RECURSIVE FORMULATION OF THE 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 gradient. In the appendices, we derive this formulation of the gradient for the cases of regression as well as classification, and finally we show that **this formulation also applies to the more general case where the supervisory variable has a conditional probability density modeled as an arbitrary Generalized Linear Model(GLM)'s "standard-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** (Regression and Classification are special cases of this generic GLM structure).

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 $l = 0, 1, \ldots, L-1$. 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 prediction for input X (that will be supervised by Y).

Note that the number of neurons in layer l will be the number of outputs of layer l $(= |O_l|)$. Note that $|O_L| = 1$ for regression and $|O_L| = K$ for classification where K is the number of classes. We will denote the supervision error on this network as E (for the said training data point). For regression, E is the scalar $O_L - Y$. For classification, E is denoted by the vector $O_L - Y$. For classification, if we assume that the supervisory variable Y takes on the value $k \in [1, \ldots, K]$ (for the said training data point) and if we denote the r^{th} element of O_L (output of r^{th} neuron in the output layer) as $O_L^{(r)}$ for all $r = 1, \ldots K$, then the k^{th} element of the vector $O_L - Y$ is $O_L^{(k)} - 1$ and the other (non-k) elements of the vector $O_L - Y$ 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 $A_l(\cdot)$ for all $l = 0, 1, \ldots, L-1$ (the activation function for layer L will be the identity function for regression and softmax for classification).

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)
$A_l(\cdot)$	Activation function for layer l for $l = 0, 1, \ldots, L-1$
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 $l = 0, 1, \ldots, L$:

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

with the recursion terminating as:

$$P_L = E = O_L - Y$$

Appendix A provides a proof of the above for the 2 layer case for regression (where the activation function of the output layer is the identity function and the loss function is mean-squared-error). Appendix B shows that these formulas also hold for classification (where the activation function of the output layer is softmax and the loss function is cross-entropy loss). **Appendix C proves that these formulas are applicable more generically to the case where the supervisory variable has a conditional probability density modeled as the Generalized Linear Model(GLM)'s "standard-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.**

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 $A_l'(O_l)$ to yield the $|O_l|$ size vector P_l.

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

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

$P_l \otimes I_l$ 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:

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

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

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

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

Appendices

A. Gradient for Regression for a Two Layer Network

Here we provide a proof of the gradient formulation in terms of "proxy error" for the 2 layer case for regression (where the activation function of the output layer is the identity function and the loss function is mean-squared-error).

Consider a dense feed-forward neural network with two layers (one hidden layer and one output layer). Let the input training data point to this network be denoted by the vector $X = (X_1, X_2, \ldots, X_m)$ and the associated supervisory value be denoted by the scalar Y (regression problem). Let there be n neurons in the hidden layer with the parameters in neuron j denoted by W_{ji} for all $i = 1, 2, \ldots, m$ for all $j = 1, 2, \ldots, n$. Let the parameters in the output layer be denoted by $W_1, W_2, \ldots, W_n$. For ease of exposition, we ignore the bias parameters (as we can always treat X as augmented by a "fake" feature that is always 1). So we have $mn + m$ parameters to train. Let the activation function of the hidden layer be denoted by $g(\cdot)$. Let the regularization parameter (assuming L2 regularization) be λ_1 for the hidden layer and λ_2 for the output layer.

The Loss function

$$Loss(W_{11}, W_{12}, \ldots, W_{1m}, \ldots\ldots, W_{n1}, W_{n2}, \ldots, W_{nm}, W_1, W_2, \ldots W_n)$$

$$= (\sum_{j=1}^{n} W_j g(\sum_{i=1}^{m} X_i W_{ji}) - Y)^2 + \lambda_1 \sum_{j=1}^{n} \sum_{i=1}^{m} W_{ji}^2 + \lambda_2 \sum_{j=1}^{n} W_j^2$$

The gradient of the Loss function with respect to the parameters in the output layer is defined by:

$$\frac{\partial Loss}{\partial W_j} = 2(\sum_{j=1}^{n} W_j g(\sum_{i=1}^{m} X_i W_{ji}) - Y) \cdot g(\sum_{i=1}^{m} X_i W_{ji}) + 2\lambda_2 W_j$$

$$= 2 \cdot LayerProxyError \otimes LayerInput + 2 \cdot LayerLambda \cdot LayerParameters$$

The gradient of the Loss function with respect to the parameters in the hidden layer is defined by:

$$\frac{\partial Loss}{\partial W_{ji}} = 2(\sum_{j=1}^{n} W_j g(\sum_{i=1}^{m} X_i W_{ji}) - Y) \cdot W_j \cdot g'(\sum_{i=1}^{m} X_i W_{ji}) \cdot X_i + 2\lambda_1 W_{ji}$$

$$= 2 \cdot ((NextLayerProxyError \cdot NextLayerParameters) \circ LayerActivationDerivative) \otimes LayerInput$$

$$+ 2 \cdot LayerLambda \cdot LayerParameters$$

$$= 2 \cdot LayerProxyError \otimes LayerInput + 2 \cdot LayerLambda \cdot LayerParameters$$

If we have more hidden layers, the same differentiation chain-rule structure applies because the network structure/functional forms will be the same across all the hidden layers. Hence, the recursive formulation of the gradient will apply across all the hidden layers.

B. Gradient with respect to Output Layer Parameters for Classification

Here we show that the formula for gradient with respect to the output layer parameters is the same for classification as the formula we derived for regression in Appendix A.

Consider a dense feed-forward neural network for classification where the output layer produces K probability outputs, one for each of the classes $k = 1, 2, \ldots, K$ (the K probabilities summing to 1). Let the output of the final hidden layer (input to the output layer) for a given training data point be denoted by $H = (H_1, H_2, \ldots, H_n)$ and let the corresponding output from the output layer be denoted by $R = (R_1, R_2, \ldots, R_K)$. Let the supervisory value for the said training data point be denoted by $Y \in [1, 2, \ldots, K]$. Define"one-hot" variables Y_k such that $Y_k = 1$ if $Y = k$ and $Y_k = 0$ if $Y \neq k$ for all $k = 1, 2, \ldots K$. Let the parameters in the output layer be denoted by W_{kj} for all $j = 1, 2, \ldots, n$ for all $k = 1, 2, \ldots K$.

The classification cross-entropy loss is given by the Loss function

$$Loss(W_{11}, W_{12}, \ldots, W_{1n}, \ldots\ldots, W_{K1}, W_{K2}, \ldots, W_{Kn}) = -\sum_{k=1}^{K}(Y_k \cdot \log R_k + (1-Y_k) \cdot \log(1-R_k))$$

where

$$R_k = \frac{e^{S_k}}{\sum_{i=1}^{K} e^{S_i}}$$

where

$$S_k = \sum_{j=1}^{n} H_j W_{kj}$$

By the chain rule,

$$\frac{\partial Loss}{\partial W_{kj}} = \frac{\partial Loss}{\partial R_k} \cdot \frac{\partial R_k}{\partial S_k} \cdot \frac{\partial S_k}{\partial W_{kj}}$$

$$\frac{\partial Loss}{\partial R_k} = -\frac{Y_k}{R_k} + \frac{1-Y_k}{1-R_k} = \frac{R_k - Y_k}{R_k \cdot (1-R_k)}$$

Note that R_k can be written as:

$$R_k = \frac{1}{1 + \alpha \cdot e^{-S_k}}$$

where

$$\alpha = \sum_{i \neq k} e^{S_i}$$

Noting that α is independent of S_k,

$$\frac{\partial R_k}{\partial S_k} = \frac{\alpha \cdot e^{-S_k}}{(1 + \alpha \cdot e^{-S_k})^2} = \frac{1}{1 + \alpha \cdot e^{-S_k}} \cdot \frac{\alpha \cdot e^{-S_k}}{1 + \alpha \cdot e^{-S_k}} = \frac{1}{1 + \alpha \cdot e^{-S_k}} \cdot (1 - \frac{1}{1 + \alpha \cdot e^{-S_k}}) = R_k \cdot (1 - R_k)$$

$$\frac{\partial S_k}{\partial W_k j} = H_j$$

Hence,

$$\frac{\partial Loss}{\partial W_{kj}} = \frac{\partial Loss}{\partial R_k} \cdot \frac{\partial R_k}{\partial S_k} \cdot \frac{\partial S_k}{\partial W_{kj}} = \frac{R_k - Y_k}{R_k \cdot (1 - R_k)} \cdot R_k \cdot (1 - R_k) \cdot H_j = (R_k - Y_k) \cdot H_j$$

$R_k - Y_k$ is the "error" of the output from the output layer and H_j is the input to the output layer. Hence, the gradient of the classification cross-entropy loss function with respect to the output layer parameters can be expressed as the outer product of the "error" of the output from the output layer and the input to the output layer, which is exactly the form of the gradient of the regression loss function with respect to the parameters of its output layer.

Note that the gradient with respect to the hidden-layer parameters will have the same recursive structure as our derivation for the regression case because the structure of the network/functional forms prior to the output layer (hidden layers) is exactly the same as that of regression, and hence, the same differentiation chain-rule structure applies as in the case of our derivation for regression.

C. Generalization: GLM conditional probability density and Canonical Link

Here we show that the formula for gradient with respect to the output layer parameters that we previously derived for regression and classification applies more generically to the case where the supervisory variable has a conditional probability density modeled as the Generalized Linear Model(GLM)'s "standard-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.

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

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

where θ should be thought of as the "location" parameter of the probability distribution and τ should be thought of as the "scale" parameter 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.

When we specialize the above probability density form to the so-called "standard-form", the conditional probability density of the supervisory variable is modeled as:

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

Further, if we specialize the GLM link function g (defined as $g(X \cdot \beta) = Y$ where X is the input vector and β is the parameter vector) to be the canonical link function, then we have:

$$\mu = E[Y] = g(\theta)$$

Let us denote the parameters in the output layer by the vector W. Let us assume that the activation function of the output layer is the canonical link function $g(\cdot)$. Let us denote the output of the final hidden layer (input to the output layer) for a given training data point by the vector H and so, the corresponding output of the network would be $g(H \cdot W)$. Let the supervisory value for the said training data point be denoted by Y and we assume that $Probability(Y|g(H \cdot W))$ follows the above-mentioned "standard-form" probability density.

In this setting,

$$\mu = E[Y] = g(\theta) = g(H \cdot W) = A'(\theta)$$

and

$$Var[Y] = A''(\theta) \cdot d(\tau)$$

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

$$Loss(W) = -\log(h(Y,\tau) * e^{\frac{H \cdot W * Y - A(H \cdot W)}{d(\tau)}}) = -\log(h(y,\tau)) + \frac{A(H \cdot W) - (H \cdot W) * Y}{d(\tau)}$$

$$\nabla_W Loss = \frac{\partial Loss}{\partial \theta} \cdot \nabla_W \theta$$

So all we are left to prove is that $\frac{\partial Loss}{\partial \theta}$ is equal to a constant factor of $g(H \cdot W) - Y$ (the "error" in the output from the output layer).

Expressing the loss function in terms of θ,

$$Loss(\theta) = -\log(h(y,\tau)) + \frac{A(\theta) - \theta * Y}{d(\tau)}$$

$$\frac{\partial Loss}{\partial \theta} = \frac{A'(\theta) - Y}{d(\tau)} = \frac{\mu - Y}{d(\tau)} = \frac{g(\theta) - Y}{d(\tau)} = \frac{g(H \cdot W) - Y}{d(\tau)}$$

Since $d(\tau)$ is a constant and since $g(H \cdot W) - Y$ is the "error" in the output from the output layer, this completes the proof.

Note that the gradient with respect to the hidden-layer parameters will have the same recursive structure as our derivation for the regression case because the structure of the network/functional forms prior to the output layer (hidden layers) is exactly the same as that of regression, and hence, the same differentiation chain-rule structure applies as in the case of our derivation for regression.

We note that regression is a special case of this GLM formulation because the regression loss function is simply the cross-entropy loss function (negative log likelihood) when the supervisory variable Y follows a conditional normal distribution and the canonical link function for the normal distribution is the identity function. Likewise, classification is a special case of this GLM formulation because the supervisory variable Y follows a conditional Multinoulli distribution and the canonical link function for Multinoulli is softmax.