CS231N Assignment 1 - Probability Recap, Logistic Regression & Softmax

1 Probability Recap

1.1 Probability Space

Provides a formal model of a random process or "experiment"

Probability space consists of three elements . . .

- *Sample space* Ω, which is the set of all possible outcomes
- *Event space* $\mathcal{F}$, representing a set of events
  - An event is a set of outcomes in the sample space
- *Probability function* P, assigns to each event in the event space a probability

Take a **single** throw of a six-sided die as an example . . .

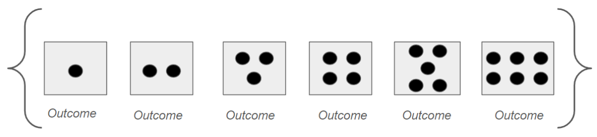


Figure 1: Sample space Ω
All possible outcomes of throwing a six-sided dice **once**

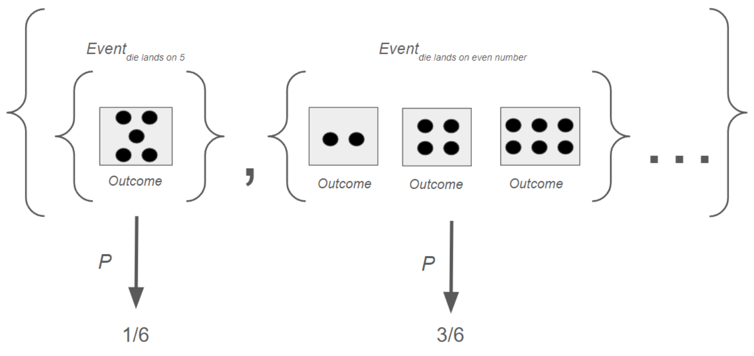
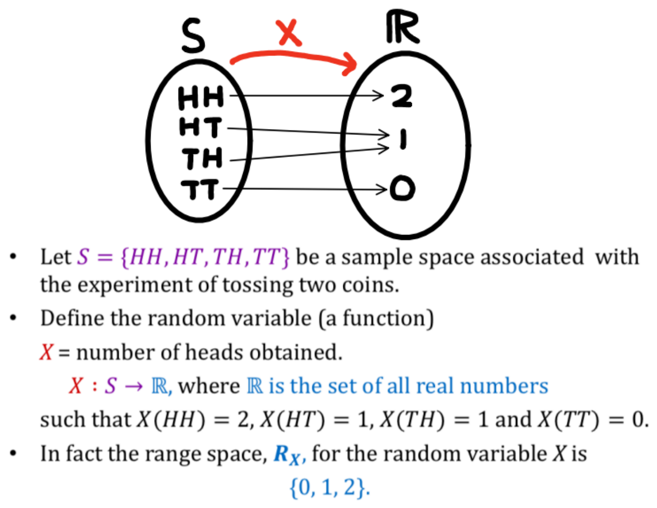


Figure 2: Event space $\mathcal{F}$ could be the set of all subsets
P maps each event as $\frac{\text{numOutcomes}}{6}$

1.2 Random Variables

Random variable $\boldsymbol{X}$ is a **function** $\boldsymbol{X} : \Omega \to \mathbb{R}$, from the sample space Ω to real numbers $\mathbb{R}$. Intuitively, this allows us to calculate probabilities of 'abstract' events that are happening in our sample space Ω



- Let $S = \{HH, HT, TH, TT\}$ be a sample space associated with the experiment of tossing two coins.
- Define the random variable (a function) X = number of heads obtained.
  $X : S \to \mathbb{R}$, where $\mathbb{R}$ is the set of all real numbers
  such that $X(HH) = 2, X(HT) = 1, X(TH) = 1$ and $X(TT) = 0$.
- In fact the range space, $\boldsymbol{R_X}$, for the random variable X is $\{0, 1, 2\}$.

Figure 3: 1D RV

1.2.1 Probability distribution of an RV is not the underlying probability function

We often specify the *probability distribution* (think PMF, PDF) of a random variable directly without explicit mention of the underlying probability function defining the random variable.

In these cases, you can think of the probability function as being the probability distribution of the random variable, and the function defining the random variable to be the identity function.

1.2.2 RV is not it's probability distribution

The probability distribution of an RV is determined by ...

- **Underlying** probability function P, which represents all assumptions about our random phenomenon
- Random Variable $\boldsymbol{X}$ itself, ie. the function which maps sample space outcomes to real numbers in $\mathbb{R}$

Changing either the underlying probability function or the RV itself, will change the probability distribution of the RV.

To illustrate this point, consider the sample space Ω of two rolls of a four-sided die under the following scenarios ...

1. Die is fair and $\boldsymbol{X}$ is the sum of the two rolls
   - $\boldsymbol{X}$ takes on value 2 with probability $\frac{1}{16}$
2. Die is fair and $\boldsymbol{X}$ is the larger of the two rolls
   - $\boldsymbol{X}$ takes on value 2 with probability $\frac{3}{16}$
3. Die is weighted to land on 1 with probability 0.1, 4 with probability 0.4 and $\boldsymbol{X}$ is the sum of the two rolls
   - $\boldsymbol{X}$ takes on value 2 with probability 0.01

- In scenario 1 and 2, the underlying probability function is the same, but the RV is different
- In scenario 1 and 3, the underlying probability function is different, but the RV is the same

1.3 Expectation

1.3.1 Expectation of a random variable

- *Probability mass function (PMF)*
  - Suppose that the underlying probability function P is discrete
  - For any event E containing outcomes s of the sample space Ω, we have that $P(E) = \sum_{s \in E} p(s)$, where p is the PMF
  - Similarly, we can define $\boldsymbol{X}$ to have a PMF p_X
    * $p_X(r) = P(X = r)$; Probability that $\boldsymbol{X}$ takes on some value $r \in \mathbb{R}$
      $= P(s_i \in \Omega : \boldsymbol{X}(s_i) = r)$; Think of it as an event
      $= \sum_{s_i \in \Omega \text{ st. } \boldsymbol{X}(s_i) = r} p(s_i)$; Sum of all probabilities of all sample points in Ω that are assigned the value of r
- Expectation of a random variable $\boldsymbol{X}$
  - $$\mathbb{E}[\boldsymbol{X}] = \sum_{s \in \Omega} \boldsymbol{X}(S) P(\{s\}) = \sum_{s \in \Omega} \boldsymbol{X}(S) p(s)$$
    $$= \sum_r r P(\boldsymbol{X} = r) = \sum_r r p_X(r)$$

1.3.2 Expectation of a function of a random variable

- What does $\mathbb{E}_{\boldsymbol{X} \sim p}[f(\boldsymbol{X})]$ mean?
  - Note that $\boldsymbol{X}$ is a random variable, **not** the *realization* of the random variable
  - $\boldsymbol{X} \sim p$ means that RV $\boldsymbol{X}$ is distributed according to some PMF/PDF $p(\boldsymbol{X})$. Note that p is **not** referring to the underlying probability function P
  - f($\boldsymbol{X}$) is a **function** of the random variable $\boldsymbol{X}$
  - Therefore, we want the expected value of $f(\boldsymbol{X})$ if $\boldsymbol{X}$ is distributed according to p (ie. expectation of the **function** of an RV, closely related to expectation of an RV)
- Probability distribution (PMF,PDF) is not a random variable
  - Suppose we have a RV $\boldsymbol{X}$ with associated PMF p_X
  - $p_X(r)$ is not an RV. It's simply a function assigning RV values to probabilities (there is nothing random happening)
  - However, $p_X(\boldsymbol{X})$ is another RV (since it is a function of the RV $\boldsymbol{X}$)

2 Recap on Logistic Regression

Logistic *regression* is used to solve *classification* problems

2.1 Entropy, Cross Entropy, KL Divergence

- *Entropy of a **random variable*** is the average level of "information" inherent to a variable's possible outcomes
  - Suppose we have discrete RV $\boldsymbol{X}$, which takes in values (outcomes) x in the set Ω and is distributed according to $p_X : \Omega \to [0,1]$ (ie. p_X is the PMF)
  - Define entropy of random variable as

$$\begin{aligned}
H(\boldsymbol{X}) &:= -\sum_{x\in\mathbb{R}} p_X(x)\ log\ p_X(x) && \text{; Expectation of negative log probability}\\
&:= \mathbb{E}_{\boldsymbol{X}\sim p_X}[f(\boldsymbol{X})] && \text{; Formulation from Expectation POV}\\
&:= \mathbb{E}_{\boldsymbol{X}\sim p_X}\big[log(p_X(\boldsymbol{X}))\big] && \text{; Define } f = -log(p_X)\\
&:= \mathbb{E}_{\boldsymbol{X}\sim p_X}\big[\boldsymbol{Y}\big] && \text{; } \boldsymbol{Y} = log(p_X(\boldsymbol{X})),\ \text{our new RV}\\
&= \sum_t \Big\{t * p_Y(t)\Big\} && \text{; } t \in image\big(\boldsymbol{Y}\big)\\
&= t_1 p_Y(t_1) + t_2 p_Y(t_2) + \ldots\\
&= -log(p_X(r_1)) * p_Y(t_1) - log(p_X(r_2)) * p_Y(t_2) - \ldots\\
&= -log(p_X(r_1)) * p_X(r_1) - log(p_X(r_2)) * p_X(r_2) - \ldots\\
&= -\sum_{r_i} log(p_X(r_i)) * p_X(r_i) && \text{; } r_i \in \boldsymbol{X(s_i)}, s_i \in \Omega\ st.\ log(p_X(r_i)) = t_i
\end{aligned}$$

- *KL Divergence* is a measure of difference between probability distributions P and Q. Let's not involve random variables in this discussion
  - Suppose we have discrete **probability distributions** P and Q defined on the same sample space Ω
  - $$\begin{aligned} D_{KL}(P\|Q) &:= \sum_{x\in\Omega} P(x)\ log\ (\frac{P(x)}{Q(x)})\\ &= -\sum_{x\in\Omega} P(x)\ log\ (\frac{Q(x)}{P(x)})\end{aligned}$$
- *Cross Entropy* is another measure of difference between probability distributions P and Q.
  - A single RV cannot have multiple probability distributions (have reference material on this)
  - Suppose we have discrete **probability distributions** P and Q defined on the same sample space Ω
  - Cross Entropy is defined for two different RVs; $\boldsymbol{X}$ with probability distribution P, $\boldsymbol{Y}$ with probability distribution Q
    * $\boldsymbol{X}$ and $\boldsymbol{Y}$ have the same image, but they create different distributions on this image (hence they are different functions). $P(\boldsymbol{X} = x), Q(\boldsymbol{Y} = x)$, where $x \in \mathbb{R}$
    * More research needs to be done on this point
  - $H(P,Q) := -\sum_{x\in\Omega} P(x)log(Q(x))$

2.1.1 Relationship between KL Divergence and Cross Entropy

TBD
Have reference material on entropy of RV vs entropy of a distribution, more research needs to be done on this

2.1.2 Derivation of Binary Cross Entropy from Cross Entropy

TBD
CS3244 derives BCE from a probabilistic perspective. Let's derive BCE from Cross Entropy loss directly.

3 Softmax Classifier

To be more precise, using *Softmax Activation function* **minimizes** the *Cross-Entropy Loss*

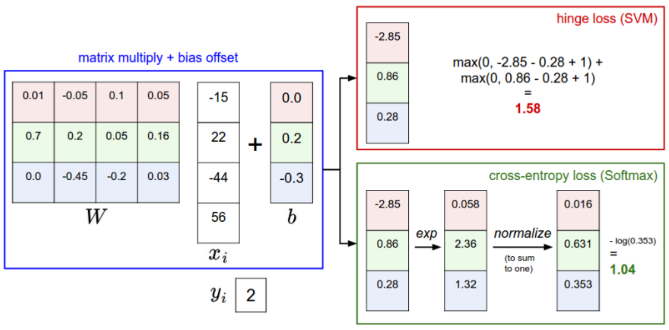


Figure 4: SVM (**minimizing** hinge loss) vs Softmax (**minimizing** cross-entropy loss)
Note that both are utilized to solve **multiclass** classification

See CS3244 notes for more on how Softmax is minimizing cross entropy

3.1 Numerical Stability of Softmax function

- We know that the softmax function for some datapoint x is $\frac{exp(W^T_{(j)}x)}{\sum_i exp(w^T_{(i)})x}$, $\forall j \in \{1, 2, \dots, k\}$
- Recognize that the exponential terms may get very large, leading to numerical instability
- One way to rectify is this . . .

$$\frac{exp(W^T_{(j)}x)}{\sum_i exp(w^T_{(i)})x} = \frac{C * exp(W^T_{(j)}x)}{\sum_i \{C * exp(W^T_{(i)})x\}}$$
$$= \frac{exp(W^T_{(j)}x + log(C))}{\sum_i exp(W^T_{(i)}x + log(C))} \text{ ; Set } log(C) = -max(W^T_{(i)}x) \text{ (ie. Class } i \text{ that has largest } W^T_{(i)}x \text{ score)}$$

3.2 Gradient of Softmax function

3.2.1 Prerequisites

- We have training data $D = \{x_i, t_i\}^C_1$
  - C = number of classes
  - t_i = label of datapoint x_i
- **Per-element** softmax function for some datapoint x is $s_i = p(t = i \mid x) = \frac{exp(W^T_{(i)}x)}{\sum_{n=1}^{C} exp(W^T_{(n)}x)} = \frac{exp(z_i)}{\sum_{n=1}^{C} exp(z_n)}$ for $i \in \{1, 2, ..., C\}$
- Hence, given $W^T = \begin{bmatrix} — & w_1 & — \\ — & w_2 & — \\ & \vdots & \\ — & w_C & — \end{bmatrix} \in \mathbb{R}^{CxD}$; (D = dimension of data)
  - $softmax \left(\begin{bmatrix} W^T_1 x \\ W^T_2 x \\ \vdots \\ W^T_C x \end{bmatrix} \right) = softmax\left(W^T x\right) = \begin{bmatrix} P(t=1|x) \\ P(t=2|x) \\ \vdots \\ P(t=C|x) \end{bmatrix}$; $softmax$: $\mathbb{R}^N \to \mathbb{R}^N$
- Recognize that derivative of Softmax is it's Jacobian matrix, $J_{softmax} = \begin{bmatrix} \frac{\partial s_1}{\partial z_1} & \frac{\partial s_1}{\partial z_2} & \dots & \frac{\partial s_1}{\partial z_C} \\ \frac{\partial s_2}{\partial z_1} & \frac{\partial s_2}{\partial z_2} & \dots & \frac{\partial s_2}{\partial z_C} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial s_C}{\partial z_1} & \frac{\partial s_C}{\partial z_2} & \dots & \frac{\partial s_C}{\partial z_C} \end{bmatrix}$

3.2.2 Gradient Computation

Let's consider what is $\frac{\partial s_i}{\partial z_j} = \frac{\partial}{\partial z_j} \frac{exp(z_i)}{\sum\limits_{n=1}^{C} exp(z_n)}$

(1.1) Let's consider the *logarithmic derivative* (avoids the derivative quotient rule later on)

(1.2) $\frac{\partial}{\partial z_j} log(s_i) = \frac{1}{s_i} \frac{\partial s_i}{\partial z_j}$

(1.3) Rearranging, $\frac{\partial s_i}{\partial z_j} = s_i \cdot \frac{\partial}{\partial z_j} log(s_i)$. Observe that the LHS is the original derivative we are interested in

(2.1) $$log(s_i) = log\left(\frac{exp(z_i)}{\sum\limits_{n=1}^{C} exp(z_n)}\right)$$
$$= log\Big(exp(z_i)\Big) - log\Big(\sum_{n=1}^{C} exp(z_n)\Big)$$
$$= z_i - log\Big(\sum_{n=1}^{C} exp(z_n)\Big)$$

(2.2) Hence, $$\frac{\partial}{\partial z_j} log(s_i) = \frac{\partial}{\partial z_j}\Big\{z_i - log\Big(\sum_{n=1}^{C} exp(z_n)\Big)\Big\}$$
$$= \frac{\partial z_i}{\partial z_j} - \frac{\partial}{\partial z_j}\Big\{log\Big(\sum_{n=1}^{C} exp(z_n)\Big)\Big\}$$

(2.3) Notice that $\frac{\partial z_i}{\partial z_j} = \begin{cases} 1 \text{ if i=j} \\ 0 \text{ if i} \neq \text{j} \end{cases}$

(2.4) Thus, $$\frac{\partial}{\partial z_j} log(s_i) = \mathbb{1}_{i=j} - \frac{\partial}{\partial z_j}\Big\{log\Big(\sum_{n=1}^{C} exp(z_n)\Big)\Big\}$$
$$= \mathbb{1}_{i=j} - \frac{1}{\sum\limits_{n=1}^{C} exp(z_n)} \cdot \left(\frac{\partial}{\partial z_j} \sum_{n=1}^{C} exp(z_n)\right)$$
$$= \mathbb{1}_{i=j} - \frac{1}{\sum\limits_{n=1}^{C} exp(z_n)} \cdot \left(\frac{\partial}{\partial z_j}\Big\{e^{z_1} + e^{z_2} + \cdots + e^{z_C}\Big\}\right)$$
$$= \mathbb{1}_{i=j} - \frac{1}{\sum\limits_{n=1}^{C} exp(z_n)} \cdot e^{z_j}$$
$$= \mathbb{1}_{i=j} - s_j$$

(2.5) Lets multiply by s_i on both sides to make it look like (1.3)

$$s_i \cdot \frac{\partial}{\partial z_j} log(s_i) = s_i \cdot (\mathbb{1}_{i=j} - s_j)$$

$\frac{\partial s_i}{\partial z_j} = s_i \cdot (\mathbb{1}_{i=j} - s_j)$; This gives us the elements of $J_{softmax}$

(3) $$J_{softmax} = \begin{bmatrix} \frac{\partial s_1}{\partial z_1} & \frac{\partial s_1}{\partial z_2} & \cdots & \frac{\partial s_1}{\partial z_C} \\ \frac{\partial s_2}{\partial z_1} & \frac{\partial s_2}{\partial z_2} & \cdots & \frac{\partial s_2}{\partial z_C} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial s_C}{\partial z_1} & \frac{\partial s_C}{\partial z_2} & \cdots & \frac{\partial s_C}{\partial z_C} \end{bmatrix} = \begin{bmatrix} s_1 \cdot (1 - s_1) & -(s_1 \cdot s_2) & \dots & -(s_1 \cdot s_C) \\ -(s_2 \cdot s_1) & s_2 \cdot (1 - s_2) & \dots & -(s_2 \cdot s_C) \\ \vdots & \vdots & \ddots & \vdots \\ -(s_C \cdot s_1) & -(s_C \cdot s_2) & \dots & s_C \cdot (1 - s_C) \end{bmatrix}$$

3.2.3 Backpropagation with Softmax function

(1.1) We know that $\dfrac{\partial s_i}{\partial z_j} = s_i \cdot (\mathbb{1}_{i=j} - s_j)$, where $z_j = W_{(j)}^T x$

(1.2) Cross-Entropy Loss for a particular datapoint n is $L_n(\boldsymbol{y}, \boldsymbol{s}) = -\sum_{i=1}^{C} y_i \cdot log(s_i);$ $\boldsymbol{y}$ is one-hot encoded label

$= -log(s_k);$ k is the label of the particular datapoint n

(1.3)
$$\begin{aligned}
\frac{\partial L_n}{\partial z_j} &= \frac{\partial}{\partial z_j}\Big\{-\sum_{i=1}^{C} y_i \cdot log(s_i)\Big\} \\
&= -\frac{\partial}{\partial z_j}\sum_{i=1}^{C}\Big\{y_i \cdot log(s_i)\Big\} \\
&= -\sum_{i=1}^{C}\Big\{y_i \cdot \frac{\partial}{\partial z_j} log(s_i)\Big\} \\
&= -\sum_{i=1}^{C}\Big\{y_i \cdot \frac{1}{s_i} \cdot \frac{\partial s_i}{\partial z_j}\Big\} \\
&= -\sum_{i=1}^{C}\Big\{y_i \cdot \frac{1}{s_i} \cdot s_i \cdot (\mathbb{1}_{i=j} - s_j)\Big\} \\
&= -\sum_{i=1}^{C}\Big\{y_i \cdot (\mathbb{1}_{i=j} - s_j)\Big\} \\
&= -\sum_{i=1}^{C}\Big\{(y_i \cdot \mathbb{1}_{i=j}) - (y_i \cdot s_j)\Big\} \\
&= \sum_{i=1}^{C}\Big\{(y_i \cdot s_j) - (y_i \cdot \mathbb{1}_{i=j})\Big\} \\
&= \sum_{i=1}^{C}\Big\{(y_i \cdot s_j) - y_j\Big\} \\
&= \sum_{i=1}^{C}(y_i \cdot s_j) - \sum_{i=1}^{C} y_j \\
&= s_j \sum_{i=1}^{C} y_i - y_j \\
&= s_j - y_j \text{ ; Since } \boldsymbol{y} \text{ is a one-hot encoded vector} \\
&= s_j - \mathbb{1}_{n=j}(1) \text{ ; } y_j \text{ (class label of score } z_j\text{) is only '1' if class label of datapoint } n \text{ is the same}
\end{aligned}$$

(1.4)
$$\begin{aligned}
\frac{\partial L_n}{\partial W_{(j)}} &= \frac{\partial L_n}{\partial z_j} \cdot \frac{\partial z_j}{W_{(j)}} \\
&= (s_j - \mathbb{1}_{n=j}(1)) \cdot x
\end{aligned}$$

3.2.4 Gradient Computation - Vectorized

(1.1) Denote our minibatch of data as $\boldsymbol{X} = \begin{bmatrix} —x_1— \\ —x_2— \\ \dots \\ —x_N— \end{bmatrix} \in \mathbb{R}^{NxD}$; N = number of samples, D = dimension of data

(1.2) Denote our weight matrix as $\boldsymbol{W} = \begin{bmatrix} | & | & | & | \\ w_1 & w_2 & \dots & w_C \\ | & | & | & | \end{bmatrix} \in \mathbb{R}^{DxC}$; C = number of classes

(1.3) Denote our softmax function as $\boldsymbol{s} = \begin{bmatrix} s_1 \\ s_2 \\ \dots \\ s_C \end{bmatrix}$, where $s_i = p(t = i \mid x) = \dfrac{exp(W_{(i)}^T x)}{\sum\limits_{n=1}^{C} exp(W_{(n)}^T x)} = \dfrac{exp(z_i)}{\sum\limits_{n=1}^{C} exp(z_n)}$

(1.4) For a single datapoint $x_n \in \mathbb{R}^{1xD}$,
$$\frac{\partial L_n}{\partial \boldsymbol{W}} = \begin{bmatrix} | & | & | & | \\ \frac{\partial L_n}{\partial W_{(1)}} & \frac{\partial L_n}{\partial W_{(2)}} & \dots & \frac{\partial L_n}{\partial W_{(C)}} \\ | & | & | & | \end{bmatrix} \in \mathbb{R}^{DxC}$$
$$= \begin{bmatrix} | & | & | & | \\ (s_1 - \mathbb{1}_{i=1}) \cdot x_n & (s_2 - \mathbb{1}_{i=2}) \cdot x_n & \dots & (s_C - \mathbb{1}_{i=C}) \cdot x_n \\ | & | & | & | \end{bmatrix}$$
$$= \begin{bmatrix} (s_1 - \mathbb{1}_{i=1}) \cdot x_{n_1} & (s_2 - \mathbb{1}_{i=2}) \cdot x_{n_1} & \dots & (s_C - \mathbb{1}_{i=C}) \cdot x_{n_1} \\ (s_1 - \mathbb{1}_{i=1}) \cdot x_{n_2} & (s_2 - \mathbb{1}_{i=2}) \cdot x_{n_2} & \dots & (s_C - \mathbb{1}_{i=C}) \cdot x_{n_2} \\ \vdots & \vdots & \ddots & \vdots \\ (s_1 - \mathbb{1}_{i=1}) \cdot x_{n_D} & (s_2 - \mathbb{1}_{i=2}) \cdot x_{n_D} & \dots & (s_C - \mathbb{1}_{i=C}) \cdot x_{n_D} \end{bmatrix}$$
$$= \begin{bmatrix} x_{n_1} \\ x_{n_2} \\ \dots \\ x_{n_D} \end{bmatrix} \cdot \begin{bmatrix} (s_1 - \mathbb{1}_{i=1}) & (s_2 - \mathbb{1}_{i=2}) & \dots & (s_C - \mathbb{1}_{i=C}) \end{bmatrix}$$
$$= x_n^T \cdot \begin{bmatrix} (s_1 - \mathbb{1}_{i=1}) & (s_2 - \mathbb{1}_{i=2}) & \dots & (s_C - \mathbb{1}_{i=C}) \end{bmatrix}$$

(1.5) For all datapoints $\boldsymbol{X} \in \mathbb{R}^{NxD}$, we have $\dfrac{\partial L}{\partial \boldsymbol{W}} \in \mathbb{R}^{DxC}$; Shape makes sense (think about it)

(1.6)
$$\frac{\partial L}{\partial \boldsymbol{W}} = \frac{1}{N} \sum_i \frac{\partial L_i}{\partial \boldsymbol{W}}$$
$$= \begin{bmatrix} x_{1_1} \\ x_{1_2} \\ \dots \\ x_{1_D} \end{bmatrix} \cdot \begin{bmatrix} (s_1 - \mathbb{1}_{i=1})_1 & (s_2 - \mathbb{1}_{i=2})_2 & \dots & (s_C - \mathbb{1}_{i=C})_N \end{bmatrix} + \begin{bmatrix} x_{2_1} \\ x_{2_2} \\ \dots \\ x_{2_D} \end{bmatrix} \cdot \begin{bmatrix} (s_1 - \mathbb{1}_{i=1})_2 & (s_2 - \mathbb{1}_{i=2})_2 & \dots & (s_C - \mathbb{1}_{i=C})_2 \end{bmatrix} + \dots$$
$$= \begin{bmatrix} x_{1_1}(s_1 - \mathbb{1}_{i=1})_1 + x_{2_1}(s_1 - \mathbb{1}_{i=1})_2 + \dots + x_{N_1}(s_1 - \mathbb{1}_{i=1})_N \\ x_{1_2}(s_1 - \mathbb{1}_{i=1})_1 + x_{2_2}(s_1 - \mathbb{1}_{i=1})_2 + \dots + x_{N_2}(s_1 - \mathbb{1}_{i=1})_N \\ \dots \\ x_{1_D}(s_1 - \mathbb{1}_{i=1})_1 + x_{2_D}(s_1 - \mathbb{1}_{i=1})_2 + \dots + x_{N_D}(s_1 - \mathbb{1}_{i=1})_N \end{bmatrix}$$
$$= \begin{bmatrix} x_{1_1} & x_{2_1} & \dots & x_{N_1} \\ x_{1_2} & x_{2_2} & \dots & x_{N_2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1_D} & x_{2_D} & \dots & x_{N_D} \end{bmatrix} \cdot \begin{bmatrix} (s_1 - \mathbb{1}_{i=1})_1 & (s_2 - \mathbb{1}_{i=2})_1 & \dots & (s_C - \mathbb{1}_{i=C})_1 \\ (s_1 - \mathbb{1}_{i=1})_2 & (s_2 - \mathbb{1}_{i=2})_2 & \dots & (s_C - \mathbb{1}_{i=C})_2 \\ \vdots & \vdots & \ddots & \vdots \\ (s_1 - \mathbb{1}_{i=1})_N & (s_2 - \mathbb{1}_{i=2})_N & \dots & (s_C - \mathbb{1}_{i=C})_N \end{bmatrix}$$
$$= \begin{bmatrix} | & | & | & | \\ x_1 & x_2 & \dots & x_N \\ | & | & | & | \end{bmatrix} \cdot \begin{bmatrix} | & | & | & | \\ (s_1 - \mathbb{1}_{i=1})_{1\dots N} & (s_2 - \mathbb{1}_{i=2})_{1\dots N} & \dots & (s_C - \mathbb{1}_{i=C})_{1\dots N} \\ | & | & | & | \end{bmatrix}$$
$$= \begin{bmatrix} | & | & | & | \\ x_1 & x_2 & \dots & x_N \\ | & | & | & | \end{bmatrix} \cdot \begin{bmatrix} — & (\boldsymbol{s} - \mathbb{1}_{i=1\dots C})_1 & — \\ — & (\boldsymbol{s} - \mathbb{1}_{i=1\dots C})_2 & — \\ — & \dots\dots\dots\dots\dots & — \\ — & (\boldsymbol{s} - \mathbb{1}_{i=1\dots C})_N & — \end{bmatrix}$$
$= \boldsymbol{X}^T \boldsymbol{Q}$; Where $\boldsymbol{Q}$ is defined as above