Due Date : February 16th, 2019

Name : Aditya Joshi
Matricule :

Instructions

- *For all questions, show your work!*
- *Use a document preparation system such as LaTeX.*
- *Submit your answers electronically via the course studium page, and via Gradescope.*

Question 1. Using the following definition of the derivative and the definition of the Heaviside step function :

$$\frac{d}{dx}f(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon} \qquad H(x) = \begin{cases} 1 & \text{if } x > 0 \\ \frac{1}{2} & \text{if } x = 0 \\ 0 & \text{if } x < 0 \end{cases}$$

1. Show that the derivative of the rectified linear unit $g(x) = \max\{0, x\}$, **wherever it exists**, is equal to the Heaviside step function.

   The ReLU is defined as :

   $$g(x) = \text{ReLU}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$

   The derivative is given by :

   $$g(x) = \text{ReLU}'(x) = \begin{cases} 0 & \text{if } x < 0 \\ \text{undefined} & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

   This is equivalent to the Heaviside step function.

2. Give two alternative definitions of $g(x)$ using $H(x)$.

   - $g(x) = x \cdot H(x)$
   - $g(x) = x \cdot (1 - H(-x))$

3. Show that $H(x)$ can be well approximated by the sigmoid function $\sigma(x) = \frac{1}{1+e^{-kx}}$ asymptotically (i.e for large k), where k is a parameter.

   (a) For a **fixed value of** x **where** $x > 0$, as $k \to \infty$, $\sigma(x)$ approaches the value of 1. In other words, $\lim_{k\to\infty} \dfrac{1}{1+e^{-kx}} = 1$

   (b) When $x < 0$, as $k \to \infty$, $\sigma(x)$ approaches the value of 0, because $\lim_{k\to\infty} \dfrac{1}{1+e^{kx}} = 0$.

   (c) When $x = 0$, we get $\sigma(x) = \dfrac{1}{1+e^0} = \dfrac{1}{2}$.

   Therefore we see that we can approximate $H(x)$ by the sgimoid function $\sigma(x)$ asymptotically.

Thus we can approximate the Heaviside step function asymptotically with the sigmoid function.

*4. Although the Heaviside step function is not differentiable, we can define its **distributional derivative**. For a function F, consider the functional $F[\phi] = \int_{\mathbb{R}} H(x)\phi(x)dx$, where ϕ is a smooth function (infinitely differentiable) with compact support ($\phi(x) = 0$ whenever $|x| \geq A$, for some $A > 0$).

Show that whenever F is differentiable, $F'[\phi] = -\int_{\mathbb{R}} F(x)\phi'(x)dx$. Using this formula as a definition in the case of non-differentiable functions, show that $H'[\phi] = \phi(0)$. ($\delta[\phi] \doteq \phi(0)$ is known as the Dirac delta function.)

Answer 1. Write your answer here.

Question 2. Let x be an n-dimentional vector. Recall the softmax function : $S : \boldsymbol{x} \in \mathbb{R}^n \mapsto S(\boldsymbol{x}) \in \mathbb{R}^n$ such that $S(\boldsymbol{x})_i = \frac{e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}}$; the diagonal function : $\text{diag}(\boldsymbol{x})_{ij} = \boldsymbol{x}_i$ if $i = j$ and $\text{diag}(\boldsymbol{x})_{ij} = 0$ if $i \neq j$; and the Kronecker delta function : $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$.

1. Show that the derivative of the softmax function is $\frac{dS(\boldsymbol{x})_i}{d\boldsymbol{x}_j} = S(\boldsymbol{x})_i\,(\delta_{ij} - S(\boldsymbol{x})_j)$.

   There are two cases to be considered.

   (a) $\boldsymbol{i = j}$: Using the quotient rule of derivatives, we have that :

   $$\frac{dS(\boldsymbol{x})_i}{dx_j} = \frac{e^{x_i} \cdot \sum_j e^{x_j} - e^{x_i} \cdot e^{x_j}}{(\sum_j e^{x_j})^2}$$

   (b) $\boldsymbol{i \neq j}$: Notice that in the expression below the $e^{x_i} \cdot \sum_j e^{x_j}$ part is equal to 0 as the derivative of e^{x_i} with respect to e^{x_j} is equal to 0 since $i \neq j$.

   $$\frac{dS(\boldsymbol{x})_i}{dx_j} = \frac{-e^{x_i} \cdot e^{x_j}}{(\sum_j e^{x_j})^2}$$

   We can combine these two expressions by introducing the Kronecker delta function δ_{ij} as follows :

   $$\begin{aligned}
   \frac{dS(\boldsymbol{x})_i}{dx_j} &= \frac{e^{x_i} \cdot \sum_j e^{x_j} \cdot \delta_{ij} - e^{x_i} \cdot e^{x_j}}{(\sum_j e^{x_j})^2} \\
   &= \frac{e^{x_i} \cdot \sum_j e^{x_j} \cdot \delta_{ij}}{(\sum_j e^{x_j})^2} - \frac{e^{x_i} \cdot e^{x_j}}{(\sum_j e^{x_j})^2} \\
   &= \frac{e^{x_i}}{\sum_j e^{x_j}} \cdot \delta_{ij} - \left[\frac{e^{x_i}}{\sum_j e^{x_j}} \cdot \frac{e^{x_j}}{\sum_j e^{x_j}}\right] \\
   &= S(\boldsymbol{x})_i \cdot \delta_{ij} - S(\boldsymbol{x})_i \cdot S(\boldsymbol{x})_j \\
   &= \boxed{S(\boldsymbol{x})_i(\delta_{ij} - S(\boldsymbol{x})_j)}
   \end{aligned}$$

2. Express the Jacobian matrix $\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}}$ using matrix-vector notation. Use $\text{diag}(\cdot)$.

   $$\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}} = \text{diag}\Big(S(\boldsymbol{x})\Big) - S(\boldsymbol{x}) \cdot S(\boldsymbol{x})^\top$$

3. Compute the Jacobian of the sigmoid function $\sigma(\boldsymbol{x}) = 1/(1+e^{-\boldsymbol{x}})$.
   Let us first derive the derivative of the sigmoid with respect to a scalar parameter.

$$\begin{aligned}
\sigma(x) &= \frac{1}{1+e^{-x}} \\
\frac{d\sigma(x)}{dx} &= \frac{d}{dx} \cdot (1+e^{-x})^{-1} \\
&= -1 \cdot (1+e^{-x})^{-2} \cdot (-e^{-x}) \\
&= (1+e^{-x})^{-2} \cdot e^{-x} \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \sigma(x) \cdot (1-\sigma(x))
\end{aligned}$$

   For the sigmoid transformation of the vector, each element of the vector is given by :

$$\sigma(\boldsymbol{x})_i = \frac{1}{1+e^{-(\boldsymbol{x})_i}}$$

   There are two cases as before for the derivative $\dfrac{\partial \sigma(\boldsymbol{x})_i}{(\boldsymbol{x})_j}$

   - $\boldsymbol{i = j}$ **:** $\sigma(\boldsymbol{x})_i \cdot (1 - \sigma(\boldsymbol{x})_j)$
   - $\boldsymbol{i \neq j}$ **:** 0

   Thus, in vector notation, where $\mathbf{1}$ is a vector of the appropriate size,

$$\frac{\partial \sigma(\boldsymbol{x})}{\partial \boldsymbol{x}} = \operatorname{diag}\Big(\sigma(\boldsymbol{x})(\mathbf{1} - \sigma(\boldsymbol{x}))^\top\Big)$$

4. Let $\boldsymbol{y}$ and $\boldsymbol{x}$ be $n-$dimensional vectors related by $\boldsymbol{y} = f(\boldsymbol{x})$, L be an unspecified differentiable loss function. According to the chain rule of calculus, $\nabla_{\boldsymbol{x}} L = (\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}})^\top \nabla_{\boldsymbol{y}} L$, which takes up $\mathcal{O}(n^2)$ computational time in general. Show that if $f(\boldsymbol{x}) = \sigma(\boldsymbol{x})$ or $f(\boldsymbol{x}) = S(\boldsymbol{x})$, the above matrix-vector multiplication can be simplified to a $\mathcal{O}(n)$ operation.

   - $f(\boldsymbol{x}) = \sigma(\boldsymbol{x})$:

$$\begin{aligned}
\nabla_{\boldsymbol{x}} L &= \Bigg(\operatorname{diag}\Big(\sigma(\boldsymbol{x}) \cdot (\mathbf{1} - \sigma(\boldsymbol{x}))\Big)\Bigg)^\top \nabla_{\boldsymbol{y}} L \\
&= \operatorname{diag}\Big(\sigma(\boldsymbol{x})(\mathbf{1} - \sigma(\boldsymbol{x}))\Big) \nabla_{\boldsymbol{y}} L
\end{aligned}$$

     The above product is of a diagonal matrix of size $n \times n$ and a column vector of size n. This is an element wise scaling operation that can be performed in $\mathcal{O}(n)$ time.

- $f(\boldsymbol{x}) = S(\boldsymbol{x})$:

$$\begin{aligned}
\nabla_{\boldsymbol{x}} L &= \left(\operatorname{diag}\left(S(\boldsymbol{x})\right) - S(\boldsymbol{x})S(\boldsymbol{x})^\top\right)^\top \nabla_{\boldsymbol{y}} L \\
&= \left(\operatorname{diag}\left(S(\boldsymbol{x})\right)^\top - (S(\boldsymbol{x})S(\boldsymbol{x})^\top)^\top\right)\nabla_{\boldsymbol{y}} L \\
&= \left(\operatorname{diag}\left(S(\boldsymbol{x})\right) - S(\boldsymbol{x})S(\boldsymbol{x})^\top\right)\nabla_{\boldsymbol{y}} L \\
&= \operatorname{diag}\left(S(\boldsymbol{x})\right)\nabla_{\boldsymbol{y}} L - S(\boldsymbol{x})S(\boldsymbol{x})^\top\nabla_{\boldsymbol{y}} L
\end{aligned}$$

Now the $\operatorname{diag}\left(S(\boldsymbol{x})\right)\nabla_{\boldsymbol{y}} L$ term is an element wise scaling operation of complexity $\mathcal{O}(n)$ (product of a diagonal matrix of size $n \times n$ and a column vector of size n) as seen before for the sigmoid derivation. Secondly, $S(\boldsymbol{x})S(\boldsymbol{x})^\top\nabla_{\boldsymbol{y}} L$ can be simplified to an $\mathcal{O}(n)$ operation by using the property of matrix associativity.

$$\begin{aligned}
&= S(\boldsymbol{x})\left(S(\boldsymbol{x})^\top\nabla_{\boldsymbol{y}} L\right) \\
&= S(\boldsymbol{x})\,\underbrace{\left(S(\boldsymbol{x})^\top\nabla_{\boldsymbol{y}} L\right)}_{\mathcal{O}(n)} \\
&= \underbrace{S(\boldsymbol{x}) \cdot c}_{\mathcal{O}(n)}
\end{aligned}$$

Thus, finally, we have that

$$\nabla_{\boldsymbol{x}} L = \underbrace{\operatorname{diag}\left(S(\boldsymbol{x})\right)\nabla_{\boldsymbol{y}} L}_{\mathcal{O}(n)} - \overbrace{S(\boldsymbol{x})S(\boldsymbol{x})^\top\nabla_{\boldsymbol{y}} L}^{\mathcal{O}(n)}$$

And therefore it is an $\mathcal{O}(n)$ operation.

Question 3. Recall the definition of the softmax function : $S(\boldsymbol{x})_i = e^{\boldsymbol{x}_i}/\sum_j e^{\boldsymbol{x}_j}$.

1. Show that softmax is translation-invariant, that is : $S(\boldsymbol{x}+c) = S(\boldsymbol{x})$, where c is a scalar constant.

   The softmax function is defined as follows :

$$S(\boldsymbol{x})_i = \frac{e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}}$$

   If we scale the input vector $\boldsymbol{x}$ by a scalar c, that is, $\boldsymbol{x}+c$, we have the following result :

$$\begin{aligned} S(\boldsymbol{x}+c)_i &= \frac{e^{\boldsymbol{x}_i+c}}{\sum_j e^{\boldsymbol{x}_j+c}} \\ &= \frac{e^{\boldsymbol{x}_i} \cdot e^c}{\sum_j e^{\boldsymbol{x}_j} \cdot e^c} \\ &= \frac{e^c \cdot e^{\boldsymbol{x}_i}}{e^c \cdot \sum_j e^{\boldsymbol{x}_j}} \\ &= \frac{e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}} \\ S(\boldsymbol{x}+c)_i &= S(\boldsymbol{x})_i \end{aligned}$$

   This is the same for all input elements i in the vector, and thus we have shown that the softmax function is translation-invariant, that is, $S(\boldsymbol{x}+c) = S(\boldsymbol{x})$.

2. Show that softmax is not invariant under scalar multiplication. Let $S_c(\boldsymbol{x}) = S(c\boldsymbol{x})$ where $c \geq 0$. What are the effects of taking c to be 0 and arbitrarily large ?

   After scaling the softmax function we get :

$$S(c \cdot \boldsymbol{x})_i = \frac{e^{c \cdot \boldsymbol{x}_i}}{\sum_j e^{c \cdot \boldsymbol{x}_j}} \neq \frac{e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}}$$

   - $\boldsymbol{c \rightarrow 0}$: We will obtain the uniform distribution, that is, $S(\boldsymbol{x})_i = \dfrac{1}{|\boldsymbol{x}|}$
   - $\boldsymbol{c \rightarrow \infty}$: The vector with the largest value will tend towards having a very large probability. We will shift probability mass from all other points to the ones that have a higher numeric value. That is, if $(\boldsymbol{x})_i$ has the largest value, $S(\boldsymbol{x})_i \rightarrow 1$.

3. Let $\boldsymbol{x}$ be a 2-dimentional vector. One can represent a 2-class categorical probability using softmax $S(\boldsymbol{x})$. Show that $S(\boldsymbol{x})$ can be reparameterized using sigmoid function, i.e. $S(\boldsymbol{x}) = [\sigma(z), 1-\sigma(z)]^\top$ where z is a scalar function of $\boldsymbol{x}$.

   For a two dimensional vector $\boldsymbol{x}$, the softmax function is given by :

$$
\begin{aligned}
S(\boldsymbol{x})_1 &= \frac{e^{\boldsymbol{x}_1}}{e^{\boldsymbol{x}_1} + e^{\boldsymbol{x}_2}} & S(\boldsymbol{x})_2 &= \frac{e^{\boldsymbol{x}_2}}{e^{\boldsymbol{x}_1} + e^{\boldsymbol{x}_2}} \\
S(\boldsymbol{x})_1 &= \frac{e^{\boldsymbol{x}_1}}{e^{\boldsymbol{x}_1} + e^{\boldsymbol{x}_2}} & S(\boldsymbol{x})_2 &= 1 - \frac{e^{\boldsymbol{x}_1}}{e^{\boldsymbol{x}_1} + e^{\boldsymbol{x}_2}} \\
S(\boldsymbol{x})_1 &= \frac{e^{\boldsymbol{x}_1}}{e^{\boldsymbol{x}_1} + e^{\boldsymbol{x}_2}} & S(\boldsymbol{x})_2 &= 1 - S(\boldsymbol{x}_1) \\
S(\boldsymbol{x}_1) &= \frac{1}{1 + \frac{e^{\boldsymbol{x}_2}}{e^{\boldsymbol{x}_1}}} \\
&= \frac{1}{1 + e^{\boldsymbol{x}_2} \cdot e^{-\boldsymbol{x}_1}} \\
&= \frac{1}{1 + e^{\boldsymbol{x}_2 - \boldsymbol{x}_1}} \\
&= \frac{1}{1 + e^{-(\boldsymbol{x}_1 - \boldsymbol{x}_2)}} \\
S(\boldsymbol{x})_1 &= \sigma(\boldsymbol{x}_1 - \boldsymbol{x}_2) & S(\boldsymbol{x})_2 &= 1 - \sigma(\boldsymbol{x}_1 - \boldsymbol{x}_2)
\end{aligned}
$$

   Thus, we can define a scalar function of $\boldsymbol{x}$ as the following $z = (\boldsymbol{x})_1 - (\boldsymbol{x})_2$ and reparameterize the softmax function as $[\sigma(z), 1 - \sigma(z)]^\top$.

4. Let $\boldsymbol{x}$ be a K-dimentional vector ($K \geq 2$). Show that $S(\boldsymbol{x})$ can be represented using $K-1$ parameters, i.e. $S(\boldsymbol{x}) = S([0, y_1, y_2, ..., y_{K-1}]^\top)$ where y_i is a scalar function of $\boldsymbol{x}$ for $i \in \{1, ..., K-1\}$.

For a K-dimensional vector $\boldsymbol{x}$, we have

$$
\begin{aligned}
S(\boldsymbol{x})_1 &= \frac{e^{\boldsymbol{x}_1}}{e^{\boldsymbol{x}_1} + e^{\boldsymbol{x}_2} + \cdots e^{\boldsymbol{x}_k}} & S(\boldsymbol{x})_j &= \frac{e^{\boldsymbol{x}_j}}{e^{\boldsymbol{x}_1} + \cdots e^{\boldsymbol{x}_j} + \cdots + e^{\boldsymbol{x}_k}} \\
&= \frac{1}{1 + e^{\boldsymbol{x}_2 - \boldsymbol{x}_1} + \cdots e^{\boldsymbol{x}_k - \boldsymbol{x}_1}} & & \frac{e^{\boldsymbol{x}_j - \boldsymbol{x}_1}}{1 + \cdots e^{\boldsymbol{x}_j - \boldsymbol{x}_1} + \cdots + e^{\boldsymbol{x}_k - \boldsymbol{x}_1}} \\
&= \frac{e^0}{e^0 + e^{\boldsymbol{x}_2 - \boldsymbol{x}_1} + \cdots e^{\boldsymbol{x}_k - \boldsymbol{x}_1}} & & \frac{e^{\boldsymbol{x}_j - \boldsymbol{x}_1}}{e^0 + \cdots e^{\boldsymbol{x}_j - \boldsymbol{x}_1} + \cdots + e^{\boldsymbol{x}_k - \boldsymbol{x}_1}}
\end{aligned}
$$

We can express this using $K-1$ parameters y_i, for all i where $1 \leq i \leq K-1$, such that $y_i = \boldsymbol{x}_i - \boldsymbol{x}_1$. Thus, we see that $S(\boldsymbol{x}) = S([0, y_1, y_2, ..., y_{K-1}]^\top)$.

Question 4. Consider a 2-layer neural network $y : \mathbb{R}^D \to \mathbb{R}^K$ of the form :

$$y(x, \Theta, \sigma)_k = \sum_{j=1}^{M} \omega_{kj}^{(2)} \sigma \left(\sum_{i=1}^{D} \omega_{ji}^{(1)} x_i + \omega_{j0}^{(1)} \right) + \omega_{k0}^{(2)}$$

for $1 \le k \le K$, with parameters $\Theta = (\omega^{(1)}, \omega^{(2)})$ and logistic sigmoid activation function σ. Show that there exists an equivalent network of the same form, with parameters $\Theta' = (\tilde{\omega}^{(1)}, \tilde{\omega}^{(2)})$ and tanh activation function, such that $y(x, \Theta', \tanh) = y(x, \Theta, \sigma)$ for all $x \in \mathbb{R}^D$, and express Θ' as a function of Θ.

The tanh function is defined as follows :

$$\begin{aligned}
\tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \\
&= \frac{1 - e^{-2x}}{1 + e^{-2x}} \\
&= \frac{2 - (1 + e^{-2x})}{1 + e^{-2x}} \\
&= \frac{2}{1 + e^{-2x}} - \frac{1 + e^{-2x}}{1 + e^{-2x}} \\
&= 2 \cdot \frac{1}{1 + e^{-2x}} - 1 \\
&= 2 \cdot \sigma(2x) - 1
\end{aligned}$$

Therefore, the tanh function is a **re-scaled** version of the sigmoid function. We also have that $\sigma(x) = \frac{1}{2}\left(\tanh\left(\frac{x}{2}\right) + 1 \right)$

Consider the original equation,

$$\begin{aligned}
y(x, \Theta, \sigma)_k &= \sum_{j=1}^{M} \omega_{kj}^{(2)} \sigma \left(\sum_{i=1}^{D} \omega_{ji}^{(1)} x_i + \omega_{j0}^{(1)} \right) + \omega_{k0}^{(2)} \\
&= \sum_{j=1}^{M} \omega_{kj}^{(2)} \cdot \frac{1}{2} \cdot \left(\tanh \left(\frac{1}{2} \left(\sum_{i=1}^{D} \omega_{ji}^{(1)} x_i + \omega_{j0}^{(1)} \right) \right) + 1 \right) + \omega_{k0}^{(2)} \\
&= \sum_{j=1}^{M} \frac{\omega_{kj}^{(2)}}{2} \left(\tanh \left(\frac{1}{2} \left(\sum_{i=1}^{D} \omega_{ji}^{(1)} x_i + \omega_{j0}^{(1)} \right) \right) + 1 \right) + \omega_{k0}^{(2)} \\
&= \sum_{j=1}^{M} \left(\frac{\omega_{kj}^{(2)}}{2} \cdot \tanh \left(\sum_{i=1}^{D} \frac{\omega_{ji}^{(1)} x_i}{2} + \frac{\omega_{j0}^{(1)}}{2} \right) \right) + \sum_{j=1}^{M} \frac{\omega_{kj}^{(2)}}{2} + \omega_{k0}^{(2)} \\
&= \sum_{j=1}^{M} \left(\tilde{\omega}_{kj}^{(2)} \cdot \tanh \left(\sum_{i=1}^{D} \tilde{\omega}_{ji}^{(1)} x_i + \tilde{\omega}_{j0}^{(1)} \right) \right) + \tilde{\omega}_{k0}^{(2)}
\end{aligned}$$

Thus we have defined an equivalent network with the following changes in Θ with Θ' defined as :

$$\textbf{Layer 2} \qquad \tilde{\omega}_{kj}^{(2)} = \frac{\omega_{kj}^{(2)}}{2} \qquad \tilde{\omega}_{k0}^{(2)} = \sum_{j=1}^{M} \frac{\omega_{kj}^{(2)}}{2} + \omega_{k0}^{(2)}$$

$$\textbf{Layer 1} \qquad \tilde{\omega}_{ji}^{(1)} = \frac{\omega_{ji}^{(1)}}{2} \qquad \tilde{\omega}_{j0}^{(1)} = \frac{\omega_{j0}^{(1)}}{2}$$

Question 5. Given $N \in \mathbb{Z}^+$, we want to show that for any $f : \mathbb{R}^n \to \mathbb{R}^m$ and any sample set $\mathcal{S} \subset \mathbb{R}^n$ of size N, there is a set of parameters for a two-layer network such that the output $y(\boldsymbol{x})$ matches $f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{S}$. That is, we want to interpolate f with y on any finite set of samples $\mathcal{S}$.

1. Write the generic form of the function $y : \mathbb{R}^n \to \mathbb{R}^m$ defined by a 2-layer network with $N-1$ hidden units, with linear output and activation function ϕ, in terms of its weights and biases $(\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)})$ and $(\boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)})$.

$$y(\boldsymbol{x}) = \boldsymbol{W}^{(2)}\phi\left(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}$$

2. In what follows, we will restrict $\boldsymbol{W}^{(1)}$ to be $\boldsymbol{W}^{(1)} = [\boldsymbol{w}, \cdots, \boldsymbol{w}]^T$ for some $\boldsymbol{w} \in \mathbb{R}^n$ (so the rows of $\boldsymbol{W}^{(1)}$ are all the same). Show that the interpolation problem on the sample set $\mathcal{S} = \{\boldsymbol{x}^{(1)}, \cdots \boldsymbol{x}^{(N)}\} \subset \mathbb{R}^n$ can be reduced to solving a matrix equation : $\boldsymbol{M}\tilde{\boldsymbol{W}}^{(2)} = \boldsymbol{F}$, where $\tilde{\boldsymbol{W}}^{(2)}$ and $\boldsymbol{F}$ are both $N \times m$, given by

$$\tilde{\boldsymbol{W}}^{(2)} = [\boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)}]^\top \qquad \boldsymbol{F} = [f(\boldsymbol{x}^{(1)}), \cdots, f(\boldsymbol{x}^{(N)})]^\top$$

Express the $N \times N$ matrix $\boldsymbol{M}$ in terms of $\boldsymbol{w}$, $\boldsymbol{b}^{(1)}$, ϕ and $\boldsymbol{x}^{(i)}$.

$$\begin{aligned}
\boldsymbol{F} &= [f(\boldsymbol{x}^{(1)}), \cdots, f(\boldsymbol{x}^{(N)})]^\top \\
&= [(\boldsymbol{W}^{(2)}\phi(\boldsymbol{W}^{(1)}\boldsymbol{x}^{(1)} + \boldsymbol{b}^{(1)}) + \boldsymbol{b}^{(2)}), \cdots, (\boldsymbol{W}^{(2)}\phi(\boldsymbol{W}^{(1)}\boldsymbol{x}^{(N)} + \boldsymbol{b}^{(1)}) + \boldsymbol{b}^{(2)})]^\top \\
&= \begin{bmatrix} (\boldsymbol{W}^{(2)}\phi(\boldsymbol{W}^{(1)}\boldsymbol{x}^{(1)} + \boldsymbol{b}^{(1)}) + \boldsymbol{b}^{(2)})^\top \\ \cdots \\ (\boldsymbol{W}^{(2)}\phi(\boldsymbol{W}^{(1)}\boldsymbol{x}^{(N)} + \boldsymbol{b}^{(1)}) + \boldsymbol{b}^{(2)})^\top \end{bmatrix} \\
&= \begin{bmatrix} (\phi(\boldsymbol{x}^{(1)\top}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)\top})\boldsymbol{W}^{(2)\top} + \boldsymbol{b}^{(2)\top}) \\ \cdots \\ (\phi(\boldsymbol{x}^{(N)\top}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)\top})\boldsymbol{W}^{(2)\top} + \boldsymbol{b}^{(2)\top}) \end{bmatrix} \\
&= \begin{bmatrix} (\phi(\boldsymbol{x}^{(1)\top}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)\top})\boldsymbol{W}^{(2)\top} \\ \cdots \\ (\phi(\boldsymbol{x}^{(N)\top}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)\top})\boldsymbol{W}^{(2)\top} \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}^{(2)\top} \\ \cdots \\ \boldsymbol{b}^{(2)\top} \end{bmatrix} \\
&= \begin{bmatrix} (\phi(\boldsymbol{x}^{(1)\top}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)\top}) \\ \cdots \\ (\phi(\boldsymbol{x}^{(N)\top}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)\top}) \end{bmatrix} \boldsymbol{W}^{(2)\top} + \begin{bmatrix} 1 \\ \cdots \\ 1 \end{bmatrix} \boldsymbol{b}^{(2)\top} \\
&= \begin{bmatrix} (\phi(\boldsymbol{x}^{(1)\top}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)\top}) & 1 \\ \cdots & \\ (\phi(\boldsymbol{x}^{(N)\top}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)\top}) & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{W}^{(2)\top} \\ \boldsymbol{b}^{(2)\top} \end{bmatrix} \\
&= \begin{bmatrix} (\phi(\boldsymbol{x}^{(1)\top}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)\top}) & 1 \\ \cdots & \\ (\phi(\boldsymbol{x}^{(N)\top}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)\top}) & 1 \end{bmatrix} [\boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)}]^\top = \tilde{\boldsymbol{W}}^{(2)}
\end{aligned}$$

$$F = \boldsymbol{M}\tilde{\boldsymbol{W}}^{(2)}$$

*3. **Proof with Relu activation.** Assume $\boldsymbol{x}^{(i)}$ are all distinct. Choose $\boldsymbol{w}$ such that $\boldsymbol{w}^\top \boldsymbol{x}^{(i)}$ are also all distinct (Try to prove the existence of such a $\boldsymbol{w}$, although this is not required for the assignment - See Assignment 0). Set $\boldsymbol{b}_j^{(1)} = -\boldsymbol{w}^\top \boldsymbol{x}^{(j)} + \epsilon$, where $\epsilon > 0$. Find a value of ϵ such that $\boldsymbol{M}$ is triangular with non-zero diagonal elements. Conclude. (Hint : assume an ordering of $\boldsymbol{w}^\top \boldsymbol{x}^{(i)}$.)

$$\begin{aligned}\boldsymbol{b}_j^{(1)} &= -\boldsymbol{w}^\top \boldsymbol{x}^{(j)} + \epsilon \\ \epsilon &= \boldsymbol{b}_j^{(1)} + \boldsymbol{w}^\top \boldsymbol{x}^{(j)}\end{aligned}$$

(a) The diagonal elements are $\phi(\boldsymbol{w}^\top \boldsymbol{x}^{(j)} + \boldsymbol{b}_j^{(1)}) = \phi(\epsilon)$. $\epsilon > 0$, and thus $\phi(\epsilon) > 0$ and the diagonal elements are non-zero.

(b) The non-diagonal elements below the diagonal will be $\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + \boldsymbol{b}_j^{(j)} < 0$. Assuming an ordering of $\boldsymbol{w}^\top \boldsymbol{x}^{(i)}$, $i > j$ and substituting $\boldsymbol{b}_j^{(1)} = -\boldsymbol{w}^\top \boldsymbol{x}^{(j)} + \epsilon$, we have :

$$\begin{aligned}\boldsymbol{w}^\top \boldsymbol{x}^{(i)} - \boldsymbol{w}^\top \boldsymbol{x}^{(j)} + \epsilon &< 0 \\ \boldsymbol{w}^\top \boldsymbol{x}^{(j)} - \boldsymbol{w}^\top \boldsymbol{x}^{(i)} &> \epsilon \\ \boldsymbol{w}^\top (\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(i)}) &> \epsilon\end{aligned}$$

Thus we can choose an epsilon such that $0 < \epsilon < \boldsymbol{w}^\top (\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(i)})$. Since we have an upper triangular matrix $\boldsymbol{M}$, $F = \boldsymbol{M}\tilde{\boldsymbol{W}}^{(2)}$ can be solved efficiently for $\boldsymbol{W}^{(2)}$.

*4. **Proof with sigmoid-like activations**. Assume ϕ is continuous, bounded, $\phi(-\infty) = 0$ and $\phi(0) > 0$. Decompose $\boldsymbol{w}$ as $\boldsymbol{w} = \lambda \boldsymbol{u}$. Set $\boldsymbol{b}_j^{(1)} = -\lambda \boldsymbol{u}^\top \boldsymbol{x}^{(j)}$. Fixing $\boldsymbol{u}$, show that $\lim_{\lambda \to +\infty} \boldsymbol{M}$ is triangular with non-zero diagonal elements. Conclude. (Note that doing so preserves the distinctness of $\boldsymbol{w}^\top \boldsymbol{x}^{(i)}$.)

(a) The diagonal elements are $\phi(\boldsymbol{w}^\top \boldsymbol{x}^{(j)} + \boldsymbol{b}_j^{(1)})$. When we substitute $\boldsymbol{b}_j^{(1)} = -\lambda \boldsymbol{u}^\top \boldsymbol{x}^{(j)}$ and $\boldsymbol{w} = \lambda \boldsymbol{u}$, we get $\phi(\lambda \boldsymbol{u}^\top \boldsymbol{x}^{(j)}) = \phi(\lambda \boldsymbol{u}^\top \boldsymbol{x}^{(j)} - \lambda \boldsymbol{u}^\top \boldsymbol{x}^{(j)}) = \phi(0)$. We know that $\phi(0) > 0$. Thus diagonal elements are non-zero.

(b) The non-diagonal elements are $\phi(\lambda \boldsymbol{u}(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}))$. Assuming an ordering of $\boldsymbol{w}^\top \boldsymbol{x}^{(i)}$, for elements under the diagonal we have that $\boldsymbol{w}^\top (\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(i)}) > 0$. Now substituting the value of $\boldsymbol{w}$, we have that $\lambda \boldsymbol{u}^\top (\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(i)}) < 0$. Thus as $\lambda \to \infty$, these elements will tend to 0. We will then get a triangular matrix with non-zero diagonal elements.

Since we have an upper triangular matrix $\boldsymbol{M}$, $F = \boldsymbol{M}\tilde{\boldsymbol{W}}^{(2)}$ can be solved efficiently for $\boldsymbol{W}^{(2)}$.

Question 6. Compute the *full*, *valid*, and *same* convolution (with kernel flipping) for the following 1D matrices : $\begin{bmatrix}1 & 2 & 3 & 4\end{bmatrix} * \begin{bmatrix}1 & 0 & 2\end{bmatrix}$

- **Full convolution :** We pad the input with 2 0's on each side giving us : $\begin{bmatrix}0 & 0 & 1 & 2 & 3 & 4 & 0 & 0\end{bmatrix} * \begin{bmatrix}2 & 0 & 1\end{bmatrix} = \begin{bmatrix}1 & 2 & 5 & 8 & 6 & 8\end{bmatrix}$
- **Valid convolution :** $\begin{bmatrix}1 & 2 & 3 & 4\end{bmatrix} * \begin{bmatrix}2 & 0 & 1\end{bmatrix} = \begin{bmatrix}5 & 8\end{bmatrix}$
- **Same convolution :** For this convolution, we pad the input with 0's on both sides giving us $\begin{bmatrix}0 & 1 & 2 & 3 & 4 & 0\end{bmatrix} * \begin{bmatrix}2 & 0 & 1\end{bmatrix} = \begin{bmatrix}2 & 5 & 8 & 6\end{bmatrix}$

Question 7. Consider a convolutional neural network. Assume the input is a colorful image of size 256×256 in the RGB representation. The first layer convolves 64 8×8 kernels with the input, using a stride of 2 and no padding. The second layer downsamples the output of the first layer with a 5×5 non-overlapping max pooling. The third layer convolves 128 4×4 kernels with a stride of 1 and a zero-padding of size 1 on each border.

The first layer transforms an $256 \times 256 \times 3$ image to an output of $125 \times 125 \times 64$ by using 64 8×8 kernels.

The next layer performs a 5×5 non-overlapping max pooling operation. This is effectively a stride equal to 5. This produces an output volume of dimensions $25 \times 25 \times 64$.

The final layer has a padding of 1, which effectively changes the dimensions to $27 \times 27 \times 64$. Then we convolve with 128 kernels of dimensions 4×4 giving us an output volume of $24 \times 24 \times 128$.

1. What is the dimensionality (scalar) of the output of the last layer ?
   The dimensionality of the output from the last layer is **73728.**
2. Not including the biases, how many parameters are needed for the last layer ?
   Each kernel has $4 \times 4 \times 64 = 1024$ parameters, and there are 128 kernels, so we get a total of $1024 \times 128 =$ **131072** parameters.

Question 8. Assume we are given data of size $3 \times 64 \times 64$. In what follows, provide the correct configuration of a convolutional neural network layer that satisfies the specified assumption. Answer with the window size of kernel (k), stride (s), padding (p), and dilation (d, with convention $d = 1$ for no dilation). Use square windows only (e.g. same k for both width and height).

Answers are marked in **bold.**

1. The output shape of the first layer is $(64, 32, 32)$.
   (a) Assume $k = 8$ without dilation.

Kernel Size (k)	**Stride (s)**	**Padding (p)**	Dilation (d)
8	**2**	**3**	1

   (b) Assume $d = 7$, and $s = 2$.

Kernel Size (k)	Stride (s)	**Padding (p)**	Dilation (d)
2	2	**3**	7

2. The output shape of the second layer is $(64, 8, 8)$. Assume $p = 0$ and $d = 1$.
   (a) Specify k and s for pooling with non-overlapping window.

Kernel Size ($\boldsymbol{k}$)	**Stride ($\boldsymbol{s}$)**	Padding (p)	Dilation (d)
4	**4**	0	1

   (b) What is output shape if $k = 8$ and $s = 4$ instead ?

   The output dimension is $\frac{32-8}{4} + 1 = 7$ and the output shape is $\mathbf{(64, 7, 7)}$.

3. The output shape of the last layer is $(128, 4, 4)$.
   (a) Assume we are not using padding or dilation.

Kernel Size ($\boldsymbol{k}$)	**Stride ($\boldsymbol{s}$)**	Padding (p)	Dilation (d)
2	**2**	0	1

   (b) Assume $d = 2$, $p = 2$.

Kernel Size ($\boldsymbol{k}$)	**Stride ($\boldsymbol{s}$)**	Padding (p)	Dilation (d)
3	**2**	2	2

   (c) Assume $p = 1$, $d = 1$.

Kernel Size ($\boldsymbol{k}$)	**Stride ($\boldsymbol{s}$)**	Padding (p)	Dilation (d)
1	**3**	1	1