



Homework 2  
Due 5pm, Monday, March 18, 2024

**Problem 1:** *Logistic regression via SGD.* Use SGD to solve the logistic regression optimization problem

$$\underset{\theta \in \mathbb{R}^p}{\text{minimize}} \quad \frac{1}{N} \sum_{i=1}^N \log(1 + \exp(-Y_i X_i^\top \theta)),$$

where  $X_1, \dots, X_N \in \mathbb{R}^p$  and  $Y_1, \dots, Y_N \in \{-1, 1\}$ . Use the data

```
N, p = 30, 20
np.random.seed(0)
X = np.random.randn(N, p)
Y = 2*np.random.randint(2, size = N) - 1
```

where  $X_1^\top, \dots, X_N^\top$  are the rows of  $X$ .

**Solution.** See the file `p1_sol.py`. ■

**Problem 2:** *SVM via SGD.* Use SGD to solve the non-differentiable SVM optimization problem

$$\underset{\theta \in \mathbb{R}^p}{\text{minimize}} \quad \frac{1}{N} \sum_{i=1}^N \max\{0, 1 - Y_i X_i^\top \theta\} + \lambda \|\theta\|^2,$$

where  $X_1, \dots, X_N \in \mathbb{R}^p$ ,  $Y_1, \dots, Y_N \in \{-1, 1\}$ , and  $\lambda = 0.1$ . Use the data of Problem 1. Empirically, does the SGD ever encounter a point of non-differentiability?

**Solution.** See the file `p2_sol.py`. ■

**Problem 3:** Consider the data generated by the Python code

```
N=30
np.random.seed(0)
X = np.random.randn(2, N)
y = np.sign(X[0, :]**2 + X[1, :]**2 - 0.7)
theta = 0.5
c, s = np.cos(theta), np.sin(theta)
X = np.array([[c, -s], [s, c]]) @ X
X = X + np.array([[1], [1]])
```

Observe (by plotting) that the data is not linearly separable. Consider the transformation

$$\phi\left(\begin{bmatrix} u \\ v \end{bmatrix}\right) = \begin{bmatrix} 1 \\ u \\ u^2 \\ v \\ v^2 \end{bmatrix}.$$

Using the logistic regression or SVM, show that the data  $\phi(X_1), \dots, \phi(X_N) \in \mathbb{R}^5$  with labels  $Y_1, \dots, Y_N \in \{-1, +1\}$  is linearly separable. Visualize in  $\mathbb{R}^2$  the data and the decision boundary.

*Hint.* Visualize the decision boundary given by

```
0 == w[0]+w[1]*x+w[2]*(x**2)+w[3]*y+w[4]*(y**2)
```

with the code

```
xx = np.linspace(-4, 4, 1024)
yy = np.linspace(-4, 4, 1024)
xx, yy = np.meshgrid(xx, yy)
Z = w[0] + (w[1] * xx + w[2] * xx**2) + (w[3] * yy + w[4] * yy**2)
plt.contour(xx, yy, Z, 0)
```

*Remark.* This is the basis of kernel methods.

**Solution.** See the file `p3_sol.py`. ■

**Problem 4: Nonnegativity of KL-divergence.** A set  $C \subseteq \mathbb{R}^m$  is said to be convex if

$$x_1, x_2 \in C \Rightarrow \eta x_1 + (1 - \eta)x_2 \in C, \quad \forall \eta \in (0, 1).$$

A function  $\varphi: C \rightarrow \mathbb{R}$  is said to be convex if  $C \subseteq \mathbb{R}^m$  is convex and

$$\varphi(\eta x_1 + (1 - \eta)x_2) \leq \eta \varphi(x_1) + (1 - \eta)\varphi(x_2), \quad \forall x_1, x_2 \in C, \eta \in (0, 1).$$

Jensen's inequality [1] states that if  $X \in C$  is a random variable and  $\varphi$  is convex, then

$$\varphi(\mathbb{E}[X]) \leq \mathbb{E}[\varphi(X)].$$

Use this to show that

$$D_{\text{KL}}(p||q) \geq 0$$

for any probability mass functions  $p, q \in \mathbb{R}^n$ .

*Hint.* First show that  $-\log(x)$  is a convex function.

**Solution.** We first show that  $x \mapsto -\log(x)$  is a convex function on a convex set  $C = \{x \in \mathbb{R} \mid x > 0\}$ . Let  $x_3 = \eta x_1 + (1 - \eta)x_2$  for any  $x_1, x_2 \in C$  and  $\eta \in (0, 1)$ , and also let  $\varphi = -\log$ . Since  $\varphi'(x) = -1/x$  is a strictly increasing function on  $C$ , we have

$$\begin{aligned} \varphi(x_2) - \varphi(x_3) &= \int_{x_3}^{x_2} \varphi'(x) dx > \int_{x_3}^{x_2} \varphi'(x_3) dx = \varphi'(x_3)(x_2 - x_3) \\ \varphi(x_1) - \varphi(x_3) &= - \int_{x_1}^{x_3} \varphi'(x) dx > - \int_{x_1}^{x_3} \varphi'(x_3) dx = \varphi'(x_3)(x_1 - x_3) \end{aligned}$$

multiplying the first inequality by  $(1 - \eta)$  and the second by  $\eta$  and summing them gives us

$$(1 - \eta)\varphi(x_2) + \eta\varphi(x_1) - \varphi(x_3) > \varphi'(x_3) \{(1 - \eta)(x_2 - x_3) + \eta(x_1 - x_3)\} = \varphi'(x_3) \cdot 0 = 0,$$

which is the strict convexity inequality.

Now we prove that  $D_{\text{KL}}(p||q) \geq 0$ . If there is an  $i$  such that  $q_i = 0$  and  $p_i > 0$ , then  $D_{\text{KL}}(p||q) = \infty$ . Now assume that  $q_i > 0$  for all  $i$  such that  $p_i > 0$ . Then

$$D_{\text{KL}}(p||q) = \mathbb{E}_I [-\log(q_I/p_I)] \geq -\log(\mathbb{E}_I [q_I/p_I]) = -\log\left(\sum_{i=1}^n q_i\right) = -\log(1) = 0.$$

(The assumption is used to ensure the expectation is finite and therefore well-defined.) ■

**Problem 5: Positivity of KL-divergence.** A function  $\varphi: C \rightarrow \mathbb{R}$  is said to be *strictly convex* if  $C \subseteq \mathbb{R}^m$  is convex and

$$\varphi(\eta x_1 + (1 - \eta)x_2) < \eta\varphi(x_1) + (1 - \eta)\varphi(x_2), \quad \forall x_1, x_2 \in C, x_1 \neq x_2, \eta \in (0, 1).$$

Strict Jensen's inequality states that if  $X \in C$  is a non-constant random variable and  $\varphi$  is strictly convex, then

$$\varphi(\mathbb{E}[X]) < \mathbb{E}[\varphi(X)].$$

Use this to show that

$$D_{\text{KL}}(p||q) > 0$$

for any probability mass functions  $p, q \in \mathbb{R}^n$  such that  $p \neq q$ .

**Solution.** If  $p \neq q$ , then the random variable  $q_I/p_I$ , where  $I$  is a random index with  $\mathbb{P}(I = i) = p_i$ , is not a constant variable. By the same reasoning as in the previous problem, we get  $D_{\text{KL}}(p||q) > 0$ . ■

**Problem 6: Differentiating 2-layer neural networks.** Consider the 2-layer neural network

$$f_\theta(x) = u^\top \sigma(ax + b) = \sum_{j=1}^p u_j \sigma(a_j x + b_j),$$

where  $a, b, u \in \mathbb{R}^p$  and  $\theta = (a_1, \dots, a_p, b_1, \dots, b_p, u_1, \dots, u_p) \in \mathbb{R}^{3p}$ . Assume the univariate function  $\sigma: \mathbb{R} \rightarrow \mathbb{R}$  is differentiable. The notation  $\sigma(ax + b)$  means  $\sigma$  is applied elementwise to the vector in  $\mathbb{R}^p$ . Show that

$$\begin{aligned} \nabla_u f_\theta(x) &= \sigma(ax + b) \\ \nabla_b f_\theta(x) &= \sigma'(ax + b) \odot u = \text{diag}(\sigma'(ax + b))u \\ \nabla_a f_\theta(x) &= (\sigma'(ax + b) \odot u)x = \text{diag}(\sigma'(ax + b))ux, \end{aligned}$$

where  $\sigma'(ax + b)$  means the univariate function  $\sigma'$  is applied elementwise to the vector  $ax + b$ ,  $\odot$  denotes the element-wise product, and  $\text{diag}(\cdot)$  denotes the diagonal matrix with the diagonal elements equal to the elements of the input vector.

**Solution.** By applying standard rules from vector calculus, we get

$$\begin{aligned} \frac{\partial f_\theta(x)}{\partial u_j} &= \sigma(a_j x + b_j) \\ \frac{\partial f_\theta(x)}{\partial b_j} &= \frac{\partial f_\theta(x)}{\partial(a_j x + b_j)} \frac{\partial(a_j x + b_j)}{\partial b_j} = u_j \sigma'(a_j x + b_j) \\ \frac{\partial f_\theta(x)}{\partial a_j} &= \frac{\partial f_\theta(x)}{\partial(a_j x + b_j)} \frac{\partial(a_j x + b_j)}{\partial a_j} = u_j \sigma'(a_j x + b_j)x \end{aligned}$$

for  $j = 1, \dots, p$ . Vectorizing these partial derivatives gives us the stated results. ■

**Problem 7: SGD with 2-layer neural networks.** Consider the univariate function

$$f_{\star}(x) = (x - 2) \cos(4x).$$

Let

$$f_{\theta}(x) = \sum_{j=1}^p u_j \sigma(a_j x + b_j),$$

be the same 2-layer neural network as in the previous problem. For this problem, use the sigmoid activation function, i.e.,  $\sigma(x) = (1 + e^{-x})^{-1}$ . Given data  $X_i$  generated as IID unit Gaussians and corresponding labels  $Y_i = f_{\star}(X_i)$  for  $i = 1, \dots, N$ , define loss functions

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N \ell_{\theta}(X_i, Y_i)$$

and

$$\ell_{\theta}(X, Y) = \frac{1}{2} (f_{\theta}(X) - Y)^2.$$

Consider the minimization problem

$$\underset{\theta \in \mathbb{R}^{3p}}{\text{minimize}} \quad \mathcal{L}(\theta).$$

Without using PyTorch (so using NumPy), implement

$$\begin{aligned} i(k) &\sim \text{Uniform}\{1, \dots, N\} \\ \theta^{k+1} &= \theta^k - \alpha \nabla_{\theta} \ell_{\theta}(X_{i(k)}, Y_{i(k)}). \end{aligned}$$

Use the parameters  $K = 10000$ ,  $\alpha = 0.007$ ,  $N = 30$ , and  $p = 50$  and use independent initializations with distributions  $a_j^0 \sim \mathcal{N}(0, 4^2)$ ,  $b_j^0 \sim \mathcal{N}(0, 4^2)$ , and  $u_j^0 \sim \mathcal{N}(0, 0.05^2)$  for  $j = 1, \dots, p$ . (These parameters and initializations are implemented in the starter code `twolayerSGD.py`.) Plot the final trained function with  $f_{\theta^K}(x)$  as a function of  $x$ . How does it compare with  $f_{\star}(x)$ ?

*Remark.* In order to fit the nonlinear function  $f_{\star}$ , it is essential that we use the nonlinear activation function  $\sigma$ ; without it,

$$f_{\theta}(x) = \sum_{j=1}^p u_j (a_j x + b_j),$$

will be linear in  $x$ , and a linear function cannot approximate the nonlinear function  $f_{\star}(x)$  well.

**Solution.** See the file `twolayerSGD_sol.py`. ■

## References

- [1] J. L. W. V. Jensen, Sur les fonctions convexes et les inégalités entre les valeurs moyennes, *Acta Mathematica*, 1906.