Stats 102B - Week 3, Lecture 2

Miles Chen, PhD

Department of Statistics

Week 3 Wednesday

UCLA

Copyright Miles Chen. For personal use only. Do not distribute.

Section 1

Quick Review

Copyright Miles Chen. For personal use only. Do not distribute.

Quick review of Gradient Descent:

- We are trying to find the values of $\mathbf{w}$ that will minimize the function $F(\mathbf{w})$
- Gradient Descent is an iterative algorithm.
- If we begin at some arbitrary starting location $\mathbf{w}_{(t=0)}$ (w at time 0), how do we select our next location, $\mathbf{w}_{(t=1)}$?

Copyright Miles Chen. For personal use only. Do not distribute.

Quick review of Gradient Descent:

- We are trying to find the values of $\mathbf{w}$ that will minimize the function $F(\mathbf{w})$
- Gradient Descent is an iterative algorithm.
- If we begin at some arbitrary starting location $\mathbf{w}_{(t=0)}$ (w at time 0), how do we select our next location, $\mathbf{w}_{(t=1)}$?

$$\mathbf{w}_{(n+1)} = \mathbf{w}_{(n)} - \gamma \nabla F(\mathbf{w}_{(n)})$$

Copyright Miles Chen. For personal use only. Do not distribute.

Section 2

Introducing Neural Networks

Copyright Miles Chen. For personal use only. Do not distribute.

Required Viewing: Video Resources

The following series of videos are excellent introductions to Neural Networks.

Our lectures will largely follow the example in the Welch Labs series.

Welch Labs: Neural Networks Demystified:
https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

3Blue1Brown: Neural Networks:
https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

Copyright Miles Chen. For personal use only. Do not distribute.

A Disclaimer

In this class, we will explore only the simplest kind of neural network.

Neural Networks have been around since the 1970's and 80's and have gained great popularity in the last decade.

Many advances have been made in neural networks in the past few years, with new kinds of layers and activation function. Neural networks are frequently used in image recognition systems and power many AI technologies.

In our class we will learn only the simplest kind of neural network. I like to introduce them in this class because I think they are really neat, and we use gradient descent.

Copyright Miles Chen. For personal use only. Do not distribute.

What is a Neural Network?

In an overly simplistic sense, a neural network is a design for a function: It takes an input, does a series of computations and produces and output.

The inspiration for a neural network's design is the brain. The brain consists of billions of neurons. Each neuron is connected by synapses to thousands of other neurons.

In a biological brain, a neuron receives input signals. Depending on the signal and the neuron's function, the neuron may send a pulse (action potential) to other neurons.

An artificial neural network will consist of nodes that receive input values. Depending on the input value and the node's activation function, the node will send a new output value to other nodes.

Copyright Miles Chen. For personal use only. Do not distribute.

Silly Example (based on Welch Labs Video)

We begin with a very simple example.

Our input matrix $\mathbf{X}$ has two columns: Hours of sleep, and hour spent studying.

The output we are trying to predict is $\mathbf{y}$, the exam score.

The training data will have N rows. In the video, the $N = 3$.

$$\mathbf{X} = \begin{bmatrix} 3 & 5 \\ 5 & 1 \\ 10 & 2 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} 75 \\ 82 \\ 93 \end{bmatrix}$$

Copyright Miles Chen. For personal use only. Do not distribute.

Creating our Neural Network

All neural networks begin with an input layer and end with an output layer.

We have two variables (two columns) in our input matrix $\mathbf{X}$. So our input layer will have two nodes, one for each variable.

The output is one variable, the exam score. So the output layer will have only one node.

A neural network will also have one or more hidden layers. For our simple example, we will have one hidden layer with three nodes. (The choice of one layer with three nodes is a bit arbitrary. We chose small numbers to keep our introductory example simple.)

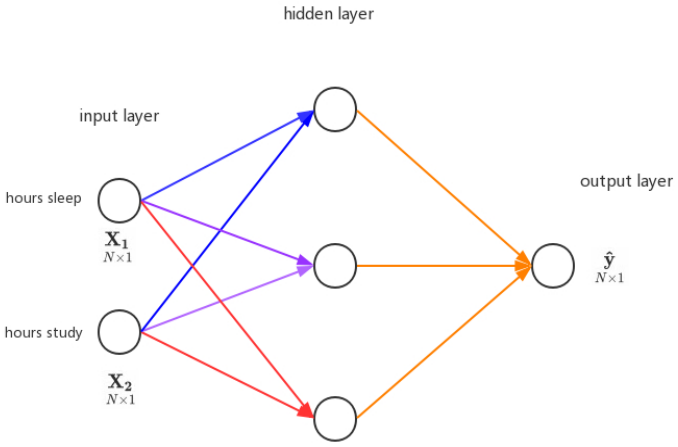
The arrangement of layers is:

$$\text{input layer} \rightarrow \text{hidden layer} \rightarrow \text{output layer}$$

Each node in each layer is connected to all nodes in the next layer.

Copyright Miles Chen. For personal use only. Do not distribute.

Diagram



Copyright Miles Chen. For personal use only. Do not distribute.

Side note: Deep Learning

In our example, we have one hidden layer. A neural network can have multiple hidden layers.

If a network has many hidden layers (100+ layers), we might call it a deep neural network, or a deep learning model.

Deep neural networks have been shown to be incredibly powerful.

For example, deep neural networks have been applied to the ImageNet dataset which consists of 14 million images with thousands of possible categorical labels. In 2015, a deep convolutional neural network surpassed the human error rate for labeling images, and performance continues to improve.

Some neat stuff you can do with neural networks:
https://www.youtube.com/watch?v=0OT3UIXZztE

Copyright Miles Chen. For personal use only. Do not distribute.

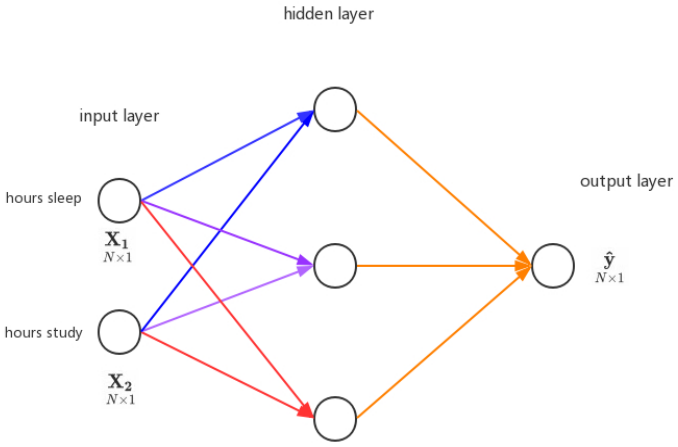
Side Note:

Neural networks are great for predictive performance.

They are not good for interpretability. The function of each node often remains a mystery.

Copyright Miles Chen. For personal use only. Do not distribute.

Back to our network



Copyright Miles Chen. For personal use only. Do not distribute.

Forward Propagation

In our neural network, we have:

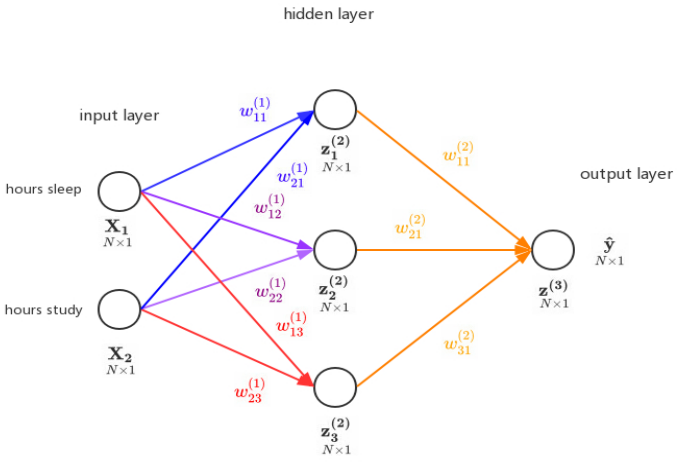
- 6 connections between the input layer and the hidden layer
- 3 connections between the hidden layer and the output layer

Each of these connections has an associated weight, which effectively informs how 'important' a connection is.

Training our model consists of figuring out the weights of these connections.

Copyright Miles Chen. For personal use only. Do not distribute.

Forward propagation - labeled connections



Copyright Miles Chen. For personal use only. Do not distribute.

The Weight matrices

We can organize the weights for our neural network in two matrices:

- $\mathbf{W}^{(1)}$ will be the weight matrix from the first layer $\mathbf{X}$ to the hidden layer
  - it has two rows (one for each node in layer 1)
  - it has three columns (one for each node in layer 2)
- $\mathbf{W}^{(2)}$ will be the weight matrix from the second layer (hidden layer) to the output layer
  - it has three rows (one for each node in layer 2)
  - it has one column (one for each node in final layer)

$$\mathbf{W}^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \end{bmatrix}$$

$$\mathbf{W}^{(2)} = \begin{bmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \\ w_{31}^{(2)} \end{bmatrix}$$

Copyright Miles Chen. For personal use only. Do not distribute.

Forward Propagation

When we multiply $\mathbf{X}$ by $\mathbf{W}^{(1)}$ it will produce a new matrix $\mathbf{z}^{(2)}$

$$\underset{N\times 2}{\mathbf{X}}\ \underset{2\times 3}{\mathbf{W}^{(1)}} = \underset{N\times 3}{\mathbf{z}^{(2)}}$$

$$\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \times \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \end{bmatrix} =$$

$$\begin{bmatrix} x_{11}w_{11}^{(1)} + x_{12}w_{21}^{(1)} & x_{11}w_{12}^{(1)} + x_{12}w_{22}^{(1)} & x_{11}w_{13}^{(1)} + x_{12}w_{13}^{(1)} \\ x_{21}w_{11}^{(1)} + x_{22}w_{21}^{(1)} & x_{21}w_{12}^{(1)} + x_{22}w_{22}^{(1)} & x_{21}w_{13}^{(1)} + x_{22}w_{13}^{(1)} \\ \vdots & \vdots & \vdots \\ x_{N1}w_{11}^{(1)} + x_{N2}w_{21}^{(1)} & x_{N1}w_{12}^{(1)} + x_{N2}w_{22}^{(1)} & x_{N1}w_{13}^{(1)} + x_{N2}w_{13}^{(1)} \end{bmatrix}$$

Copyright Miles Chen. For personal use only. Do not distribute.

Forward Propagation

After producing our matrix $\mathbf{z}^{(2)}$ for the hidden layer, we apply a function called an activation function to each element in $\mathbf{z}$.

Thus the function will not change the shape of the matrix $\mathbf{z}$, only the values inside.

$$f(\underset{N\times 3}{\mathbf{z}^{(2)}}) = \underset{N\times 3}{\mathbf{a}^{(2)}}$$

Copyright Miles Chen. For personal use only. Do not distribute.

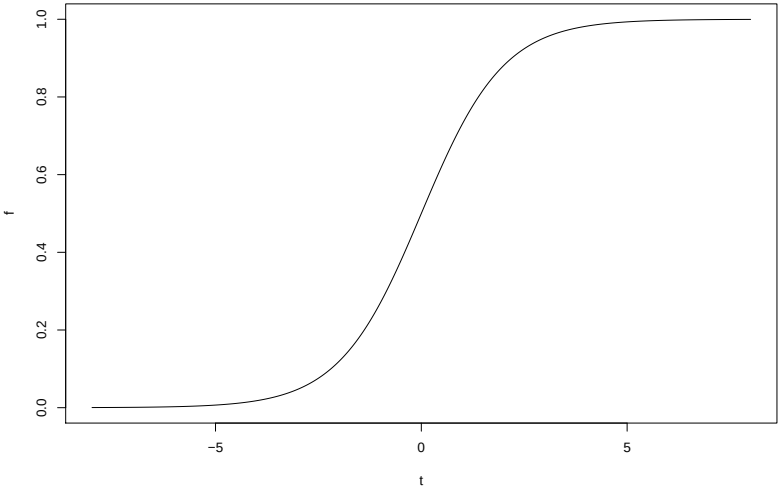
Forward Propagation - the Sigmoid activation function

We will use a sigmoid activation function. This will 'squish' values in the domain $(-\infty, +\infty)$ to the range $(0, 1)$.

$$f(t) = \frac{1}{1 + e^{-t}}$$

Copyright Miles Chen. For personal use only. Do not distribute.

Forward Propagation - the Sigmoid activation function



Copyright Miles Chen. For personal use only. Do not distribute.

Side note: Activation functions

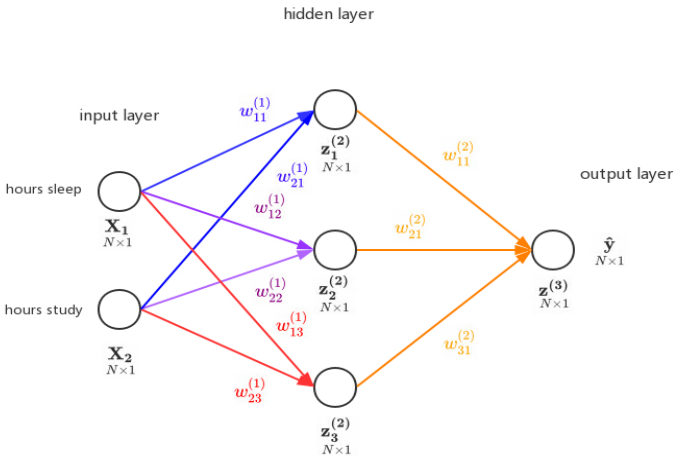
Recently, the sigmoid activation function has fallen out of favor. Neural networks today seem mostly employ a rectified linear unit (ReLu) activation function.

Other possible activation functions include tanh and leaky relu. See: https://en.wikipedia.org/wiki/Activation_function

We will continue to use the sigmoid function because: it's used in the video series, it is differentiable, and it has historic popularity.

Copyright Miles Chen. For personal use only. Do not distribute.

Forward propagation - labeled connections



Copyright Miles Chen. For personal use only. Do not distribute.

Forward Propagation

Apply the activation function to $\mathbf{z}^{(2)}$ and get $\mathbf{a}^{(2)}$. Multiply $\mathbf{a}^{(2)}$ by our second weight matrix $\mathbf{W}^{(2)}$ to get $\mathbf{z}^{(3)}$.

$$\underset{N\times 3}{\mathbf{a}^{(2)}}\underset{3\times 1}{\mathbf{W}^{(2)}} = \underset{N\times 1}{\mathbf{z}^{(3)}}$$

$$\begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} \\ a_{21}^{(2)} & a_{22}^{(2)} & a_{23}^{(2)} \\ \vdots & \vdots & \vdots \\ a_{N1}^{(2)} & a_{N2}^{(2)} & a_{N3}^{(2)} \end{bmatrix} \times \begin{bmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \\ w_{31}^{(2)} \end{bmatrix} = \begin{bmatrix} a_{11}^{(2)} w_{11}^{(2)} + a_{12}^{(2)} w_{21}^{(2)} + a_{13}^{(2)} w_{31}^{(2)} \\ a_{21}^{(2)} w_{11}^{(2)} + a_{22}^{(2)} w_{21}^{(2)} + a_{23}^{(2)} w_{31}^{(2)} \\ \vdots \\ a_{N1}^{(2)} w_{11}^{(2)} + a_{N2}^{(2)} w_{21}^{(2)} + a_{N3}^{(2)} w_{31}^{(2)} \end{bmatrix}$$

Copyright Miles Chen. For personal use only. Do not distribute.

Activate

Our last step is to apply the activation function to $\mathbf{z}^{(3)}$.

$$f(\underset{N\times 1}{\mathbf{z}^{(3)}}) = \underset{N\times 1}{\hat{\mathbf{y}}}$$

Copyright Miles Chen. For personal use only. Do not distribute.

Summary

$$\underset{N\times 2}{\mathbf{X}}\ \underset{2\times 3}{\mathbf{W}^{(1)}} = \underset{N\times 3}{\mathbf{z}^{(2)}}$$

$$f(\underset{N\times 3}{\mathbf{z}^{(2)}}) = \underset{N\times 3}{\mathbf{a}^{(2)}}$$

$$\underset{N\times 3}{\mathbf{a}^{(2)}}\underset{3\times 1}{\mathbf{W}^{(2)}} = \underset{N\times 1}{\mathbf{z}^{(3)}}$$

$$f(\underset{N\times 1}{\mathbf{z}^{(3)}}) = \underset{N\times 1}{\hat{\mathbf{y}}}$$

$$\hat{\mathbf{y}} = f(f(\mathbf{X}\mathbf{W}^{(1)})\mathbf{W}^{(2)})$$

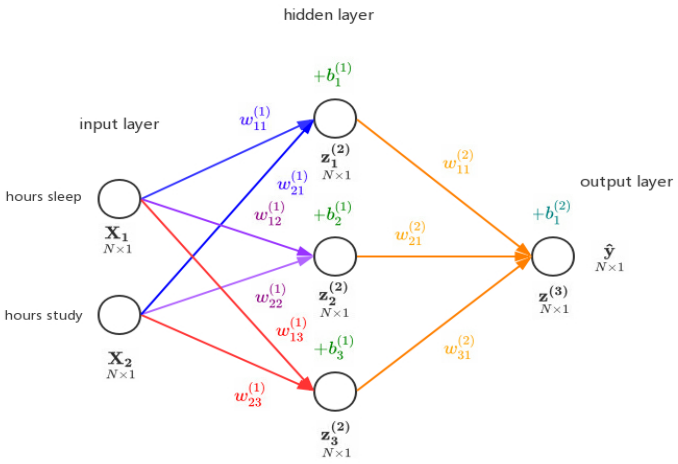
Copyright Miles Chen. For personal use only. Do not distribute.

Bias Terms

The Welch Labs video did not include bias terms. Bias terms simply add a constant at each node before applying the activation function.

Copyright Miles Chen. For personal use only. Do not distribute.

Neural Network with Biases labeled connections



Copyright Miles Chen. For personal use only. Do not distribute.

Bias Terms

There is one bias term for each node, so our matrix of Biases $\mathbf{B}^{(1)}$ for the hidden layer is 3×1

In order for the dimensionality to work, I have to multiply the Bias matrix by a matrix of ones, which we'll call $\mathbf{J}$.

Thus:

$$\underset{N\times 2}{\mathbf{X}}\ \underset{2\times 3}{\mathbf{W}^{(1)}} + \underset{N\times 1}{\mathbf{J}}\ \underset{1\times 3}{\mathbf{B}^{(1)^T}} = \underset{N\times 3}{\mathbf{z}^{(2)}}$$

Copyright Miles Chen. For personal use only. Do not distribute.

Bias Terms

$$\underset{N\times 2}{\mathbf{X}}\ \underset{2\times 3}{\mathbf{W}^{(1)}} + \underset{N\times 1}{\mathbf{J}}\ \underset{1\times 3}{\mathbf{B}^{(1)^T}} = \underset{N\times 3}{\mathbf{z}^{(2)}}$$

$$\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \times \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \times \begin{bmatrix} b_1^{(1)} & b_2^{(1)} & b_3^{(1)} \end{bmatrix} =$$

$$\begin{bmatrix} x_{11}w_{11}^{(1)} + x_{12}w_{21}^{(1)} + b_1^{(1)} & x_{11}w_{12}^{(1)} + x_{12}w_{22}^{(1)} + b_2^{(1)} & x_{11}w_{13}^{(1)} + x_{12}w_{13}^{(1)} + b_3^{(1)} \\ x_{21}w_{11}^{(1)} + x_{22}w_{21}^{(1)} + b_1^{(1)} & x_{21}w_{12}^{(1)} + x_{22}w_{22}^{(1)} + b_2^{(1)} & x_{21}w_{13}^{(1)} + x_{22}w_{13}^{(1)} + b_3^{(1)} \\ \vdots & \vdots & \vdots \\ x_{N1}w_{11}^{(1)} + x_{N2}w_{21}^{(1)} + b_1^{(1)} & x_{N1}w_{12}^{(1)} + x_{N2}w_{22}^{(1)} + b_2^{(1)} & x_{N1}w_{13}^{(1)} + x_{N2}w_{13}^{(1)} + b_3^{(1)} \end{bmatrix}$$

Copyright Miles Chen. For personal use only. Do not distribute.

Bias Terms

Similarly, we have a bias matrix (with one element) for the node in the output layer.

$$\underset{N\times 3}{\mathbf{a}^{(2)}}\underset{3\times 1}{\mathbf{W}^{(2)}} + \underset{N\times 1}{\mathbf{J}}\underset{1\times 1}{\mathbf{B}^{(2)^T}} = \underset{N\times 1}{\mathbf{z}^{(3)}}$$

$$\begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} \\ a_{21}^{(2)} & a_{22}^{(2)} & a_{23}^{(2)} \\ \vdots & \vdots & \vdots \\ a_{N1}^{(2)} & a_{N2}^{(2)} & a_{N3}^{(2)} \end{bmatrix} \times \begin{bmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \\ w_{31}^{(2)} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \times \begin{bmatrix} b_1^{(2)} \end{bmatrix} =$$

$$\begin{bmatrix} a_{11}^{(2)} w_{11}^{(2)} + a_{12}^{(2)} w_{21}^{(2)} + a_{13}^{(2)} w_{31}^{(2)} + b_1^{(2)} \\ a_{21}^{(2)} w_{11}^{(2)} + a_{22}^{(2)} w_{21}^{(2)} + a_{23}^{(2)} w_{31}^{(2)} + b_1^{(2)} \\ \vdots \\ a_{N1}^{(2)} w_{11}^{(2)} + a_{N2}^{(2)} w_{21}^{(2)} + a_{N3}^{(2)} w_{31}^{(2)} + b_1^{(2)} \end{bmatrix}$$

Copyright Miles Chen. For personal use only. Do not distribute.

Summary

$$\underset{N\times 2}{\mathbf{X}}\ \underset{2\times 3}{\mathbf{W}^{(1)}} + \underset{N\times 1}{\mathbf{J}}\ \underset{1\times 3}{\mathbf{B}^{(1)^T}} = \underset{N\times 3}{\mathbf{z}^{(2)}}$$

$$f(\underset{N\times 3}{\mathbf{z}^{(2)}}) = \underset{N\times 3}{\mathbf{a}^{(2)}}$$

$$\underset{N\times 3}{\mathbf{a}^{(2)}}\underset{3\times 1}{\mathbf{W}^{(2)}} + \underset{N\times 1}{\mathbf{J}}\ \underset{1\times 1}{\mathbf{B}^{(2)^T}} = \underset{N\times 1}{\mathbf{z}^{(3)}}$$

$$f(\underset{N\times 1}{\mathbf{z}^{(3)}}) = \underset{N\times 1}{\hat{\mathbf{y}}}$$

$$\hat{\mathbf{y}} = f(f(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{J}\mathbf{B}^{(1)T})\mathbf{W}^{(2)} + \mathbf{J}\mathbf{B}^{(2)T})$$

Copyright Miles Chen. For personal use only. Do not distribute.

The Loss Function

To see how good the weights are, we need a Loss function to measure its performance.

We will use the SSE

$$\mathcal{L} = \sum (y - \hat{y})^2 = (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}})$$

Keep in mind that

$$\hat{\mathbf{y}} = f(f(\mathbf{X}\mathbf{W}^{(1)})\mathbf{W}^{(2)})$$

, or if you use biases, $\hat{\mathbf{y}} = f(f(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{J}\mathbf{B}^{(1)T})\mathbf{W}^{(2)} + \mathbf{J}\mathbf{B}^{(2)T})$

Next class: find the gradient of the loss function with respect to $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ so we can search for a local minimum.

Copyright Miles Chen. For personal use only. Do not distribute.