MATH 156S HOMEWORK 3

Due Apr. 21

Say we have training examples $(\boldsymbol{\phi}(\mathbf{x}_1), y_1), \dots, (\boldsymbol{\phi}(\mathbf{x}_N), y_N)$ for binary classification. Recall that the probit regression training is done by solving the following optimization problem

$$\hat{\mathbf{w}} = \operatorname{argmin}_{\mathbf{w}\in\mathbb{R}^p} \left[\ell_{PR}(\mathbf{w}) := -\sum_{i=1}^{N} y_i \log \Psi(\boldsymbol{\phi}(\mathbf{x}_i)^T \mathbf{w}) + (1-y_i) \log \Psi(-\boldsymbol{\phi}(\mathbf{x}_i)^T \mathbf{w}) \right], \tag{1}$$

where $\Psi(x)$ denotes the CDF of standard normal random variable.

Exercise 1 (Convexity of the probit regression loss)**.** Let $\ell_{PR}(\mathbf{w})$ denote the probit regression loss defined in (1). Let $\psi(x) = (2\pi)^{-1/2}\exp(-x^2/2)$ denote the PDF of standard normal distribution and let $\Psi(x) = \int_{-\infty}^{x} \psi(t)\,dt$ denote the corresponding CDF. Let $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), \dots, \boldsymbol{\phi}(\mathbf{x}_N)] \in \mathbb{R}^{p\times N}$ be the design matrix.

(i) Show that

$$\begin{aligned}
\nabla_{\mathbf{w}} \ell_{PR}(\mathbf{w}) &= \sum_{i=1}^{N} \psi(\boldsymbol{\phi}(\mathbf{x}_i)^T \mathbf{w}) \left[\frac{1-y_i}{\Psi(-\boldsymbol{\phi}(\mathbf{x}_i)^T\mathbf{w})} - \frac{y_i}{\Psi(\boldsymbol{\phi}(\mathbf{x}_i)^T\mathbf{w})} \right] \boldsymbol{\phi}(\mathbf{x}_i) && (2)\\
&= \boldsymbol{\Phi}(\mathbf{Q}_1 - \mathbf{Q}_2), && (3)
\end{aligned}$$

where

$$\mathbf{Q}_1 = \begin{bmatrix} \frac{(1-y_1)\,\psi(\boldsymbol{\phi}(\mathbf{x}_1)^T\mathbf{w})}{\Psi(-\boldsymbol{\phi}(\mathbf{x}_1)^T\mathbf{w})} & \cdots & \frac{(1-y_N)\,\psi(\boldsymbol{\phi}(\mathbf{x}_N)^T\mathbf{w})}{\Psi(-\boldsymbol{\phi}(\mathbf{x}_N)^T\mathbf{w})} \end{bmatrix}^T, \quad \mathbf{Q}_2 = \begin{bmatrix} \frac{y_1\,\psi(\boldsymbol{\phi}(\mathbf{x}_1)^T\mathbf{w})}{\Psi(\boldsymbol{\phi}(\mathbf{x}_1)^T\mathbf{w})} & \cdots & \frac{y_N\,\psi(\boldsymbol{\phi}(\mathbf{x}_N)^T\mathbf{w})}{\Psi(\boldsymbol{\phi}(\mathbf{x}_N)^T\mathbf{w})} \end{bmatrix}^T \in \mathbb{R}^N. \tag{4}$$

(ii) Show the following properties of PDF and CDF of standard normal distribution:

$$\psi(x) = \psi(-x), \qquad \Psi(-x) = 1-\Psi(x), \qquad \psi'(x) = -x\psi(x), \tag{5}$$

$$\frac{\partial}{\partial x}\frac{\psi(x)}{\Psi(x)} = \frac{\psi(x)\left[-x\Psi(x) - \psi(x)\right]}{\Psi(x)^2}, \qquad \frac{\partial}{\partial x}\frac{\psi(x)}{\Psi(-x)} = \frac{\psi(x)\left[-x\Psi(-x) + \psi(x)\right]}{\Psi(-x)^2}. \tag{6}$$

(iii) Show that

$$\frac{\partial(\mathbf{Q}_1^T - \mathbf{Q}_2^T)}{\partial \mathbf{w}} = \begin{bmatrix} \uparrow & & \uparrow \\ G_1\boldsymbol{\phi}(\mathbf{x}_1) & \cdots & G_N\boldsymbol{\phi}(\mathbf{x}_N) \\ \downarrow & & \downarrow \end{bmatrix} = \boldsymbol{\Phi}\mathbf{D}, \tag{7}$$

where for $z_i := \boldsymbol{\phi}(\mathbf{x}_i)^T\mathbf{w}$, we denote

$$G_i := (1-y_i)\frac{\psi(z_i)\left[-z_i\Psi(-z_i) + \psi(z_i)\right]}{\Psi(-z_i)^2} + y_i\frac{\psi(z_i)\left[z_i\Psi(z_i) + \psi(z_i)\right]}{\Psi(z_i)^2}. \tag{8}$$

and $\mathbf{D}$ is the $N\times N$ diagonal matrix with diagonal entries $\mathbf{D}_{ii} = G_i$.

(iv) Show that

$$\mathbf{H} := \frac{\partial^2 \ell_{PR}(\mathbf{w})}{\partial\mathbf{w}\,\partial\mathbf{w}^T} = \nabla_{\mathbf{w}}(\mathbf{Q}_1^T - \mathbf{Q}_2^T)\boldsymbol{\Phi}^T = \boldsymbol{\Phi}\mathbf{D}\boldsymbol{\Phi}^T \in \mathbb{R}^{p\times p}. \tag{9}$$

(v) (Optional*) Argue that the Hessian $\mathbf{H}$ is positive semi-definite. Conclude that the probit regression loss ℓ_{PR} is convex. (Hint: $\mathbf{D}$ is a diagonal matrix with positive diagonal entries. This was clear for logistic regression. Here, one can show the funtions $-z\Psi(-z)+\psi(z)$ and $-z\Psi(z)+\psi(z)$ always assume positive values by some calculus argument. See [ZL12, Lem. 1])

Exercise 2. The gradient descent (GD) algorithm for probit regression training

$$\mathbf{w}_{t+1} \longleftarrow \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} \ell_{PR}(\mathbf{w}_t) \tag{10}$$

is implemented as the python function 'fit_PR_GD' in the Jupyter notebook provided in the course repository.

```
def fit_PR_GD(Y, H, W0=None, sub_iter=100, stopping_diff=0.01):
        '''
        Convex optimization algorithm for Probit Regression using Gradient Descent
        Y = (n x 1), H = (p x n) (\Phi in lecture note), W = (p x 1)
        Logistic Regression: Y ~ Bernoulli(Q), Q = Probit(H.T @ W)
        '''
        if W0 is None:
            W0 = np.random.rand(H.shape[0],1) #If initial coefficients W0 is None, randomly initialize

        W1 = W0.copy()
        i = 0
        grad = np.ones(W0.shape)
        while (i < sub_iter) and (np.linalg.norm(grad) > stopping_diff):
            Q = norm.pdf(H.T @ W1) * ( (1-Y)/norm.cdf(-H.T @ W1) - Y/norm.cdf(H.T @ W1) )
            grad = H @ Q
            W1 = W1 - (np.log(i+1) / (((i + 1) ** (0.5)))) * grad
            i = i + 1
            # print('iter %i, grad_norm %f' %(i, np.linalg.norm(grad)))
        return W1
```

FIGURE 1. A python implementation of the gradient descent algorithm for logistic regression training (10).

(i) Modify the code (or make your own function) so that each gradient descent step (10), the current value of objective function ℓ_{PR} in (1) is printed. Test it on the MNIST example with digits $['0','1']$. Does the loss monotonically decrease?

(ii) The implemented version of GD for (10) uses the step size $\eta_t = \gamma \log(t+1)/\sqrt{(t+1)}$ with $\gamma = 1$. Test the same step sizes for $\gamma = 10$ and also $\eta_t = 1$ and $\eta_t = 1/(t+1)$. Test it on the MNIST example with digits $['0','1']$. Is there any difference in the way the loss function decrease?

(iii) If we use the step size of the form $\eta_t = \gamma t^{-\delta}$ for $0 \le \delta \le 1$, what would be your optimal choice of the hyperparameters γ and δ? (There is no answer and trying out a few choices and make your observation.)

Exercise 3. In this exercise, we investigate the 'robustness' of the logistic and the probit regression for binary classification.

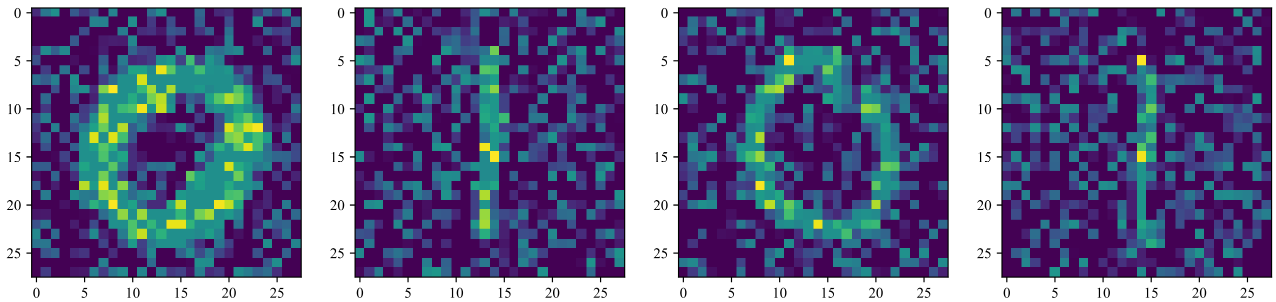

FIGURE 2. Corrupted MNIST images where Uniform([0, 1]) is added to 50% of randomly chosen pixels.

(i) Modify the data preprocessing function "sample_binary_MNIST" so that it have the option to add noise with given ratio. Namely, if 'noise_ratio' is 0.5, then it will add independent Uniform([0, 1]) variables to 50% of randomly chosen pixels. (See Figure 2)

(ii) For logistic regression, re-generate figures similar to Figure 7 in lecture notes with varying noise ratios in $\{0.1, 0.5, 0.9\}$.

(iii) For probit regression, re-generate figures similar to Figure 13 in lecture notes with varying noise ratios in $\{0.1, 0.5, 0.9\}$.

(iv) From the previous experiments, compare the robustness of LR and PR against random noise you gave. Can you conclude that one is significantly more robust than the other?

REFERENCES

[ZL12] Songfeng Zheng and Weixiang Liu, *Functional gradient ascent for probit regression*, Pattern recognition **45** (2012), no. 12, 4428–4437.