ENGN 2520 Spring 2024
Homework 4

In this assignment you will implement a multiclass SVM to recognize handwritten digits. You will use the data available on the course website.

A multiclass SVM learns functions from $\mathbb{R}^D$ to $\{1, \ldots, k\}$ where k is the number of classes. The classifier is defined by k weight vectors $w_1, \ldots, w_k \in \mathbb{R}^D$ using the classification rule

$$f(x) = \operatorname*{argmax}_y w_y^T x$$

where y ranges from 1 to k. We can think of $w_y^T x$ as a score of class y on example x, and the classifier selects the class with maximum score.

Let $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a set of training examples. We would like to learn weight vectors $w_1, \ldots, w_k$ such that

$$y_i = \operatorname*{argmax}_y w_y^T x_i$$

To obtain a large-margin classifier we look for weight vectors $w_1, \ldots, w_k$ that have low norms and such that the score of class y_i on x_i is above the score of all other classes by at least 1. In analogy to the binary SVM objective function we can define an objective function for the multiclass case as follows

$$E(w_1, \ldots, w_k) = \sum_{j=1}^{k} \frac{1}{2} ||w_j||^2 + C \sum_{i=1}^{n} L((w_1, \ldots, w_k), (x_i, y_i)).$$

where the loss of $(w_1, \ldots, w_k)$ on (x, y) is a multiclass version of the hinge loss

$$L((w_1, \ldots, w_k), (x, y)) = \begin{cases} 0 & \text{if } w_y^T x > (\max_{y' \neq y} w_{y'}^T x) + 1 \\ (\max_{y' \neq y} w_{y'}^T x) + 1 - w_y^T x & \text{otherwise} \end{cases}$$

Problem 1

In this problem you will derive the gradient of L on a fixed example (x, y).

Let $\hat{y} = \operatorname{argmax}_{y' \neq y} w_{y'}^T x$.

Let $w_{j,l}$ be the l-th entry in w_j.

Let x_l be the l-th entry in x.
(a) Suppose $w_y^T x > w_{\hat{y}}^T x + 1$. What is

$$\frac{\partial L((w_1, \ldots, w_k), (x, y))}{\partial w_{j,l}}$$

(b) Suppose $w_y^T x < w_{\hat{y}}^T x + 1$ and $j = y$. What is

$$\frac{\partial L((w_1, \ldots, w_k), (x, y))}{\partial w_{j,l}}$$

(c) Suppose $w_y^T x < w_{\hat{y}}^T x + 1$ and $j = \hat{y}$. What is

$$\frac{\partial L((w_1, \ldots, w_k), (x, y))}{\partial w_{j,l}}$$

(d) Suppose $w_y^T x < w_{\hat{y}}^T x + 1$ and $j \neq y$ and $j \neq \hat{y}$. What is

$$\frac{\partial L((w_1, \ldots, w_k), (x, y))}{\partial w_{j,l}}$$

Problem 2

A multiclass SVM partitions the feature space $\mathbb{R}^D$ into decision regions using several linear functions, one per class.

(a) Show that when the number of classes equals 2, a multiclass SVM partitions the feature space using a single hyperplane. Let w_1 and w_2 be the weight vectors associated with a multiclass SVM classifier. Show that the classifier defined by (w_1, w_2) is a linear classifier defined by a single weight vector w. What is the relationship between w and (w_1, w_2)?

(b) Now consider multiclass SVM with K classes. What do the boundaries between decision regions look like? Draw an example 3 class problem in the plane ($D = 2$) where a multiclass SVM is used to correctly classify a dataset. You should indicate which training points belong to each class and indicate the decision regions and decision boundaries defined by the SVM.

Problem 3

Implement a matlab function for evaluating $E(w)$ and $\nabla E(w)$ for a fixed model $w = (w_1, \ldots, w_k)$ and a fixed training set.

Problem 4

To optimize E we are going to use gradient descent. This involves starting with a model w_0 and taking T steps in the negative gradient direction. In each step we set

$$w_t = w_{t-1} - r\nabla E(w_{t-1})$$

where $r \in \mathbb{R}$ is a learning rate.

(a)

Implement gradient descent for optimizing the multiclass SVM objective function. The input to the optimization function should be: the training set, the value of C, the step size r, and the number of gradient descent interations T.

Your implementation can simply use a fixed small step size r.

(b)

Use gradient descent to train a multiclass SVM model for digit recognition. You should use the data available on the class website.

IMPORTANT: for this assignment you should augment each example x with a constant feature 1 to learn decision boundaries (hyperplanes) that don't go through the origin.

You will need to experiment with different values of C, r and T.

You should train models with different values of C such as $0.01, 0.1, 1, 10, 100$ and report the results that lead to the best performance at test time.

You need to pick r small enough so that the objective function goes down consistently over time, but not too small so as to require too many steps to reach a good value. For a given choice of r and T you should look at a plot of the objective function over time. The value of T should be large enough that in the end the plot is more or less flat. The value of r should be small enough so that the objective is going down consistently over time.

Show a plot of the objective function value over time for one of your experiments.

(c)

Consider the model you obtained using the different values of C. What fraction of the test digits were correctly classified for each choice of C?

For the choice of C leading to the minimum number of errors at test time, compute a 10x10 confusion matrix where entry (i,j) specifies how often digit i was classified as digit j.

(d)

Consider the choice of C leading to the minimum number of errors at test time. Make a visualization of the model for each digit by drawing each vector w_y as a 28x28 image. The brightness of a pixel should depend on the value of the relevant entry in w_y. Note that w_y will have both positive and negative values and you need to take this into account to make a meaningful picture.