Machine Learning (H)
Southern University of Science and Technology
Mengxuan Wu
12212006

Assignment 5

Mengxuan Wu

Question 1

(a)

$$p(D|w) = \prod_{n=1}^{N} p(t_n|x_n, w) = \prod_{n=1}^{N} N(t_n|y(x_n, w), \Sigma)$$

The log likelihood is

$$\ln p(D|w) = -\frac{1}{2}\sum_{n=1}^{N}\{t_n - y(x_n, w)\}^T\Sigma^{-1}\{t_n - y(x_n, w)\} - \frac{N}{2}\ln|\Sigma| - \frac{N}{2}\ln(2\pi)$$

Taking the gradient with respect to w and setting it to zero, we have

$$\frac{\partial}{\partial w}\ln p(D|w) = \frac{\partial}{\partial w}\left\{-\frac{1}{2}\sum_{n=1}^{N}\{t_n - y(x_n, w)\}^T\Sigma^{-1}\{t_n - y(x_n, w)\}\right\} = 0$$

By solving the above equation, we can get the maximum likelihood solution w_{ML}.

(b)

Taking the gradient of the log likelihood with respect to Σ and setting it to zero, we have

$$\frac{\partial}{\partial \Sigma}\ln p(D|w) = -\frac{1}{2}\sum_{n=1}^{N}\{t_n - y(x_n, w)\}\{t_n - y(x_n, w)\}^T + \frac{N}{2}\Sigma^{-1} = 0$$

Thus, we have

$$\Sigma_{ML} = \frac{1}{N}\sum_{n=1}^{N}\{t_n - y(x_n, w_{ML})\}\{t_n - y(x_n, w_{ML})\}^T$$

Question 2

We can write the output as

$$y(a) = 2\sigma(a) - 1$$

which will be in the range $[-1, 1]$.
Then, the conditional distribution of t given x and w is

$$p(t|x, w) = [\frac{1 + y(x, w)}{2}]^{\frac{1+t}{2}} [\frac{1 - y(x, w)}{2}]^{\frac{1-t}{2}}$$

The error function is

$$\begin{aligned}
E(w) &= -\ln p(D|w) \\
&= -\sum_{n=1}^{N} \ln p(t_n|x_n, w) \\
&= -\sum_{n=1}^{N} \left\{ \frac{1 + t_n}{2} \ln \frac{1 + y(x_n, w)}{2} + \frac{1 - t_n}{2} \ln \frac{1 - y(x_n, w)}{2} \right\} \\
&= -\frac{1}{2} \sum_{n=1}^{N} \{(1 + t_n) \ln(1 + y(x_n, w)) + (1 - t_n) \ln(1 - y(x_n, w))\} + N \ln 2
\end{aligned}$$

We can also write the output as

$$\begin{aligned}
y(a) &= 2\sigma(a) - 1 \\
&= \frac{2}{1 + \exp(-a)} - 1 \\
&= \frac{1 - \exp(-a)}{1 + \exp(-a)} \\
&= \frac{\exp(a) - 1}{\exp(a) + 1} \\
&= \tanh(\frac{a}{2})
\end{aligned}$$

Thus, the appropriate choice of the activation function is $\tanh(\frac{a}{2})$.

Question 3

(a)

$$\begin{aligned}
E[t|x] &= \int t p(t|x) dt \\
&= \int t \sum_{k=1}^{K} \pi_k \mathcal{N}(t|\mu_k, \sigma_k^2) dt \\
&= \sum_{k=1}^{K} \pi_k \int t \mathcal{N}(t|\mu_k, \sigma_k^2) dt \\
&= \sum_{k=1}^{K} \pi_k \mu_k
\end{aligned}$$

(b)

$$
\begin{aligned}
s^2(x) &= E[||t - E[t|x]||^2|x] \\
&= \int (t - E[t|x])^2 p(t|x) dt \\
&= \int (t - \sum_{k=1}^{K} \pi_k \mu_k)^2 \sum_{k=1}^{K} \pi_k \mathcal{N}(t|\mu_k, \sigma_k^2) dt \\
&= \sum_{k=1}^{K} \pi_k \int (t - \mu_k)^2 \mathcal{N}(t|\mu_k, \sigma_k^2) dt \\
&= \sum_{k=1}^{K} \pi_k (\sigma_k^2 + ||\mu_k - \sum_{l=1}^{K} \pi_l \mu_l||^2)
\end{aligned}
$$

Question 4

Yes. The weight for A is 1 and the weight for B is -1 with a bias of -0.5. The decision boundary is $A - B - 0.5 = 0$.

Question 5

(a)

There are $3 \times 4 \times 4 = 48$ parameters in the convolutional layer.

(b)

There are $3 \times 5 \times 5 = 75$ ReLU operations.

(c)

There are 48 parameters in the convolutional layer and $3 \times 5 \times 5 \times 4 = 300$ parameters in the fully connected layer. Thus, there are $48 + 300 = 348$ parameters in total.

(d)

True

(e)

With the same layer size, the convolutional neural network has fewer parameters than the fully connected neural network. Thus, the convolutional neural network is less likely to overfit and easier to train.

Question 6

(a)

Since the old function is

$$Y = \begin{bmatrix} w_5 & w_6 \end{bmatrix} \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

It can be simplified as

$$Y = \begin{bmatrix} w_5 * w_1 + w_6 * w_2 & w_5 * w_3 + w_6 * w_4 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

Let the two inputs directly connected to the output unit. And the new weights are $C * (w_5 * w_1 + w_6 * w_2)$ and $C * (w_5 * w_3 + w_6 * w_4)$.

(b)

No, such neural network cannot capture any non-linear relationship in the input data.

(c)

To make the neural network behaves like a XOR gate, it should satisfy the following conditions (generated by truth table)

$$\begin{aligned} w_5\sigma(w_1 + w_3) + w_6\sigma(w_2 + w_4) &\leq 0 \\ w_5\sigma(w_1) + w_6\sigma(w_2) &> 0 \\ w_5\sigma(w_3) + w_6\sigma(w_4) &> 0 \end{aligned}$$

By randomly guessing the weights, we can find that the weights $w_1 = 8$, $w_2 = 1$, $w_3 = 7$, $w_4 = 1$, $w_5 = 6$, $w_6 = -7$, satisfy the conditions.

```
import numpy as np

def sigmoid(x):
  return 1 / (1 + np.exp(-x))
def threshold(x):
  return np.where(x > 0, 1, 0)
def random_init():
  w = np.random.randint(-10, 10, size=6)
  return w
def nn(w, x):
  w_1, w_2, w_3, w_4, w_5, w_6 = w
  x_1, x_2 = x
  z_1 = w_1 * x_1 + w_3 * x_2
  a_1 = sigmoid(z_1)
  z_2 = w_2 * x_1 + w_4 * x_2
  a_2 = sigmoid(z_2)
  return threshold(w_5 * a_1 + w_6 * a_2)
```

```
XOR_x = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
XOR_y = np.array([0, 1, 1, 0])
while True:
  flag = False
  w = random_init()
  for x, y in zip(XOR_x, XOR_y):
    if nn(w, x) != y:
      flag = True
      break

  if not flag:
    break
print(w)
```