Forward-Feeding Neural Networks (FFNNs): Canonical Transformation Functions, Their Derivatives & Their Corresponding Cost Functions

STUFF TO KNOW BY HEART - EVEN WHEN DRUNK!

*1. Three most common FFNN transformation functions: **Linear**, **Logistic** and **Softmax***

*2. An FFNN's **top layer** almost always involves one of these three functions*

*3. **Logistic** function models probabilities of 2 binary states (OFF & ON); **Softmax** models relative probabilities of more than 2 states*

*4. Corresponding canonical cost functions: **Squared Error**, **Cross Entropy (Logistic)** and **Cross Entropy (Softmax)***

This note discusses the three most common/standard ("canonical") transformation functions used in layers of FFNNs. These are **continous real-valued** functions that have **known, mathematically-convenient partial derivatives**. The top layer of an FFNN almost always involves one of these three transformation functions, and the FFNN's hypothesized output is evaluated by one of three corresponding canonical cost functions.

Linear Function

The **Linear** function transforms a “fan-in” activation matrix $\mathbf{A}_{\text{in}}$ (dimensions: <# of cases> x <# of “in” nodes>) to “fan-out” activation matrix $\mathbf{A}_{\text{out}}$ (dimensions: <# of cases> x <# of “out” nodes>) via weights $\mathbf{W}$ (dimensions: <# of “in” nodes> x <# of “out” nodes>) by linearly multiplying $\mathbf{A}_{\text{in}}$ and $\mathbf{W}$. This is the same as the Linear Regression model.

The Linear function and its backward partial derivative functions are defined below:

$$\mathbf{A}_{\text{out}} = f(\mathbf{A}_{\text{in}}, \mathbf{W}) = \mathbf{Z}, \text{ where } \mathbf{Z} = \mathbf{A}_{\text{in}}\mathbf{W}$$

$$\frac{\partial v}{\partial \mathbf{A}_{\text{in}}} = b_A\left(\frac{\partial v}{\partial \mathbf{A}_{\text{out}}}, \mathbf{A}_{\text{in}}, \mathbf{W}, \mathbf{A}_{\text{out}}\right) = \mathbf{W}\frac{\partial v}{\partial \mathbf{A}_{\text{out}}}$$

$$\frac{\partial v}{\partial \mathbf{W}} = b_W\left(\frac{\partial v}{\partial \mathbf{A}_{\text{out}}}, \mathbf{A}_{\text{in}}, \mathbf{W}, \mathbf{A}_{\text{out}}\right) = \frac{\partial v}{\partial \mathbf{A}_{\text{out}}}\mathbf{A}_{\text{in}}$$

The cost function corresponding to a Linear FFNN top layer is the **Squared Error** function, which measures the Euclidean geometric distance between Hypothesized Output $\mathbf{H}$ from the “right-answer” Target Output $\mathbf{Y}$. This function has a convenient partial derivative with resprect to $\mathbf{H}$.

$$c(\mathbf{H}, \mathbf{Y}) = \frac{|\mathbf{H} - \mathbf{Y}|^2}{\text{\# of Cases}}$$

$$\frac{\partial c}{\partial \mathbf{H}} = \frac{(\mathbf{H} - \mathbf{Y})^{\mathrm{T}}}{\text{\# of Cases}}$$

Logistic Function

The **Logistic** function transforms a “fan-in” activation matrix $\mathbf{A}_{\text{in}}$ (dimensions: <# of cases> x <# of “in” nodes>) to “fan-out” activation matrix $\mathbf{A}_{\text{out}}$ (dimensions: <# of cases> x <# of “out” nodes>) via weights $\mathbf{W}$ (dimensions: <# of “in” nodes> x <# of “out” nodes>) by first linearly multiplying $\mathbf{A}_{\text{in}}$ and $\mathbf{W}$, and then “squashing” the resulting values into the $(0, 1)$ unit interval. This is the same as the Logistic Regression model: the output represents probabilities, with values near 0 representing “likely OFF” and values near 1 representing “likely ON”.

The Logistic function and its backward partial derivative functions are defined below:

$$\mathbf{A}_{\text{out}} = f(\mathbf{A}_{\text{in}}, \mathbf{W}) = \mathbf{1} \;./\; \left(\mathbf{1} + \exp(-\mathbf{Z})\right), \text{ where } \mathbf{Z} = \mathbf{A}_{\text{in}}\mathbf{W}$$

$$\frac{\partial v}{\partial \mathbf{A}_{\text{in}}} = b_A\left(\frac{\partial v}{\partial \mathbf{A}_{\text{out}}}, \mathbf{A}_{\text{in}}, \mathbf{W}, \mathbf{A}_{\text{out}}\right) = \mathbf{W}\mathbf{B}$$

$$\frac{\partial v}{\partial \mathbf{W}} = b_W\left(\frac{\partial v}{\partial \mathbf{A}_{\text{out}}}, \mathbf{A}_{\text{in}}, \mathbf{W}, \mathbf{A}_{\text{out}}\right) = \mathbf{B}\mathbf{A}_{\text{in}}$$

$$\text{where: } \mathbf{B} = \frac{\partial v}{\partial \mathbf{A}_{\text{out}}} \;.\!*\; (\mathbf{A}_{\text{out}})^{\mathrm{T}} \;.\!*\; (\mathbf{1} - \mathbf{A}_{\text{out}})^{\mathrm{T}}$$

The Target Output $\mathbf{Y}$ for a Logistic FFNN top layer is a matrix of binary values 0 (“OFF”) and 1 (“ON”), and the cost function corresponding to a Logistic FFNN top layer is the **Cross Entropy (Logistic)** function, which is defined below together with its partial derivatives:

$$c(\mathbf{H}, \mathbf{Y}) = -\frac{\text{sumAllDims}\left(\mathbf{Y} \;.\!*\; \ln(\mathbf{H}) + (\mathbf{1} - \mathbf{Y}) \;.\!*\; \ln(\mathbf{1} - \mathbf{H})\right)}{\text{\# of Cases}}$$

$$\frac{\partial c}{\partial \mathbf{H}} = -\frac{\left(\mathbf{Y} \;./\; \mathbf{H} - (\mathbf{1} - \mathbf{Y}) \;./\; (\mathbf{1} - \mathbf{H})\right)^{\mathrm{T}}}{\text{\# of Cases}}$$

Softmax Function

The **Softmax** function generalizes the Logistic function. While the Logistic function models probabilities of 2 binary states OFF and ON, the Softmax function models **relative probabilities of more than 2 states**, with such probabilities summing to 1.

The Softmax function and its partial derivatives are defined below:

$$\begin{aligned}&\mathbf{A}_{\text{out}} = f(\mathbf{A}_{\text{in}}, \mathbf{W}) = \exp(\mathbf{Z}) \text{ ./ } \text{rowwiseSum}\big(\exp(\mathbf{Z})\big), \text{ where } \mathbf{Z} = \mathbf{A}_{\text{in}}\mathbf{W}\\ &\frac{\partial v}{\partial \mathbf{A}_{\text{in}}} = b_A(\frac{\partial v}{\partial \mathbf{A}_{\text{out}}}, \mathbf{A}_{\text{in}}, \mathbf{W}, \mathbf{A}_{\text{out}}) = \mathbf{WB}\\ &\frac{\partial v}{\partial \mathbf{W}} = b_W(\frac{\partial v}{\partial \mathbf{A}_{\text{out}}}, \mathbf{A}_{\text{in}}, \mathbf{W}, \mathbf{A}_{\text{out}}) = \mathbf{BA}_{\text{in}}\\ &\text{where } \mathbf{B} \text{ is a complicated, cumbersome but computable function}\end{aligned}$$

The Target Output $\mathbf{Y}$ (dimensions: <# of cases> x <# of ouput states>) for a Logistic FFNN top layer is a matrix of binary values 0 ("OFF") and 1 ("ON"), with each row having only one single 1 value representing the "ON" state for the case. The cost function corresponding to a Softmax FFNN top layer is the **Cross Entropy (Softmax)** function, which is defined below:

$$\begin{aligned}&c(\mathbf{H}, \mathbf{Y}) = -\frac{\text{sumAllDims}\big(\mathbf{Y} \text{ .* } \ln(\mathbf{H})\big)}{\text{\# of Cases}}\\ &\frac{\partial c}{\partial \mathbf{H}} = -\frac{(\mathbf{Y} \text{ ./ } \mathbf{H})^{\text{T}}}{\text{\# of Cases}}\end{aligned}$$

IMPLEMENTATION NOTE

In actual implementation, the partial derivative $\frac{\partial c}{\partial \mathbf{H}}$ is often numerically unstable to compute for a Logistic or Softmax FFNN top layer. One convenient and stable work-around applicable for all three canonical Linear, Logistic and Softmax functions is to compute $\frac{\partial c}{\partial \mathbf{Z}}$, where $\mathbf{Z} = \mathbf{A}_{\text{in}}\mathbf{W}$ for the FFNN's top layer. Interestingly, all three canonical functions have the very same and extremely convenient form for $\frac{\partial c}{\partial \mathbf{Z}}$:

$$\frac{\partial c}{\partial \mathbf{Z}} = \frac{(\mathbf{H} - \mathbf{Y})^{\text{T}}}{\text{\# of Cases}}$$