ID2223
Scalable Machine Learning and Deep Learning

Review Questions 2

FALL 2021

Yuchen Gao
Weikai Zhou

November 19, 2021

KTH EECS Embedded Systems

1 Which of the following is/are true about individual tree in Random Forest?

(a) Individual tree is built on a subset of the features.

(b) Individual tree is built on all the features.

(c) Individual tree is built on a subset of instances.

(d) Individual tree is built on full set of instances.

(a) and (c) are correct, while (b) and (d) are incorrect.

Random forest builds multiple decision trees that are most of the time trained with the bagging method. The bagging method uses the same training algorithm for every estimator, but to train them on different random subsets of the training set. Therefore, (a) and (c) are correct.

2 Ensemble model estimators (such as Random Forest) in Spark have a parameter called featureSubsetStrategy. What does it do?

It specifies the number of features to use as candidates for splitting at each tree node, as a fraction of the total number of features. Possible values are:

- auto: If "auto" is set, this parameter is set based on numTrees: if numTrees == 1, set to "all"; if numTrees > 1 (forest) set to "sqrt" for classification and to "onethird" for regression.
- all: It uses all the features.
- onethird: It uses $\frac{1}{3}\times$numFeatures of the features.
- sqrt: It uses $\sqrt{\text{numFeatures}}$ of the features.
- log2: It uses $\log_2$ (numFeatures) of the features.

3 Explain why the entropy becomes zero when all class partitions are pure?

We can calculate the entropy as

$$\text{entropy(D)} = -\sum_{i=1}^{m} p_i \log_2 (p_i)$$

When the class partitions are pure, we know that every instance in D belongs to the same class, i.e., the k-th class. Therefore, we know $1 \leq k \leq m$ and $p_i = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases}$, and

$$\text{entropy(D)} = -\sum_{i=1}^{m} p_i \log_2 (p_i) = -p_k \log_2 (p_k) - \sum_{\substack{i=1 \\ i\neq k}}^{m} p_i \log_2 (p_i) = -1 \times \log_2 (1) - 0 = 0$$

4 Explain why the Gini impurity becomes zero when all class partitions are pure?

We can calculate the Gini impurity as

$$\text{Gini(D)} = 1 - \sum_{i=1}^{m} p_i^2$$

When the class partitions are pure, we know that every instance in D belongs to the same class, i.e., the k-th class. Therefore, we know $1 \leq k \leq m$ and $p_i = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases}$, and

$$\text{Gini(D)} = 1 - \sum_{i=1}^{m} p_i^2 = 1 - p_k^2 - \sum_{\substack{i=1 \\ i \neq k}}^{m} p_i^2 = 1 - 1 - 0 = 0$$

5 Assume a feedforward neural network with one hidden layer, in which the output of the hidden units and output units are computed by functions h = f(x) and out = g(h), respectively. Show that if we use linear functions in f and g, e.g., $\mathbf{h = f(x) = w_1^T x}$ and $\mathbf{out = g(h) = w_2^T h}$, then the feedforward network as a whole would remain a linear function of its input.

Suppose we have n inputs, m hidden, and z output neurons. If we use $w_{1,1}$ to $w_{1,mn}$ to represent elements in the $m \times n$ matrix w_1^T, and use $w_{2,1}$ to $w_{2,mz}$ to represent the elements in the $z \times m$ matrix w_2^T, then, we can have

$$h = w_1^T x = \begin{bmatrix} \sum_{i=1}^{n} w_{1,i} x_i \\ \sum_{i=1}^{n} w_{1,n+i} x_i \\ \dots \\ \sum_{i=1}^{n} w_{1,n\times(m-1)+i} x_i \end{bmatrix}$$

$$out = g(h) = w_2^T h = w_2^T (w_1^T x) = \begin{bmatrix} \sum_{j=1}^{m} w_{2,j} (\sum_{i=1}^{n} w_{1,n\times(j-1)+i} x_i) \\ \sum_{j=1}^{m} w_{2,m+j} (\sum_{i=1}^{n} w_{1,n\times(j-1)+i} x_i) \\ \dots \\ \sum_{j=1}^{m} w_{2,m\times(z-1)+j} (\sum_{i=1}^{n} w_{1,n\times(j-1)+i} x_i) \end{bmatrix}$$

From the equation, we can tell the output linearly depend on the input features $(x_1, x_2, ..., x_n)$.

6 What's the problem of using step function as an activation function in deep feedforward neural networks?

The step function is a non-differentiable function. In deep feedforward neural networks, we need to compute the gradient. However, we cannot compute the gradient with a non-differentiable function. This means that the gradient descent won't be able to make progresses in updating the weights. Thus, it is not possible for us to implement the backpropagation.

Also, the primitive object of the neural network is to make predictions basing on features, and the step function makes the prediction change drastically, which is what we do not desire, as it makes the prediction near the splitting line changes from time to time.

7 **Compute the value of w_2 and w_8 after the first iteration of the backpropagation in the following figure (Figure 1). Assume all the neurons use the ReLU activation function and we use squared error function as the cost function. In this figure, red and orange colors indicate the initial values of the weights and biases, while the numbers in blue show the input and true output values.**

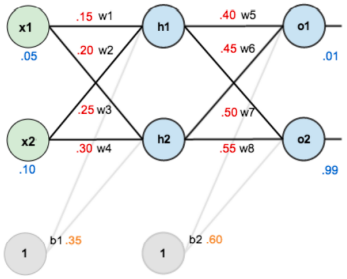

Figure 1: Figure for problem 7.

Forward Pass:

$$net_{h1} = w_1x_1 + w_2x_2 + b_1 = 0.15 \times 0.05 + 0.2 \times 0.1 + 0.35 = 0.3775$$

$$net_{h2} = w_3x_1 + w_4x_2 + b_1 = 0.25 \times 0.05 + 0.3 \times 0.1 + 0.35 = 0.3925$$

$$out_{h1} = max(0, net_{h1}) = 0.3775$$

$$out_{h2} = max(0, net_{h2}) = 0.3925$$

$$net_{o1} = w_5h_1 + w_6h_2 + b_2 = 0.4 \times 0.3775 + 0.45 \times 0.3925 + 0.6 = 0.927625$$

$$net_{o2} = w_7h_1 + w_8h_2 + b_2 = 0.5 \times 0.3775 + 0.55 \times 0.3925 + 0.6 = 1.004625$$

$$out_{o1} = max(0, net_{o1}) = 0.927625$$

$$out_{o2} = max(0, net_{o2}) = 1.004625$$

$$E_{o1} = \frac{1}{2}(target_{o1} - output_{o1})^2 = \frac{1}{2}(0.01 - 0.927625)^2 = 0.4210178203$$

$$E_{o2} = \frac{1}{2}(target_{o2} - output_{o2})^2 = \frac{1}{2}(0.99 - 1.004625)^2 = 0.0001069453$$

$$E_{total} = E_{o1} + E_{o2} = 0.4211247656$$

Backward Pass:

- w_2

$$
\begin{aligned}
\frac{\partial E_{total}}{\partial w_2} &= \frac{\partial E_{total}}{\partial out_{h1}} \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_2} \\
&= (\frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}) \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_2} \\
&= (\frac{\partial E_{o1}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{o2}} \times \frac{\partial out_{o2}}{\partial net_{o2}} \times \frac{\partial net_{o2}}{\partial out_{h1}}) \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_2} \\
&= [-(target_{o1} - out_{o1}) \times (1) \times (w_5) - (target_{o2} - out_{o2}) \times (1) \times (w_7)] \times (1) \times (x_2) \\
&= [-(0.01 - 0.927625) \times (1) \times (0.4) - (0.99 - 1.004625) \times (1) \times (0.5)] \times (1) \times (0.1) \\
&= 0.03744
\end{aligned}
$$

$$
w_2^{next} = w_2 - \eta \frac{\partial E_{total}}{\partial w_2} = 0.2 - 0.03744\eta
$$

- w_8

$$
\begin{aligned}
\frac{\partial E_{total}}{\partial w_8} &= \frac{\partial E_{total}}{\partial out_{o2}} \times \frac{\partial out_{o2}}{\partial net_{o2}} \times \frac{\partial net_{o2}}{\partial w_8} \\
&= -(target_{o2} - out_{o2}) \times (1) \times (h_2) \\
&= -(0.99 - 1.004625) \times (1) \times (0.3925) \\
&= 0.00574
\end{aligned}
$$

$$
w_8^{next} = w_8 - \eta \frac{\partial E_{total}}{\partial w_8} = 0.55 - 0.00574\eta
$$