Noname manuscript No.
(will be inserted by the editor)

Entropy loss function for unsupervised deep learning

Dylan M. Paiton
Sheng Y. Lundquist

Abstract Here we describe a loss function for training a typical deep network in an unsupervised way. The objective of the loss is to encourage the deep network to minimize the entropy of its output while still conveying information. This should result in an intelligent clustering of object categories without the need of human generated object labels.

Keywords entropy · deep network · loss function · unsupervised learning

1. Introduction

Here we describe a loss function for the unsupervised training of a deep neural network. This is motivated by the cost of labeling data and we want to learn a more general representation.

The goal of the project is to develop a cost function that applies to any generic neural network output. In this document we will first define our loss function in terms of a cost that must be propagated through the deep network to modify parameters. Then we will derive the gradient of the cost function with respect to an individual output element from the deep network. This gradient can then be applied to the network parameters via the back propagation algorithm. We assume that the deep network is being trained via stochastic gradient descent with a batch of images. For each image, we expect an output vector of floats (they could be positive only, but it is not required). We assume one vector per image in the batch, which can then be assembled into a matrix.

DM Paiton
Vision Science Graduate Group
Redwood Center for Theoretical Neuroscience
University of California, Berkeley
E-mail: dpaiton@berkeley.edu

2. Forward Function - Entropy Computation

The input to the entropy loss function is X, a matrix of N neurons by B batch inputs. We note that the indices i, j, k will be used to index the node in X and l, m, n will be used to index the batch number in X. We want the network output distribution for a given image to be a delta function (i.e. one-hot distribution), indicating that the input falls into a particular category. Additionally, we want the average network output distribution over all images to be uniform. These objectives can be expressed in terms of entropy. First we define a probability distribution across the N nodes in X:

$$Q_{i,l} = \frac{e^{-\beta X_{i,l}}}{\sum_j^N e^{-\beta X_{j,l}}}, \tag{1}$$

where Q is computed per neuron for a single input, $i \in N$ indexes the neurons, and $l \in B$ indexes the batch input. For $\beta = -1$, equation 1 is the softmax function on X. We include the β term to allow us to modify the temperature of our probability distribution. Next we define a probability distribution across our batch of B inputs:

$$P_{i,l} = \frac{e^{-\beta X_{i,l}}}{\sum_m^B e^{-\beta X_{i,m}}}. \tag{2}$$

The entropy for a given batch image (summed over nodes) and for a given node (summed over the batch) are

$$H_l = -\sum_i Q_{i,l} \ln Q_{i,l} \tag{3}$$

and

$$H_i = -\sum_l P_{i,l} \ln P_{i,l}, \tag{4}$$

respectively. We wish for all nodes to be active over the corpus of our inputs and for only one node to be active for a single input. Thus, we want to minimize **positive** entropy per image and minimize **negative** entropy per batch. This should encourage the network to cluster inputs into discrete categories and produce a sparse output. Therefore, our cost function to minimize will combine equation 3 with the negative of equation 4:

$\mathrm{C} = \sum_l H_l - \lambda \sum_i H_i$

$\mathrm{C} = \text{-} \sum_l \sum_i Q_{i,l} \ln Q_{i,l} + \lambda \sum_i \sum_l P_{i,l} \ln P_{i,l}$

$$C = \sum_{i,l} \frac{e^{-\beta X_{i,l}}}{\sum_j e^{-\beta X_{j,l}}} \left(\beta X_{i,l} + \ln \sum_j e^{-\beta X_{j,l}} \right) - \lambda \sum_{i,l} \frac{e^{-\beta X_{i,l}}}{\sum_m e^{-\beta X_{i,m}}} \left(\beta X_{i,l} + \ln \sum_m e^{-\beta X_{i,m}} \right). \tag{5}$$

It should be clear here that the two entropy calculations are nearly identical, and therefore the derivatives are going to be similar. We will take the derivative of one entropy function and apply it to both terms in our cost function.

$$H_l = -\sum_i Q_{i,l} \ln Q_{i,l}$$

$$H_l = \sum_i \frac{\beta X_{i,l} e^{-\beta X_{i,l}}}{\sum_j e^{-\beta X_{j,l}}} + \sum_i \frac{e^{-\beta X_{i,l}}}{\sum_j e^{-\beta X_{j,l}}} \ln \sum_j e^{-\beta X_{j,l}}$$

$$H_l = \overbrace{\sum_i \frac{\beta X_{i,l} e^{-\beta X_{i,l}}}{\sum_j e^{-\beta X_{j,l}}}}^{\text{Left Term}} + \overbrace{\frac{\ln \sum_j e^{-\beta X_{j,l}}}{\sum_j e^{-\beta X_{j,l}}} \sum_i e^{-\beta X_{i,l}}}^{\text{Right Term}}. \tag{6}$$

3. Backward Function - Entropy Gradient

The gradient of the forward function will contain the partial derivative of equation 5 with respect to an individual element, $X_{k,l}$. For the sake of brevity, we will only derive a single entorpy term, given in equation 6. First we will take the derivative of the left term, as defined in equation 6, then we will take the derivative of the right term, then we will combine them.

3.1 Useful Derivatives

The following are a handful of derivatives that have to be taken multiple times for the entropy gradient calculation.

$$\frac{\partial \sum_i X_i}{\partial X_k} = \frac{\partial X_k}{\partial X_k} + \frac{\partial \sum_{i \neq k} X_i}{\partial X_k} = 1 \tag{7}$$

$$\frac{\partial \sum_i e^{-\beta X_i}}{\partial X_k} = \frac{\partial e^{-\beta X_k}}{\partial X_K} + \frac{\partial \sum_{i \neq k} e^{-\beta X_i}}{\partial X_k} = -\beta e^{-\beta X_k} \tag{8}$$

$$\begin{aligned}\frac{\partial \sum_i \beta X_i e^{-\beta X_i}}{\partial X_k} &= \frac{\partial \beta X_k e^{-\beta X_k}}{\partial X_k} + \frac{\partial \sum_{i \neq k} \beta X_i e^{-\beta X_i}}{\partial X_k} \\ &= \frac{\partial \beta X_K}{\partial X_k} e^{-\beta X_k} + \beta X_k \frac{\partial e^{-\beta X_k}}{\partial X_k} \\ &= \beta e^{-\beta X_k} + \beta X_k (-\beta e^{-\beta X_k}) \\ &= \beta e^{-\beta X_k} (1 - \beta X_k)\end{aligned} \tag{9}$$

$$\frac{\partial \ln \sum_j e^{-\beta X_j}}{\partial X_k} = \frac{1}{\sum_j e^{-\beta X_j}} \frac{\partial \sum_j e^{-\beta X_j}}{\partial X_k} = \frac{-\beta e^{-\beta X_k}}{\sum_j e^{-\beta X_j}} \tag{10}$$

$$\frac{\partial Q_i}{\partial X_i} = \frac{-\sum_j e^{-\beta X_j} \beta e^{-\beta X_i} + \beta (e^{-\beta X_i})^2}{(\sum_j e^{-\beta X_j})^2} = \beta Q_i \left(Q_i - 1\right) \tag{11}$$

3.2 Left Term Entropy Derivative

$$\frac{\partial H_l}{\partial X_{k,n}} = \frac{\sum_j e^{-\beta X_{j,n}} \frac{\partial \sum_i \beta X_{i,n} e^{-\beta X_{i,n}}}{\partial X_{k,n}} - \sum_i \beta X_{i,n} e^{-\beta X_{i,n}} \frac{\partial \sum_j e^{-\beta X_{j,n}}}{\partial X_{k,n}}}{\left(\sum_j e^{-\beta X_{j,n}}\right)^2} + \dots$$

$$\frac{\partial H_l}{\partial X_{k,n}} = \frac{\sum_j e^{-\beta X_{j,n}} \beta e^{-\beta X_{k,n}} \left(1 - \beta X_{k,n}\right) + \beta^2 e^{-\beta X_{k,n}} \sum_i X_{i,n} e^{-\beta X_{i,n}}}{\left(\sum_j e^{-\beta X_{j,n}}\right)^2} + \dots$$

$$\frac{\partial H_l}{\partial X_{k,n}} = \frac{\beta e^{-\beta X_{k,n}} \left(1 - \beta X_{k,n}\right)}{\sum_j e^{-\beta X_{j,n}}} + \frac{\beta^2 e^{-\beta X_{k,n}} \sum_i X_{i,n} e^{-\beta X_{i,n}}}{\left(\sum_j e^{-\beta X_{j,n}}\right)^2} + \dots$$

$$\frac{\partial H_l}{\partial X_{k,n}} = \frac{\beta e^{-\beta X_{k,n}}}{\sum_j e^{-\beta X_{j,n}}} \left(1 - \beta X_{k,n} + \frac{\beta \sum_i X_{i,n} e^{-\beta X_{i,n}}}{\sum_j e^{-\beta X_{j,n}}}\right) + \dots$$

$$\frac{\partial H_l}{\partial X_{k,n}} = \beta Q_{k,n} \left(1 - \beta X_{k,n} + \beta \sum_i X_{i,n} Q_{i,n}\right) + \dots$$

$$\frac{\partial H_l}{\partial X_{k,n}} = \beta Q_{k,n} - \beta^2 X_{k,n} Q_{k,n} + \beta^2 Q_{k,n} \sum_i X_{i,n} Q_{i,n} + \dots$$

3.3 Right Term Entropy Derivative

$$\frac{\partial H_l}{\partial X_{k,n}} = \ldots + \sum_i e^{-\beta x_{i,n}} \left(\frac{\sum_j e^{-\beta X_{j,n}} \left(\frac{-\beta e^{-\beta X_{k,n}}}{\sum_j e^{-\beta X_{j,n}}} \right) + \beta e^{-\beta X_{k,n}} \ln \sum_j e^{-\beta X_{j,n}}}{\left(\sum_j e^{-\beta X_{j,n}} \right)^2} \right) + \frac{\ln \sum_j e^{-\beta X_{j,n}}}{\sum_j e^{-\beta X_{j,n}}} \left(-\beta e^{-\beta X_{k,n}} \right)$$

$$\frac{\partial H_l}{\partial X_{k,n}} = \ldots + \frac{-\beta e^{-\beta X_{k,n}}}{\sum_j e^{-\beta X_{j,n}}} + \frac{\beta e^{-\beta X_{k,n}} \ln \sum_j e^{-\beta X_{j,n}}}{\sum_j e^{-\beta X_{j,n}}} - \frac{\beta e^{-\beta X_{k,n}} \ln \sum_j e^{-\beta X_{j,n}}}{\sum_j e^{-\beta X_{j,n}}}$$

$$\frac{\partial H_l}{\partial X_{k,n}} = \ldots - \beta Q_{k,n} \tag{12}$$

3.4 Combine Left and Right Entropy Derivatives

$$\frac{\partial H_l}{\partial X_{k,n}} = \beta Q_{k,n} - \beta^2 X_{k,n} Q_{k,n} + \beta^2 Q_{k,n} \sum_i X_{i,n} Q_{i,n} - \beta Q_{k,n}$$

$$\frac{\partial H_l}{\partial X_{k,n}} = -\beta^2 X_{k,n} Q_{k,n} + \beta^2 Q_{k,n} \sum_i X_{i,n} Q_{i,n}$$

$$\frac{\partial H_l}{\partial X_{k,n}} = \beta^2 Q_{k,n} \left(\sum_i X_{i,n} Q_{i,n} - X_{k,n} \right), \tag{13}$$

where $Q_{k,n}$ is given in equation 1.

3.5 Cost Derivative

Equation 5 has two entropy terms in it, the gradient of which can be derived by following the same process outlined above.

$$\frac{\partial C}{\partial X_{k,n}} = \beta^2 Q_{k,n} \left(\sum_i X_{i,n} Q_{i,n} - X_{k,n} \right) - \lambda \beta^2 P_{k,n} \left(\sum_l X_{k,l} P_{k,l} - X_{k,n} \right)$$

$$\frac{\partial C}{\partial X_{k,n}} = \beta^2 \left(Q_{k,n} \left(\sum_i X_{i,n} Q_{i,n} - X_{k,n} \right) - \lambda P_{k,n} \left(\sum_l X_{k,l} P_{k,l} - X_{k,n} \right) \right) \tag{14}$$

3.6 Semi-Supervised LCA

The loss function described in equation can be combined with the LCA sparse solver. First we add the supervised objective to the traditional Sparse Coding cost function.

$$\underset{a}{\operatorname{argmin}} \left(E = \overbrace{\tfrac{1}{2}\|S - R\|_2^2}^{\text{Preserve Information}} + \overbrace{\lambda \sum_m C(a_m)}^{\text{Limit Activations}} \right) \tag{15}$$

$$\underset{a}{\operatorname{argmin}} \left(E = \overbrace{\tfrac{1}{2}\|I - R\|_2^2}^{\text{Preserve Information}} + \overbrace{\lambda \sum_m C(a_m)}^{\text{Limit Activations}} + \overbrace{\gamma \mathscr{L}\left(W \frac{a_m}{||a_m||_2}, y\right)}^{\text{Categorical Cost}} \right) \tag{16}$$

Taking the derivative gives us a new membrane update rule for LCA inference:

$$\begin{aligned} \dot{u_i}(t) &\propto -\frac{\partial E}{\partial a_i(t)} \\ &= \frac{1}{\tau}\left[b_i - u_i(t) - \sum_{m \neq i}^{M} G_{i,m} a_m(t) - \sum_{m}^{M} W_m \delta\left(\hat{y}\right) \right], \end{aligned} \tag{17}$$

where $\hat{y}$ represents the softmax output of the classification layer. Finally, we want to construct a new objective function for a hierarchical sparse coding model:

$$\underset{a}{\operatorname{argmin}} \left(E = \tfrac{1}{2}\|I - \phi a^0\|_2^2 + \sum_i \frac{a_i^l}{\sigma_i} + \log\left(\sigma_i\right) + \lambda \sum_i |a_i^{l+1}| \right)$$

where

$$\log\left(\sigma\right) = \sum_j B_{ij} a_j^{l+1}$$