CS 188: Artificial Intelligence

Reinforcement Learning (RL)

Pieter Abbeel – UC Berkeley

Many slides over the course adapted from Dan Klein, Stuart Russell, Andrew Moore

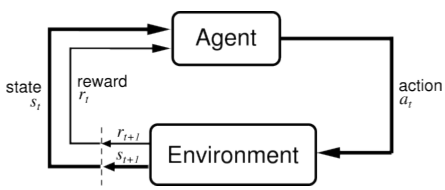
MDPs and RL Outline

- Markov Decision Processes (MDPs)
  - ✓ Formalism
  - ✓ Planning
    - Value iteration
    - Policy Evaluation and Policy Iteration
- Reinforcement Learning --- MDP with T and/or R unknown
  - Model-based Learning
  - Model-free Learning
    - Direct Evaluation [performs policy evaluation]
    - Temporal Difference Learning [performs policy evaluation]
    - Q-Learning [learns optimal state-action value function Q*]
    - Policy search [learns optimal policy from subset of all policies]
  - Exploration vs. exploitation
- Large state spaces: feature-based representations

Reinforcement Learning

- Basic idea:
  - Receive feedback in the form of rewards
  - Agent's utility is defined by the reward function
  - Must (learn to) act so as to maximize expected rewards

[DEMOS on next slides]



Example: learning to walk

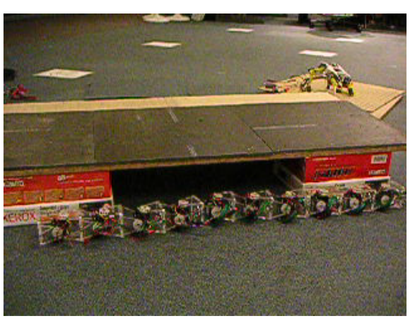

Before learning (hand-tuned)


One of many learning runs


After learning [After 1000 field traversals]

[Kohl and Stone, ICRA 2004]

Example: snake robot

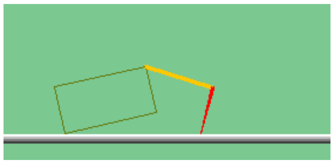


[Andrew Ng]

Example: Toddler

Tedrake, Zhang and Seung, 2005

In your project 3



Reinforcement Learning

- Still assume a Markov decision process (MDP):
  - A set of states s ∈ S
  - A set of actions (per state) A
  - A model T(s,a,s')
  - A reward function R(s,a,s')
- Still looking for a policy π(s)
- New twist: don't know T or R
  - I.e. don't know which states are good or what the actions do
  - Must actually try actions and states out to learn

MDPs and RL Outline

- Markov Decision Processes (MDPs)
  - ✓ Formalism
  - ✓ Planning
    - Value iteration
    - Policy Evaluation and Policy Iteration
- ➡ Reinforcement Learning --- MDP with T and/or R unknown
  - Model-based Learning
  - Model-free Learning
    - Direct Evaluation [performs policy evaluation]
    - Temporal Difference Learning [performs policy evaluation]
    - Q-Learning [learns optimal state-action value function Q*]
    - Policy search [learns optimal policy from subset of all policies]
  - Exploration vs. exploitation
- Large state spaces: feature-based representations

Example to Illustrate Model-Based vs. Model-Free: Expected Age

Goal: Compute expected age of cs188 students

Known P(A)

$$E[A] = \sum_a P(a) \cdot a \quad = 0.35 \times 20 + \ldots$$

Without P(A), instead collect samples [a_1, a_2, … a_N]

Unknown P(A): "Model Based"

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$

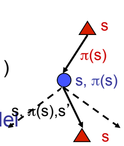
$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

Unknown P(A): "Model Free"

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Model-Based Learning

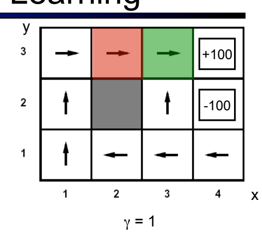
- Idea:
  - Step 1: Learn the model empirically through experience
  - Step 2: Solve for values as if the learned model were correct
- Step 1: Simple empirical model learning
  - Count outcomes for each s,a
  - Normalize to give estimate of **T(s,a,s')**
  - Discover **R(s,a,s')** when we experience (s,a,s')
- Step 2: Solving the MDP with the learned model
  - Value iteration, or policy iteration



Example: Learning the Model in Model-Based Learning

- Episodes:

(1,1) up -1	(1,1) up -1
(1,2) up -1	(1,2) up -1
(1,2) up -1	(1,3) right -1
(1,3) right -1	(2,3) right -1
(2,3) right -1	(3,3) right -1
(3,3) right -1	(3,2) up -1
(3,2) up -1	(4,2) exit -100
(3,3) right -1	(done)
(4,3) exit +100	
(done)	



T(<3,3>, right, <4,3>) = 1 / 3

T(<2,3>, right, <3,3>) = 2 / 2

Learning the Model in Model-Based Learning

- Estimate P(x) from samples
  - Samples: $x_i \sim P(x)$
  - Estimate: $\hat{P}(x) = \text{count}(x)/k$
- Estimate P(s' | s, a) from samples
  - Samples: $s_0, a_0, s_1, a_1, s_2, \ldots$
  - Estimate: $\hat{P}(s'|s,a) = \frac{\text{count}(s_{t+1}=s', a_t=a, s_t=s)}{\text{count}(s_t=s, a_t=a)}$
- Why does this work? Because samples appear with the right frequencies!

13

Model-based vs. Model-free

- Model-based RL
  - First act in MDP and learn T, R
  - Then value iteration or policy iteration with learned T, R
  - Advantage: efficient use of data
  - Disadvantage: requires building a model for T, R
- Model-free RL
  - Bypass the need to learn T, R
  - Methods to evaluate V^π, the value function for a fixed policy π without knowing T, R:
    - (i) Direct Evaluation
    - (ii) Temporal Difference Learning
  - Method to learn π^*, Q*, V* without knowing T, R
    - (iii) Q-Learning

14

Direct Evaluation

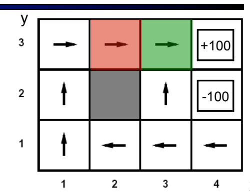
- Repeatedly execute the policy π
- Estimate the value of the state s as the average over all times the state s was visited of the sum of discounted rewards accumulated from state s onwards

15

Example: Direct Evaluation

- Episodes:

(1,1) up -1	(1,1) up -1
(1,2) up -1	(1,2) up -1
(1,2) up -1	(1,3) right -1
(1,3) right -1	(2,3) right -1
(2,3) right -1	(3,3) right -1
(3,3) right -1	(3,2) up -1
(3,2) up -1	(4,2) exit -100
(3,3) right -1	(done)
(4,3) exit +100	
(done)	



γ = 1, R = -1

V(2,3) ~ (96 + -103) / 2 = -3.5

V(3,3) ~ (99 + 97 + -102) / 3 = 31.3

16

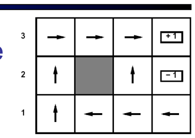
Model-Free Learning

- Want to compute an expectation weighted by P(x):
  $$E[f(x)] = \sum_x P(x) f(x)$$
- Model-based: estimate P(x) from samples, compute expectation
  $$x_i \sim P(x)$$
  $$\hat{P}(x) = \text{num}(x)/N$$
  $$E[f(x)] \approx \sum_x \hat{P}(x) f(x)$$
- Model-free: estimate expectation directly from samples
  $$x_i \sim P(x) \qquad E[f(x)] \approx \frac{1}{N} \sum_i f(x_i)$$
- Why does this work? Because samples appear with the right frequencies!

17

Limitations of Direct Evaluation

- Assume random initial state
- Assume the value of state (1,2) is known perfectly based on past runs
- Now for the first time encounter (1,1) --- can we do better than estimating V(1,1) as the rewards outcome of that run?

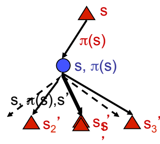


18

Sample-Based Policy Evaluation?

$$V_{i+1}^{\pi}(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_i^{\pi}(s')]$$

- Who needs T and R? Approximate the expectation with samples of s' (drawn from T!)



$$sample_1 = R(s,\pi(s),s_1') + \gamma V_i^{\pi}(s_1')$$
$$sample_2 = R(s,\pi(s),s_2') + \gamma V_i^{\pi}(s_2')$$
$$\dots$$
$$sample_k = R(s,\pi(s),s_k') + \gamma V_i^{\pi}(s_k')$$

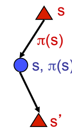
$$V_{i+1}^{\pi}(s) \leftarrow \frac{1}{k}\sum_i sample_i$$

Almost! But we can't rewind time to get sample after sample from state s.

20

Temporal-Difference Learning

- Big idea: learn from every experience!
  - Update V(s) each time we experience (s,a,s',r)
  - Likely s' will contribute updates more often
- Temporal difference learning
  - Policy still fixed!
  - Move values toward value of whatever successor occurs: running average!



Sample of V(s): $sample = R(s,\pi(s),s') + \gamma V^{\pi}(s')$

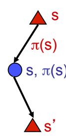
Update to V(s): $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

21

Temporal-Difference Learning

- Big idea: learn from every experience!
  - Update V(s) each time we experience (s,a,s',r)
  - Likely s' will contribute updates more often
- Temporal difference learning
  - Policy still fixed!
  - Move values toward value of whatever successor occurs: running average!



Sample of V(s): $sample = R(s,\pi(s),s') + \gamma V^{\pi}(s')$

Update to V(s): $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

22

Exponential Moving Average

- Exponential moving average
  - Makes recent samples more important

$$\bar{x}_n = \frac{x_n + (1-\alpha)\cdot x_{n-1} + (1-\alpha)^2 \cdot x_{n-2} + \dots}{1 + (1-\alpha) + (1-\alpha)^2 + \dots}$$

  - Forgets about the past (distant past values were wrong anyway)
  - Easy to compute from the running average

$$\bar{x}_n = (1-\alpha)\cdot \bar{x}_{n-1} + \alpha \cdot x_n$$

- Decreasing learning rate can give converging averages

23

Policy Evaluation when T (and R) unknown --- recap

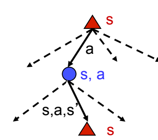
- Model-based:
  - Learn the model empirically through experience
  - Solve for values as if the learned model were correct
- Model-free:
  - Direct evaluation:
    - V(s) = sample estimate of sum of rewards accumulated from state s onwards
  - Temporal difference (TD) value learning:
    - Move values toward value of whatever successor occurs: running average!

$$sample = R(s,\pi(s),s') + \gamma V^{\pi}(s')$$
$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$$

25

Problems with TD Value Learning

- TD value leaning is a model-free way to do policy evaluation
- However, if we want to turn values into a (new) policy, we're sunk:



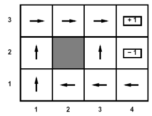
$$\pi(s) = \arg\max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V^*(s')\right]$$

- Idea: learn Q-values directly
- Makes action selection model-free too!

26

Active RL



- Full reinforcement learning
  - You don't know the transitions T(s,a,s')
  - You don't know the rewards R(s,a,s')
  - You can choose any actions you like
  - Goal: learn the optimal policy / values
  - … what value iteration did!
- In this case:
  - Learner makes choices!
  - Fundamental tradeoff: exploration vs. exploitation
  - This is NOT offline planning! You actually take actions in the world and find out what happens…

27

Detour: Q-Value Iteration

- Value iteration: find successive approx optimal values
  - Start with $V_0^*(s) = 0$, which we know is right (why?)
  - Given V_i^*, calculate the values for all states for depth i+1:

$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

- But Q-values are more useful!
  - Start with $Q_0^*(s,a) = 0$, which we know is right (why?)
  - Given Q_i^*, calculate the q-values for all q-states for depth i+1:

$$Q_{i+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_i(s',a') \right]$$

28

Q-Learning

- Q-Learning: sample-based Q-value iteration
- Learn Q*(s,a) values
  - Receive a sample (s,a,s',r)
  - Consider your old estimate: $Q(s,a)$
  - Consider your new sample estimate:

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \right]$$

$$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

  - Incorporate the new estimate into a running average:

$$Q(s,a) \leftarrow (1-\alpha) Q(s,a) + (\alpha) \left[sample \right]$$

29

Q-Learning Properties

- Amazing result: Q-learning converges to optimal policy
  - If you explore enough
  - If you make the learning rate small enough
  - … but not decrease it too quickly!
  - Basically doesn't matter how you select actions (!)
- Neat property: off-policy learning
  - learn optimal policy without following it

31

Exploration / Exploitation

- Several schemes for forcing exploration
  - Simplest: random actions (ε greedy)
    - Every time step, flip a coin
    - With probability ε, act randomly
    - With probability 1-ε, act according to current policy
  - Problems with random actions?
    - You do explore the space, but keep thrashing around once learning is done
    - One solution: lower ε over time
    - Another solution: exploration functions

33

Exploration Functions

- When to explore
  - Random actions: explore a fixed amount
  - Better idea: explore areas whose badness is not (yet) established
- Exploration function
  - Takes a value estimate and a count, and returns an optimistic utility, e.g. $f(u,n) = u + k/n$ (exact form not important)

$$Q_{i+1}(s,a) \leftarrow (1-\alpha) Q_i(s,a) + \alpha \left(R(s,a,s') + \gamma \max_{a'} Q_i(s',a') \right)$$

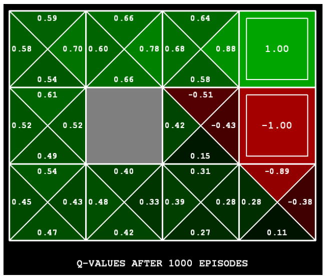
now becomes:

$$Q_{i+1}(s,a) \leftarrow (1-\alpha) Q_i(s,a) + \alpha \left(R(s,a,s') + \gamma \max_{a'} f(Q_i(s',a'), N(s',a')) \right)$$

35

Q-Learning

- Q-learning produces tables of q-values:



The Story So Far: MDPs and RL

Things we know how to do:

- If we know the MDP
  - Compute V*, Q*, π* exactly
  - Evaluate a fixed policy π
- If we don't know the MDP
  - We can estimate the MDP then solve
  - We can estimate V for a fixed policy π
  - We can estimate Q*(s,a) for the optimal policy while executing an exploration policy

Techniques:

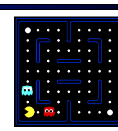
- Model-based DPs
  - Value Iteration
  - Policy evaluation
- Model-based RL
- Model-free RL
  - Value learning
  - Q-learning

Q-Learning

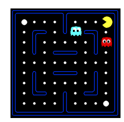
- In realistic situations, we cannot possibly learn about every single state!
  - Too many states to visit them all in training
  - Too many states to hold the q-tables in memory
- Instead, we want to generalize:
  - Learn about some small number of training states from experience
  - Generalize that experience to new, similar states
  - This is a fundamental idea in machine learning, and we'll see it over and over again

Example: Pacman

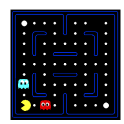
- Let's say we discover through experience that this state is bad:



- In naïve q learning, we know nothing about this state or its q states:

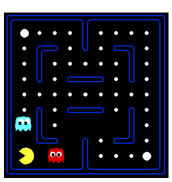


- Or even this one!



Feature-Based Representations

- Solution: describe a state using a vector of features
  - Features are functions from states to real numbers (often 0/1) that capture important properties of the state
  - Example features:
    - Distance to closest ghost
    - Distance to closest dot
    - Number of ghosts
    - 1 / (dist to dot)^2
    - Is Pacman in a tunnel? (0/1)
    - …… etc.
  - Can also describe a q-state (s, a) with features (e.g. action moves closer to food)



Linear Feature Functions

- Using a feature representation, we can write a q function (or value function) for any state using a few weights:

$$V(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

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

- Advantage: our experience is summed up in a few powerful numbers
- Disadvantage: states may share features but be very different in value!

Function Approximation

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

- Q-learning with linear q-functions:

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

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

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

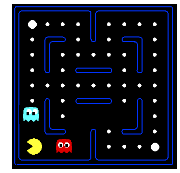
  $w_i \leftarrow w_i + \alpha$ [difference] $f_i(s,a)$ Approximate Q' s

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

43

Example: Q-Pacman

$Q(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$



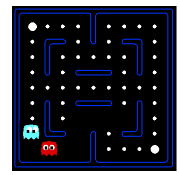
$f_{DOT}(s, \text{NORTH}) = 0.5$

$f_{GST}(s, \text{NORTH}) = 1.0$

$Q(s,a) = +1$

$R(s,a,s') = -500$

$error = -501$



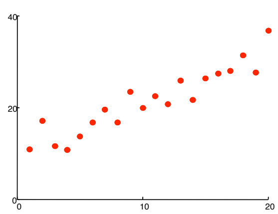
$w_{DOT} \leftarrow 4.0 + \alpha\,[-501]\,0.5$

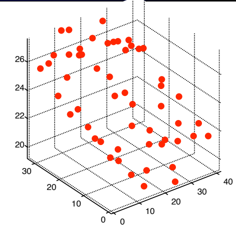
$w_{GST} \leftarrow -1.0 + \alpha\,[-501]\,1.0$

$Q(s,a) = 3.0 f_{DOT}(s,a) - 3.0 f_{GST}(s,a)$

44

Linear regression



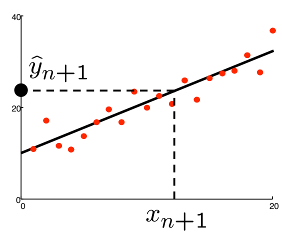


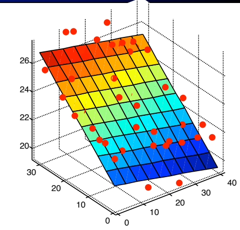
Given examples $(x_i, y_i)_{i=1\ldots n}$

Predict y_{n+1} given a new point x_{n+1}

45

Linear regression



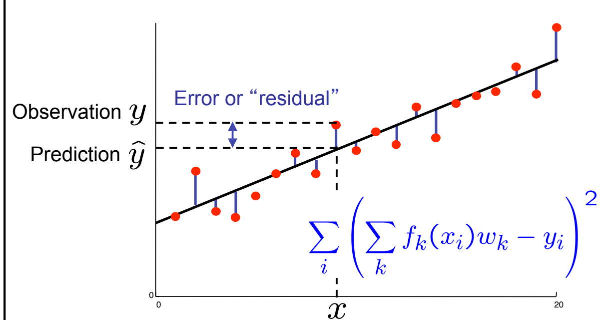


Prediction
$\hat{y}_i = w_0 + w_1 x_i$

Prediction
$\hat{y}_i = w_0 + w_1 x_{i,1} + w_2 x_{i,2}$

46

Ordinary Least Squares (OLS)



47

Minimizing Error

$$E(w) = \frac{1}{2}\sum_i \left(\sum_k f_k(x_i) w_k - y_i\right)^2$$

$$\frac{\partial E}{\partial w_m} = \sum_i \left(\sum_k f_k(x_i) w_k - y_i\right) f_m(x_i)$$

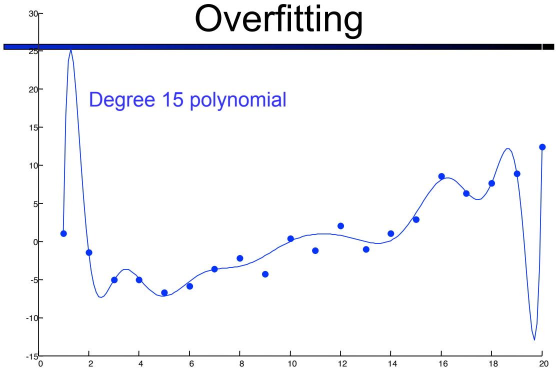
$$E \leftarrow E + \alpha \sum_i \left(\sum_k f_k(x_i) w_k - y_i\right) f_m(x_i)$$

Value update explained:

$$w_i \leftarrow w_i + \alpha\,[error]\,f_i(s,a)$$

48

Overfitting



[DEMO]

49

MDPs and RL Outline

- Markov Decision Processes (MDPs)
  - ✓ Formalism
  - ✓ Planning
    - Value iteration
    - Policy Evaluation and Policy Iteration
  - Reinforcement Learning --- MDP with T and/or R unknown
    - ✓ Model-based Learning
    - Model-free Learning
      - ✓ Direct Evaluation [performs policy evaluation]
      - ✓ Temporal Difference Learning [performs policy evaluation]
      - ✓ Q-Learning [learns optimal state-action value function Q*]
      - ➡ Policy search [learns optimal policy from subset of all policies]
    - ✓ Exploration vs. exploitation
  - ✓ Large state spaces: feature-based representations

50

Policy Search

51

Policy Search

- Problem: often the feature-based policies that work well aren't the ones that approximate V / Q best
  - E.g. your value functions from project 2 were probably horrible estimates of future rewards, but they still produced good decisions
  - We'll see this distinction between modeling and prediction again later in the course
- Solution: learn the policy that maximizes rewards rather than the value that predicts rewards
- This is the idea behind policy search, such as what controlled the upside-down helicopter

52

Policy Search

- Simplest policy search:
  - Start with an initial linear value function or q-function
  - Nudge each feature weight up and down and see if your policy is better than before
- Problems:
  - How do we tell the policy got better?
  - Need to run many sample episodes!
  - If there are a lot of features, this can be impractical

53

Policy Search*

- Advanced policy search:
  - Write a stochastic (soft) policy:

$$\pi_w(s) \propto e^{\sum_i w_i f_i(s,a)}$$

  - Turns out you can efficiently approximate the derivative of the returns with respect to the parameters w (details in the book, but you don't have to know them)
  - Take uphill steps, recalculate derivatives, etc.

54

MDPs and RL Outline

- Markov Decision Processes (MDPs)
  - Formalism
  - Value iteration
  - Expectimax Search vs. Value Iteration
  - Policy Evaluation and Policy Iteration
- Reinforcement Learning
  - Model-based Learning
  - Model-free Learning
    - Direct Evaluation [performs policy evaluation]
    - Temporal Difference Learning [performs policy evaluation]
    - Q-Learning [learns optimal state-action value function Q*]
    - Policy Search [learns optimal policy from subset of all policies]

55

To Learn More About RL

- Online book: Sutton and Barto

  http://www.cs.ualberta.ca/~sutton/book/ebook/the-book.html

- Graduate level course at Berkeley has reading material pointers online:*

  http://www.cs.berkeley.edu/~russell/classes/cs294/s11/

56

Take a Deep Breath…

- We're done with search and planning!
- Next, we'll look at how to reason with probabilities
  - Diagnosis
  - Tracking objects
  - Speech recognition
  - Robot mapping
  - … lots more!
- Last part of course: machine learning

57