Name: QIU Yaowen
ID : 20784389

Q1

(a)

$P(y = c_i \mid \mathbf{x}) \propto P(x_1 \mid y = c_i) * P(x_2 \mid y = c_i) * P(y = c_i)$

$P(y = 1) = 5/8$ $P(y = 0) = 3/8$

$P(x_1 = 0 \mid y = 0) = 0$ $P(x_1 = 1 \mid y = 0) = 1$ $P(x_1 = 0 \mid y = 1) = 4/5$ $P(x_1 = 1 \mid y = 1) = 1/5$

$P(x_2 = 0 \mid y = 0) = 2/3$ $P(x_2 = 1 \mid y = 0) = 1/3$ $P(x_2 = 0 \mid y = 1) = 3/5$ $P(x_2 = 1 \mid y = 1) = 2/5$

(b)

For instances 1:

The posterior probability of y= 1:

$P(x_1 = 0 \mid y = 1) * P(x_2 = 0 \mid y = 1) = 0.8 * 0.4 = 0.32$

The posterior probability of y= 0:

$P(x_1 = 0 \mid y = 0) * P(x_2 = 0 \mid y = 0) = 0 * (2/3) = 0.67$

For instances 2:

The posterior probability of y= 1:

$P(x_1 = 1 \mid y = 1) * P(x_2 = 1 \mid y = 1) = 0.2 * 0.6 = 0.12$

The posterior probability of y= 0:

$P(x_1 = 1 \mid y = 0) * P(x_2 = 1 \mid y = 0) = 1 * (1/3) = 0.33$

Q2

(a)

$$\mathrm{Var}(\bar{h}) = Var\left(\frac{1}{K}\sum_{k=1}^{K} h_k\right) = \frac{1}{K} * \frac{1}{K} * Var\left(\sum_{k=1}^{K} h_k\right) = \frac{1}{K} * \frac{1}{K} * \sum_{k=1}^{K} Var(h_k) = \frac{1}{K} * \frac{1}{K} * K\sigma^2$$

$$= \frac{1}{K}\sigma^2$$

(b)

1. Given a set of Data with size = N, at each iteration i, a training set D_i with size = N will be obtained through sampling with replacement from D, is called bootstrap.
2. A classifier / Regressor M_i is learned from each D_i
3. When all the M_i is trained, the prediction is based on the majority voting of the prediction from M_i (for classification) or the mean value of the prediction from M_i (for regression).

The bagging method trains the sub-classifier using data through sampling with replacement, thus, the training data in each iteration is i.i.d. when the aggregated model $\bar{h}$ makes a prediction, it considers each prediction from all sub-classifier h_k, from (a) we know $\mathrm{Var}(\bar{h}) = \frac{1}{K}\sigma^2$. Therefore,

the bagging method can reduce the variance.

Q3

(a)

$$g(z) = \tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$$

Output of h_{11}:

$H_{11} = w_{11}^{(1)} * x_1 + w_{21}^{(1)} * x_2 = -1$

$U_{11} = -0.7616$

Output of h_{12}:

$H_{12} = w_{12}^{(1)} * x_1 + w_{22}^{(1)} * x_2 = 1$

$U_{12} = 0.7616$

Output of h_{21}:

$H_{21} = w_{11}^{(2)} * U_{11} + w_{21}^{(2)} * U_{12} = 1.5232$

$U_{21} = 0.9093$

Output of h_{22}:

$H_{22} = w_{12}^{(2)} * U_{11} + w_{22}^{(2)} * U_{12} = 1.5232$

$U_{22} = 0.9093$

Output of z:

$Z = U_{21} * w_1^{(3)} + U_{22} * w_2^{(3)} = 0.9093 + 0.9093 = 1.8186$

Sigmoid(z) = 0.8604

The distribution is Bernoulli distribution where

$P(y \mid x_1 = 1, x_2 = 2, \theta) = \text{Ber}(y|\text{sigmoid}(z))$

(b)

$$\frac{\partial L}{\partial z} = \sigma(z) - y = \sigma(z) = 0.8604$$

The derivative of tanh activation is:

$$g'(x) = 1 - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2} = 1 - \tanh(x) * \tanh(x)$$

Error for weights between two hidden layers:

$$\delta_{21} = \frac{\partial u_j}{\partial z_j} \sum_k W_{kj} \delta_k = (1 - 0.9093^2) * 1 * 0.8604 = 0.1490$$

$$\delta_{22} = \frac{\partial u_j}{\partial z_j} \sum_k W_{kj} \delta_k = (1 - 0.9093^2) * 1 * 0.8604 = 0.1490$$

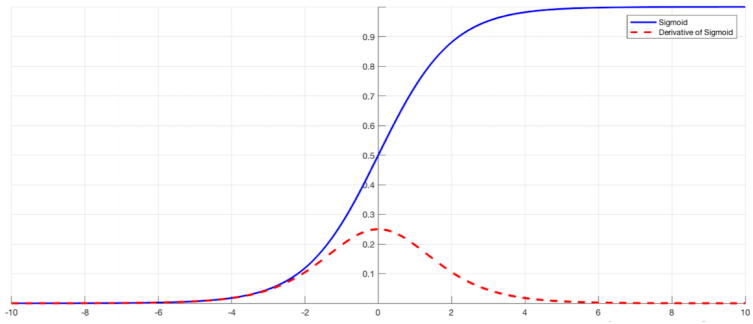
Error for weights between input and first hidden:

$$\delta_{11} = \frac{\partial u_j}{\partial z_j} \sum_k W_{kj}\delta_k = (1 - 0.7616^2)(-1 * 0.1490 * 2) = -0.1251$$
$$\delta_{12} = \frac{\partial u_j}{\partial z_j} \sum_k W_{kj}\delta_k = (1 - 0.7616^2)(1 * 0.1490 * 2) = 0.1251$$

$$\frac{\partial L}{\partial w_{22}^{(2)}} = u_{12} * \delta_{22} = 0.7616 * 0.1490 = 0.1135$$
$$\frac{\partial L}{\partial w_{22}^{(1)}} = \text{x2} * \delta_{12} = 2 * 0.1251 = 0.2502$$

We should both decrease these two parameters since their gradients all greater than zero.

Q4



The figure demonstrates the curve of sigmoid function and its derivative.

The sigmoid unit has small gradients across most of its domain, a neuron predominantly outputs values close to the asymptotic ends of the bounded activation function, which will lead to slow learning. Therefore, it is not recommended for hidden units.

However, it is fine for the output unit because the negative log-likelihood function helps to avoid the problem:

When $\sigma(z) - y$ close to 0, means z $\gg$ 0, y = 1 OR z $\ll$ 0, y = 0, in this case, slow learning only occurs when the model gives the right answer.

Q5

For each minibatch in training, randomly sample values for mask variables with a certain probability. Then, during the process of parameters updating, only parameter for the units with mask value 1 are updated. The others have gradient zero.

At each step, training is only conducted in a part of network, thus it can reduce overfitting by preventing complex co-adaptations of parameters on the training data.

Q6

It combines the idea of Momentum $s \leftarrow \rho_1 s + (1 - \rho_1)\boldsymbol{g}$ and RMSProp $\rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$ with bias correction to calculate the updated term of weights. The momentum reduces the variation in overall gradient directions and speeds up learning. And RMSProp uses an exponentially decaying average to discard history from the past while controlling the parameters which have changed a lot to change less. Adam takes these two ideas into the updated term and fix the bias.