Stony Brook University
CSE512 – Machine Learning – Spring 21
Homework 4, Version 2, updated 2021/04/05
Due: Apr 13 at midnight 11:59pm

This homework contains three questions. The last two questions require programming. This homework provides you an opportunity to explore two popular machine learning libraries. Although these are well-implemented libraries, you will soon see that you cannot treat them as black boxes. In order to obtain good results, you will need to know what the important hyper parameters are and how to tune them. The maximum score for this home work is 100 + 15 bonus points.

1 Question 1 – Support Vector Machines (20 points)

1.1 Linear case (10 points)

Consider training a linear SVM on linearly separable dataset consisting of n points. Let m be the number of support vectors obtained by training on the entire set. Show that the LOOCV error is bounded above by $\frac{m}{n}$.

Hint: Consider two cases: (1) removing a support vector data point and (2) removing a non-support vector data point.

1.2 General case (10 points)

Now consider the same problem as above. But instead of using a linear SVM, we will use a general kernel. Assuming that the data is linearly separable in the high dimensional feature space corresponding to the kernel, does the bound in previous section still hold? Explain why or why not.

2 Question 2 – XGBoost (30 points)

In this question, you will use XGBoost to predict the income of a person. The data for this question is under HW4_q2.

You can use the implementation from `https://github.com/dmlc/xgboost`. To install it, use *pip install xgboost* and then import it using *from xgboost import XGBClassifier*. Here is a tutorial on how to install and use it: `https://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/`.

You will use the Census Income Dataset (`https://archive.ics.uci.edu/ml/datasets/census+income`). This dataset includes attributes of 48842 people, such as their age and education. The task is to predict whether their income exceeds \$50K per year. For each person, there are 14 attributes in total. Some attributes are categorical, like the education, and other attributes have continuous values, like the age. Review the attributes and their possible values at the link provided above.

The first step is to load your training and test data that are provided in the files $adult.data$ and $adult.test$ correspondingly. (Tip: To load the data, you can use numpy.genfromtxt). Inspect your data carefully: The first 14 columns of each file correspond to the features (attributes) of the persons. The last column corresponds to the labels (income $\leq 50K$ or $> 50K$). Since you have both categorical and continuous attributes, you will first need to load the data as strings. Use "dtype=np.object" to load them to numpy arrays.

Then, you need to convert them to float. Keep the continuous attributes as they are. For the categorical attributes, you will need to convert them to integers. To do that, you can use sklearn.preprocessing.OrdinalEncoder.

For each column that corresponds to a categorical attribute, you can construct a new encoder that converts its values to integers. For example, the education attribute can be encoded as "Bachelors" = 0, "Masters" = 1, "Doctorate" = 2, etc. Same for the labels column (e.g. "$\leq 50K$" = 0 and "$> 50K$" = 1). At the end, convert your whole feature matrix and labels to float.

Important: Be careful when you convert the categorical attributes and labels of your test data. They need to correspond to the same integer values as in your training data. For example, if "Bachelors" corresponds to 0 in your training data, then it should also correspond to 0 in your test data. To ensure that, for each column of your training data, you can use the fit_transform method of sklearn.preprocessing.OrdinalEncoder and then use the transform method for the corresponding column of your test data.

2.1 Question 2.1 (10 points)

Using the whole training set, train an XGBoost classifier and and test it on the test set. Report the accuracy and the confusion matrix on both the train and test sets.

2.2 Question 2.2 (20 points)

Perform k-fold cross-validation to find the best set of hyper-parameters for XGBoost. To do that, you will need to split your training data (X) to train (X_{train}) and validation (X_{val}) sets. For each fold (split), you will train a model on the training data X_{train} (subset of the original training data X) and test it on the validation set X_{val}. Use $k = 10$ folds. You can use sklearn.model_selection.KFold.

Report the best cross-validation accuracy. Use this set of hyper-parameters and train XGBoost on the entire training data, and report the accuracy and confusion matrix on the test data.

3 Question 3 – SVM for object detection (50 points + 15 bonus points)

In this question, you will train an SVM and use it for detecting human hands in images. You can use the SVM implementation of sklearn. For this question, it is sufficient and also much faster to use linear SVM (sklearn.svm.LinearSVC), but you can also experiment with non-linear kernel using sklearn.svm.SVC (it might take long time though).

As features, you will use deep features extracted from detectron2
`https://github.com/facebookresearch/detectron2`. We provide the utility functions for feature extraction.

To detect human hands in images, we need a classifier that can distinguish between hand image patches from non-hand patches. To train such a classifier, we can use SVMs. The training data is typically a set of images with bounding boxes of the hands. Positive training examples are image patches extracted at the annotated locations. A negative training example can be any image patch that does not significantly overlap with the annotated hands. Thus there potentially many more negative training examples than positive training examples. Due to memory limitation, it will not be possible to use all negative training examples at the same time. In this question, you will implement hard-negative mining to find hardest negative examples and iteratively train an SVM.

3.1 Preparation

3.1.1 For Conda environment

Assume you have download the homework file and store in director hw4.

```
cd hw4

# Create new environment
conda create -n cse512_spring21_hw4 python=3.7
```

```
conda activate cse512_spring21_hw4

# Install pytorch
conda install pytorch torchvision cudatoolkit=10.0 -c pytorch

# Install dependency libraries
python -m pip install opencv-python mean_average_precision
python -m pip install pycocotools scikit-learn

# Install detectron
git clone https://github.com/facebookresearch/detectron2.git \
--branch v0.1.1 detectron2_v0.1.1
cd detectron2_v0.1.1
git checkout db1614e
python -m pip install -e .

# Move HW4_q3 into detectron2_v0.1.1
mv ../HW4_q3 .
cd HW4_q3
```

3.1.2 For Google Colab

For this question, you will need to use a GPU. You can use Google Colab. If so, remember to change the runtime type to GPU. Then, install the following prerequisites:

```
!pip install mean_average_precision
!git clone https://github.com/facebookresearch/detectron2.git --branch
v0.1.1 detectron2_v0.1.1
%cd detectron2_v0.1.1
!git checkout db1614e
!pip install -e .
```

Inside the detectron2_v0.1.1 directory, unzip the given HW4_q3.zip:

```
!unzip HW4_q3.zip
%cd HW4_q3
```

3.1.3 Data download

Download the ContactHands dataset and put it inside the HW4_q3/ directory from `http://vision.cs.stonybrook.edu/~supreeth/ContactHands_data_website/` or by running:

```
!wget https://public.vinai.io/ContactHands.zip
!unzip ContactHands
```

The file ContactHands/README.md provides useful information regarding the structure of this dataset. For more information about the dataset, see:

'Detecting Hands and Recognizing Physical Contact in the Wild.' S. Narasimhaswamy, T. Nguyen, M. Hoai. *Advances in Neural Information Processing Systems (**NeurIPS**)*, 2020.

3.1.4 Data split

Under HW4_q3/sets/ you can find the data split that you will use for this question: `train.txt` corresponds to the training set, `validation.txt` corresponds to the validation set, `extra-train.txt` corresponds to more data for training (optional), and `test.txt` corresponds to the test set. Copy those files under ContactHands/ImageSets/Main/.

3.1.5 Annotations

Under HW4_q3/Annotations/ you can find the annotations that you will use for this question. The folder contains annotations for the training, validation and extra-train data. The annotations for the test data will not be released, but they will be used for testing your final submission result. Copy the Annotations/ folder and replace the ContactHands/Annotations/.

3.2 Helper functions

To help you, a number of utility functions and classes are provided in HW4_q3/. The most important functions are in `hw4_utils.py`:

1. Run `python hw4_utils.py -va` to visualize some annotated samples.
2. Use `get_pos_and_random_neg()` to get initial training/validation data (dataset = 'train' or dataset = 'validation' correspondingly). This function returns the training/validation feature matrix D, the corresponding training/validation labels lb. There are 2 classes: positive (1) and negative (-1) class. Positive instances are deep features extracted at the locations of hands. Negative instances are deep features at random locations of the images. Important: You first need to initialize `feat_extractor = prepare_second_stream()` before calling this function.
3. Use `detect()` to run the sliding window detector. This returns a numpy array of bounding box locations and corresponding SVM scores. This function can be used for detecting hands in an image. It can also be used to find hardest negative examples in an image.
4. Use `generate_result_file()` to generate a result file (dataset = 'validation' or dataset = 'test' for the validation or test set correspondingly). Set the argument num_img to run the detection for a subset of test images (e.g. num_img=100).
5. Use `compute_mAP()` to compute the Average Precision for the result file.
6. Use `get_iou()` to compute the overlap between two rectangular regions. The overlap is defined as the area of the intersection over the area of the union. A returned detection region is considered correct (true positive) if there is an annotated hand such that the overlap between the two boxes is more than 0.5.
7. Some useful OpenCV functions to work with images are: imread, imshow, imresize.

In addition, `detect.py` includes the feature extraction using the detectron2.

3.3 What to implement

1. *(15 points)* Use the `get_pos_and_random_neg()` function to get the training data and train an SVM classifier *clf*. You can use the sklearn.svm.LinearSVC. Since you have a large number of data, you can limit the maximum number of iterations (e.g. max_iter=1000).

   Use the trained classifier to generate a result file (use `generate_result_file()`) for the validation data. Then, run the `compute_mAP()` to compute the AP and plot the precision recall curve. Submit your AP and precision recall curve on the validation data.

Algorithm 1 Hard negative mining algorithm

$PosD \leftarrow$ all annotated hands
$NegD \leftarrow$ random image patches
$(\mathbf{w}, b) \leftarrow$ trainSVM$(PosD, NegD)$
for $iter = 1, 2, \cdots$ **do**
  $\mathbf{A} \leftarrow$ All non support vectors in $NegD$.
  $\mathbf{B} \leftarrow$ Hardest negative examples ▷ Run UB detection and find negative patches that ▷ violate the SVM margin constraint the most
  $NegD \leftarrow (NegD \setminus \mathbf{A}) \cup \mathbf{B}$.
  $(\mathbf{w}, b) \leftarrow$ trainSVM$(PosD, NegD)$
end for

2. Implement hard negative mining algorithm given in Algorithm 1. Positive training data and random negative training data can be generated using the `get_pos_and_random_neg()` function. At each iteration, you should remove negative examples that do not correspond to support vectors from the negative set. Use the function `detect()` on train images to identify hardest negative examples and include them in the negative training set.

   Hints: (1) a negative example should not have significant overlap with any annotated hand. You can experiment with different threshold but 0.3 is a good starting point. (2) you should compute the objective value at each iteration; the objective values should not decrease. (3) to speed up you can write a modified version of `detect()` that uses a different set of bounding box proposals for training.

3. *(20 points)* Run the negative mining for 10 iterations. Assume your computer is not so powerful and so you cannot add more than 10000 new negative training examples at each iteration. Record the objective values (on train data) and the APs (on validation data) through the iterations. Plot the objective values. Plot the APs. On the validation data, you can also use `get_pos_and_random_neg` to sample 10 negative patches per validation image. To calculate AP, use `sklearn.metrics.average_precision_score`.

4. *(15 points)* For this question, you will need to generate a .npy result file for the test data using the function `generate_result_file()`. You will need to submit this file by uploading to `https://forms.gle/Y5qzA6Mi5Sz5SB2u9` to receive the AP on test data. Report the AP in your answer file. **Important Note:** You MUST use your Stony Brook ID as the name of your submission file, i.e., `your_SBU_ID.npy` (e.g., 012345679.npy). Your submission will not be evaluated if you don't use your SBU ID. For this question, you don't need to have the highest AP to earn full marks.

5. *(15 bonus points)* Your submitted result file for test data will be automatically entered in a competition for fame (`https://bit.ly/31L9Cov`). We will maintain a leader board and the top three entries at the end of the competition (due date) will receive 15, 10, and 5 bonus points. The ranking is based on AP.

   You can submit the result as frequent as you want. However, the evaluation server will only evaluate all submissions two times a day, at 09:00am and 09:00pm. The system only keeps the recent submission file, and your new submission will override the previous ones. Therefore, you have two chances a day to evaluate your method.

   You are allowed to use any feature types and classifiers for this part of the homework. In addition, you are allowed to fine-tune any part of the given code. For example, you can try tuning the sliding window detection, e.g., try different image scales, window sizes and strides. You can use more training data.

You can run hard negative mining algorithm for as many iterations as you want, and the number of negative examples added at each iteration is not limited by 10000. You can train with all available data, including "train", "validation", "extra-train". You can also use data from other datasets. For example, see `https://www3.cs.stonybrook.edu/~cvl/projects/hand_det_attention/`.

Check the following papers for the state-of-the-art performance on hand detection. If your method significantly outperforms these papers, we invite you to write a paper with us! Please email us directly if you think you have an awesome technique that obtains good results.

'Detecting Hands and Recognizing Physical Contact in the Wild.' S. Narasimhaswamy, T. Nguyen, M. Hoai. *Advances in Neural Information Processing Systems (**NeurIPS**)*, 2020.

'Contextual Attention for Hand Detection in the Wild.' S. Narasimhaswamyy, Z. Wei, Y. Wang, J. Zhang, M. Hoai. *Proceedings of International Conference on Computer Vision (**ICCV**)*, 2019.

4 What to submit

You will need to submit both your code and your answers to questions on Blackboard. Put the answer file and your python code in a folder named: SUBID_FirstName_LastName (e.g., *10947XXXX_Barack_Obama*). Zip this folder and submit the zip file on Blackboard. Your submission must be a zip file, i.e, SUBID_FirstName_LastName.zip.

The answer file should be named: hw4-answers.pdf. You can use Latex if you wish, but it is not compulsory. The first page of the hw4-answers.pdf should be the filled cover page at the end of this homework. The remaining of the answer file should contain answers to Questions 1, 2, 3.

Your Python code for Questions 2 and 3 can be in separate notebooks or python files. Make sure that the name of each file is self-explanatory.

Make sure you follow the instructions carefully.

5 Cheating warnings

Don't cheat. You must do the homework yourself, otherwise you won't learn. You cannot ask and discuss with students from previous years. You cannot look up the solution online.

Cover page for answers.pdf
CSE512 Spring 2021 - Machine Learning - Homework 4

Your Name:

Solar ID:

NetID email address:

Names of people whom you discussed the homework with: