

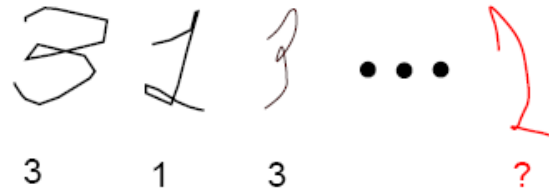
Inductive Learning (2/2)

Neural Nets

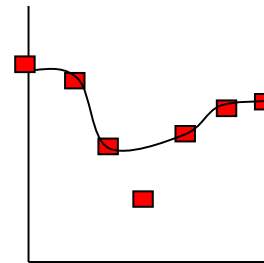
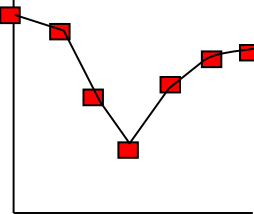
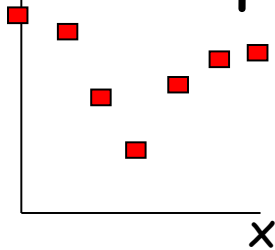
R&N: Chap. 20, Sec. 20.5

Function-Learning Formulation

- Goal function f
- Training set: $(\mathbf{x}^{(i)}, f(\mathbf{x}^{(i)}))$, $i = 1, \dots, n$



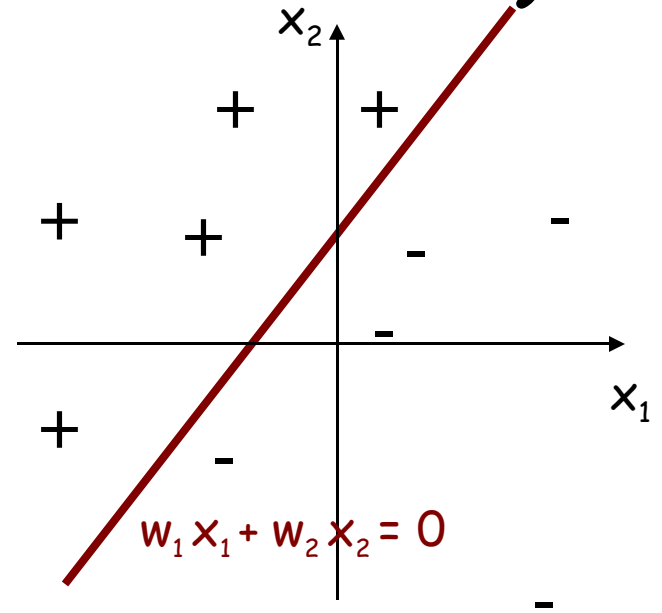
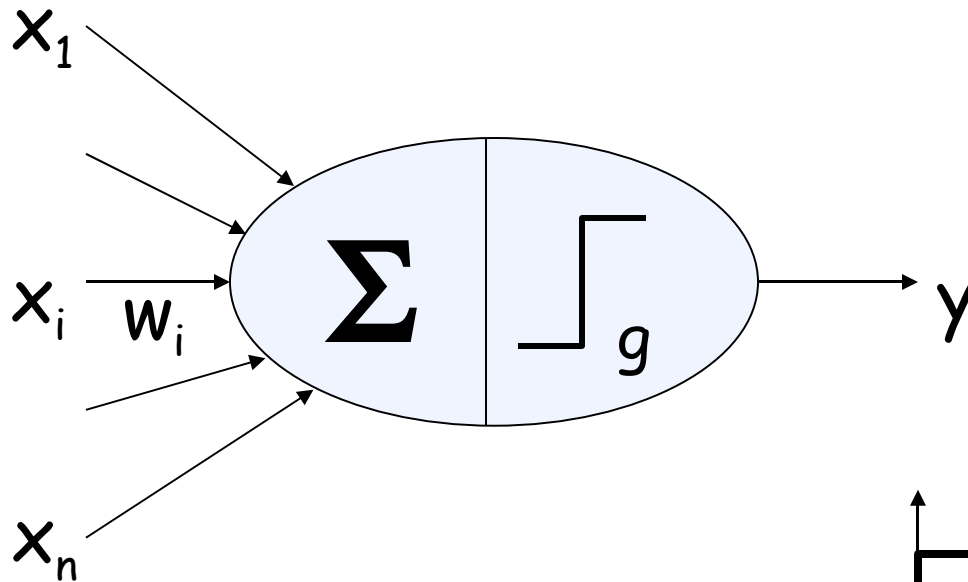
- Inductive inference: find a function h that fits the points well



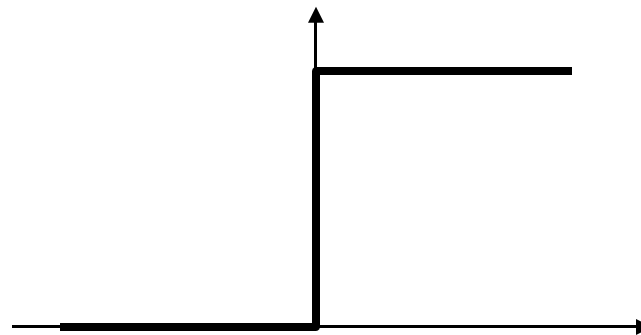
- Some Keep It Simple bias

Perceptron

(The goal function f is a boolean one)

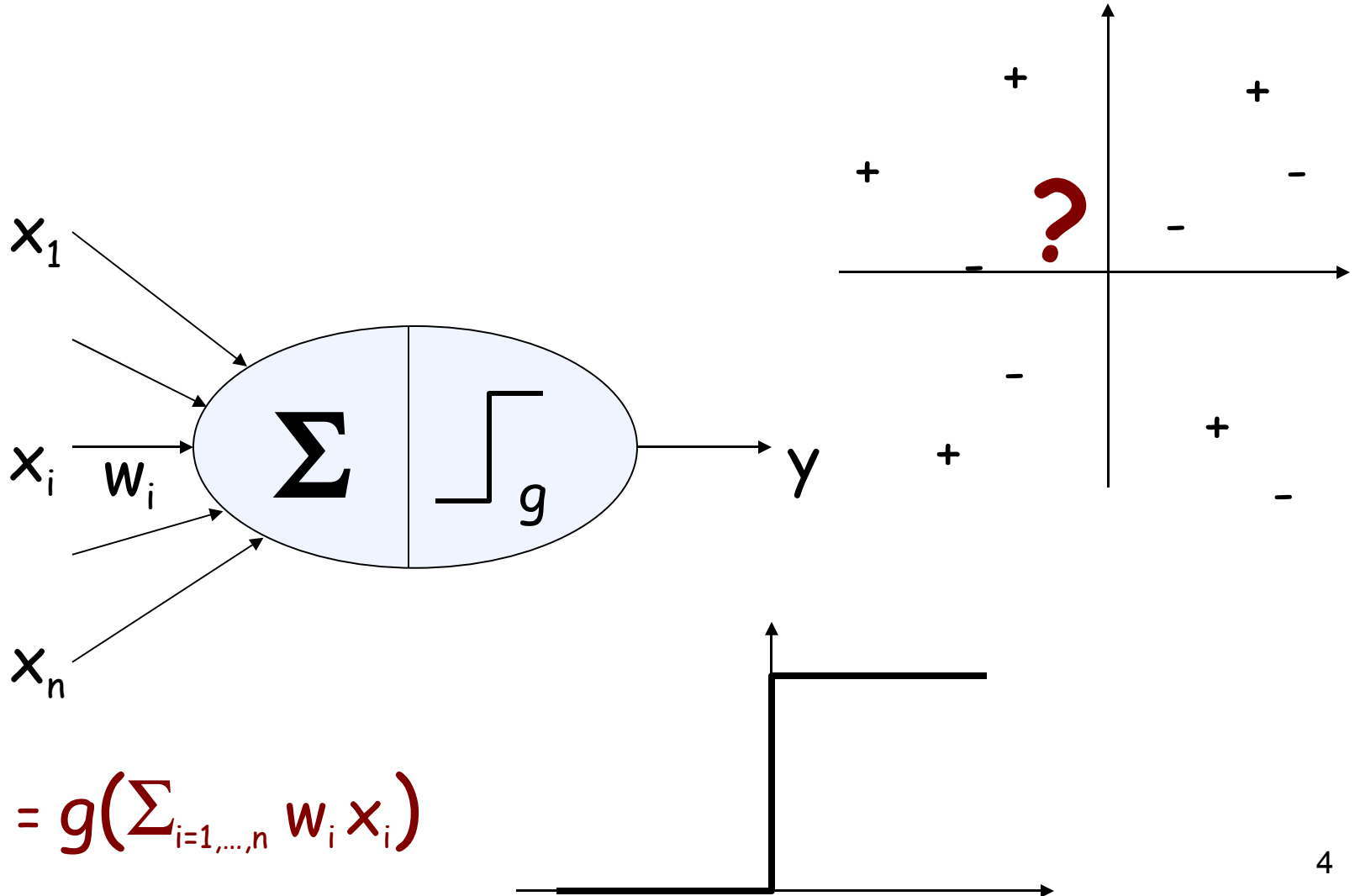


$$y = g\left(\sum_{i=1, \dots, n} w_i x_i\right)$$



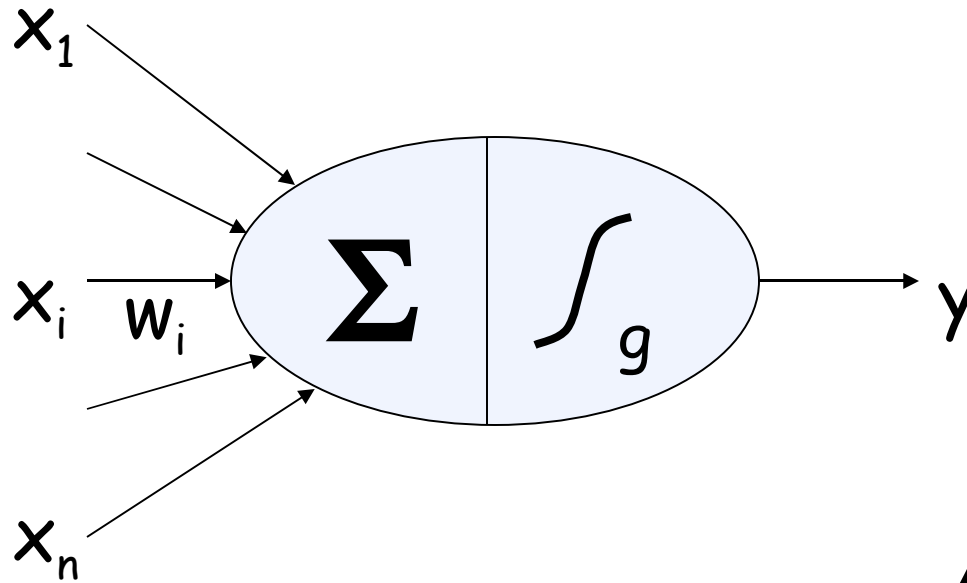
Perceptron

(The goal function f is a boolean one)

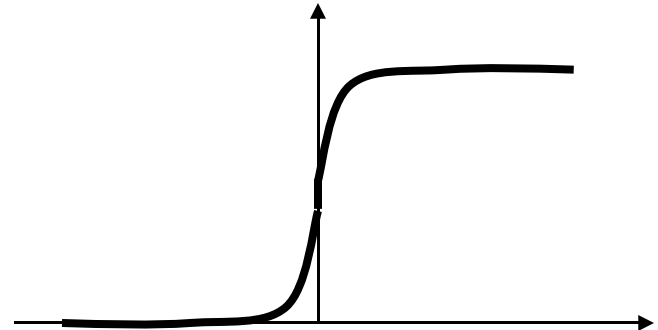


$$y = g\left(\sum_{i=1, \dots, n} w_i x_i\right)$$

Unit (Neuron)

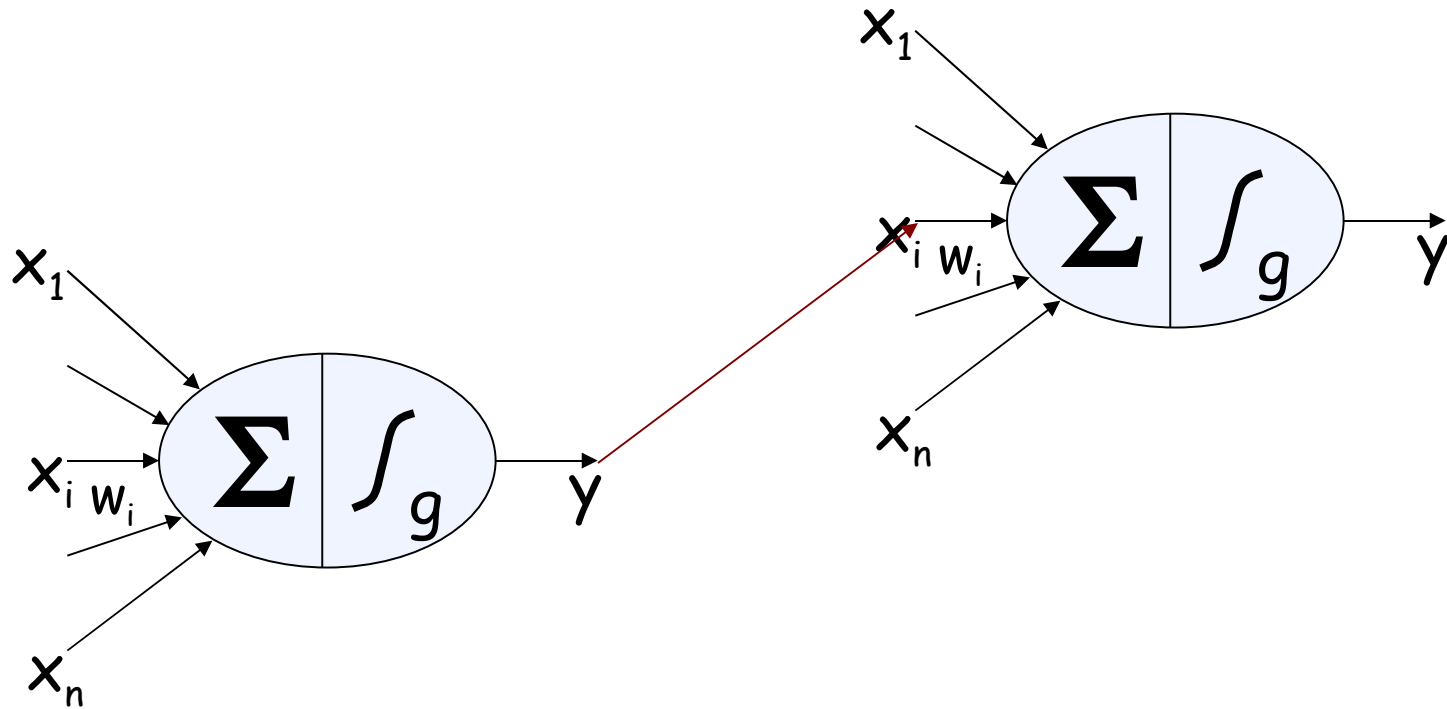


$$y = g\left(\sum_{i=1, \dots, n} w_i x_i\right)$$
$$g(u) = 1/[1 + \exp(-\alpha \times u)]$$



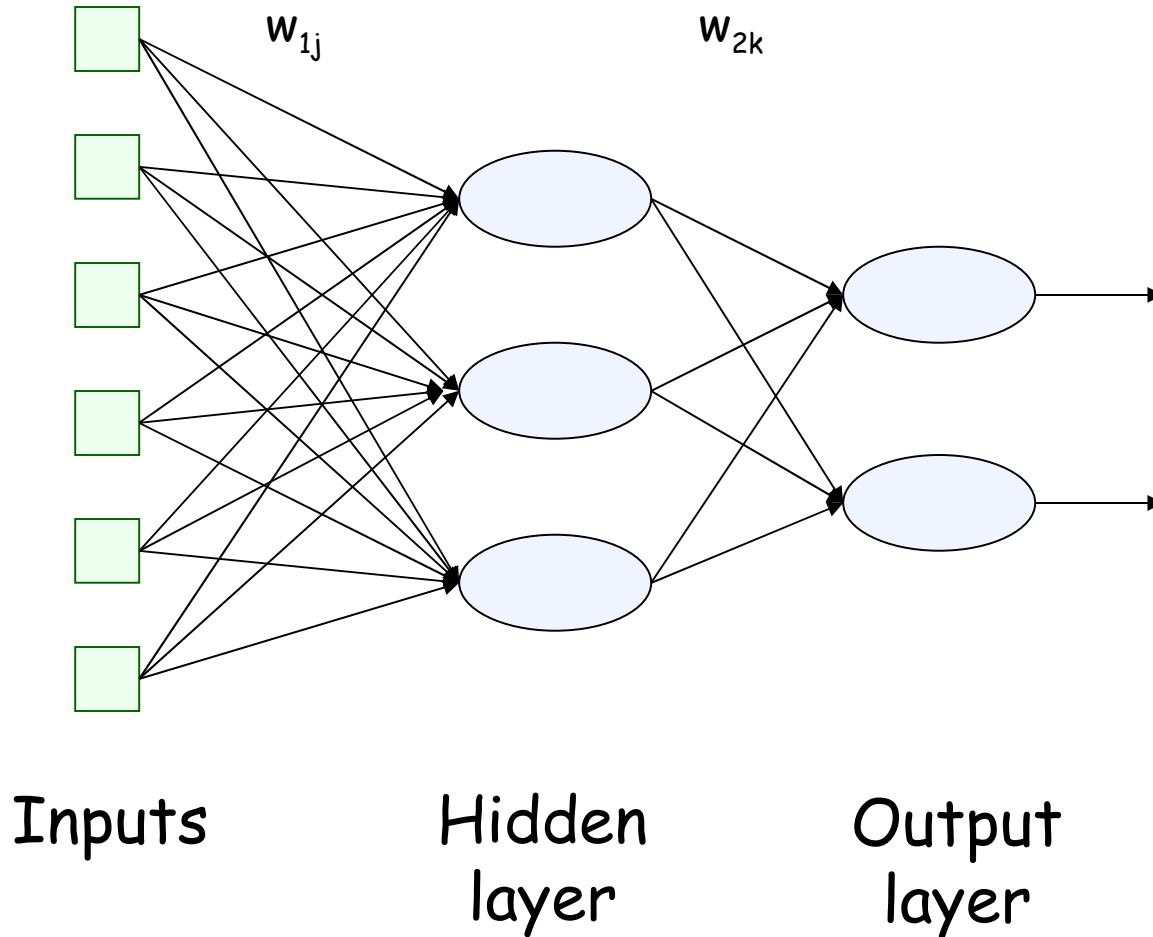
Neural Network

Network of interconnected neurons



Acyclic (feed-forward) vs. recurrent networks

Two-Layer Feed-Forward Neural Network



Backpropagation (Principle)

- New example $y^{(k)} = f(x^{(k)})$
- $\varphi^{(k)}$ = outcome of NN with weights $w^{(k-1)}$ for inputs $x^{(k)}$
- Error function: $E^{(k)}(w^{(k-1)}) = ||\varphi^{(k)} - y^{(k)}||^2$
- $w_{ij}^{(k)} = w_{ij}^{(k-1)} - \varepsilon \times \partial E^{(k)} / \partial w_{ij}$ ($w^{(k)} = w^{(k-1)} - \varepsilon \times \nabla E$)
- **Backpropagation algorithm:**
Update the weights of the inputs to the last layer, then the weights of the inputs to the previous layer, etc.

Comments and Issues

- How to choose the size and structure of networks?
 - If network is too large, risk of over-fitting (data caching)
 - If network is too small, representation may not be rich enough
- Role of representation: e.g., learn the concept of an odd number
- Incremental learning

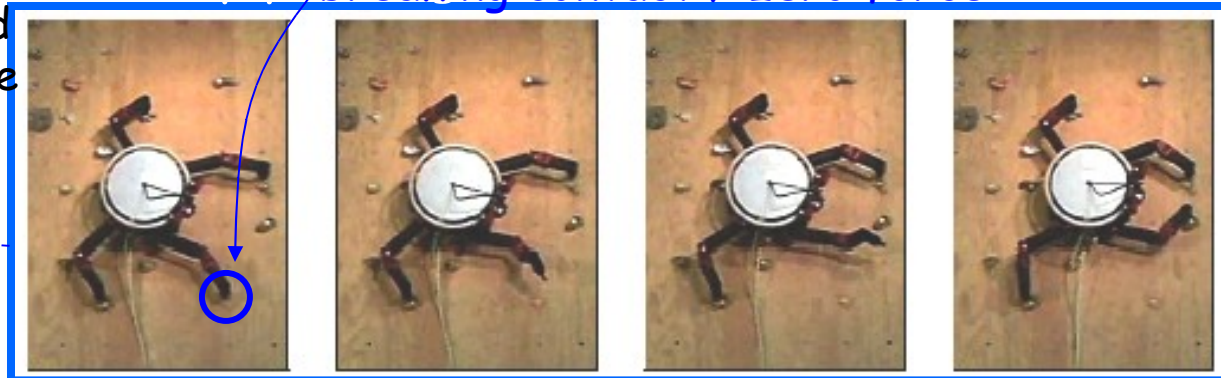
Application of NN to Motion Planning (Climbing Robot)

Transition

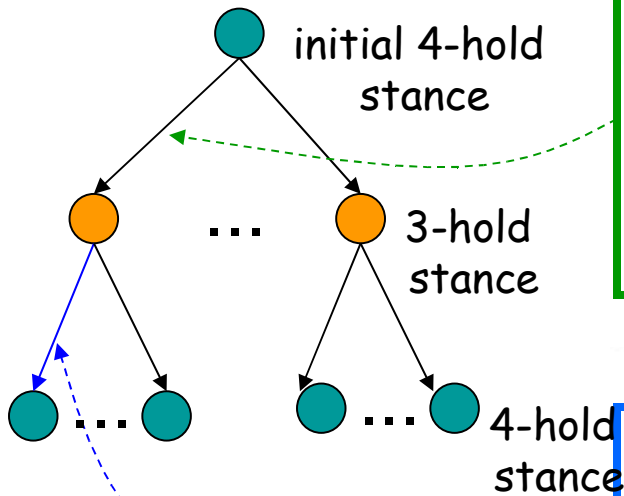
one-step planning



A one-step motion with a 4-hold stance, to remove the bottom right hand.

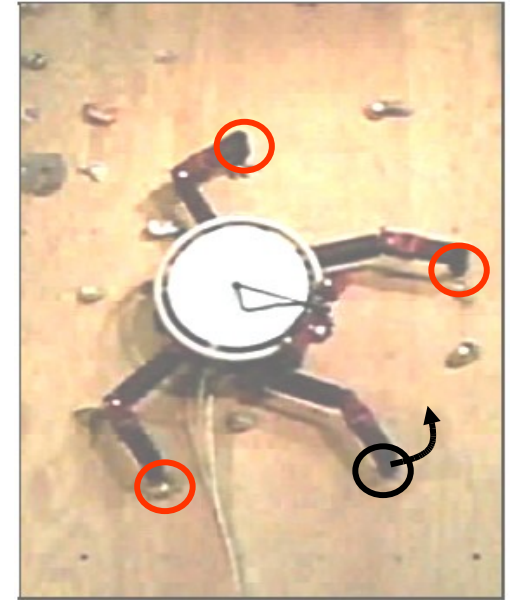


A one-step motion with a 3-hold stance, to place the bottom right hand.



Idea: Learn Feasibility

- Create a large database of labeled transitions



- Train a NN classifier

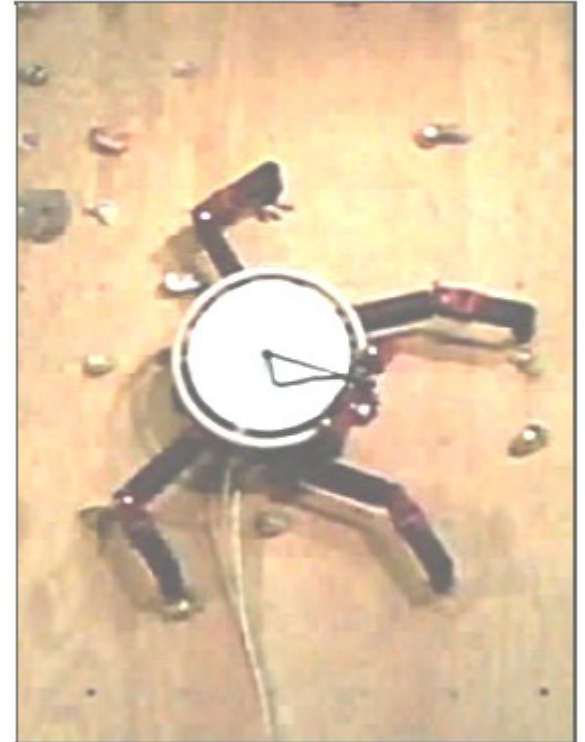
⊕ : transition \rightarrow {feasible, not feasible}

- Learning is possible because:

Shape of a feasible space is mostly determined by the equilibrium condition that depends on relatively few parameters

Creation of Database

- Sample transitions at random (by picking 4 holds at random within robot's limb span)
- Label each transition - feasible or infeasible - by sampling with high time limit
 - over 95% infeasible transitions
- **Re-sample around feasible transitions**
 - **35-65% feasible transitions**
- ~1 day of computation to create a database of 100,000 labeled transitions



Training of a NN Classifier

- NN with 9 input units, 100 hidden units, and 1 output unit
- Training on 50,000 examples (~3 days of computation)
- Validation on the remaining 50,000 examples
 - ~78% accuracy ($\epsilon = 0.22$)
 - 0.003ms average running time

Transition

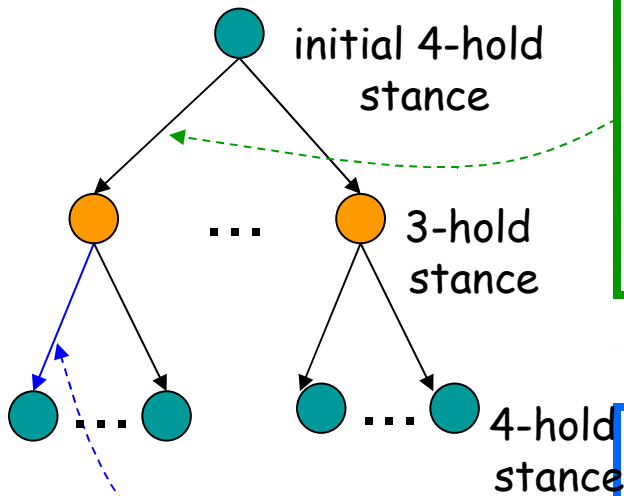
one-step planning



A one-step motion with a 4-hold stance, to remove the bottom right hand.



A one-step motion with a 3-hold stance, to place the bottom right hand.



Summary

- Machine learning is very useful in many robot motion planning scenarios
- Neural Networks is one type of learning algorithm
- Good luck with HW 6 and the programming
- See you next week!