colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=jIF12AOVj1Kt

Paused

+ Code + Text All changes saved

T4 RAM Disk

1s

```
import torch
import numpy as np

# Input (temp, rainfall, humidity)
inputs = np.array([[73, 67, 43],
                   [91, 88, 64],
                   [87, 134, 58],
                   [102, 43, 37],
                   [69, 96, 70],
                   [85, 100, 60],
                   [95, 80, 55],
                   [105, 120, 75],
                   [78, 90, 50],
                   [82, 70, 45]], dtype='float32')

# Targets (apples, oranges)
targets = np.array([[56, 70],
                    [81, 101],
                    [119, 133],
                    [22, 37],
                    [103, 119],
                    [98, 110],
                    [88, 95],
                    [115, 140],
                    [76, 85],
                    [65, 75]], dtype='float32')
```

2s

```
inputs = torch.from_numpy(inputs)
```

0s completed at 2:14 PM

colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=jIF12AOVj1Kt

Paused

+ Code + Text All changes saved

T4 RAM Disk

1s

```
import torch
import numpy as np

# Input (temp, rainfall, humidity)
inputs = np.array([[73, 67, 43],
                   [91, 88, 64],
                   [87, 134, 58],
                   [102, 43, 37],
                   [69, 96, 70],
                   [85, 100, 60],
                   [95, 80, 55],
                   [105, 120, 75],
                   [78, 90, 50],
                   [82, 70, 45]], dtype='float32')

# Targets (apples, oranges)
targets = np.array([[56, 70],
                    [81, 101],
                    [119, 133],
                    [22, 37],
                    [103, 119],
                    [98, 110],
                    [88, 95],
                    [115, 140],
                    [76, 85],
                    [65, 75]], dtype='float32')
```

2s

```
inputs = torch.from_numpy(inputs)
```

0s completed at 2:14 PM

colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=jIF12AOVj1Kt

Paused

+ Code + Text All changes saved

T4 RAM Disk

[1]

```
targets = np.array([[56, 70],
                    [81, 101],
                    [119, 133],
                    [22, 37],
                    [103, 119],
                    [98, 110],
                    [88, 95],
                    [115, 140],
                    [76, 85],
                    [65, 75]], dtype='float32')
```

```
inputs = torch.from_numpy(inputs)
targets = torch.from_numpy(targets)
print(inputs)
print(targets)
```

```
tensor([[ 73.,  67.,  43.],
        [ 91.,  88.,  64.],
        [ 87., 134.,  58.],
        [102.,  43.,  37.],
        [ 69.,  96.,  70.],
        [ 85., 100.,  60.],
        [ 95.,  80.,  55.],
        [105., 120.,  75.],
        [ 78.,  90.,  50.],
        [ 82.,  70.,  45.]])
tensor([[ 56.,  70.],
        [ 81., 101.],
```

CODES & DATA ARE AVAILBLE AT
WWW.AISCIENCES.ACADEMY/COURSE-PYTORCH

AISCIENCES

0s completed at 2:14 PM

colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=6DG1zS7iKBCP

Paused

+ Code + Text All changes saved

T4 RAM Disk

[2] 2s

```
tensor([[ 56.,  70.],
        [ 81., 101.],
        [119., 133.],
        [ 22.,  37.],
        [103., 119.],
        [ 98., 110.],
        [ 88.,  95.],
        [115., 140.],
        [ 76.,  85.],
        [ 65.,  75.]])
```

```
w = torch.randn(3,2, requires_grad=True)
b = torch.randn(2, requires_grad=True)
print(w)
print(b)
```

```
tensor([[ 0.3865,  0.6463],
        [ 0.6664,  0.2998],
        [ 1.5070, -0.1798]], requires_grad=True)
tensor([ 0.8740, -0.7764], requires_grad=True)
```

[ ]

[ ]

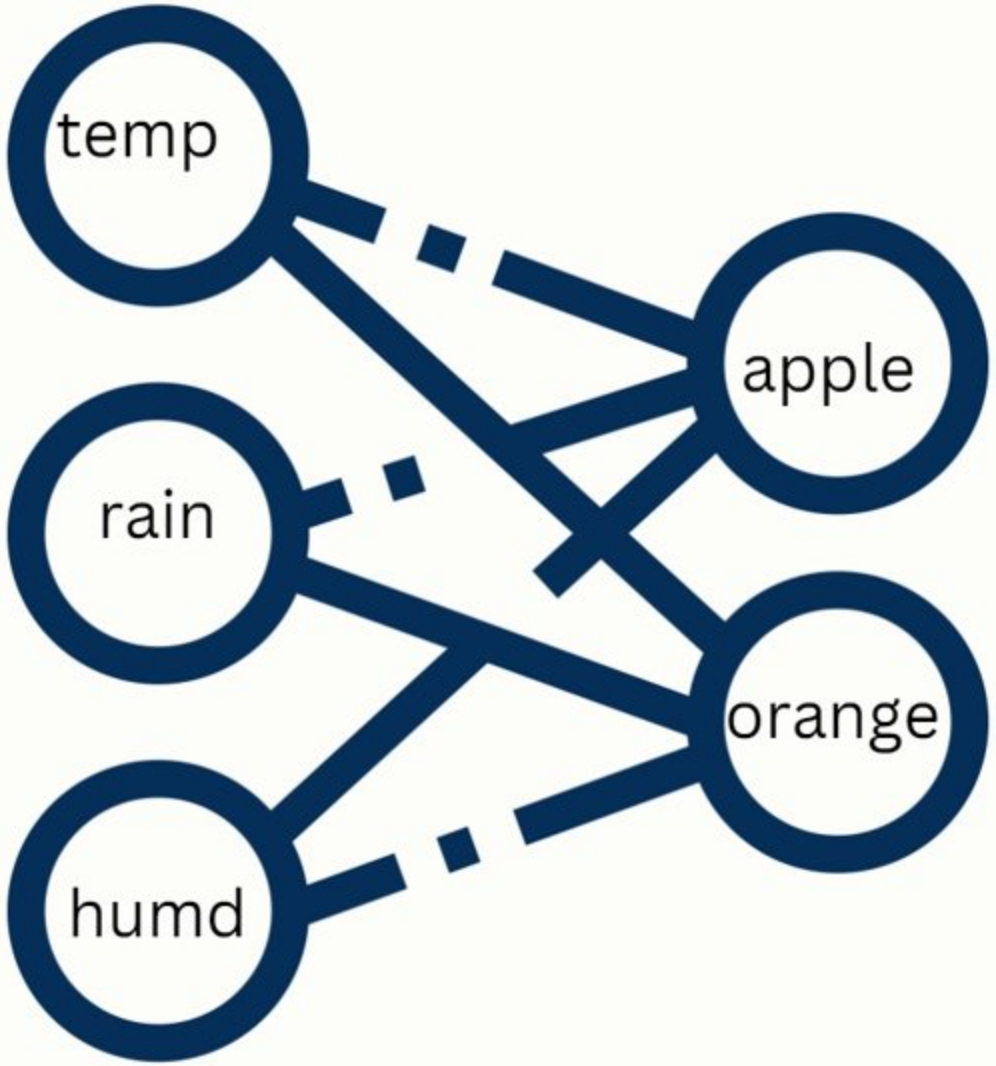
[ ]

[ ]

AISCIENCES

0s completed at 2:14 PM

SIMPLE NEURAL NETWORK



$$X * W + b$$

colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=ecfwVB8-W3JY

Paused

+ Code + Text All changes saved

T4 RAM Disk

```
[ ] w = torch.randn(3,2, requires_grad=True)
    b = torch.randn(2, requires_grad=True)
    print(w)
    print(b)
```

```
tensor([[ 0.3865,  0.6463],
        [ 0.6664,  0.2998],
        [ 1.5070, -0.1798]], requires_grad=True)
tensor([ 0.8740, -0.7764], requires_grad=True)
```

```
def model(x):
  return (x*w)+b
```

[ ]

[ ]

[ ]

[ ]

[ ]

[ ]

[ ]

AISCIENCES

0s completed at 2:14 PM

udemy

colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=ecfwVB8-W3JY

Paused

+ Code + Text

T4 RAM Disk

```
w = torch.randn(3,2, requires_grad=True)
b = torch.randn(2, requires_grad=True)
print(w)
print(b)
```

```
tensor([[ 0.3865,  0.6463],
        [ 0.6664,  0.2998],
        [ 1.5070, -0.1798]], requires_grad=True)
tensor([ 0.8740, -0.7764], requires_grad=True)
```

```
def model(x):
  return (x @ w)+ b
```

[ ]

[ ]

[ ]

[ ]

[ ]

[ ]

[ ]

AISCIENCES

0s completed at 2:14 PM

udemy

colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=ecfwVB8-W3JY

Paused

+ Code + Text All changes saved

T4 RAM Disk

```
w = torch.randn(3,2, requires_grad=True)
b = torch.randn(2, requires_grad=True)
print(w)
print(b)
```

```
tensor([[ 0.3865,  0.6463],
        [ 0.6664,  0.2998],
        [ 1.5070, -0.1798]], requires_grad=True)
tensor([ 0.8740, -0.7764], requires_grad=True)
```

```
def model(x):
  # return (x @ w) + b
  return torch.matmul()
```

[ ]

[ ]

[ ]

[ ]

[ ]

[ ]

[ ]

AISCIENCES

0s completed at 2:14 PM

udemy

colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=ecfwVB8-W3JY

Paused

+ Code + Text All changes saved

T4 RAM Disk

```
[ ] w = torch.randn(3,2, requires_grad=True)
    b = torch.randn(2, requires_grad=True)
    print(w)
    print(b)
```

```
tensor([[ 0.3865,  0.6463],
        [ 0.6664,  0.2998],
        [ 1.5070, -0.1798]], requires_grad=True)
tensor([ 0.8740, -0.7764], requires_grad=True)
```

```
# Defining Model
def model(x):
  # return (x @ w) + b
  return torch.matmul(x,w)+b

#MSE Loss function
def mse(p1, p2):
  diff = p1-p2
  return torch.sum()
```

[ ]

[ ]

[ ]

AISCIENCES

0s completed at 2:14 PM

colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=ecfwVB8-W3JY

Paused

+ Code + Text

T4 RAM Disk

```
[ ] w = torch.randn(3,2, requires_grad=True)
    b = torch.randn(2, requires_grad=True)
    print(w)
    print(b)
```

```
tensor([[ 0.3865,  0.6463],
        [ 0.6664,  0.2998],
        [ 1.5070, -0.1798]], requires_grad=True)
tensor([ 0.8740, -0.7764], requires_grad=True)
```

```
# Defining Model
def model(x):
  # return (x @ w) + b
  return torch.matmul(x,w)+b

#MSE Loss function
def mse(p1, p2):
  diff = p1-p2
  return torch.sum(diff**2)/ diff.numel()
```

[ ]

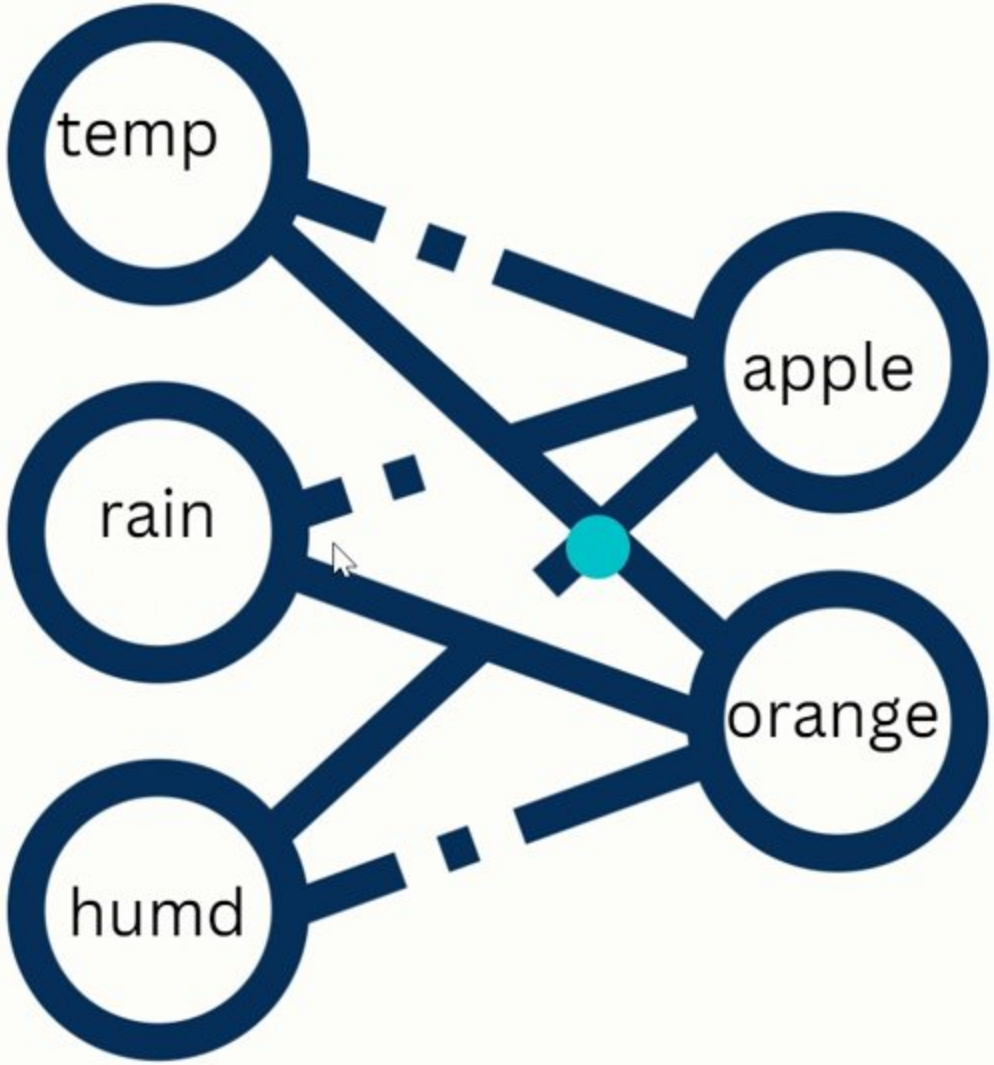
[ ]

[ ]

AISCIENCES

0s completed at 2:14 PM

SIMPLE NEURAL NETWORK



$$X * W + b$$

```
        [ 1.5070, -0.1798]], requires_grad=True)
tensor([ 0.8740, -0.7764], requires_grad=True)
```

```python
# Defining Model
def model(x):
  # return (x @ w) + b
  return torch.matmul(x,w)+b

#MSE Loss function
def mse(p1, p2):
  diff = p1-p2
  return torch.sum(diff**2)/ diff.numel()

# Training step
for i in range(100):
  preds = model(inputs)
  loss = mse(preds, targets)
  loss.backward()

  with torch.no_grad():
    w -= w.grad * 0.00001
    b -= b.grad * 0.00001
    w.grad.zero_()
    b.grad.zero_()
```

```
[ ]
```

colab.google

Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=ecfwVB8-W3JY

Paused

+ Code + Text All changes saved

T4 RAM Disk

```
for i in range(100):
  preds = model(inputs)
  loss = mse(preds, targets)
  loss.backward()

  with torch.no_grad():
    w -= w.grad * 0.00001
    b -= b.grad * 0.00001
    w.grad.zero_()
    b.grad.zero_()

preds = model(inputs)
loss = mse(preds, targets)
print(loss)
```

[ ]

AISCIENCES

0s completed at 2:29 PM

colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=ecfwVB8-W3JY

Paused

+ Code + Text All changes saved

T4 RAM Disk

```
def model(x):
  # return (x @ w) + b
  return torch.matmul(x,w)+b

#MSE Loss function
def mse(p1, p2):
  diff = p1-p2
  return torch.sum(diff**2)/ diff.numel()

# Training step
for i in range(1000):
  preds = model(inputs)
  loss = mse(preds, targets)
  loss.backward()

  with torch.no_grad():
    w -= w.grad * 0.00001
    b -= b.grad * 0.00001
    w.grad.zero_()
    b.grad.zero_()

preds = model(inputs)
loss = mse(preds, targets)
print(loss)
```

1s

```
tensor(74.1262, grad_fn=<DivBackward0>)
```

[ ]

AISCIENCES

0s completed at 2:29 PM

colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=ecfwVB8-W3JY

Paused

+ Code + Text

T4 RAM Disk

```
def model(x):
  # return (x @ w) + b
  return torch.matmul(x,w)+b

#MSE Loss function
def mse(p1, p2):
  diff = p1-p2
  return torch.sum(diff**2)/ diff.numel()

# Training step
for i in range(1000):
  preds = model(inputs)
  loss = mse(preds, targets)
  loss.backward()
  if i%100 == 99:

  with torch.no_grad():
    w -= w.grad * 0.00001
    b -= b.grad * 0.00001
    w.grad.zero_()
    b.grad.zero_()

preds = model(inputs)
loss = mse(preds, targets)
print(loss)
```

```
tensor(74.1262, grad_fn=<DivBackward0>)
```

[ ]

AISCIENCES

0s completed at 2:29 PM

udemy

colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=ecfwVB8-W3JY

Paused

+ Code + Text All changes saved

T4 RAM Disk

```
# Defining Model
def model(x):
  # return (x @ w) + b
  return torch.matmul(x,w)+b

#MSE Loss function
def mse(p1, p2):
  diff = p1-p2
  return torch.sum(diff**2)/ diff.numel()

# Training step
for i in range(1000):
  preds = model(inputs)
  loss = mse(preds, targets)
  loss.backward()
  if i%100 == 99:
    print(loss.item())

  with torch.no_grad():
    w -= w.grad * 0.00001
    b -= b.grad * 0.00001
    w.grad.zero_()
    b.grad.zero_()

preds = model(inputs)
loss = mse(preds, targets)
print(loss)
```


0s completed at 2:30 PM

colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=ecfwVB8-W3JY

Paused

+ Code + Text All changes saved

T4 RAM Disk

0s

```
# Defining Model
def model(x):
  # return (x @ w) + b
  return torch.matmul(x,w)+b

#MSE Loss function
def mse(p1, p2):
  diff = p1-p2
  return torch.sum(diff**2)/ diff.numel()

# Training step
for i in range(1000):
  preds = model(inputs)
  loss = mse(preds, targets)
  loss.backward()
  if i%100 == 99:
    print(loss.item())

  with torch.no_grad():
    w -= w.grad * 0.00001
    b -= b.grad * 0.00001
    w.grad.zero_()
    b.grad.zero_()

preds = model(inputs)
loss = mse(preds, targets)
print(loss)
```

AISCIENCES

0s completed at 2:30 PM

colab.google | Pytorch.ipynb - Colab

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=ecfwVB8-W3JY

Paused

+ Code + Text All changes saved

T4 RAM Disk

0s

```
# Defining Model
def model(x):
  # return (x @ w) + b
  return torch.matmul(x,w)+b

#MSE Loss function
def mse(p1, p2):
  diff = p1-p2
  return torch.sum(diff**2)/ diff.numel()

# Training step
for i in range(1000):
  preds = model(inputs)
  loss = mse(preds, targets)
  loss.backward()
  if i%100 == 99:
    print(loss.item())

  with torch.no_grad():
    w -= w.grad * 0.00001
    b -= b.grad * 0.00001
    w.grad.zero_()
    b.grad.zero_()

preds = model(inputs)
loss = mse(preds, targets)
```

CODES & DATA ARE AVAILBLE AT
WWW.AISCIENCES.ACADEMY/COURSE-PYTORCH

AISCIENCES

0s completed at 2:30 PM