Project Proposal

Name of the Project: Combining Labeled and Unlabeled Data with Co-Training

Team Members:

1. Kukkapalli Shravya - 2021101051
2. Penumalla Aditya Pavani - 2021101133

Reference Paper Link:

https://www.cs.cmu.edu/~avrim/Papers/cotrain.pdf

Introduction:

- The co-training algorithm considers the problem of using a large unlabeled sample to boost the performance of a learning algorithm when only a small set of labeled examples is available.
- Co-training assumes that
  - Features can be split into two sets
  - Each sub-feature set is sufficient to train a good classifier
  - The two sets are conditionally independent given the class.
- Consider a setting in which the description of each example can be partitioned into two distinct views, motivated by the task of learning to classify web pages.
- For example, the description of a web page can be partitioned into
  - The words occurring on that page
  - The words occurring in hyperlinks that point to that page.
- The algorithm stated in the paper is as follows:
  - For each example webpage x, x_1 was considered to be the bag of words appearing on the web page, and x_2 was considered to be the bag of words

underlined in all links pointing to the web page from other pages in the database. Classifiers were trained separately for x_1 and for x_2, using the naive Bayes algorithm.

- Consider a set L of labeled examples and a set U of unlabeled examples.
- The algorithm first creates a smaller pool U' containing u unlabeled examples. It then iterates the following procedure.
- First, use L to train two distinct classifiers: h_1 and h_2. h_1 is a naive Bayes classifier based only on the x_1 portion of the instance and h_2 is a naive Bayes classifier based only on the x_2 portion.
- Second, allow each of these two classifiers to examine the unlabeled set U' and select the p examples it most confidently labels as positive and the n examples it most confidently labels negative.
- Each example selected in this way is added to L, along with the label assigned by the classifier that selected it.
- Finally, the pool U is replenished by drawing $2p + 2n$ examples from U at random.

Given:

- a set L of labeled training examples
- a set U of unlabeled examples

Create a pool U' of examples by choosing u examples at random from U
Loop for k iterations:

Use L to train a classifier h_1 that considers only the x_1 portion of x
Use L to train a classifier h_2 that considers only the x_2 portion of x
Allow h_1 to label p positive and n negative examples from U'
Allow h_2 to label p positive and n negative examples from U'
Add these self-labeled examples to L
Randomly choose $2p + 2n$ examples from U to replenish U'

Deliverables:

1. Implementing the co-training algorithm mentioned above (for binary classification) to the dataset given in the paper.

- In this phase, we'll create a program that follows the Co-Training algorithm as described in the paper.
- This algorithm is designed to work with a dataset where we will be trying to classify items into one of two categories (binary classification).
- We'll apply this program to the dataset mentioned in the paper.

Data:

http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-51/www/co-training/data/

- This data set contains a subset of the WWW pages collected from computer science departments of various universities.
- The 1051 pages were manually classified into the following categories.
  - Course (230)
  - Non-Course (821)
- The files are organized into a directory structure with two directories at the top level
  - Fulltext - This directory contains the text on the web pages
  - Inlinks - This directory contains the anchor text on the hyperlinks pointing to the page.
- Under each of the two directories, there is one directory for each class (course, non-course). These directories in turn contain the Web pages. The file name of each page corresponds to its URL, where '/' was replaced with '^'.

Data Pre-processing:

- Each file in the data corresponds to an HTML file.

- First, we need to extract the text from the html files and convert it into a “bag of words”.

2. Implementing the co-training algorithm for multi-class classification.

- Following the completion of the binary classification part, we'll try to extend our program to work with datasets where we have more than two categories to classify (multi-class classification).
- The dataset that we will be using for this task is mentioned below.

Data:

http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

- This data set contains WWW pages collected from computer science departments of various universities.
- The 8,282 pages were manually classified into the following categories:
  - Student (1641)
  - Faculty (1124)
  - Staff (137)
  - Department (182)
  - Course (930)
  - Project (504)
  - Other (3764)
- For each class, the data set contains pages from the four universities
  - Cornell (867)
  - Texas (827)
  - Washington (1205)
  - Wisconsin (1263)

and 4,120 miscellaneous pages collected from other universities.

- The files are organized into a directory structure, one directory for each class. Each of these seven directories contains 5 subdirectories, one for each of the 4 universities and one for the miscellaneous pages. These directories in turn contain the Web pages. The file name of each page corresponds to its URL, where '/' was replaced with '^'.
- Data pre-processing can be done as mentioned above.

Train/Test Splits:

- Since each university's web pages have their own idiosyncrasies, training can be done on three of the universities plus the *misc* collection, and testing on the pages from a fourth, held-out university.

3. Generating, creating, or finding other relevant datasets that could be used to evaluate this method.

- Here, We'll either create new datasets or locate existing ones that are relevant for evaluating the Co-Training method.
- These datasets should allow us to test how well the algorithm performs.

4. Comparing against the current methods like self-training algorithm which uses labeled and unlabelled datasets.

- To assess the effectiveness of the Co-Training algorithm, we'll compare its performance against other semi-supervised methods like the Self-Training algorithm.

Analysis:

- Comparing the performance of co-training and the standard supervised learning.
- Comparing the performance of co-training and the other methods that use labeled and unlabelled datasets.

Deliverables for checkpoint - 1

- Implementing the co-training algorithm mentioned above (for binary classification) to the dataset given in the paper. **(8th October - 21st October)**
- Implementing the co-training algorithm for multi-class classification. **(22nd October to 30th October)**

Deliverables for checkpoint - 2

- Generating, creating, or finding other relevant datasets which could be used to evaluate this method. **(31st October to 11th November)**
- Comparing against the current methods like self-training algorithm which uses labeled and unlabelled datasets. **(12th November to 27th November)**