Activation Functions

1

Variables Characterizing Neural Networks

- Architecture: How units connect to one another
  - Feed-forward, feedback, recurrent, auto-associative, hybrids
  - Will encounter various of these throughout the course
- Input function:
  - Function for collecting input from other units
  - $n_i = \sum w_{ij} a_j$ which in vector notation is $\mathbf{w} \cdot \mathbf{a}$
- Activation function:
  - Function for converting input to activation
- Learning rule:
  - Function for updating connection weights to/from other units
  - Hebbian, error correction, …

2

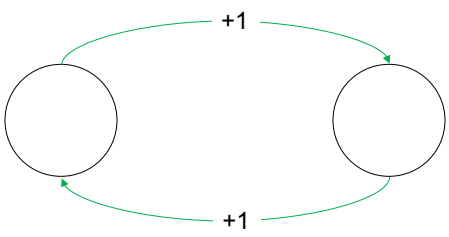
Activation Functions

- A function for converting input to activation
- Can have counterintuitive effects on network behavior
- Important properties:
  - Linear vs. nonlinear
    - Recurrent: Feedback loops
    - Feedforward: Doing actual work
  - If nonlinear: Differentiable vs. not
  - Instantaneous response vs. temporal integration
  - Others (complexity, temporal dynamics…)

3

Linear Activation Functions

- Almost useless
- $a_i^t = n_i^{t-1}$ (basic linear)
- $a_i^t = n_i^{t-1} - \theta_i$ (linear with threshold)



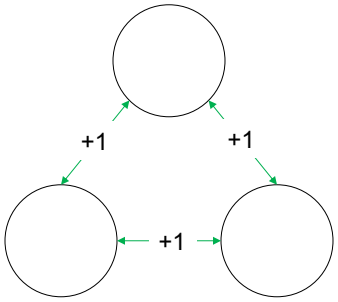
What will this network do if both nodes start with activation = 1.0 ($\theta = 0$)?

What will this network do if one node starts with activation = 1.0 and the other with activation = 0.0?

4

Linear Activation Functions

- Almost useless
- $a_i^t = n_i^{t-1}$ (basic linear)
- $a_i^t = n_i^{t-1} - \theta_i$ (linear with threshold)



What will this network do if two nodes start with activation = 1.0 ($\theta = 0$)?

What will this network do if one node starts with activation = 1.0 and the other with activation = 0.0?

NB: It's getting harder to do the simulation in our heads…

5

Linear Activation Functions

- Almost useless
- $a_i^t = n_i^{t-1}$ (basic linear)
- $a_i^t = n_i^{t-1} - \theta_i$ (linear with threshold)
- In a layered network, a linear activation function is useless
  - Any *n*-layered linear network is formally equivalent to a one-layered network
  - There are some functions (e.g., XOR) a one-layered network cannot compute
  - We'll come back to this
- So let's talk about nonlinear activation functions…

6

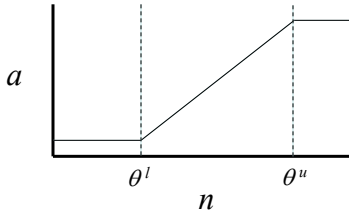
Binary Threshold Unit

- The simplest possible nonlinearity:
- $a_i^t = \begin{cases} 1 \; if \; n_i^{t-1} \geq \theta_i \\ 0 \; otherwise \end{cases}$
- Advantages:
  - Activation *bounded* so no exploding to infinity
  - Can compute complex functions (like XOR) in a layered system
- Disadvantage:
  - Information is completely lost above and below θ

7

Rectified Linear Unit ("RLU")

- Preserves information between upper and lower thresholds:
- $a_i^t = \begin{cases} 1 \; if \; n_i^{t-1} \geq \theta_i^u \\ 0 \; if \; n_i^{t-1} < \theta_i^l \\ kn_i^{t-1} otherwise \end{cases}$



- Advantages:

  Activation *bounded* so no exploding to infinity

  Can compute complex functions (like XOR) in a layered system

  Information preserved between θ^l and θ^u

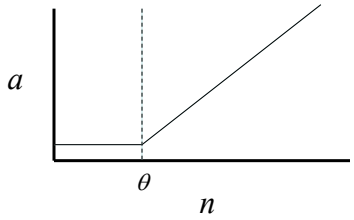
  Semi-differentiable and nonlinear
- Disadvantages:

  A hack

8

Rectified Linear Unit ("RLU")

- OR Preserves information above lower thresholds:
- $a_i^t = \begin{cases} 0 \; if \; n_i^{t-1} < \theta \\ k n_i^{t-1} \; otherwise \end{cases}$

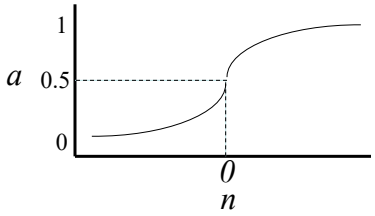


- Advantages:
  - Can compute complex functions (like XOR) in a layered system
  - Information preserved above θ
  - Semi-differentiable and nonlinear; rapid convergence w/ back prop
- Disadvantages:
  - Activation *unbounded* so can explode to infinity in autoassociator
  - A hack

9

Logistic Function

- $a_i^t = \frac{1}{1+e^{-n_i^{t-1}}}$

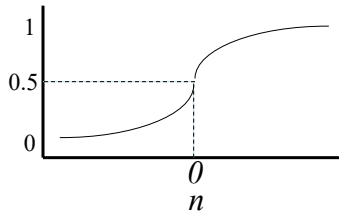


- Advantages:
  - Activation *bounded* so no exploding to infinity
  - Can compute complex functions (like XOR) in a layered system
  - Information is not completely lost
  - Fully differentiable: $a_i(1-a_i)$
- Disadvantage:
  - Instantaneous only

10

Probabilistic Logistic Function

- $p(a_i^t = 1) = \frac{1}{1+e^{-n_i^{t-1}/T}}$



- Advantages:

  Simulated annealing by varying T (temperature) over time

- Disadvantage:

  Instantaneous only

11

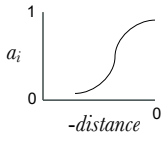
Gaussian Radial Basis Function

The Gaussian, a probability density function:

$$a_i^t = \frac{1}{\sqrt{2\Pi\sigma^2}} e^{-0.5\left(\frac{\mu-x}{\sigma}\right)^2}$$

For Gaussians in a high dimensional space,
μ and x are both vectors, so $(\mu - x)$ is the Euclidean distance, d, between μ and x. The equation therefore simplifies to

$$a_i^t = \frac{1}{\sqrt{2\Pi\sigma^2}} e^{-0.5\left(\frac{d}{\sigma}\right)^2}$$



The height of any Gaussian, $G()$, at it's mean, μ, will be less than 1.0. Therefore, to ensure that a_i is bounded between 0 and 1, you must set a_i to the value of the Gaussian at $-distance$, $G(-distance)$, by $1/G(\mu)$.

12

Temporal Integrators

- Integrate information over time:
- Activation at time *t* is a function of both input at *t*-1 and activation at *t*-1.
- **Simple integrator**:

$$a_i^t = a_i^{t-1} + \gamma n_i^{t-1}(1 - a_i^{t-1})$$

Rewrite in terms of *change* in activation:

$$\Delta a_i = \gamma n_i(1 - a_i) \quad \text{where} \quad a_i^t = a_i^{t-1} + \Delta a_i^{t-1}$$

- Advantages:

  Integrates over time, activation bounded to <= 1.0

- Disadvantage:

  Asymptotic activation is always 1.0; unbounded on lower range; unstable

13

"Leaky" Integrator

$$\Delta a_i = \gamma n_i(1 - a_i) - \delta a_i$$

- Advantages:

  Integrates over time, activation bounded to <= 1.0

  Asymptotic activation proportional to net input

- Disadvantage:

  Unbounded on lower range; unstable

14

Leaky Integrator Asymptotic Activation

$$\Delta a_i = \gamma n_i(1 - a_i) - \delta a_i$$

Set Δa_i to zero and solve for a_i:

$$0 = \gamma n_i(1 - a_i) - \delta a_i$$

Distribute with n and γ:

$$0 = \gamma n - a\gamma n - \delta a$$

Get all the as on the left:

$$a\gamma n + \delta a = \gamma n$$

Get a on the outside:

$$a(\gamma n + \delta) = \gamma n$$

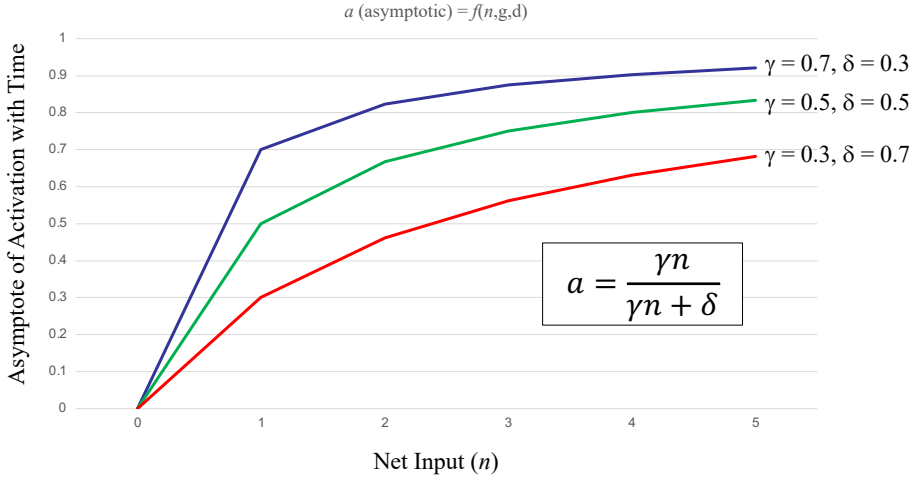
Divide both sides by $(\gamma n + \delta)$:

$$a = \frac{\gamma n}{\gamma n + \delta}$$

As n_i grows toward infinity, asymptote of a_i grows toward 1.0.

15

Leaky Integrator Asymptotic Activation



- a_i (asymptote) asymptotically approaches 1.0 in n_i.
- The rate at which it does do depends on γ and δ.

16

Leaky Integrator

$$\Delta a_i = \gamma n_i (1 - a_i) - \delta a_i$$

- Advantages:
  - Integrates over time, activation bounded to <= 1.0
  - Asymptotic activation proportional to net input
- Disadvantage:
  - Unbounded on lower range; unstable
  - Activation slow to change near asymptote

17

Grossberg's Leaky Integrator

$$\Delta a_i = \gamma (e_i (1 - a_i) + i_i (1 + a_i)) - \delta a_i$$

where e_i is excitatory input and i_i is inhibitory input.

- Advantages:
  - Integrates over time, activation bounded to -1…1 (bounded lower range)
  - Asymptotic activation proportional to net input
  - Activation responds quickly to inputs unlike its current value
- Disadvantage:
  - Unstable

18

"Unstable" Integrator?

$$\Delta a_i = \gamma n_i (1 - a_i) - \delta a_i$$

- What happens if:
  - $a_i = 0.5$
  - $n_i = 10$
  - $\gamma = 0.5$
  - $\delta = 0.5$?
- $\Delta a_i = 0.5 * 10 * (1 - 0.5) - 0.5 * 0.5 = 2.5 - 0.25 = 2.25$

Next iteration: $a_i = 0.5 + 2.25 = 2.75$!

And things just go crazy from here.

19

Dealing with Unstable Integrators

$$\Delta a_i = \gamma n_i (1 - a_i) - \delta a_i$$

- Choose γ and δ wisely (know the range of possible n_i)
- Incorporate a *time step* constant, $t < 1.0$:

$$\Delta a_i = t(\gamma n_i (1 - a_i) - \delta a_i)$$

- After you make the change: $a_i = a_i + \Delta a_i$,

  use *if* statements to ensure that $0 \leq a_i \leq 1$:

  if self.activation > 1.0: self.activation = 1.0

  if self.activation < 0.0: self,activation = 0.0
- Inelegant, but better than a runtime error

20

```
class Unit(object):
        def __init__(self):
                self.connections = []

class Connection(object):
        def __init__(self, recipient, sender, weight = 0.0):

# construct network
units = []
for i in range(10):
        units.append(Unit())

# connects every unit to every other unit except itself
for unit1 in units:
        for unit2 in units:
                if not unit1 is unit2:
                        unit1.add_connection(unit2)
```

21