Problem Set 4

Basil R. Yap
50.021 Artificial Intelligence - Term 8

June 12, 2018

1 Theory Component

[Q1]. Consider a layer in CNN that takes in a single channel input of 64×64, and has 96 filters. In each of the following cases, compute the number of parameters that are learned in this layer. We assume that bias is present for each weight.

1. A convolution layer with filters of same size as the input.
2. A convolution layer with 8×8 filters with stride of 4
3. A convolution layer with 1×1 filter and a stride of 1

Solution

1. one convolution filter:

$$\begin{aligned} \text{no of parameters} &= \text{size of filter} \times \text{number of channels} + 1 \\ &= 64 \times 64 \times 1 + 1 \\ &= 4097 \end{aligned}$$

full convolution layer:

$$\begin{aligned} \text{no of parameters} &= 4097 \times 96 \\ &= 393312 \end{aligned}$$

number of parameters for one convolution filter:

$$\begin{aligned} \text{no of parameters} &= 8 \times 8 \times 1 + 1 \\ &= 65 \end{aligned}$$

full convolution layer:

$$\begin{aligned}\text{no of parameters} &= 65 \times 96 \\ &= 6240\end{aligned}$$

2. one convolution filter:

$$\begin{aligned}\text{no of parameters} &= 1 \times 1 \times 1 + 1 \\ &= 2\end{aligned}$$

full convolution layer:

$$\begin{aligned}\text{no of parameters} &= 2 \times 96 \\ &= 192\end{aligned}$$

[Q2].

Suppose you would have a neuron which has an RBF kernel as activation function (remember the evil wolf? Drop your linear style of thoughts. Circumferential thoughts can be nice too.)

$$y = \exp(-(x_1^2 + x_2^2)) + b$$

with parameter b. What would be the shapes realized by the set of points $\{(x_1, x_2) : y((x_1, x_2)) = 0\}$ as a function of b ? Explain in at most 2 sentences and/or a little math.

Suppose now we add weights:

$$y = \exp(-(w_1 x_1^2 + w_2 x_2^2)) + b$$

What shapes could you realize now? Explain in at most 5 sentences and/or a little math. You can make references to publicly available in the internet materials to explain.

Solution For $y = \exp(-(x_1^2 + x_2^2)) + b$, when $(x_1, x_2) : y((x_1, x_2)) = 0$

$$\begin{aligned}0 &= \exp(-(x_1^2 + x_2^2)) + b \\ -b &= \exp(-(x_1^2 + x_2^2)) \\ -\ln(-b) &= x_1^2 + x_2^2\end{aligned}$$

The shape of the function is a circle with centre, $x_1 = 0,\ x_2 = 0$ and radius, $\sqrt{-\ln(-b)}$.

$$\begin{aligned}\sqrt{-\ln(-b)} &\geq 0 \\ -\ln(-b) &\geq 0 \\ \ln(-b) &\leq 0\end{aligned}$$

$$\begin{aligned} -b \leq & \ \exp(0) & -b & \geq 0 \\ b & \geq -1 & b & \leq 0 \end{aligned}$$

$$\text{where } -1 \geq b \geq 0$$

For $y = \exp(-(w_1 x_1^2 + w_2 x_2^2)) + b$, when $(x_1, x_2) : y((x_1, x_2)) = 0$

$$\begin{aligned} 0 &= \exp(-(w_1 x_1^2 + w_2 x_2^2)) + b \\ -b &= \exp(-(w_1 x_1^2 + w_2 x_2^2)) \\ -\ln(-b) &= w_1 x_1^2 + w_2 x_2^2 \\ \frac{-\ln(-b)}{w_1 w_2} &= \frac{x_1^2}{w_2} + \frac{x_2^2}{w_1} \end{aligned}$$

The shape of the function is a ellipse with centre, $x_1 = 0,\ x_2 = 0$,
x_1-axis radius, $\sqrt{\frac{-\ln(-b)}{w_1}}$ and x_2-axis radius, $\sqrt{\frac{-\ln(-b)}{w_2}}$.
where $-1 \geq b \geq 0$

[Q3].

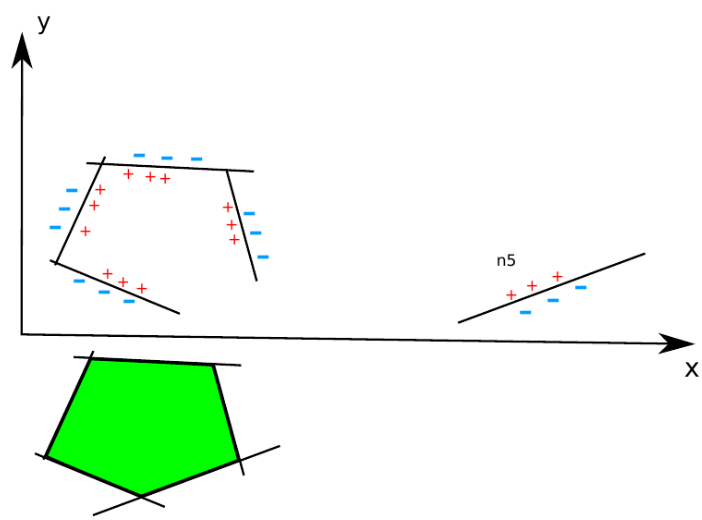

Figure 1: shapes

Suppose you have five linear neurons neurons $n_1, \ldots, n_5$, realizing above decision boundaries as shown in Figure 1. That is: for every decision boundary we have outputs are $= 0.5$ in the zones marked with red plusses, and $= 0.2$ in the zones marked with the blue minuses. As you know, each neuron is realized by:

$$n_i = 0.3H(w_1^{(i)}x_1 + w_2^{(i)}x_2 + b^{(i)}) + 0.2, H(z) \in \{0, 1\}$$

where H is the threshold activation function.

You want to predict positive values in a shape marked in green in Figure 1. You want to achieve this prediction by combining these neurons using a threshold neuron H:

$$y = H(\sum_i v_i^* n_i + b^*)$$

1. what do you have to do with the weights of n_5 so that you can move the decision boundary of n_5 so that you can realize the shape in green shown above (in the sense of having positive values inside and negative values outside.)? Give a qualitative description. Note: Give a qualitative description in 3 sentences at most. Note that there is an x- and an y-axis, which helps you to express vectors qualitatively.
2. after moving the decision boundary of n_5 appropriately, the green shape looks a bit like an logical AND-combination of the +-zones for every neuron. How to choose the weights v_i^* and the bias b^* in

$$y = H(\sum_i v_i^* n_i + b^*)$$

so that you can realize the green shape (in the sense of having positive values inside and negative values outside that shape)?

Note: n_i gives out values either 0.5 or 0.2.

I hope that exercise explains you more what neural networks with 2 layers can achieve as shapes. With 3 layers you can realize an OR-combination of green shapes as above.

Solution

1. The decision boundary of n_5 is a linear function with gradients that is a function of the ratio of w_1 and w_2, and bias that is a function of the ratio of b and w_2.
   In order to shift the decision boundary, we need to decrease the value of b.
2. Let $H(x) = 1[z > 0]$

$$\text{where } 1[x] = \begin{cases} 1 & \text{If } x \text{ is True} \\ 0 & \text{If } x \text{ is False} \end{cases}$$

Consider when within the green region $y = 1$ and $n_i = 0.5\ \forall i \in [1, n]$

$$
\begin{aligned}
1 &= H(\sum_i 0.5v_i^* + b^*) \\
1 &= 1[(\sum_i 0.5v_i^* + b^*) > 0] \\
\sum_i 0.5v_i^* + b^* &> 0 \\
b^* &> -\sum_i 0.5v_i^*
\end{aligned}
$$

Consider when outside the green region $y = 0$, $n_j = 0.2$ and $n_i = 0.5\ \forall i \in ([1, n] \setminus j)$

$$
\begin{aligned}
0 &= H(\sum_i 0.5v_i^* + 0.2v_j^* + b^*) \\
0 &= 1[(\sum_i 0.5v_i^* + 0.2v_j^* + b^*) > 0] \\
\sum_i 0.5v_i^* + 0.2v_j^* + b^* &\leq 0 \\
\sum_i 0.5v_i^* + 0.2v_j^* + b^* &\leq 0 \\
b^* &\leq -\sum_i 0.5v_i^* - 0.2v_j^* \\
-\sum_i 0.5v_i^* - 0.5v_j^* < b^* &\leq -\sum_i 0.5v_i^* - 0.2v_j^*
\end{aligned}
$$

We can get the bounded green region as long as we satisfy the above inequality for all values of $v_k\ \forall k \in [1, n]$ with all combinations $\forall j \in [1, n]$ with $i \in ([1, n] \setminus j)$.

2 Coding Component

```python
from PIL import Image
from torch import Tensor, tensor
from torch.utils.data import Dataset, DataLoader
from torchvision.models import resnet18
from torchvision.transforms import *
from getimagenetclasses import parsesynsetwords, parseclasslabel
import sys


class cropSet(Dataset):

    def __init__(self, path, xmlDir, size):
        self.path = path
        self.xmlDir = xmlDir
        self.size = size
        self.transform = None

    def __len__(self):
        return self.size

    def __getitem__(self, idx):
        path = self.path.format(idx + 1)
        im = Image.open(path).convert("RGB")

        idx, name = parseclasslabel(self.xmlDir.format(idx + 1), syn2idx)

        if self.transform:
            im = self.transform(im)

        return im, idx


def center(img):
    C, W, H = img.size()
    crop = img.new_empty((3, 224, 224))
    x = img.narrow(1, (W - 224) // 2, 224)
    crop = x.narrow(2, (H - 224) // 2, 224)

    return crop


class centerSet(cropSet):
```

```
    def __init__(self, *arg):
        super().__init__(*arg)
        self.transform = \
            Compose([
                Resize(224),
                ToTensor(),
                Normalize(mean=[0.485, 0.456, 0.406],
                          std=[0.229, 0.224, 0.225]),
                center,
            ])


def five(img):
    C, W, H = img.size()
    crop5 = img.new_empty((5, 3, 224, 224))
    x = img.narrow(1, 0, 224)
    x = x.narrow(2, 0, 224)
    crop5[0] = x
    x = img.narrow(1, W - 224, 224)
    x = x.narrow(2, 0, 224)
    crop5[1] = x
    x = img.narrow(1, W - 224, 224)
    x = x.narrow(2, H - 224, 224)
    crop5[2] = x
    x = img.narrow(1, 0, 224)
    x = x.narrow(2, H - 224, 224)
    crop5[3] = x
    x = img.narrow(1, (W - 224) // 2, 224)
    x = x.narrow(2, (H - 224) // 2, 224)
    crop5[4] = x

    return crop5


class fiveSet(cropSet):

    def __init__(self, *arg):
        super().__init__(*arg)
        self.transform = \
            Compose([
                Resize(280),
                ToTensor(),
                Normalize(mean=[0.485, 0.456, 0.406],
                          std=[0.229, 0.224, 0.225]),
```

```python
                five,
            ])


def test(model, dataloader):
    total = 0
    correct = 0
    for n, sample_batched in enumerate(dataloader):
        data, descs = sample_batched
        X = data.view(-1, 3, 224, 224)
        out = model.forward(X)
        out = out.view(batch_size, -1, 1000)
        out = out.mean(1)
        val, pred = out.max(1)
        cmp = pred.eq(descs)
        total += cmp.size(0)
        correct += cmp.sum()

    return int(correct), total

def main(dataDir, xmlDir, synDir, n=250, batch_size=10, is_shuffle=False):
    idx2syn, syn2idx, syn2desc = parsesynsetwords(synDir)
    fivecropset = fiveSet(dataDir, xmlDir, n)
    fivecroploader = DataLoader(
        fivecropset, batch_size=batch_size, shuffle=is_shuffle)
    centercropset = centerSet(dataDir, xmlDir, n)
    centercroploader = DataLoader(
        centercropset, batch_size=batch_size, shuffle=is_shuffle)
    model = resnet18(pretrained=True)
    model.eval()
    fiveResult = test(model, fivecroploader)
    centerResult = test(model, centercroploader)

    return fiveResult, centerResult


if __name__ == "__main__":
    fiveResult, centerResult = main(sys.argv[1], sys.argv[2], sys.argv[3])
    print("Five Crop Accuracy: %s" % fiveResult)
    print("Center Crop Accuracy: %s" % centerResult)
```