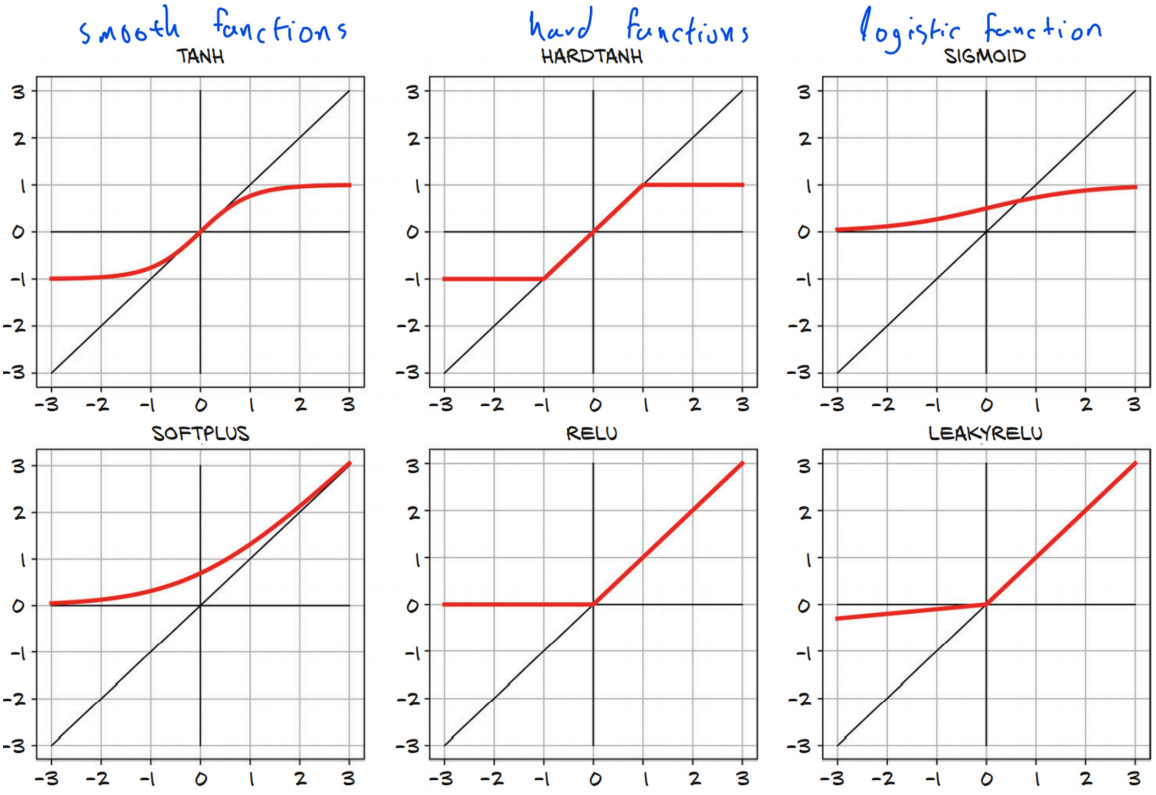
Chapter 6

Friday, April 23, 2021 12:23 PM

6.1 Artificial neurons

At the core of deep learning are neural networks: mathematical entities capable of representing complicated functions through a composition of simpler functions.
Neural networks do not have that same property of a convex error surface. There's no single right answer for each parameter we're attempting to approximate
A big part of the reason neural networks have non-convex error surfaces is due to the activation function.
Different activation functions:



The nonlinearity allows the overall network to approximate more complex functions.

The following are true for the functions:

- They have at least one sensitive range, where nontrivial changes to the input result in a corresponding nontrivial change to the output. This is needed for training.
- Many of them have an insensitive (or saturated) range, where changes to the input result in little or no change to the output.

What makes using deep neural networks so attractive is that it saves us from worrying too much about the exact function that represents our data

6.2. The PyTorch nn module

PyTorch has a whole submodule dedicated to neural networks, called torch.nn.
Those building blocks are called modules in PyTorch parlance. A PyTorch module is a Python class deriving from the nn.Module base class.

Using __call__ rather than forward

All PyTorch-provided subclasses of nn.Module have their __call__ method defined.

```
y = model(x)              <-- Correct!
y = model.forward(x)      <-- Silent error. Don't do it!
```

6.3. Finally a neural network

The first linear + activation layer is commonly referred to as a hidden layer for historical reasons, since its outputs are not observed directly but fed into the output layer.

When inspecting parameters of a model made up of several submodules, it is handy to be able to identify parameters by name. There's a method for that, called named_parameters.

```
from collections import OrderedDict

seq_model = nn.Sequential(OrderedDict([
    ('hidden_linear', nn.Linear(1, 8)),
    ('hidden_activation', nn.Tanh()),
    ('output_linear', nn.Linear(8, 1))
]))
```