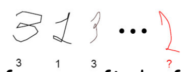
Inductive Learning (2/2)
Neural Nets

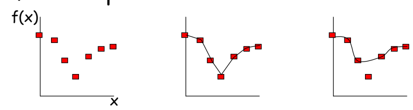
R&N: Chap. 20, Sec. 20.5

Function-Learning Formulation

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

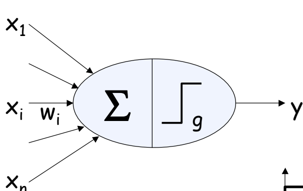


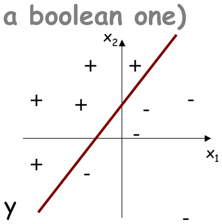
- Inductive inference: find a function h that fits the points well



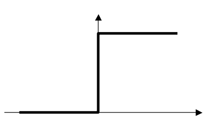
- Same Keep-It-Simple bias

Perceptron
(The goal function f is a boolean one)

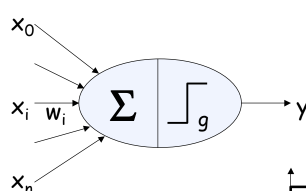




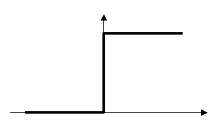
$y = g(\Sigma_{i=1,\dots,n}\, w_i x_i)$



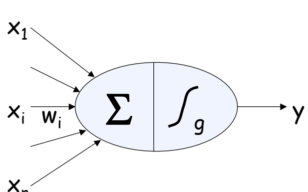
Perceptron
(The goal function f is a boolean one)



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

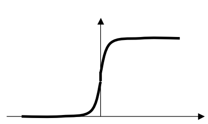


Unit (Neuron)



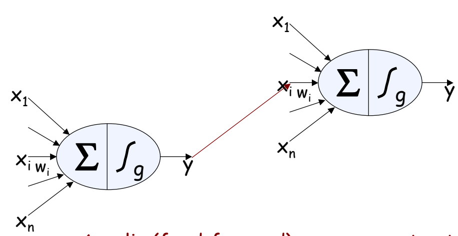
$y = g(\Sigma_{i=1,\dots,n}\, w_i x_i)$

$g(u) = 1/[1 + \exp(-\alpha\times u)]$



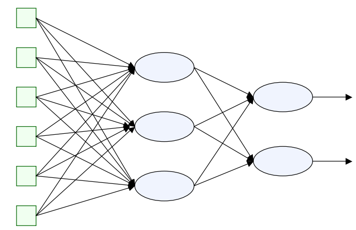
Neural Network

Network of interconnected neurons



Acyclic (feed-forward) vs. recurrent networks

Two-Layer Feed-Forward Neural Network



Backpropagation (Principle)

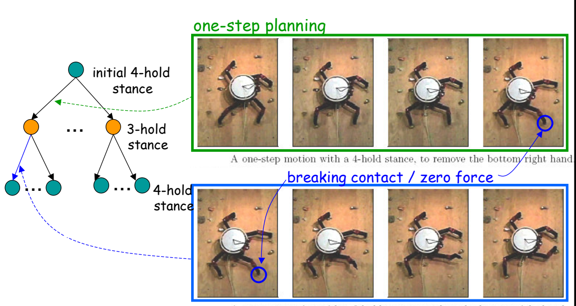
- New example $y^{(k)} = f(x^{(k)})$
- $\varphi^{(k)}$ = outcome of NN with weights $w^{(k-1)}$ for inputs $x^{(k)}$
- Error function: $E^{(k)}(\mathbf{w}^{(k-1)}) = ||\varphi^{(k)} - y^{(k)}||^2$
- $w_{ij}^{(k)} = w_{ij}^{(k-1)} - \varepsilon \times \partial E/\partial w_{ij}$
- Backpropagation: Update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

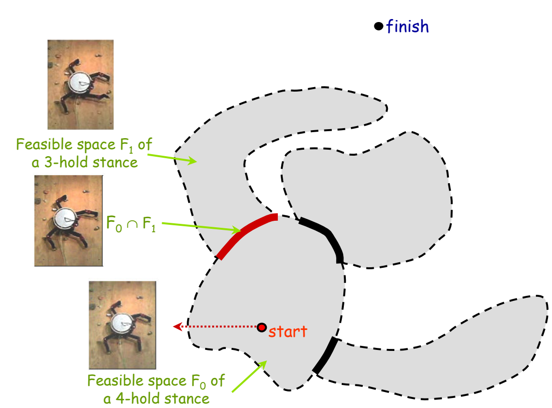
Comments and Issues

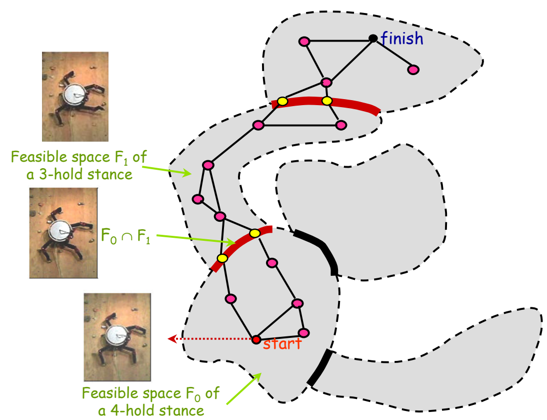
- How to choose the size and structure of networks?
  - If network is too large, risk of over-fitting (data caching)
  - If network is too small, representation may not be rich enough
- Role of representation: e.g., learn the concept of an odd number
- Incremental learning

Application of NN to Motion Planning (Climbing Robot)

Multi-Step Planning

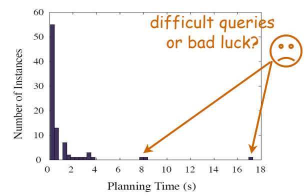






Dilemma

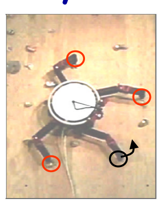
- More than 1,000,000 stances, but only 20,000 feasible ones
- Several 1000 path queries
- A large fraction of them have no solution, but **sampling can't detect it**
- The running times for (feasible) queries making up an 88-step path are highly variable



How much time for each query?

Idea: Learn Feasibility

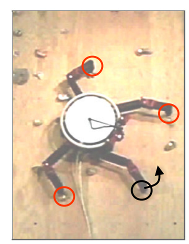
- Create a large database of labeled transitions
- Train a NN classifier
  Θ : transition → {feasible, not feasible)
- Learning is possible because:
  Shape of a feasible space is mostly determined by the equilibrium condition that depends on relatively few parameters



Parameterization of a Transition

- A transition is always between a 3-hold and a 4-hold stance
- Defining parameters:
  - Relative positions of 4 holds
  - Orientation of 3 holds
  - (Friction coefficient at 3 holds)

→ 9 parameters



Creation of Database

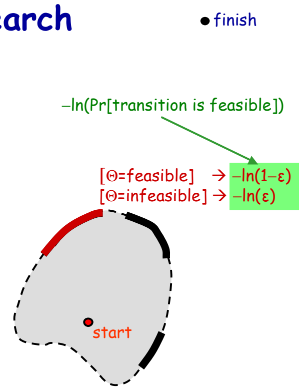
- Sample transitions at random (by picking 4 holds at random within robot's limb span)
- Label each transition – feasible or infeasible – by sampling with high time limit
  → over 95% infeasible transitions
- Re-sample around feasible transitions
  → 35-65% feasible transitions
- ~1 day of computation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

Training of a NN Classifier

- 3-layer 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 (ε = 0.22)
  → 0.003ms average running time

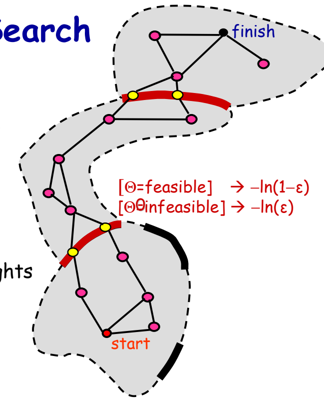
Fuzzy Search

- Use Θ in initial phase of search to attach weights to transitions
- Never fully trust the classifier
- Perform uniform-cost search



Fuzzy Search

- Use Θ in initial phase of search to attach weights to transitions
- Never fully trust the classifier
- Perform uniform-cost search
- Sampling transitions leads to updating weights



Experimental Results

Terrain with 34 holds (> million stances)

Time (sec.)	Total	Classification of transitions	Sampling of transitions	One-step planning
Basic	210	--	179	24
Fuzzy	31	10	7	13

#	Transition sampling failures	One-step motion planning failures	Steps in path
Basic	4948	0.8	20.6
Fuzzy	62	2.2	29.2

What Have We Learned?

- Useful methods
- Connection between fields, e.g., control theory, game theory, operational research
- Impact of hardware (chess software → brute-force reasoning, case-base reasoning)
- Relation between high-level (e.g., logic) and low-level (e.g., neural nets) representations: from pixels to predicates
- Beyond learning: What concepts to learn?
- What is intelligence? Impact of other aspects of human nature: fear of dying, appreciation for beauty, self-consciousness, ...
- Should AI be limited to information-processing tasks?
- Our methods are better than our understanding

Some Important Achievements

- Logic reasoning (data bases)
- Search and game playing
- Knowledge-based systems
- Bayesian networks (diagnosis)
- Machine learning
- Data mining
- Military logistics
- Autonomous robots

What is AI?

Discipline that systematizes and automates intellectual tasks to create machines that:

Act like humans	Act rationally
Think like humans	Think rationally

Some Other AI Classes

- Intros to AI: CS121 and CS221
- CS 222: Knowledge Representation
- CS 223A: Intro to Robotics
- CS 223B: Intro to Computer Vision
- CS 224M: Multi-Agent Systems
- CS 224N: Natural Language Processing
- CS 225A: Experimental Robotics
- CS 227: Reasoning Methods in AI
- CS 228: Probabilistic Models in AI
- CS 229: Machine Learning
- CS 257: Automated Deduction and Its Applications
- CS 323: Common Sense Reasoning in Logic
- CS 324: Computer Science and Game Theory
- CS 326A: Motion Planning
- CS 327A: Advanced Robotics
- CS 328: Topics in Computer Vision
- CS 329: Topics in AI