Machine Learning HW3

B09902005 資工三盧冠綸

October 29, 2022

programming part

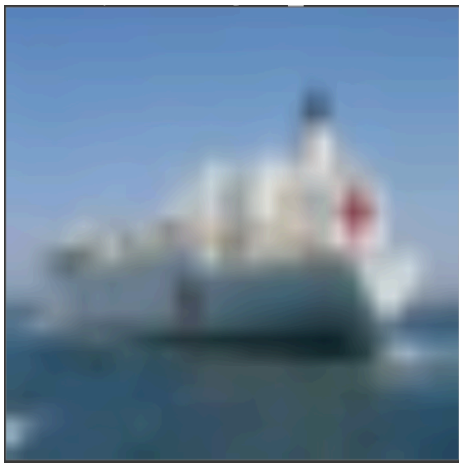
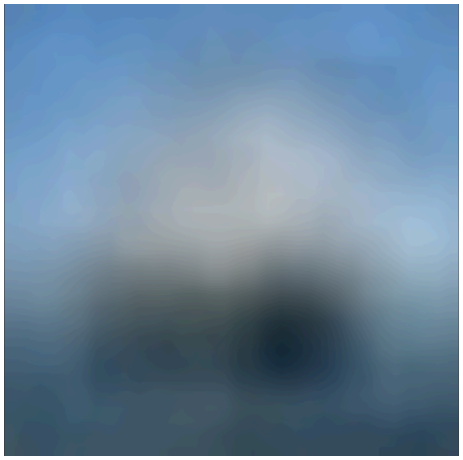
problem 1: (1%) 請附上你在 kaggle 競賽上表現最好的降維以及分群方式，並條列五種不同降維維度的設定對應到的表現 (public / private accuracy)
Below is the chart that lists my public accuracy and private accuracy given different combinations of `LATENT_DIM` and `REDUCED_DIM` when using PCA.

`LATENT_DIM`	`REDUCED_DIM`	public accuracy	private accuracy
32	8	0.77466	0.77622
32	16	0.78600	0.78555
64	8	0.76111	0.76266
64	16	0.78488	0.78622
64	32	0.77822	0.77933
128	16	0.79044	0.79044
128	32	0.79000	0.78911
256	32	0.78711	0.78933

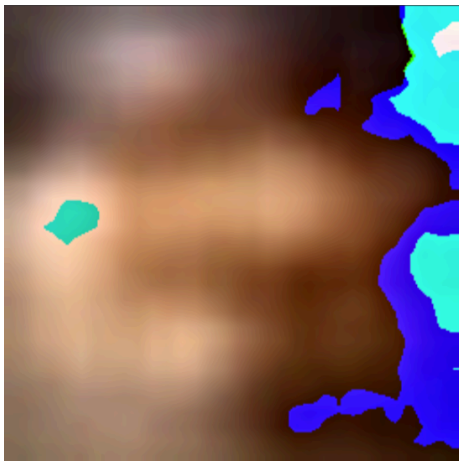
We can see that the model performs better when `LATENT_DIM` is 128 or 256. And, when `RUDUCED_DIM` is 16 or 32, the model will also perform better than when `RUDUCED_DIM` is 8.

problem 2: (1%) 從 train.npy 選出不同類別的 2 張圖，貼上原圖以及你的 autoencoder reconstruct 的圖片。用 Mean Square Error 計算這兩張圖的 reconstruction error, 並說明該 error 與 kaggle score 的關係。
Here, I use PCA with (`LATENT_DIM`, `RUDUCED_DIM`) = (32, 16) to deal with these pictures.

Below is the original picture (left) and the reconstructed picture (right) of a scenery photo (the 2022-th image in `visualization_X.npy`). Their reconstruction error is 0.0041.

Below is the original picture (left) and the reconstructed picture (right) of a photo of human face (the 4044-th image in `visualization_X.npy`). Their reconstruction error is 0.0274.

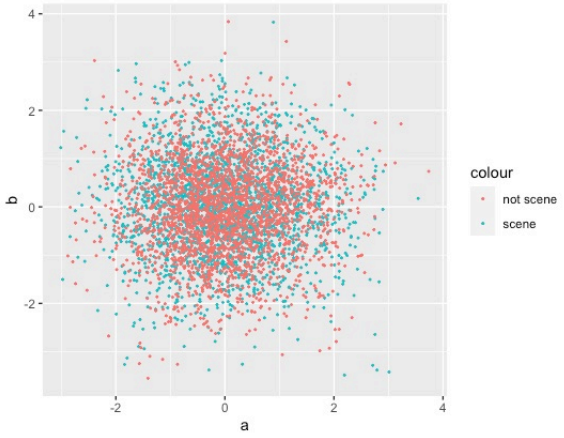



And here, I use three different PCA models, to compare between the reconstruction error and public score.

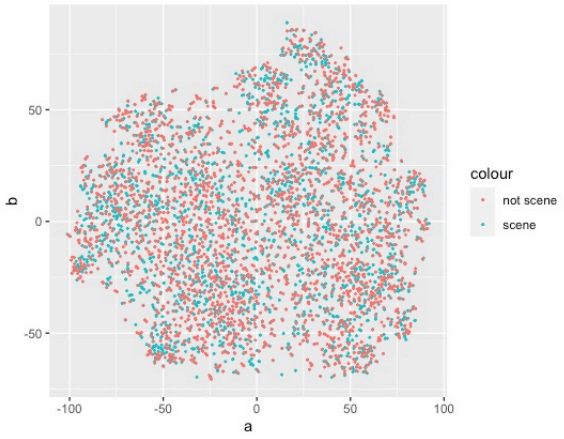
`LATENT_DIM`	`REDUCED_DIM`	E_0	E_1	public score
32	16	0.0131	0.0167	0.78600
64	16	0.0090	0.0109	0.78488
128	16	0.0060	0.0077	0.79044

On the above table, E_0 means the average reconstruction error of the 2500 scenery photos, and E_1 means the average reconstruction error of the 2500 photos that are not scenery photos.
We can see, if `LATENT_DIM` gets larger, then it is easy to know that reconstruction error will decrease. However, we cannot tell whether kaggle score will increase or decrease because we may use too much useless information if `LATENT_DIM` gets too large, which may lead to a bad performance.

problem 3: (2%) 請使用 pca 以及 tsne 兩種方法, 將 visualization.npy 的圖片經過 autoencoder 降維後得到之 latent vector，進一步降維至二維平面並作圖。並說明兩張圖之差異。
Below is the distribution of latent vectors on 2-D plane after PCA.



Below is the distribution of latent vectors on 2-D plane after TSNE.



There is no difference between the distribution of blue dots (scenery photos) and red dots (other pictures) in both pictures. (I think this is a bit abnormal, maybe there's some mistakes in my implementation...).

However, the distribution of all dots after PCA seems more centralized then that after TSNE. I think this is because what PCA does is to directly find a plane and project all points on it, so it is understandable that all the points tend to be closer to the middle of the plane. On the other hand, TSNE checks neighbors of all points and then find a surface and project points on it, thus it is more likely for a point to be at the corner of the 2D plane if the neighbors of the point form a group and the group is far from any other points. So, the distribution of points after PCA may be more centralized than the distribution of points after TSNE.

mathematics part

problem 1: Convolution

First, the total shape after padding is $(B, W+2p_1, H+2p_2, output\_channels)$, and we want to find maximum number of integer n and m, so that it is possible to put nm kernels with size (k_1, k_2) with stride (s_1, s_2) in the range $(W+2p_1, H+2p_2)$. Which is,

$$k_1 + (m-1)s_1 \leq W + 2p_1, \text{ and } k_2 + (n-1)s_2 \leq W + 2p_2$$

So,

$$m \leq \frac{W + 2p_1 - k_1}{s_1} + 1, \text{ and } n \leq \frac{W + 2p_2 - k_2}{s_2} + 1$$

.

So, the shape of the image data will become

$$= (B, \lfloor \frac{W + 2p_1 - k_1}{s_1} \rfloor + 1, \lfloor \frac{H + 2p_2 - k_2}{s_2} \rfloor + 1, output\_channels)$$

problem 2: Batch Normalization

Note that

$$\frac{\partial l}{\partial \hat{x_i}} = \frac{\partial l}{\partial y_i}\frac{\partial y_i}{\partial \hat{x_i}} = \gamma \frac{\partial l}{\partial y_i}$$

$$\frac{\partial l}{\partial \sigma_B^2} = \Sigma_{i=1}^{m}(\frac{\partial l}{\partial \hat{x_i}}\frac{\partial \hat{x_i}}{\partial \sigma_B^2})$$

And, since $\frac{\partial \sigma_B^2}{\partial \mu_B} = \frac{1}{m}\Sigma_{i=1}^m(-2(x_i - \mu_B)) = 0$, so

$$\frac{\partial l}{\partial \mu_B} = \Sigma_{i=1}^m(\frac{\partial l}{\partial \hat{x_i}}\frac{\partial \hat{x_i}}{\partial \mu_B}) + \frac{\partial l}{\partial \sigma_B^2}\frac{\partial \sigma_B^2}{\partial \mu_B} = \Sigma_{i=1}^m(\frac{\partial l}{\partial \hat{x_i}}\frac{\partial \hat{x_i}}{\partial \mu_B})$$

From the above equations, we can see

$$\frac{\partial l}{\partial x_i} = \frac{\partial l}{\partial \hat{x_i}}\frac{\partial \hat{x_i}}{\partial x_i} + \frac{\partial l}{\partial \sigma_B^2}\frac{\partial \sigma_B^2}{\partial x_i} + \frac{\partial l}{\partial \mu_B}\frac{\partial \mu_B}{\partial x_i}$$

$$= \frac{\partial l}{\partial \hat{x_i}}\frac{\partial \hat{x_i}}{\partial x_i} + \Sigma_{j=1}^m(\frac{\partial l}{\partial \hat{x_j}}\frac{\partial \hat{x_j}}{\partial \sigma_B^2})\frac{2(x_i - \mu_B)}{m} + \Sigma_{j=1}^m(\frac{\partial l}{\partial \hat{x_j}}\frac{\partial \hat{x_j}}{\partial \mu_B})\frac{1}{m}$$

$$= \gamma\frac{\partial l}{\partial y_i}\frac{1}{\sqrt{\sigma_B^2 + \epsilon}} + \Sigma_{j=1}^m(\gamma\frac{\partial l}{\partial y_j}\frac{-(x_j - \mu_B)}{2\sqrt{\sigma_B^2 + \epsilon}^3})\frac{2(x_i - \mu_B)}{m} + \Sigma_{j=1}^m(\gamma\frac{\partial l}{\partial y_j}\frac{-1}{\sqrt{\sigma_B^2 + \epsilon}})\frac{1}{m}$$

$$= \frac{\gamma}{m(\sigma_B^2 + \epsilon)^{1.5}}\Sigma_{j=1}^m(\frac{\partial l}{\partial y_i}(\sigma_B^2 + \epsilon) + \frac{\partial l}{\partial y_j}(-x_j + \mu_B)(x_i - \mu_B) + \frac{\partial l}{\partial y_j}(-\sigma_B^2 - \epsilon))$$

And, finally,

$$\frac{\partial l}{\partial \gamma} = \Sigma_{i=1}^m\frac{\partial l}{\partial y_i}\frac{\partial y_i}{\partial \gamma} = \Sigma_{i=1}^m\frac{\partial l}{\partial y_i}\hat{x_i}$$

$$\frac{\partial l}{\partial \beta} = \Sigma_{i=1}^m\frac{\partial l}{\partial y_i}\frac{\partial y_i}{\partial \beta} = \Sigma_{i=1}^m\frac{\partial l}{\partial y_i}$$

problem 3: Softmax Function and Cross Entropy

1. Consider the case $i = j$:

$$\frac{\partial S(z_i)}{\partial z_j} = \frac{\partial S(z_i)}{\partial e^{z_j}}\frac{\partial e^{z_j}}{\partial z_j} = \frac{(\Sigma_{k=1}^N e^{z_k}) - e^{z_i}}{(\Sigma_{k=1}^N e^{z_k})^2}e^{z_j}$$

   Consider the case $i \neq j$:

$$\frac{\partial S(z_i)}{\partial z_j} = \frac{\partial S(z_i)}{\partial e^{z_j}}\frac{\partial e^{z_j}}{\partial z_j} = \frac{-e^{z_i}}{(\Sigma_{k=1}^N e^{z_k})^2}e^{z_j}$$

   Then, we can combine the two formulas together:

$$\frac{\partial S(z_i)}{\partial z_j} = \frac{\delta_{ij}(\Sigma_{k=1}^N e^{z_k}) - e^{z_i}}{(\Sigma_{k=1}^N e^{z_k})^2}e^{z_j}$$

2. We can see that

$$\frac{\partial L}{\partial z_i} = \frac{-\Sigma_j y_j \log(S(z_j))}{\partial z_i}$$

$$= \Sigma_j \frac{-y_j \log(S(z_j))}{\partial(S(z_j))} \frac{\partial(S(z_j))}{\partial z_i}$$

$$= \Sigma_j \frac{-y_j}{S(z_j)} \frac{\delta_{ji}(\Sigma_{k=1}^N e^{z_k}) - e^{z_j}}{(\Sigma_{k=1}^N e^{z_k})^2} e^{z_i}$$

$$= (\Sigma_{j \neq i} \frac{-y_j}{S(z_j)}(-S(z_i)S(z_j))) + \frac{-y_i}{S(z_i)}(S(z_i) - S(z_i)^2)$$

$$= (\Sigma_{j \neq i} y_j S(z_i)) + (-y_i + y_i S(z_i))$$

$$= S(z_i) - y_i$$

$$= \hat{y}_i - y_i$$

problem 4: Constrained Mahalanobis Distance Minimization Problem

1. Since Σ is a symmetric semi-definite matrix, so all eigenvalues of Σ are non-negative. Thus, we can let $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_m)$, where λ_i is an eigenvalue of Σ for all i. Then, $\Sigma = U\Lambda U^T$, where $U = [u_1, u_2, ..., u_m]$, where u_i is an eigenvector of Σ corresponding to λ_i. This way,

$$\Sigma = U\Lambda U^T = \Sigma_{i=1}^m \lambda_i u_i u_i^T$$

Then, we may let $n = 2m$, and for all $1 \leq i \leq m$, we let

$$x_i = \mu + u_i \sqrt{m\lambda_i}$$

$$x_{m+i} = \mu - u_i \sqrt{m\lambda_i}$$

This way,

$$\frac{1}{n}\Sigma_{i=1}^n x_i = \frac{1}{2m}\Sigma_{i=1}^m (x_i + x_{m+i}) = \frac{1}{2m}\Sigma_{i=1}^m 2\mu = \mu$$

$$\frac{1}{n}\Sigma_{i=1}^{n}(x_i-\mu)(x_i-\mu)^T = \frac{1}{2m}\Sigma_{i=1}^{m}((x_i-\mu)(x_i-\mu)^T + (x_{i+m}-\mu)(x_{i+m}-\mu)^T)$$

$$= \frac{1}{2m}\Sigma_{i=1}^{m}2(\mu_i\sqrt{m\lambda_i})(\mu_i\sqrt{m\lambda_i})^T$$

$$= \frac{1}{2m}\Sigma_{i=1}^{m}2m\lambda_i\mu_i\mu_i^T$$

$$= \Sigma$$

2. Since we're able to construct a set of points $x_1, ..., x_n \in R^m$, s.t. $\frac{1}{n}\Sigma_{i=1}^{n}x_ix_i^T = \Sigma$ (This can be proved by the previous subproblem and set $\mu = 0$). Here, we let $X = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$, then $\Sigma = U\Lambda U^T = \frac{1}{n}XX^T$. So,

$$\text{trace}(\Phi^T\Sigma\Phi) = \frac{1}{n}\text{trace}(\Phi^TXX^T\Phi) = \frac{1}{n}\Sigma_{i=1}^{n}||\Phi^Tx_i||^2$$

Since $\Phi^T\Phi = I_k$, so if we let $\Phi = \begin{bmatrix} \Phi_1 & \Phi_2 & \cdots & \Phi_k \end{bmatrix}$, then $||\Phi_i|| = 1$, and $\Phi_i \cdot \Phi_j = 0$ for all $i \neq j$. (Φ is an orthogonal matrix)

Then,

$$\frac{1}{n}\Sigma_{i=1}^{n}||\Phi^Tx_i||^2 = \frac{1}{m}\Sigma_{i=1}^{m}||\Phi^Tx_i||^2 = \Sigma_{i=1}^{m}\lambda_i||\Phi^Tu_i||^2 = \Sigma_{i=1}^{m}\lambda_i\Sigma_{j=1}^{k}(\Phi_j \cdot u_i)^2$$

Here, $||\mu_i|| = 1$ and $\mu_i \cdot \mu_j = 0$ for all $i \neq j$, so we have the following constraints:

$$0 \leq \Sigma_{j=1}^{k}(\Phi_j \cdot u_i)^2 \leq 1, \text{ and } \Sigma_{i=1}^{m}\Sigma_{j=1}^{k}(\Phi_j \cdot u_i)^2 = \Sigma_{j=1}^{k}1 = k$$

So, if we let $\lambda_1 \geq \lambda \geq ... \geq \lambda_m \geq 0$, then to minimize $\text{trace}(\Phi^T\Sigma\Phi)$, we may let $\Sigma_{j=1}^{k}(\Phi_j \cdot u_i)^2 = 0$ if $1 \leq i \leq m-k$, and let $\Sigma_{j=1}^{k}(\Phi_j \cdot u_i)^2 = 1$ if $m-k+1 \leq i \leq m$. This is the solution that minimizes $\text{trace}(\Phi^T\Sigma\Phi)$ among all solutions that follows the two constraints.

And, this solution is indeed feasible. We only have to let $\Phi_j = u_{m+1-j}$. This way,

$$\text{trace}(\Phi^T\Sigma\Phi) = \Sigma_{i=1}^{m}\lambda_i\Sigma_{j=1}^{k}(\Phi_j \cdot u_i)^2 = \Sigma_{i=m-k+1}^{m}\lambda_i$$