

Homework 2 - Practical component

One-versus-all, L2 loss SVM

This part consists of the implementation of a one-versus-all, L2 loss SVM, which is commonly used for multiclass classification. The L2 SVM loss is differentiable and imposes a bigger penalty on points that violate the margin. In the one-versus-all (OVA) approach, we train m binary classifiers, one for each class and during inference time, we select the class which classifies the test data with maximum margin.

Given a training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^p$, $y_i \in \{1, \dots, m\}$ where p is the number of features and m is the number of classes, we would like to minimize the following objective function:

$$\frac{1}{2} \sum_{j'=1}^m \|\mathbf{w}^{j'}\|_2^2 + \frac{C}{n} \sum_{(\mathbf{x}_i, y_i) \in S} \sum_{j'=1}^m l(\mathbf{w}^{j'}; (\mathbf{x}_i, y_i))^2$$

where

$$l(\mathbf{w}^{j'}; (\mathbf{x}_i, y_i)) = \max\{0, 1 - (\langle \mathbf{w}^{j'}, \mathbf{x}_i \rangle) \mathbb{1}\{y_i = j'\}\}$$

and

$$\mathbb{1}\{y_i = j'\} = \begin{cases} 1 & \text{if } y_i = j' \\ -1 & \text{if } y_i \neq j' \end{cases}$$

In order to update the parameters \mathbf{w} of the SVM, gradient descent techniques are used. (Note: in the dataset provided with this assignment, the last element of each row is a dummy element with the value 1. That means there is no separate bias parameter b ; it is implicitly included in \mathbf{w} as the weight for the dummy element.)

The training set for this part can be downloaded from <https://www.dropbox.com/s/nf8zriqjgw02m1a/hw2-cifar-dataset.zip?dl=0>.

1. **Question.** What is the derivative of the regularization term

$$\frac{1}{2} \sum_{j'=1}^m \|\mathbf{w}^{j'}\|_2^2$$

with respect to w_k^j (the k th weight of the weight vector for the j th class)? Show all your work and write your answer in the report.

2. **Question.** What is the derivative of the hinge loss term

$$\frac{C}{n} \sum_{(\mathbf{x}_i, y_i) \in S} \sum_{j'=1}^m l(\mathbf{w}^{j'}; (\mathbf{x}_i, y_i))^2$$

with respect to w_k^j ?

Express your answer in terms of $\mathbf{x}_{i,k}$ (the k th entry of the i th training example \mathbf{x}_i).

Assume that

$$\frac{\partial}{\partial a} \max\{0, a\} = \begin{cases} 1 & \text{if } a > 0 \\ 0 & \text{if } a \leq 0 \end{cases}$$

(This is not exactly true: at $a = 0$, the derivative is undefined. However, for this problem, it's OK to make this assumption.)

3. **Question.** Fill in the following in the code:

- (a) `SVM.make_one_versus_all_labels`: Given an array of integer labels and the number of classes m , this function should create a 2-d array corresponding to the $\mathbb{1}\{y_i = j'\}$ term above. In this array, each row is filled with -1 , except for the entry corresponding to the correct label, which should have the entry 1 . For example, if the array of labels is $[1, 0, 2]$ and $m = 4$, this function would return the following array: $[[-1, 1, -1, -1], [1, -1, -1, -1], [-1, -1, 1, -1]]$. The inputs are y (a numpy array of shape (number of labels,)) and m (an integer representing the number of classes), and the output should be a numpy array of shape (number of labels, m). For this homework, m will be 10, but you should write this function to work for any $m > 2$.
- (b) `SVM.compute_loss`: Given a minibatch of examples, this function should compute the loss function. The inputs are x (a numpy array of shape (minibatch size, 401)), y (a numpy array of shape (minibatch size, 10)), and the output should be the computed

loss, a single float.

- (c) `SVM.compute_gradient`: Given a minibatch of examples, this function should compute the gradient of the loss function with respect to the parameters \mathbf{w} . The inputs are X (a numpy array of shape (minibatch size, 401)), y (a numpy array of shape (minibatch size, 10)), and the output should be the computed gradient, a numpy array of shape (401, 10), the same shape as the parameter matrix \mathbf{w} . (Hint: use the expressions you derived above.)
 - (d) `SVM.infer`: Given a minibatch of examples, this function should infer the class for each example, i.e. which class has the highest score. The input is X (a numpy array of shape (minibatch size, 401)), and the output is $y_inferred$ (a numpy array of shape (minibatch size, 10)). The output should be in the one-versus-all format, i.e. -1 for each class other than the inferred class, and $+1$ for the inferred class.
 - (e) `SVM.compute_accuracy`: Given an array of inferred labels and an array of true labels, this function should output the accuracy as a float between 0 and 1. The inputs are $y_inferred$ (a numpy array of shape (minibatch size, 10)) and y (a numpy array of shape (minibatch size, 10)), and the output is a single float.
4. **Question.** The method `SVM.fit` uses the code you wrote above to train the SVM. After each epoch (one pass through the training set), `SVM.fit` computes the training loss, the training accuracy, the test loss, and the test accuracy.

Plot the value of these four quantities for every epoch for $C = 0.1, 1, 30, 50$. Use 200 epochs, a learning rate of 0.001, and a minibatch size of 5000.

You should have four plots: one for each quantity, with the curves for all four values of C . Include these four plots in your report.