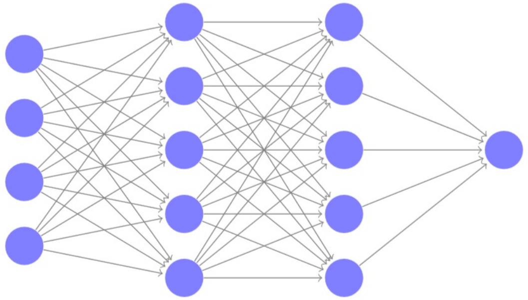
ENLP Lecture 14

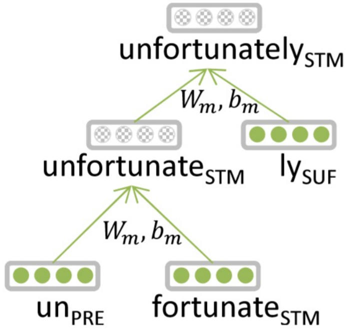
Deep Learning & Neural Networks

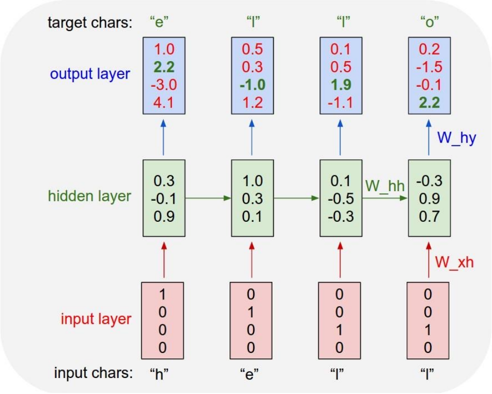
Austin Blodgett & Nathan Schneider

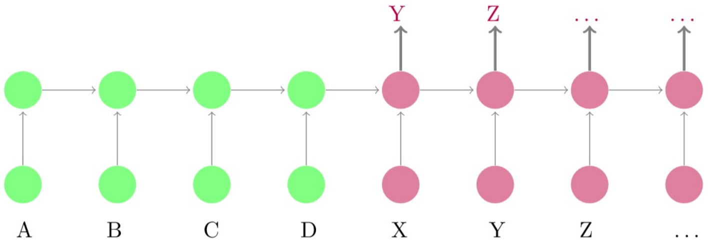
ENLP | March 18, 2021

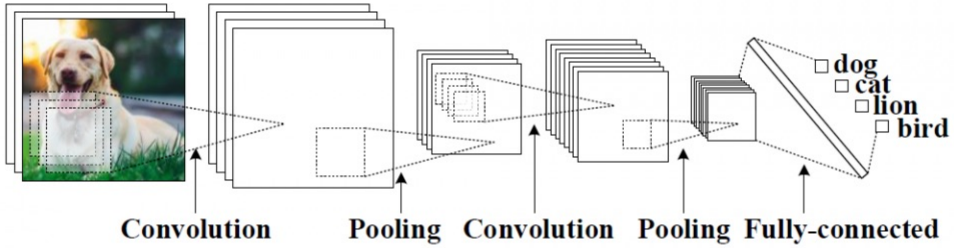
a family of algorithms











NN Task	Example Input	Example Output
Binary classification		
Multiclass classification		
Sequence		
Sequence to Sequence		
Tree/Graph Parsing		

NN Task	Example Input	Example Output
Binary classification	features	+/-
Multiclass classification	features	decl, imper, …
Sequence	sentence	POS tags
Sequence to Sequence	(English) sentence	(Spanish) sentence
Tree/Graph Parsing	sentence	dependency tree or AMR parsing

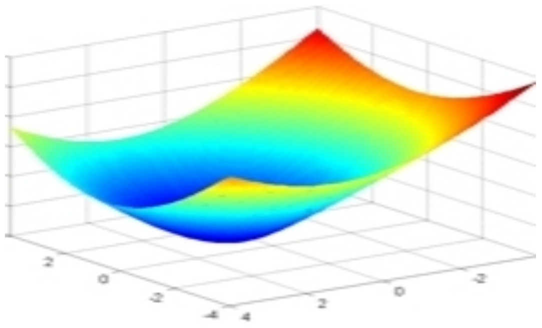
2. What's Deep Learning (DL)?

- **Deep learning** is a subfield of **machine learning**

- Most machine learning methods work well because of **human-designed representations** and **input features**
  - For example: features for finding named entities like locations or organization names (Finkel et al., 2010):

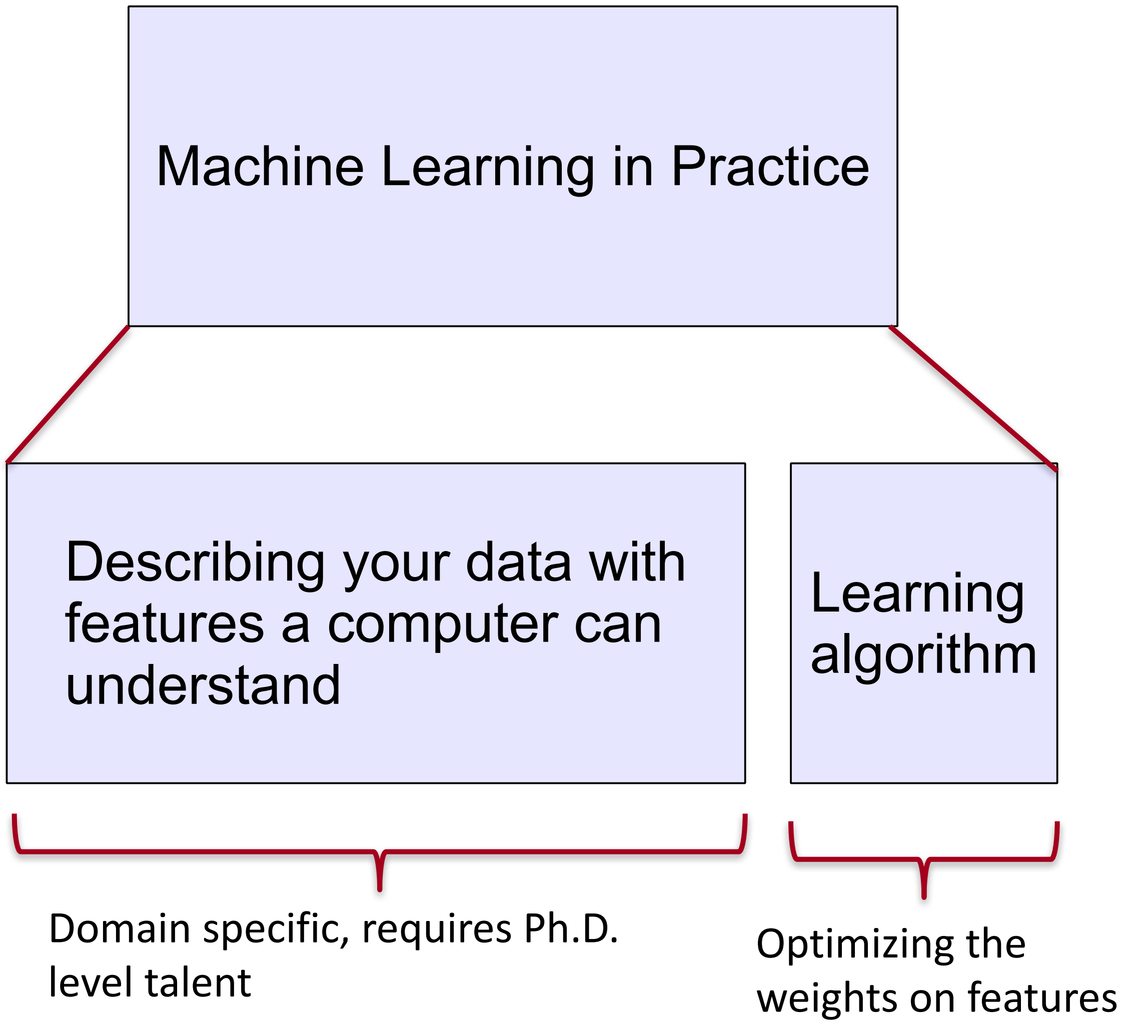
Feature	NER
Current Word	✓
Previous Word	✓
Next Word	✓
Current Word Character n-gram	all
Current POS Tag	✓
Surrounding POS Tag Sequence	✓
Current Word Shape	✓
Surrounding Word Shape Sequence	✓
Presence of Word in Left Window	size 4
Presence of Word in Right Window	size 4

- Machine learning becomes just optimizing weights to best make a final prediction



(Slide from Manning and Socher)

Machine Learning vs. Deep Learning



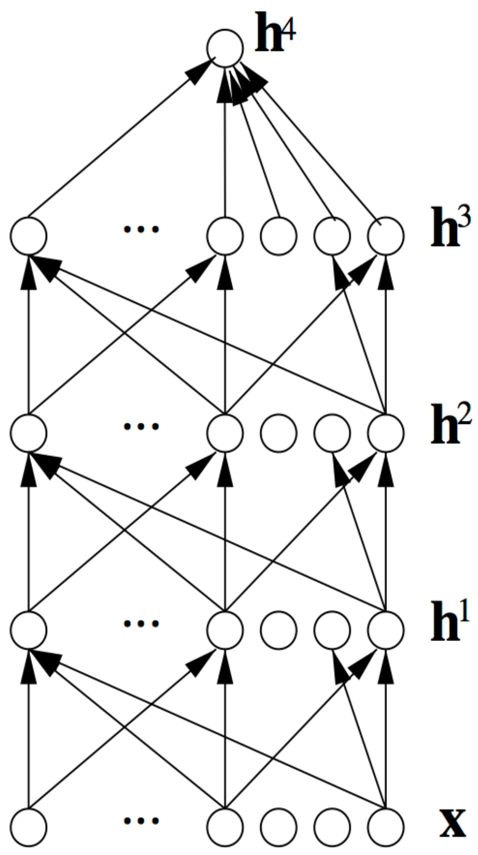
(Slide from Manning and Socher)

What's Deep Learning (DL)?

- Representation learning attempts to automatically learn good features or representations

- Deep learning algorithms attempt to learn (multiple levels of) representation and an output

- From "raw" inputs **x** (e.g., sound, characters, or words)



(Slide from Manning and Socher)

On the history of and term "Deep Learning"

- We will focus on different kinds of **neural networks**
- The dominant model family inside deep learning

- We will not take a historical approach but instead focus on methods which work well on NLP problems now
- For a long (!) history of deep learning models (starting ~1960s), see: Deep Learning in Neural Networks: An Overview by Jürgen Schmidhuber

(Slide from Manning and Socher)

Reasons for Exploring Deep Learning

- In ~2010 **deep** learning techniques started outperforming other machine learning techniques. Why this decade?

- Large amounts of training data favor deep learning
- Faster machines and multicore CPU/GPUs favor Deep Learning
- New models, algorithms, ideas
  - Better, more flexible learning of intermediate representations
  - Effective end-to-end joint system learning
  - Effective learning methods for using contexts and transferring between tasks

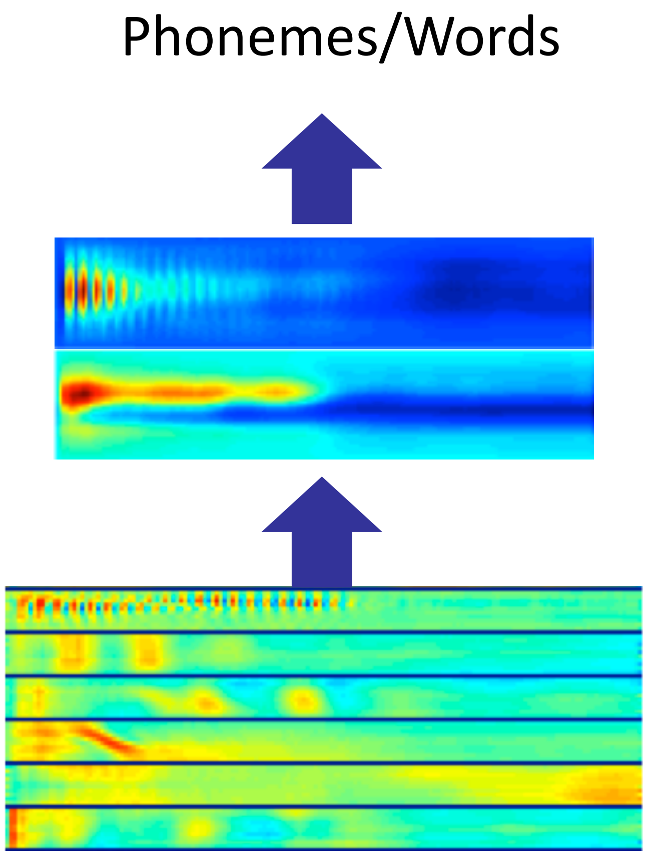
→ **Improved performance** (first in speech and vision, then NLP)

(Slide from Manning and Socher)

Deep Learning for Speech

- The first breakthrough results of "deep learning" on large datasets happened in speech recognition
- Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition Dahl et al. (2010)

Acoustic model	Recog WER	RT03S FSH	Hub5 SWB
Traditional features	1-pass −adapt	**27.4**	**23.6**
Deep Learning	1-pass −adapt	**18.5** (−33%)	**16.1** (−32%)

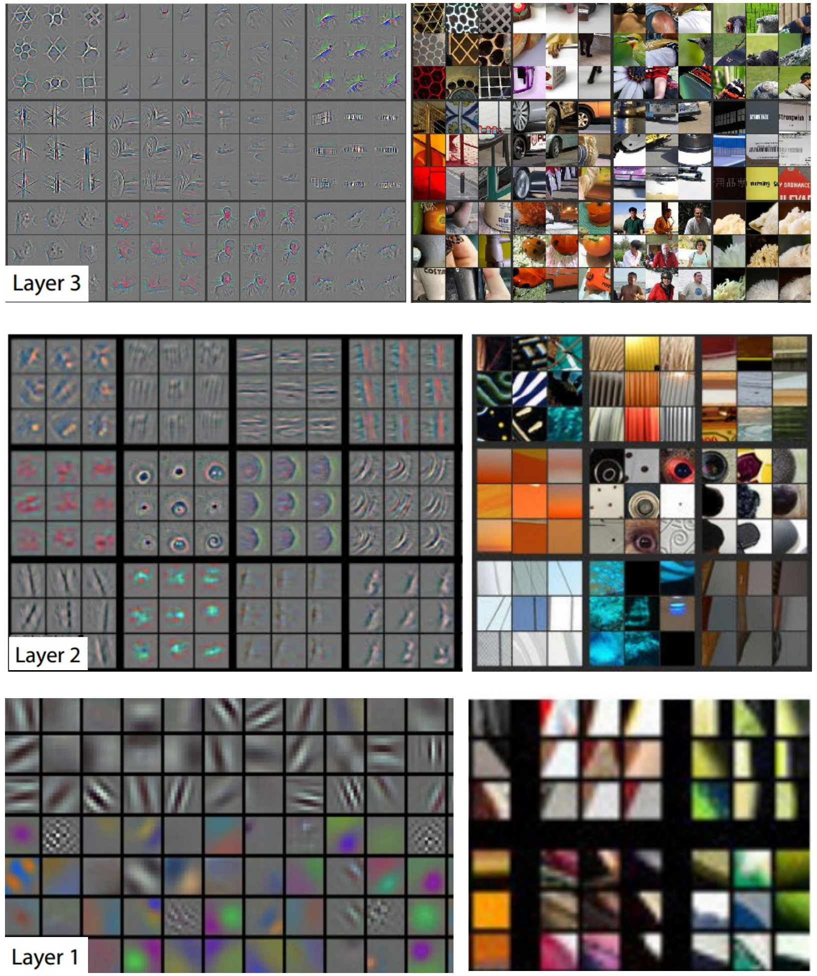


(Slide from Manning and Socher)

Deep Learning for Computer Vision

Most deep learning groups have focused on computer vision (at least till 2 years ago)

The breakthrough DL paper: ImageNet Classification with Deep Convolutional Neural Networks by Krizhevsky, Sutskever, & Hinton, 2012, U. Toronto. 37% error red.

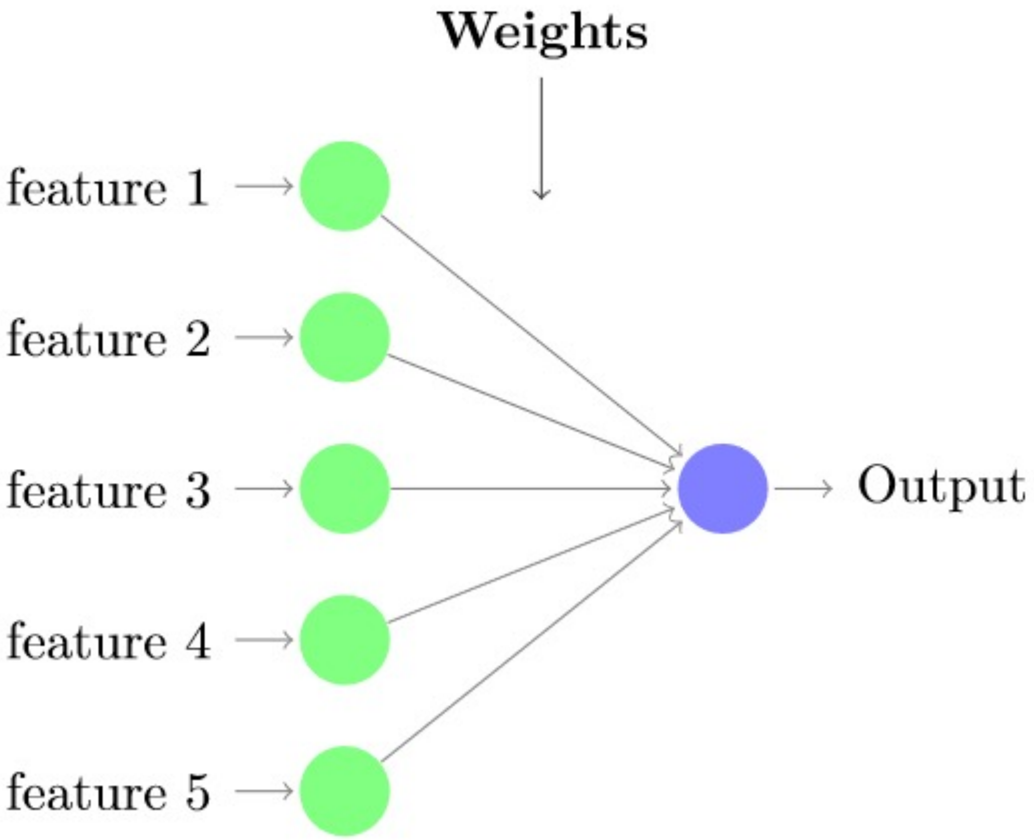


Zeiler and Fergus (2013)

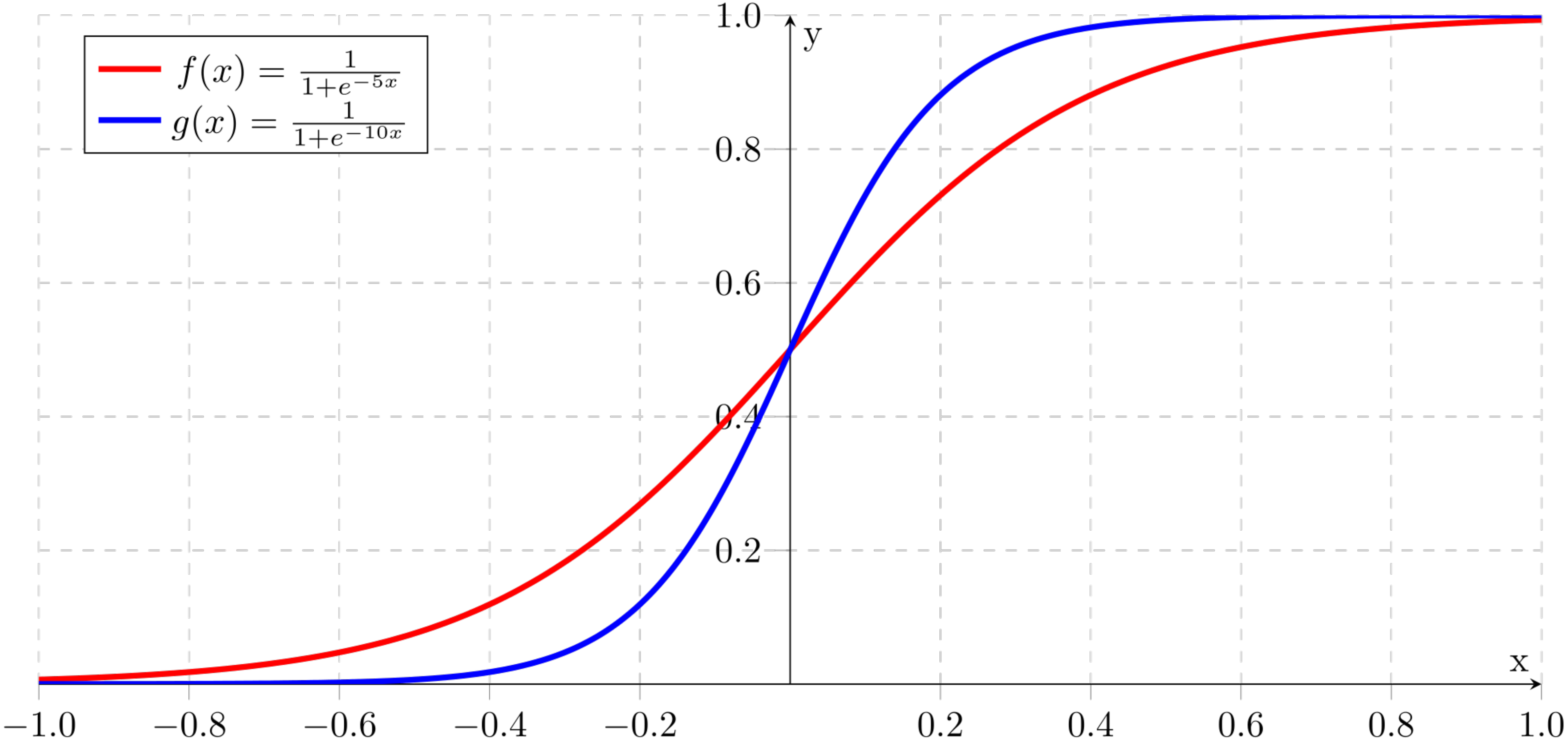
(Slide from Manning and Socher)

Perceptron

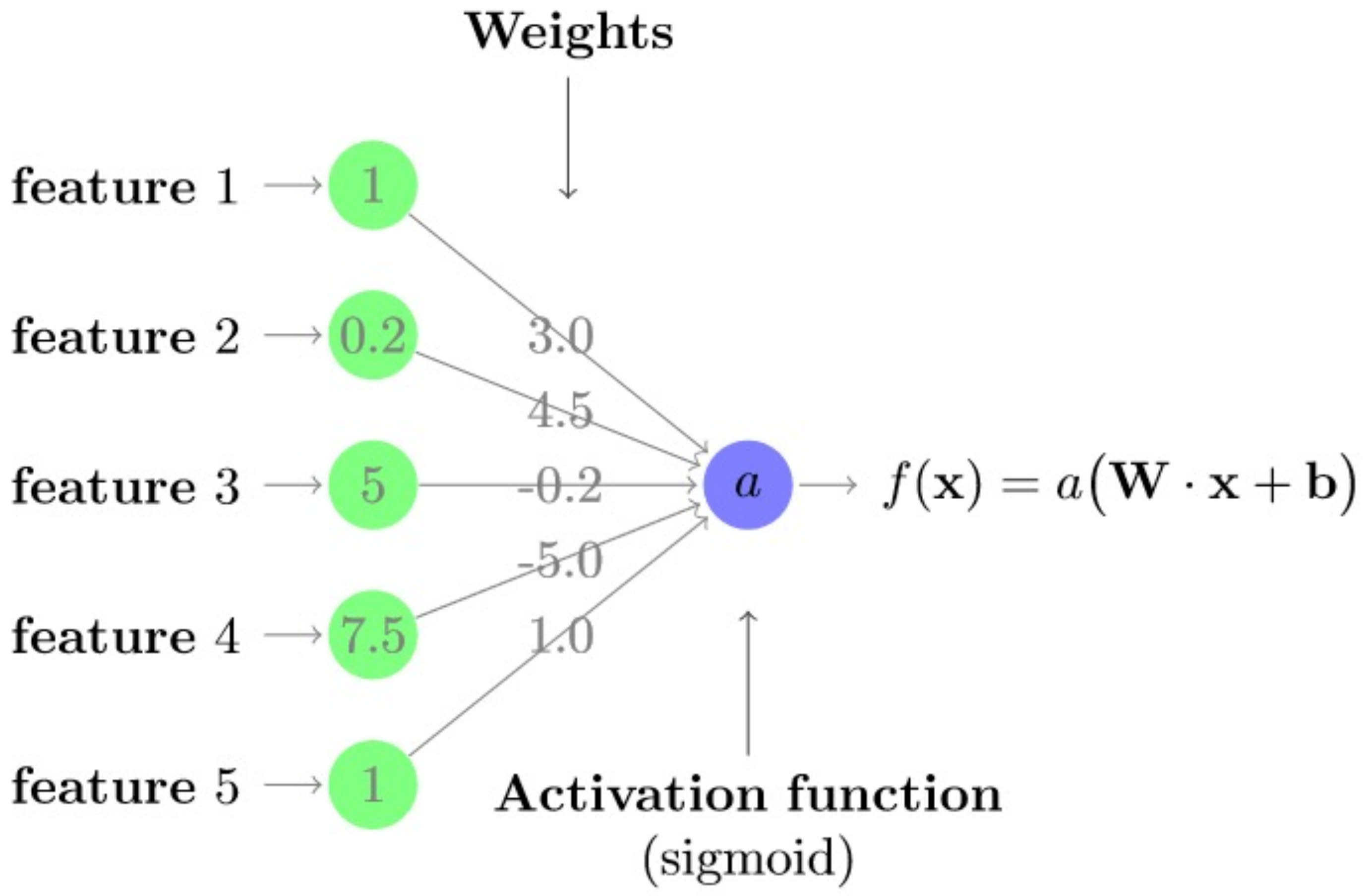
(as in the classifier, not the learning algorithm)



Sigmoid Activation Function

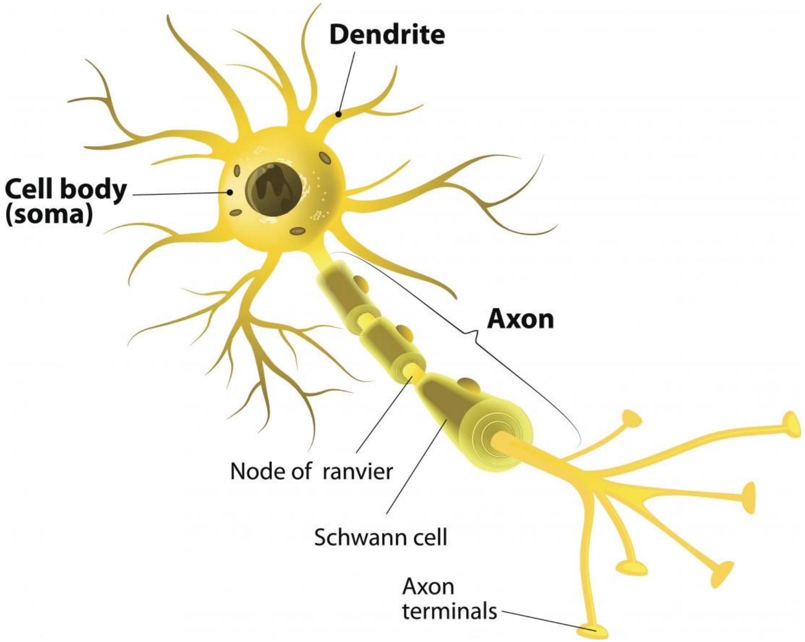


Perceptron


Weights
feature 1
feature 2
feature 3
feature 4
feature 5
1
0.2
5
7.5
1
3.0
4.5
-0.2
-5.0
1.0
a
$f(\mathbf{x}) = a(\mathbf{W} \cdot \mathbf{x} + \mathbf{b})$
Activation function
(sigmoid)

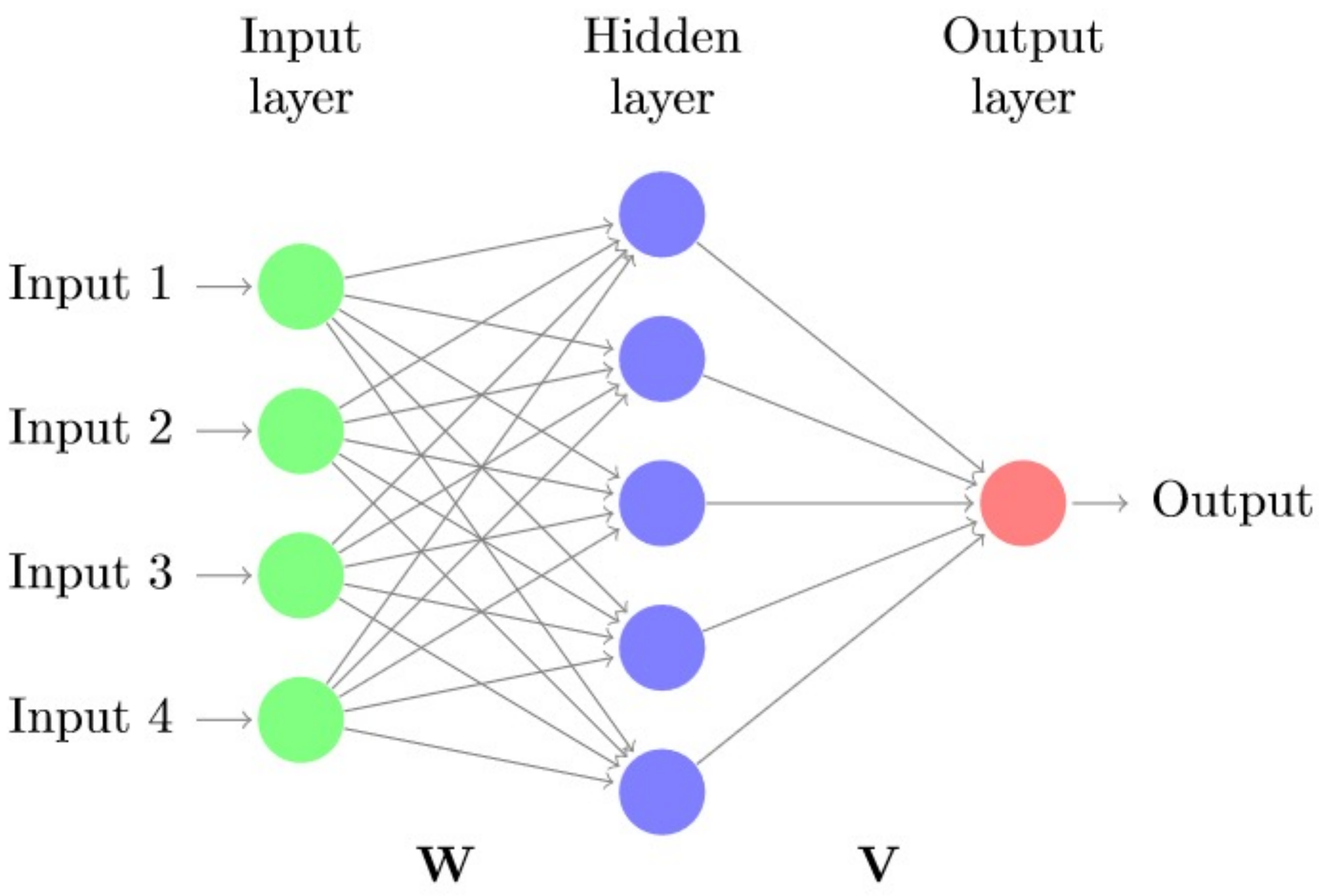
"Neuron"

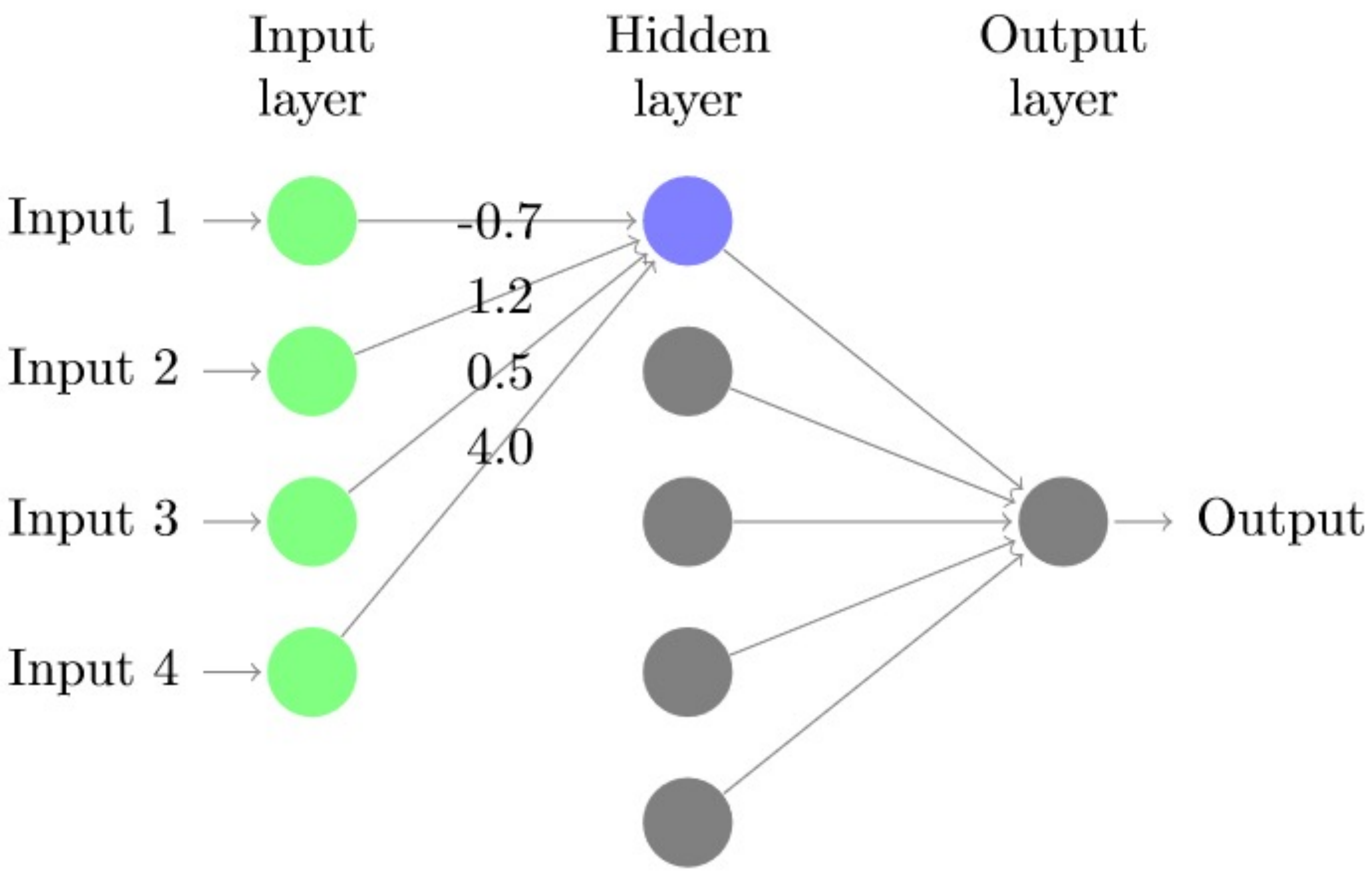
- A biological neuron receives electric signals as input and uses them to compute an electrical signal as output
- The perceptron in an artificial neural network is **loosely inspired** by the biological neuron
- The artificial neural networks we use for machine learning are NOT models of the brain!

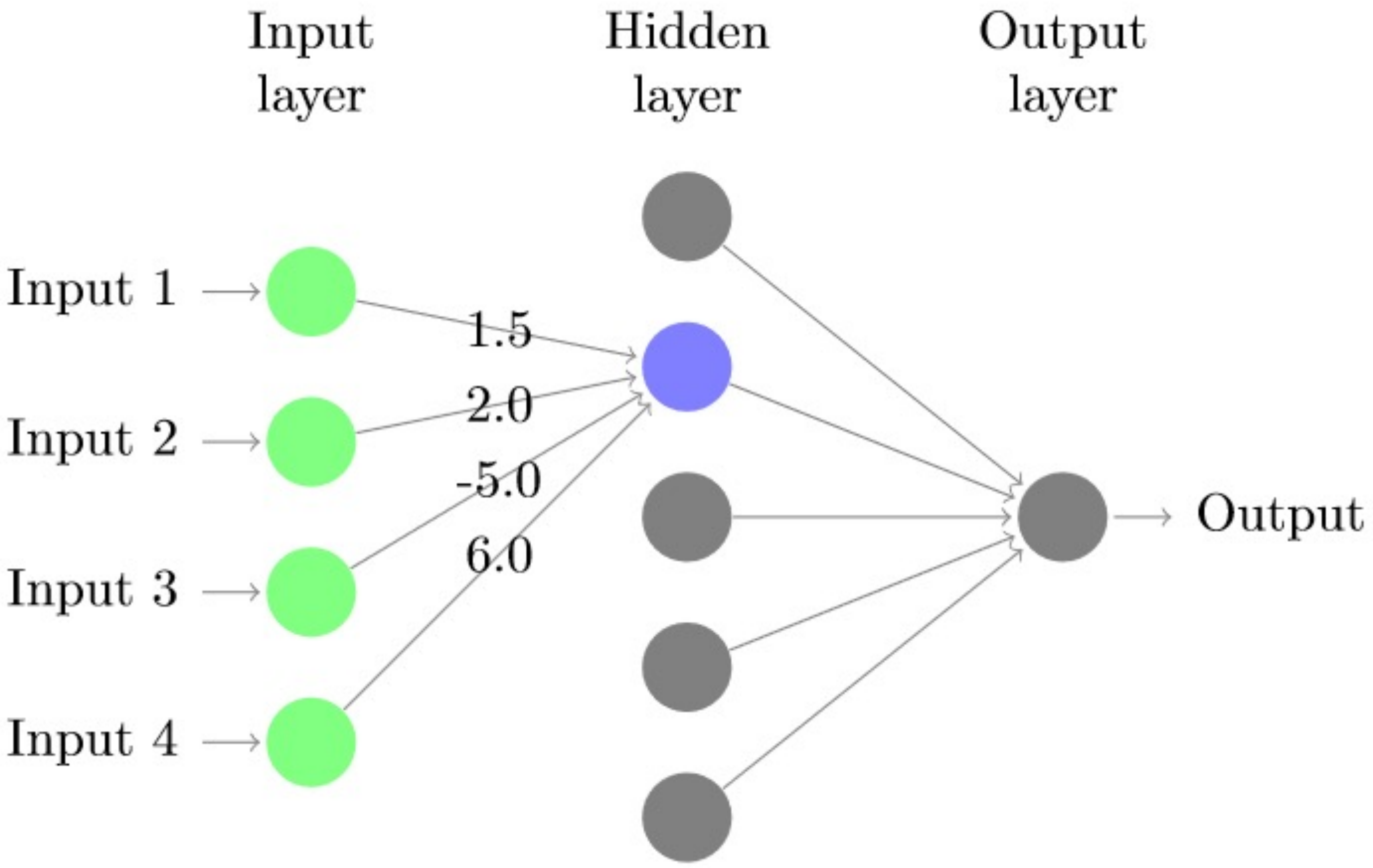


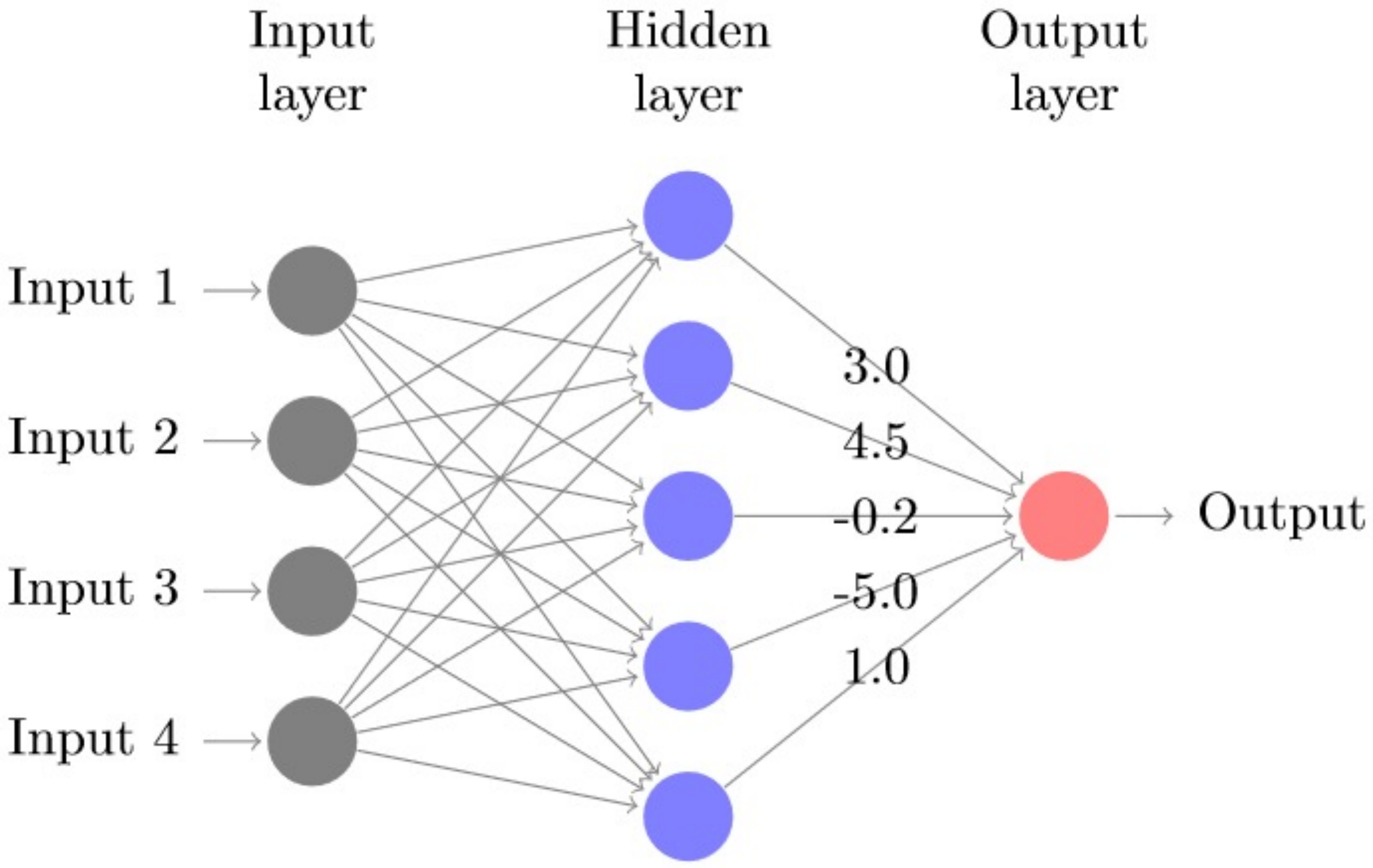
FFNNs

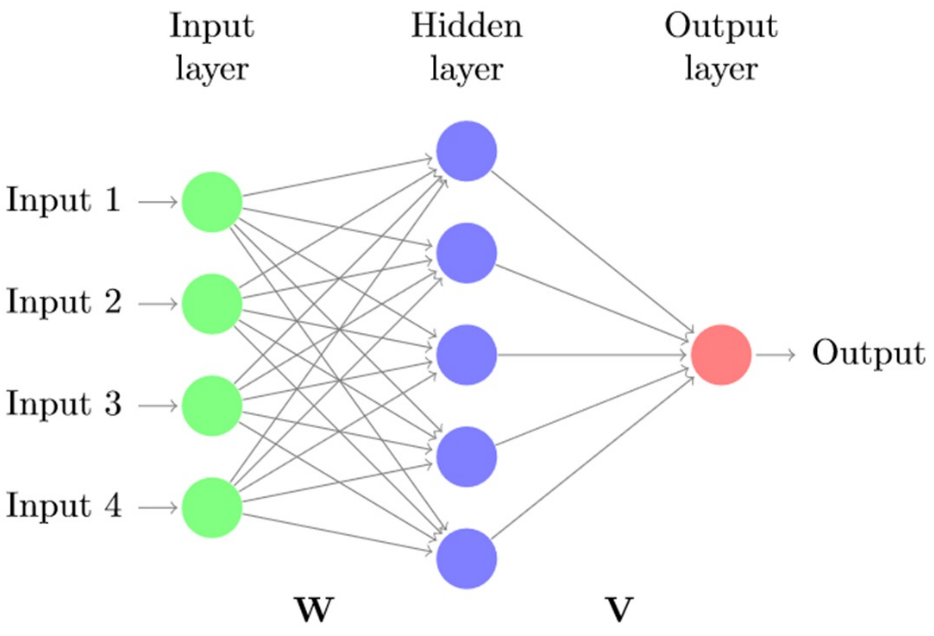
- **Feed Forward Neural Net** – Multiple layers of neurons
- *Can solve non-linearly separable problems*
- (All arrows face the same direction)
- Applications:
  - *Text classification* – sentiment analysis, language detection, …
  - *Unsupervised learning* – dimension reduction, word2vec


Input
layer
Hidden
layer
Output
layer
Input 1
Input 2
Input 3
Input 4
Output
W
V


Input
layer
Hidden
layer
Output
layer
Input 1
Input 2
Input 3
Input 4
-0.7
1.2
0.5
4.0
Output

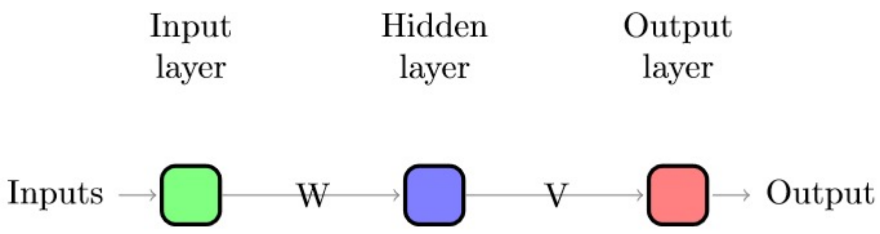

Input layer
Hidden layer
Output layer
Input 1
Input 2
Input 3
Input 4
1.5
2.0
-5.0
6.0
Output


Input
layer
Hidden
layer
Output
layer
Input 1
Input 2
Input 3
Input 4
3.0
4.5
-0.2
-5.0
1.0
Output



=

Compact diagram



FAQ

- How do I interpret an NN?
  - An NN performs *function approximation*
  - Connections in an NN posit *relatedness*
  - Lack of connection posits *independence*

FAQ

- What do the weights mean?
  - *Functional perspective* – these weights optimize NN's task performance
  - *Representation perspective* – weights represent **unlabeled, distributed** knowledge (*useful* but not generally *interpretable*)

FAQ

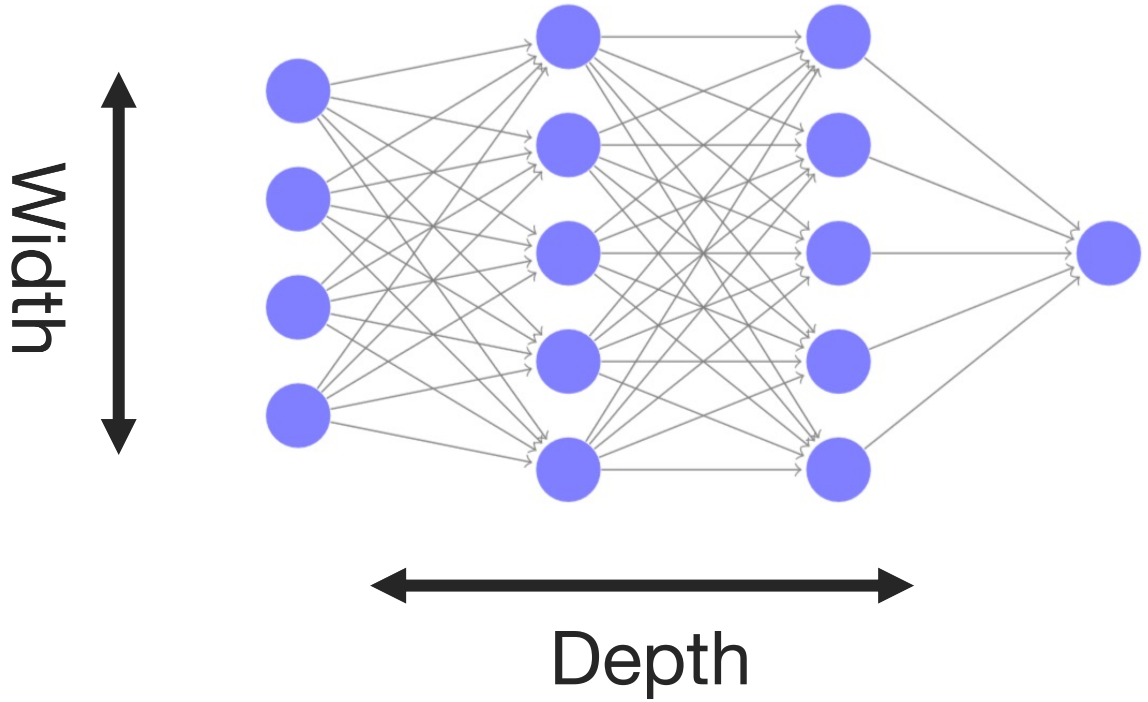
- Can an NN learn anything?
  - No, but …

    Theorem: ‘One hidden layer is enough to represent (*not learn*) an approximation of any function to an arbitrary degree of accuracy’

  - *(Given infinite training data, memory, etc.)*

FAQ

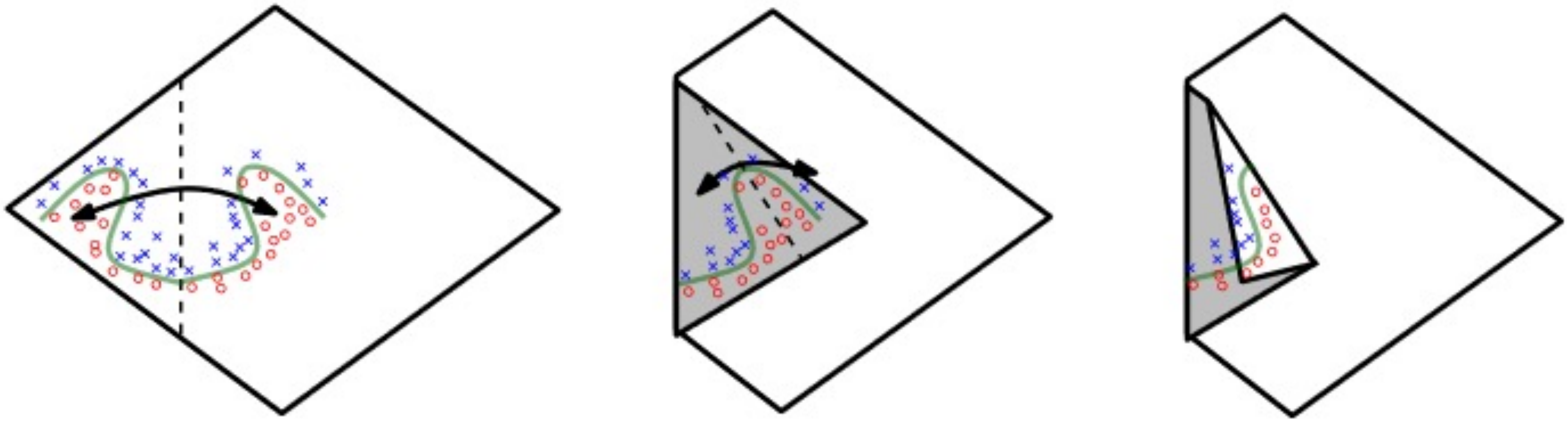
- What happens if I make an NN deeper?



Width controls overfitting/underfitting

Depth allows complex functions, can reduce overfitting

Exponential Representation Advantage of Depth

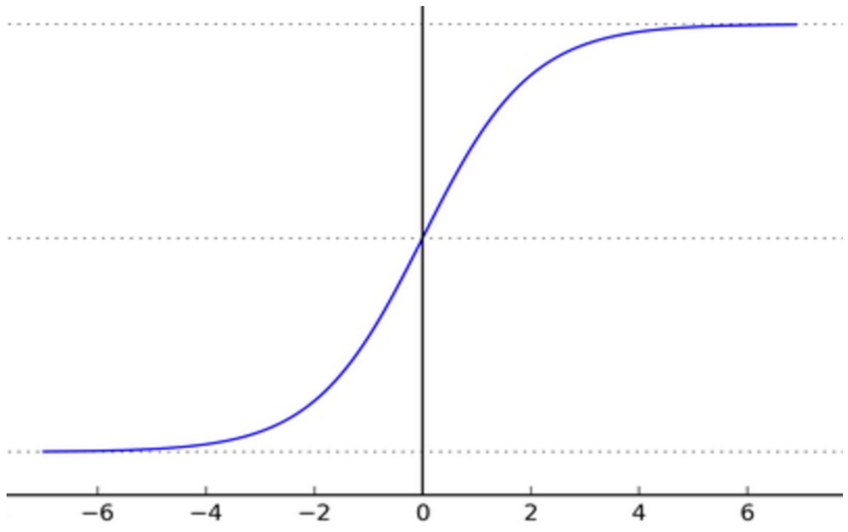


(Goodfellow 2017)

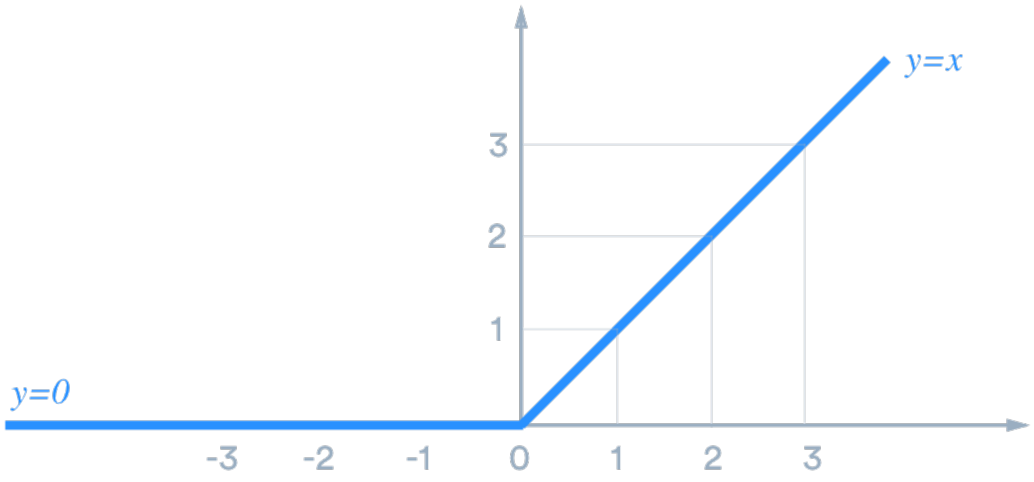
activation functions

- **Activation function –** "squishes" neuron inputs into an output
  - Use in output layer – *Sigmoid (binary class), Softmax (Multiclass)*
  - Use in hidden layers – *ReLU, Leaky ReLU*

Sigmoid



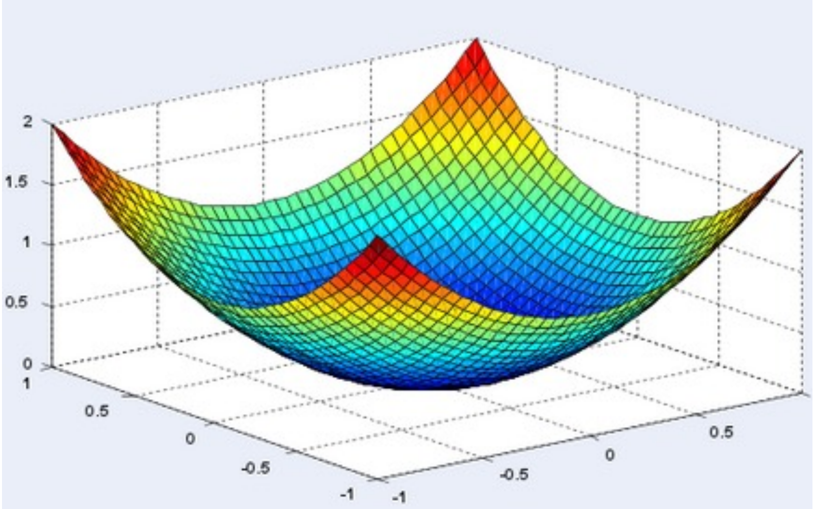
ReLU (**Re**ctified **L**inear **U**nit)



training

- To train an NN, you need:
  - **Training set -** ordered pairs each with an input and target output
  - **Loss function -** a function to be optimized, e.g. *Cross Entropy*
  - **Optimizer** - a method for adjusting the weights, e.g. *Gradient Descent*

Gradient Descent – use gradient to find lowest point in a function



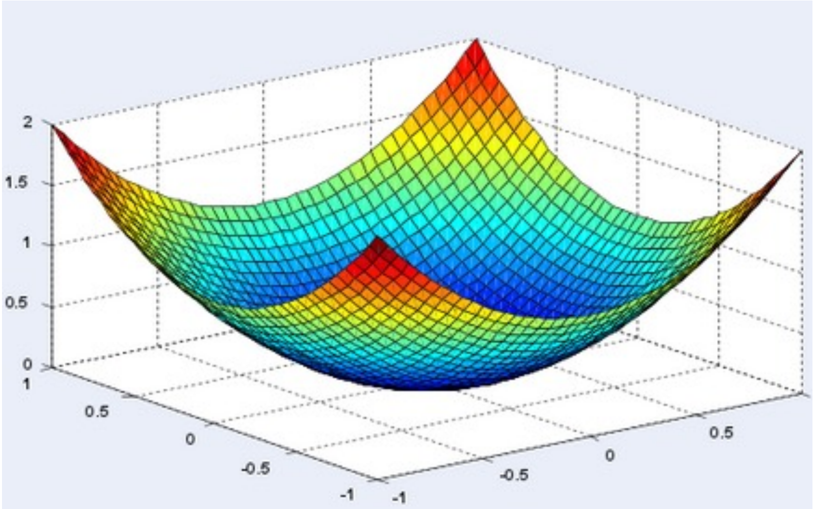
backpropagation

- **Backpropagation =** Chain Rule + Dynamic Programming

Loss function – measures NN's performance.

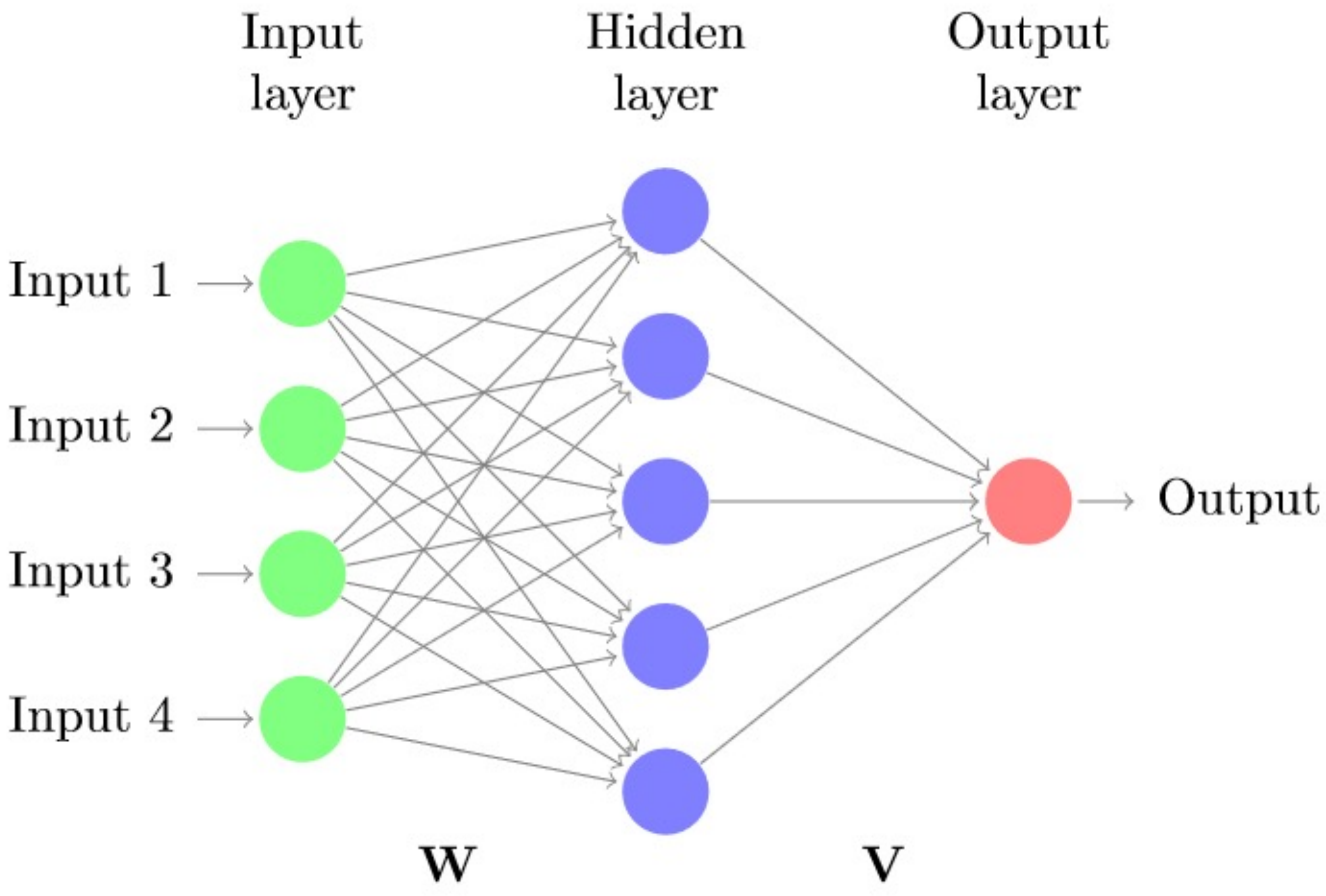
Adjust weights by gradient (using a *learning rate*) of the loss. Save repeated partial computations along the way.

$$\Delta w_i = \frac{\partial}{\partial w_i} Loss(f(\mathbf{W}, \mathbf{V}, \ldots, \mathbf{x}), target)$$



loss functions

- **Loss function –** measures NN’s performance.
  - Probabilistic interpretation
    - Binary output - **Binary Cross Entropy** and Sigmoid
    - Multiclass/Sequence output - **Categorical Cross Entropy** and Softmax
    - either *Generative* or *Discriminative*
  - Geometric interpretation
    - **Mean Squared Error** or **Hinge Loss** (like in Structured Perceptron)


Input
layer
Hidden
layer
Output
layer
Input 1
Input 2
Input 3
Input 4
Output
W
V

Embeddings

- **Embeddings -** Dense vector representations of *words, characters, documents, etc.*
- *Used as input features for most Neural NLP models*
- Prepackaged – *word2vec, GloVe*
- Use pre-trained word embeddings *and* train them yourself!
- Pretrained models that give **contextualized** word embeddings: *ELMo*, *BERT*, *OpenAI GPT-2*

Word meaning as a neural word vector – visualization

$$expect = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$

(Slide from Manning and Socher)

Some References

- **NN Packages –** *TensorFlow*, *PyTorch*, *Keras*
- **Some Books**
  - Goldberg book (free from Georgetown)
  - Goodfellow book (Chapters and Videos)

Other architectures

- The layout of a network is called the **architecture**.
- Vanilla architecture: **Feed-forward**, with every node in the 1st layer connected to every node in the 2nd layer, etc.,
- Other architectures: **convolutional**, **recurrent**, …