Basic information **Learning representations by back-propagating errors**
Author: David E. Rumelhart(USCD), Geoffrey E. Hinton(CMU), Ronald J. Williams(UCSD). Published on Nature Vol. 323 9 Oct 1986 Page 533-536.

Summary of the paper This paper mainly introduces backward propagation for self-weight-adjusting in neural networks. After forward propagation, applying chain rules to compute derivative from the output layer to the input layer and then applying gradient descent enables multi-layer networks to update the weights according to the error between prediction and desired output. This structure can handle problems that perceptrons cannot, such as symmetry detection and family relation learning. The main problem of back-propagation is it is not guaranteed to convege to the global minimum.

Comments .

Related work Before back-propagation, people mainly use perceptrons and forward propagation for machine learning. The problem of perceptrons is that they require hand-fixed connection instead of self-learned representations. While the main problem of forward propagation is that computing partial derivative for each weight is too expensive. By introducing back-propagation from control theory, the author solve these two problems.

Strong points
1. **Applicability.** By giving an example, the author state that recurrent networks can be converted into a layered network. In this way, we can also apply back-propagation to sequence problems.
2. **Parallel-ability.** Historically, people focus more on parallel distributed processing. Update information is stored in the neuron units so back propagation requires no separate graph-structured memory. In this way, it is computationally simpler than methods using second derivatives and can be easily adapted to parallel version. This also reduce the computational complexity from forward propagation.
3. **Accelerate.** Though first-order derivative will leads to slower convergence than methods using second order derivative, we can use velocity to accelerate the training process. This encourages momentum and other optimization algorithms.

Weakness and improvement
1. **Stuck in a local minimum.** SGD using back-propagation is guaranteed to give the optimal only when the loss function is convex and differentiable. However, neural networks with loss functions are usually not convex problems. The obvious drawback is that gradient descent may converge to a local minimum on non-convex problems. However, adding more than enough connections will provide paths around the barriers outside poor local minima. Another reasonable argument is that in high dimension, we are more likely to find a saddle point instead of a local minimum. There will always be a way out the valley. This algorithm also has trouble traveling across plateaus (However, Yann LeCun thinks it is not a major issue). In practice, SGD will converge to a local minimum but not obviously worse than the global minimum, which means back propagation seems to perform well on most networks. However, the effectiveness hasn't been proved theoretically. Lots of researches have been done to determine which problem can be converted to convex problems, in which problems local minimums are similarly good as global minimum.
2. **Not biological plausibility.** Back in the '80s, people still focus more on biological plausibility. So one drawback of back-propagation is that there is no reasonable biological explanation for doing back-propagation. But now, since thousands of amazing tasks have been solved due to back propagation, we focus more on the ability for stimulating a certain behavior instead of its biological plausibility. Maybe it will be better if we try to understand the human brain and then understand more about the neural networks.
3. **Loss function and activation function choosing.** In this paper, the authors use sigmoid as activation function and least square error as loss function. Later in the practice, people find that sigmoid function will lead to gradient vanish. The least square error loss function is not always reasonable for some tasks. Although the authors said they are not the only choice, it had a great influence on the research community then.
4. **Initial value and normalization.** The author said that we should use small random numbers as initial weights to break symmetry. However, they didn't give the reason why we should use small initial values. In practice, people find it will face gradient explosion with large initial weights. Back-propagation does not require normalization. However, we usually have to normalize the input to avoid unbalanced features.
5. **Experiments.** The to examples given in the paper, symmetry detection and family relation learning, are all from artificial data. Thus, is not strong enough to convince the effectiveness on real data.

Follow ups 1. People try lots of different loss function and activation functions later. Nowadays, people use the rectified linear unit (ReLU) as an activation function to avoid gradient vanishing.
2. Hinton himself said we should start all over and get rid of back-propagation. The reason is that it is not biologically plausible. People do not need big-data to learn, but current neural networks need abundant labeled data to train. So if we want to do unsupervised learning, there will be no way out through back-propagation.