1 Reinforcement Learning Basics

Melih Kandemir

Özyeğin University
Computer Science Department
melih.kandemir@ozyegin.edu.tr

26 Sep 2017

Outline

- Machine learning (now)
- Deep learning (next week)
- Reinforcement learning (rest of the semester)
- Deep reinforcement learning (second half of the semester)

What is machine learning?

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured in P, improves with experience E".

[Mitchell, 1997]

What is machine learning FOR?

To accomplish tasks that are impractical, infeasible, or impossible to program for humans.

e.g. How would you program a face recognizer?

Machine learning problems

- **Supervised learning:** N observations are provided as input-output pairs $(\mathbf{x}, y)$ and we are asked to learn a function f that maps inputs to the outputs $y = f(\mathbf{x})$.
- **Unsupervised learning:** N observations $\mathbf{x}$ are provided. We are asked to learn either the data distribution $p(\mathbf{x})$ (density estimation) or to form groups of similar observations (clusterings).
- **Reinforcement learning:** Coming soon.
- **Other:** There are many others, which are off the topic.

Some nomenclature

- **Hyperparameter:** Parameters control the model, hyperparameters control the parameters.
- **Regularization:** Any change in the objective function to avoid overfitting.
- **Training set:** The observations accessible to the model during training (the lecture).
- **Test set:** All the remaining obsevations the model is expected to predict well (the exam, or life).
- **Validation set:** Part of the training set used to tune the hyperparameters (sample exam questions).
- **Cross-validation:** A recipe to assess generalization performance. A partition of the data set is assigned to training and test in a round robin fashion.

A closer look at supervised learning

We are given a $N \times D$ matrix, called a *design matrix*, constituting $D-$dimensional observations (examples) in its rows:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}$$

where the nth row corresponds to a multidimensional sample (also called a *feature vector*):

$$\mathbf{x}_n = [x_{n1}, \cdots, x_{nD}]$$

consisting of singular measurements describing the object or even of interest x_{ij} (also called a *feature*).

A closer look at supervised learning

We are also given the corresponding outputs for the samples:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

Combined, $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ is called a *data set*.

The learning problem is called

- **classification** if $y_n \in \mathbb{Z}$ (set of integers, the magnitudes of which we do not care).
- **regression** if $y_n \in \mathbb{R}$ (set of real numbers).

Supervised learning example: Linear regression

We posit that the predictor function is defined as

$$\hat{y}_n = f(\mathbf{x}_n) = \mathbf{w}^T \mathbf{x}_n,$$

where $\mathbf{w}$ is the vector of model parameters.

For this model, learning means finding such a $\mathbf{w}$ that gives us accurate predictions for y.

Supervised learning example: Linear regression

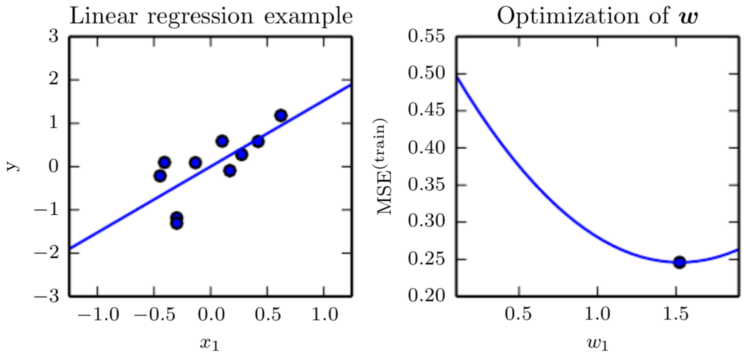


Figure. Goodfellow et al., Deep Learning, MIT Press, 2016

How would one quantify the goodness of a predictor?

Mean Squared Error (MSE): Euclidean distance between the observed output y_n and the predicted output $\hat{y}_n$. Formally,

$$MSE = \frac{1}{N} \sum_{n=1}^{N} ||y_n - \hat{y}_n||_2^2.$$

How would one quantify the goodness of a good predictor?

Mean Squared Error (MSE): Euclidean distance between the observed output y_n and the predicted output $\hat{y}_n$. Formally,

$$MSE = \frac{1}{N} \sum_{n=1}^{N} ||y_n - \hat{y}_n||_2^2.$$

Remember how we predict

$$\hat{y}_n = \mathbf{w}^T \mathbf{x}_n,$$

and plug it into the MSE definition

$$MSE = \frac{1}{N} \sum_{n=1}^{N} ||y_n - \underbrace{\mathbf{w}^T \mathbf{x}_n}_{\hat{y}_n}||_2^2.$$

How to learn?

We need to maximize our performance (remember Mitchell's definition of learning), hence minimize MSE

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} ||y_n - \hat{y}_n||_2^2.$$

How to minimize MSE?

Hint:

$$\begin{aligned} ||y_n - \hat{y}_n||_2^2 &= (y_n - \hat{y}_n)^2 \\ &= y_n^2 + (\mathbf{w}^T\mathbf{x}_n)^2 - 2y_n\mathbf{w}^T\mathbf{x}_n \\ &= y_n^2 + \mathbf{w}^T\mathbf{x}_n\mathbf{x}_n^T\mathbf{w} - 2y_n\mathbf{w}^T\mathbf{x}_n \end{aligned}$$

How to minimize MSE?

Calculate the gradient with respect to the model parameters $\mathbf{w}$ we aim to learn

$$
\begin{aligned}
\nabla_{\mathbf{w}} MSE &= \nabla_{\mathbf{w}} \sum_{n=1}^{N} ||y_n - \mathbf{w}^T \mathbf{x}_n||_2^2 \triangleq 0 \\
&= \nabla_{\mathbf{w}} \sum_{n=1}^{N} (y_n^2 + \mathbf{w}^T \mathbf{x}_n \mathbf{x}_n^T \mathbf{w} - 2 y_n \mathbf{w}^T \mathbf{x}_n) \\
&= \sum_{n=1}^{N} \nabla_{\mathbf{w}} (\mathbf{w}^T \mathbf{x}_n \mathbf{x}_n^T \mathbf{w}) - 2 \sum_{n=1}^{N} \nabla_{\mathbf{w}} y_n \mathbf{w}^T \mathbf{x}_n \\
&= \sum_{n=1}^{N} 2 \mathbf{x}_n \mathbf{x}_n^T \mathbf{w} - 2 \sum_{n=1}^{N} y_n \mathbf{x}_n
\end{aligned}
$$

How to minimize MSE?

Cont'd...

$$\begin{aligned}\nabla_{\mathbf{w}} MSE &= \sum_{n=1}^{N} 2\mathbf{x}_n\mathbf{x}_n^T\mathbf{w} - 2\sum_{n=1}^{N} y_n\mathbf{x}_n \\ &= 2\Big[\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^T\Big]\mathbf{w} - 2\sum_{n=1}^{N} y_n\mathbf{x}_n \\ &= \mathbf{X}\mathbf{X}^T\mathbf{w} - \mathbf{X}\mathbf{y}\end{aligned}$$

Now solve for the gradient at zero:

$$\mathbf{X}\mathbf{X}^T\mathbf{w} - \mathbf{X}\mathbf{y} \triangleq 0$$

$$\mathbf{X}\mathbf{X}^T\mathbf{w} \triangleq \mathbf{X}\mathbf{y}$$

$$\Big[\mathbf{X}\mathbf{X}^T\Big]^{-1}\Big[\mathbf{X}\mathbf{X}^T\Big]\mathbf{w} \triangleq \Big[\mathbf{X}\mathbf{X}^T\Big]^{-1}\mathbf{X}\mathbf{y}$$

$$\mathbf{w} \triangleq \Big[\mathbf{X}\mathbf{X}^T\Big]^{-1}\mathbf{X}\mathbf{y}$$

Underfitting and overfitting

We expect a good machine learning model to exhibit two properties

- make the training error small (*underfitting* otherwise),
- make the gap between training and test error small (*overfitting* otherwise).

The continuum between underfitting and overfitting can be traversed by tuning the *model capacity* (e.g. the degree of a polynomial, number of neurons/layers in a deep neural net).

Underfitting and overfitting

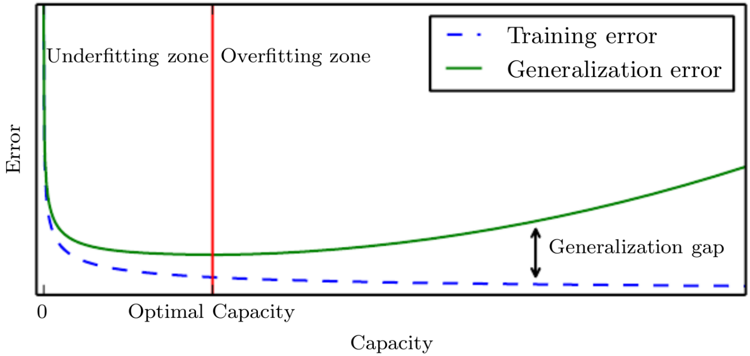


Figure. Goodfellow et al., Deep Learning, MIT Press, 2016

Occam's razor principle

"Among hypotheses explaining a set of observations with equal success, choose the simplest one."

(a.k.a. The Law of Parsimony)

Applied to machine learning: *"Among models performing equally well, choose the simplest one."*

How would you quantify model capacity?

Vapnik-Chervonenskis (VC) dimension

- **Definition.** Maximum number of arbitrarily placed data points that can be perfectly predicted.
- VC dimension of linear regression on D dimensional input and an intercept is $D + 1$ (i.e. number of model parameters).

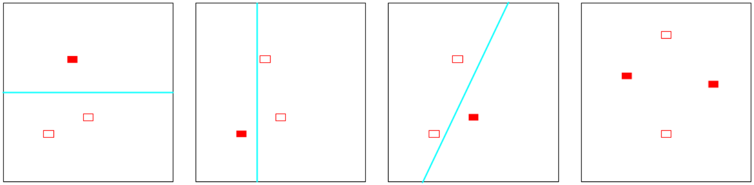


Figure. T. Hastie et al., The Elements of Statistical Learning, Springer, 2001

VC dimension of the sine function

- High capacity does not necessarily imply high parameter count.
- $\text{sine}(\alpha \cdot x)$ has one parameter α, but infinite VC dimension.

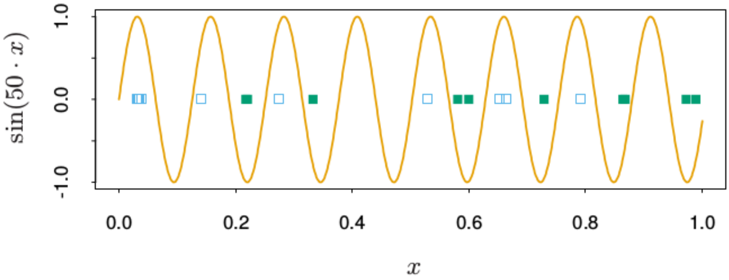


Figure. T. Hastie et al., The Elements of Statistical Learning, Springer, 2001

Quantifying the optimism of training error

(Only for binary classification for simplicity)

$$\mathrm{Err}_{Ts} = \mathrm{err}_{Tr} + \frac{\epsilon}{2}\left(1 + \sqrt{1 + \frac{4 \cdot \mathrm{err}_{Tr}}{\epsilon}}\right),$$

where

$$\epsilon = a_1 \frac{h[log(a_2 N/h) + 1] - \log(\eta/2)}{N}.$$

Here, h is the VC dimension of the predictor, err_{Tr} is the **training error** (or in-sample error) and $\mathrm{Err}_t s$ is the **test error** (or generalization error), and $0 < a_1 \leq 4$ and $0 < a_2 \leq 2$ are arbitrary coefficients satisfying the given inequalities.

Lastly, $opt = \mathrm{Err}_{Ts} - \mathrm{err}_{Tr}$ is the **optimism of the training error.**

For more details, see Cherkassky and Mulier, Learning from Data, Wiley, 2007.

What happens if $h \to +\infty$?

$$\mathrm{Err}_{Ts} = \mathrm{err}_{Tr} + \frac{\epsilon}{2}\left(1 + \sqrt{1 + \frac{4 \cdot \mathrm{err}_{Tr}}{\epsilon}}\right),$$

where

$$\epsilon = a_1 \frac{h[log(a_2 N/h) + 1] - \log(\eta/2)}{N}.$$

What happens if $h \to +\infty$?

$$\mathrm{Err}_{Ts} = \mathrm{err}_{Tr} + \frac{\epsilon}{2}\left(1 + \sqrt{1 + \frac{4 \cdot \mathrm{err}_{Tr}}{\epsilon}}\right),$$

where

$$\epsilon = a_1 \frac{h[log(a_2 N/h) + 1] - \log(\eta/2)}{N}.$$

Let's check out the asymptotic behavior: $\lim_{h \to +\infty} \epsilon = \infty$, hence $opt \to +\infty$ (overfitting!).

What happens if $N \to +\infty$?

$$\mathrm{Err}_{Ts} = \mathrm{err}_{Tr} + \frac{\epsilon}{2}\left(1 + \sqrt{1 + \frac{4 \cdot \mathrm{err}_{Tr}}{\epsilon}}\right),$$

where

$$\epsilon = a_1 \frac{h[log(a_2 N/h) + 1] - \log(\eta/2)}{N}.$$

What happens if $N \to +\infty$?

$$\mathrm{Err}_{Ts} = \mathrm{err}_{Tr} + \frac{\epsilon}{2}\left(1 + \sqrt{1 + \frac{4 \cdot \mathrm{err}_{Tr}}{\epsilon}}\right),$$

where

$$\epsilon = a_1 \frac{h[log(a_2 N/h) + 1] - \log(\eta/2)}{N}.$$

Let's check out the asymptotic behavior: $\lim_{N \to +\infty} \epsilon = 0$, hence $opt \to 0$ (actual fitting!).

Point estimation

- Attempt to find the single best prediction of some quantity of interest.
- Given a data set $\mathcal{D} = \{x_1, \cdots, x_N\}$, any function $f(\mathcal{D}) = \hat{\theta}_N$ is a *point estimator*.
- According to the *frequentist approach*, the true θ exists but it is unknown. Its point estimate $\hat{\theta}_N$ is a function of data. As data comes from a distribution, $\hat{\theta}_N$ is a random variable.

Bias of an estimator

A bias of an estimator $\hat{\theta}_N$ is defined as

$$\mathrm{bias}(\hat{\theta}_N) = \mathbb{E}[\hat{\theta}_N] - \theta.$$

An estimator is said to be **unbiased** if

$$\mathrm{bias}(\hat{\theta}_N) = 0,$$

or equivalently,

$$\mathbb{E}[\hat{\theta}_N] = \theta.$$

An estimator is said to be **asymptotically unbiased** if

$$\lim_{N\to\infty} \mathrm{bias}(\hat{\theta}_N) = 0.$$

Example 1: Sample mean is an unbiased estimator

Assume we have N samples $x_1, \cdots, x_N$ coming from a normal distribution $\mathcal{N}(\mu, \sigma^2)$. We are interested in finding out the true mean μ. Choose the **sample mean** as the estimator for μ:

$$\hat{\mu}_N = \frac{1}{N}\sum_{n=1}^{N} x_n.$$

Now check the bias of the estimator:

$$\begin{aligned}\text{bias}(\hat{\mu}_N) &= \mathbb{E}[\hat{\mu}_N] - \mu \\ &= \mathbb{E}\Big[\frac{1}{N}\sum_{n=1}^{N} x_n\Big] - \mu \\ &= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[x_n] - \mu = 0.\end{aligned}$$

It is unbiased!

Example 2: Sample variance is a biased estimator

$$\mathbb{E}[\hat{\sigma}_N^2] = \frac{N-1}{N}\sigma^2 \neq \sigma^2$$

The bias is $-\sigma^2/N$, which is ignorable if N is large [1].

A small fix, called *Bessel's correction*, solves the issue. The sample variance is unbiased if defined as

$$\hat{\sigma}_N^2 = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \hat{\mu})^2.$$

[1] Derivation is lengthy but simple, see: `https://www.marcovicentini.it/wp-content/uploads/2014/07/La-correlazione-di-Bessel.pdf`

Standard error

The square-root of the variance of an estimator is called its **standard error**:

$$SE(\hat{\theta}) = \sqrt{Var(\hat{\theta})}.$$

Taking the sample mean as the estimator, we have

$$SE(\hat{\theta}) = \sqrt{Var\left[\frac{1}{M}\sum_{m=1}^{M} x_m\right]} = \frac{\sigma}{\sqrt{M}},$$

which is called the **standard error of the mean.**

The bias-variance trade-off

Let f be a predictor that explains an observation set as $y = f(x) + \epsilon$, where $\mathbb{E}[\epsilon] = 0$ and $Var(\epsilon) = \sigma^2$, and $\hat{f}(x)$ an estimator for the predictor. The expected prediction error reads

$$\begin{aligned}
\mathbb{E}[(y - \hat{f}(x))^2] &= \mathbb{E}[y^2 - 2y\hat{f}(x) + \hat{f}(x)^2] \\
&= \mathbb{E}[(f(x) + \epsilon)^2 - 2(f(x) + \epsilon)\hat{f}(x) + \hat{f}(x)^2] \\
&= \mathbb{E}[f(x)^2 + 2f(x)\epsilon + \epsilon^2 - 2f(x)\hat{f}(x) \\
&\qquad - 2\epsilon\hat{f}(x) + \hat{f}(x)^2] \\
&= f(x)^2 + 2f(x)\underbrace{\mathbb{E}[\epsilon]}_{0} + \underbrace{\mathbb{E}[\epsilon^2]}_{\sigma^2} - 2f(x)\mathbb{E}[\hat{f}(x)] \\
&\qquad - 2\underbrace{\mathbb{E}[\epsilon]}_{0}\mathbb{E}[\hat{f}(x)] + \underbrace{\mathbb{E}[\hat{f}(x)^2]}_{\mathbb{E}[\hat{f}(x)]^2 + Var(\hat{f}(x)}
\end{aligned}$$

The bias-variance trade-off

$$\begin{aligned}\mathbb{E}[(y-\hat{f}(x))^2] &= \mathbb{E}[\epsilon^2] + f(x)^2 - 2f(x)\mathbb{E}[\hat{f}(x)] \\ &\quad + \mathbb{E}[\hat{f}(x)]^2 + Var(\hat{f}(x)) \\ &= \underbrace{\sigma^2}_{\text{Irreducible error}} + \underbrace{\mathbb{E}[f(x)-\hat{f}(x)]^2}_{\text{Bias}^2} + \underbrace{Var(\hat{f}(x))}_{\text{Variance}}\end{aligned}$$

Here,

- Irreducible error: Inaccuracy of the *ground-truth* predictor, which cannot be corrected by its estimator.
- Bias^2: Degree of inflexibility of the estimator (its level of ignorance to observations).
- Variance: Unstability of the estimator due to the limited sample size.

Where is the trade-off here?

- High bias reduces variance due to less dependence on data (e.g. always predict toss) $\Rightarrow$ Underfitting.
- Low bias assumes the model captures all the details of the data generating process from training data, hence increases dependence on the particular sample set. The outcome is subject to dramatical changes on a new sample set $\Rightarrow$ Overfitting.

Example: k-nearest neighbor regression

The estimator is $\hat{f}(x) = \frac{1}{k}\sum_{i=1}^{k} f(x_i)$, where $\{x_1, \cdots, x_k\}$ is the set of k nearest neighbors to x. Then,

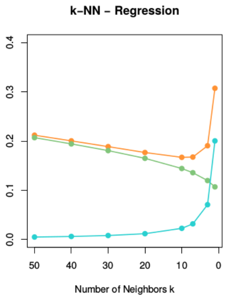
$$\text{err} = \sigma^2 + \left[f(x) - \frac{1}{k}\sum_{i=1}^{k} f(x_k)\right] + \underbrace{Var\Big(\frac{1}{k}\sum_{i=1}^{k} f(x_k)\Big)}_{SE(\hat{f}(x))^2}$$

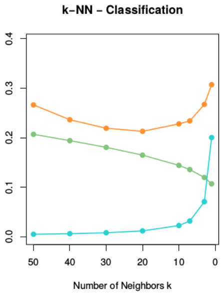
$$= \sigma^2 + \left[f(x) - \frac{1}{k}\sum_{i=1}^{k} f(x_k)\right] + \frac{\sigma^2}{k}.$$

- Increasing k increases bias. If $k = N$, the model outputs the sample mean regardless of the input.
- Increasing k decreases variance (third term).

No way to decrease both bias and variance, keeping the model family of f fixed.

Example: k-nearest neighbor regression





Regression   Classification

Orange: Expected prediction error (MSE), **Green:** squared bias, **Blue:** Variance

Figure. T. Hastie et al., The Elements of Statistical Learning, Springer, 2001

The bias-variance trade-off

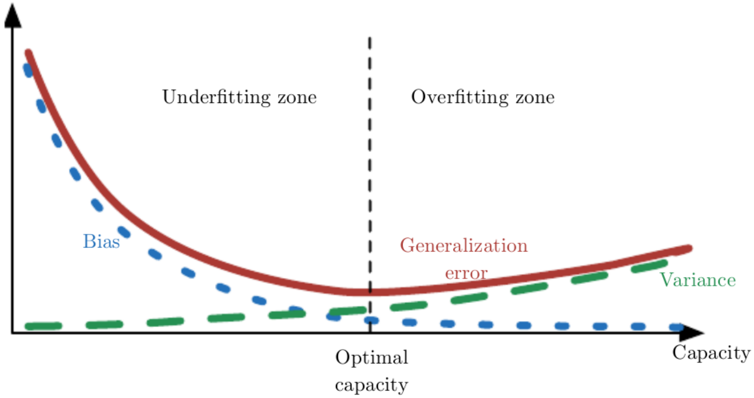


Figure. Goodfellow et al., Deep Learning, MIT Press, 2016

What the hell are we talking about? This is a Deep RL course!

- Nowadays, RL makes sense as much as you can plug a deep NN into your learning pipeline.
- Deep NNs are ultimately practical tools for playing with the model capacity (VC dimension).
- Know what risks to expect when you increase the number of neurons.
- Understand in advance why a huge NN is not all we need to solve an arbitrary ML problem.

The “No Free Lunch” theorem

Also understand in advance why a giant NN is not all we need to solve all ML problems. The obstacle facing this naive way out comes (magically) from the nature!

“Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.” [Wolpert, 1996]

Artists are required to sculp problem-specific biases.

We, *machine learners*, are those artists!

Position of RL in science

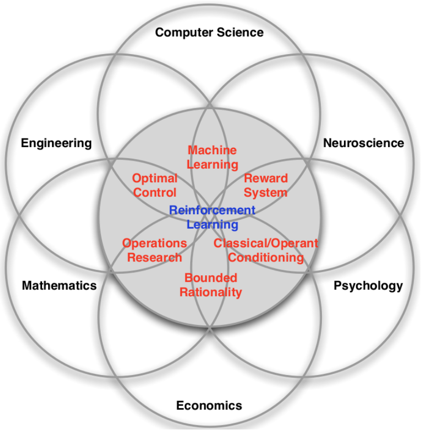


Figure. D. Silver, Lecture at UCL.

Characteristics of the RL setup

- The model *interacts* with the environment by actions it takes and the feedback, called the *reward*, it receives.
- Reward is the only supervision available.
- Reward is usually a sparse signal. Informative feedback could be delayed. May need to sacrifice immediate reward for the long-term goals.
  - Car driving (immediate reward)
    +: keep the lane, -: hit the barriers
  - Board game (delayed reward)
    +: win, -: lose
- Data is non-i.i.d. Present decisions affect future observations. The order of the observations matters!
- The goal of RL is to *learn to maximize the cumulative reward*.

Components of the RL setup

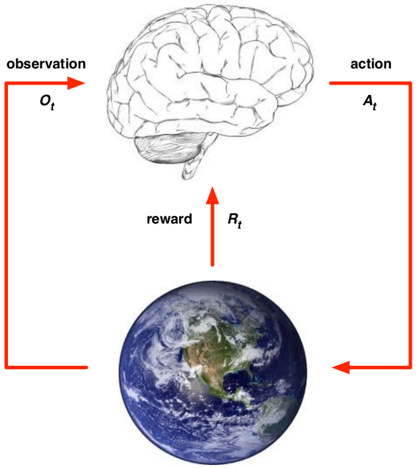


Figure D. Silver, Lecture at UCL.

Dynamics of the RL setup

	Agent	**Environment**
Action	Execute, A_t	Receive, A_t
Observation	Observe, O_t	Emit, O_{t+1}
Reward	Receive, R_t	Emit, R_{t+1}

History. Collection of all agent-environment interactions.
$H_t = O_1, R_1, A_1, O_2, R_2, A_2, \cdots, O_{t-1}, R_{t-1}, A_{t-1}, O_t, R_t$

Agent State. Any function of the history the agent uses to represent the situation of itself. $S_t^a = f(H_t)$.

Environment State. A representation of the environment that contains all useful details relevant for the learning task. S_t^e.

Markov state

(a.k.a. information state)

Definition

A state S_t is Markov if and only if

$$P(S_{t+1}|S_t) = P(S_{t+1}|S_1, \cdots, S_t).$$

$$\blacksquare$$

The future is independent of the past given the present.

$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$

Example

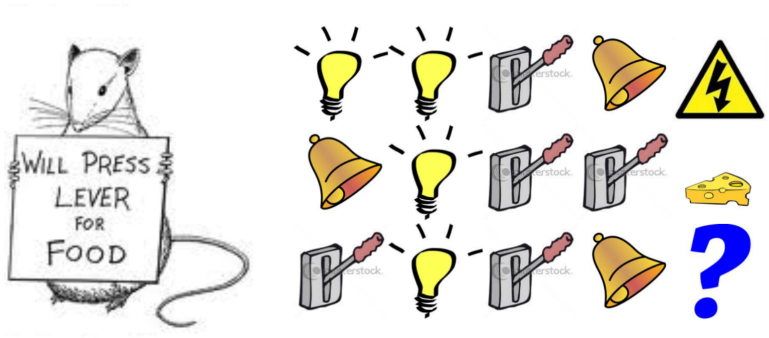


Figure D. Silver, Lecture at UCL.

Fully and partially observable environment

- **Full:** $O_t = S_t$, formally a Markov Decision Process (MDP). e.g. chessboard.
- **Partial:** Agent must construct its own representation of the environment, formally a Partially-Observed MDP (POMDP).
  e.g. the physical conditions of the sky when flying a helicopter

Ingredients of an RL algorithm

- **Policy:** Agent's behavior function that maps states to actions.
  - **Deterministic policy:** $A_t = \pi(S_t)$
  - **Stochastic policy:** $\pi(A_t|S_t) = P(A_t|S_t)$
- **Value Function:** Goodness measure for states and actions. Formally defined as the expected ($\mathbb{E}$) cumulative ($+\cdots+$) reward (R_t) of a state (S_t) for a given policy (π).

$$v_\pi(S_t) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t]$$

- **Model:** Agent's representation of the environment behavior.

$$\begin{aligned} \mathcal{P}^a_{ss'} &= P(S_{t+1} = s' | S_t = s, A_t = a), \qquad (i) \\ \mathcal{R}^a_s &= \mathbb{E}[R_{t+1} | S_t = s, A_t = a]. \qquad (ii) \end{aligned}$$

  i) How do the states evolve? **ii)** How are the rewards generated?

Maze example

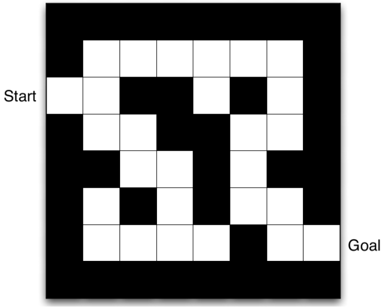


Figure D. Silver, Lecture at UCL.

- Rewards: -1 per move
- Actions: N, E, S, W
- States: Location

Maze example: The policy

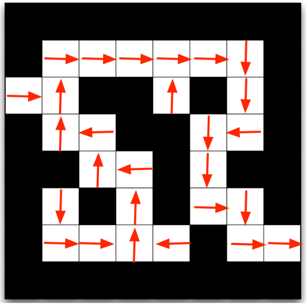


Figure D. Silver, Lecture at UCL.

Cells are states, arrows are actions.

Maze example: Value function

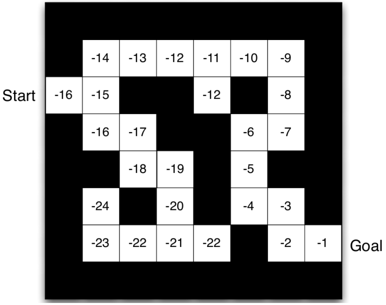


Figure D. Silver, Lecture at UCL.

Values of maze positions (states) shown in cells.

Maze example: Model

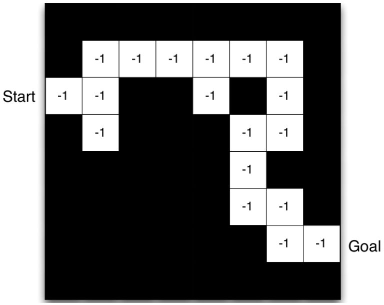


Figure D. Silver, Lecture at UCL.

- Grid layout is the transition model ($\mathcal{P}^a_{ss'}$).
- Numbers are the immediate rewards of states ($\mathcal{R}^a_s$).

Types of RL agents

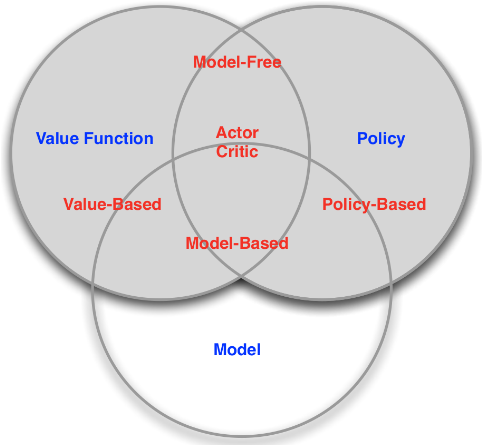


Figure D. Silver, Lecture at UCL.

What shall I be able to do with the Deep RL course material?

The AI engine of an atari game. Mind that the observation is a raw set of pixels!

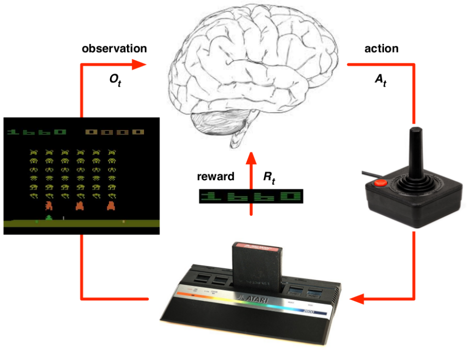


Figure D. Silver, Lecture at UCL.

Exploration and Exploitation

- **Exploration** discovers the environment. e.g. try a new restaurant.
- **Exploitation** maximizes the reward exploiting what it knows at present. e.g. go to your favorite restaurant.

An ideal model finds an optimal trade-of between exploration and exploitation.

Prediction and Control

- **Prediction:** Given a policy, evaluate the future.
- **Control:** Learn the best policy from experience.

What is RL for? i) Robot control

https://www.youtube.com/watch?v=Fhb26WdqVuE

What is RL for? ii) AI engine development

https://www.youtube.com/watch?v=efbOGCO_Ac8

What is RL for? iii) Autonomous driving

https://www.youtube.com/watch?v=n737OdzCZ1o