



Watson Studio democratizes machine learning and deep learning to accelerate infusion of AI in your business to drive innovation. Watson Studio provides a suite of tools and a collaborative environment for data scientists, developers and domain experts.

(http://cocl.us/pytorch_link_top)



Activation Functions

Table of Contents

In this lab, you will cover logistic regression by using PyTorch.

- [Logistic Function](#)
- [Tanh](#)
- [Relu](#)
- [Compare Activation Functions](#)

Estimated Time Needed: **15 min**

We'll need the following libraries

In [1]:

```
# Import the libraries we need for this lab

import torch.nn as nn
import torch

import matplotlib.pyplot as plt
torch.manual_seed(2)
```

Out[1]:

```
<torch._C.Generator at 0x7fbb7fd6f550>
```

Logistic Function

Create a tensor ranging from -10 to 10:

In [2]:

```
# Create a tensor

z = torch.arange(-10, 10, 0.1,).view(-1, 1)
```

When you use sequential, you can create a sigmoid object:

In [3]:

```
# Create a sigmoid object

sig = nn.Sigmoid()
```

Apply the element-wise function Sigmoid with the object:

In [4]:

```
# Make a prediction of sigmoid function

yhat = sig(z)
```

Plot the results:

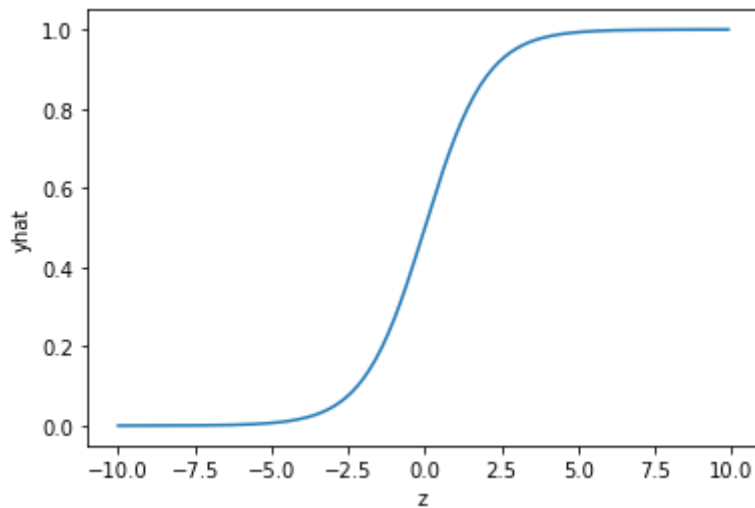
In [5]:

```
# Plot the result

plt.plot(z.detach().numpy(),yhat.detach().numpy())
plt.xlabel('z')
plt.ylabel('yhat')
```

Out[5]:

Text(0, 0.5, 'yhat')



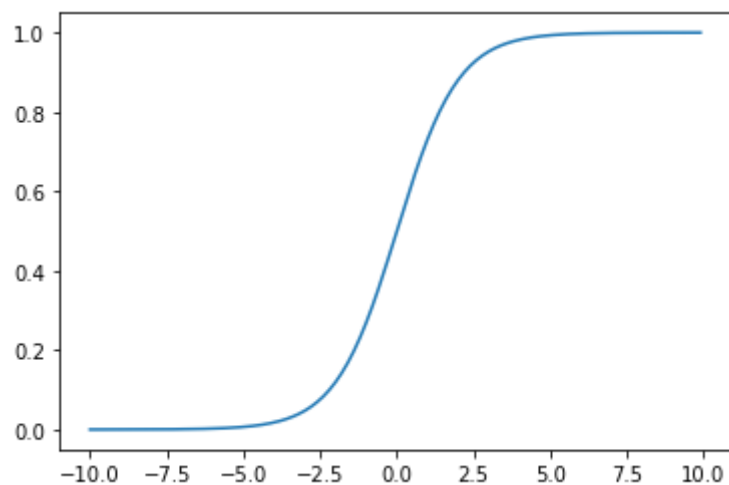
For custom modules, call the sigmoid from the torch (`nn.functional` for the old version), which applies the element-wise sigmoid from the function module and plots the results:

In [6]:

```
# Use the build in function to predict the result

yhat = torch.sigmoid(z)
plt.plot(z.numpy(), yhat.numpy())

plt.show()
```



Tanh

When you use sequential, you can create a tanh object:

In [7]:

```
# Create a tanh object

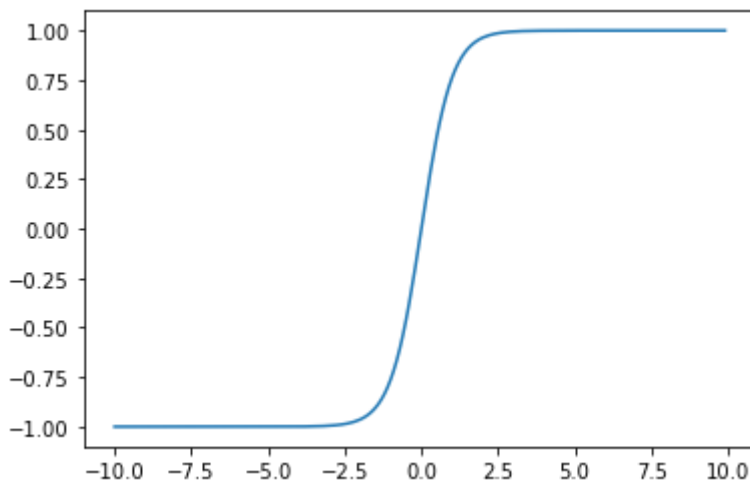
TANH = nn.Tanh()
```

Call the object and plot it:

In [8]:

```
# Make the prediction using tanh object

yhat = TANH(z)
plt.plot(z.numpy(), yhat.numpy())
plt.show()
```

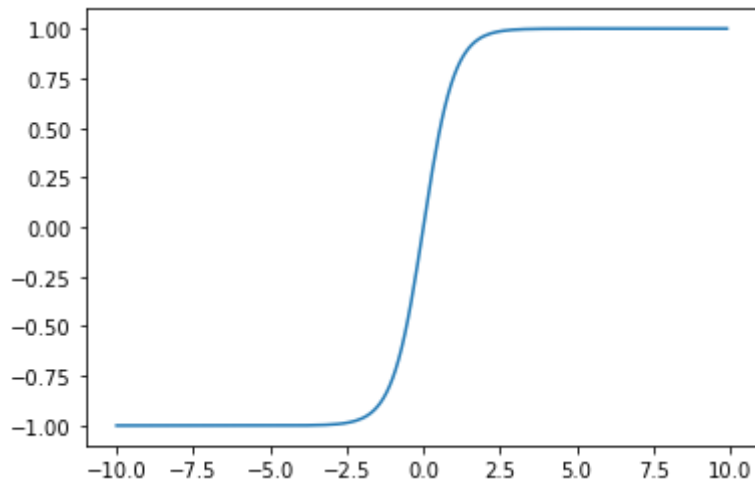


For custom modules, call the Tanh object from the torch (nn.functional for the old version), which applies the element-wise sigmoid from the function module and plots the results:

In [9]:

```
# Make the prediction using the build-in tanh object
```

```
yhat = torch.tanh(z)  
plt.plot(z.numpy(), yhat.numpy())  
plt.show()
```



Relu

When you use sequential, you can create a Relu object:

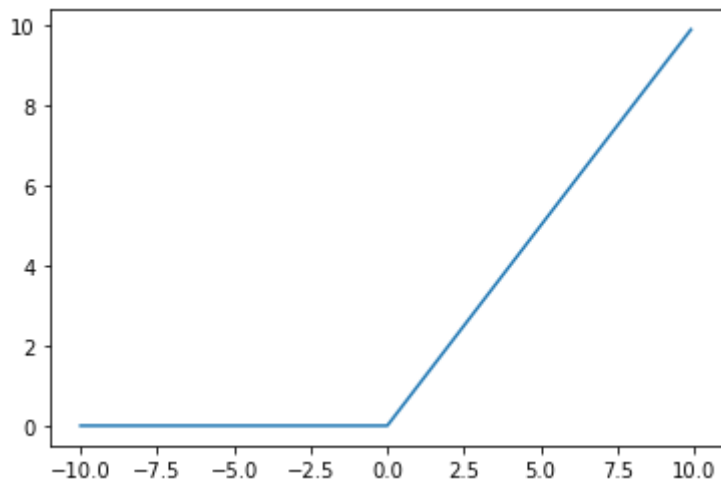
In [10]:

```
# Create a relu object and make the prediction
```

```
RELU = nn.ReLU()  
yhat = RELU(z)  
plt.plot(z.numpy(), yhat.numpy())
```

Out[10]:

```
[<matplotlib.lines.Line2D at 0x7fbb77cbfda0>]
```

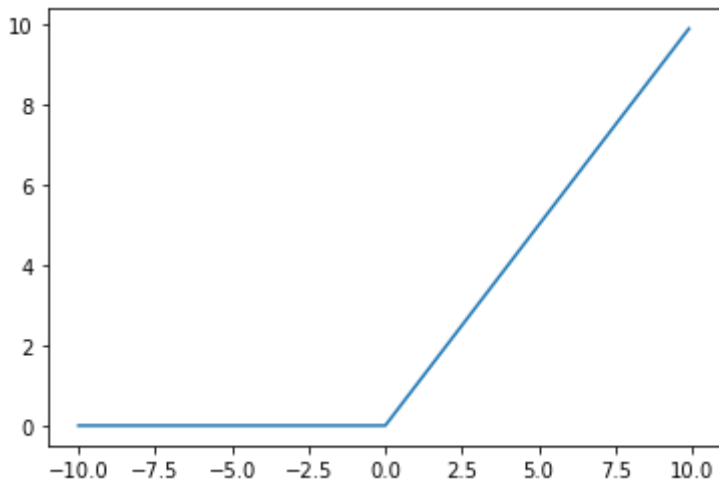


For custom modules, call the relu object from the nn.functional, which applies the element-wise sigmoid from the function module and plots the results:

In [11]:

```
# Use the build-in function to make the prediction
```

```
yhat = torch.relu(z)  
plt.plot(z.numpy(), yhat.numpy())  
plt.show()
```



Compare Activation Functions

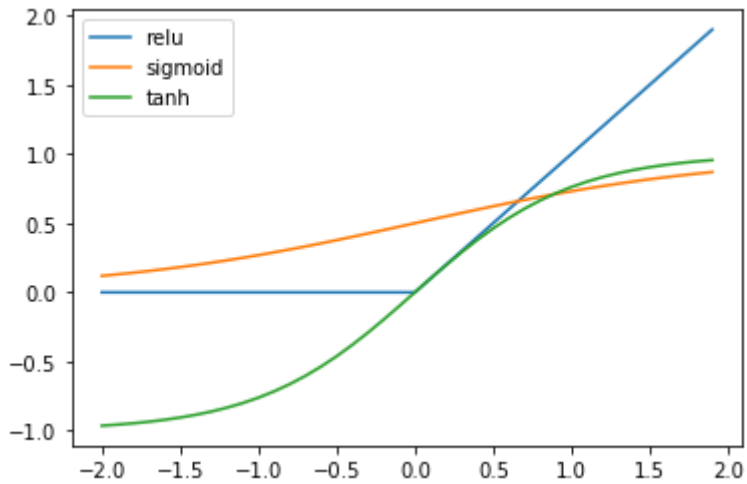
In [12]:

```
# Plot the results to compare the activation functions

x = torch.arange(-2, 2, 0.1).view(-1, 1)
plt.plot(x.numpy(), torch.relu(x).numpy(), label='relu')
plt.plot(x.numpy(), torch.sigmoid(x).numpy(), label='sigmoid')
plt.plot(x.numpy(), torch.tanh(x).numpy(), label='tanh')
plt.legend()
```

Out[12]:

<matplotlib.legend.Legend at 0x7fbb77dab0b8>



Practice

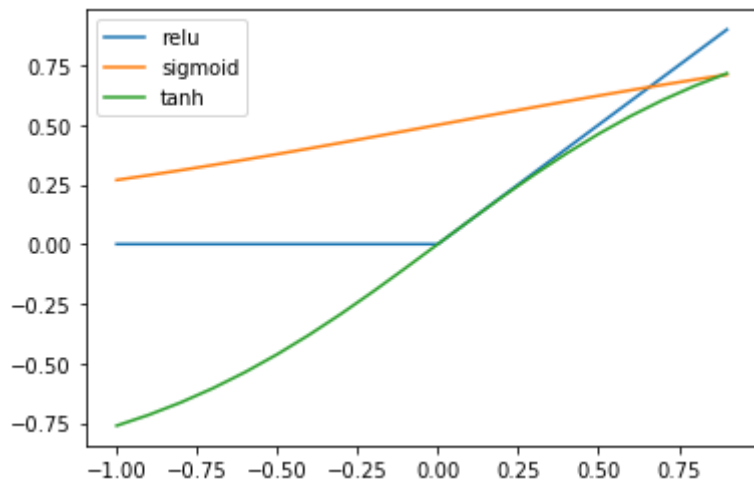
Compare the activation functions with a tensor in the range $(-1, 1)$

In [14]:

```
# Practice: Compare the activation functions again using a tensor in the range (-1, 1)
x = torch.arange(-1, 1, 0.1).view(-1, 1)
plt.plot(x.numpy(), torch.relu(x).numpy(), label = 'relu')
plt.plot(x.numpy(), torch.sigmoid(x).numpy(), label = 'sigmoid')
plt.plot(x.numpy(), torch.tanh(x).numpy(), label = 'tanh')
plt.legend()
```

Out[14]:


<matplotlib.legend.Legend at 0x7fbb7bbe7dd8>



Double-click **here** for the solution.


Get IBM Watson Studio free of charge!

Build and train AI & machine learning models, prepare and analyze data – all in a flexible, hybrid cloud environment. Get IBM Watson Studio Lite Plan free of charge.




Learn

Get started or get better with built-in learning.



Create

Use the best of open source tooling with IBM innovation.



Collaborate

Work smarter using community, work faster with your team.

[Sign Up For a Free Trial](#)

(<http://cocl.us/pytorch> link bottom)

About the Authors:

[Joseph Santarcangelo](https://www.linkedin.com/in/joseph-s-50398b136/) (<https://www.linkedin.com/in/joseph-s-50398b136/>) has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Other contributors: [Michelle Carey](https://www.linkedin.com/in/michelleccarey/) (<https://www.linkedin.com/in/michelleccarey/>), [Mavis Zhou](https://www.linkedin.com/in/jiahui-mavis-zhou-a4537814a) (www.linkedin.com/in/jiahui-mavis-zhou-a4537814a).

Copyright © 2018 cognitiveclass.ai (cognitiveclass.ai?utm_source=bducopyrightlink&utm_medium=dswb&utm_campaign=bdu). This notebook and its source code are released under the terms of the [MIT License](https://bigdatauniversity.com/mit-license/) (<https://bigdatauniversity.com/mit-license/>).