Introduction to Machine Learning (Spring 2019)

Homework #3 (50 Pts, May 22)

Student ID _____

Name _____

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).

(a) Show your codes for 'utils/Answers.py'.

```
def Accuracy(label, pred):
    # ======================== EDIT HERE ========================
    total = len(label)
    correct = len(np.where(label == pred)[0])
    Acc = correct / total
    # ===========================================================
    return Acc

def Precision(label, pred):
    # ======================== EDIT HERE ========================
    true_pred = np.where(pred == 1)[0]
    true_total = len(true_pred)

    if true_total == 0:
        precision = 1

    else:
        true_correct = 0

        for ind in true_pred:
            if label[ind] == 1:
                true_correct += 1

        precision = true_correct / true_total
    # ===========================================================
    return precision
```

```
def Recall(label, pred):

    # ======================== EDIT HERE ========================
    true_label = np.where(label == 1)[0]
    true_total = len(true_label)

    if true_total == 0:
        recall = 1

    else:
        true_correct = 0
        for ind in true_label:
            if pred[ind] == 1:
                true_correct += 1

        recall = true_correct / true_total
    # ===========================================================
    return recall

def F_measure(label, pred):

    # ======================== EDIT HERE ========================
    rec = Recall(label, pred)
    prec = Precision(label, pred)

    if rec + prec == 0:
        F_score = 1
    else:
        F_score = (2 * rec * prec) / (rec + prec)
    # ===========================================================
    return F_score
```

(b) 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>

(2) [30 pts] Implement two functions in 'utils/Answer.py'. ('MAP' , 'nDCG'). You don't need to implement the part of model.

(a) 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):
    query_num, query_len = label.shape

    rel_num = [0] * query_num
    ap = [0] * query_num

    for query in range(query_num):
        rel_num[query] = len(np.where(label[query] == 1)[0])

        pred = np.zeros_like(label[0], dtype=float)
        label_rearr = []

        sorted_ind = np.flip(np.asarray(hypo[query]).argsort())

        for ind in sorted_ind:
            label_rearr.append(label[query][ind])

        for ind in range(at):
            pred[ind] = 1

        for ind in range(at):
            if label_rearr[ind] == 1:
                ap[query] += Precision(label_rearr[:ind+1], pred[:ind+1])

        ap[query] /= rel_num[query]

    Map = np.average(ap)
    # ============================================================
    return Map
```

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

    def DCG(label, hypo, at=10):
        # ======================== EDIT HERE ========================
        dcg = 0

        sorted_ind = np.flip(np.asarray(hypo).argsort())
        label_rearr = []

        for ind in sorted_ind:
            label_rearr.append(label[ind])

        label_rearr = label_rearr[:at]

        for i in range(len(label_rearr)):
            if label_rearr[i] == 1:
                dcg += 1 / np.log2(i + 2)
        # ===========================================================
        return dcg

    def IDCG(label, hypo, at=10):
        # ======================== EDIT HERE ========================
        idcg = 0

        ideal_label = np.zeros_like(label)
        cnt = len(np.where(label == 1)[0])

        for i in range(cnt):
            ideal_label[i] += 1

        ideal_label = ideal_label[:at]

        for i in range(len(ideal_label)):
            if ideal_label[i] == 1:
                idcg += 1 / np.log2(i + 2)
        # ===========================================================
        return idcg
    # ======================== EDIT HERE ========================
    ndcg = 0
    query_num, query_len = label.shape

    for query in range(query_num):
        ndcg_query = DCG(label[query], hypo[query], at) / IDCG(label[query], hypo[query], at)
        ndcg += ndcg_query

    ndcg /= query_num
    # ===========================================================
    return ndcg
```

(b) 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