RECURSIVE FORMULATION OF LOSS GRADIENT IN A DENSE FEED-FORWARD DEEP NEURAL NETWORK

ASHWIN RAO

1. Motivation

The purpose of this short paper is to derive the recursive formulation of the gradient of the loss function with respect to the parameters of a dense feed-forward deep neural network. We start by setting up the notation and then we state the recursive formulation of the loss gradient. **We derive this formulation for a fairly general case - where the supervisory variable has a conditional probability density modeled as an arbitrary Generalized Linear Model(GLM)'s "normal-form" probability density, when the output layer's activation function is the GLM canonical link function, and when the loss function is cross-entropy loss**. Finally, we show that Linear Regression and Softmax Classification are special cases of this generic GLM structure, and so this formulation of the gradient is applicable to the special cases of dense feed-forward deep neural networks whose output layer activation function is linear (for regression) or softmax (for classification).

2. Notation

Consider a "vanilla" (i.e., dense feed-forward) deep neural network with L layers. Layers $l = 0, 1, \ldots, L-1$ carry the hidden layer neurons and layer $l = L$ carries the output layer neurons.

Let I_l be the inputs to layer l and let O_l be the outputs of layer l for all $l = 0, 1, \ldots, L$. We know that $I_{l+1} = O_l$ for all $l = 0, 1, \ldots, L-1$. Note that the number of neurons in layer l will be the number of outputs of layer l $(= |O_l|$. In the following exposition, we assume that the input to this network is a single training data point (input features) denoted as X and the output of the network is supervised by the supervisory value Y associated with X (we work with a single training data point in this entire paper because the gradient calculation can be independently applied to each training data point and finally summed over the training data points). So, $I_0 = X$ and O_L is the network's prediction for input X (that will be supervised by Y).

We denote K as the number of elements in the supervisory variable Y (K will also be the number of neurons in the output layer as well as the number of outputs produced by the output layer). So, $|O_L| = |Y| = K$. We denote the r^{th} element of O_L (output of r^{th} neuron in the output layer) as $O_L^{(r)}$ and the r^{th} element of Y as $Y^{(r)}$ for all $r = 1, \ldots K$.

Note that $K = |O_L| = |Y| = 1$ for regression. For classification, K is the number of classes with $K = |O_L|$ neurons in the output layer. The supervisory variable Y will be have a "one-hot" encoding of length K as follows: . If the supervisory variable is meant to represent class $k \in [1, K]$, then $Y^{(k)} = 1$ and $Y^{(r)} = 0$ for all $r = 1, \ldots, k-1, k+1, \ldots K$ (the non-k elements of Y are 0).

The error $E = O_L - Y$ is a scalar for regression. For classification, $E = O_L - Y$ is a vector of length K whose k^{th} element is $O_L^{(k)} - 1$ and the other (non-k) elements of E are $O_L^{(r)}$ for all $r = 1, \ldots, k-1, k+1, \ldots K$.

We denote the parameters ("weights") for layer l as W_l (W_l is a matrix of size $|O_l| \times |I_l|$). For ease of exposition, we will ignore the bias parameters (as we can always treat X as augmented by a "fake" feature that is always 1). We denote the activation function of layer l as $g_l(\cdot)$ for all $l = 0, 1, \ldots, L-1$. Let $S_l = I_l \cdot W_l$ for all $l = 0, 1, \ldots, L$, so $O_l = g_l(S_l)$

Notation	Description
I_l	Inputs to layer l for all $l = 0, 1, \ldots, L$
O_l	Outputs of layer l for all $l = 0, 1, \ldots, L$
X	Input Features to the network for a single training data point
Y	Supervisory value associated with input X
W_l	Parameters ("weights") for layer l for all $l = 0, 1, \ldots, L$
E	Error $O_L - Y$ for the single training data point (X, Y)
$g_l(\cdot)$	Activation function for layer l for $l = 0, 1, \ldots, L-1$
S_l	$S_l = I_l \cdot W_l$, $O_l = g_l(S_l)$ for all $l = 0, 1, \ldots L$
P_l	"Proxy Error" for layer l for all $l = 0, 1, \ldots, L$
λ_l	Regularization coefficient for layer l for all $l = 0, 1, \ldots, L$

3. Recursive Formulation

We define "proxy error" P_l for layer l recursively as follows for all $l = 0, 1, \ldots, L$:

$$P_l = (P_{l+1} \cdot W_{l+1}) \circ g_l'(S_l)$$

with the recursion terminating as:

$$P_L = E = O_L - Y$$

Then, the gradient of the loss function with respect to the parameters of layer l for all $l = 0, 1, \ldots, L$ will be:

$$\frac{\partial Loss}{\partial W_l} = P_l \otimes I_l$$

Section 4 provides a proof of the above when the supervisory variable has a conditional probability density modeled as the Generalized Linear Model(GLM)'s "normal-form" probability density (i.e. fairly generic family of probability distributions), when the output layer's activation function is the GLM canonical link function, and when the loss function is cross-entropy loss. We show in the final two sections that Linear Regression and Softmax Classification are special cases of this generic GLM structure, and so this formulation of the gradient is applicable to the special cases of dense feed-forward deep neural networks whose output layer activation function is linear (for regression) or softmax (for classification). For now, we give an overview of the proof in the following 3 bullet points:

- "Proxy Error" P_l is defined as $\frac{\partial Loss}{\partial S_l}$, and so gradient $\frac{\partial Loss}{\partial W_l} = \frac{\partial Loss}{\partial S_l} \otimes \frac{\partial S_l}{\partial W_l} = P_l \otimes I_l$
- The important result $P_L = E = O_L - Y$ is due to the important GLM result $A'(S_L) = g_L(S_L) = O_L$ where $A(\cdot)$ is the key function in the probability density functional form for GLM.
- The recursive formulation for P_l has nothing to do with GLM, it is purely due to the structure of the feed-forward network: $S_{l+1} = g_l(S_l) \cdot W_{l+1}$. Differentiation chain rule yields $\frac{\partial Loss}{\partial S_l} = \frac{\partial Loss}{\partial S_{l+1}} \cdot \frac{\partial S_{l+1}}{\partial S_l}$ which gives us the recursive formulation $P_l = (P_{l+1} \cdot W_{l+1}) \circ g_l'(S_l)$

The detailed proof is in Section 4.

Note that $P_{l+1} \cdot W_{l+1}$ is the inner-product of the $|O_{l+1}|$ size vector P_{l+1} and the $|O_{l+1}| \times |I_{l+1}|$ size matrix W_{l+1}, and the resultant $|I_{l+1}| = |O_l|$ size vector $P_{l+1} \cdot W_{l+1}$ is multiplied pointwise (Hadamard product) with the $|O_l|$ size vector $g_l'(S_l)$ to yield the $|O_l|$ size vector P_l.

$P_l \otimes I_l$ is the matrix outer-product of the $|O_l|$ size vector P_L and the $|I_l|$ size vector I_l. Hence, $\frac{\partial Loss}{\partial W_l}$ is a matrix of size $|O_l| \times |I_l|$

If we do L2 regularization (with λ_l as the regularization coefficient in layer l), then:

$$\frac{\partial Loss}{\partial W_l} = P_l \otimes I_l + 2\lambda_l W_l$$

If we do L1 regularization (with λ_l as the regularization coefficient in layer l), then:

$$\frac{\partial Loss}{\partial W_l} = P_l \otimes I_l + \lambda_l sign(W_l)$$

where $sign(W_l)$ is the sign operation applied pointwise on the elements of the matrix W_l.

4. Derivation of Gradient Formulation for GLM conditional probability density and Canonical Link Function

Here we show that the formula for gradient shown in the previous section is applicable when the supervisory variable has a conditional probability density modeled as the Generalized Linear Model(GLM)'s "normal-form" probability density (i.e., fairly generic family of probability distributions), when the output layer's activation function is the GLM canonical link function, and when the loss function is cross-entropy loss. First, we give a quick summary of GLM (*independent of neural networks*).

4.1. **The GLM Setting.**

In the GLM setting, the conditional probability density of the supervisory variable Y is modeled as:

$$f(Y|\theta,\tau) = h(Y,\tau) \cdot e^{\frac{b(\theta)\cdot T(Y) - A(\theta)}{d(\tau)}}$$

where θ should be thought of as the "center" parameter (related to the mean) of the probability distribution and τ should be thought of as the "dispersion" parameter (related to the variance) of the distribution. $h(\cdot,\cdot), b(\cdot), A(\cdot), d(\cdot)$ are general functions whose specializations define the family of distributions that can be modeled in the GLM framework. Note that this form is a generalization of the exponential-family functional form incorporating the "dispersion" parameter τ (for this reason, the above form is known as the "overdispersed exponential family" of distributions).

It is important to mention here that the GLM framework operates even when the supervisory variable Y is multi-dimensional. We will denote the dimension of Y and θ as K (eg: for classification, this would mean K classes). Note that $b(\theta) \cdot T(Y)$ is the inner-product between the vectors $b(\theta)$ and $T(Y)$.

When we specialize the above probability density form to the so-called "normal-form" (which means both $b(\cdot)$ and $T(\cdot)$ are identity functions), then the conditional probability density of the supervisory variable is (this is the form we work with in this paper):

$$f(Y|\theta,\tau) = h(Y,\tau) \cdot e^{\frac{\theta\cdot Y - A(\theta)}{d(\tau)}}$$

The GLM link function $q(\cdot)$ is the function that transforms the linear predictor to the mean of the supervisory variable Y conditioned on the linear predictor η (linear predictor η is the inner-product of the input vector and the parameters matrix). Therefore,

$$E[Y|\theta] = q(\eta)$$

We will specialize the GLM link function $q(\cdot)$ to be the canonical link function ("canonical" simply means that θ is equal to the linear predictor η). So for a canonical link function $q(\cdot)$,

$$E[Y|\theta] = q(\theta)$$

Note that q transforms a vector of length K to a vector of length K ($\theta \rightarrow E[Y|\theta]$).

4.2. **Key Result.** Since

$$\int_y f(y|\theta, \tau) dy = 1,$$

it's partial derivatives with respect to θ will be zero. In other words,

$$\frac{\partial (\int_y f(y|\theta, \tau) dy)}{\partial \theta} = 0$$

Hence,

$$\frac{\partial (\int_y h(y, \tau) \cdot e^{\frac{\theta \cdot y - A(\theta)}{d(\tau)}} dy)}{\partial \theta} = 0$$

Taking the partial derivative inside the integral, we get:

$$\int_y h(y, \tau) \cdot e^{\frac{\theta \cdot y - A(\theta)}{d(\tau)}} \cdot \frac{y - \frac{\partial A}{\partial \theta}}{d(\tau)} dy = 0$$

$$\int_y f(y|\theta, \tau) \cdot (y - \frac{\partial A}{\partial \theta}) dy = 0$$

$$E[(Y|\theta) - \frac{\partial A}{\partial \theta}] = 0$$

This gives us the key result (which we will utilize later in our proof):

$$E[Y|\theta] = \frac{\partial A}{\partial \theta} \text{ (which as we know from above is } = q(\theta))$$

As an aside, we want to mention that apart from allowing for a non-linear relationship between the input and the mean of the supervisory variable, GLM allows for heteroskedastic models (i.e., the variance of the supervisory variable is a function of the input variables). Specifically,

$$Variance[Y|\theta] = \frac{\partial^2 A}{\partial \theta^2} \cdot d(\tau) \text{ (derivation similar to that of } E[Y|\theta] \text{ above)}$$

4.3. **Common Examples of GLM.** Several standard probability distributions paired with their canonical link function are special cases of "normal-form" GLM. For example,

- Normal distribution $N(\mu, \sigma)$ paired with the identity function ($q(\theta) = \theta$) is Linear Regression ($\theta = \mu, \tau = \sigma, h(Y, \tau) = \frac{e^{\frac{-Y^2}{2\tau^2}}}{\sqrt{2\pi}\tau}, A(\theta) = \frac{\theta^2}{2}, d(\tau) = \tau^2$).
- Bernoulli distribution parameterized by p paired with the logistic function ($q(\theta) = \frac{1}{1+e^{-\theta}}$) is Logistic Regression ($\theta = \log(\frac{p}{1-p}), \tau = h(Y, \tau) = d(\tau) = 1, A(\theta) = \log(1 + e^{\theta})$).
- Poisson distribution parameterized by λ paired with the exponential function ($q(\theta) = e^{\theta}$) is Log-Linear Regression ($\theta = \log \lambda, \tau = d(\tau) = 1, h(Y, \tau) = \frac{1}{Y!}, A(\theta) = e^{\theta}$).

4.4. **GLM in a Dense Feed-Forward Deep Neural Network.** Now we come to our dense feed-forward deep neural network setting. Let us assume that the supervisory variable Y has a conditional probability distribution of the GLM "normal-form" expressed above and let us assume that the activation function of the output layer $g_L(\cdot)$ is the canonical link function $q(\cdot)$. Then,

$$E[Y|\theta] = \frac{\partial A}{\partial \theta} = q(\theta) = q(I_L \cdot W_L) = g_L(I_L \cdot W_L) = g_L(S_L) = O_L$$

In the above equation, we note that each of Y, θ, S_L, O_L is a vector of length K.

The Cross-Entropy Loss (Negative Log-Likelihood) of the training data point (X, Y) is given by:

$$Loss = -\log\left(h(Y, \tau)\right) + \frac{A(\theta) - \theta \cdot Y}{d(\tau)}$$

We define the "Proxy Error" P_l for layer l (for all $l = 0, 1, \ldots L$) as:

$$P_l = d(\tau) \cdot \frac{\partial Loss}{\partial S_l}$$

4.5. **Recursion Termination.** We want to first establish the termination of the recursion for P_l, i.e., we want to establish that:

$$P_L = d(\tau) \cdot \frac{\partial Loss}{\partial S_L} = d(\tau) \cdot \frac{\partial Loss}{\partial \theta} = O_L - Y = E$$

To calculate P_L, we have to evaluate the partial derivatives of $Loss$ with respect to the θ vector.

$$\frac{\partial Loss}{\partial \theta} = \frac{\frac{\partial A}{\partial \theta} - Y}{d(\tau)}$$

It follows that:

$$P_L = d(\tau) \cdot \frac{\partial Loss}{\partial \theta} = \frac{\partial A}{\partial \theta} - Y = q(\theta) - Y = g_L(\theta) - Y = O_L - Y$$

This establishes the termination of the recursion for P_l, i.e. $P_L = O_L - Y = E$ (the "error" in the output of the output layer). It pays to re-emphasize that this result ($P_L = E$) owes a Thank You to the key result $\frac{\partial A}{\partial \theta} = q(\theta)$, which is applicable when the conditional probability density of the supervisory variable is in GLM "normal form" and when the link function $q(\cdot)$ is canonical. We are fortunate that this is still a fairly general setting and so, it applies to a wide range of regressions and classifications.

4.6. **Recursion.** Next, we want to establish the following recursion for P_l for all $l = 0, 1, \ldots, L-1$:

$$P_l = (P_{l+1} \cdot W_{l+1}) \circ g_l'(S_l)$$

We note that:

$$S_{l+1} = g_l(S_l) \cdot W_{l+1}$$

Hence,

$$P_l = d(\tau) \cdot \frac{\partial Loss}{\partial S_l} = d(\tau) \cdot \frac{\partial Loss}{\partial S_{l+1}} \cdot \frac{\partial S_{l+1}}{\partial S_l} = d(\tau) \cdot \frac{\partial Loss}{\partial S_{l+1}} \cdot W_{l+1} \circ g_l'(S_l) = (P_{l+1} \cdot W_{l+1}) \circ g_l'(S_l)$$

This establishes the recursion for P_l for all $l = 0, 1, \ldots, L-1$. Again, it pays to re-emphasize that this recursion has nothing to do with GLM, it only has to do with the structure of the feed-forward network: $S_{l+1} = g_l(S_l) \cdot W_{l+1}$.

4.7. **Loss Gradient Formulation.** Finally, we are ready to express the gradient of the loss function with respect to the parameters of layer l (for all $l = 0, 1, \ldots L$).

$$\frac{\partial Loss}{\partial W_l} = \frac{\partial Loss}{\partial S_l} \cdot \frac{\partial S_l}{\partial W_l} = P_l \otimes I_l \text{ (ignoring the constant factor } d(\tau))$$

5. Regression on a dense feed-forward deep neural network

Linear Regression is a special case of the GLM family because the linear regression loss function is simply the cross-entropy loss function (negative log likelihood) when the supervisory variable Y follows a linear-predictor-conditional normal distribution and the canonical link function for the normal distribution is the identity function. Here we consider a dense feed-forward deep neural network whose output layer activation function is the identity function and we define the loss function to be mean-squared error (as follows):

$$(I_L \cdot W_L - Y)^2$$

The Cross-Entropy Loss (Negative Loss Likelihood) when $Y|(I_L \cdot W_L)$ is a normal distribution $N(\mu, \sigma)$ is as follows:

$$-\log\big(\frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(Y-(I_L \cdot W_L))^2}{2\sigma^2}}\big) = \frac{(Y-(I_L \cdot W_L))^2}{2\sigma^2} - \log\big(\frac{1}{\sqrt{2\pi}\sigma}\big)$$

Since σ is a constant, the gradient of cross-entropy loss is same as the gradient of mean-squared error up to a constant factor. Hence, the gradient formulation we derived for GLM is applicable here.

6. Classification on a dense feed-forward deep neural network

Softmax Classification is a special case of the GLM family where the supervisory variable Y follows a linear-predictor-conditional Multinoulli distribution and the canonical link function for Multinoulli is Softmax. Here we consider a dense feed-forward deep neural network whose output layer activation function is softmax and we define the loss function to be cross-entropy loss.

We assume K classes. We denote the r^{th} element of O_L (output of r^{th} neuron in the output layer) as $O_L^{(r)}$, the r^{th} element of the training data point Y as $Y^{(r)}$ and the r^{th} element of S_L as $S_L^{(r)}$ for all $r = 1, \ldots K$. The supervisory variable Y will be have a "one-hot" encoding of length K as follows: . If the supervisory variable is meant to represent class $k \in [1, K]$, then $Y^{(k)} = 1$ and $Y^{(r)} = 0$ for all $r = 1, \ldots, k-1, k+1, \ldots K$ (the non-k elements of Y are 0).

The softmax classification cross-entropy loss function is given by:

$$-\sum_{r=1}^{K} (Y^{(r)} \cdot \log O_L^{(r)})$$

where

$$O_L^{(r)} = \frac{e^{S_L^{(r)}}}{\sum_{i=1}^{K} e^{S_L^{(i)}}}$$

We can write the above cross-entropy loss function as the negative log-likelihood of a Multinoulli $((p_1, p_2, \ldots, p_K))$ distribution that is a special case of a GLM distribution:

$$-\log f(Y|\theta,\tau) = -\log\left(h(Y,\tau)\cdot e^{\frac{\theta\cdot Y - A(\theta)}{d(\tau)}}\right) = -\log\left(h(Y,\tau)\right)\cdot\frac{A(\theta)-\theta\cdot Y}{d(\tau)}$$

with the following specializations:

$$\tau = h(Y,\tau) = d(\tau) = 1$$
$$\theta^{(r)} = \log(p_r) = \log(O_L^{(r)}) = S_L^{(r)} \text{ for all } r = 1,\ldots,K$$
$$A(\theta) = A(\theta^{(1)},\ldots,\theta^{(K)}) = \log\left(\sum_{r=1}^{K} e^{\theta^{(r)}}\right)$$

The canonical link function for Multinoulli distribution is the softmax function $g(\theta)$ given by:

$$\frac{\partial A}{\partial \theta} = g(\theta) = g(\theta^{(1)},\theta^{(2)},\ldots,\theta^{(K)}) = \left(\frac{e^{\theta^{(1)}}}{\sum_{r=1}^{K} e^{\theta^{(r)}}}, \frac{e^{\theta^{(2)}}}{\sum_{r=1}^{K} e^{\theta^{(r)}}}, \ldots, \frac{e^{\theta^{(K)}}}{\sum_{r=1}^{K} e^{\theta^{(r)}}}\right)$$

So we can see that:

$$O_L = g(\theta) = g(S_L)$$

Hence, the gradient formulation we derived for GLM is applicable here.