Fast Dropout

K. Pirov

April 15, 2019

1 How would you summarize the main idea of this paper in one sentence?

The main idea is that if the input to each node in a neural network is a weighted sum of its inputs – and if some of those inputs are randomly being set to zero, then the total input is actually a weighted sum of Bernoulli random variables and we can approximate the sum by a Gaussian and derive the mean and variance of these Gaussians.

2 In which cases the fast dropout approximation is exact? Why is this the case?

Dropout training of regression with squared error loss can be computed exactly. From bias-variance decomposition

$$\begin{aligned} E_{\hat{Y}\sim\mathcal{N}(\mu,s^2)}\left[(\hat{Y}-y)^2\right] &= \int_{-\infty}^{\infty}(\hat{y}-y)^2\mathcal{N}\left(\hat{y}|\mu,s^2\right)d\hat{y} \\ &= (\mu-y)^2+s^2 \end{aligned}$$

. We understand that this expectation is determined by the mean and variance of $\hat{Y}$. The exact example follows from Ridge regression. For dropout training and ridge regression we have

$$\begin{aligned} w &= \left(X^TX+\lambda\,\mathrm{diag}\left(X^TX\right)\right)^{-1}X^Ty \\ w &= \left(X^TX+\lambda I\right)^{-1}X^Ty \end{aligned}$$

. So if we normalize the data, unit variances will give us ridge regression, thus meaning that there is a scale invariance of dropout regularisation.

3 Where do the time savings come from?

When they approximate the output of each neuron with a Gaussian, by locally-linearizing the nonlinearity in each neuron, and derive deterministic update rules for the multi-layer case. Also they state the speedup in part 2.3, as the partial derivatives as well as $E_{S\sim\mathcal{N}(\mu s,\sigma_S^2)}[f(S)]$ only need to be computed once per training case, as they independent of i, α,β,γ can be computed by drawing K samples from S, taking time $O(K)$, whereas K samples of $Y(Z)$ take $O(mK)$ time.

4 Can fast dropout be used to optimize w.r.t. dropout rates p in a meaningful way? If so, how? If not, why?