## SEEM2460 Introduction to Data Science

**The solutions for Lab Assignment**

---

**Question 1.** Assume we have a simplified version of the Animal Classification dataset[1] which includes properties of animals as descriptive features and the animal species as target feature. In our dataset, the animals are classified as being Mammals or Reptiles based on whether they are toothed and have legs, as shown in Table 1. In this question, you are asked to develop a decision tree based on this simplified dataset.

Table 1: Animal Classification dataset

| Instance | Toothed | Legs | Species |
|---|---|---|---|
| 1 | T | T | Mammal |
| 2 | T | T | Mammal |
| 3 | T | F | Reptile |
| 4 | F | T | Mammal |
| 5 | T | T | Mammal |
| 6 | T | T | Mammal |
| 7 | T | F | Reptile |
| 8 | T | F | Reptile |
| 9 | T | T | Mammal |
| 10 | F | T | Reptile |

(a) Calculate the resulting Gini index when splitting on the attribute "Toothed" and "Legs", respectively (i.e. $Gini_{split"Toothed"}$ and $Gini_{split"Legs"}$). Show your calculation details. Which attribute would be chosen as the first splitting attribute? (10 pts)

(b) Based on the decision in Question 1.a, draw a two-level decision tree if needed using both attributes for splitting. Mark the class label in each leaf node. In case of a tie on the "Mammal" and "Reptile" instances in a leaf node, mark the node as "-". (4 pts)

(c) **WEKA Tool Practice**. Use the WEKA tool to classify the data with decision tree (J48) under the test option "Use training set". Copy the result in 'classifier output' window to your assignment. (6 pts)

[1] the UCI Zoo Dataset

**Solution:**

(a)

After splitting on Attribute “Toothed”,

| | Toothed=T | Toothed=F |
|---|---|---|
| Mammal | 5 | 1 |
| Reptile | 3 | 1 |

$\text{Gini}(\text{Toothed} = \text{T}) = 1 - \left(\frac{5}{8}\right)^2 - \left(\frac{3}{8}\right)^2 = \frac{15}{32}$ (1 pts)

$\text{Gini}(\text{Toothed} = \text{F}) = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = \frac{1}{2}$ (1 pts)

$Gini_{split"Toothed"} = \frac{8}{10} * \frac{15}{32} + \frac{2}{10} * \frac{1}{2} = \frac{19}{40}$ (2 pts)

After splitting on Attribute “Legs”,

| | Legs=T | Legs=F |
|---|---|---|
| Mammal | 6 | 0 |
| Reptile | 1 | 3 |

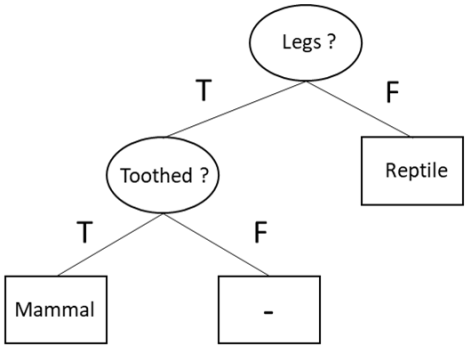
$\text{Gini}(\text{Legs} = \text{T}) = 1 - \left(\frac{6}{7}\right)^2 - \left(\frac{1}{7}\right)^2 = \frac{12}{49}$ (1 pts)

$\text{Gini}(\text{Legs} = \text{F}) = 1 - \left(\frac{0}{3}\right)^2 - \left(\frac{3}{3}\right)^2 = 0$ (1 pts)

$Gini_{split"Legs"} = \frac{7}{10} * \frac{12}{49} + \frac{3}{10} * 0 = \frac{6}{35}$ (2 pts)

Since $Gini_{split"Toothed"} > Gini_{split"Legs"}$, we will choose Attribute “Legs” as the first splitting attribute. (2 pts)

(b)



Tree hierarchical structure 2 pts, and tree leaf node class label 2 pts.

(c)

(6 pts)

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2

Relation: Lab_Assignment

Instances: 10

Attributes: 3

Toothed

Legs

Species

Test mode: evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree

---

Legs = T: Mammal (7.0/1.0)

Legs = F: Reptile (3.0)

Number of Leaves : 2

Size of the tree : 3

Time taken to build model: 0 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

| | | | |
|---|---|---|---|
| Correctly Classified Instances | 9 | 90 | % |
| Incorrectly Classified Instances | 1 | 10 | % |
| Kappa statistic | 0.7826 | | |
| Mean absolute error | 0.1714 | | |
| Root mean squared error | 0.2928 | | |
| Relative absolute error | 35.468 | % | |
| Root relative squared error | 59.7269 | % | |
| Total Number of Instances | 10 | | |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | MCC | ROC Area | PRC Area | Class |
|---|---|---|---|---|---|---|---|---|---|
| | 1.000 | 0.250 | 0.857 | 1.000 | 0.923 | 0.802 | 0.875 | 0.857 | Mammal |
| | 0.750 | 0.000 | 1.000 | 0.750 | 0.857 | 0.802 | 0.875 | 0.850 | Reptile |
| Weighted Avg. | 0.900 | 0.150 | 0.914 | 0.900 | 0.897 | 0.802 | 0.875 | 0.854 | |

=== Confusion Matrix ===

| a | b | <-- classified as |
|---|---|---|
| 6 | 0 | a = Mammal |
| 1 | 3 | b = Reptile |

**Question 2.** In class, we learn how to solve the sparse recovery problem:

$$\min \ |x_1| + |x_2| + |x_3| + |x_4| + |x_5|$$

$$s.t. \quad \underbrace{\begin{bmatrix} 0 & -1 & 0 & -1 & 1 & 0 \\ -2 & 1 & 0 & 2 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 & 2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 2 \\ 1 \\ 1 \\ -3 \end{bmatrix}}_{b}$$

We find a solution $x = (0,1,0,0,3,0)$ with four entries being zero. Now, instead of finding a solution $x$ to $Ax = b$ with as many zero entries as possible, we want to find a solution to $Ax = b$ that minimizes the first two entries. This motivates the following optimization formulation:

$$\min \ |x_1| + |x_2|$$

$$s.t. \quad \begin{bmatrix} 0 & -1 & 0 & -1 & 1 & 0 \\ -2 & 1 & 0 & 2 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1 \\ -3 \end{bmatrix}$$

Please use "cvxpy" to solve the above sparse recovery problem. (10 pts: 8pts for code, 2pts for answer)

**Solution:**

```
status: optimal
Found a feasible x in R^6 that has 4 nonzeros.
x_L1 is:  [-0.000000 0.000000 -2.000000 1.000000 3.000000 1.000000]
optimal objective value: 3.625802090241523e-23
```

```
import cvxpy as cp
import numpy as np

A = np.array([[0,-1,0,-1,1,0],
              [-2,1,0,2,0,-1],
              [0,1,0,0,0,1],
              [0,0,1,0,-1,2]])

b = np.array([2,1,1,-3])

c = np.array([[1,0,0,0,0,0],
              [0,1,0,0,0,0],
              [0,0,0,0,0,0],
              [0,0,0,0,0,0],
              [0,0,0,0,0,0],
              [0,0,0,0,0,0]])
x_L1 = cp.Variable(shape=6)
constraints = [A * x_L1 == b]
```

```
obj = cp.Minimize(cp.norm(x_L1*c, 1))
prob = cp.Problem(obj, constraints)
prob.solve()
print("status: {}".format(prob.status))

nnz_l1 = (np.absolute(x_L1.value) > 1e-8).sum()
np.set_printoptions(formatter={'float_kind':'{:f}'.format})
print('Found a feasible x in R^{} that has {} nonzeros.'.format(6, nnz_l1))
print('x_L1 is: ', x_L1.value)
print('optimal objective value: {}'.format(obj.value))
```

**Question 3**. In lecture we have learned two ideas to tackle the problem of background extraction. These two ideas lead to an optimization formulation as follows:

$$\begin{array}{ll} \text{minimize} & (|y_1^1| + |y_2^1| + \cdots + |y_n^1|) + \cdots + (|y_1^m| + |y_2^m| + \cdots + |y_n^m|) \\ \text{subject to} & f^i = x + y^i \quad \text{for } i = 1, \ldots, m. \end{array} \tag{$S_1$}$$

(a) We have three figures extracted from one video, i.e., m = 3. The three figures are denoted as M1, M2 and M3, with the same size of 130*160, i.e., n = 130*160. Part of the cvxpy code is shown as below. Can you modify the code to obtain an implementation of the above formulation? You can find the figures in attachment. (10 pts)

(b) Please run your code for ($S_1$) with your own figure(s) and examine the result. How is the background extraction? Please attach the figures you use and the results you obtain. (bonus: 10 pts)

*Cvxpy Code for Background-Extraction Formulation ($S_\infty$)*

```
import numpy as np
import cv2
import cvxpy as cp
from cvxpy import *
import matplotlib.pyplot as plt
```

```
im1 = cv2.imread('/content/Figure1.png',cv2.IMREAD_GRAYSCALE)
im2 = cv2.imread('/content/Figure2.png',cv2.IMREAD_GRAYSCALE)
im3 = cv2.imread('/content/Figure3.png',cv2.IMREAD_GRAYSCALE)
M_size = im1.shape
size_a = M_size[0]
size_b = M_size[1]
n = size_a*size_b


M1 = im1.reshape(n,-1)
M2 = im2.reshape(n,-1)
M3 = im3.reshape(n,-1)


w = cp.Variable((n,1))
# Please trying to implementing you code here:
################################################
plt.figure(figsize=(6,6))
plt.imshow((M1 - w.value).reshape(size_a, size_b), cmap='gray')
plt.figure(figsize=(6,6))
plt.imshow((M2 - w.value).reshape(size_a, size_b), cmap='gray')
plt.figure(figsize=(6,6))
plt.imshow((M3 - w.value).reshape(size_a, size_b), cmap='gray')
plt.figure(figsize=(6,6))
plt.imshow((w.value).reshape(size_a, size_b), cmap='gray')
--END--
```

**Solution:**

(a)

*Cvxpy Code for Background-Extraction Formulation ($S_\infty$)*

```
import numpy as np
import cv2
import cvxpy as cp
from cvxpy import *
import matplotlib.pyplot as plt

im1 = cv2.imread('/content/Figure1.png',cv2.IMREAD_GRAYSCALE)
im2 = cv2.imread('/content/Figure2.png',cv2.IMREAD_GRAYSCALE)
im3 = cv2.imread('/content/Figure3.png',cv2.IMREAD_GRAYSCALE)
M_size = im1.shape
size_a = M_size[0]
size_b = M_size[1]
n = size_a*size_b

M1 = im1.reshape(n,-1)
M2 = im2.reshape(n,-1)
M3 = im3.reshape(n,-1)

w = cp.Variable((n,1))
# Please trying to implementing you code here:
################################################
obj = cp.norm(M1-w,1) + cp.norm(M2-w,1) + cp.norm(M3-w,1)
prob = cp.Problem(cp.Minimize(obj))
prob.solve()
################################################
```

```
plt.figure(figsize=(6,6))
plt.imshow((M1 - w.value).reshape(size_a, size_b), cmap='gray')
plt.figure(figsize=(6,6))
plt.imshow((M2 - w.value).reshape(size_a, size_b), cmap='gray')
plt.figure(figsize=(6,6))
plt.imshow((M3 - w.value).reshape(size_a, size_b), cmap='gray')
plt.figure(figsize=(6,6))
plt.imshow((w.value).reshape(size_a, size_b), cmap='gray')
--END--
```