Paper Q&A

1. **What is the main problem that the paper proposes to solve?**
    - Neural networks develop an internal structure to learn a particular task. As some intermediate(or hidden) layers are introduced between the input and output layer, the learning procedure should also decide when those hidden layers are activated, or in other words, what those hidden layers represent. The backpropagation method proposed in this paper is simple but powerful for the hidden layers to learn representation.
2. **Describe the artificial neural network in terms of Equation 1) and 2).**
    - Artificial neural network is a set of layers, where the inputs of each hidden layer are represented as a linear combination of the outputs from the previous hidden layer and the weights. In addition, the outputs of a certain layer is calculated by a non-linear function, known as a sigmoid function.
    - There is one constraint: the output function should have a bounded derivative.
3. **Equation 3), what is the total error and why do we need it?**
    - In the equation, c denotes the cases, or data points, and j denotes each unit of the output layer.
    - Then the total error is sum of the squared difference between the actual estimate and the desired(or true) state among all units and among all cases(or observations).
    - The reason why we need the total error, or in other words, loss function, is that we need to minimize the total error so that we can find a set of parameters making the estimate sufficiently close to the desired(or true) state.
4. **Explain Equation 4-6. Why did the authors derive them and how are they related to the chain rule.**
    - With the given input and output, the actual output state and the loss are determined by the weight w_{ji} in each layer. Thus the total error E can be seen as a function of weights.
    - Here we need to obtain the $\arg\min_{w_{j,c}} E$, and since the total error is quadratic thus differentiable, we compute the partial derivative of E with

respect to the weights.

- Mathematically, we need to obtain $\frac{\partial E}{\partial w_{ji}}$. Here, since x_j is a function of w_{ji}, y_j is a function of x_j, and E is a function of y_j, we can obtain $\frac{\partial E}{\partial w_{ji}}$ using the chain rule. Specifically, the formula is like below.

$$\begin{aligned}\frac{\partial L}{\partial w_{ji}} &= \frac{\partial L}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ji}} \\ &= (y_j - d_j) \cdot (y_j \cdot (1 - y_j)) \cdot y_i\end{aligned}$$

5. **Explain Equation 8 and 9.**

- The above formula is all about the computing gradient with respect to a particular weight w_{ji}. However, by this particular w_{ji}, the value y_j, the actual state, is already determined and with this particular weight, the equation $(y_j - d_j) \cdot (y_j \cdot (1 - y_j)) \cdot y_i$ might not be zero.
- Therefore, we need to update the weight and re-compute the whole procedure, until the equation becomes (sufficienly close to) zero for every weight.
- This method is called the Gradient Descent, and the equation 8 shows the algorithm of updating the weight. Δw represents the amount of change in the weight, or in other words, difference between the original weight and the updated weight. The right-hand side, denoting the amount of update, is the negative gradient of the weight, muliplied by a scalar ϵ.
    - For a convex function, if the gradient at a particular point is positive, it means that the point is positively far from, or greather than, the $\arg\min$. In that case, we need to move the point leftward. On the other hand, if the gradient is negative, it means that the point is less than the $\arg\min$, thus we need to move it rightward. Furthermore, the scalar ϵ prevents us from moving the point too much at a time. This is why the amount of update is $-\epsilon \cdot \frac{\partial E}{\partial w}$
    - and ϵ here is well known as a learning rate.
- As a variation, we can update the weight following the equation 9. Unlike the equation 8, this algorithm tries to reflect the contribution of the earlier gradients to the weight change.
    - Specifically, since the algorithm follows a recursive process and the coefficient $\alpha \in (0, 1)$, the impact of earlier gradients becomes less as

the update process continues. This is why the name of α is an exponential decay factor.

6. **What is the main message in Figure 4?**
    - The main message in Figure 4 is that by adjusting the magnitude of each weight in a particular layer, the deep learning model can learn what the layer represents. After the adjustment is done, each weight indicates the importance of its corresponding feature.
7. **Which part is the most impressive in the paper?**
    - Before reading this paper, I had no idea that the most deep learning models, with this simple calculation. The whole procedure of algorithm, such as iteratively computing the gradient at each weight and update it with the proportional amount of that gradient was impressive. Moreover, the fact that we can obtain the predicted values with high accuracy using this simple logic was more impressive.