DALHOUSIE
COMPUTATIONAL NEUROSCIENCE GROUP
Studying Minds

Reinforcement learning

HAL
Hierarchical
Anticipatory
Learning

Thomas Trappenberg

Three kinds of learning:

1. Supervised learning

Detailed teacher that provides desired output **y** for a given input x: training set {**x**,**y**}

→ find appropriate mapping function **y**=h(**x**;**w**) [= W φ(x)]

2. Unsupervised Learning

Unlabeled samples are provided from which the system has to figure out good representations: training set {**x**}

→ find sparse basis functions b_i so that $x=\Sigma_i c_i b_i$

3. Reinforcement learning

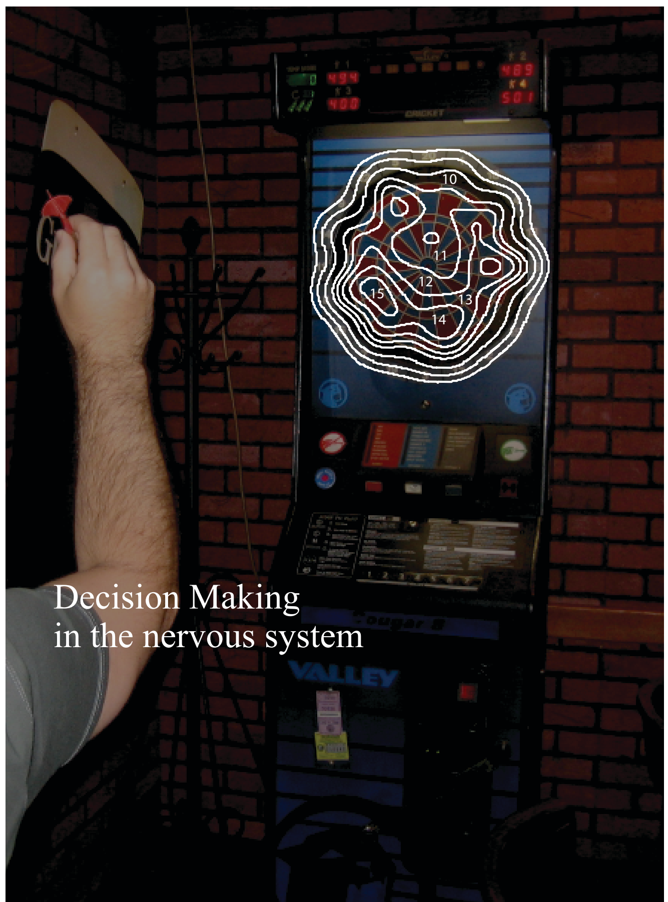
Delayed feedback from the environment in form of reward/punishment when reaching state **s** with action **a**: reward r(**s**,**a**)

→ find optimal policy **a**=π*(**s**)

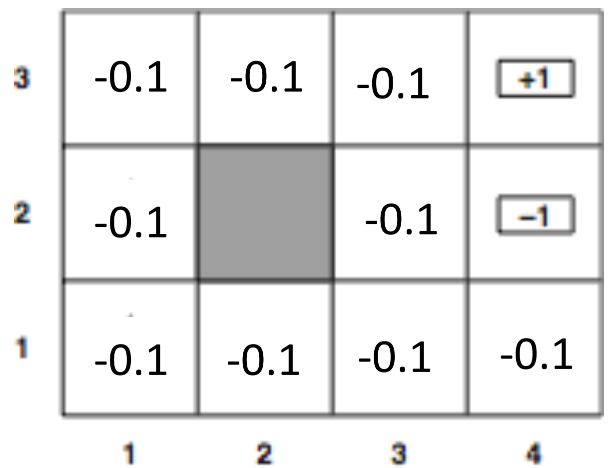
Most general learning circumstances

Maximize expected Utility

$$\hat{\pi} = \arg\max_{\pi} \sum_{o} U(o)p(o|\pi)$$



2. Reinforcement learning



From Russel and Norvik

Markov Decision Process (MDP)

$$(S, A, T(s'|s,a), R(r|s,a), \theta)$$

- S is a set of states.
- A is a set of actions.
- $T(s'|s,a)$ is a **transition probability**, for reaching state s' when taking action a from state s. This transition probability only depends on the previous state, which is called the Markov condition; hence the name of the process.
- $R(r|s)$ is the probability of receiving **reward** when getting to state s. This quantity provides feedback from the environment. r is a numeric value with positive values indicating reward and negative values indicating punishment.
- θ are specific parameters for some of the different kinds of RL settings. This will be the **discount factor** γ in our first examples.

Two important quantities

policy: $\pi(a|s)$

value function:

$$Q^{\pi}(s,a) = E\{r(s) + \gamma r(s_1) + \gamma^2 r(s_2) + \gamma^3 r(s_3) + \ldots\}_{\pi}$$

Goal: maximize total expected payoff

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

Optimal Control

$$\pi^{*}(a|s) = \arg\max_{\pi} Q^{\pi}(s,a)$$

Calculate value function (dynamic programming)

Deterministic policies to simplify notation $Q^\pi(s,a) = V^\pi(s)$

$$\begin{aligned} V^\pi(s) &= E\{r(s) + \gamma r(s_1) + \gamma^2 r(s_2) + \gamma^3 r(s_3) + \ldots\}_\pi \\ &= E\{r(s)\}_\pi + \gamma E\{r(s_1) + \gamma r(s_2) + \gamma^2 r(s_3) + \ldots\}_\pi \\ &= r(s) + \gamma \sum_{s'} T(s'|s,a) E\{r(s') + \gamma R(s'_1) + \gamma^2 R(s'_2) + \ldots\}_\pi \end{aligned}$$

$$V^\pi(s) = r(s) + \gamma \sum_{s'} T(s'|s,a) V^\pi(s').$$

Bellman Equation for policy π

Richard Bellman
1920-1984

Solution: Analytic or Incremental

$$\mathbf{r} = (\mathbb{1} - \gamma \mathbf{T}) \mathbf{V}^\pi$$

$$\mathbf{V}^\pi = (\mathbb{1} - \gamma \mathbf{T})^{-1} \mathbf{r}^t$$

$$\mathbf{V} \leftarrow \mathbf{r} + \gamma \mathbf{T} \mathbf{V}$$

Remark on different formulations:

Some (like Sutton, Alpaydin, but not Russel & Norvik) define the value as the reward at the next state plus all the following reward:

$$V^\pi(s) = \sum_{s'} (r(s') + \gamma T(s'|s,a) V^\pi(s'))$$

instead of

$$V^\pi(s) = r(s) + \gamma \sum_{s'} T(s'|s,a) V^\pi(s').$$

Policy Iteration

```
Choose initial policy and value function
Repeat until policy is stable {
        1. Policy evaluation
        Repeat until change in values is sufficiently small {
                For each state {
                        Calculate the value of neighbouring states when taking
                          action according to current policy.
                        Update estimate of optimal value function.
                } each state
        } convergence
        2. Policy improvement
        new policy according to equation 10.21, assuming V* ≈ current V^π
} policy
```

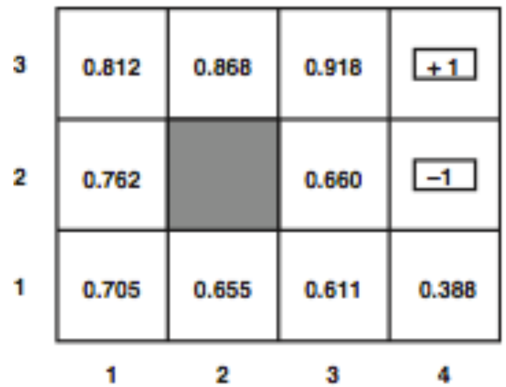
Value Iteration

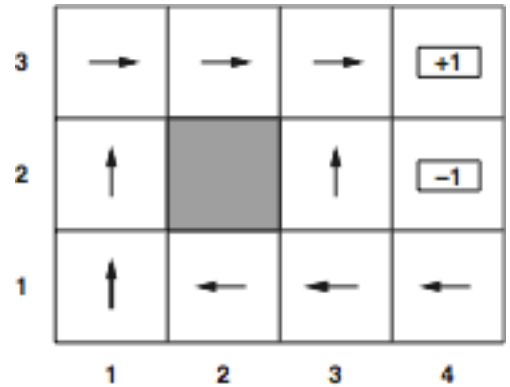
Bellman Equation for optimal policy

$$V^*(s) = r(s) + \max_a \gamma \sum_{s'} T(s'|s,a) V^*(s')$$

Choose initial estimate of optimal value function
Repeat until change in values is sufficiently small {
    For each state {
        Calculate the maximum expected value of neighbouring states for each possible action.
        Use maximal value of this list to update estimate of optimal value function.
    } each state
} convergence
Calculate optimal value function from equation 10.21

Solution:





But:

Environment not known a priori → Online (TD)
Observability of states → POMDP
Curse of Dimensionality → Model-based RL

POMDP:

Partially observable MDPs can be reduced to MDPs by considering believe states b:

$$Q(s,a) = r + \sum_{b'} \gamma P(b'|b,a) Q(s',a)$$

What if the environment is not completely known ?

Online value function estimation (TD learning)

If the environment is not known,
use Monte Carlo method with bootstrapping

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha\{r(s) + \gamma V^{\pi}(s') - V^{\pi}(s)\}$$

Expected payoff before taking step

Expected reward after taking step = actual reward plus discounted expected payoff of next step

Temporal Difference

Online optimal control: Exploitation versus Exploration

ϵ-greedy policy $\pi(a = \arg\max_a Q(s,a)) = \epsilon$

softmax policy $\pi(a|s) = \frac{e^{Q(s,a)}}{\sum_{a'} e^{Q(s,a')}}$

On-policy TD learning: Sarsa

$$Q(s,a) \leftarrow Q(s,a) + \alpha\{r(s) + \gamma Q(s',a') - Q(s,a)\}$$

Off-policy TD learning: Q-learning

$$Q(s,a) \leftarrow Q(s,a) + \alpha\{r(s) + \max_{a'} \gamma Q(s',a') - Q(s,a)\}$$

Model-based RL: TD(1)

Instead of tabular methods as mainly discussed before, use function approximator with parameters θ and gradient descent step (Satton 1988):

$$V_t(\mathbf{x}_t) \approx V_t(\mathbf{x}_t; \theta)$$

For example by using a neural network with weights θ and corresponding delta learning rule

$$\Delta\theta = \alpha \sum_{t=1}^{m} (r - V_t) \frac{\partial V_t}{\partial \theta}$$

when updating the weights after an episode of m steps.
The only problem is that we receive the feedback r only after the t-th step. So we need to keep a memory (trace) of the sequence.

Model-based RL: TD(1) … alternative formulation

We can write

$$r - V_t = \sum_{k=t}^{m} (V_{k+1} - V_k)$$

An putting this into the formula and rearranging the sum gives

$$\begin{aligned}\Delta\theta &= \alpha \sum_{t}^{m} \sum_{k=t}^{m} (V_{k+1} - V_k) \frac{\partial V_t}{\partial \theta} \\ &= \alpha \sum_{t=1}^{m} (V_{t+1} - V_t) \sum_{k=1}^{t} \frac{\partial V_k}{\partial \theta}\end{aligned}$$

We still need to keep the cumulative sum of the derivative terms, but otherwise it looks already closer to bootstrapping.

Model-based RL: TD(λ)

We now introduce a new algorithm by weighting recent gradients more than ones in the distance

$$\Delta_t \theta = \alpha (V_{t+1} - V_t) \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial V_k}{\partial \theta}$$

This is called the TD(λ) rule. For $\lambda=1$ we recover the TD(1) rule. Interesting is also the the other extreme of TD(0)

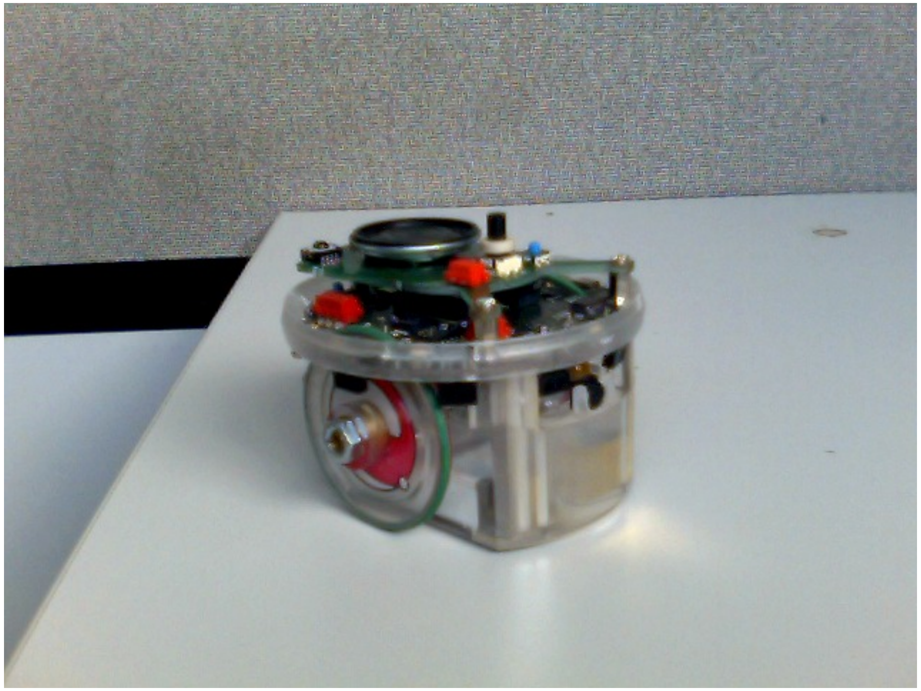
$$\Delta_t \theta = \alpha (V_{t+1} - V_t) \frac{\partial V_t}{\partial \theta}$$

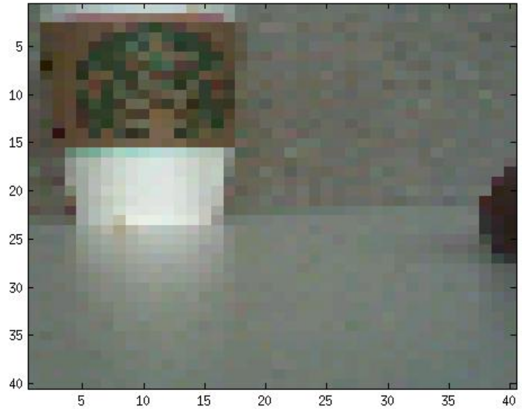
Which uses the prediction of V(t+1) as supervision signal for step t. Otherwise this is equivalent to supervised learning and can easily be generalized to hidden layer networks.

Free-Energy-Based RL:

This can be generalized to Boltzmann machines
(Sallans & Hinton 2004)

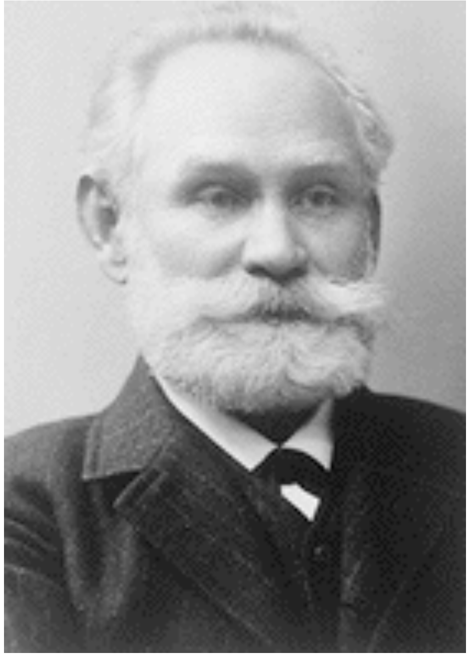
Paul Hollensen:
Sparse, topographic RBM successfully learns to drive the e-puck and avoid obstacles, given training data (proximity sensors, motor speeds)






rbm_driving_epuck.avi

Classical Conditioning

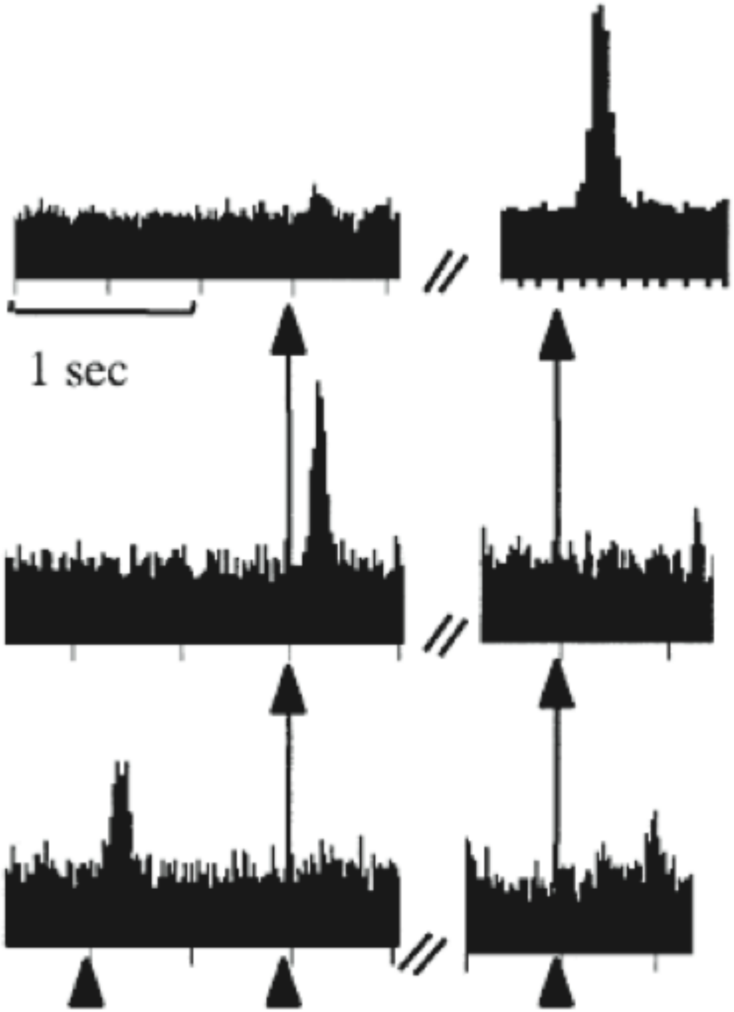


Ivan Pavlov
1849-1936
Nobel Prize 1904

Rescorla-Wagner Model (1972)

$$\Delta V_i = \alpha_i \beta(\lambda - \Sigma V_j)$$

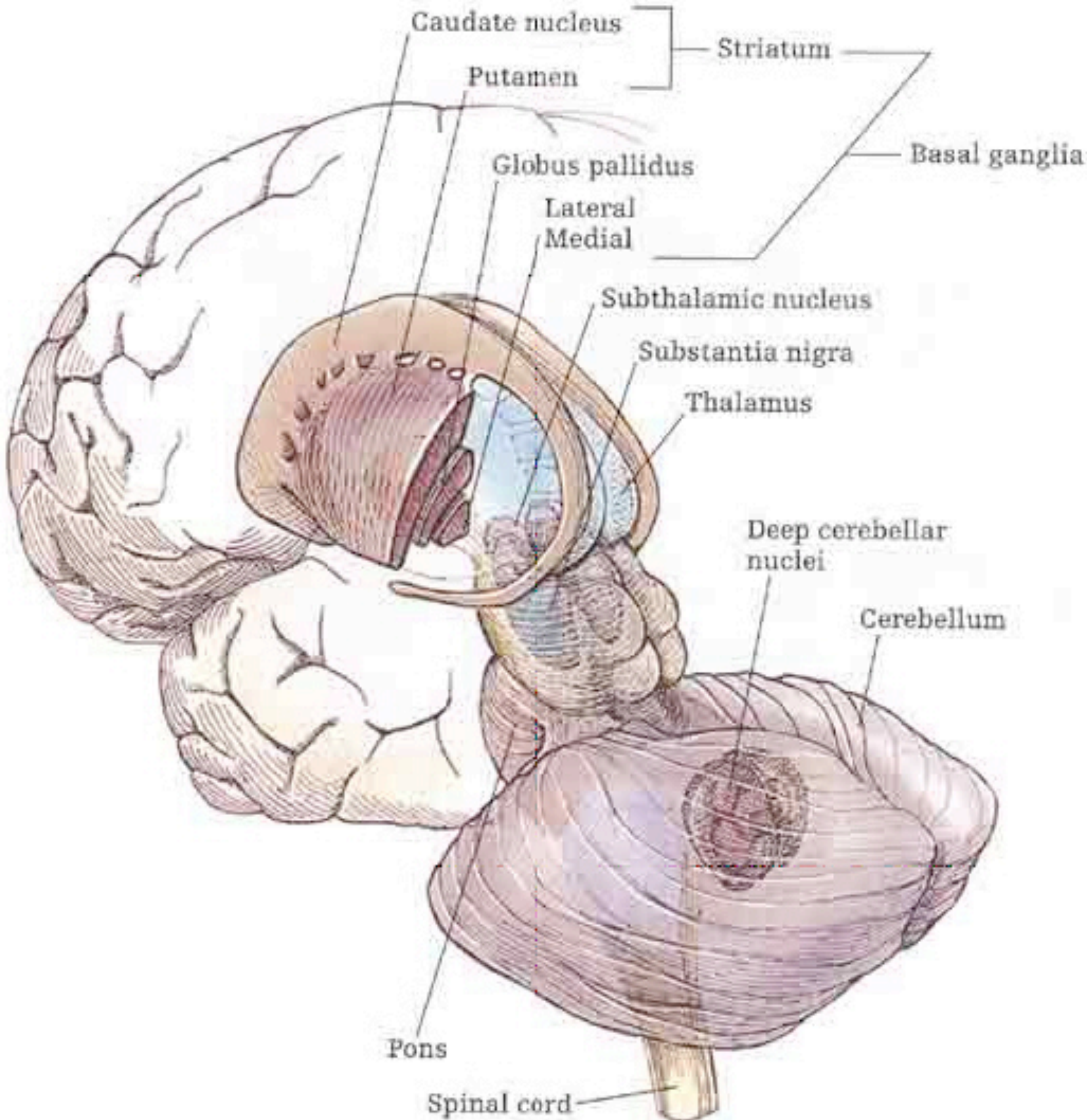
Reward Signals in the Brain



Stimulus B Stimulus A Reward

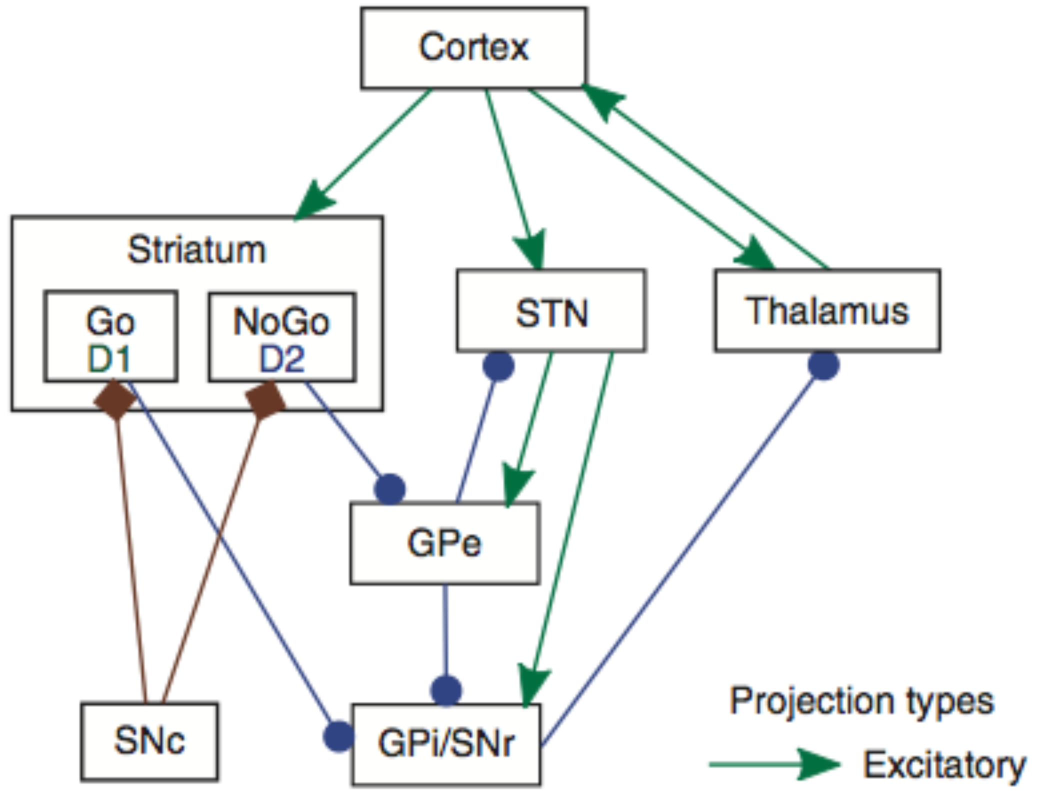
Wolfram Schultz

Stimulus A No reward


Caudate nucleus
Putamen
Striatum
Basal ganglia
Globus pallidus
Lateral
Medial
Subthalamic nucleus
Substantia nigra
Thalamus
Deep cerebellar
nuclei
Cerebellum
Pons
Spinal cord

Disorders with effects
On dopamine system:

Parkinson's disease
Tourett's syndrome
ADHD
Drug addiction
Schizophrenia



Maia & Frank 2011

Conclusion and Outlook

Three basic categories of learning:

Supervised: Lots of progress through statistical learning theory
Kernel machines, graphical models, etc

Unsupervised: Hot research area with some progress,
deep temporal learning

Reinforcement: Important topic in animal behavior,
model-based RL