hw6

November 14, 2024

1 Homework 6 - Ishaan Sathaye

1.1 Section B: Computer Implementation

[16]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
```

Mapped Indica as 0 and Sativa as 1:

[30]:
```
cannabis = pd.read_csv('../hw3/cannabis_full.csv')
cannabis_hybrid = cannabis[cannabis['Type'] == 'hybrid']

# remove hybrid strains
cannabis = cannabis[cannabis['Type'] != 'hybrid']

# remove effects and flavors
cannabis = cannabis.drop(columns=['Effects', 'Flavor'])

# map type to 0 or 1
cannabis['Type'] = cannabis['Type'].map({'indica': 0, 'sativa': 1})

# drop null values
cannabis = cannabis.dropna()

cannabis.head()
```

[30]:

	Strain	Type	Rating	Creative	Energetic	Tingly	Euphoric	Relaxed	\
2	1024	1	4.4	1.0	1.0	0.0	0.0	1.0	
5	3-Bears-Og	0	0.0	0.0	0.0	0.0	0.0	0.0	
7	303-Og	0	4.2	0.0	0.0	0.0	1.0	1.0	
8	3D-Cbd	1	4.6	0.0	0.0	0.0	0.0	1.0	
9	3X-Crazy	0	4.4	0.0	0.0	1.0	1.0	1.0	

	Aroused	Happy	…	Ammonia	Minty	Tree	Fruit	Butter	Pineapple	Tar	\
2	0.0	1.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
5	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
7	0.0	1.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

```
8      0.0    1.0  …      0.0    0.0   0.0    0.0     0.0         0.0  0.0
9      0.0    1.0  …      0.0    0.0   0.0    0.0     0.0         0.0  0.0

   Rose  Plum  Pear
2   0.0   0.0   0.0
5   0.0   0.0   0.0
7   0.0   0.0   0.0
8   0.0   0.0   0.0
9   0.0   0.0   0.0

[5 rows x 67 columns]
```

1.1.1 Step 1: Regression

[18]:
```
effect_cols = cannabis.columns[cannabis.columns.get_loc("Creative"):cannabis.
 ↪columns.get_loc("Mouth")+1]
flavors_cols = cannabis.columns[cannabis.columns.get_loc("Earthy"):cannabis.
 ↪columns.get_loc("Pear")+1]

# Logistic Regression using only effect predictors
m1 = LinearRegression()
m1.fit(cannabis[effect_cols], cannabis['Type'])

# Logistic Regression using only flavor predictors
m2 = LinearRegression()
m2.fit(cannabis[flavors_cols], cannabis['Type'])

# Logistic Regression using only rating predictor
m3 = LinearRegression()
m3.fit(cannabis[['Rating']], cannabis['Type'])
```

[18]:
```
LinearRegression()
```

1.1.2 Step 2: Weight Sets

[29]:
```
w1 = np.concatenate([m1.coef_, np.zeros(len(flavors_cols) + 1)])
w2 = np.concatenate([np.zeros(len(effect_cols)), m2.coef_, np.zeros(1)])
w3 = np.concatenate([np.zeros(len(effect_cols) + len(flavors_cols)), m3.coef_])

c1 = m1.intercept_
c2 = m2.intercept_
c3 = m3.intercept_
```

1.1.3 Step 3: Perceptron Function

[44]:
```
# activation function: sigmoid
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def perceptron(preds, w1, w2, w3, c1, c2, c3):
    preds = np.array(preds)

    z1 = np.dot(preds, w1) + c1
    z2 = np.dot(preds, w2) + c2
    z3 = np.dot(preds, w3) + c3

    u1 = sigmoid(z1)
    u2 = sigmoid(z2)
    u3 = sigmoid(z3)

    u = np.array([u1, u2, u3])
    v = np.array([1/3, 1/3, 1/3])
    d = 0

    hidden_layer = np.dot(u, v) + d
    output = sigmoid(hidden_layer)

    return output
```

1.1.4 Step 4: Prediction

[60]:
```
cannabis_hybrid_preds = cannabis_hybrid[effect_cols.tolist() + flavors_cols.
 ↪tolist() + ['Rating']]
preds = cannabis_hybrid_preds.apply(lambda x: perceptron(x, w1, w2, w3, c1, c2,␣
 ↪c3), axis=1)

cannabis_hybrid['Predicted Sativa'] = preds
cannabis_hybrid[['Strain', 'Predicted Sativa']].head()
```

[60]:

	Strain	Predicted Sativa
0	100-Og	0.652103
1	98-White-Widow	0.656424
3	13-Dawgs	0.652649
4	24K-Gold	0.651774
6	3-Kings	0.641211

- The first strain 100 OG is balanced between indica and sativa, and the perceptron model predicted around 65% sativa, which means it is more sativa dominant.
- The second strain 98 White Widow is 70% sativa nd 30% indica, and the perceptron model predicted around 65% sativa, which is pretty close to the actual value.

- 13-Dawgs is also a balance between indica and sativa, and the perceptron model predicted around 65% sativa, which means it is more sativa dominant.
- 24K Gold is 60% indica and 40% sativa, and the perceptron model predicted around 65% sativa, which means here it predicted more sativa than the actual value.
- 3 Kings is 60% sativa and 40% indica, and the perceptron model predicted around 65% sativa, which means it was relatively close to the actual value.
- The perceptron model overall is not very accurate in predicting the strain type, but it was good on the 2nd and 5th strain.

1.1.5 Step 5: The Hidden Layer

[58]:
```
def perceptron_weighted(preds, w1, w2, w3, c1, c2, c3):
    preds = np.array(preds)

    z1 = np.dot(preds, w1) + c1
    z2 = np.dot(preds, w2) + c2
    z3 = np.dot(preds, w3) + c3

    u1 = sigmoid(z1)
    u2 = sigmoid(z2)
    u3 = sigmoid(z3)

    u = np.array([u1, u2, u3])
    v = np.array([0.3, 0.4, 0.3])
    d = -0.55

    hidden_layer = np.dot(u, v) + d
    output = sigmoid(hidden_layer)

    return output
```

[59]:
```
weighted_preds = cannabis_hybrid_preds.apply(lambda x: perceptron_weighted(x,␣
 ↪w1, w2, w3, c1, c2, c3), axis=1)

cannabis_hybrid['Predicted Sativa with Weights'] = weighted_preds
cannabis_hybrid[['Strain', 'Predicted Sativa', 'Predicted Sativa with␣
 ↪Weights']].head()
```

[59]:

	Strain	Predicted Sativa	Predicted Sativa with Weights
0	100-Og	0.652103	0.519938
1	98-White-Widow	0.656424	0.524961
3	13-Dawgs	0.652649	0.522664
4	24K-Gold	0.651774	0.520088
6	3-Kings	0.641211	0.507320

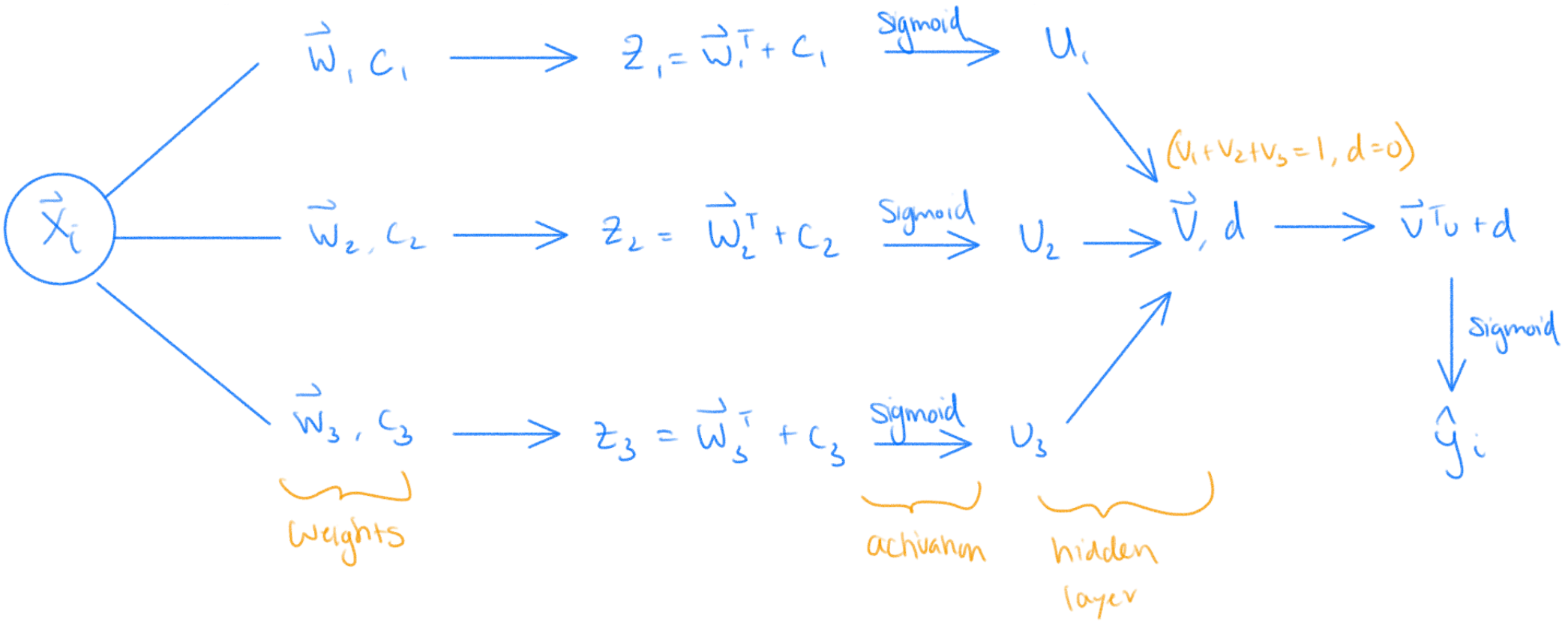
- Instead of doing a majority vote, to determine the final output, I multiplied all the net inputs by a weight which sum to 1. This way, the output is a weighted average of the net inputs, which is a more accurate representation of the final output. I choose this custom set of weights

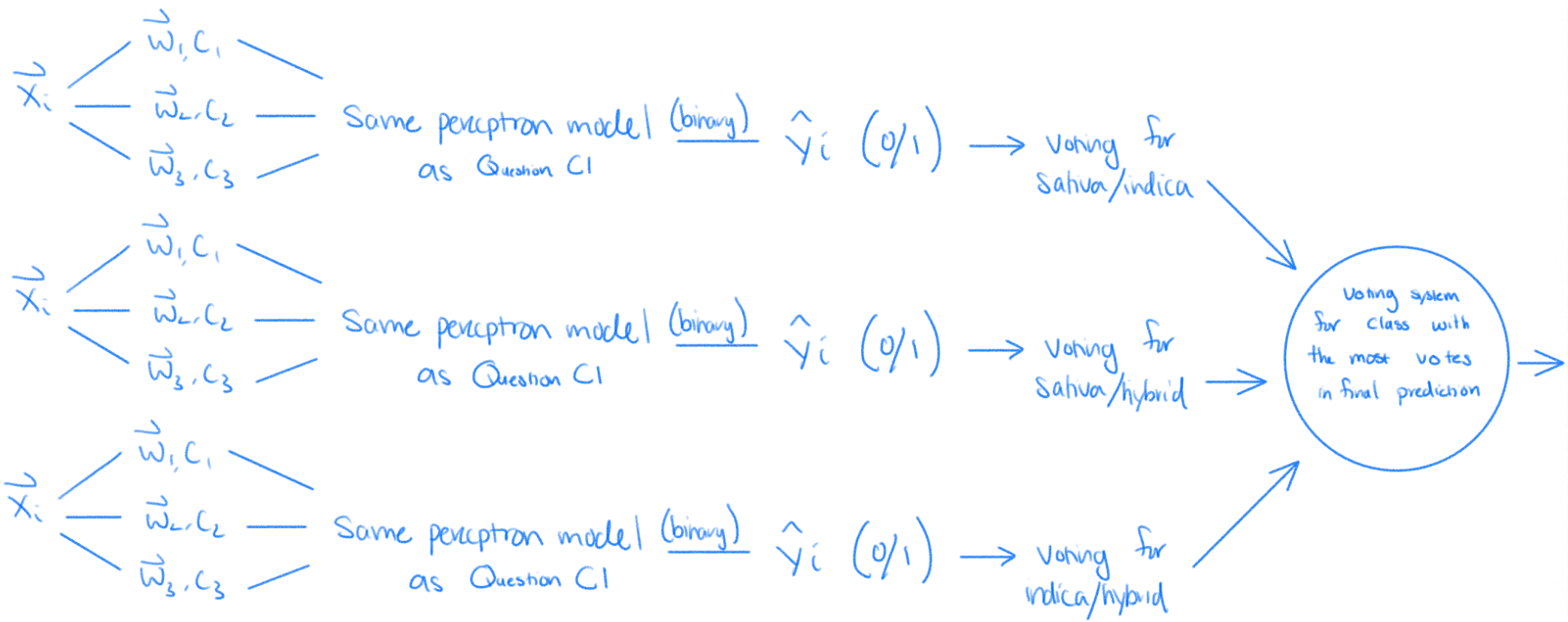
as it performed better with a bias value of -0.5.

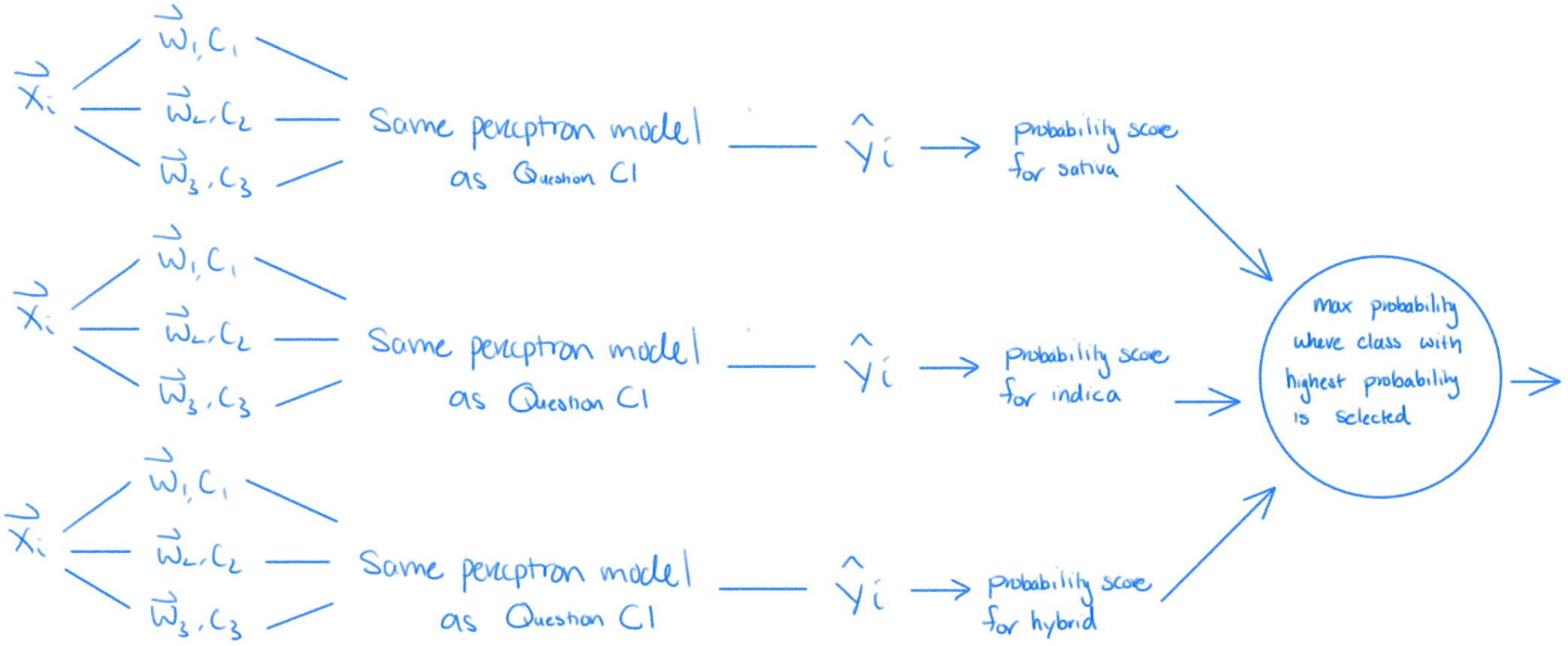
- In this perceptron model, the performance was much better since it got better at predicting the the first and third strain. For the second strain it got worse overall since it got further from the actual value, but it was still on the correct side. The 4th strain prediction was much closer than the previous model, and the 5th strain prediction was also much closer to the actual value.

1.2 Section C: Concepts

1. Draw the configuration of the perceptron we built in Section B.
   - Drawing 1 is the first image below.
2. In the setup of Section B, we "cheated" on the first step: We chose our weight sets ahead of time instead of **fitting** a perceptron. Suppose instead, we used the typical unpenalized Logistic Regression loss function as our loss for the perceptron. What would you expect to get for w1, w2, and w3, in this case? Why?
   - If instead of cheating by choosing the weight sets ahead of time and using the typical unpenalized Logistic Regression loss function as our loss for the perceptron, the weights would be different from the weights from fitting individually. This solely due to the fact that we would be using a model that includes all the predictors with the same loss function. The process of fitting would adjust the weights to minimize the error in classification. So each weight set would not be fitted independently, but rather together so that the final weights would be the combined effect of all the predictors.
3. Suppose we wanted to train a perceptron for all three categories: Sativa, Indica, and Hybrid. Draw a perceptron configuration for (a) a one-vs-one approach and (b) a one-vs-all approach. Make sure to be specific about what your **final output(s)** are and what your **loss function** is for each of the two configurations.
   - For the one-vs-one approach, I would have a separate binary classifier for each pair of classes. There would be 3 classifiers then which would be sativa vs indica, sativa vs hybrid, and indica vs hybrid. The final output would be the majority vote of the 3 classifiers. For the loss function, I chose to go with a binary cross-entropy loss function, which can distinguish between the two classes and measures how well each perceptron separates each of its 2 classes
     - Drawing 2 is the second image below.
   - For the one-vs-all approach, I would have separate classifiers for each class and in total there would be 3. These 3 would be sativa vs not sativa, indica vs not indica, and hybrid vs not hybrid. The final output would be the class with the highest probability. For the loss function, I chose to go with a categorical cross-entropy loss function since there will be probabilities for each class and multiple classes.
     - Drawing 3 is the third image below.


$\vec{x}_i$
$\vec{w}_1, c_1$
$z_1 = \vec{w}_1^T + c_1$
Sigmoid
u_1
$\vec{w}_2, c_2$
$z_2 = \vec{w}_2^T + c_2$
Sigmoid
u_2
$\vec{w}_3, c_3$
$z_3 = \vec{w}_3^T + c_3$
Sigmoid
u_3
$(v_1 + v_2 + v_3 = 1, d = 0)$
$\vec{v}, d$
$\vec{v}^T u + d$
Sigmoid
$\hat{y}_i$
Weights
activation
hidden layer


$\vec{x}_i$
$\vec{w}_1, c_1$
$\vec{w}_2, c_2$
$\vec{w}_3, c_3$
Same perceptron model (binary) as Question C1
$\hat{y}_i$ (0/1)
Voting for sativa/indica
$\vec{x}_i$
$\vec{w}_1, c_1$
$\vec{w}_2, c_2$
$\vec{w}_3, c_3$
Same perceptron model (binary) as Question C1
$\hat{y}_i$ (0/1)
Voting for sativa/hybrid
$\vec{x}_i$
$\vec{w}_1, c_1$
$\vec{w}_2, c_2$
$\vec{w}_3, c_3$
Same perceptron model (binary) as Question C1
$\hat{y}_i$ (0/1)
Voting for indica/hybrid
Voting system for class with the most votes in final prediction


$\vec{x_i}$
$\vec{w_1}, c_1$
$\vec{w_2}, c_2$
$\vec{w_3}, c_3$
Same perceptron model as Question C1
$\hat{y_i}$
probability score for sativa
$\vec{x_i}$
$\vec{w_1}, c_1$
$\vec{w_2}, c_2$
$\vec{w_3}, c_3$
Same perceptron model as Question C1
$\hat{y_i}$
probability score for indica
$\vec{x_i}$
$\vec{w_1}, c_1$
$\vec{w_2}, c_2$
$\vec{w_3}, c_3$
Same perceptron model as Question C1
$\hat{y_i}$
probability score for hybrid
max probability where class with highest probability is selected