Advanced Python for Neuroscientists
Lecture 6: Neural network II

Summer 2022

Princeton Neuroscience Institute
Instructors: Yisi Zhang & Bichan Wu

July 14, 2022

Recap

Lecture 5

- Feedforward neural network
- Gradient descent

Outline

- Backpropagation
- Stochastic Gradient Descent
- Application

6.1 Backpropagation

Cost function of a neural network

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta^k(x^{(i)})) + (1 - y_k^{(i)}) \log(1 - h_\Theta^k(x^{(i)}))$$
$$+ \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{j,i}^{(l)})^2$$

How do we minimize this cost function?

6.1 Backpropagation

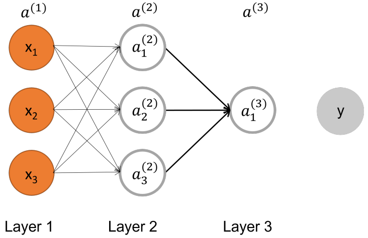
"Backpropagation" in neural network is for minimizing the cost function and finding the optimal set of Θ.

To use gradient descent, we need to compute the partial derivative $\frac{\partial}{\partial \Theta_{i,j}^{(l)}} J(\Theta)$.

6.1 Backpropagation

Forward propagation:

$$
\begin{aligned}
a^{(1)} &= x \\
z^{(2)} &= \Theta^{(1)} a^{(1)} \\
a^{(2)} &= g(z^{(2)}) \\
z^{(3)} &= \Theta^{(2)} a^{(2)} \\
a^{(3)} &= g(z^{(3)}) = h_{\Theta}(x)
\end{aligned}
$$



6.1 Backpropagation

Backpropagation: Layer 3 to Layer 2

$$J(\Theta) = -y\log(h_\Theta(x)) - (1-y)\log(1-h_\Theta(x))$$

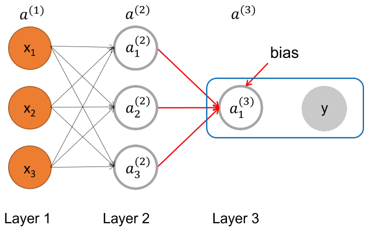
Chain rule

$$h_\Theta(x) = g(z^{(3)}) = a^{(3)}$$

$$z^{(3)} = \Theta^{(2)} a^{(2)}$$

$$\frac{\partial J}{\partial \Theta^{(2)}} = \frac{\partial J}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial \Theta^{(2)}}$$

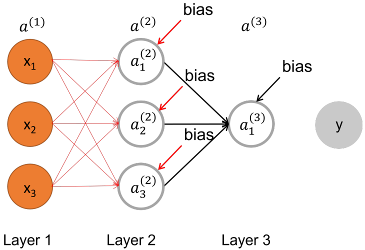
$$= (a^{(3)} - y) a^{(2)T}$$



6.1 Backpropagation

Backpropagation: Layer 2 to Layer 1

$$
\begin{aligned}
h_\Theta(x) &= g(z^{(3)}) = a^{(3)} \\
z^{(3)} &= \Theta^{(2)} a^{(2)} \\
a^{(2)} &= g(z^{(2)}) \\
z^{(2)} &= \Theta^{(1)} a^{(1)} \\
\frac{\partial J}{\partial \Theta^{(1)}} &= \frac{\partial J}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial a^{(2)}} \\
&\quad \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial \Theta^{(1)}} \\
&= \Theta^{(2)T} (a^{(3)} - y) \\
&\quad (a^{(2)} - a^{(2)2}) a^{(1)T}
\end{aligned}
$$



6.2 Stochastic gradient descent

Learning with large datasets

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log(h_\theta(x^{(i)})) + (1 - y^{(i)})\log(1 - h_\theta(x^{(i)}))$$

Gradient descent:
Repeat{
$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$
for every $j = 0, ..., n$
}
We compute the gradient with all samples and then update θ. This can be very time consuming when m is large.

6.2 Stochastic gradient descent

Instead, we can update θ with gradient contributed by each sample:

1. Randomly shuffle the training set.
2. Repeat{

    for i=1:m{

    $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$

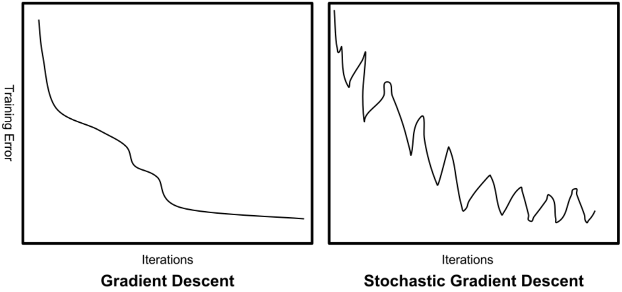
    for every $j = 0, \ldots, n$

    }

   }

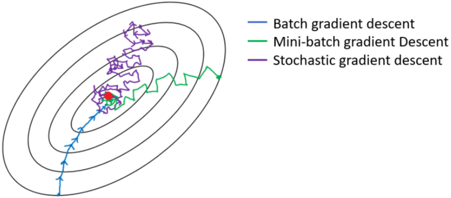
6.2 Stochastic gradient descent

The cost function does not monotonically decrease with SGD.



6.2 Stochastic gradient descent

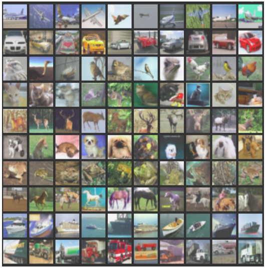
More often, we use a batch of training samples (between 1 and m) to update θ each time. This is called mini-batch gradient descent.



The batch size is a hyper-parameter for us to determine.
The complete pass of the entire training set is called an epoch, whose number is also a hyper-parameter.

6.3 Application

Data collection
Explore the open datasets: MNIST, CIFAR-10, SVHN, IMDB Reviews ...

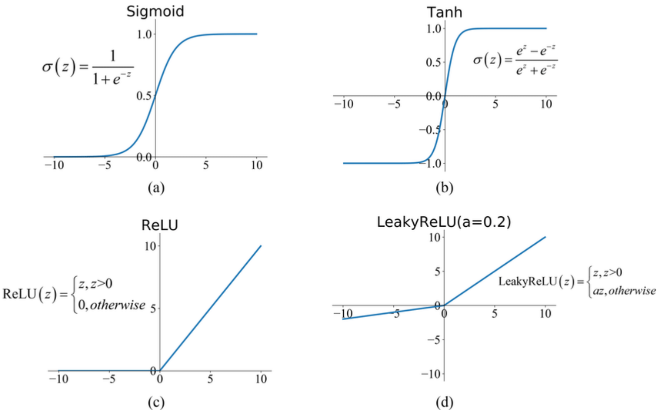


6.3 Application

Data split

6.3 Application

Activation functions



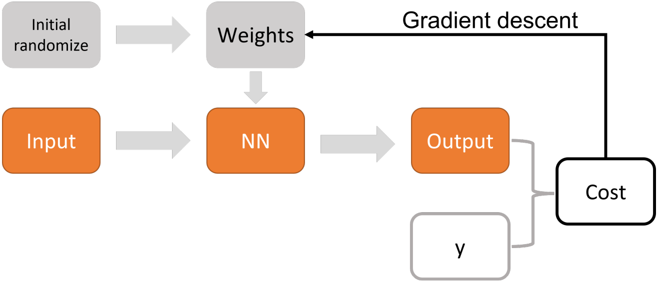
6.3 Application

Considerations before training

Goal: e.g. classify neuron vs. non-neuron			
Data	Images	Number?	e.g. 10,000
Input	Image	Size?	e.g. (64,64,3)
Output	Categorical	No. categories?	e.g. y=0 or 1
Architecture	Neural net	Shallow/Deep	e.g. 3 layers...
Activation function	Sigmoid, tanh, ReLU,...		e.g. ReLU, leakyReLU
Cost function	Cross entropy, MSE,... (depending on the goal)		

6.3 Application

Train the network



Estimate accuracy in the test set.

Homework

- Make sure you understand all the exercises above
- Run through the codes here that should replicate all the figures
  https://github.com/yisiszhang/AdvancedPython/blob/main/colab/Lecture6.ipynb
- Try different neural network layouts
- Try different learning rates