Approximate Q-Learning

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

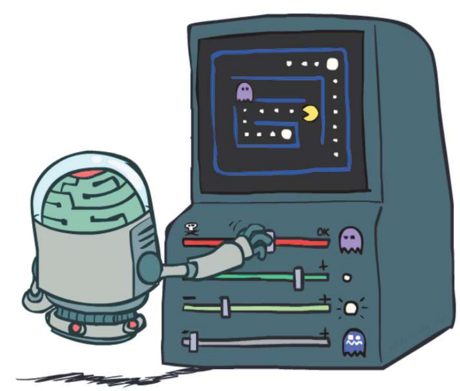
- Q-learning with linear Q-functions:

  transition $= (s, a, r, s')$

  $$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a')\right] - Q(s,a)$$

  $$Q(s,a) \leftarrow Q(s,a) + \alpha\,[\text{difference}]$$ Exact Q's

  $$w_i \leftarrow w_i + \alpha\,[\text{difference}]\, f_i(s,a)$$ Approximate Q's



- Intuitive interpretation:
  - Adjust weights of active features
  - E.g., if something unexpectedly bad happens, blame the features that were on: disprefer all states with that state's features
- Formal justification: online least squares

Q-learning with Linear Approximation

Algorithm 4: Q-learning with linear approximation.

Initialize q-value function Q with random weights w: $Q(s, a; w) = \sum_m w_m f_m(s, a)$;
for *episode* $= 1 \rightarrow M$ **do**
  Get initial state s_0;
  for $t = 1 \rightarrow T$ **do**
    With prob. ϵ, select a random action a_t;
    With prob. $1 - \epsilon$, select $a_t \in \text{argmax}_a \, Q(s_t, a; w)$;
    Execute selected action a_t and observe reward r_t and next state s_{t+1};
    Set target $y_t = \begin{cases} r_t & \text{if episode terminates at step } t+1 \\ r_t + \gamma \max_{a'} Q(s_{t+1}, a'; w) & \text{otherwise} \end{cases}$;
    Perform a gradient descent step to update w: $w_m \leftarrow w_m + \alpha \left[y_t - Q(s_t, a_t; w) \right] f_m(s, a)$;