Residual Networks Notes

Piyush Patil

September 19, 2017

Residual networks are motivated by the desire to be able to efficiently and effectively make use of very deep neural models. However, the problem with deep networks is that the number of parameters quickly becomes very large, leading to the risk of overfitting. Further, the problems of vanishing (or exploding) gradients (depending on which activation function is used) are exacerbated by adding more layers and squashing gradients even more. Even worse, there exist many negative results that show that often deep networks perform worse than their more shallow counterparts for reasons other than overfitting - for whatever reason, deeper models simply degrade performance.

This so-called degradation problem is quite counter-intuitive, and should, in theory, not even exist, since after all a deeper model's expressivity ought to be a strict superset of that of a more shallow model. This can be seen by construction - given a shallow model with L layers, a deeper model with N layers could simply learn to converge precisely to the shallow model in its first L layers and have the final $N - L$ layers simply learn the identity mapping. In this case, both models would have the exact same accuracy. The fact that learning such models doesn't happen in practice and that instead deeper models simply perform worse, even when not overfitting, seems to suggest that the "reference mapping" learned by layers in a deep network is poor. Instead, we'd like layers to, by default, learn the identity mapping, and only learn more sophisticated mappings by necessarily adding complexity when absolutely necessary. To do so, we can take an arbitrary (meaning any number of neurons, any activation function, and any augmentation such as batch normalization) layer of the network and make it a residual layer as follows: we take the output of our layer and rather than passing it as input to the next layer, we first add in the unadulterated input first. Thus, given weights W and activation F, the output of our layer is $F(Wx) + x$ rather than simply $F(Wx)$.