Neural Network Initialization

Vladimir Feinberg

July 17, 2017

Initialization is an important aspect of ANN training. It contributes to improvements both in terms of training speed and generalization. Content is mostly from Dr. Goodfellow's Deep Learning Book.

Parameters should not be initialized uniformly so as to break symmetry between hidden units; otherwise, the parameters will remain the same during training. This motivates random initialization. Biases, however, are typically constants, heuristically chosen.

Weights are typically Gaussian or uniform, but scale is important. Large scale, which breaks symmetry strongly, must be balanced with stability of the learning algorithm. ReLU activations, since they're linear, suggest that that the scale of the output for a basic ReLU MLP with $\|y\| \sim \|\mathbf{x}\| \prod_i \left\|W^{(i)}\right\|$. In general, the scale of weights should be treated as a hyperparameter. Importantly, loss (such as softmax) not scale-invariant; keep an eye out for initialization.

Output layer bias should be initialized with the marginal statistics of the training data. Certain initialization schemes may be compatible with positive constant bias initialization, which lets gradients start on the linear part of a ReLU. (Jozefowicz et al 2015). If a unit acts as a control gate for letting other information pass (like a forget gate in an LSTM), then bias should be initialized as positive there.

1 Uniform Random Initialization

(Glorot and Bengio 2010). Normed initialization suggests that a fully connected layer initializes an $n \times m$ weight matrix entrywise with Uniform $(-u, u)$ where $u = \sqrt{\frac{6}{m+n}}$. This approach intends to keep gradient variance the same across layers, and it is very commonly used because of its simplicity.

2 Orthogonal Matrix Initialization

(Saxe et al 2013, Susillo 2014). Orthogonal matrix initialization with a scaling factor dependent on layer activation is motivated by making hidden units track a varied set of functions, resulting in training time independent of depth (if nonlinearities are the identity).

2.1 Greedy Pre-training

(Bengio et al 2007). Greedy pre-training is a form of initialization where a FFNN is trained by first training a shallow network to completion, then appending new layers after lower layers are trained. This greedy form of optimization isn't optimal from a training loss perspective, but in addition to speeding up convergence it can be viewed as a form of regularization. However, it has fallen out of favor as other modern techniques (ReLU, dropout, BN, unsupervised pre-training) have been developed.

2.2 Unsupervised Pre-training

(Erhan et al 2010, Paine et al 2014). Unsupervised approaches have found success in both improving generalization error and optimization time. Here, layers from unsupervised learners, such as autoencoders or

GANs, might be used (Salimans et al 2017). An early technique was to use TICA (Hyv arinen et al 2001), introduced by Le et al 2010.

2.3 Layer-sequential Unit-variance (LSUV)

(Mishkin and Matas 2015). Tested out on FFNNs with results comparable to BN, greedy pre-training, and other initialization schemes, LSUV offers a cheap initialization procedure which effectively does a single round of BN at initialization time. There's not a lot of background on LSUV use, nor its extensions to RNNs.