CIE6032 and MDS6232: Homework 1

Jian Zhan, 219012058

March 6, 2020

1. Answer to question 1

   (1)

   Objective function of cross-entropy with softmax activation function:

$$\mathbf{J}(\mathbf{w}) = -\sum_{k=1}^{c} y_k log(z_k)$$

   we first consider the derivative of softmax function F with respect to weights:

   If $i \neq j$,

$$\frac{\partial F(z_i)}{\partial z_j} = \frac{-\mathbf{e}^{z_i}\mathbf{e}^{z_j}}{(\sum_{j=1}^{c}\mathbf{e}^{z_j})^2} = -z_i z_j$$

   if $i = j$,

$$\frac{\partial F(z_i)}{\partial z_i} = \frac{\mathbf{e}_i^z * \sum_{j=1}^{c}\mathbf{e}^{z_j} - (\mathbf{e}^{z_i})^2}{(\sum_{j=1}^{c}\mathbf{e}^{z_j})^2} = z_i(1 - z_i)$$

   Further, we consider the sensitivity of unit j,

$$\delta_j = -\frac{\partial \mathbf{J}(w)}{\partial z_j} = \frac{y_j}{z_j} * z_i(1 - z_i) + \frac{y_j}{z_j} * \sum_{j \neq i}^{c}(-z_i z_j) = y_j - z_j$$

   The gradient of hidden-to-output weights,

$$\frac{\partial \mathbf{J}(w)}{\partial w} = -\delta_j * y_j = y_j(z_j - y_j)$$

   Denotes net activation for hidden unit as net_k, it computes the weighted sum of its input. And denotes $f(x)$ as activation function.

   Sensitivity for a hidden unit k,

$$\delta_k = -\frac{\partial \mathbf{J}(w)}{\partial net_k} = f^{'}(net_k)\sum_{j=1}^{c} w_{jk}\delta_j$$

   The gradient of input-to-output weights,

$$\frac{\partial \mathbf{J}(w)}{\partial w} = -\delta_k * x_i = -x_i(f^{'}(net_k)\sum_{j=1}^{c} w_{jk}\delta_j)$$

(2)

If we make a single observation, and we observe outcome j, then the likelihood is simply $\hat{y}_j$. If we represent the actual observation as a vector $\mathbf{y}$ with one-hot encoding (i.e., the jth element is 1 and all other elements are 0 when we observe the jth outcome).
Since the training samples are independent, then the likelihood of the same single observation can be represented as $\prod_{j=1}^{N}(\hat{y}_j)^{y_j}$, since each term in the product except that corresponding to the observed value will be equal to 1.
The negative log likelihood is then

$$-\sum_{j=1}^{N} y_j log(\hat{y}_j)$$

which is equivalent to the cross entropy.

2. Answer to question 2

(1)

$$h_{11} = f_{11}(x) = \begin{cases} -1 & x \leq 0.5 \\ 1 & x > 0.5 \end{cases}$$

$$h_{12} = f_{12}(x) = \begin{cases} 1 & x \leq 0.5 \\ -1 & x > 0.5 \end{cases}$$

$$g(h_{11}, h_{12}) = \begin{cases} 1 & h_{11}h_{12} = -1 \\ -1 & h_{11}h_{12} = 1 \end{cases}$$

(2)

Add one more layer between input and existed hidden layer,

$$h_{21} = f_{21}(x) = \begin{cases} f_{11}(x) & x \leq 1 \\ f_{11}(2-x) & x > 1 \end{cases}$$

$$h_{22} = f_{22}(x) = \begin{cases} f_{12}(x) & x \leq 1 \\ f_{12}(2-x) & x > 1 \end{cases}$$

Then, $h_{11} = f_{11}(h_{21})$, $h_{12} = f_{12}(h_{22})$, output remains$g(h_{11}, h_{12})$

(3)

Obviously, the figure1(b) of decision boundaries is symmetric, and each $2 * 2$grid is rotated from the origin figure 1(a).

Draw the decision boundaries when the range of x1 and x2 to [0, 4].

+	-	-	+	+	-	-	+
-	+	+	-	-	+	+	-
-	+	+	-	-	+	+	-
+	-	-	+	+	-	-	+
+	-	-	+	+	-	-	+
-	+	+	-	-	+	+	-
-	+	+	-	-	+	+	-
+	-	-	+	+	-	-	+

(4)

The strategy of building the network structure is similar to question(2), for larger range n, we add more layers between input and existed hidden layers recursively.
The transform function can be generalized as following:

$$h_{n1} = f_{n1}(x) = \begin{cases} f_{n-11}(x) & x \leq 2^{n-1} \\ f_{n-11}(2^n - x) & x > 2^{n-1} \end{cases}$$

$$h_{n2} = f_{n2}(x) = \begin{cases} f_{n-12}(x) & x \leq 2^{n-1} \\ f_{n-12}(2^n - x) & x > 2^{n-1} \end{cases}$$

(5)

Notice that the decision boundary has similar properties as cosine function.

The network structure is the same as figure1(a), while transform functions:

$$h_{11} = f_{11}(x_1) = cos(\pi x_1)$$
$$h_{12} = f_{12}(x_2) = cos(\pi x_2)$$

$$g(h_{11}, h_{12}) = sign(h_{11} * h_{12})$$

3. Answer to question 3:

(1)

$$\nabla_W \mathbf{J} = \|h1 - h2\|_F * [\sigma^{'}(\mathbf{W}x_1 + \mathbf{b}) * x_1 - \sigma^{'}(\mathbf{W}x_2 + \mathbf{b}) * x_2] + \lambda \|\mathbf{W}\|_F$$

$$\nabla_b \mathbf{J} = \|h1 - h2\|_F * [\sigma^{'}(\mathbf{W}x_1 + \mathbf{b}) - \sigma^{'}(\mathbf{W}x_2 + \mathbf{b})]$$

(2)

$$\mathbf{W} := \mathbf{W} - \alpha * \nabla_W \mathbf{J}$$

$$\mathbf{b} := \mathbf{b} - \alpha * \nabla_b \mathbf{J}$$

(3)

W has five parameters, **b** is a parameter.
In total, there are $5 + 1 = 6$ parameters.

(4)

We have known that the sensitivity of cross-entropy loss:

$$\delta_j = y_j - \hat{y}_j$$

Then, we have

$$\nabla_{h1}\mathbf{J} = -\frac{1}{N} * \sum_{j=1}^{N}(y_j - \hat{y}_j) * W_3$$

Similarly, we have

$$\nabla_{h2}\mathbf{J} = -\frac{1}{N} * \sum_{j=1}^{N}(y_j - \hat{y}_j) * W_3$$

And for $\nabla_x\mathbf{J}$, we further calculate the derivative of h1,h2 on x.

For Relu function, we need to discuss whether $\mathbf{W}_2 x + \mathbf{b}_2$ is less than zero.

$$\nabla_x\mathbf{J} = -\frac{1}{N} * \sum_{j=1}^{N}(y_j - \hat{y}_j) * W_3 * [\sigma^{'}(\mathbf{W}_1 + \mathbf{b}_1) * \mathbf{W}_1 + \mathbf{W}_2], \quad if\ \mathbf{W}_2 x + \mathbf{b}_2 \geq 0$$

$$\nabla_x\mathbf{J} = -\frac{1}{N} * \sum_{j=1}^{N}(y_j - \hat{y}_j) * W_3 * [\sigma^{'}(\mathbf{W}_1 + \mathbf{b}_1) * \mathbf{W}_1], \quad if\ \mathbf{W}_2 x + \mathbf{b}_2 < 0$$

(5)

$\mathbf{W}_2$ is likely to train faster. It is because h2 is the Relu function, whose properties are efficient for updating the parameters. Empirically, a deep network with ReLu tended to converge much more quickly and reliably than other activation function.