With PyTorch 0.2.0_2

PYTORCH

Lab 2

Linear 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-02-1%262-linear_regression.py

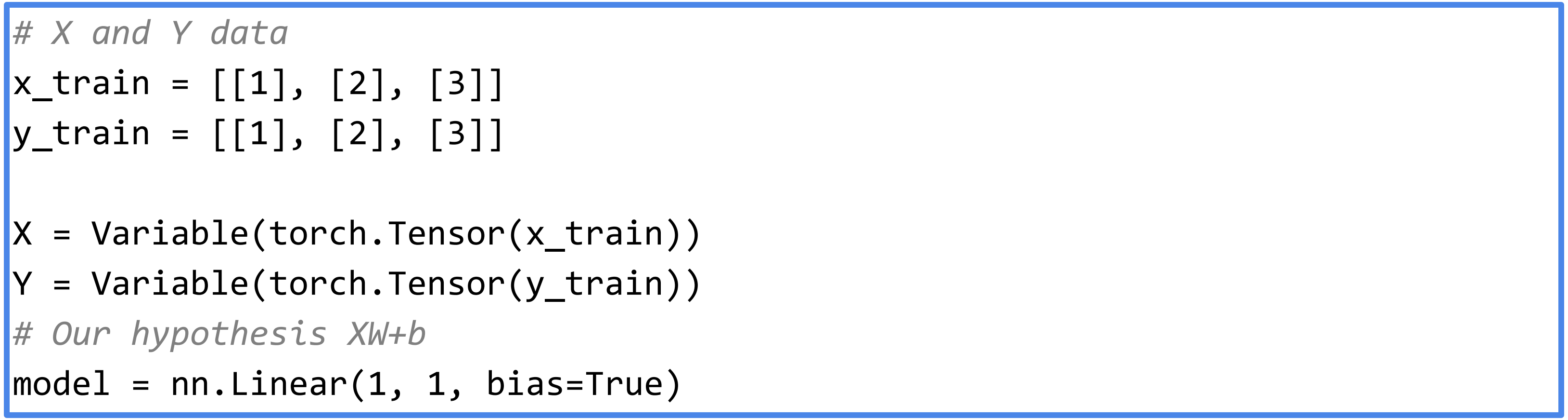
Hypothesis and cost function

$$H(x) = Wx + b$$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

1 Build model using PyTorch operations

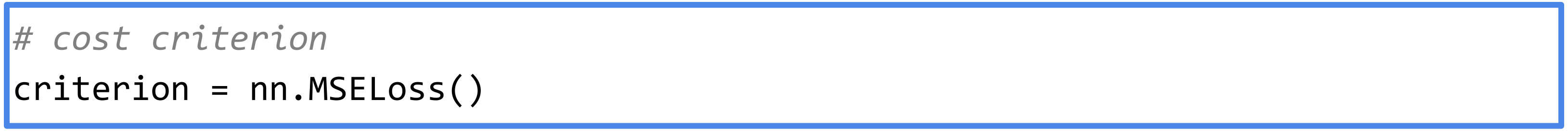
$$H(x) = Wx + b$$



```
# X and Y data
x_train = [[1], [2], [3]]
y_train = [[1], [2], [3]]

X = Variable(torch.Tensor(x_train))
Y = Variable(torch.Tensor(y_train))
# Our hypothesis XW+b
model = nn.Linear(1, 1, bias=True)
```

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$



```
# cost criterion
criterion = nn.MSELoss()
```

1 Build model using PyTorch operations

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

```
# cost criterion
criterion = nn.MSELoss()
```

Optimizer

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

2 Train the model

```
# Train the model
 for step in range(2001):
     optimizer.zero_grad()
     # Our hypothesis
     hypothesis = model(X)
     cost = criterion(hypothesis, Y)
     cost.backward()
     optimizer.step()

     if step % 20 == 0:
         print(step, cost.data.numpy(), model.weight.data.numpy(),
               model.bias.data.numpy())
```

3 Test the model

```
# Testing our model
predicted = model(Variable(torch.Tensor([[5]])))
print(predicted.data.numpy())
predicted = model(Variable(torch.Tensor([[2.5]])))
print(predicted.data.numpy())
predicted = model(Variable(torch.Tensor([[1.5], [3.5]])))
print(predicted.data.numpy())
```

Full code

```python
# lab 2 Linear Regression
import torch
import torch.nn as nn
from torch.autograd import Variable

torch.manual_seed(777)   # for reproducibility

# X and Y data
x_train = [[1], [2], [3]]
y_train = [[1], [2], [3]]

X = Variable(torch.Tensor(x_train))
Y = Variable(torch.Tensor(y_train))

# Our hypothesis XW+b
model = nn.Linear(1, 1, bias=True)

# cost criterion
criterion = nn.MSELoss()

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

```python
# Train the model
 for step in range(2001):
     optimizer.zero_grad()
     # Our hypothesis
     hypothesis = model(X)
     cost = criterion(hypothesis, Y)
     cost.backward()
     optimizer.step()

     if step % 20 == 0:
         print(step, cost.data.numpy(), model.weight.data.numpy(),
               model.bias.data.numpy())

# Testing our model
predicted = model(Variable(torch.Tensor([[5]])))
print(predicted.data.numpy())
predicted = model(Variable(torch.Tensor([[2.5]])))
print(predicted.data.numpy())
predicted = model(Variable(torch.Tensor([[1.5], [3.5]])))
print(predicted.data.numpy())
```

With PyTorch 0.2.0_2

PYTORCH

Lab 4

Multi-variable linear regression

Sung Kim <hunkim+ml@gmail.com>