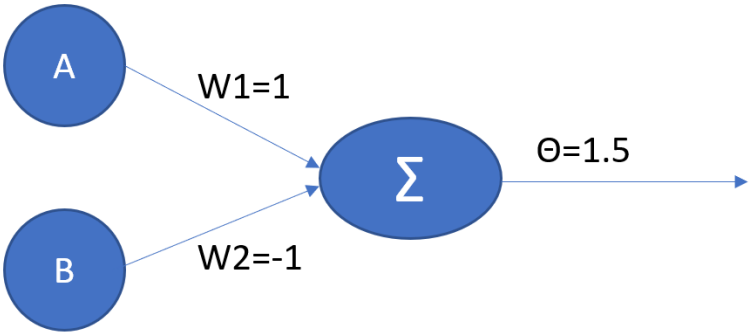
Problem set 1

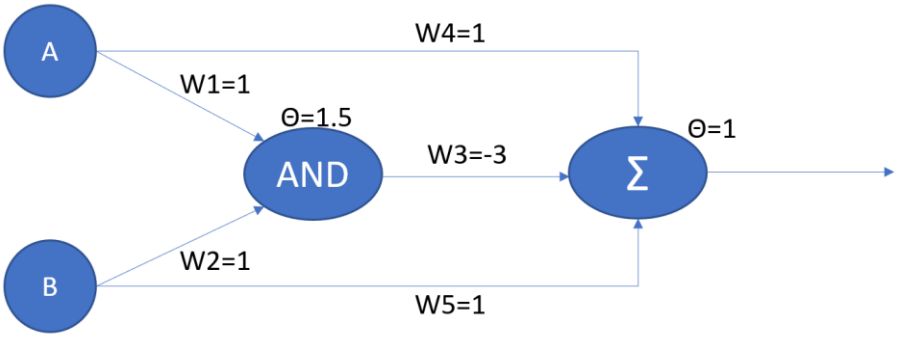
Jun Wang

903750524

Problem 2.



A AND NOT B:



A XOR B:

Problem 4.

To use decision tree for regression, the tree structure can be generated in the same way as DT classifier. But instead of assigning a single class to each leaf, a regression tree can have unique prediction value for each leaf. For each leaf, a squared error function equals to using the mean of class as the class for every instance. That is $\sum(Y_i - h_i)^2 = \sum(mean(Y) - h_i)^2$. So, we can use the mean of Y as expected values at each leaf. Using the Boston housing data, a tree of depth 5 gave the smallest mean squared error.

```
import numpy as np
import pandas as pd
import graphviz
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import mean_squared_error

data = pd.read_csv("Desktop/train.csv")
X = data.iloc[:,1:14]
y = data.iloc[:,14]
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0,
test_size=0.25)

regr_1 = DecisionTreeRegressor(random_state=0, max_depth=4)
regr_1.fit(X_train, y_train)
y_1 = regr_1.predict(X_test)
print(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(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(mean_squared_error(y_3, y_test))
```

Output:

```
10.4497593528
9.11738246771
9.66406200934
```

Problem 5.

A lazy decision tree algorithm works like nearest neighbor method. As it didn't build a whole tree of all classes, but instead, it build a route in the tree that leads to one specific instance. So for each instance I in the test data, we use all possible split functions that pass certain information gain threshold, and then take all training instances with the same split value as I, and then repeat until all remaining training instances have the same class. The final leaf with most training instances will be the prediction. The advantages of such lazy function are faster classification due to no training process, and it handles missing values easily. The disadvantages include higher memory usages because of heavy recursive calls of all possible splits.

Problem 6.

I would choose decision tree. Because KNN will perform poorly for testing points near the actual line. As for decision tree, if I add multiple features created by apply a linear combination of X1 and X2, then after learning a decision with depth 1 for multiple times. I can narrow down the linear combination to basically the same as the actual line. Then this selected linear combination can serve as a very good classifier.