Assessed Coursework 1

CID 01252821

Problem 1

Part 1

For a transformation to be linear, it needs to satisfy the following two properties:

Additivity: $T(u+v) = T(u) + T(v)$ for every $u, v \in V$

Homogeneity: $T(\lambda u) = \lambda T(u)$ for every $u \in V, \lambda \in R$

Let's test the transformations against these properties, given $u = (u_1, u_2) \in V$, and $v = (v_1, v_2) \in V$:

a. $\mathbf{T(v) = (v_2, v_2)}$: $T(u+v) = T((u_1+v_1, u_2+v_2)) = (u_2+v_2, u_2+v_2) = (u_2, u_2)+(v_2, v_2) = T(u)+T(v)$, $T(\lambda u) = T((\lambda u_1, \lambda u_2)) = (\lambda u_2, \lambda u_2) = \lambda(u_2, u_2) = \lambda T(u)$. The additivity and homogeneity properties are both satisfied, therefore, we have a linear transformation.

b. $\mathbf{T(v) = (7, v_1)}$: $T(u+v) = T((u_1+v_1, u_2+v_2)) = (7, u_1+v_1)$, $T(u)+T(v) = (7, u_1)+(7, v_1) = (14, u_1+v_1) \neq T(u+v)$. The additivity property is not satisfied. $T(\lambda u) = T((\lambda u_1, \lambda u_2)) = (7, \lambda u_1)$, $\lambda T(u) = \lambda(7, u_1) = (\lambda 7, \lambda u_1)$ which is not equal to $T(\lambda u)$ for all $\lambda \in R$. The homogeneity property is not satisfied, therefore, we have a non linear transformation.

c. $\mathbf{T(v) = (0.2, 4)}$: $T(u+v) = T((u_1+v_1, u_2+v_2)) = (0.2, 4)$, $T(u)+T(v) = (0.2, 4)+(0.2, 4) = (0.4, 8) \neq T(u+v)$. The additivity property is not satisfied. $T(\lambda u) = T((\lambda u_1, \lambda u_2)) = (0.2, 4)$, $\lambda T(u) = \lambda(0.2, 4) = (\lambda 0.2, \lambda 4)$ which is not equal to $T(\lambda u)$ for all $\lambda \in R$. The homogeneity property is not satisfied, therefore, we have a non linear transformation.

d. $\mathbf{T(v) = (3v_2, v_1)}$: $T(u+v) = T((u_1+v_1, u_2+v_2)) = (3(u_2+v_2), u_1+v_1) = (3u_2, u_1)+(3v_2, v_1) = T(u)+T(v)$, $T(\lambda u) = T((\lambda u_1, \lambda u_2)) = (3\lambda u_2, \lambda u_1) = \lambda(3u_2, u_1) = \lambda T(u)$. The additivity and homogeneity properties are both satisfied, therefore, we have a linear transformation.

e. $\mathbf{T(v) = v_1 v_2}$: $T(u+v) = T((u_1+v_1, u_2+v_2)) = (u_1+v_1)(u_2+v_2) = u_1u_2 + u_1v_2 + v_1u_2 + v_1v_2 = T(u)+T(v)+u_1v_2+v_1u_2$ which is not equal to $T(u)+T(v)$ for every $u, v \in V$. $T(\lambda u) = T((\lambda u_1, \lambda u_2)) = \lambda^2 u_1 u_2$, $\lambda T(u) = \lambda u_1 u_2$ which is not equal to $T(\lambda u)$ for all $\lambda \in R$. The homogeneity property is not satisfied, therefore, we have a non linear transformation.

Part 2

a. **Reflection $T(v) = -v$:** Checking if $T(v) = -v$ is linear: $T(u+v) = -(u+v) = -u + (-v) = T(u) + T(v)$, $T(\lambda u) = -(\lambda u) = \lambda(-u) = \lambda T(u)$. The additivity and homogeneity properties are both satisfied, therefore, we have a linear transformation. Let $T'(v) = T(T(v)) = T(-v) = -(-v) = v$, now let's show that $T'(v)$ is also linear: $T'(u+v) = u+v = T'(u) + T'(v)$, $T'(\lambda u) = \lambda u = \lambda T'(u)$. The additivity and homogeneity properties are both satisfied, therefore, $T'(v)$ a linear transformation.

b. **Translation** $T(v) = v + (1,1)$**:** $T(T(v)) = T(v + (1,1)) = (v + (1,1)) + (1,1) = v + (2,2)$. Checking if $T(v) = v + (1,1)$ is linear: $T(u + v) = (u + v) + (1,1)$, $T(u) + T(v) = u + (1,1) + v + (1,1) = u + v + (2,2) \neq T(u + v)$. The additivity property is not satisfied. $T(\lambda u) = \lambda u + (1,1)$, $\lambda T(u) = \lambda(u + (1,1)) = \lambda u + (\lambda, \lambda)$, which is not equal to $T(\lambda u)$ for all $\lambda \in R$. The homogeneity property is not satisfied, therefore, we have a non linear transformation.

c. **90 degree rotation** $T(v) = (-v_2, v_1)$**:** Checking if $T(v) = (-v_2, v_1)$ is linear: $T(u + v) = T((u_1 + v_1, u_2 + v_2)) = (-(u_2 + v_2), u_1 + v_1) = (-u_2 + (-v_2), u_1 + v_1) = (-u_2, u_1) + (-v_2, v_1) = T(u) + T(v)$, $T(\lambda u) = T((\lambda u_1, \lambda u_2)) = (-\lambda u_2, \lambda u_1) = \lambda(-u_2, u_1) = \lambda T(u)$. The additivity and homogeneity properties are both satisfied, therefore, $T(v)$ a linear transformation. Let $T'(v) = T(T(v)) = T((-v_2, v_1)) = (-v_1, -v_2)$, now let's show that $T'(v)$ is also linear: $T'(u + v) = T'((u_1 + v_1, u_2 + v_2)) = (-(u_1 + v_1), -(u_2 + v_2)) = (-u_1 + (-v_1), -u_2 + (-v_2)) = (-u_1, -u_2) + (-v_1, -v_2) = T'(u) + T'(v)$, $T'(\lambda u) = T'((\lambda u_1, \lambda u_2)) = (-\lambda u_1, -\lambda u_2) = \lambda(-u_1, -u_2) = \lambda T'(u)$. The additivity and homogeneity properties are both satisfied, therefore, $T'(v)$ is a linear transformation.

d. **Projection** $T(v) = \frac{1}{2}(v_1 + v_2, v_1 + v_2)$**:** Checking if $T(v) = \frac{1}{2}(v_1 + v_2, v_1 + v_2)$ is linear: $T(u + v) = T((u_1 + v_1, u_2 + v_2)) = \frac{1}{2}(u_1 + v_1 + u_2 + v_2, u_1 + v_1 + u_2 + v_2) = \frac{1}{2}(u_1 + u_2, u_1 + u_2) + \frac{1}{2}(v_1 + v_2, v_1 + v_2) = T(u) + T(v)$. $T(\lambda u) = T((\lambda u_1, \lambda u_2)) = \frac{1}{2}(\lambda u_1 + \lambda u_2, \lambda u_1 + \lambda u_2) = \lambda \frac{1}{2}(u_1 + u_2, u_1 + u_2) = \lambda T(u)$. The additivity and homogeneity properties are both satisfied, therefore, $T(v)$ is a linear transformation. Let $T'(v) = T'((\frac{v_1+v_2}{2}, \frac{v_1+v_2}{2})) = (\frac{v_1+v_2}{2}, \frac{v_1+v_2}{2}) = \frac{1}{2}(v_1 + v_2, v_1 + v_2) = T(v)$, which we already prooved is a linear transformation.

Part 3

We can determine the coordinates (a, b) of $(0, 1)$ by the following equation:

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}.$$

Problem 2

Part 1

Forward Pass:

Layer 1: $x_1 = 12$, $x_2 = 1$, $x_3 = 4$, $x_4 = 3.2$

Layer 2: $y_1 = x_1 + x_2 = 13$, $y_2 = x_3 + x_4 = 7.2$

Output: $z = y_1 y_2 = (x_1 + x_2)(x_3 + x_4) = x_1 x_3 + x_1 x_4 + x_2 x_3 + x_2 x_4 = 93.6$.

Backpropagation:

Second layer:

$$\frac{\partial z}{\partial y_1} = y_2 = 7.2, \ \frac{\partial z}{\partial y_2} = y_1 = 13$$

First layer:

$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x_1} = 7.2 \times 1 = 7.2, \ \frac{\partial z}{\partial x_2} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x_2} = 7.2 \times 1 = 7.2$$

$$\frac{\partial z}{\partial x_3} = \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x_3} = 13 \times 1 = 13, \ \frac{\partial z}{\partial x_4} = \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x_4} = 13 \times 1 = 13$$

Part 2

In the absence of any specified stride or padding the dimension of the final matrix after convolution is $(M-m+1)\times(N-n+1)$. In the case of having stride `s` and padding `p` the final matris is $\left(\frac{M-m+2p}{s}+1\right)\times\left(\frac{N-n+2p}{s}+1\right)$.

Part 3

The data used is the one provided in the problem statement https://www.kaggle.com/ chetankv/dogs-cats-images and the aim is to apply a CNN model to classify the images. I have adapted the CNN code given in the lecture notes to the current use case.

First, the data is loaded by converting the images to tnesors, resizing them to 28 by 28 pixels, and normalizing them. Then, train and test loders are created with batch size of 100. A CNN with two convolutional layers is defined, aiming to classify pictures in two classes: cats and dogs. The first layer accepts 3 channels because the images are RGB, the output channels are set to 16, the kernel is of size 5, the stride is 1, and the padding with zeros is 2. The stride along with the padding are set such that the size of the images won't change, it will remain 28. Then the outputs are normlised, ReLu is applied making the negative values 0 and keeping the positive ones the same, and finally pooling with kernel size 2 and stride 2 is applied which picks tha max value. The max pooling will result in reducing the size of the images in half. The second layer accepts as input channels 16 becasue this was the number of channels from the first layer, now the output channels are set to 96, and th kernel size, padding, and stride are kept the same. As for the first layer there is normalizing, ReLu and max pooling applied. In the end the output is transformed to one vector. We also define a forward pass, containing the ouput of the layer consisting of 16 batches of 100 images but now with half the pixels, or 14. Similarly, the output of the second layer consists of 96 batches of 100 images, and 7 by 7 pixels. In `Figure 1` we can see the average loss for each epoch, obtained after training the model with 5 epochs, and learning rate 0.001. We can observe that the average loos is decreasing for every next epoch.

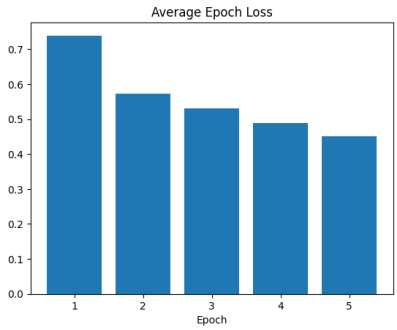


Figure 1: Average loss for each epoch from training the model.

Problem 3

Part 1

Let $h(x,y)=g(x)g(y)=(2\pi)^{-1}s^{-2}\exp\{-\frac{x^2+y^2}{2s^2}\}$, then

$$\frac{\partial h}{\partial x}=-\frac{1}{2\pi s^4}x\exp\{-\frac{x^2+y^2}{2s^2}\}$$

$$\frac{\partial^2 h}{\partial x^2}=-\frac{1}{2\pi s^4}\left[\exp\{-\frac{x^2+y^2}{2s^2}\}-\frac{x^2}{s^2}\exp\{-\frac{x^2+y^2}{2s^2}\}\right]=-\frac{1}{2\pi s^4}\exp\{-\frac{x^2+y^2}{2s^2}\}\left[1-\frac{x^2}{s^2}\right].$$

Analogucally,

$$\frac{\partial^2 h}{\partial y^2} = -\frac{1}{2\pi s^4}\exp\{-\frac{x^2+y^2}{2s^2}\}\left[1-\frac{y^2}{s^2}\right],$$

therefore

$$\nabla^2 = \frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2} = -\frac{1}{\pi s^4}\exp\{-\frac{x^2+y^2}{2s^2}\}\left[1-\frac{x^2+y^2}{2s^2}\right],$$

and

$$H(x,y) = -\nabla^2\{g(x)g(y)\} = \frac{1}{\pi s^4}\exp\{-\frac{x^2+y^2}{2s^2}\}\left[1-\frac{x^2+y^2}{2s^2}\right]$$

For $s=1$:

$$H(x,y) = -\nabla^2\{g(x)g(y)\} = \frac{1}{\pi}\exp\{-\frac{x^2+y^2}{2}\}\left[1-\frac{x^2+y^2}{2}\right].$$

The 5×5 filter matrix with calculated values in Python is:

Results from code:

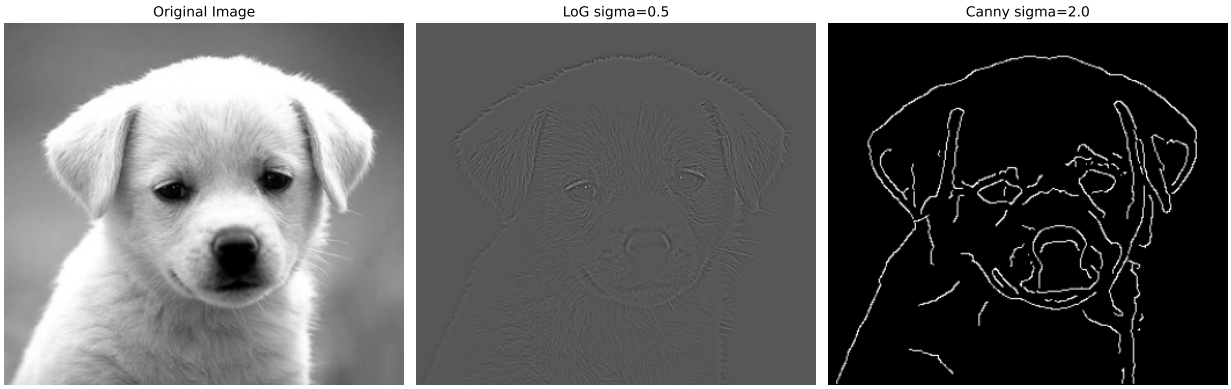
$$H := \begin{bmatrix} H(-2,-2) & H(-1,-2) & H(0,-2) & H(1,-2) & H(2,-2) \\ H(-2,-1) & H(-1,-1) & H(0,-1) & H(1,-1) & H(2,-1) \\ H(-2,0) & H(-1,0) & H(0,0) & H(1,0) & H(2,0) \\ H(-2,1) & H(-1,1) & H(0,1) & H(1,1) & H(2,1) \\ H(-2,2) & H(-1,2) & H(0,2) & H(1,2) & H(2,2) \end{bmatrix} = \begin{bmatrix} -0.017 & -0.039 & -0.043 & -0.039 & -0.017 \\ -0.039 & 0. & 0.097 & 0. & -0.039 \\ -0.043 & 0.097 & 0.318 & 0.097 & -0.043 \\ -0.039 & 0. & 0.097 & 0. & -0.039 \\ -0.017 & -0.039 & -0.043 & -0.039 & -0.017 \end{bmatrix}$$

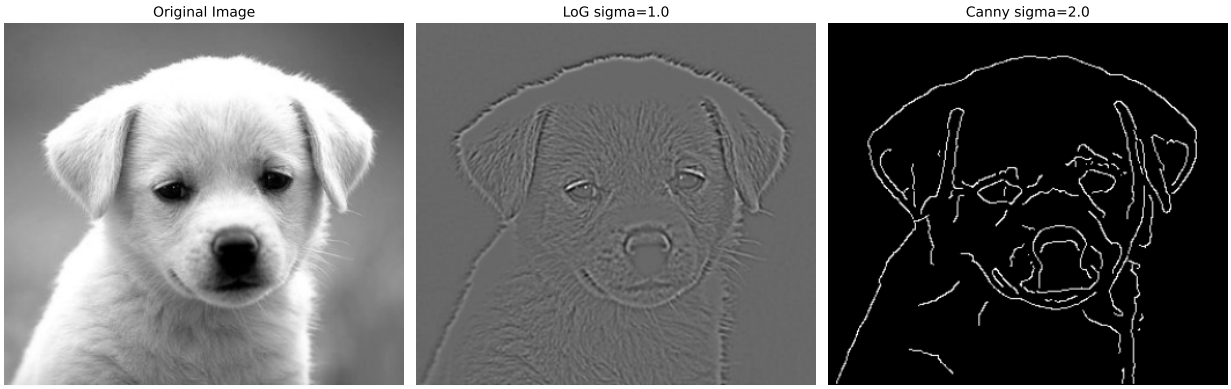
Part 2

$$S_x(r,c) = 16I(r,c) -$$
$$-2I(r+1,c) - 2I(r,c+1) - 2I(r-1,c) - 2I(r,c-1) -$$
$$-I(r+1,c+1) - I(r-1,c-1) - I(r+1,c-1) - I(r-1,c+1) - I(r+2,c) - I(r,c+2) - I(r-2,c) - I(r,c-2)$$

Part 3

Here are shown the results from the LoG implementation in Python using `skimage`. The code was executed for different values of sigma, which indirectly changes the effective mask size.





The first two results point to the observation that for the LoG filter increasing sigma leads to captruing more deatils from the image. We can see sharper (whiter) contrours as well as more details, for example the fur is more distinct. The LoG filter produces a lot more details, whereas the Canny filter manages to capture the main edges, and no details

Decreasing sigma in the Canny filter leads to captruing more details, but again on a contour level, no distinct and clear fur can be observed as in the result from the LoG filter.

Finally, increasing even more the sigma in the LoG filter produces a not so sharp, rather blurry results, meaning the edges are not very distinct.