Training Very Deep Networks

1 Citation

Srivastava, Rupesh K., Klaus Greff, and Jürgen Schmidhuber. "Training very deep networks." Advances in neural information processing systems. 2015.

`http://papers.nips.cc/paper/5850-training-very-deep-networks.pdf`

2 Abstract

To enable training deep networks, we introduce highway networks, which are stacks of highway modules that have a gate that decides whether how much of the layer input and layer output should be let through.

3 Introduction and Previous Work

Deeper networks have enabled us to improve accuracy on many machine learning problems (e.g. ImageNet). To deal with the difficulty of training deep nets, people have developed better optimizers, intialization strategies, skip connections, distillation (i.e. teacher network teaches student network with soft targets), and layerwise training. The key problem is that of vanishing and exploding gradients. LSTMs mitigate this problem for recurrent neural nets with their gating mechanism, so we apply the same principle here.

4 Highway Networks

A network layer computes $\boldsymbol{y} = H(\boldsymbol{x}, W_H)$ where W_H is the weight matrix and H is usually an affine transformation (or convolution) followed by a nonlinearity. For highway networks, we replace this with:

$$\boldsymbol{y} = H(\boldsymbol{x}, W_H) \cdot T(\boldsymbol{x}, W_T) + \boldsymbol{x} \cdot (1 - T(\boldsymbol{x}, W_T)) \tag{1}$$

where T is the transform gate that decides how much of the input and output should be allowed through. To make $\boldsymbol{x}$ have the same dimension as T and H, you can zero-pad it (if too short) or subsample it (if too long). You can also apply T convolutionally in the case of conv layers. We set $T = \sigma(W_T^T \boldsymbol{x} + \boldsymbol{b}_T)$. We initialize the bias with negative values so the sigmoid is near zero and thus the transform gate starts out by allowing the input to flow through. We were able to train 1000 layer networks usign this highway module described above.

5 Experiments

We train with SGD with momentum. The learning rate starts at γ and decays according to factor λ. We use ReLU activation.

We train on MNIST. We have thin networks with either 50 highway blocks or 71 plain blocks (to make sure highway and non-highway nets have about the same number of parameters). Hyperparameters were chosen with random search. Our highway networks converge faster than plain networks and outperform plain nets with the same number of layers. We are competitive with state of the art with many fewer parameters.

We also train on CIFAR-100, where we use Fitnets as our basis. These networks use maxout units and are hard to train when they are deep. We add highway blocks to these networks and are able to train them without problems. We also make them deeper and can train those as well.

6 Analysis

We initialized transform gate biases with negative values, but we find that the trained network actually makes them more negative (where deeper layer are even more negative). At the lower layers, this is becuase the network is more selective - producing transformations for specific kinds of samples and just letting others flow through. For a single example, it won't trigger many blocks, but the set of blocks triggered likely varies sample to sample. So, the gating mechanism not only eases training, but also serves as a routing mechanism that is essential for computation.

We try lesioning (i.e. set all the transform gates to zero so it just copies the input) layers. For MNIST, the early layers are essential but the layers after layer 10 have no impact on test error. This indicates that MNIST is pretty easy and thus the network doesn't need most of its layers. For the more difficult CIFAR-100, on the other hand, lesioning any layer but the last few hurts test error, which indicates the network uses almost all of its layers.

7 Discussion

Good intialization, local response normalization, and deep supervision all help train deeper models, but they struggle with very deep models. Highway networks can train very deep models, allowing the network to learn which layers to keep and which ones to ignore (i.e. close transform gate). They can also help visualize how much computation depth is needed for a problem.