A Trend Of Moving Deep Learning Towards Multi-Agent Setting

Lisheng Wu

I. ABSTRACT

This article aims at analysing neural network as the hierarchical multi-agent decision system and tries to figure out the key points for more complex behaviours from CapsNet[2].

II. BACK PROPAGATION

For each layer in deep neural network trained with back propagation[1], the weights change in the way to minimise the final loss. Use f_i^l to represent the i_{th} neuron activation on layer l. In most neural networks, we have

$$f_j^{l+1} = \sum_i f_i^l w_{ij}^l \tag{1}$$

Though we get different values $f_i^l w_{ij}^l$ from the layer below, we push each equally when we apply the gradients.

$$s_{ij}^l = f_i^l w_{ij}^l \tag{2}$$

$$\frac{\partial L}{\partial f_j^{l+1}} = \frac{\partial L}{\partial s_{ij}^l}, \forall i \tag{3}$$

The features at each layer often only make sense as a whole vector because we treat each element equally. For example, we can't divide the features into halves and expect each half to represent something.

III. MULTI AGENT SETTING

Consider each neural network connection as an agent and the neuron activation as the state, the agent connecting the i_{th} neuron on layer l and the j_{th} neuron on layer $l+1$ makes a constant continuous action a_{ij}^l to any state s_i^l it encounters. Then, we have

$$s_j^{l+1} = \sum_i s_i^l a_{ij}^l = \sum_i f_i^l w_{ij}^l \tag{4}$$

State s^{l+1} is passed to the next layer to be dealt with. Between every two subsequent layers, there could be thousands of such agents taking actions simultaneously. Finally, we get the loss L and do the credit assignment layer by layer through back propagation. Through back propagation, each agent gets it own credit from the layer above. As we could see in equation 3 that the gradients push each component s_{ij}^l equally, so there is only uniform credit assignment during the back propagation. Then we can regard those thousands of agents between two layers as one single agent with a continuous action space which has thousands of dimensions because it doesn't make sense to distinguish them.

This credit setting could be too naive in updating the agent's actions because both s_j^{l+1} and a_{ij} are one dimensional and the change in a_{ij} could only increase or decrease s_j^{l+1}. If we could extend both the state s_j^{l+1} and action a_{ij} to multi-dimensional, s_j^{l+1} could have more freedom in changes and we could also have more complex updates in actions from the credits assigned, like involving the cosine similarities in higher dimensional space. In other words, the credit assigned could be more expressive.

IV. MODEL COMPLEXITY

If we increases the width or depth of one neural network layer, the model complexity as well as the model capacity increase. When we increase the model complexity, it is easier to find a local minimum and has the potential to find a better solution. However, neural network could also reach a bad local minimum.

We might choose to use some regularisation techniques to alleviate this problem and still benefit from the model complexity. Some regularisation techniques could reduce the model complexity or the parameter search space during each forward pass. For example, (1)Dropout reduces the model complexity by omitting parts of the features. (2)Weight decay directly limits the parameter space to search from.

It could be more straightforward to understand the parameter search space as the continuous action space as we talked in previous section. In one way, we could associate the action space size with the model complexity.

V. PROS AND CONS

Cons: There are three potential shortcomings of the traditional deep learning. At first, when the network has a large action space to search, it is easier to cause overfitting. Second, there is only uniform credit assignment during the back propagation and the whole feature vector only makes sense as a whole, which makes it more difficult to learn structured components and complex interactions among them. Third, all the agents make constant actions regardless of the state, which could limit the network's capability.

Pros: It is easier to find a local minimum on training dataset in a large parameter space.

VI. ANALYSIS

We could divide the feature vector and weights of each layer to multiple groups if we introduce some dependencies to the neurons for each layer. Each group has its own feature

representation called as a *group embedding* and its own *group action*. We could search in different group action space in parallel while remaining the expressive ability of the network, i.e. keeping the model complexity while factor the search space into several smaller ones. Moreover, each group has a smaller parameter space to search parameters, it would reduce the possibilities of overfitting.

Each group can also be considered as an agent which has its own action space and can interact with other groups to form more complex behaviours.

However, even if we divide the search space into different groups, we can't ensure each group can learn meaningful group embedding. The reason is that back propagation only does uniform credit assignments. In other words, each group doesn't receive its own learning signal from traditional back propagation but only learn how to make the whole feature vector meaningful.

VII. Capsule Network

Capsule Network[2] proposed by Hinton introduced capsules as groups. Each layer is composed of several capsules while each capsule C_i^l in layer l has its own weights for computing output $u_{j|i}^l$ corresponding to each capsule C_j^{l+1} in layer $l+1$. The routing process in fig.1 is like a recurrent architecture. The rightmost summation in fig.1 takes all the capsule outputs as inputs where

$$v_j^{l+1} = squash(s_j^l) = \sum_i c_{ij}^l u_{j|i}^l \tag{5}$$

In capsule setting, we consider each agent as the combination of the capsule and the routing process. For the i_{th} agent at layer l, its action is $(w_{C_i^l}, \{c_{ij}\}_{j=1...N^{l+1}})$, consisting of the weights of capsule C_i^l and the routing coefficients decision. What's more, the routing coefficients process involves the communication among all capsules in layer l because equation 5 is used to update the routing coefficients.

In equation 6, the c_{ij}^l in $c_{ij}^l \frac{\partial L}{\partial v_j^{l+1}}$ does ununiform credit assignment during the back propagation for the capsule output $u_{j|i}^l$, which helps the capsule learn meaningful feature embedding. As for $\frac{\partial c_{ij}^l}{\partial u_{j|i}^l}$ in equation 6, it needs to back propagate through the recurrent architecture as fig.1 shows and the process involves all the capsules. Refer to the paper[2] for more details.

$$\begin{aligned} \frac{\partial L}{\partial v_j^{l+1}} &= \frac{\partial L}{\partial c_{ij}^l u_{j|i}^l} \\ \frac{\partial L}{\partial u_{j|i}^l} &= \frac{\partial L}{\partial v_j^{l+1}} \frac{\partial c_{ij}^l u_{j|i}^l}{\partial u_{j|i}^l} \\ &= \frac{\partial L}{\partial v_j^{l+1}} \left(c_{ij}^l + \frac{\partial c_{ij}^l}{\partial u_{j|i}^l} \right) \end{aligned} \tag{6}$$

CapsNet makes it possible for each agent to interact with each other to form more complex behaviours. Moreover, as we talked in **Section III**, because of the cosine similarity distance, all weights in a capsule are also updated together in a compact space from the credit assigned.

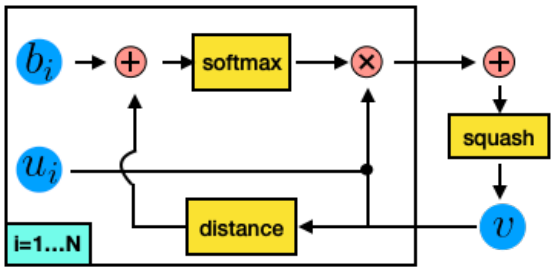

Fig. 1. Visualisation of Dynamic Routing Process in CapsNet

VIII. Gradient Comparison

From equation (3), we also know

$$\frac{\partial L}{\partial f_i^l} = \sum_j w_{ij}^l \frac{\partial L}{\partial f_j^{l+1}} \tag{7}$$

In traditional back propagation, $\frac{\partial L}{\partial f_j^l}$ is independent of $\frac{\partial L}{\partial f_i^l}$ conditioned on L and the weights of higher level $w_{L \geq l}$. This also infers that the gradient decent direction was almost controlled only by the final loss during the back propagation except nonlinear function like *softmax*. We can define this kind of gradients as *loss guided gradient*. However, in capsule setting, the dependency introduced makes the updates of all weights become more dependent. In equation 6, the gradients to $u_{j|i}^l$ would involves all the capsule outputs because of the part $\frac{\partial c_{ij}^l}{\partial u_{j|i}^l}$. This kind of gradients is called *peer aware gradient*.

Instead of looking for a local minimum, we could also understand that all capsules are looking for the Nash equilibrium that each capsule doesn't want to change its weights, then both the activation and the routing coefficients won't change any more.

IX. Conclusion

In summary, the CapsNet has the following advantages:

1) The capsule as group reduce the parameter space to search and help avoid overfitting.
2) The ununiform credit assignments help each capsule learn meaningful features and they can interact with each other.
3) The cosine similarity calculation involves all capsules and makes the updates from the credit/gradients to be more compact rather than just guided by the final loss and the weights of above layers.

In the same time, CapsNet also has more restrictions are added to the neural network updating which makes it more difficult to train.

References

[1] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[2] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.