MFDNN HW2

MinGyu Shin

Problem 1 : *Logistic Regression via SGD.*

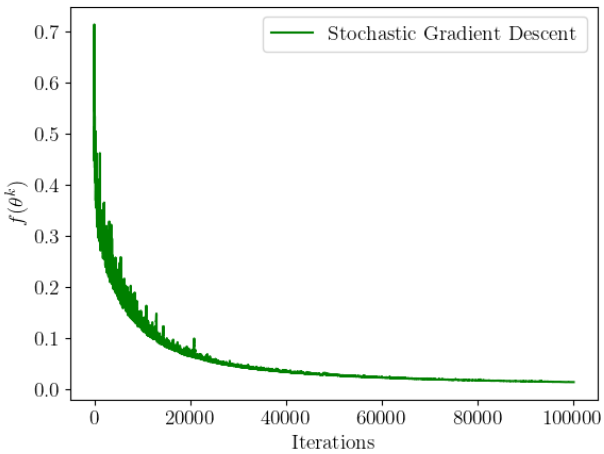


Figure 1: Result

Problem 2 : *SVM via SGD.*

With 100,000 iteration, differentiation at a point of non-differentiability never occurs.

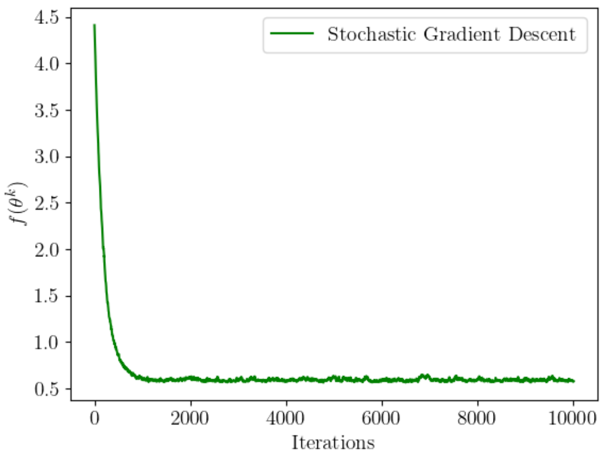
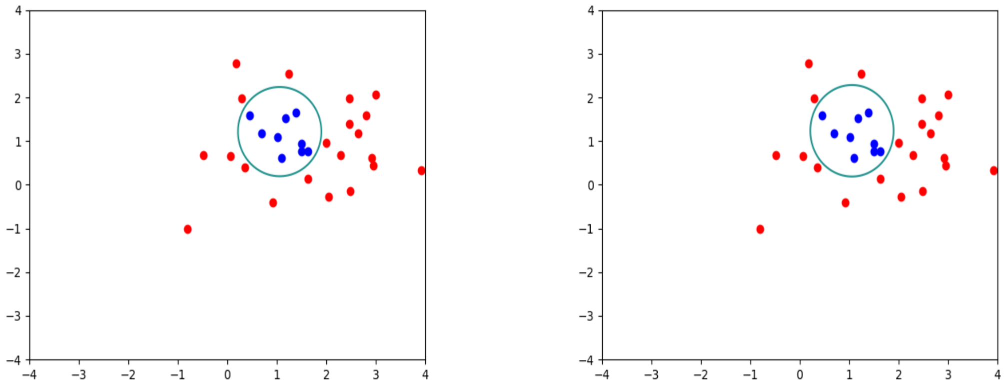


Figure 2: Result

Problem 3 : *Kernel Methods.*

The data is not linearly separable with respect to original dimension. But with the transformation, it is linearly separable.



(a) Logistic Regression (b) SVM without regulirizer

Figure 3: Kernel Methods with SGD

Problem 4 : *Non-negativity of KL-divergence.*

Let $f(x) = -\log x$. A set $C = \{x|x > 0\}$ is a convex set, $f'(x) = -1/x$ is strictly increasing function on C. For $x, y \in C, \lambda \in (0, 1)$, let $z = \lambda x + (1-\lambda)y$. (It's obvious that $z \in C$) Let assume $x < y$, WLOG. (The case $x = y$ is also trivial) Then

$$f(y) = f(z) + \int_z^y f'(t)dt \geq f(z) + \int_z^y f'(z)dt = f(z) - \frac{\lambda(y-x)}{z} \tag{1}$$

$$f(x) = f(z) - \int_x^z f'(t)dt \geq f(z) - \int_x^z f'(z)dt = f(z) - \frac{(1-\lambda)(y-x)}{z} \tag{2}$$

since $f'(a) < f'(z) < f'(b) \forall a \in [x, z), b \in (z, y]$. Consider $\lambda(2) + (1-\lambda)(1)$, $\lambda f(x) + (1-\lambda)f(y) \geq f(\lambda x + (1-\lambda)y)$, thus, f is convex. For probability mass function p, q, if $q_i = 0$ and $p_i \neq 0$ for some i, then $D_{KL}(p||q) = \infty > 0$. So, assume that $q_i \neq 0$ for all i s.t. $p_i \neq 0$.

$$D_{KL}(p||q) = \mathop{\mathbb{E}}_I[\log \frac{p_I}{q_I}] = \mathop{\mathbb{E}}_I[-\log \frac{q_I}{p_I}] \geq -\log \mathop{\mathbb{E}}_I[\frac{q_I}{p_I}] = -\log 1 = 0, \tag{3}$$

where I is a random variable s.t. $P(I = i) = p_i (> 0)$.

Problem 5 : *Positivity of KL-divergence.*

Since if $p \neq q$, p_I/q_I is not a constant random variable, KL-divergence of any probability mass function is positive similarly with Problem 4. (The inequality of (3) is replaced with an strict inequality.)

Problem 6 : *Differentiating 2-layer neural networks.*

$$\frac{\partial}{\partial u_i} f_\theta(x) = \frac{\partial}{\partial u_i} \sum_{j=1}^{P} u_j \sigma(a_j x + b_j) = \sigma(a_i x + b_i)$$

$$\nabla_u f_\theta(x) = \begin{bmatrix} \sigma(a_1 x + b_1) \\ \dots \\ \sigma(a_p x + b_p) \end{bmatrix} = \sigma(ax + b)$$

$$\frac{\partial}{\partial b_i} f_\theta(x) = \frac{\partial}{\partial b_i} \sum_{j=1}^{P} u_j \sigma(a_j x + b_j) = u_i \sigma'(a_i x + b_i)$$

$$\nabla_b f_\theta(x) = \begin{bmatrix} u_1 \sigma'(a_1 x + b_1) \\ \dots \\ u_p \sigma'(a_p x + b_p) \end{bmatrix} = \mathrm{diag}(\sigma'(ax + b))u$$

$$\frac{\partial}{\partial a_i} f_\theta(x) = \frac{\partial}{\partial a_i} \sum_{j=1}^{P} u_j \sigma(a_j x + b_j) = u_i \sigma'(a_i x + b_i)x$$

$$\nabla_a f_\theta(x) = \begin{bmatrix} u_1 \sigma'(a_1 x + b_1)x \\ \dots \\ u_p \sigma'(a_p x + b_p)x \end{bmatrix} = \mathrm{diag}(\sigma'(ax + b))ux$$

Problem 7 : *SGD with 2-layer neural networks.*

$f_{\theta^K}(x)$ gets closer to $f_*(x)$ as $K \uparrow$.

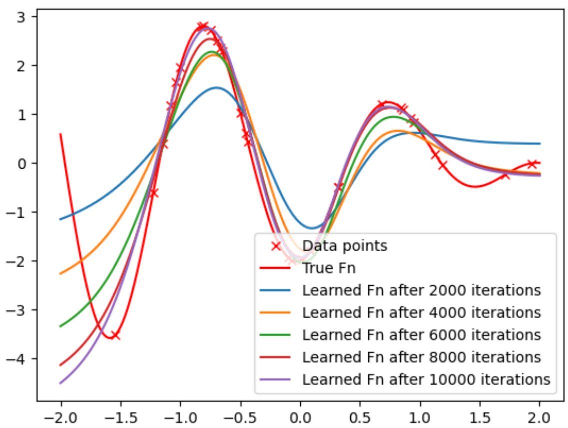


Figure 4: Result