Weekly Report
02/07/2015

Jingyu Deng

This week

- Reviewed/learned Python
- Learned Sk-learn
- Learned Numpy
- Read the source code of ensemble and tree packages in Sk-learn
- Implement a basic weighted random forest module

Weighted Random Forest

- Each tree in the forest has a weight
- The result is the sum of weighted decision
- Support updating forest by using new coming data chunk
    - Adjust the weight of the tree according to the result
    - Discard some trees according to weights

Main Functions

- __init__(n_estimators)
    - generate a forest containing ***n_estimators*** trees.
- fit(X, y)
    - ***X*** is the data set. ***y*** is the target set. Using ***X*** and ***y*** to train the forest.
- predict(X)
    - predict the target values for input data ***X.***
- score(X, y)
    - Returns the coefficient of determination R^2 of the prediction.
- update(X, y)
    - for new coming data point set (***X***, ***y***), update the forest.

Update(X, y)

```
Pseudocode:
    sum = 0
    for each tree in forest
      score = tree.score(X,y)
      //returns the coefficient of determination R^2 of the prediction
      tree.weight *= score
      sum += tree.weight
    for each tree in forest
      tree.weight /= sum
      //adjust weight
      if tree.weight < 1 / n_estimators * 0.5
    tree = new tree(last_chunk_X, last_chunk_y)
    //chunk_size is a fixed number
    //last_chunk_X means the last chunk_size coming input data
    //last_chunk_y means the last chunk_size coming target values
```

Questions

- Is there any stream data for testing?

- Should the number of trees be fixed?

Next Step

- This is a naive implementation. I need to find the appropriate strategy to adjust the forest.