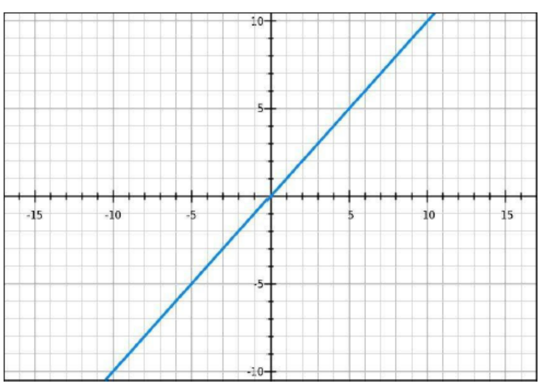
Mod-2

Multilayer Neural Networks Topology

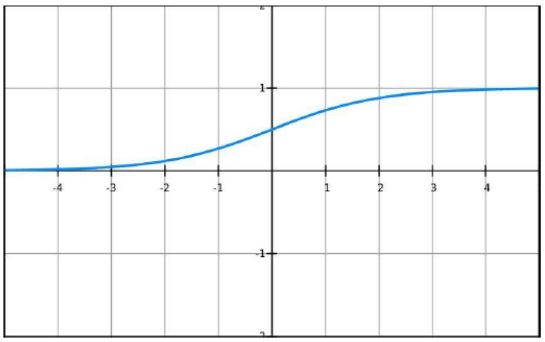
- *Connection weights*-coefficients that scale (amplify or minimize) the input signal to a given neuron in the network
- *Biases*-
  - Scalar values added to the input to ensure that at least few nodes per layer are activated regardless of signal strength.
  - Biases allow learning to happen by giving the network action in the event of low signal.
  - Allow the network to try new interpretations or behaviors.
  - Notated as *b*, and, like weights, biases are modified throughout the learning process
- *Activation functions*-The functions that govern the artificial neuron's behavior

Activation functions Linear



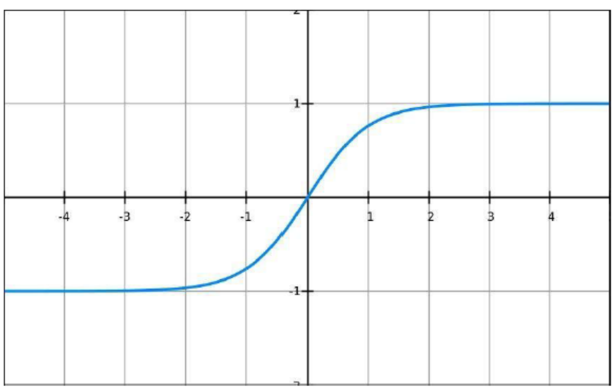
- A linear transform is basically the identity function,
- f(x) = W * x
- Dependent variable has a direct, proportional relationship with the independent variable.
- It means the function passes the signal through unchanged.
- Used in the input layer of neural networks

Activation functions Sigmoid



- Sigmoids can reduce extreme values or outliers in data without removing them.
- A sigmoid activation function outputs an independent probability for each class.
- A sigmoid function is a machine that converts independent variables of near infinite range into simple probabilities between 0 and 1, and most of its output will be very close to 0 or 1.

Activation functions Tanh

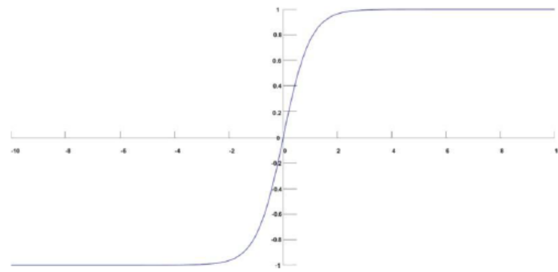


- Hyperbolic trignometric function
- Just as the tangent represents a ratio between the opposite and adjacent sides of a right triangle, tanh represents the ratio of the hyperbolic sine to the hyperbolic cosine:
- tanh(x) = sinh(x) / cosh(x).
- normalized range of tanh is −1 to 1.
- Advantage -it can deal more easily with negative numbers

Activation functions Softmax

Softmax-

- Also known as -softargmax or normalized exponential function
- Allows us to express our inputs as a discrete probability distribution
- a generalization of logistic regression in as much as it can be applied to continuous data (rather than classifying binary) and can contain multiple decision boundaries.
- It handles multinomial labeling systems.
- Generally used at the output layer of a classifier
- Defined as follows:
  - for each value in input vector, the Softmax value is the exponent of the individual input divided by a sum of the exponents of all the inputs.
- Returns the probability distribution over mutually exclusive output classes.



$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

σ = softmax

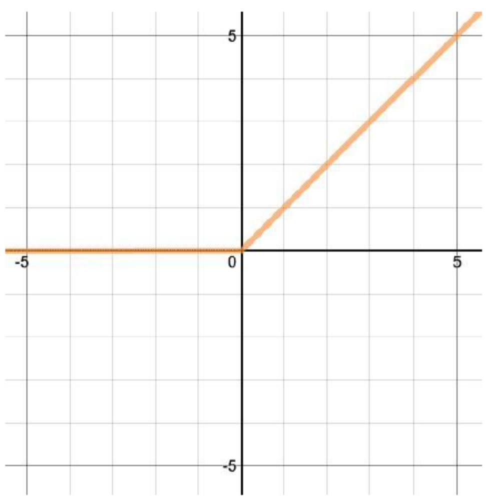
$\vec{z}$ = input vector

e^{z_i} = standard exponential function for input vector

K = number of classes in the multi-class classifier

e^{z_j} = standard exponential function for output vector

Activation functions Rectified Linear

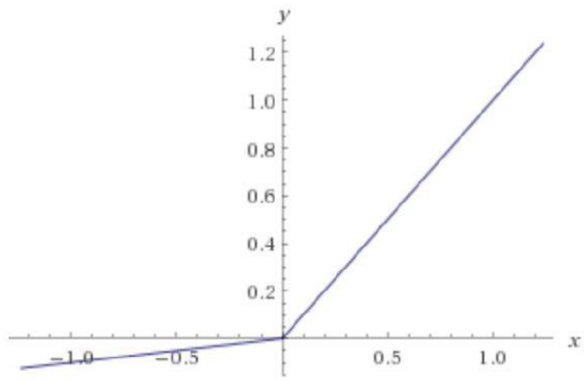


- More interesting transform that activates a node only if the input is above a certain quantity.
- While the input is below zero, the output is zero, but when the input rises above a certain threshold, it has a linear relationship with the dependent variable f(x) = max(0, x)
- ReLU activation functions have shown to train better in practice than sigmoid activation functions
- Compared to the sigmoid and tanh activation functions, the ReLU activation function does not suffer from vanishing gradient issues

Activation functions Leaky Relu

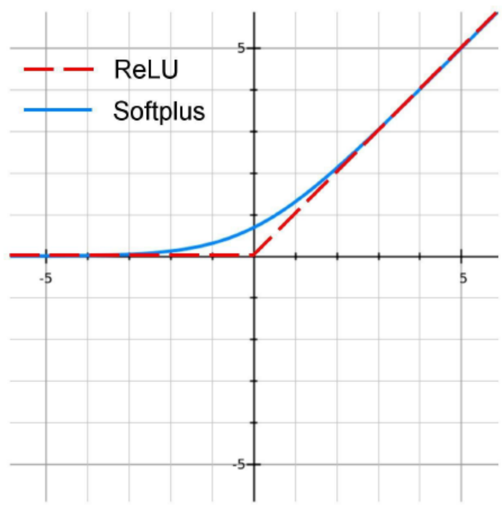
- A strategy to mitigate the "dying ReLU"
- As opposed to having the function being zero when x < 0, the leaky ReLU will instead have a small negative slope (e.g.,"around 0.01").
- Some success has been seen in practice with this ReLU variation, but results are not always consistent
- The equation is :

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$$



Activation functions Softplus

- This activation function is the "smooth version of the ReLU,"
- Compare this plot to the ReLU - shows that the softplus activation function (f(x) = ln[1 + exp(x)]) has a similar shape to the ReLU.



Architectures of deep networks

- Four major architectures of deep networks-
  - Unsupervised Pretrained Networks
  - Convolutional Neural Networks
  - Recurrent Neural Networks
  - Recursive Neural Networks

Unsupervised Pretrained Networks (UPN):

- **Purpose:** UPNs are designed for unsupervised learning, where the model learns patterns and representations from unlabeled data.
- **Training:** Initially trained on a large dataset without labeled outputs, the model captures inherent structures and features.
- **Transfer Learning:** Pretrained UPNs can be fine-tuned for specific tasks with smaller labeled datasets, leveraging the learned representations.

Convolutional Neural Networks (CNNs):

- **Purpose:** Primarily used for visual data, such as images and videos.
- **Architecture:** Consists of convolutional layers for spatial hierarchies, pooling layers for down sampling, and fully connected layers for classification.
- **Feature Extraction:** CNNs automatically learn hierarchical features, recognizing patterns like edges, textures, and shapes.

Recurrent Neural Networks (RNNs):

- **Purpose:** Suited for sequential data, like time-series, speech, and natural language.
- **Architecture:** Incorporates recurrent connections to maintain a memory of previous inputs, allowing the model to capture temporal dependencies.
- **Applications:** Used in tasks like language modeling, speech recognition, and sentiment analysis.

Recursive Neural Networks (RecNNs):

- **Purpose:** Applies to hierarchical structures, such as tree-structured data.
- **Architecture:** Operates recursively on a hierarchical structure, capturing dependencies at different levels of abstraction.
- **Applications:** Commonly used in natural language processing tasks that involve parse trees, syntactic structures, or hierarchical relationships.

Regularization-

  - a measure taken against overfitting
- Regularization for hyperparameters helps modify the gradient so that it doesn't step in directions that lead it to overfit
  - Dropout
  - DropConnect
  - L1 penalty
  - L2 penalty
- Regularization works by adding an extra term to the normal gradient computed.

Common Architectural Principles of Deep Networks

Regularization-

- *Dropout-*
  - Mechanism used to improve the training of neural networks by omitting a hidden unit.
  - Speeds training.
  - Driven by randomly dropping a neuron so that it will not contribute to the forward pass and backpropagation.
- *DropConnect*
  - Does the same thing as Dropout, but instead of choosing a hidden unit, it mutes the connection between two neurons.
- **Dropout and DropConnect mute parts of the input to each layer, such that the neural network learns other portions.**
- **Zeroing-out parts of the data causes a neural network to learn more general representations**

Regularization-

- The penalty methods L1 and L2, in contrast, are a way of preventing the neural network parameter space from getting too big in one direction.
- They make large weights smaller.

- *L1-*
  - Computationally inefficient in the nonsparse case, has sparse outputs, and includes built-in feature selection.
  - Multiplies the absolute value of weights rather than their squares.
  - This function drives many weights to zero while allowing a few to grow large, making it easier to interpret the weights

---

Common Architectural Principles of Deep Networks

Regularization-

- L2 –
  - Computationally efficient due to analytical solutions and nonsparse outputs, but it does not do feature selection automatically.
  - Common and simple hyperparameter, adds a term to the objective function that decreases the squared weights.
  - Multiply half the sum of the squared weights by a coefficient called the weight-cost.
  - Improves generalization, smooths the output of the model as input changes, and helps the network ignore weights it does not use.