Mathematical Foundations of Deep Neural Networks, M1407.001200
E. Ryu
Spring 2024

Homework 12
Due 5pm, Tuesday, June 18, 2024

Problem 1: *Gradient ascent-descent for robust logistic regression.* Consider the minimax optimization problem

$$\underset{\theta\in\mathbb{R}^p}{\text{minimize}} \quad \underset{\phi\in\mathbb{R}^p}{\text{maximize}} \quad L(\theta,\phi),$$

where

$$L(\theta,\phi)=\frac{1}{N}\sum_{i=1}^{N}\log(1+\exp(-Y_i(X_i-\phi)^\intercal\theta))-\frac{\lambda}{2}\|\phi\|^2,$$

$X_1,\dots,X_N\in\mathbb{R}^p$, $Y_1,\dots,Y_N\in\{-1,1\}$, and $\lambda=30$. 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
lamda = 30
```

where $X_1^\intercal,\dots,X_N^\intercal$ are the rows of `X`. Implement stochastic gradient ascent-descent with starting points θ^0 and ϕ^0 randomly initialized to be zero-mean IID Gaussians with standard deviation 0.1, descent and ascent stepsizes $\alpha=3\times10^{-1}$ and $\beta=10^{-4}$, and 5000 epochs. You may find the starter code `minimax_logistic.py` helpful.

Remark. We can interpret this problem as performing robust logistic regression where there is uncertaintly in the data $X_1,\dots,X_N$.

Solution. See `minimax_logistic_sol.py`. ■

Problem 2: *GAN with non-uniform weights.* Consider the variant of the GAN with non-uniform weights on type I and type II errors:

$$\underset{\theta\in\mathbb{R}^p}{\text{minimize}} \quad \underset{\phi\in\mathbb{R}^p}{\text{maximize}} \quad \mathbb{E}_{X\sim p_{\text{true}}}[\log D_\phi(X)]+\lambda\mathbb{E}_{\tilde{X}\sim p_\theta}[\log(1-D_\phi(\tilde{X}))].$$

Here, $\lambda>0$ represents the relative significance of a type II error over a type I error. Assuming the discriminator network D_ϕ is infinitely expressive, i.e., assuming $D_\phi\colon\mathbb{R}^n\to(0,1)$ can represent any function from $\mathbb{R}^n$ to $(0,1)$, show that the stated minimax problem is equivalent to

$$\underset{\theta\in\mathbb{R}^p}{\text{minimize}} \quad D_f(p_{\text{true}}\|p_\theta)$$

with

$$f(u)=\begin{cases}u\log\frac{u}{u+\lambda}+\lambda\log\frac{\lambda}{\lambda+u}+(1+\lambda)\log(1+\lambda)-\lambda\log\lambda & u\ge 0\\ \infty & \text{otherwise.}\end{cases}$$

Solution. Note

$$\begin{aligned}\mathcal{L}(\theta,\phi) &= \mathbb{E}_{X\sim p_{\text{true}}}[\log D_\phi(X)] + \lambda \mathbb{E}_{\tilde{X}\sim p_\theta}[\log(1-D_\phi(X))] \\ &= \int_x p_{\text{true}}(x)\log D_\phi(x) + \lambda p_\theta(x)\log(1-D_\phi(x))\,dx\end{aligned}$$

Since

$$\frac{d}{dy}(a\log y + b\log(1-y)) = \frac{a}{y} - \frac{b}{1-y} = \frac{a-y(a+b)}{y(1-y)} = 0$$

implies that $y = \dfrac{a}{a+b}$, for fixed θ, $L(\theta,\phi)$ is maximized by

$$D_{\phi^\star}(x) = \frac{p_{\text{true}}(x)}{p_{\text{true}}(x)+\lambda p_\theta(x)}.$$

If we plug in $D_{\phi^\star}$, then

$$\min_\theta \max_\phi \mathcal{L}(\theta,\phi) \simeq \min_\theta \mathcal{L}(\theta,\phi^\star).$$

Here we can rewrite $\mathcal{L}(\theta,\phi^\star)$ as follows:

$$\begin{aligned}\mathcal{L}(\theta,\phi^\star) &= \mathbb{E}_{X\sim p_{\text{true}}}\left[\log\frac{p_{\text{true}}(X)}{p_{\text{true}}(X)+\lambda p_\theta(X)}\right] + \lambda\mathbb{E}_{\tilde{X}\sim p_\theta}\left[\log\frac{\lambda p_\theta(\tilde{X})}{p_{\text{true}}(\tilde{X})+\lambda p_\theta(\tilde{X})}\right] \\ &= \int_x p_{\text{true}}(x)\log\frac{p_{\text{true}}(x)}{p_{\text{true}}(x)+\lambda p_\theta(x)} + \lambda p_\theta(x)\log\frac{\lambda p_\theta(x)}{p_{\text{true}}(x)+\lambda p_\theta(x)}\,dx \\ &= \int_x p_\theta(x)\left[\frac{p_{\text{true}}(x)}{p_\theta(x)}\log\frac{p_{\text{true}}(x)/p_\theta(x)}{p_{\text{true}}(x)/p_\theta(x)+\lambda} + \lambda\log\frac{\lambda}{p_{\text{true}}(x)/p_\theta(x)+\lambda}\right]dx \\ &= \int_x f\left(\frac{p_{\text{true}}(x)}{p_\theta(x)}\right)p_\theta(x)\,dx + \lambda\log\lambda - (1+\lambda)\log(1+\lambda) \\ &= D_f(p_{\text{true}}\|p_\theta) + \lambda\log\lambda - (1+\lambda)\log(1+\lambda),\end{aligned}$$

where f is defined by

$$f(u) = \begin{cases} u\log\frac{u}{u+\lambda} + \lambda\log\frac{\lambda}{\lambda+u} + (1+\lambda)\log(1+\lambda) - \lambda\log\lambda & u\ge 0 \\ \infty & \text{otherwise.}\end{cases}$$

Therefore,

$$\min_\theta \max_\phi \mathcal{L}(\theta,\phi) \simeq \min_\theta D_f(p_{\text{true}}\|p_\theta).$$

■

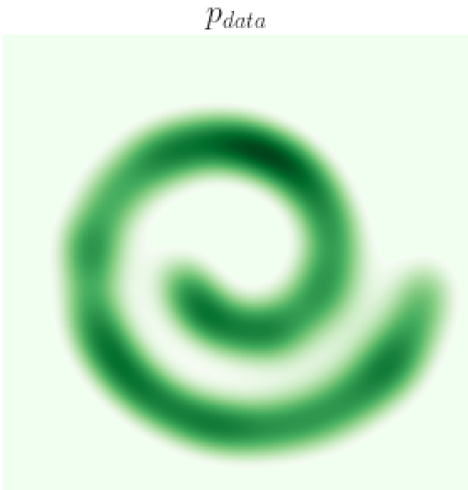

Figure 1: Data distribution p_{data} for the Swiss roll VAE and GAN problems.

Problem 3: *Swiss roll VAE.* Implement a VAE to learn the data distribution p_{data} defined by the starter code `swiss_roll.py` and illustrated in Figure 1. Use the standard VAE setup with

$$\begin{aligned} p_Z &= \mathcal{N}(0,1) && (z \in \mathbb{R}) \\ q_\phi(z \mid x) &= \mathcal{N}\left(\mu_\phi(x), \sigma_\phi^2(x)\right) \\ p_\theta(x \mid z) &= \mathcal{N}\left(f_\theta(z), \sigma^2 I\right), && \sigma = \frac{1}{\sqrt{150}} \end{aligned}$$

Let the encoder $(\mu_\phi, \log\sigma_\phi)$ be a 3-layer fully-connected network with both hidden layer widths equal to 128. Let the decoder f_θ be a 3-layer fully-connected network with both hidden layer widths equal to 64. For both the encoder and decoder networks, use the LeakyReLU activation function with negative slope 0.2 for the first hidden layer, the tanh activation function for the second hidden layer, and no activation function for the output layer. (The first hidden layer is the layer closest to the input.) Use the standard VAE loss

$$\mathcal{L}(\theta, \phi) = -\log p_\theta(X \mid Z) + D_{\text{KL}}\left(q_\phi(\cdot \mid X) \| p_Z(\cdot)\right)$$

where $X \sim p_{\text{data}}$ and $Z \sim q_\phi(z \mid X)$. Use the Adam optimizer with learning rate 5×10^{-4} and a batch size of 64. Train for 2000 epochs.

Solution. See `swiss_vae.py`. ■

Problem 4: *Swiss roll GAN.* Implement a GAN to learn the data distribution p_{data} defined by the starter code `swiss_roll.py` and illustrated in Figure 1. Use a latent distribution $p_Z = \mathcal{N}(0,1)$, with $z \in \mathbb{R}$. Let the discriminator D_ϕ be a 3-layer fully-connected network with both hidden layer widths equal to 128. Use the tanh activation function for the hidden layers and the sigmoid activation function for the output layer. Let the generator G_θ be a 2-layer fully-connected network with hidden layer width equal to 32. Use the tanh activation function for the hidden layer and no activation function for the output layer. Use the standard GAN loss

$$\mathcal{L}(\theta, \phi) = \log D_\phi(X) + \log(1 - D_\phi(G_\theta(Z))),$$

where $X \sim p_{\text{data}}$ and $Z \sim p_Z$. Use the Adam optimizer with learning rate 5×10^{-4} and a batch size of 64. Train for 2000 epochs.

Solution. See `swiss_gan.py`. ■