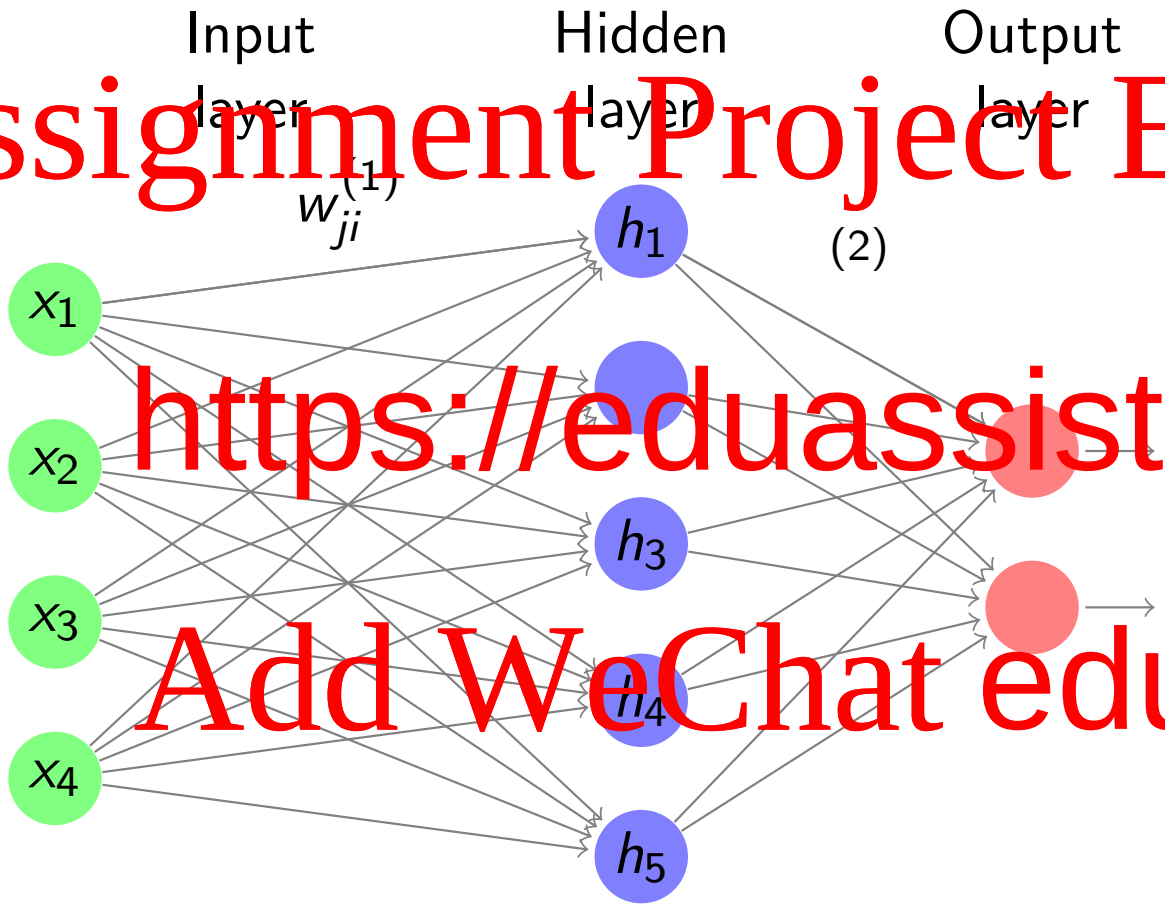
Foundations of Machine Learning

Neural Networks

ECS Southampto

November 23, 202

Multilayer Perceptron



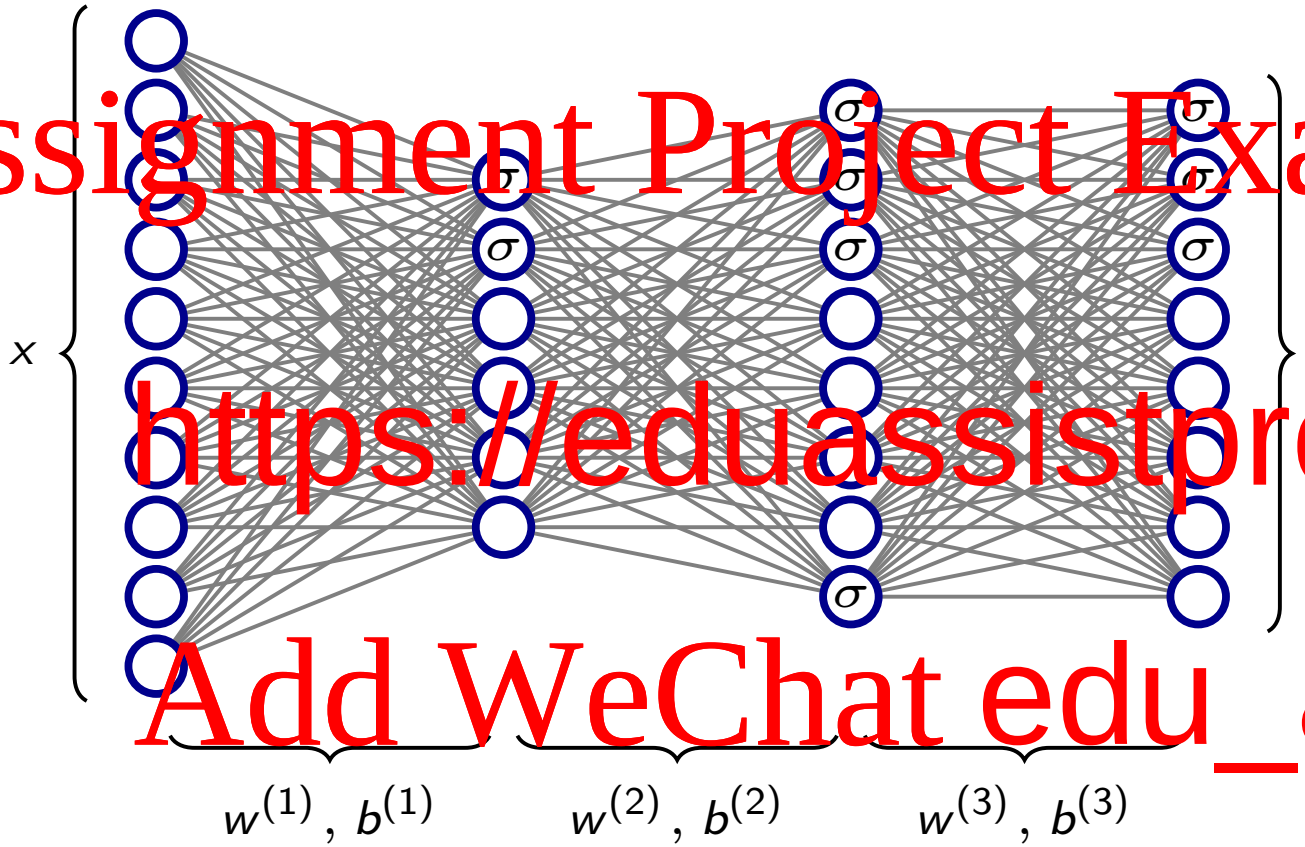
- MLPs are fully connected
-
-
-

previous slide

The Multilayer Perceptron – Output Layer

- c neurons in the output layer
-
- problem that is being solved – more details lat

Multilayer Perceptrons



Multiple layers of units

Multilayer Perceptron (MLP)

We can define the MLP formally as,

$\forall l =$

wher

Note, we will define the weighted input term,
$z^{(l)} = w^{(l)} a^{(l-1)} + b^{(l)}$, for the backprop

Multilayer Perceptron (MLP)

Define the following expressions:

1. $^{(1)}$
2.
3.
4. $a^{(2)}$
5. $a^{(2)}$

Activation Functions

- The activation function in a neural network is a function used to transform the activation level of a unit (neuron) into an output signal.
-
-
- function.
- The input space is mapped to a different spac
- There have been many kinds of activation fu over the years (640+), however, the most c the Sigmoid, Tanh, ReLU, and Softmax

The Logistic (or Sigmoid) Activation Function

- The sigmoid function is a special case of a logistic function

$$\frac{1}{}$$

-
- monotonically increasing
- NB: e is the natural logarithm

Sigmoid Function – Derivative

- The sigmoid function has an easily calculated derivative which is usesd in the back propagation algorithm

The Hyperbolic Tangent Activation Function

- The tanh function is also "s"-shaped like the sigmoidal function, but the output range is (-1, 1)
- $tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$
- $tanh'(x) = 1 - tanh^2(x)$

Rectified Linear Units (ReLU)

- The ReLU (used for hidden layer neurons) is defined as: $f(x) = max(0, x)$
- The range of the ReLU is between 0 to ∞

Softmax

The softmax is an activation function used at the output layer of a neural network that forces the outputs to sum to 1 so that they can represent a probability distribution across a discrete mutually exclu

$y_j =$ ———

Note $\frac{\partial y_i}{\partial z_i} = y_i(1 - y_i)$

- The output of a softmax layer is a set of positive nu which sum up to 1 and can be thought of as a proba distribution

Question

Question

The Cost Function (measure of discrepancy)

- Mean Squared Error (MSE) for M data points is given by $MSE = \frac{1}{2*M} \sum_{i=1}^{M} (\hat{y}_i - y_i)^2$
- $\frac{1}{2*M}$ just a constant so can be replaced by $\frac{1}{2}$ or $\frac{1}{M}$
-
-
- far off the targets [2]
- Cross-Entropy Cost function is generally cost function (discussed next)

[1] https://stats.stackexchange.com/questions/154879/a-list-of-cost-functions-used-in-neural-networks-alongside-applications

[2] http://neuralnetworksanddeeplearning.com/chap3.html

Cross-Entropy Cost Function

- $J = -\frac{1}{M}\sum_{i=1}^{M}[y_i ln(\hat{y}_i) + (1 - y_i) ln(1 - \hat{y}_i)]$
  where M is the number of training examples
-
-
  $y = 0$
- $\frac{\partial J}{\partial w_{ij}}$ is proportional to the error in the outpu
  therefore, the larger the error, the faster the n