Iterative MILP-based Learning Rule for Neural Networks

Colin McDonnell, Parker Zhao, Dan Sosa

April 22, 2016

1 Introduction

We are building off the work done by a final project done last year by David Zheng and others. We will use the same optimization model they used, but wrap it in an iterative update rule.

Without an iterative update rule, the model has to be trained on all data simultaneously; that is, each pixel of each image is a variable in the model. This scales very poorly when trying to do image classification on real images. As a result, Zheng et al reduced the resolution of their training images to 5x5, which clearly throws away a huge amount of the information content required to classify.

2 Approach

2.1 The general idea

We developed an iterative approach that doesn't require training on all data at once. We maintain an exogenous weight vector that gets updated based on optimal weight values computed from a stochastically sampled subset of the training data. Then, after many iterations of these training rule, we "inject" the learned exogenous weights into the model with equality constraints and evaluate performance of the newtrk on the test data set to determine whether the system has learned a generalizable classifier. To achieve meaningful results we updated the model significantly and introduced many hyperparameters. We will discuss these here.

2.2 The model

The new model is shown below.

$$min \sum_{i_1} o_{i_1} - \sum_{i_2} o_{i_2} + wdp * \sum dw + wdp * \sum dv$$

$$s.t.$$

$$\sum_{j=1}^{P} w_{jk}x_{ij} \geq \epsilon - M(1-h_{ik}) \qquad \forall i \in 1,...,N, \forall k \in 1,...,H$$

$$\sum_{j=1}^{P} w_{jk}x_{ij} \leq -\epsilon + Mh_{ik} \qquad \forall i \in 1,...,N, \forall k \in 1,...,H$$

$$\sum_{k=1}^{H} v_k h_{ik} \geq \epsilon - M(1-o_i) \qquad \forall i \in 1,...,N$$

$$\sum_{k=1}^{H} v_k h_{ik} \leq -\epsilon + Mo_i \qquad \forall i \in 1,...,N$$

$$-dw_{jk} \leq w_{jk} - tw_{jk} \leq dw_{jk} \qquad \forall i \in 1,...,N, \forall k \in 1,...,H$$

$$-dv_k \leq v_k - tv_k \leq dv_{jk} \qquad \forall i \in 1,...,N, \forall k \in 1,...,H$$

where
N is the number of images
P is the number of pixels in each image
H is the number of hidden nodes in the neural network
x_{ij} is a constant representing the pixel value of the jth pixel in the ith image
o_i is the binary output of the neural network for the ith image
w_{jk} is the weight between the jth pixel of the image and the kth node of the hidden layer
v_k is the weight between the kth node of the hidden layer and the output node
tw_{jk} is the value for weight w_{jk} stored in the exogenous ("true") weight vector
tv_k is the value for weight v_k stored in the exogenous ("true") weight vector
h_{ik} is the binary output of the kth node of the hidden layer for the ith image
dw_{jk} is the absolute value of $w_{jk} - tw_{jk}$
dv_k is the absolute value of $v_k - tv_k$
c is the bias of the output node
ϵ and M are a small and large positive constant respectively
wdp is the $weightdivergencepenalty$ which biases the neural net towards finding weights similar to those in the exogenous weight vector

There are two additional terms in the objective function that penalize weights that are different from those in the exogenous weight vector. We accomplish this by using a set of constraints that functionally implements the absolute value of $w - tw$ and $v - tv$. Here, tw and tv are so named because they are the "true-w" and "true-v" values, and the actual w and v values are overwritten on each step of the training algorithm.

This new objective function is required because the optimal weights for classifying one subset of the training set are typically very different from those that optimally classify another subset. MILP is very good at overfitting to the data because it explores a massive space of variables, typically identifying features that don't generalize to data points not in the training set. By including this constraint we introduce continuity into the training process: the MILP algorithm finds a way to improve the neural net's performance while taking into account past training.

3 Hyperparameters

The new training algorithm involves a number of hyperparameters that govern how training is done. These are described below. Thus far, we have yet to implement any hyperparameter optimization over the space of these variables, but based on our results we are confident we've found values that are reasonable.

p_{sample} is the sampling rate used by the minibatch generation process. Each image in the dataset (there are 1000 images of fours and 1000 images of sixes) has a $p_{s}ample$ probability of being includes in the minibatch on each iteration.

$numiter$ is the number of iterations of the algorithm.

$time_{l}imit$ is the time constraint on solving the MILP on each iteration. If an optimal solution isn't found within the time limit, the best solution found by that point is returned and used to update weights.

$mip_{g}ap$ is how the solver decides a found solution is "good enough". The MIP Gap is defined as the relative gap between the lower and upper objective bound. If the MIP Gap is 20 percent, then the best solution found thus far is within 20 percent of optimal.

wdp_{con} is the constant component of the wdp value. wdp_{var} is the variable component of the wdp value. The value of wdp is calculated on each iteration as $wdp_{con} + wdp_{var} * iteration$. Thus, we can have wdp values that vary over the course of the training process. This parameter functions as a "learning rate": early in the training process we let the weights jump around freely, but later on we let the weights "cool" and vary less in response to new information. This lets us explore the whole parameter space without letting the most recent training data seen by the network have a disproportional impact on its final weights.

4 Results

Zheng et al. They trained on 80 5x5 images, and achieved an accuracy of 89%. Using the exact same training data, we achieved an accuracy of 92%.

Moreover, because we no longer have to train a model on all data simultaneously, we were able to handle much larger image inputs than before. We created a new training dataset consisting of 2000 10x10 greyscale images: 1000 images of handwritten fours and 1000 images of handwritten sixes. This was derived from the online MNIST dataset maintained by Yann Lecun. On the larger images, we achieved 77% accuracy on the larger images.

5 Future work

In the future we want to scale the network up even further to handle full-scale MNIST images of resolution 28x28. We also hope to implement dropout, a technique for preventing overfitting, in an attempt to make a more generalizable classifier. Finally, we hope to generalize the work described here to perform classification over all 10 digits, as opposed to the two (fours and sixes) used here.