Scalable Machine Learning and Deep Learning - Review Questions 2

Anna Martignano, Daniele Montesi

November 17, 2019

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.

2. **Ensemble model estimators (such as Random Forest) in Spark have a parameter called featureSubsetStrategy. What does it do?**
   The parameter *featureSubsetStrategy* is used when training ensemble decision trees models and it refers to the strategy to use the features during the creations of the trees. It is part of the **bootstrapping**. The possible strategies are here explained in detail:

   - "auto": Features selected with an internal strategy depending on the input dataset
   - "all": All the features. No subset is chosen. This may result useless since using all the features, only the samples chosen from the dataset will differ, and the trees might be all the same/very similar
   - "sqrt": select $sqrt(num\_features)$ features
   - "log2": select $log_2(num\_features)$ features
   - "onethird". select num_features/3 features
   - Any integer $N \geq 2$ is accepted, and will get N features as subset to create the tree

3. **Explain why the entropy becomes zero when all class partitions are pure?**
   The formula of Entropy used in Decision Trees is the following:

$$Entropy(D) = -\sum_{i=1}^{m} p_i \log(p_i)$$

Where m is the total number of classes, p_i is the probability that a sample in the dataset D is belonging to class i.
In the case in which all class partitions are pure, we have that p_i will be 1 only for the corrected labelled class, let's say named **j**, and 0 for all the other classes. Being the formula a summation of terms multiplied by $\log(p_i)$, we obtain that:

$$Entropy(D) = -\sum_{i=1}^{m} p_i \log(p_i) = -1 \log 1 - \sum_{i=1}^{m-[j]} 0 = 0$$

Where m - [j] means all the possible values until i in (1, 2, 3, ... m), and i!=j

4. **Explain why the Gini impurity becomes zero when all class partitions are pure?**
The formula of Gini impurity is the following:

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

Where m is the total number of classes, p_i is the probability that a sample in the dataset D is belonging to class i.

In the case in which all class partitions are pure, we have that p_i will be 1 only for the corrected labelled class, let's say named **j**, and 0 for all the other classes. Being the formula a summation of terms multiplied by $\log(p_i)$, we obtain that:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2 = 1 - 1^2 - \sum_{i=1}^{m-[j]} 0^2 = 0$$

Where m - [j] means all the possible values until i in (1, 2, 3, ... m), and i!=j

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. $h = f(x) = w_1^T x$ and $out = g(h) = w_2^T h$, then the feedforward network as a whole would remain a linear function of its input.**

The formula for the output of a feedforwards neural network presented in the task assignment is the following:

$$y_1(x) = g((\sum_{j=0}^{J} W_j h(\sum_{i=0}^{I} w_i x_i)))$$

$$y_2(x) = g((\sum_{j=0}^{J} W_j h(\sum_{i=0}^{I} w_i x_i)))$$

$$...$$

$$...$$

$$y_O(x) = g((\sum_{j=0}^{J} W_j h(\sum_{i=0}^{I} w_i x_i)))$$

Given O all the set of possible outputs, J all possible hidden neurons, I all possible inputs; w_i weights from inputs to hidden layer neurons, W_j weights from hidden layer neurons to Output neuron.
If we substitute the following formulation $h = f(x) = w_1^T x$ and $out = g(h) = w_2^T h$ in the formula of $y_1(x)$, we obtain:

$$y_1(x) = \sum_{k=0}^{J} w_{2_k}((\sum_{j=0}^{J} W_j \sum_{h=0}^{I} w_{1_h}(\sum_{i=0}^{I} w_i x_i)))$$

We can notice that w1 is a Matrix of shape [IxJ] made of the weights connecting the inputs and the hidden layers, and w2 is a vector of the weights connecting the hidden layers' neuron to the output (let's say, output 1). Proceeding with the calculation, we will eventually obtain a series of multiplications depending linearly only on the inputs $x_1, x_2, ...x_I$.

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

   The step function is a threshold-based activation function. If the input value is above or below a certain threshold, the neuron is activated and sends exactly the same signal to the next layer. Therefore, the output will be binary, 1 or 0. Moreover, such activaction function is non-continous and therefore non differentiable. Its derivative is not possible to compute and the backpropagation would not work. (Note: we could solve the issue as in the Perceptron with binary classification, where we define a sign() function whose derivative is set multiplied by a term $-t$, where t is the target [0 or 1]. Adding the minus (-) is a trick to make the loss increasing when the sign is discordant).

7. **Compute the value of w_2 and w_8 after the first iteration of the backpropagation in the following figure. 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.

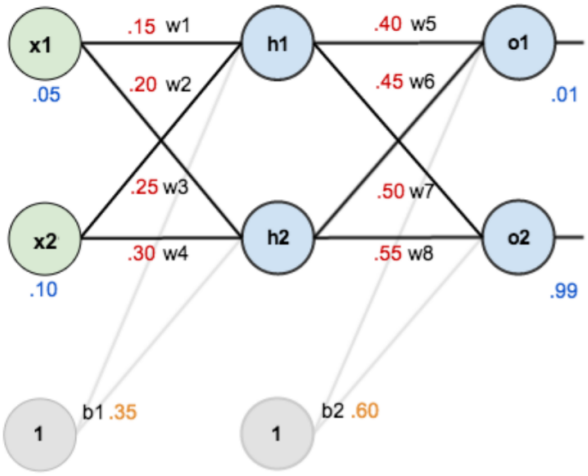

Figure 1: Back Propagation

FORWARD STEPS
Compute the output of the hidden layer

$$\begin{aligned} net_{h1} &= w_1 x_1 + w_2 x_2 + b_1 = 0.15 \times 0.05 + 0.2 \times 0.1 + 0.35 = 0.3775 \\ net_{h2} &= w_3 x_1 + w_4 x_2 + b_1 = 0.25 \times 0.05 + 0.3 \times 0.1 + 0.35 = 0.3925 \\ out_{h1} &= max(0, net_{h1}) = 0.3775 \\ out_{h2} &= max(0, net_{h2}) = 0.3925 \end{aligned} \tag{1}$$

Compute the output of the output layer

$$\begin{aligned} net_{o1} &= w_5 out_{h1} + w_6 out_{h2} + b_2 = 0.4 \times 0.3775 + 0.45 \times 0.3925 + 0.6 = 0.927625 \\ net_{o2} &= w_7 out_{h1} + w_8 out_{h2} + b_2 = 0.5 \times 0.3775 + 0.55 \times 0.3925 + 0.6 = 1.004625 \\ out_{o1} &= max(0, net_{o1}) = 0.927625 \\ out_{o2} &= max(0, net_{o2}) = 1.004625 \end{aligned} \tag{2}$$

Compute the error for each output

$$\begin{aligned} E_{o1} &= \tfrac{1}{2}(target_{o1} - output_{o1})^2 = \tfrac{1}{2}(0.01 - 0.927625)^2 = 0.42101782 \\ E_{o2} &= \tfrac{1}{2}(target_{o2} - output_{o2})^2 = \tfrac{1}{2}(0.99 - 1.004625)^2 = 0.000106945 \\ E_{total} &= \sum \tfrac{1}{2}(target - output)^2 = E_{o1} + E_{o2} = 0.421124765 \end{aligned} \tag{3}$$

BACKWARD STEPS
Since we do not know which is the value of the learning rate η, we have simply reported the computation steps to be performed.

Consider w_8

$$\begin{aligned} w_8^{(next)} &= w_8 - \eta \times \frac{\partial E_{total}}{\partial w_8} \\ \frac{\partial E_{total}}{\partial w_8} &= \frac{\partial E_{total}}{\partial out_{o2}} \times \frac{\partial out_{o2}}{\partial net_{o2}} \times \frac{\partial net_{o2}}{\partial net_{w_8}} \end{aligned} \tag{4}$$

Consider w_2

$$\begin{aligned} w_2^{(next)} &= w_2 - \eta \times \frac{\partial E_{total}}{\partial w_2} \\ \frac{\partial E_{total}}{\partial w_2} &= \frac{\partial E_{total}}{\partial out_{h1}} \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial net_{w_2}} \end{aligned} \tag{5}$$