With PyTorch 0.2.0_2

PYTORCH

Lab 5

Logistic (regression) classifier

Sung Kim <hunkim+ml@gmail.com>

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

With PyTorch 0.2.0_2

PYTORCH

Lab 5-1

Logistic regression

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-05-1-logistic_regression.py

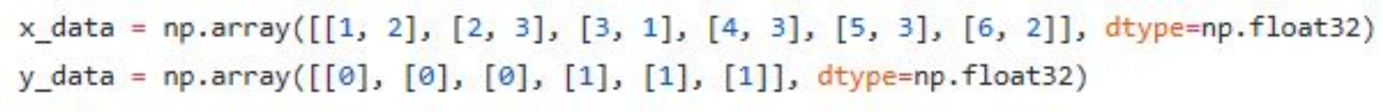
Logistic Regression

$$H(X) = \frac{1}{1 + e^{-W^T X}}$$

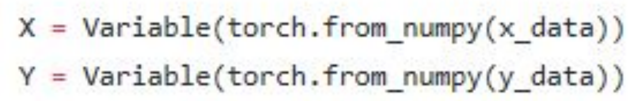
$$cost(W) = -\frac{1}{m}\sum y log(H(x)) + (1 - y)(log(1 - H(x))$$

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

Training Data

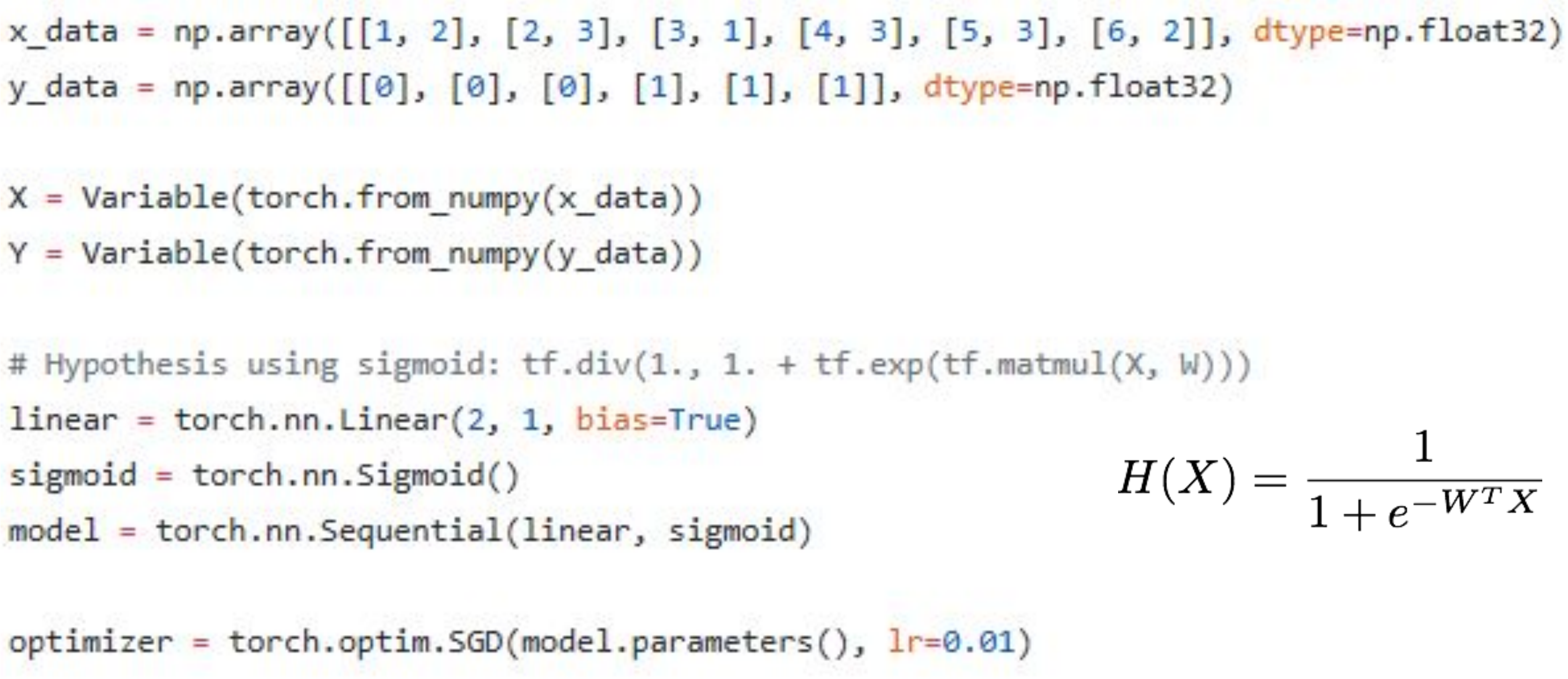


```
x_data = np.array([[1, 2], [2, 3], [3, 1], [4, 3], [5, 3], [6, 2]], dtype=np.float32)
y_data = np.array([[0], [0], [0], [1], [1], [1]], dtype=np.float32)
```



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

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-05-1-logistic_regression.py

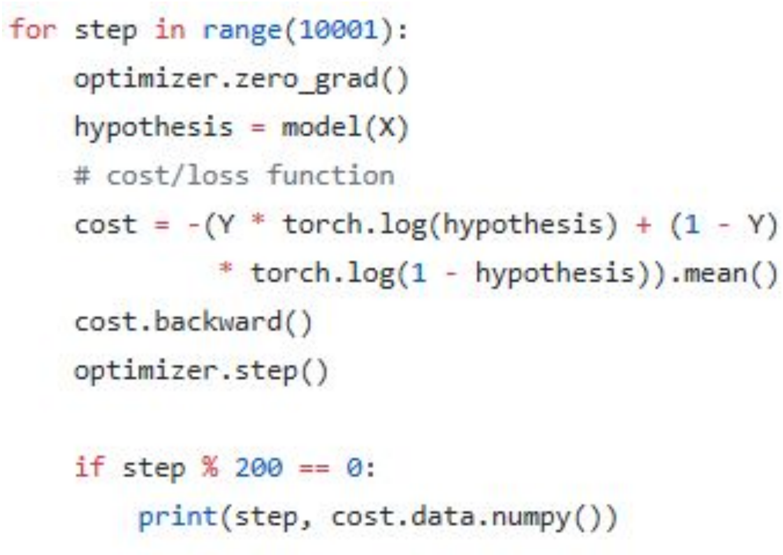


```
x_data = np.array([[1, 2], [2, 3], [3, 1], [4, 3], [5, 3], [6, 2]], dtype=np.float32)
y_data = np.array([[0], [0], [0], [1], [1], [1]], dtype=np.float32)

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

# Hypothesis using sigmoid: tf.div(1., 1. + tf.exp(tf.matmul(X, W)))
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=0.01)
```

$$H(X) = \frac{1}{1 + e^{-W^T X}}$$

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-05-1-logistic_regression.py

Train the model

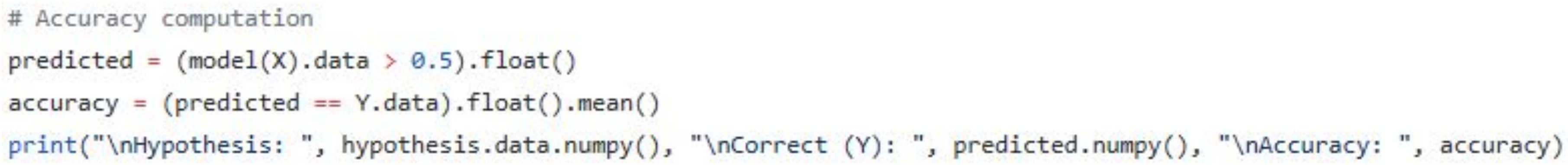


```
for step in range(10001):
    optimizer.zero_grad()
    hypothesis = model(X)
    # cost/loss function
    cost = -(Y * torch.log(hypothesis) + (1 - Y)
             * torch.log(1 - hypothesis)).mean()
    cost.backward()
    optimizer.step()

    if step % 200 == 0:
        print(step, cost.data.numpy())
```

$$H(X) = \frac{1}{1 + e^{-W^T X}}$$

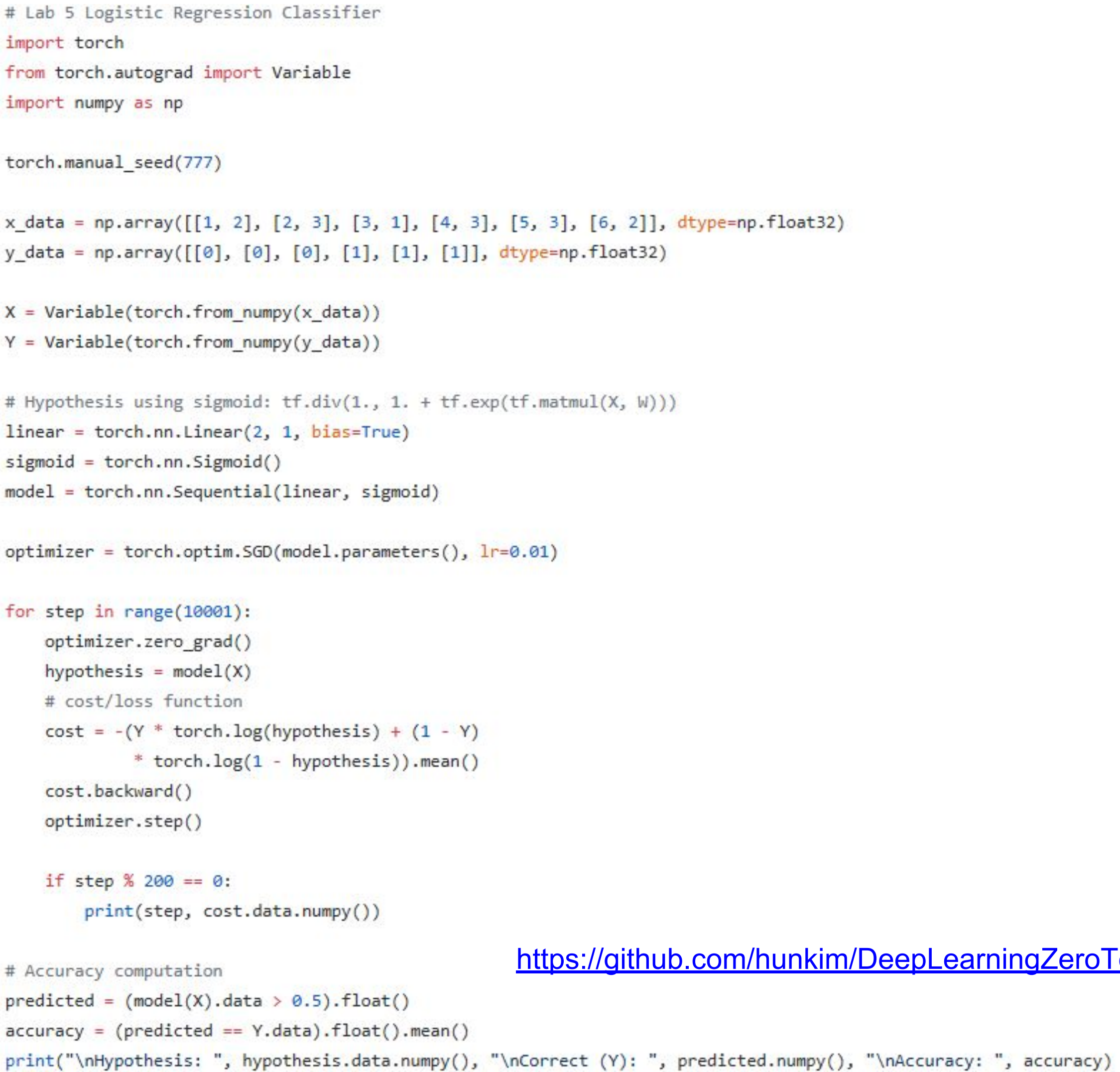
$$cost(W) = -\frac{1}{m}\sum y log(H(x)) + (1 - y)(log(1 - H(x))$$

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$



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

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-05-1-logistic_regression.py



```python
# Lab 5 Logistic Regression Classifier
import torch
from torch.autograd import Variable
import numpy as np

torch.manual_seed(777)

x_data = np.array([[1, 2], [2, 3], [3, 1], [4, 3], [5, 3], [6, 2]], dtype=np.float32)
y_data = np.array([[0], [0], [0], [1], [1], [1]], dtype=np.float32)

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

# Hypothesis using sigmoid: tf.div(1., 1. + tf.exp(tf.matmul(X, W)))
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=0.01)

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 % 200 == 0:
        print(step, cost.data.numpy())

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

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-05-1-logistic_regression.py

With PyTorch 0.2.0_2

PYTORCH

Lab 5-2

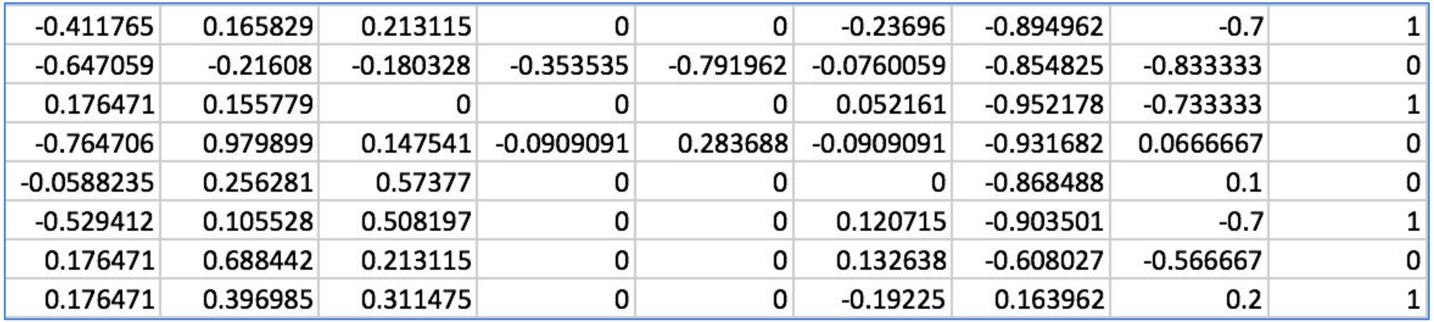
Logistic regression - diabetes

Sung Kim <hunkim+ml@gmail.com>
Code: https://github.com/hunkim/DeepLearningZeroToAll/

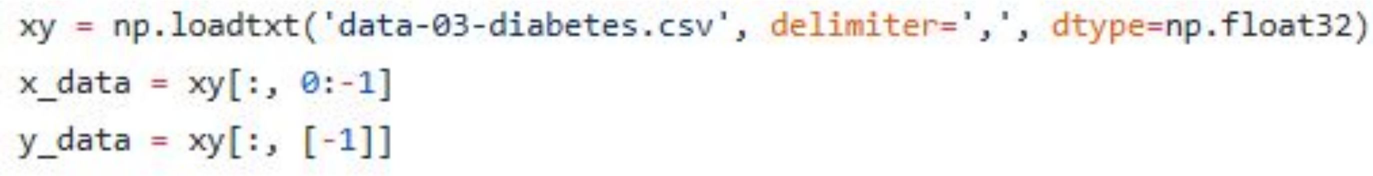
https://github.com/hunkim/DeepLearningZeroToAll/tree/master/pytorch

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-05-2-logistic_regression_diabetes.py

Classifying diabetes

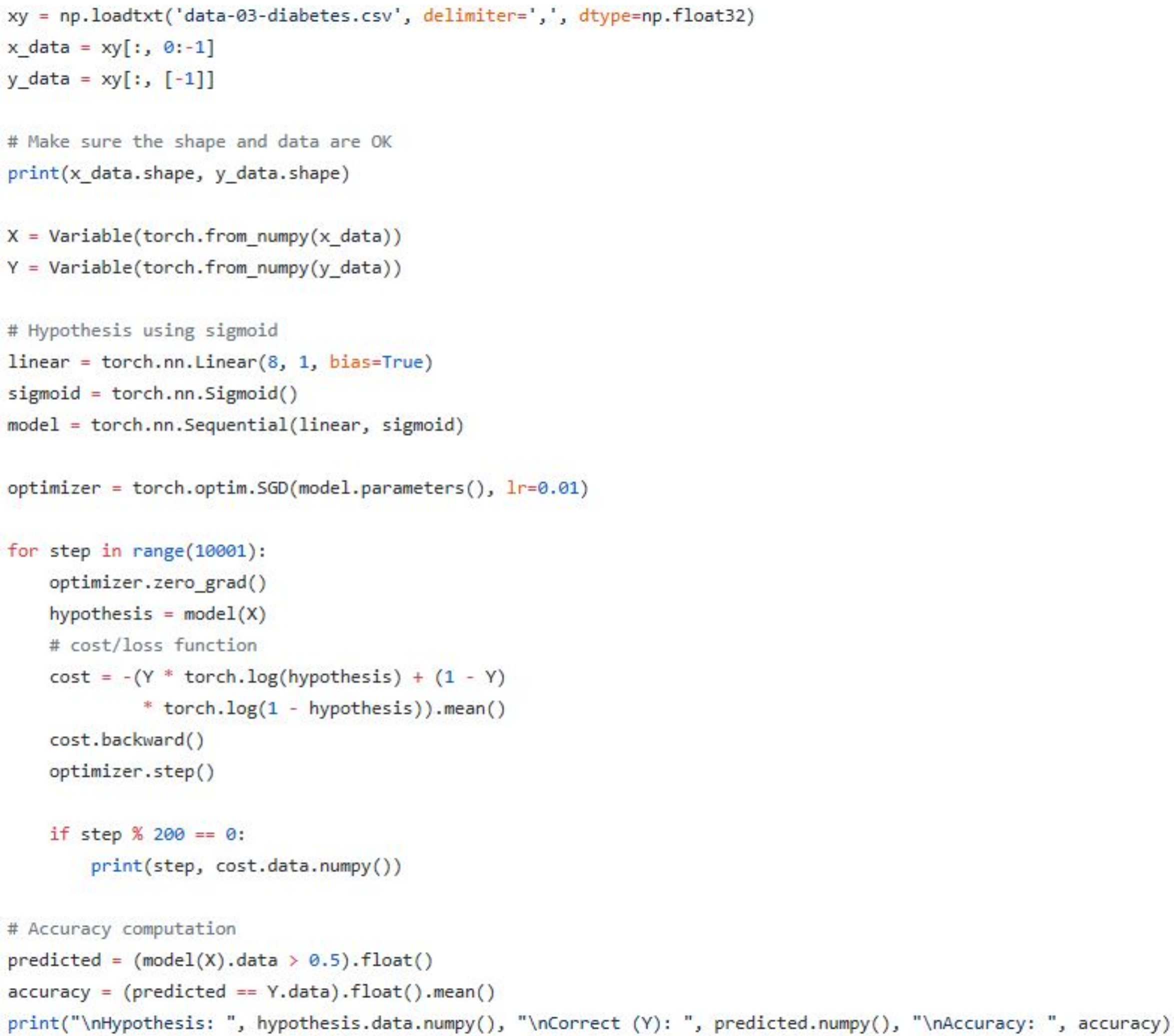


-0.411765	0.165829	0.213115	0	0	-0.23696	-0.894962	-0.7	1
-0.647059	-0.21608	-0.180328	-0.353535	-0.791962	-0.0760059	-0.854825	-0.833333	0
0.176471	0.155779	0	0	0	0.052161	-0.952178	-0.733333	1
-0.764706	0.979899	0.147541	-0.0909091	0.283688	-0.0909091	-0.931682	0.0666667	0
-0.0588235	0.256281	0.57377	0	0	0	-0.868488	0.1	0
-0.529412	0.105528	0.508197	0	0	0.120715	-0.903501	-0.7	1
0.176471	0.688442	0.213115	0	0	0.132638	-0.608027	-0.566667	0
0.176471	0.396985	0.311475	0	0	-0.19225	0.163962	0.2	1



```
xy = np.loadtxt('data-03-diabetes.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-05-2-logistic_regression_diabetes.py



```
xy = np.loadtxt('data-03-diabetes.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]

# Make sure the shape and data are OK
print(x_data.shape, y_data.shape)

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

# Hypothesis using sigmoid
linear = torch.nn.Linear(8, 1, bias=True)
sigmoid = torch.nn.Sigmoid()
model = torch.nn.Sequential(linear, sigmoid)

optimizer = torch.optim.SGD(model.parameters(), lr=0.01)

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 % 200 == 0:
        print(step, cost.data.numpy())

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

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-05-2-logistic_regression_diabetes.py

Exercise

- Try other classification data from Kaggle
    - https://www.kaggle.com

With PyTorch 0.2.0_2

PYTORCH

Lab 6

Softmax classifier

Sung Kim <hunkim+ml@gmail.com>