

in3050_2023_w6

February 15, 2023

1 IN3050/IN4050 2021: Week 06 Perceptron

1.0.1 Introduction

The goal of this week's exercise set is to get familiar with the perceptron classifier. The set has two parts. The first part does not involve programming or Python. The questions in this part are the type of questions who will meet at the exam. They can (and should) be solved by pen and paper or a word processor and, eventually, a calculator.

The second part involves implementations in Python.

1.1 Part 1: Exam type exercises

1.1.1 Exercise I

You want to predict if movies will be profitable based on their screenplays. You hire two critics A and B to read a script you have and rate it on a scale of 1 to 6. The critics are not perfect; here are five data points including the critics' scores and the performance of the movie:

Movie #	Alice	Bob	Profitable?
1	1	1	no
2	4	3	yes
3	3	5	yes
4	5	6	yes
5	2	3	no

Is the data linearly separable? Plot the data on a sheet of paper; label profitable movies with + and non-profitable movies with x and determine if the data are linearly separable.

1.1.2 Exercise II

You decide to use a perceptron to classify your data. Suppose you directly use the scores given above as features, x_1 = score given by A and x_2 = score given by B. You also add a bias term in position 0, $x_0 = -1$.

Run one pass through the data with the perceptron algorithm, filling out the table below. Go through the data points in order, e.g., using data from movie 1 at step 1. Suppose we start with the weights $(w_0, w_1, w_2) = (1, 0, 0)$. Use 1 as learning rate.

For each of the 5 steps, write the the score, and whether the prediction is correct. Calculate the new weights, enter them at the next line and repeat.

Step	Weights	Score	Correct prediction?
1	(1, 0, 0)	$1*(-1) + 0*1 + 0 * 1 = -1$	yes
2			
3			
4			
5			

1.1.3 Exercise III

Have weights been learned that separate the data? Plot the data together with the decision boundary given by the weights which you are left with after you have run all the data points the first round in the previous question.

1.1.4 Exercise IV

More generally, irrespective of the training data, you want to know if your features are powerful enough to allow you to handle a range of scenarios. Which of the following scenarios can a perceptron using the features above indeed perfectly classify movies which are profitable according to the given rules:

1. Your reviewers are awesome: if the total of their scores is more than 8, then the movie will definitely be profitable, and otherwise it won't be.
2. Your reviewers are art critics. Your movie will be profitable if and only if each reviewer gives either a score of 3 or a score of 4.
3. Your reviewers have weird but different tastes. Your movie will be profitable if and only if both reviewers agree.

1.2 Part 2: Implementing the Perceptron Classifier

1.2.1 Exercise V

We will implement the perceptron classifier. We will implement it as a class similarly to how we implemented the k NN-classifier last week. You should implement the two methods `fit` and `predict`. You don't need an `__init__` method. You may include other methods if you find them convenient.

The main part of the perceptron classifier is the `fit` method that trains the perceptron. `eta` is the learning rate. One `epoch` is a run through all the training data. We have set the default to one epoch, but you might need more. Remember to add the bias to the data.

```
[ ]: class PyClassifier():
    """Common methods to all python classifiers --- if any

    Nothing here yet"""

[ ]: class PyPerClassifier(PyClassifier):
    """Simple perceptron python classifier"""

    def fit(self, X_train, y_train, eta=1, epochs=1):
        """Train the self.weights on the training data with learning
        rate eta, running epochs many epochs"""

    def predict(self, X):
        """Predict the class for the items (in the list) X"""
```

1.3 Dataset

We will train and test the classifier on the same sythetic dataset as we used for the k NN last week.

```
[ ]: from sklearn.datasets import make_blobs
X_np, y_np = make_blobs(n_samples=200, centers=[[0,0],[1,2]],
                        n_features=2, random_state=2019)
X1 = [(X_np[i,0], X_np[i,1]) for i in range(X_np.shape[0])]
t1 = [y_np[i] for i in range(X_np.shape[0])]

[ ]: def show(X, y, marker='.'):
    labels = set(y)
    cl = {lab : [] for lab in labels}
    # cl[lab] shall contain the datapoints labeled lab
    for (a, b) in zip(X, y):
        cl[b].append(a)
    for lab in labels:
        plt.plot([a[0] for a in cl[lab]], [a[1] for a in cl[lab]],
                 marker, label="class {}".format(lab))
    plt.legend()

[ ]: show(X1, t1)

[ ]: X_np, y_np = make_blobs(n_samples=200, centers=[[0,0],[1,2]],
                        n_features=2, random_state=2020)
X2 = [(X_np[i,0], X_np[i,1]) for i in range(X_np.shape[0])]
t2 = [y_np[i] for i in range(X_np.shape[0])]
```

1.3.1 Exercise VI

Train PyPerClassifier on X1, t1 and test on X2, t2 as with the k NN-classifier. Try various numbers of epochs and see whether it makes a difference.

Is X2, t2 linearly separable?

How does the results compare to the k NN classifier from last week?

1.3.2 Exercise VII: Termination and Weights

- Modify the *fit*-method of the classifier such that training halts when there are no more updates, and it is possible to inspect how many epochs it took.
- Run this on the five points film critics dataset. How many epochs does it take to reach completion?
- What are the final weights? Can you plot the decision boundary that corresponds to these weights together with the data set?