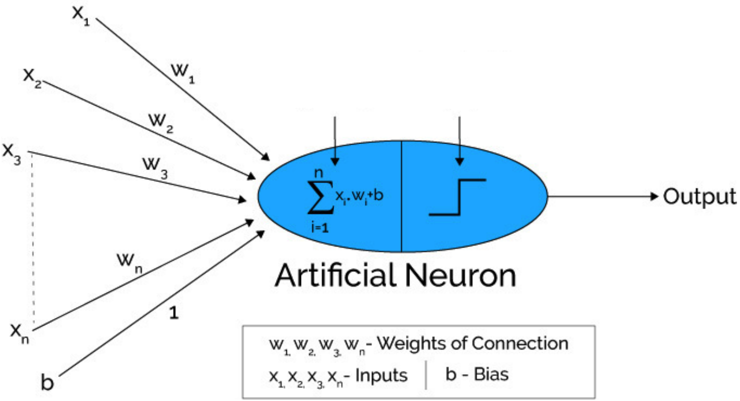
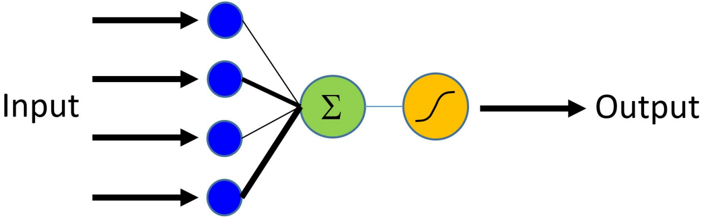
Mathematical techniques in data science

Lecture 9: Feed-forward neural networks (cont.)

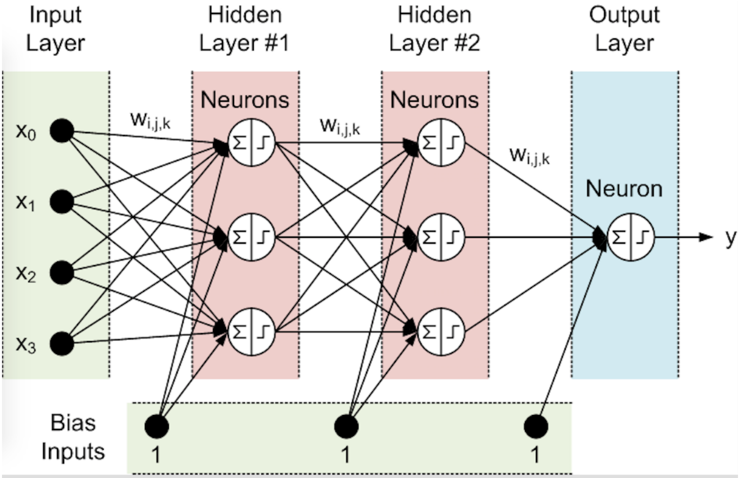
Logistic neuron



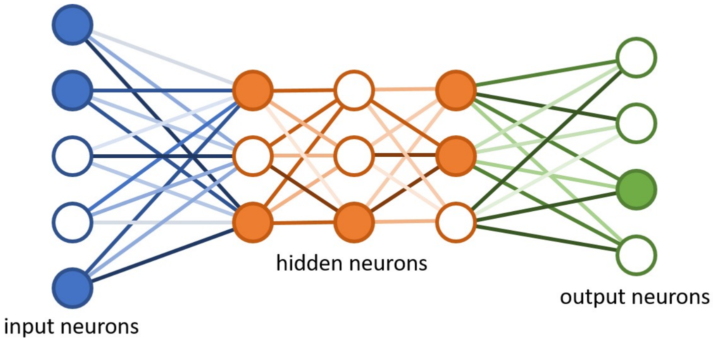
Logistic neuron



Feed-forward neural networks



Feed-forward neural networks

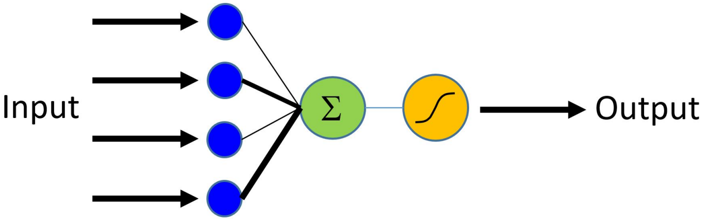


Feed-forward neural networks

- Structure:
  - Graphical representation
  - Activation functions
  - Loss functions
- Training:
  - Stochastic gradient descent
  - Back-propagation

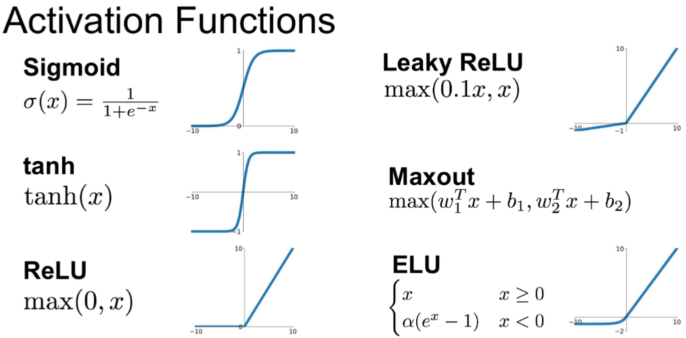
Activation functions

Activation functions



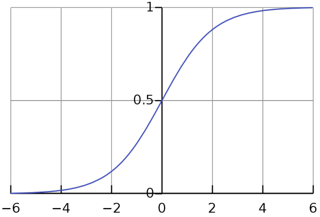
If we do not apply an activation function, then the output signal would simply be a simple linear function of the input signals

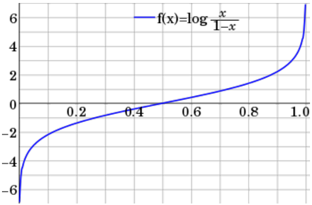
Activation functions



Logistic function (sigmoid function)

Transformation between $(-\infty, \infty)$ and $[0, 1]$

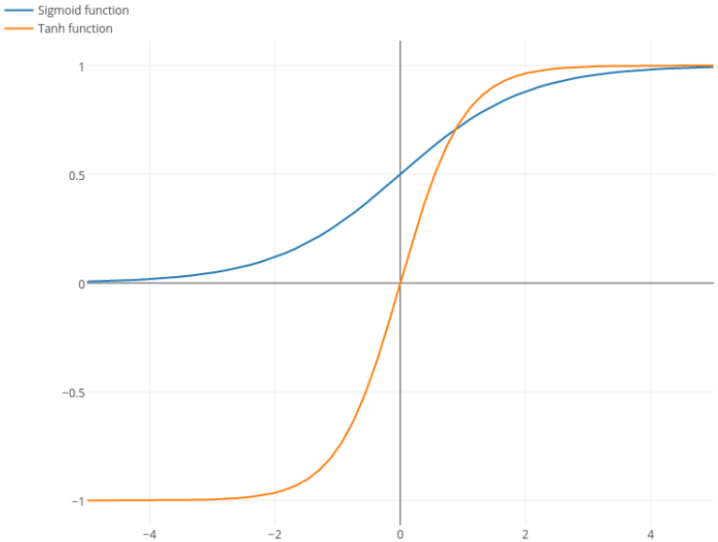




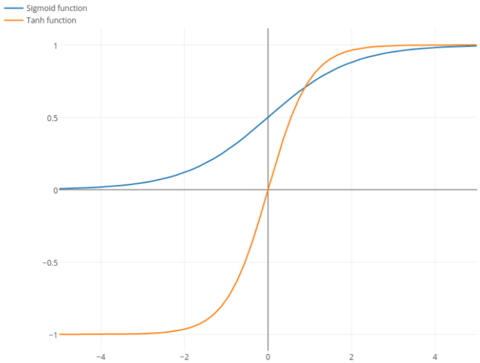
$$f(x) = \frac{e^x}{1 + e^x}$$

$$logit(p) = \log \frac{p}{1 - p}$$

Hyperbolic tangent

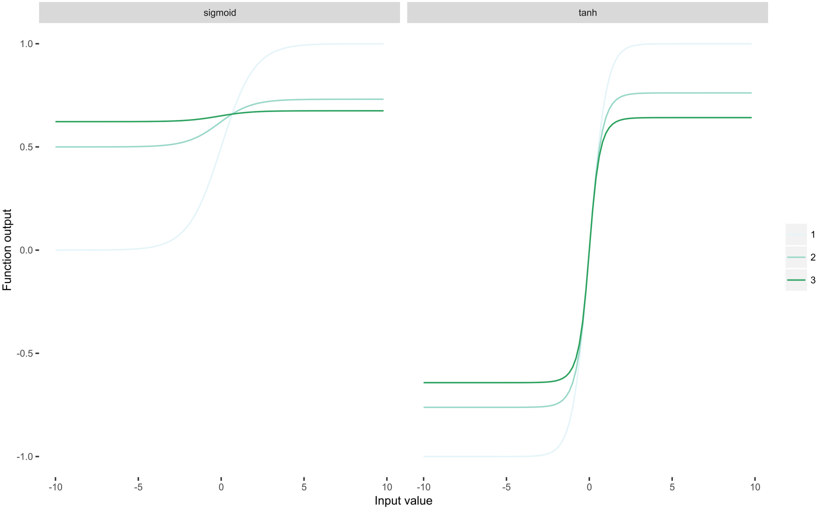


Hyperbolic tangent



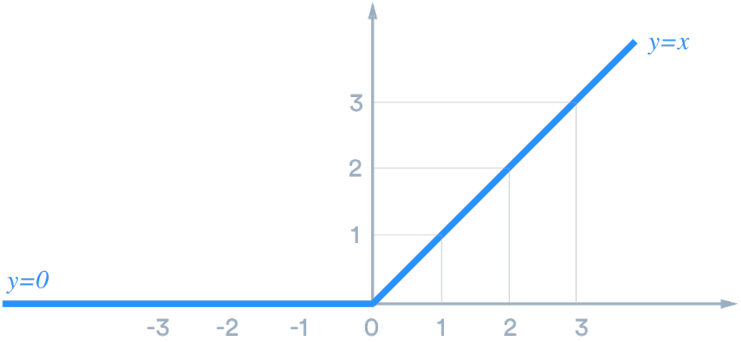
Issue: vanishing gradient problem

Hyperbolic tangent

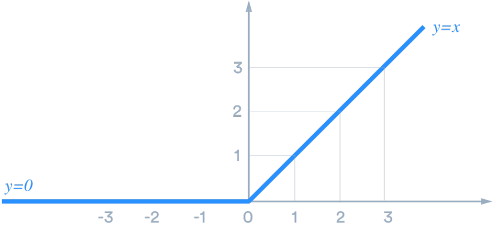


Vanishing gradient problem

Rectified linear unit (ReLU)



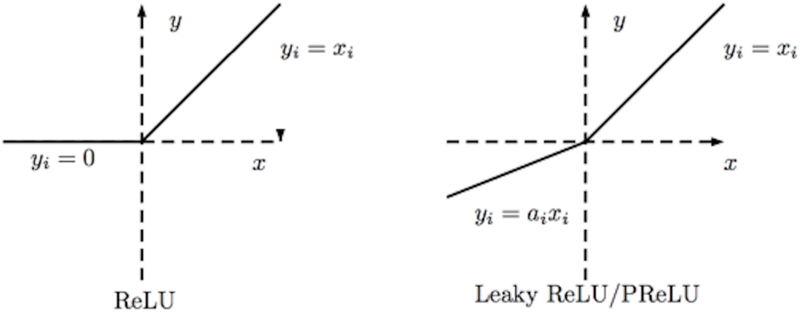
Rectified linear unit (ReLU)



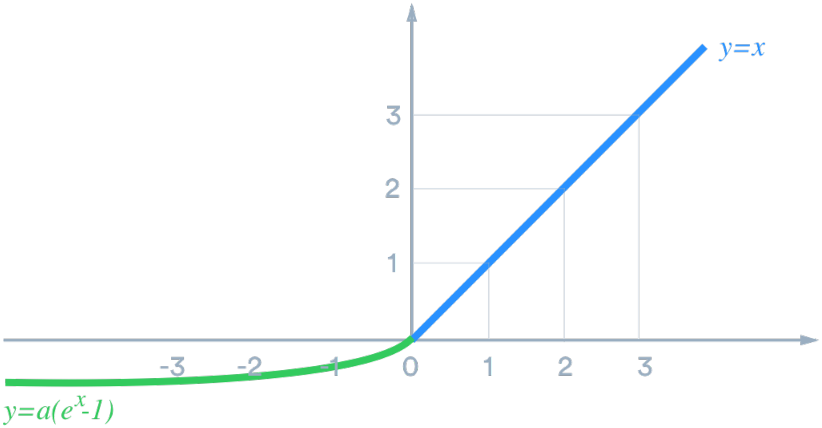
Advantage: model sparsity, cheap to compute (no complicated math), partially address the vanishing gradient problem

Issue: Dying ReLU

Leaky relu



Exponential Linear Unit (ELU, SELU)



Loss function for classification: cross-entropy

Code

```
def CrossEntropy(yHat, y):
    if y == 1:
      return -log(yHat)
    else:
      return -log(1 - yHat)
```

Math

In binary classification, where the number of classes M equals 2, cross-entropy can be calculated as:

$$-(y\log(p) + (1-y)\log(1-p))$$

If $M > 2$ (i.e. multiclass classification), we calculate a separate loss for each class label per observation and sum the result.

$$-\sum_{c=1}^{M} y_{o,c}\log(p_{o,c})$$

Note: Here $y_{o,c}$ is the 0-1 label and $p_{o,c}$ is the predicted probability for the observation o is of class c, respectively

Stochastic gradient descent

Gradient descent

Gradient Descent

Minimize a function by moving in the opposite direction of the gradient.

$$\theta_i := \theta_i - \rho \frac{\partial J}{\partial \theta_i}$$

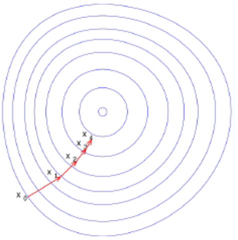


Figure: Gradient Descent. Source: `http://en.wikipedia.org/wiki/Gradient_descent`