Stony Brook University
CSE512 – Machine Learning – Spring 18
Homework 2, Version 2, last updated: 20 Feb 2018
Due: 4 Mar 2018 at midnight 23:59

This homework contains 4 questions. The last two questions require programming. Question 4 requires an SVM implementation from Question 3. The maximum number of points is 100 plus 20 bonus points.

1 Question 1 – Ridge Regression and LOOCV (20 points)

In class, you learned about using cross validation as a way to estimate the true error of a learning algorithm. The preferred solution is *Leave-One-Out Cross Validation* (LOOCV), which provides an almost unbiased estimate of this true error, but it can take a really long time to compute. In this problem, you will derive a formula for efficiently computing the LOOCV error for ridge regression.

Given a set of n data points and associated labels $\{\mathbf{x}_i, y_i | \mathbf{x}_i \in \Re^k, y_i \in \Re\}_{i=1}^n$. Ridge regression find the weight vector $\mathbf{w}$ and a bias term b to optimize the following:

$$\underset{\mathbf{w},b}{\text{minimize}} \ \lambda||\mathbf{w}||^2 + \sum_{i=1}^{n}(\mathbf{w}^T\mathbf{x}_i + b - y_i)^2. \tag{1}$$

1.1 (4 points)

Let $\overline{\mathbf{w}} = [\mathbf{w}; b], \overline{\mathbf{X}} = [\mathbf{X}; \mathbf{1}_n^T], \bar{\mathbf{I}} = [\mathbf{I}_k, \mathbf{0}_k; \mathbf{0}_k^T, 0]$, $\mathbf{C} = \overline{\mathbf{X}}\overline{\mathbf{X}}^T + \lambda\bar{\mathbf{I}}$, and $\mathbf{d} = \overline{\mathbf{X}}\mathbf{y}$. Show that the solution of Ridge regression is:

$$\overline{\mathbf{w}} = \mathbf{C}^{-1}\mathbf{d} \tag{2}$$

1.2 (3 points)

Now suppose we remove $\mathbf{x}_i$ from the training data, let $\mathbf{C}_{(i)}, \mathbf{d}_{(i)}, \overline{\mathbf{w}}_{(i)}$ be the corresponding matrices for removing $\mathbf{x}_i$. Express $\mathbf{C}_{(i)}$ in terms of $\mathbf{C}$ and $\mathbf{x}_i$. Express $\mathbf{d}_{(i)}$ in terms of $\mathbf{d}$ and $\mathbf{x}_i$.

1.3 (4 points)

Express $\mathbf{C}_{(i)}^{-1}$ in terms of $\mathbf{C}^{-1}$ and $\mathbf{x}_i$. Hint: use the Sherman-Morrison formula:

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^T)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{A}^{-1}}{1 + \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}} \tag{3}$$

1.4 (3 points)

Show that

$$\overline{\mathbf{w}}_{(i)} = \overline{\mathbf{w}} + (\mathbf{C}^{-1}\overline{\mathbf{x}}_i)\frac{-y_i + \overline{\mathbf{x}}_i^T\overline{\mathbf{w}}}{1 - \overline{\mathbf{x}}_i^T\mathbf{C}^{-1}\overline{\mathbf{x}}_i} \tag{4}$$

1.5 (3 points)

Show that the leave-one-out error for removing the i^{th} training data is:

$$\overline{\mathbf{w}}_{(i)}^T\overline{\mathbf{x}}_i - y_i = \frac{\overline{\mathbf{w}}^T\mathbf{x}_i - y_i}{1 - \overline{\mathbf{x}}_i^T\mathbf{C}^{-1}\overline{\mathbf{x}}_i} \tag{5}$$

1.6 (3 points)

The LOOCV is defined as: $\sum_{i=1}^{n}(\overline{\mathbf{w}}_{(i)}^{T}\overline{\mathbf{x}}_i - y_i)^2$. What is the algorithmic complexity of computing LOOCV error using the formula given in Question 1.5? How is it compared with the usual way of computing LOOCV? Note that the complexity of inverting a $k \times k$ matrix is $O(k^3)$.

$$(\overline{\mathbf{X}}\overline{\mathbf{X}}^T + \lambda\bar{\mathbf{I}})\overline{\mathbf{w}} = \overline{\mathbf{X}}diag(\mathbf{s})\mathbf{y} \tag{6}$$

2 Question 2 – Naive Bayes and Logisitic Regression (20 points)

2.1 Naive Bayes with both continuous and boolean variables (10 points)

Consider learning a function $\mathbf{X} \to Y$ where Y is boolean, where $\mathbf{X} = (X_1, X_2)$, and where X_1 is a boolean variable and X_2 a continuous variable. State the parameters that must be estimated to define a Naive Bayes classifier in this case. Give the formula for computing $P(Y|\mathbf{X})$, in terms of these parameters and the feature values X_1 and X_2.

2.2 Naive Bayes and Logistic Regression with Boolean variables (10 points)

In class, we showed that when Y is Boolean and $\mathbf{X} = (X_1, \cdots, X_d)$ is a vector of continuous variables, the the assumptions of the Gaussian Naive Bayes classifier imply that $P(Y|\mathbf{X})$ is given by the logistic function with appropriate parameters $\boldsymbol{\theta}$. In particular:

$$P(Y = 1|\mathbf{X}) = \frac{1}{1 + \exp(-(\sum_{i=1}^{d}\theta_i X_i + \theta_{d+1}))} \tag{7}$$

Consider instead the case where Y is Boolean and $\mathbf{X} = (X_1, \cdots, X_d)$ is a vector of *Boolean* variables. Prove for this case also that $P(Y|\mathbf{X})$ follows this same form (and hence that Logistic Regression is also the discriminative counterpart to a Naive Bayes generative classifier over Boolean features).

3 Question 3 – Implementation of SVMs (40 points + 10 bonus)

In this problem, you will implement SVMs using two different optimization techniques:(1) quadratic programming and (2) stochastic gradient descent.

3.1 Implement Kernel SVM using Quadratic Programming (15 points)

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:

$$\underset{\boldsymbol{\alpha}}{\text{maximize}} \sum_{j=1}^{n}\alpha_j - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i\alpha_i y_j\alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{8}$$

$$\text{s.t.} \sum_{j=1}^{n} y_j\alpha_j = 0 \tag{9}$$

$$0 \leq \alpha_j \leq C\ \forall j. \tag{10}$$

1. (5 points) Write the SVM dual objective as a quadratic program. Look at the `quadprog` function of Matlab, and write down what **H**, **f**, **A**, **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. (5 points) Set $C = 0.1$, train an SVM with linear kernel using `trD, trLb` in `q3_1_data.mat` (in Matlab, load the data using `load q3_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 $C = 10$.

3.2 Implement Multiclass SVM using Stochastic Gradient Descent (25 points + 10 bonus points)

In this question, you will implement mutliclass SVM with Stochastic Gradient Descent. We will consider the Crammer and Singer's SVM formulation [1]. Note that there are several different formulations for multiclass SVM (see `https://www.csie.ntu.edu.tw/~cjlin/papers/multisvm.pdf`).

Consider a multiclass classification problems with k classes. We want to train k weight vectors $\mathbf{w}_1, \cdots, \mathbf{w}_k$, one for each class. $\mathbf{w}_i \in \Re^d$, where d is the dimension of the data. Let $\mathbf{W}$ denote the matrix of all weight vectors $[\mathbf{w}_1, \cdots, \mathbf{w}_k]$. Suppose we have n training data instances $\{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \Re^d, y_i \in \{1, \cdots, k\}\}$. Crammer and Singer's formulation seeks to optimize:

$$\underset{\mathbf{W}}{\text{minimize}} \; \frac{1}{2}\sum_{j=1}^{k} ||\mathbf{w}_j||^2 + C\sum_{i=1}^{n} L(\mathbf{W}, \mathbf{x}_i, y_i) \tag{11}$$

Here $L(\mathbf{W}, \mathbf{x}_i, y_i)$ is the *Multiclass Hinge loss* of the i-th instance:

$$L(\mathbf{W}, \mathbf{x}_i, y_i) = \max\{\mathbf{w}_{\hat{y}_i}^T \mathbf{x}_i - \mathbf{w}_{y_i}^T \mathbf{x}_i + 1, 0\} \text{ where } \hat{y}_i = \underset{j \neq y_i}{\text{argmax}} \, \mathbf{w}_j^T \mathbf{x}_i. \tag{12}$$

By distributing the regularization term to each training instance, we obtain the following equivalent objective:

$$\underset{\mathbf{W}}{\text{minimize}} \sum_{i=1}^{n} \left(\frac{1}{2n}\sum_{j=1}^{k} ||\mathbf{w}_j||^2 + CL(\mathbf{W}, \mathbf{x}_i, y_i) \right) \tag{13}$$

Let $L_i = \frac{1}{2n}\sum_{j=1}^{k} ||\mathbf{w}_j||^2 + CL(\mathbf{W}, \mathbf{x}_i, y_i)$. We can use stochastic gradient descent to optimize this objective. The update rule for $\mathbf{w}_j$ with the i^{th} training instance will be:

$$\mathbf{w}_j^{new} \leftarrow \mathbf{w}_j^{cur} - \eta \partial_{\mathbf{w}_j} L_i \; \forall j \tag{14}$$

where $\partial_{\mathbf{w}_j} L_i$ denote the sub-gradient of L_i w.r.t. $\mathbf{w}_j$.

Algorithm 1 Stochastic gradient descent for multiclass SVM

```
for $epoch = 1, 2, \cdots,$ max_epoch do
    $\eta \leftarrow \eta_0/(\eta_1 + epoch)$                    ▷ Update the learning rate
    $(i_1, \cdots, i_n) = \text{permute}(1, \cdots, n)$.        ▷ Shuffle the indexes of training data
    for $i \in \{1, 2, \cdots, n\}$ do
        Update $\mathbf{W}$ using Eq. (14)
    end for
end for
```

1. *(2 points)* What is the subgradient of L_i wrt to $\mathbf{w}_{y_i}$?

2. *(2 points)* What is the subgradient of L_i wrt to $\mathbf{w}_{\hat{y}_i}$?

3. *(1 point)* What is the subgradient of L_i wrt to $\mathbf{w}_j$ for $j \neq y_i$ and $j \neq \hat{y}_i$.

4. Implement SGD for multiclass SVM given in Algorithm 1. η_0, η_1 are tunable parameters. Initially start all the weights at 0.

5. *(5 points)* Using `trD, trLb` in `q3_1_data.mat` as your training set, run 2000 epochs over the dataset using $\eta_0 = 1, \eta_1 = 100$, $C = 0.1$ and $C = 10$. Plot the loss in Eq. (13) after each epoch. Compare with the objective value obtained in 3.1.4.

6. *(5 points)* Using the $\mathbf{W}$ learned after 2000 epochs, report:

   (a) The prediction error on `valD, valLb` in `q3_1_data.mat` (test error)

   (b) The prediction error on `trD, trLb` (training error)

   (c) $\sum_{j=1}^{k} ||\mathbf{w}_j||^2$

7. *(10 points + 10 Bonus)* For this question, you will use the previous multiclass implementation (Q 3.2) or multiple binary kernel SVMs (Q 3.1) 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/c/hw2-activity-recognition-cse512-spr18/`.

   Training data are provided in `q3_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:

   `https://www.kaggle.com/c/hw2-activity-recognition-cse512-spr18/data`

   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 `q3_2_data.mat`. Play around with parameters, epochs to achieve a good score. You're not allowed to use any other classifiers for this submission. Report the best accuracy and the parameters you used to achieve that. Also report a plot of the training loss after each epoch.

   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 2 submissions per 24 hours. Your submission will be evaluated immediately and the leader board will be updated.

4 Question 4 – SVM for object detection (20 points + 10 bonus points)

In this question, you will train a SVM and use it for detecting human upper bodies in your favorite TV series The Big Bang Theory. You must use your SVM implementation in either Question 3.1 or 3.2.

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.

4.1 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. We have also extracted some image regions of test images, and the regions are saved as 64×64 jpeg images in `testRegs`. Only small portion of these images correspond to upper bodies.

4.2 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 [2] 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`.

4.3 Helper functions

To help you, a number of utility functions and classes are provided. The most important functions are in `HW2_Utils.m`.

1. Run `HW2_Utils.demo1` to see how to read and display upper body annotation
2. Run `HW2_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 `HW2_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 `HW2_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 `HW2_Utils.cmpFeat` to compute HOG feature vector for an image patch.
6. Use `HW2_Utils.genRsltFile` to generate result file.
7. Use `HW2_Utils.cmpAP` to compute the Average Precision for the result file.

Algorithm 2 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        $\triangleright$ Run UB detection and find negative patches that
                                                  $\triangleright$ 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
```

8. Use `HW2_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.

4.4 What to implement?

1. (5 points) Use the training data in `HW2_Utils.getPosAndRandomNeg()` to train an SVM classifier. Use this classifier to generate a result file (use `HW2_Utils.genRsltFile`) for validation data. Use `HW2_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 `HW2_Utils.getPosAndRandomNeg().` At each iteration, you should remove negative examples that do not correspond to support vectors from the negative set. Use the function `HW2_Utils.detect` on train images to identify hardest negative examples and include them in the negative training set. Use `HW2_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. (10 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. *(5 points)* For this question, you will need to generate a result file for test data using the function `HW2_Utils.genRsltFile`. You will need to submit this file to our evaluation sever (`https://goo.gl/forms/RFAJpiUxJvCdtjTw2`) to receive the AP on test data. Report the AP in your answer file. **Important Note:** You MUST use your Stony Brook ID to name your submission file, i.e., `your_SBU_ID.mat` (e.g., 012345679.mat). Your submission will not be evaluated if you don't use your SBU ID.

5. *(10 bonus points)* Your submitted result file for test data will be automatically entered a competition for fame. We will maintain a leader board at (`http://goo.gl/L1cSxB`) 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 can submit the result as frequent as you want. However, the evaluation server will only evaluate all submissions three times a day, at 11:00am, 5:00pm, and 11:00pm. The system only keeps the recent submission file, and your new submission will override the previous ones. Therefore, you have three chances a day to evaluate your method. The leader board will be updated in 30 minutes after every evaluation.

   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.

5 What to submit?

5.1 Blackboard submission

You will need to submit both your code and your answers to questions on Blackboard. Do not submit the provided data. 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, and it should contain:

1. Answers to Question 1 and 2
2. Answers to Question 3.1 and 3.2, including the requested plots.
3. Answers to Question 4.3, including the requested plots.

5.2 Prediction submission

For Question 3.2.7, you must submit a `.csv` file to get the accuracy through Kaggle (`https://www.kaggle.com/c/hw2-activity-recognition-cse512-spr18/`). A submission file should contain two columns: Id and Class. The file should contain a header and have the following format.

$Id,$	$Class$
1,	1
2,	10
...	...

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

For Questions 4.4.4, 4.4.5, you must submit a mat file to get the AP through `https://goo.gl/forms/RFAJpiUxJvCdtjTw2`. You MUST use your Stony Brook CS email account to submit. A submission file can be automatically generated by `HW2_Utils.genRsltFile`.

6 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] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[2] 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.