# Task Sheet 3
# Putting It All Together

**Context.** You now have a realistic scenario, an environment and a replay buffer. This week you implement *Deep Q-Learning* and watch the agent learn.

## Learning goals

- Implement the $\varepsilon$-greedy exploration schedule.
- Code the Bellman update and periodic target-network sync.
- Evaluate your trained agent against buy-and-hold.

## Exercises

1. (**Transfer from previous tasks**) Transfer the functions for environment, replay buffer etc. from previous tasks into your R document. Also, include the dataset from task 1.
2. (**Data split**) Split the return data by using the first 80 percent of returns as training and the rest as test data.
3. (**Network factory**) Wrap task 1's network definition into `build_qnet()` so the same architecture can be created twice (online & target). (Glossary:"Online (Q-)Network", "Target (Frozen) Network")
4. (**Random action function**) Study, what an $\varepsilon$-greedy action is. Then, create a function `epsilon()`, which computes the current $\varepsilon$ based on the current episode for the $\varepsilon$-greedy decision. It should start at $\varepsilon_{start}$ and exponentially decay with rate $e^{-dr*ep}$ to $\varepsilon_{final}$, dependent on the current episode $ep$ and decay rate $dr$. (Glossary: "$\varepsilon$-greedy", "Episode")
5. (**Hyper-parameters**) Choose sensible values for $\gamma$, learning rate, replay capacity, warm-up size (number of observations until replay buffer is ready for training use), `max_steps_ep` (maximum number of state transitions per episode), $\varepsilon_{start}$, $\varepsilon_{final}$ and decay rate. Justify your choice with one sentence each.
6. (**Training loop**) For each of overall 10 episodes:
   (a) Reset the environment and obtain the initial state.
   (b) Until `done` OR `max_steps_ep` are reached:
      i. select an action via $\varepsilon$-greedy (Glossary: "Exploration vs. Exploitation") ,
      ii. call `env_step` and store the transition,
      iii. once warm-up is over, sample a mini-batch and compute:

$$y = r + \gamma\,(1 - d)\,\max_a Q_{\text{target}}(s', a),$$

      using the target net. Calculate your prediction using the online net. Then update the online net using gradient descent and MSE as cost/loss function. The Keras function `train_on_batch()` is very useful here.
   (c) Every 10 episodes copy weights from the online to target network.
   Log episode reward and equity.
7. (**Visual inspection**) After training, run one episode with $\varepsilon = 0$ and plot the agent's equity over time using only the training data. Would an investor who just buys at the beginning and never sells be better off? (buy-and-hold)
8. (**Discussion**) Argue how trade costs affect the result and what you could do to improve the result.

## Reflection questions

- Explain "bootstrapping" in the Bellman update.

- Why is a separate target network more stable than using the online weights directly as target?
- How would you extend the action space to *short selling*?