Middle East Technical University Department of Computer Engineering

CENG 499

Introduction to Machine Learning

Fall 2021-2022

Homework 4: Hidden Markov Models (HMMs) and Naive Bayes

version 1.0

Due date: 5 February 2022, 23:59

1 Introduction

In this assignment, you will have a chance to get hands-on experience with Hidden Markov Models (HMMs) and Naive Bayes classifiers. No report is required for this homework. The related material can be found in **hw4_material.zip**.

2 Hidden Markov Models

In this section, you will work on the evaluation and decoding tasks of HMMs. Check the recitation slides for examples and the notation we are using for HMMs.

- For the evaluation task, implement the Forward algorithm by filling the function forward() in hmm.py.
- For the decoding task, implement the Viterbi algorithm by filling the function viterbi() in hmm.py.
- Both algorithms have the time complexity of $O(N^2T)$. Do not try brute force approaches, which have exponential time complexity. If your implementation has such complexity, you will be penalized.

3 Naive Bayes

In Naive Bayes, the hypothesis is defined as

$$h(x) = \underset{y}{\operatorname{argmax}}\, P(y|x) \tag{1}$$

where y is the label, and x is the feature. Applying the Bayes rule and assuming conditional independence between the label and dimensions of the feature, the hypothesis then becomes

$$h(x) = \underset{y}{\operatorname{argmax}}\, P(y) \prod_{j=1}^{d} P(x_j|y) \tag{2}$$

where d is the number of dimensions of the feature, and x_j is the j^{th} dimension of the feature. Multiplying many probabilities results in floating-point underflow, so we take the logarithm of the hypothesis (Page 15, NaiveBayes.pdf).

$$h(x) = \underset{y}{\operatorname{argmax}}\, log(P(y)) + \sum_{j=1}^{d} log(P(x_j|y)) \tag{3}$$

In this section, you will use Naive Bayes to classify sentences (sentiment analysis, to be more precise). Firstly, you will create a vocabulary which consists of every word in the training set. Assuming bag-of-words modelling in this homework, which do not consider the positions of the words and only count the words in a sentence, a sentence can be represented like that.

Example Vocabulary = {a, are, brown, car, cats, dogs, following, the, yellow }
Example Sentence = The brown dogs are following a brown car.
Its Representation = [1, 1, 2, 1, 0, 1, 1, 1, 0]

As stated in the previous equations, you need to estimate the following probabilities: $P(y)$ and $P(x|y)$. Let $\pi_c = P(y = c)$ be a probability of a sentence being in class c. Then, it is estimated as

$$\hat{\pi}_c = \frac{\sum_{i=1}^{n} I(y^{(i)} = c)}{n} \tag{4}$$

where n is the number of examples in the training set, $y^{(i)}$ is the label of i^{th} example, and I is a function that returns 1 if the condition holds (else 0).

In our model, we are going to think that: while writing a sentence of length m, every word is selected by rolling a die with d sides where d is the number of words in our vocabulary. Let θ_{jc} be the probability of rolling the word j in the vocabulary given that the label is c. Then, the probability of generating a sentence with length m is a multinomial distribution.

$$P(x|m, y = c) = \frac{m!}{x_1!x_2!x_3!...x_d!} \prod_{j=1}^{d} \theta_{jc}^{x_j} \tag{5}$$

We can skip calculating the first term since it will be the same for each class. Hence, the final hypothesis becomes

$$h(x) = \underset{c}{\operatorname{argmax}}\, log(\hat{\pi}_c) + \sum_{j=1}^{d} x_j log(\hat{\theta}_{jc}) \tag{6}$$

where $\hat{\theta}_{jc}$ is the estimation of θ_{jc}.

To calculate $\hat{\theta}_{jc}$, we will divide the number of occurrences of the j^{th} word in all examples labeled with class c by the number of total words in the examples labeled with c.

$$\hat{\theta}_{jc} = \frac{\sum_{i=1}^{n} I(y^{(i)} = c) x_j^{(i)}}{\sum_{i=1}^{n} [I(y^{(i)} = c) \sum_{j'=1}^{d} x_{j'}^{(i)}]} \tag{7}$$

where $y^{(i)}$ is the label of i^{th} example, $x_j^{(i)}$ is the number of occurrences of the j^{th} word in the vocabulary in the i^{th} example. The term $\sum_{j'=1}^{d} x_{j'}^{(i)}$ calculates the number of words in the i^{th} example. We will further apply Additive Smoothing with smoothing parameter 1 to deal with $log(0)$, i.e., the above equation can return zero. Therefore, we use the following equation in this homework.

$$\hat{\theta}_{jc} = \frac{1 + \sum_{i=1}^{n} I(y^{(i)} = c) x_j^{(i)}}{d + \sum_{i=1}^{n} [I(y^{(i)} = c) \sum_{j'=1}^{d} x_{j'}^{(i)}]} \tag{8}$$

where d is the number words in the vocabulary.

3.1 Task

- Implement the functions in **nb.py**.
- Create your vocabulary by extracting words from each sentence in the training set. You may or may not remove special characters. Describe your preprocessing in terms of comments in your source code.
- Train your Naive Bayes classifier with all training set. Print your test accuracy on the console by using Equation 6. You can simply skip the words in the test set that are not present in the training set.
- Your implementation should complete its execution in several seconds (not in minutes). Try to cache your variables instead of calculating them over and over again. Try to avoid multiple passes over the whole dataset. You will be penalized if your code runs for more than that.

4 Specifications

- The code must be in Python.
- You are also provided with simple testers. Note that passing a tester does not guarantee that your code works perfectly.
- Falsifying results, changing the composition of training and test data are strictly forbidden, and you will receive 0 if this is the case. Your programs will be examined to see if you have actually reached the results and if they are working correctly.

- Using any piece of code that is not your own is strictly forbidden and constitutes as cheating. This includes friends, previous homeworks, or the internet. The violators will be punished according to the department regulations.

- Follow the course page on ODTUClass for any updates and clarifications. Please ask your questions on the discussion section of ODTUClass instead of e-mailing.

- You have a total of 3 late days for **all** homeworks. The late submission penalty will be calculated using $5d^2$, that is, 1 day late submission will cost you 5 points, 2 days will cost you 20 points, and 3 days will cost you 45 points. No late submission is accepted after reaching a total of 3 late days.

5 Submission

Submission will be done via ODTUClass. You will submit a zip file called **hw4.zip** that contains all your **source code**. Also, please provide a **README** file describing how to run your code and perform the experiments.

6 Resources

Homework4's recitation slides can be accessed through the link.