**Introduction to Machine Learning (Spring 2019)**

**Homework #3 (50 Pts, May 22)**

**Student ID 2014313303**

**Name 홍태하**

**Instruction:** We provide all codes and datasets in Python. Please write your code to complete the Evaluation metric. Compress ‘Answer.py and submit with the filename ‘HW3\_STUDENT\_ID.zip’.

1. **[20 pts]** Implement four functions in ‘utils/ Answer.py’. (‘Accurcay,’ ‘Precision’, ‘Recall’, ‘F\_measure’,). You don’t need to implement the part of model(Logistic Regression) and optimizer(SGD).

**Answer: Fill your code here. You also have to submit your code to i-campus.**

[NOTE]: You should write your codes in ‘EDIT HERE’ signs. It is not recommended to edit other parts. Once you complete your implementation, you can test the each evaluation metric (Accurcay, Precision, Recall, F-measure) for the given conditions with ‘A\_check.py’. You must get the same result with the examples.

[NOTE]: Dataset: These Datasets(Contracept, Heart and Yeast) are used for binary classifiacation. Each dataset’s last column is the label(y), other part(x) are input\_features.

1. Show your codes for ‘utils/Answers.py’.

def Accuracy(label, pred):

########################################################################################

# TODO : Complete the code to calculate the accuracy for prediction.

# [Input]

# - label : (N, ), Correct label with 0 (negative) or 1 (positive)

# - hypo : (N, ), Predicted score between 0 and 1

# [output]

# - Acc : (scalar, float), Computed accuracy score

# ========================= EDIT HERE =========================

Acc = None

true = 0

for i in range(len(pred)):

if pred[i] > 0.5:

if label[i] == 1:

true += 1

else:

if label[i] == 0:

true += 1

Acc = true/len(pred)

# =============================================================

return Acc

def Precision(label, pred):

########################################################################################

# TODO : Complete the code to calculate the Precision for prediction.

# you should consider that label = 1 is positive. 0 is negative

# Notice that, if you encounter the divide zero, return 1

# [Input]

# - label : (N, ), Correct label with 0 (negative) or 1 (positive)

# - hypo : (N, ), Predicted score between 0 and 1

# [output]

# - precision : (scalar, float), Computed precision score

# ========================= EDIT HERE =========================

precision = None

TP = 0

FP = 0

for i in range(len(pred)):

if pred[i] > 0.5:

if label[i] == 1:

TP += 1

else:

FP += 1

if TP + FP == 0:

precision = 1

else:

precision = TP/(TP+FP)

# =============================================================

return precision

def Recall(label, pred):

########################################################################################

# TODO : Complete the code to calculate the Recall for prediction.

# you should consider that label = 1 is positive. 0 is negative

# Notice that, if you encounter the divide zero, return 1

# [Input]

# - label : (N, ), Correct label with 0 (negative) or 1 (positive)

# - hypo : (N, ), Predicted score between 0 and 1

# [output]

# - recall : (scalar, float), Computed recall score

# ========================= EDIT HERE =========================

recall = None

TP = 0

FN = 0

for i in range(len(pred)):

if label[i] == 1:

if pred[i] > 0.5:

TP += 1

else:

FN += 1

if TP + FN == 0:

recall = 1

else:

recall = TP/(TP+FN)

# =============================================================

return recall

def F\_measure(label, pred):

########################################################################################

# TODO : Complete the code to calculate the F-measure score for prediction.

# you can erase the code. (F\_score = 0.)

# Notice that, if you encounter the divide zero, return 1

# [Input]

# - label : (N, ), Correct label with 0 (negative) or 1 (positive)

# - hypo : (N, ), Predicted score between 0 and 1

# [output]

# - F\_score : (scalar, float), Computed F-score score

# ========================= EDIT HERE =========================

F\_score = None

recall = Recall(label, pred)

precision = Precision(label, pred)

if recall + precision == 0:

F\_score = 1

else:

F\_score = (2\*recall\*precision)/(recall+precision)

# =============================================================

return F\_score

1. In ‘A\_main.py’, you will deal with the 3 binary classification datasets, ‘Contracept’, ‘Heart’ and ‘Yeast’. Label 1 is the positive and 0 is negative. With given hyperparameters, obtain 4 measures (accuracy, precision, recall and F-measure) for 3 datasets and fill in the blank.

**Answer: Fill the blank in the table.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Contracept** | **Heart** | **Yeast** |
| Accuracy | 0.581633 | 0.766667 | 0.517857 |
| Precision | 0.640167 | 0.725000 | 0.333333 |
| Recall | 0.805263 | 0.906250 | 0.038462 |
| F-measure | 0.713287 | 0.805556 | 0.068966 |

|  |  |
| --- | --- |
| **Parameter Settings** | |
| Batch size | 32 |
| Learning rate | 0.01 |
| Optimizer | SGD |
| # of epochs | 100 |
| Numpy Random\_seed | 503 |

<A\_main.py parameter setting>

1. **[30 pts]** Implement two functions in ‘utils/Answer.py’. (‘MAP’ , ‘nDCG’). You don’t need to implement the part of model.

**Answer: Fill your code here. You also have to submit your code to i-campus.**

NOTE: You should write your codes in ‘EDIT HERE’ signs. It is not recommended to edit other parts. Once you complete your implementation, you can test the each evaluation metric (MAP, nDCG) for the given conditions with ‘B\_check.py’. You must get the same result with the examples.

1. Show your codes for ‘utils/Answers.py’. In ‘B\_check.py’ you can test the each evaluation metric(MAP, nDCG) for the given conditions. You must get the same result for examples. (When you get the MAP, nDCG, You should get the mean value.)

def MAP(label, hypo, at = 10):

########################################################################################

# TODO : Complete the code to calculate the MAP for prediction.

# Notice that, hypo is the real value array in (0, 1)

# MAP (at = 10) means MAP @10

# [Input]

# - label : (N, K), Correct label with 0 (incorrect) or 1 (correct)

# - hypo : (N, K), Predicted score between 0 and 1

# - at: (int), # of element to consider from the first. (TOP-@)

# [output]

# - Map : (scalar, float), Computed MAP score

# ========================= EDIT HERE =========================

Map = None

AP = []

for i in range(len(label)):

true\_label = 0

ap = 0

ind = np.argsort(hypo[i])

ind = ind[::-1]

for j, data in enumerate(ind):

if label[i][data] == 1:

true\_label += 1

if j < at:

ap += true\_label/(j+1)

AP.append(ap/true\_label)

Map = np.mean(AP)

# =============================================================

return Map

def nDCG(label, hypo, at = 10):

########################################################################################

# TODO : Complete the each code to calculate the nDCG for prediction.

# you can erase the code. (dcg, idcg, ndcg = 0.)

# Notice that, hypo is the real value array in (0, 1)

# nDCG (at = 10 ) means nDCG @10

# [Input]

# - label : (N, K), Correct label with 0 (incorrect) or 1 (correct)

# - hypo : (N, K), Predicted score between 0 and 1

# - at: (int), # of element to consider from the first. (TOP-@)

# [output]

# - Map : (scalar, float), Computed nDCG score

def DCG(label, hypo, at=10):

# ========================= EDIT HERE =========================

dcg = None

dcg = 0

ind = np.argsort(hypo)

ind = ind[::-1]

for i, data in enumerate(ind):

if label[data] == 1:

if i < at:

dcg += 1/(np.log2(i+2))

else:

break

# =============================================================

return dcg

def IDCG(label, hypo, at=10):

# ========================= EDIT HERE =========================

idcg = None

idcg = 0

true\_label = 0

for i in range(len(label)):

if label[i] == 1:

true\_label += 1

min\_val = min(true\_label, at)

for i in range(min\_val):

idcg += 1/(np.log2(i+2))

# =============================================================

return idcg

# ========================= EDIT HERE =========================

ndcg = None

ndcg = 0

for i in range(len(label)):

ndcg += DCG(label[i], hypo[i], at)/IDCG(label[i], hypo[i], at)

ndcg /= len(label)

# =============================================================

return ndcg

1. In ‘B\_main.py’, you will deal with the ‘Product.csv.’, ‘Movielens.csv’ dataset. But you don’t need to process this dataset. Prediction scores and correct labels are given. Test the datasets with MAP, nDCG at (25, 50, 75) and fill in the blanks.

|  |  |  |
| --- | --- | --- |
| **Dataset** | **Product** | **Movielens** |
| MAP @25 | 0.142387 | 0.337997 |
| nDCG @25 | 0.495483 | 0.829796 |
| MAP @50 | 0.269029 | 0.448933 |
| nDCG @50 | 0.505574 | 0.805522 |
| MAP @75 | 0.393560 | 0.510219 |
| nDCG @75 | 0.660047 | 0.806849 |