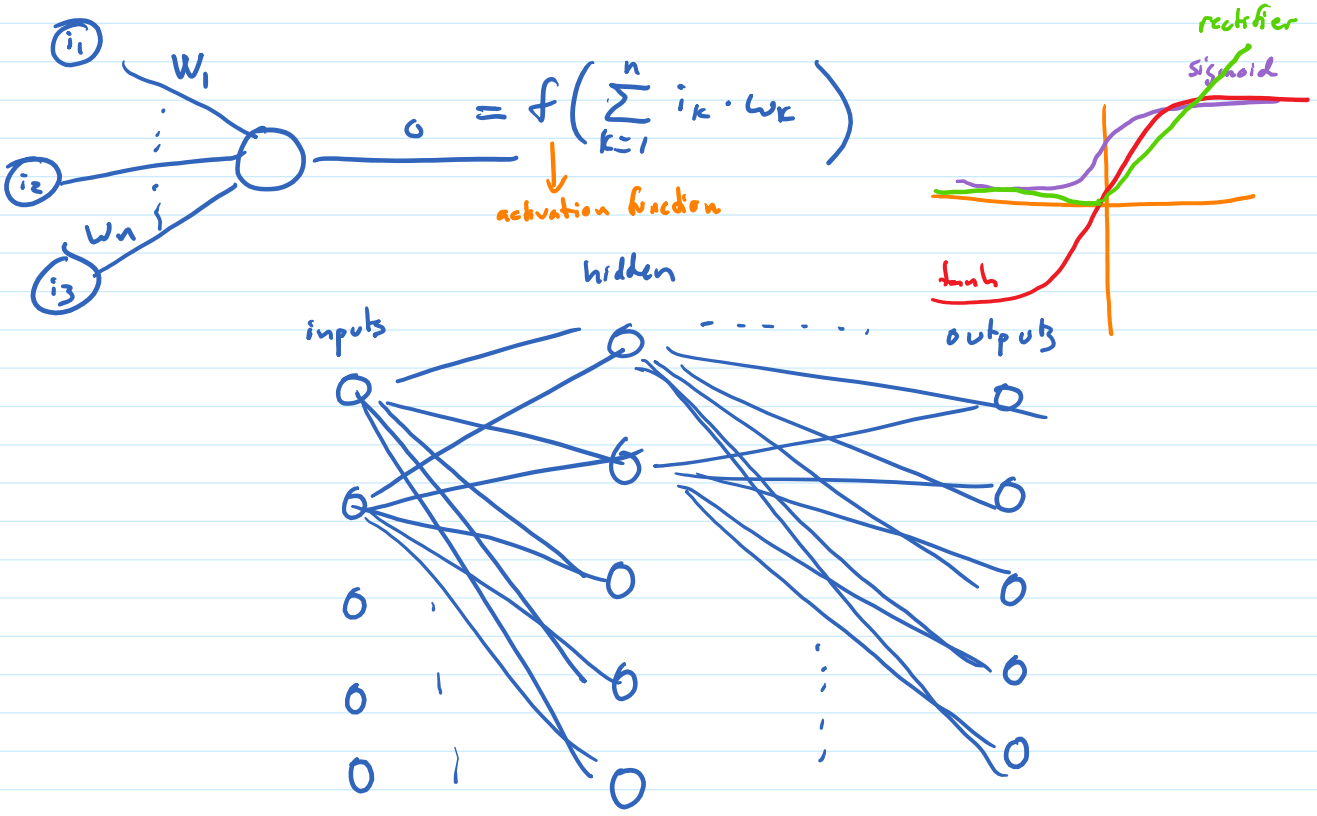
ANN Supervised Learning



supervised learning : uses example inputs with known correct outputs

initialize weights randomly

until sufficiently trained
- compute output for examples
- compute distance between network's output and correct output (ex: mean squared error)
- adjust weights to decrease that distance — ex: gradient descent back propagation

split examples into training data ~70%, test data (validation data)

"sufficiently trained" can be "until no improvement on test data"

x_1	x_2	x_3	
3	1.4	6	A
2.9	2.0	7	B
3.1	1.0	5	A

if $x_1 = 3$ and $x_2 = 1.4$ and $x_3 = 6$
  output A
else if $x_1 = 2.9$ and $x_2 = 2.0$ and $x_3 = 7$
  output B
  ⋮

overfitting

input/output representation

class input in $\{C_1, \ldots, C_k\}$ → 1 input unit for each class indicating membership in class

$0 - \frac{2}{3}$
$\frac{2}{3} - \frac{4}{3}$

0 1 2
{empty, white, black}

0.5 means equally likely empty or white

1.5 equally white or black

?? empty or black

inputs				
	empty	1	0	0
	white	0	1	0
	black	0	0	1
output		0.7	0.2	0.1

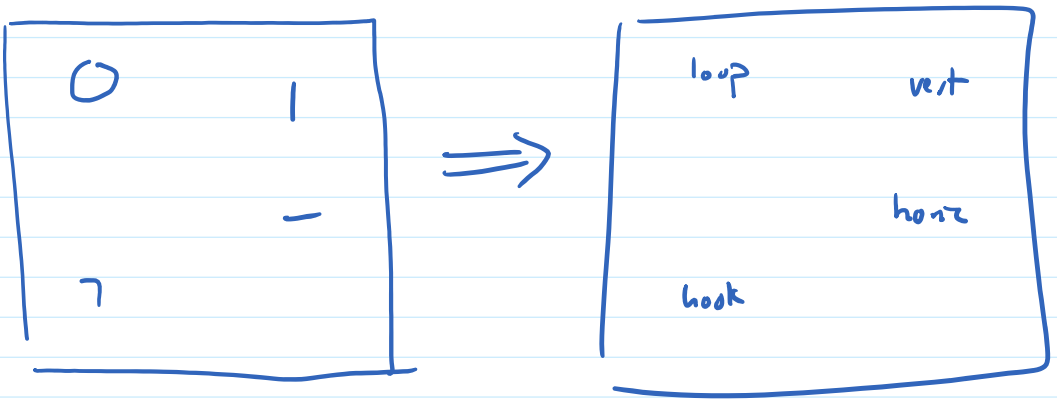
gives prob dist

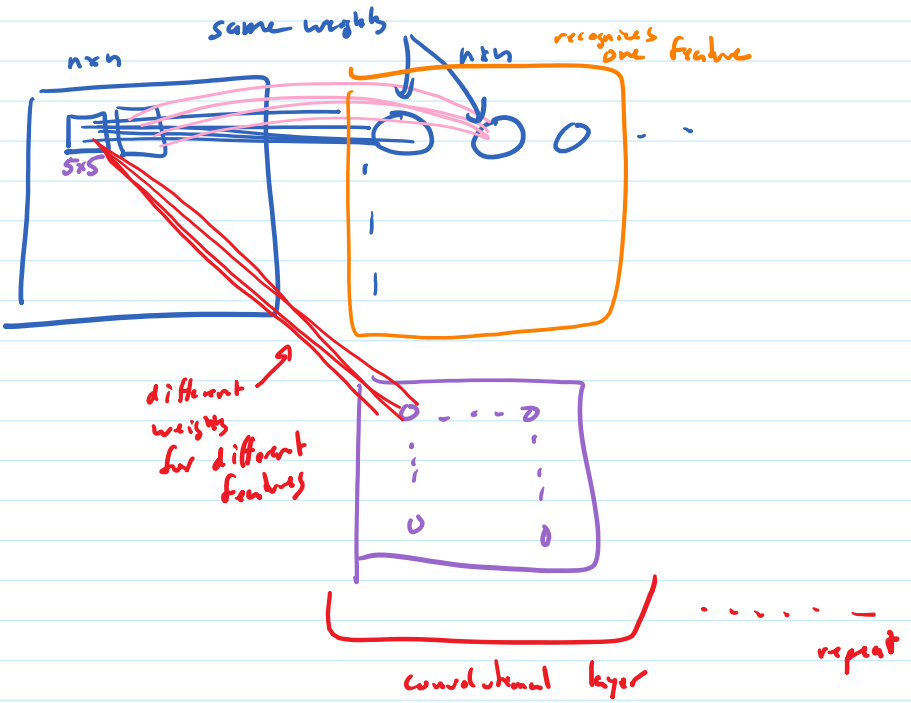
date month / day of month fuzzy encoding

	Jan						Dec
April 12	0	0	0.15	0.8	0.05	0 ...	0
Jan 1	0.52						0.48
Dec 31	0.48						0.52

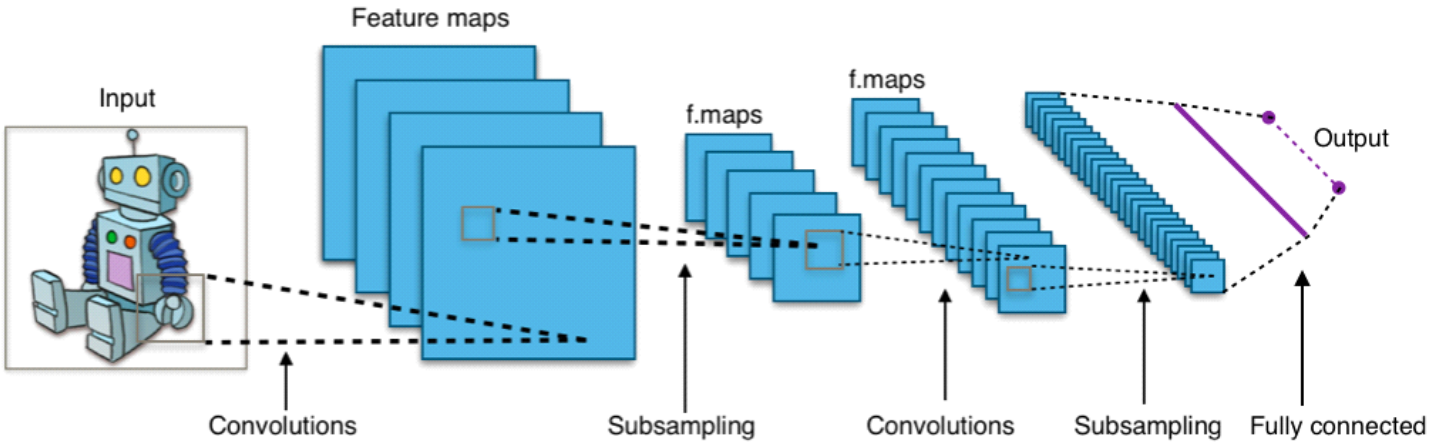
Convolutional Neural Networks — for images

Deep Q network learning to play Pong





A much better picture from Wikipedia user cpheve34, who does not endorse these notes.



https://upload.wikimedia.org/wikipedia/commons/6/63/Typical_cnn.png

AlphaGo

Step 1: supervised learning for convolutional deep neural network

- using data from expert players
- input: position
- output: expert's move
- 13 layers
- 19x19 x 48
- black/white/empty
- # opp captured
- # own captured
- liberties
- ladder capture
- ladder escape
- 3 weeks
- matched move 55% of time
- + smaller, faster network ~25% matches

Step 2: reinforcement learning for convolutional deep neural network

- 1 day
- beats SL network 80% of time

Step 3: reinforcement learning for value network

- using data from RL network playing itself 30M times (1 pos sampled per game)
- output is estimate of value:
  - +1 black wins
  - 0 draw
  - -1 white wins

Step 4: MCTS

- playout using fast network > weighted average
- estimate value using value network
- tree policy uses PUCT

$$Q(s,a) + c \cdot P(s,a) \cdot \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)}$$

- $Q(s,a)$: observed reward
- $\sum_b N(s,b)$: # times s visited
- $P(s,a)$: initialized using Step 1 (slow) SL network