Neuronal Death in Neural Networks with Group Sparsity Regularization

Nicholas Dwork

Stanford ENGINEERING
Electrical Engineering

Overview

- The structure of a neural network is determined by hand.
- Choosing the number of neurons in each layer is a shot in the dark.
- Choosing too many neurons leads to over-fitting. Choosing too few neurons cannot fit the data well.
- The purpose of this project is to develop an algorithm that automatically determines the number of neurons in each layer of a neural network.

The Optimization Problem

Neural networks are often trained by solving a model of the form:

$$\underset{x}{\text{minimize}} \underbrace{\sum_i f(x, d_i)}_{J(x)} \tag{1}$$

- x is a vector comprised of the parameters (weights and biases) of the neural network
- d_i is an element of the training set

When J is sub-differentiable, problem (1) can be solved with sub-gradient descent.

Let x_g be the parameters for the g^{th} neuron. The vector x is the concatenation of all x_g. Problem (1) can be modified to include L_2, L_1 regularization as follows:

$$\underset{x}{\text{minimize}}\; J(x) + \gamma \underbrace{\sum_g \|x_g\|_2}_{R(x)}, \tag{2}$$

where γ is a regularization parameter, $\|\cdot\|_2$ represents the L_2 norm, and R is the regularization function. This amounts to calculating the L_1 norm of a vector of L_2 norms.

When J is differentiable and R has a simple proximal operator, problems of this form can be solved with the Stochastic Proximal Gradient algorithm [1].

For $k = 1 \dots K_{\text{max}}$,

$$y^{(k)} = x^{(k)} - t_k \hat{\nabla} f\left(x^{(k)}\right)$$
$$z^{(k)} = \text{prox}_{t_k g}\left(y^{(k)}\right).$$

The prox function is the proximal operator, defined as

$$\text{prox}_{tR}(y) = \text{argmin}_x R(x) + \frac{1}{2t}\|x - y\|_2^2.$$

The proximal operator for the L_2, L_1 norm is $\text{prox}_{L_2,L_1}(x) = \sum_g \text{prox}_{L_2}(x_g)$, and the proximal operator for the L_2 norm is

$$\underset{t\|\cdot\|_2}{\mathbf{prox}}(x) = \begin{cases} (1 - t/\|x\|_2)\, x & \text{if } \|x\|_2 \geq t \\ 0 & \text{otherwise} \end{cases}.$$

This amounts shrinking the x vector by t (and setting it to 0 if less than size t).

Regularization Growing

- When J is convex, the stochastic sub-gradient algorithm is guaranteed to reach the global minima (under mild assumptions). However, when J is non-convex, the algorithm is only guaranteed to reach a local minimum.
- On the test problem (described later), I found that I reached a local minimum for regularization parameters of interest (e.g. $10^2, 10^3, \cdots, 10^8$). These parameter values (those of the local minima) yielded insignificant accuracies on the test and training sets (e.g. $\approx 15\%$ accuracy).

To generate a result that yielded good accuracy with high accuracy, I created a regularization growing algorithm:

Initialize net to random values
net = Solve problem (1)
For $p = 0, 1, \dots, P_{\text{max}}$,
  $\gamma = 10^p$
  $t = 2 \cdot 10^{-(p+2)}$
  net = Solve problem (2)

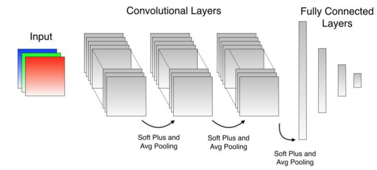
The network is "warm started" with the result of the previous iteration (which used a smaller regularization parameter).

Test Problem: CIFAR-10

- Consists of 32×32 color images of 10 categories: airplane, car, bird, cat, deer, dog, frog, horse, ship, and truck
- The dataset consists of $60,000$ images ($50,000$ in the training set and $10,000$ in the test set)
- Goal: properly classify each image

Test Network

- The network consists of 3 convolutional layers followed by 3 fully connected layers (as shown below)
- The convolutional layers start with 300, 200, and 100 neurons, respectively
- The fully connected layers have 500, 200, and 10 neurons respectively
- A `softmax` is applied to the output of the final layer



Results

- After training the network with Stochastic-Subgradient Descent, 99.9% accuracy was achieved on the test data and 73% accuracy on the test set. This suggests significant over-fitting.
- After training with regularization growing, the number of active neurons is reduced.
- Future work: once the sparsity pattern is determined, the network can be polished (train with the appropriate network without any regularization)

Acknowledgements

I would like to thank Surag Nair and Daniel O'Connor for their guidance and advice throughout this project. I would also like to thank Amazon Web Services for providing the computing resources that made this project possible.

References

[1] Lorenzo Rosasco, Silvia Villa, and Bang Công Vũ. Convergence of stochastic proximal gradient algorithm. *arXiv preprint arXiv:1403.5074*, 2014.