Malicious Dropout

Jack Maris and Iskandar Pashayev

Can intermittently dropping prominent features improve generalization?

Problem / Background

What is a zebra? We recognize that it has stripes, hooves, and a face with a certain curvature.

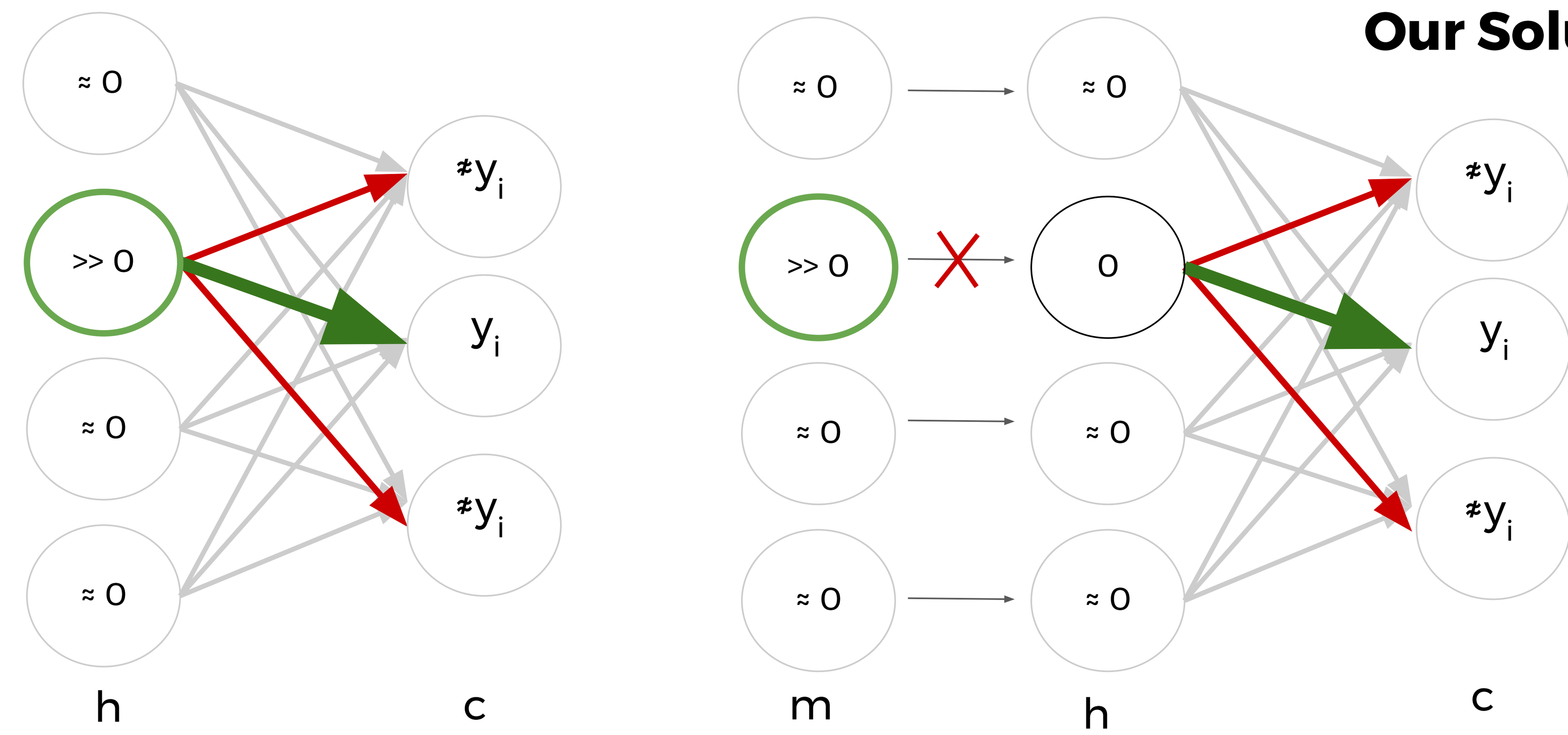
A network which only learns that it has stripes may get great Loss on training data. But it would generalize poorly: it might classify a fence as a zebra. We want the network to learn what we learn.

Dropout helps by occasionally dropping, *e.g.* stripe-ness [1]. This forces the network to learn other features. L2 normalization can prevent certain features from dominating. And Dropout modifications in the literature include Nested Dropout [2], which aims at dimensionality reduction.

But Dropout can be cumbersome. What if we were to *target* the most useful features on some trials?

We propose a method for dropping the features which cause the loss to increase the most. We then report results of various mechanisms for interleaving this method.

Our Solution



Each neuron h_j is given a score equal to the its contribution to c_{yi} (that is, $h_j w_{j,yi}$) minus ω times the sum of its contributions to $c_{\neq yi}$. The top ρ neurons are dropped. Then h is recalculated.

Models

We run a simple model as a proof-of-concept, and also append Maldrop to SqueezeNet.

- Conv (with ReLU activation function) to Spatial BatchNorm to Conv (with ReLU activation function) to MalDrop to Affine.
- ImageNet and CIFAR.
- PyTorch.
- Code: git.io/maldrop.

ρ Schedules

We try 5 different ρ schedules.
Static: ρ is the same at each iteration.
Random: ρ is sampled from ~Uniform(0,*n*) at each iteration, where *n* is the number of neurons the layer.
Forward Annealing: Steadily increase some initial ρ_0 with each epoch $i \geq 0$, with a maximum of L number of neurons.
Reverse Annealing: .Steadily decrease some initial ρ_0 with each epoch $i \geq 0$, with a minimum of 0 neurons.
Periodic: say t is the trial number, and pick a parameter s. When t % s = 0 use Maldrop; otherwise, use vanilla Dropout.

Forward Annealing

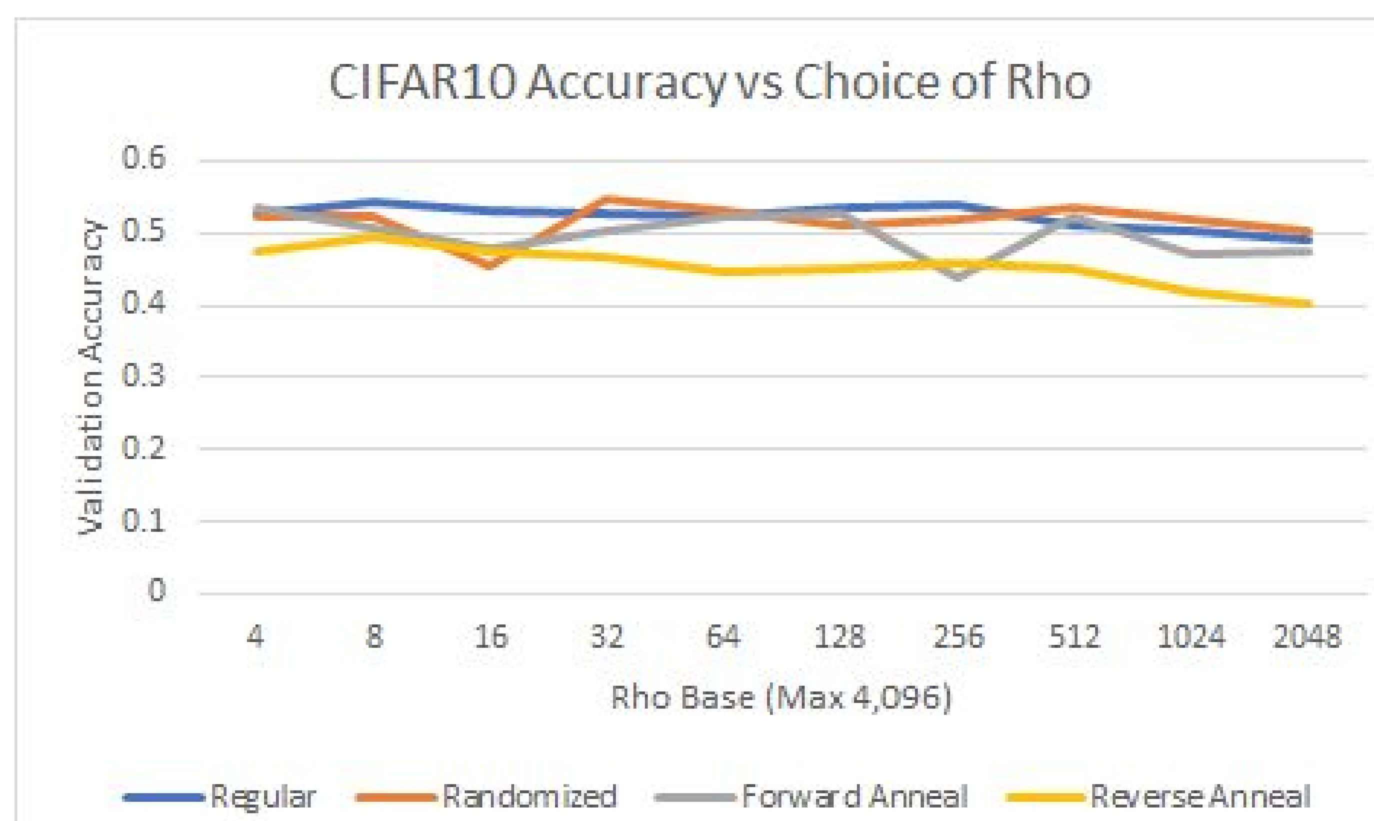
$$\rho_i = \frac{L}{1 + e^{-(\rho_0 + i)}}$$

Reverse Annealing

$$\rho_i = L \cdot \left(1 - \frac{1}{1 + e^{-(\rho_0 + i)}}\right)$$

Findings

- Most effective placement: immediately before a network's final affine layer.
  - More efficient (fewer backprop steps to compute twice).
  - Otherwise, combinatoric possibilities: d-choose-ρ !
  - Exception: networks which propagate loss from multiple places, *e.g.* GoogLeNet [4].
- It is useful to vary the method used between trials. Either change ρ, intersperse vanilla Dropout trials, or intersperse no-Dropout trials.
- We expect certain tasks to be better suited to Malicious Dropout than others. We are informed by how vanilla Dropout fares in certain domains (*e.g.* relatively poorly in recurrent networks [3]).
- Network with two CONV layers and a MalDrop layer on CIFAR10 data: performance similar to network with regular Dropout layer, with exception of reverse annealing.
- Different modes of rho yield similar results.



Future Work

- Improve efficiency.
- Time-Space tradeoff of MalDrop computation.
- Numerically stable reverse annealing.

Citations

[1] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Re- search, 15(1):1929–1958, 2014.

[2] O. Rippel, M. Gelbart, and R. Adams. Learning ordered representations with nested dropout. In International Conference on Machine Learning, pages 1746–1754, 2014.

[3] C.Szegedy, W.Liu, Y.Jia, P.Sermanet, S.Reed, D.Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1-9, 2015.

[4] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. CoRR, abs/1409.2329, 2014.