2020-14281 이서웅 HW6

#1. ① Linear − dropout − σ 와 ② Linear − σ − dropout 이 같으려면,

① $y = \sigma(\tilde{y}')$ 와 ② $(y)_i = \begin{cases} 0 & \text{with prob } p \\ \dfrac{(\sigma(Wx+b))_i}{1-p} & \text{otherwise} \end{cases}$ 가 같아야 한다.

$$\left((\tilde{y}')_i = \begin{cases} 0 & \text{with prob } p \\ \dfrac{y_i}{1-p} = \dfrac{(Wx+b)_i}{1-p} & \text{otherwise} \end{cases} \right)$$

$(y)_i$ 를 살피자.

①의 경우, $(y)_i = \sigma((\tilde{y}')_i) = \begin{cases} 0 & \text{with probability } p \\ \sigma\left(\dfrac{\sum_j W_{ij}x_j + b_i}{1-p}\right) & \text{otherwise} \end{cases}$

②의 경우, $(y)_i = \begin{cases} 0 & \text{with prob } p \\ \dfrac{\sigma\left(\sum_j W_{ij}x_j + b_i\right)}{1-p} & \text{otherwise} \end{cases}$

즉 ①과 ②가 같으려면, $\sigma(cx) = c\,\sigma(x)$ 가 성립하면 된다. $(c>0)$

즉, ReLU (a) 와 LeakyReLU (c) 는 equivalent, sigmoid (b)는 그렇지 않다.

#2. Pytorch 의 Default weight initialization을 찾아보면, $k = \dfrac{1}{\text{in\_features}}$ 라고 할 때,

A 와 b 모두 Uniform$(-\sqrt{k}, \sqrt{k})$ 으로 initialize 된다고 한다. (즉 Mean 0, Variance $= \frac{1}{3}k$)

$x_1, \ldots x_n$: IID, $E[x_i]=0$, $Var(x_i)=1$.

Then, $y_1 = A_1x + b_1$, $(y_1)_i = \sum_{j=1}^{n_0} (A_1)_{ij}(x)_j + (b_1)_i$

$$E[(y_1)_i] = \underbrace{0+0\cdots+0}_{n_0}+0 = 0$$

$$Var[(y_1)_i] = Var\left(\sum_{j=1}^{n_0}(A_1)_{ij}(x)_j + (b_1)_i\right) = \frac{1}{3}\frac{1}{n_0} + Var\left(\sum_{j=1}^{n_0}(A_1)_{ij}(x_j)\right)$$

$$= \frac{1}{3n_0} + \sum_1^{n_0}\frac{1}{3}\frac{1}{n_0}\times 1 = \frac{n_0+1}{3n_0} = \frac{1}{3}\left(\frac{1}{n_0}+1\right)$$

So we can notice that $\boxed{E[y_L]=0}$ and, $Var((y_i)_j) = \dfrac{1}{3n_{i-1}} + n_{i-1} \times \dfrac{1}{3n_{i-1}} \times Var((y_{i-1})_k)$

$$= \frac{1}{3}\left(\frac{1}{n_{i-1}} + Var((y_{i-1})_k)\right)$$

So, $\boxed{Var(y_L) = \dfrac{1}{3^L} + \sum_{i=0}^{i=L-1} \dfrac{1}{n_i}\dfrac{1}{3^{L-i}}}$

#3 <i> $\frac{\partial y_l}{\partial y_{l-1}}$ for $l=2 \dots L$.

$$\left(\frac{\partial y_L}{\partial y_{L-1}}\right)_i = \frac{\partial y_L}{\partial (y_{L-1})_i} = \frac{\partial (A_L y_{L-1} + b_L)}{\partial (y_{L-1})_i} = (A_L)_i \quad \text{Thus} \quad \boxed{\frac{\partial y_L}{\partial y_{L-1}} = A_L \in \mathbb{R}^{1\times m}.}$$

$$\left(\frac{\partial y_l}{\partial y_{l-1}}\right)_{ij} = \frac{\partial (y_l)_i}{\partial (y_{l-1})_j} = \frac{\partial\left(\left(\sigma(A_l y_{l-1} + b_l) + y_{l-1}\right)_i\right)}{\partial (y_{l-1})_j} = \frac{\partial\left((y_{l-1})_i + \sigma\left(\sum_{j=1}^{m} (A_l)_{ij} (y_{l-1})_j + (b_l)_i\right)\right)}{\partial (y_{l-1})_j}$$

$\in \mathbb{R}^{m\times m}$

$$= \begin{cases} \sigma'\left(\sum_{j=1}^{m} (A_l)_{ij} (y_{l-1})_j + (b_l)_i\right) \cdot (A_l)_{ij} & \text{if } i \neq j \\ 1 + \sigma'\left(\sum_{j=1}^{m} (A_l)_{ij} (y_{l-1})_j + (b_l)_i\right) \cdot (A_l)_{ij} & \text{else} \end{cases}$$

$$\therefore \boxed{\underbrace{\left(\frac{\partial y_l}{\partial y_{l-1}}\right)}_{\in \mathbb{R}^{m\times m}} = I + \mathrm{diag}\left(\sigma'(A_l y_{l-1} + b_l)\right) A_l \quad \text{for } l=2, \dots L-1}$$

<ii> $l = 1, \dots L$.

$$\frac{\partial y_L}{\partial b_l} = \frac{\partial y_L}{\partial y_{L-1}} \cdot \frac{\partial y_{L-1}}{\partial y_{L-2}} \cdots \frac{\partial y_{l+1}}{\partial y_l} \cdot \frac{\partial y_l}{\partial b_l} =$$

$$\frac{\partial y_L}{\partial b_L} = 1. \quad \left(\frac{\partial y_l}{\partial b_l}\right)_{ij} = \frac{\partial (y_l)_i}{\partial (b_l)_j} = \frac{\partial\left(\left(\sigma(A_l y_{l-1} + b_l) + y_{l-1}\right)_i\right)}{\partial (b_l)_j} = \begin{cases} \sigma'\left(\sum_{j=1}^{m} (A_l)_{ij} (y_{l-1})_j + (b_l)_i\right) & \text{if } i=j \\ 0 & \text{else} \end{cases}$$

$$\Rightarrow \frac{\partial y_l}{\partial b_l} = \mathrm{diag}\left(\sigma'(A_l y_{l-1} + b_l)\right)$$

$$\therefore \boxed{\frac{\partial y_L}{\partial b_L} = 1. \quad \underbrace{\frac{\partial y_L}{\partial b_l}}_{\in \mathbb{R}^{1\times m}} = \frac{\partial y_L}{\partial y_l}\left(\mathrm{diag}\left(\sigma'(A_l y_{l-1} + b_l)\right)\right).}$$

* $\frac{\partial y_l}{\partial y_{l-1}}$ is calculated in <i>.

$$\Rightarrow \frac{\partial y_L}{\partial y_l} = \frac{\partial y_L}{\partial y_{L-1}} \cdots \frac{\partial y_{l+1}}{\partial y_l}$$

* $\frac{\partial y_L}{\partial b_l} \in \mathbb{R}^m$ 으로 생각하고 싶다면, $\mathrm{diag}\left(\sigma'(A_l y_{l-1} + b_l)\right)\left(\frac{\partial y_L}{\partial y_l}\right)^T$ 로 표현하는 것이 좋음.

$$\frac{\partial y_L}{\partial A_l} \quad l = 1 \dots L.$$

$$\left(\frac{\partial y_L}{\partial A_l}\right) \in \mathbb{R}^{n_l \times n_{l-1}}, \quad \left(\frac{\partial y_L}{\partial A_l}\right)_{ij} = \frac{\partial y_L}{\partial (A_l)_{ij}}$$ (clarifying notation).

- $\frac{\partial y_L}{\partial A_L} \in \mathbb{R}^{1 \times m}$ $\quad \left(\frac{\partial y_L}{\partial A_L}\right)_{1,j} = \frac{\partial y_L}{\partial (A_L)_{1j}} = (y_{L-1})_j \Rightarrow \boxed{\frac{\partial y_L}{\partial A_L} = y_{L-1}^T}$

$\frac{\partial y_L}{\partial A_l} \in \mathbb{R}^{n \times m}$ $(l = 1 \dots L-1)$

$$\left(\frac{\partial y_L}{\partial A_l}\right)_{ij} = \frac{\partial y_L}{\partial (A_l)_{ij}} = \left(\frac{\partial y_L}{\partial y_{L-1}} \cdots \frac{\partial y_{l+1}}{\partial y_l}\right) \frac{\partial y_l}{\partial (A_l)_{ij}}$$

$$\frac{\partial y_l}{\partial (A_l)_{ij}} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \sigma'\left(\sum_{k=1}^{m} (A_l)_{ik} (y_{l-1})_k + (b_l)_i\right)(y_{l-1})_j \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow i\text{번째 행}$$

$$\therefore \left(\frac{\partial y_L}{\partial A_l}\right)_{ij} = \left(\frac{\partial y_L}{\partial y_l}\right) \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \sigma'\left(\sum_{k=1}^{m} (A_l)_{ik} (y_{l-1})_k + (b_l)_i\right)(y_{l-1})_j \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow i\text{번째 행}$$

$$\Rightarrow \underbrace{\left(\frac{\partial y_L}{\partial A_l}\right)}_{\in\,\mathbb{R}^{n \times m}} = \underbrace{\operatorname{diag}\left(\sigma'(A_l y_{l-1} + b_l)\right)}_{\in\,\mathbb{R}^{m \times m}} \underbrace{\left(\frac{\partial y_L}{\partial y_l}\right)^T}_{\in\,\mathbb{R}^{m \times 1}} \underbrace{(y_{l-1})^T}_{\in\,\mathbb{R}^{1 \times m}}$$

<iii> $\frac{\partial y_L}{\partial b_i}$ 와 $\frac{\partial y_L}{\partial A_i}$ 를 살펴보면, 모두 $\left(\frac{\partial y_L}{\partial y_l}\right)^T$ term이 존재한다. 그러나, residual connection이 있는 경우,

$$\left(\frac{\partial y_L}{\partial y_l}\right) = (A_L)\left(I + \operatorname{diag}\left(\sigma'(A_{L-1} y_{L-2} + b_{L-1})\right) A_{L-1}\right) \cdots \left(I + \operatorname{diag}\left(\sigma'(A_{l+1} y_l + b_{l+1})\right) A_{l+1}\right)$$

와 같이 계산되기 때문에, A_i 혹은 $\sigma'(A_i y_{i-1} + b_i) = 0$ 이 되는 상황이 나타나 $\operatorname{diag}(\sigma'(A_{L-1} y_{L-2} + b_{L-1})) A_{L-1}$ 가 0이 되더라도 Identity Matrix가 남게 된다. 그렇기에 이전의 Naive MLP 와는 다르게, $\frac{\partial y_L}{\partial b_i}$, $\frac{\partial y_L}{\partial A_i}$ 가 0으로 vanish 하지 않게 된다.

#4

(a) Parameter Count

<i> Naive structure : $128\times(256\times1\times1+1) + 128\times(128\times3\times3+1) + 256\times(128\times1\times1+1)$

$= 213504$ 개

<ii> Split-transform-Merge

: $32\times\left(4\times(256\times1\times1+1) + 4\times(4\times3\times3+1) + 256\times(4\times1\times1+1)\right)$

$= 78592$개

(b) Implement

* (init 함수 안)

```
self.convlayers1 = nn.ModuleList([nn.Conv2d(256, 4, 1) for i in range(32)])
self.convlayers2 = nn.ModuleList([nn.Conv2d(4, 4, 3, padding=1) for i in range(32)])
self.convlayers3 = nn.ModuleList([nn.Conv2d(4, 256, 1) for i in range(32)])
```

* forward 함수 안.

```
answer = torch.zeros(x.size())
for i in range(32):
    out = torch.nn.functional.relu(self.convlayers1[i](x))
    out = torch.nn.functional.relu(self.convlayers2[i](x))
    out = torch.nn.functional.relu(self.convlayers3[i](x))
    answer += out
out = answer
```

#5

다음과 같이 짠다

ddescent.py

```python
import matplotlib.pyplot as plt
import numpy as np


"""
Step 1 : Generate Toy data
"""

d = 35
n_train, n_val, n_test = 300, 60, 30
np.random.seed(0)
beta = np.random.randn(d)
beta_true = beta / np.linalg.norm(beta)
# Generate and fix training data
X_train = np.array([np.random.multivariate_normal(np.zeros(d), np.identity(d)) for _ in range(n_train)])
Y_train = X_train @ beta_true + np.random.normal(loc = 0.0, scale = 0.5, size = n_train)
# Generate and fix validation data (for tuning lambda).
X_val = np.array([np.random.multivariate_normal(np.zeros(d), np.identity(d)) for _ in range(n_val)])
Y_val = X_val @ beta_true
# Generate and fix test data
X_test = np.array([np.random.multivariate_normal(np.zeros(d), np.identity(d)) for _ in range(n_test)])
Y_test = X_test @ beta_true


"""
Step 2 : Solve the problem
"""
def f(X,W):
    L=[]
    for i in range(len(X)):
        L.append((W @ X[i]) * (W @ X[i]>0))
    return np.array(L)

fixed_lambda = 0.01
lambda_list = [2 ** i for i in range(-6, 6)]
num_params = np.arange(1,1501,10)

errors_opt_lambda = []
errors_fixed_lambda = []
for p in num_params :
    # fix W, calculate Xtilda based on fixed W
    W=np.random.normal(0, 1/np.sqrt(p), size=(p,d))
    X_tilda = f(X_train, W)
    X_val_tilda = f(X_val, W)
    X_test_tilda = f(X_test, W)

    # theta based on fixed lambda
    theta_fixed = np.linalg.inv(X_tilda.T @ X_tilda + fixed_lambda * np.identity(p) )@ X_tilda.T @ Y_train

    # find optimal lambda using validation data
    minloss = 10000000000000
    theta_optimal = np.zeros(p)
    for l in lambda_list:
        theta =  np.linalg.inv(X_tilda.T @ X_tilda + l * np.identity(p) )@ X_tilda.T @ Y_train
        loss = np.linalg.norm(X_val_tilda@theta - Y_val)
        if(loss < minloss):
            minloss = loss
            theta_optimal = theta


    #use test set to calculate accuracy
    fixed_loss = np.linalg.norm(X_test_tilda@theta_fixed - Y_test)
    errors_fixed_lambda.append(fixed_loss)

    optimal_loss = np.linalg.norm(X_test_tilda@theta_optimal - Y_test)
    errors_opt_lambda.append(optimal_loss)

    #just for debugging
    print(p, " done")

```

```python
"""
Step 3 : Plot the results
"""

plt.figure(figsize = (24, 8))
plt.rc('text', usetex = False)
plt.rc('font', family = 'serif')
plt.rc('font', size = 24)


plt.scatter(num_params, errors_fixed_lambda, color = 'black',
            label = r"Test error with fixed $\lambda = 0.01$",
            )
plt.legend()

plt.plot(num_params, errors_opt_lambda, 'k', label = r"Test error with tuned $\lambda$")
plt.legend()
plt.xlabel(r'$\#$ parameters')
plt.ylabel('Test error')
plt.title(r'Test error vs. $\#$ params : SeoyongLee')

plt.savefig('double_descent.png')
plt.show()
```

그러면 똑같은 결과를 얻게 된다.