In [126…

```
import numpy as np
```

    - 1.3 - Assume $X^TX = 2I$ and LS estimate $\hat{\theta}_{LS} = [-.75 \;\; 1.3]^T$. Derive expression for $\hat{\theta}_R$ and determine its value.
    - 1.4 - Apply l_1 regulartization using $\lambda = 0.5$. Using IST for one iteration, determine the lowest value of μ that results in **exactly one** component of $\theta^{(i)}$ being zero.
    - 1.5 - Ridge regression ($\lambda = 0.5$). Assume $\sigma^2 = 0.3$ for the noise. How does this relate to the prior distribution of θ? Determine the prior variance.
  - Problem 2 - Linear Filtering (FIR)
    - 2.1 - Given input sequence and filter weight coeff. w, determine the output of the filter at $n = 3$.
    - 2.2 - Derive expressions for correlation functions r_u, r_{du} in terms of r_s, r_x and r_η.
    - 2.3 - Given correlation function values, determine coefficient values w using $l = 3$ and minimum mean square error MMSE.
    - 2.4 - Consider LMS and $\mu = 0.1$ and weights given $n = 2$ as described in problem before, same with input. Assume $d_3 = 3$. Determine the new value of filter coefficient at $n = 3$ using LMS.
    - 2.5 - Consider RLS with $l = 2$. Determine steady-state excess MSE for RLS using $\beta = 0.95$. Determine β that results in lowest excess MSE.
  - Problem 3 - Dictionary learning (ICA)
    - 3.1 - Four microphones, four sources, attentuated by distance d and various filters. Write up mixing matrix A.
    - 3.2 - What is **false** about ICA using Mutual Information?
  - Problem 4 - Hidden Markov Models (HMM)
    - 4.1 - 2-state HMM with sequence of $N = 37$ elements and P_{ij}. What are the most likely transition probabilities? Argue.
    - 4.2 - Given Table, compute probability of being in state $k = 1$ at time $n = 2$ given observations $y_{1:5}$.
  - Problem 5 - Kalman Filtering (AR(3) w. white noise)
    - 5.1 - Given noisy version of signal, setup components (x_n, F, η_n, H, and v_n) such that Kalman filter algorithm can be used.
  - Problem 6 - Kernel methods (Laplacaion and Polynomial kernel $r = 1$)
    - 6.1 - Given figure, which kernel is most appropriate? Explain and relate to representer theorem, ie. what are possible values of θ_n.
    - 6.2 - Kernel ridge regression w. Laplacian using $a = 0.5$ on data fitted using $C = 0.01$. Compute regression value for $\hat{y}$ at $t = 1$, disregard any kernel value below 0.1.
    - 6.3 - Support Vector Regression with ϵ-insensitive loss using $C = 1$. Support vectors are crosses. Identify ϵ used for data-fit.
- Exam 2023
  - Problem 1 - Parameter Estimation

  - 1.1 - Polynomial regression, determine parameters $\hat{\theta}$ that minimize the error.
  - 1.2 - Linear model with non-white Gaussian noise and variance of noise $\sigma_\eta = 1$, covariance two successive samples is 0.2, determine parameters.
  - 1.3 - Compute $\hat{\theta}_{agg}$ across $N = 10$ different realizations using unbiased estimator $\hat{\theta}_i$, where each individual estimator's variance is $\sigma^2 = 1$.
- Problem 2 - Sparse Learning ($l1$/linear regression)
  - 2.1 - IST, determine smallest value of λ that results in **both** components in $\theta^{(i)} = 0$ using $\mu = 0.1$
  - 2.2 - Assume $\lambda = 1$ and reliable estimate of noise $\sigma^2 = 1$. How does linear/LASSO regression relate to prior distribution of θ? Define the parameters. (Laplacian)
- Problem 3 - Linear Filtering (FIR)
  - 3.1 - Determine filter weight w_2 used to find $\hat{y}_4$ given that $n = 4$ and $\hat{y}_4 = 3.1$ and $w = [0.50 \;\; 0.30 \;\; w_2]^T$
  - 3.2 - (WSS) Derive analytical expression for cross-correlation function $r_{yu}(k)$.
  - 3.3 - Given $r_u(k)$ and $r_{du}(k)$ for d_n and u_n determine filter coefficients w using $l = 3$. Also, find highest tolerable value for $r_d(0)$ if the MMSE is at most 0.1.
  - 3.4 - Determine maximum value of step-size μ using LMS and given Σ_u. Ensure that **filter converges** AND weight estimator has **bounded variance**.
  - 3.5 - Using NLMS determine range of μ that ensures the excess MSE is at most 0.1, while NLMS has to guaranteed to remain **stable** as well.
- Problem 4 - Dictionary Learning (ICA)
  - 4.1 - ICA: Given 2 sources s_1 and s_2, identify umixing matrix W such that the average mutual information is zero. Show formally!
  - 4.2 - Write short proof that ICA has scaling ambiguity.
- Problem 5 - Hidden Markov Models (HMM)
  - 5.1 - Suppose 3-state HMM with given P_k and P_{ij} and sub-sequences A, B, C, D. Specify composition that creates most likely state sequence, and justify choices.
  - 5.2 - HMM allows $i = 3$ possible observable actions a_i. Compute initial state probability vector P_k.
- Problem 6 - Kalman filtering
  - 6.1 - Calculate updated state estimate $\hat{x}_1$ and error covariance P_1 after receiving first measurement given initial state and probability (Multiple choice).
- Problem 7 - Kernel methods
  - 7.1 - Gaussian kernel given 8 data points (figure). Write kernel matrix $\mathcal{K}$ for the data using $\sigma = 2$.

- 7.2 - Assume two test points belonging to each of two classes are given. Use representer theorem to determine how these can be classified correctly.
- 7.3 **[NOT DONE]** - Support vector regression on 5 data points using $C = 1$ and $\epsilon = 0.1$ with $\hat{\theta}_0$ = 0 as bias. Compute either exact or tight bound for $\hat{y}(x^{test})$

In [127…

```python
### Helper functions
def J_exc_LMS(Sigma_x, Sigma_omega, sigma_eta_squared, mu, precision=3):
    """Calculate the excess MSE for the LMS algorithm. (Table 6.1)"""
    LHS = 1/2 * mu * sigma_eta_squared * np.trace(Sigma_x)
    RHS = 1/2 * mu**(-1) * np.trace(Sigma_omega)
    return round(LHS + RHS, precision)

def J_exc_RLS(Sigma_x, Sigma_omega, sigma_eta_squared, filter_length, beta, prec
    """Calculate the excess MSE for the RLS algorithm. (Table 6.1)"""
    LHS = 1/2 * (1-beta) * sigma_eta_squared * filter_length
    RHS = 1/2 * (1-beta)**(-1) * np.trace(Sigma_omega @ Sigma_x)
    return round(LHS + RHS, precision)

def J_exc_NLMS(Sigma_x, Sigma_omega, sigma_eta_squared, mu, norm_input, precisic
    """
    Calculate the excess MSE for the NLMS algorithm.
    expectation_q is the expected value of q=1 over the squared norm of the inpu
    If used, please refer to Table 6.1 in the book.
    """
    LHS = 1/2 * mu * sigma_eta_squared * np.trace(Sigma_x) * ( 1 / norm_input )
    RHS = 1/2 * mu**(-1) * np.trace(Sigma_omega) * np.trace(Sigma_x)
    return LHS + RHS

def biased_crosscor(x, y, k):
    """
    Calculate the biased cross-correlation between two signals x and y for lag k
    denoted as ^r_xy(k).It is called auto-correlation if x = y.
    !! See exercise 3.1.3. !!
    """
    N = len(x)
    if len(y) != N:
        raise ValueError("Signals x and y must have the same length.")
    # Compute the biased cross-correlation
    correlation = 0
    for n in range(k, N):
        correlation += x[n] * y[n - k]
        print(f"n={n}: x(n)= {x[n]}, y(n-k)= {y[n - k]}")

    return correlation / N

def gaussian_kernel(x, y, sigma):
    return np.exp( - ((x - y)**2) / (2 * sigma**2) )

def homogeneous_polynomial_kernel(x, y, r):
    return (x.T @ y)**r

def laplacian_kernel(x, y, a):
    return np.exp((a) * np.linalg.norm(np.abs(x-y)))
```

Exam 2021

21 Problem 1

Correlation functions

21 1.1

Consider the following sequence starting at $n = 0$:

$$y_n = \{5, 3, 6, 2\}$$

Calculate the biased auto-correlation value for $r_y(0)$ and $r_y(3)$.

Answer

The biased cross-correlation is defined in exercise 3.1.3 as:

$$\hat{r}_{xy} = \frac{1}{N} \sum_{n=k}^{N-1} x(n)y(n-m), \quad \text{for } k = 0, 1, \dots, N-1$$

Inserting we have:

In [128…

```
y_n = [5, 3, 6, 2]
k = 0
print(f"r_xy({k}) = {biased_crosscor(y_n, y_n, k)}")

k = 3
print(f"r_xy({k}) = {biased_crosscor(y_n, y_n, k)}")
```

```
n=0: x(n)= 5, y(n-k)= 5
n=1: x(n)= 3, y(n-k)= 3
n=2: x(n)= 6, y(n-k)= 6
n=3: x(n)= 2, y(n-k)= 2
r_xy(0) = 18.5
n=3: x(n)= 2, y(n-k)= 5
r_xy(3) = 2.5
```

21 1.2

Consider the signal $x_n = \alpha y_{n-d_1} + \beta z_{n-d_2}$. The signals y_n and z_n are wide-sense stationary processes (WSS). Assume $r_y(k)$, $r_z(k)$ and $r_yz(k)$ are known.

Determine an analytic expression of the auto-correlation function $r_x(k)$

Answer

Exercise 3.1.1 has the same structure. Recall, WSS processes can be freely time-shifted! We have to expand the following:

$$\begin{aligned} r_x(k) &= \mathbb{E}[x_n x_{n-k}] \\ &= \mathbb{E}[(\alpha y_{n-d_1} + \beta z_{n-d_2})(\alpha y_{n-d_1-k} + \beta z_{n-d_2-k})] \end{aligned}$$

Now one can either $d = d_2 - d_1$ to get the same form as exercise 3.1.1, or can take the long way:

$$\begin{aligned} r_x(k) &= \mathbb{E}[x_n x_{n-k}] \\ &= \mathbb{E}[(\alpha y_{n-d_1} + \beta z_{n-d_2})(\alpha y_{n-d_1-k} + \beta z_{n-d_2-k})] \\ &= \mathbb{E}[\alpha^2 y_{n-d_1} y_{n-d_1-k} + \beta^2 z_{n-d_2} z_{n-d_2-k} + \alpha\beta y_{n-d_1} z_{n-d_2-k} + \alpha\beta z_{n-} \\ &= \mathbb{E}[\alpha^2 y_n y_{n-k} + \beta^2 z_n z_{n-k} + \alpha\beta y_{n-d_1+d_1} z_{n-d_2-k+d_1} + \alpha\beta z_{n-d_2+d_2} y \\ &= \mathbb{E}[\alpha^2 y_n y_{n-k} + \beta^2 z_n z_{n-k} + \alpha\beta y_n z_{n-(d_2+k-d_1)} + \alpha\beta z_n y_{n-(d_1+k-d_2)} \\ &= \alpha^2 r_y(k) + \beta^2 r_z(k) + \alpha\beta r_{yz}(d_2 + k - d_1) + \alpha\beta r_{zy}(d_1 + k - d_2) \\ &= \alpha^2 r_y(k) + \beta^2 r_z(k) + \alpha\beta r_{yz}(d_2 + k - d_1) + \alpha\beta r_{yz}(d_2 - k - d_1) \end{aligned}$$

Recall that the following property holds for cross-correlation (property 2.120) in ML book:

$$r_{uv}(k) = r^*_{vu}(-k)$$

Since both y_n and y_{n-k} (same for z) are shifted by d_1 and d_2, respectively, we can simply shift them by the same amount when it only concerns the two expressions. Note, for the rest, we want a form of $\mathbb{E}[\cdot_n \cdot_{n-(k)}]$, so we can shift the entire sequence by adding $+d_1$ and $+d_2$ onto the y and z component, respectively. See above. In the cross-over, we add either $+d_1$ or $+d_2$ to the entire sequence dependent on which one yields the form above.

The parenthesis is made and we **CHANGE THE SIGN**. Then we use linearity of expectations and recognize each element as their respective correlation functions.

21 1.3

Consider same x_n. Assume $y_n \sim AR(1)$, ie. $y_n = a_1 y_{n-1} + \eta_n$. z_n is a random variable taking *one* value and is drawn from $\text{Uniform}(0, 1)$. Assume $\mathbb{E}[y_n] = 0$

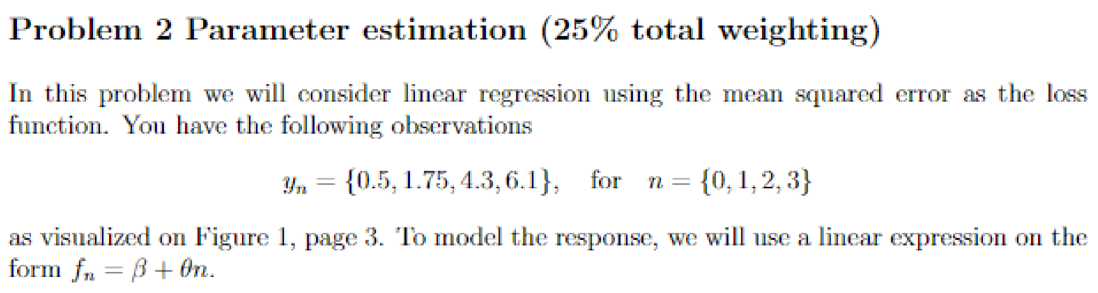
Determine if it is a second order ergodic process.

Answer

- The signal is WSS (mean and auto-correlation depend only on time differences)
  - ie. must be stationary
- Time average for mean and auto-correlation converge to respective ensemble averages.
- Has finite variance.

Since z_n is a single realization, its mean and auto-correlation will become non-stationary as the specific time realizations will have different mean values. In other words z_n is not random across realizations, but fixed ($z_n = z_{n+1}$). Hence, x_n is not WSS, which is a prerequisite for second order ergodic processes.

21 Problem 2



Problem 2 Parameter estimation (25% total weighting)

In this problem we will consider linear regression using the mean squared error as the loss function. You have the following observations

$$y_n = \{0.5, 1.75, 4.3, 6.1\}, \quad \text{for} \quad n = \{0, 1, 2, 3\}$$

as visualized on Figure 1, page 3. To model the response, we will use a linear expression on the form $f_n = \beta + \theta n$.

21 2.1

Determine the least squares (LS) estimate of the coefficients b and θ. Write up the matrices and vectors used to calculate the estimate.

Answer

Recall:

$$\hat{\theta}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Remember the design matrix X contains to the bias and the number of predictors. The model response was $f_n = \beta + \theta n$, hence the predictors are simply n

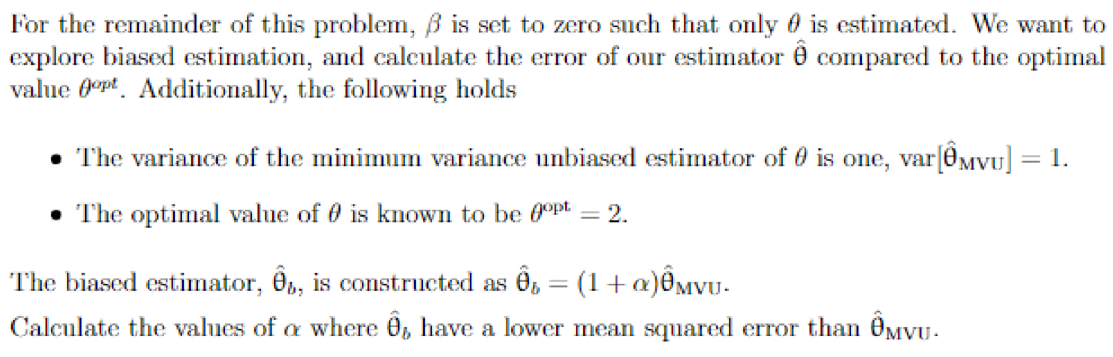
In [129…

```
y_n = np.array([0.5, 1.75, 4.3, 6.1])
n = np.arange(len(y_n))
bias = np.ones(len(y_n))
X = np.array([bias, n]).T
```

```
print(f"X = \n{X}")
print(f"y_n = {y_n}")
print(f"LS estimate:\n[b, theta] = {np.linalg.inv(X.T @ X) @ X.T @ y_n}")
```

```
X = 
[[1. 0.]
 [1. 1.]
 [1. 2.]
 [1. 3.]]
y_n = [0.5  1.75 4.3  6.1 ]
LS estimate:
[b, theta] = [0.26  1.935]
```

21 2.2



For the remainder of this problem, β is set to zero such that only θ is estimated. We want to explore biased estimation, and calculate the error of our estimator $\hat{\theta}$ compared to the optimal value θ^{opt}. Additionally, the following holds

- The variance of the minimum variance unbiased estimator of θ is one, $\mathrm{var}[\hat{\theta}_{\mathrm{MVU}}] = 1$.
- The optimal value of θ is known to be $\theta^{\mathrm{opt}} = 2$.

The biased estimator, $\hat{\theta}_b$, is constructed as $\hat{\theta}_b = (1+\alpha)\hat{\theta}_{\mathrm{MVU}}$.

Calculate the values of α where $\hat{\theta}_b$ have a lower mean squared error than $\hat{\theta}_{\mathrm{MVU}}$.

Answer

Recall that we may find the optimal parameters that are biased such as $\hat{\theta}_b = (1+\alpha)\hat{\theta}_u$ then value of α where our biased estimater has a lower MSE than its unbiased counterpart is given by (*ML 3.5.1, (3.21) and (3.26)*):

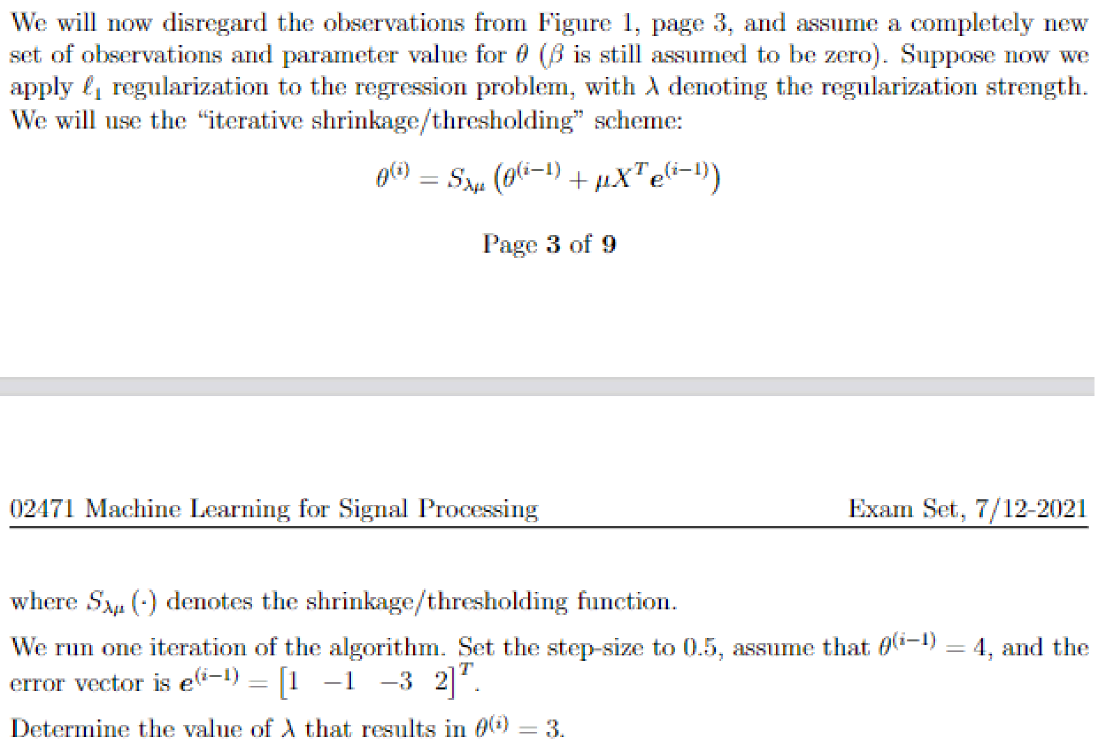
$$-\frac{2\mathrm{MSE}(\hat{\theta}_u)}{\mathrm{MSE}(\hat{\theta}_u)+\theta_o^2} < \alpha < 0$$

where we have $\mathrm{VAR}[\hat{\theta}_{MVU}]$ and θ_o, so:

$$-\frac{2\mathrm{MSE}(\hat{\theta}_u)}{\mathrm{MSE}(\hat{\theta}_u)+\theta_o^2} = -\frac{2\cdot 1}{1+2^2} = -\frac{2}{5} < \alpha < 0$$

21 2.3

keywords: (IST, iterative, shrinkage, thresholding, λ, l_1 regularization, step-size)



We will now disregard the observations from Figure 1, page 3, and assume a completely new set of observations and parameter value for θ (β is still assumed to be zero). Suppose now we apply ℓ_1 regularization to the regression problem, with λ denoting the regularization strength. We will use the "iterative shrinkage/thresholding" scheme:

$$\theta^{(i)} = S_{\lambda\mu}\left(\theta^{(i-1)} + \mu X^T \boldsymbol{e}^{(i-1)}\right)$$

Page **3** of **9**

02471 Machine Learning for Signal Processing Exam Set, 7/12-2021

where $S_{\lambda\mu}(\cdot)$ denotes the shrinkage/thresholding function.

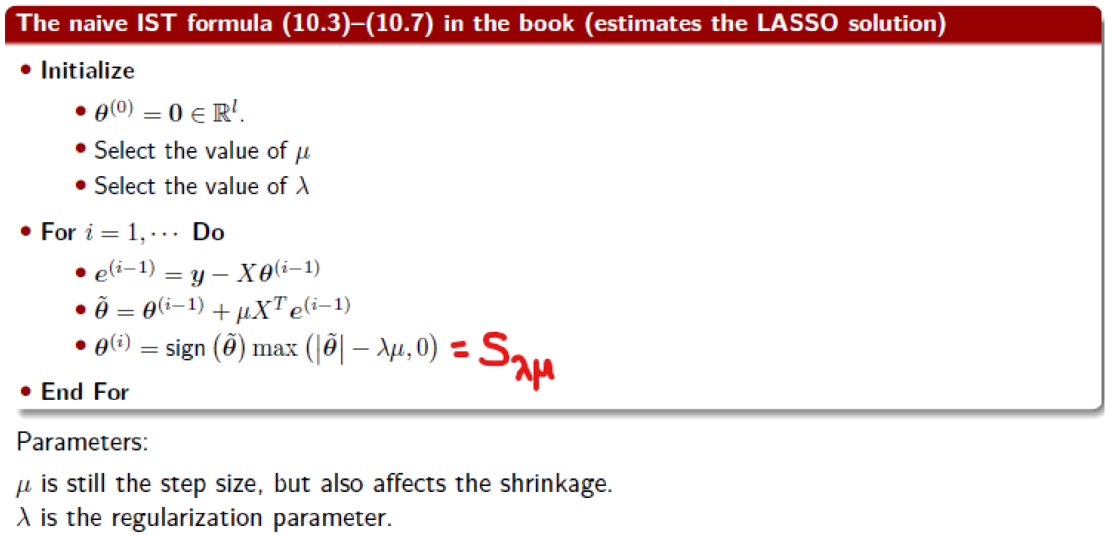
We run one iteration of the algorithm. Set the step-size to 0.5, assume that $\theta^{(i-1)} = 4$, and the error vector is $\boldsymbol{e}^{(i-1)} = \begin{bmatrix}1 & -1 & -3 & 2\end{bmatrix}^T$.

Determine the value of λ that results in $\theta^{(i)} = 3$.

Answer

NOTE $X^TX = I$ IS NOT SPECIFIED SO WE USE THE BELOW, OTHERWISE WE WOULD USE FORMULA FROM ex 6.2 OR chapter 9.3

$\theta^{(i)} = S_{\lambda\mu}(\tilde{\theta})$ is given by (10.7) in the book, where $S_{\lambda\mu}(\tilde{\theta})$ is given by slides as :



The naive IST formula (10.3)–(10.7) in the book (estimates the LASSO solution)

```
• Initialize
    • $\boldsymbol{\theta}^{(0)} = \mathbf{0} \in \mathbb{R}^l$.
    • Select the value of $\mu$
    • Select the value of $\lambda$
• For $i = 1, \cdots$ Do
    • $\boldsymbol{e}^{(i-1)} = \boldsymbol{y} - X\boldsymbol{\theta}^{(i-1)}$
    • $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}^{(i-1)} + \mu X^T \boldsymbol{e}^{(i-1)}$
    • $\boldsymbol{\theta}^{(i)} = \text{sign}(\tilde{\boldsymbol{\theta}}) \max(|\tilde{\boldsymbol{\theta}}| - \lambda\mu, 0)$ $= S_{\lambda\mu}$
• End For
```

Parameters:

μ is still the step size, but also affects the shrinkage.
λ is the regularization parameter.

Hence:

$$\begin{aligned}
\theta^{(i)} &= S_{\lambda\mu}(\tilde{\theta}) = S_{\lambda\mu}(\theta^{(i-1)} + \mu X^T e^{(i-1)}) \\
&= S_{\lambda\mu}(4 + 0.5 \cdot [0 \quad 1 \quad 2 \quad 3][1 \quad -1 \quad -3 \quad 2]^T) \\
&= S_{\lambda\mu}(3.5) \\
&= \text{sign}(\theta)\max(|\theta| - \lambda\mu, 0) = \text{sign}(3.5)\max(|3.5| - \lambda 0.5, 0)
\end{aligned}$$

In [130…

```
step = 0.5
theta_prev = 4
error_prev = np.array([1, -1, -3, 2]).T
X = np.array([0, 1, 2, 3]).T
S = (theta_prev + step * X.T @ error_prev)
for reg_choice in range(0,3):
    theta_i = np.sign(S) * np.maximum(np.abs(S) - reg_choice * step, 0)
    print(f"lambda={reg_choice}: {theta_i}")
```

```
lambda=0: 3.5
lambda=1: 3.0
lambda=2: 2.5
```

21 2.4

Suppose now we apply ℓ_2 regularization to our problem, with λ denoting the regularization strength of the ℓ_2. Suppose that we choose $\lambda = 2$, and we interpret Ridge regression from a Bayesian viewpoint. In this viewpoint, we choose a distribution on our θ with a variance set to 0.5. Answer the two questions:

- What does these parameters indicate about our assumption of the statistical properties of measurement noise?
- What is the assumed value of the measurement noise?

Answer

You are supposed to read off that this pertains to Gaussian white noise (exercise 9.1.2). ML 12.2.2 p598 under (12.12) a small note says $\lambda = \frac{\sigma_\eta^2}{\sigma_\theta^2}$

- Measurement noise is Gaussian white noise which is statistically independent from sample to sample, ie $\eta \sim \mathcal{N}(\mu_\eta, \Sigma_\eta)$ with uncorrelated residuals
- $\lambda = 2 = \frac{\sigma_\eta^2}{\sigma_\theta^2} \Leftrightarrow \sigma_\eta^2 = 2 \cdot \sigma_\theta^2 = 2 \cdot 0.5 = 1$

21 2.5

Now assume that we take a Bayesian approach, and have a likelihood that is Gaussian with mean $X\theta$ and covariance $\sigma_\eta^2 I$, and a prior distribution on θ that is Gaussian with zero mean and variance denoted as σ_θ^2.

The measurement noise can now be estimated using the Expectation-Maximization algorithm by specifying the complete log-likelihood

$$\ln p(\boldsymbol{y}, \theta; \boldsymbol{\xi}), \quad \boldsymbol{\xi} = \{\sigma_\eta^2, \sigma_\theta^2\}$$

Determine an expression for $\ln p(\boldsymbol{y}, \theta; \boldsymbol{\xi})$ as a function of $\sigma_\eta^2, \sigma_\theta^2$. Define the meaning of other terms that is in your expression.

Answer

Look at ML p614, where it is noted we instead would use precision variables. Hence $\alpha = \frac{1}{\sigma_\theta^2}$ and $\beta = \frac{1}{\sigma_\eta^2}$. Inserting we then have:

Relating the expression to the setup in the problem, we get

$$\ln p(\boldsymbol{y}, \theta; \sigma_\theta^2, \sigma_\eta^2) = \frac{N}{2}\ln\frac{1}{\sigma_\eta^2} + \frac{K}{2}\ln\frac{1}{\sigma_\theta^2} - \frac{1}{2\sigma_\eta^2}\|\boldsymbol{y} - \boldsymbol{x}\theta\|^2 - \frac{1}{2\sigma_\theta^2}\theta^2 - \left(\frac{N}{2} + \frac{K}{2}\right)\ln(2\pi)$$

Recall K is the number of weights and N is the number of points, ie. $K = 1$ and $N = 4$, so:

$$\ln p(\boldsymbol{y}, \theta; \sigma_\theta^2, \sigma_\eta^2) = 2\ln\frac{1}{\sigma_\eta^2} + \frac{1}{2}\ln\frac{1}{\sigma_\theta^2} - \frac{1}{2\sigma_\eta^2}\|\boldsymbol{y} - \boldsymbol{x}\theta\|^2 - \frac{1}{2\sigma_\theta^2}\theta^2 - \frac{5}{2}\ln(2\pi)$$

21 Problem 3

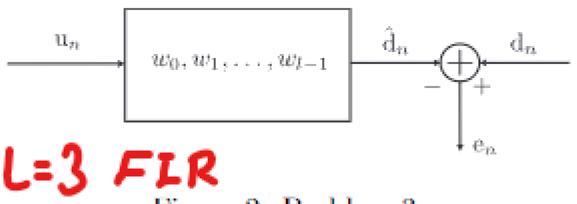


Figure 2: Problem 3.

21 3.1

You get the following correlation functions:

$r_u(0)$	$r_u(1)$	$r_u(2)$	$r_u(3)$
1.7	0.7	0.4	0.2

$r_{du}(0)$	$r_{du}(1)$	$r_{du}(2)$	$r_{du}(3)$
1.2	0.6	0.35	0.05

$r_d(0)$	$r_d(1)$	$r_d(2)$	$r_d(3)$
1.4	1.0	0.7	0.3

Determine the filter coefficient values w and the minimum mean squared error (MMSE) as achieved by the filter.

Answer

This is the exact system from Figure 4.5 in the ML book p. 135, where we have a new form of the normal equation (4.5) given by p. 136 as:

$$\Sigma_u w = R_u w = p \Leftrightarrow w = R_u^{-1} p$$

Since we have been given R_u and p, we can readily find that below.

The MMSE is given by eq. (4.9) in the book p. 123 as:

$$MMSE = J(\theta_*) = \sigma_y^2 - p^T\theta_*$$

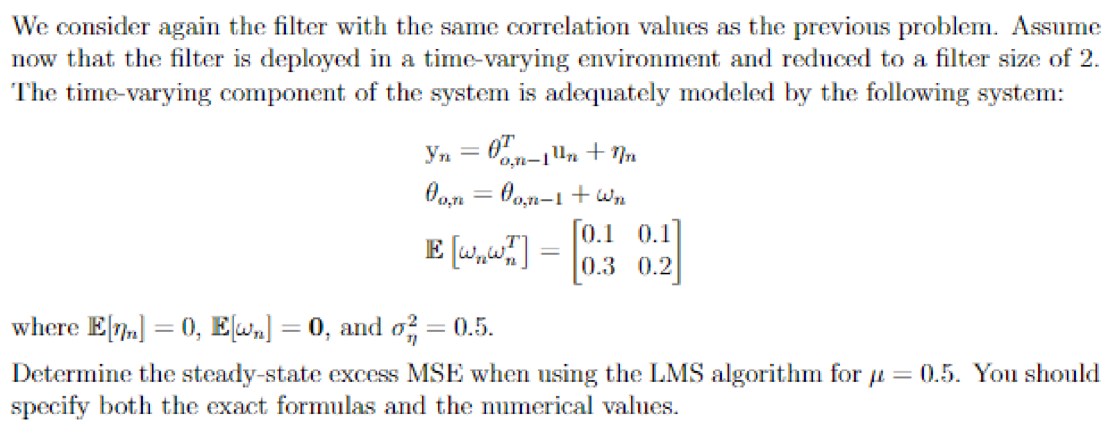
However, here we remark that w is indeed the parameter, so $\theta_* = w_*$ and $\sigma_y^2 = \sigma_d^2 = r_d(0)$

In [131…

```
np.set_printoptions(3)
ru0=1.7; ru1=0.7; ru2=0.4; ru3=0.2
rdu0=1.2; rdu1=0.6; rdu2=0.35
R_u = np.array([[ru0, ru1, ru2],
                [ru1, ru0, ru1],
                [ru2, ru1, ru0]])
p = np.array([[rdu0],
              [rdu1],
              [rdu2]])
print(f"R_u = \n{R_u}")
print(f"p = \n{p}")
w = np.linalg.inv(R_u) @ p
print(f"w = \n{w}")

rd0 = 1.4 # This is the same as \sigma^2_d
MMSE = rd0 - w.T @ p
print(f"MMSE = {MMSE}")
```

```
R_u =
[[1.7 0.7 0.4]
 [0.7 1.7 0.7]
 [0.4 0.7 1.7]]
p =
[[1.2 ]
 [0.6 ]
 [0.35]]
w =
[[0.673]
 [0.068]
 [0.02 ]]
MMSE = [[0.544]]
```

21 3.2



We consider again the filter with the same correlation values as the previous problem. Assume now that the filter is deployed in a time-varying environment and reduced to a filter size of 2. The time-varying component of the system is adequately modeled by the following system:

$$\begin{aligned} y_n &= \theta_{o,n-1}^T u_n + \eta_n \\ \theta_{o,n} &= \theta_{o,n-1} + \omega_n \\ \mathbb{E}\left[\omega_n \omega_n^T\right] &= \begin{bmatrix} 0.1 & 0.1 \\ 0.3 & 0.2 \end{bmatrix} \end{aligned}$$

where $\mathbb{E}[\eta_n] = 0$, $\mathbb{E}[\omega_n] = \mathbf{0}$, and $\sigma_\eta^2 = 0.5$.

Determine the steady-state excess MSE when using the LMS algorithm for $\mu = 0.5$. You should specify both the exact formulas and the numerical values.

Answer

Recognize $l = 2$ now. Hence Σ_u changes! The steady-state excess can be found in the ML book in table 6.1, p276:

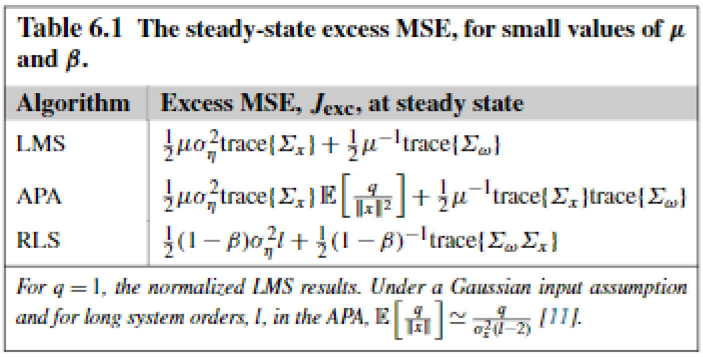


Table 6.1 The steady-state excess MSE, for small values of μ and β.

Algorithm	Excess MSE, J_{exc}, at steady state
LMS	$\frac{1}{2}\mu\sigma_\eta^2\text{trace}\{\Sigma_x\} + \frac{1}{2}\mu^{-1}\text{trace}\{\Sigma_\omega\}$
APA	$\frac{1}{2}\mu\sigma_\eta^2\text{trace}\{\Sigma_x\}\mathbb{E}\left[\frac{q}{\lVert x\rVert^2}\right] + \frac{1}{2}\mu^{-1}\text{trace}\{\Sigma_x\}\text{trace}\{\Sigma_\omega\}$
RLS	$\frac{1}{2}(1-\beta)\sigma_\eta^2 l + \frac{1}{2}(1-\beta)^{-1}\text{trace}\{\Sigma_\omega\Sigma_x\}$

For $q = 1$, the normalized LMS results. Under a Gaussian input assumption and for long system orders, l, in the APA, $\mathbb{E}\left[\frac{q}{\lVert x\rVert}\right] \simeq \frac{q}{\sigma_x^2(l-2)}$ [11].

Hence we get, using y_n which takes input u_n:

$$\begin{aligned} J_{exc} &\simeq \frac{1}{2}\mu\sigma_\eta^2\text{trace}\{\Sigma_x\} + \frac{1}{2}\mu^{-1}\text{trace}\{\Sigma_\omega\} \\ &= \frac{1}{2}\cdot 0.5\cdot 0.5\cdot (r_u(0) + r_u(0)) + \frac{1}{2}\cdot\left(\frac{1}{2}\right)^{-1}\cdot(0.1+0.2) \\ &= 0.725 \end{aligned}$$

In [132…

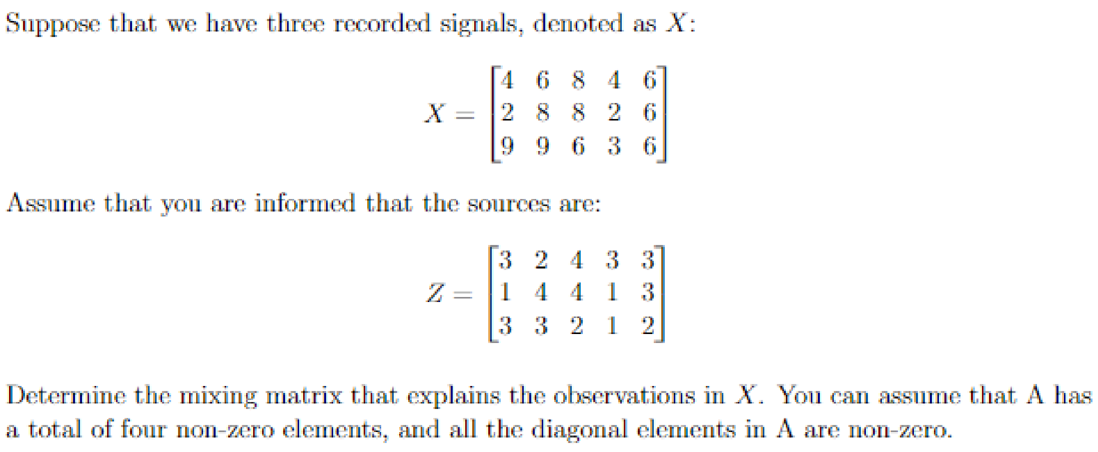
```
J_exc = 1/2 * 1/2 * 1/2 * (1.7 + 1.7) + 1/2 * (1/2)**(-1) * (0.1 + 0.2)
l = 2; mu = 1/2; sigma_eta_squared = 0.5
Sigma_omega = np.array([[0.1, 0.1],
                        [0.3, 0.2]])
J_exc_LMS = J_exc_LMS(R_u[:l, :l], Sigma_omega, sigma_eta_squared, mu, precision
print(J_exc)
print(J_exc_LMS)
```

```
0.7250000000000001
0.725
```

21 Problem 4

Independent Component Analysis (ICA)

21 4.1



Suppose that we have three recorded signals, denoted as X:

$$X = \begin{bmatrix} 4 & 6 & 8 & 4 & 6 \\ 2 & 8 & 8 & 2 & 6 \\ 9 & 9 & 6 & 3 & 6 \end{bmatrix}$$

Assume that you are informed that the sources are:

$$Z = \begin{bmatrix} 3 & 2 & 4 & 3 & 3 \\ 1 & 4 & 4 & 1 & 3 \\ 3 & 3 & 2 & 1 & 2 \end{bmatrix}$$

Determine the mixing matrix that explains the observations in X. You can assume that A has a total of four non-zero elements, and all the diagonal elements in A are non-zero.

Answer

Recall in ICA that the mixing matrix A is part of the equation given in slide 29 week 8 (although given by (19.44) as $Z = s$):

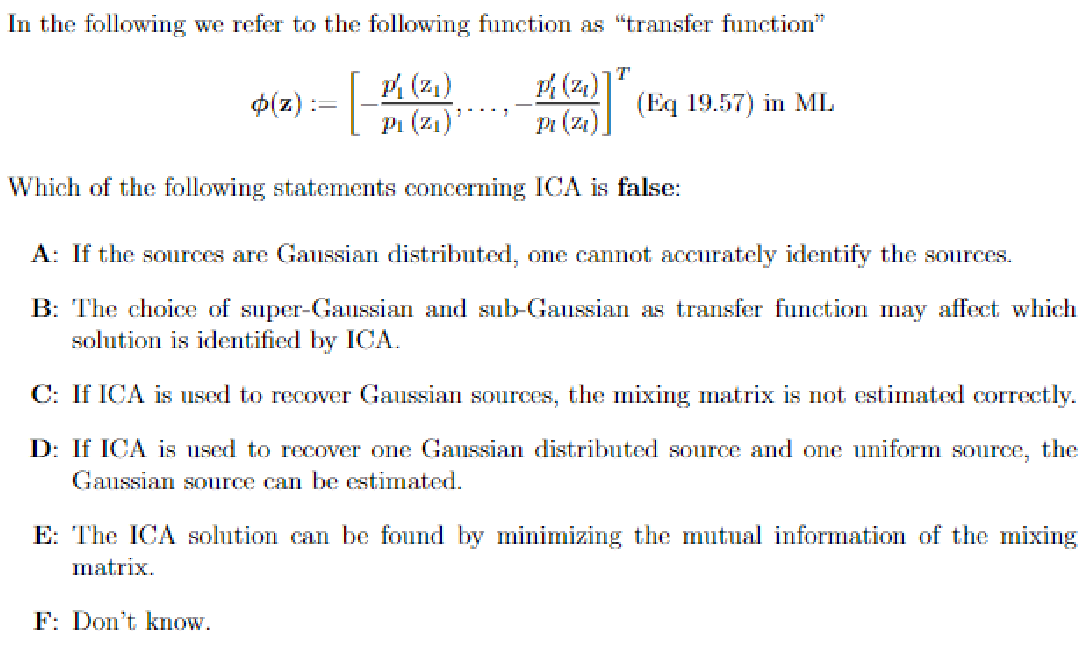
$$X = AZ \Leftrightarrow XZ^T = AZZ^T \Leftrightarrow XZ^T(ZZ^T)^{-1} = A$$

In [133…

```
np.set_printoptions(suppress=True)
X = np.array([[4, 6, 8, 4, 6],
              [2, 8, 8, 2, 6],
              [9, 9, 6, 3,6]])
Z = np.array([[3, 2, 4, 3, 3],
              [1, 4, 4, 1, 3],
              [3, 3, 2, 1, 2]])
A = X@Z.T @ np.linalg.inv(Z@Z.T)
print(f"A = \n{A}")
```

```
A =
[[ 1.  1.  0.]
 [-0.  2.  0.]
 [-0.  0.  3.]]
```

21 4.2



In the following we refer to the following function as "transfer function"

$$\phi(\mathbf{z}) := \left[-\frac{p_1'(z_1)}{p_1(z_1)}, \ldots, -\frac{p_l'(z_l)}{p_l(z_l)}\right]^T \text{(Eq 19.57) in ML}$$

Which of the following statements concerning ICA is **false**:

A: If the sources are Gaussian distributed, one cannot accurately identify the sources.

B: The choice of super-Gaussian and sub-Gaussian as transfer function may affect which solution is identified by ICA.

C: If ICA is used to recover Gaussian sources, the mixing matrix is not estimated correctly.

D: If ICA is used to recover one Gaussian distributed source and one uniform source, the Gaussian source can be estimated.

E: The ICA solution can be found by minimizing the mutual information of the mixing matrix.

F: Don't know.

Answer

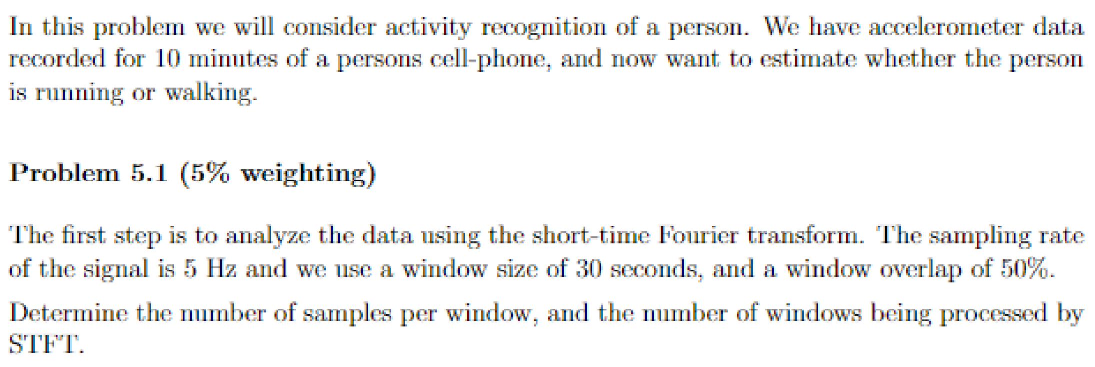
- If the sources are Gaussian distributed we **can** accurately identify the sources. In other cases, maybe not.
- Super/sub-Gaussian transfer function does indeed affect which solution ICA finds, e.g. in slides and the exercises.
- Using ICA for Gaussian sources the mixing matrix A is easily estimated correctly.
- ICA for Gaussian source and uniform source, the Gaussian can be estimated always.

- ICA solution minimizes the mutual information (MI) of the sources (z), not the mixing matrix! Thus this is incorrect.

21 Problem 5

State-space models and time-frequency analysis.

21 5.1



In this problem we will consider activity recognition of a person. We have accelerometer data recorded for 10 minutes of a persons cell-phone, and now want to estimate whether the person is running or walking.

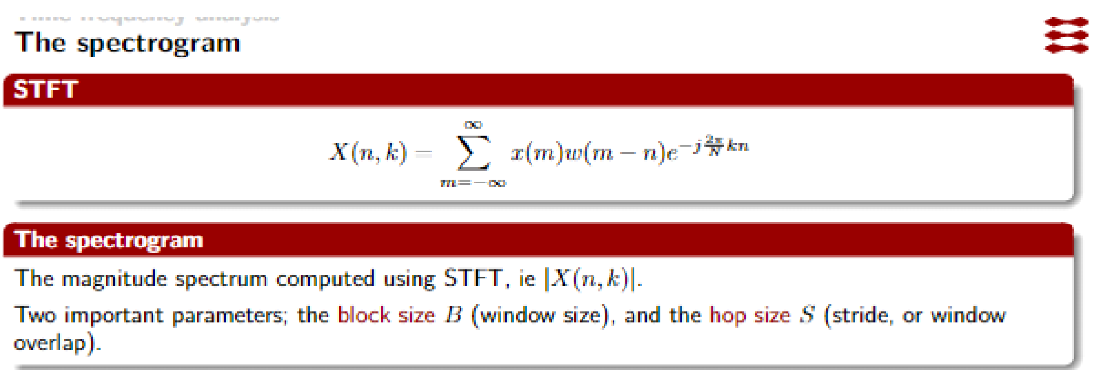
Problem 5.1 (5% weighting)

The first step is to analyze the data using the short-time Fourier transform. The sampling rate of the signal is 5 Hz and we use a window size of 30 seconds, and a window overlap of 50%.

Determine the number of samples per window, and the number of windows being processed by STFT.

Answer

See slide 33 week 7, which states the STFT is given by:



The spectrogram

STFT

$$X(n,k) = \sum_{m=-\infty}^{\infty} x(m)w(m-n)e^{-j\frac{2\pi}{N}kn}$$

The spectrogram

The magnitude spectrum computed using STFT, ie $|X(n,k)|$.

Two important parameters; the block size B (window size), and the hop size S (stride, or window overlap).

In other words, the signal x is 10 minutes so $10 \times 60 = 600s$ and sampled at a rate of $F_s = 5Hz$ so there is a total of $600s \times 5Hz = 3000$ samples.

Furthermore, we identify that the window size is $B = 30s$ and the hop size $S = 50\%$.

Recall, the number of samples per window is:

$$\text{Samples per window} = B \times F_s = 30 \times 5 = 150\text{samples}$$
$$\text{Hop size in samples} = \text{Samples per window} \times \frac{1}{2} = \frac{150}{2} = 75\text{samples}$$
$$\text{No. windows} = \frac{\text{Total samples} - \text{Samples per window}}{\text{Hop Size}} + 1$$
$$= \frac{3000-150}{75} + 1 = 38 + 1 = 39.$$

Alternatively the solution found:

$$\text{No. windows} = \frac{\text{Total signal length}}{\text{Window size} \times \text{Window overlap}} - 1$$

In [134…

```
signal_seconds = 10 * 60
window_size = 30
hop_size = 1/2
nr_windows = (signal_seconds / (window_size * hop_size)) - 1
print(f"Number of windows: {nr_windows}")
```

```
Number of windows: 39.0
```

21 5.2

Problem 5.2 (5% weighting)

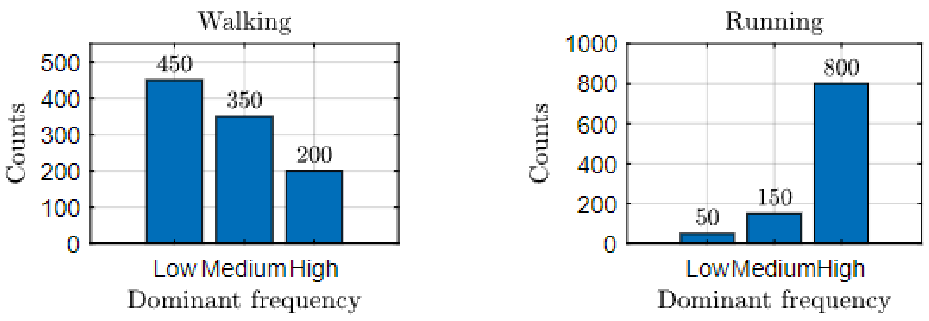


Figure 3: Problem 5.2.

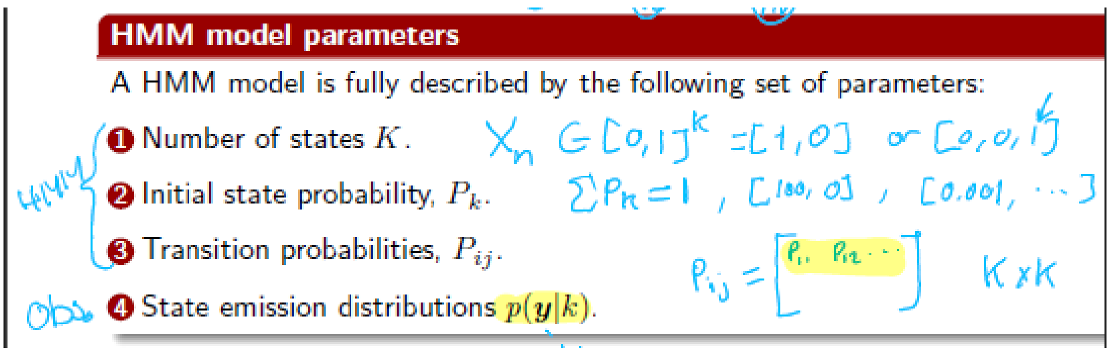
We now consider a Hidden Markov model for the activity recognition of the person.

For this task, we collect five hours of walking data and five hours of running data, and segment the data into 1000 bins of each class. For each time segment, we calculate the total energy in three frequency ranges (low, medium and high frequency) and record the frequency range with the highest energy. The counts are shown on Figure 3, page 7.

Estimate as many parameter of the Hidden Markov model as possible with the information you currently have.

Answer

HMM is fully described by the parameters:



HMM model parameters

A HMM model is fully described by the following set of parameters:

HMM
1. Number of states K. $X_n \in [0,1]^K = [1,0]$ or $[0,0,1]$ K
2. Initial state probability, P_k. $\sum P_k = 1$, $[100, 0]$, $[0.001, \cdots]$
3. Transition probabilities, P_{ij}. $P_{ij} = \begin{bmatrix} P_{11} & P_{12} \cdots \\ & \end{bmatrix}$ $K \times K$

Obs 4. State emission distributions $p(y|k)$.

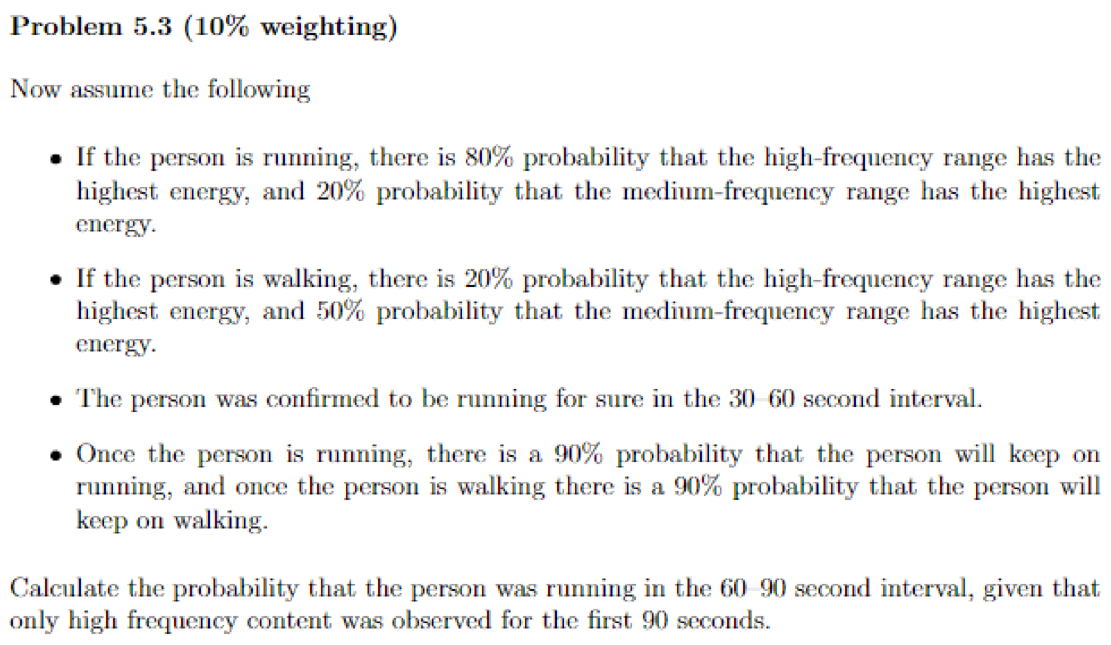
So we can recognize there are $K = 2$ total states (walking/running). The initial state probabilities P_k can be identified as:

$$P_k(Y|k=\text{Walking}) = \begin{bmatrix} \frac{450}{1000} & \frac{350}{1000} & \frac{200}{1000} \end{bmatrix}$$

$$P_k(Y|k=\text{Running}) = \begin{bmatrix} \frac{50}{1000} & \frac{150}{1000} & \frac{800}{1000} \end{bmatrix}$$

We cannot estimate any other parameters with the given information provided.

21 5.3



Problem 5.3 (10% weighting)

Now assume the following

- If the person is running, there is 80% probability that the high-frequency range has the highest energy, and 20% probability that the medium-frequency range has the highest energy.
- If the person is walking, there is 20% probability that the high-frequency range has the highest energy, and 50% probability that the medium-frequency range has the highest energy.
- The person was confirmed to be running for sure in the 30–60 second interval.
- Once the person is running, there is a 90% probability that the person will keep on running, and once the person is walking there is a 90% probability that the person will keep on walking.

Calculate the probability that the person was running in the 60–90 second interval, given that only high frequency content was observed for the first 90 seconds.

Answer

Here the trick is to use Markov property, ie. we only need the previous state to determine the present. In other words, we discard the observations for the first 60 seconds. The initial probability $P_{running} = 1$ then.

We seek:

$$P(x_n = \text{running}|y_n = \text{high frequency})$$

We can find the probability described in ML ch. 16.5.1, p.850/851, where we do not have future data β, hence with message passing function $\alpha()$:

$$P(x_n|y_{[1:n]}) = \frac{\alpha(x_n)}{p(y_{[1:n]})} = \frac{\alpha(x_1=\text{running})}{p(y_1)}$$
$$= \frac{\alpha(x_1=\text{running})}{\alpha(x_1=\text{walking})+\alpha(x_1=\text{running})}$$

Given (16.41) we have the $\alpha(x_n) = p(y_1, y_2, \ldots, y_n, x_n)$ so:

$$\alpha(x_1) = p(y_1, x_1) = p(y_1|x_1)p(x_1)$$
$$\Rightarrow$$
$$(1) \quad \alpha(x_1=r) = p(y_1=hf|x_1=r)p(x_1=r)$$
$$(2) \; \alpha(x_1=w) = p(y_1=hf|x_1=w)p(x_1=w)$$

where r is running, w is walking and hf is high frequency.

We know the person is running, so the transition probabilities P_{rw} can be found:

$$\alpha(x_1 = r) = p(y_1 = hf|x_1 = r)p(x_1 = r) = 0.8 \times 0.9 = 0.72$$
$$\alpha(x_1 = w) = p(y_1 = hf|x_1 = w)p(x_1 = w) = 0.2 \times 0.1 = 0.02$$

where we have used the information that the previous state we were for sure running, ie. $p(w) = 1 - 0.9$ and $p(r) = 0.9$ in the problem info. So:

$$\begin{aligned}P(x_1 = r|y_1 = hf) &= \frac{\alpha(x_1 = r)}{\alpha(x_1 = w) + \alpha(x_1 = r)} \\ &= \frac{0.72}{0.02 + 0.72} \\ &= 0.973\end{aligned}$$

In [135…

```
round(0.72 / (0.02+0.72),3)
```

Out[135…

```
0.973
```

21 Problem 6

In this problem we consider first a kernel function for a classification problem, and then kernel regression. For all problems, we will use the Gaussian kernel defined as

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2}\right)$$

21 6.1

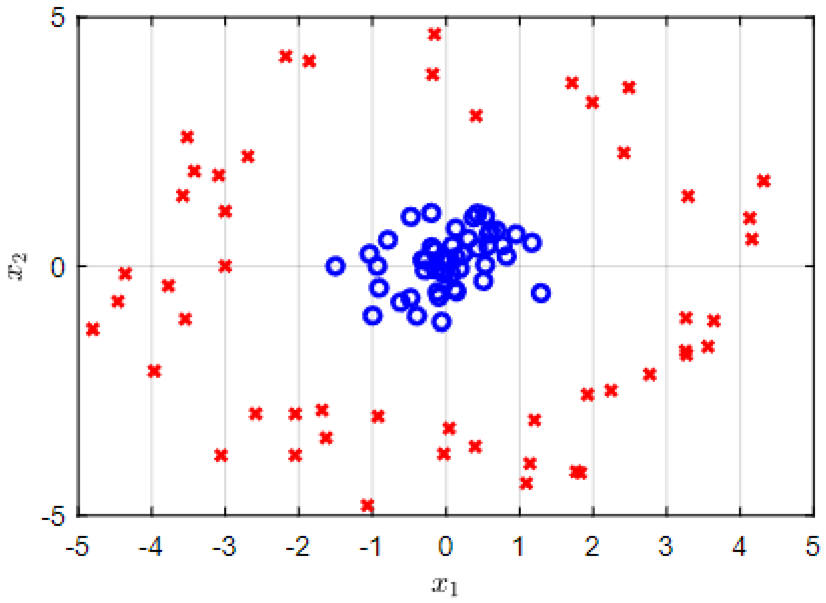


Figure 4: Problem 6.1

Consider the points in the figure as training points in a dataset where red crosses are class one and blue are class two. Assume you observe a test point of class blue $x_{test} = (-2, 0)$.

- Apply kernel function to the closest points of each class in the data set using $\sigma = 1$.
- Argue whether the kernel can be useful or not for seperating the two classes.

Answer

The two closest points of each class can be read off as :

$$x^{train}_{red} = (-3, 0) \;\; , \;\; x^{train}_{blue} = (-1.5, 0)$$
$$\kappa(x_{test}, x^{train}_{red}) = \exp\left(-\frac{||(-2,0)-(-3,0)||^2}{2\cdot 1^2}\right)$$
$$= \exp\left(-\frac{||(-2-(-3))||^2}{2}\right) = \exp\left(\frac{1^2}{2}\right) =$$

```
In [136…
```

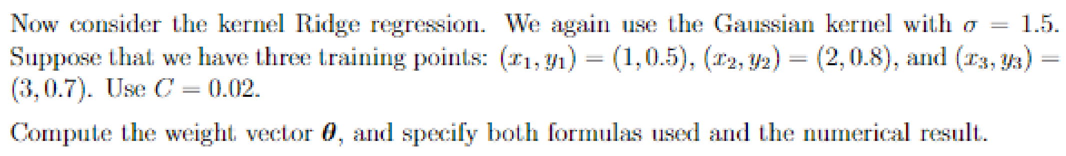
```python
x_test = np.array([-2,0])
x_red = np.array([-3,0])
x_blue = np.array([-1.5, 0])
sigma = 1
np.set_printoptions(precision=2)

print(gaussian_kernel(x_red, x_test, sigma))
print(gaussian_kernel(x_blue, x_test, sigma))
```

```
[0.61 1.  ]
[0.88 1.  ]
```

Since the kernel κ assigns higher value for points of the correct class it can be used.

Much like the exercise 12.1.X and slide 15 week 12 it is definitely linearly seperable in RKHS with the representer theorem.

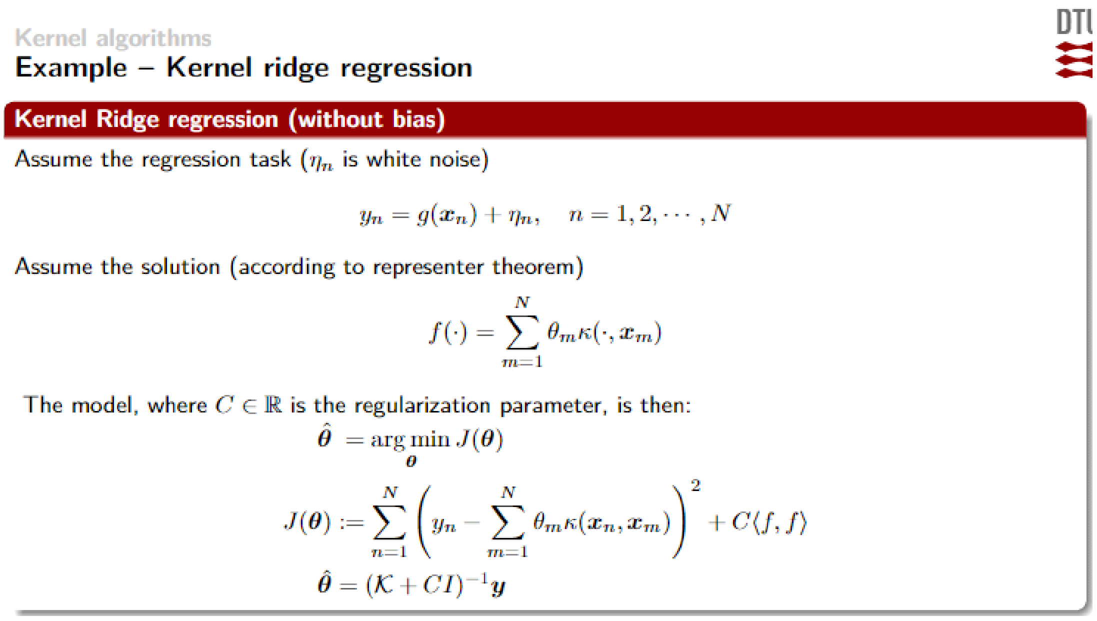
21 6.2


Now consider the kernel Ridge regression. We again use the Gaussian kernel with $\sigma = 1.5$. Suppose that we have three training points: $(x_1, y_1) = (1, 0.5)$, $(x_2, y_2) = (2, 0.8)$, and $(x_3, y_3) = (3, 0.7)$. Use $C = 0.02$.

Compute the weight vector $\boldsymbol{\theta}$, and specify both formulas used and the numerical result.

Answer

We find on slide 29 from week 12 that:



Kernel algorithms

Example – Kernel ridge regression

Kernel Ridge regression (without bias)

Assume the regression task (η_n is white noise)

$$y_n = g(\boldsymbol{x}_n) + \eta_n, \quad n = 1, 2, \cdots, N$$

Assume the solution (according to representer theorem)

$$f(\cdot) = \sum_{m=1}^{N} \theta_m \kappa(\cdot, \boldsymbol{x}_m)$$

The model, where $C \in \mathbb{R}$ is the regularization parameter, is then:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$J(\boldsymbol{\theta}) := \sum_{n=1}^{N} \left(y_n - \sum_{m=1}^{N} \theta_m \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) \right)^2 + C\langle f, f \rangle$$

$$\hat{\boldsymbol{\theta}} = (\mathcal{K} + CI)^{-1} \boldsymbol{y}$$

Hence:

$$\hat{\theta} = (\mathcal{K} + CI)^{-1} y$$

where $\mathcal{K}$ is given by (11.11), so:

$$\mathcal{K} = \begin{bmatrix} \kappa(x_1, x_1) & \dots & \kappa(x_1, x_N) \\ \vdots & \ddots & \vdots \\ \kappa(x_N, x_1) & \dots & \kappa(x_N, x_N) \end{bmatrix}$$

In [137…

```python
sigma = 1.5
C = 0.02
x =  np.array([1, 2, 3]) # x-coordinate values
y = np.array([0.5, 0.8, 0.7]) # y-coordinate values
K = np.zeros((len(x),len(x))) # placeholder

np.set_printoptions(precision=3)
for i in range(K.shape[0]):
    for j in range(K.shape[1]):
        K[i,j] = gaussian_kernel(x[i], x[j], sigma)
KCI = K + C * np.eye(K.shape[0])
theta = np.linalg.inv(KCI) @ y
print(f"K + CI: \n{KCI}")
print(f"theta:\n{theta}")
## First coordinate is slightly diff from solution? Typo?
```

```
K + CI:
[[1.02  0.801 0.411]
 [0.801 1.02  0.801]
 [0.411 0.801 1.02 ]]
theta:
[-0.325  1.037  0.003]
```

21 6.3

Consider again kernel Ridge regression with three observations. Assume now that a weight vector of $\theta_1 = 1$, $\theta_2 = 3$ was computed. You can assume that $\kappa(\boldsymbol{x}_{\text{test}}, \boldsymbol{x}_1) = 0.3$, $\kappa(\boldsymbol{x}_{\text{test}}, \boldsymbol{x}_2) = 0$, and $\kappa(\boldsymbol{x}_{\text{test}}, \boldsymbol{x}_3) = 0.6$.

We are also informed that the regression value at $\boldsymbol{x}_{\text{test}} = 3$ yielded a value of 3, i.e. $\hat{y}(\boldsymbol{x}_{\text{test}}) = 3$.

Determine the value of θ_3 used in the regression problem.

Answer

According to eq. (11.27) the prediction of the kernel ride regression can be found as (although $y = f$ in slides):

$$\hat{y}(x) = \sum_{n=1}^{N} \theta_n \kappa(\cdot, x_n) = \theta_1 \kappa(x, x_1) + \theta_2 \kappa(x, x_2) + \theta_3 \kappa(x, x_3)$$

$$\Leftrightarrow$$

$$\theta_3 = \frac{\hat{f(x)} - \theta_1 \kappa(x, x_1) + \theta_2 \kappa(x, x_2)}{\kappa(x, x_3)}$$

Knowing $\theta_1 = 1, \theta_2 = 2, \kappa(x, x_1) = 0.3, \kappa(x, x_2) = 0, \kappa(x, x_3) = 0.6, \hat{y}(x) = 3$:

$$\theta_3 = \frac{3 - 1 \cdot 0.3 + 3 \cdot 0}{0.6} = 4.5$$

Exam 2022

22 Problem 1

22 1.1

Consider linear regression using squared error. You have the following observations:

n	0	1	2
y_n	-0.4	0.3	1.8

To model the response y_n, we will use a linear model on the form $f_n = \theta_0 + \theta_1 \cdot n$. Determine the parameters $\hat{\boldsymbol{\theta}} = [\theta_0 \ \theta_1]^T$ of the model that minimizes the error. Write up the relevant matrices and vectors used to calculate the estimate of $\hat{\boldsymbol{\theta}}$.

Answer

The parameters of linear regression with squared error loss can be readily found as:

$$\hat{\theta} = (X^T X)^{-1} X^T y$$

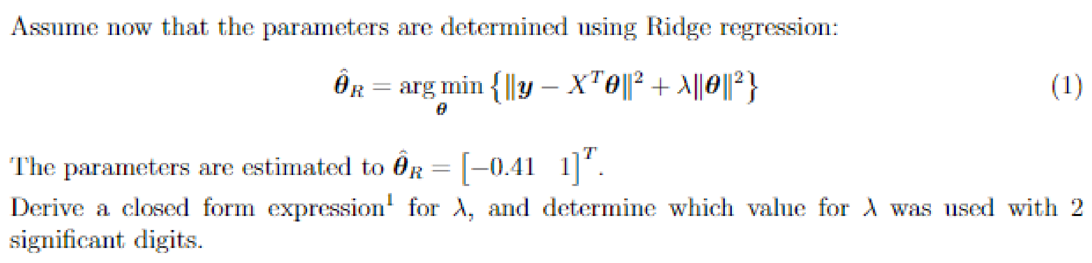
where we have $X = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$, where the columns correspond to the bias b of ones and the observations or predictors n. The vector $y_n = [-0.4 \;\; 0.3 \;\; 1.8]^T$ is readily seen from the data.

```
In [138…
import numpy as np
np.set_printoptions(precision=2)
X = np.array([[1,1,1],[0,1,2]]).T
y = np.array([-0.4, 0.3, 1.8])
theta = np.linalg.inv(X.T @ X) @ X.T @ y
print(f"theta = {theta}")
```

```
theta = [-0.53  1.1 ]
```

22 1.2

Assume now the parameters are determined with Ridge Regression:



Assume now that the parameters are determined using Ridge regression:

$$\hat{\boldsymbol{\theta}}_R = \arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - X^T\boldsymbol{\theta}\|^2 + \lambda\|\boldsymbol{\theta}\|^2 \right\} \tag{1}$$

The parameters are estimated to $\hat{\boldsymbol{\theta}}_R = \begin{bmatrix} -0.41 & 1 \end{bmatrix}^T$.
Derive a closed form expression[1] for λ, and determine which value for λ was used with 2 significant digits.

Derive a closed from expression for λ and determine its value. Hint: $\lambda \cdot a = b \Rightarrow \lambda = \frac{a^T b}{a^T a}$

Answer

We start off by finding the solution for θ_R in the ML book (6.29) or slide 34 week 6:

$$\begin{aligned} \theta_R &= (X^T X + \lambda I)^{-1} X^T y \\ &\Leftrightarrow (X^T X + \lambda I)\theta_R = X^T y = X^T X \theta_R + \lambda\theta_R = X^T y \\ &\Leftrightarrow \lambda\theta_R = X^T y - X^T X\theta_R \Leftrightarrow \lambda = \frac{\theta_R^T (X^T y - X^T X \theta_R)}{\theta_R^T \theta_R} \end{aligned}$$

Inserting and computing gives:

```
In [139…
theta_R = np.array([-0.41, 1])
l = (theta_R.T @ (X.T@y - X.T@X@theta_R)) / (theta_R.T@theta_R)
print(f"lambda = {np.round(l, 2)}")
```

```
lambda = 0.14
```

22 1.3

We now consider a different situation where we still assume a linear model and Ridge regression, but we assume that $X^TX = 2I$. Also assume that the least squares estimate (also under the assumption that $X^TX = 2I$) is $\hat{\boldsymbol{\theta}}_{LS} = \begin{bmatrix}-.75 & 1.3\end{bmatrix}^T$. Additionally, assume $\lambda = 0.5$.

Derive an expression for $\hat{\boldsymbol{\theta}}_R$ under these circumstances, and determine the value of $\hat{\boldsymbol{\theta}}_R$.

Answer

This is almost equivalent to the exercise we did in week 6 for θ_{LS}, albeit $X^TX = 2I$, so by direct substitution we can derive that, then use the formula which relates the LS estimate to the RR estimate given on slide 35-ish:

$$\theta_R = \frac{1}{1+\lambda}\theta_{LS}$$

For the LS estimate:

$$\theta_{LS} = (X^TX)^{-1}X^Ty = (2I)^{-1}X^Ty = \frac{1}{2}X^Ty \qquad \Leftrightarrow X^Ty = 2\cdot\theta_{LS}$$

For the Ridge Regression:

$$\begin{aligned}\theta_R &= (X^TX+\lambda I)^{-1}X^Ty = (2I+\lambda I)^{-1}X^Ty \\ &= (I(2+\lambda))^{-1}X^Ty = \frac{1}{2+\lambda}X^Ty = \frac{1}{2+\lambda}2\cdot\theta_{LS} \\ &= \frac{2}{2+\lambda}\theta_{LS}\end{aligned}$$

Inserting:

```
In [140…
theta_LS = np.array([-0.75, 1.3])
l = 0.5
theta_R = (2 /  (2 + l)) * theta_LS
print(f"theta_R = {theta_R}")
```

```
theta_R = [-0.6   1.04]
```

22 1.4

Apply l_1 regularization and disregarding the assumption $X^TX = 2I$ and instead use IST, we run one iteration of the algorithm. Assume $\theta^{(i-1)} = \theta_{LS}$ from 1.3 and $X^Te^{(i-1)} = [0.25 \quad -0.35]^T$

$$\boldsymbol{\theta}^{(i)} = S_{\lambda\mu}\left(\boldsymbol{\theta}^{(i-1)} + \mu X^T\boldsymbol{e}^{(i-1)}\right) \quad : \operatorname{sign}(\tilde{\theta})\max\left(|\tilde{\theta}| - \lambda\mu, 0\right) = S_{\lambda\mu}$$

Determine lowest value of μ that results in exactly **one** component of $\theta^{(i)}$ being zero.

Answer

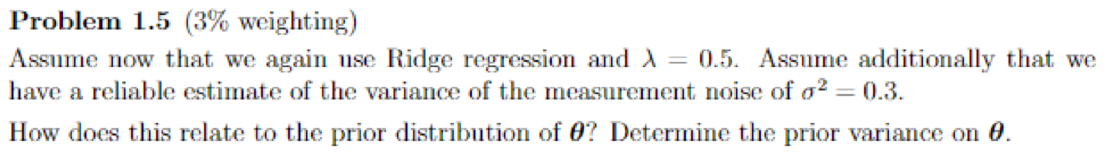
$$\theta^{(i)} = S_{\lambda\mu}\left(\begin{bmatrix}-0.75\\1.3\end{bmatrix} + \mu\begin{bmatrix}0.25\\-0.35\end{bmatrix}\right)$$
$$= \text{sign}\left(\begin{bmatrix}-0.75\\1.3\end{bmatrix} + \mu\begin{bmatrix}0.25\\-0.35\end{bmatrix}\right)\max\left(|\begin{bmatrix}-0.75\\1.3\end{bmatrix} + \mu\begin{bmatrix}0.25\\-0.35\end{bmatrix}| - \lambda\mu, 0\right)$$

If simply one component has to be zero we can isolate the difference in the max function and set it to zero, so we can rearrange it as $|\theta| - \lambda\mu = 0 \Leftrightarrow |\theta| = \lambda\mu$

$$|-0.75 + \mu 0.25| = \lambda\mu = 0.5\mu \Leftrightarrow \mu = 1 \quad \text{(MAPLE or SYMBOLAB)}$$
$$|1.3 - 0.35\mu| = 0.5\mu \Leftrightarrow \mu = \frac{26}{17}$$

Hence the lowest value of μ for one component to be zero is $\mu = 1$

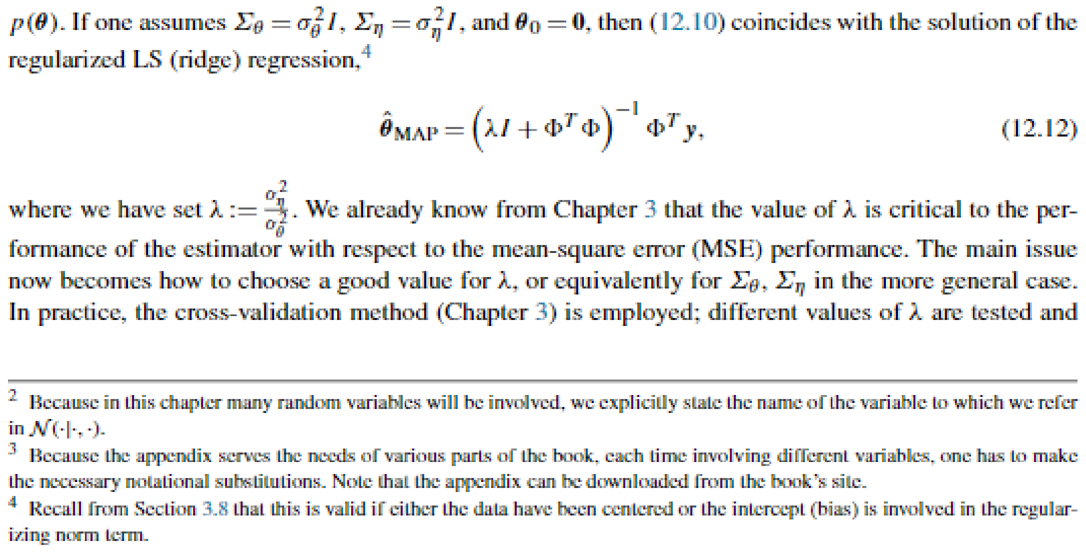
22 1.5



Problem 1.5 (3% weighting)
Assume now that we again use Ridge regression and $\lambda = 0.5$. Assume additionally that we have a reliable estimate of the variance of the measurement noise of $\sigma^2 = 0.3$.
How does this relate to the prior distribution of $\boldsymbol{\theta}$? Determine the prior variance on $\boldsymbol{\theta}$.

Answer

Chapter 12.2.2 in the book eq. (12.12) relates regularized LS (ridge) regression to the MAP estimator as:



$p(\boldsymbol{\theta})$. If one assumes $\Sigma_\theta = \sigma_\theta^2 I$, $\Sigma_\eta = \sigma_\eta^2 I$, and $\boldsymbol{\theta}_0 = \mathbf{0}$, then (12.10) coincides with the solution of the regularized LS (ridge) regression,[4]

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \left(\lambda I + \Phi^T\Phi\right)^{-1}\Phi^T \mathbf{y}, \tag{12.12}$$

where we have set $\lambda := \frac{\sigma_\eta^2}{\sigma_\theta^2}$. We already know from Chapter 3 that the value of λ is critical to the performance of the estimator with respect to the mean-square error (MSE) performance. The main issue now becomes how to choose a good value for λ, or equivalently for Σ_θ, Σ_η in the more general case. In practice, the cross-validation method (Chapter 3) is employed; different values of λ are tested and

[2] Because in this chapter many random variables will be involved, we explicitly state the name of the variable to which we refer in $\mathcal{N}(\cdot|\cdot,\cdot)$.
[3] Because the appendix serves the needs of various parts of the book, each time involving different variables, one has to make the necessary notational substitutions. Note that the appendix can be downloaded from the book's site.
[4] Recall from Section 3.8 that this is valid if either the data have been centered or the intercept (bias) is involved in the regularizing norm term.

where $\phi = X$ will coincide in the ridge solution. The prior distribution of θ is assumed Gaussian in eq. (12.8) preceding the above:

$$p(\theta) = \mathcal{N}(\theta|\theta_0 = 0, \Sigma_\theta = \sigma_\theta^2 I)$$

In other words, there is a direct relation between λ, measurement noise σ_η^2 and prior variance σ_θ^2:

$$\lambda = \frac{\sigma_\eta^2}{\sigma_\theta^2} \Leftrightarrow \sigma_\theta^2 = \frac{\sigma_\eta^2}{\lambda} = \frac{0.3}{0.5} = 0.6$$

22 Problem 2

Linear FIR filtering with 3 coefficients and $l = 3$.

In this problem we are considering the Linear filtering situation as depicted in Figure 1, page 3, where we have a FIR filter. We will use a filter with 3 coefficients ($l = 3$).

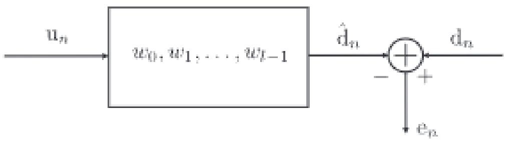


Figure 1: Problem 2 filter setup.

Assume that we have an input sequence to the filter as

n	0	1	2	3	4	5
u_n	3	2	1	5	4	5

The filter has the coefficients $\boldsymbol{w} = [0.6 \quad 1.5 \quad -1]^T$.

22 2.1

Determine the output of the filter at $n = 3$

Answer

We are asked to find $\hat{d}_{n=3}$. We use eq. (4.41) which states:

$$\boxed{\hat{d}_n = \sum_{i=0}^{l-1} w_i u_{n-i} = \boldsymbol{w}^T \mathbf{u}_n : \quad \text{convolution sum,}} \qquad (4.41)$$

$$\hat{d}_{n=3} = \sum_{i=0}^{3-1} w_i u_{3-i} = w_0 u_3 + w_1 u_2 + w_2 u_1 =$$

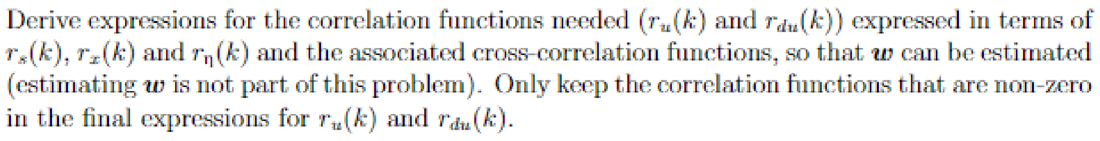
In [141…

```
l = 3
n = 3
w = np.array([0.6, 1.5, -1])
u_n = np.array([3,2,1,5,4,5])
d_hat_n = w.T @ np.array([u_n[n], u_n[n-1], u_n[n-2]])
print(f"d_hat_3 = {d_hat_n.item()}")
```

```
d_hat_3 = 2.5
```

22 2.2

Learning-based setup to recover s_n. $u_n = s_n + x_n$, where $x_n, s_n \sim AR(1)$. Additionally $v_n = s_n + \eta_n$ where follows a white-noise sequence ie. $\eta_n \sim \mathcal{N}(0, \sigma^2)$



Derive expressions for the correlation functions needed ($r_u(k)$ and $r_{du}(k)$) expressed in terms of $r_s(k)$, $r_x(k)$ and $r_\eta(k)$ and the associated cross-correlation functions, so that $\boldsymbol{w}$ can be estimated (estimating $\boldsymbol{w}$ is not part of this problem). Only keep the correlation functions that are non-zero in the final expressions for $r_u(k)$ and $r_{du}(k)$.

Answer

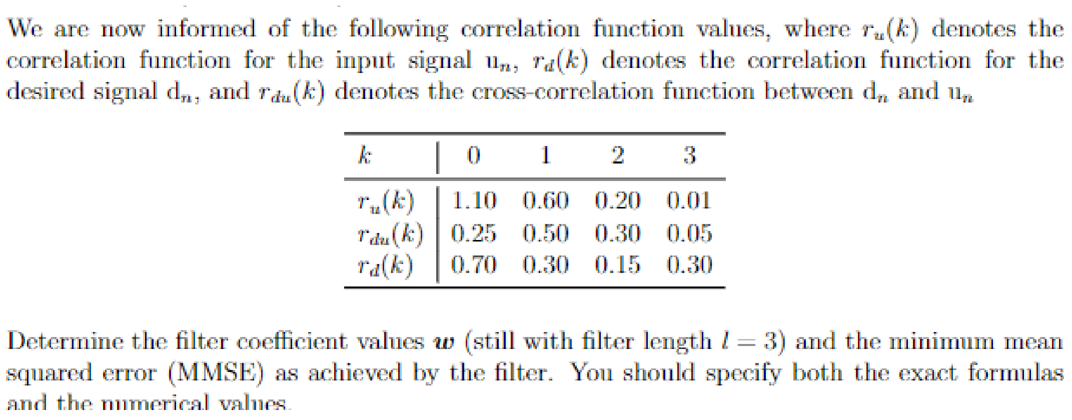
$$\begin{aligned}
r_u(k) &= \mathbb{E}[u_n u_{n-k}] = \mathbb{E}[(s_n + x_n)(s_{n-k} + x_{n-k})] \\
&= \mathbb{E}[s_n s_{n-k} + x_n x_{n-k} + s_n x_{n-k} + x_n s_{n-k}] \\
&= \mathbb{E}[s_n s_{n-k}] + \mathbb{E}[x_n x_{n-k}] + \mathbb{E}[s_n x_{n-k}] + \mathbb{E}[x_n s_{n-k}] \\
&= r_s(k) + r_x(k) + r_{sx}(k) + r_{xs}(k) \\
&= r_s(k) + r_x(k)
\end{aligned}$$

Since AR processes are generated with white noise they are assumed uncorrelated, ie. $r_{xs} = r_{sx} = 0$. Now, here the problem is a bit weirdly put, but $d_n = v_n = s_n + \eta_n$ is what we should use...

$$\begin{aligned}
r_{du}(k) &= \mathbb{E}[d_n u_{n-k}] = \mathbb{E}[(s_n + \eta_n)(s_{n-k} + x_{n-k})] \\
&= \mathbb{E}[s_n s_{n-k} + s_n x_{n-k} + \eta_n s_{n-k} + \eta_n x_{n-k}] \\
&= r_s(k)
\end{aligned}$$

Here, since η_n is a white-noise sequence (zero mean $\mathbb{E}[\eta_n] = 0$) and s_n and x_n are $AR(1)$-processes, they all vanish, respectively, leaving us with $r_s(k)$

22 2.3



We are now informed of the following correlation function values, where $r_u(k)$ denotes the correlation function for the input signal u_n, $r_d(k)$ denotes the correlation function for the desired signal d_n, and $r_{du}(k)$ denotes the cross-correlation function between d_n and u_n

k	0	1	2	3
$r_u(k)$	1.10	0.60	0.20	0.01
$r_{du}(k)$	0.25	0.50	0.30	0.05
$r_d(k)$	0.70	0.30	0.15	0.30

Determine the filter coefficient values $\boldsymbol{w}$ (still with filter length $l = 3$) and the minimum mean squared error (MMSE) as achieved by the filter. You should specify both the exact formulas and the numerical values.

Answer

To determine the filter coefficient (with $l = 3$) and the MMSE we need to consult the book, particularly p.136 where:

$$R_u = \Sigma_u$$
$$\Sigma_u w = p \Leftrightarrow w = \Sigma_u^{-1} p = R_u^{-1} p$$

The MMSE can be found in ch. 4.2.1 p. 123 eq. (4.9), where we simply choose the one applicable to our case. In this case θ is not known, hence we have to use the first option listed here:

$$J(\theta) = \sigma_y^2 - p^T \Sigma_x^{-1} p = \sigma_y^2 \theta_*^T \Sigma_x \theta_* = \sigma_y^2 - p^T \theta_*$$

where $\Sigma_x = \Sigma_u$, as u is our input and $y = d$ is our output, where we recall that the first option $r_d(0) = \sigma_d^2$ of its cross-correlation matrix Σ_d.

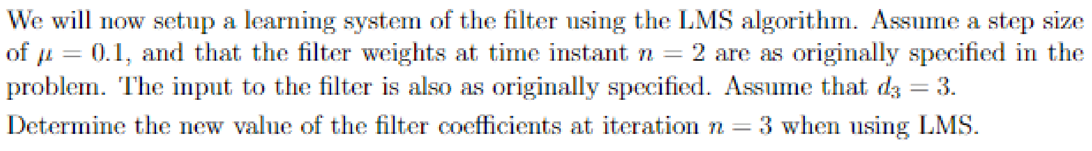
In [142…

```python
#### First part
r_u0 = 1.10; r_u1 = 0.6; r_u2 = 0.20; r_u3 = 0.01
# recall it is populated by r(0) -> r(l-1) square matrix, symmetric
R_u = np.array([[r_u0, r_u1, r_u2],
                [r_u1, r_u0, r_u1],
                [r_u2, r_u1, r_u0]])
r_du0 = 0.25; r_du1 = 0.5; r_du2 = 0.3; r_du3 = 0.05
p = np.array([[r_du0], [r_du1], [r_du2]]) # p is a column vector
w = np.linalg.inv(R_u) @ p
print(f"w = \n{w}\n")

#### Second part
r_d0 = 0.70
J = r_d0 - p.T @ np.linalg.inv(R_u) @ p
print(f"MMSE = {J.item()}")
```

```
w = 
[[-0.02]
 [ 0.45]
 [ 0.03]]

MMSE = 0.47132237871674487
```

22 2.4

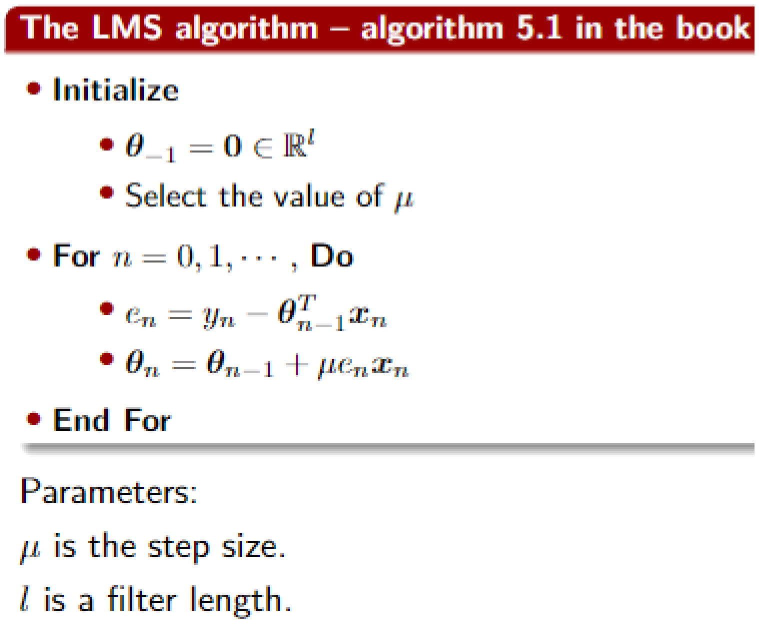
Learning system of filter with LMS. Stepsize, filter weights.


We will now setup a learning system of the filter using the LMS algorithm. Assume a step size of $\mu = 0.1$, and that the filter weights at time instant $n = 2$ are as originally specified in the problem. The input to the filter is also as originally specified. Assume that $d_3 = 3$.
Determine the new value of the filter coefficients at iteration $n = 3$ when using LMS.

Answer

If we assume $n = 2$ filter weights, then we have the same u, w and n as given in the original problem statement.

LMS is given by the algorithm:



The LMS algorithm – algorithm 5.1 in the book

```
• Initialize
    • $\theta_{-1} = 0 \in \mathbb{R}^l$
    • Select the value of $\mu$
• For $n = 0, 1, \cdots$, Do
    • $e_n = y_n - \theta_{n-1}^T x_n$
    • $\theta_n = \theta_{n-1} + \mu e_n x_n$
• End For
```

Parameters:

μ is the step size.

l is a filter length.

so for $n = 3$, which means $x_n = u_3 = [u_3, u_{3-1}, u_{3-2}] = [5 \;\; 1 \;\; 2]^T$ (see (4.41)), $\mu = 0.1$ and $y_n = d_3 = 3$.

$$e_3 = y_3 - \theta_{3-1}^T x_3 = d_3 - w_{3-1}^T [5 \;\; 1 \;\; 2]^T$$
$$\theta_3 = \theta_{3-1} + \mu e_3 x_3 = w_{3-1}^T + \mu e_3 u_3$$

In [143…

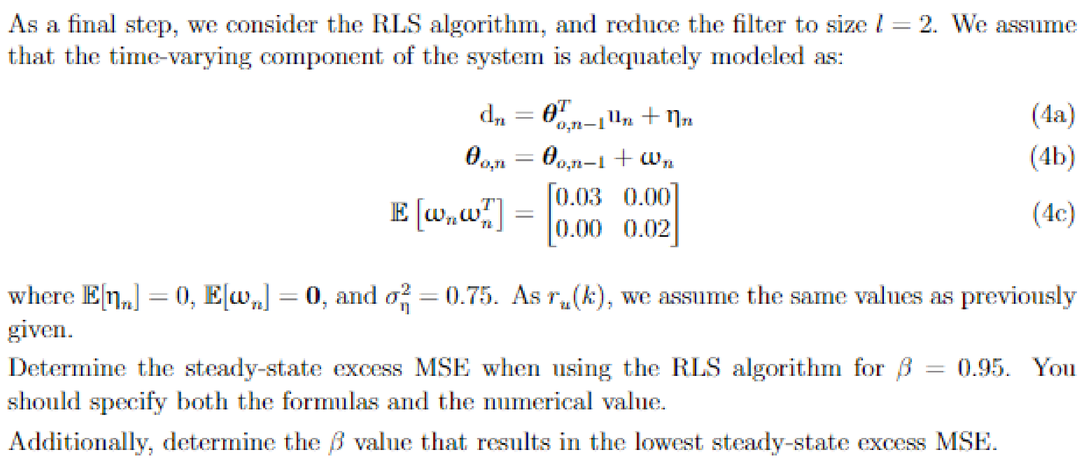
```python
w = np.array([0.6, 1.5, -1]) # column vector
u_3 = np.array([5,1,2]) # column vector
d_3 = 3
mu = 0.1

e_3 = d_3 - w.T @ u_3
theta_3 = w + mu * e_3 * u_3
print(f"e_3 = {e_3}")
print(f"theta_3 = {theta_3}")
```

```
e_3 = 0.5
theta_3 = [ 0.85  1.55 -0.9 ]
```

22 2.5

RLS with $l = 2$. Time varying component is adequately modeled. Determine steady-state excess MSE using RLS for $\beta = 0.95$. Also find β that gives the lowest steady-state excess

MSE



As a final step, we consider the RLS algorithm, and reduce the filter to size $l = 2$. We assume that the time-varying component of the system is adequately modeled as:

$$\mathrm{d}_n = \boldsymbol{\theta}_{o,n-1}^T \mathrm{u}_n + \eta_n \tag{4a}$$

$$\boldsymbol{\theta}_{o,n} = \boldsymbol{\theta}_{o,n-1} + \boldsymbol{\omega}_n \tag{4b}$$

$$\mathbb{E}\left[\boldsymbol{\omega}_n \boldsymbol{\omega}_n^T\right] = \begin{bmatrix} 0.03 & 0.00 \\ 0.00 & 0.02 \end{bmatrix} \tag{4c}$$

where $\mathbb{E}[\eta_n] = 0$, $\mathbb{E}[\boldsymbol{\omega}_n] = \mathbf{0}$, and $\sigma_\eta^2 = 0.75$. As $r_u(k)$, we assume the same values as previously given.

Determine the steady-state excess MSE when using the RLS algorithm for $\beta = 0.95$. You should specify both the formulas and the numerical value.

Additionally, determine the β value that results in the lowest steady-state excess MSE.

Answer

This looks oddly like ch. 6.8 "Steady-state performance of the RLS". Table 6.1 on p 276 describes the steady-state excess MSE for LMS, RLS and APA (and NLMS).

$$J_{exc}^{RLS} \simeq \frac{1}{2}(1-\beta)\sigma_\eta^2 l + \frac{1}{2}(1-\beta)^{-1}\text{trace}\{\Sigma_\omega \Sigma_x\}$$

Where the input x is now u, hence we need to use Σ_u.

The optimal β can be found using equation on page 277:

$$\beta_{opt} = 1 - \sqrt{\frac{\text{trace}\{\Sigma_\omega \Sigma_x\}}{\sigma_\eta^2 \cdot l}}$$

In [144…

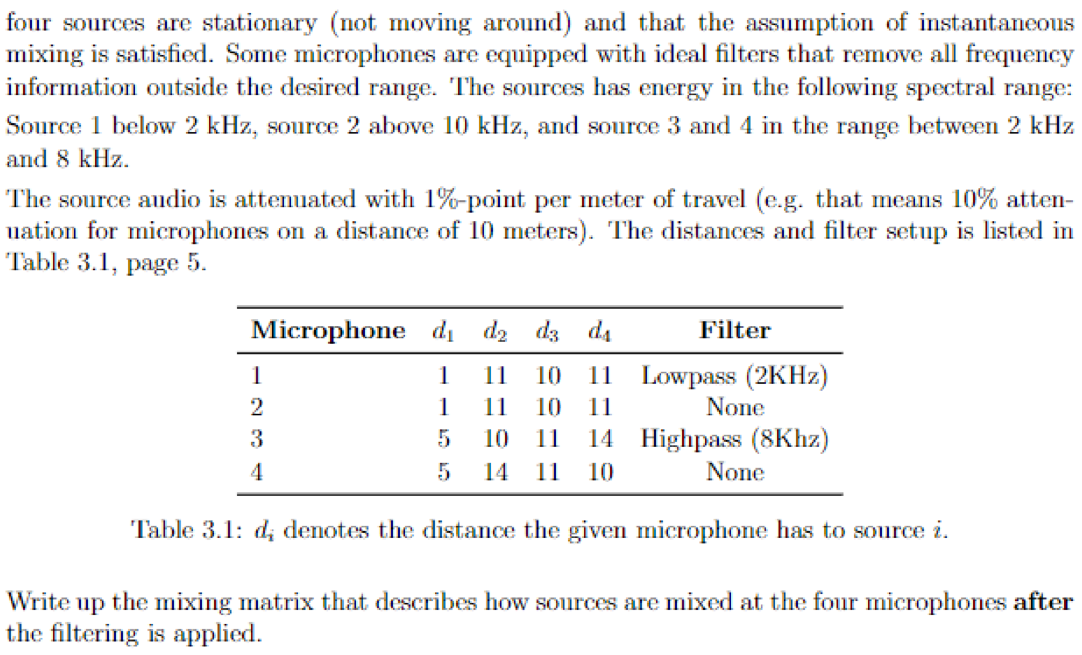
```
beta = 0.95
l = 2
Sigma_omega = np.array([[0.03, 0.00],
                        [0.00, 0.02]])
Sigma_u = np.array([[r_u0, r_u1],
                    [r_u1, r_u0]]) # Since l = 2, (r(0), r(l-1))
print(f"Sigma_u = \n{Sigma_u}")
sigma2_eta = 0.75

J_rls_excess = 1/2 * (1-beta) * sigma2_eta * l + 1/2 * (1-beta)**(-1) * np.trace
print(f"J_RLS_excess = {round(J_rls_excess,2)}")

beta_optimal = 1 - np.sqrt(  np.trace(Sigma_omega@Sigma_u) / (sigma2_eta * l)  )
print(f"Optimal Beta = {round(beta_optimal,2)}")

# Double checking this indeed gives a lower excess MSE:
J_opt =  1/2 * (1-beta_optimal) * sigma2_eta * l + 1/2 * (1-beta_optimal)**(-1)
print(f"\nChecking if new J_exc is lower with new beta:\nJ_opt = {round(J_opt,2)
Sigma_u =
[[1.1 0.6]
 [0.6 1.1]]
J_RLS_excess = 0.59
Optimal Beta = 0.81

Checking if new J_exc is lower with new beta:
J_opt = 0.29
```

22 Problem 3

Concerns ICA using Mutual Information

22 3.1

Suppose we are in a room with 4 microphones and 4 sources, which are assumed stationary. The assumption of instantaneous mixing is satisfied.



four sources are stationary (not moving around) and that the assumption of instantaneous mixing is satisfied. Some microphones are equipped with ideal filters that remove all frequency information outside the desired range. The sources has energy in the following spectral range: Source 1 below 2 kHz, source 2 above 10 kHz, and source 3 and 4 in the range between 2 kHz and 8 kHz.

The source audio is attenuated with 1%-point per meter of travel (e.g. that means 10% attenuation for microphones on a distance of 10 meters). The distances and filter setup is listed in Table 3.1, page 5.

Microphone	d_1	d_2	d_3	d_4	Filter
1	1	11	10	11	Lowpass (2KHz)
2	1	11	10	11	None
3	5	10	11	14	Highpass (8Khz)
4	5	14	11	10	None

Table 3.1: d_i denotes the distance the given microphone has to source i.

Write up the mixing matrix that describes how sources are mixed at the four microphones **after** the filtering is applied.

Answer

The mixing matrix A can be found by careful inspection of the above picture and information. Note, it is *after* the filtering is applied. So, first we inspect every microphone and what source d_n for $n \in \{1,2,3,4\}$ is "recorded" by the microphone after passing through the filter. We note the following:

$$d_1 < 2kHz$$
$$d_2 > 10kHz$$
$$2kHz < d_3 \text{ and } d_4 < 8kHz$$

Microphone 1:

- d_1 passes through filter, and is attenuated by 1% due to a distance of 1, so $A_{1,1} = 1 - 0.01 = 0.99$

- d_2, d_3 and d_4 all do not pass through, despite the distance so $A_{1,234} = 0$

Microphone 2:

- No filter, so we just need to attenuate. $A_{2,1} = 0.99$, $A_{2,2} = 1 - 0.11 = 0.89$, $A_{2,3} = 1 - 0.10 = 0.90$ and $A_{2,4} = 1 - 0.11 = 0.89$

Microphone 3:

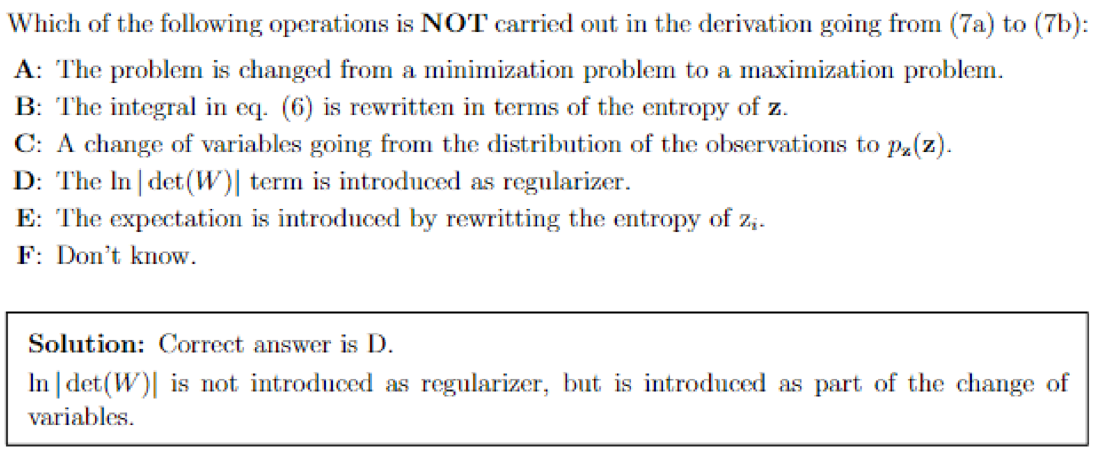
- Only source 2 is large enough to pass through, so $A_{3,2} = 0.90$ and $A_{3,134} = 0$

Microphone 4:

- No filter, so we just need to attenuate. $A_{4,1} = 0.95$, $A_{4,2} = 0.86$, $A_{4,3} = 0.89$ and $A_{4,4} = 0.90$

$$A = \begin{bmatrix} 0.99 & 0 & 0 & 0 \\ 0.99 & 0.89 & 0.90 & 0.89 \\ 0 & 0.90 & 0 & 0 \\ 0.95 & 0.86 & 0.89 & 0.90 \end{bmatrix}$$

22 3.2



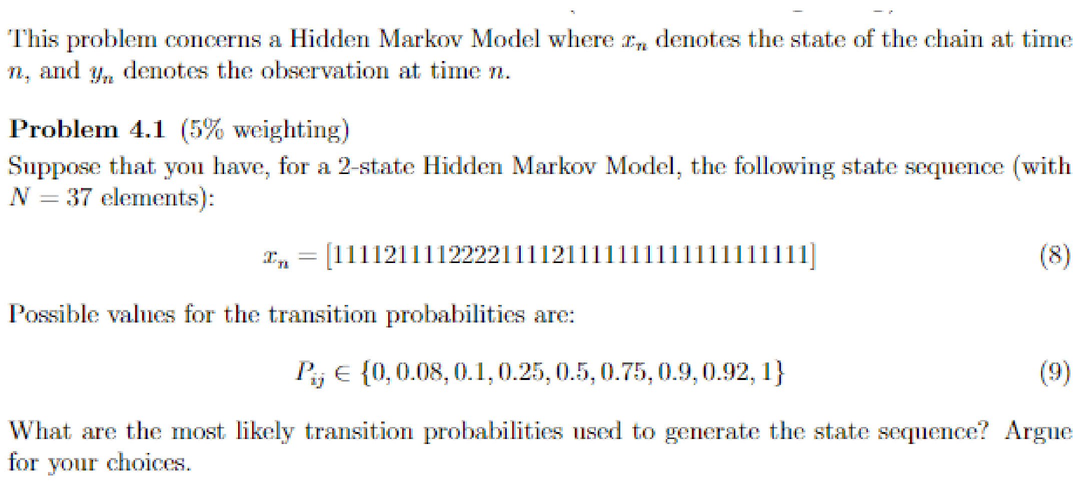
Which of the following operations is **NOT** carried out in the derivation going from (7a) to (7b):

A: The problem is changed from a minimization problem to a maximization problem.
B: The integral in eq. (6) is rewritten in terms of the entropy of $\mathbf{z}$.
C: A change of variables going from the distribution of the observations to $p_{\mathbf{z}}(\mathbf{z})$.
D: The $\ln|\det(W)|$ term is introduced as regularizer.
E: The expectation is introduced by rewritting the entropy of z_i.
F: Don't know.

Solution: Correct answer is D.
$\ln|\det(W)|$ is not introduced as regularizer, but is introduced as part of the change of variables.

22 Problem 4

22 4.1



This problem concerns a Hidden Markov Model where x_n denotes the state of the chain at time n, and y_n denotes the observation at time n.

Problem 4.1 (5% weighting)
Suppose that you have, for a 2-state Hidden Markov Model, the following state sequence (with $N = 37$ elements):

$$x_n = [1111211112222111121111111111111111111] \tag{8}$$

Possible values for the transition probabilities are:

$$P_{ij} \in \{0, 0.08, 0.1, 0.25, 0.5, 0.75, 0.9, 0.92, 1\} \tag{9}$$

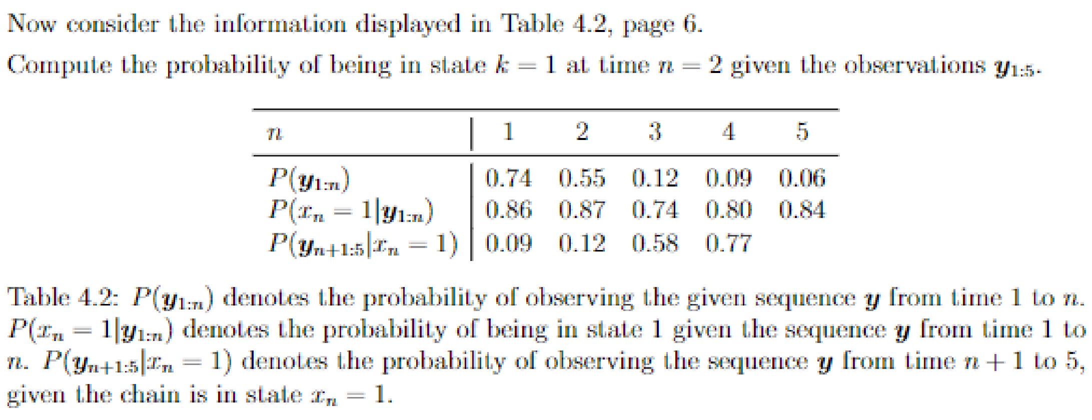
What are the most likely transition probabilities used to generate the state sequence? Argue for your choices.

Answer

The best estimate we can have is the frequentist point of view, ie. counting the state transitions in chains of each state 1 and 2, respectively. This means:

$$\begin{aligned}
1 \to 1: \;\; 27 \mid P_{11} &= \frac{27}{27+3} = 0.9 \\
1 \to 2: \;\; 3 \mid P_{12} &= \frac{3}{27+3} = 0.1 \\
2 \to 1: \;\; 3 \mid P_{21} &= \frac{3}{3+3} = 0.5 \\
2 \to 2: \;\; 3 \mid P_{22} &= \frac{3}{3+3} = 0.5
\end{aligned}$$

22 4.2



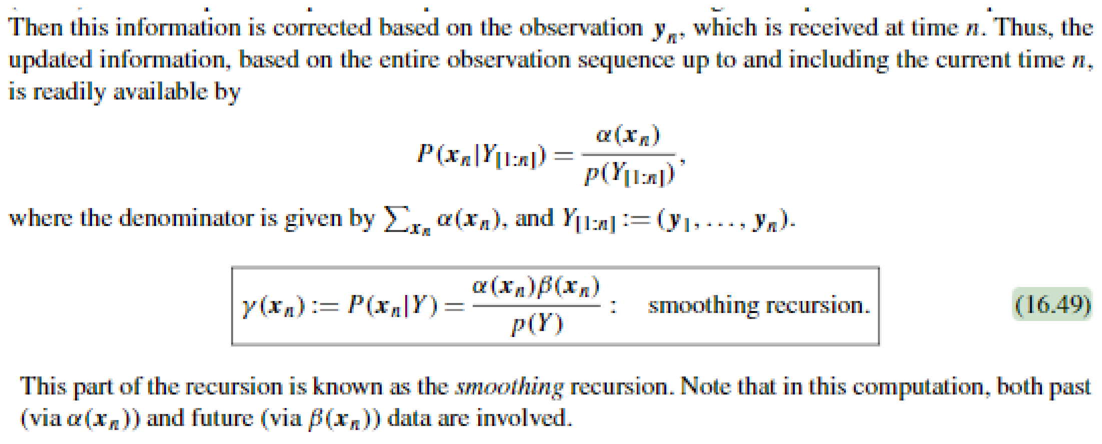
Now consider the information displayed in Table 4.2, page 6.
Compute the probability of being in state $k = 1$ at time $n = 2$ given the observations $\boldsymbol{y}_{1:5}$.

n	1	2	3	4	5
$P(\boldsymbol{y}_{1:n})$	0.74	0.55	0.12	0.09	0.06
$P(x_n = 1 \mid \boldsymbol{y}_{1:n})$	0.86	0.87	0.74	0.80	0.84
$P(\boldsymbol{y}_{n+1:5} \mid x_n = 1)$	0.09	0.12	0.58	0.77	

Table 4.2: $P(\boldsymbol{y}_{1:n})$ denotes the probability of observing the given sequence $\boldsymbol{y}$ from time 1 to n. $P(x_n = 1 \mid \boldsymbol{y}_{1:n})$ denotes the probability of being in state 1 given the sequence $\boldsymbol{y}$ from time 1 to n. $P(\boldsymbol{y}_{n+1:5} \mid x_n = 1)$ denotes the probability of observing the sequence $\boldsymbol{y}$ from time $n+1$ to 5, given the chain is in state $x_n = 1$.

Answer

We are asked to compute $P(x_2 = 1 \mid y_{1:5})$. HMM on p. 850 and p. 851 states:



Then this information is corrected based on the observation $\boldsymbol{y}_n$, which is received at time n. Thus, the updated information, based on the entire observation sequence up to and including the current time n, is readily available by

$$P(\boldsymbol{x}_n|Y_{[1:n]}) = \frac{\alpha(\boldsymbol{x}_n)}{p(Y_{[1:n]})},$$

where the denominator is given by $\sum_{\boldsymbol{x}_n} \alpha(\boldsymbol{x}_n)$, and $Y_{[1:n]} := (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$.

$$\boxed{\gamma(\boldsymbol{x}_n) := P(\boldsymbol{x}_n|Y) = \frac{\alpha(\boldsymbol{x}_n)\beta(\boldsymbol{x}_n)}{p(Y)} : \quad \text{smoothing recursion.}} \tag{16.49}$$

This part of the recursion is known as the *smoothing* recursion. Note that in this computation, both past (via $\alpha(\boldsymbol{x}_n)$) and future (via $\beta(\boldsymbol{x}_n)$) data are involved.

In our problem we have future observations, so we should actually consider the second part with β! Here α is the past and β the future. So, concerning rewriting we use:

$$\alpha(x_n) = p(y_n|x_n) \sum_{x_{n-1}} \alpha(x_{n-1}) P(x_n|x_{n-1})$$
$$\beta(x_n) = p(y_{n+1}, y_{n+2}, \ldots, y_N|x_n) = p(y_{n+1:N}|x_n)$$

So:

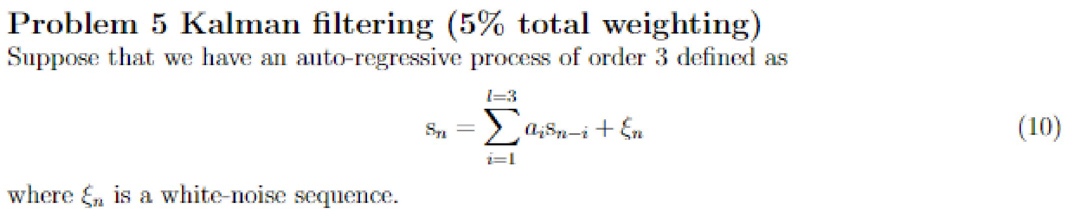
$$\begin{aligned} P(x_n|y_{1:5}) &= \frac{\alpha(x_n)\beta(x_n)}{P(y_{1:N})} \\ &= \frac{\alpha(x_n)P(y_{n+1:N}|x_n)}{P(y_{1:N})} \\ &= \frac{P(x_n|y_{1:n})P(y_{1:n})P(y_{n+1:N}|x_n)}{P(y_{1:N})} \end{aligned}$$

Inserting for $n = 2$ and $N = 5$

$$\begin{aligned} P(x_2 = 1|y_{1:5}) &= \frac{P(x_2 = 1|y_{1:2})P(y_{1:2})P(y_{3:5}|x_2 = 1)}{P(y_{1:5})} \\ &= \frac{0.87 \cdot 0.55 \cdot 0.12}{0.06} \\ &= 0.96 \end{aligned}$$

Or with Bayes', albeit a bit of the same idea, except one has to split up the probability and do some more rewriting that I find confusing, tbh.

22 Problem 5

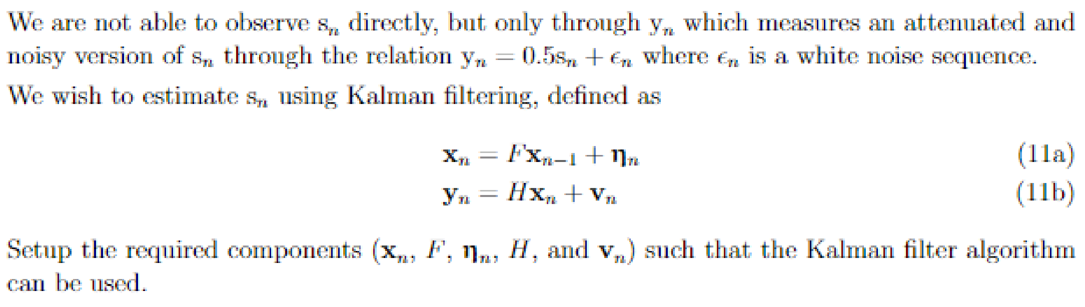


Problem 5 Kalman filtering (5% total weighting)
Suppose that we have an auto-regressive process of order 3 defined as

$$s_n = \sum_{i=1}^{l=3} a_i s_{n-i} + \xi_n \tag{10}$$

where ξ_n is a white-noise sequence.

22 5.1



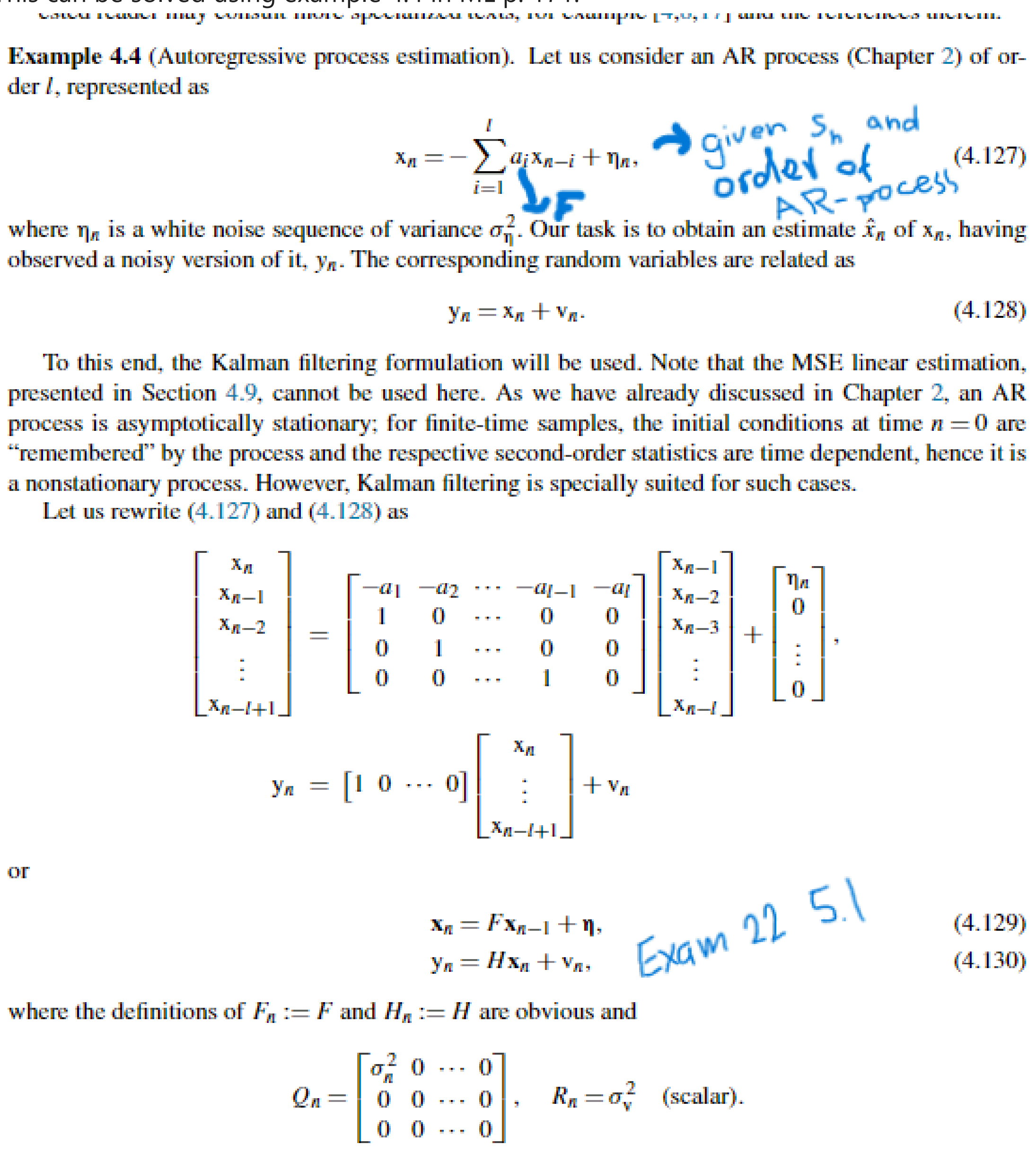
We are not able to observe s_n directly, but only through y_n which measures an attenuated and noisy version of s_n through the relation $y_n = 0.5s_n + \epsilon_n$ where ϵ_n is a white noise sequence.
We wish to estimate s_n using Kalman filtering, defined as

$$\mathbf{x}_n = F\mathbf{x}_{n-1} + \boldsymbol{\eta}_n \quad (11a)$$
$$\mathbf{y}_n = H\mathbf{x}_n + \mathbf{v}_n \quad (11b)$$

Setup the required components ($\mathbf{x}_n$, F, $\boldsymbol{\eta}_n$, H, and $\mathbf{v}_n$) such that the Kalman filter algorithm can be used.

Answer

This can be solved using example 4.4 in ML p. 171.



ested reader may consult more specialized texts, for example [4,8,17] and the references therein.

Example 4.4 (Autoregressive process estimation). Let us consider an AR process (Chapter 2) of order l, represented as

$$\mathrm{x}_n = -\underbrace{\sum_{i=1}^{l} a_i \mathrm{x}_{n-i}}_{\downarrow F} + \eta_n, \quad \rightarrow \text{given } S_n \text{ and order of AR-process} \quad (4.127)$$

where η_n is a white noise sequence of variance σ_η^2. Our task is to obtain an estimate $\hat{x}_n$ of x_n, having observed a noisy version of it, y_n. The corresponding random variables are related as

$$\mathrm{y}_n = \mathrm{x}_n + \mathrm{v}_n. \quad (4.128)$$

To this end, the Kalman filtering formulation will be used. Note that the MSE linear estimation, presented in Section 4.9, cannot be used here. As we have already discussed in Chapter 2, an AR process is asymptotically stationary; for finite-time samples, the initial conditions at time $n = 0$ are "remembered" by the process and the respective second-order statistics are time dependent, hence it is a nonstationary process. However, Kalman filtering is specially suited for such cases.

Let us rewrite (4.127) and (4.128) as

$$\begin{bmatrix} \mathrm{x}_n \\ \mathrm{x}_{n-1} \\ \mathrm{x}_{n-2} \\ \vdots \\ \mathrm{x}_{n-l+1} \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{l-1} & -a_l \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{x}_{n-1} \\ \mathrm{x}_{n-2} \\ \mathrm{x}_{n-3} \\ \vdots \\ \mathrm{x}_{n-l} \end{bmatrix} + \begin{bmatrix} \eta_n \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$\mathrm{y}_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \mathrm{x}_n \\ \vdots \\ \mathrm{x}_{n-l+1} \end{bmatrix} + \mathrm{v}_n$$

or

$$\mathbf{x}_n = F\mathbf{x}_{n-1} + \boldsymbol{\eta}, \quad (4.129)$$
$$\mathrm{y}_n = H\mathbf{x}_n + \mathrm{v}_n, \quad (4.130)$$

Exam 22 5.1

where the definitions of $F_n := F$ and $H_n := H$ are obvious and

$$Q_n = \begin{bmatrix} \sigma_n^2 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \quad R_n = \sigma_v^2 \quad \text{(scalar)}.$$

Here we have to recognize that x_n follows the form of (4.127) with slight alterations, one of them being the sign of the sum, and since ξ_n is a white-noise sequence we should recognize the following:

$$x_n = \begin{bmatrix} s_n \\ s_{n-1} \\ s_{n-2} \end{bmatrix}$$

$$F = \begin{bmatrix} a_1 & a_2 & a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\eta_n = \begin{bmatrix} \xi_n \\ 0 \\ 0 \end{bmatrix}, \quad H = [0.5 \quad 0 \quad 0], \quad v_n = \epsilon_n$$

22 Problem 6

Problem 6 Kernel methods (15% total weighting)
In this problem we will consider kernel methods, and we will use two kernel functions, the homogeneous polynomial kernel with $r = 1$ and the Laplacian kernel, denoted κ_p and κ_l respectively

$$\kappa_p(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^T \boldsymbol{y})^r \tag{12a}$$
$$\kappa_l(\boldsymbol{x}, \boldsymbol{y}) = \exp(-a\|\boldsymbol{x} - \boldsymbol{y}\|) \tag{12b}$$

22 6.1

Problem 6.1 (5% weighting)
Consider the classification problem with the points listed in Figure 2, page 8.

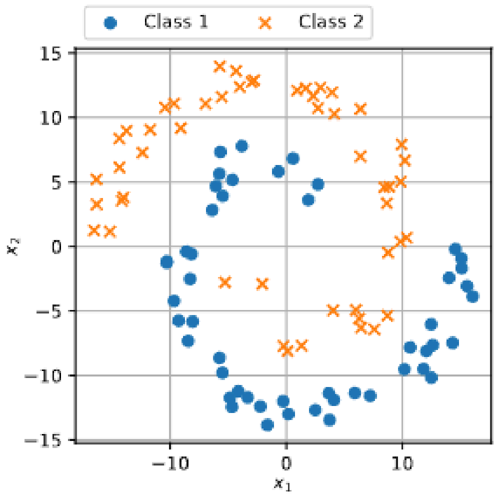


Figure 2: Problem 6.1.

Which of the two kernel functions $\kappa_p(\boldsymbol{x}, \boldsymbol{y})$ and $\kappa_l(\boldsymbol{x}, \boldsymbol{y})$ will be most appropriate?
Explain your decision for the kernel, and relate the solution to the Representer theorem (e.g. what could possible values for the θ_n in the Representer theorem be?)

Answer

- The Laplacian kernel is better as it measures the distancy between the points, whereas the polynomial kernel with $r = 1$ becomes the linear kernel, which is unfit for non-linearly separable data, whereas Laplacian is. Clearly, the data is non-linearly separable.

The representer theorem given in ML (11.17):

$$f(\cdot) = \sum_{n=1}^{N} \theta_n \kappa_l(\cdot, x_n),$$

where x_n are the training points, and θ_n are coefficients determined during training using a test point x.

- θ_n associated with class 1 could be 1, while class 2 could have a -1. Then the decision boundary could be around 0, such that $f(x_{test}) > 0$ is class 1 and $f(x_{test}) < 0$ is class 2.

These images are taken from the solution to aid understanding and would not have been possible to do since we do not have any data of the problem

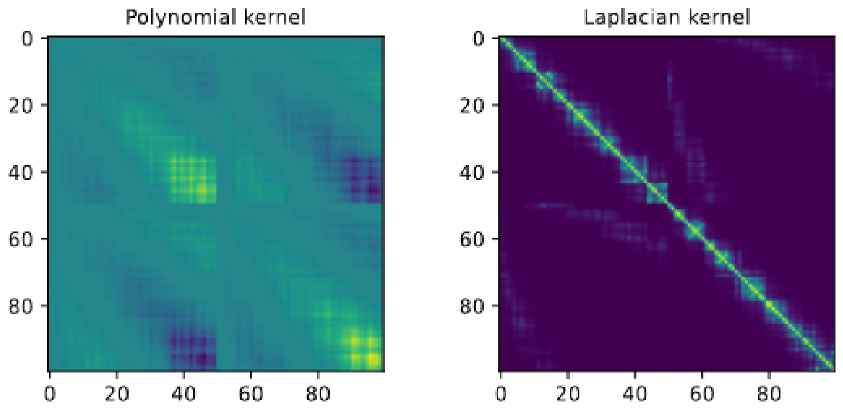


Figure 3: Problem 6.1 solution. The points are first sorted according to the class label, and there after sorted according to the angle relative to the x-axis. This makes the kernel matrix easy to interpret in terms of the block-structure.

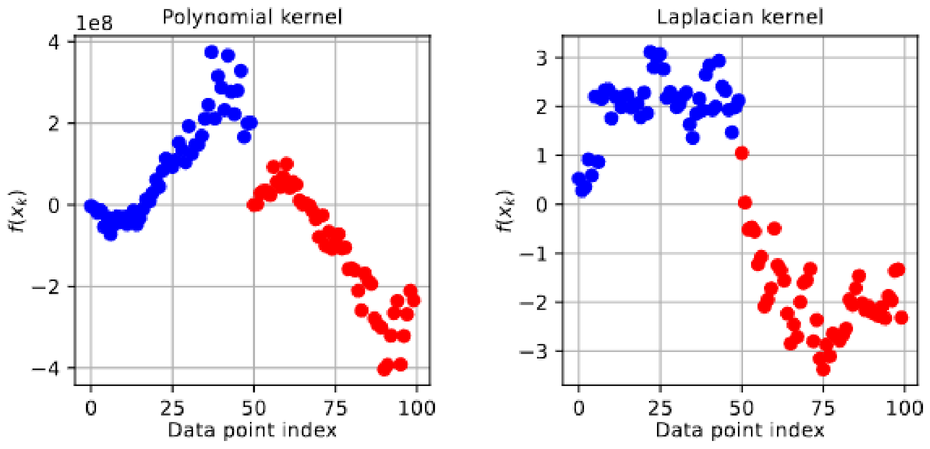


Figure 4: Problem 6.1 solution. Plots of the Representer theorem value for the data points computed using leave-out-out.

22 6.2

Problem 6.2 (5% weighting)
Now consider the kernel Ridge regression for a one-dimensional problem. We will use the Laplacian kernel $\kappa_l(\boldsymbol{x}, \boldsymbol{y})$ with $a = 0.5$. Suppose we have five observations, and that we have fitted a Kernel ridge regression problem with $C = 0.01$. The data and data-fit is:

n	1	2	3	4	5
t	0	2	3	6	7
y_n	1	3	2	3	2
θ_n	-0.11	2.82	-0.28	2.48	0.30

where n denotes the point index, t the time, y_n is the value for point n, and θ_n is the coefficient estimated using Kernel Ridge regression associated with point n.

Compute the regression value for $\hat{y}$ at $t = 1$, disregarding all points that have a kernel value lower that 0.1.

Answer

We know that the Laplacian kernel is:

$$\kappa(x, y) = \exp(-a||x - y||)$$

We are asked for $\hat{y}$ at $t = 1$ disregarding points below kernel value of 0.1.

First we need to find the distance between points x and y that would produce a kernel value lower than 0.1. We start by taking the log:

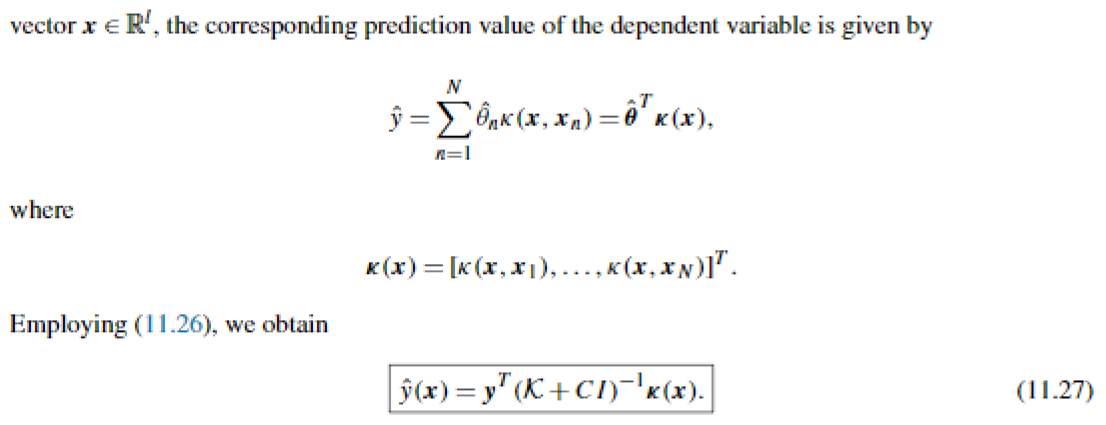
$$\ln \kappa(x, y) = -a||x - y||$$
$$\Leftrightarrow$$
$$||x - y|| = \frac{\ln \kappa(x, y)}{-a}$$
$$= \frac{\ln 0.1}{-0.5}$$

In [145…
```
xy_dist = np.log(0.1)/(-0.5)
round(xy_dist.item(),2)
```

Out[145…
```
4.61
```

For $t = 1$ we can ignore all the points above $t > 4.61$ (Solution actually states it is 5.61, but I cannot see why???)

p 553 we have


vector $x \in \mathbb{R}^l$, the corresponding prediction value of the dependent variable is given by

$$\hat{y} = \sum_{n=1}^{N} \hat{\theta}_n \kappa(\boldsymbol{x}, \boldsymbol{x}_n) = \hat{\boldsymbol{\theta}}^T \boldsymbol{\kappa}(\boldsymbol{x}),$$

where

$$\boldsymbol{\kappa}(\boldsymbol{x}) = [\kappa(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, \kappa(\boldsymbol{x}, \boldsymbol{x}_N)]^T.$$

Employing (11.26), we obtain

$$\boxed{\hat{y}(\boldsymbol{x}) = \boldsymbol{y}^T(\mathcal{K} + CI)^{-1}\boldsymbol{\kappa}(\boldsymbol{x}).} \qquad (11.27)$$

Hence:

$$\hat{y} = \sum_{n=1}^{N} \hat{\theta}_n \kappa(x, x_n) = \sum_{n=1}^{3} \hat{\theta}_n \kappa(t = 1, t_n),$$

where $x = t = 1$ and $x_n = t_n$ from the table. Inserting:

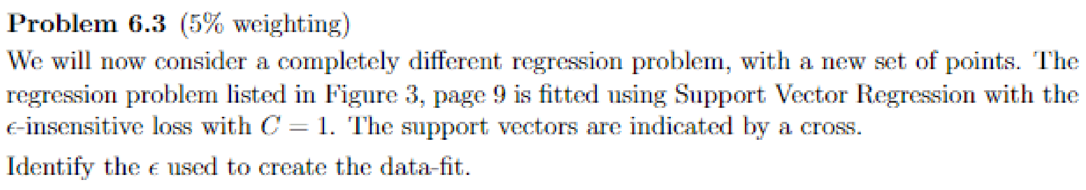
$$\hat{y} = \theta_1\kappa(1, 0) + \theta_2\kappa(1, 2) + \theta_3\kappa(1, 3)$$
$$= -0.11\exp(-0.5 \cdot ||1 - 0||) + 2.28\exp(-0.5 \cdot ||1 - 2||) - 0.28\exp(-0.5 \cdot ||1 - 3||$$

In [146…
```
a = -0.5
kappa1 = laplacian_kernel(1, 0, a)
kappa2 = laplacian_kernel(1, 2, a)
```

```
kappa3 = laplacian_kernel(1, 3, a)
print(f"kappa1 = {round(kappa1,2)}")
print(f"kappa2 = {round(kappa2,2)}")
print(f"kappa3 = {round(kappa3,2)}")
y1 =-0.11*kappa1 + 2.82*kappa2 - 0.28*kappa3
print(f"y1 = {round(y1,2)}")
```

```
kappa1 = 0.61
kappa2 = 0.61
kappa3 = 0.37
y1 = 1.54
```

22 6.3



Problem 6.3 (5% weighting)

We will now consider a completely different regression problem, with a new set of points. The regression problem listed in Figure 3, page 9 is fitted using Support Vector Regression with the ϵ-insensitive loss with $C = 1$. The support vectors are indicated by a cross.

Identify the ϵ used to create the data-fit.

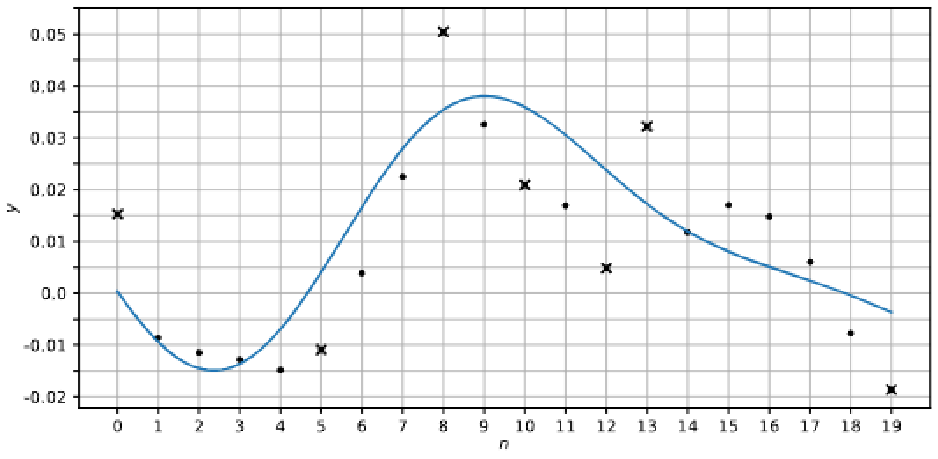
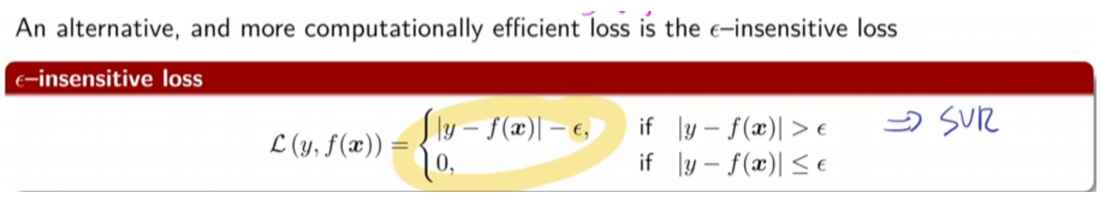


Figure 3: Problem 6.3.

Answer

We know that support vectors have to be *on* the epsilon tube or outside its region. Hence, the smallest error between the data fit provided and the support vector would be the ϵ that has been used. Going through each point, there is no support vector with an error smaller than 0.015, hence $\epsilon = 0.015$.

Another way to look at it is to see which point of the data fit (points) compared to the support vectors (cross-points) has the largest distance from the regression line (blue) and compare it to a support vector with the closest distance.



An alternative, and more computationally efficient loss is the ϵ–insensitive loss

ϵ–insensitive loss

$$\mathcal{L}(y, f(\boldsymbol{x})) = \begin{cases} |y - f(\boldsymbol{x})| - \epsilon, & \text{if } |y - f(\boldsymbol{x})| > \epsilon \\ 0, & \text{if } |y - f(\boldsymbol{x})| \le \epsilon \end{cases} \Rightarrow \text{SVR}$$

From slide 11 week 13!

Exam 2023

23 Problem 1

23 1.1

In this problem we will consider the following data

n	0	1	2	3
y_n	0.5	-1.2	2.0	4.1

Problem 1.1 (5% weighting)
Consider polynomial regression using the squared error as the loss function and assume i.i.d noise. For modeling the response y_n, we will use a polynomial model of the form $f_n = \theta_0 + \theta_1 \cdot n + \theta_2 \cdot n^2$.
Determine the parameters $\hat{\boldsymbol{\theta}} = [\theta_0\ \theta_1\ \theta_2]^T$ of the model that minimizes the error. Describe the process for calculating this estimate, including the relevant matrices and vectors.

Answer

$\theta = (X^T X)^{-1} X^T y$ where the design matrix is given by

$$X = \begin{bmatrix} 1 & 0 & 0^2 \\ 1 & 1 & 1^2 \\ 1 & 2 & 2^2 \\ 1 & 3 & 3^2 \end{bmatrix} \text{ and } y = [0.5 \quad -1.2 \ \ 2.0 \ \ 4.1]^T$$

In [147…

```
X = np.array([[1,1,1,1], [0,1,2,3], [0,1,4,9]]).T
y = np.array([0.5, -1.2, 2.0, 4.1])
theta = np.linalg.inv(X.T @ X) @ X.T @ y
print(f"theta = {theta}")
```

```
theta = [ 0.2   -1.45   0.95]
```

23 1.2

Using the same data, we consider a linear model on the form $f_n = \theta_0 + \theta_1 \cdot n$, and we assume non-white Gaussian noise, where the variance of the noise is 1, and the covariance of the noise between two successive samples is 0.2.
Determine the parameters $\hat{\boldsymbol{\theta}} = [\theta_0\ \theta_1]^T$ of the model that minimizes the squared error using this noise assumption. Write the relevant matrices and vectors used to estimate $\hat{\boldsymbol{\theta}}$.

Answer

Now we have a linear model assuming non-white Gaussian noise and variance of 1 and covariance of 0.2. Hence, chapter 3.10.1 matches perfectly "Linear regression: The nonwhite gaussian noise case", particularly eq (3.61).

$$\hat{\theta}_{ML} = (X^T\Sigma_\eta^{-1}X)^{-1}X^T\Sigma_\eta^{-1}y$$

From the information provided we know Σ_η has to be the same size of the number of observations, ie 4×4, where the diagonal is $\sigma_\eta^2 = 1$ and the two successive samples are given by 0.2 is the super- and subdiagonal.

Furthermore, we know that X is now only 2 columns given by model $f_n = \theta_0 + \theta_1 \cdot n$

$$\Sigma_\eta = \begin{bmatrix} 1 & 0.2 & 0 & 0 \\ 0.2 & 1 & 0.2 & 0 \\ 0 & 0.2 & 1 & 0.2 \\ 0 & 0 & 0.2 & 1 \end{bmatrix}$$

Plugging it in:

In [148…

```
Sigma_eta = np.array([[1.0, 0.2, 0.0, 0.0],
                      [0.2, 1.0, 0.2, 0.0],
                      [0.0, 0.2, 1.0, 0.2],
                      [0.0, 0.0, 0.2, 1.0]])
X = np.array([[1, 1, 1, 1],
              [0, 1, 2, 3]]).T
theta_ml = np.linalg.inv((X.T @ np.linalg.inv(Sigma_eta) @ X))@X.T@np.linalg.inv
print(f"theta_ml = {theta_ml}")
```

```
theta_ml = [-0.52  1.31]
```

23 1.3

General parameter estimation assuming parameter is estimated with 10 different datasets (realizations). For each one we have an unbiased estimator $\hat{\theta}_i$, where the variance of each individual estimator is $\sigma = 1$. We aggregate them using:

$$\hat{\theta}_{agg} = \frac{1}{N}\sum_{i=1}^{N}\hat{\theta}_i$$

Where it is given that $N = 10$. Compute the variance of the estimator $\hat{\theta}_{aagg} and specify formulas, numerical value and assumptions made.

Answer

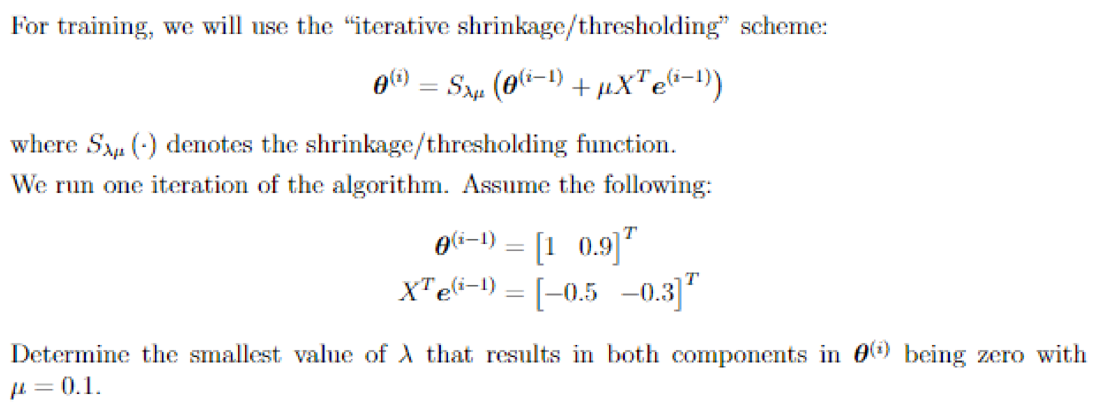
We CRTL+F'd unbiased estimator, and we see in ML book p 81 and 82 the exact problem definition is described. Here it is described as an average. **Hence, we know the aggregate (or average) estimator is an unbiased estimator and assuming they are mutually uncorrelated and have same variance, ie $\sigma_i = 1$ for all i.** Then, the variance of the new estimator is much smaller and given by:

$$\sigma^2_{\hat{\theta}_{agg}} = \mathbb{E}[(\hat{\theta}_{agg} - \theta_o)^2] = \frac{\sigma^2}{N}$$
$$= \frac{1}{10} = 0.1$$

23 Problem 2

In this problem we consider sparse learning and apply l_1 regularization and linear regression (LASSO).

23 2.1



For training, we will use the "iterative shrinkage/thresholding" scheme:

$$\boldsymbol{\theta}^{(i)} = S_{\lambda\mu}\left(\boldsymbol{\theta}^{(i-1)} + \mu X^T \boldsymbol{e}^{(i-1)}\right)$$

where $S_{\lambda\mu}(\cdot)$ denotes the shrinkage/thresholding function.
We run one iteration of the algorithm. Assume the following:

$$\boldsymbol{\theta}^{(i-1)} = \begin{bmatrix}1 & 0.9\end{bmatrix}^T$$
$$X^T\boldsymbol{e}^{(i-1)} = \begin{bmatrix}-0.5 & -0.3\end{bmatrix}^T$$

Determine the smallest value of λ that results in both components in $\boldsymbol{\theta}^{(i)}$ being zero with $\mu = 0.1$.

Answer

Using the IST (iterative shrinkage and thresholding) algorithm. We must determine the smallest value of λ that results in **both** components in $\theta^{(i)}$ being zero with $\mu = 0.1$.

Hence, since $S_{\lambda\mu}(\tilde{\theta}) = \text{sign}(\tilde{\theta})\text{max}(|\tilde{\theta}| - \lambda\mu, 0)$, we can rewrite the max() function terms as:

$$|\tilde{\theta}| - \lambda\mu = 0 \Leftrightarrow \lambda = \frac{|\tilde{\theta}|}{\mu}$$

First, we have to compute what $\tilde{\theta}$ is, given by:

$$\tilde{\theta} = \theta^{(i-1)} + \mu X^T e^{(i-1)} = \begin{bmatrix}1\\0.9\end{bmatrix} + 0.1 \cdot \begin{bmatrix}-0.5\\-0.3\end{bmatrix}$$

Inserting into it gives (below), hence λ has to be 9.5 and 8.7, respectively, for each parameter to be zero. Hence, for **both** to be zero the smallest value is $\lambda = 9.5$

In [149…

```
mu = 0.1
prev_theta = np.array([1, 0.9])
prev_error = np.array([-0.5, -0.3])
theta_tilde = prev_theta + mu * prev_error
print(f"theta_tilde = {theta_tilde}")

l = np.abs(theta_tilde) / mu
print(f"lambda = {l}")
```

```
theta_tilde = [0.95 0.87]
lambda = [9.5 8.7]
```

23 2.2

Assume we have chosen $\lambda = 1$ and an reliable estimate of the noise of the data is obtained such that $\sigma^2 = 1$. How does the original regression problem relate to the prior distribution of θ? Determine the parameters of the prior distribution of θ.

Answer

We know from exercise 9.1.7 that the LASSO regression is equivalent to the MAP estimate, if using a normal likelihood and a univariate zero-mean Laplace distribution on each weight component in θ.

Furthermore, we also see that if we reparametrize with $b = \frac{2\sigma^2}{\lambda} = \frac{2\cdot 1}{1} = 2$ we get the LASSO. So the Laplacian would then be:

$$\begin{aligned} p(x|\mu, b=2) &= \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right) \\ &= \frac{1}{2\cdot 2}\exp\left(-\frac{|x-\mu|}{2}\right) \\ &= \frac{1}{4}\exp\left(-\frac{|x-\mu|}{2}\right) \end{aligned}$$

23 Problem 3

Echo cancelling setup of a linear FIR filter with l coefficients depicted below:

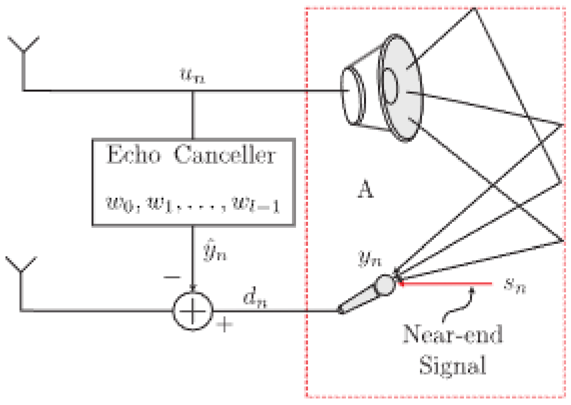
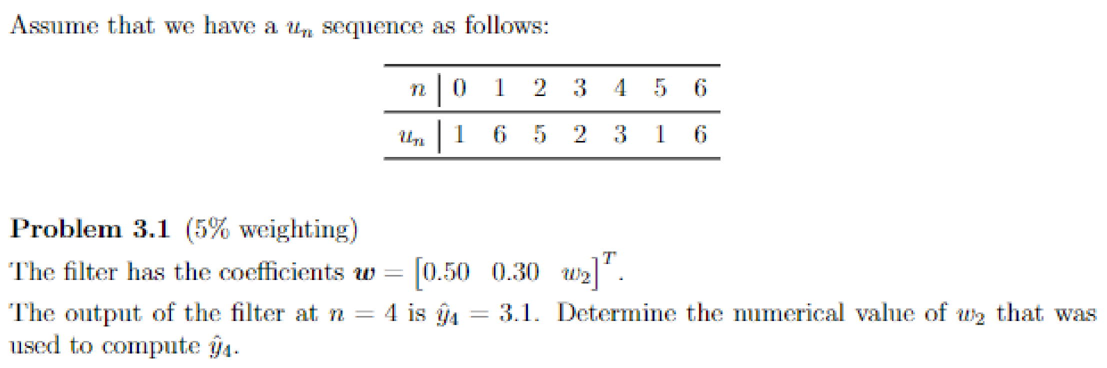


Figure 1: Problem 3 setup.

23 3.1

Assume we have a signal sequence u_n.


Assume that we have a u_n sequence as follows:

n	0	1	2	3	4	5	6
u_n	1	6	5	2	3	1	6

Problem 3.1 (5% weighting)
The filter has the coefficients $\boldsymbol{w} = \begin{bmatrix} 0.50 & 0.30 & w_2 \end{bmatrix}^T$.
The output of the filter at $n = 4$ is $\hat{y}_4 = 3.1$. Determine the numerical value of w_2 that was used to compute $\hat{y}_4$.

Given the filter coefficients w, and that the output of the filter at $n = 4$ is $\hat{y}_4 = 3.1$, determine the numerical value of w_2 that was used to compute $\hat{y}_4$

Answer

$\hat{y}_n$ is given by the convolution of u_n with the filter weights $w_{0:l-1}$ given by eq (4.41) p 135:

$$\hat{d}_n = \sum_{i=0}^{l-1} w_i u_{n-i}$$

$$\Rightarrow \hat{d}_4 = 3.1 = \sum_{i=0}^{3-1} w_i u_4 = w_0 u_4 + w_1 u_3 + w_2 u_2$$
$$= 0.50 \cdot 3 + 0.30 \cdot 2 + w_2 \cdot 5$$
$$\Leftrightarrow w_2 = \frac{3.1 - (0.5 \cdot 3 + 0.3 \cdot 2)}{5} = 0.2$$

The filter length l is hidden, but we can find it by observing w only had up to w_2 components, ie. $2 = l - 1 \Leftrightarrow l = 3$ long!

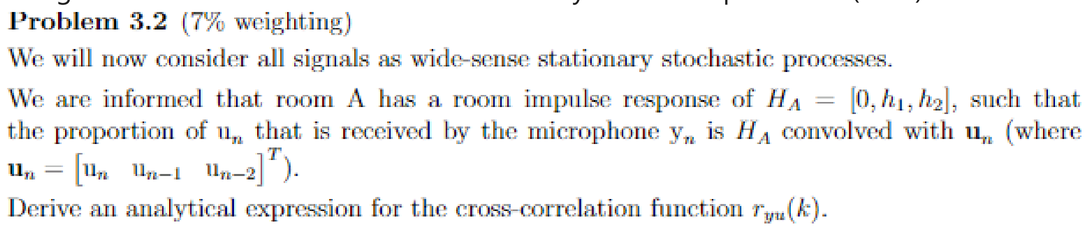
In [150…

```
w2 = (3.1-(0.5*3 + 0.3*2)) / 5
print(f"w2 = {w2}")
```

```
w2 = 0.2
```

23 3.2

All signals are considered wide-sense stationary stochastic processes (WSS).


Problem 3.2 (7% weighting)
We will now consider all signals as wide-sense stationary stochastic processes.
We are informed that room A has a room impulse response of $H_A = [0, h_1, h_2]$, such that the proportion of u_n that is received by the microphone y_n is H_A convolved with $\mathbf{u}_n$ (where $\mathbf{u}_n = \begin{bmatrix} \mathrm{u}_n & \mathrm{u}_{n-1} & \mathrm{u}_{n-2} \end{bmatrix}^T$).
Derive an analytical expression for the cross-correlation function $r_{yu}(k)$.

Derive an analytical expression for the cross-correlation function $r_{yu}(k)$.

Answer

We are told that y_n is a **room** impulse response H_A convolved with u_n (in addition to the signal!), ie. $y_n = s_n + H_A * u_n$

We know that:

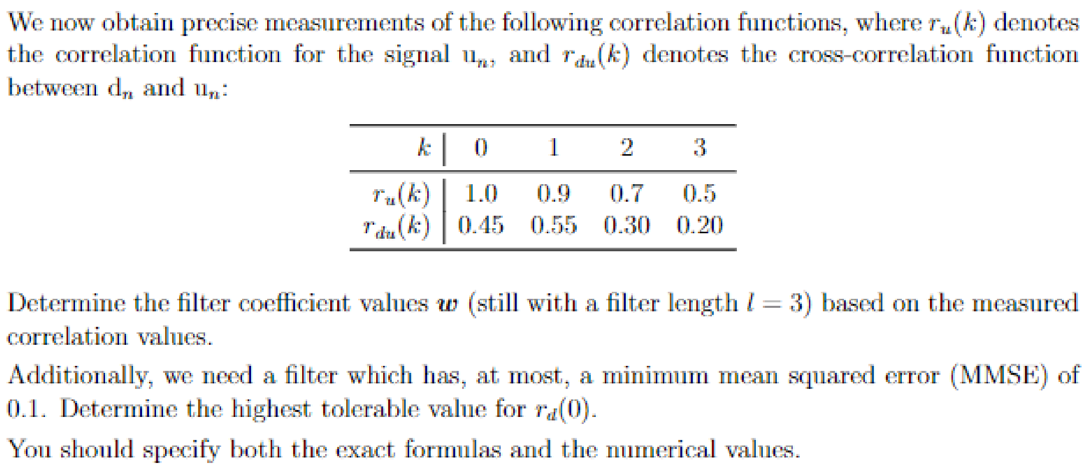
$$\begin{aligned}
r_{yu}(k) &= \mathbb{E}[y_n u_{n-k}] = \mathbb{E}[(s_n + H_A * u_n)(u_{n-k})] && (1)\\
&= \mathbb{E}[s_n u_{n-k} + H_A * u_n u_{n-k}] && (2)\\
&= \mathbb{E}[s_n u_{n-k} + (\sum_{i=0}^{2} h_i u_{n-i}) u_{n-k}] && (3)\\
&= \mathbb{E}[s_n u_{n-k} + h_0 u_{n-0} u_{n-k} + h_1 u_{n-1} u_{n-k} + h_2 u_{n-2} u_{n-k}] && (4)\\
&= \mathbb{E}[s_n u_{n-k} + 0 + h_1 u_{n-1} u_{n-k} + h_2 u_{n-2} u_{n-k}] && (5)\\
&= \mathbb{E}[s_n u_{n-k}] + \mathbb{E}[h_1 u_{n-1} u_{n-k}] + \mathbb{E}[h_2 u_{n-2} u_{n-k}] && (6)\\
&= \mathbb{E}[s_n u_{n-k}] + h_1 \mathbb{E}[u_{n-1} u_{n-k}] + h_2 \mathbb{E}[u_{n-2} u_{n-k}] && (7)\\
&= \mathbb{E}[s_n u_{n-k}] + h_1 \mathbb{E}[u_n u_{n-(k-1)}] + h_2 \mathbb{E}[u_n u_{n-(k-2)}] && (8)\\
&= r_{su}(k) + h_1 \cdot r_u(k-1) + h_2 \cdot r_u(k-2) && (9)
\end{aligned}$$

Here we have used the definition of convolution (3)-(4), the definition of $h_0 = 0$ (5), linearity of expectations (6), and moving constants outside the expectation (7), and lastly that it is a WSS signal, hence we can shift freely. Recall we want it on the form $\mathbb{E}[u_n u_{n-(k \pm D)}]$ where D is the displacement in total!

23 3.3

We are now given r_u and r_{du} for k of the signal u_n and d_n. Now, determine the values of the filter coefficient using a filter length of $l = 3$ based on the measurement values.

Additionally, we need a filter with **at most** minimum mean squared error (MMSE) of 0.1! Determine the highest tolerable value of $r_d(0)$. Specify formulas and numerical values!



We now obtain precise measurements of the following correlation functions, where $r_u(k)$ denotes the correlation function for the signal u_n, and $r_{du}(k)$ denotes the cross-correlation function between d_n and u_n:

k	0	1	2	3
$r_u(k)$	1.0	0.9	0.7	0.5
$r_{du}(k)$	0.45	0.55	0.30	0.20

Determine the filter coefficient values $\boldsymbol{w}$ (still with a filter length $l = 3$) based on the measured correlation values.

Additionally, we need a filter which has, at most, a minimum mean squared error (MMSE) of 0.1. Determine the highest tolerable value for $r_d(0)$.

You should specify both the exact formulas and the numerical values.

Answer

To determine the filter coefficient (with $l = 3$) and the MMSE we need to consult the book, particularly p.136 where:

$$R_u = \Sigma_u$$
$$\Sigma_u w = p \Leftrightarrow w = \Sigma_u^{-1} p = R_u^{-1} p$$

The MMSE can be found in ch. 4.2.1 p. 123 eq. (4.9), where we simply choose the one applicable to our case. In this case θ is not known, hence we have to use the first option listed here:

$$J(\theta) = \sigma_y^2 - p^T \Sigma_x^{-1} p = \sigma_y^2 \theta_*^T \Sigma_x \theta_* = \sigma_y^2 - p^T \theta_*$$

where $\Sigma_x = \Sigma_u$, as u is our input and $y = d$ is our output, where we recall that the first option $r_d(0) = \sigma_d^2$ of its cross-correlation matrix Σ_d.

We are informed $J(\theta) = MMSE = 0.1$, so we are solving for σ_d^2:

$$J(\theta) = 0.1 = \sigma_y^2 - p^T \Sigma_x^{-1} p$$
$$\Leftrightarrow$$
$$\sigma_d^2 = r_d(0) = 0.1 + p^T \Sigma_u^{-1} p = 0.8$$

$$w = \begin{bmatrix} r_u(0) & r_u(1) & r_u(2) \\ r_u(1) & r_u(0) & r_u(1) \\ r_u(2) & r_u(1) & r_u(0) \end{bmatrix}^{-1} \begin{bmatrix} r_{du}(0) \\ r_{du}(1) \\ r_{du}(2) \end{bmatrix} =$$

In [151…

```python
np.set_printoptions(3)
ru0=1.0; ru1=0.9; ru2=0.7; ru3=0.5
rdu0=0.45; rdu1=0.55; rdu2=0.30; rdu3=0.20
R_u = np.array([[ru0, ru1, ru2],
                [ru1, ru0, ru1],
                [ru2, ru1, ru0]])
p = np.array([[rdu0],
              [rdu1],
              [rdu2]])
print(f"R_u = \n{R_u}")
print(f"p = \n{p}")
w = np.linalg.inv(R_u) @ p
print(f"w = \n{w}")

rd0 = 0.1 + p.T @ np.linalg.inv(R_u) @ p
print(f"r_d(0) = {round(rd0.item(), 2)}")
```

```
R_u = 
[[1.  0.9 0.7]
 [0.9 1.  0.9]
 [0.7 0.9 1. ]]
p = 
[[0.45]
 [0.55]
 [0.3 ]]
w = 
[[-1.25]
 [ 3.25]
 [-1.75]]
r_d(0) = 0.8
```

23 3.4

Now the signal u_n is changed with new correlation values and we decide to learn the filter weights w with the LMS algorithm. We are given the knowledge of the covariance matrix of u_n where u_n is asumed wide-sense stationary stochastic process (WSS), ie:

$$\Sigma_u = \begin{bmatrix} 2.00 & 1.40 & 0.70 \\ 1.40 & 2.00 & 1.40 \\ 0.70 & 1.40 & 2.00 \end{bmatrix}$$

We want to choose a step size μ such that we are sure the filter converges and the weight estimator has **bounded** variance. Determine the maximum value for step-size μ.

Answer

To ensure that the filter both **converges** and the weight estimator has a **bounded variance** for a wide-sense stationary signal, we can use the following formula(s) to determine the maximum step size μ.

- p. 200 states that, in a stationary environment (WSS), LMS **converges** to optimal MSE solution in the mean:

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

- However, in order for the **variance to be bounded** we must refer to eq. (5.45), where we have to check:

$$0 < \mu < \frac{2}{\text{trace}\{\Sigma_x\}}$$

where $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix R_u.

Hence we will take the lower of the two, such that it fulfill **both** conditions.

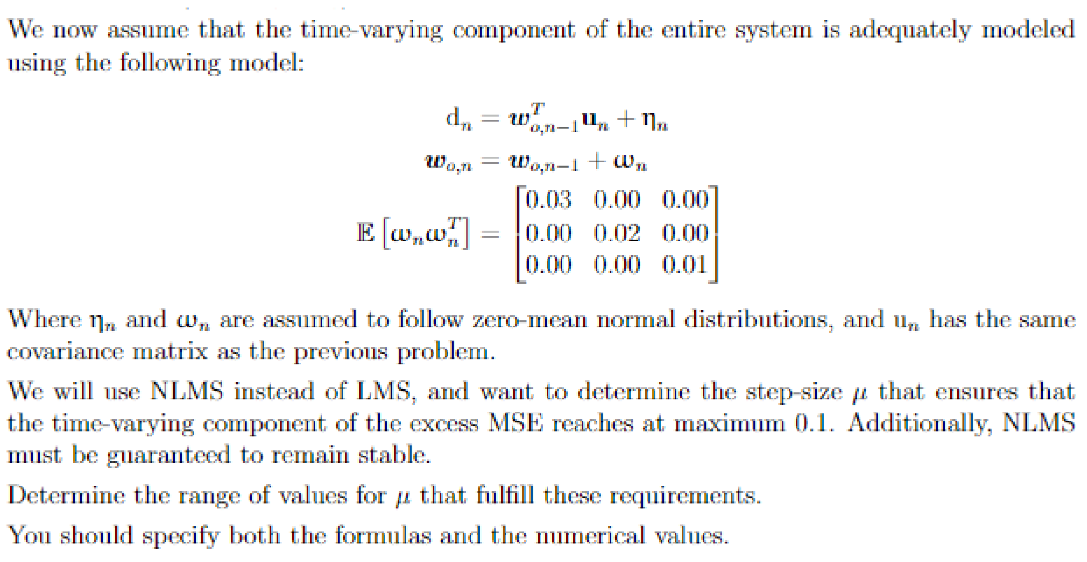
In [152…

```
R_u = np.array([[2.0, 1.40, 0.70],
                [1.40, 2.00, 1.40],
                [0.70, 1.40, 2.00]])
eigenvalues = np.linalg.eigvals(R_u)
lambda_max = np.max(eigenvalues)
mu_max = 2 / lambda_max

print(f"Max mu (convergence in LMS): {round(mu_max,2)}")
print(f"Maximum mu (bounded variance): {round(2 / np.trace(R_u),2)}")
```

```
Max mu (convergence in LMS): 0.46
Maximum mu (bounded variance): 0.33
```

23 3.5

Assume the time-varying component of the system is adequately modeled:



We now assume that the time-varying component of the entire system is adequately modeled using the following model:

$$\mathrm{d}_n = \boldsymbol{w}_{o,n-1}^T \mathbf{u}_n + \eta_n$$
$$\boldsymbol{w}_{o,n} = \boldsymbol{w}_{o,n-1} + \boldsymbol{\omega}_n$$
$$\mathbb{E}\left[\boldsymbol{\omega}_n \boldsymbol{\omega}_n^T\right] = \begin{bmatrix} 0.03 & 0.00 & 0.00 \\ 0.00 & 0.02 & 0.00 \\ 0.00 & 0.00 & 0.01 \end{bmatrix}$$

Where η_n and $\boldsymbol{\omega}_n$ are assumed to follow zero-mean normal distributions, and $\mathbf{u}_n$ has the same covariance matrix as the previous problem.

We will use NLMS instead of LMS, and want to determine the step-size μ that ensures that the time-varying component of the excess MSE reaches at maximum 0.1. Additionally, NLMS must be guaranteed to remain stable.

Determine the range of values for μ that fulfill these requirements.

You should specify both the formulas and the numerical values.

We will use NLMS instead of LMS to determine step-size that ensures the excess MSE reaches at maximum 0.1, and NLMS must be guaranteed to remain stable. Determine a step size that fulfills these requirements.

Answer

The formula can be found for the excess MSE in table 6.1 in the ML book p276:

$$J_{exc} = \frac{1}{2}\mu\sigma_\eta^2 \text{trace}\{\Sigma_x\}\mathbb{E}\left[\frac{q}{||x||^2}\right] + \frac{1}{2}\mu^{-1}\text{trace}\{\Sigma_x\}\text{trace}\{\Sigma_\omega\}$$

where $q = 1$ gives the NLMS result.

p. 212 states stability of the NLMS is guaranteed iff. $0 < \mu < 2$!

The $\text{trace}\{\Sigma_\omega\}$ part is the contributor only time-varying part, so we can look at that specifically.

$$J_{MSE}^{NLMS} = \frac{1}{2}\mu^{-1}\text{trace}\{\Sigma_x\}\text{trace}\{\Sigma_\omega\}$$
$$\Leftrightarrow$$
$$\mu = \frac{1}{2 \cdot J_{MSE}^{NLMS}}\text{trace}\{\Sigma_x\}\text{trace}\{\Sigma_\omega\}$$

In [153…

```python
J_MSE = 0.1
R_u = np.array([[2.0, 1.40, 0.70],
                [1.40, 2.00, 1.40],
                [0.70, 1.40, 2.00]])
R_omega = np.array([[0.03, 0.00, 0.00],
                    [0.00, 0.02, 0.00],
                    [0.00, 0.00, 0.01]])
mu_ = 1 / (2*0.1) * np.trace(R_u) * np.trace(R_omega)
print(f"{round(mu_,2)} < mu < 2")
```

```
1.8 < mu < 2
```

23 Problem 4

Concerns ICA with mutual information.

23 4.1

Assume we have two sources s_1 and s_2 which are statistically independent, and two observable variables x_1 and x_2 defined as:

$$x_1 = s_1 + s_2, \quad x_2 = s_2$$

Unmixing the signals using ICA can be done using a unmixing matrix W:

$$\begin{bmatrix} \hat{s}_1 \\ \hat{s}_2 \end{bmatrix} = W \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Identify W such that $\hat{s}_1$ and $\hat{s}_2$ such that the average mutual information $I(\hat{s}_1, \hat{s}_2) = 0$. Show formally that it is true for your solution.

Answer

The mixing matrix given $x = As$ can be easily read from the exercise as:

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

An obvious choice of unmixing would then be $W = A^{-1}$, since we just move it to the other side by inversion.

The average mutual information is defined for (2.158) and (2.150), however since we are in the continuous case the former is preferred, however since s_1 and s_2 are statistically independent, it can be seen that $p(s_1, s_2) = p(s_1)p(s_2)$, and this would result in $\ln(1)$ which is 0, hence the integrand itself goes to 0. So the choice of W does not matter.

Given two continuous random variables, the average mutual information is defined as

$$\boxed{I(\mathrm{x};\mathrm{y}) := \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}\,dx\,dy} \tag{2.158}$$

23 4.2

We know that the ICA model cannot identify the scale and direction of the identified vectors of the unmixing matrix W, and consequently in the mixing matrix A.

Write a short proof that shows the ICA model has a scaling ambiguity, that is, for a specific identified W (or A), this W (or A) can be scaled, vector-wise, and still result in sources that are the same, up to scaling factor.

Answer

Solution: This can be shown in a few ways. One approach is to start from the ICA model. We have $\mathbf{x} = A\mathbf{s}$. We can now create a new matrix, denoted C, with the same dimensions as A, as

$$\begin{bmatrix} c_1 & 0 & \cdots & 0 \\ 0 & c_2 & \cdots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \cdots & c_D \end{bmatrix}$$

where c_i is a non-zero scalar. In that case, C is invertible, and $CC^{-1} = I$. We can thus write the ICA model as

$$\begin{aligned} \mathbf{x} &= ACC^{-1}\mathbf{s} \\ &= A'\mathbf{s}' \end{aligned}$$

where $A' = AC$ and $\mathbf{s}' = C^{-1}\mathbf{s}$. Since the matrix C is a diagonal matrix, the mixing matrix and source matrix is only scaled, and the resulting $\mathbf{x}$ will be the same.

23 Problem 5

Hidden Markov Models

23 5.1

Suppose a 3-state HMM with initial state probability vector P_k and transition probabilities P_{ij}. We will construct a state sequence of length 8 by using sub-sequences of each state A,B,C,D exactly once. Specify the composition that creates the most likely state sequence given the defined HMM and justify your choices.

Suppose a 3-state HMM have the initial state probability vector

$$P_k = \begin{bmatrix} 0.25 & 0.25 & 0.50 \end{bmatrix}^T$$

and the following state transition probabilities

$$P_{ij} = \begin{bmatrix} 0.5 & 0.5 & 0.0 \\ 0.0 & 0.2 & 0.8 \\ 0.0 & 0.9 & 0.1 \end{bmatrix}$$

We will construct a state sequence of length 8, by using each of the following state sub-sequences exactly once:

$$A = \begin{bmatrix} 1 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 3 \end{bmatrix}, \quad C = \begin{bmatrix} 3 & 2 \end{bmatrix}, \quad D = \begin{bmatrix} 3 & 3 \end{bmatrix}$$

E.g., the composition 'ABCD' will give the state sequence

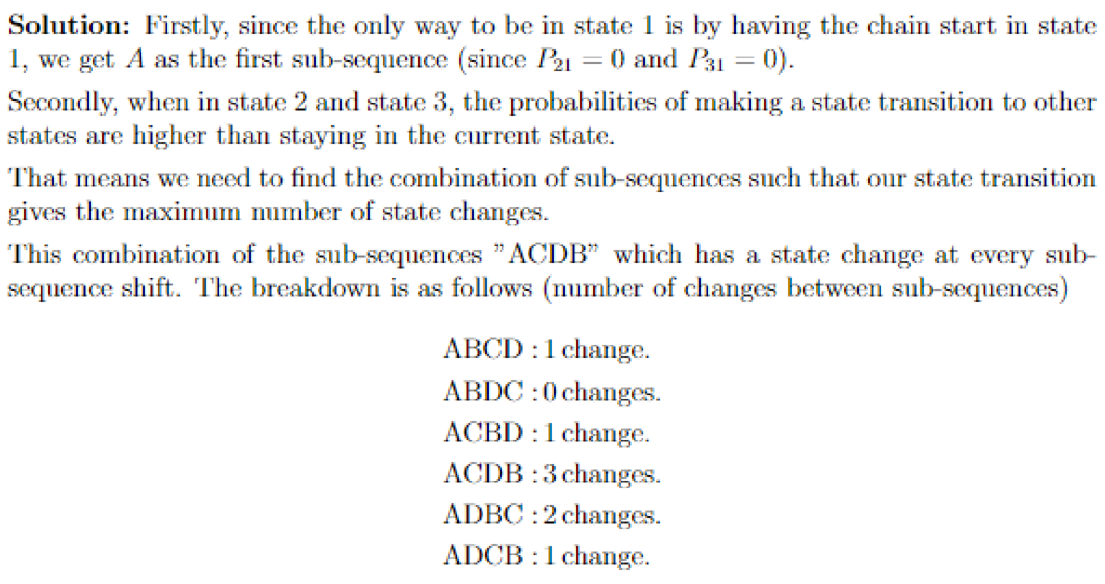
$$x_n = \begin{bmatrix} 1 & 2 & 2 & 3 & 3 & 2 & 3 & 3 \end{bmatrix}$$

Specify the composition that creates the most likely state sequence given the defined HMM. Justify your choices.

Answer

First, we most carefully inspect the P_k and P_{ij}. The only way to be in state 1 is to start the chain in state 1! So A is the first sub-sequence (since $P_{21} = 0$ and $P_{31} = 0$)

When in state 2 or 3, probability of transitioning to other states is higher than staying. Hence, combination of subsequences that change the most is the goal.



Solution: Firstly, since the only way to be in state 1 is by having the chain start in state 1, we get A as the first sub-sequence (since $P_{21} = 0$ and $P_{31} = 0$).

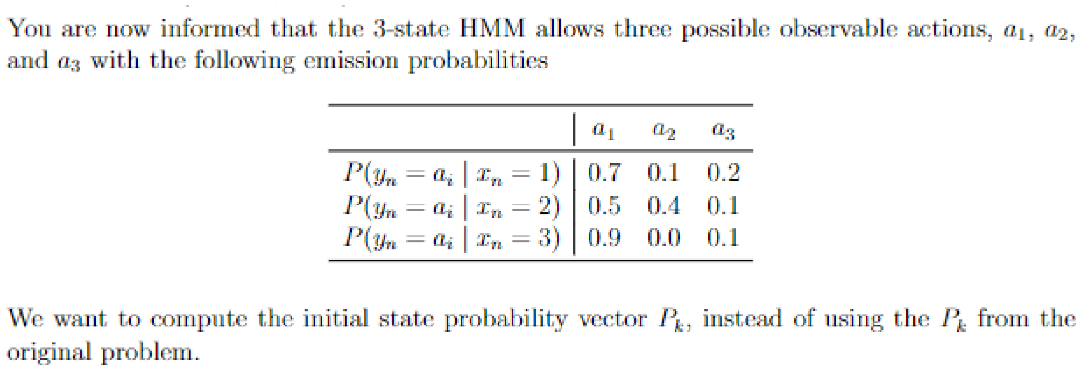
Secondly, when in state 2 and state 3, the probabilities of making a state transition to other states are higher than staying in the current state.

That means we need to find the combination of sub-sequences such that our state transition gives the maximum number of state changes.

This combination of the sub-sequences "ACDB" which has a state change at every sub-sequence shift. The breakdown is as follows (number of changes between sub-sequences)

$$\begin{aligned}&\text{ABCD : 1 change.}\\&\text{ABDC : 0 changes.}\\&\text{ACBD : 1 change.}\\&\text{ACDB : 3 changes.}\\&\text{ADBC : 2 changes.}\\&\text{ADCB : 1 change.}\end{aligned}$$

23 5.2



You are now informed that the 3-state HMM allows three possible observable actions, a_1, a_2, and a_3 with the following emission probabilities

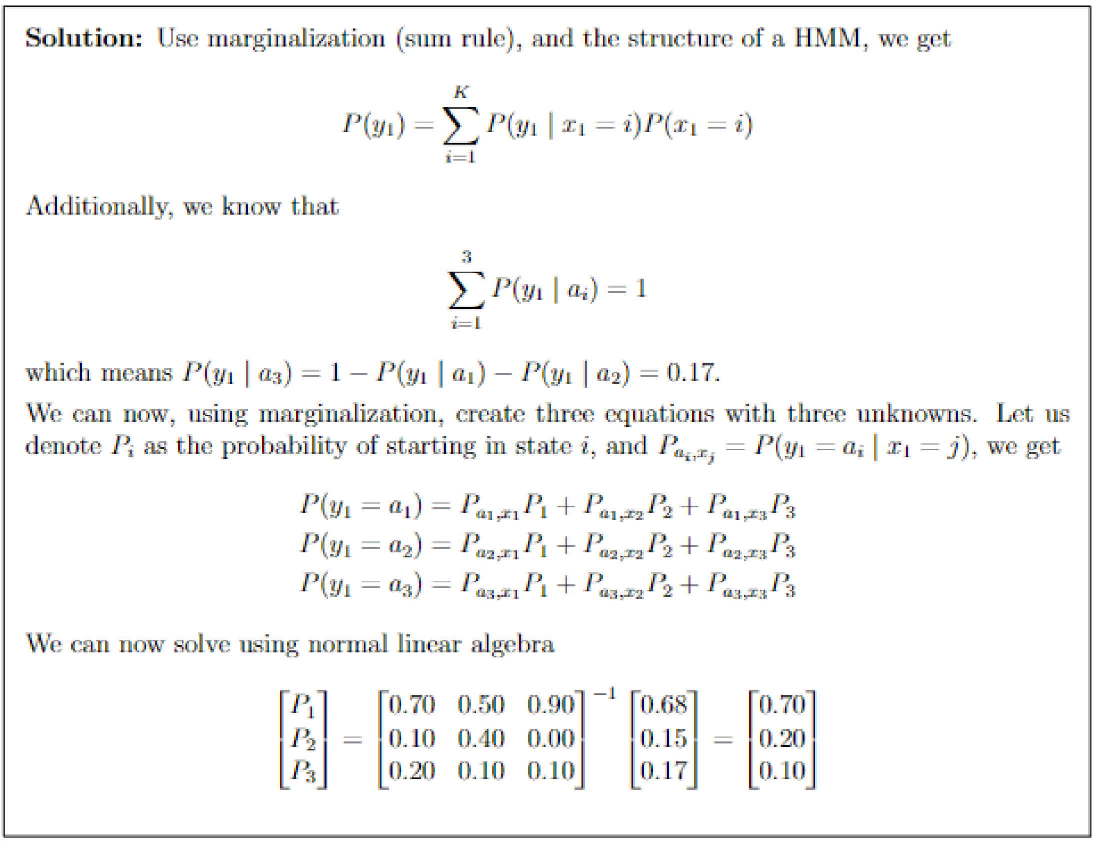
	a_1	a_2	a_3
$P(y_n = a_i \mid x_n = 1)$	0.7	0.1	0.2
$P(y_n = a_i \mid x_n = 2)$	0.5	0.4	0.1
$P(y_n = a_i \mid x_n = 3)$	0.9	0.0	0.1

We want to compute the initial state probability vector P_k, instead of using the P_k from the original problem.

Based on the observations you know that $P(y_1 = a_1) = 0.68$ and $P(y_1 = a_2) = 0.15$.

Compute the initial state probability vector P_k given this information.

Answer



Solution: Use marginalization (sum rule), and the structure of a HMM, we get

$$P(y_1) = \sum_{i=1}^{K} P(y_1 \mid x_1 = i)P(x_1 = i)$$

Additionally, we know that

$$\sum_{i=1}^{3} P(y_1 \mid a_i) = 1$$

which means $P(y_1 \mid a_3) = 1 - P(y_1 \mid a_1) - P(y_1 \mid a_2) = 0.17$.

We can now, using marginalization, create three equations with three unknowns. Let us denote P_i as the probability of starting in state i, and $P_{a_i,x_j} = P(y_1 = a_i \mid x_1 = j)$, we get

$$\begin{aligned} P(y_1 = a_1) &= P_{a_1,x_1}P_1 + P_{a_1,x_2}P_2 + P_{a_1,x_3}P_3 \\ P(y_1 = a_2) &= P_{a_2,x_1}P_1 + P_{a_2,x_2}P_2 + P_{a_2,x_3}P_3 \\ P(y_1 = a_3) &= P_{a_3,x_1}P_1 + P_{a_3,x_2}P_2 + P_{a_3,x_3}P_3 \end{aligned}$$

We can now solve using normal linear algebra

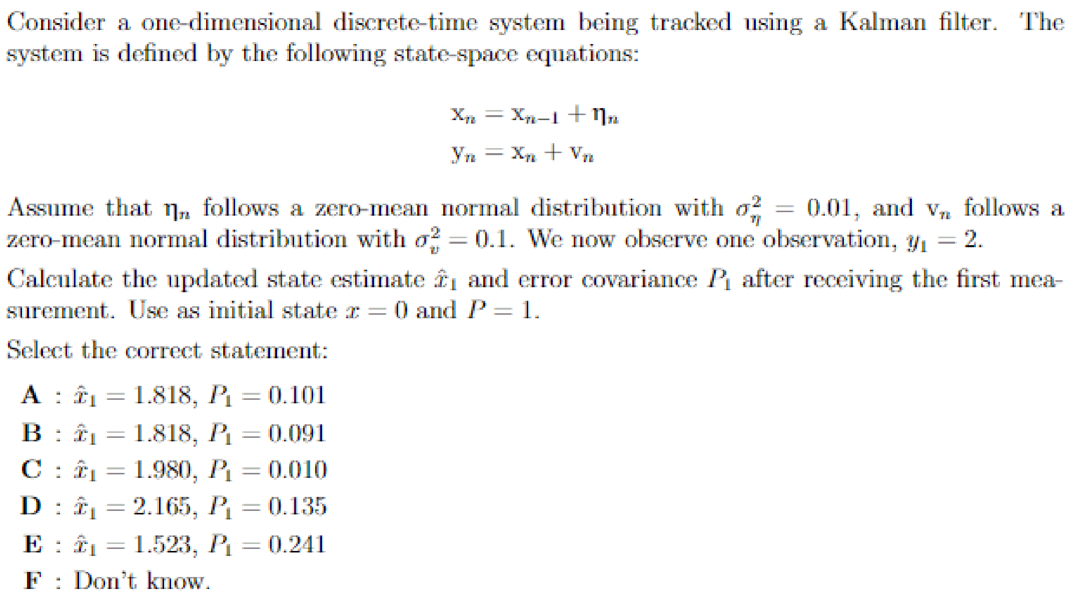
$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = \begin{bmatrix} 0.70 & 0.50 & 0.90 \\ 0.10 & 0.40 & 0.00 \\ 0.20 & 0.10 & 0.10 \end{bmatrix}^{-1} \begin{bmatrix} 0.68 \\ 0.15 \\ 0.17 \end{bmatrix} = \begin{bmatrix} 0.70 \\ 0.20 \\ 0.10 \end{bmatrix}$$

23 Problem 6

Kalman filtering

23 6.1

Consider a one-dimensional discrete time system tracked by a Kalman filter. It is defined by the following state-space equations... Calculate the updated estimate and error covariance after the first measurement is in. Use initial state $x =$ and $P = 1$



Consider a one-dimensional discrete-time system being tracked using a Kalman filter. The system is defined by the following state-space equations:

$$\begin{aligned} \mathrm{x}_n &= \mathrm{x}_{n-1} + \eta_n \\ \mathrm{y}_n &= \mathrm{x}_n + \mathrm{v}_n \end{aligned}$$

Assume that η_n follows a zero-mean normal distribution with $\sigma_\eta^2 = 0.01$, and v_n follows a zero-mean normal distribution with $\sigma_v^2 = 0.1$. We now observe one observation, $y_1 = 2$.

Calculate the updated state estimate $\hat{x}_1$ and error covariance P_1 after receiving the first measurement. Use as initial state $x = 0$ and $P = 1$.

Select the correct statement:

A : $\hat{x}_1 = 1.818$, $P_1 = 0.101$
B : $\hat{x}_1 = 1.818$, $P_1 = 0.091$
C : $\hat{x}_1 = 1.980$, $P_1 = 0.010$
D : $\hat{x}_1 = 2.165$, $P_1 = 0.135$
E : $\hat{x}_1 = 1.523$, $P_1 = 0.241$
F : Don't know.

Answer

p 169 has algorithm 4.2 the Kalman Filtering algorithm. We must calculate after correction, ie. $\hat{x}_{n|n}$ and $P_{n|n}$ is our goal.

Let's denote what we know from the problem, ie.

- $R_n = \sigma_v^2 = 0.1$
- $H_n = 1$
- $F_n = 1$
- $P_{n|n-1} = P_{1|0} = 1$
- $y_n = y_1 = 2$
- $\hat{x}_{n|n-1} = x_0 = 0$

Using the algorithm:

$$\begin{aligned}
S_n &= R_n + H_n P_{n|n-1} H_n^T = 0.1 + 1 \cdot 1 \cdot 1 = 1.1 \\
K_n &= P_{n|n-1} H_n^T S_n^{-1} = 1 \cdot 1 \cdot 1.1^{-1} = \frac{1}{1.1} = 0.909 \\
\hat{x}_{n|n} &= \hat{x}_{n|n-1} + K_n(y_n - H_n \hat{x}_{n|n-1}) = 0 + 0.909 \cdot (2 - 1 \cdot 0) = 1.818 \\
P_{n|n} &= P_{n|n-1} - K_n H_n P_{n|n-1} = 1 - 0.909 \cdot 1 \cdot 1 = 0.091
\end{aligned}$$

So correct is B.

23 Problem 7

Kernel methods

23 7.1

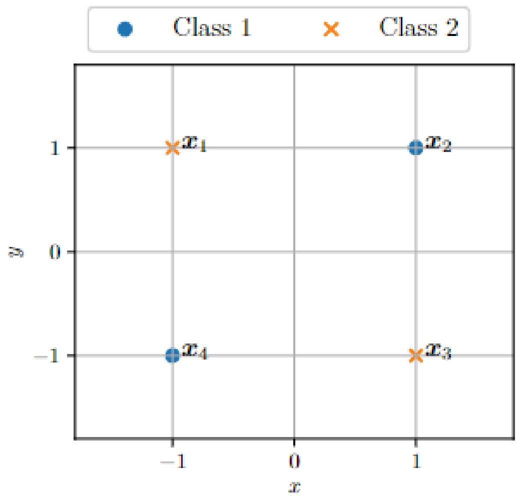


Figure 2: Problem 7.1.

In this problem we will use the Gaussian kernel

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2}\right)$$

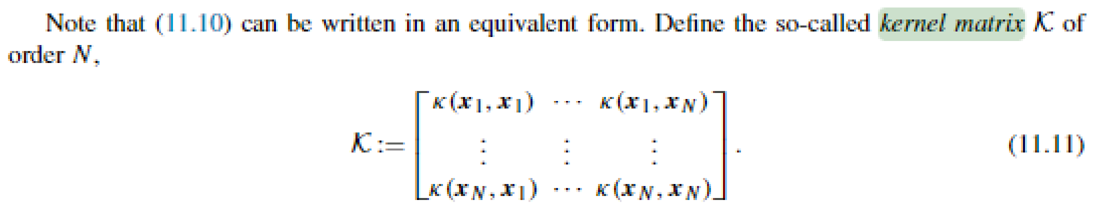
We have observed the data points as depicted on Figure 2, page 15 as training data.

Write up the complete kernel matrix $\mathcal{K}$ for the data, where you have used the kernel function to evaluate the points. Use $\sigma = 2$.

You should specify both the formulas and the numerical values.

Answer

We use



Note that (11.10) can be written in an equivalent form. Define the so-called *kernel matrix* $\mathcal{K}$ of order N,

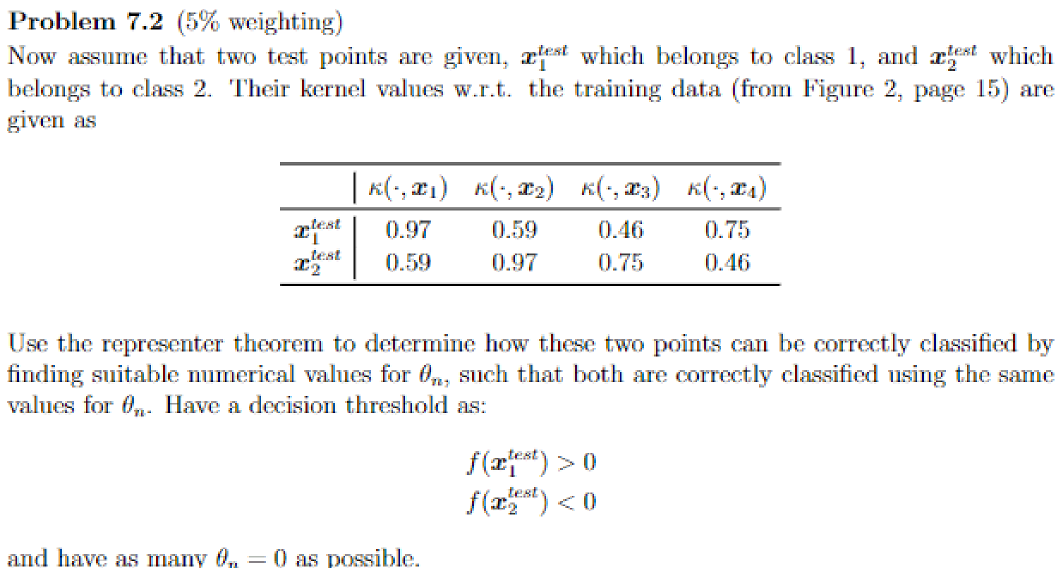
$$\mathcal{K} := \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}. \quad (11.11)$$

In [154…

```
K = np.zeros((4,4)) # placeholder
# Problem definitions
sigma = 2
x1 = np.array([-1,1])
x2 = np.array([1,1])
x3 = np.array([1,-1])
x4 = np.array([-1,-1])
all_x = [x1, x2, x3, x4]
for i in range(K.shape[0]):
    for j in range(K.shape[1]):
        K[i,j] = np.exp(-np.linalg.norm(all_x[i] - all_x[j])**2 / (2*sigma**2))
print(f"K = \n{K}")
```

```
K = 
[[1.    0.607 0.368 0.607]
 [0.607 1.    0.607 0.368]
 [0.368 0.607 1.    0.607]
 [0.607 0.368 0.607 1.   ]]
```

23 7.2



Problem 7.2 (5% weighting)
Now assume that two test points are given, $\boldsymbol{x}_1^{test}$ which belongs to class 1, and $\boldsymbol{x}_2^{test}$ which belongs to class 2. Their kernel values w.r.t. the training data (from Figure 2, page 15) are given as

	$\kappa(\cdot,\boldsymbol{x}_1)$	$\kappa(\cdot,\boldsymbol{x}_2)$	$\kappa(\cdot,\boldsymbol{x}_3)$	$\kappa(\cdot,\boldsymbol{x}_4)$
$\boldsymbol{x}_1^{test}$	0.97	0.59	0.46	0.75
$\boldsymbol{x}_2^{test}$	0.59	0.97	0.75	0.46

Use the representer theorem to determine how these two points can be correctly classified by finding suitable numerical values for θ_n, such that both are correctly classified using the same values for θ_n. Have a decision threshold as:

$$f(\boldsymbol{x}_1^{test}) > 0$$
$$f(\boldsymbol{x}_2^{test}) < 0$$

and have as many $\theta_n = 0$ as possible.

Answer

The representer theorem given in ML (11.17):

$$f(\cdot) = \sum_{n=1}^{N} \theta_n \kappa_l(\cdot, x_n),$$

where x_n are the training points, and θ_n are coefficients determined during training using a test point x.

Most times I've seen a decision boundary around 0 it corresponds to at least 2 different θ 's which have values $\theta_1 = 1$ and $\theta_2 = -1$, respectively for each class. This would give "opposite" signs for each test point x_n^{test}, so let's try and see if that's enough and we can set the rest to $\theta_n = 0$.

$$f(x_1^{test}) = \theta_1\kappa(\cdot, x_1) + \theta_2\kappa(\cdot, x_2) = 1 \cdot 0.97 + (-1) \cdot 0.59 = 0.38$$
$$f(x_2^{test}) = \theta_1\kappa(\cdot, x_1) + \theta_2\kappa(\cdot, x_2) = 1 \cdot 0.59 + (-1) \cdot 0.97 = -0.38$$

Hence, for $f(x) > 0$ gives class 1 and $f(x) < 0$ gives class two, which satisfies the decision threshold!

23 7.3

We consider a regression problem using support vector regression, fitted on five data points. We know that the data is fitted using $C = 1$ and $\epsilon = 0.1$, and the bias is estimated to $\hat{\theta}_0 = 0$. We have the following information:

n	1	2	3	4	5
$y_n - \hat{y}_n$	-0.14	0.04	0.11	-0.01	0.10
$\kappa(\boldsymbol{x}^{test}, \boldsymbol{x}_n)$	0.38	0.92	0.84	0.28	0.03

Computer either an exact value for $\hat{y}(\boldsymbol{x}^{test})$, or a tight bound for $\hat{y}(\boldsymbol{x}^{test})$, whichever is possible. You should specify both the formulas and the numerical values.

Answer

Solution: From ML p 557, we have the prediction formula eq (11.36) given as

$$\hat{y}(\boldsymbol{x}^{test}) = \sum_{n=1}^{N}(\tilde{\lambda}_n - \lambda_n)\kappa(\boldsymbol{x}^{test}, \boldsymbol{x}_n) + \hat{\theta}_0$$

From ML p 556 eq. (11.31-11.32), and the remarks on ML p 559, we know that $\tilde{\lambda} = C$ when the error $e_n = y_n - \hat{y}_n > \epsilon$, and $\lambda = C$ when the error $e_n = y_n - \hat{y}_n < \epsilon$ (for $n = 1$ and $n = 3$).
For $|e_n| < \epsilon$, we have $\tilde{\lambda}_n = \lambda_n = 0$ ($n = 2$ and $n = 4$).
Finally, for $\boldsymbol{x}_5$, the error is exactly equal to ϵ, so we know possible values are $0 \leq \tilde{\lambda}_5 \leq C$.
Putting all this information together, we get

$$\begin{aligned}\hat{y}(\boldsymbol{x}^{test}) &= -\lambda_1\kappa(\boldsymbol{x}^{test}, \boldsymbol{x}_1) + \tilde{\lambda}_3\kappa(\boldsymbol{x}^{test}, \boldsymbol{x}_3) + \tilde{\lambda}_5\kappa(\boldsymbol{x}^{test}, \boldsymbol{x}_5) \\ &= -0.38 + 0.84 + \tilde{\lambda}_5 0.03 \\ &= 0.46 + \tilde{\lambda}_5 0.03\end{aligned}$$

Thus we can conclude $0.46 \leq \hat{y}(\boldsymbol{x}^{test}) \leq 0.49$.