NEURAL NETWORKS
LTAT.02.001
MATRIX CALCULUS SURVIVAL GUIDE

ANTI INGEL

1. Prerequisites

1.1. Algebra

Prerequisite 1 (Matrix)**.** From algebra, we need matrices and matrix multiplications. Recall that matrix (over real numbers) can be thought of as a table of numbers

$$A = \begin{pmatrix} A_{11} & \dots & A_{1m} \\ \vdots & \ddots & \vdots \\ A_{n1} & \dots & A_{nm} \end{pmatrix}.$$

If a matrix has n rows and m columns it is said to be $n \times m$ matrix. We can also think of it as a function $A : \{1,\dots,n\} \times \{1,\dots,m\} \to \mathbb{R}$ that takes two numbers as input, row and column, and gives us the element at this cell.

Prerequisite 2 (Matrix multiplication)**.** Recall the definition of matrix multiplication. Given $n \times m$ matrix A and $m \times k$ matrix B we have

$$AB = \begin{pmatrix} A_{11} & \dots & A_{1m} \\ \vdots & \ddots & \vdots \\ A_{n1} & \dots & A_{nm} \end{pmatrix} \begin{pmatrix} B_{11} & \dots & B_{1k} \\ \vdots & \ddots & \vdots \\ B_{m1} & \dots & B_{mk} \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^m A_{1i}B_{i1} & \dots & \sum_{i=1}^m A_{1i}B_{ik} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^m A_{ni}B_{i1} & \dots & \sum_{i=1}^m A_{ni}B_{ik} \end{pmatrix},$$

that is the result is the matrix whose element at row j and column ℓ is

$$(AB)_{j\ell} = \sum_{i=1}^{n} A_{ji}B_{i\ell}.$$

We need matrix multiplication because it is used to calculate activations of the next layer given previous layer values and connection weights. It essentially means that we are applying linear transformation to the input of the network/output of previous layer.

Prerequisite 3 (Transpose)**.** Suppose again that we have an $n \times m$ matrix

$$A = \begin{pmatrix} A_{11} & \dots & A_{1m} \\ \vdots & \ddots & \vdots \\ A_{n1} & \dots & A_{nm} \end{pmatrix}.$$

Its transpose is defined as an $m \times n$ matrix

$$A^T = \begin{pmatrix} A_{11} & \dots & A_{n1} \\ \vdots & \ddots & \vdots \\ A_{1m} & \dots & A_{nm} \end{pmatrix}$$

that we get by switching the rows and columns of A, or equivalently, by reflecting its elements over its main diagonal. Element-wise we have

$$(A^T)_{ij} = A_{ji}.$$

Date: 26. jaanuar 2022. a.

1.2. Calculus

Prerequisite 4 (Derivative)**.** We now recall the notion of derivative of a function $f : \mathbb{R} \to \mathbb{R}$ and introduce some notation. Let us denote the parameter of function f as x, that means, we are given a definition of f in the following form

$$f(x) = \dots.$$

Then the derivative of f with respect to its parameter x is itself a function $\frac{df}{dx} : \mathbb{R} \to \mathbb{R}$ (here we have assumed that f is differentiable), defined as

$$\frac{df}{dx}(a) = \lim_{x \to a} \frac{f(x) - f(a)}{x - a}.$$

Intuitively, it shows us the instantaneous change of the value of function f at the point a. From the derivative we get the information of whether the function is increasing, decreasing, or not changing at the point a; note again that we are talking about instantaneous change, so this information might not be useful for very wobbly functions, but it is useful for sufficiently smooth functions.

Prerequisite 5 (Partial derivative)**.** Now we recall the notion of partial derivatives of a function $g : \mathbb{R}^n \to \mathbb{R}$. Let us denote the parameters of function g as $x_1, \dots, x_n$, that means, we are given a definition of g in the following form

$$g(x_1, \dots, x_n) = \dots.$$

The partial derivative of g with respect to its parameter x_i is itself a function $\frac{\partial g}{\partial x_i} : \mathbb{R}^n \to \mathbb{R}$ (here we have assumed that g is differentiable) defined as

$$\frac{\partial g}{\partial x_i}(a_1, \dots, a_n) = \lim_{x_i \to a_i} \frac{g(a_1, \dots, a_{i-1}, x_i, a_{i+1}, \dots, a_n) - g(a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n)}{x_i - a_i}.$$

Note that if we define a function h as

$$h(x_i) = g(a_1, \dots, a_{i-1}, x_i, a_{i+1}, \dots, a_n)$$

then the derivative of h is partial derivative of g

$$\frac{dh}{dx_i} = \frac{\partial g}{\partial x_i}.$$

Thus partial derivative g is itself a derivative of a function h that we get if we fix all the other coordinates except x_i. Thus all the properties and intuition about derivative holds also for partial derivative, we only look at the change in the direction of one coordinate axis.

Prerequisite 6 (Chain rule)**.** You should also be very comfortable with using chain rule. Suppose we are given functions $f_1, \dots, f_n : \mathbb{R}^m \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$. Let us denote their composition as

$$h(y_1, \dots, y_m) = g(f_1(y_1, \dots, y_m), \dots, f_n(y_1, \dots, y_m)).$$

Then

$$\frac{\partial h}{\partial y_i}(a) = \sum_{j=1}^{n} \frac{\partial g}{\partial x_j}(f_j(a)) \frac{\partial f_j}{\partial y_i}(a).$$

Often in this notation arguments are omitted and it is written as

$$\frac{\partial h}{\partial y_i} = \sum_{j=1}^{n} \frac{\partial g}{\partial x_j} \frac{\partial f_j}{\partial y_i}.$$

Chain rule also works for if we compose more than two functions. Suppose we have functions $F_1, \dots, F_m : \mathbb{R}^k \to \mathbb{R}$

$$h(z_1, \dots, z_k) = g(f_1(F_1(z_1, \dots, z_k), \dots, F_m(z_1, \dots, z_k)), \dots, f_n(F_1(z_1, \dots, z_k), \dots, F_m(z_1, \dots, z_k))).$$

Then

$$\frac{\partial h}{\partial z_i} = \sum_{j=1}^{n} \frac{\partial g}{\partial x_j} \frac{\partial f(F_1(z_1, \dots, z_k), \dots, F_m(z_1, \dots, z_k))}{\partial z_i} = \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial g}{\partial x_j} \frac{\partial f_j}{\partial y_k} \frac{\partial F_k}{\partial z_i}.$$

As you can see, the notation is becoming more and more inconvenient. Thus, we are going to use matrix calculus to alleviate this problem.

Prerequisite 7 (Gradient)**.** Finally, the reason we need to calculate partial derivatives is that we need to calculate the gradient of a function to use gradient descent algorithm. For a function f with parameters $x_1,\dots,x_n$, its gradient at a point a is simply the derivative with respect to $\mathbf{x} = (x_1,\dots,x_n)^T$ (see next section for explanation of the notation)

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}.$$

It shows the direction of fastest increase and $-\frac{\partial f}{\partial \mathbf{x}}$ shows the direction of fastest decrease of the function value at a point. Again, we are looking at instantaneous changes.

2. Derivatives involving vectors/matrices

Notation 1 (Derivative wrt. matrix)**.** There are different layout conventions for defining derivatives involving vectors and matrices so it is important that we agree on one of them. Suppose we are given a function f with parameters $x_1,\dots,x_n$ and a function g with parameters $Y_{11},\dots,Y_{h_y w_y}$. Denote

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \qquad Y = \begin{pmatrix} Y_{11} & \dots & Y_{1w_y} \\ \vdots & \ddots & \vdots \\ Y_{h_y 1} & \dots & Y_{h_y w_y} \end{pmatrix}$$

Then we denote

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}, \qquad \frac{\partial g}{\partial Y} = \begin{pmatrix} \frac{\partial g}{\partial Y_{11}} & \cdots & \frac{\partial g}{\partial Y_{1w_x}} \\ \vdots & \ddots & \vdots \\ \frac{\partial g}{\partial Y_{h_x 1}} & \cdots & \frac{\partial g}{\partial Y_{h_x w_x}} \end{pmatrix}.$$

Thus derivative with respect to column vector is itself a column vector and derivative with respect to matrix is itself a matrix of the same dimensions. Elements of the matrix are the corresponding partial derivatives. We could also define derivatives of the form $\frac{\partial \mathbf{f}}{\partial x}$ and $\frac{\partial G}{\partial y}$ where f is vector valued function and G is matrix valued function but we are not going to use them in this course. But just to mention it, the standard notation in this case is similar to $\frac{\partial f}{\partial \mathbf{x}}$ and $\frac{\partial g}{\partial Y}$, but the resulting matrices are transposed to have compatibility between chain rule and matrix multiplication.

Notation 2 (Derivative of matrix wrt. matrix)**.** Let us generalize this notation and define derivative of a matrix with respect to another matrix. Denote

$$X = \begin{pmatrix} X_{11} & \dots & X_{1w_x} \\ \vdots & \ddots & \vdots \\ X_{h_x 1} & \dots & X_{h_x w_x} \end{pmatrix}, \qquad Y = \begin{pmatrix} Y_{11} & \dots & Y_{1w_y} \\ \vdots & \ddots & \vdots \\ Y_{h_y 1} & \dots & Y_{h_y w_y} \end{pmatrix}$$

Then we define $\frac{\partial Y}{\partial X}$ as an object with shape $(h_y \times w_y) \times (h_x \times w_x)$ that has elements

$$\left(\frac{\partial Y}{\partial X}\right)^{i_1 i_2}_{j_1 j_2} = \frac{\partial Y_{i_1 i_2}}{\partial X_{i_1 i_2}}.$$

Upper indices are for Y and lower indices are for X. Similarly we can use this definition for higher or lower dimensional objects.

Notation 3 (Multiplication)**.** The reason for using upper and lower indices is that in this case we can define an operation similar to matrix multiplication. Given matrices X, Y, Z we have $\frac{\partial X}{\partial Y}$ of shape $(h_x \times w_x) \times (h_y \times w_y)$ and $\frac{\partial Y}{\partial Z}$ of shape $(h_y \times w_y) \times (h_z \times w_z)$. We define the product $\frac{\partial X}{\partial Y}\frac{\partial Y}{\partial Z}$ as object with shape $(w_x \times h_x) \times (h_z \times w_z)$ and elements

$$\left(\frac{\partial X}{\partial Y}\frac{\partial Y}{\partial Z}\right)^{i_1 i_2}_{j_1 j_2} = \sum_{k_1=1}^{h_y}\sum_{k_2=1}^{w_y} \frac{\partial X_{i_2 i_1}}{\partial Y_{k_1 k_2}}\frac{\partial Y_{k_1 k_2}}{\partial Z_{j_1 j_2}}.$$

Notation 4 (Chain rule)**.** Let us now explain why these notations are useful. Suppose Y is a function of X. Suppose further that f is a scalar function. Define $L = f(Y(X))$. Then

$$\frac{\partial L}{\partial X} = \begin{pmatrix} \frac{\partial L}{\partial X_{11}} & \cdots & \frac{\partial L}{\partial X_{1w_x}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial X_{h_x 1}} & \cdots & \frac{\partial L}{\partial X_{h_x w_x}} \end{pmatrix}$$

where

$$\left(\frac{\partial L}{\partial X}\right)_{ij} = \frac{\partial L}{\partial X_{ij}} = \frac{\partial}{\partial X_{ij}} f(Y(X)) = \sum_{k=1}^{h_y}\sum_{\ell=1}^{w_y} \frac{\partial f}{\partial Y_{k\ell}}\frac{\partial Y_{k\ell}}{\partial X_{ij}} = \left(\frac{\partial f}{\partial Y}\frac{\partial Y}{\partial X}\right)_{ij}.$$

Note that the only prior knowledge that we used besides our defined notation is the chain rule we know from calculus. Since on the left and right hand side we have exactly the same indices, the objects themselves must be equal

$$\frac{\partial L}{\partial X} = \frac{\partial f}{\partial Y}\frac{\partial Y}{\partial X}.$$

Similar formula holds if f is not a scalar function, but similarly to X and Y is itself a higher dimensional object. Thus our notation lets us to use the chain rule in matrix form, which is very convenient as it hides a lot of technicalities like keeping track of all the indices.

3. Let's calculate some derivatives

Example 3.1 (deriving useful formulas)**.** Consider now matrices

$$X = \begin{pmatrix} X_{11} & \dots & X_{1m} \\ \vdots & \ddots & \vdots \\ X_{n1} & \dots & X_{nm} \end{pmatrix}, \qquad Y = \begin{pmatrix} Y_{11} & \dots & Y_{1k} \\ \vdots & \ddots & \vdots \\ Y_{m1} & \dots & Y_{mk} \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} b_1 & \dots & b_k \end{pmatrix}, \qquad 1_n = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Denote $Z = XY + 1_n\mathbf{b}$. Then

$$Z = \begin{pmatrix} \sum_{i=1}^m X_{1i}Y_{i1} + b_1 & \dots & \sum_{i=1}^m X_{1i}Y_{ik} + b_k \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^m X_{ni}Y_{i1} + b_1 & \dots & \sum_{i=1}^m X_{ni}Y_{ik} + b_k \end{pmatrix}.$$

Now note that

$$\frac{\partial Z_{i_1 i_2}}{\partial X_{j_1 j_2}} = \frac{\partial}{\partial X_{j_1 j_2}}\left(\sum_{k=1}^m X_{i_1 k}Y_{k i_2} + b_{i_2}\right) = \begin{cases} Y_{j_2 i_2} & i_1 = j_1 \\ 0 & i_1 \neq j_1 \end{cases}$$
$$\frac{\partial Z_{i_1 i_2}}{\partial Y_{j_1 j_2}} = \frac{\partial}{\partial Y_{j_1 j_2}}\left(\sum_{k=1}^m X_{i_1 k}Y_{k i_2} + b_{i_2}\right) = \begin{cases} X_{i_1 j_1} & i_2 = j_2 \\ 0 & i_2 \neq j_2 \end{cases}.$$

Let us apply chain rule. Suppose we have a scalar valued function f and denote $L = f(Z)$. Then by the chain rule

$$\frac{\partial L}{\partial X_{ij}} = \frac{\partial L}{\partial Z}\frac{\partial Z}{\partial X_{ij}} = \sum_{k=1}^{h_z}\sum_{\ell=1}^{w_z}\frac{\partial L}{\partial Z_{k\ell}}\frac{\partial Z_{k\ell}}{\partial X_{ij}} = \sum_{\ell=1}^{w_z}\frac{\partial L}{\partial Z_{i\ell}}\frac{\partial Z_{i\ell}}{\partial X_{ij}} = \sum_{\ell=1}^{w_z}\frac{\partial L}{\partial Z_{i\ell}}Y_{j\ell} = \sum_{\ell=1}^{w_z}\frac{\partial L}{\partial Z_{i\ell}}(Y^T)_{\ell j} = \left(\frac{\partial L}{\partial Z}Y^T\right)_{ij}$$

and similarly

$$\frac{\partial L}{\partial Y_{ij}} = \frac{\partial L}{\partial Z}\frac{\partial Z}{\partial Y_{ij}} = \sum_{k=1}^{h_z}\sum_{\ell=1}^{w_z}\frac{\partial L}{\partial Z_{k\ell}}\frac{\partial Z_{k\ell}}{\partial Y_{ij}} = \sum_{k=1}^{h_z}\frac{\partial L}{\partial Z_{kj}}\frac{\partial Z_{kj}}{\partial Y_{ij}} = \sum_{k=1}^{h_z}\frac{\partial L}{\partial Z_{kj}}X_{ki} = \sum_{k=1}^{h_z}(X^T)_{ik}\frac{\partial L}{\partial Z_{kj}} = \left(X^T\frac{\partial L}{\partial Z}\right)_{ij}$$

Thus we have obtained formulas

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Z}Y^T, \qquad \frac{\partial L}{\partial Y} = X^T\frac{\partial L}{\partial Z}.$$

4. Softmax

4.1. Notation

Let N denote the number of samples in a batch, D denote the number of features in the input data and C denote the number of classes. Classes themselves are denoted by $\{1,\dots,C\}$. Denote input data as X which is $N \times D$ matrix, each row corresponds to features of one training sample

$$X = \begin{pmatrix} X_{11} & \dots & X_{1D} \\ \vdots & \ddots & \vdots \\ X_{N1} & \dots & X_{ND} \end{pmatrix}.$$

Let us organize correct classes into Y which is $N \times C$ matrix

$$Y = \begin{pmatrix} Y_{11} & \dots & Y_{1C} \\ \vdots & \ddots & \vdots \\ Y_{N1} & \dots & Y_{NC} \end{pmatrix}, \qquad Y_{ij} = \begin{cases} 1 & \text{training sample } i \text{ belongs to class } j \\ 0 & \text{training sample } i \text{ does not belong to class } j \end{cases}$$

Denote the weights in as W which is $D \times C$ matrix

$$W = \begin{pmatrix} W_{11} & \dots & W_{1C} \\ \vdots & \ddots & \vdots \\ W_{D1} & \dots & W_{DC} \end{pmatrix}.$$

Define a function (parameter is a matrix Z of dimension $h_z \times w_z$)

$$\operatorname{softmax}(Z) = \begin{pmatrix} \frac{e^{Z_{11}}}{\sum_{k=1}^{w_z} e^{Z_{1k}}} & \cdots & \frac{e^{Z_{1w_z}}}{\sum_{k=1}^{w_z} e^{Z_{1k}}} \\ \vdots & \ddots & \vdots \\ \frac{e^{Z_{h_z1}}}{\sum_{k=1}^{w_z} e^{Z_{h_zk}}} & \cdots & \frac{e^{Z_{h_zw_z}}}{\sum_{k=1}^{w_z} e^{Z_{h_zk}}} \end{pmatrix}.$$

4.2. Feed-forward pass

Denote activations of the output layer

$$Z = XW.$$

This means that input X is pushed through the connections on the layer and the result is Z. Now we apply softmax to softly take the maximum

$$P = \operatorname{softmax}(Z).$$

Finally, the loss is calculated as

$$L = -\frac{1}{N} \sum_{i=1}^{N} Y_{i\cdot} \odot \log P_{i\cdot}$$

where we have used the notation $Y_{i\cdot}$, which means the i-th row of Y and similarly for P. Logarithm is taken element-wise and $\odot$ denotes element-wise multiplication. If we denote the correct classes as a vector $\mathbf{c} = (c_1,\dots,c_N)$, so i-th sample belongs to class c_i, then by the definition of Y we get

$$L = -\frac{1}{N} \sum_{i=1}^{N} \log P_{ic_i}.$$

4.3. Gradient

Let us now calculate the gradient of the loss function L with respect to the weights W. Note that we can write L as

$$L = f(Z) = f(XW)$$

for some function f. Thus using our previously derived formula we get

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Z} \frac{\partial Z}{\partial W} = X^T \frac{\partial L}{\partial Z}.$$

Let us now tackle $\frac{\partial L}{\partial Z}$. Let us do it element-wise

$$\frac{\partial L}{\partial Z_{ij}} = -\frac{1}{N}\sum_{k=1}^{N}\frac{\partial}{\partial Z_{ij}}\log P_{kc_k}.$$

Now depending whether $i=k$ or $j=c_k$ holds, we have different results. If $i\neq k$ then we are looking at different rows of Z and P and we get derivative 0, since P elements depend only on the values of Z that are in the same row as them. In case $i=k$ and $j\neq c_k$ we get

$$\frac{\partial}{\partial Z_{ij}}\log P_{kc_k} = \frac{\frac{\partial}{\partial Z_{ij}}\frac{e^{Z_{kc_k}}}{\sum_{\ell=1}^{C}e^{Z_{k\ell}}}}{P_{kc_k}} = \frac{-\frac{e^{Z_{kc_k}}e^{Z_{ij}}}{\left(\sum_{\ell=1}^{C}e^{Z_{k\ell}}\right)^2}}{P_{kc_k}} = -\frac{P_{kc_k}P_{kj}}{P_{kc_k}} = -P_{kj}$$

and if $j=c_k$ we get

$$\frac{\partial}{\partial Z_{ij}}\log P_{kc_k} = \frac{\frac{e^{Z_{kc_k}}\left(\sum_{\ell=1}^{C}e^{Z_{k\ell}}\right)-e^{Z_{kc_k}}e^{Z_{ij}}}{\left(\sum_{\ell=1}^{C}e^{Z_{k\ell}}\right)^2}}{P_{kc_k}} = 1-P_{kj}.$$

Therefore we get

$$\begin{aligned}\frac{\partial L}{\partial Z_{ij}} &= \begin{cases}\frac{1}{N}P_{ij}, & j\neq c_i\\ \frac{1}{N}(P_{ij}-1), & j=c_i\end{cases}\\ &= \frac{1}{N}(P_{ij}-Y_{ij})\end{aligned}$$

and we have actually shown that

$$\frac{\partial L}{\partial Z} = \frac{1}{N}(P-Y).$$

Therefore

$$\frac{\partial L}{\partial W} = \frac{1}{N}X^T(P-Y).$$

5. Two-layer network

5.1. Notation

Let N denote the number of samples in a batch, D denote the number of features in the input data and C denote the number of classes. Classes themselves are denoted by $\{1,\dots,C\}$. Denote input data as X which is $N\times D$ matrix, each row corresponds to features of one training sample

$$X = \begin{pmatrix}X_{11} & \dots & X_{1D}\\ \vdots & \ddots & \vdots\\ X_{N1} & \dots & X_{ND}\end{pmatrix}.$$

Let us organize the correct classes into Y which is $N\times C$ matrix

$$Y = \begin{pmatrix}Y_{11} & \dots & Y_{1C}\\ \vdots & \ddots & \vdots\\ Y_{N1} & \dots & Y_{NC}\end{pmatrix}, \qquad Y_{ij} = \begin{cases}1 & \text{training sample } i \text{ belongs to class } j\\ 0 & \text{training sample } i \text{ does not belong to class } j\end{cases}$$

Denote the number of hidden layer nodes as M. Denote weights in the first layer as $W^{(1)}$ which is $D\times M$ matrix

$$W^{(1)} = \begin{pmatrix}W^{(1)}_{11} & \dots & W^{(1)}_{1M}\\ \vdots & \ddots & \vdots\\ W^{(1)}_{D1} & \dots & W^{(1)}_{DM}\end{pmatrix},$$

where $W_{ij}^{(1)}$ corresponds to the weight on the connection between input feature i and hidden node j. Similarly define the weight matrix $W^{(2)}$ for the second layer, it is $M \times C$ matrix

$$W^{(2)} = \begin{pmatrix} W_{11}^{(2)} & \dots & W_{1C}^{(2)} \\ \vdots & \ddots & \vdots \\ W_{M1}^{(2)} & \dots & W_{MC}^{(2)} \end{pmatrix},$$

where W_{ij} is the weight of the connection between hidden node i and output node j. Denote bias terms for the first layer as $1 \times M$ vector $\mathbf{b}^{(1)}$ and for the second layer $1 \times C$ matrix $\mathbf{b}^{(2)}$. So bias for hidden node i is $b_i^{(1)}$ and for output node j it is $b_j^{(2)}$. For convenience, we define $N \times 1$ vector 1_N that only contains ones. Now we define two functions

$$\mathrm{ReLU}(Z) = \begin{pmatrix} \max(Z_{11}, 0) & \dots & \max(Z_{1w_z}, 0) \\ \vdots & \ddots & \vdots \\ \max(Z_{h_z 1}, 0) & \dots & \max(Z_{h_z w_z}, 0) \end{pmatrix},$$

$$\mathrm{softmax}(Z) = \begin{pmatrix} \frac{e^{Z_{11}}}{\sum_{k=1}^{w_z} e^{Z_{1k}}} & \cdots & \frac{e^{Z_{1w_z}}}{\sum_{k=1}^{w_z} e^{Z_{1k}}} \\ \vdots & \ddots & \vdots \\ \frac{e^{Z_{h_z 1}}}{\sum_{k=1}^{w_z} e^{Z_{h_z k}}} & \cdots & \frac{e^{Z_{h_z w_z}}}{\sum_{k=1}^{w_z} e^{Z_{h_z k}}} \end{pmatrix}$$

so ReLU is applying $\max(z, 0)$ element-wise to the matrix and softmax is applying e^z to each element and normalizing the rows so they add up to one.

5.2. Feed-forward pass

Denote the activations of first layer as $A^{(1)}$ which is $N \times M$ matrix defined as

$$A^{(1)} = XW^{(1)} + 1_N \mathbf{b}^{(1)}.$$

We get the term $XW^{(1)}$ by pushing input data X through the connections to the first layer and adding $1_N \mathbf{b}^{(1)}$ means that to each row of $XW^{(1)}$ we add the vector $\mathbf{b}^{(1)}$. Note that each row of $XW^{(1)}$ contains activations for a single input sample. Next we apply non-linearity

$$H^{(1)} = \mathrm{ReLU}(A^{(1)}).$$

Similarly for the second layer we get $N \times C$

$$A^{(2)} = H^{(1)} W^{(2)} + 1_N \mathbf{b}^{(2)}.$$

Finally we apply softmax

$$P = \mathrm{softmax}(A^{(2)}).$$

We have completed the feed-forward pass, data has gone from input layer to the output layer. Cross-entropy loss is now calculated as

$$L_{ce} = -\frac{1}{N} \sum_{i=1}^{N} Y_{i\cdot} \odot \log P_{i\cdot}$$

Here we have used notation $Y_{i\cdot}$ which means the i-th row of Y and similarly for P. Logarithm is taken element-wise. If we denote the correct classes as a vector $\mathbf{c} = (c_1, \dots, c_N)$, so i-th sample belongs to class c_i, then by the definition of Y we get

$$L_{ce} = -\frac{1}{N} \sum_{i=1}^{N} \log P_{ic_i}.$$

5.3. Gradient

Now let us calculate the gradient of the loss function L_{CE} with respect to the weights of first layer $W^{(1)}$. Note that we can write L_{ce} as

$$L_{ce} = f(A^{(1)}) = f(XW^{(1)} + 1_N \mathbf{b}^{(1)})$$

for some function f. Thus our previously derived formula (which also holds in case we add the bias term!) gives us

$$\frac{\partial L_{ce}}{\partial W^{(1)}} = \frac{\partial L_{ce}}{\partial A^{(1)}} \frac{\partial A^{(1)}}{\partial W^{(1)}} = X^T \frac{\partial L_{ce}}{\partial A^{(1)}}.$$

Now using chain rule we get

$$\frac{\partial L_{ce}}{\partial W^{(1)}} = X^T \frac{\partial L_{ce}}{\partial H^{(1)}} \frac{\partial H^{(1)}}{\partial A^{(1)}}$$

Note that

$$\left(\frac{\partial L_{ce}}{\partial H^{(1)}} \frac{\partial H^{(1)}}{\partial A^{(1)}} \right)_{k\ell} = \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{\partial L_{ce}}{\partial H_{ij}^{(1)}} \frac{\partial H_{ij}^{(1)}}{\partial A_{k\ell}^{(1)}}$$

where

$$\frac{\partial H_{ij}^{(1)}}{\partial A_{k\ell}^{(1)}} = \begin{cases} 0 & i \neq k \text{ or } j \neq \ell \text{ or } A_{k\ell}^{(1)} \leq 0 \\ 1 & \text{otherwise.} \end{cases}$$

Thus denoting

$$(A^{(1)} > 0) = \begin{pmatrix} a_{11} & \dots & a_{1M} \\ \vdots & \ddots & \vdots \\ a_{N1} & \dots & a_{NM} \end{pmatrix}, \qquad a_{ij} = \begin{cases} 1 & A_{ij} > 0 \\ 0 & A_{ij} \leq 0 \end{cases}$$

we get

$$\frac{\partial L_{ce}}{\partial H^{(1)}} \frac{\partial H^{(1)}}{\partial A^{(1)}} = (A^{(1)} > 0) \odot \frac{\partial L_{ce}}{\partial H^{(1)}}$$

where $\odot$ is element-wise multiplication. Thus we have

$$\frac{\partial L_{ce}}{\partial W^{(1)}} = X^T \left((A^{(1)} > 0) \odot \frac{\partial L_{ce}}{\partial H^{(1)}} \right).$$

Again noting that we can write L_{ce} as

$$L_{ce} = g(A^{(2)}) = g(H^{(1)} W^{(2)} + 1_N \mathbf{b}^{(2)})$$

for some function g. Thus by our derived formula

$$\frac{\partial L_{ce}}{\partial H^{(1)}} = \frac{\partial L_{ce}}{\partial A^{(2)}} \frac{\partial A^{(2)}}{\partial H^{(1)}} = \frac{\partial L_{ce}}{\partial A^{(2)}} (W^{(2)})^T$$

and hence

$$\frac{\partial L_{ce}}{\partial W^{(1)}} = X^T \left((A^{(1)} > 0) \odot \left(\frac{\partial L_{ce}}{\partial A^{(2)}} (W^{(2)})^T \right) \right).$$

Calculating $\frac{\partial L_{ce}}{\partial A^{(2)}}$ is exactly the same as in the softmax gradient section

$$\frac{\partial L_{ce}}{\partial A^{(2)}} = \frac{1}{N}(P - Y).$$

Therefore

$$\begin{aligned} \frac{\partial L_{ce}}{\partial W^{(1)}} &= X^T \left((A^{(1)} > 0) \odot \left(\frac{1}{N}(P - Y) \right) (W^{(2)})^T \right) \\ &= \frac{1}{N} X^T \left((A^{(1)} > 0) \odot \left((P - Y)(W^{(2)})^T \right) \right). \end{aligned}$$