CS671 HW2

1 Logistic regression (25 points)

On a dataset $D = \{(\mathbf{x}_k, y_k)\}_{k=1}^n$ ($\|\mathbf{x}_k\|_2 \le 1, y_k \in \{0,1\}$ for all $k = 1, 2, \cdots, n$) , the loss of logistic regression can be written as:

$$L_D(\mathbf{w}) = -\sum_{k=1}^{n} \left[y_k \log\left(\frac{1}{1+\exp(-f_{\mathbf{w}}(\mathbf{x}_k))}\right) + (1-y_k)\log\left(1 - \frac{1}{1+\exp(-f_{\mathbf{w}}(\mathbf{x}_k))}\right)\right], \tag{1}$$

where $f_{\mathbf{w}}(\mathbf{x})$ is a function in $\mathbf{x}$ parametrized by $\mathbf{w}$. Here we consider a non-linear logistic regression where $f_{\mathbf{w}}(\mathbf{x})$ satisfies the following conditions:

1. (**C1**) $f_{\mathbf{w}}$ is twice continuously differentiable over $\mathbb{R}^d$, i.e., $\frac{\partial^2 f_{\mathbf{w}}(\mathbf{x})}{\partial \mathbf{w}_i \partial \mathbf{w}_j}$ are well-defined and continuous for all $i, j = 1, 2, \cdots, d$ and $\mathbf{x}, \mathbf{w} \in \mathbb{R}^d$. This means the Hessian matrix $\nabla^2 f_{\mathbf{w}}(\mathbf{x})$ is symmetric for all $\mathbf{x}, \mathbf{w} \in \mathbb{R}^d$.
2. (**C2**) For all $\mathbf{x}$ such that $\|\mathbf{x}\|_2 \le 1$, $|f_{\mathbf{w}}(\mathbf{x})|$ satisfies $\|\nabla f_{\mathbf{w}}(\mathbf{x})\|_2^2 \le g$ for some constant g.
3. (**C3**) For any $\mathbf{x}$ such that $\|\mathbf{x}\|_2 \le 1$ and any $\mathbf{w} \in \mathbb{R}^d$, the eigenvalues of the Hessian matrix $\nabla^2 f_{\mathbf{w}}(\mathbf{x})$ are all absolutely bounded by u for some positive constant u. In other words, if λ is an eigenvalue of $\nabla^2 f_{\mathbf{w}}(\mathbf{x})$, then $|\lambda| \le u$ for some constant $u > 0$, i.e., $\|\nabla^2 f_{\mathbf{w}}(\mathbf{x})\|_2 \le u$.

With this not-necessarily-linear model $f_{\mathbf{w}}(\mathbf{x})$, the objective in (1) is not necessarily convex. Now we consider the following L_2-regularized new objective

$$\widetilde{L}_{D,\alpha}(\mathbf{w}) = L_D(\mathbf{w}) + \frac{\alpha}{2}\|\mathbf{w}\|_2^2. \tag{2}$$

Can this new objective be convex for some α? If yes, name an α value such that $\widetilde{L}_{D,\alpha}(\mathbf{w})$ is convex and justify your claim. If no, why not?

Hint: Compute the Hessian matrix of $\widetilde{L}_{D,\alpha}(\mathbf{w})$ and try to find an α value such that this Hessian matrix is positive semi-definite. Recall $\sigma(z) = \frac{1}{1+\exp(-z)}$. The following properties of σ may be useful:

- $1 - \sigma(z) = 1 - \frac{1}{1+\exp(-z)} = \frac{\exp(-z)}{1+\exp(-z)} = \frac{1}{1+\exp(z)} = \sigma(-z)$.
- $\frac{d\sigma(z)}{dz} = \exp(-z)(1+\exp(-z))^{-2} = \frac{1}{1+\exp(-z)}\frac{\exp(-z)}{1+\exp(-z)} = \sigma(z)(1-\sigma(z))$.

2 SVM with a squared loss (25 points)

Given a dataset $\{x_i, y_i\}_{i=1}^n, x_i \in \mathbb{R}^p, y_i \in \{-1, +1\}$, the soft (linear) SVM learns the classifier by the following optimization problem:

$$\min L(w,\xi) = \frac{1}{2}w^\top w + C\sum_{i=1}^n \xi_i, \quad s.t. \begin{cases} y_i[w^\top x_i + b] \ge 1 - \xi_i \\ \xi_i \ge 0 \end{cases}.$$

Alternatively, we can construct an SVM that uses the sum of squared errors within the objective function of the standard SVM. This changes the optimization problem to:

$$\min L(w,b,e) = \frac{1}{2}w^\top w + \frac{1}{2}C\sum_{i=1}^n e_i^2, \quad s.t. \quad y_i - w^\top x_i = e_i.$$

Using similar techniques (KKT conditions) to the standard soft-margin SVM, write out w as a function of the data points.

3 Linear SVM (25 points)

For this problem, you need will need to train several linear SVM models on a toy classification dataset as well as on a handwritten digits dataset (MNIST: http://yann.lecun.com/exdb/mnist).
You will program in Python3[1] and will need the packages scikit-learn (https://pypi.org/project/scikit-learn/)[2] and LIBSVM (https://pypi.org/project/libsvm/)[3].
You can find more documentation on LIBSVM in the following resources:

- Python documentation: https://github.com/cjlin1/libsvm/tree/master/python
- Practical guide: https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
- FAQs: https://www.csie.ntu.edu.tw/~cjlin/libsvm/faq.html

We recommend you to use Conda to manage python packages and edit an iPython Notebook. The installation guide can be found here.
We have also provided the starting code in a notebook **(linear_svm.ipynb)**. Please answer the questions, which are copied in the notebook, in the same write-up as the previous problems. You also need to upload the completed iPython Notebook to Gradescope, under the Homework2 Coding assignment.

1. (a) Generate a training set of size 100 with 2D features (X) drawn at random as follows:
      - $Xneg \sim \mathcal{N}([-5, -5], 5*I_2)$ and corresponds to negative labels (-1)
      - $Xpos \sim \mathcal{N}([5, 5], 5*I_2)$ and corresponds to positive labels (+1)

      Accordingly, $X = [Xneg, Xpos]$ is a 100×2 array, and Y is a 100×1 array of values $\in \{-1, 1\}$.
      Draw a scatter plot of the full training dataset with the points colored according to their labels.

   (b) Train a support vector machine on the data with $C = 1$ and draw the decision boundary line that separates the positives and negatives. Mark the support vectors separately (e.g., a circle around the point). [4]

   (c) Draw a line that separates the data for 8 different C values (between perhaps 10^{-5} to 10^{7}) for a total of 8 lines to be displayed on a scatter plot of the training data. Plot the number of support vectors vs. C (plot x-axis on a log scale). How does the number of support vectors change as C increases and why does it change like that? [5]

   (d) Now rescale the features (X) to the [0,1] range and repeat the steps of the previous question (3.1.3) over the same range of C values. Are the decision boundaries closer or further apart than those from the previous question? What does the slope change imply about the change of each variable's importance.[6]

2. Let us perform multi-class classification on MNIST. Now we will use support vector machines to classify handwritten digits from 0-9 based on the pixel values given as features.

   (a) Load in the MNIST data using from the provided *mnist-original.mat* file on Sakai. Follow the instructions in the notebook on how to subsample the train and test datasets.

   (b) Draw 3 different digits using pyplot.imshow().

   (c) Train the support vector machine classifier with a linear kernel on the first 5000 datapoints and test the accuracy on the following 5000 points. Plot test accuracy and the number of support vectors (two separate plots) vs. C for $C = 10^{-12}$ 10^{12} (plot 7 points or more with the x-axis on a log scale).

[1] version >=3.8.0
[2] version >=0.20.0
[3] version >=3.10
[4] You should use the libsvm.svmutil functions with the kernel_type set to 0, indicating a linear kernel and svm_type set to 0 indicating C-SVC. Also note that the support_vector coefficients returned by the LIBSVM model are the dual coefficients.
[5] You might prefer to use the command-line style of svm_parameter initialization such as: **svm_parameter('-s 0 -t 0')** to indicate a linear kernel and C-SVC as the SVM type.
[6] E.g. if a single variable/dimension dominated the other dimensions before rescaling, how would that have affected the slope of the decision boundary and how would that change post-rescaling?

4 AdaBoost (25 points)

1. Assume that the weak learning assumption holds (on the training set), show that the boosted model eventually classifies the training set perfectly (i.e. training error equals to zero).
   Hint: look at the class notes.
2. Suppose that the points are weighted differently in the training set. More specifically, the training set is $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where each point $(\mathbf{x}_i, y_i)$ in the training set has weight w_i and the objective is defined as

$$R^{train}(\boldsymbol{\lambda}) = \sum_{i=1}^{n} w_i e^{-(\boldsymbol{M\lambda})_i}$$

   Derive the weighted version of AdaBoost based on this new objective.
   Hint: the change to AdaBoost is surprisingly small.
3. In this problem, you will implement your own boosted decision stumps (trees with two leafs). Follow the skeleton code in ***adaboost.ipynb*** and fill in the holes in the code (marked with "YOUR CODE HERE"). Make sure that you pass all check points and answer the questions raised in the skeleton code. The skeleton code is in Python 3.