Stony Brook University
CSE512 – Machine Learning – Fall 18
Homework 4, Version 1
Due: 15 Oct 2018 at midnight 23:59

This homework contains 3 questions. The last two questions require programming. Question 3 requires the SVM implementation from Question 2. The maximum number of points is 100 plus 10 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 – Implementation of SVMs (35 points + 10 bonus points)

In this problem, you will implement SVMs using quadratic programming. Quadratic programs refer to optimization problems in which the objective function is quadratic and the constraints are linear. Quadratic programs are well studied in optimization literature, and there are efficient solvers. Many Machine Learning algorithms are reduced to solving quadratic programs. In this question, we will use the quadratic program solver of Matlab to optimize the dual objective of a kernel SVM.

The dual objective of kernel SVM can be written as:

$$\underset{\boldsymbol{\alpha}}{\text{maximize}} \sum_{j=1}^{n} \alpha_j - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i \alpha_i y_j \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{1}$$

$$\text{s.t.} \sum_{j=1}^{n} y_j \alpha_j = 0 \tag{2}$$

$$0 \leq \alpha_j \leq C \ \forall j. \tag{3}$$

1. (10 points) Write the SVM dual objective as a quadratic program. Look at the `quadprog` function of Matlab, and write down what $\mathbf{H}$, $\mathbf{f}$, $\mathbf{A}$, $\mathbf{b}$, **Aeq**, **beq, lb, ub** are.

2. Use quadratic programming to optimize the dual SVM objective. In Matlab, you can use the function `quadprog`.

3. Write a program to compute $\mathbf{w}$ and b of the primal from $\boldsymbol{\alpha}$ of the dual. You only need to do this for linear kernel.

4. (10 points) Set $C = 0.1$, train an SVM with linear kernel using `trD, trLb` in `q2_1_data.mat` (in Matlab, load the data using `load q2_1_data.mat`). Test the obtained SVM on `valD, valLb`, and report the accuracy, the objective value of SVM, the number of support vectors, and the confusion matrix.

5. (5 points) Repeat the above question with $C = 10$.

6. *(10 points + 10 Bonus)* For this question, you will use multiple binary kernel SVMs to do activity recognition on the UCF101 data (see `http://crcv.ucf.edu/data/UCF101.php`). Originally, this data has 101 classes but for this homework you will be using just 10 classes of data to train your multiclass SVM classifier and compete in an in-class Kaggle competition:

   `https://www.kaggle.com/t/c370f405783f4d38b0383b3580d182b9`.

   Training data are provided in `q2_2_data.mat`. Use `trD, trLb` for training your SVM classifier. Validate your obtained SVM on `valD, valLb`, then provide the prediction for `tstD` in a `.csv` file. You can download the data from the competition webpage.

   We have already computed feature vectors for you. Each feature vector has 4096 features. For reference, we also provide the jpeg images from which the feature vectors were extracted, but you are not required to use them. The training and validation labels are correspondence to `trLb` and `valLb` in `q2_2_data.mat`. For multi-class classification, you can use one-versus-one or one-versus-rest approaches. You're not allowed to use any other classifiers for this submission. Report the best accuracy and the approach, the kernel, the parameters you used to achieve that.

   We will maintain a leader board, and the top three entries at the end of the competition (assignment due date) will receive 10 bonus points. Any submission that rises to top three after the assignment deadline is not eligible for bonus points. The ranking will be based on the Categorization accuracy (percentage of correct label).

   To prevent exploiting test data, you are allowed to make a maximum of 3 submissions per 24 hours. Your submission will be evaluated immediately and the leader board will be updated.

   For this question, you don't need to have the highest accuracy to earn full points. However, you might loose all or some points if your performance is much lower than a certain threshold. The threshold will be determined by us, based on what we believe to be the minimum value that a correct implementation should achieves.

3 Question 3 – SVM for object detection (45 points + 10 bonus points)

In this question, you will train an SVM and use it for detecting human upper bodies in your favorite TV series The Big Bang Theory. You must use your SVM implementation from Question 2.

To detect human upper bodies in images, we need a classifier that can distinguish between upper-body image patches from non-upper-body patches. To train such a classifier, we can use SVMs. The training data is typically a set of images with bounding boxes of the upper bodies. 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 upper bodies. 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 Data

Training images are provided in the subdirectory `trainIms`. The annotated locations of the upper bodies are given in `trainAnno.mat`. This file contains a cell structure `ubAnno`; `ubAnno{i}` is the annotated locations of the upper bodies in the i^{th} image. `ubAnno{i}` is $4 \times k$ matrix, where each column corresponds to an upper body. The rows encode the left, top, right, bottom coordinates of the upper bodies (the origin of the image coordinate is at the top left corner).

Images for validation and test are given in `valIms, testIms` respectively. The annotation file for test images is not released.

3.2 External library

Raw image intensity values are not robust features for classification. In this question, we will use Histogram of Oriented Gradient (HOG) as image features. HOG uses the gradient information instead of intensities, and this is more robust to changes in color and illumination conditions. See [1] for more information about HOG, but it is not required for this assignment.

To use HOG, you will need to install an `VL_FEAT`: `http://www.vlfeat.org`. This is an excellent cross-platform library for computer vision and machine learning. However, in this homework, you are only allowed to use the HOG calculation and visualization function `vl_hog`. In fact, you should not call `vl_hog` directly. Use the supplied helper functions instead; they will call `vl_hog`.

3.3 Helper functions

To help you, a number of utility functions and classes are provided. The most important functions are in `HW4_Utils.m`.

1. Run `HW4_Utils.demo1` to see how to read and display upper body annotation
2. Run `HW4_Utils.demo2` to display image patches and HOG feature images. Compare HOG features for positive and negative examples, can you see why HOG would be useful for detect upper bodies?
3. Use `HW4_Utils.getPosAndRandomNeg()` to get initial training and validation data. Positive instances are HOG features extracted at the locations of upper bodies. Negative instances are HOG features at random locations of the images. The data used in Question 3 is actually generated using this function.
4. Use `HW4_Utils.detect` to run the sliding window detector. This returns a list of locations and SVM scores. This function can be used for detecting upper bodies in an image. It can also be used to find hardest negative examples in an image.
5. Use `HW4_Utils.cmpFeat` to compute HOG feature vector for an image patch.
6. Use `HW4_Utils.genRsltFile` to generate result file.
7. Use `HW4_Utils.cmpAP` to compute the Average Precision for the result file.
8. Use `HW4_Utils.rectOverlap` 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 upper body such that the overlap between the two boxes is more than 0.5.
9. Some useful Matlab functions to work with images are: imread, imwrite, imshow, rgb2gray, imresize.

Algorithm 1 Hard negative mining algorithm

$PosD \leftarrow$ all annotated upper bodies
$NegD \leftarrow$ 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

3.4 What to implement?

1. (15 points) Use the training data in `HW4_Utils.getPosAndRandomNeg()` to train an SVM classifier. Use this classifier to generate a result file (use `HW4_Utils.genRsltFile`) for validation data. Use `HW4_Utils.cmpAP` to compute the AP and plot the precision recall curve. Submit your AP and precision recall curve (on validation data).

2. Implement hard negative mining algorithm given in Algorithm 2. Positive training data and random negative training data can be generated using `HW4_Utils.getPosAndRandomNeg().` At each iteration, you should remove negative examples that do not correspond to support vectors from the negative set. Use the function `HW4_Utils.detect` on train images to identify hardest negative examples and include them in the negative training set. Use `HW4_Utils.cmpFeat` to compute HOG feature vectors.

   Hints: (1) a negative example should not have significant overlap with any annotated upper body. You can experiment with different threshold but 0.3 is a good starting point. (2) make sure you normalize the feature vectors for new negative examples. (3) you should compute the objective value at each iteration; the objective values should not decrease.

3. (15 points) Run the negative mining for 10 iterations. Assume your computer is not so powerful and so you cannot add more than 1000 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.

4. *(15 points)* For this question, you will need to generate a result file for test data using the function `HW4_Utils.genRsltFile`. You will need to submit this file on our evaluation server to receive the AP on test data. Report the AP in your answer file. For this question, you don't need to have the highest AP to earn full marks. However, you might loose all or some points if your performance is much lower than a certain threshold. The threshold will be determined by us, based on what we believe to be the minimum value that a correct implementation should achieve.

   Note: Further details about the evaluation server and how to make submissions will be released later.

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

   You are allowed to use any feature types for this part of the homework. For example, you can use different parameter settings for HOG feature computation. You can even combine multiple HOG features. You can also append HOG features with geometric features (e.g., think about the locations of the upper body). You are allowed to perform different types of feature normalization (e.g, L_1, L_2). You can use both training and validation data to train your classifier. You are allowed to use SVMs, Ridge Regression, Lasso Regression, or any technique that we have covered. 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 1000. You are not allowed to use Deep Learning features.

4 What to submit?

4.1 Blackboard submission

You will need to submit both your code and your answers to questions on Blackboard. Put the answer file and your code in a folder named: SUBID_FirstName_LastName (e.g., *10947XXXX_lionel_messi*). 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: answers.pdf. The first page of the answers.pdf should be the filled cover page at the end of this homework. The remaining of the answer file should contain:

1. Answers to Question 1 and 2
2. Answers to Question 3.4, including the requested plots.

4.2 Prediction submission

For Question 2.6, you must submit a `.csv` file to get the accuracy through Kaggle (`https://www.kaggle.com/t/c370f405783f4d38b0383b3580d182b9`. A submission file should contain two columns: Id and Class. The file should contain a header and have the following format.

$$\begin{array}{cc} Id, & Class \\ 1, & 1 \\ 2, & 10 \\ ... & ... \end{array}$$

A sample submission file is available from the competition site and our handout.

For Questions 3.4.4, 3.4.5, you must submit a mat file. A submission file can be automatically generated by `HW4_Utils.genRsltFile`. You need to upload this result file on our evaluation server to obtain AP.

5 Cheating warnings

Don't cheat. You must do the homework yourself, otherwise you won't learn. You must use your SBU ID as your file name for the competition. Do not fake your Stony Brook ID to bypass the submission limitation per 24 hours. Doing so will be considered cheating.

References Cited

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

Cover page for answers.pdf

CSE512 Fall 2018 - Machine Learning - Homework 4

Your Name:

Solar ID:

NetID email address:

Names of people whom you discussed the homework with: