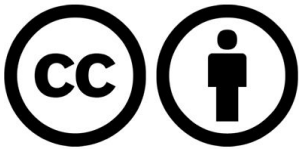
With PyTorch 0.2.0_2

Lab 9

NN for XOR

Sung Kim <hunkim+ml@gmail.com>

Code: https://github.com/hunkim/DeepLearningZeroToAll/



https://github.com/hunkim/DeepLearningZeroToAll/tree/master/pytorch

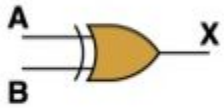
https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-06-1-softmax_classifier.py

XOR data set

Boolean Expression

$X = A \oplus B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

```
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0],    [1],    [1],    [0]], dtype=np.float32)
```

http://templ25.mandaringardencity.com/xor-gate-truth-table-2/

```
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)

X = Variable(torch.from_numpy(x_data))
Y = Variable(torch.from_numpy(y_data))

# Hypothesis using sigmoid
linear = torch.nn.Linear(2, 1, bias=True)
sigmoid = torch.nn.Sigmoid()
model = torch.nn.Sequential(linear, sigmoid)

optimizer = torch.optim.SGD(model.parameters(), lr=0.1)

for step in range(10001):
    optimizer.zero_grad()
    hypothesis = model(X)
    # cost/loss function
    cost = -(Y * torch.log(hypothesis) + (1 - Y)
             * torch.log(1 - hypothesis)).mean()
    cost.backward()
    optimizer.step()

    if step % 100 == 0:
        print(step, cost.data.numpy())

# Accuracy computation
# True if hypothesis>0.5 else False
predicted = (model(X).data > 0.5).float()
accuracy = (predicted == Y.data).float().mean()
print("\nHypothesis: ", hypothesis.data.numpy(), "\nCorrect: ", predicted.numpy(), "\nAccuracy: ", accuracy)
```

XOR with logistic regression?

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-06-1-softmax_classifier.py

```
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)

X = Variable(torch.from_numpy(x_data))
Y = Variable(torch.from_numpy(y_data))

# Hypothesis using sigmoid
linear = torch.nn.Linear(2, 1, bias=True)
sigmoid = torch.nn.Sigmoid()
model = torch.nn.Sequential(linear, sigmoid)

optimizer = torch.optim.SGD(model.parameters(), lr=0.1)

for step in range(10001):
    optimizer.zero_grad()
    hypothesis = model(X)
    # cost/loss function
    cost = -(Y * torch.log(hypothesis) + (1 - Y)
             * torch.log(1 - hypothesis)).mean()
    cost.backward()
    optimizer.step()

    if step % 100 == 0:
        print(step, cost.data.numpy())

# Accuracy computation
# True if hypothesis>0.5 else False
predicted = (model(X).data > 0.5).float()
accuracy = (predicted == Y.data).float().mean()
print("\nHypothesis: ", hypothesis.data.numpy(), "\nCorrect: ", predicted.numpy(), "\nAccuracy: ", accuracy)
```

XOR with logistic regression?

But it doesn't work!

```
Hypothesis:  [[ 0.49999997]
 [ 0.5       ]
 [ 0.5       ]
 [ 0.5       ]]
Correct:  [[ 0.]
 [ 0.]
 [ 0.]
 [ 0.]]
Accuracy:  0.5
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-06-1-softmax_classifier.py

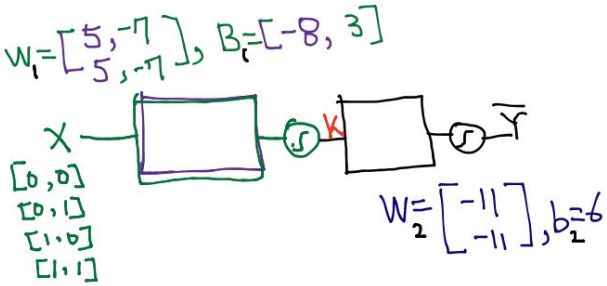
https://github.com/hunkim/DeepLearningZeroToAll/tree/master/pytorch

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-09-2-xor-nn.py

Neural Net

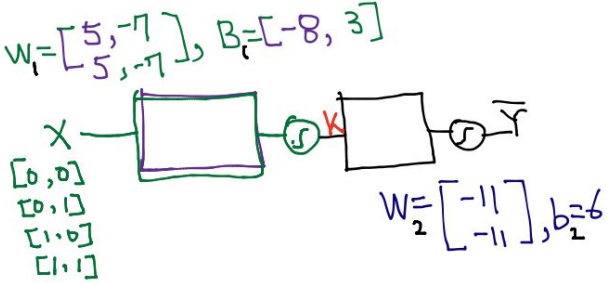
```
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)

X = Variable(torch.from_numpy(x_data))
Y = Variable(torch.from_numpy(y_data))
```



https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-09-2-xor-nn.py

Neural Net




```
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)

X = Variable(torch.from_numpy(x_data))
Y = Variable(torch.from_numpy(y_data))
```

```
linear1 = torch.nn.Linear(2, 2, bias=True)
linear2 = torch.nn.Linear(2, 1, bias=True)
sigmoid = torch.nn.Sigmoid()
model = torch.nn.Sequential(linear1, sigmoid, linear2, sigmoid)

optimizer = torch.optim.SGD(model.parameters(), lr=0.1)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-09-2-xor-nn.py

NN for XOR

```
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)

X = Variable(torch.from_numpy(x_data))
Y = Variable(torch.from_numpy(y_data))

linear1 = torch.nn.Linear(2, 2, bias=True)
linear2 = torch.nn.Linear(2, 1, bias=True)
sigmoid = torch.nn.Sigmoid()
model = torch.nn.Sequential(linear1, sigmoid, linear2, sigmoid)

optimizer = torch.optim.SGD(model.parameters(), lr=0.1)

for step in range(10001):
    optimizer.zero_grad()
    hypothesis = model(X)
    # cost/loss function
    cost = -(Y * torch.log(hypothesis) + (1 - Y)
             * torch.log(1 - hypothesis)).mean()
    cost.backward()
    optimizer.step()

    if step % 100 == 0:
        print(step, cost.data.numpy())

# Accuracy computation
# True if hypothesis>0.5 else False
predicted = (model(X).data > 0.5).float()
accuracy = (predicted == Y.data).float().mean()
print("\nHypothesis: ", hypothesis.data.numpy(), "\nCorrect: ", predicted.numpy(), "\nAccuracy: ", accuracy)
```

```
Hypothesis:  [[ 0.0216833 ]
 [ 0.97211885]
 [ 0.97253156]
 [ 0.04630803]]
Correct:  [[ 0.]
 [ 1.]
 [ 1.]
 [ 0.]]
Accuracy:  1.0
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-09-2-xor-nn.py

With PyTorch 0.2.0_2

Lab 10

NN, Xavier, Dropout

Sung Kim <hunkim+ml@gmail.com>

Code: https://github.com/hunkim/DeepLearningZeroToAll/