Evaluating a Policy

Policy : function π : observations (current roll + scoresheet) → action (dice to keep or category to use)

observations = position s ; action = set of actions $A(s)$

$v(\pi)$ = expected reward (score) following policy π

can compute exactly for finite game (slow)

$P_\pi(s)$ = prob reach position s when following policy π

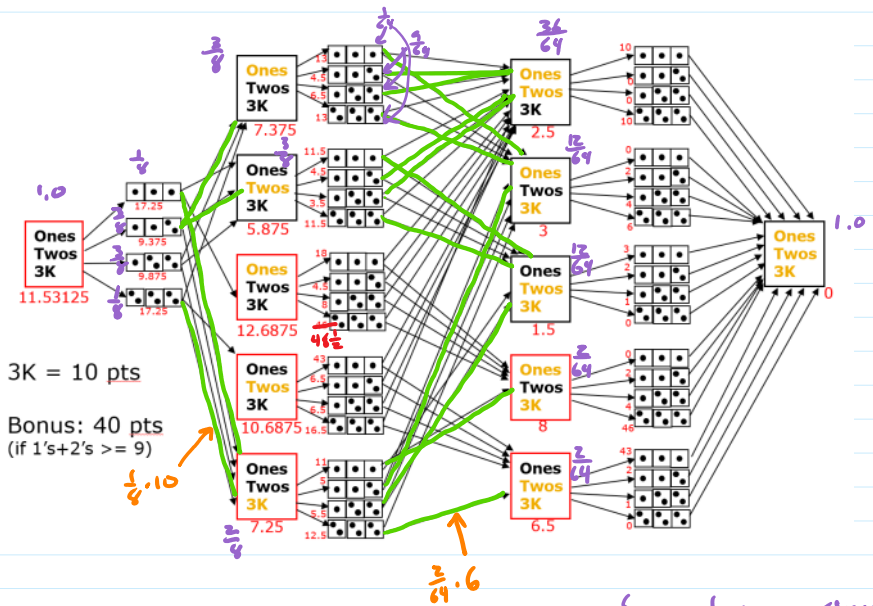
$$= \begin{cases} 1 & \text{if } s = s_0 \text{ (initial position)} \\ \sum_{q \text{ leads to } s} P_\pi(q) \cdot P_{\pi(q)}(q \to s) & \end{cases}$$

$P_\pi(q)$: prob π leads to q ; $P_{\pi(q)}(q \to s)$: prob action selected at q results in s

(special case of Markov decision process) - finite, no discount

$$v(\pi) = \sum_{\pi(q)=s} P_\pi(q) \cdot r_{\pi(q)}(q \to s)$$

reward earned by chosen action



find roll showing most keep them

{ ones, twos, ..., sixes, straight, yahtzee, full house, chance }

Yahtzee : $A(s) =$ { up to 32 combos of dice to keep } for reroll positions ; { c | category c is open } if s is end of turn

```
if rerolls == 2
    keep 5's, 6's
else
    keep 4's, 5's, 6's
```

else

keep 4's, 5's, 6's

normal computations for computing $\pi_{opt}(s)$: (for finite process)

$V_a(s)$ = value of action a at position s

$$= \sum_{P_a(s \to s') \neq 0} \underbrace{P_a(s \to s')}_{\text{prob action results in } s'} \cdot \left(\underbrace{R_a(s \to s')}_{\text{reward for action}} + \underbrace{V_{\pi_{opt}}(s')}_{\substack{\text{value at} \\ \text{resulting position}}} \right)$$

$$\pi_{opt}(s) = \underset{a}{\text{argmax}} \; V_a(s)$$

works even with a restricted set of actions to determine opt policy wrt those actions

$$V_{\pi_{opt}}(s) = V_{\pi(s)}(s)$$

Classifiers

Classifier : function — positions (attributes) → actions (class)

Iris flower data set

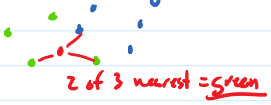
MNIST database http://yann.lecun.com/exdb/mnist/

Learning : supervised — examples available

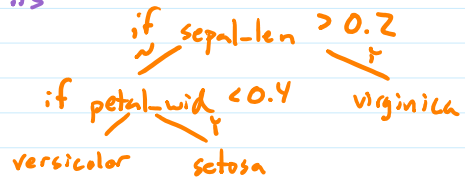
reinforcement — reward observable

unsupervised (clustering)

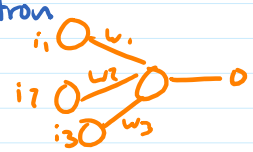
Methods : k-nearest neighbors — treat example inputs as pts in d-dim space, classify new input according to class of majority of the k nearest neighbors



decision trees — nested ifs



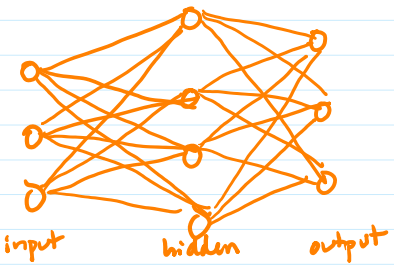
perceptron



$\text{output} = f\left(\sum_{i=1}^{k} i_k \cdot w_k\right)$

ex: threshold $f(y) = \begin{cases} 1 & \text{if } x > 5 \\ 0 & \text{otherwise} \end{cases}$

multi-layer perceptron (a type of artificial neural network)



Relu

introduces non-linearity

Deep Q network learning to play Pong