Residual Networks

Rohit Amarnath

February 24, 2024

1 Residual Function

Suppose we wanted to fit a function $f(x)$. The usual approach with any model is to find $h(x)$ such that for any $x, h(x) \approx f(x)$. In the residual network approach, we set $h(x) = x + r(x)$ so that $r(x) \approx f(x) - x$.

Experiments show that networks trained with this approach perform better on object recognition tasks. There is no clear reason why ResNet works better. However, one line of reasoning is inspired by Taylor approximations. If $f(x)$ is approximately linear, then x is a good estimate for $f(x)$. Subsequently, it follows that a better estimate for $f(x)$ is $x + r(x)$.

ResNet was designed to solve the problem of degradation. As model depth increases, accuracy gets saturated and degrades rapidly. Sometimes, such degradation is caused by overfitting, but He et. al asserts that the degradation they experienced was not overfitting as deeper models led to higher training error [1]. The motivation for ResNet was the idea that deeper models should perform atleast as well as shallower models as any unnecessary layer in the deeper model can learn an identity mapping. However, experiments showed that deeper models struggled to do this and ended up performing worse than shallower counterparts.

It is incorrectly believed that residual blocks perform better due to mitigating the vanishing gradients. However, [1] argues that vanishing gradients is unlikely to be the reason residual blocks perform better as vanishing gradients were taken care by batch normalization. Anyhow, ResNet certainly does address the vanishing gradient, and we will see this below.

2 Backpropagation with residual layer

Let $x_1, x_2, \dots x_n$ be a set of inputs into the i-th residual block of the network. Note that for some F, we have $x_{i+1} = F(x_i) + x_i$, meaning

$$x_n = x_1 + \sum_{i=1}^{n-1} F(x_i)$$

Denote $\mathcal{L}$ to be the loss. Then,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x_1} &= \frac{\partial \mathcal{L}}{\partial x_n}\frac{\partial x_n}{\partial x_1}\\ &= \frac{\partial \mathcal{L}}{\partial x_n}\left(1 + \frac{\partial}{\partial x_1}\sum_{i=1}^{n-1} F(x_i)\right)\\ &= \frac{\partial \mathcal{L}}{\partial x_n} + \frac{\partial \mathcal{L}}{\partial x_n}\frac{\partial}{\partial x_1}\sum_{i=1}^{n-1} F(x_i)\end{aligned}$$

Notice that the gradient of the loss with respect to x_1 has a term involving the gradient of the loss with respect to x_n, preserving some of the long-term dependencies. In a sense, the gradient *skips* the layers in between while preserving this dependency; hence, residual networks are also called skip-networks.

3 Alternate residual connections

It is also worth mentioning that there is an alternative residual network structure. Earlier, we stated that $h(x) = x + r(x)$. Instead, we can perform a linear projection W on the shortcut connection, which gives us $h(x) = Wx + r(x)$. However, experiments suggest that the improvement in loss is marginal between the two variations.

References

[1] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: `1512.03385`. URL: `http://arxiv.org/abs/1512.03385`.