CS534 — Implementation Assignment 2 Reference Information

Logistic regression with L2 and L1 regularizations

For this assignment, you need to implement and test logistic regression, which learns from a set of N training examples $\{\mathbf{x}_i, y_i\}_{i=1}^N$ an weight vector $\mathbf{w}$ that maximize the log likelihood objective. You will examine two different regularization methods: L2 (ridge) and L1 (Lasso).

Data. This dataset consists of health insurance customer demographics, as well as collected information related to the customers' driving situation. Your goal is to use this data to predict whether or not a customer may be interested in purchasing vehicular insurance as well (this is your "Response" variable). The dataset description (dictionary) is included and provided on canvas. We provide to you a training set: **IA2-train.csv** and a validation set: **IA2-dev.csv**. Furthermore, for experimenting with noise in data, we also provided IA2-train-noisy.csv, which is created by flipping the label of 30% of training examples (randomly selected). You will use this noisy data to explore the effect of random label noise on learning.

Preprocessing Information We have already done some pre-processing of the data. In particular, we have treated [**Gender, Driving_License, Region_Code, Previously_Insured, Vehicle_Age, Vehicle_Damage, Policy_Sales_Channel**] as categorical features and converted those into one-hot vectors. Additionally, the dataset has been pre-processed to be relatively class balanced (close to the same number of 1's and 0's for Response). This was not the case in the original raw data, but we downsampled for easier training purposes. Handling class imbalance is beyond the scope of this assignment, but it is a common and important problem in real-world data. There are 3 numerical features, **these need to be normalized so that your training can converge faster.**

Guidelines for training For this assignment, you are responsible for finding the right learning rate that works for your training. Note that you will need to adjust the learning rate for different regularization parameter λ values. In the colab notebook, some basic range information is provided.
A common strategy for tuning the learning rate is via exponential search:

1. Start with an initial guess for the learning rate γ.
2. If the loss function diverges, reduce γ by a factor of 2, 5, or 10.
3. If the loss converges too slowly, increase γ by a similar factor.
4. repeat step 2 or 3 (using different factors as needed) until you achieve reasonable convergence.

What to expect for this data. For reference, the best training accuracy for this data set will hover around 0.8, and the validation accuracy will be slightly lower than that. Given that we are experimenting with a very simple learning algorithm that has low complexity. The difference we observe will tend to be small. If you don't run your algorithm to close to convergence (e.g., using a stringent convergence criterion), you might not be able to see clear performance difference. It is also important to plot the results with a proper value range to clearly observe any variations in performance.

Algorithm 1: Logistic regression with L2 (Ridge) regularization. Recall, Logistic regression with L2 regularization aims to minimize the following loss function[1]:

$$\frac{1}{N}\sum_{i=1}^{N}\left[-y_i \log \sigma(\mathbf{w}^T\mathbf{x}_i) - (1-y_i)\log(1-\sigma(\mathbf{w}^T\mathbf{x}_i)\right] + \frac{\lambda}{2}\sum_{j=1}^{d} w_j^2 \tag{1}$$

[1]In class we presented the log likelihood function as the objective to maximize. It is, however, more common to put a negative in the front and turn it into a loss function, which is called "negative loglikelihood".

See the following algorithm for batch gradient descent [2] optimization of Equation 1.

Algorithm 1: Gradient descent for Ridge logistic regression

```
Input: $\{(\mathbf{x}_i, y_i)_{i=1}^N\}$(training data), $\alpha$(learning rate), $\lambda$(regularization parameter)
Output: learned weight vector $\mathbf{w}$
Initialize $\mathbf{w} = \mathbf{0}$;
while not converged do
    $\mathbf{w} \leftarrow \mathbf{w} + \frac{\alpha}{N} \sum_{i=1}^N (y_i - \sigma(\mathbf{w}^T \mathbf{x}_i))\mathbf{x}_i$ ;    // normal gradient without the L2 norm
    for $j = 1$ to $d$ do
        $w_j \leftarrow w_j - \alpha\lambda w_j$ ;    // L2 norm contribution excluding $w_0$ for the dummy feature
    end
end
```

Algorithm 2: Logistic Regression with L1(Lasso) regularization You will also need to implement L1 regularized logistic regression. Recall that the loss function for L1 regularized logistic regression is:

$$\frac{1}{N} \sum_{i=1}^N \left[-y_i \log \sigma(\mathbf{w}^T \mathbf{x}_i) - (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i) \right] + \lambda \sum_{j=1}^d |w_j| \tag{2}$$

The following algorithm minimizes Equation 2 via a procedure called proximal gradient descent. For L_1 regularized loss functions, Proximal gradient descent often leads to substantially faster convergence than simple gradient (or subgradient in this case since the L_1 norm is not differentiable everywhere) descent. You can refer to Ryan Tibshirani's note (`http://www.stat.cmu.edu/~ryantibs/convexopt/lectures/prox-grad.pdf`) for an introduction to this method.

Algorithm 2: Proximal gradient descent for LASSO logistic regression

```
Input: $\{(\mathbf{x}_i, y_i)_{i=1}^N\}$(training data), $\alpha$ (learning rate), $\lambda$ (regularization parameter)
Output: learned weight vector $\mathbf{w}$
Initialize $\mathbf{w} = \mathbf{0}$;
while not converged do
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha \frac{1}{N} \sum_{i=1}^N (y_i - \sigma(\mathbf{w}^T \mathbf{x}_i))\mathbf{x}_i$ ;    // normal gradient descent without the L1 norm
    for $j = 1$ to $d$ do
        $w_j \leftarrow sign(w_j) \max(|w_j| - \alpha\lambda, 0)$ ;    // soft thresholding each $w_j$: if $|w_j| < \alpha\lambda$, $w_j \leftarrow 0$
    end
end
```

[2] Our lecture presented gradient ascent, here since we are working with loss function, we use gradient descent instead.