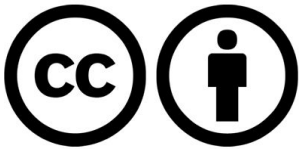
With PyTorch 0.2.0_2


# Lab 4

## Multi-variable linear regression

Sung Kim <hunkim+ml@gmail.com>

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



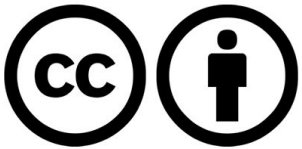
With PyTorch 0.2.0_2


# Lab 4-1

## Multi-variable 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-04-2-multi_variable_linear_regression.py

# Hypothesis using matrix

$$H(x_1, x_2, x_3) = x_1w_1 + x_2w_2 + x_3w_3$$

| $x_1$ | $x_2$ | $x_3$ | Y |
|---|---|---|---|
| 73 | 80 | 75 | 152 |
| 93 | 88 | 93 | 185 |
| 89 | 91 | 90 | 180 |
| 96 | 98 | 100 | 196 |
| 73 | 66 | 70 | 142 |

Test Scores for General Psychology

# Hypothesis using matrix

$$H(x_1, x_2, x_3) = x_1 w_1 + x_2 w_2 + x_3 w_3$$

| $x_1$ | $x_2$ | $x_3$ | Y |
|---|---|---|---|
| 73 | 80 | 75 | 152 |
| 93 | 88 | 93 | 185 |
| 89 | 91 | 90 | 180 |
| 96 | 98 | 100 | 196 |
| 73 | 66 | 70 | 142 |

Test Scores for General Psychology

```
# X and Y data
x_data = [[73., 80., 75.], [93., 88., 93.],
          [89., 91., 90.], [96., 98., 100.], [73., 66., 70.]]
y_data = [[152.], [185.], [180.], [196.], [142.]]

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

# Our hypothesis XW+b
model = nn.Linear(3, 1, bias=True)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-04-1-multi_variable_linear_regression.py

```python
# Lab 4 Multi-variable 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_data = [[73., 80., 75.], [93., 88., 93.],
          [89., 91., 90.], [96., 98., 100.], [73., 66., 70.]]
y_data = [[152.], [185.], [180.], [196.], [142.]]

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

# Our hypothesis XW+b
model = nn.Linear(3, 1, bias=True)

# cost criterion
criterion = nn.MSELoss()

# Minimize
optimizer = torch.optim.SGD(model.parameters(), lr=1e-5)
```

```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 % 10 == 0:
        print(step, "Cost: ", cost.data.numpy(), "\nPrediction:\n", hypothesis.data.numpy())
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-04-2-multi_variable_linear_regression.py

# Matrix

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix} \qquad H(X) = XW$$

# Matrix

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix} \qquad H(X) = XW$$

```
# X and Y data
x_data = [[73., 80., 75.], [93., 88., 93.],
          [89., 91., 90.], [96., 98., 100.], [73., 66., 70.]]
y_data = [[152.], [185.], [180.], [196.], [142.]]

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

# Our hypothesis XW+b
model = nn.Linear(3, 1, bias=True)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-04-2-multi_variable_linear_regression.py

```python
# Lab 4 Multi-variable 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_data = [[73., 80., 75.], [93., 88., 93.],
          [89., 91., 90.], [96., 98., 100.], [73., 66., 70.]]
y_data = [[152.], [185.], [180.], [196.], [142.]]

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

# Our hypothesis XW+b
model = nn.Linear(3, 1, bias=True)

# cost criterion
criterion = nn.MSELoss()

# Minimize
optimizer = torch.optim.SGD(model.parameters(), lr=1e-5)
```

```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 % 10 == 0:
        print(step, "Cost: ", cost.data.numpy(), "\nPrediction:\n", hypothesis.data.numpy())
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-04-2-multi_variable_matmul_linear_regression.py

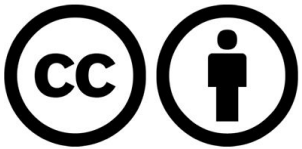
With TF 1.0!

# Lab 4-2

## Loading Data from File

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-04-3-file_input_linear_regression.py

# Loading data from file

**data-01-test-score.csv**

```
# EXAM1,EXAM2,EXAM3,FINAL
73,80,75,152
93,88,93,185
89,91,90,180
96,98,100,196
73,66,70,142
53,46,55,101
```

```python
import numpy as np

torch.manual_seed(777)  # for reproducibility

xy = np.loadtxt('data-01-test-score.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, x_data, len(x_data))
print(y_data.shape, y_data)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-04-3-file_input_linear_regression.py

# Slicing

```
nums = range(5)    # range is a built-in function that creates a list of integers
print nums         # Prints "[0, 1, 2, 3, 4]"
print nums[2:4]    # Get a slice from index 2 to 4 (exclusive); prints "[2, 3]"
print nums[2:]     # Get a slice from index 2 to the end; prints "[2, 3, 4]"
print nums[:2]     # Get a slice from the start to index 2 (exclusive); prints "[0, 1]"
print nums[:]      # Get a slice of the whole list; prints ["0, 1, 2, 3, 4]"
print nums[:-1]    # Slice indices can be negative; prints ["0, 1, 2, 3]"
nums[2:4] = [8, 9] # Assign a new sublist to a slice
print nums         # Prints "[0, 1, 8, 9, 4]"
```

http://cs231n.github.io/python-numpy-tutorial/

# Indexing, Slicing, Iterating

- Arrays can be indexed, sliced, iterated much like lists and other sequence types in Python
- As with Python lists, slicing in NumPy can be accomplished with the colon ( : ) syntax
- Colon instances ( : ) can be replaced with dots ( ... )

```
a = np.array([1, 2, 3, 4, 5])
# array([1, 2, 3, 4, 5])

a[1:3]
# array([2, 3])

a[-1]
# 5

a[0:2] = 9

a
# array([9, 9, 3, 4, 5])
```

```
b = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
# array([[ 1,  2,  3,  4],
#        [ 5,  6,  7,  8],
#        [ 9, 10, 11, 12]])

b[:, 1]
# array([ 2,  6, 10])

b[-1]
# array([ 9, 10, 11, 12])

b[-1, :]
# array([ 9, 10, 11, 12])

b[-1, ...]
# array([ 9, 10, 11, 12])

b[0:2, :]
# array([[1, 2, 3, 4],
#        [5, 6, 7, 8]])
```

# Loading data from file

**data-01-test-score.csv**

```
# EXAM1,EXAM2,EXAM3,FINAL
73,80,75,152
93,88,93,185
89,91,90,180
96,98,100,196
73,66,70,142
53,46,55,101
```

```python
import numpy as np

torch.manual_seed(777)  # for reproducibility

xy = np.loadtxt('data-01-test-score.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, x_data, len(x_data))
print(y_data.shape, y_data)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-04-3-file_input_linear_regression.py

```python
# Lab 4 Multi-variable linear regression
import torch
import torch.nn as nn
from torch.autograd import Variable
import numpy as np

torch.manual_seed(777)   # for reproducibility

xy = np.loadtxt('data-01-test-score.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, x_data, len(x_data))
print(y_data.shape, y_data)

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

# Our hypothesis XW+b
model = nn.Linear(3, 1, bias=True)

# cost criterion
criterion = nn.MSELoss()

# Minimize
optimizer = torch.optim.SGD(model.parameters(), lr=1e-5)
```

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

    if step % 10 == 0:
        print(step, "Cost: ", cost.data.numpy(), "\nPrediction:\n", hypothesis.data.numpy())

# Ask my score
print("Your score will be ", model(Variable(torch.Tensor([[100, 70, 101]]))).data.numpy())
print("Other scores will be ", model(Variable(torch.Tensor([[60, 70, 110], [90, 100, 80]]))).data.numpy())
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-04-3-file_input_linear_regression.py

# Output

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

    if step % 10 == 0:
        print(step, "Cost: ", cost.data.numpy(), "\nPrediction:\n", hypothesis.data.numpy())

# Ask my score
print("Your score will be ", model(Variable(torch.Tensor([[100, 70, 101]]))).data.numpy())
print("Other scores will be ", model(Variable(torch.Tensor([[60, 70, 110], [90, 100, 80]]))).data.numpy())
```

```
Your score will be  [[ 181.73277283]]
Other scores will be  [[ 145.86265564]
 [ 187.23129272]]
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-04-3-file_input_linear_regression.py

```python
# Lab 4 Multi-variable linear regression
import torch
import torch.nn as nn
from torch.autograd import Variable
import numpy as np

torch.manual_seed(777)   # for reproducibility

xy = np.loadtxt('data-01-test-score.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, x_data, len(x_data))
print(y_data.shape, y_data)

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

# Our hypothesis XW+b
model = nn.Linear(3, 1, bias=True)

# cost criterion
criterion = nn.MSELoss()

# Minimize
optimizer = torch.optim.SGD(model.parameters(), lr=1e-5)
```

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

    if step % 10 == 0:
        print(step, "Cost: ", cost.data.numpy(), "\nPrediction:\n", hypothesis.data.numpy())

# Ask my score
print("Your score will be ", model(Variable(torch.Tensor([[100, 70, 101]]))).data.numpy())
print("Other scores will be ", model(Variable(torch.Tensor([[60, 70, 110], [90, 100, 80]]))).data.numpy())
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/pytorch/lab-04-3-file_input_linear_regression.py

With PyTorch 0.2.0_2


# Lab 5

## Logistic (regression) classifier

Sung Kim <hunkim+ml@gmail.com>