Untitled

November 9, 2021

1 Mathematical Foundations on Deep Neural Network

1.1 Homework #6

1.1.1 2017-11362

1.2 Problem 1.

[2]:
```
import torch.nn as nn
from torch.utils.data import DataLoader
import torch
import torchvision
import torchvision.transforms as transforms


# Instantiate model with BN and load trained parameters
class smallNetTrain(nn.Module) :
    # CIFAR-10 data is 32*32 images with 3 RGB channels
    def __init__(self, input_dim=3*32*32) :
        super().__init__()

        self.conv1 = nn.Sequential(
                            nn.Conv2d(3, 16, kernel_size=3, padding=1),
                            nn.BatchNorm2d(16),
                            nn.ReLU()
                            )
        self.conv2 = nn.Sequential(
                            nn.Conv2d(16, 16, kernel_size=3, padding=1),
                            nn.BatchNorm2d(16),
                            nn.ReLU()
                            )
        self.fc1 = nn.Sequential(
                            nn.Linear(16*32*32, 32*32),
                            nn.BatchNorm1d(32*32),
                            nn.ReLU()
                            )
        self.fc2 = nn.Sequential(
                            nn.Linear(32*32, 10),
                            nn.ReLU()
```

```python
                              )
    def forward(self, x) :
        x = self.conv1(x)
        x = self.conv2(x)
        x = x.float().view(-1, 16*32*32)
        x = self.fc1(x)
        x = self.fc2(x)

        return x

model = smallNetTrain()
model.load_state_dict(torch.load("./smallNetSaved",map_location=torch.
 ↪device('cpu')))


# Instantiate model without BN
class smallNetTest(nn.Module) :
    # CIFAR-10 data is 32*32 images with 3 RGB channels
    def __init__(self, input_dim=3*32*32) :
        super().__init__()

        self.conv1 = nn.Sequential(
                              nn.Conv2d(3, 16, kernel_size=3, padding=1),
                              nn.ReLU()
                              )
        self.conv2 = nn.Sequential(
                              nn.Conv2d(16, 16, kernel_size=3, padding=1),
                              nn.ReLU()
                              )
        self.fc1 = nn.Sequential(
                              nn.Linear(16*32*32, 32*32),
                              nn.ReLU()
                              )
        self.fc2 = nn.Sequential(
                              nn.Linear(32*32, 10),
                              nn.ReLU()
                              )
    def forward(self, x) :
        x = self.conv1(x)
        x = self.conv2(x)
        x = x.float().view(-1, 16*32*32)
        x = self.fc1(x)
        x = self.fc2(x)

        return x

model_test = smallNetTest()
```

```
# Initialize weights of model without BN

conv1_bn_beta, conv1_bn_gamma = model.conv1[1].bias, model.conv1[1].weight
conv1_bn_mean, conv1_bn_var = model.conv1[1].running_mean, model.conv1[1].
 ↪running_var
conv2_bn_beta, conv2_bn_gamma = model.conv2[1].bias, model.conv2[1].weight
conv2_bn_mean, conv2_bn_var = model.conv2[1].running_mean, model.conv2[1].
 ↪running_var
fc1_bn_beta, fc1_bn_gamma = model.fc1[1].bias, model.fc1[1].weight
fc1_bn_mean, fc1_bn_var = model.fc1[1].running_mean, model.fc1[1].running_var
eps = 1e-05


##################################################################################
# Initialize the following parameters
model_test.conv1[0].weight.data = model.conv1[0].weight * conv1_bn_gamma.
 ↪reshape(-1,1,1,1) \
                                  / torch.sqrt(conv1_bn_var.reshape(-1,1,1,1) +␣
 ↪eps)
model_test.conv1[0].bias.data = conv1_bn_beta + conv1_bn_gamma * (model.
 ↪conv1[0].bias - conv1_bn_mean) \
                                  / torch.sqrt(conv1_bn_var + eps)

model_test.conv2[0].weight.data = model.conv2[0].weight * conv2_bn_gamma.
 ↪reshape(-1,1,1,1) \
                                  / torch.sqrt(conv2_bn_var.reshape(-1,1,1,1) +␣
 ↪eps)
model_test.conv2[0].bias.data = conv2_bn_beta + conv2_bn_gamma*(model.conv2[0].
 ↪bias - conv2_bn_mean) \
                                  / torch.sqrt(conv2_bn_var + eps)

model_test.fc1[0].weight.data = fc1_bn_gamma.reshape(-1,1) * model.fc1[0].
 ↪weight \
                                  / torch.sqrt(fc1_bn_var.reshape(-1,1) + eps)
model_test.fc1[0].bias.data = fc1_bn_beta + fc1_bn_gamma * (model.fc1[0].bias -␣
 ↪fc1_bn_mean) \
                                  / torch.sqrt(fc1_bn_var + eps)

model_test.fc2[0].weight.data = model.fc2[0].weight
model_test.fc2[0].bias.data = model.fc2[0].bias
##################################################################################
```

```python
# Verify difference between model and model_test

model.eval()
# model_test.eval()  # not necessary since model_test has no BN or dropout


test_dataset = torchvision.datasets.CIFAR10(root='./cifar_10data/',
                                train=False,
                                transform=transforms.ToTensor(), download =␣
 ↪True)
test_loader = torch.utils.data.DataLoader(dataset=test_dataset, batch_size=100,␣
 ↪shuffle=False)

diff = []
with torch.no_grad():
    for images, _ in test_loader:
        diff.append(torch.norm(model(images) - model_test(images))**2)

print(max(diff)) # If less than 1e-08, you got the right answer.
```

```
Files already downloaded and verified
tensor(8.3320e-09)
```

1.3 Problem 2.

Problem 2.

Pytorch $\Rightarrow$ A : kaiming_uniform(weight, $a=\sqrt{5}$) : unif$\left(-\sqrt{\frac{1}{n_{in}}}, \sqrt{\frac{1}{n_{in}}}\right)$, b : unif$\left(-\sqrt{\frac{1}{n_{in}}}, \sqrt{\frac{1}{n_{in}}}\right)$.

$x \sim (0, I_{n_0})$, $(A_1)_{ij}, (b_1)_i \sim$ iid $U\left(-\sqrt{\frac{1}{n_0}}, \sqrt{\frac{1}{n_0}}\right)$ $\left(E(A_1)_{ij} = 0, \ Var(A_1)_{ij} = \frac{1}{12}\left(2\sqrt{\frac{1}{n_0}}\right)^2 = \frac{1}{3n_0} \right)$

$\Rightarrow E(y_1) = E(A_1 x + b_1) = E(A_1)E(x) + E(b_1) = 0.$

$Var(y_1) = Var(A_1 x + b_1) = Var(A_1 x) + Var(b_1)$

$Cov((A_1 x)_i, (A_1 x)_j) = Cov\left(\sum_{k=1}^{n_0} (A_1)_{ik} x_k, \sum_{k=1}^{n_0} (A_1)_{jk} x_k\right)$

$= \sum_{k=1}^{n_0} Cov\left((A_1)_{ik} x_k, (A_1)_{jk} x_k\right)$

$$Cov(XZ, YZ) = E(XYZ^2) = \begin{cases} E(X^2)E(Z^2) = Var(X)Var(Z), & X = Y \\ E(X)E(Y)E(Z^2) = 0, & X \neq Y \end{cases}$$

$$\Rightarrow Cov((A_1)_{ik} x_k, (A_1)_{jk} x_k) = \begin{cases} \frac{1}{3n_0}, & i = j \\ 0, & i \neq j \end{cases} \quad \Rightarrow Var(A_1 x) = \frac{1}{3} I_{n_1}$$

$\therefore \ Var(y_1) = \left(\frac{1}{3} + \frac{1}{3n_0}\right) I_{n_1}$

$y_1 \sim \left(0, \left(\frac{1}{3} + \frac{1}{3n_0}\right) I_{n_1}\right)$

$\Rightarrow E(y_2) = E(A_2 y_1 + b_2) = E(A_2)E(y_1) + E(b_2) = 0$

$Var(y_2) = Var(A_2 y_1 + b_2) = Var(A_2 y_1) + Var(b_2) = \frac{1}{3}\left(\frac{1}{3} + \frac{1}{3n_0}\right) I_{n_2} + \frac{1}{3n_1} I_{n_2}$

$= \left(\frac{1}{3n_1} + \frac{1}{3^2 n_0} + \frac{1}{3^2}\right) I_{n_2}$

$\Rightarrow y_2 \sim \left(0, \left(\frac{1}{3n_1} + \frac{1}{3^2 n_0} + \frac{1}{3^2}\right) I_{n_2}\right)$

...

$$y_L \sim \Big(\underbrace{0}_{E(y_L)}, \underbrace{\frac{1}{3n_{L-1}} + \frac{1}{3^2 n_{L-2}} + \cdots + \frac{1}{3^L n_0} + \frac{1}{3^L}}_{Var(y_L)}\Big)$$

1.4 Problem 3.

$$\frac{\partial y_L}{\partial b_L} = 1.$$

$$\frac{\partial y_L}{\partial b_\ell} = \frac{\partial y_L}{\partial \beta_\ell}\frac{\partial \beta_\ell}{\partial b_\ell}, \quad \beta_\ell = b_\ell \cdot 1_{n_\ell} \Rightarrow \frac{\partial \beta_\ell}{\partial b_\ell} = 1_{n_\ell}.$$

From HW4-6, $\frac{\partial y_L}{\partial \beta_\ell} = \operatorname{diag}(\sigma'(A_\ell y_{\ell-1} + \beta_\ell))$

$$\therefore \frac{\partial y_L}{\partial b_\ell} = \operatorname{diag}(\sigma'(A_\ell y_{\ell-1} + b_\ell \cdot 1_{n_\ell}))\, 1_{n_\ell} = \sigma'(A_\ell y_{\ell-1} + b_\ell \cdot 1_{n_\ell}), \quad \ell = 1, 2, \cdots, L-1.$$

$$\frac{\partial y_L}{\partial w_L} = \frac{\partial}{\partial w_L}\left(\sum_{i=1}^{f_L} w_{Li}\, y_{L-1} + b_L\right) = y_{L-1}^T.$$

$$\frac{\partial y_L}{\partial (w_\ell)_k} = \sum_{i,j} \frac{\partial y_L}{\partial (A_\ell)_{ij}} \frac{\partial (A_\ell)_{ij}}{\partial (w_\ell)_k}, \quad A_\ell = \begin{bmatrix} w_{\ell 1} & \cdots & w_{\ell f_\ell} & 0 & \cdots & 0 \\ 0 & w_{\ell 1} & \cdots & w_{\ell f_{\ell-1}} & w_{\ell f_\ell} & 0 \cdots 0 \\ \vdots & & & & & \\ 0 & \cdots & 0 & w_{\ell 1} & \cdots & w_{\ell f_\ell} \end{bmatrix} \Rightarrow \frac{\partial A_\ell}{\partial (w_\ell)_k} = \begin{array}{c} \text{kth} \\ \downarrow \\ \begin{bmatrix} 0 \cdots & 1 & \cdots & 0 & 0 \\ & & 1 & & \\ & & & 1 & \\ & & & & \ddots \\ 0 & & & & 1 \end{bmatrix} \end{array}$$

From HW4-6, $\frac{\partial y_L}{\partial A_\ell} = \operatorname{diag}(\sigma'(A_\ell y_{\ell-1} + b_\ell 1_{n_\ell}))\left(\frac{\partial y_L}{\partial y_\ell}\right)^T y_{\ell-1}^T.$

$$\frac{\partial y_L}{\partial (w_\ell)_k} = \sum_{i=1}^{n_\ell} \left(\frac{\partial y_L}{\partial A_\ell}\right)_{i,k+i-1}, \quad k = 1, \cdots, f_\ell.$$

1.5 Problem 4.

$Y \sim N_n(0, I)$, $X = AY + b \Rightarrow Y = A^{-1}(X - b)$

$f_Y(y) = (2\pi)^{-n/2}\exp(-\frac{1}{2}y^T y)$ $\quad \left|\frac{dx}{dy}\right| = \det(A) \Rightarrow \left|\frac{dy}{dx}\right| = \frac{1}{\det(A)}$

$$f_Y(y)\,dy = (2\pi)^{-n/2}\exp(-\tfrac{1}{2}y^T y)\,dy$$

$$= (2\pi)^{-n/2}\exp(-\tfrac{1}{2}[A^{-1}(x-b)]^T \cdot A^{-1}(x-b))\left|\tfrac{dy}{dx}\right| dx$$

$$= (2\pi)^{-n/2}\exp(-\tfrac{1}{2}(x-b)^T (A^T)^{-1} A^{-1}(x-b))\,\det(A)^{-1}\,dx$$

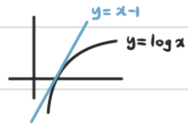
$$= \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}}\exp\left\{-\tfrac{1}{2}(x-b)^T \Sigma^{-1}(x-b)\right\} dx \quad \left(\Sigma = AA^T,\ \Sigma^{-1} = (AA^T)^{-1} = (A^T)^{-1}A^{-1},\ \det(\Sigma) = \det(A)\det(A^T) = \det(A)^2\right)$$

$$\therefore f_X(x) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}}\exp\left\{-\tfrac{1}{2}(x-b)^T \Sigma^{-1}(x-b)\right\}$$

1.6 Problem 5.

Problem 5. $D_{KL}(X\|Y) = \int_{\mathbb{R}^d} f(x)\log\left(\frac{f(x)}{g(x)}\right)dx.$

(a) $D_{KL}(X\|Y) \geq 0.$



$\Rightarrow x-1 \geq \log x,\ {}^{\forall} x>0 \iff -\log x \geq -x+1,\ {}^{\forall} x>0$

$$D_{KL}(X\|Y) = \int_{\mathbb{R}^d} f(x)\log\left(\frac{f(x)}{g(x)}\right)dx$$

$$= \int_{\mathbb{R}^d} f(x)\left(-\log\left(\frac{g(x)}{f(x)}\right)\right)dx$$

$$\geq \int_{\mathbb{R}^d} f(x)\left(-\frac{g(x)}{f(x)}+1\right)dx$$

$$= \underbrace{\int_{\mathbb{R}^d} f(x)\,dx}_{\text{pdf of } X} - \underbrace{\int_{\mathbb{R}^d} g(x)\,dx}_{\text{pdf of } Y} = 0$$

(b) $X=(X_1,\cdots,X_d)$, $Y=(Y_1,\cdots,Y_d)$, X_i : indep, Y_i : indep.

$$D_{KL}(X\|Y) = \int\cdots\int f(x_1,\cdots,x_d)\log\left(\frac{f(x_1,\cdots,x_d)}{g(x_1,\cdots,x_d)}\right)dx_1\cdots dx_n$$

$$= \int\cdots\int f(x_1)\cdots f(x_d)\log\left(\frac{f(x_1)\cdots f(x_d)}{g(x_1)\cdots f(x_d)}\right)dx_1\cdots dx_n$$

$$= \int\cdots\int f(x_1)\cdots f(x_d)\left[\log\left(\frac{f(x_1)}{g(x_1)}\right)+\cdots+\log\left(\frac{f(x_d)}{g(x_d)}\right)\right]dx_1\cdots dx_n$$

$$= \sum_{i=1}^{d}\int\cdots\int f(x_1)\cdots f(x_d)\log\left(\frac{f(x_i)}{g(x_i)}\right)dx_1\cdots dx_d$$

$$= \sum_{i=1}^{d}\underbrace{\int f(x_1)dx_1}_{1}\cdots\underbrace{\int f(x_{i-1})dx_{i-1}}_{1}\underbrace{\int f(x_{i+1})dx_{i+1}}_{1}\cdots\underbrace{\int f(x_d)dx_d}_{1}\underbrace{\int f(x_i)\log\left(\frac{f(x_i)}{g(x_i)}\right)dx_i}_{D_{KL}(X_i\|Y_i)}$$

$$= D_{KL}(X_1\|Y_1)+\cdots+D_{KL}(X_d\|Y_d).$$

1.7 Problem 6.

$X\sim N(\mu_0,\Sigma_0)$, $Y\sim N(\mu_1,\Sigma_1)$

$$D_{KL}(X\|Y) = \int_{\mathbb{R}^d} f_X(x)\log\left(\frac{f_X(x)}{f_Y(x)}\right)dx$$

$$= \int_{\mathbb{R}^d} f_X(x)\left[\frac{1}{2}\log\left(\frac{\det\Sigma_0}{\det\Sigma_1}\right)-\frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)+\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right]dx$$

$$= \frac{1}{2}\log\left(\frac{\det\Sigma_0}{\det\Sigma_1}\right)-\frac{1}{2}\int_{\mathbb{R}^d} f(x)\cdot\mathrm{tr}(\Sigma_0^{-1}(x-\mu_0)(x-\mu_0)^T)\,dx+\frac{1}{2}\int_{\mathbb{R}^d} f_X(x)\,\mathrm{tr}(\Sigma_1^{-1}(x-\mu_1)(x-\mu_1)^T)\,dx$$

$$= \frac{1}{2}\log\left(\frac{\det\Sigma_0}{\det\Sigma_1}\right)-\frac{1}{2}\mathbb{E}(\mathrm{tr}(\Sigma_0^{-1}(X-\mu_0)(X-\mu_0)^T))+\frac{1}{2}\mathbb{E}(\mathrm{tr}(\Sigma_1^{-1}(X-\mu_1)(X-\mu_1)^T))$$

$$= \frac{1}{2}\log\left(\frac{\det\Sigma_0}{\det\Sigma_1}\right)-\frac{1}{2}\mathrm{tr}(\Sigma_0^{-1}\underbrace{\mathbb{E}(X-\mu_0)(X-\mu_0)^T}_{\Sigma_0})+\frac{1}{2}\mathrm{tr}(\Sigma_1^{-1}\underbrace{\mathbb{E}(X-\mu_1)(X-\mu_1)^T}_{\mathbb{E}(X-\mu_0+\mu_0-\mu_1)(X-\mu_0+\mu_0-\mu_1)^T})$$

$$= \mathbb{E}(X-\mu_0)(X-\mu_0)^T+(\mu_0-\mu_1)\underbrace{\mathbb{E}(X-\mu_0)}_{0}+(\mu_0-\mu_1)(\mu_0-\mu_1)^T$$

$$= \Sigma_0+(\mu_0-\mu_1)(\mu_0-\mu_1)^T$$

$$= \frac{1}{2}\left[\log\left(\frac{\det\Sigma_0}{\det\Sigma_1}\right)-\mathrm{tr}(I_d)+\mathrm{tr}(\Sigma_1^{-1}\Sigma_0)\right.$$
$$\left.+\,\mathrm{tr}(\Sigma_1^{-1}(\mu_0-\mu_1)(\mu_0-\mu_1)^T)\right]$$

$$= \frac{1}{2}\left[\log\left(\frac{\det\Sigma_0}{\det\Sigma_1}\right)-d+\mathrm{tr}(\Sigma_1^{-1}\Sigma_0)+(\mu_0-\mu_1)\Sigma_1^{-1}(\mu_0-\mu_1)^T\right].$$