1] Consider modified objective function of regularized logistic regression, Zack Berger

$$J(w) = -\sum_{n=1}^{\tilde{N}} [\, y_n \log \sigma(w^T x_n) + (1-y_n) \log(1-\sigma(w^T x_n)) \,] + \frac{1}{2}\sum_i w_i^2$$

Find partial derivative $\frac{\partial J}{\partial w_j}$, then derive gradient descent update rule.

$$\frac{\partial J}{\partial w_j} = -\sum_{n=1}^{\tilde{N}} \frac{\partial}{\partial w_j} [\, y_n \log \sigma(w^T x_n) + (1-y_n)\log(1-\sigma(w^T x_n)) \,] + \frac{1}{2}\sum_i \frac{\partial}{\partial w_j}(w_i^2)$$

$$= -\sum_{n=1}^{\tilde{N}} \Big[y_n \cdot \frac{1}{\sigma(w^T x_n)} \cdot \sigma(w^T x_n)(1-\sigma(w^T x_n)) x_{nj} + (1-y_n) \cdot \frac{-1}{(1-\sigma(w^T x_n))} \sigma(w^T x_n)(1-\sigma(w^T x_n)) x_{nj} \Big] + \frac{1}{2} \cdot 2 w_j$$

$$= -\sum_{n=1}^{\tilde{N}} [\, y_n (1-\sigma(w^T x_n)) x_{nj} + (y_n - 1)\sigma(w^T x_n) x_{nj} \,] + w_j$$

$$= -\sum_{n=1}^{\tilde{N}} [\, y_n x_{nj} - y_n \sigma(w^T x_n) x_{nj} + y_n \sigma(w^T x_n) x_{nj} - \sigma(w^T x_n) x_{nj} \,] + w_j$$

$$= -\sum_{n=1}^{\tilde{N}} [\, y_n x_{nj} - \sigma(w^T x_n) x_{nj} \,] + w_j$$

$$= \sum_{n=1}^{\tilde{N}} [\, x_{nj} (\sigma(w^T x_n) - y_n) \,] + w_j \implies \boxed{\frac{\partial J}{\partial w_j} = \sum_{n=1}^{\tilde{N}} [\, x_{nj}(\sigma(w^T x_n) - y_n) \,] + w_j}$$

Gradient Descent update rule for j^{th} weight is given by

$$w_j^{(k+1)} = w_j^{(k)} - \eta \cdot \frac{\sum_{n=1}^{\tilde{N}} [\, x_{nj}(\sigma(w^T x_n) - y_n) \,] + w_j^{(k)}}{\left\| \sum_{n=1}^{\tilde{N}} [\, x_{nj}(\sigma(w^T x_n) - y_n) \,] + w_j^{(k)} \right\|}$$

where k denotes the iteration and η denotes the learning rate. For all the weights as a vector, we get

$$w^{(k+1)} = w^{(k)} - \eta \cdot \frac{\sum_{n=1}^{\tilde{N}} [\, x_n(\sigma(w^T x_n) - y_n) \,] + w^{(k)}}{\left\| \sum_{n=1}^{\tilde{N}} [\, x_n(\sigma(w^T x_n) - y_n) \,] + w^{(k)} \right\|}$$

2| Show that MAP estimation is equivalent to minimizing $J(w)$ from Q1, assuming standard Gaussian prior $N(0, I)$ for w. For $Z \sim N(0, I_m)$, we have $f_Z(z) = \frac{1}{(2\pi)^{m/2}} \exp\left(-\sum_{i=1}^{m} \frac{z_i^2}{2}\right)$. Also, MAP estimation: $w^* = \arg\max_w \prod_{i=1}^{n} P(y_i | x_i, w) f(w)$.

Goal: maximize $\prod_{i=1}^{n} P(y_i | x_i, w) f(w)$.

we have that $\max \prod_{i=1}^{n} P(y_i | x_i, w) f(w)$

$$\| $$

$$\max \log\left(\prod_{i=1}^{n} P(y_i | x_i, w) f(w)\right)$$

$$\| $$

$$\max \sum_{i=1}^{n} \log\left(P(y_i | x_i, w) f(w)\right)$$

$$\| $$

$$\min -\sum_{i=1}^{n} \log\left(P(y_i | x_i, w) f(w)\right).$$

Noting that $P(y_i | x_i, w) = [\sigma(w^T x_i)]^{y_i} [1 - \sigma(w^T x_i)]^{1-y_i}$, the goal amounts to taking $\min\left(-\sum_{i=1}^{n} [y_i \log(\sigma(w^T x_i)) + (1-y_i)\log(1-\sigma(w^T x_i)) + \log f(w)]\right)$.

Let $L(w) = -\sum_{i=1}^{n} [y_i \log(\sigma(w^T x_i)) + (1-y_i)\log(1-\sigma(w^T x_i))] - \sum_{i=1}^{n} \log f(w)$

Then, $\frac{\partial L}{\partial w} = \sum_{i=1}^{n} x_i(\sigma(w^T x_i) - y_i) - \sum_{i=1}^{n} \frac{\partial}{\partial w} \log f(w)$.

↳ $\frac{\partial}{\partial w_j} \log f(w) = \frac{\partial}{\partial w_j} \log\left(\frac{1}{(2\pi)^{m/2}} e^{-\sum_{i=1}^{m} \frac{w_i^2}{2}}\right)$ for $w \in \mathbb{R}^m$.

$$= \frac{\partial}{\partial w_j} \log\left(\frac{1}{(2\pi)^{m/2}}\right) + \frac{\partial}{\partial w_j} \log\left(e^{-\sum_{i=1}^{m} \frac{w_i^2}{2}}\right)$$

$$= 0 + \frac{\partial}{\partial w_j} \left(-\sum_{i=1}^{m} \frac{w_i^2}{2}\right) \log e$$

$$= \frac{\partial}{\partial w_j} -\frac{1}{2} \sum_{i=1}^{m} w_i^2 = -w_j$$

$\Rightarrow \frac{\partial}{\partial w} \log f(w) = -w$.

→

Therefore, $\frac{\partial L}{\partial w} = \sum_{i=1}^{n} x_i (\sigma(w^T x_i) - y_i) - \sum_{i=1}^{n} (-w)$

$= \sum_{i=1}^{n} x_i (\sigma(w^T x_i) - y_i) + w$

In summation, the problem of MAP estimation amounts to minimizing
$L(w) = -\sum_{i=1}^{n} [y_i \log(\sigma(w^T x_i)) + (1-y_i) \log(1 - \sigma(w^T x_i)) + \log f(w)]$.

As derived, $\frac{\partial L}{\partial w} = \sum_{i=1}^{n} x_i (\sigma(w^T x_i) - y_i) + w$.

This gradient is identical to that obtained in Q1 when one takes $\frac{\partial J}{\partial w}$, because $\frac{\partial J}{\partial w} = \sum_{i=1}^{n} x_i (\sigma(w^T x_i) - y_i) + w$. There, the goal is to minimize $J(w)$.

In both the MAP estimation and also in the problem of minimizing the logistic regression objective with L2 regularization, obtaining w amounts to performing gradient descent with the same gradient.

Hence, the two problems are equivalent.

3| Build a restaurant decision tree from chart, answer the questions...

a) What is entropy of IsGoodRestaurant? Let IsGoodRestaurant = IGR.

↳ $P(\text{IsGoodRestaurant} = 1) = \frac{6}{8}$ $P(IGR = 0) = \frac{2}{8}$.

$\Rightarrow H(IGR) = -\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} \approx 0.811278$

$H(\text{IsGoodRestaurant}) = 0.8113$

b) Calculate $H(IGR \mid \text{HasOutdoorSeating})$. For brevity, let HasOutdoorSeating = HOS.

↳ $P(HOS = 0) = \frac{3}{8}$ $P(HOS = 1) = \frac{5}{8}$

$P(IGR = 1 \mid HOS = 1) = \frac{P(IGR = 1, HOS = 1)}{P(HOS = 1)} = \frac{3/8}{5/8} = \frac{3}{5}$

$P(IGR = 0 \mid HOS = 1) = \frac{2}{5}$

$P(IGR = 1 \mid HOS = 0) = \frac{P(IGR = 1, HOS = 0)}{P(HOS = 0)} = \frac{3/8}{3/8} = 1$

$P(IGR = 0 \mid HOS = 0) = 0$

$\Rightarrow H(IGR \mid HOS = 1) = -P(IGR = 1 \mid HOS = 1)\log_2 P(IGR = 1 \mid HOS = 1) - P(IGR = 0 \mid HOS = 1)\log_2 P(IGR = 0 \mid HOS = 1)$

$= -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97095$

$\Rightarrow H(IGR \mid HOS = 0) = -P(IGR = 1 \mid HOS = 0)\log_2 P(IGR = 1 \mid HOS = 0) - P(IGR = 0 \mid HOS = 0)\log_2 P(IGR = 0 \mid HOS = 0)$

$= -1\log_2 1 - 0\log_2 0 = 0$

$H(IGR \mid HOS) = P(HOS = 0)H(IGR \mid HOS = 0) + P(HOS = 1)H(IGR \mid HOS = 1)$

$= \frac{3}{8}(0) + \frac{5}{8}(0.97095) = 0.606844$

$H(\text{IsGoodRestaurant} \mid \text{HasOutdoorSeating}) \approx 0.606844$

c) Calculate conditional entropy conditioning on other three features.
For brevity, let HasBar = HB, IsClean = IC and HasGoodAtmosphere = HGA.

i. $H(IGR \mid HB)$

- $P(HB=0) = \frac{4}{8}$ $\quad P(HB=1) = \frac{4}{8}$
- $P(IGR=1 \mid HB=1) = \frac{2/8}{4/8} = \frac{2}{4}$ $\quad P(IGR=0 \mid HB=1) = \frac{2}{4}$
- $P(IGR=1 \mid HB=0) = 1$ $\quad P(IGR=0 \mid HB=0) = 0$

$\Rightarrow H(IGR \mid HB=1) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1$

$H(IGR \mid HB=0) = -1\log_2 1 - 0\log_2 0 = 0$

$\Rightarrow H(IGR \mid HB) = \frac{4}{8}\cdot 0 + \frac{4}{8}\cdot 1 = \frac{4}{8} = \frac{1}{2} = H(IGR \mid HB)$

ii. $H(IGR \mid IC)$

- $P(IC=0) = \frac{4}{8}$ $\quad P(IC=1) = \frac{4}{8}$
- $P(IGR=1 \mid IC=1) = \frac{3}{4}$ $\quad P(IGR=0 \mid IC=1) = \frac{1}{4}$
- $P(IGR=1 \mid IC=0) = \frac{3}{4}$ $\quad P(IGR=0 \mid IC=0) = \frac{1}{4}$

$\Rightarrow H(IGR \mid IC=1) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.811278$

$H(IGR \mid IC=0) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.811278$

$\Rightarrow H(IGR \mid IC) = \frac{4}{8}(0.811278) + \frac{4}{8}(0.811278) = 0.811278 = H(IGR \mid IC)$

iii. $H(IGR \mid HGA)$

- $P(HGA=0) = \frac{3}{8}$, $P(HGA=1) = \frac{5}{8}$ $\quad$ • $P(IGR=1 \mid HGA=1) = 1$, $P(IGR=0 \mid HGA=1) = 0$
- $P(IGR=1 \mid HGA=0) = \frac{1}{3}$, $P(IGR=0 \mid HGA=0) = \frac{2}{3}$

$\Rightarrow H(IGR \mid HGA=1) = -1\log_2 1 - 0\log_2 0 = 0$

$H(IGR \mid HGA=0) = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.918295$

$\Rightarrow H(IGR \mid HGA) = \frac{3}{8}(0.918295) + \frac{5}{8}(0) = 0.34436 = H(IGR \mid HGA)$

$H(\text{IsGoodRestaurant} \mid \text{HasBar}) = 0.5$
$H(\text{IsGoodRestaurant} \mid \text{IsClean}) = 0.811278$
$H(\text{IsGoodRestaurant} \mid \text{HasGoodAtmosphere}) = 0.34436$

d) Calculate info gain for splitting on each attribute.

- $IG(IGR \mid HOS) = H(IGR) - H(IGR \mid HOS) = 0.8113 - 0.606844 = 0.20446$
- $IG(IGR \mid HB) = 0.8113 - 0.5 = 0.3113$
- $IG(IGR \mid IC) = 0.8113 - 0.8113 = 0.0$
- $IG(IGR \mid HGA) = 0.8113 - 0.34436 = 0.4669$

$IG(IGR \mid HOS) = 0.204$
$IG(IGR \mid HB) = 0.311$
$IG(IGR \mid IC) = 0.0$
$IG(IGR \mid HGA) = 0.467$

e) Based on the information gain, the first attribute to split on is HasGoodAtmosphere.

f) Initial tree:

→ noting that $H(IsGoodRestaurant \mid HasGoodAtmosphere = 1) = 0$, we stop splitting on that branch and label its leaf TRUE (1).

→ Now, need to split on the branch with 3 data points, summarized here:

Sample	HasOutdoorSeating	HasBar	IsClean	IsGoodRestaurant
2	1	1	0	0
5	1	1	1	0
6	1	0	1	1

→

By observation, splitting on HasBar will give us our max info gain. In fact, it will allow us to terminate splitting, as evidenced by the following computation showing $H(IGR \mid HasBar = 0) = H(IGR \mid HasBar = 1) = 0$

↳ $P(HB=0) = \frac{1}{3}$, $P(HB=1) = \frac{2}{3}$ $P(IGR=1 \mid HB=1) = 0$, $P(IGR=0 \mid HB=1) = 1$

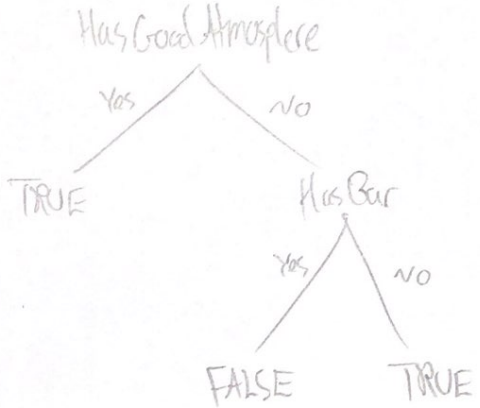
$P(IGR=1 \mid HB=0) = 1$, $P(IGR=0 \mid HB=0) = 0$

$\Rightarrow H(IGR \mid HB=0) = -1\log_2 1 - 0\log_2 0 = 0$

$H(IGR \mid HB=1) = -1\log_2 1 - 0\log_2 0 = 0$

$\Longrightarrow H(IGR \mid HB=0) = H(IGR \mid HB=1) = 0$

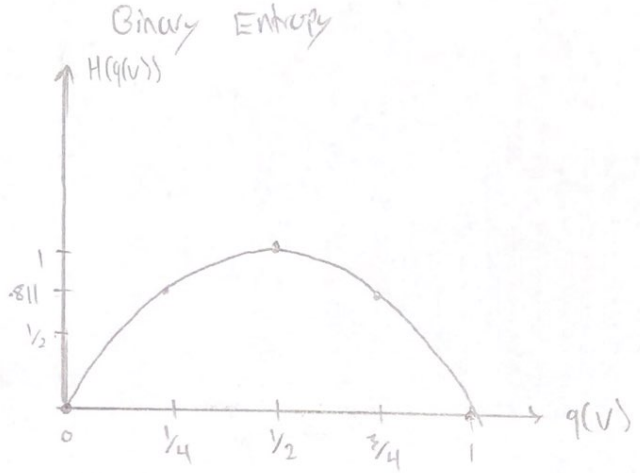
Hence, the full decision tree to answer the question "IsGoodRestaurant" is...

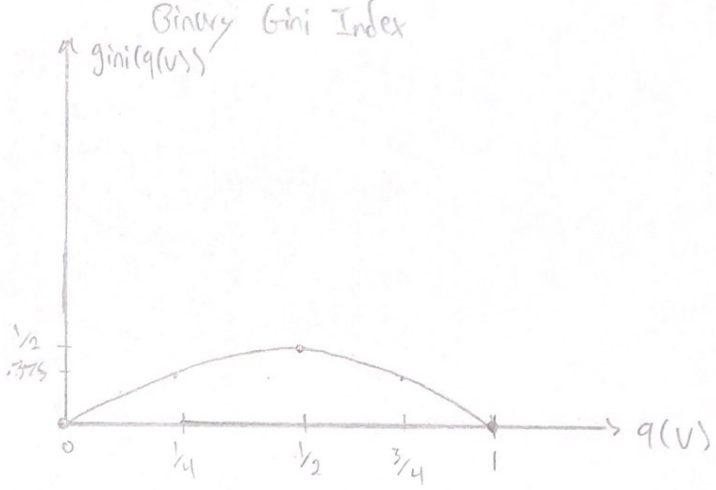


9) Based on the above decision tree, restaurants 9 of 10 have a good atmosphere, so they <u>are</u> good restaurants.

4] Consider a formal definition of impurity measures, and in extension consider binary entropy and binary Gini index.

a) Plot of Gini index and binary entropy function for $q(v)$ from 0 to 1.
   ↳ Plot for $q(v)$ at 5 points in its range, $0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1$.





Similarities: Both graphs have zeros at $q(v)=0$, $q(v)=1$.
Both graphs are concave down (upside down parabolas).
Both graphs have maximum at $q(v) = 1/2$.

b) Prove: if $i(q(v))$ is concave in $q(v)$, then $I(V_1, V_2, V) \geq 0 \quad \forall V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$.
This means every split does not lose any info.

Suppose $i(q(v))$ is concave in $q(v)$.

Then, $\forall \lambda \in [0,1]$ & $\forall q_1, q_2$, $i(\lambda q_1 + (1-\lambda) q_2) \geq \lambda i(q_1) + (1-\lambda) i(q_2)$.

*Claim: $q(V) = \frac{|V_1|}{|V|} q(V_1) + \frac{|V_2|}{|V|} q(V_2)$.

Let $\lambda = \frac{|V_1|}{|V|}$, and note that $0 \leq \lambda \leq 1$ because $V_1 \subseteq V$.
Also, because $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$, $\frac{|V_2|}{|V|} = 1 - \lambda$.

Then, $i(q(V)) = i(\lambda q(V_1) + (1-\lambda) q(V_2))$
$\geq \lambda i(q(V_1)) + (1-\lambda) i(q(V_2))$ by concavity.

$\Rightarrow i(\lambda q(V_1) + (1-\lambda) q(V_2)) - [\lambda i(q(V_1)) + (1-\lambda) i(q(V_2))] \geq 0$

→

Subbing in for λ and $1-\lambda$, and recalling that $q(V) = \lambda q(V_1) + (1-\lambda) q(V_2)$,

$$i(q(V)) - \left[\frac{|V_1|}{|V|} i(q(V_1)) + \frac{|V_2|}{|V|} i(q(V_2)) \right] \geq 0.$$

By definition, $p(V_1, V) = \frac{|V_1|}{|V|}$ and $p(V_2, V) = \frac{|V_2|}{|V|}$.

Then, $i(q(V)) - [p(V_1, V) i(q(V_1)) + p(V_2, V) i(q(V_2))]$

$$= I(V_1, V_2, V)$$

$$\geq 0$$

$$\Rightarrow I(V_1, V_2, V) \geq 0, \text{ as desired.}$$

□

* Proof of claim, $q(V) = \frac{|V_1|}{|V|} q(V_1) + \frac{|V_2|}{|V|} q(V_2)$.

$$q(V) = \frac{|\{i : i \in V, y_i = 1\}|}{|V|}$$

$$= \frac{|\{i : i \in V_1, y_i = 1\}| + |\{i : i \in V_2, y_i = 1\}|}{|V|} \quad \text{because } V_1 \cup V_2 = V, V_1 \cap V_2 = \emptyset.$$

$$= \frac{|V_1| |\{i : i \in V_1, y_i = 1\}|}{|V_1| |V|} + \frac{|V_2| |\{i : i \in V_2, y_i = 1\}|}{|V_2| |V|}$$

$$= \frac{|V_1|}{|V|} q(V_1) + \frac{|V_2|}{|V|} q(V_2) \text{ by definition.}$$

□

c) Show that binary entropy is concave.

↳ $\frac{d}{dq(v)} H(q(v)) = \frac{-d}{dq(v)} \left(q(v)\log q(v) + (1-q(v))\log(1-q(v)) \right)$

$= -\left(q(v)\frac{1}{q(v)} + \log q(v) + (1-q(v))\cdot\frac{-1}{(1-q(v))} - \log(1-q(v)) \right)$

$= -\left(\log q(v) - \log(1-q(v)) - 1 \right)$

$= -\log q(v) + \log(1-q(v)) + 1 \quad \forall\, q(v)$

$\frac{d^2}{dq(v)^2} H(q(v)) = \frac{d}{dq(v)} \left[-\log q(v) + \log(1-q(v)) + 1 \right]$

$= \frac{-1}{q(v)} - \frac{1}{1-q(v)} = -\left(\frac{1}{q(v)} + \frac{1}{1-q(v)} \right) \quad \forall\, q(v)$

Noting that $1 \geq q(v) \geq 0$, it follows that $\frac{1}{q(v)} + \frac{1}{1-q(v)} \geq 0$.

Then, $\frac{d^2}{dq(v)^2} H(q(v)) = -\left(\frac{1}{q(v)} + \frac{1}{1-q(v)} \right) \leq 0$

$\Rightarrow \frac{d^2}{dq(v)^2} H(q(v)) \leq 0 \quad \forall\, q(v)$

$\Rightarrow$ Binary Entropy $H(q(v))$ is concave.

□

d) Show that binary Gini index is concave.

↳ $\frac{\partial}{\partial q(v)} gini(q(v)) = \frac{\partial}{\partial q(v)} 2q(v)(1-q(v))$

$= 2q(v)(-1) + 2(1-q(v))$

$= -2q(v) + 2 - 2q(v)$

$= -4q(v) + 2$

$\frac{\partial^2}{\partial q(v)^2} gini(q(v)) = \frac{\partial}{\partial q(v)}(-4q(v) + 2)$

$= -4$

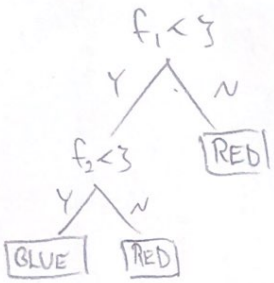
Noting that $-4 < 0$, it follows that $\frac{\partial^2}{\partial q(v)^2} gini(q(v)) < 0$

$\Rightarrow$ binary gini index is concave. □

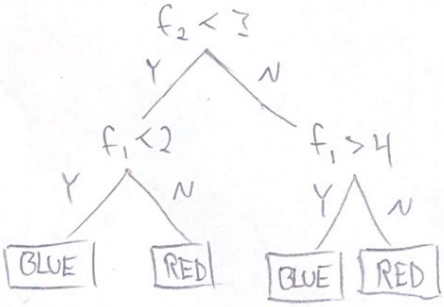
5] Use binary split decision tree to classify points. 2 attributes, f_1 and f_2. Ask bool questions comparing f_1, f_2 to real numbers.

a) Examples 1 and 3 can be fully separated with a depth 2 decision tree:

example 1



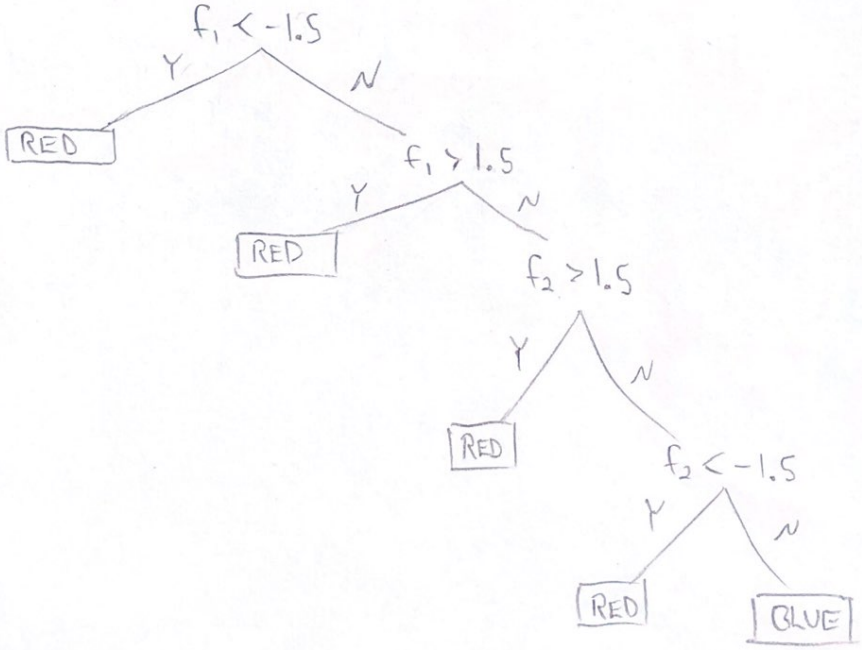
example 3



The other examples cannot be fully separated with a depth 2 decision tree.

b) Example 2 would have the most complex decision tree in terms of number of splits. Intuitively, this is because example 2 has a non-axis aligned boundary. In practice, this is because examples 1 and 3 can be represented by depth 2 trees, example 4 can be represented by a depth 4 tree, yet example 2 requires a greater-depth decision tree.

c) Using a depth 4 decision tree, example 4 becomes separable, as follows:



Zack Berger

Question 6

In order to obtain values for accuracy, I did the following:

1. Ran each dataset through the classifier
2. Counted the number of correctly labeled data points when compared to the true labels for each data set
3. Divided the number of correct labels by the total number of labels

My code is attached to the end of the pdf of this homework. My results are summarized in part A and part B…

Part A
I found the following using a baseline accuracy test:

> Accuracy of baseline classifier on training set: 0.594366197183
> Accuracy of baseline classifier on testing set: 0.694915254237

Part B
I used Python to complete this part of the assignment. I found the following:

> Accuracy of trained decision tree on training set: 0.981690140845

The implementation in sklearn.tree.DecisionTreeClassifier uses sampling with replacement, leading to a level of randomization and nondeterminism in the algorithm. As a consequence, I got different values for the accuracy of the trained decision tree on the testing set.

These accuracy values included:

> 0.779661016949
> 0.785310734463
> 0.796610169492
> 0.802259887006
> 0.80790960452

```
#!/usr/bin/env python
"""Analysis of the titanic dataset using a decision tree classifier"""

import numpy as np
from sklearn import tree

def main():

    # Import all training and testing data
    training_dataX = np.loadtxt("/Users/zackberger/Desktop/MI/HW/HW_3/dataTraining_X.csv", delimiter=',')
    training_dataY = np.loadtxt("/Users/zackberger/Desktop/MI/HW/HW_3/dataTraining_Y.csv", delimiter=',')
    testing_dataX = np.loadtxt("/Users/zackberger/Desktop/MI/HW/HW_3/dataTesting_X.csv", delimiter=',')
    testing_dataY = np.loadtxt("/Users/zackberger/Desktop/MI/HW/HW_3/dataTesting_Y.csv", delimiter=',')

    # Basline classification
    baseline_label = baseline_classification(training_dataY)

    correct_training_predictions = 0
    for i in range (0, len(training_dataY)):
        if training_dataY[i] == baseline_label:
            correct_training_predictions += 1

    correct_testing_predictions = 0
    for i in range (0, len(testing_dataY)):
        if testing_dataY[i] == baseline_label:
            correct_testing_predictions += 1

    # Compute training and testing accuracy
    accuracy_training = float(correct_training_predictions) / len(training_dataY)
    accuracy_testing = float(correct_testing_predictions) / len(testing_dataY)

    print("Accuracy of baseline classifier on training set: " + str(accuracy_training))
    print("Accuracy of baseline classifier on testing set: " + str(accuracy_testing) + "\n")

    # Train a scikit decision tree
    clf = tree.DecisionTreeClassifier(criterion='entropy')
    clf = clf.fit(training_dataX, training_dataY)

    # Test the decision tree
    train_predictions = clf.predict(training_dataX)
    correct_training_predictions = count_matches(train_predictions, training_dataY)
    accuracy_training = float(correct_training_predictions) / len(train_predictions)

    test_predictions = clf.predict(testing_dataX)
    correct_testing_predictions = count_matches(test_predictions, testing_dataY)
    accuracy_testing = float(correct_testing_predictions) / len(test_predictions)

    print("Accuracy of trained decision tree on training set: " + str(accuracy_training))
    print("Accuracy of trained decision tree on testing set: " + str(accuracy_testing))
```

```python
# Count the number of matches between two arrays of equal length
def count_matches(arr1, arr2):

    matches = 0
    for i in range (0, len(arr1)):
        if arr1[i] == arr2[i]:
            matches += 1

    return matches



# Obtain baseline classification result by majority count
# Return the value (1 or 0) that appears the most in the array
def baseline_classification(arr):

    one_count = 0
    zero_count = 0

    # Obtain how many zeros vs. ones there are in the set
    for i in range (0,len(arr)):
        if arr[i] == 0:
            zero_count += 1
        else:
            one_count += 1

    # Return the majority
    if one_count > zero_count:
        majority = 1
    else:
        majority = 0

    return majority



# Name guard
if __name__ == "__main__":
    main()
```