Problem Set 6

Basil R. Yap
50.021 Artificial Intelligence - Term 8

June 27, 2018

1 Theory Component

[Q1] The following is the AdaGrad algorithm for weight update.

$$cache_i = cache_i + (\nabla_{w_i} L)^2$$
$$w_i = w_i - \frac{\eta}{\sqrt{cache_i} + \epsilon} \nabla_{w_i} L$$

where w_i is the weight to be updated, $\nabla_{w_i} L$ is the gradient of the loss w.r.t w_i, ϵ is a hyperparemeter between 10^{-8} and 10^{-4} and η is a hyperparameter similar to step size in SGD. List one difference between AdaGrad and SGD in terms of step size and **explain** what effects you expect from this difference.

Solution

Solution

Solution

2 Coding Component

```
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
```

[Q2] The following are the defining equations for a LSTM cell,

$$
\begin{aligned}
i_t &= \sigma(W^i x_t + U^i h_{t-1}) \\
f_t &= \sigma(W^f x_t + U^f h_{t-1}) \\
o_t &= \sigma(W^o x_t + U^o h_{t-1}) \\
\tilde{c}_t &= \tanh(W^c x_t + U^c h_{t-1}) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
$$

The symbol $\circ$ denotes element-wise multiplication and $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. Answer True/False to the following questions and give not more than 2 sentences explanation.

1. If $x_t = 0$ vector then $h_t = h_{t-1}$.
2. If f_t is very small or zero, then the error will not be back-propagated to earlier time steps.
3. The entries of f_t, i_t, o_t are non-negative.
4. f_t, i_t, o_t can be seen as probability distributions, which means that their entries are non-negative and their entries sum to 1.

[Q3] The defining equations for a GRU cell are,

$$
\begin{aligned}
z_t &= \sigma(W^z x_t + U^z h_{t-1}) \\
r_t &= \sigma(W^r x_t + U^r h_{t-1}) \\
\tilde{h}_t &= \tanh(W x_t + r_t \circ U h_{t-1}) \\
h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t
\end{aligned}
$$

1. Draw a diagram of this GRU cell.
2. Assume h_t and x_t are column vectors, with dimensions d_h and d_x respectively. What are the dimensions (rows $\times$ columns) of the weight matrices W^z, W^r, W, U^z, U^r, and U?
3. Like LSTM cells, GRU cells can tackle vanishing or exploding gradient problem too. By taking a look at the formula for LSTM in Q2, what is the main advantage of using GRU cells over LSTMs for some problems? Give an answer it at most 5 sentences.
   Hint: We expect a qualitative answer (deep math proofs are not required) that comes with an explanation of the answer.

```python
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]),
                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)
```