Hierarchical Reinforcement Learning

Aravind Rajeswaran and Kendall Lowrey
May 30, 2018

Lecture Contents

Where does RL work, not work?

How can we scale our current algorithms to bigger problems?

What other efficiencies can we seek?

How Much Information Does the Machine Need to Predict?

Y LeCun

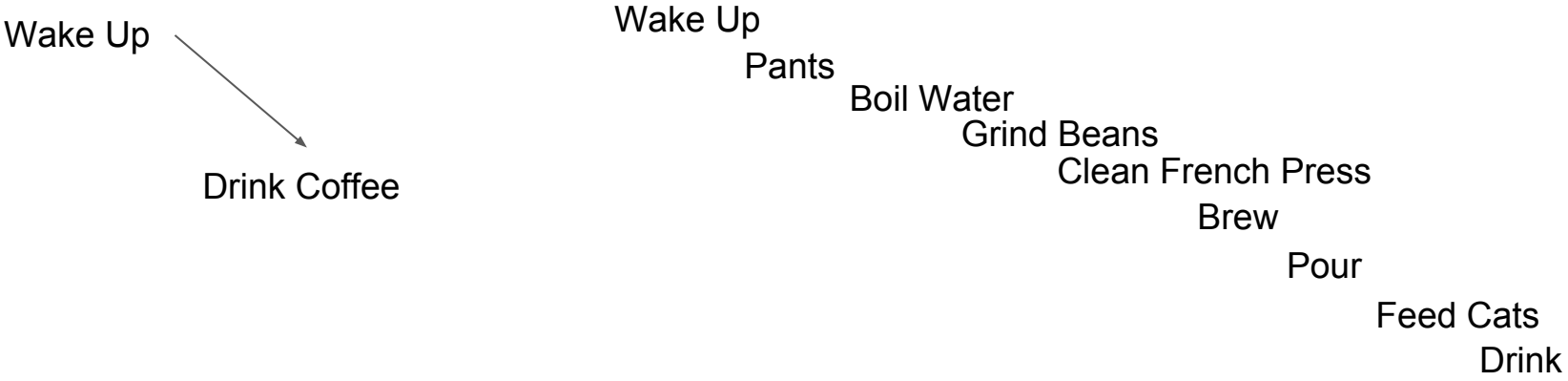
- "Pure" Reinforcement Learning (cherry)
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- Supervised Learning (icing)
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- Unsupervised/Predictive Learning (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

Hierarchies for RL

If the algorithms we're familiar with can solve 'shallow' problems, what can we do to solve more complex ones?

Use the 'shallow' solutions to solve more complex problems: need help with exploration and knowledge re-use.

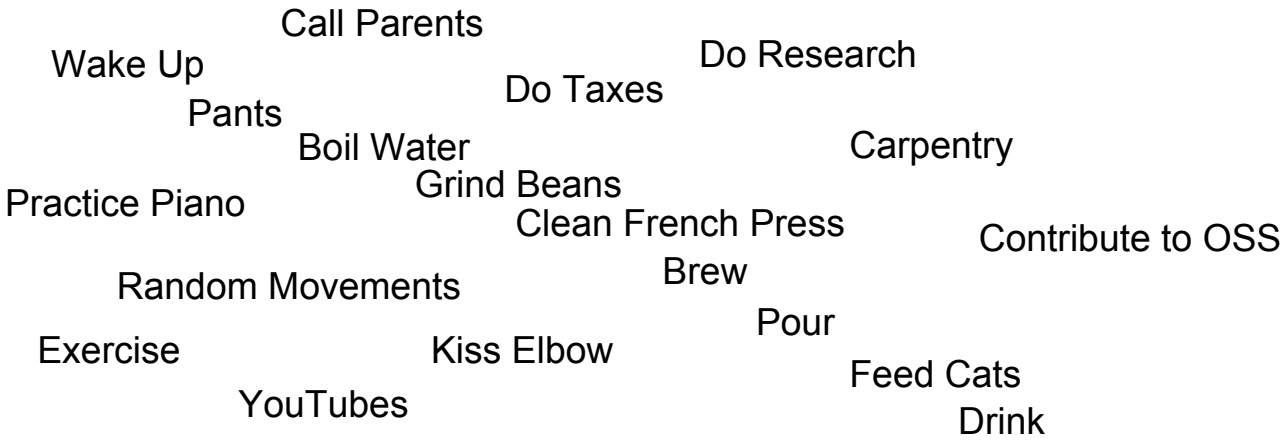


Hierarchies for RL

RL is informed by the problem structure (the MDP) to be efficient

Hierarchical RL structures the potential solution

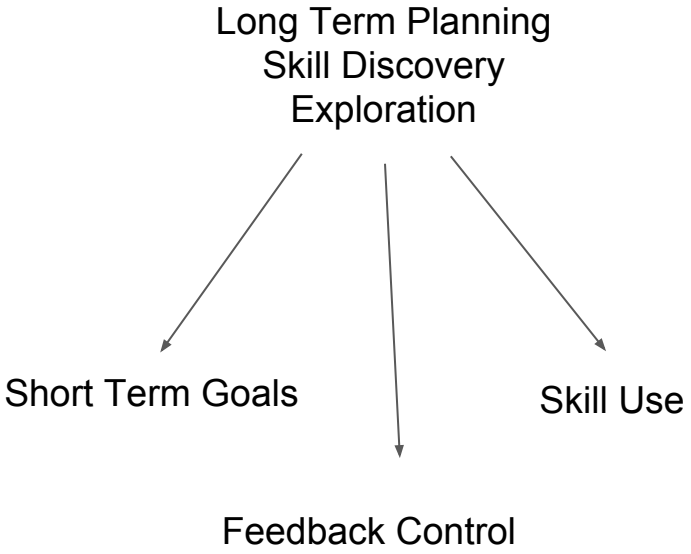
Limiting or directing exploration helps efficiency



Hierarchies for RL

A goal in a hierarchy is to allow for some specialization.

Instead of one policy doing everything, we can try to assign roles to the layers of the hierarchy.



What are our Options?

Options Framework: an option is an encapsulation of multiple low level actions

They can represent accomplishing ‘higher level’ goal, a policy, a different reward function, etc.

Training a policy (Q-function) that selects among options is the same as a policy that selects among actions through the use of a semi-MDP.

Can be viewed as temporal abstraction, sometimes spatial.

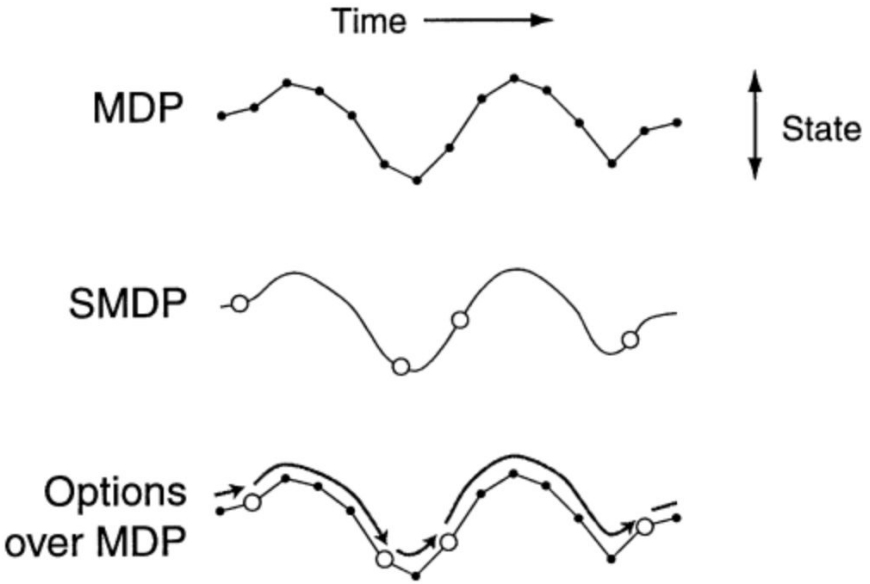
http://www-anw.cs.umass.edu/~barto/courses/cs687/Sutton-Precup-Singh-AIJ99.pdf

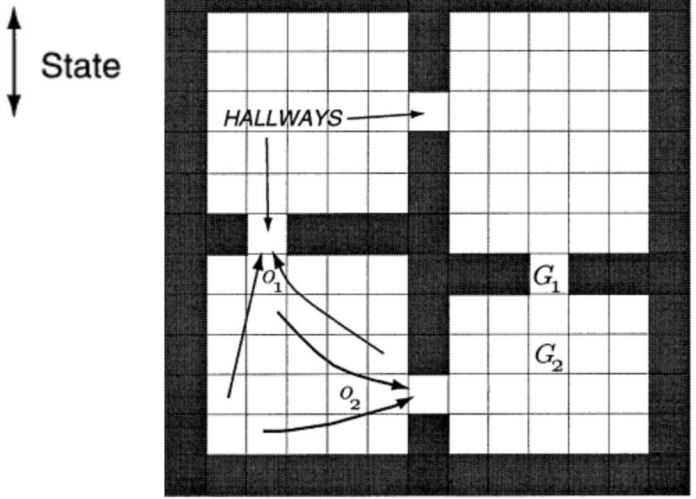
Learning with Options

$$
\begin{aligned}
Q^*(s,a) &= \max_{\pi} Q^{\pi}(s,a) \\
&= r_s^a + \gamma \sum_{s'} p_{ss'}^a \max_{a'} Q^*(s',a')
\end{aligned}
$$

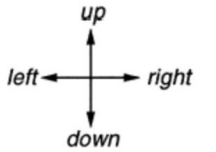
$$
\begin{aligned}
Q_{\mathcal{O}}^*(s,o) &\stackrel{\text{def}}{=} \max_{\mu \in \Pi(\mathcal{O})} Q^{\mu}(s,o) \\
&= E\left\{ r_{t+1} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^k V_{\mathcal{O}}^*(s_{t+k}) \,\middle|\, \mathcal{E}(o,s,t) \right\} \\
&\qquad \text{(where } k \text{ is the duration of } o \text{ from } s\text{)} \\
&= E\left\{ r_{t+1} + \cdots + \gamma^{k-1} r_{t+k} + \gamma^k \max_{o' \in \mathcal{O}_{s_{t+k}}} Q_{\mathcal{O}}^*(s_{t+k}, o') \,\middle|\, \mathcal{E}(o,s,t) \right\}, \\
&= r_s^o + \sum_{s'} p_{ss'}^o \max_{o' \in \mathcal{O}_{s'}} Q_{\mathcal{O}}^*(s',o') \\
&= E\left\{ r + \gamma^k \max_{o' \in \mathcal{O}_{s'}} Q_{\mathcal{O}}^*(s',o') \,\middle|\, \mathcal{E}(o,s) \right\},
\end{aligned}
$$

Options





4 stochastic primitive actions



Fail 33% of the time

8 multi-step options

(to each room's 2 hallways)

What is an Option Doing?

Combining a sequence of states/actions into a subgoal (i.e. do task)

Searching among a set of options should be easier than searching in state/action space directly: helps our agent to explore.

> If not, need better options!

Where do our options come from? If we think of an option as optimizing for a subgoal, need to have the correct subgoals provided.

See also: HAMs, MAXQ

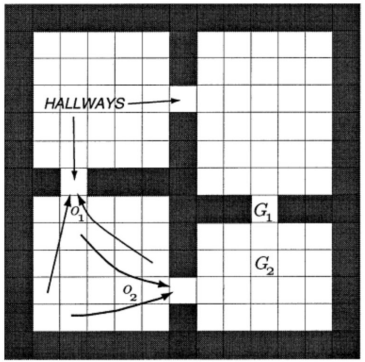
Limited Options

Options are pre-defined; even if you may not use all options, there is a set provided.

Searching for options is akin to the original non-hierarchical problem: at the limit, each state/action pair is an action.

‘Simple’ options may be discoverable, but still require prior knowledge or changing the problem. (see Feudal RL)

What is the most generic way we can help an agent?



Intrinsic Motivation

The goal of an RL agent is to accomplish a given task: ***extrinsic motivation***

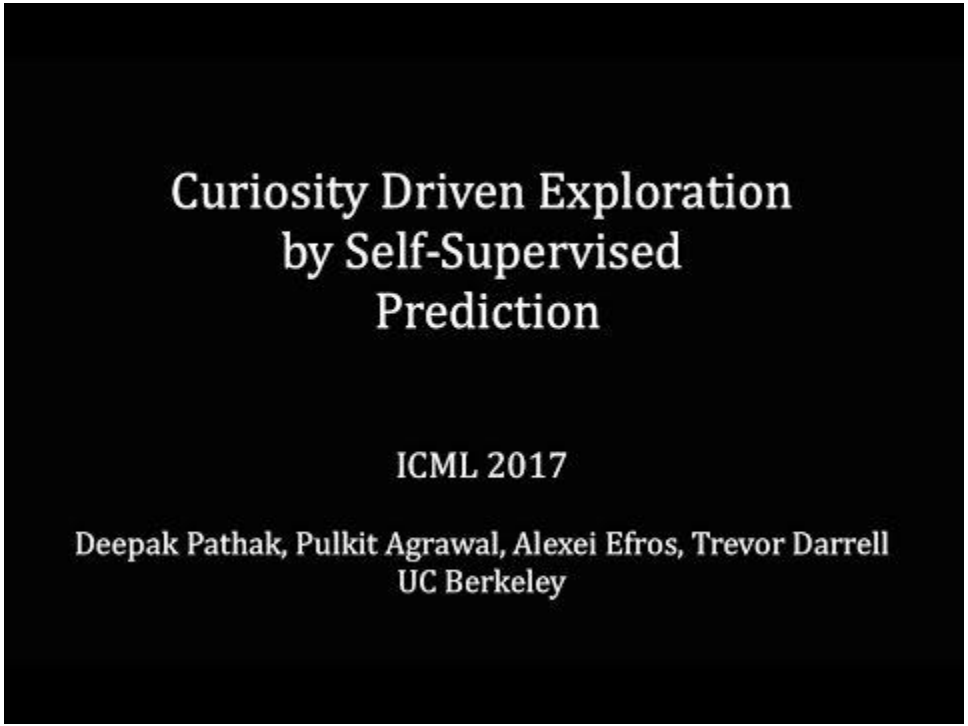
Intrinsic motivation: some agent specific goals to accomplish alongside the task

Or… if you cannot do the task, do this instead!

Can be used in lieu of extrinsic motivation…

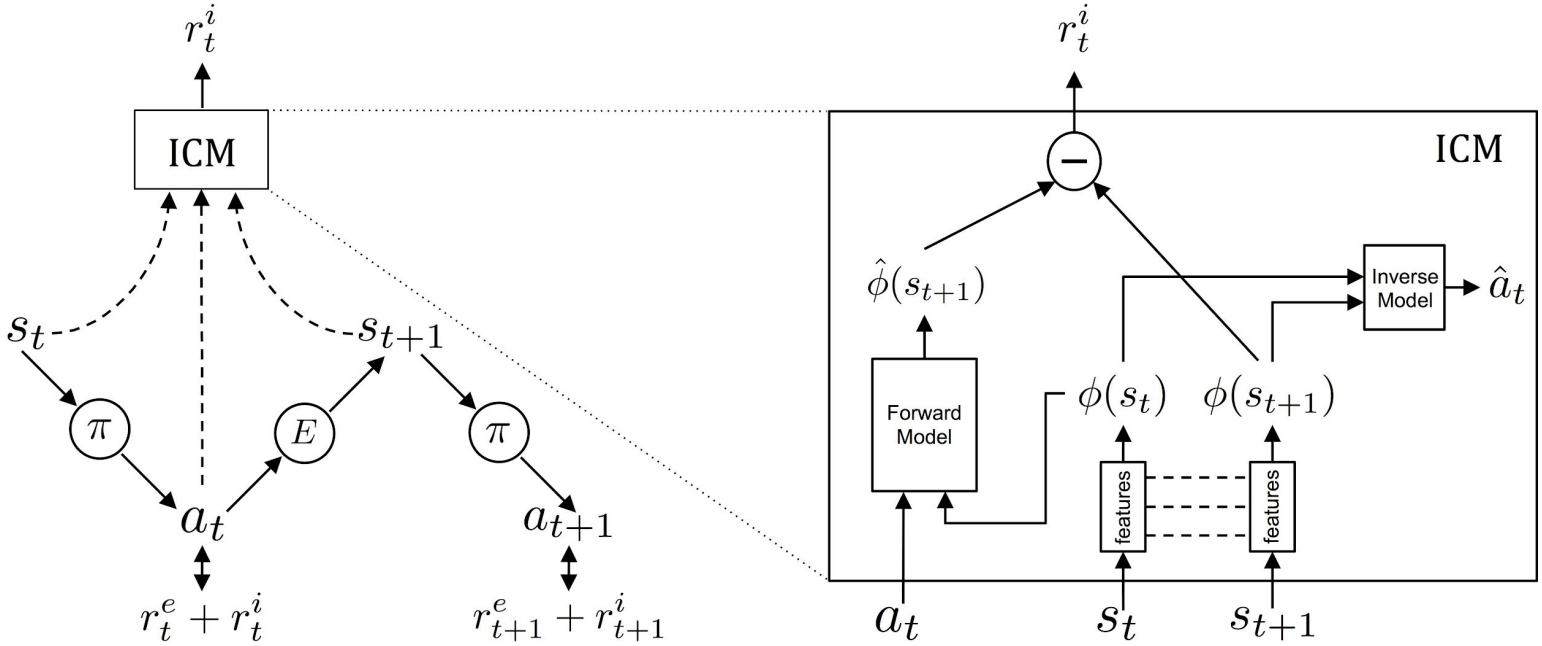
“Curiosity” has been a recently popular intrinsic motivation criteria.

Curiosity Driven Exploration



https://pathak22.github.io/noreward-rl/

Curiosity Driven Exploration

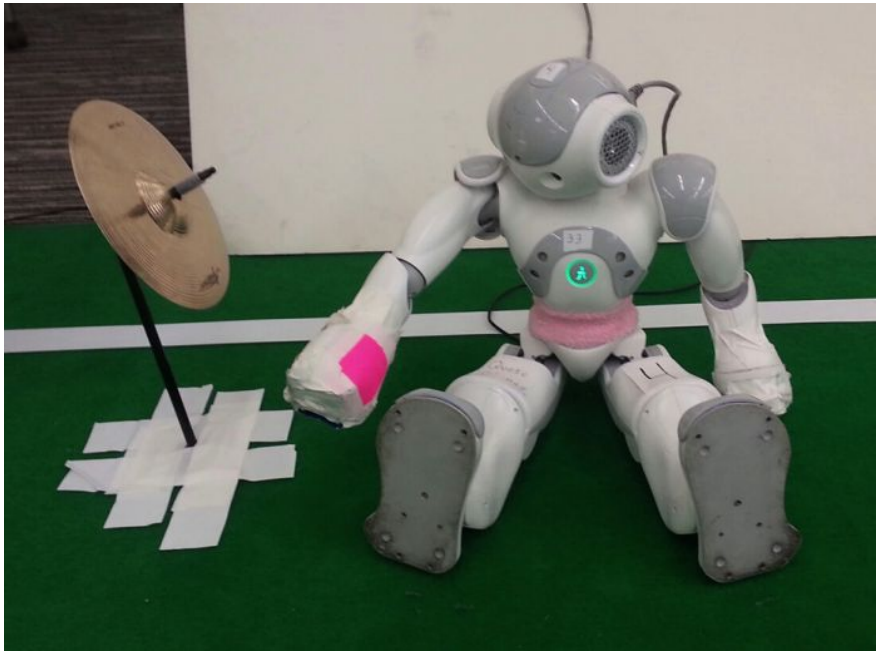


Intrinsically Motivated Model Learning for Developing Curious Robots

Keep a model of the dynamics using Random Forests

When model (forests) disagree with each other, then be optimistic wrt the most rewarding state transition.

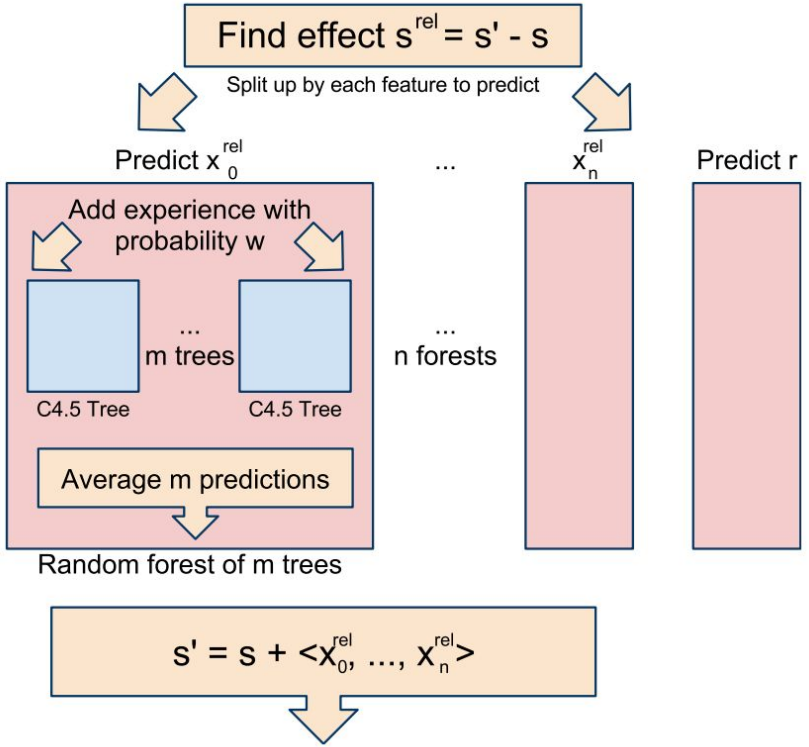
Encourages exploration of entire state space



http://www.cs.utexas.edu/~ai-lab/pubs/ICDL12-hester.pdf or https://ac.els-cdn.com/S0004370215000764/1-s2.0-S0004370215000764-main.pdf?_tid=5cf10552-8809-48a0-95a9-b4ed9278c758&acdnat=1527708608_6fcfacebf4025649c3e9c33cf9659d74

Intrinsically Motivated Model Learning for Developing Curious Robots

$$Q(s,a) = \frac{1}{m}\sum_{i=1}^{m} R_i(s,a) + \gamma \frac{1}{m}\sum_{i=1}^{m}\sum_{s'} P_i(s'|s,a)\max_{a'} Q(s',a')$$



Diversity is all you need

Previously we saw ways of encouraging exploration of the state

Maybe during that exploration, we also did the extrinsic task…

How can we re-use the exploration?

https://sites.google.com/view/diayn

Diversity is all you need

Algorithm 1 DIAYN

Input: skill distribution $p(z)$
repeat
  Sample skill $z \sim p(z)$ and initial state $s_0 \sim p_0(s)$.
  for $t = 1$ **to** $steps\_per\_episode$ **do**
    Sample action $a_t \sim \pi_\theta(a_t \mid s_t, z)$ from skill.
    Step environment: $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$.
    Compute $q_\phi(z \mid s_{t+1})$ with discriminator.
    Set skill reward $r_t = \log q_\phi(z \mid s_{t+1}) - \log p(z)$
    Update policy (θ) to maximize r_t with SAC.
    Update discriminator (ϕ) with SGD.
  end for
until

Is Diversity enough?

Seems like a way of finding options, but are limited to options that are different between states.

A hand crafted option can be an entire extrinsic task with a reward function designed by you; options thus have diversity in task space instead of diversity in state space.

Can we bridge this gap?

- Augment the state vector with task signals
- Diversity in some kind of ‘concept’ space or feature space

Other links...

HAMs https://papers.nips.cc/paper/1384-reinforcement-learning-with-hierarchies-of-machines.pdf

Feudal http://www.cs.toronto.edu/~fritz/absps/dh93.pdf

OO-MDP http://carlosdiuk.github.io/papers/OORL.pdf

Semi-mdp http://www-anw.cs.umass.edu/~barto/courses/cs687/Sutton-Precup-Singh-AIJ99.pdf

RL summary with modern HRL list: https://arxiv.org/pdf/1701.07274.pdf