CSE 493G/599G: Deep Learning

Section 3: Convolutions & Vectorization & Matrix-Vector Backprop

Welcome to section! We hope you have a great day.

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.

(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 strides of 1, 2, and 3. Then, compute the output sizes if we use strides of 2 and 3 with a padding of 3.

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

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?

(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.

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

(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?

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?

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

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

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.

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

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

[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.

4. Vector Virtuosity

Consider the following function,

$$f(W, x) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W * x)_i^2$$

where $W \in \mathbb{R}^{n \times n}$ and $x \in \mathbb{R}^n$.

First draw the function's computation graph. Then compute the forward pass for the following inputs.

$$W = \begin{bmatrix} 0.1 & 0.5 \\ -0.3 & 0.8 \end{bmatrix} \qquad x = \begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix}$$

Lastly, compute the backward pass. Verify your answer by deriving the closed forms of $\nabla_W f$ and $\nabla_x f$.