PyTorch Tutorials

PyTorch

0.4.0

Search docs

BEGINNER TUTORIALS

Deep Learning with PyTorch: A 60 Minute Blitz

PyTorch for former Torch users

Learning PyTorch with Examples

Tensors

Autograd

nn module

Examples

Tensors

Warm-up: numpy

PyTorch: Tensors

Autograd

nn module

Transfer Learning tutorial

Data Loading and Processing Tutorial

Deep Learning for NLP with Pytorch

INTERMEDIATE TUTORIALS

Classifying Names with a Character-Level RNN

Generating Names with a Character-Level RNN

Translation with a Sequence to Sequence Network and Attention

Reinforcement Learning (DQN) tutorial

Writing Distributed Applications with PyTorch

Warm-up: numpy

A fully-connected ReLU network with one hidden layer and no biases, trained to predict y from x using Euclidean error.

This implementation uses numpy to manually compute the forward pass, loss, and backward pass.

A numpy array is a generic n-dimensional array; it does not know anything about deep learning or gradients or computational graphs, and is just a way to perform generic numeric computations.

```python
import numpy as np

# N is batch size; D_in is input dimension;
# H is hidden dimension; D_out is output dimension.
N, D_in, H, D_out = 64, 1000, 100, 10

# Create random input and output data
x = np.random.randn(N, D_in)
y = np.random.randn(N, D_out)

# Randomly initialize weights
w1 = np.random.randn(D_in, H)
w2 = np.random.randn(H, D_out)

learning_rate = 1e-6
for t in range(500):
    # Forward pass: compute predicted y
    h = x.dot(w1)
    h_relu = np.maximum(h, 0)
    y_pred = h_relu.dot(w2)

    # Compute and print loss
    loss = np.square(y_pred - y).sum()
    print(t, loss)

    # Backprop to compute gradients of w1 and w2 with respect to loss
    grad_y_pred = 2.0 * (y_pred - y)
    grad_w2 = h_relu.T.dot(grad_y_pred)
    grad_h_relu = grad_y_pred.dot(w2.T)
    grad_h = grad_h_relu.copy()
    grad_h[h < 0] = 0
    grad_w1 = x.T.dot(grad_h)

    # Update weights
    w1 -= learning_rate * grad_w1
    w2 -= learning_rate * grad_w2
```

Total running time of the script: (0 minutes 0.000 seconds)

Download Python source code: two_layer_net_numpy.py

Download Jupyter notebook: two_layer_net_numpy.ipynb

Gallery generated by Sphinx-Gallery

Previous Next

Built with Sphinx using a theme provided by Read the Docs.