Math 514

Neural Networks

(updated: 2019-02-28)

Neural Networks

Networks so far:

1. Perceptron
   - Linearly separable data
2. Regression (Linear/Logistic)
   - Linear decision boundary
3. Softmax
   - Multi-category, linear decisions.

Pluses

- Covered the most important cost functions
- Gradient descent is easy when inputs connect directly to output

Single layer Network

- Good warm up for deep nets/backpropagation.

Will more layers make solutions more expressive?

Classification

- Classify pictures as cat or not cat.
- 3 colors x 720 x 480 pixels = 1,036,800

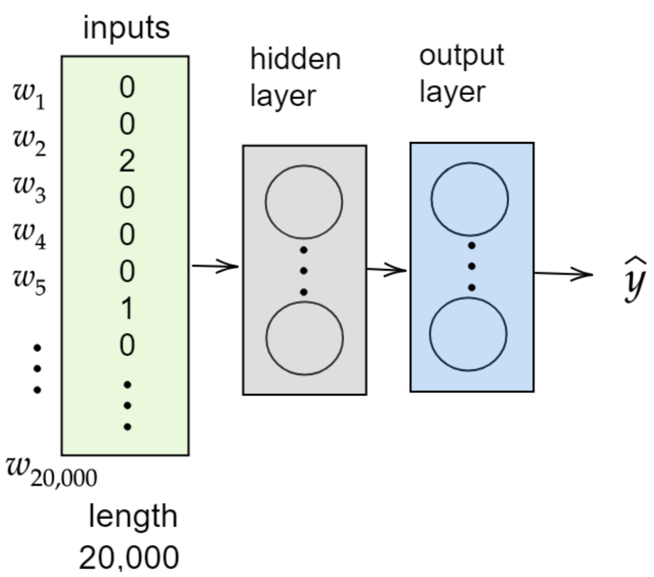
- Each picture is a point in 10^6 dimensional space.
  - MNIST images are 28 x 28 gray scale
- Images are very redundant

Natural Language Processing

Bag of words

Text classification

- Spam/Not Spam
- Positive/Negative Review
- Relevant/Not Relevant
- Create vocabulary list
    - Length 10,000? 20,000?
- Convert document to vocabulary count vector



The Learning Problem (Supervised)

Given data $\mathbf{x}^{(i)}$ and labels $t^{(i)}$

$$\{\,\mathbf{x}^{(i)}, \quad t^{(i)}\,\}, \quad i = 1, \ldots, m$$

Find a function $f(\mathbf{x})$ such that

$$f(\mathbf{x}^{(i)}) \sim t^{(i)}$$

Think of each $x^{(i)}$ as an image and $t^{(i)}$=1 if it's a cat and $t^{(i)}$=0 otherwise.

The assumption is that $(\mathbf{x}^{(i)})$ and $t^{(i)}$ are sampled from a fixed but unknown distribution $p(\mathbf{x}, t)$.

Each model generates a candidate function f. The goal is to find one that **generalizes**

Generalization

> We should be careful to get out of an experience only the wisdom that is in it, and stop there, lest we be like the cat that sits down on a hot stove-lid. She will never sit down on a hot stove-lid again - and that is well; but also she will never sit down on a cold one anymore."[1]
>
> **Mark Twain, *Following the Equator***

Two elements:

1. Model must learn to predict $t^{(i)}$ well on data it is trained with.
2. Must also predict well when given a sample $(\mathbf{x}^{(j)}, t^{(j)})$ not part of the training data.

1. From URL hagan.okstate.edu/13_Generalization.pdf

Generalization

Classification models are evaluated based on predictive ability.

- Not memorization. Must generalize to new data.
- Not explanatory. Unlike most statistical/economic models.
- Understanding generalization is an active area of ongoing research.

> "In practice it is often found that large over-parameterized neural networks generalize better than their smaller counterparts, an observation that appears to conflict with classical notions of function complexity, which typically favor smaller models."
>
> ***Sensitivity and Generalization in Neural Networks: An Empirical Study,"*** Novak Bahri, Abolafia, Pennington and Sohl-Dickstein, Google Brain 2018.

The Learning Problem

Training data [1]

Note 1.Example from Daume

The Learning Problem

Classify the test data as Class A or Class B.

Daume images http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf

The Learning Problem

Is the training data drawn from the same joint distribution?

- Fly/Can't Fly -- AABB, 1/3 of people
- Bird/Not Bird -- ABBA, 2/3 of peopale

This example shows inductive bias.

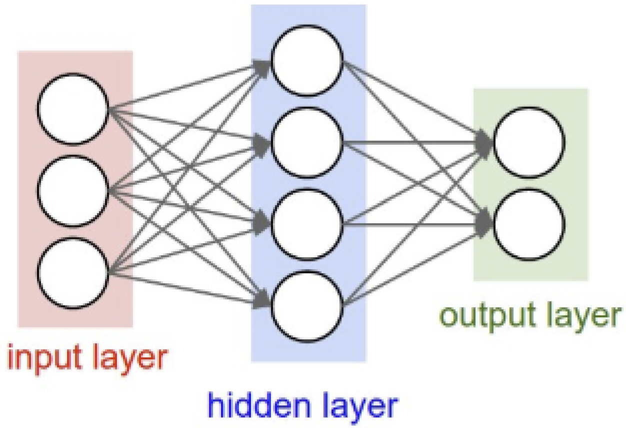
- Models have implicit inductive bias.

There are many examples of neural networks learning the wrong thing. What about ABAB for "B=background in focus"/"A=background not in focus"

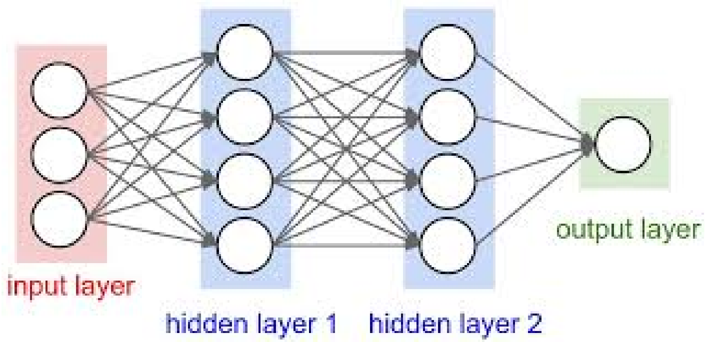
Neural Networks

Network depth equals the number of sets of adjustable parameters

2 layer network (1 hidden layer)



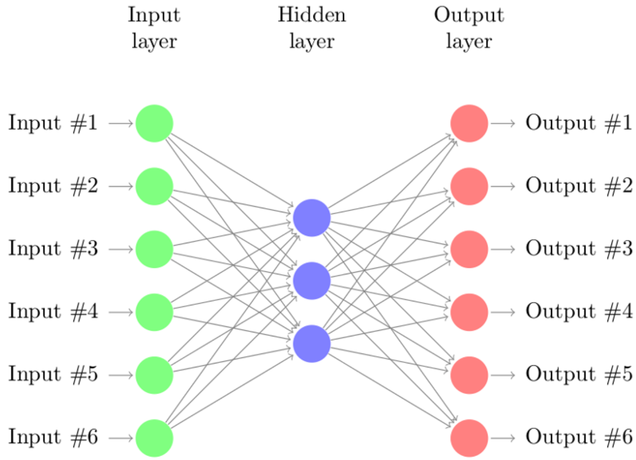
3 layer network (2 hidden layers)



Neural Networks

Increasing Depth - Positives

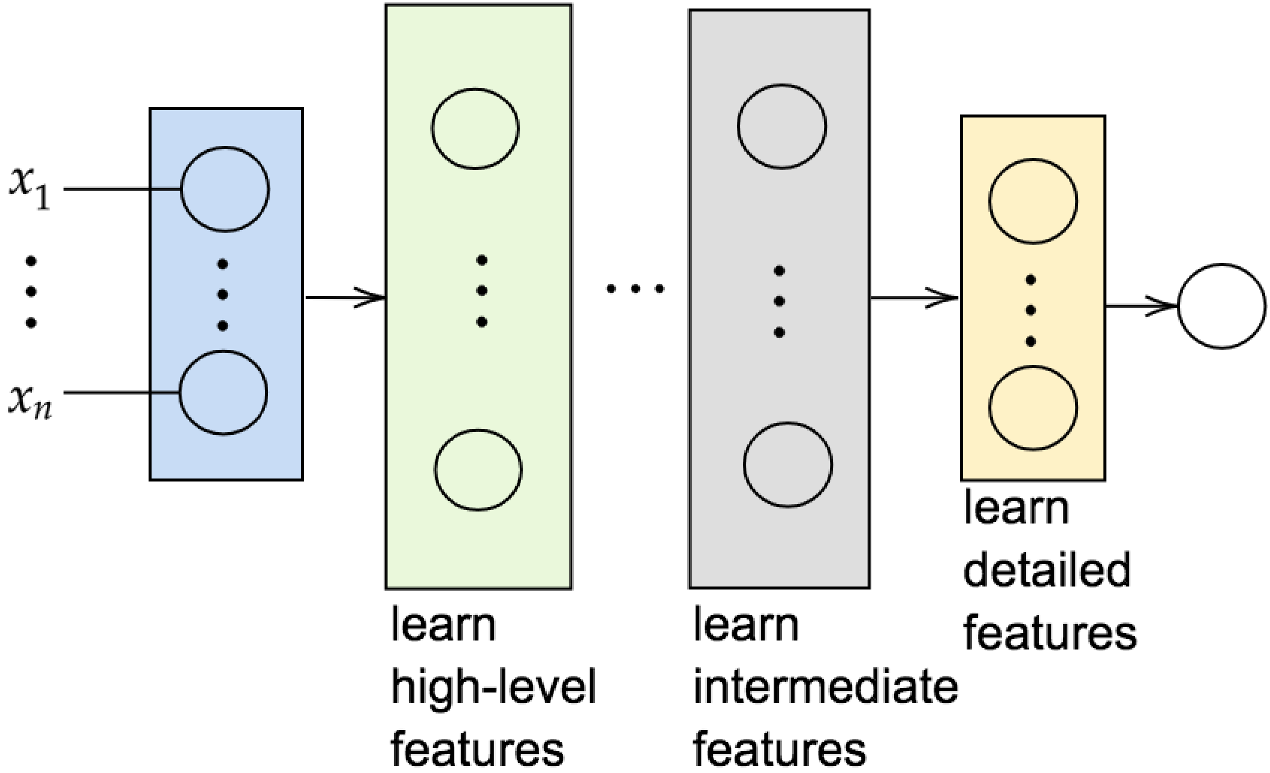
- Increases expressiveness - will allow non-linear decision boundaries.
- Universal approximators - to be shown.
- Universal classifier - to be shown.
- Many interesting architectures.



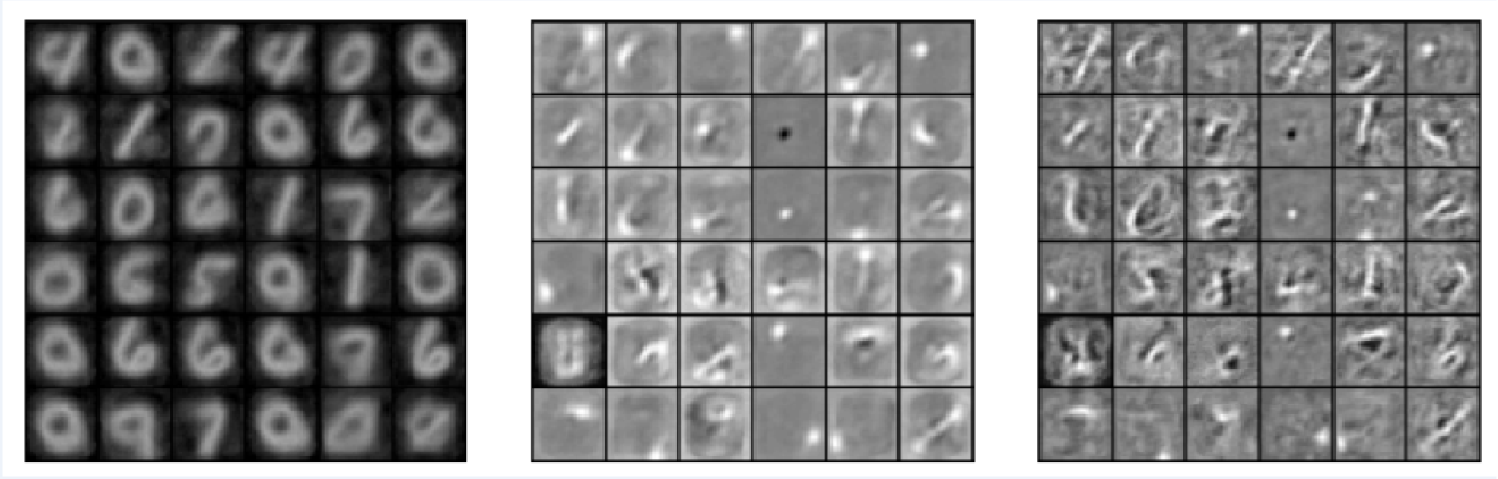
Most data is unlabeled. Extracting useful information from unlabeled data is sometimes the desired output from network. Learning new representations for data

Neural Networks

Increasing Depth



Layer Visualization



Neural Networks

Increasing Depth

Negatives:

- Non-convex cost function
  - Get a different result each time
- Are all local-minima good enough?
- More complex gradient descent
  - Will show backpropagation is efficient.

Why Neural Networks

Massively parallel

- Can take advantage of GPUs
- Layered structure is key

Efficient training

- backpropagation

Computationally powerful

- Universal approximators
- Universal classifiers

Empirical evidence they work

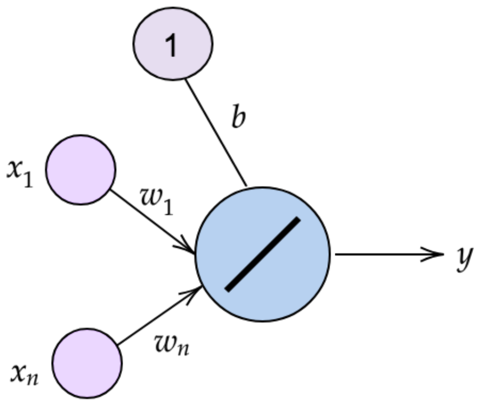
- academically and comercially

Architecturally flexible - can be applied to a wide range of problems.

- FFNN classification
- CNN Vision
- RNN translation

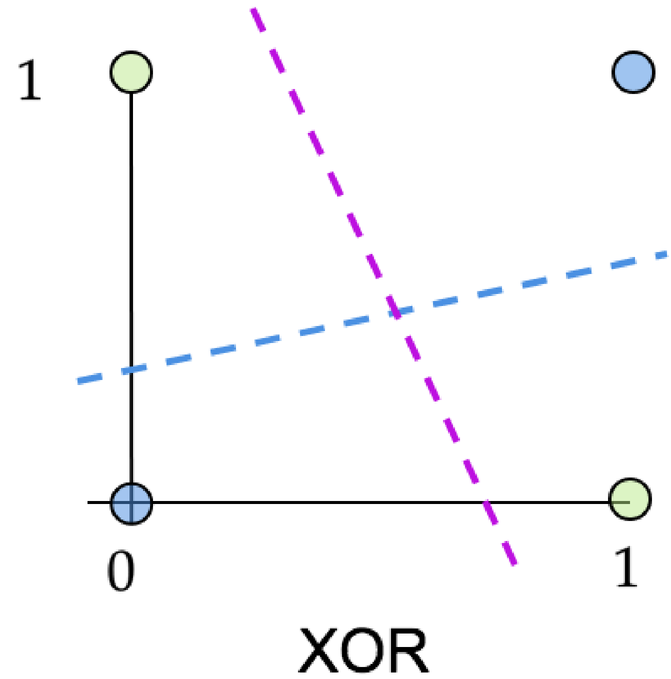
Inductive Bias toward generalization

Nonlinear Activations



XOR

x_1	x_2	t
0	0	0
0	1	1
1	0	1
1	1	0



Nonlinear Activations

$$
\begin{aligned}
y &= b + \mathbf{w}^T \mathbf{x} \\
C(t, \mathbf{x}) &= \frac{1}{2} \sum_{i=1}^{4} (y^{(i)}) - t^{(i)})^2 \\
&= \frac{1}{2} \sum_{i=1}^{4} (b + \mathbf{w}^T \mathbf{x}^{(i)}) - t^{(i)})^2 \\
&= \frac{1}{2} \sum_{i=1}^{4} (e^{(i)})^2
\end{aligned}
$$

Minimize cost by setting partials to zero:

$$
\frac{\partial C}{\partial b} = \sum_{i}^{4} e^{(i)} \frac{\partial y^{(i)}}{\partial b} = \sum_{i}^{4} e^{(i)}
$$

$$
\frac{\partial C}{\partial \mathbf{w}} = \sum_{i}^{4} e^{(i)} \frac{\partial y^{(i)}}{\partial \mathbf{w}} = \sum_{i}^{4} e^{(i)} \mathbf{x}^{(i)}
$$

Error Terms:

$$
e^{(i)} = b + \mathbf{w}_1 \mathbf{x}_1^{(i)} + \mathbf{w}_2 \mathbf{x}_2^{(i)} - t^{(i)}
$$

XOR

x_1	x_2	t	e^i
0	0	0	b
0	1	1	$b + w_2 - 1$
1	0	1	$b + w_1 - 1$
1	1	0	$b + w_1 + w_2$

Nonlinear Activations

Condition:

$\sum_{i=1}^{4} e^{(i)}\mathbf{x}^{(i)} = 0$

$$b\begin{bmatrix}0\\0\end{bmatrix} + (b+\mathbf{w}_1-1)\begin{bmatrix}1\\0\end{bmatrix} + (b+\mathbf{w}_2-1)\begin{bmatrix}0\\1\end{bmatrix} + (b+\mathbf{w}_1+\mathbf{w}_2)\begin{bmatrix}1\\1\end{bmatrix} = 0$$

or

$$(1)\ b+\mathbf{w}_1-1+b+\mathbf{w}_1+\mathbf{w}_2 = 0$$

$$(2)\ b+\mathbf{w}_2-1+b+\mathbf{w}_1+\mathbf{w}_2 = 0$$

$$\Rightarrow \mathbf{w}_1-\mathbf{w}_2 = 0$$
$$\Rightarrow \mathbf{w}_1 = \mathbf{w}_2$$

Nonlinear Activations

Using $\mathbf{w}_1 = \mathbf{w}_2$ in (1) shows

$$2b + 3\mathbf{w} - 1 = 0$$

$$b = (1 - 3\mathbf{w})/2$$

Now

$$\begin{aligned} \sum_{i=1}^{4} e^{(i)} &= 4b + 4\mathbf{w} - 2 = 0 \\ b &= \frac{(1 - 2\mathbf{w})}{2} \\ \frac{1 - 3\mathbf{w}}{2} &= \frac{1 - 2\mathbf{w}}{2} \\ 3\mathbf{w} &= 2\mathbf{w} \\ \mathbf{w} &= 0 \end{aligned}$$

If $w = 0$ then $b = 1/2$

Linear activation predicts:

$y(x) = b + \mathbf{w}^T\mathbf{x} = \frac{1}{2} + (0, 0) \cdot \mathbf{x} = \frac{1}{2}$

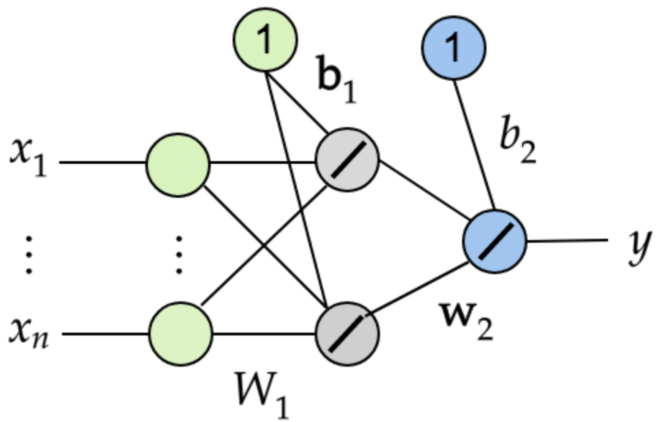
For all $\mathbf{x}$.

Not very useful.

Nonlinear Activation

What if we add another layer, but keep all activations linear?

Note that W_1 below is a 2×2 matrix and $\mathbf{b}_1$ is a vector



$$
\begin{aligned}
y(\mathbf{x}) &= b_2 + \mathbf{w}_2^T(\mathbf{b}_1 + W_1 \cdot \mathbf{x}) \\
&= (b_2 + \mathbf{w}_2^T \mathbf{b}_1) + (\mathbf{w}_2^T W_1) \cdot \mathbf{x} \\
&= \tilde{\mathbf{b}} + \tilde{\mathbf{w}} \cdot \mathbf{x}
\end{aligned}
$$

- Adding layer to linear system is equivalent to original linear system
    - System is over-parameterized, but still won't solve the XOR problem

Nonlinear Activation

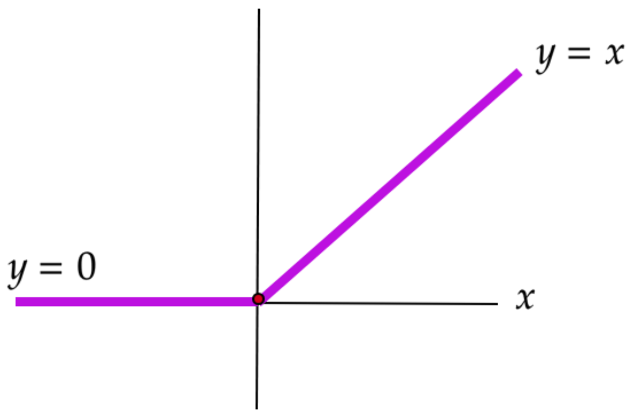
A commonly used nonlinear activation function is the ReLU (Rectified Linear Unit)

$$\mathrm{relu}(x) = \max(0, x)$$

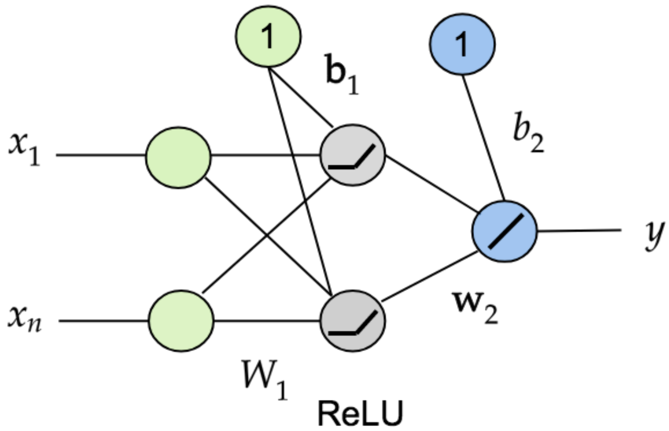
- Clearly nonlinear

$$\mathrm{relu}(-x) \neq -\mathrm{relu}(x))$$

- Convex
- Differentiable everywhere except $x = 0$



Nonlinear Activation



$$y(x) = b_2 + \mathbf{w}_2^T \max(0, \mathbf{b}_1 + W_1\mathbf{x}))$$

Assume W_1 and $\mathbf{b}_1$ are as shown below

$$W_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$b_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

Nonlinear Activation

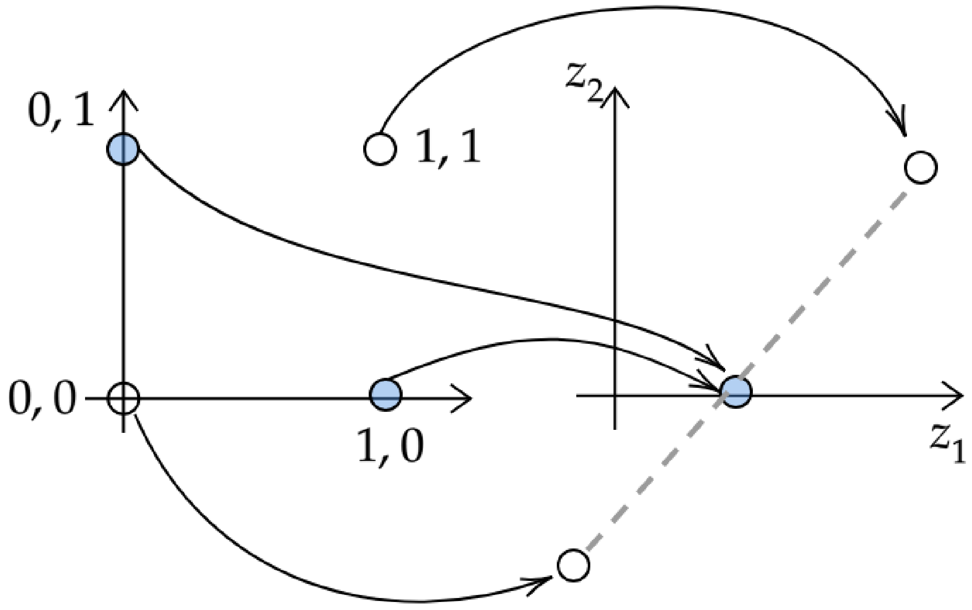
Map the input data to a new intermediate representation

$$Z_1 = \underbrace{\begin{bmatrix} 0 \\ -1 \end{bmatrix}}_{b_1} + \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}}_{W_1} \underbrace{\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}}_{X} = \begin{bmatrix} 0 & 1 & 1 & 2 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

$$A = \mathrm{ReLU}(Z) = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Nonlinear Activation

Linear Activation Mapping of input data before application of ReLU. Data fall on a straight line and still cannot be linearly separated



Still not separable.

Nonlinear Activation

Mapping of input data after application of ReLU. New representation shows the data can now be linearly separated

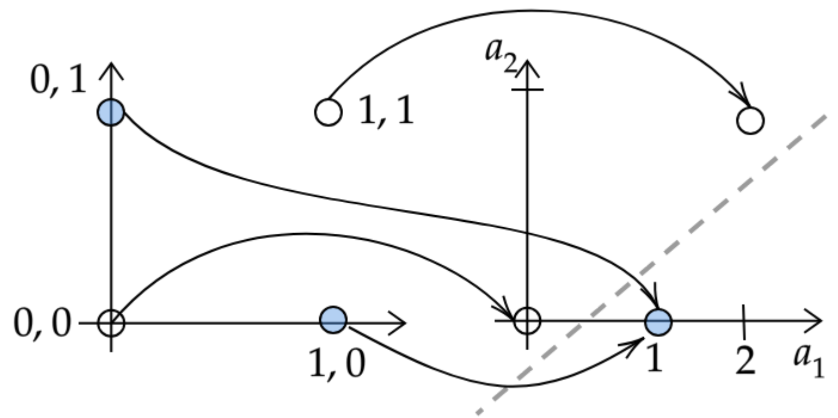
Setting

$$b_2 = 0, \quad \mathbf{w}_2 = [\,1 \;\; -2\,]$$

gives exact mapping to correct tags

$$\mathbf{w}_2^T A = [\,0 \quad 1 \quad 1 \quad 0\,]$$

This would not be possible without the ReLU



Universal Approximation

Weierstrass approximation theorem:

Suppose f is a continuous-valued function defined on the real interval $[a, b]$. For every $\epsilon > 0$, there exists a polynomial p such that for all x in $[a, b]$ we have $|f(x) - p(x)| < \epsilon$

- Can approximate continuous functions with polynomials.
- Kolmogorov showed that a continuous function of several variables can be represented exactly by a superposition of continuous one-dimensional functions of the original input variables.
    - Sprecher and Hecht-Nielsen introduced the result to the neural network community.
    - There are questions about applicability due to lack of smoothness in theoretical results

Universal Approximation

Neural networks with 1 hidden layer are universal approximators

Universal approximation property is not rare:

- Polynomials
- Trig polynomials (Fourier series)
- Kernels
- Wavelets

Neural networks are powerful enough to approximate most functions

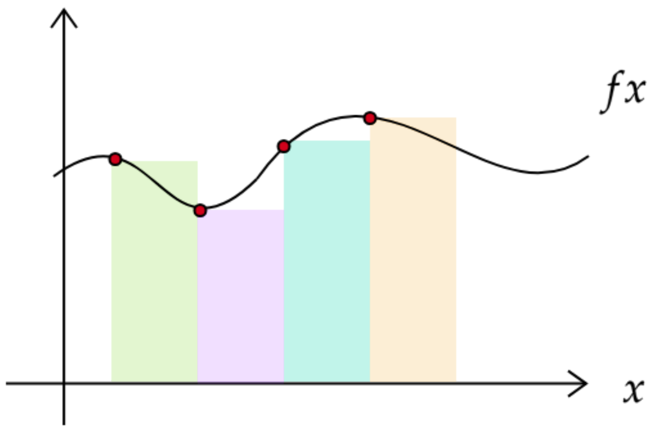
- But this property is not unique

Existence proofs only:

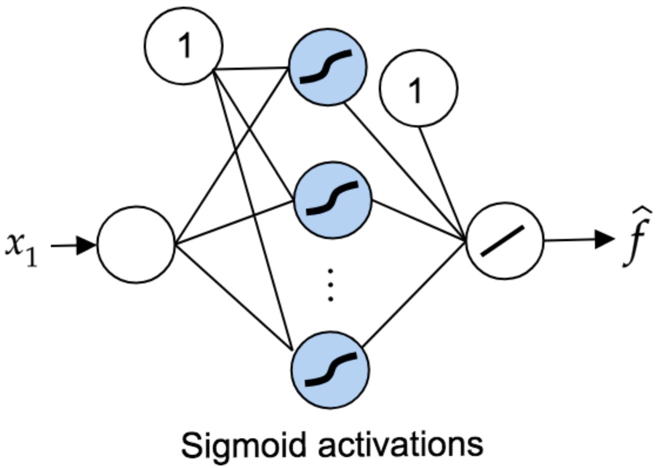
- Does not put limit on number of nodes needed.
- Does not show how to find parameters .

If target function is discontinuous then two layers are needed.

Universal Approximation



A function f can be approximated by piece-wise constant functions. The approximation can be close if partition is fine enough.



Will show that pairs of sigmoid nodes can form a partition

Universal Approximation

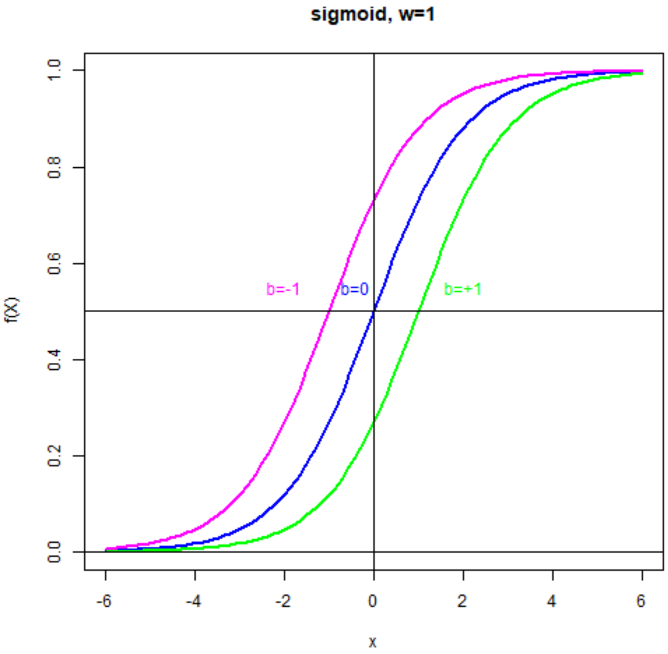
$$\sigma(x; b, w) = \frac{1}{1 + e^{-b-wx}}$$

$$\sigma(0) = \frac{1}{2}$$

$$-b - wx = 0$$

$$x_0 = -\frac{b}{w}$$

Shows adjusting b moves the sigmoid curve left/right

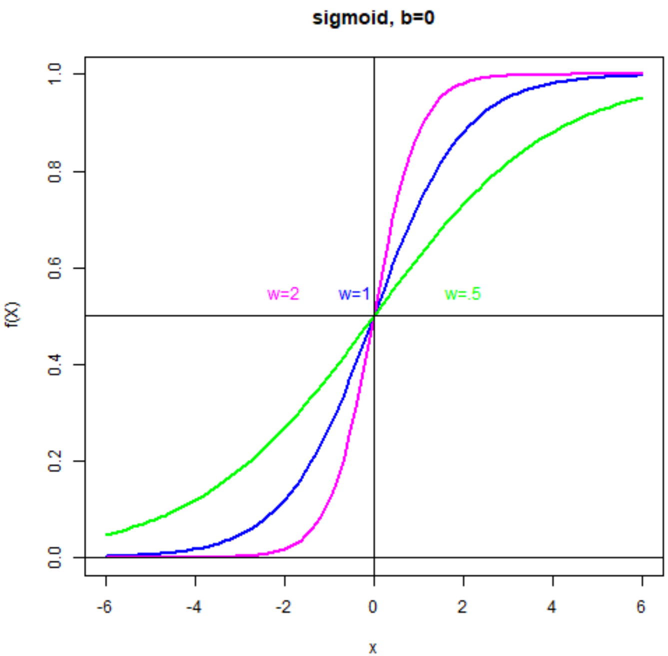


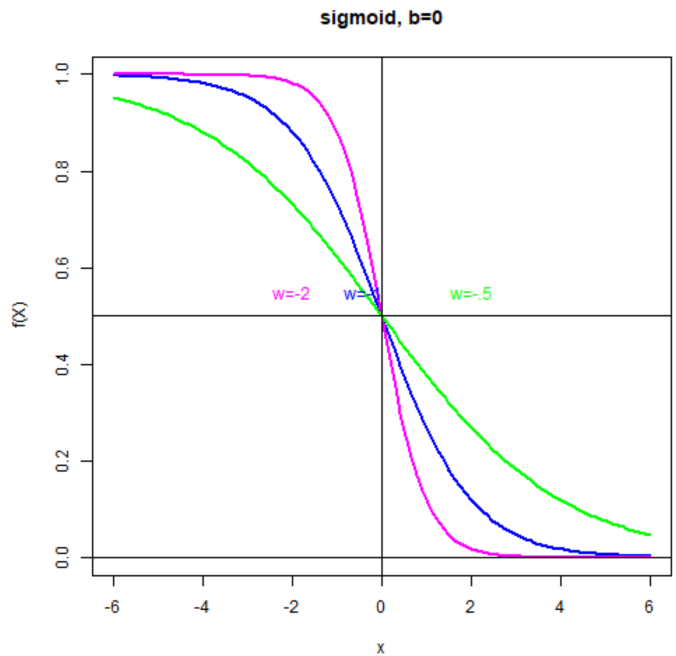
Universal Approximation

$$\frac{\partial \sigma}{\partial x} = \frac{\partial \sigma}{\partial z}\frac{\partial z}{\partial x} = \sigma(1-\sigma)\cdot w$$

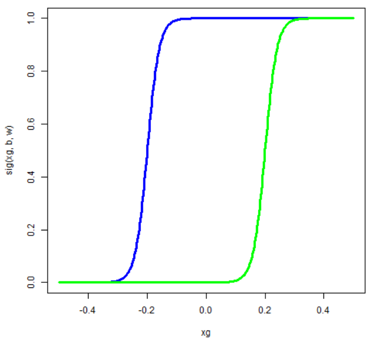
$$\sigma > 0, \qquad (1-\sigma) > 0$$

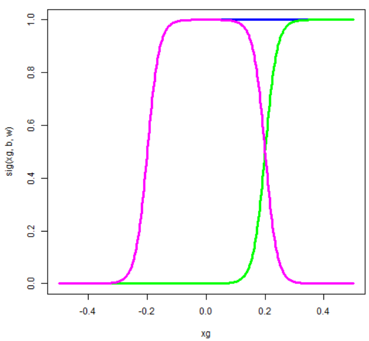
The sign and magnitude of w controls the slope



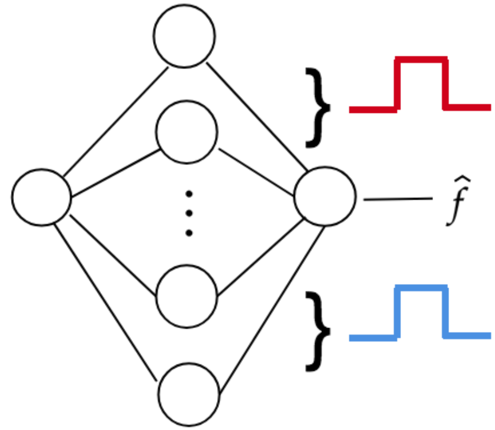


Universal Approximation





Blue curve minus green curve approximates a square wave



Each pair of nodes approximates a small part of f

Demonstrates that a network with enough nodes can approximate a function f. It does not mean that gradient descent will necessarily find the solution that approximates f.

Universal Approximation

What if output activation is sigmoid instead of identity?

Same problem but now estimate hidden layer parameters b, w to fit steps of

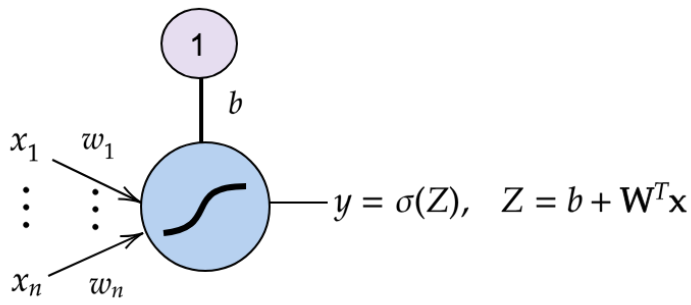
$$\sigma^{-1}(f(x))$$

On output get

$$\sigma(\sigma^{-1}(f(x)) = f(x)$$

See Nielsen for an argument on the approximation of functions $f : \mathbb{R}^m \to \mathbb{R}^n$

Universal Classifier



For a single neuron, the output value $\sigma \to 0$ as $\mathbf{z} \to -\infty$ and $\sigma \to 1$ as $\mathbf{z} \to +\infty$

The gradient of $\sigma(z)$ is in the direction of $\mathbf{w}$:

$$\frac{\partial \sigma}{\partial \mathbf{x}} = \sigma' \frac{\partial z}{\partial \mathbf{x}} = \sigma(1-\sigma)\mathbf{w}$$

The directional derivative is:

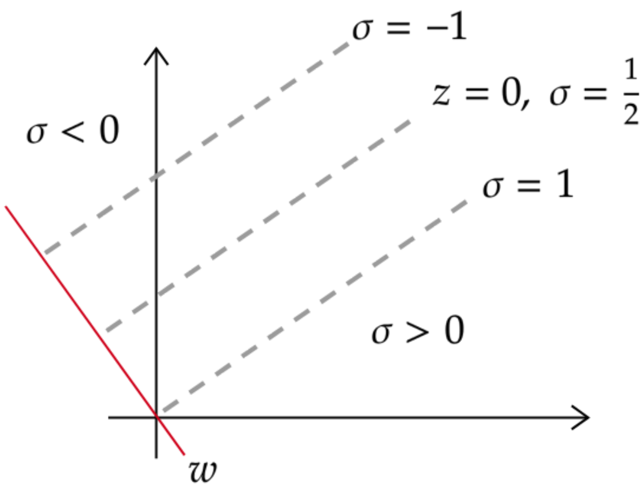
$$\partial_{\mathbf{u}}\, \sigma = \sigma(1-\sigma)\hat{\mathbf{u}} \cdot \mathbf{w}$$

It follows that σ is a constant along the lines $\perp$ to $\mathbf{w}$ and the magnitude of the gradient is

$$\left\| \frac{\partial \sigma}{\partial \mathbf{x}} \right\| = \sigma(1-\sigma)\, \|\mathbf{w}\|$$

Universal Classifier

Can create arbitrarily precise transitions across a decision boundary



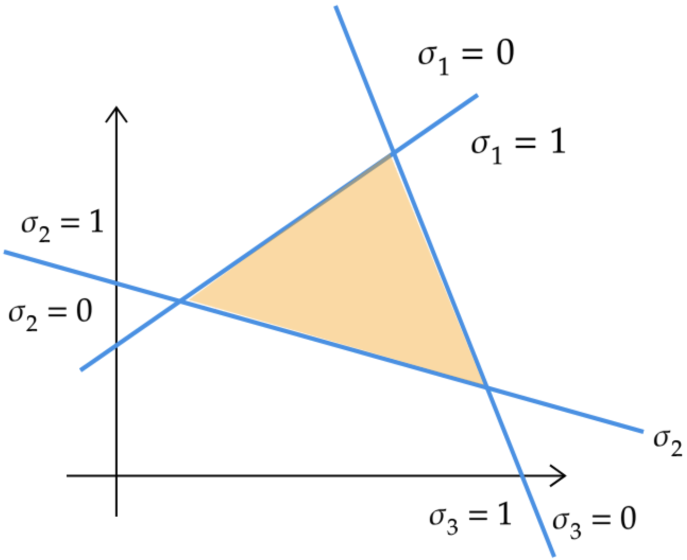
The logistic activation function:

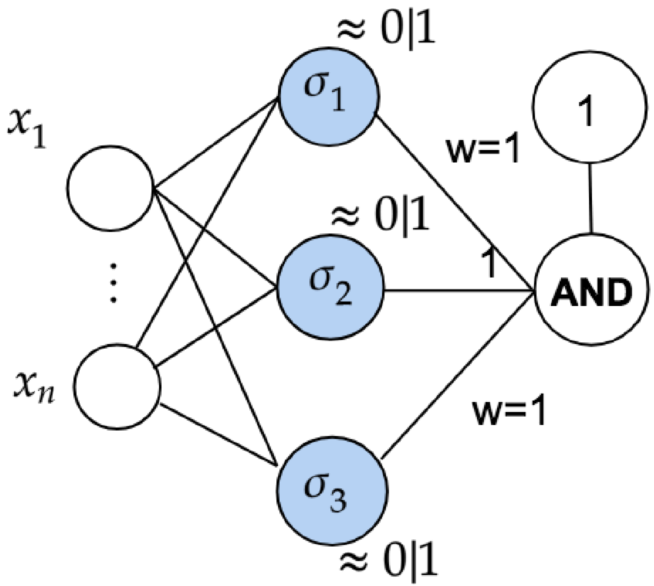
- Has a linear decision boundary
- Is constant along lines $\perp$ to $\mathbf{w}$
- The gradient has magnitude $\frac{1}{4}\|\mathbf{w}\|$ at decision boundary
- The transition from 0 to 1 can be made arbitrarily abrupt by increasing $\|\mathbf{w}\|$
- The decision boundary distance from the origin along $\mathbf{w}$ is

$$b + \mathbf{w}^T(\alpha \frac{\mathbf{w}}{\|\mathbf{w}\|}) = 0$$

$$\alpha = -\frac{b}{\|\mathbf{w}\|}$$

Universal Classifier





If output σ makes rapid transition from 0 to 1 at $\sum \mathbf{x}_i = 2.5$ then this portion of network will classify triangular region correctly

Universal Classifier