CSE 493G1/599G1: Deep Learning

Solutions for Section 3: Convolutions & Vectorization

Thanks for attending section, we hope you found it helpful.

0. Reference Material

Equations for Convolutions

Assuming the following variables, which imply that the input image has size (W, H, C),

- W is the width of the input image
- H is the height of the input image
- C is the number of channels in the input image
- F is the receptive field size (i.e., the height and width of the conv field)
- S is the stride with which the convolution is applied
- P is the padding
- K is the depth of the conv layer (i.e., the number of filters applied)

The output size will be,

$$\left(\frac{W - F + 2P}{S} + 1,\ \frac{H - F + 2P}{S} + 1,\ K\right)$$

The conv layer will have $K\left(F^2C + 1\right)$ trainable parameters.

Manhattan Distance

L1 distance, also known as taxicab distance or Manhattan distance, is a measure of distance between two different points are in a n-dimensional coordinate space. If we think of two images with 3072 pixel values each as points in $\mathbb{R}^{3072}$, then their L1 distance tells us how "similar" the two images are.

Formally, if $I_1 = \left(I_1^{(1)}, I_1^{(2)}, \ldots, I_I^{(p)}\right)$ and $I_2 = \left(I_2^{(1)}, I_2^{(2)}, \ldots, I_2^{(p)}\right)$ are two images with p pixel values, then their L1 distance is given by,

$$\sum_{i=1}^{p} \left|I_1^{(i)} - I_2^{(i)}\right|$$

L1 distance has a variety of applications in image processing and computer vision. For instance, in lecture we introduced L1 distance in the context of k nearest neighbors classifiers.

Note that the nicknames come from Manhattan's grid-like street structure, which forces taxicabs to travel along the x-axis or y-axis of the grid (as the L1 distance calculation does) instead of taking diagonals (which is what L2 does).

1. As Convoluting As Possible

(a) What's the formula for determining a conv layer's output size? Assume that the receptive field is a square. Define all the variables you use.

Solution:

For a (W, H, C) input, the output of a conv layer with a (F, F, C) receptive field applied with padding P, stride S, and depth K is given by:

$$\left(\frac{W - F + 2P}{S} + 1, \frac{H - F + 2P}{S} + 1, K\right)$$

(b) Consider a conv layer that takes a $32 \times 32 \times 3$ input and applies a $5 \times 5 \times 3$ filter with no padding. Compute the output sizes if we use a stride of 1, 2, and 3. Then, compute the output size if we use a stride of 2 and 3 with a padding of 3.

Hint: certain strides may result in an invalid configuration for this conv layer.

Solution:

First, identify that $W = 32$, $H = 32$, $C = 3$, $F = 5$. Since $W = H$, we will only perform one of these computations.

Now, let's compute the output sizes for each of the configurations.

- $P = 0$ and $S = 1$: output size $(28, 28, 1)$ since

$$\frac{W - F + 2P}{S} + 1 = \frac{32 - 5 + 2 \cdot 0}{1} + 1 = 27 + 1 = 28$$

- $P = 0$ and $S = 2$: invalid configuration because

$$\frac{W - F + 2P}{S} + 1 = \frac{32 - 5 + 2 \cdot 0}{2} + 1 = 13.5 + 1 = 14.5 \notin \mathbb{Z}^+$$

- $P = 0$ and $S = 3$: output size $(10, 10, 1)$ since

$$\frac{W - F + 2P}{S} + 1 = \frac{32 - 5 + 2 \cdot 0}{3} + 1 = 9 + 1 = 10$$

- $P = 3$ and $S = 2$: invalid configuration because

$$\frac{W - F + 2P}{S} + 1 = \frac{32 - 5 + 2 \cdot 3}{2} + 1 = 16.5 + 1 = 17.5 \notin \mathbb{Z}^+$$

- $P = 3$ and $S = 3$: output size $(12, 12, 1)$ since

$$\frac{W - F + 2P}{S} + 1 = \frac{32 - 5 + 2 \cdot 3}{3} = 11 + 1 = 12$$

Note that, no matter what padding we apply, we cannot make a stride of 2 work.

Note: People will often leave out the channels dimension when writing out a filter (i.e., they might refer to a $5 \times 5 \times 3$ filter as a 5×5 filter). We will now adopt this shorthand as well.

(c) Consider the first conv layer of AlexNet, which takes an input of size $227 \times 227 \times 3$ and applies 96 separate 11×11 convolutional filters with stride of 4 and no padding. People will sometimes write this as applying one 11×11 filter with depth 96; it means the same thing. What are our output dimensions?

Solution:

Our outut will have size $(55, 55, 96)$ since we applied 96 filters and

$$\frac{H - F}{S} + 1 = \frac{W - F}{S} + 1 = \frac{227 - 11}{4} + 1 = \frac{216}{4} + 1 = 55$$

Notice that we didn't bother including the $2P$ term in our calculation above since the convolution is applied with zero padding.

(d) Develop a formula for the number of trainable parameters in a conv layer. Assume that the receptive field is a square and that the conv layer has biases. Define all the variables you use.

Solution:

There are K filters. Each filter can be thought of as its own linear classifier of sorts with weights and biases. The filter has size (F, F, C), so it has $F \cdot F \cdot C = F^2C$ weights. By convention, the filter has 1 bias term. Combining these two, we see that each filter has $F^2C + 1$ trainable parameters. Thus, across K filters, a conv layer will have $K(F^2C + 1)$ trainable parameters.

(e) Consider the first conv layer of AlexNet mentioned above. How many trainable parameters are there?

Solution:

$$K(CF^2 + 1) = 96(3 \cdot 11^2 + 1) = 96(3 \cdot 121 + 1) = 96(364) = 34944$$

(f) Consider a conv layer which takes a $31 \times 31 \times 5$ input and applies a 3×3 filter with depth 25, stride 2, and padding 1. How many trainable parameters does it have?

Solution:

$$K(CF^2 + 1) = 25(5 \cdot 3^2 + 1) = 25(5 \cdot 9 + 1) = 25(46) = 1150$$

Takeaway: The values you set K and F to (these are hyperparameters!) will have a significant impact on the number of parameters your model has. You must be careful not to add too many parameters to your model.

(g) Recall your answer for the output of AlexNet's first conv layer from above. This output is fed directly into a max pool layer which applies a 3x3 pool filter at stride 2 with no padding. What will the output size be? How many trainable parameters does this layer introduce?

Solution:

The pool layer takes an input of $(55, 55, 96)$. The pool layer slides a pool filter across the image, similar to how the conv layer slid a convulational filter across the image. Thus, we can use the same formula as before:

$$\frac{H-F}{S}+1=\frac{W-F}{S}+1=\frac{55-3}{2}+1=26+1=27$$

This gives us an output of size $(27, 27, 96)$. Note that the channel dimension stays constant since the pool is applied separately to each of the 96 input channels.

This layer introduces 0 trainable parameters since the pool filter uses a fixed function ($\max$).

(h) Did the pool layer change the number of channels? Does this pattern generalize to pool layers of all sizes?

Solution:

Pool layers generally do NOT change the number of channels.

(i) AlexNet precipitated the deep learning revolution. Explain one of the paper's key contributions.

Solution:

Some of AlexNet's major contributions include:

- Although previous papers had implemented neural networks on GPUs, Alex wrote a highly-optimized GPU implementation of convolutions code and successfully parallelized training across 2 GPUs.
- Popularized the use of ReLU as an activation function.
- Popularized the use of dropout to prevent overfitting.
- Proved that, when designed correctly, deeper neural networks could perform better than the shallower neural networks that were popular at the time. (Note that AlexNet was not the first paper to demonstrate the plausibility of deep neural nets; for example, see this).

This is not a comprehensive list.

2. Kernel of Truth

Consider the following input matrix I and filter F.

As an aside, you may also hear a convolution filter referred to as a kernel, mask, or convolutional matrix.

$$I = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ -1 & 0 & 1 & 2 \\ 0 & -2 & 4 & 0 \end{bmatrix} \qquad F = \begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix}$$

(a) Apply F on I with padding 0 and stride 1.

Solution:

Applying the convolution,

$$\begin{bmatrix} 1(1) - 1(2) + 0(4) + 2(3) & 1(2) - 1(3) + 0(3) + 2(2) & 1(3) - 1(4) + 0(2) + 2(1) \\ 1(4) - 1(3) + 0(-1) + 2(0) & 1(3) - 1(2) + 0(0) + 2(1) & 1(2) - 1(1) + 0(1) + 2(2) \\ 1(-1) + -1(0) + 0(0) + 2(-2) & 1(0) - 1(1) + 0(-2) + 2(4) & 1(1) - 1(2) + 0(4) + 2(0) \end{bmatrix}$$

Simplifying,

$$\begin{bmatrix} 5 & 3 & 1 \\ 1 & 3 & 5 \\ -5 & 7 & -1 \end{bmatrix}$$

(b) Apply F on I with padding 1 and stride 2.

Solution:

Note that after padding I looks like this:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 0 \\ 0 & 4 & 3 & 2 & 1 & 0 \\ 0 & -1 & 0 & 1 & 2 & 0 \\ 0 & 0 & -2 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Applying the convolution yields,

$$\begin{bmatrix} 2 & 6 & 0 \\ -6 & 3 & 1 \\ 0 & -6 & 0 \end{bmatrix}$$

(c) Apply a max pool on I with a 3×3 filter using a padding of 1 and a stride of 3.[1]

Solution:

$$\begin{bmatrix} 4 & 4 \\ 0 & 4 \end{bmatrix}$$

[1] In practice, pool layers are usually applied after conv layers; they are typically *not* applied directly on the input.

3. The City That Never Skips (Diagonals)

Examine the following code.

```
import numpy as np

X_test = np.random.rand(2, 3)
X_train = np.random.rand(4, 3)

num_test = X_test.shape[0]
num_train = X_train.shape[0]

dists = np.zeros((num_test, num_train))

# ***** START OF SOLUTION CODE *****

# ***** END OF SOLUTION CODE *****

return dists
```

In the space below, write out a naive, two-loop implementation of L1 distance, followed by a one-loop implementation and a zero-loop implementation. Assume that the code you write will be placed inside the `SOLUTION CODE` section above.

Solution:

Here is the naive, two-loop implementation.

```
for i in range(num_test):
    for j in range(num_train):
        abs_diff = np.abs(X_test[i] - X_train[j])
        dists[i, j] = np.sum(abs_diff)
```

And the one-loop version.

```
for i in range(num_test):
    abs_diff = np.abs(X_test[i] - X_train)
    dists[i] = np.sum(abs_diff, axis=1)
```

Finally, the vectorized one.

```
abs_diff = np.sum(X_test[:, None, :] - X_train)
dists = np.sum(abs_diff, axis=2)
```