . \tag{176}$$

Thus in terms of the matrices of this problem we have $U \otimes V$ given by the matrix

$$\begin{bmatrix} U(0,0)V & U(0,1)V & U(0,2)V & \cdots & U(0,N-1)V \\ U(1,0)V & U(1,1)V & U(1,2)V & \cdots & U(1,N-1)V \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ U(N-1,0)V & U(N-1,1)V & U(N-1,2)V & \cdots & U(N-1,N-1)V \end{bmatrix}.$$

Then when we multiply this by the lexicographic ordered vector x we get

$$\begin{bmatrix} U(0,0)VX(0,:)^T + U(0,1)VX(1,:)^T + \cdots + U(0,N-1)VX(N-1,:)^T \\ U(1,0)VX(0,:)^T + U(1,1)VX(1,:)^T + \cdots + U(1,N-1)VX(N-1,:)^T \\ \vdots \\ U(N-1,0)VX(0,:)^T + U(N-1,1)VX(1,:)^T + \cdots + U(N-1,N-1)VX(N-1,:)^T \end{bmatrix}.$$

This is a block column matrix of size $N \times 1$ where the blocks are $N \times N$ matrices with the m block element given by

$$\sum_{i=0}^{N-1} U(m,i)VX(i,:)^T .$$

Since $X(i,:)^T$ is a column vector the product $VX(i,:)^T$ is another column vector and the above is the sum of column vectors. The nth element of this column vector is given by

$$\sum_{j=0}^{N-1} V(n,j)X(i,j) .$$

Thus the nth element of the mth block in the product $(U \otimes V)x$ is

$$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X(i,j)U(m,i)V(n,j) .$$

If we have the desired equality this should equal the value $Y(m,n)$. To show this we can simply recall that $Y = UXV^T$ and as such we can compute the (m,n)th element of this product. Using the summation definition of a matrix product we find

$$\begin{aligned} Y(m,n) &= \sum_{i=0}^{N-1}(UX)(m,i)(V^T)(i,n) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} U(m,j)X(j,i)V(n,i) \\ &= \sum_{j=0}^{N-1}\sum_{i=0}^{N-1} X(j,i)U(m,j)V(n,i) , \end{aligned} \tag{177}$$

which is equivalent to the expression above showing the desired equivalence.

Problem 6.3 (minimizing the MSE by using the eigenvectors of R_x)

For a *fixed* orthonormal basis e_i for $i = 0, 1, 2, \cdots, N-1$, a random vector x the decomposition

$$x = \sum_{i=0}^{N-1} y(i)e_i , \tag{178}$$

with $y(i) = e_i^T x$. Note that since x is random the $y(i)$'s are also random. The projection of x in the m-dimensional subspace spanned by the vectors $e_0, e_1, \cdots, e_{m-1}$ is given by

$$\hat{x} = \sum_{i=0}^{m-1} y(i) e_i \,. \tag{179}$$

Then as in the book the expectation of the error ϵ defined as $\epsilon = x - \hat{x}$ can be shown to be given by

$$E\left[||\epsilon||^2\right] = \sum_{i=m}^{N-1} e_i^T R_x e_i \,, \tag{180}$$

with R_x the correlation matrix of x i.e. $R_x = E[xx^T]$. Considering $E\left[||\epsilon||^2\right]$ as the objective function to be minimized we seek vectors e_i that will achieve this minimum. Obviously $E\left[||\epsilon||^2\right] \geq 0$ and $e_i = 0$ will make the right-hand-side of Equation 180 zero. To avoid this trivial solution we need to introduced the constraint that the vectors e_i are normalized or $e_i^T e_i = 1$.

Question: I'm not sure why we don't have to *also* introduce the orthogonality constraint of $e_i^T e_j = 0$ for $i \neq j$. I think the answer might be because of the functional form for our objective function $E\left[||\epsilon||^2\right]$. For example, if the vectors for i and j appeared together as a product like $e_i A e_j$ for some matrix A in the objective function we would have to also introduce the constraint $e_i^T e_j = 0$. If anyone knows more about this or has an opinion on this please contact me.

Part (a): Given that we have the constraint $e_i^T e_i = 1$ we use the methods of constrained optimization to seek the optimum. That is we introduce Lagrange multipliers λ_i and form the Lagrangian

$$\mathcal{L} = \sum_{i=m}^{N-1} e_i^T R_x e_i - \sum_{i=0}^{N-1} \lambda_i (e_i^T e_i - 1) \,.$$

Then taking the derivative with respect to e_i for $i = m, m+1, \cdots, N-1$ and setting the result equal to zero gives

$$2R_x e_i - 2\lambda_i e_i = 0 \quad \Rightarrow \quad R_x e_i = \lambda_i e_i \,.$$

Thus e_i is the eigenvector and λ_i is the eigenvalue of R_x.

Part (b): In this case using the normalization properties of e_i we have that

$$E\left[||\epsilon||^2\right] = \sum_{i=m}^{N-1} \lambda_i \,.$$

Thus to make this as small as possible we want to take e_i to be the eigenvectors with the smallest eigenvalues.

Part (c): For the given approximation above consider the magnitude of the variance of $\hat{x}$.

We find

$$\begin{aligned}
\text{Var}(\hat{x}) &= E[\hat{x}^T\hat{x}] = E\left[\left(\sum_{i=0}^{m} y(i)e_i^T\right)\left(\sum_{i=0}^{m} y(i)e_i\right)\right] \qquad (181)\\
&= E\left[\sum_{i=0}^{m}\sum_{i=0}^{m} y(i)y(j)e_i^T e_j\right] \\
&= E\left[\sum_{i=0}^{m} y(i)^2\right] = E\left[\sum_{i=0}^{m}(e_i^T x)^2\right] = E\left[\sum_{i=0}^{m} e_i^T xx^T e_i\right] = \sum_{i=0}^{m} e_i^T E\left[xx^T\right] e_i \\
&= \sum_{i=0}^{m} e_i^T R_x e_i = \sum_{i=0}^{m}\lambda_i\,.
\end{aligned}$$

Thus since e_i are chosen to be the eigenvectors of R_x ordered from largest eigenvalue to smallest eigenvalue we see that this sum is maximal.

Problem 6.4 (Karhunen-Loeve with the covariance matrix Σ_x)

In the same way as earlier we have a representation of our random variable x given by Equation 178 and now we will take our approximation of x given by

$$\hat{x} = \sum_{i=0}^{m-1} y(i)e_i + \sum_{i=m}^{N-1} c_i e_i\,, \qquad (182)$$

with $y(i) = e_i^T x$.

Part (a): Consider the expected square error $E[||x-\hat{x}||^2]$. We find

$$\begin{aligned}
E[||x-\hat{x}||^2] &= E\left[\left|\left|\sum_{i=m}^{N-1}(y_i - c_i)e_i\right|\right|^2\right] \\
&= E\left[\sum_{i=m}^{N-1}\sum_{j=m}^{N-1}(y_i-c_i)(y_j-c_j)e_i^T e_j\right] \\
&= E\left[\sum_{i=m}^{N-1}(y_i-c_i)^2\right] \\
&= \sum_{i=m}^{N-1}(E[y_i^2] - 2E[y_i]c_i + c_i^2)\,.
\end{aligned}$$

If we want to pick c_i's that make this as small as possible, we can take the derivative with respect to c_i set the result equal to zero and solve for c_i we find

$$\frac{\partial}{\partial c_i}E[||x-\hat{x}||^2] = 0 \quad \Rightarrow \quad -2E[y_i] + 2c_i = 0\,.$$

This gives $c_i = E[y_i]$, for $i = m, m+1, \cdots, N-1$.

Part (b): We now want to ask for an approximation to x given by

$$\hat{x} = \sum_{i=0}^{m-1} y_i e_i + \sum_{i=m}^{N-1} E[y_i] e_i \,,$$

how do we pick the orthonormal basis vectors e_i. We do that by minimizing the square norm of the error ϵ defined as $\epsilon = x - \hat{x}$. We find using the same techniques as the sequence of steps around Equation 181 and recalling that $y_i = e_i^T x$ we have

$$\begin{aligned} E[||\epsilon||^2] &= E\left[\sum_{i=m}^{N-1}(y_i - E[y_i])^2\right] = E\left[\sum_{i=m}^{N-1}(e_i^T x - e_i^T E[x])^2\right] \\ &= E\left[\sum_{i=m}^{N-1}(e_i^T(x - E[x]))^2\right] = E\left[\sum_{i=m}^{N-1} e_i^T(x - E[x])(x - E[x])^T e_i\right] \\ &= \sum_{i=m}^{N-1} e_i^T E[(x - E[x])(x - E[x])^T] e_i = \sum_{i=m}^{N-1} e_i^T \Sigma_x e_i \,. \end{aligned} \tag{183}$$

Thus to pick the orthonormal basis that minimizes $E[||\epsilon||^2]$ we minimize Equation 183 subject to the constraint that $e_i^T e_i = 1$. Introducing Lagrange multipliers like in the previous problem we find e_i are the eigenvectors of Σ_x.

Part (b): To make the expression for $E[||\epsilon||^2]$ as small as possible we we order these eigenvectors so that they are ranked in decreasing order of their eigenvalues, therefore the vectors $e_m, e_{m+1}, \cdots, e_{N-1}$ will be the eigenvectors of Σ_x corresponding to the $N-m$ smallest eigenvalues.

Problem 6.5 (the eigenvalues of $X^H X$ and XX^H are the same)

Let λ be a nonzero eigenvalue of XX^H with eigenvector v. Then by definition $XX^H v = \lambda v$. Now consider the vector $\hat{v}$ defined by $\hat{v} = X^H v$. Then

$$X^H X \hat{v} = X^H X X^H v = \lambda X^H v = \lambda \hat{v} \,.$$

This last expression shows that $\hat{v}$ is an eigenvector of $X^H X$ with eigenvalue λ. Thus both XX^H and $X^H X$ have the same eigenvalues.

Problem 6.6 (proving $\epsilon^2 \equiv \sum_{m=0}^{N} \sum_{n=0}^{N} |X(m,n) - \hat{X}(m,n)|^2 = \sum_{i=k}^{r-1} \lambda_i$)

Recall the definition of ϵ^2 where we have

$$\epsilon^2 = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} |X(m,n) - \hat{X}(m,n)|^2 \,. \tag{184}$$

Since we have our rank k approximate matrix $\hat{X}$ given by

$$\hat{X} = \sum_{i=0}^{k-1} \sqrt{\lambda_i} u_i v_i^H \,,$$

while the full decomposition for X is

$$X = \sum_{i=0}^{r-1} \sqrt{\lambda_i} u_i v_i^H \,.$$

We have the matrix difference $X - \hat{X}$ given by

$$X - \hat{X} = \sum_{i=k}^{r-1} \sqrt{\lambda_i} u_i v_i^H \,.$$

Thus the m, nth element of this matrix difference $X - \hat{X}$ is given by using Equation 173

$$\sum_{i=k}^{r-1} \sqrt{\lambda_i}\, U(m,i) V(n,i)^* \,.$$

Now recall that for a complex number $|x|^2 = xx^*$ when we "square" the above expression we have

$$|X(m,n) - \hat{X}(m,n)|^2 = \sum_{i=k}^{r-1} \sum_{j=k}^{r-1} \sqrt{\lambda_i}\sqrt{\lambda_j} U(m,i) V(n,i)^* U(m,j)^* V(n,j) \,.$$

It is this expression that we will sum for m and n both running from 0 to $N-1$. When we apply this summation, then exchange the order of the sums and use Equation 175 we get

$$\begin{aligned} \epsilon^2 &= \sum_{i=k}^{r-1} \sum_{j=k}^{r-1} \sqrt{\lambda_i}\sqrt{\lambda_j} \sum_{m=0}^{N-1} U(m,i) U(m,j)^* \sum_{n=0}^{N-1} V(n,j) V(n,i)^* \\ &= \sum_{i=k}^{r-1} \sum_{j=k}^{r-1} \sqrt{\lambda_i}\sqrt{\lambda_j} I(i,j) I(j,i) = \sum_{i=k}^{r-1} \lambda_i \,, \end{aligned}$$

as we were to show.

Problem 6.7 (an example with the SVD)

The SVD decomposition of a matrix X is given by

$$X = \sum_{i=0}^{r-1} \sqrt{\lambda_i} u_i v_i^H \,,$$

where u_i and v_i are the eigenvectors (with common eigenvalue λ_i) of XX^H and $X^H X$ respectively or

$$\begin{aligned} XX^H u_i &= \lambda_i u_i \\ X^H X v_i &= \lambda_i v_i \end{aligned}$$

It turns out that u_i and v_i are related more directly as $u_i = \frac{1}{\sqrt{\lambda_i}} X v_i$. In the MATLAB script `chap_6_prob_7.m` we compute XX^H and $X^H X$ and the eigenvector and eigenvalues for these two matrices. We first find that $X^H X$ has u_i eigenvectors (stored as columns) and eigenvalues given by

$$\begin{bmatrix} 0.8452 & 0.0998 & 0.5251 \\ -0.1690 & -0.8821 & 0.4397 \\ -0.5071 & 0.4604 & 0.7286 \end{bmatrix} \quad \text{and} \quad 0.0\,, 1.93\,, 18.06\,.$$

We next find that XX^H has v_i eigenvectors (again stored as columns) with eigenvalues given by

$$\begin{bmatrix} -0.8649 & 0.5019 \\ 0.5019 & 0.8649 \end{bmatrix} \quad \text{and} \quad 1.93\,, 18.06\,.$$

To use the SVD decomposition one has to match the eigenvectors of XX^H and $X^H X$ to use in the inner product so that their eigenvalues match. This means the decomposition of X is given by

$$\sqrt{18.06} \begin{bmatrix} 0.5251 \\ 0.4397 \\ 0.7286 \end{bmatrix} \begin{bmatrix} 0.5019 & 0.8649 \end{bmatrix} + \sqrt{1.937} \begin{bmatrix} 0.0998 \\ -0.8821 \\ 0.4604 \end{bmatrix} \begin{bmatrix} -0.8649 & 0.5019 \end{bmatrix}\,.$$

Problem 6.8 (the orthogonality of the DFT)

We begin by proving the following sum

$$\frac{1}{N} \sum_{n=0}^{N-1} \exp\left(j \frac{2\pi}{N} (k-l) n \right) = \begin{cases} 1 & l = k + rN \quad r = 0, \pm 1, \pm 2, \cdots \\ 0 & \text{otherwise} \end{cases} \tag{185}$$

Since the sum above is a geometric sum from [6] we can evaluate it as

$$\begin{aligned} \sum_{n=0}^{N-1} \exp\left(j \frac{2\pi}{N} (k-l) n \right) &= \frac{1 - \exp\left(j \frac{2\pi}{N} (k-l) N \right)}{1 - \exp\left(j \frac{2\pi}{N} (k-l) \right)} \\ &= \frac{1 - \exp\left(j 2\pi (k-l) \right)}{1 - \exp\left(j \frac{2\pi}{N} (k-l) \right)}\,. \end{aligned}$$

This is true only if the expression we are summing powers of is not identically 1 (for which the denominator would be 0 and division is undefined). This latter will happen if the argument of the exponential $\exp\left(j \frac{2\pi}{N} (k-l) \right)$ is a multiple of 2π. This means the above sum is valid if

$$\frac{k-l}{N} \neq r\,,$$

where r is an integer. If this previous condition holds then the numerator vanishes

$$1 - \exp\left(j 2\pi (k-l) \right) = 0\,.$$

since $k-l$ is an integer. If $\frac{k-l}{N}$ is actually equal to and integer r then $l = k + rN$ and the sum is N and we have proven Equation 185.

Now consider the (m,n) element of the product $W^H W$. Using Equation 160 and 162 we get

$$\begin{aligned}(W^H W)(m,n) &= \frac{1}{N}\sum_{k=0}^{N-1} W_N^{mk} W_N^{-kn} \\ &= \frac{1}{N}\sum_{k=0}^{N-1} W_N^{k(m-n)} = \frac{1}{N}\sum_{k=0}^{N-1} \exp\left(-j\frac{2\pi}{N}(m-n)k\right) = \delta_{mn}\,,\end{aligned}$$

when we use Equation 185. Here δ_{mn} is the Kronecker delta.

Problem 6.9 (an example computing a 2-d DFT)

Using Equation 164 or $Y = W^H X W^H$ we can compute the two-dimensional DFT by computing the required matrix product. To generate the matrix W^H of order N (as defined in the book) we can use the MATLAB command `dftmtx(N)/sqrt(N)`. In the MATLAB script `chap_6_prob_9.m` given in the input matrix X we do this and perform the required matrix multiplications. We find that we get

$$Y = \begin{bmatrix} 3.6667 & -0.3333 & 0.1667 + 0.8660j \\ -0.3333 & 0.1667 - 0.8660j & 0.1667 - 0.8660j \\ -0.3333 & 0.1667 + 0.8660j & -0.3333 \end{bmatrix}.$$

Problem 6.11 (orthogonality of the discrete cosine transform)

The discrete cosine transform (DCT) matrix C has elements $C(n,k)$ given by

$$\begin{aligned}C(n,k) &= \frac{1}{\sqrt{N}} \quad \text{when} \quad k = 0 \\ C(n,k) &= \sqrt{\frac{2}{N}}\cos\left(\frac{\pi(2n+1)k}{2N}\right) \quad \text{when} \quad k > 1\,,\end{aligned} \tag{186}$$

and for $0 \le n \le N-1$. We want to show that $C^T C = I$. To do this consider the (i,j)th element of this product (denoted by $(C^T C)(i,j)$) we have

$$(C^T C)(i,j) = \sum_{k=0}^{N-1} C^T(i,j) C(k,j) = \sum_{k=0}^{N-1} C(k,i) C(k,j)\,. \tag{187}$$

Lets evaluate this for various values of i and j. When $i = 0$ and $j = 0$ Equation 187 gives

$$\sum_{k=0}^{N-1} C(k,0)C(k,0) = \sum_{k=0}^{N-1} C(k,0)^2 = \sum_{k=0}^{N-1} \frac{1}{N} = 1\,.$$

When $i = 0$ and $j \geq 1$ Equation 187 gives

$$\begin{aligned}(C^T C)(0,j) &= \sum_{k=0}^{N-1} C(k,0)C(k,j) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} C(k,j) \\ &= \frac{\sqrt{2}}{N} \sum_{k=0}^{N-1} \cos\left(\frac{\pi(2k+1)j}{2N}\right)\end{aligned}$$

By writing the summand above as

$$\cos\left(\frac{\pi(2k+1)j}{2N}\right) = \cos\left(\frac{\pi j}{2N} + \frac{\pi j}{N}k\right),$$

we can use the following identity [5]

$$\sum_{k=0}^{N-1} \cos(\alpha + \beta k) = \cos\left(\alpha + \frac{N-1}{2}\beta\right) \frac{\sin\left(\frac{N}{2}\beta\right)}{\sin\left(\frac{\beta}{2}\right)}, \tag{188}$$

with $\alpha = \frac{\pi j}{2N}$ and $\beta = \frac{\pi j}{N}$ to evaluate it. In that case we have

$$\frac{\beta}{2} = \frac{\pi j}{2N}, \quad \text{so} \quad \frac{N\beta}{2} = \frac{\pi j}{2}, \quad \text{and} \quad \alpha + \frac{N-1}{2} = \frac{\pi j}{2N} + \left(\frac{N-1}{2}\right)\frac{\pi j}{N} = \frac{\pi j}{2}.$$

Thus we have

$$\sum_{k=0}^{N-1} \cos\left(\frac{\pi(2k+1)}{2N}\right) = \cos\left(\frac{\pi j}{2}\right) \frac{\sin\left(\frac{\pi j}{2}\right)}{\sin\left(\frac{\pi j}{2N}\right)}.$$

Since for $j = 1, 2, \cdots, N-1$ the value of $\frac{\pi}{2}j$ is a multiple of $\frac{\pi}{2}$ where we have $\cos\left(\frac{\pi j}{2}\right) = 0$ thus

$$(C^T C)(0,j) = \sum_{k=0}^{N-1} \cos\left(\frac{\pi(2k+1)}{2N}\right) = 0.$$

Next let $i \geq 1$ and $j = 0$ and we have

$$\begin{aligned}(C^T C)(i,0) &= \sum_{k=0}^{N-1} C(k,i)C(k,0) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} C(k,i) \\ &= \frac{\sqrt{2}}{N} \sum_{k=0}^{N-1} \cos\left(\frac{\pi(2k+1)i}{2N}\right) = 0,\end{aligned}$$

since this is the same sum we evaluated earlier. Finally, let $i \geq 1$ and $j \geq 1$ to get

$$(C^T C)(i,j) = \sum_{k=0}^{N-1} C(k,i)C(k,j) = \frac{2}{N} \sum_{k=0}^{N-1} \cos\left(\frac{\pi(2k+1)i}{2N}\right) \cos\left(\frac{\pi(2k+1)j}{2N}\right).$$

To evaluate this sum we could convert the trigonometric functions into exponentials and then use the sum of a geometric series identity to evaluate each sum, or we can evaluate it using Mathematica. In the Mathematica notebook `chap_6_prob_11.nb` we find it equals

$$\frac{1}{4}\left(\frac{\sin(\pi(i-j))}{\sin\left(\frac{\pi(i-j)}{2N}\right)} + \frac{\sin(\pi(i+j))}{\sin\left(\frac{\pi(i+j)}{2N}\right)}\right),$$

when neither of the denominators is zero, or in this case that $i \neq j$. When $i \neq j$ both terms in the numerator vanish and this expression is zero. If $i = j$ then we want to evaluate

$$(C^T C)(i,i) = \frac{2}{N} \sum_{k=0}^{N-1} \cos\left(\frac{\pi(2k+1)i}{2N} \right)^2 .$$

Using the following identity

$$\sum_{k=0}^{N-1} \cos\left(\frac{\pi(2k+1)i}{2N} \right)^2 = \frac{N}{2} + \frac{\sin(2\pi i)}{4\sin\left(\frac{\pi i}{N}\right)} , \tag{189}$$

as i is an integer the second term vanishes and we have shown that

$$(C^T C)(i,i) = 1 .$$

All of these elements show that $C^T C = I$ the desired expression.

Problem 6.12 (the discrete cosign transform)

The discrete cosign transform (DCT) Y of an image X is given by computing

$$Y = C^T X C ,$$

where C is the matrix with elements given by Equation 186. This matrix can be constructed using that MATLAB function `mk_DCT_matrix_C.m`. Then using the X matrix given in for this problem in the MATLAB script `chap_6_prob_12.m` we find Y to be

$$Y = \begin{bmatrix} 3.6667 & -0.4082 & -0.2357 \\ -0.4082 & 0 & -1.1547 \\ 1.1785 & 0.5774 & 0.3333 \end{bmatrix} .$$

Problem 6.14 (orthogonality of the Hadamard transform)

To begin recall the recursive definition of H_n given by

$$\begin{aligned} H_n &= H_{n-1} \otimes H_1 \\ &= H_{n-1} \otimes \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \right) \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix} . \end{aligned} \tag{190}$$

We will show that $H_n^T = H_n$ and $H_n^{-1} = H_n$. To do that we will use recursion, where we will show that these two relationships are true for $n = 1$ and then assume that they hold up to

and equal to some index n. We will then show that we can prove that the relationships hold for the index $n+1$. Consider H_1 we see that $H_1^T = H_1$ and

$$H_1^T H_1 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = I\,.$$

Then assume that $H_n^T = H_n$ and $H_n^{-1} = H_n^T$. Consider

$$H_{n+1}^T = \frac{1}{\sqrt{2}}\begin{bmatrix} H_n^T & H_n^T \\ H_n^T & -H_n^T \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix} = H_{n+1}\,,$$

showing that H_{n+1} is symmetric. Now consider

$$\begin{aligned} H_{n+1}^T H_{n+1} &= H_{n+1}H_{n+1} = \frac{1}{2}\begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}\begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix} 2H_n^2 & 0 \\ 0 & 2H_n^2 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} = I\,, \end{aligned}$$

showing that $H_{n+1}^{-1} = H_{n+1}^T$.

Problem 6.15 (computing the Hadamard transform)

We can use the MATLAB command `hadamard` to compute the Hadamard matrix H_n as defined in the book. Specifically, we have

$$H_n = \frac{1}{2^{n/2}}\text{hadamard}(2^n)\,.$$

In the MATLAB script `chap_6_prob_15.m` given in the input matrix X we compute $Y = H_1\hat{X}H_1$, where $\hat{X}$ is a submatrix of the original matrix X.

Problem 6.17 (the Noble identities)

Noble Downsampling Identity: Recall that downsampling by M produces a new sequence $y(k)$ generated by the old sequence $\hat{y}(k)$ according to

$$y(k) = \hat{y}(Mk)\,. \tag{191}$$

The transfer function for this operation, $D(z)$, when the input signal is $\hat{y}(k)$ is given by [8].

$$D(z) = \frac{1}{M}\sum_{k=0}^{M-1}\hat{Y}\left(z^{1/M}e^{-\frac{2\pi j}{M}k}\right)\,, \tag{192}$$

where $i = \sqrt{-1}$. Using that we can write the serial affect of downsampling followed by filtering with $H(z)$ as

$$\frac{1}{M}\sum_{k=0}^{M-1}\hat{Y}\left(z^{1/M}e^{-\frac{2\pi j}{M}k}\right)H(z)\,.$$

If we consider the combined system of filtering with $H(z^M)$, to get $H(z^M)\hat{Y}(z)$, and then downsampling we have that the combined affect is given by

$$\frac{1}{M}\sum_{k=0}^{M-1}\hat{Y}\left(z^{1/M}e^{\frac{2\pi j}{M}k}\right)H\left(\left(z^{1/M}e^{\frac{2\pi j}{M}k}\right)^M\right)=\frac{1}{M}\sum_{k=0}^{M-1}\hat{Y}\left(z^{1/M}e^{\frac{2\pi j}{M}k}\right)H\left(z\right)\,,$$

the same expression as before, showing the equivalence.

Noble Upsampling Identity: Recall that upsampling by M produces the new sequence $y(k)$ generated by the old sample function $\hat{y}(k)$ according to

$$y(k)=\begin{cases}\hat{y}\left(\frac{k}{M}\right) & \text{when } \frac{k}{M} \text{ is an integer}\\ 0 & \text{otherwise}\end{cases} \tag{193}$$

Then the transfer function for this operation $U(z)$ when the input signal is $\hat{y}$ is given by [8]

$$U(z)=\hat{Y}(z^M)\,. \tag{194}$$

Then the affect of filtering with $H(z)$ a signal with transfer function $\hat{Y}(z)$ is $H(z)\hat{Y}(z)$. Following this by upsampling by M gives the transfer function

$$H(z^M)\hat{Y}(z^M)\,.$$

This is the same as taking the input $\hat{Y}(z)$ upsampling by M to get $\hat{Y}(z^M)$ and then passing that output through the linear system $H(z^M)$, showing the equivalence of the two Noble upsampling forms.

Problem 6.18 (an equivalent filter bank representation)

In figure 6.5 we have three paths to generate the outputs y_0, y_1, and y_2. When drawn without the traditional system box notation we get these three paths given by

$$\begin{array}{lllllllll}
\cdots,x(2),x(1),x(0) & \to & H_0(z) & \to & \downarrow 2 & \to y_0 & & & \\
\cdots,x(2),x(1),x(0) & \to & H_1(z) & \to & \downarrow 2 & \to H_0(z) & \to & \downarrow 2 & \to y_1 \\
\cdots,x(2),x(1),x(0) & \to & H_1(z) & \to & \downarrow 2 & \to H_1(z) & \to & \downarrow 2 & \to y_2\,.
\end{array}$$

The system for y_0 matches the same system in Figure 6.6b for y_0 when we take $\hat{F}_0(z)=H_0(z)$. If we then use Noble's downsampling identity we can write the output expressions for y_1 and y_2 as

$$\begin{array}{lllllllll}
\cdots,x(2),x(1),x(0) & \to & H_1(z) & \to H_0(z^2) & \to & \downarrow 2 & \to & \downarrow 2 & \to y_1 \\
\cdots,x(2),x(1),x(0) & \to & H_1(z) & \to H_1(z^2) & \to & \downarrow 2 & \to & \downarrow 2 & \to y_2\,.
\end{array}$$

We can simplify the two downsampling procedures of size 2 on the right side of these expressions into one downsampling expression of size 4, and combine the two serial systems to get

$$\begin{array}{llllll}
\cdots,x(2),x(1),x(0) & \to & H_1(z)H_0(z^2) & \to & \downarrow 4 & \to y_1 \\
\cdots,x(2),x(1),x(0) & \to & H_1(z)H_1(z^2) & \to & \downarrow 4 & \to y_2\,.
\end{array}$$

This matches the system drawn in Figure 6.6 when we take $\hat{F}_1(z)=H_1(z)H_0(z^2)$ and $\hat{F}_2(z)=H_1(z)H_1(z^2)$ proving the equivalence.

Feature Generation II

Notes on the text

Notes on co-occurrence matrices

The co-occurrence matrices provide a way to measure the relative position of gray levels in an image and as such are functions of a pixel displacement d and an angle offset ϕ. They rely on the joint probabilities that the given image takes gray level values at the angular direction ϕ and a distance d. Typically $d = 1$ while ϕ is taken to be from $\{0, 45, 90, 135\}$ in degrees. That is for $\phi = 0$ we need to compute

$$P(I(m,n) = I_1, I(m \pm d, n) = I_2) = \frac{\text{number of pairs of pixels with levels} = (I_1, I_2)}{\text{total number of possible pixel pairs}} .$$

Probability density functions can be obtained for $\phi = 45$ where the probability we are extracting can be written

$$P(I(m,n) = I_1, I(m \pm d, n \mp d) = I_2) .$$

To make things easier to understand assume that we have *four* gray levels then $I(m,n) \in \{0,1,2,3\}$ and we can define the co-occurrence matrix A (with elements $P(I_1, I_2)$ above) when given a specification of (d, ϕ) as

$$A(d,\phi) = \frac{1}{R} \begin{bmatrix} \eta(0,0) & \eta(0,1) & \eta(0,2) & \eta(0,3) \\ \eta(1,0) & \eta(1,1) & \eta(1,2) & \eta(1,3) \\ \eta(2,0) & \eta(2,1) & \eta(2,2) & \eta(2,3) \\ \eta(3,0) & \eta(3,1) & \eta(3,2) & \eta(3,3) \end{bmatrix} , \tag{195}$$

The book writes these A matrices with ϕ as a superscript as $A^{\phi}(d)$. Here $\eta(I_1, I_2)$ is the *number* of pixel pairs at a relative position of (d, ϕ) which have a gray level pair (I_1, I_2) respectively and R is the total number of pixel pairs in that orientation in the given image. Lets consider in some detail how to compute the expression R, the *number* of pairs of pixels we have to consider, in the co-occurrence matrix above for some common orientations. Lets first consider the case where $d = 1$ and $\phi = 0$. Then anchored at each pixel of the image we will try to look left and right (since $\phi = 0$) by one (since $d = 1$) and consider the resulting image values (I_1, I_2). For simplicity assume our input image has four rows/columns. Now at the position $(0,0)$ (upper left corner) we can only look right (otherwise we are looking off the image) which gives one pixel pair. If we are at the pixel $(0,1)$ we can look left and right for two pairs of pixels. At the pixel located at $(0,2)$ we can look left and right giving two pixel pairs, and finally at the pixel $(0,3)$ we can look only left giving one pixel pair. In total this is

$$1 + 2 + 2 + 1 = 6 ,$$

left/right pairs per row. Since we have four rows we have $R = 6 \cdot 4 = 24$ possible pairs. In general for an image of dimension $M \times N$ with M rows and N columns we have

$$1 + 2(N-2) + 1 = 2(N-2) + 2 = 2N - 2 ,$$

pixel pairs in each row, so

$$R(d=1,\phi=0) = M(2N-2)\,.$$

If $\phi = 90$ (again with $d=1$) we have $1+2(M-2)+1 = 2M-2$ pixel pairs in each column so with N columns we have

$$R(d=1,\phi=90) = N(2M-2)\,,$$

pixel pairs to consider. If $\phi = 45$ the first row has *no* pixel pairs in the northeast direction or of the form $(I(m,n), I(m-d,n+d))$ since we would be looking off the image, but has $N-1$ pairs in the southwest direction of the form $(I(m,n), I(m+d,n-d))$. All rows but the last have $2(N-2)+2 = 2N-2$ pixel pairs to consider. The last row has $N-1$ pixel pairs of the form $(I(m,n), I(m-d,n+d))$ and none like $(I(m,n), I(m+d,n-d))$. Thus in total we have

$$R(d=1,\phi=45) = 2(N-1) + (M-2)(2N-2) = 2(N-1)(M-1)\,.$$

The pair count for $R(d=1,\phi=135)$ should equal that of $R(d=1,\phi=45)$ but with N and M exchanged,

$$R(d=1,\phi=135) = 2(M-1)(N-1)\,.$$

In practice, a very simple algorithm for computing the co-occurrence matrix for given values of d and ϕ is to start with $\eta(I_1,I_2) = 0$ and then walk the image over all pixels looking at the image values (I_1,I_2) of the two pixels specified by the (d,ϕ) inputs and incrementing the corresponding $\eta(I_1,I_2)$. The value of A can then be obtained after the fact by summing all the elements of the η array. In the MATLAB function `co_occurrence.m` using this very simple algorithm we compute the co-occurrence matrix for an input (d,ϕ). For the sample input image

$$I = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 1 & 1 & 0 & 0 \\ 3 & 2 & 3 & 3 \\ 3 & 2 & 2 & 2 \end{bmatrix},$$

for $(d,\phi) = (1,0)$ this routine gives

$$A(1,0) = \frac{1}{24}\begin{bmatrix} 4 & 1 & 1 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 0 & 6 & 3 \\ 0 & 0 & 3 & 2 \end{bmatrix},$$

the same as in the book. In the MATLAB script `dup_co_occurrence_example.m` we duplicate all of the co-occurrence matrices from this section.

Notes on second-order statistics features

We implemented several of the summary second-order statistics discussed in this section as MATLAB functions. In particular we implemented

- The *Angular second moment* in `ASM.m`.
- The *Contrast* measure in `CON.m`.
- The *Inverse difference moment* measure in `IDF.m`.
- The *Entropy* measure in `entropy.m`.

In general, these routines can take additional input argument parameters d and ϕ that specify the displacement and angular offset that are used in computing the given co-occurrence matrix one uses in the feature definition. If these additional arguments are not given then the functions above compute all four possible co-occurrence matrices, the corresponding feature from each, and then return the average of the four feature measurements.

Notes on gray level run lengths

For these run length features we pick a direction ϕ as before, then the matrix $Q_{RL}(\phi)$ has its (i,j)th element given by the number of time that the gray level i for $i = 0, 1, \cdots N_g - 1$ appears with a run length $j = 1, 2, \cdots N_r$ in the direction ϕ. Thus Q_{RL} is a matrix of dimension $N_g \times N_r$. We implemented the computation of the gray level run length matrix Q_{RL} in the MATLAB function `Q_RL.m`. Using that in the MATLAB function `dup_run_length_example.m` we duplicated the examples computing $Q_{RL}(0)$ and $Q_{RL}(45)$. We then implemented a number of derivative features based on $Q_{RL}(\phi)$ such as

- The *short run emphasis* in `SRE.m`.
- The *long run emphasis* measure in `LRE.m`.
- The *gray level nonuniformity* measure in `GLNU.m`.
- The *run length nonuniformity* measure in `RLN.m`.
- The *run percentage* measure in `RP.m`.

The above computations are function of the angle ϕ. As in the second order statistics above, to make these features rotation invariant we compute them for each of the four possible $\phi \in \{0, 45, 90, 135\}$ values, and then return the average of these four feature measurements.

Notes on local linear transformations

Given the three basis vectors: $b_1 = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$, for a local average, $b_2 = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ for local edge detection, and $b_3 = \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$ for local spot detection then the N^2 local filters are given by all possible outer products or

$$b_1 b_1^T \, , b_1 b_2^T \, , b_1 b_3^T \, , b_2 b_1^T \, , b_2 b_2^T \, , b_2 b_3^T \, , b_3 b_1^T \, , b_3 b_2^T \, , b_3 b_3^T \, .$$

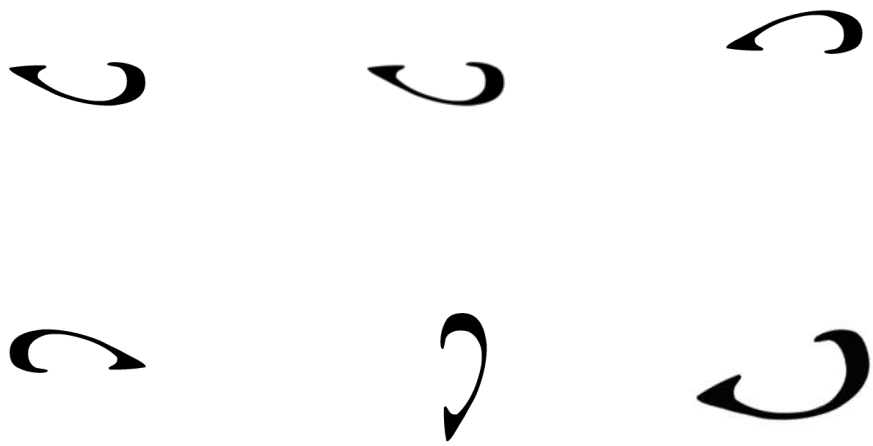

Figure 17: Duplicate "petasti" figures used to extract Hu invariant moments from.

For example

$$b_1^T b_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

$$b_1^T b_2 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}.$$

All outer products are computed in the MATLAB script `gen_local_linear_transformations.m`, which duplicate the 9 matrices presented in the text.

Notes on geometric moments

In this section I attempted to duplicate the results in the book on using the *Hu* moments for image features extraction. I first got a set of images of the "petasti" symbol, see Figure 17. Then in the MATLAB code `dup_Hu_moments.m` we load in `gif` versions of these images and call the function `Hu_moments.m`. I scaled each image to the x and y ranges $[-1, +1]$ before computing the moments. The dynamic ranges of the higher center moments is significantly smaller that the earlier ones (and some seem to vanish to zero), thus to compare the extracted moments from each images we extract and plot the logarithm of the absolute value of the direct moments this is suggested in [4]. This gives the following

```
            phi_1     phi_2     phi_3     phi_4     phi_5     phi_6     phi_7
image a:  -14.6794  -43.7024  -59.5204  -55.4081 -115.3626  -80.8707 -112.8758
image b:  -15.3285  -43.7243  -60.1944  -56.1308 -118.0532  -80.5011 -114.2937
```

```
image c:  -14.6794  -43.7024  -59.5204  -55.4081 -115.3626  -80.8707 -112.8758
image d:  -14.6794  -43.7024  -59.5204  -55.4081 -115.3626  -80.8707 -112.8758
image e:  -14.6794  -43.7024  -59.5204  -55.4081 -115.3626  -80.8707 -112.8758
image f:  -17.0347  -36.9026  -57.2860  -56.1632 -113.1657  -74.8205 -113.3142
```

In general these numbers look very similar for each of the images. I was, however, unable to duplicate the exact numerical results for the value of ϕ_i from the book. If anyone sees anything wrong with what I have done or a way that I can better match the books results please let me know.

Notes on Zernike moments

The indices for the Zernike moments require that $p = 0, 1, 2, \cdots$ and $|q| \leq p$ with $p - |q|$ even. This means that we have the following valid combinations (for a few value of p only)

- $p = 0$ so $q = 0$ only.
- $p = 1$ so $q = 1$ only.
- $p = 2$ so $q \in \{-2, 0, +2\}$.
- $p = 3$ so $q \in \{-3, -1, +1, +3\}$.
- $p = 4$ so $q \in \{-4, -2, 0, +2, +4\}$.
- $p = 5$ so $q \in \{-5, -3, -1, +1, +3, +5\}$.

The pattern at this point seems clear.

Notes on Fourier features

Consider the complex boundary u_k with the origin shifted by the index k_0 or the signal u_{k-k_0}. Then this shifted boundary has Fourier features given by

$$f'_l = \sum_{k=0}^{N-1} u_{k-k_0} e^{-j\frac{2\pi}{N}lk} = \sum_{k=-k_0}^{N-1-k_0} u_k e^{-j\frac{2\pi}{N}l(k+k_0)} = e^{-j\frac{2\pi}{N}lk_0} \sum_{k=-k_0}^{N-1-k_0} u_k e^{-j\frac{2\pi}{N}lk} .$$

This last sum is $e^{-j\frac{2\pi}{N}lk_0} f_l$, since both u_k and $e^{-j\frac{2\pi}{N}lk}$ are periodic in the index k with a period N. Thus we have shown

$$f'_l = e^{-j\frac{2\pi}{N}lk_0} f_l . \tag{196}$$

Note that a shift of origin does not change the *magnitude* of the Fourier coefficients.

The book then presents an argument as to why the *normalized* Fourier coefficients are the index location invariant. We present a somewhat different version of that argument here. Since the Fourier coefficients change depending on the k origin (see Equation 196 above) we would like to define Fourier features that are independent of this choice in origin. One way to do that is the following. One simply computes the Fourier features directly using the definition

$$f_l = \sum_{k=0}^{N-1} u_k e^{-j\frac{2\pi}{N}lk}\,, \tag{197}$$

ignoring any issue of k origin. Then we explicitly write the *first* Fourier complex number f_1 in polar form as $f_1 = |f_1|e^{-j\phi_1}$ (note the negative sign in the angle). With this definition of ϕ_1 as the polar angle for the first Fourier feature we define the **normalized Fourier coefficients** $\hat{f}_l$ as

$$\hat{f}_l = f_l \exp(jl\phi_1)\,, \tag{198}$$

that is we multiply each of the previously computed Fourier coefficient by a power of the complex phase of f_1. We claim that these normalized Fourier coefficients are invariant to the choice of the sampling origin in k. To show this imagine that we had selected a different origin (say k_0) to evaluate u_k at. Then from Equation 196 the first Fourier feature would be transformed to f_1' given by the product of the old value f_1 times a phase shift or

$$f_1' = f_1 e^{-j2\pi\frac{k_0}{N}} = |f_1|e^{-j\phi_1}e^{-j2\pi\frac{k_0}{N}} = |f_1|e^{-j\left(\phi_1+2\pi\frac{k_0}{N}\right)}\,.$$

Thus the angular phase we would extract as the polar angle from f_1' is given by

$$\phi_1' = \phi_1 + 2\pi\frac{k_0}{N}\,.$$

The normalized Fourier coefficients we compute for this shifted k origin path u_k' using Equation 198 again

$$\hat{f}_l' = f_l' \exp(jl\phi_1')\,.$$

Again using Equation 196 to evaluate f_l' and replacing ϕ_1' by what we found above we get

$$\hat{f}_l' = f_l e^{-j\frac{2\pi}{N}lk_0} \exp\left(jl(\phi_1 + 2\pi\frac{k_0}{N})\right) = f_l \exp(jl\phi_1) = \hat{f}_l\,,$$

showing that the normalized Fourier coefficients are indeed invariant with respect to where we start sampling u_k in k.

Problem Solutions

Problem 7.1 (the ASM and the CON features)

This problem is worked in the MATLAB script `prob_7_1.m`.

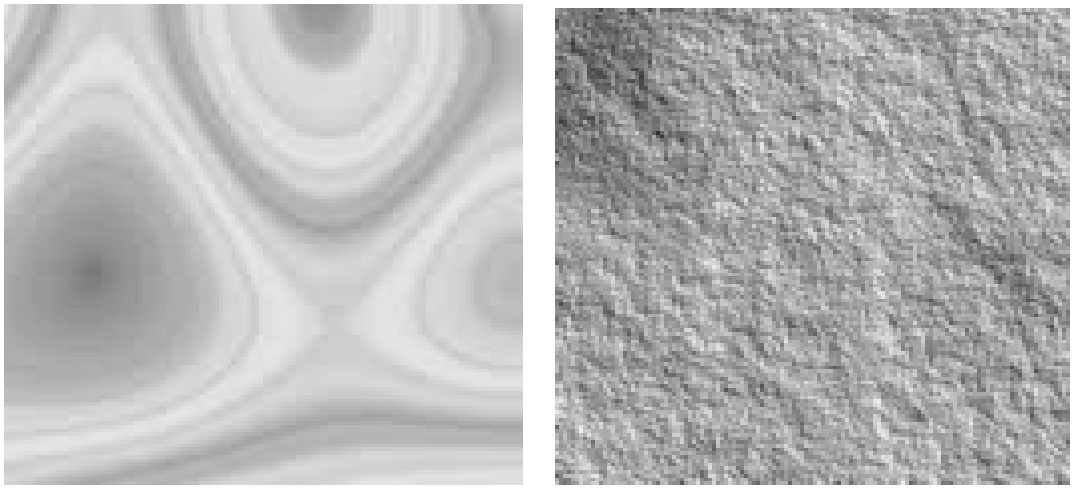

Figure 18: **Left:** A plot of the books "image1.png". **Right:** A plot of the books "image3.png".

Problem 7.2 (the run-length matrices)

This problem is worked in the MATLAB script `prob_7_2.m`.

Problem 7.3 (image contrast)

Consider the image

$$I = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Then notice that this image alternates between 0 and 1 when we look along the $\phi = 0$ direction but has the same value when we look along the $\phi = 45$ direction. Our routines give $CON(I, 1, 0) = 1$ while $CON(I, 1, 45) = 0$. This problem is worked in the MATLAB script `prob_7_3.m`.

Problem 7.4 (feature extraction on various test images)

For this problem we use the MATLAB command `imread` to load in two test images from the book. The `imread` command creates a "matrix" with quantized gray levels that can be processed by the routines developed on Page 123. The two test images selected (and converted to postscript for display in this document) are shown in Figure 18. This problem is worked in the MATLAB script `prob_7_4.m`. When that script is run we get the outputs given in Table 1.

Feature	Image #1	Image #3
ASM	0.0023	$1.69\,10^{-4}$
CON	25.6573	725.08
IDF	0.332	0.04
H_{xy}	9.6371	12.86
SRE	0.8878	0.9890
LRE	2.1792	1.0452
GLNU	269.6925	176.5214
RLN	$1.0051\,10^4$	$1.5680\,10^4$

Table 1: Extracted features for the two images shown in Figure 18. Note that the Image #1 has larger values for the ASM, IDF, LRE features while Image #3 has larger values for the CON, H_{xy}, and SRE features. From the discussion in the book this means that Image #1 is smoother, has less contrast, and more long runs, while Image #3 has more contrast, has more disorder, and has more short runs. All of these properties can be heuristically verified as true by looking at the two images given in Figure 18.

Problem 7.5 (a constrained minimization problem)

This is a constrained optimization problem so to solve it we will use the method of Lagrange multipliers. Technically this procedure is for finding *unbounded* optimal. Since in addition to the summation constraint $\sum_{i=1}^N P_i = 1$ we have the constraints $0 \leq P_i \leq 1$ we need to make sure that the solution to the unconstrained problem also satisfies these constraints. To use this method we first form the Lagrangian

$$\mathcal{L}((P_1, P_2, \cdots, P_N); \lambda) \equiv \sum_{i=1}^N P_i^2 - \lambda \left(\sum_{i=1}^N P_i - 1 \right) ,$$

and then look for stationary points with respect to $\mathbf{P} \equiv (P_1, P_2, \cdots, P_N)$ and λ. We have

$$\frac{\partial \mathcal{L}}{\partial P_i} = 0 \Rightarrow 2P_i - \lambda = 0 \tag{199}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \Rightarrow -\sum_{i=1}^N P_i + 1 = 0 \,. \tag{200}$$

From Equation 199 we have $P_i = \frac{\lambda}{2}$, which when we put these into Equation 200 we get

$$-\frac{\lambda}{2} N + 1 = 0 \,.$$

Thus $\lambda = \frac{2}{N}$. When we put this back into Equation 199 we have

$$P_i = \frac{\lambda}{2} = \frac{1}{N} \,,$$

as we were to show. Note that these solutions also satisfy $0 \leq P_i \leq 1$ as required.

Problem 7.6 (moments translational and scaling invariance)

We will consider the translated image I' defined by $I'(x,y) = I(x-a, y-b)$. Then

$$\mu'_{00} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I'(x,y)dxdy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x-a,y-b)dxdy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(u,v)dudv = \mu_{00}\,,$$

when we make the substitution $u = x-a$ and $v = y-b$. The same expression holds for m_{00}. That is $m'_{00} = m_{00} = \mu_{00} = \mu'_{00}$. Now for m'_{10} we have

$$m'_{10} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xI(x-a,y-b)dxdy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (u+a)I(u,v)dudv = m_{10} + am_{00}\,.$$

In the same way we find

$$m'_{01} = m_{01} + bm_{00}\,.$$

Thus the new means $\bar{x}'$ and $\bar{y}'$ are given by

$$\begin{aligned}\bar{x}' &= \frac{m'_{10}}{m'_{00}} = \frac{m_{10}}{m_{00}} + a = \bar{x} + a \\ \bar{y}' &= \frac{m'_{01}}{m'_{00}} = \frac{m_{01}}{m_{00}} + b = \bar{y} + b\,.\end{aligned}$$

We find for μ'_{pq}

$$\begin{aligned}\mu'_{pq} &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-\bar{x})^p (y-\bar{y})^q I'(x,y)dxdy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-\bar{x}-a)^p (y-\bar{y}-b)^q I(x-a,y-b)dxdy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (u-\bar{x})^p (v-\bar{y})^q I(u,v)dudv = \mu_{pq}\,.\end{aligned}$$

Showing that the central moments are invariant to translations. From all of this we also see that

$$\eta'_{pq} = \frac{\mu'_{pq}}{(\mu'_{00})^\gamma} = \frac{\mu_{pq}}{\mu_{00}^\gamma}\,,$$

showing that the normalized central moments are invariant to translations as we were to show.

Now consider a scaled image I' defined as $I'(x,y) = I(\alpha x, \alpha y)$. Then

$$\mu'_{00} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I'(x,y)dxdy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(\alpha x,\alpha y)dxdy = \frac{1}{\alpha^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(u,v)dudv = \frac{1}{\alpha^2}\mu_{00}\,,$$

when we make the substitution $u = \alpha x$ and $v = \alpha y$. This also equals m'_{00} and m_{00}. Now for m'_{10} we have

$$m'_{10} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xI(\alpha x,\alpha y)dxdy = \frac{1}{\alpha^3}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} uI(u,v)dudv = \frac{1}{\alpha^3}m_{10}\,.$$

In the same way we have

$$m'_{01} = \frac{1}{\alpha^3} m_{01} \,.$$

Thus the new means $\bar{x}'$ and $\bar{y}'$ are given by

$$\bar{x}' = \frac{m'_{10}}{m'_{00}} = \frac{1}{\alpha}\frac{m_{10}}{m_{00}} = \frac{1}{\alpha}\bar{x}$$
$$\bar{y}' = \frac{m'_{01}}{m'_{00}} = \frac{1}{\alpha}\bar{y} \,.$$

Using these we find for μ'_{pq}

$$\begin{aligned}\mu'_{pq} &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(\alpha x, \alpha y)(x-\bar{x}')^p (y-\bar{y}')^q dxdy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(\alpha x, \alpha y)\left(x - \frac{1}{\alpha}\bar{x}\right)^p \left(y - \frac{1}{\alpha}\bar{y}\right)^q dxdy \,.\end{aligned}$$

Let $u = \alpha x$ and $v = \alpha y$ to get

$$\begin{aligned}\mu'_{pq} &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(u,v)\left(\frac{u}{\alpha} - \frac{\bar{x}}{\alpha}\right)^p \left(\frac{v}{\alpha} - \frac{\bar{y}}{\alpha}\right)^q \frac{du}{\alpha}\frac{dv}{\alpha} \\ &= \frac{1}{\alpha^{p+q+2}}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(u,v)(u-\bar{x})^p (v-\bar{y})^q dudv = \frac{1}{\alpha^{p+q+2}}\mu_{pq} \,.\end{aligned}$$

Thus the central moments μ'_{pq} are not invariant to scaling but if we consider the normalized central moments η'_{pq} we see that

$$\eta'_{pq} = \frac{\mu'_{pq}}{(\mu'_{00})^\gamma} = \frac{\alpha^{2\gamma}}{\alpha^{p+q+2}}\frac{\mu_{pq}}{\mu_{00}^\gamma} \,.$$

Now from the definition of γ we have $2\gamma = p+q+2$ thus $\eta'_{pq} = \eta_{pq}$ as we were to show.

Problem 7.8 (rotational invariants)

If our image is rotated by and angle θ_0 then when we express the image in terms of polar coordinates we have that the new image $I'(\rho,\theta)$ is given in terms of the old image $I(\rho,\theta)$ by

$$I'(\rho,\theta) = I(\rho, \theta - \theta_0) \,.$$

Then the Zernike moments of the rotated image are given by

$$\begin{aligned}A'_{pq} &= \frac{p+1}{\pi}\int\int_{x^2+y^2\le 1} I'(x,y)V^*(\rho,\theta)\rho d\rho d\theta \\ &= \frac{p+1}{\pi}\int\int_{x^2+y^2\le 1} I(\rho,\theta-\theta_0)V^*(\rho,\theta)\rho d\rho d\theta \,.\end{aligned}$$

In this integral let $\phi = \theta - \theta_0$ so that $d\phi = d\theta$ and we get

$$A'_{pq} = \frac{p+1}{\pi}\int\int_{x^2+y^2\le 1} I(\rho,\phi)V^*(\rho,\phi+\theta_0)\rho d\rho d\phi$$

Since

$$V_{pq}^*(\rho, \phi + \theta_0) = R_{pq}(\rho)e^{-jq(\phi+\theta_0)} = V_{pq}^*(\rho, \phi)e^{-jq\theta_0}\,,$$

the above expression for A'_{pq} becomes

$$A'_{pq} = \left(\frac{p+1}{\pi}\int\int_{x^2+y^2\leq 1} I(\rho,\phi)V^*(\rho,\phi)\rho d\rho d\phi\right)e^{-jq\theta_0} = A_{pq}e^{-jq\theta_0}\,.$$

In deriving this result we have needed the fact that both $I(\rho, \phi)$ and $V_{pq}^*(\rho, \phi)$ are periodic in ϕ with period 2π.

Problem 7.9 (computing the moments of Hu)

See the notes on Page 125 for a discussion on this.

Problem 7.10 (computing the Zernike moments)

See the MATLAB script `dup_Zernike_moments.m` which uses the functions `Zernike_moments.m` and `Zernike_polynomial.m` to compute the Zernike moments for several values of p and q for the petasti images. When that code is run (after some time) we find the absolute values of the $A_{1,1}$, $A_{2,-2}$, $A_{2,0}$ and $A_{2,2}$ moments given by

	A	B	C	D	E	F
A_{1,1}	0.0016	0.0056	0.0016	0.0016	0.0016	0.3993
A_{2,-1}	0.0008	0.0029	0.0008	0.0008	0.0008	0.3204
A_{2,0}	0.0098	0.0351	0.0098	0.0098	0.0098	2.9708
A_{2,2}	0.0008	0.0029	0.0008	0.0008	0.0008	0.3204

all multiplied by 10^6. From these numbers we can see the invariance of the Zernike moments with rotation.

Problem 7.11 (the variance of the error or σ_η^2)

We can compute the desired variance σ_η^2 as the expectation of

$$\left(I(m,n) - \sum_{k,l} a(k,l)I(m-k,n-l)\right)^2 = I(m,n)^2 - 2I(m,n)\sum_{k,l} a(k,l)I(m-k,n-l) + \sum_{k_1,l_1,k_2,l_2} a(k_1,l_1)a(k_2,l_2)I(m-k_1,n-l_1)I(m-k_2,n-l_2)\,.$$

Taking the expectation of the right-hand-side of the above expression gives

$$\sigma_\eta^2 = r(0,0) - 2\sum_{k,l} a(k,l)r(k,l) + \sum_{k_1,l_1}\sum_{k_2,l_2} a(k_1,l_1)a(k_2,l_2)r(k_2-k_1,l_2-l_1)\,. \tag{201}$$

The equation used to solve for $a(k,l)$ requires that

$$\sum_{k_2,l_2} a(k_2,l_2)r(k_2-k_1,l_2-l_1) = \sum_{k_2,l_2} r(k_1-k_2,l_1-l_2) = r(k_1,l_1)\,. \tag{202}$$

Using this expression in the (k_2,l_2) summation in Equation 201 gives

$$\begin{aligned}\sigma_\eta^2 &= r(0,0) - 2\sum_{k,l} a(k,l)r(k,l) + \sum_{k_1,l_1} a(k_1,l_1)r(k_1,l_1) \\ &= r(0,0) - \sum_{k,l} a(k,l)r(k,l)\,,\end{aligned}$$

as we were to show.

Problem 7.13 (a rotation on u_k)

To compute the new point (x',y') when a point (x,y) is rotated counterclockwise by an angle θ we compute

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}\,.$$

Thus the new complex point on the boundary u_k' is given by

$$\begin{aligned}u_k' &= x_k' + jy_k' = (\cos(\theta)x_k - \sin(\theta)y_k) + j(\sin(\theta)x_k + \cos(\theta)y_k) \\ &= (\cos(\theta) + j\sin(\theta))x_k + (-\sin(\theta) + j\cos(\theta))y_k \\ &= e^{j\theta}x_k + j(\cos(\theta) + j\sin(\theta))y_k \\ &= e^{j\theta}(x_k + jy_k) = e^{j\theta}u_k\,,\end{aligned} \tag{203}$$

as we were to show.

Problem 7.14 (computing the Fourier series coefficients a_n, b_n, c_n and d_n)

This problem is discussed motivated in more detail in the reference [7]. For the given Fourier expansions of $x(t)$ we can compute the coefficients a_0, a_n, and b_n by integration

$$\begin{aligned}a_0 &= \frac{1}{T}\int_0^T x(t)dt \\ a_n &= \frac{2}{T}\int_0^T x(t)\cos\left(\frac{2n\pi t}{T}\right)dt \\ b_n &= \frac{2}{T}\int_0^T x(t)\sin\left(\frac{2n\pi t}{T}\right)dt\,.\end{aligned}$$

The same type of expressions hold for the coefficients in the Fourier expansion of $y(t)$. Since we assume that $x(t)$ is piecewise linear it might be easier to consider the t derivative of $x(t)$ We can express $\dot{x}(t)$ in a different Fourier series as

$$\dot{x}(t) = \sum_{n=1}^{\infty} \alpha_n \cos\left(\frac{2n\pi t}{T}\right) + \beta_n \sin\left(\frac{2n\pi t}{T}\right) .$$

The Fourier coefficients α_n and β_n of $\dot{x}(t)$ are given by the same type of expression used to compute a_n and b_n. Namely for α_n we have

$$\alpha_n = \frac{2}{T}\int_0^T \dot{x}(t)\cos\left(\frac{2n\pi t}{T}\right) dt .$$

But since the the boundary curve in $\dot{x}(t)$ is constant over the range of times $t_{i-1} < t < t_i$ we can evaluate this integral by summing several smaller integrals over these segments as

$$\alpha_n = \frac{2}{T}\sum_{i=1}^{m}\int_{t_{i-1}}^{t_i} \dot{x}(t)\cos\left(\frac{2n\pi t}{T}\right) dt .$$

Now on the range $t_{i-1} < t < t_i$ we can introduce $\Delta x_i = x_i - x_{i-1}$ and $\Delta t_i = t_i - t_{i-1}$ and then take $\dot{x} \approx \frac{\Delta x_i}{\Delta t_i}$, which is independent of t and the above integral becomes

$$\begin{aligned}\alpha_n &= \frac{2}{T}\sum_{i=1}^{m}\frac{\Delta x_i}{\Delta t_i}\int_{t_{i-1}}^{t_i}\cos\left(\frac{2n\pi t}{T}\right) dt = \frac{2}{T}\sum_{i=1}^{m}\frac{\Delta x_i}{\Delta t_i}\left(\frac{T}{2\pi n}\sin\left(\frac{2n\pi t}{T}\right)\Big|_{t_{i-1}}^{t_i}\right. \\ &= \frac{1}{\pi n}\sum_{i=1}^{m}\frac{\Delta x_i}{\Delta t_i}\left(\sin(\phi_i) - \sin(\phi_{i-1})\right) ,\end{aligned}$$

where we have defined $\phi_i \equiv \frac{2\pi n t_i}{T}$. The coefficient β_n is defined in the same way and is given by

$$\beta_n = -\frac{1}{n\pi}\sum_{i=1}^{m}\frac{\Delta x_i}{\Delta t_i}\left(\cos(\phi_i) - \cos(\phi_{i-1})\right) .$$

Given the original Fourier expansion of $x(t)$ we can compute the time derivative explicitly where we get

$$\dot{x}(t) = \frac{2\pi}{T}\sum_{n=1}^{\infty} n b_n \cos\left(\frac{2n\pi t}{T}\right) - n a_n \sin\left(\frac{2n\pi t}{T}\right) .$$

Equating the coefficients of $\cos\left(\frac{2n\pi t}{T}\right)$ and $\sin\left(\frac{2n\pi t}{T}\right)$ with the definitions of α_n and β_n we get

$$\begin{aligned}\frac{2\pi}{T} n b_n = \alpha_n &= \frac{1}{\pi n}\sum_{i=1}^{m}\frac{\Delta x_i}{\Delta t_i}\left(\sin(\phi_i) - \sin(\phi_{i-1})\right) \\ -\frac{2\pi}{T} n a_n = \beta_n &= -\frac{1}{n\pi}\sum_{i=1}^{m}\frac{\Delta x_i}{\Delta t_i}\left(\cos(\phi_i) - \cos(\phi_{i-1})\right) .\end{aligned}$$

Thus solving for a_n and b_n we get

$$\begin{aligned}a_n &= \frac{T}{2\pi^2 n^2}\sum_{i=1}^{m}\frac{\Delta x_i}{\Delta t_i}\left(\cos(\phi_i) - \cos(\phi_{i-1})\right) \\ b_n &= \frac{T}{2\pi^2 n^2}\sum_{i=1}^{m}\frac{\Delta x_i}{\Delta t_i}\left(\sin(\phi_i) - \sin(\phi_{i-1})\right) .\end{aligned}$$

Expressions for c_n and d_n for $n \geq 1$ can be obtained in the same way but working with the function $y(t)$.

Problem 7.15 (invariants of the Fourier coefficients a_n, b_n, c_n, and d_n)

If we consider a rotation of the parametrically described curve $z(t)$ then the transformed curve $z'(t) = z(t)e^{j\theta}$ and the real and imaginary components of $z'(t)$ are given in terms of the real and imaginary components of $z(t)$ by

$$\begin{aligned} x_i' &= \cos(\theta)x_i - \sin(\theta)y_i \\ y_i' &= \sin(\theta)x_i + \cos(\theta)y_i \,. \end{aligned}$$

See Equation 203. Therefore the discrete changes in x' and y' are given by

$$\begin{aligned} \Delta x_i' &= \cos(\theta)\Delta x_i - \sin(\theta)\Delta y_i \\ \Delta y_i' &= \sin(\theta)\Delta x_i + \cos(\theta)\Delta y_i \,. \end{aligned}$$

From the forms of a_n, b_n, c_n and d_n and how the depend on Δx_i and Δy_i given in Problem 7.14 we have that the Fourier coefficients a_n transforms as

$$\begin{aligned} a_n' &= \frac{T}{2\pi^2 n^2} \sum_{i=1}^{m} \frac{\Delta x_i'}{\Delta t_i} \left(\cos\left(\phi_i\right) - \cos(\phi_{i-1})\right) \\ &= \cos(\theta)a_n - \sin(\theta) \left(\frac{T}{2\pi^2 n^2} \sum_{i=1}^{m} \frac{\Delta y_i}{\Delta t_i} (\cos(\phi_i) - \cos(\phi_{i-1})) \right) \\ &= \cos(\theta)a_n - \sin(\theta)c_n \,. \end{aligned}$$

In the same way we find that b_n, c_n, and d_n transform as follows

$$\begin{aligned} b_n' &= \cos(\theta)b_n - \sin(\theta)d_n \\ c_n' &= \sin(\theta)a_n + \cos(\theta)c_n \\ d_n' &= \sin(\theta)b_n + \cos(\theta)d_n \,. \end{aligned}$$

Using these expressions we can evaluate I_n, J_n, and $K_{1,n}$ in the rotated frame. Using the MATHEMATICA file `chap_7_prob_15.nb` we simplify each expression using the above relationships for a_n', b_n' etc and show that they equal the original definitions in the original frame.

Problem 7.16 (more Fourier invariants)

If our original boundary is given in parametric form as $z(t) = x(t) + jy(t)$ then a rotation counter-clockwise by the angle θ produces the boundary $z'(t)$ given by $z'(t) = z(t)e^{j\theta}$. This

causes the complex Fourier coefficients a_n to transform as $a'_n = a_n e^{j\theta}$. Then the suggested features b_n and d_{mn} transform as

$$b'_n = \frac{a'_{1+n}a'_{1-n}}{a'^2_1} = \frac{a_{1+n}a_{1-n}e^{2j\theta}}{a_1^2 e^{2j\theta}} = \frac{a_{1+n}a_{1-n}}{a_1^2} = b_n$$
$$d'_{mn} = \frac{a'_{1+m}{}^n a'_{1-n}{}^m}{a'_1{}^{m+n}} = \frac{a_{1+m}{}^n a_{1-n}{}^m e^{j(n+m)\theta}}{a_1{}^{m+n} e^{j(n+m)\theta}} = d_{mn}\,,$$

showing the invariance. Multiplying our boundary by a scalar say α produces the new curve $z'(t) = \alpha z(t)$ which gives new Fourier coefficients of $a'_n = \alpha a_n$. The proof above again shows scale invariance.

Problem 7.18 (derivation of the orientation angle θ)

Expand the quadratic in the expression given for $I(\theta)$ to get

$$\begin{aligned} I(\theta) &= \sum_{i,j} \left[(i-\bar{x})^2\cos(\theta)^2 - 2(i-\bar{x})(j-\bar{y})\sin(\theta)\cos(\theta) + (j-\bar{y})^2\sin(\theta)^2\right] \\ &= \mu_{20}\cos(\theta)^2 - 2\mu_{11}\sin(\theta)\cos(\theta) + \mu_{02}\sin(\theta)^2\,. \end{aligned}$$

The derivative of this expression with respect to θ is given by

$$\begin{aligned} I'(\theta) &= -2\mu_{20}\cos(\theta)\sin(\theta) - 2\mu_{11}\cos(\theta)^2 + 2\mu_{11}\sin(\theta)^2 + 2\mu_{02}\sin(\theta)\cos(\theta) \\ &= 2(\mu_{02}-\mu_{20})\cos(\theta)\sin(\theta) - 2\mu_{11}(\cos(\theta)^2 - \sin(\theta)^2) \\ &= (\mu_{02}-\mu_{20})\sin(2\theta) - 2\mu_{11}\cos(2\theta)\,. \end{aligned}$$

When we set this equal to zero and then solve for θ we get

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{2\mu_{11}}{\mu_{02}-\mu_{20}}\right)\,.$$

For some reason the above representation of θ has the expression $\mu_{02} - \mu_{20}$ rather than the desired $\mu_{20} - \mu_{02}$ (i.e. there is a negative sign difference) if any one sees an error in what I have done above please contact me.

Template Matching

Problem Solutions

Problem 8.1 (the edit distance)

The MATLAB function `edit_distance.m` computes the edit distance between two words. The call `edit_distance('poem','poten')` gives the "distance matrix" (which represents the optimal path one would take)

```
0     1     2     3     4     5
1     0     1     2     3     4
2     1     0     1     2     3
3     2     1     1     1     2
4     3     2     2     2     2
```

The value of the corner element (here the value in the 5th row and 6th column) is the edit distance. From the above we see that it is 2 representing the deletion of the character "t" and the change of the "n" to an "m".

Problem 8.2 (slopes of the Sakoe-Chiba constraints)

In the Sakoe-Chiba constraints part a we can take the transition from $(i-1, j)$ to (i, j) giving a slope of

$$\frac{i-(i-1)}{j-j} = \infty \,.$$

In the Sakoe-Chiba constraints part b we can take the transition from $(i-2, j-1)$ to (i, j) giving a slope of

$$\frac{i-(i-2)}{j-(j-1)} = 2 \,.$$

In the Sakoe-Chiba constraints part c we can take the transition from $(i-3, j-1)$ to (i, j) giving a slope of

$$\frac{i-(i-3)}{j-(j-1)} = 3 \,.$$

In the Sakoe-Chiba constraints part d we can take the transition from $(i-3, j-2)$ to (i, j) giving a slope of

$$\frac{i-(i-3)}{j-(j-2)} = \frac{3}{2} \,.$$

Problem 8.4 (a measure of image similarity)

The expression given represents the cross-correlation coefficient between the seven Hu moments of the test and the reference image. Since the Hu moments have great number of invariants this measure of similarity might be more robust than using a direct correlation metric.

Problem 8.5 (the Mellin transform)

Consider the scaled function f' defined by $f'(x,y) = f(\alpha x, \alpha y)$, where $\alpha > 0$ is a real constant. Then the Mellin transform for this function is given by

$$M'(u,v) = \iint f(\alpha x, \alpha y) x^{-ju-1} y^{-jv-1} dx dy \,.$$

Let $x' = \alpha x$ and $y' = \alpha y$ in the above to get

$$\begin{aligned} M'(u,v) &= \iint f(x',y') \left(\frac{x'}{\alpha}\right)^{-ju-1} \left(\frac{y'}{\alpha}\right)^{-jv-1} \frac{dx'}{\alpha}\frac{dy'}{\alpha} \\ &= \frac{1}{\alpha^{-ju-1}\alpha^{-jv-1}\alpha^2} \iint f(x',y') x'^{-ju-1} y'^{-jv-1} dx' dy' \\ &= \frac{1}{\alpha^{-ju}\alpha^{-jv}} M(u,v) \,, \end{aligned}$$

since α, and u, v are a real numbers we have $|\alpha^{-ju}| = 1$ and $|\alpha^{-jv}| = 1$ showing that the *magnitude* of the Mellin transform is invariant to scaling.

Problem 8.6 (computational resources for correlation based matching)

In the motion compensation step we assume we have the original frame taken at the "time" t and of size $I \times J$ that we break up into subblocks of a smaller size say $M \times N$. The camera or visual recording device then records another frame at the time $t+1$ of the same size $M \times N$. To develop a mapping of the frame at time t to the new frame at time $t+1$ each of the smaller $M \times N$ subblocks in the first frame must be searched for in the second frame. Since there are on order of

$$\frac{IJ}{MN} \,,$$

smaller subblocks in the larger $I \times J$ frame we have to do this many searches for optimal cross-correlation points. For the numbers given in this problem $M = N = 16$, $I = 720$, $J = 480$, and $f = 30$ frames per second this number becomes 1350.

In the case where we are doing the full cross-correlation computation on every pixel (m,n) in the test image we must compute

$$c(m,n) = \sum_{i=m}^{m+M-1} \sum_{j=n}^{n+N-1} t(i,j) r(i-m, j-n) \,, \tag{204}$$

at each of the $(2p+1)^2$ pixels in the test image. Each of the sums above requires $O(MN)$ multiplications and additions giving a total computational cost of

$$(2p+1)^2 NM\,.$$

for the "full search" technique. Thus the computational resources for the full search technique is

$$\frac{IJ}{MN}(2p+1)^2 MNf = IJ(2p+1)^2 f = 9.96\,10^9\,,$$

flops per second.

If we consider a two-dimensional logarithmic search we start with a test box of dimension $[-p,+p]\times[-p,+p]$ and in this box compute the cross-correlation at 8 points (and the center) spaced at a distance of $\frac{p}{2}$ around this center. This means that we compute the cross-correlation at the points

$$(0,0)\,,\left(\frac{p}{2},\frac{p}{2}\right)\,,\left(\frac{p}{2},0\right)\,,\left(\frac{p}{2},-\frac{p}{2}\right)\,,\left(0,-\frac{p}{2}\right)\,,\left(-\frac{p}{2},-\frac{p}{2}\right)\,,\left(-\frac{p}{2},0\right)\,,\left(-\frac{p}{2},\frac{p}{2}\right)\,,\left(0,\frac{p}{2}\right)\,.$$

From the cross-correlation computed at these points we find the point with the largest cross-correlation. We now impose a box of size $\left[-\frac{p}{2},\frac{p}{2}\right]\times\left[-\frac{p}{2},\frac{p}{2}\right]$ on the maximal point and search the 8 points on the perimeter of a box with edge spaced $\frac{p}{4}$ from this new center. We will specify a new center to search about $k=\lceil\log_2(p)\rceil$ times. Each new center requires 8 cross-correlation searches thus including the cross-correlation taken at the first center we have $8k+1$ cross-correlations. Since each of these cross-correlations takes MN calculations the total operations required using this method is

$$\frac{IJ}{MN}(8k+1)MNf = IJ(8k+1)f = 0.342\,10^9\,,$$

flops per second.

The linear variant of the two-dimensional logarithmic search searches the top and bottom locations but not the diagonal locations searched in the full two-dimensional search. We search $\left(0,\frac{p}{2}\right)$ and $\left(0,-\frac{p}{2}\right)$ for the vertical search and $\left(-\frac{p}{2},0\right)$ and $\left(\frac{p}{2},0\right)$ for the horizontal search. This gives four cross-correlation calculations for every central point and again we have $k=\lceil\log_2(p)\rceil$ total computations. Since each of these cross-correlations takes MN calculations the total operations required using this method is

$$\frac{IJ}{MN}(4k+1)MNf = IJ(8k+1)f = 0.176\,10^9\,,$$

flops per second.

Context-Dependent Classification

Notes on the text

Notes on Channel Equalization

The total number of clusters will be equal to the number of possible vectors of the form

$$\mathbf{x}_k^T = \begin{bmatrix} x_k & x_{k-1} & x_{k-2} & \cdots & x_{k-l+2} & x_{k-l+1} \end{bmatrix},$$

where there are l samples of the noisy x_k. The last samples is x_{k-l+1} and from the model of the noisy transmission channel is given in terms of the binary inputs I_k as

$$x_k = f(I_k, I_{k-1}, \cdots, I_{k-n+2}, I_{k-n+1}),$$

will need a total of n samples in the past of I_k to determine the value of x_{k-l+1}. Thus to know the value of $\mathbf{x}_k$ defined above we need

$$I_k, I_{k-1}, I_{k-2}, \cdots, I_{k-l+2}, I_{k-l+1}, I_{k-l}, \cdots, I_{k-l-n+3}, I_{k-l-n+2},$$

or $k-(k-l-n+2)+1 = l+n-1$ samples of I_k starting from I_k and working backwards to $I_{k-l-n+2}$. If the value of I_k are binary valued then we have a total of 2^{l+n-1} distinct inputs to produce 2^{l+n-1} distinct vectors $\mathbf{x}$ (this ignores the value of the noise term η_t which would only spread the samples $\mathbf{x}_k$ about the cluster centers).

From this discussion we can construct Table 9.1 by forming all possible sequences of three bit patterns for the inputs I_k that we would need to know to determine the lagged vector $\mathbf{x}_k$ having two elements i.e. $\mathbf{x}_k^T = \begin{bmatrix} x_k & x_{k-1} \end{bmatrix}$. That is we can have (I_k, I_{k-1}, I_{k-2}) be $(0,0,0)$, $(0,0,1)$, $(0,1,0)$, $(1,0,0)$ etc. and the observed outputs of x_k and x_{k-1} are computed from the model (ignoring noise)

$$x_k = 0.5I_k + I_{k-1}.$$

Notes on Hidden Markov Models

From the initial definition of $\alpha(i_{k+1})$ we can express it as

$$\begin{aligned} \alpha(i_{k+1}) &= p(x_1, \ldots, x_{k+1}, i_{k+1}|\mathcal{S}) && (205) \\ &= \sum_{i_k} \alpha(i_k) P(i_{k+1}|i_k) p(x_{k+1}|i_{k+1}), && (206) \end{aligned}$$

by using the rule of total probability when we sum over all ways in which we can get into the state i_{k+1} from states $i_k \in \{1, 2, \cdots, M\}$.

Problem Solutions

Problem 9.2 (self-transition probabilities)

To be in state i for d successful strategies means that we take $d-1$ transitions that result in no movement of our underlying state. That is $d-1$ times we stay put (this happens with probability $P(i|i)$) and on the dth transition we move somewhere else (which happens with probability $1-P(i|i)$). Thus the probability we stay in state i for d successive stages is

$$P(i|i)^{d-1}(1-P(i|i))\,.$$

Note that this expression works for $d=1$ since the start state is i. This is called a geometric distribution [10] and has an expected number of transitions needed to leave the state i given by

$$\frac{1}{1-P(i|i)}\,.$$

If $P(i|i) \approx 1$ then $\bar{d}$ can be very large resulting in many self transitions.

Chapter 10 (System Evaluation)

Notes on the text

Notes on the error counting approach

When, for a given classifier, our empirical error rate for the ith class $1 \leq i \leq M$ is given by the frequency count $\hat{P}_i = \frac{k_i}{N_i}$. With this, an estimate for the total error rate $\hat{P}$ of this classifier is given by

$$\hat{P} = \sum_{i=1}^{M} P(\omega_i)\hat{P}_i \,.$$

Since the random variables k_i (the number of misclassifications in class i) is a binomial random variable with parameters (P_i, N_i) we have that the variance in k_i given by the standard formula

$$\sigma_{k_i}^2 = N_i P_i (1 - P_i) \,. \tag{207}$$

Here P_i is the true error rate in the ith class. We can estimate the variance in our estimate of the total error probability $\hat{P}$ by assuming that the k_i random variables are independent. We then have that

$$\sigma_{\hat{P}}^2 = \sum_{i=1}^{M} \frac{P(\omega_i)^2}{N_i^2}\sigma_{k_i}^2 = \sum_{i=1}^{M} P(\omega_i)^2 \frac{P_i(1-P_i)}{N_i} \,, \tag{208}$$

when we put in Equation 207. **Warning:** I'm not sure I follow the argument that the k_i random variables are independent. I would assume that the error made in classifying class i would influence whether or not one made an error in class j for $j \neq i$, even if the feature vectors $\mathbf{x}$ are independent. If anyone has a better argument or explanation please contact me.

Problem Solutions

Problem 10.1 (the expectation and variance of a binomial RV)

For a binomial random variable (RV) with probability of success p and defining $q = 1 - p$ we find its expectation given by $E(X)$

$$\begin{aligned}
E(K) &= \sum_{k=0}^{N} k \left(\begin{array}{c} N \\ k \end{array} \right) p^k q^{N-k} \\
&= \sum_{k=1}^{N} k \frac{n!}{k!(n-k)!} p^k q^{N-k} = \sum_{k=1}^{N} \frac{n!}{(k-1)!(n-k)!} p^k q^{N-k} \\
&= n \sum_{k=1}^{N} \frac{(n-1)!}{(k-1)!((n-1)-(k-1))!} p^{k-1} q^{(n-1)-(k-1)} \\
&= np \sum_{k=1}^{N} \left(\begin{array}{c} n-1 \\ k-1 \end{array} \right) p^{k-1} q^{(n-1)-(k-1)} \\
&= np \sum_{k=0}^{N-1} \left(\begin{array}{c} n-1 \\ k \end{array} \right) p^{k} q^{(n-1)-k} \\
&= np \cdot 1 = np \,.
\end{aligned}$$

Next we need to evaluate $E(K^2)$. We find

$$\begin{aligned}
E(K^2) &= \sum_{k=0}^{N} k^2 \left(\begin{array}{c} N \\ k \end{array} \right) p^k q^{N-k} \\
&= \sum_{k=1}^{N} k \frac{n(n-1)!}{(k-1)!(n-k)!} p^{k-1+1} q^{(n-1)-(k-1)} \\
&= np \sum_{k=1}^{N} (k-1+1) \left(\begin{array}{c} n-1 \\ k-1 \end{array} \right) p^{k-1} q^{(n-1)-(k-1)} \\
&= np \sum_{k=1}^{N} (k-1) \left(\begin{array}{c} n-1 \\ k-1 \end{array} \right) p^{k-1} q^{(n-1)-(k-1)} + np \sum_{k=1}^{N} \left(\begin{array}{c} n-1 \\ k-1 \end{array} \right) p^{k-1} q^{(n-1)-(k-1)} \\
&= np \sum_{k=2}^{N} (k-1) \left(\begin{array}{c} n-1 \\ k-1 \end{array} \right) p^{k-1} q^{(n-1)-(k-1)} + np \sum_{k=0}^{N-1} \left(\begin{array}{c} n-1 \\ k \end{array} \right) p^{k} q^{(n-1)-k} \\
&= np \sum_{k=2}^{N} \frac{(n-1)(n-2)!}{(k-2)!((n-1)-(k-1))!} p^{k-2+1} q^{(n-2)-(k-2)} + np \\
&= n(n-1)p^2 \sum_{k=2}^{N} \left(\begin{array}{c} n-2 \\ k-2 \end{array} \right) p^{k-2} q^{(n-2)-(k-2)} + np \\
&= n(n-1)p^2 \sum_{k=0}^{N-2} \left(\begin{array}{c} n-2 \\ k \end{array} \right) p^{k} q^{(n-2)-k} + np \\
&= n(n-1)p^2 + np \,.
\end{aligned}$$

Thus the variance of a binomial random variable is given by combining these two results as

$$\begin{aligned}
\mathrm{Var}(K) &= E(K^2) - E(K)^2 = n(n-1)p^2 + np - n^2 p^2 \\
&= np(1-p) = npq \,. \qquad (209)
\end{aligned}$$

Clustering: Basic Concepts

Notes on the text

Notes on similarity measures for real-valued vectors

The Tanimoto measure/distance is given by

$$s_T(x,y) = \frac{x^T y}{||x||^2 + ||y||^2 - x^T y}\,. \tag{210}$$

Since $(x-y)^T(x-y) = ||x||^2 + ||y||^2 - 2x^T y$ we have that $s_T(x,y)$ becomes

$$s_T(x,y) = \frac{x^T y}{(x-y)^T(x-y) + x^T y} = \frac{1}{1 + \frac{(x-y)^T(x-y)}{x^T y}}\,.$$

If $||x|| = ||y|| = a$, then we have

$$s_T(x,y) = \frac{x^T y}{2a^2 - x^T y} = \frac{1}{-1 + 2\frac{a^2}{x^T y}}\,.$$

Problem Solutions

Problem 11.1

If s is a similarity measure on X with $s(x,y) > 0$ for all $x, y \in X$ by defining $d(x,y) = \frac{a}{s(x,y)}$ with $a > 0$ we claim that $d(x,y)$ is a dissimilarity measure. To be a dissimilarity measure we need to satisfy several things. Note that since $s(x,y)$ is a positive similarity measure we have that $0 < s(x,y) \leq s_0$ and thus

$$\frac{a}{s_0} \leq \frac{a}{s(x,y)} < +\infty\,.$$

Thus $d(x,y)$ is bounded as $\frac{a}{s_0} \leq d(x,y) < +\infty$. Next for notational simplification lets define $d_0 \equiv \frac{a}{s_0}$. Note that $d(x,x)$ is given by

$$d(x,x) = \frac{a}{s(x,x)} = \frac{a}{s_0} = d_0\,.$$

Next the arguments of d are symmetric in that

$$d(x,y) = \frac{a}{s(x,y)} = \frac{a}{s(y,x)} = d(y,x)\,.$$

Thus we have shown that d is a dissimilarity measure (DM) on X. If we have $d(x,y) = d_0$ then this implies that $s(x,y) = s_0$ which happens if and only if $x = y$, since $s(x,y)$ is a metric similarity measure. Finally, by the property of $s(x,y)$ we have

$$s(x,y)s(y,z) \leq [s(x,y) + s(y,z)]s(x,z) \quad \forall x, y \in X\,. \tag{211}$$

We can write this in terms of $d(x,y)$ as

$$\frac{a}{d(x,y)}\,\frac{a}{d(y,z)} \leq \left[\frac{a}{d(x,y)} + \frac{a}{d(y,z)}\right]\frac{a}{d(x,z)}\,.$$

As $d(x,y) > 0$ this equals

$$d(x,z) \leq d(y,z) + d(x,y)\,,$$

or the final condition for a *metric* dissimilarity measure.

Problem 11.2

Note that if we take $p = 2$ in the Minkowski inequality we have

$$||x + y||_2 \leq ||x||_2 + ||y||_2\,. \tag{212}$$

To make this match the normal definition of the triangle inequality lets introduce three new vectors a, b, and c such that

$$\begin{aligned} x + y &= a - c \\ x &= a - b\,. \end{aligned}$$

These two equations require that y is given by

$$y = a - c - x = a - c - a + b = b - c\,.$$

Then using Equation 212 we get

$$||a - c||_2 \leq ||a - b||_2 + ||b - c||_2\,,$$

or

$$d(a,c) \leq d(a,b) + d(b,c)\,,$$

which is the triangle inequality.

Problem 11.4

Consider $d_2(x,y) = f(d(x,y))$ where $d(x,y)$ is a metric dissimilarity measure. Then

$$d_2(x,x) = f(d(x,x)) = f(d_0) \quad \forall x \in X\,,$$

and

$$d_2(x,y) = f(d(x,y)) = f(d(y,x)) = d_2(y,x) \quad \forall x, y \in X\,.$$

If we have $d_2(x,y) = f(d_0)$ or $f(d(x,y)) = f(d_0)$ then since f is monotonic we can invert the above equation to get $d(x,y) = d_0$. From the properties of d we know that this happens if and only if $x = y$.

Next since $f(\cdot)$ is increasing and $d(x,z) \leq d(x,y) + d(y,z)$ we have

$$f(d(x,z)) \leq f(d(x,y) + d(y,z)) \,.$$

Using the stated properties of f this expression on the right is bounded above by

$$f(d(x,y)) + f(d(y,z)) = d_2(x,y) + d_2(y,z) \,.$$

These show that $d_2(x,y)$ is a dissimilarity metric.

Problem 11.5

For this problem we will look at the various properties that a dissimilarity metric must satisfy and then show that $d(x,y) \equiv f(s(x,y))$ satisfies them, when f is a function that has the properties specified. To begin note that $d(x,x) = f(s(x,x)) = f(s_0)$ for all $x \in X$. Lets define $d_0 \equiv f(s_0)$ for notational simplicity. As a second property of d note that d is symmetric in its arguments since

$$d(x,y) = f(s(x,y)) = f(s(y,x)) = d(y,x) \,.$$

Now if $d(x,y) = d_0 = f(s_0)$ then since f is monotone and increasing we can invert this last equation to get $s(x,y) = s_0$ which imply that $x = y$. Next consider $d(x,y) + d(y,z)$ which from the assumed hypothesis is *greater* than

$$f\left(\frac{1}{\frac{1}{s(x,y)} + \frac{1}{s(y,z)}}\right) = f\left(\frac{s(x,y)s(y,z)}{s(x,y) + s(y,z)}\right) \,. \tag{213}$$

Since $s(x,y)$ is a similarity metric it must satisfy Equation 211 so that

$$\frac{s(x,y)s(y,z)}{s(x,y) + s(y,z)} \leq s(x,z) \,.$$

Since f is monotonically increasing the right-hand-side of Equation 213 is less than $f(s(x,z)) = d(x,z)$. Thus we have shown that $d(x,y) + d(y,z) \leq d(x,z)$ so d is a dissimilarity metric.

Problem 11.6

For this problem we want to show that

$$d_\infty(x,y) \leq d_2(x,y) \leq d_1(x,y) \,, \tag{214}$$

when all of the weights w_i in their definitions are equal to one. This is equivalent to showing

$$\max_{1\leq i\leq l}|x_i-y_i| \leq \left(\sum_{i=1}^{l}|x_i-y_i|^2\right)^{1/2} \leq \sum_{i=1}^{l}|x_i-y_i|\,.$$

To show this later expression consider the first inequality. In that expression let i^* be defined as

$$i^* = \operatorname{argmax}_{1\leq i\leq l}|x_i-y_i|\,.$$

Then we have

$$\begin{aligned}\left(\sum_{i=1}^{l}|x_i-y_i|^2\right)^{1/2} &= \left(|x_{i^*}-y_{i^*}|^2+\sum_{i=1;i\neq i^*}^{l}|x_i-y_i|^2\right)^{1/2}\\ &= |x_{i^*}-y_{i^*}|\left(1+\frac{1}{|x_{i^*}-y_{i^*}|}\sum_{i=1;i\neq i^*}^{l}|x_i-y_i|^2\right)^{1/2}.\end{aligned}$$

Since $\frac{1}{|x_{i^*}-y_{i^*}|}\sum_{i=1;i\neq i^*}^{l}|x_i-y_i|^2 > 0$ we see that the right-hand-side of the above equality is greater than or equal to $|x_{i^*}-y_{i^*}|$, showing

$$d_2(x,y) \geq d_\infty(x,y)\,.$$

Next consider $d_2^2(x,y)=\sum_{i=1}^{l}|x_i-y_i|^2$ in comparison to $d_1^2(x,y)$. This later expression is equal to

$$d_1^2(x,y)=\left(\sum_{i=1}^{l}|x_i-y_i|\right)^2=\sum_{i=1}^{l}|x_i-y_i|^2+2\sum_{i=1}^{l}\sum_{j=i+1}^{l}|x_i-y_i||x_j-y_j|\,.$$

Note that the right-hand-side of the above is larger than the sum

$$\sum_{i=1}^{l}|x_i-y_i|^2 = d_2^2(x,y)\,.$$

Thus

$$d_1^2(x,y)\geq d_2^2(x,y) \quad \text{or} \quad d_1(x,y)\geq d_2(x,y)\,,$$

showing the second half of the requested identity.

Problem 11.7

Part (a): That the maximum of $s_F^q(x,y)$ is $l^{1/q}$ can be seen since each term in its sum has the property $0\leq s(x_i,y_i)\leq 1$ and so

$$\sum_{i=1}^{l}s(x_i,y_i)^q \leq \sum_{i=1}^{l}1 = l\,.$$

Thus

$$s_F^q(x,y) \leq l^{1/q} \tag{215}$$

Question: How show that $s_F(x,y) \geq \frac{1}{2} l^{1/q}$?

Part (b): If we let $i^* = \text{argmax}_i s(x_i, y_i)$ then

$$\frac{s(x_i,y_i)}{s(x_{i^*},y_{i^*})} \leq 1 \quad \text{for} \quad 1 \leq i \leq l\,.$$

Then writing $s_F^q(x,y)$ as

$$s_F^q(x,y) = s(x_{i^*},y_{i^*}) \left(\sum_{i=1;i\neq i^*}^{l} \left(\frac{s(x_i,y_i)}{s(x_{i^*},y_{i^*})} \right)^q + 1 \right)^{1/q}.$$

Since $\lim_{q\to\infty} x^q = 0$ if $|x| < 1$ we have

$$\lim_{q\to\infty} s_F^q(x,y) = s(x_{i^*},y_{i^*}) = \max_{1\leq i\leq l} s(x_i,y_i)\,.$$

Problem 11.8

Question: How to show for these similarity functions that

$$s(x,y)s(y,z) \leq [s(x,y) + s(y,z)]s(x,z)$$

for all $x, y, z \in X$.

Problem 11.9

A proximity measure is a general notation for either a dissimilarity measure or a similarity measure. Consider the definition of $s_{\text{avg}}^{\text{ps}}(x,C)$ which is the point-set average similarly between the set C and the point x given by

$$s_{\text{avg}}^{\text{ps}}(x,C) = \frac{1}{n_C} \sum_{y\in C} s(x,y)\,.$$

Since $d_{\text{avg}}^{\text{ps}}(x,C)$ is equal to

$$d_{\text{avg}}^{\text{ps}}(x,C) = \frac{1}{n_C} \sum_{y\in C} d(x,y)\,,$$

and $d_{\max}$ can be written as

$$d_{\max} = \frac{1}{n_C} \sum_{y\in C} d_{\max}\,,$$

the difference between these two expressions gives

$$\begin{aligned} d_{\max} - d^{\text{ps}}_{\text{avg}}(x, C) &= \frac{1}{n_C}\sum_{y\in C} d_{\max} - \frac{1}{n_C}\sum_{y\in C} d(x,y) \\ &= \frac{1}{n_C}\sum_{y\in C}(d_{\max} - d(x,y)) \\ &= \frac{1}{n_C}\sum_{y\in C} s(x,y) \equiv s^{\text{ps}}_{\text{avg}}(x, C)\,. \end{aligned}$$

Problem 11.10

Recall that $d_{\text{Hamming}}(x, y)$ is equal to the number of places where the two vectors differ. Using the contingency table A we can write

$$d_{\text{Hamming}}(x, y) = \sum_{i=1}^{k-1}\sum_{j=0; j\neq i}^{k-1} a_{ij}\,,$$

or the sum of the off-diagonal elements of the contingency table A. Recall that

$$d_2(x, y) = \sqrt{\sum_{i=1}^{l}(x_i - y_i)^2}\,.$$

Now if $x, y \in \{0, 1\}^l$ then if $x_i = y_i$ then $(x_i - y_i)^2 = 0$ (as always) while if $x_i \neq y_i$ then $(x_i - y_i)^2 = 1$. Thus the sum above $\sum_{i=1}^{l}(x_i - y_i)^2$ equals the number of elements that differ. This is the same definition of the Hamming distance.

Problem 11.13

In general we can determine proximity functions between a point and a set from proximity functions between two sets by converting one of the sets to a set with a single point $\{x\}$. For example the "max" set-set similarity measure

$$s^{\text{ss}}_{\max}(D_i, D_j) = \max_{x\in D_i, y\in D_j} s(x, y)\,,$$

would be converted to a point-set similarity function in the straight forward way as

$$s^{\text{ps}}_{\max}(x, D_j) = \max_{y\in D_j} s(x, y)\,.$$

Clustering Algorithms I: Sequential Algorithms

Notes on the text

Notes on the number of possible clusterings

In this section of the text we would like to evaluate the numerical value of $S(N,m)$ so that we can determine if we need a clustering with m clusters from N points how many different clusterings would we have to search over to find the optimal clustering given some clustering criterion. We can compute $S(N,m)$ in terms of smaller values of N and m in the following way. Assume that we have $N-1$ points and we will add another point to get a total of N. We can add this additional point N in two ways and end up with m clusters

- If we have m clusters of $N-1$ points we can add this new point as a member of any of the m clusters to create m clusters of N points. This can be done in $mS(N-1,m)$ ways.
- If we have $m-1$ clusters of $N-1$ points we can add this new point as a *new* singleton cluster to create m clusters of N points. This can be done in $S(N-1,m-1)$ ways.

Since each of these is exclusive we can enumerate the totality of $S(N,m)$ ways to form m clusters from N points as

$$S(N,m) = mS(N-1,m) + S(N-1,m-1)\,.$$

We are told that the solution to this is given by the *Stirling number of the second kind* or

$$S(N,m) = \frac{1}{m!}\sum_{i=0}^{m}(-1)^{m-i}\left(\begin{array}{c} m \\ i \end{array}\right) i^N\,.$$

If $m=2$ we can evaluate the above expression as

$$\begin{aligned} S(N,2) &= \frac{1}{2}\sum_{i=0}^{2}(-1)^{2-i}\left(\begin{array}{c} 2 \\ i \end{array}\right) i^N = \frac{1}{2}\left(\left(\begin{array}{c} 2 \\ 0 \end{array}\right)0^N - \left(\begin{array}{c} 2 \\ 1 \end{array}\right)1^N + 2^N\right) \\ &= 2^{N-1}-1\,, \end{aligned}$$

or the books equation 12.3.

Notes on sequential clustering algorithms (BSAS) and (MBSAS)

In the MATLAB/Octave code `BSAS.m` and `MBSAS.m` we have implemented the basic and modified sequential algorithm schemes. To verify their correctness, in the script `dup_figure_12_1.m`

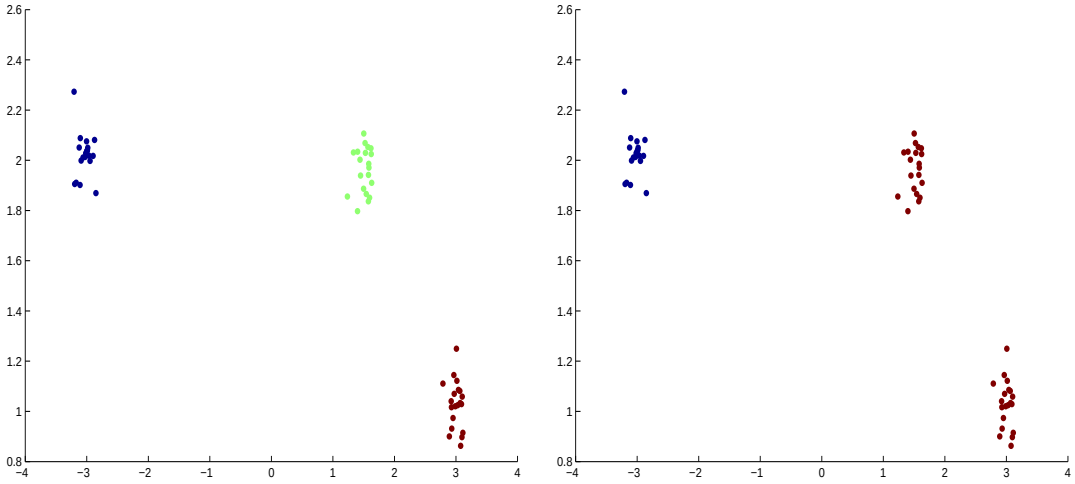

Figure 19: A duplicate of the books figure 12.1. **Left:** Running BSAS with q, the maximum number of clusters, taken to be $q = 3$ (or larger) **Right:** Running BSAS with $q = 2$.

we duplicated data like that presented in the books figure 12.1. Next we provide Matlab/Octave code that duplicates the books figure 12.2. The script `dup_figure_12_2.m` generates a very simple two cluster data set and then calls the function `estimateNumberOfClusters.m` which runs the BSAS algorithm many times each time with a randomly selected data ordering. When we run that script we obtain the two plots shown in Figure 20.

Notes on a two-threshold sequential scheme (TTSAS)

Next we provide Matlab/Octave code that duplicates the books figure 12.3. The script `dup_figure_12_3.m` creates the data suggested in example 12.3 (a very simple two cluster data set) and then calls the function `MBSAS.m` and the `TTSAS.m`. Where the Matlab/Octave code in `TTSAS.m` is an implementation of the two-threshold sequential scheme described in the book. When we run that script we obtain the two plots shown in Figure 21. Note that I was *not* able to get the `MBSAS` algorithm to show the *three* class clustering claimed in the book. Using the suggested value of $\Theta = 2.5$ gave the plot shown in Figure 21 (left). Increasing the threshold value for Θ to 3 however gave the same two clustering result that `TTSAS` gave shown in Figure 21 (right).

Notes on refinement stages

On the web site that accompanies this text one can find the Matlab procedure `merging.m` that implements the merging procedure discussed in in section of the text. To demonstrate the usage of the `merging.m` routine, in the Matlab script `merging_example.m` we consider the same data from Example 12.3 and clustered using the routine `MBSAS.m`. The results from running this clustering are presented in Figure 22 (left) there we see four clusters have been found. We next run the `merging.m` code on the resulting clusters and obtain

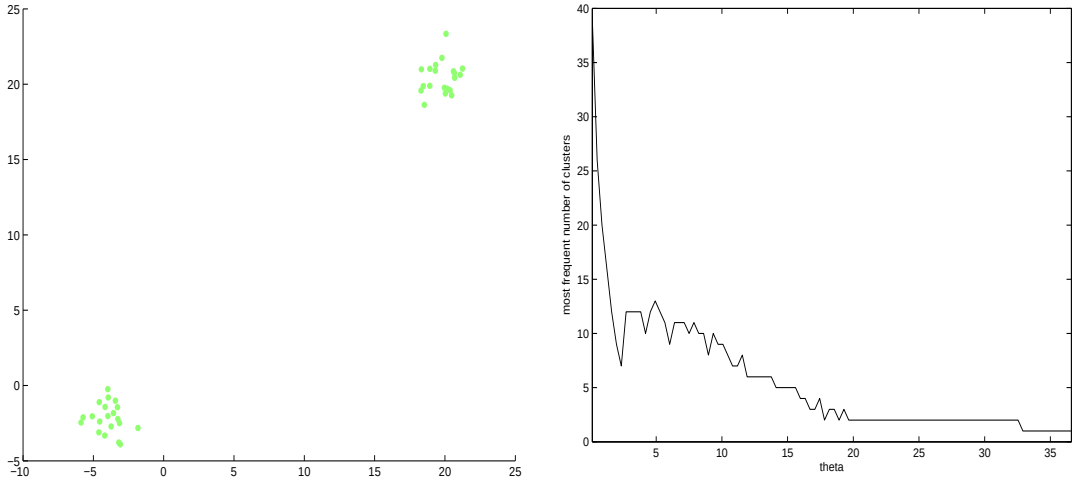

Figure 20: A duplicate of the books figure 12.2. **Left:** The initial data chosen to run the `estimateNumberOfClusters.m` function on. **Right:** A plot of the most frequent (mode) number of clusters found for each value of Θ. This plot looks similar to the one presented in the book. The long stretch of values of Θ where the mode is 2 indicates that 2 maybe a good value for the number of clusters present.

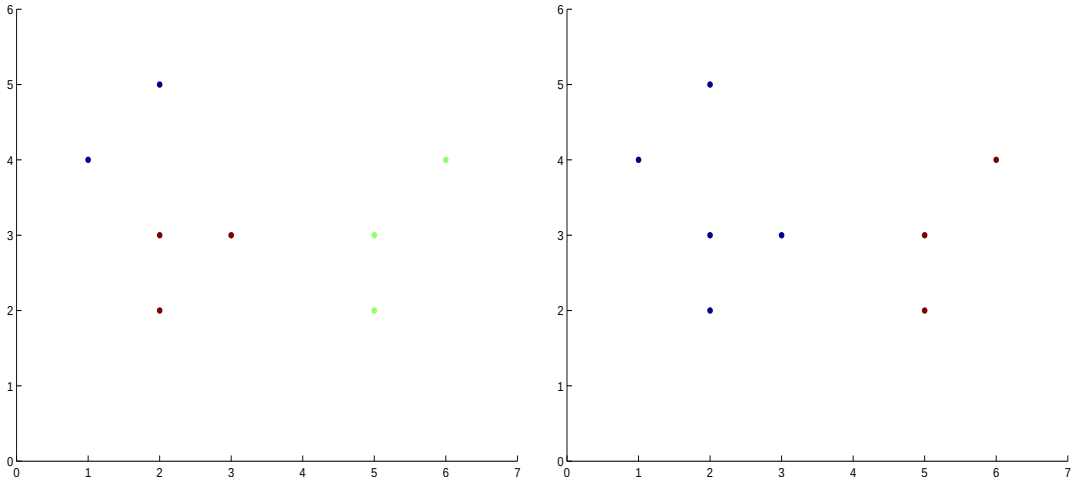

Figure 21: Plots duplicating the books figure 12.3. **Left:** The cluster labels produced using the `MBSAS` algorithm on the data set of example 12.3, and using the parameters given in the book. Note that I was *not* able to exactly duplicate the cluster results given in the book. **Right:** The result of apply the `TTSAS` algorithm on the data set of example 12.3.

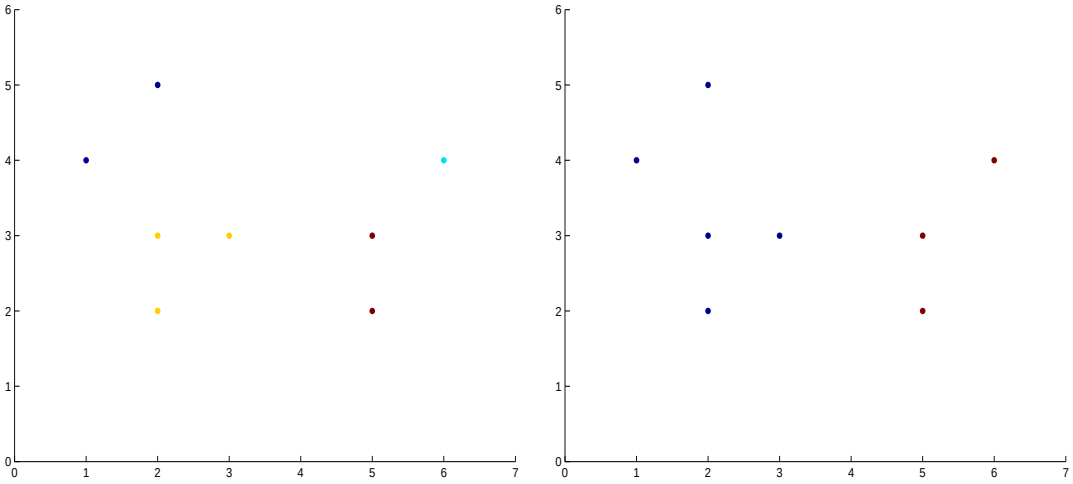

Figure 22: Plots to show the results from using the `merging.m` routine. **Left:** The cluster labels produced using the `MBSAS` algorithm on the data set from example 12.3. Note that we found four clusters. **Right:** The result of applying the `merging` algorithm on the resulting data set. We have merged two of the original clusters leaving two larger clusters.

Figure 22 (right). Note that we need to use a *larger* value for $M_1 = 2.85$ than the value of $\Theta = 1.5$ that which we used in clustering with `MBSAS`.

Problem Solutions

Problem 12.1 (the number of binary divisions of N points)

In this problem we prove by induction that given a set with N points the number of binary divisions (divisions into two non empty sets) is given by $2^{N-1} - 1$. See also page 160 where some alternative derivations of this same result are given. To begin note that from example 12.1 in the book that when $N = 3$ the number of binary divisions $S(3, 2) = 3$ which also equals the expression $2^{N-1} - 1$ when $N = 3$. Thus we have shown the required base case for an induction proof. Lets assume that

$$S(N, 2) = 2^{N-1} - 1 \quad \text{for} \quad N \leq N_1 \,,$$

and consider the evaluation of $S(N_1 + 1, 2)$. We can count the number of binary divisions of a set with $N_1 + 1$ points (and evaluate $S(N_1 + 1, 2)$) in the following way. First we can take each of the pairs of sets formed from N_1 points (of which there are $S(N_1, 2)$ of them) and introduce this $N_1 + 1$-st point into *either* of the pairs. This would give $2S(N_1, 2)$ sets with $N_1 + 1$ points. In addition, we can add this N_1-st point as a *singleton* set (a set with only one element) to the set with all other N_1 points. This can be done in only *one* way. Thus we have expressed $S(N_1 + 1, 2)$ as

$$S(N_1 + 1, 2) = 2S(N_1, 2) + 1 \,.$$

Using the induction hypothesis we find

$$S(N_1+1,2) = 2(2^{N_1-1}-1)+1 = 2^{N_1}-1\,,$$

as the number of binary divisions of a set with N_1+1 points. As this satisfies our desired expression for N_1+1 points we have proven the requested expression is true.

Problem 12.2 (recursively updating the cluster mean vector)

Since the mean vector of an "old" cluster C_k^{old} is defined as

$$m_{C_k^{\text{old}}} = \frac{1}{n_{C_k^{\text{old}}}} \sum_{x_i \in C_k^{\text{old}}} x_i\,,$$

we can represent the sum over all points in C_k^{old} cleanly as the product $n_{C_k^{\text{old}}} m_{C_k^{\text{old}}}$. If we merge another cluster D with n_D points into C_k^{old} to make a new cluster such that $C_k^{\text{new}} = C_k^{\text{old}} \cup D$ then the new mean vector over this merged cluster is given by

$$m_{C_k^{\text{new}}} = \frac{1}{n_{C_k^{\text{new}}}} \left(\sum_{x_i \in C_k^{\text{old}}} x_i + \sum_{x_i \in D} x_i \right) = \frac{1}{n_{C_k^{\text{new}}}} \left(n_{C_k^{\text{old}}} m_{C_k^{\text{old}}} + \sum_{x_i \in D} x_i \right)\,.$$

If we only have *one* point in D denoted as x then $n_{C_k^{\text{new}}} = n_{C_k^{\text{old}}} + 1$ and using the above we get

$$m_{C_k^{\text{new}}} = \frac{(n_{C_k^{\text{new}}} - 1) m_{C_k^{\text{old}}} + x}{n_{C_k^{\text{new}}}}$$

the requested expression.

Problem 12.3 (experiments with the BSAS and MBSAS algorithm)

Problem 12.4 (more experiments with BSAS and MBSAS)

Problem 12.5 (clustering in a square)

Problem 12.6 (estimating the number of clusters)

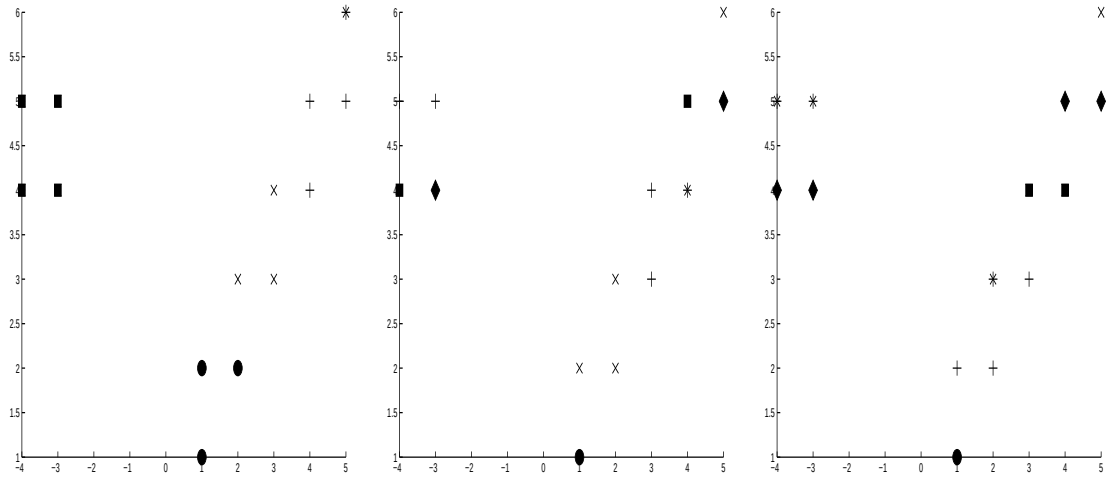

Figure 23: Some examples using the MATLAB function `histfit` function to plot normal densities on some "toy" data sets. **Left: Center: Right:**

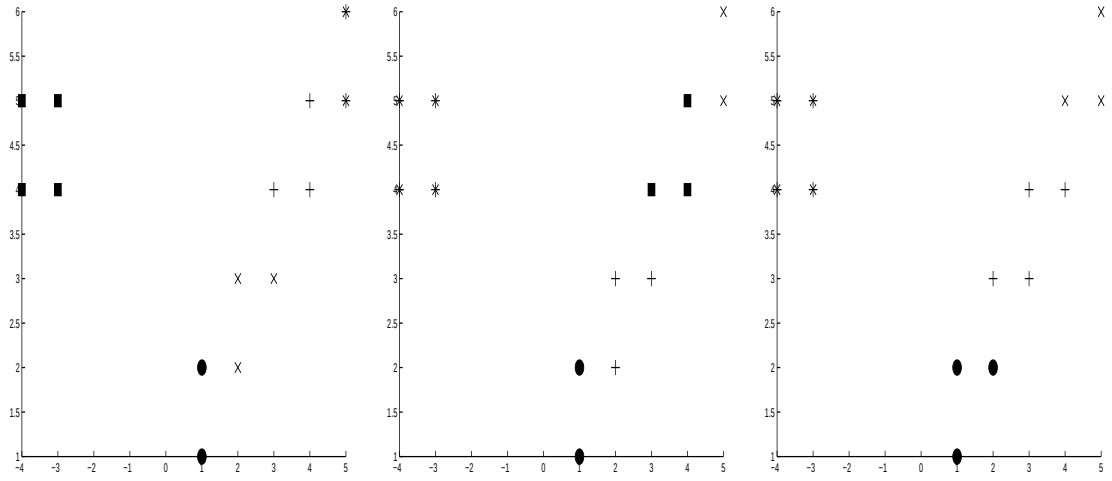

Figure 24: Some examples using the MATLAB function `histfit` function to plot normal densities on some "toy" data sets. **Left: Center: Right:**

Clustering Algorithms II: Hierarchical Algorithms

Notes on the text

Notes on agglomerative algorithms based on matrix theory

In this section of the book some algorithms for agglomerative clustering are introduced. These algorithms are presented in an way that involves modifications of the starting input dissimilarity matrix $P_0 = P(X)$ by modifying this matrix as clustering proceeds in stages from the initial singleton clusters of $\mathcal{R}_0$ which consist of only the points $\{x_i\}$ to the cluster $\mathcal{R}_{N-1}$ that contains all points in one cluster. What is not sufficiently emphasized in this section is that these clusters can be formed based only on the specified pairwise proximity measure between points. In normal clustering problem we initially have access to the sample points $\mathbf{x}_i$ from which we can then form the initial proximity matrix $P_0 = P(X)$, which has elements $\mathcal{P}(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j \in \{1, 2, \cdots, N\}$. Once this initial matrix is formed we can effectively "forget" the original data points $\mathbf{x}_i$ and work only with these proximity matrices P_t. At the step $t > 1$, once we have decided to merge two clusters C_i and C_j into a new cluster C_q the first step in obtaining the *new* proximity matrix P_t from the old P_{t-1} is to remove the ith and jth rows and columns of P_{t-1}. This corresponds to removing clusters C_i and C_j since their data are now merged into the new cluster C_q. Next, a new row and column is added to the modified proximity matrix P_{t-1}. This added row measures the cluster distances between the new cluster C_q and all existing unmodified clusters C_s for $s \in \{1, 2, \cdots, N-t\}$ and $s \notin \{i, j\}$. The values in the new proximity matrix P_t are created using elements derived from

$$d(C_q, C_s) = f(d(C_i, C_s), d(C_j, C_s), d(C_i, C_j)) , \tag{216}$$

for some function $f(\cdot, \cdot, \cdot)$ of three arguments. As a final comment most clustering algorithms can be express f in a simpler form as

$$\begin{aligned} d(C_q, C_s) &= a_i d(C_i, C_s) + a_j d(C_j, C_s) + b d(C_i, C_j) \\ &+ c|d(C_i, C_s) - d(C_j, C_s)| , \end{aligned} \tag{217}$$

for various values of a_i, a_j, b, and c. Note that in the above expression all of these cluster distances $d(\cdot, \cdot)$ have already been computed previously and can be found in the old proximity matrix P_{t-1}.

Notes on the minimum variance clustering algorithm (Ward)

In this section of these notes we derive the result that Wards linkage algorithm is equivalent to merging the two clusters that lead to the *smallest* possible increase in the total variance. We assume that the clusters C_i and C_j are to be merged at the iteration $t+1$ into a cluster denoted C_q and all other clusters remain the same. We begin by defining the total variance

E_t over all clusters at step t by

$$E_t = \sum_{r=1}^{N-t} e_r^2 \,, \tag{218}$$

where e_r is the scatter around the rth cluster given by

$$e_r = \sum_r ||x - m_r||^2 \,. \tag{219}$$

Then the *change* in the total variance denoted by $E_{t+1} - E_t$ where E_t is the total cluster variance at step t defined above under the merge of the clusters C_i and C_j will be

$$\Delta E_{t+1}^{ij} = e_q^2 - e_i^2 - e_j^2 \,. \tag{220}$$

Noting that we can express $\sum_{x \in C_r} ||x - m_r||^2$ as

$$\begin{aligned}
\sum_{x \in C_r} ||x - m_r||^2 &= \sum_{x \in C_r} (||x||^2 - 2x^T m_r + ||m_r||^2) \\
&= \sum_{x \in C_r} ||x||^2 - 2 \left(\sum_{x \in C_r} x^T \right) m_r + n_r ||m_r||^2 \\
&= \sum_{x \in C_r} ||x||^2 - 2 n_r m_r^T m_r + n_r ||m_r||^2 \\
&= \sum_{x \in C_r} ||x||^2 - 2 n_r ||m_r||^2 + n_r ||m_r||^2 \\
&= \sum_{x \in C_r} ||x||^2 - n_r ||m_r||^2 \,.
\end{aligned} \tag{221}$$

This last equation is the books equation 13.16. If we use expressions like this to evaluate the three terms e_q, e_i and e_j we have

$$\begin{aligned}
\Delta E_{t+1}^{ij} &= \sum_{x \in C_q} ||x||^2 - n_q ||m_q||^2 - \sum_{x \in C_i} ||x||^2 + n_i ||m_i||^2 - \sum_{x \in C_j} ||x||^2 + n_j ||m_j||^2 \\
&= n_i ||m_i||^2 + n_j ||m_j||^2 - n_q ||m_q||^2 \,.
\end{aligned} \tag{222}$$

Since when we merge cluster C_i and C_j into C_q we take all of the points from C_i and C_j in forming C_q we have

$$\sum_{x \in C_q} ||x||^2 = \sum_{x \in C_i} ||x||^2 + \sum_{x \in C_j} ||x||^2 \,.$$

An equivalent statement of the above in terms of the means m_i and the number of elements summed in each mean n_i is

$$n_i m_i + n_j m_j = n_q m_q \,,$$

since the product $n_i m_i$ is just the sum of the x vectors in C_i. From this last expression we compute

$$\begin{aligned}
||m_q||^2 &= \frac{1}{n_q^2} ||n_i m_i + n_j m_j||^2 \\
&= \frac{1}{n_q^2} (n_i^2 ||m_i||^2 + 2 n_i n_j m_i^T m_j + n_j^2 ||m_j||^2) \,.
\end{aligned}$$

Thus using this expression in Equation 222 we can write ΔE_{t+1}^{ij} as

$$\Delta E_{t+1}^{ij} = n_i||m_i||^2 + n_j||m_j||^2 - \frac{1}{n_q}(n_i^2||m_i||^2 + 2n_i n_j m_i^T m_j + n_j^2||m_j||^2)\,.$$

Since we have $n_q = n_i + n_j$ this simplifies to

$$\begin{aligned}\Delta E_{t+1}^{ij} &= \frac{1}{n_q}\left[n_i^2||m_i||^2 + n_i n_j||m_i||^2 + n_i n_j||m_j||^2 + n_j^2||m_j||^2\right. \\ &- \left. n_i^2||m_i||^2 - 2n_i n_j m_i^T m_j - n_j^2||m_j||^2\right] \\ &= \frac{n_i n_j}{n_q}\left[||m_i||^2 - 2m_i^T m_j + ||m_j||^2\right] \\ &= \frac{n_i n_j}{n_q}||m_i - m_j||^2\,.\end{aligned}$$

This last equation is the books equation 13.19.

Notes on agglomerative algorithms based on graph theory

In this section it can be helpful to discuss Example 13.4 in some more detail. Now in going from the clustering $\mathcal{R}_2 = \{\{x_1, x_2\}, \{x_3\}, \{x_4, x_5\}\}$ to $\mathcal{R}_3$ we are looking for the smallest value of a such that the threshold graph $G(a)$ has the property $h(k)$. Thus we compute $g_{h(k)}(C_r, C_s)$ for all pairs (C_r, C_s) of possible merged clusters. In this case the two clusters we would consider merging in the new clustering $\mathcal{R}_3$ are

$$\{x_1, x_2\} \cup \{x_3\}\,, \quad \{x_3\} \cup \{x_4, x_5\}\,, \quad \text{or} \quad \{x_1, x_2\} \cup \{x_4, x_5\}\,.$$

The smallest value of $g_{h(k)}$ using these pairs is given by 1.8 since when $a = 1.8$, the threshold graph $G(a)$ of $\{x_3\} \cup \{x_4, x_5\}$ is connected (which is property $h(k)$ for this example).

Another way to view the numerical value of the function $g_{h(k)}(C_r, C_s)$ is to express its meaning in words. In words, the value of the function

$$g_{h(k)}(C_r, C_s)\,,$$

is the *smallest* value of a such that the threshold graph $G(a)$ of the set $C_r \cup C_s$ is *connected* and has *either* one of two properties:

- The set $C_r \cup C_s$ has the property $h(k)$.
- The set $C_r \cup C_s$ is *complete*.

Thus we see that the value of $g_{h(k)}(\{x_1, x_2\}, \{x_3, x_4, x_5\})$ is 2.5 since in this example this is where the threshold graph $G(2.5)$ of $\{x_1, x_2\} \cup \{x_3, x_4, x_5\} = X$ is now connected which is property $h(k)$ for the single link algorithm.

Notes on divisive algorithms

The book claims the result that the number of possible partition of of N points into two sets is given by $2^{N-1}-1$ but no proof is given. After further study (see Page 151) this result can be derived from the Stirling numbers of the second kind $S(N,m)$ by taking the number of clusters $m=2$ but we present two alternative derivations here that might be simpler to understand.

Method 1: The first way to derive this result is to consider in how many ways could we label each of the points x_i such that each point is in one cluster. Since each point can be in one of two clusters we can denote its membership by a 0 or a 1 depending on which cluster a given point should be assigned. Thus the first point x_1 has 2 possible labelings (a 0 or a 1) the second point x_2 has the same two possible labelings etc. Thus the total number of labeling for all N points is

$$2 \times 2 \cdots 2 \times 2 = 2^N \,.$$

This expression over counts the total number of two cluster divisions in two ways. The first is that it includes the labeling where every point x_i gets the *same* label. For example all points are labeled with a 1 or a 0, of which there are two cases giving

$$2^N - 2 \,.$$

This number also over counts the number of two cluster divisions in that it includes *two* labelings for each allowable cluster. For example, using the above procedure when $N=3$ we have the possible labelings

x_1	x_2	x_3	
0	0	0	X
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	3
1	0	1	2
1	1	0	1
1	1	1	X

In the above we have separated the labelings and an additional piece of information with a vertical pipe |. We present the two invalid labelings that don't result in two nonempty sets with an X. Note also that the labeling $0\,,0\,,1$ and $1\,,1\,,0$ are equivalent in that they have the same points in the two clusters. To emphasis this we have denoted the 3 pairs of equivalent labelings with the integers $1,2$ and 3. Thus we see that the above counting represents twice as many clusters. Thus the number of two cluster divisions is given by

$$\frac{1}{2}(2^N - 2) = 2^{N-1} - 1 \,,$$

as we were to show.

Method 2: In this method we recognize that the total problem of counting the number of partitions of the N points into two clusters has as a subproblem counting the number of

ways we can partition of the N points into two sets of size k and $N-k$. This subproblem can be done in $\left(\begin{array}{c} N \\ k \end{array} \right)$ ways. Since we don't care how many points are in the individual clusters the total number of ways in which we can perform a partition into two sets might be represented like

$$\sum_{k=1}^{N-1} \left(\begin{array}{c} N \\ k \end{array} \right) .$$

Here we have exclude from the above sum the sets with $k = 0$ and $K = N$ since they correspond to all of the N points in one set and no points in the other set. The problem with this last expression is that again it over counts the number of sets. This can be seen from the identity $\left(\begin{array}{c} N \\ k \end{array} \right) = \left(\begin{array}{c} N \\ N-k \end{array} \right)$. Thus to correctly count these we need to divide this expression by two to get

$$\frac{1}{2} \sum_{k=1}^{N-1} \left(\begin{array}{c} N \\ k \end{array} \right) .$$

We can evaluate this expression if we recall that

$$\sum_{k=0}^{N} \left(\begin{array}{c} N \\ k \end{array} \right) = 2^N , \tag{223}$$

Using this the above becomes

$$\frac{1}{2} \sum_{k=1}^{N-1} \left(\begin{array}{c} N \\ k \end{array} \right) = \frac{1}{2} \left(2^N - \left(\begin{array}{c} N \\ 0 \end{array} \right) - \left(\begin{array}{c} N \\ N \end{array} \right) \right) = \frac{1}{2}(2^N - 2) = 2^{N-1} - 1 ,$$

the same expression as earlier.

Problem Solutions

Problem 13.1 (the definitions of the pattern / proximity matrix)

Part (a): The pattern matrix $D(X)$ is the $N \times l$ matrix whos ith row is the transposed ith vector of X. This matrix thus contains the N feature vectors as rows stacked on top of each other. Since the proximity matrix $P(X)$ is the $N \times N$ with (i,j)th elements that are given by either $s(x_i, x_j)$ if the proximity matrix corresponds to a similarity matrix or to $d(x_i, x_j)$ if the proximity matrix corresponds to a dissimilarity matrix. The term "proximity matrix" covers both cases. Thus given the pattern matrix $D(X)$ an application of the proximity function will determine a unique proximity matrix

Part (b): To show that a proximity matrix does not determine the pattern matrix one would need to find two sets feature vectors that are the same under the Euclidean distance. The scalar measurements $1, 2, 3$ with the dissimilarity metric $d(x,y) = |x-y|$ has a proximity

matrix given by

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}.$$

While the scalar values $3, 4, 5$ have the same proximity matrix. These two sets have different pattern matrices given by

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}.$$

Problem 13.2 (the unweighted group method centroid (UPGMC))

To solve this problem lets first show that

$$n_1||m_1 - m_3||^2 + n_2||m_2 - m_3||^2 = \frac{n_1 n_2}{n_1 + n_2}||m_1 - m_2||^2 \tag{224}$$

Since $n_3 = n_1 + n_2$ and $C_3 = C_1 \cup C_2$, we can write m_3 as

$$\begin{aligned} m_3 &= \frac{1}{n_3}\sum_{x \in C_3} x = \frac{1}{n_3}\left[\sum_{x \in C_1} x + \sum_{x \in C_2} x\right] \\ &= \frac{1}{n_3}\left[n_1 m_1 + n_2 m_2\right]. \end{aligned}$$

Now the left-hand-side of Equation 224 denoted by

$$\text{LHS} = n_1||m_1 - m_3||^2 + n_2||m_2 - m_3||^2,$$

using the above for m_3 is given by

$$\begin{aligned} \text{LHS} &= n_1\left|\left|\left(1 - \frac{n_1}{n_3}\right)m_1 - \frac{n_2}{n_3}m_2\right|\right|^2 + n_2\left|\left|\left(1 - \frac{n_2}{n_3}\right)m_2 - \frac{n_1}{n_3}m_1\right|\right|^2 \\ &= n_1\left(1 - \frac{n_1}{n_3}\right)^2||m_1||^2 - 2n_1\left(1 - \frac{n_1}{n_3}\right)\left(\frac{n_2}{n_3}\right)m_1^T m_2 + n_1\frac{n_2^2}{n_3^2}||m_2||^2 \\ &+ n_2\left(1 - \frac{n_2}{n_3}\right)^2||m_2||^2 - 2n_2\left(1 - \frac{n_2}{n_3}\right)\left(\frac{n_1}{n_3}\right)m_1^T m_2 + n_2\frac{n_1^2}{n_3^2}||m_2||^2 \\ &= \left[n_1\left(1 - 2\frac{n_1}{n_3} + \frac{n_1^2}{n_3^2}\right) + n_2\left(\frac{n_1^2}{n_3^2}\right)\right]||m_1||^2 \\ &- 2\left[n_1\left(1 - \frac{n_1}{n_3}\right)\left(\frac{n_2}{n_3}\right) + n_2\left(1 - \frac{n_2}{n_3}\right)\left(\frac{n_1}{n_3}\right)\right]m_1^T m_2 \\ &+ \left[n_1\frac{n_2^2}{n_3^2} + n_2\left(1 - 2\frac{n_2}{n_3} + \frac{n_2^2}{n_3^2}\right)\right]||m_2||^2. \end{aligned}$$

Next consider the coefficient of $||m_1||^2$. We see that it is equal to

$$\begin{aligned} \frac{n_1}{n_3^2}(n_3^2 - 2n_1 n_3 + n_1^2) + n_2\left(\frac{n_1^2}{n_3^2}\right) &= \frac{n_1}{n_3^2}(n_3 - n_1)^2 + n_2\left(\frac{n_1^2}{n_3^2}\right) \\ &= \frac{n_1 n_2}{n_3^2}(n_2 + n_1) = \frac{n_1 n_2}{n_3}. \end{aligned}$$

The same type of transformation changes the coefficient of $||m_2||^2$ into $\frac{n_1 n_2}{n_3}$ also. For the coefficient of $m_1^T m_2$ we have

$$\begin{aligned}\left[n_1\left(1-\frac{n_1}{n_3}\right)\left(\frac{n_2}{n_3}\right)+n_2\left(1-\frac{n_2}{n_3}\right)\left(\frac{n_1}{n_3}\right)\right] &= \frac{n_1 n_2}{n_3}\left[1-\frac{n_1}{n_3}+1-\frac{n_2}{n_3}\right] \\ &= \frac{n_1 n_2}{n_3}\left[2-\frac{n_1+n_2}{n_3}\right] \\ &= \frac{n_1 n_2}{n_3}\,.\end{aligned}$$

Using all three of these results we have that

$$\text{LHS} = \frac{n_1 n_2}{n_3}\left[||m_1||^2 - 2m_1^T m_2 + ||m_2||^2\right] = \frac{n_1 n_2}{n_3}||m_1 - m_2||^2\,,$$

which we were to show. Now we can proceed to solve the requested problem. We begin by recalling that the recursive matrix update algorithm for UPGMC when we merge cluster C_i and C_j into C_q is given by

$$d_{qs} = \left(\frac{n_i}{n_i+n_j}\right)||m_i - m_s||^2 + \frac{n_j}{n_i+n_j}||m_j - m_s||^2 - \frac{n_i n_j}{(n_i+n_j)^2}||m_i - m_j||^2\,. \tag{225}$$

If we use Equation 224 with $m_1 = m_i$, $m_2 = m_j$ and $m_3 = m_q$ to express $||m_i - m_j||^2$ in the above as

$$||m_i - m_j||^2 = \left(\frac{n_i+n_j}{n_i n_j}\right)\left[n_i||m_i - m_q||^2 + n_j||m_j - m_q||^2\right]$$

Using this we can then write d_{qs} as

$$\begin{aligned} d_{qs} &= \frac{n_i}{n_q}||m_i - m_s||^2 + \frac{n_j}{n_q}||m_j - m_s||^2 \\ &- \frac{n_i}{n_q}||m_i - m_q||^2 - \frac{n_j}{n_q}||m_j - m_q||^2 \\ &= \frac{1}{n_q}\left[n_i(||m_i - m_s||^2 - ||m_i - m_q||^2) + n_j(||m_j - m_s||^2 - ||m_j - m_q||^2)\right]\,. \end{aligned}$$

To simplify this recall that for any two vectors a and b we have

$$||a||^2 - ||b||^2 = (a-b)^T(a+b) = (a+b)^T(a-b)\,,$$

as one can prove by expanding out the product in the right-hand-side. Using this we can write d_{qs} as

$$\begin{aligned} d_{qs} &= \frac{1}{n_q}\left[n_i(m_i - m_s + m_i - m_q)^T(m_i - m_s - (m_i - m_q))\right. \\ &+ \left. n_j(m_i - m_s + m_j - m_q)^T(m_j - m_s - (m_j - m_q))\right] \\ &= \frac{1}{n_q}\left[n_i(2m_i - m_s - m_q)^T(m_q - m_s) + m_j(2m_j - m_s - m_q)^T(m_q - m_s)\right] \\ &= \frac{1}{n_q}\left[n_i(2m_i - m_s - m_q) + m_j(2m_j - m_s - m_q)\right]^T(m_q - m_s) \\ &= \frac{1}{n_q}\left[2n_i m_i + 2n_j m_j - n_i m_s - n_i m_q - n_j m_s - n_j m_q\right]^T(m_q - m_s)\,. \end{aligned}$$

Since $n_i m_i + n_j m_j = n_q m_q$ the above becomes

$$\begin{aligned} d_{qs} &= \frac{1}{n_q}\left[(n_i+n_j)m_q - n_i m_s - n_j m_s\right]^T (m_q - m_s) \\ &= \frac{1}{n_q}\left[n_i(m_q - m_s) + n_j(m_q - m_s)\right]^T (m_q - m_s) \\ &= \frac{1}{n_q}(n_i+n_j)(m_q - m_s)^T(m_q - m_s) = ||m_q - m_s||^2 , \end{aligned}$$

which is the result we wanted to show.

Problem 13.3 (properties of the WPGMC algorithm)

The weighted pair group mean centroid (WPGMC) algorithm has an update equation for d_{qs} given by

$$d_{qs} = \frac{1}{2}d_{is} + \frac{1}{2}d_{js} - \frac{1}{4}d_{ij} .$$

That there exists cases where $d_{qs} \leq \min(d_{is}, d_{js})$ is easy to see. Consider any existing cluster C_s that is equally distant between C_i and C_j or $d_{is} = d_{js} = \min(d_{is}, d_{js})$. A specific example of three clusters like this could be created from single points if needed. Then in this case using the above WPGMC update algorithm we would have

$$d_{qs} = d_{is} - \frac{1}{4}d_{ij} \leq d_{is} = \min(d_{is}, d_{js}) .$$

Problem 13.4 (writing the Ward distance as a MUAS update)

The Ward or minimum variance algorithm defines a distance between two clusters C_i and C_j as

$$d'_{ij} = \frac{n_i n_j}{n_i + n_j}||m_i - m_j||^2 .$$

We want to show that this distance update can be written in the form of a MUAS algorithm

$$d(C_q, C_s) = a_i d(C_i, C_s) + a_j d(C_j, C_s) + b d(C_i, C_j) + c|d(C_i, C_s) - d(C_j, C_s)| . \tag{226}$$

In problem 3.2 we showed that

$$d_{qs} = \frac{n_i}{n_i + n_j}d_{is} + \frac{n_j}{n_i + n_j}d_{js}d_{is} - \frac{n_i n_j}{(n_i + n_j)^2}d_{ij} = ||m_q - m_s||^2 .$$

As suggested in the hint lets multiply both sides of this expression by the expression $\frac{(n_i+n_j)n_s}{n_i+n_j+n_s}$ to get

$$\begin{aligned} \frac{(n_i+n_j)n_s}{n_i+n_j+n_s}||m_q - m_s||^2 &= \frac{n_i n_s}{n_i+n_j+n_s}d_{is} + \frac{n_j n_s}{n_i+n_j+n_s}d_{js} \\ &- \frac{n_i n_j n_s}{(n_i+n_j)(n_i+n_j+n_s)}d_{ij} . \end{aligned}$$

Writing $n_q = n_i + n_j$ the left-hand-side of the above becomes

$$\frac{n_q n_s}{n_q + n_s} d_{qs} = d'_{qs} \,,$$

while the right-hand-side becomes

$$\frac{n_i + n_s}{n_i + n_j + n_s} \left(\frac{n_i n_s}{n_i + n_j} \right) d_{is} + \frac{n_j + n_s}{n_i + n_j + n_s} \left(\frac{n_j n_s}{n_j + n_s} \right) d_{js} - \frac{n_s}{n_i + n_j + n_s} \left(\frac{n_i n_j}{n_i + n_j} \right) d_{ij} \,,$$

or introducing thee definition of d' and equating these two expressions we have

$$d'_{qs} = \frac{n_i + n_s}{n_i + n_j + n_s} d'_{is} + \frac{n_j + n_s}{n_i + n_j + n_s} d'_{js} - \frac{n_s}{n_i + n_j + n_s} d'_{ij} \,.$$

This is the books equation 13.13 which we were to show.

Problem 13.5 (Wards algorithm is the smallest increase in variance)

This problem is worked on Page 157 of these notes.

Problem 13.7 (clusters distances from the single link algorithm)

The single link algorithm in the matrix updating algorithmic scheme (MUAS) has a dissimilarity between the new cluster C_q formed from the clusters C_i and C_j and an old cluster C_s given by

$$d(C_q, C_s) = \min(d(C_i, C_s), d(C_j, C_s)) \,. \tag{227}$$

We desire to show that this distance is equal to the smallest pointwise distance for point taken from the respective clusters or

$$d(C_q, C_s) = \min_{x \in C_q, y \in C_s} d(x, y) \,. \tag{228}$$

At step R_0 when every cluster is a single point Equation 228 is true since C_q and C_s are singleton point sets i.e. sets with only one element in them. Assuming that at level t the clusters at that level in $\mathcal{R}_t$ have clusters distances where Equation 228 holds we will now prove that the clusters at level $t+1$ will also have distances that satisfy this property. Consider the next cluster level R_{t+1}, where we form the cluster C_q from the sets C_i and C_j say picked as specified by the Generalized Agglomerative Scheme (GAS) with

$$g(C_i, C_j) = \min_{r,s} g(C_r, C_s) \,,$$

where g is a dissimilarity measure. Thus to show that Equation 228 holds between the new set C_q and all the original unmerged sets from R_t we use Equation 227 to write

$$d(C_q, C_s) = \min(d(C_i, C_s), d(C_j, C_s)) \,,$$

Figure 25: **Left:** Single link clustering on the dissimilarity matrix for problem 13.10. **Right:** Complete link clustering on the dissimilarity matrix for problem 13.10.

and then use the induction hypothesis to write the above as

$$\begin{aligned} d(C_q, C_s) &= \min\left(\min_{x_1 \in C_i, y_1 \in C_s} d(x_1, y_1), \min_{x_2 \in C_j, y_2 \in C_s} d(x_2, y_2) \right) \\ &= \min_{x \in C_i \cup C_j, y \in C_s} d(x, y)\,. \end{aligned}$$

This last expression is what we wanted to prove.

Problem 13.8 (clusters distances from the complete link algorithm)

All of the arguments in Problem 13.7 are still valid for this problem when we replace minimum with maximum.

Problem 13.9 (similarity measures vs. dissimilarity measures)

For this problem one simply replaces dissimilarities $d(x, y)$ with similarities $s(x, y)$ and replaces minimizations with maximizations in the two earlier problems. All of the arguments are the same.

Problem 13.10 (some simple dendrograms)

Note that for this proximity matrix we do have ties in P (for example $P(2, 3) = 1 = P(4, 5)$) and thus as discussed in the book we may not have a unique representations for the dendrogram produced by the complete link clustering algorithm. The single link clustering

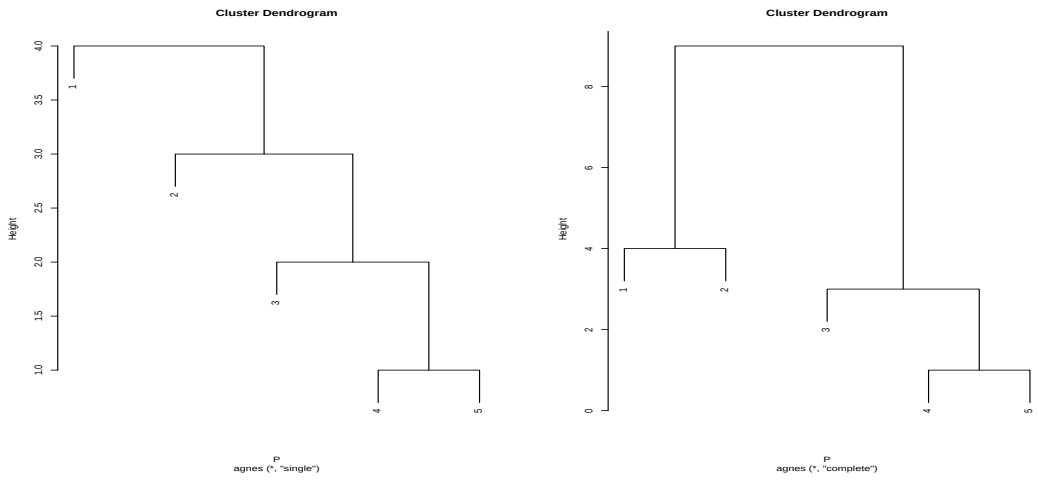

Figure 26: **Left:** Single link clustering on the dissimilarity matrix for problem 13.12. **Right:** Complete link clustering on the dissimilarity matrix for problem 13.12.

algorithm should produce a unique dendrogram however. We can use the `R` language to plot the associated single and complete link dendrograms. See the `R` file `chap_13_prob_10.R` for the code to apply these clustering procedures to the given proximitry matrix. Two dendrograms for this problem that are output from the above script are shown in Figure 25.

Problem 13.12 (more simple dendrograms)

Part (a): Note that for this proximity matrix we have ties in P (for example $P(2,3) = 3 = P(4,5)$) and thus we may not have a unique representation of the dendrogram for the complete link clustering algorithm. The histogram may depend on the order in which the two tied results are presented to the clustering algorithm. The single link clustering algorithm should be unique however. See the `R` file `chap_13_prob_12.R` for numerical code to perform complete and single link clustering on the given proximity matrix. Results from running this code are presented in Figure 26.

Problem 13.13 (a specification of the general divisive scheme (GDS))

To begin this problem, recall that in the general divisive clustering scheme the rule 2.2.1 is where given a cluster from the previous timestep, $C_{t-1,i}$, we consider *all* possible pairs of clusters (C_r, C_s) that could form a partition of the cluster $C_{t-1,i}$. From all possible pairs we search to find the pair $(C^1_{t-1,i}, C^2_{t-1,i})$ that gives the maximum value for $g(C_r, C_s)$ where $g(\cdot,\cdot)$ is some measure of cluster dissimilarity. In this problem we are further restricting the general partioning above so that we only consider pairs $(C^1_{t-1,i}, C^2_{t-1,i})$ where $C_{t-1,i} = C^1_{t-1,i} \cup C^2_{t-1,i}$ and $C^1_{t-1,i}$ only has *one* point. We can consider the total number of cluster comparisons required by this process as follows.

- At $t = 0$ there is only one cluster in $\mathcal{R}_0$ and we need to do N comparisons of the values $g(\{x_i\}, X - \{x_i\})$ for $i = 1, 2, \cdots, N$ to find the single point to split first.
- At $t = 1$ we have two clusters in $\mathcal{R}_1$ where one cluster a singleton (has only a single point) and the other cluster has $N - 1$ points. Thus we can only possibly divide the cluster with $N - 1$ points. Doing that will require $N - 1$ cluster comparisons to give $\mathcal{R}_2$.
- In general, we see that at step t we have $N - t$ comparisons to make to derive the new clustering $\mathcal{R}_{t+1}$ from $\mathcal{R}_t$.

Thus we see that this procedure would require

$$N + (N-1) + (N-2) + \cdots + 3\,,$$

comparisons. The above summation stops at three because this is the number of comparisons required to find the single split point for a cluster of three points. The above sum can be evaluated as

$$\left(\sum_{k=1}^{N} k\right) - 1 - 2 = \frac{N(N+1)}{2} - 3 = \frac{N^2 + N - 6}{2}\,.$$

The merits of this procedure is that it is not too computationally demanding since it is an $O(N^2)$ procedure. From the above discussion we would expect that this procedure will form clusters that are similar to that formed under single link clustering i.e. the clusters will most likely possess chaining. Note that in searching over such a restricted space for the two sets $C^1_{t-1,i}$ and $C^2_{t-1,i}$ this procedure will not fully explore the space of all possible partitions of $C_{t-1,i}$ and thus could result in non optimal clustering.

Problem 13.14 (the alternative divisive algorithm)

The general divisive scheme (GDS) for $t > 0$ procedes by consideing all clusters $C_{t-1,i}$ for $i = 1, 2, \cdots, t$ from the clustering $\mathcal{R}_{t-1}$ and for each cluster all their $2^{|C_{t-1,i}|-1} - 1$ possible divisions into two sets. Since the procedure we apply in the GDS is the same for each timestep t we can drop that index from the cluster notation that follows.

The *alternative* divisive algoithm searches over much fewer sets than $2^{|C_i|-1} - 1$ required by the GDS by instead performing a *linear* search on the $|C_i|$ elements of C_i. Since for large values of $|C_i|$ we have

$$|C_i| \ll 2^{|C_i|-1} - 1\,,$$

this alternative procedure has a much smaller search space and can result in significant computational savings over the GDS.

The discription of this alternative partiioning procedure for C_i is as follows. We start with an "empty set" $C^1_i = \emptyset$ and a "full set" C^2_i, where the "full set" is initialized to be C_i. As a first step, we find the vector x in the full set, C^2_i, who's average distance with the remaining vectors in C^2_i is the largest and move that point x into the emtpy set C^1_i. If we define $g(x, C)$

to be a function that measures the average dissimilarity between a point x and the set C the first point we move into C_i^1 will be the point x that maximizes

$$g(x, C_i^2 - \{x\}) ,$$

where $C_i^2 - \{x\}$ means the set C_i^2 but without the point $\{x\}$ in it. After this initial point has been moved into C_i^1 we now try to move more points out of the full set C_i^2 and into the empty set C_i^1 as follows. For all remaining $x \in C_i^2$ we would like to move a point x into C_i^1 from C_i^2 if the dissimilarity of x with C_i^1 is smaller than that of x with C_i^2 (without x) or

$$g(x, C_i^1) < g(x, C_i^2 - \{x\}) .$$

Motivated by this expression we compute

$$D(x) \equiv g(x, C_i^2 - \{x\}) - g(x, C_i^1) ,$$

for all $x \in C_i^2$. We then take the point x^* that makes $D(x)$ largest but only if at x^* the value of $D(x^*)$ is still positive. If no such point exists that is $D(x) < 0$ for all x or equivalently

$$g(x, C_i^1) > g(x, C_i^2 - \{x\}) ,$$

then we stop and return the two sets (C_i^1, C_i^2). Note that this procedure *cannot* gaurentee to give the optimal partition of the set C_i since we are limiting our search of possible splits over the much smaller space of pairwise sets than the full general divisive scheme would search over.

Problem 13.15 (terminating the number of clusters on $\theta = \mu + \lambda\sigma$)

This problem refers to Method 1 for determining the number of clusters suggested by the data. To use this method one needs to introduce a set function $h(C)$ that provides a way of measuring how dissimilar the vectors in a given set are. Common measures for $h(C)$ might be

$$\begin{aligned} h(C) &= \max_{x,y \in C} d(x, y) \\ h(C) &= \text{median}_{x,y \in C} d(x, y) , \end{aligned}$$

where $d(\cdot, \cdot)$ is a dissimilarity measure. Then we stop clustering with $\mathcal{R}_t$ at level t when there is a cluster in $\mathcal{R}_{t+1}$ (the next level) that is so "large" that it has points that are too dissimilar to continue. Mathematically that means that we keep the $\mathcal{R}_t$ clustering (and cluster farther than the $\mathcal{R}_{t+1}$ clustering) if

$$\exists C_j \in \mathcal{R}_{t+1} \quad \text{such that} \quad h(C_j) > \theta ,$$

where the threshold θ still has to be determined experimentally. To help in evaluating the value of θ we might write it as

$$\theta = \mu + \lambda\sigma ,$$

where μ is the average dissimilarity between two points in the full set X and σ is the variance of that distance. These can be computed as

$$\begin{aligned} \mu &= \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} d(x_i, x_j) \\ \sigma^2 &= \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(d(x_i, x_j) - \mu\right)^2 . \end{aligned}$$

Thus when the choice of the value of the threshold θ is transfered to the choice of a value for λ we are effectivly saying that we will stop clustering when we get sets that have a average dissimilatrity greater than λ standard deviations from the average pointwise dissimilarity.

Clustering Algorithms III: Schemes Based on Function Optimization

Notes on the text

Notes on compact and hyperellipsoidal clusters

The equations presented in the book for estimating μ_j and Σ_j are derived earlier in these notes, see Page 29.

Note I believe there is a typo in the books more general expression for the M-step in the expectation maximization (EM) algorithm. The book starts by defining our objective function $Q(\Theta; \Theta(t))$ given by

$$Q(\Theta; \Theta(t)) = \sum_{i=1}^{N} \sum_{j=1}^{m} P(C_j|x_i; \Theta(t)) \ln(p(x_i|C_j; \theta) P_j) \,, \tag{229}$$

and then argues that the parameters of the jth cluster θ_j are functionally independent of the parameters of the kth cluster when $k \neq j$. Thus when we take the θ_j derivative of the above expression to find the maximum of $Q(\Theta; \Theta(t))$ we *loose* any reference to any other clusters $k \neq j$. Thus the sum over j falls away and we get

$$\sum_{i=1}^{N} P(C_j|x_i; \Theta(t)) \frac{\partial}{\partial \theta_j} \ln(p(x_i|C_j; \theta_j) = 0 \,. \tag{230}$$

The book has an additional sum over the index j which I believe should not be there.

Even though the EM algorithm for the multidimensional case is derived on Page 29. I found the derivation given in the Appendix of this chapter informative and wanted to further elucidate the discussion there. In this derivation we wish to consider Equation 230 where the elements of θ_j are the individual elements of the jth covariance matrix Σ_j. To this end we take the derivative of $ln(\cdot)$ with respect to the (r, s) element of the *inverse* of Σ_j, which we denote as σ_{rs}. This means that

$$\begin{aligned}
\frac{\partial}{\partial \sigma_{rs}} \ln(p(x|C_j; \theta_j) &= \frac{\partial}{\partial \sigma_{rs}} \left(\ln \left(\frac{|\Sigma_j^{-1}|}{(2\pi)^{l/2}} \right) - \frac{1}{2}(x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j) \right) \\
&= \frac{1}{2} \frac{1}{|\Sigma_j^{-1}|} \frac{\partial}{\partial \sigma_{rs}} |\Sigma_j^{-1}| - \frac{1}{2}(x_r - \mu_{jr})(x_s - \mu_{js}) \\
&= \frac{1}{2} |\Sigma_j| \frac{\partial}{\partial \sigma_{rs}} |\Sigma_j^{-1}| - \frac{1}{2}(x_r - \mu_{jr})(x_s - \mu_{js}) \,.
\end{aligned}$$

To evaluate the partial derivative of $|\Sigma_j^{-1}|$ with respect to one of its elements we will use one way of computing the determinate [11]. For example with a general matrix A, by using the cofactor expansion of the determinant about the ith row we can write

$$|A| = a_{i1} C_{i1} + a_{i2} C_{i2} + \cdots + a_{in} C_{in} \,.$$

Where $C_{ij} = (-1)^{i+j}|M_{ij}|$ is the cofactor of the (i,j)th element and M_{ij} is the minor of the (i,j)th element. In that case we see that the partial derivative of $|A|$ with respect to one of its element, say a_{ij} is given by

$$\frac{\partial |A|}{\partial a_{ij}} = C_{ij} \,. \tag{231}$$

From this fact we see that

$$\frac{\partial}{\partial \sigma_{rs}} \ln(p(x|C_j;\theta_j) = \frac{1}{2}|\Sigma_j| C_{rs} - \frac{1}{2}(x_r - \mu_{jr})(x_s - \mu_{js}) \,.$$

Thus letting $\mathbf{C}_j$ be the matrix with cofactor elements C_{ij} corresponding to the matrix Σ_j^{-1} we have that in matrix notation the above is

$$\frac{\partial}{\partial \Sigma_j^{-1}} \ln(p(x|C_j;\theta_j) = \frac{1}{2}|\Sigma_j| \mathbf{C}_j - \frac{1}{2}(x - \mu_j)(x - \mu_j)^T \,.$$

The cofactor matrix $\mathbf{C}_j$ is special in that it is related to the inverse of the generating matrix (here Σ_j^{-1}). One can show [11] that

$$(\Sigma_j^{-1})^{-1} = \frac{\mathbf{C}_j^T}{|\Sigma_j^{-1}|} \,.$$

Thus $\mathbf{C}_j^T = |\Sigma_j^{-1}|\Sigma_j$ and since Σ_j is a symmetric matrix this gives that the product $|\Sigma_j|\mathbf{C}_j$ in $\frac{\partial}{\partial \sigma_{rs}} \ln(p(x|C_j;\theta_j)$ above simplifies to $|\Sigma_j|\mathbf{C} = \Sigma_j$ and we have

$$\frac{\partial}{\partial \Sigma_j^{-1}} \ln(p(x|C_j;\theta_j) = \frac{1}{2}\Sigma_j - \frac{1}{2}(x - \mu_j)(x - \mu_j)^T \,. \tag{232}$$

Now when we put this into Equation 230 we get

$$\frac{1}{2}\Sigma_j \sum_{i=1}^{N} P(C_j|x_i;\Theta(t)) - \frac{1}{2}\sum_{i=1}^{N} P(C_j|x_i;\Theta(t))(x_i - \mu_j)(x_i - \mu_j)^T = 0 \,,$$

or when we solve for Σ_j we get

$$\Sigma_j = \frac{\sum_{i=1}^{N} P(C_j|x_i;\Theta(t))(x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^{N} P(C_j|x_i;\Theta(t))} \,. \tag{233}$$

The expression given in the book.

Notes on Example 14.4 (hard clustering vs. fuzzy clustering)

Recall the expression for the fuzzy objective function $J_q(\theta, U)$ which is given by

$$J_q(\theta, U) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^q d(x_i, \theta_j) \,.$$

In this example for the hard clustering we take $U_{\text{hard}} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$ so that J becomes

$$\begin{aligned} J_q^{\text{hard}}(\theta, U) &= \sum_{i=1}^{4}\sum_{j=1}^{2} u_{ij}^q d(x_i, \theta_j) = d(x_1,\theta_1) + d(x_2,\theta_1) + d(x_3,\theta_2) + d(x_4,\theta_2) \\ &= 1+1+1+1 = 4\,, \end{aligned}$$

which has a value independent of q. Now take $q = 1$ and fuzzy clustering where we find

$$\begin{aligned} J_1^{\text{fuzzy}}(\theta, U) &= u_{11}d(x_1,\theta_1) + u_{12}d(x_1,\theta_2) \\ &\quad + u_{21}d(x_2,\theta_1) + u_{22}d(x_2,\theta_2) \\ &\quad + u_{31}d(x_3,\theta_1) + u_{32}d(x_3,\theta_2) \\ &\quad + u_{41}d(x_4,\theta_1) + u_{42}d(x_4,\theta_2)\,. \end{aligned}$$

From facts like $d(x_1,\theta_1) = d(x_2,\theta_1) = 1$ and $d(x_1,\theta_2) = \sqrt{1+3^2} = \sqrt{10} = d(x_2,\theta_2)$ the above is given by

$$\begin{aligned} J_1^{\text{fuzzy}}(\theta, U) &= u_{11} + u_{12}\sqrt{10} + u_{21} + u_{22}\sqrt{10} + u_3\sqrt{10} + u_{32} + u_{41}\sqrt{10} + u_{42} \\ &= \sum_{i=1}^{2}(u_{i1} + u_{i2}\sqrt{10}) + \sum_{i=3}^{4}(u_{i1}\sqrt{10} + u_{i2})\,. \end{aligned}$$

Since $0 \le u_{i1} \le 1$ and $u_{i2} = 1 - u_{i2}$ we can write the argument of the first summation as

$$u_{i1} + u_{i2}\sqrt{10} = \sqrt{10} + (1-\sqrt{10})u_{i1}\,.$$

Since $1 - \sqrt{10} < 0$ to make this expression as small as possible we take u_{i1} as large as possible. Thus take $u_{i1} = 1$ and we get

$$u_{i1} + u_{i2}\sqrt{10} \ge 1\,.$$

For the argument of the second summation we have

$$u_{i1}\sqrt{10} + u_{i2} = u_{i1}\sqrt{10} + (1-u_{i1}) = 1 + (\sqrt{10}-1)u_{i1}\,.$$

Since $\sqrt{10} - 1 > 0$ to make this expression as small as possible we take u_{i1} as small as possible. Thus take $u_{i1} = 0$ and we get

$$u_{i1}\sqrt{10} + u_{i2} \ge 1\,.$$

Thus we have shown that

$$J_1^{\text{fuzzy}}(\theta, U) \ge \sum_{i=1}^{2} 1 + \sum_{i=3}^{4} 1 = 4\,.$$

If we next take $q = 2$ then our fuzzy objective function is given by

$$J_2^{\text{fuzzy}}(\theta, U) = \sum_{i=1}^{2}(u_{i1}^2 + u_{i2}^2\sqrt{10}) + \sum_{i=3}^{4}(u_{i1}^2\sqrt{10} + u_{i2}^2)\,.$$

Consider the argument of the first sum $u_{i1}^2 + u_{i2}^2\sqrt{10}$ as a function of u_{i2} or the expression $(1-u_{i2})^2+u_{i2}^2\sqrt{10}$. If we seek to find the extrema of this function by taking the u_{i2} derivative of it we see that we need to solve

$$2(1-u_{i2})(-1) + 2u_{i2}\sqrt{10} = 0\,,$$

for u_{i2}. A second derivative of this expression is given by

$$2 + 2\sqrt{10} > 0\,,$$

showing that the extrema found would be a *minimum*. Thus the *maximum* of this expression happens at the end points of the u_{i2} domain. If we consider the two end points $u_{i2} = 0$ and $u_{i2} = 0.48$ we get

$$(1-u_{i2})^2 + u_{i2}^2\sqrt{10}\Big|_{u_{i2}=0} = 1$$
$$(1-u_{i2})^2 + u_{i2}^2\sqrt{10}\Big|_{u_{i2}=0.48} = 0.998\,,$$

where we see that we have a maximum value of one over the interval $u_{i2} \in [0, 0.48]$. If the same expression holds for the argument of the second sum we have

$$J_2^{\text{fuzzy}}(\theta, U) \leq \sum_{i=1}^{2} 1 + \sum_{i=3}^{4} 1 \leq 4\,,$$

as we were to show. Thus fuzzy clustering with $q = 2$ produces an objective function of lesser value that of hard clustering.

If we next take $q = 3$ then our fuzzy objective function is given by

$$J_3^{\text{fuzzy}}(\theta, U) = \sum_{i=1}^{2}(u_{i1}^3 + u_{i2}^3\sqrt{10}) + \sum_{i=3}^{4}(u_{i1}^3\sqrt{10} + u_{i2}^3)\,.$$

Consider the argument of the first sum $u_{i1}^3 + u_{i2}^3\sqrt{10}$ as a function of u_{i2} or the expression $(1-u_{i2})^3+u_{i2}^3\sqrt{10}$. If we seek to find the extrema of this function by taking the u_{i2} derivative of it we see that we need to solve

$$3(1-u_{i2})^2(-1) + 3u_{i2}^2\sqrt{10} = 0\,,$$

for u_{i2}. A second derivative of this expression is given by

$$6(1-u_{i2}) + 6\sqrt{10}u_{i2} = 6 + 6(\sqrt{10}-1)u_{i2} > 0\,,$$

showing that the extrema found would be a minimum. Thus the maximum of this expression happens at the end points of the u_{i2} domain. If we consider the two end points $u_{i2} = 0$ and $u_{i2} = 0.67$ we get

$$(1-u_{i2})^3 + u_{i2}^3\sqrt{10}\Big|_{u_{i2}=0} = 1$$
$$(1-u_{i2})^3 + u_{i2}^3\sqrt{10}\Big|_{u_{i2}=0.67} = 0.987\,,$$

again showing that the maximum is given by the value 1. The same expression holds for the argument of the second sum and in the same way as before we have

$$J_3^{\text{fuzzy}}(\theta, U) \leq 4\,,$$

as we were to show.

Notes on the minimization of $J_q(\theta, U)$)

Recall that the objective function J_q that we want to minimize is given by

$$J_q(\theta; U) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^q d(x_i, \theta_j) \,. \tag{234}$$

where θ_j are "cluster representatives" (numerical parameters that determine the type of cluster) and U is a matrix with (i,j)th element $u_j(x_i)$ or the membership of the x_i sample $i = 1, 2, \cdots, N$ in the jth cluster $j = 1, 2, \cdots m$. We would like to minimize this with respect to both the cluster representatives θ_j and the sample memberships U. Since the cluster memberships are constrained such that $u_{ij} \in [0, 1]$ and

$$\sum_{j=1}^{m} u_{ij} = 1 \quad \text{for} \quad i = 1, 2, \cdots N \,,$$

we must use the methods of constrained optimizing. In this direction we introduce Lagrange multipliers λ_i (one for each sample) and the Lagrangian $\mathcal{J}$ defined by

$$\mathcal{J}(\theta, U) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^q d(x_i; \theta_j) - \sum_{i=1}^{N} \lambda_i \left(\sum_{j=1}^{m} u_{ij} - 1 \right) \,. \tag{235}$$

With this Lagrangian we first take the derivative of $\mathcal{J}$ with respect to the (r, s) element of U, set the result equal to zero and solve for u_{rs}. We find the derivative set equal to zero given by

$$\frac{\partial \mathcal{J}}{\partial u_{rs}} = q u_{rs}^{q-1} d(x_r, \theta_s) - \lambda_r = 0 \,.$$

or u_{rs} given by

$$u_{rs} = \left(\frac{\lambda_r}{q d(x_r, \theta_s)} \right)^{\frac{1}{q-1}} \,. \tag{236}$$

When we put this into the constraints

$$\sum_{j=1}^{m} u_{ij} = 1 \quad \Rightarrow \quad \left(\frac{\lambda_r}{q} \right)^{\frac{1}{q-1}} \sum_{j=1}^{m} \frac{1}{d(x_r, \theta_j)^{\frac{1}{q-1}}} = 1 \,,$$

or

$$\left(\frac{\lambda_r}{q} \right)^{\frac{1}{q-1}} = \frac{1}{\sum_{j=1}^{m} \left(\frac{1}{d(x_r, \theta_j)} \right)^{\frac{1}{q-1}}} \,, \tag{237}$$

and solving for λ_r we get

$$\lambda_r = \frac{q}{\left(\sum_{j=1}^{m} \left(\frac{1}{d(x_r, \theta_j)} \right)^{\frac{1}{q-1}} \right)^{q-1}} \tag{238}$$

When we put this expression for λ_r in the form of Equation 237 into Equation 236 we see that u_{rs} is given by

$$\begin{aligned} u_{rs} &= \frac{1}{d(x_r,\theta_s)^{\frac{1}{q-1}}} \left(\frac{1}{\sum_{j=1}^{m} \left(\frac{1}{d(x_r,\theta_j)} \right)^{\frac{1}{q-1}}} \right) \\ &= \frac{1}{\sum_{j=1}^{m} \left(\frac{d(x_r,\theta_s)}{d(x_r,\theta_j)} \right)^{\frac{1}{q-1}}} \end{aligned} \tag{239}$$

which holds for $r = 1, \cdots N$ and $s = 1, 2, \cdots m$. Now that we have found the optimal U given a fixed values for θ_j we now search for the optimal θ_j given values of U. To do this we need to solve $\frac{\partial \mathcal{J}}{\partial \theta_j} = 0$ for θ_j. We find this last equation given by

$$\frac{\partial \mathcal{J}}{\partial \theta_j} = \sum_{i=1}^{N} u_{ij}^q \frac{\partial d(x_i,\theta_j)}{\partial \theta_j} = 0 \,. \tag{240}$$

Unless we specify a functional form for $d(\cdot,\cdot)$ we cannot go further.

The algorithm for finding the full solution (both U and θ_j for $j = 1, 2, \cdots, m$) then becomes to iterate between the two routines above. One way to do this is to pick initial values for the cluster representatives $\theta_j(0)$ for each j and then use Equation 239 to compute $u_{ij}(1)$. With these new values for $u_{ij}(1)$ we solve for $\theta_j(1)$ in Equation 240. This procedure is iterated by stepping from $\theta_j(t)$ to $u_{ij}(t)$ to $\theta_j(t+1)$ to $u_{ij}(t+1)$ etc. until convergence.

Notes on the minimization of $J_q(\theta, U)$ with point represented clusters)

We can now specify Equation 240 to some special cluster types and representatives. As a first case we consider θ_j to be a point representation of a cluster (and thus it is simply a vector of dimension l) and take $d(x_i,\theta_j)$ to be a typical dissimilarity metric. Two simple cases are

- A Mahalanobis type distance for $d(x_i,\theta_j)$

$$d(x_i,\theta_j) = (\theta_j - x_i)^T A (\theta_j - x_i) \,. \tag{241}$$

  Then the derivative of $d(x_i,\theta_j)$ with respect to θ_j is

$$\frac{\partial d(x_i,\theta_j)}{\partial \theta_j} = 2A(\theta_j - x_i) \,.$$

  With this Equation 240 is

$$\sum_{i=1}^{N} u_{ij}^q (t-1) A(\theta_j - x_i) = 0 \,.$$

or by multiplying by A^{-1} we get

$$\left(\sum_{i=1}^{N} u_{ij}^q(t-1)\right)\theta_j - \sum_{i=1}^{N} u_{ij}^q(t-1)x_i = 0\,,$$

so

$$\theta_j = \frac{\sum_{i=1}^{N} u_{ij}^q(t-1)x_i}{\sum_{i=1}^{N} u_{ij}^q(t-1)}\,. \tag{242}$$

- A Minkowski distance for $d(x_i, \theta_j)$

$$d(x_i, \theta_j) = \left(\sum_{k=1}^{l} |x_{ik} - \theta_{jk}|^p\right)^{\frac{1}{p}}\,. \tag{243}$$

Then the derivative of $d(x_i, \theta_j)$ with respect to the rth component of θ_j is

$$\begin{aligned}\frac{\partial d(x_i, \theta_j)}{\partial \theta_{jr}} &= \frac{1}{p}\left(\sum_{k=1}^{l}(x_{ik} - \theta_{jk})^p\right)^{\frac{1}{p}-1} p\,(x_{ir} - \theta_{jr})^{p-1} \\ &= \frac{(\theta_{jr} - x_{ir})^{p-1}}{\left(\sum_{k=1}^{l}(x_{ik} - \theta_{jk})^p\right)^{\frac{1}{p}-1}}\,. \end{aligned} \tag{244}$$

for $r = 1, 2, \cdots, l$. This means that using Equation 240 specified to the θ_{jr} derivative we must solve

$$\sum_{i=1}^{N} u_{ij}^q(t-1)\left[\frac{(\theta_{jr} - x_{ir})^{p-1}}{\left(\sum_{k=1}^{l}(x_{ik} - \theta_{jk})^p\right)^{\frac{1}{p}-1}}\right] = 0 \quad \text{for} \quad r = 1, 2, \cdots, l\,.$$

Since there are l equations above and l components of θ_j we expect there to be a unique solution.

Notes on quadratic surfaces as representatives

In the quadratic surface representation

$$x^T A x + b^T x + c = 0\,, \tag{245}$$

since x is of dimension l and the matrix A is symmetric it therefore has $\frac{1}{2}l(l-1)$ unique elements in its upper (or lower) triangular part. There are l additional elements on its diagonal. Thus to specify the unique values of the matrix A we have to specify

$$l + \frac{1}{2}l(l-1) = \frac{1}{2}l(l+1)\,,$$

numbers. There are l numbers needed to specify the vector b and 1 number needed to specify the number c. Thus if we recast the quadratic surface representation above into the form

$$q(x)^T p = 0\,, \tag{246}$$

the vector p must have

$$\frac{1}{2}l(l+1)+l+1\,,$$

numbers.

Notes on computing the perpendicular distance

In this section of these notes we discuss how to evaluate the perpendicular distance between a point x and a quadratic surface Q defined by

$$d_p^2(x,Q)=\min_z ||x-z||^2\,, \tag{247}$$

subject to the constraint on z such that $z^TAz+b^Tz+c=0$. We form the Lagrangian $\mathcal{D}$

$$\mathcal{D}(x,Q)=||x-z||^2-\lambda(z^TAz+b^Tz+c)\,. \tag{248}$$

and take derivatives in the normal way. The z derivative gives

$$\frac{\partial\mathcal{D}}{\partial z}=2(x-z)-2\lambda Az-\lambda b=0\,,$$

On expanding

$$2x-(2I+2\lambda A)z-\lambda b=0\,,$$

or

$$z=(2I+2\lambda A)^{-1}(2x-\lambda b)=\frac{1}{2}(I+\lambda A)^{-1}(2x-\lambda b)\,. \tag{249}$$

We then put this into $z^TAz+b^Tz+c=0$ to get a polynomial in λ which gives several roots for λ_k. For each root λ_k we can evaluate $z_k=z(\lambda_k)$ using Equation 249 and then select the value for $d_p^2(x,Q)$ that gives the smallest value

$$d_p^2(x,Q)=\min_{\lambda_k}||x-z(\lambda_k)||^2\,.$$

Notes on adaptive fuzzy C-shells (AFCS) clustering algorithms

We start with our objective function

$$J_{nr}(\theta,U)=\sum_{i=1}^{N}\sum_{j=1}^{m}u_{ij}^q d_{nr}^2(x_i,Q_j)\,, \tag{250}$$

and set derivatives equal to zero to get the equations we must solve for optimal solutions. We start with $\frac{\partial}{\partial c_j}$ of $d_{nr}^2(x,;c_j,A_j)$ where we get

$$\begin{aligned}\frac{\partial}{\partial c_j}d_{nr}^2(x;c_j,A_j) &= 2\left(((x-c_j)^TA_j(x-c_j))^{1/2}-1\right)\frac{\partial}{\partial c_j}\left[(x-c_j)A_j(x-c_j)\right]^{1/2}\\ &= d_{nr}(x;c_j,A_j)\left[(x-c_j)A_j(x-c_j)\right]^{-1/2}\frac{\partial}{\partial c_j}(x-c_j)^TA_j(x-c_j)\\ &= \frac{d_{nr}(x;c_j,A_j)}{\left[(x-c_j)A_j(x-c_j)\right]^{1/2}}(2A_j(c_j-x))\\ &= -2\frac{d_{nr}(x;c_j,A_j)}{\phi(x;c_j,A_j)}A(x-c_j)\,. \end{aligned} \tag{251}$$

Using the definition of $\phi^2(x;c_j,A_j)$. Next we let a_{rs} be the (r,s) element of A_j. Then $\frac{\partial}{\partial a_{rs}}d_{nr}^2(x;c_j,A_j)$ is given by

$$\begin{aligned}\frac{\partial}{\partial a_{rs}}d_{nr}^2(x;c_j,A_j) &= 2d_{nr}(x;c_j,A_j)\left(\frac{1}{2}\right)\frac{1}{\phi(x;c_j,A_j)}\frac{\partial}{\partial a_{rs}}(x-c_j)A_j(x-c_j)\\ &= \frac{d_{nr}(x;c_j,A_j)}{\phi(x;c_j,A_j)}(x_r-c_{jr})(x_s-c_{js})\,.\end{aligned}$$

When we write this in matrix form we get

$$\frac{\partial}{\partial A_j}d_{nr}^2(x;c_j,A_j) = \frac{d_{nr}(x,c_j,A_j)}{\phi(x;c_j,A_j)}(x-c_j)(x-c_j)^T\,. \tag{252}$$

We then put Equation 251 and 252 into the "parameter update" step in the Generalized Fuzzy Algorithmic Scheme" (GFAS) which for reminder is the expression

$$\sum_{i=1}^{N}u_{ij}^q(t-1)\frac{\partial}{\partial\theta_j}d(x_i;\theta_j)=0\,,$$

we get the parameter update step for this algorithm quoted in the book.

Notes on the fuzzy C-ellipsoid shells (FCES) clustering algorithms

Using the definition of $\phi^2=(x-c)^TA(x-c)$ with Equations 268 and 269 developed below we have that

$$d_r^2(x;c,A)=(1-a)^2||x-c||^2=\left(1-\frac{1}{\phi}\right)^2||x-c||^2\,.$$

Using this expression we will take the derivatives needed for the parameter updating algorithm. We find

$$\begin{aligned}\frac{\partial}{\partial c}d_r^2(x;c,A) &= 2\left(1-\frac{1}{\phi}\right)\left(\frac{1}{\phi^2}\right)\frac{\partial\phi}{\partial c}||x-c||^2+\left(1-\frac{1}{\phi}\right)^2(-2(x-c))\\ &= \frac{2}{\phi^2}\left(1-\frac{1}{\phi}\right)\frac{\partial\phi}{\partial c}||x-c||^2-2\left(1-\frac{1}{\phi}\right)^2(x-c)\,.\end{aligned}$$

Now from the definition of ϕ we have that

$$\frac{\partial \phi}{\partial c} = \frac{1}{2}\frac{1}{\phi}(2A(c-x)) = -\frac{1}{\phi}A(x-c)\,.$$

Using this we get for the derivative $\frac{\partial}{\partial c}d_r^2(x;c,A)$

$$\begin{aligned}\frac{\partial}{\partial c}d_r^2(x;c,A) &= -\frac{2}{\phi^3}\left(1-\frac{1}{\phi}\right)||x-c||^2A(x-c) - 2\left(1-\frac{1}{\phi}\right)(x-c) \\ &= -\frac{2}{\phi^4}\left(1-\phi\right)||x-c||^2A(x-c) - 2\left(1-\frac{1}{\phi}\right)(x-c)\,.\end{aligned}$$

Thus one of the parameter updating equations

$$\sum_{i=1}^{N} u_{ij}^q(t-1)\frac{\partial}{\partial c}d_r^2(x;c,A) = 0\,,$$

when we divide by 2 and put back in the index j specifying the cluster becomes

$$\sum_{i=1}^{N} u_{ij}^q(t-1)\left[\frac{||x-c||^2(1-\phi)}{\phi^4}A_j - \left(1-\frac{1}{\phi}\right)^2 I\right](x-c_j) = 0\,, \tag{253}$$

the same equation as in the book. Next we need to evaluate $\frac{\partial}{\partial A}d_r^2(x;c,A)$ where we find

$$\frac{\partial}{\partial A}d_r^2(x;c,A) = 2\left(1-\frac{1}{\phi}\right)\left(\frac{1}{\phi^2}\right)\frac{\partial \phi}{\partial A}||x-c||^2\,.$$

Since

$$\frac{\partial \phi}{\partial A} = \frac{1}{2}(\phi^2)^{-1/2}\frac{\partial}{\partial A}((x-c)^TA(x-c)) = \frac{1}{2}\frac{1}{\phi}(x-c)(x-c)^T\,,$$

we have that

$$\frac{\partial}{\partial A}d_r^2(x;c,A) = \left(1-\frac{1}{\phi}\right)\left(\frac{1}{\phi^3}\right)||x-c||^2(x-c)^T(x-c)^T\,.$$

Thus the second parameter updating equations

$$\sum_{i=1}^{N} u_{ij}^q(t-1)\frac{\partial}{\partial A}d_r^2(x;c,A) = 0\,,$$

with the index j specifying the cluster becomes

$$\sum_{i=1}^{N} u_{ij}^q(t-1)\left(\frac{\phi-1}{\phi^4}\right)||x-c||^2(x-c)(x-c)^T = 0\,, \tag{254}$$

the same equation as in the book.

Notes on the fuzzy C-quadric shells (FCQS) algorithm

For this clustering our distance function for a point x and a cluster Q is given by the algebraic distance

$$d_a^2(x;Q) = (x^T A x + b^T x + c)^2 = p^T M(x) p\,,$$

where we have written M as a function of x since it depends on the point x where we want to evaluate this distance. Here p is a parameter vector that determines the "shape" of the quadric that we are considering. We want to impose the constraint on the vector p_j of the form

$$\left|\left| \sum_{k=1}^{l} p_{jk}^2 + \frac{1}{2} \sum_{k=l+1}^{r} p_{jk}^2 \right|\right|^2 = 1\,. \tag{255}$$

Since the quadratic we seek to describe is given by the equation $q(x)^T p = 0$ we can modify the definitions of q and p by introducing $\sqrt{2}$ so that the inner product remains unchanged and so that we can explicitly introduce the constraint Equation 255. To this end we introduce the vector a, b, r, and t such that

$$\tilde{p} = \left[\begin{array}{cc} a^T & b^T \end{array}\right]^T \quad \text{and} \quad \tilde{q} = \left[\begin{array}{cc} r^T & t^T \end{array}\right]^T\,,$$

then $q^T p = \tilde{q}^T \tilde{p} = r^T a + t^T b$ and $p_j^T M_i p_j = \tilde{p}_j \tilde{M}_i \tilde{p}_j$. The cost function (without any constraint) for the FCQS algorithm can then be written as

$$\begin{aligned} J &= \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^q \tilde{p}_j \tilde{M}(x_i) \tilde{p}_j \\ &= \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^q \left[\begin{array}{cc} a_j^T & b_j^T \end{array}\right] \left[\begin{array}{cc} r_i r_i^T & r_i t_i^T \\ t_i r_i^T & t_i t_i^T \end{array}\right] \left[\begin{array}{c} a_j \\ b_j \end{array}\right]\,. \end{aligned} \tag{256}$$

To enforce the constraint from Equation 255 we need to modify J to include this constraint by adding

$$-\sum_{j=1}^{m} \lambda_j (||a_j||^2 - 1)\,,$$

to get $\mathcal{J}_a(\theta, J)$. Expanding the block inner product in Equation 256 we get

$$\sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^q \left[a_j^T r_i r_i^T a_j + b_j^T t_i t_i^T b_j + 2 a_j^T r_i t_i^T b_j\right]\,.$$

Consider now $\frac{\partial}{\partial b_j} \mathcal{J}_a$. When we recall that for symmetric A

$$\frac{\partial}{\partial x}(y^T A x) = A^T y \quad \text{and} \quad \frac{\partial}{\partial x}(x^T A x) = 2Ax\,,$$

we then find $\frac{\partial}{\partial b_j} \mathcal{J}_a$ given by

$$\frac{\partial}{\partial b_j} \mathcal{J}_a = \sum_{i=1}^{N} u_{ij}^q (2 T_i b_j + 2 S_i a_j)\,.$$

Let

$$H_j \equiv \sum_{i=1}^{N} u_{ij}^q T_i \quad \text{and} \quad G_j \equiv \sum_{i=1}^{N} u_{ij}^q S_i \,,$$

and setting $\frac{\partial}{\partial b_j}\mathcal{J}_a$ equal to zero we get for b_j

$$b_j = -H_j^{-1} G_j a_j \,. \tag{257}$$

Now we need to evaluate $\frac{\partial}{\partial a_j}\mathcal{J}_a$. When we recall that

$$\frac{\partial}{\partial x}(x^T A y) = A y \,,$$

we get for this derivative

$$\frac{\partial}{\partial a_j}\mathcal{J}_a = \sum_{i=1}^{N} u_{ij}^q \left[2R_i a_j + 2r_i t_i^T b_j\right] - \lambda_j (2a_j) = \sum_{i=1}^{N} u_{ij}^q \left[2R_i a_j + 2S_i^T b_j\right] - 2\lambda_j a_j \,.$$

Let

$$F_j \equiv \sum_{i=1}^{N} u_{ij}^q R_i \,,$$

and set $\frac{\partial}{\partial a_j}\mathcal{J}_a$ equal to zero we get $F_j a_j - \lambda_j a_j = -G_j^T b_j$. Since we know b_j using Equation 257 we get $F_j a_j - \lambda_j a_j = G_j^T H_j^{-1} G_j a_j$ or

$$(F_j - G_j^T H_j^{-1} G_j) a_j = \lambda_j a_j \,. \tag{258}$$

Thus λ_j is an eigenvalue of the matrix $F_j - G_j^T H_j^{-1} G_j$ and a_j is the corresponding eigenvector with length 1 due to the constraint Equation 255. We now specify how to pick λ_j from all of the eigenvalues of $F_j - G_j^T H_j^{-1} G_j$. Since we know that $b_j = -H_j^{-1} G_j a_j$ when we are at the optimum solution the value of $\mathcal{J}_a$ will be given by the value of J at this optimal solution where J is given by Equation 256. This is because the constraints must all be satisfied (and therefore vanish) at the optimum solution and won't contribute to the value of the cost function. The argument of the summation in Equation 256 using the value for b_j calculated above is given by

$$a_j^T R_i a_j + b_j^T T_j b_j + 2 a_j^T S_i^T b_j = a_j^T R_j a_j + a_j^T G_j^T H_j^{-T} T_i H_j^{-1} G_j a_j - 2 a_j^T S_i^T H_j^{-1} G_j a_j \,.$$

When we multiply this by u_{ij}^q and sum over i we get

$$a_j^T \left(\left(\sum_{i=1}^{N} u_{ij}^q R_i \right) + G_j^T H_j^{-T} \left(\sum_{i=1}^{N} u_{ij}^q T_i \right) H_j^{-1} G_j - 2 \left(\sum_{i=1}^{N} u_{ij}^q S_i^T \right) H_j^{-1} G_j \right) a_j \,,$$

or recalling our previous definitions of F_j, G_j, and H_j we have

$$a_j^T \left(F_j + G_j^T H_j^{-T} H_j H_j^{-1} G_j - 2 G_j^T H_j^{-1} G_j \right) a_j \,.$$

Since H_j is a symmetric matrix this becomes

$$a_j^T (F_j - G_j^T H_j^{-1} G_j) a_j \,.$$

If a_j is an eigenvector (with unit norm) of the matrix $F_j - G_j H_j^{-1} G_j$ then this inner product is given by

$$\lambda_j a_j^T a_j = \lambda_j \,.$$

Thus to make the sum of these terms over j as small as possible (to minimize J) we pick λ_j as small as possible. This motivates picking the smallest values for the eigenvalues of $F_j - G_j H_j^{-1} G_j$ in this algorithm.

Notes on hyperplane representatives (the Gustafson-Kessel algorithm)

In this formulation the distance we use is given by

$$d^2_{\text{GK}}(x;\theta_j) = |\Sigma_j|^{1/l}(x-c_j)^T \Sigma_j^{-1}(x-c_j)\,, \tag{259}$$

and our objective function is given by

$$J_{\text{GK}}(\theta;U) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^q d^2_{\text{GK}}(x_i;\theta_j)\,.$$

To find the minimum of this expression we take derivatives with respect to c_j and Σ_j, set the results equal to zero and solve for these expressions. Taking the derivative of $J_{\text{GK}}(\theta;U)$ with respect to c_j we have

$$\frac{\partial}{\partial c_j} J_{\text{GK}}(\theta;U) = \sum_{i=1}^{N} u_{ij}^q \frac{\partial}{\partial c_j} d^2_{\text{GK}}(x_i;c_j,\Sigma_j)\,.$$

Note in the above expression that since we are explicitly specifying the jth cluster in this derivative (and in the subsequent Σ_j derivative) the derivative of the other terms with different indices are zero. Thus we can drop the j index on c and Σ and only consider how to evaluate $\frac{\partial}{\partial c} d^2_{\text{GK}}(x;c,\Sigma)$. This procedure makes the notation cleaner. Thus

$$\frac{\partial}{\partial c} d^2_{\text{GK}}(x;c,\Sigma) = |\Sigma|^{1/l}(2\Sigma^{-1}(c-x)) = -2|\Sigma|^{1/l}\Sigma^{-1}(x-c)\,. \tag{260}$$

When we put this last expression into the previous one, equate the result to zero and cancel common terms, we get

$$\sum_{i=1}^{N} u_{ij}^q (x_i - c) = 0\,,$$

or solving for c we find

$$c = \frac{\sum_{i=1}^{N} u_{ij}^q x_i}{\sum_{i=1}^{N} u_{ij}^q}\,. \tag{261}$$

This depends on j via the right-hand-side. We now need to evaluate the optimal value for Σ via taking the derivative of J_{GK} with respect to Σ. Rather than evaluate $\frac{\partial}{\partial \Sigma} d^2_{\text{GK}}(x;c,\Sigma)$ directly we will evaluate $\frac{\partial}{\partial \Sigma^{-1}} d^2_{\text{GK}}(x;c,\Sigma)$ or the derivative of d^2_{GK} with respect to Σ^{-1}. To do this we first write $d^2_{\text{GK}}(x;c,\Sigma)$ as

$$d^2_{\text{GK}}(x;c,\Sigma) = |\Sigma^{-1}|^{-1/l}(x-c)^T\Sigma^{-1}(x-c)\,.$$

Let f_{rs} be the (r,s) element of Σ^{-1}. Then

$$\frac{\partial}{\partial f_{rs}} d^2_{\text{GK}}(x;c,\Sigma) = -\frac{1}{l}|\Sigma^{-1}|^{-\frac{1}{l}-1}\frac{\partial|\Sigma^{-1}|}{\partial f_{rs}}(x-c)^T\Sigma^{-1}(x-c) + |\Sigma^{-1}|^{-1/l}(x_r-c_r)(x_s-c_s)\,.$$

As we discussed on Page 171 and expressed via Equation 231 we have

$$\frac{\partial|\Sigma^{-1}|}{\partial f_{rs}} = \sigma_{rs}\,,$$

where σ_{rs} is the (r,s) cofactor of the matrix Σ^{-1}. When we put this into the above and then consider the *matrix* form of the above expression we get

$$\frac{\partial}{\partial \Sigma^{-1}} d^2_{\text{GK}}(x;c,\Sigma) = -\frac{1}{l}|\Sigma^{-1}|^{-\frac{1}{l}-1}\mathbf{C}(x-c)^T\Sigma^{-1}(x-c) + |\Sigma^{-1}|^{-1/l}(x-c)(x-c)^T\,,$$

where $\mathbf{C}$ is the matrix of cofactors of Σ^{-1} i.e. the (r,s) element of $\mathbf{C}$ is σ_{rs}. Then again from earlier we know that the cofactor matrix $\mathbf{C}$ and the inverse of Σ^{-1} are related as

$$(\Sigma^{-1})^{-1} = \frac{\mathbf{C}^T}{|\Sigma^{-1}|} \quad \text{or} \quad \mathbf{C}^T = |\Sigma^{-1}|\Sigma\,.$$

Since Σ is symmetric so is $\mathbf{C}$ (just take the transpose of the previous equation) and we have

$$\frac{\partial}{\partial \Sigma^{-1}} d^2_{\text{GK}}(x;c,\Sigma) = |\Sigma^{-1}|^{-1/l}\left[-\frac{1}{l}(x-c)^T\Sigma^{-1}(x-c)\Sigma + (x-c)(x-c)^T\right]\,.$$

Setting $\frac{\partial}{\partial\Sigma^{-1}} J^2_{\text{GK}}(\theta;U)$ equal to zero using this result we get

$$\frac{1}{l}\sum_{i=1}^N u_{ij}^q (x_i-c)^T\Sigma^{-1}(x_i-c)\Sigma = \sum_{i=1}^N u_{ij}^q (x_i-c)(x_i-c)^T\,. \tag{262}$$

The book then presents

$$\Sigma = \frac{\sum_{i=1}^N u_{ij}^q (x_i-c)(x_i-c)^T}{\sum_{i=1}^N u_{ij}^q}\,, \tag{263}$$

as the solution to Equation 262.

Warning: Note I was not able to derive the given expression for Σ. If anyone knows how to get this expression please email me.

Notes on possibilistic clustering

For the cost function in the possibilistic framework of

$$J(\theta;U) = \sum_{i=1}^N\sum_{j=1}^m u_{ij}^q d(x_i;\theta_j) + \sum_{j=1}^m \eta_j \sum_{i=1}^N (1-u_{ij})^q\,, \tag{264}$$

setting the u_{ij} derivative equal to zero gives

$$\frac{\partial}{\partial u_{ij}} J(\theta; U) = q u_{ij}^{q-1} d(x_i; \theta_j) - q\eta_j (1 - u_{ij})^{q-1} = 0 \,.$$

Dividing this by u_{ij}^{q-1} and by q we get

$$d(x_i; \theta_j) - \eta_j \left(\frac{1}{u_{ij}} - 1 \right)^{q-1} = 0 \,. \tag{265}$$

Solving for u_{ij} gives

$$u_{ij} = \frac{1}{1 + \left(\frac{d(x_i;\theta_j)}{\eta_j} \right)^{\frac{1}{q-1}}} \,. \tag{266}$$

When we solve Equation 265 for $d(x_i; \theta_j)$ we get

$$d(x_i; \theta_j) = \eta_j \left(\frac{1 - u_{ij}}{u_{ij}} \right)^{q-1} \,.$$

When we put this into

$$J_j = \sum_{i=1}^{N} u_{ij}^q d(x_i; \theta_j) + \eta_j \sum_{i=1}^{N} (1 - u_{ij})^q \,,$$

we get

$$\begin{aligned} J_j &= \eta_j \sum_{i=1}^{N} u_{ij} (1 - u_{ij})^{q-1} + \eta_j \sum_{i=1}^{N} (1 - u_{ij})^q = \eta_j \sum_{i=1}^{N} [u_{ij} + (1 - u_{ij})](1 - u_{ij})^{q-1} \\ &= \eta_j \sum_{i=1}^{N} (1 - u_{ij})^{q-1} \,. \end{aligned} \tag{267}$$

Problem Solutions

Problem 14.1 (a known covariance matrix in the GMDAS)

If the covariance matrix is *known* then we don't need to take the derivative with respect to Σ_i in the M-step of the expectation maximization (EM) algorithm. Thus the Σ_i update step drops away and there is only the μ_i update step.

Problem 14.2 (GMDAS when the covariance matrices are diagonal)

If the covariance matrix is diagonal then

$$|\Sigma_j| = \prod_{k=1}^{l} \sigma_{jk}^2 \,,$$

so

$$\ln(p(x_i|C_j;\Theta_j)) = -\frac{l}{2}\ln(2\pi) - \sum_{k=1}^{l}\ln(\sigma_{jk}) - \frac{1}{2}\sum_{k=1}^{l}\frac{(x_{ik}-\mu_{jk})^2}{\sigma_{jk}^2}\,.$$

The M-step for estimating θ_j (where θ_j is a vector representing the mean μ_j and the elements of the covariance matrix) is given by (see the notes on Page 171 around the Equation 230).

$$\sum_{i=1}^{N} P(C_j|x_i;\Theta(t))\frac{\partial}{\partial\theta_j}\ln(p(x_i|C_j;\theta_j) = 0\,.$$

The evaluation of $\frac{\partial}{\partial\theta_j}$ does not change for the elements of the mean vector μ_j from before and we need to evaluate $\frac{\partial}{\partial\theta_j}$ for the elements of the covariance matrix. Consider just one term $\frac{\partial}{\partial\sigma_{jk}}$ where we get

$$\frac{\partial}{\partial\sigma_{jk}}\ln(p(x_i|C_j;\theta_j) = -\frac{1}{\sigma_{jk}} + \frac{(x_{ik}-\mu_{jk})^2}{\sigma_{jk}^3}\,.$$

Thus in Equation 230 we get

$$\sum_{i=1}^{N} P(C_j|x_i;\Theta(t))\left(-\frac{1}{\sigma_{jk}} + \frac{(x_{ik}-\mu_{jk})^2}{\sigma_{jk}^3}\right) = 0 \quad \text{for} \quad 1 \le k \le l\,.$$

If we solve for σ_{jk}^2 in the above expression we get

$$\sigma_{jk}^2 = \frac{\sum_{i=1}^{N} P(C_j|x_i;\Theta(t))(x_{ik}-\mu_{jk})^2}{\sum_{i=1}^{N} P(C_j|x_i;\Theta(t))} \quad \text{for} \quad 1 \le k \le l\,.$$

Note that this gives the same result for the diagonal elements as does Equation 233 which computes the full covariance matrix.

Problem 14.4 (running GMDAS on a given data set)

For this problem I considered the second set of points x_i for $i = 9, 10, \cdots, 16$ to be generated by

$$x_i = x_{16-i+1} + 6 \quad \text{for} \quad i = 9, 10, \cdots, 16\,.$$

Thus $x_9 = x_8 + 6$, $x_{10} = x_7 + 6$, etc. This problem is implemented in the MATLAB script `chap_14_prob_4.m`. For this problem we use MATLAB code from the `NETLAB` toolbox for pattern recognition [1]. The EM algorithm finds

$$\mu_1 = \begin{bmatrix} 6.0069 \\ -0.0053 \end{bmatrix} \quad \mu_2 = \begin{bmatrix} 0.0225 \\ 0.0052 \end{bmatrix},$$

for the means and for Σ_i it finds

$$\Sigma_1 = \begin{bmatrix} 2.0047 & -1.4131 \\ -1.4131 & 2.0016 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 2.0886 & 0.0232 \\ 0.0232 & 1.9984 \end{bmatrix}.$$

If one looks at a scatter plot of the initial points one sees that the returned EM results look like they "fit" the generated data shown in Figure 27 (left).

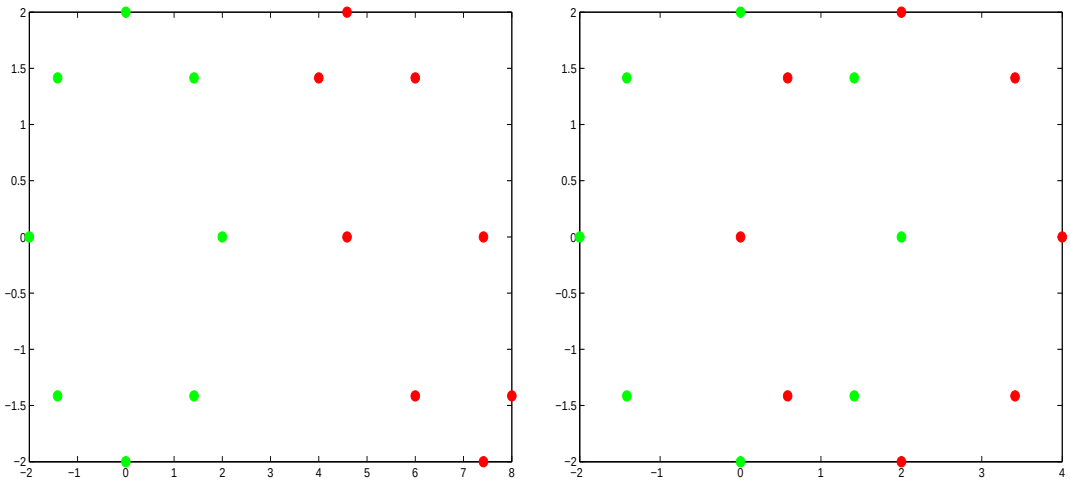

Figure 27: **Left:** Sample data for Problem 4. **Right:** Sample data for Problem 5. Green points correspond to the first eight samples while red points correspond to the remaining eight points.

Problem 14.5 (running GMDAS on another data set)

This problem is implemented in the MATLAB script `chap_14_prob_5.m`. For this problem we again use MATLAB code from the `NETLAB` toolbox for pattern recognition [1]. The EM algorithm finds

$$\mu_1 = \begin{bmatrix} 0.4761 \\ 0.0 \end{bmatrix} \quad \mu_2 = \begin{bmatrix} 3.6173 \\ 0 \end{bmatrix} .$$

and for Σ_i

$$\Sigma_1 = \begin{bmatrix} 1.9393 & 0 \\ 0 & 2.1388 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 0.0777 & 0 \\ 0 & 1.3067 \end{bmatrix} .$$

The data for this problem is shown in Figure 27 (right). In this case the value found don't exactly correspond to the centers of the generated data. This "error" is compounded the closer the clusters are to each other (and how much over lap they have).

Problem 14.7 (the objective function J after clustering)

See the notes on Page 172 for this exercise.

Problem 14.8 (an equivalent relationship to $x^T Ax + b^T x + c = 0$)

When we write out in components the relationship given by Equation 245 we have

$$\sum_{i=1}^{l} \sum_{j=1}^{l} a_{ij} x_i x_j + \sum_{i=1}^{l} b_i x_l + c = 0 .$$

or factoring out the x_i^2 terms

$$\sum_{i=1}^{l} a_{ii}x_i^2 + \sum_{i,j;i\neq j}^{l} a_{ij}x_ix_j + \sum_{i=1}^{l} b_ix_l + c = 0\,.$$

If we expand the second sum by taking $i = 1$ with $j = 2, 3, \ldots l$, then taking $i = 2$ with $j = 1, 3, \cdots, l$, then take $i = 3$ and $j = 1, 2, 4, \cdots, l$ etc we can group the sum of the terms $a_{ij}x_ix_j$ where $i \neq j$ (i.e. the second sum above) as

$$\sum_{i=1}^{l}\sum_{j=l+1}^{l} (a_{ij} + a_{ji})x_ix_j\,.$$

This gives the representation for the vector p as

$$p = \begin{bmatrix} a_{11} & a_{22} & a_{33} & \cdots & a_{ll} & a_{12}+a_{21} & a_{13}+a_{31} & \cdots & a_{l-1,l}+a_{l,l-1} & b_1 & b_2 & \cdots & b_l & c \end{bmatrix}.$$

Problem 14.9 (finding $d_p^2(x, Q)$ via polynomial roots)

When $l = 2$ then A is 2×2 symmetric matrix $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix}$, b is a 2×1 vector, x is a 2×1 vector and then

$$\frac{1}{2}(I + \lambda A)^{-1} = \left(\frac{1}{(1+\lambda a_{11})(1+\lambda a_{22}) - \lambda^2 a_{12}^2}\right)\begin{bmatrix} 1+a_{22}\lambda & -a_{12}\lambda \\ -a_{12}\lambda & 1+a_{11}\lambda \end{bmatrix}.$$

Thus the expression used for z given by Equation 249 is given by

$$\begin{aligned} z &= \frac{1}{2}(I + \lambda A)^{-1}(2x - \lambda b) \\ &= \left(\frac{1}{2((1+\lambda a_{11})(1+\lambda a_{22}) - \lambda^2 a_{12}^2)}\right)\begin{bmatrix} (1+\lambda a_{22})(2x_1 - \lambda b_1) - \lambda a_{12}(2x_2 - \lambda b_2) \\ -\lambda a_{12}(2x_1 - \lambda b_1) + (1+\lambda a_{11})(2x_2 - \lambda b_2) \end{bmatrix}. \end{aligned}$$

When we put this into Equation 245 or $z^TAz + b^Tz + c = 0$ and multiply by the denominator

$$(1+\lambda a_{11})(1+\lambda a_{22}) - \lambda^2 a_{12}^2\,,$$

squared (because of the quadratic term z^TAz) we will obtain a *fourth* order polynomial in λ.

Problem 14.10 (a relationship between $d_{nr}^2(x, Q)$ and $d_r^2(x, Q)$)

Since $d_r^2(x, Q) = ||x - z||^2$ with z on a line from x to c on the hyperellipsoid i.e. $z - c = a(x - c)$ and $(z - c)^TA(z - c) = 1$. When we put this first equation into the second equation we see that

$$a^2(x-c)^TA(x-c) = 1 \quad \text{or} \quad a^2 = \frac{1}{(x-c)^TA(x-c)}\,. \tag{268}$$

Then since $z - c = a(x - c)$ we have

$$||z - c||^2 = a^2||x - c||^2 \quad \text{or} \quad ||x - c||^2 = \frac{1}{a^2}||z - c||^2 \,,$$

with a^2 given by Equation 268. Now consider

$$\begin{aligned} d_r^2(x, Q) &= ||x - z||^2 = ||x - c - a(x - c)||^2 = (1 - a)^2||x - c||^2 \qquad (269) \\ &= \left(\frac{1}{a} - 1\right)^2 ||z - c||^2 \\ &= (((x - c)^T A(x - c))^{1/2} - 1)||z - c||^2 \,, \qquad (270) \end{aligned}$$

as we were to show.

Problem 14.11 (deriving the fuzzy C-ellipsoidal shells algorithm)

See the notes on this algorithm on Page 179.

Problem 14.12 (deriving the fuzzy C-quadric shells algorithm)

See the notes on this algorithm on Page 181.

Problem 14.13 (the modified fuzzy C quadratic shells algorithm)

In this scheme the goal is to use the generalized fuzzy algorithmic scheme (GFAS) with the *perpendicular* distance, $d_p(x, \theta)$, for the degree of membership. Rather than use this distance measure everywhere in the GFAS, in the parameter updating step we replace it with a distance measure that is easier to calculate. Recall that the perpendicular distance is computationally difficult to compute since it relies on finding roots of polynomial equations. Due to the iterative nature of many root finding algorithms that would be used in the parameter updating step we would need to compute $d_p(x, \tilde{\theta})$ for many values of $\tilde{\theta}$. To avoid this computational difficulty in the parameter updating step we will use the *algebraic* cluster-point distance instead. This distance is given by

$$d_a^2(x, Q) = (x^T Ax + b^T x + c)^2 = p^T Mp \,.$$

Problem 14.14 (relationships between the radian and perpendicular distance)

In general the perpendicular distance will be the smaller distance. That is $d_p(x; Q) \leq d_r(x; Q)$ will hold.

Problem 14.15 (spherical clusters)

The AFCS algorithm was derived on Page 178. Here we modify the arguments there to account for the new distance measure for the jth cluster is $d_s^2(x;\theta_j) = (||x-c_j|| - r_j)^2$. Thus the parameters θ_j are given by the vector c_j and the scalar r_j. We start with out criterion function to minimize

$$J(\theta; U) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^q d_s^2(x_i;\theta_j)\,.$$

Then to minimize this expression we take the derivatives with respect to c_j and r_j, set the results equal to zero and then solve (numerically or analytically) for c_j and r_j. We find the c derivative of d_s^2 given by

$$\begin{aligned}
\frac{\partial}{\partial c} d_s^2(x;\theta) &= 2(||x-c|| - r)\frac{\partial}{\partial c}||x-c|| \\
&= 2(||x-c|| - r)\frac{\partial}{\partial c}((x-c)^T(x-c))^{1/2} \\
&= 2(||x-c|| - r)\frac{1}{2}((x-c)^T(x-c))^{-1/2}\frac{\partial}{\partial c}(x-c)^T(x-c) \\
&= \frac{(||x-c|| - r)}{||x-c||}(-2(x-c)) = -2\frac{(||x-c|| - r)}{||x-c||}(x-c)\,.
\end{aligned}$$

We find the r derivative of d_s^2 given by

$$\frac{\partial}{\partial c} d_s^2(x;\theta) = 2(||x-c|| - r)(-1) = -2(||x-c|| - r)\,.$$

Then setting $\frac{\partial J}{\partial r_j}$ equal to zero we have to solve

$$\sum_{i=1}^{N} u_{ij}^q(||x_i - c_j|| - r_j) = 0\,,$$

for r_j. This gives

$$r_j = \frac{\sum_{i=1}^N u_{ij}^q ||x_i - c_j||}{\sum_{i=1}^N u_{ij}^q}\,.$$

Setting $\frac{\partial J}{\partial c_j}$ equal to zero we need to solve

$$\sum_{i=1}^{N} u_{ij}^q \left[\frac{(||x_i - c_j|| - r_j)}{||x_i - c_j||}\right](x_i - c_j) = 0\,,$$

for c_j and r_j given the equation on r_j above. Once we have solved these equations for c_j and r_j we can update the value of $u_{ij}(t)$ as in the Generalized Fuzzy Algorithmic Scheme (GFAS).

Problem 14.16 (the possibilistic algorithm)

If $J_1(\theta, U)$ given by

$$J_1(\theta, U) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij} d(x_i, \theta_j) + \sum_{j=1}^{m} \eta_j \sum_{i=1}^{N} (u_{ij}\ln(u_{ij}) - u_{ij})\,.$$

Then taking the u_{ij} derivative of J_1 and setting it equal to zero gives

$$\frac{\partial J_1}{\partial u_{ij}} = d(x_i, \theta_j) + \eta_j\left[\ln(u_{ij}) + 1 - 1\right] = 0\,.$$

When we solve for u_{ij} we get

$$u_{ij} = \exp\left(-\frac{d(x_i, \theta_j)}{\eta_j}\right)\,. \tag{271}$$

This is the update step of u_{ij} for fixed parameters θ_j then we update the parameters θ_j via a parameter updating step by taking the $\frac{\partial}{\partial \theta_j}$ of $J_1(\theta, J)$, setting the result equal to zero and solving for θ_j i.e. solving

$$\sum_{i=1}^{N} u_{ij}(t-1)\frac{\partial}{\partial \theta_j} d(x_i; \theta_j) = 0\,,$$

this last set depends the type of clustering desired.

Problem 14.17 (plots of u_{ij} vs. $d(x_i, \theta_j)/\eta_j$

See the MATLAB script `chap_14_prob_17.m` where these expressions for u_{ij} are plotted. When that script is run we get the plot shown in Figure 28.

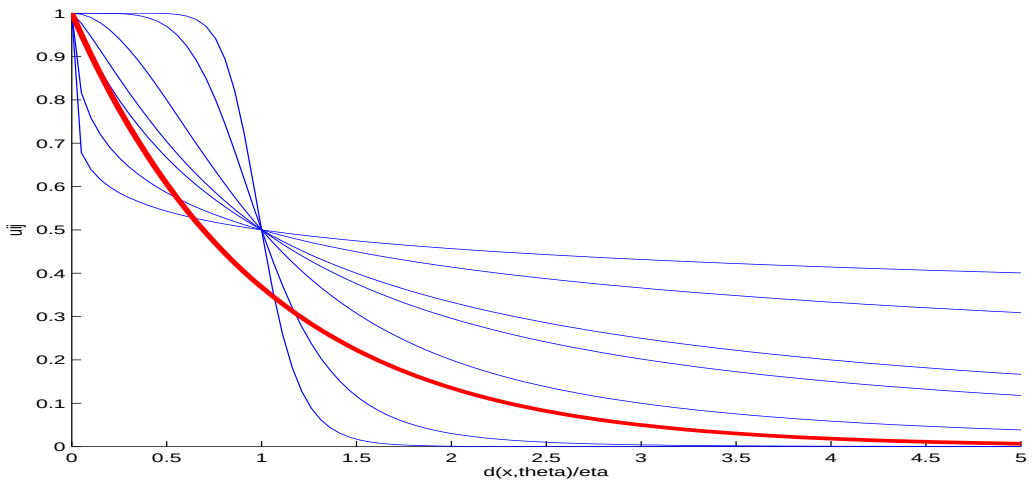

Figure 28: The plots of u_{ij} given by Equation 266 for various value of q (in blue) and by Equation 271 (in red).

Clustering Algorithms IV

Notes on the text

Notes on Competitive Learning-Like Algorithms

We are to consider the objective function $J(W)$ defined by

$$\begin{aligned} J(W) &= \frac{1}{2m}\sum_{i=1}^{N}\sum_{j=1}^{m} z_j(x_i)||x_i - w_j||^2 \qquad (272)\\ &= \frac{1}{2m}\sum_{i=1}^{N}\sum_{j=1}^{m}\left(\frac{||x_i - w_j||^{-2}}{\sum_{r=1}^{m}||x_i - w_r||^{-2}}\right)||x_i - w_j||^2 \\ &= \frac{1}{2m}\sum_{i=1}^{N}\left(\frac{1}{\sum_{r=1}^{m}||x_i - w_r||^{-2}}\right) \\ &= \frac{1}{2}\sum_{i=1}^{N}\left(\sum_{r=1}^{m}||x_i - w_r||^{-2}\right)^{-1} . \qquad (273) \end{aligned}$$

Now to take the derivative of the above expression with respect to the vector w_k we get

$$\begin{aligned} \frac{\partial J}{\partial w_k} &= \frac{1}{2}\sum_{i=1}^{N} -1\left(\sum_{r=1}^{m}||x_i - w_r||^{-2}\right)^{-2}\cdot\frac{\partial}{\partial w_k}\left(||x_i - w_k||^{-2}\right) \\ &= -\frac{1}{2}\sum_{i=1}^{N}(z_k^2(x_i)||x_i - w_k||^4)\frac{\partial}{\partial w_k}(||x_i - w_k||^{-2}) \\ &= -\frac{1}{2}\sum_{i=1}^{N}(z_k^2(x_i)||x_i - w_k||^4)(-2)||x_i - w_k||^{-3}\frac{\partial}{\partial w_k}(||x_i - w_k||) \\ &= \sum_{i=1}^{N} z_k(x_i)^2||x_i - w_k||\frac{\partial}{\partial w_k}||x_i - w_k|| . \end{aligned}$$

To continue, we recall that the vector derivative of the vector norm is given by

$$\frac{\partial}{\partial w_k}||x_i - w_k|| = -\frac{(x_i - w_r)}{||x_i - w_k||} ,$$

which we can be shown by taking the derivative of $||x_i - w_k||$ with respect to each component one at a time. Thus we get

$$\frac{\partial J}{\partial w_k} = -\sum_{i=1}^{N} z_k^2(x_i)(x_i - w_k) , \qquad (274)$$

which is the result given in the book.

Notes on the branch and bound clustering algorithms

We are given $\mathcal{C}_r = [c_1, c_2, \ldots, c_r]$ or an assignment of the first r points from the data set X into the m clusters $c_i \in \{1, 2, \ldots, m\}$. Note that in this section there are two objects that are denoted via the "c" character. The first, c_i, represents is the cluster that the point x_i is assigned to. The second, $\mathcal{C}_r$, is all assignments to clusters for the first r points from X. With these definitions this we can define an objective function for this cluster assignment as follows

$$J(\mathcal{C}_r) = \sum_{i=1}^{r} ||x_i - m_{c_i}(\mathcal{C}_r)||^2 . \tag{275}$$

Here $m_j(\mathcal{C}_r)$ is defined as

$$m_j(\mathcal{C}_r) = \frac{1}{n_j(\mathcal{C}_r)} \sum_{q=1,\ldots,r:C_q=j} x_q . \tag{276}$$

This is a more complicated way of writing the fact that $m_j(\mathcal{C}_r)$ is the mean of all vectors assigned to the jth cluster. Based on the above and assuming that we assign the point x_{r+1} to cluster j we can write

$$\begin{aligned} J(\mathcal{C}_{r+1}) &= \sum_{i=1}^{r+1} ||x_i - m_{c_i}(\mathcal{C}_{r+1})||^2 \\ &= \sum_{i=1}^{r} ||x_i - m_{c_i}(\mathcal{C}_{r+1})||^2 + ||x_{r+1} - m_j(\mathcal{C}_{r+1})||^2 \\ &= \sum_{i=1:c_i \neq j}^{r} ||x_i - m_{c_i}(\mathcal{C}_{r+1})||^2 + \sum_{i=1:c_i=j}^{r} ||x_i - m_{c_i}(\mathcal{C}_{r+1})||^2 + ||x_{r+1} - m_j(\mathcal{C}_{r+1})||^2 . \end{aligned}$$

In the last step we have broken the original sum up into two additional sums. The first is the sum over all the points x_i that are *not* assigned to the cluster j and the second is the sum over all the points x_i that *are* assigned to cluster j. Now with these two new sum, in the first sum since we are ignoring the cluster j it can be written as

$$\sum_{i=1:c_i \neq j}^{r} ||x_i - m_{c_i}(\mathcal{C}_r)||^2 ,$$

where we now have $m_{c_j}(\mathcal{C}_r)$ rather than $m_{c_j}(\mathcal{C}_{r+1})$ since the means over $\mathcal{C}_r$ and $\mathcal{C}_{r+1}$ are the same for all clusters that are not equal to j. With this we now have

$$J(\mathcal{C}_{r+1}) = \sum_{i=1:c_i \neq j}^{r} ||x_i - m_{c_i}(\mathcal{C}_r)||^2 + \sum_{i=1:c_i=j}^{r} ||x_i - m_j(\mathcal{C}_{r+1})||^2 + ||x_{r+1} - m_j(\mathcal{C}_{r+1})||^2 . \tag{277}$$

Now lets consider the mean vector of the j cluster or $m_j(\mathcal{C}_{r+1})$ which shows up in two places above. Since the new point x_{r+1} is placed in that cluster we have

$$\begin{aligned} m_{c_j}(\mathcal{C}_{r+1}) &= \frac{1}{n_j(C_r)+1}\left(\sum_{q=1,\cdots,r:c_q=j} x_q + x_{r+1}\right) \\ &= \frac{n_j(\mathcal{C}_r)}{n_j(\mathcal{C}_r)+1} m_j(\mathcal{C}_r) + \frac{1}{n_j(\mathcal{C}_r)+1} x_{r+1} \quad (278) \\ &= m_j(\mathcal{C}_r) + \frac{1}{n_j(\mathcal{C}_r)+1}(x_{r+1} - m_j(\mathcal{C}_r))\,. \quad (279) \end{aligned}$$

Then with this we find $x_i - m_j(\mathcal{C}_{r+1})$ can be written as

$$x_i - m_j(\mathcal{C}_{r+1}) = x_i - m_j(\mathcal{C}_r) - \frac{1}{n_j(\mathcal{C}_r)+1}(x_{r+1} - m_j(\mathcal{C}_r))\,.$$

Thus the norm needed in the second sum in Equation 277 is

$$\begin{aligned} ||x_i - m_j(\mathcal{C}_{r+1})||^2 &= ||x_i - m_j(\mathcal{C}_r)||^2 - \frac{2}{n_j(\mathcal{C}_r)+1}(x_i - m_j(\mathcal{C}_r))(x_{r+1} - m_j(\mathcal{C}_r)) \\ &\quad + \frac{1}{(n_j(\mathcal{C}_r)+1)^2}||x_{r+1} - m_j(\mathcal{C}_r)||^2\,. \end{aligned}$$

Now when we sum this expression over $\sum_{i=1:c_i=j}^{r}$ the middle term vanishes due to the fact that $\sum_{i=1:c_i=j}^{r}(x_i - m_j(\mathcal{C}_r)) = 0$. Thus we find for the second sum in Equation 277 the terms

$$\sum_{i=1:c_i=j}^{r} ||x_i - m_j(\mathcal{C}_r)||^2 + \frac{n_j(\mathcal{C}_r)}{n_j(\mathcal{C}_r)+1)^2}||x_{r+1} - m_j(\mathcal{C}_r)||^2\,.$$

Now for the lone term $||x_{r+1} - m_j(\mathcal{C}_{r+1})||^2$ in Equation 277 we find

$$x_{r+1} - m_j(\mathcal{C}_{r+1}) = \left(1 - \frac{1}{n_j(\mathcal{C}_r)+1}\right)x_{r+1} - \frac{n_j(\mathcal{C}_r)}{n_j(\mathcal{C}_r)+1} m_j(\mathcal{C}_r) \quad = \frac{n_j(\mathcal{C}_r)}{n_j(\mathcal{C}_r)+1}(x_{r+1} - m_j(\mathcal{C}_r))\,.$$

Thus

$$||x_{r+1} - m_j(\mathcal{C}_{j+1})||^2 = \frac{n_j(\mathcal{C}_r)^2}{(n_j(\mathcal{C}_r)+1)^2}||x_{r+1} - m_j(\mathcal{C}_r)||^2\,.$$

Thus combining these we find

$$\begin{aligned} J(\mathcal{C}_{r+1}) &= J(\mathcal{C}_r) + \frac{n_j(\mathcal{C}_r)}{n_j(\mathcal{C}_r)+1)^2}||x_{r+1} - m_j(\mathcal{C}_r)||^2 + \frac{n_j(\mathcal{C}_r)^2}{(n_j(\mathcal{C}_r)+1)^2}||x_{r+1} - m_j(\mathcal{C}_r)||^2 \\ &= J(\mathcal{C}_r) + \frac{n_j(\mathcal{C}_r)}{n_j(\mathcal{C}_r)+1}||x_{r+1} - m_j(\mathcal{C}_r)||^2\,. \end{aligned}$$

This last expression verifies the books expression for $\Delta J(\mathcal{C}_r)$.

Notes on the boundary detection algorithm

Recall that the form of $J(\theta)$ given in this section is

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} f^2(g(x_i;\theta)) - \left(\frac{1}{N}\sum_{i=1}^{N} f(g(x_i;\theta))\right)^{2q} . \tag{280}$$

Then if we consider the positive expression given in the book (denoted as E)

$$E = \frac{1}{N}\sum_{i=1}^{N}\left(f(g(x_i;\theta)) - \frac{1}{N}\sum_{k=1}^{N} f(g(x_k;\theta))\right)^2 ,$$

by expanding we have

$$\begin{aligned} E &= \frac{1}{N}\sum_{i=1}^{N}\left(f^2 g(x_i;\theta)) - \frac{2}{N} f(g(x_i;\theta))\sum_{k=1}^{N} f(g(x_i;\theta)) + \left(\frac{1}{N}\sum_{k=1}^{N} f(g(x_i;\theta))\right)^2\right) \\ &= \frac{1}{N}\sum_{i=1}^{N} f^2 g(x_i;\theta)) - \frac{2}{N^2}\left(\sum_{k=1}^{N} f(g(x_i;\theta))\right)^2 + \frac{1}{N^2}\left(\sum_{k=1}^{N} f(g(x_i;\theta))\right)^2 \\ &= \frac{1}{N}\sum_{i=1}^{N} f^2 g(x_i;\theta)) - \frac{1}{N^2}\left(\sum_{k=1}^{N} f(g(x_i;\theta))\right)^2 . \end{aligned} \tag{281}$$

As q is a positive integer and the sum $\frac{1}{N}\sum_{i=1}^{N} f(g(x_i;\theta))$ is inside $[-1,+1]$ we have that

$$\left(\frac{1}{N}\sum_{i=1}^{N} f(g(x_i;\theta))\right)^{2q} \leq \left(\frac{1}{N}\sum_{i=1}^{N} f(g(x_i;\theta))\right)^2$$

When we negate this expression and add $\frac{1}{N}\sum_{i=1}^{N} f^2(g(x_i;\theta))$ to both sides we get using Equation 281 that

$$\frac{1}{N}\sum_{i=1}^{N} f^2(g(x_i;\theta)) - \left(\frac{1}{N}\sum_{i=1}^{N} f(g(x_i;\theta))\right)^{2q} \geq \frac{1}{N}\sum_{i=1}^{N} f^2(g(x_i;\theta)) - \left(\frac{1}{N}\sum_{i=1}^{N} f(g(x_i;\theta))\right)^2 ,$$

or

$$J(\theta) \geq E ,$$

the expression stated in the book.

Problem Solutions

Problem 15.3 (leaky learning with $\eta_w = \eta_l$)

In this case *every* cluster representative w_j gets updated via

$$w_j(t) = w_j(t-1) + \eta(x - w_j(t-1)) .$$

Since each cluster representative w_j is initialized randomly (perhaps with a random sample from X) each of these cluster centers is a particular step in a Robbins-Monro iterations for solving for w in $E_X[h(X,w)] = 0$ where $h(X,w) = X - w$. For this $h(X,w)$ the limiting value of $w_j(t)$ (for all j) should be the mean of all the samples in X.

Problem 15.4 (the *von Malsburg learning rule*)

Part (b): If w_j looses the competition for x then it does not change its value so if $\sum_k w_{jk} = 1$ before the assignment of a sample it will still hold afterwards. If w_j wins the competition for x then

$$\sum_k w_{jk}^{\text{new}} = \sum_k w_{jk} + \eta \left(\sum_k \frac{x_k}{n_k} - \sum_k w_{jk} \right) .$$

But $\sum_k \frac{x_k}{n_k} = 1$ and $\sum_k w_{jk} = 1$ so $\sum_k w_{jk}^{\text{new}} = 1$.

Problem 15.5 (deriving the expression for $\Delta J(\mathcal{C}_r)$

See the notes on Page 194 where this problem is worked.

Problem 15.9 (a derivative)

The given expression can have the same value for $g(x;\theta)$ but with different values for w_i. This is due to the symmetry in the products $x_s x_r$. Thus it should be written as

$$g(x;\theta) = w_0 + \sum_{i=1}^{l} w_i x_i + \sum_{s=1}^{l} \sum_{r>s} w_{sr} x_s x_r \, .$$

Where we have $\frac{1}{2} l(l+1)$ terms in the second sum. Derivatives of this expression can then be computed with respect to the components of w_0, w_i and w_{sr}.

Problem 15.13 (time till convergence)

If we take $T_{\max} = 5$ and $T_{\min} = 0.5$ then using the equation suggested in the book we have

$$\ln(1 + t_{\text{end}}) = \frac{T_{\max}}{T_{\min}} = \frac{5}{0.5} = 10 \, .$$

Thus solve we find $t_{\text{end}} = 22025.46$. A large number of iterations.

Cluster Validity

Notes on the text

Notes on comparing $\mathcal{P}$ and $\mathcal{C}$

These are some simple notes on bootstrapping statistics when imposing an *external* clustering criterion. In this section we assume that we have some external clustering (represented by $\mathcal{P}$) and we desire to see if the computed clustering (represented by $\mathcal{C}$) duplicates/describes the same phenomena as $\mathcal{P}$. Note that to compute the Rand statistics, Jaccard coefficient, or the Fowlkes & Mallows index we need two clusterings $\mathcal{C}'$ and $\mathcal{P}'$. For the hypothesis testing for cluster structure discussed in this section we need to generate these two clusterings and then construct many samples of the measure (Rand statistic, Jaccard coefficient, etc). Thus we need $\mathcal{C}'$ from our algorithm and $\mathcal{P}$ from our external criterion. We thus need to generate bootstrap data samples of both $\mathcal{C}'$ and $\mathcal{P}'$. From these samples we can compute a distribution over the given measure. We can then test whether any given sample of the measure comes from this distribution or is an outlier.

Problem Solutions

Problem 16.1 (Rand, Jaccard, Fowlkes & Mallows)

Recall that the Rand statistic is given by

$$R = \frac{a+d}{a+d+b+c}\,. \tag{282}$$

As all of the variables a, b, c, d are nonnegative the rand statistic will be less than one if $b+c>0$. The smallest value that $b+c$ can be is 0. In order for $b+c=0$ we must have both $b=0$ and $c=0$. Since b is the number points of the "same–different" (SD) category and c is the number points in the "different–same" (DS) category both b and c cannot as long as the number of m clusters in $\mathcal{C}$ and the number q partitions in $\mathcal{P}$ are not the same, there must be points in at least one of these two groups. In other words not both b and c can be zero. Thus $b+c>0$ and the rand coefficient is less than one. Because the forms for the Jaccard coefficient

$$J = \frac{a}{a+b+c}\,, \tag{283}$$

and the Fowlkes & Mallows index

$$\mathrm{FM} = \sqrt{\frac{a}{a+b}\frac{a}{a+c}}\,, \tag{284}$$

have fractions that are less than one if either b or c is nonzero we have that both these expressions are less than one.

Problem 16.2 (an expression for $\hat{\Gamma}$)

Recall that $X(i,j) = 1$ if x_i and x_j are in the same cluster in $\mathcal{C}$, and $Y(i,j) = 1$ if x_i and x_j are in the same group in $\mathcal{P}$ and are zero otherwise. Now note that the definitions of μ_X and μ_Y can be simplified as

$$\mu_X \equiv \frac{1}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} X(i,j) = \frac{1}{M}m_1 \quad \text{and}$$
$$\mu_Y \equiv \frac{1}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} Y(i,j) = \frac{1}{M}m_2\,.$$

Using these the expressions for σ_X^2 and σ_Y^2 can be simplified as

$$\sigma_X^2 \equiv \frac{1}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} X(i,j)^2 - \mu_X^2 = \frac{1}{M}m_1 - \frac{1}{M^2}m_1^2 \quad \text{and}$$
$$\sigma_Y^2 = \frac{1}{M}m_2 - \frac{1}{M^2}m_2^2\,.$$

Now note that we can write the double sum in the expression for $\hat{\Gamma}$ (denoted by E for expression) as

$$\begin{aligned}
E &= \frac{1}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(X(i,j)-\mu_X)(Y(i,j)-\mu_Y) \\
&= \frac{1}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} X(i,j)Y(i,j) - \frac{\mu_X}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} Y(i,j) \\
&\quad - \frac{\mu_Y}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} X(i,j) + \frac{\mu_X\mu_Y}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} 1 \\
&= \Gamma - \mu_X\mu_Y - \mu_X\mu_Y + \mu_X\mu_Y \\
&= \Gamma - \mu_X\mu_Y\,.
\end{aligned}$$

Where we have use the definition of Hubert's Γ statistic

$$\Gamma \equiv \frac{1}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} X(i,j)Y(i,j)\,. \tag{285}$$

Note that since $\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} X(i,j)Y(i,j) = a$ we can write $\Gamma = \frac{a}{M}$. Thus when we use these expressions to evaluate $\hat{\Gamma}$ we find

$$\begin{aligned}
\hat{\Gamma} \equiv \frac{\Gamma - \mu_X\mu_Y}{\sigma_X\sigma_Y} &= \frac{\frac{a}{M} - \frac{1}{M^2}m_1m_2}{\sqrt{\left(\frac{1}{M}m_1 - \frac{1}{M^2}m_1^2\right)\left(\frac{1}{M}m_2 - \frac{1}{M^2}m_2^2\right)}} \\
&= \frac{(Ma - m_1m_2)}{\sqrt{(Mm_1 - m_1^2)(Mm_2 - m_2^2)}} = \frac{(Ma - m_1m_2)}{\sqrt{m_1m_2(M-m_1)(M-m_2)}}\,,
\end{aligned} \tag{286}$$

the desired expression.

Problem 16.4 (the $CPCC$ is bounded between $[-1, +1]$)

Note that the expression for $CPCC$ is a correlation between two vectors. One vector has values from the elements of the upper diagonal cophenetic matrix P_c and the other has elements from the upper diagonal of the proximity matrix P. As correlations ρ are always bounded $|\rho| \leq 1$ so must this measure.

Problem 16.6 (The modified Hubert Γ statistics)

Recall that to define the *modified* Hubert statistic, we start from the samples x_i directly by first computing the proximity matrix $\mathcal{P}$. From the hard clustering centers w_i for $i = 1, \ldots, m$, we define the cluster index c_i to be $c_i = k$ if the sample x_i is a member of the kth cluster $\mathcal{C}_k$. Then for each of the N x_i data points we define the matrix $Q(i, j)$ to have elements equal to $d(w_{c_i}, w_{c_j})$ where w_i is the hard cluster representative/center.

Note that if we already have a algorithm or subroutine that takes as input data samples x_i and computes proximity matrices $\mathcal{P}$ we can use it to create the matrix Q by creating a surrogate derived data set. This derived set is obtained by listing the cluster centers w_{c_i} associated with each data sample x_i. We then call our proximity matrix subroutine on the derived data set. Then using $\mathcal{P}$ and Q the modified Γ Hubert statistic for general symmetric matrices $X(i, j)$ and $Y(i, j)$ is obtained by computing

$$
\begin{aligned}
M &= \frac{1}{2}N(N-1) \\
\mu_X &= \frac{1}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} X(i,j) \quad \text{the same for } \mu_Y \\
\sigma_X^2 &= \frac{1}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} X(i,j)^2 - \mu_X^2 \quad \text{the same for } \sigma_Y^2 \\
\Gamma &= \frac{1}{M}\sum_{i=1}^{N}\sum_{j=i+1}^{N} X(i,j)Y(i,j) \quad \text{and} \\
\hat{\Gamma} &= \frac{1}{\sigma_X \sigma_Y}\sum_{i=1}^{N}\sum_{j=i+1}^{N} (X(i,j)-\mu_X)(Y(i,j)-\mu_Y)\,.
\end{aligned}
$$

Part (a): In this part of this problem we have

$$
\begin{aligned}
c_i &= 1 \quad \text{for} \quad i \in \{1, 2, \ldots, 8\} \\
c_i &= 2 \quad \text{for} \quad i \in \{9, 10, \ldots, 16\}\,.
\end{aligned}
$$

Problem 16.7 (computing Dunn's index)

We let the distance between two clusters be given by

$$d(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y) \,.$$

and the cluster diameter as

$$\text{diam}(C) = \max_{x,y \in C} d(x, y) \,.$$

Then with these, the Dunn index for a fixed number of clusters m is given by

$$D_m = \min_{i=1,2,\ldots,m} \left\{ \min_{j=i+1,i+2,\ldots,m} \left(\frac{d(C_i, C_j)}{\max_{k=1,2,\ldots,m} \text{diam}(C_k)} \right) \right\} . \tag{287}$$

Now based on the above formula and for a fixed value of m in the numerator we see that we need to compute $d(C_i, C_j)$ for $i = 1, 2, \ldots, m$ (all clusters) and $j = i+1, i+2, \ldots, m$ (all "other" clusters). While for the denominator we need to compute $\text{diam}(C_k) = \max_{x,y \in C_k} d(x, y)$ over all m. Each of these calculations involves the pairwise distances between all the samples in the two clusters. If we extend the limit of j to include the $j = i$ case we will have all of the pairwise distances needed to evaluate $\text{diam}(C)$. Thus we need the evaluate $d(C_i, C_j)$ for $i = 1, \ldots, n$ and $j = i, i+1, \ldots, m$ (note the index j starts at i). Thus there are $O(\frac{m(m+1)}{2})$ pairwise cluster distances we need to calculate to compute the Dunn index for a fixed m. Each of these calculations takes $O(\frac{1}{2} n_i n_j)$ distance calculations between data points, where n_i and n_j are the number of data points in the ith and jth cluster respectively. We assume that with m clusters there will be $O(\frac{N}{m})$ data points in each cluster and thus we have

$$O\left(\frac{N}{m} \cdot \frac{m}{2}(m+1) \right) = O\left(\frac{N}{2}(m+1) \right) ,$$

distance calculations for each m. If we do this for $m = 1, 2, \ldots, N$ we need to sum these numbers and find the total number of computations given by

$$\sum_{m=1}^{N} O\left(\frac{N}{2}(m+1) \right) = \frac{N}{2} O\left(\sum_{m=1}^{N} m \right) = \frac{N^2(N+1)}{4} = O(N^3) \,,$$

which can be a large number of calculations.

Problem 16.8 (two more Dunn like indices)

In the same way that E_i^{MST} is the minimum spanning graph (MSG) derived from the complete graph G_i from on the samples in the ith cluster C_i, we define the graphs E_i^{RNG} to be the *relative neighborhood graph* (RNG) and E_i^{GG} to be the *Gabriel graph* (GG) based on the clusters complete graph G_i. Given these two graphs we will define the RNG or GG diameter of the given cluster to be the length of the largest edge in the relative neighborhood or Gabriel graphs respectively. Once we have defined the GG and RNG diameters as above, we

can compute Dunn like indices using a definition similar to Equation 287. We have assumed the dissimilarity between the two clusters C_i and C_j is related to the distance between the cluster representatives m_i and m_j i.e.

$$d(C_i, C_j) = d(m_i, m_j)\,,$$

Problem 16.9 (bounds on various Dunn indices)

Since we are told that for a cluster C_i that

$$E_i^{MST} \subset E_i^{RNG} \subset E_i^{GG}\,, \tag{288}$$

based on this we know that the graph diameters must satisfy a similar ordering

$$\text{diam}_i^{MST} \leq \text{diam}_i^{RNG} \leq \text{diam}_i^{GG}\,. \tag{289}$$

Thus the maximum in the denominator of the Dunn index Equation 287, will get sequentially larger as we consider the MST, the RNG, and finally the GG graph. Thus the Dunn index will get smaller and we have

$$D_m^{GG} \leq D_m^{RNG} \leq D_m^{MST}\,.$$

Problem 16.10 (conditions C1-C5)

Part (a): Take the diameter to be the measure of dispersion or spread s_i around a clusters mean value. Then we define R_{ij}^{MST} as

$$R_{ij}^{MST} = \frac{s_i^{MST} + s_j^{MST}}{d_{ij}} = \frac{\text{diam}^{MST}(C_i) + \text{diam}^{MST}(C_j)}{d_{ij}}\,. \tag{290}$$

From this we see that C1, C2, C3 are satisfied. If $s_i^{MST} > s_k^{MST}$ and $d_{ij} = d_{ik}$ then we have

$$R_{ij}^{MST} = \frac{s_j^{MST} + s_i^{MST}}{d_{ij}} > \frac{s_k^{MST} + s_i^{MST}}{d_{ik}} = R_{ik}^{MST}\,,$$

showing that C4 is true. Given the condition for C5 we have

$$R_{ij}^{MST} = \frac{s_i^{MST} + s_j^{MST}}{d_{ij}} > \frac{s_i^{MST} + s_k^{MST}}{d_{ik}} = R_{ik}^{MST}\,,$$

showing C5.

Part (b): The only change to compute R_{ij}^{RNG} and R_{ij}^{GG} is to compute the diameter of the graph based on the relative neighborhood graph (RNG) or the Gabriel graph (GG) respectively and use that number to evaluate the spread of a cluster around its center. Thus R_{ij}^{RNG} and R_{ij}^{GG} should satisfy C1-C5 as R_{ij}^{MST} does.

Problem 16.11 (inequalities of DB_m)

Using Equation 289 we see that when $s_i^{method} \equiv \text{diam}_i^{method}$ for the methods MST, RNG, and GG we have

$$s_i^{MST} \leq s_i^{RNG} \leq s_i^{GG} .$$

So form Equation 290 we have

$$R_{ij}^{MST} \leq R_{ij}^{RNG} \leq R_i^{GG} .$$

Finally since $DB_m^{method} = \frac{1}{m} \sum_{i=1}^{m} R_i^{method}$ we thus get

$$DB_m^{MST} \leq DB_m^{RNG} \leq DB_m^{GG} ,$$

as we were to show.

Problem 16.12 (robustness of MST DB)

Recall that the minimum spanning tree (MST) graph looks only at the smallest tree that we can construct from the given complete graph G_i of the points belonging to the i cluster. Even if cluster i has some outliers if we define s_i^{MST} to be the "diameter" (the length of the longest edge in the MST) these outlying points will not affect the value of s_i^{MST}, since the MST is considering the smallest tree. In general, another form of s_i to be used in R_{ij} (not the MST version) would have its value changed due to these outliers. For example, if we are using a direct cluster diameter $\text{diam}(C_i) = \max_{x,y \in C} d(x, y)$ as the definition of s_i we expect outliers to affect its value. Since the values of R_{ij}^{MST} are less susceptible to outliers, minimizing $DB_m^{MST} = \sum_{i=1}^{m} R_i^{MST}$ as a function of m should be also.

Problem 16.13 (PC and PE as a function of the fuzzifier q)

Part (a): As stated in the book in the chapter on fuzzy clustering as $q \to 1^+$ then no fuzzy clustering is better than the best hard clustering. Thus $u_{ij} = 1$ when $j = k$ where the kth cluster is the one that the best hard clustering would put the sample x_i into and while $u_{ij} = 0$ for all other js. Based on this we see that

$$PC = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^2 \to \frac{1}{N} \sum_{i=1}^{N} 1 = 1 ,$$

and

$$PE = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij} \log_a(u_{ij}) \to -\frac{1}{N} \sum_{i=1}^{N} 0 = 0 ,$$

since $\lim_{u \to 0} u \log(u) = 0$.

Part (b): Now as $q \to \infty$ we have that $u_{ij} \to \frac{1}{m}$ as each sample has an equal weight in all clusters and we thus get

$$PC \to \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{m}\frac{1}{m^2} = \frac{1}{m}$$

$$PE \to -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{m}\frac{1}{m}\log_a\left(\frac{1}{m}\right) = \frac{\log_a(m)}{mN}(mN) = \log_a(m)\,.$$

Problem 16.14 (limits of XB with respect to q)

We are told that when $q \to +\infty$ we have $\lim_{q\to\infty} w_i = w$ where w is the mean vector over the set of data X. Then because of this we see that

$$d_{\min} = \min_{i,j=1,\ldots,m;i\neq j} ||w_i - w_j|| \to 0\,.$$

At the same time as $q \to \infty$ we get $u_{ij} \to \frac{1}{m}$ thus

$$\sigma_j^2 = \sum_{i=1}^{N} u_{ij}^2 ||x_i - w_j||^2 \to \sum_{i=1}^{N}\left(\frac{1}{m}\right)^2 ||x_i - w||^2\,,$$

which is a positive constant independent of j. Thus the total variation or the sum of the m of these σ_j^2 is also a constant say C. Thus we have shown that as $q \to \infty$ that

$$XB \to \frac{C}{0} \to \infty\,.$$

When we want to consider the XB_q case we recall that

$$XB_q = \frac{\sigma_q}{N d_{\dim}}\,. \tag{291}$$

We have already shown that $d_{\dim} \to 0$ as $q \to \infty$. Consider now the value of σ_q as $q \to \infty$. From the definitions given in the book we have

$$\sigma_q = \sum_{j=1}^{m}\sigma_j^q = \sum_{j=1}^{m}\sum_{i=1}^{N} u_{ij}^q ||x_i - w_j||\,.$$

As $u_{ij} \to \frac{1}{m}$ as $q \to \infty$ we see that $u_{ij}^q \to 0$ as $q \to \infty$. Thus each term in the expression for σ_q goes to zero. Thus we have $XB_q \to \frac{0}{0}$ which is indeterminate.

Problem 16.15 (limits of FS_q)

To begin, we first recall that FS_q is given by

$$FS_q = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^q(||x_i - w_j||_A^2 - ||w_j - w||_A^2)\,. \tag{292}$$

Part (a): If $q \to 1^+$ then $u_{ij} = 1$ when the index j corresponds to the cluster k that x_i is a member of and $u_{ij} = 0$ for all other j. Thus when $q \to 1^+$ we have a hard clustering. If we let $j(i)$ be the cluster index that the point x_i is a member of we get

$$FS_q = \sum_{i=1}^{N}(||x_i - w_{j(i)}||_A^2 - ||w_{j(i)} - w||_A^2) = \sum_{i=1}^{N}||x_i - w_{j(i)}||_A^2 - \sum_{i=1}^{N}||w_{j(i)} - w||_A^2 .$$

Note that the first sum in the above is the sum of the data samples x_i around the individual cluster representatives w_j, and the second sum is the sum of cluster representatives w_j around the global center w. If we change the sums above, which are over the points x_i to sums over the m clusters and the n_j points inside each we get

$$\begin{aligned} FS_q &= \sum_{j=1}^{m}\sum_{i=1}^{n_j}||x_i - w_j||_A^2 - \sum_{j=1}^{m} n_j||w_j - w||_A^2 \\ &= N\sum_{j=1}^{m}\frac{n_j}{N}\left(\frac{1}{n_j}\sum_{i=1}^{n_i}||x_i - w_j||_A^2\right) - N\sum_{j=1}^{m}\frac{n_j}{N}||w_j - w||_A^2 \\ &= N\text{trace}(S_w) - N\text{trace}(S_b) , \end{aligned}$$

where we have introduced the within S_w and between scatter matrices S_b as

$$\begin{aligned} S_w &\equiv \sum_{j=1}^{m}\frac{n_j}{N}\left(\frac{1}{n_j}\sum_{i=1}^{n_j}(x_i - w_j)A(x_i - w_j)^T\right) \\ S_b &\equiv \sum_{j=1}^{m}\frac{n_j}{N}(w_j - w)A(w_j - w)^T . \end{aligned}$$

From an earlier chapter in the book we can write the between scatter matrix S_b as $S_b = S_m - S_w$ where S_m is

$$S_m \equiv \frac{1}{N}\sum_{i=1}^{N}(x_i - w)A(x_i - w)^T ,$$

to get

$$\begin{aligned} FS_q &= N\text{trace}(S_w) - N\text{trace}(S_b) = N\text{trace}(S_w) - N(\text{trace}(S_m) - \text{trace}(S_w)) \\ &= 2N\text{trace}(S_w) - N\text{trace}(S_m) , \end{aligned}$$

the result we wanted to show.

Part (b): If $q \to +\infty$ then earlier we have shown that $u_{ij} \to \frac{1}{m}$ so $u_{ij}^q \to 0$ and $w_j \to w$ as $q \to +\infty$. Thus FS_q given by Equation 292 is the sum of terms all of which are going to zero and is therefore equal to zero in this limit.

Problem 16.16 (distance to the closest point in a sphere)

Our sphere is defined as the points x such that $||x - c_j||^2 = r_j^2$. We want to find the point x^* that is on the sphere and closest to some exterior point x_i. Let the distance (squared)

between x_i and x^* be denoted by $d^2 = ||x_i - x^*||^2$. We can phrase this problem as a constrained optimization problem where we want to minimize d^2 as a function of x^* subject to $||x^* - c_j||^2 = r_j^2$. To do this we form the Lagrangian

$$\mathcal{L} = ||x^* - x_i||^2 - \lambda(||x^* - c_j||^2 - r_j^2)\,.$$

Then the two needed derivatives (and set equal to zero) are

$$\frac{\partial \mathcal{L}}{\partial x^*} = 2(x^* - x_i) - 2\lambda(x^* - c_j) = 0$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = ||x^* - c_j||^2 - r_j^2 = 0\,.$$

The first equation states that the vectors $x^* - x_i$ is a scalar multiple of the vector $x^* - c_j$ meaning that they two vectors are parallel. Thus x^* is on the line between the point x_i and the center of the sphere c_j. Solving the first equation for x^* gives the point

$$x^* = \frac{1}{1-\lambda}(x_i - \lambda c_j)\,. \tag{293}$$

If we put this point into the constraint $||x^* - c_j||^2 = r_j^2$ we get

$$\left|\left|\frac{1}{1-\lambda}(x_i - \lambda c_j) - c_j\right|\right|^2 = r_j^2\,,$$

or simplifying some

$$\left|\left|\frac{1}{1-\lambda}x_i - \frac{1}{1-\lambda}\right|\right|^2 = r_j^2\,.$$

If we solve this equation for λ we get

$$\lambda = 1 - \frac{||x_i - c_j||}{r_j}\,.$$

Now that we know λ we can use its value in Equation 293 to compute the difference $x_i - x^*$, where we find

$$x_i - x^* = x - \frac{1}{1-\lambda}(x_i - \lambda c_j) = -\frac{\lambda}{1-\lambda}x_i + \frac{\lambda}{1-\lambda}c_j\,.$$

From what λ is we can also compute that

$$\frac{\lambda}{1-\lambda} = \frac{r_j}{||x - c_j||} - 1\,,$$

so the above difference $x_i - x^*$ equals

$$\begin{aligned} x_i - x^* &= x_i - \frac{r_j}{||x - c_j||}x_i - c_j + \frac{r_j}{||x - c_j||}c_j \\ &= x_i - c_j - \frac{r_j}{||x_i - c_j||}(x_i - c_j)\,, \end{aligned} \tag{294}$$

which is defined to be the vector τ_{ij} and is the desired expression

Problem 16.17 (some moments)

We are told that Σ is defined as

$$\Sigma = \frac{1}{L_\phi}\int_{-\phi/2}^{\phi/2} \mathbf{x}\mathbf{x}^T dl - \mathbf{m}\mathbf{m}^T .$$

With $\mathbf{x}^T = \begin{bmatrix} r\cos(\theta) & r\sin(\theta) \end{bmatrix}$, $dl = rd\theta$ and L_ϕ an arc length. First we have the value of L_ϕ as

$$L_\phi = \frac{\phi}{2\pi}(2\pi r) = r\phi .$$

Next we have $\mathbf{m}$ given by

$$\begin{aligned}\mathbf{m} &= \frac{1}{L_\phi}\int_{-\phi/2}^{\phi/2} \mathbf{x} dl = \frac{1}{r\phi}\int_{-\phi/2}^{\phi/2} \begin{bmatrix} r\cos(\theta) \\ r\sin(\theta) \end{bmatrix} rd\theta \\ &= \frac{r}{\phi}\begin{bmatrix} \sin(\theta)|_{-\phi/2}^{\phi/2} \\ -\cos(\theta)|_{-\phi/2}^{\phi/2} \end{bmatrix} = \frac{r}{\phi}\begin{bmatrix} 2\sin(\phi/2) \\ -\cos(\phi/2)+\cos(\phi/2) \end{bmatrix} = \frac{r}{\phi}\begin{bmatrix} 2\sin(\phi/2) \\ 0 \end{bmatrix} .\end{aligned}$$

Thus

$$\mathbf{m}\mathbf{m}^T = \frac{r^2}{\phi^2}\begin{bmatrix} 4\sin^2(\phi/2) & 0 \\ 0 & 0 \end{bmatrix} .$$

Next we compute the second moment

$$\begin{aligned}\frac{1}{L_\phi}\int_{-\phi/2}^{\phi/2} \mathbf{x}\mathbf{x}^T dl &= \frac{1}{r\phi}\int_{-\phi/2}^{\phi/2} \begin{bmatrix} r\cos(\theta) \\ r\sin(\theta) \end{bmatrix}\begin{bmatrix} r\cos(\theta) & r\sin(\theta) \end{bmatrix} rd\theta \\ &= \frac{r^2}{\phi}\int_{-\phi/2}^{\phi/2} \begin{bmatrix} \cos^2(\theta) & \cos(\theta)\sin(\theta) \\ \cos(\theta)\sin(\theta) & \sin^2(\theta) \end{bmatrix} d\theta \\ &= \frac{r^2}{\phi}\begin{bmatrix} \frac{1}{2}(\phi+\sin(\phi)) & 0 \\ 0 & \frac{1}{2}(\phi-\sin(\phi)) \end{bmatrix} ,\end{aligned}$$

Thus using this, we find for Σ the following

$$\begin{aligned}\Sigma &= r^2\begin{bmatrix} \frac{1}{2}+\frac{\sin(\phi)}{2\phi} & 0 \\ 0 & \frac{1}{2}-\frac{\sin(\phi)}{2\phi} \end{bmatrix} - r^2\begin{bmatrix} 4\frac{\sin^2(\phi/2)}{\phi^2} & 0 \\ 0 & 0 \end{bmatrix} \\ &= r^2\begin{bmatrix} \frac{1}{2}+\frac{1}{2}\frac{\sin(\phi)}{\phi}-\frac{4\sin^2(\phi/2)}{\phi^2} & 0 \\ 0 & \frac{1}{2}-\frac{1}{2}\frac{\sin(\phi)}{\phi} \end{bmatrix} .\end{aligned}$$

Since $r_{\text{eff}}^2 = \text{trace}(\Sigma)$ we get

$$r_{\text{eff}}^2 = r^2\left(1 - \frac{4\sin^2(\phi/2)}{\phi^2}\right) .$$

If $S = \sum_{j:x_j\in X'} u_j = \phi r$ then we get

$$\delta = \frac{S}{2\pi r_{\text{eff}}} = \frac{\phi r}{2\pi\sqrt{\text{trace}(\Sigma)}} = \frac{\phi}{2\pi\sqrt{1-\frac{4\sin^2(\phi/2)}{\phi^2}}}$$

which is the result we wanted to show. If $\phi = 2\pi$ then since $\sin^2(\phi/2) = \sin^2(\pi) = 0$ we find $\delta = 1$.

Hints from Probability and Statistics

Moments of a Quadratic Form

Suppose x is a $l \times 1$ random vector with $E[x] = \mu$ and $\mathrm{Cov}(x) = \Sigma$ and let A be a $l \times l$ symmetric matrix not dependent on x then the quadratic expectation $E[x^T Ax]$ is given by

$$E[x^T Ax] = \mu^T A\mu + \mathrm{trace}(\Sigma A)\,. \tag{295}$$

Optimization for constrained problems

Notes on the text

We can show the expression $\frac{\partial(A\theta)}{\partial\theta} = A^T$ is true, by explicitly computing the vector derivative on the left-hand-side. We begin by considering the expression $A\theta$. Recall that it can be expressed in component form as

$$A\theta = \begin{bmatrix} a_{11}\theta_1 + a_{12}\theta_2 + a_{13}\theta_3 + \cdots + a_{1l}\theta_l \\ a_{21}\theta_1 + a_{22}\theta_2 + a_{23}\theta_3 + \cdots + a_{2l}\theta_l \\ \vdots \\ a_{m1}\theta_1 + a_{m2}\theta_2 + a_{m3}\theta_3 + \cdots + a_{ml}\theta_l \end{bmatrix}.$$

Using the above expression the vector derivative of $A\theta$ with respect to the vector θ is then given by

$$\begin{aligned} \frac{\partial(A\theta)}{\partial\theta} &= \begin{bmatrix} \frac{\partial(A\theta)_1}{\partial\theta_1} & \frac{\partial(A\theta)_2}{\partial\theta_1} & \frac{\partial(A\theta)_3}{\partial\theta_1} & \cdots & \frac{\partial(A\theta)_m}{\partial\theta_1} \\ \frac{\partial(A\theta)_1}{\partial\theta_2} & \frac{\partial(A\theta)_2}{\partial\theta_2} & \frac{\partial(A\theta)_3}{\partial\theta_2} & \cdots & \frac{\partial(A\theta)_m}{\partial\theta_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial(A\theta)_1}{\partial\theta_l} & \frac{\partial(A\theta)_2}{\partial\theta_l} & \frac{\partial(A\theta)_3}{\partial\theta_l} & \cdots & \frac{\partial(A\theta)_m}{\partial\theta_l} \end{bmatrix} \\ &= \begin{bmatrix} a_{11} & a_{21} & a_{31} & \cdots & a_{m1} \\ a_{12} & a_{22} & a_{32} & \cdots & a_{m2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{1l} & a_{2l} & a_{3l} & \cdots & a_{ml} \end{bmatrix} = A^T. \end{aligned} \tag{296}$$

In the first equation above the notation $\frac{\partial(A\theta)_i}{\partial\theta_j}$ means the θ_j's derivative of the ith row of $A\theta$. Now that we have shown that the vector derivative of $A\theta$ with respect to θ is A^T we will use this result in discussing the first order optimality conditions under minimization of a function $J(\theta)$ subject to linear constraints on θ.

The first order optimality constraint for *constrained* optimization where the constraints are linear say given by $A\theta = b$, states that at the optimum value of θ (denoted by θ^*) there is a vector λ such that

$$\left.\frac{\partial J(\theta)}{\partial\theta}\right|_{\theta^*} = A^T\lambda. \tag{297}$$

Since $A^T\lambda$ is a linear combination of the *rows* of A this equation states that at the optimum point θ^* the vector direction of maximum increase of the objective J is in a direction spanned by the rows of A. The rows of A (by definition) are also in the directions of the linear constraints in $A\theta = b$. Since the vector θ derivative of the expression $\lambda^T A\theta$ is given by

$$\frac{\partial(\lambda^T A\theta)}{\partial\theta} = (\lambda^T A)^T = A^T\lambda,$$

we can write the first order optimality constraint expressed by Equation 297 as

$$\frac{\partial}{\partial\theta}\left(J(\theta) - \lambda^T A\theta\right) = 0.$$

To this expression we can add the term $\lambda^T b$ since it does not depend on θ and has a derivative that is zero. With this we get

$$\frac{\partial}{\partial\theta}\left(J(\theta) - \lambda^T(A\theta - b)\right) = 0\,. \tag{298}$$

Now if we define a function $\mathcal{L}(\theta;\lambda)$ as

$$\mathcal{L}(\theta;\lambda) \equiv J(\theta) - \lambda^T(A\theta - b)\,,$$

we see that our first order constrained *optimality* condition given by Equation 298 in terms of the function $\mathcal{L}$ is given by

$$\frac{\partial}{\partial\theta}\mathcal{L}(\theta;\lambda)) = 0\,,$$

which looks like an *unconstrained* optimality condition. Note that because $\mathcal{L}(\theta;\lambda)$ is a scalar we can take the transpose of it to write it as

$$\mathcal{L}(\theta;\lambda) \equiv J(\theta) - (A\theta - b)^T\lambda\,.$$

From this using Equation 296 we see that the *constraint* given by $A\theta - b = 0$ in terms of the function $\mathcal{L}$ is equivalent to the vector λ derivative set equal to zero or

$$\frac{\partial}{\partial\lambda}\mathcal{L}(\theta;\lambda)) = 0\,,$$

which is another expression that looks like a first order *unconstrained* optimality condition. Thus the functional expression $\mathcal{L}(\theta;\lambda)$ provides a convenient way to represent the solution to linearly constrained optimization problem in the exact same form as an *unconstrained* optimization problem but with a larger set of independent variables given by (θ,λ).

Notes on optimization with inequality constraints

In this section of these notes we document at a very high level (without much motivation or background) how to solve constrained optimization problems. These notes can then be referenced, as needed, when working with specific optimization problems. The general optimization problem with inequality constraints is given by

$$\begin{aligned} &\text{minimize} \quad J(\theta) \\ &\text{subject to} \quad f_i(\theta) \geq 0 \quad \text{for} \quad i = 1, 2, \cdots, m\,. \end{aligned}$$

To solve this problem we first form the *Lagrangian*, $\mathcal{L}$, defined by

$$\mathcal{L}(\theta;\lambda) \equiv J(\theta) - \sum_{i=1}^{m}\lambda_i f_i(\theta)\,. \tag{299}$$

The variables λ_i in the above expression are called Lagrange multipliers. Using this definition, a set of *necessary* conditions for a local minimizer θ^* to exist is the following:

1. $\frac{\partial}{\partial\theta}\mathcal{L}(\theta^*;\lambda) = 0$.

2. $\lambda_i \geq 0$ for $i = 1, 2, \cdots, m$.

3. $\lambda_i f_i(\theta^*) = 0$ for $i = 1, 2, \cdots, m$.

These three conditions are called the *Karush-Kuhn-Tucker* or KKT conditions. The third conditions are called the *complementary slackness conditions*. A given complementary slackness condition say $\lambda_i f_i(\theta^*) = 0$ mean that when this product is zero and $\lambda_i \neq 0$ we have the original nonlinear constraint $f_i(\theta^*) \geq 0$ *active* i.e. at the optimal point θ^* it is the hard constraint $f_i(\theta^*) = 0$. Given these conditions we next ask how to use them to actually *find* the optimal point θ^*. One approach, that might work for small problems, is to explicitly specify which nonlinear constraints we want to have active that is assume $f_i(\theta^*) = 0$, from some set of i. We can than solve the remaining equations for the respective Lagrange multipliers. To verify that we indeed have a solution we would then need to check that the values computed for these Lagrange multipliers were non-negative. This can be hard to do in general when there are many constraints, since there are many possible sets $f_i(\theta^*) = 0$ to consider. An alternative approach is to express the problem in its *Wolfe Dual Form*. This later form expresses the fact that in the situation where the objective function $J(\theta)$ is convex while the constraint functions $f_i(\theta)$ are concave then the above programming problem is equivalent to a simpler convex *maximization* programming problem

$$\begin{array}{cc} \text{maximize}_{\lambda \geq 0} & \mathcal{L}(\theta; \lambda) \\ \text{subject to} & \dfrac{\partial}{\partial \theta} \mathcal{L}(\theta; \lambda) = 0 \\ \text{and} & \lambda \geq 0 \, . \end{array}$$

The benefit of this later formulation is that the relatively complicated nonlinear inequality constraints of the original problem, $f_i(\theta) \geq 0$, are replaced with the simpler equality constraint $\frac{\partial}{\partial \theta} \mathcal{L}(\theta; \lambda) = 0$ and a maximization over $\lambda \geq 0$. This later problem (if needed) can be solved with more standard convex programming codes.

References

[1] *NETLAB: algorithms for pattern recognition*. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[2] M. H. DeGroot. *Optimal Statistical Decisions*. 2004.

[3] P. A. Devijver and J. Kittler. *Pattern recognition: A statistical approach*. Prentice Hall, 1982.

[4] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[5] I. S. Gradshteyn and I. M. Ryzhik. *Table of Integrals, Series, and Products*. Academic Press, New York, fourth edition, 1965.

[6] W. G. Kelley and A. C. Peterson. *Difference Equations. An Introduction with Applications*. Academic Press, New York, 1991.

[7] F. Kuhl and C. Giardina. Elliptic Fourier Features of a Closed Contour. *Computer Graphics and Image Processing*, 18:236–258, 1982.

[8] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. Prentice–Hall, 1975.

[9] C. Rorres and H. Anton. *Applications of Linear Algebra*. Wiley, 1st edition, 1977.

[10] S. Ross. *A First Course in Probability*. Macmillan, 3rd edition, 1988.

[11] G. Strang. *Linear Algebra and Its Applications*. Brooks/Cole, 3 edition, 1988.