Backpropagation for Continual Learning

Sanjit Neelam [1], Bryan Xu [2]

[1]sneelam@stanford.edu [2]bryanaxu@stanford.edu

Motivation

- Continual learning involves non-stationary input distribution and/or target function; traditional methods fail in continual learning due to loss of plasticity.
- The backpropagation algorithm used to train deep neural networks is unable to perform well on continual learning problem, as the benefits of random initialization step are lost after learning multiple tasks.
- The Continual Backprop (CBP) algorithm, which can learn continually is proposed by [1] as a modified version of backprop. We propose and evaluate extensions of CBP which are intended to address its limitations.

Background

- The pipeline for CBP is split into two parts: (1) a generator finds new features by randomly sampling from an initial weight distribution, with new features protected by some maturity threshold m, and (2) a tester sorts the features at a time-step t by utility and removes a fraction of them at replacement rate ρ. For each feature i at layer ℓ and time t, we compute its *mean-corrected contribution utility* $z_{\ell,i,t}$. Define

$$f_{\ell,i,t} = (1-\eta)\cdot h_{\ell,i,t} + \eta \cdot f_{\ell,i,t-1}, \tag{1}$$

$$\hat{f}_{\ell,i,t} = \frac{f_{\ell,i,t-1}}{1-\eta^{a_{\ell,i,t}}}, \tag{2}$$

$$z_{\ell,i,t} = (1-\eta)\cdot |h_{\ell,i,t} - \hat{f}_{\ell,i,t}| \cdot \sum_{k=1}^{n_{\ell+1}} |w_{\ell,i,k,t}| + \eta \cdot z_{\ell,i,t-1}. \tag{3}$$

Here, $h_{\ell,i,t}$ is the features' output, $w_{\ell,i,k,t}$ is the weight connected feature k in layer ℓ to feature i in layer $\ell+1$, $n_{\ell+1}$ is the number of features in layer $\ell+1$, $a_{\ell,i,t}$ is the age of a feature at time t, and η is a decay hyperparameter. Here, $f_{\ell,i,t}$ represents a running average of $h_{\ell,i,t}$ so as to keep track of its utility over the course of learning, and $\hat{f}_{\ell,i,t}$ is a bias-correction term that scales with the age of the feature.

- Finally, we multiply this by the *adaptation utility*, the inverse of the average magnitude of the features' input weights, to get the overall utility $\hat{u}_{\ell,i,t}$ of a feature. Thus, we get the system

$$u_{\ell,i,t} = \frac{|h_{\ell,i,t} - \hat{f}_{\ell,i,t}| \cdot \sum_{k=1}^{n_{\ell+1}} |w_{\ell,i,k,t}|}{\sum_{j=1}^{n_{\ell-1}} |w_{\ell-1,j,i,t}|}, \tag{4}$$

$$u_{\ell,i,t} = (1-\eta)\cdot u_{\ell,i,t} + \eta \cdot u_{\ell,i,t-1}, \tag{5}$$

$$\hat{u}_{\ell,i,t} = \frac{u_{\ell,i,t-1}}{1-\eta^{a_{\ell,i,t}}}. \tag{6}$$

Goal and Methods

The CBP algorithm aims to identify and reset the least important features in a layer based on their contribution to the next layer in order to improve continual learning. The authors acknowledge the need for a global measure of importance and a more principled tester, and so we propose modifications to address these limitations.

(1) Natural Importance

- Intuitively, a feature has high utility if predictions are significantly worse when it is absent.
- Define the loss as $\mathcal{L}(\Theta_t)$, where Θ_t represents the parameters of the network. To compute the loss when feature i is absent, define a function g which "masks" the contribution of feature i. The loss with feature i removed can be computed as $\mathcal{L}(g(\Theta, i))$. We can then define the relative importance of feature i in terms of the increase in the loss when feature i is removed:

$$J_i(\Theta_t) = \mathcal{L}(g(\Theta_t, i)) - \mathcal{L}(\Theta_t). \tag{7}$$

To compute $\mathcal{L}(g(\Theta_t, i))$ in practice, we set the appropriate columns in a weight matrix θ in Θ to zero and perform a forward pass. Then we let $u_{\ell,i,t} = J_i(\Theta)/(\sum_{j=1}^{n_{\ell-1}} |w_{\ell-1,j,i,t}|)$ in Eq. 4.

- **Sparse Natural Importance:** We also investigate a naïve relaxation of the combinatorial search, in which we only compute $J_i(\Theta)$ for a feature i with some probability $p \in [0,1]$ for each training example. For the features for which we don't compute $J_i(\Theta)$, we let $u_{\ell,i,t} = 0$.
- **Taylor Natural Importance:** To estimate the value of an objective function J when some of the weights in a parameter are set to zero, we can use a Taylor approximation to speed up computation. E.g.

$$J_i(\Theta) \approx J(\Theta) + \frac{\partial J(\Theta)}{\partial W^{(2)}}(\tilde{W}^{(2)} - W^{(2)})^T, \tag{8}$$

where $\tilde{W}^{(2)}$ is a copy of $W^{(2)}$ with the first column set to zero.

(2) Step Sizes

- An alternative global measure of utility can be obtained by considering the size of the update $\theta_{t+1} - \theta_t$ for each parameter θ in the network.
- A parameter θ_t for which the step $\theta_{t+1} - \theta_t$ over the last few training examples has been small may be a 'non-plastic' parameter, or one that no longer learns, making it a likely candidate for resetting.
- For a parameter θ_t, we let

$$u_{\ell,i,t} = \frac{||(\theta_{t+1} - \theta_t)_i||_2^2}{\sum_{j=1}^{n_{\ell-1}} |w_{\ell-1,j,i,t}|}$$

in Equation 4, where $(\theta_{t+1} - \theta_t)_i$ is the i-th column of $\theta_{t+1} - \theta_t$.

Experiments and Results

- We test our algorithms on the *slowly changing regression* problem: the input at a time step t is represented by $\mathbf{x}_t \in \{0,1\}^m$, where $x_{i,t} \in \{0,1\}$ for each $i \in 1, \ldots, m$. After every T time-steps one of the first f bits is flipped at random. Each of the next $m-f$ bits is randomly sampled from $U[0,1]$ at every time-step. We choose $m = 20$, $f = 15$, and $T = 1e4$, and test against backprop.

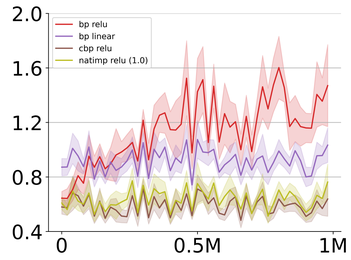

Figure 1. Natural Importance vs. Baseline

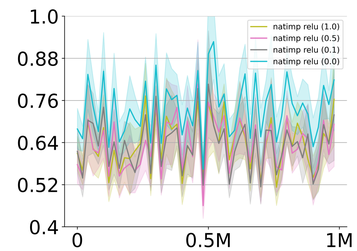

Figure 2. Sparse Nat. Importance Variation

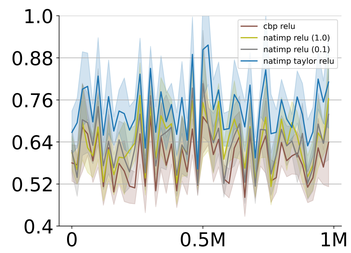

Figure 3. Taylor Nat. Importance vs. CBP

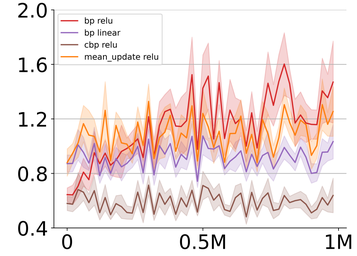

Figure 4. Step Size vs. Baseline

Next Steps

- The next step would be to test our algorithms in more complex settings such as the permuted MNIST problem, which would require a deeper network, or completely non-stationary problems such as the Slippery Ant problem. We would also like to test other activation functions and optimizers (i.e. ADAM).

References

[1] Shibhansh Dohare, Richard S. Sutton, and A. Rupam Mahmood. Continual backprop: Stochastic gradient descent with persistent randomness. 2022.