Vectorized Back-propagation Derivation

VERSION: 1.0

Courtesy of Nate Rojas 2019

Nate Rojas 2019 rojasinate@gmail.com

Introduction

This is a derivation for the vectorized back propagation equations of a standard feed forward neural network with input matrix X where there are m column feature vectors.

Disclaimer

There are many pictographic presentations that provide intuition. Considering this, I have *not* attempted to recreate the hard work put into those articles and blogs. Therefore, what I have done is attempt to move from that intuition into rigorous mathematics in a succinct manner. Additionally, I have tried to use only those mathematical concepts that are required to derive the back propagation equations. Specifically, I do not assume familiarity with matrix calculus. That being said, one should be familiar with basic linear algebra and enough calculus to understand the chain rule.

Before Diving In

If you have any questions/concerns/edits please contact the author (rojasinate@gmail.com). Feel free to print this out for your own reference if you find it useful (note: please credit the author if you use the content for more than your own personal use - thanks).

Notation

This section defines notation that will be used later. Note that this derivation and notation was motivated by and constructed to be as much inline with A. Ng's presentation of back propagation in his Deep Learning Specialization as possible. (Though I need to double check, I think that the indexing used in this derivation are slightly different from that used in the specialization.)

Operations

$\odot$ - denotes Hadamard product (element wise product)

$\times$ - denotes matrix product

$\cdot$ - denotes scalar product

Index Notation

Superscripts:

- Square brackets are used to index layers
- Parentheses are used to index along the examples

Subscripts:These are standard row and column

For example, $\Theta^{[l]}_{i,j}$ is the entry in the *i-th* row and *j-th* column of the weight matrix of the l*-th* layer.

Index Notation Justification

If you are okay with the notation you may skip this.

The subscripts don't require justification.

The superscripts are done this way because not every object under consideration is indexed with respect to a layer and similarly not every object is indexed with respect to an example. Specifically, a target y is only indexed with respect to examples and subscripts namely they are

of form $y_j^{(i)}$. Furthermore, some targets only require the superscript (eg binary classification). Similarly, activation functions are only indexed with respect to layer - namely they are of form $\alpha^{[l]}$. Thus, this indexing convention makes it very clear *what* object is under consideration and avoids syntactic complications.

Forward Prop Structure
Here the flow of data through the network is presented in a form that is graphical without sacrificing rigor and succinctness.

$$(1) \quad X = A^{[0]} \xrightarrow{\Theta^{[1]}+b^{[1]}} [Z^{[1]} \xrightarrow{\alpha^{[1]}} A^{[1]}] \xrightarrow{\Theta^{[2]}+b^{[2]}} \dots \xrightarrow{\Theta^{[L]}+b^{[L]}} [Z^{[L]} \xrightarrow{\alpha^{[L]}} A^{[L]}] \xrightarrow{\frac{1}{m}\sum_m \sum_{n_L} err\{\dots,y\}} J$$

Note: the final data transformation leaves the indices implicit.

Inputs

$$X = \begin{bmatrix} \vdots & & \vdots \\ x^{(1)} & \dots & x^{(m)} \\ \vdots & & \vdots \end{bmatrix}$$

Weights
The weight matrix for the l-th layer.

$$\Theta^{[l]} = \begin{bmatrix} \dots & \theta_1^{[l]} & \dots \\ & \vdots & \\ \dots & \theta_{n_l}^{[l]} & \dots \end{bmatrix}$$

Hyperparameters
n_l - denotes the number of neurons in the l-th layer.

Forward Prop: Intermediate Terms
This derivation considers back propagation with respect to an entire batch. Though, one can simply consider the batch matrix as a mini-batch and the derivation is still valid. Here, the forward propagation terms are defined as:

$$Z^{[l]} = \begin{bmatrix} Z_1^{[l](1)} & \dots & Z_1^{[l](m)} \\ \vdots & & \vdots \\ Z_{n_l}^{[l](1)} & \dots & Z_{n_l}^{[l](m)} \end{bmatrix} = \begin{bmatrix} z_1^{[l](1)} + b_1^{[l]} & \dots & z_1^{[l](m)} + b_1^{[l]} \\ \vdots & & \vdots \\ z_{n_l}^{[l](1)} + b_{n_l}^{[l]} & \dots & z_{n_l}^{[l](m)} + b_{n_l}^{[l]} \end{bmatrix}.$$

Where $z_i^{[l](j)} = row_i(\Theta^{[l]}) \times col_j(A^{[l-1]})$. Observe that this formalism makes use of bias terms as opposed to bias units - this choice simplifies the derivations.

$$A^{[l]} = \begin{bmatrix} A_1^{[l](1)} & \dots & A_1^{[l](m)} \\ \vdots & & \vdots \\ A_{n_l}^{[l](1)} & \dots & A_{n_l}^{[l](m)} \end{bmatrix} = \begin{bmatrix} \alpha^{[l]}(Z_1^{[l](1)}) & \dots & \alpha^{[l]}(Z_1^{[l](m)}) \\ \vdots & & \vdots \\ \alpha^{[l]}(Z_{n_l}^{[l](1)}) & \dots & \alpha^{[l]}(Z_{n_l}^{[l](m)}) \end{bmatrix}.$$

With degenerate $A^{[0]} = X$.

Cost Function J

Next we need to consider the cost function which is the object of most concern.

$J(\Theta)$ is the cost *function* with respect the *all* the weights/parameters. That is, Θ denotes the set of *all* weight matrices which can be indexed by layer as mentioned above.

From (1) we can form $J(\Theta)$ explicitly:

$$J(\Theta) = \frac{1}{m}\sum_{c=1}^{m}\sum_{i=1}^{n_L} err\{A_i^{[L](c)}, y_i^{(c)}\}.$$

In the case where the output layer is a single real number we can drop the summation $\sum_{i=1}^{n_L}(\ldots)$.

Continuing, we can "expand" the inner term $A_i^{[L](c)}$ yielding:

$$J(\Theta) = \frac{1}{m}\sum_{c=1}^{m}\sum_{i=1}^{n_L} err\{\alpha^{[L]}(Z_i^{[L](c)}), y_i^{(c)}\}.$$

Finally, this expansion results in the long form where we see the inputs X in the deepest nested sum (blue):

(2) $$J(\Theta) = \frac{1}{m}\sum_{c=1}^{m}\sum_{i=1}^{n_L} err\{\alpha^{[L]}(\sum_{j=1}^{n_{L-1}}\Theta_{i,j}^{[L]}\cdot\alpha^{[L-1]}(\ldots(\sum_{s=1}^{n_1}\Theta_{r,s}^{[1]}\cdot X_s^{(c)} + b_s^{[1]})\ldots) + b_i^{[L]}), y_i^{(c)}\}.$$

Thus, we see *exactly* how J is a *function* of Θ.

Back Prop.

Without making appeals to matrix calculus one can still motive the formalism as follows:
In reality what we are doing is trying to calculate the partial derivative of $J(\Theta)$ with respect to each *individual* parameter occurring in Θ. Additionally, since our network parameters are stored in structured matrices, the updates should *also* be structured such that we may make vectorized updates.

Preliminaries

For backward propagation we want to calculate the gradient by working on (1) from right to left. Before we can do this, let's inspect (1) & (2) - repeated here for convenience:

(1) $$X = A^{[0]} \xrightarrow{\Theta^{[1]}+b^{[1]}} [Z^{[1]} \xrightarrow{\alpha^{[1]}} A^{[1]}] \xrightarrow{\Theta^{[2]}+b^{[2]}} \dots \xrightarrow{\Theta^{[L]}+b^{[L]}} [Z^{[L]} \xrightarrow{\alpha^{[L]}} A^{[L]}] \xrightarrow{\frac{1}{m}\sum_{m}\sum_{n_L} err\{\dots,y\}} J$$

(2) $$J(\Theta) = \frac{1}{m}\sum_{c=1}^{m}\sum_{i=1}^{n_L} err\{\alpha^{[L]}(\sum_{j=1}^{n_{L-1}} \Theta_{i,j}^{[L]} \cdot \alpha^{[L-1]}(\dots(\sum_{s=1}^{n_1} \Theta_{r,s}^{[1]} \cdot X_s^{(c)} + b_s^{[1]})\dots) + b_i^{[L]}), y_i^{(c)}\}$$

Observations

A. Every parameter $\Theta_{i,j}^{[l]}$ is deeply nested in compositions of functions. This implies, extensive use of the chain rule.
B. The parameters for the layers *closer to the output* are not as deeply embedded in function composition paths. This implies, that as we calculate deeper and deeper partial derivatives we will have to make longer and longer appeals to the chain rule.
C. Observations (A) and (B) imply that we should *first* calculate the partial derivatives for the parameters of the *later* layers and *work our way backward*. This intuition is key for our later inspections which will present the vectorized back propagation equations.

The Derivation

Step one of the derivation is to calculate the partial derivatives of $J(\Theta)$ with respect to each entry in $A^{[L]}$[1]. Namely:

(3) $$\frac{\partial}{\partial ent_{i,j}(A^{[L]})} J(\Theta) \, .$$

Store each result of form (3) in a matrix. Denote this matrix as

(4) $$\frac{\partial}{\partial A^{[L]}} J(\Theta) = dA^{[L]} \, .$$

Evaluating the entries in (3) and calculating the partials we see that

(5) $$dA^{[L]} = \begin{bmatrix} \frac{\partial}{\partial A_1^{[L](1)}} J(\Theta) & \dots & \frac{\partial}{\partial A_1^{[L](m)}} J(\Theta) \\ \vdots & & \\ \frac{\partial}{\partial A_{n_L}^{[L](1)}} J(\Theta) & \dots & \frac{\partial}{\partial A_{n_L}^{[L](m)}} \end{bmatrix} = \begin{bmatrix} err'\{A_1^{[L](1)}, y_1\} & \dots & err'\{A_1^{[L](m)}, y_1\} \\ \vdots & & \vdots \\ err'\{A_{n_L}^{[L](1)}, y_{n_L}\} & \dots & err'\{A_{n_L}^{[L](m)}, y_{n_L}\} \end{bmatrix}.$$

The next step is to repeat steps (3-5) for $Z^{[L]}$ we obtain

(6) $$\frac{\partial}{\partial Z^{[L]}} J(\Theta) = dZ^{[L]} := \begin{bmatrix} \frac{\partial}{\partial Z_1^{[L](1)}} J(\Theta) & \dots & \frac{\partial}{\partial Z_1^{[L](m)}} J(\Theta) \\ \vdots & & \\ \frac{\partial}{\partial Z_{n_L}^{[L](1)}} J(\Theta) & \dots & \frac{\partial}{\partial Z_{n_L}^{[L](m)}} \end{bmatrix} =$$

[1] see post scriptum at end for an alternative starting

$$(7) \quad \begin{bmatrix} err'\{A_1^{[L](1)}, y_1\} \cdot \alpha'^{[L]}(Z_1^{[L](1)}) & \dots & err'\{A_1^{[L](m)}, y_1\} \cdot \alpha'^{[L]}(Z_1^{[L](m)}) \\ \vdots & & \vdots \\ err'\{A_{n_L}^{[L](1)}, y_{n_L}\} \cdot \alpha'^{[L]}(Z_{n_L}^{[L](1)}) & \dots & err'\{A_{n_L}^{[L](m)}, y_{n_L}\} \cdot \alpha'^{[L]}(Z_{n_L}^{[L](m)}) \end{bmatrix}.$$

By inspection we can see that (7) grants the following formula:

$$(8) \quad dZ^{[L]} = dA^{[L]} \odot \alpha'^{[L]}(Z^{[L]}) \,.$$

The next step is where the magic happens and retrospectively motivates (3-7). First, calculate the partials of every entry in $\Theta^{[L]}$ by hand and place this into a structured matrix.

As before denote this matrix as:

$$(9) \quad \frac{\partial}{\partial \Theta^{[L]}} J(\Theta) = d\Theta^{[L]} = \begin{bmatrix} \frac{\partial}{\partial \Theta_{1,1}^{[L](1)}} J(\Theta) & \dots & \frac{\partial}{\partial \Theta_{1,n_{L-1}}^{[L](m)}} J(\Theta) \\ \vdots & & \vdots \\ \frac{\partial}{\partial \Theta_{n_L,1}^{[L](1)}} J(\Theta) & \dots & \frac{\partial}{\partial \Theta_{n_L,n_{L-1}}^{[L](m)}} J(\Theta) \end{bmatrix}.$$

The evaluation of the entries in (9) yields:

$$(10) \quad \frac{\partial}{\partial \Theta_{i,j}^{[L]}} J(\Theta) = \frac{1}{m} \sum_{c=1}^{m} err'\{A_i^{[L](c)}, y_i\} \cdot \alpha'^{[L]}(Z_i^{[L](c)}) \cdot A_j^{[L-1](c)} \,.$$

One can verify by differentiating (2) w.r.t. an arbitrary parameter in $\Theta^{[L]}$

From (10) one can establish valid identities for (10) and finally obtain an equivalence:

$$(10.a) \quad \frac{1}{m} \sum_{c=1}^{m} ent_{i,c}(dZ^{[L]}) \cdot ent_{j,c}(A^{[L-1]})$$

$$(10.b) \quad \frac{1}{m} row_i(dZ^{[L]}) \times col_j((A^{[L-1]})^T) \quad \Leftrightarrow$$

$$(11) \quad d\Theta^{[L]} = \frac{1}{m} dZ^{[L]} \times (A^{[L-1]})^T \,.$$

The result in (11) follows from the definition of matrix multiplication.

Similarly, one can derive the update for the bias terms and obtain:

$$(12) \quad \frac{\partial}{\partial b^{[L]}} J(\Theta) = db^{[L]} = \frac{1}{m} \sum_{c=1}^{m} dZ^{[L](c)}$$

The final step to complete the derivation is to calculate $dA^{[L-1]}$. We proceed as before:

$$(13) \quad \frac{\partial}{\partial A^{[L-1]}} J(\Theta) = dA^{[L-1]} := \begin{bmatrix} \frac{\partial}{\partial A_1^{[L-1](1)}} J(\Theta) & \dots & \frac{\partial}{\partial A_1^{[L-1](m)}} J(\Theta) \\ \vdots & & \\ \frac{\partial}{\partial A_{n_{L-1}}^{[L-1](1)}} J(\Theta) & \dots & \frac{\partial}{\partial A_{n_{L-1}}^{[L-1](m)}} \end{bmatrix}.$$

Again - calculating however one likes - from (13) we obtain:

$$(14) \quad dA^{[L-1]} = \begin{bmatrix} \sum_{i=1}^{n_L} err'\{A_i^{[L](1)}, y_i\} \cdot \alpha'^{[L]}(Z^{[L](1)}) \cdot \Theta_{i,1}^{[L]} & \dots & \sum_{i=1}^{n_L} err'\{A_i^{[L](1)}, y_i\} \cdot \alpha'^{[L]}(Z^{[L](1)}) \cdot \Theta_{i,1}^{[L]} \\ \vdots & & \vdots \\ \sum_{i=1}^{n_L} err'\{A_i^{[L](1)}, y_i\} \cdot \alpha'^{[L]}(Z^{[L](1)}) \cdot \Theta_{i,n_{L-1}}^{[L]} & \dots & \sum_{i=1}^{n_L} err'\{A_i^{[L](1)}, y_i\} \cdot \alpha'^{[L]}(Z^{[L](1)}) \cdot \Theta_{i,n_{L-1}}^{[L]} \end{bmatrix}$$

By inspection or from sum-to-vector arguments (like those seen in steps (10.a, 10.b)) we obtain[2]:

$$(15) \quad dA^{[L-1]} = (\Theta^{[L]})^T \times dZ^{[L]}$$

At this point the calculations begin to repeat and we can extract the generalized definition. It should be noted that one could demonstrate the validity theoretically via an induction argument but this is unnecessary in this document as these equations have been verified *empirically.*

Summary

(A) $$d\Theta^{[l]} = \frac{1}{m} dZ^{[l]} \times (A^{[l-1]})^T$$

(B) $$db^{[l]} = \frac{1}{m} \sum_{c=1}^{m} dZ^{[l](c)}$$

(C) $$dZ^{[l]} = dA^{[l]} \odot \alpha'^{[l]}(Z^{[l]})$$

(D) $$dA^{[k]} = (\Theta^{[k+1]})^T \times dZ^{[k+1]}$$

(with the degenerate case when $k = L$ where $dA^{[L]}$ is defined at (5))

QED ■

PS This is a work in progress and there are many edits to come to improve the intuitive flow. For example, one could start at (10) and then motivate (3-8) via some inspection argument. Additionally, there are (with high probability) some typos - please forgive these and notify the author.

---

[2] Stay posted: future updates will make this explicit