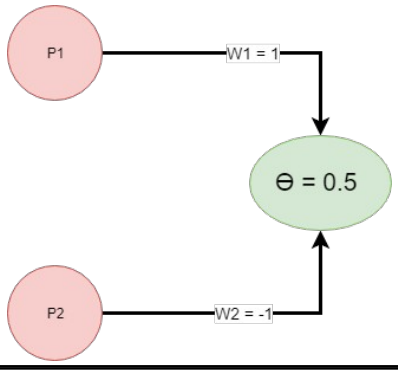
Problem Set 1

smatthews39@gatech.edu

Sean Matthews

2a) Design a two-input perceptron that implements the boolean function $A \wedge \neg B$

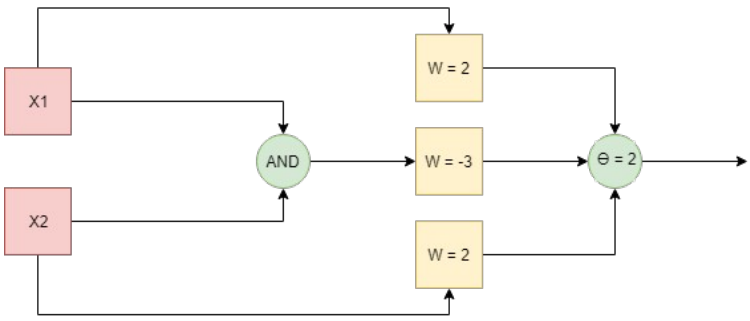


2b) Design a two-layer network of perceptrons that implements $A \oplus B$ ($\oplus$ is XOR).

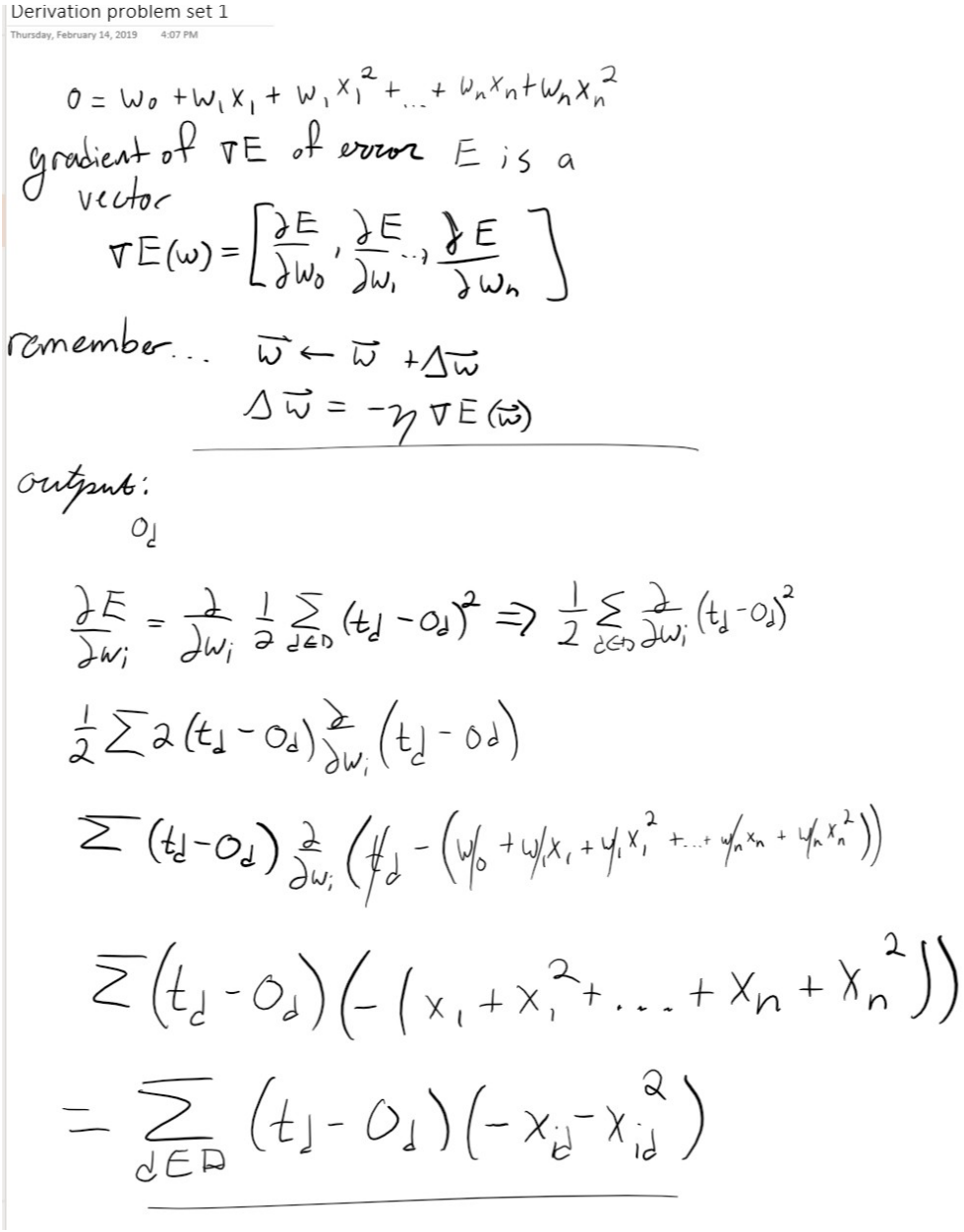
A $\oplus$ B - A XOR B

$A\ XOR\ B = (A \wedge \neg B) \vee (B \wedge \neg A)$

X1	X2	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0



3)



4) A decision tree can be used for regression by replacing information gain with standard deviation reduction. Standard deviation is used for creating the tree, and the other parameters are Coefficient of Deviation which is used to tell the model when to stop branching, and Average, which is the value within the leaf nodes.

Standart deviation reduction is based on the decrease in standard deviation after a dataset is split on an attribute. The construction of a decision tree is based on finding the attribute that has the most alike branches. This would signify that it returns the highest standard deviation reduction.

The process of using Decision trees for reduction are below:

1. Calculate the Standard Deviation of the target
2. Split the data on different attributes and calculate the STD for each branch. Subtract that STD from the STD calculated before the split and the result is the STD reduction
3. The LARGEST STD reduction is the chosen decision node
4. Split the data by the selected attribute, and recursively run the data for the next level on the decision tree

On each leaf, a squared error function is equal to using the mean of the class for each instance, therefore, the mean of Y is used as the expected value at each leaf:

$$\sum \left(Y_i - h_i\right)^2 = \sum \left(mean\left(Y\right) - h_i\right)^2$$

```python
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error


data = pd.read_csv("housing.csv")
X = data.iloc[:,0:13]
y = data.iloc[:,13]
training_size= 60

print("Training Model and testing with: ", 100-training_size, "%")
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=100-training_size)
regr_1 = DecisionTreeRegressor(random_state=0, max_depth=4)
regr_1.fit(X_train, y_train)
y_1 = regr_1.predict(X_test)
print("Error with Max Depth 4: ", mean_squared_error(y_1, y_test))

regr_2 = DecisionTreeRegressor(random_state=0, max_depth=5)
regr_2.fit(X_train, y_train)
y_2 = regr_2.predict(X_test)
print("Error with Max Depth 5: ", mean_squared_error(y_2, y_test))

regr_3 = DecisionTreeRegressor(random_state=0, max_depth=6)
regr_3.fit(X_train, y_train)
y_3 = regr_3.predict(X_test)
print("Error with Max Depth 6: ", mean_squared_error(y_3, y_test))
```

```
C:\Users\Sean\PycharmProjects\DTRegression\venv\Scripts\python.exe C:/Users/Sean/PycharmProjects/DTRegression/housing_data.py
Training Model and testing with:  40 %
Error with Max Depth 4:  7.326168811753211
Error with Max Depth 5:  7.234786420281537
Error with Max Depth 6:  7.617227360554128

Process finished with exit code 0
```

5)

ID3 (training examples, class attributes, attributes):

Loop:

- A ← best attribute
- Assign A as decision attribute for node
- For Each Value of A create a descendant of node
- Sort training examples to leaves
- If examples perfectly classified, STOP
- Else iterate over leaves

Lazy DT (unclassified instance, training examples)

- Select a test, X (the one that maximizes the decreases in entropy)
- Let x be the value of the test on the unclassified instance
- Assign the set of instances with X = x to training examples
- Iterate (unclassified instance, training examples)

This Lazy DT takes advantage of the additional information given by a single instance and fixes some problems that occur in the regular DT algorithms such as replication and fragmentation. Fragmentation is where subsequent tests on some attributes have lower significance due being based on fewer instances. Lazy decision trees select better tests with respect to a particular instance and it avoids splitting on unnecessary attributes which produce a more accurate classifier and shorter paths. Another advantage of lazy DTs is they never branch on a missing value in the instance

Lazy learning requires significantly less time during training then eager methods, but they require more time to classify each query instance. They also require large space requirements to store the entire training dataset.

6) I would use kNN. Decision trees would have a hard time determining the classification for those which are on the line y=x. They are both nonlinear classifiers, but I think kNN, would be able to determine the classification

pretty well whether it was on the (+) side or the (-), as well as on the line y=x. This is true for a dense amount of data as well as few samples.

7)

1. An origin-centered circle (2D)

   VC(H) = 3

   Figure 7.3 in the book represents a 2D space, and shows 2^3 hypothesis shattering the 3 points. For each of the dichotomies, there must exist a hypothesis that covers it. In the case where the 3 points are colinear, there isn't a way to find 2^3 linear surfaces that shatter, however, the VC dimension definition states if there are *ANY* set of instances of size *d* that can be shattered, then VC(H) ≥ d

   However after rereading the questions… I'm picturing that the circle cannot be moved, it can just be grown (considering it is origin-centered). If the circle can't be moved, then I believe it would be VC(H)=2, I may not be understanding the questions then.

2. I would also say for a sphere it is VC(H) = 3

   I can't come up with a solution where there are 4 points in 3D space that can be shattered. Therefore it is still the same as a 2D space of VC(H)=3