CS543 / ECE 549: Computer Vision, Homework 0

Semester: Spring 2021

Name: Francis Yu (hangy6)

1 Calculus Review

1.1 Shift invariance

$\mathbf{z} - C\mathbf{1}$ shift each element in the vector $\mathbf{z}$ by C, and we have:

$$\begin{aligned} y_i &= \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)} \\ &= \frac{\exp(z_i)/\exp(C)}{\frac{1}{\exp(C)} \sum_{j=1}^{k} \exp(z_j)} \\ &= \frac{\exp(z_i - C)}{\sum_{j=1}^{k} \exp(z_j - C)} \end{aligned}$$

Thus softmax function is shift invariant.

1.2 Derivative

If j = i:

$$\begin{aligned} \frac{\partial y_i}{\partial z_j} &= \frac{\partial y_i}{\partial z_i} \\ &= \frac{\exp(z_i)(\sum_{j=1}^{k} \exp(z_j)) - exp(z_i)^2}{(\sum_{j=1}^{k} \exp(z_j))^2} \\ &= y_i - y_i^2 \end{aligned}$$

else if j $\neq$ i:

$$\begin{aligned} \frac{\partial y_i}{\partial z_j} &= \frac{-exp(z_i)exp(z_j)}{(\sum_{j=1}^{k} \exp(z_j))^2} \\ &= -y_i y_j \end{aligned}$$

1.3 Chain Rule

Based on the question 2, we can vectorize the derivative of y_i with respect to vector $\mathbf{z}$ as:

$$\frac{\partial y_i}{\partial \mathbf{z}} = \mathbf{1}[j = i] - y_i \mathbf{y}$$

where $\mathbf{1}[j = i]$ is an indicator function vector that represents only the i-th element in the vector is 1, otherwise 0. Thus we have:

$$\begin{aligned}\frac{\partial y_i}{\partial \mathbf{x}} &= \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \cdot \frac{\partial y_i}{\partial \mathbf{z}} \\ &= \mathbf{W}(\mathbf{1}[j = i] - y_i \mathbf{y}) \\ &= \mathbf{w}_i - y_i \mathbf{W}\mathbf{y}\end{aligned}$$

Also, we notice that $z_i = \mathbf{w}_i^T \mathbf{x} + u_i$, thus only the j-th element in vector $\mathbf{z}$ is related to $\mathbf{w}_j$ similarly we have:

$$\begin{aligned}\frac{\partial y_i}{\partial \mathbf{w}_j} &= \frac{\partial z_j}{\partial \mathbf{w}_j} \cdot \frac{\partial y_i}{\partial z_j} \\ &= \mathbf{x}^T(\mathbf{1}[j = i] - y_i \mathbf{y}) \\ &= x_i - y_i \mathbf{x}^T \mathbf{y}\end{aligned}$$

And:

$$\begin{aligned}\frac{\partial y_i}{\partial \mathbf{u}} &= \frac{\partial \mathbf{z}}{\partial \mathbf{u}} \cdot \frac{\partial y_i}{\partial \mathbf{z}} \\ &= \mathbf{I}(\mathbf{1}[j = i] - y_i \mathbf{y}) \\ &= \mathbf{1}[j = i] - y_i \mathbf{y}\end{aligned}$$

where $\mathbf{I} \in \mathbb{R}^{k \times k}$ is an identity matrix.

2 Linear Algebra Review

2.1 Matrix Multiplication

$$\mathbf{V}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}-\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}-\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}}\end{bmatrix}$$

$$\mathbf{V}\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}-\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}-\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}}\end{bmatrix}$$

If we project the two vector on a 2D coordinate, we will find that multiply by $\mathbf{V}$ will rotate the vector 135° counterclockwise.

2.2 Matrix Transpose

Since:

$$\mathbf{V}^T = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Thus we have:

$$\mathbf{V}\mathbf{V}^T = \begin{bmatrix} -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \cdot \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Thus $\mathbf{V}^{-1} = \mathbf{V}^T$ Thus we know that:

$$\mathbf{V}\mathbf{V}^T\mathbf{x} = \mathbf{V}^T\mathbf{V}\mathbf{x} \qquad\qquad = \mathbf{x}$$

From question 2 we know that $\mathbf{Vx}$ will rotate vector $\mathbf{x}$ 135° counterclockwise, hence we know that $\mathbf{V}^T\mathbf{x}$ will rotate $\mathbf{x}$ 360° - 135° = 225° counterclockwise.

2.3 Diagonal Matrix

$$\sum \mathbf{V}^T \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} = \begin{bmatrix} \sqrt{8} & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} = \begin{bmatrix} -\sqrt{2} \\ -1 \end{bmatrix}$$

$$\sum \mathbf{V}^T \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \sqrt{8} & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \sqrt{2} \\ -1 \end{bmatrix}$$

$$\sum \mathbf{V}^T \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} = \begin{bmatrix} \sqrt{8} & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} = \begin{bmatrix} \sqrt{2} \\ 1 \end{bmatrix}$$

$$\sum \mathbf{V}^T \begin{bmatrix} 0 \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \sqrt{8} & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} -\sqrt{2} \\ 1 \end{bmatrix}$$

Thus, it is a rectangle with $2\sqrt{2}$ length and 2 width.

2.4 Matrix Multiplication II

Rotate $\mathbf{x}$ 150° counterclockwise.

2.5 Geometric Interpretation

We can use the eigendecomposition to factorize a general matrix as $\mathbf{B} = \mathbf{Q\lambda Q^{-}1}$ where i-th column of $\mathbf{Q}$ is the i-th eigenvector and $\boldsymbol{\lambda}$ is a diagnal matrix with and the trace values are eigenvalues of matrix $\mathbf{B}$. Then the effect of $\mathbf{B}$ to vector $\mathbf{x}$ is to first rotate by $\mathbf{Q}^-1$ and then scale by $\boldsymbol{\lambda}$ and finally again rotate by $\mathbf{Q}$.

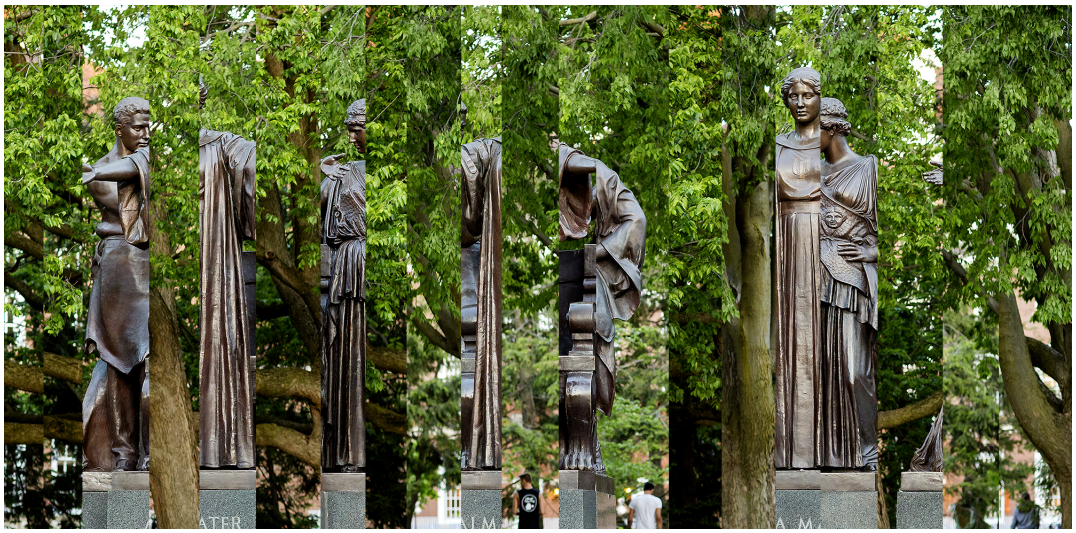

Figure 1: simple alma mater

3 Un-shredding images

3.1 Combine

See **Figure 1, Figure 2.**

3.2 Re-order

See **Figure 3, Figure 4**.
First we define the similarity function as the question given in reorder.py. Then there is an adjacency matrix where each element $edges[i][j]$ represents the similarity function value of shred i being the left neighbor of shred j. Also, we record the most likely neighbor (which has the maximum similarity value as defined) on both left side and right side of a shred. It is unlikely to occur a tie when the shreds are different at their edges. So when two of the shred has the same left or right neighbor, it is certain that one of them is the right or left edge of the whole image. Then we only need to choose the one with smaller similarity value and set it as an edge and set corresponding value as $-\infty$ (e.g. shred i is the right edge, then we set its right similarity value to $-\infty$).
After that, we can start from shred 0 and see which side of the similarity value is large, and we add and update the corresponding side of shreds and the side similarity value until there is no shreds to add and we are done.
The total computation cost is $O(n^2 + n) = O(n^2)$ where n is the number of shreds and the computation cost is dominant by the adjacency matrix computing.

3.3 Align and Re-order

See **Figure 5, Figure 6, Figure 7.** The greedy algorithm I applied is largely derived from the question 3.2. However, the similarity function is changed to compute the dot product

Figure 2: simple larry roberts

normalized by overlapping length after the overlapped part of the image edges are ZNCC processed. By sliding the left shred within range of 20% of its length, different similarity values are presented and only the largest similarity value will be recorded as well as the offset.

Using the similar method in question 3.2, the correct orders of the shreds are derived and then by converting the relative sliding distance to the absolute sliding distance using the left edge shred as the reference, each shred is slided to the best match position as the algorithm guaranteed.

One thing noticeable is that my algorithm failed only on hard_text image. The reason I found is that there are multiple pairs of shreds that is computed to have the same left neighbor or right neighbor and there are even several shreds that have the same left neighbor or right neighbor. For example, in hard_text, shred 03, 19, 23 has the same left neighbor 20

Figure 3: Reordered simple alma mater

because the have very similar pattern in their text position. Since this only contains black and white pixels, unlike other pictures having more colors to differentiate the neighbors, it is more tricky to differentiate among them. This can be a limitation of this algorithm.
The second limitation is that the search space is large. The most accuracy solution would be to slide pixel by pixel, which can be extremely expensive. The current granularity selected will still cause some disalignment.
A third limitation is that the greedy algorithm causes some unnecessary computation cost. $O(n^2)$ is not an optimized time complexity for this problem and it can be troublesome especially when combined with a large search space for alignment.
Another limitation which is an extreme case not shown in this example is that if the shreds are wrapped with a single color frame, then none of the shreds will be able to tell who is its neighbor.
The total computation cost is $O(k(n^2 + n) + n) = O(kn^2)$ where k is the number of slidings in alignment and n is the number of shreds and the computation cost is dominant by the adjacency matrix computing and also the granularity of slidings.

Figure 4: Reordered simple larry roberts

Figure 5: Aligned Alma Mater

Figure 6: Aligned Building

Figure 7: Aligned Texture