Python Implementation for KalmanTD: A Unifying Probabilistic View of Associative Learning

Anish Thankachan, Diksha Yadav, Shubhamkar B. Ayare

April 21, 2022

Contents

1 MOTIVATION

The most basic form of learning, through associations between stimulus and reward, can be observed across many species and happens to be one of the most important tools of behavioral modifications. **Rescorla-Wagner model** has been an important development to understand how learning is affected by every presentation of stimulus-reward. While the model predicts two important principles of associative learning – reward prediction errors drive learning and simultaneously presented stimuli summate to predict reward – it however ignores uncertainty about weights (or associative strengths) which have been shown to play an important role in human learning. This issue can be resolved by the **Kalman filter**, which postulates a Bayesian view of learning and incorporates uncertainty into the learning paradigm. Kalman filter still cannot resolve a fundamental issue with the Rescorla-Wagner Model: It is a trial-level model, making predictions only at the level of each trial and thereby ignoring the intra-trial structure such as stimulus duration and inter-stimulus interval, deemphasizing on long-term learning and prediction. On the contrary, the **Temporal Difference models** of associative learning focus on long term reward prediction and learning. TD models perform wonderfully in conditions where Rescorla-Wagner cannot: accounting for effect of intra-trial phenomenon, but it lacks the uncertainty tracking

mechanism of the Kalman Filter. Samuel J. Gershman provided a 'Unifying View of Associative Learning' (Gershman, 2015) through **Kalman Temporal Difference** (Geist and Pietquin, 2014), by integrating various aspects of different associative models.

In this project, we provide a python implementation for KalmanTD, along with code for running various paradigms mentioned in (Gershman, 2015).

2 KALMAN TEMPORAL DIFFERENCE

The code for KalmanTD found in `KalmanTD.py` is given below. This consists of the main class `KalmanTD` and functions/methods `predict_reward` and `update`. A function `default_kalman_td` is provided as a helper function.

```
# Implementation based on Gershman (2015)
#   "A Unifying Probabilistic View of Associative Learning"
# and Algorithm 1 from Geist and Pietquin (2010)
#   "Kalman Temporal Differences"

from numpy import *

# Keep methods separate from classes with Julia and Common Lisp style.
# Aids rapid prototyping while using emacs integrated REPL, since we
# do not need to reconstructKalmanTD instances.
class KalmanTD:
    # We are assuming num_reward_features = 1, otherwise calculation
    # for the Kalman Gain in the update function below do not seem to
    # make sense in terms of matrix dimensions
    def __init__(self, num_stimulus_features, tau, gamma, sigma_w, sigma_r):
        """
        tau     : (tau^2 * I) is added to the weights_cov at every time step
        gamma   : used to compute discounted_time_derivative
                  stimulus_features_old - gamma * stimulus_features_new
        sigma_w : initially, weights_cov = sigma_w^2 * identity_matrix
        sigma_r : actual_reward ~ normal(expected_noise, sigma_r)
        """
        self.weights_mean = zeros((num_stimulus_features, 1))
        self.num_weights  = num_stimulus_features
        self.weights_cov  = sigma_w**2 * identity(self.num_weights)
        self.tau          = tau
        self.weights_cov_noise = tau**2 * identity(self.num_weights)
        self.gamma = gamma
        self.sigma_w      = sigma_w
        self.sigma_r      = sigma_r
        self.old_kalman_gain  = None

def _compute_discounted_time_derivative(
        ktd:KalmanTD, stimulus_features_new, stimulus_features_old
):
```

```
    # Ht = Xt-Gamma*(Xt+1) : Correction Paper
    return stimulus_features_old - ktd.gamma * stimulus_features_new

def predict_reward(ktd:KalmanTD, stimulus_features_new, stimulus_features_old):
    # First compute discounted time derivative (h_t in the paper)
    discounted_time_derivative = _compute_discounted_time_derivative(
        ktd, stimulus_features_new, stimulus_features_old
    )

    # Then, check for correct dimensionality
    Ns1, _ = ktd.weights_mean.shape
    Ns2, _ = discounted_time_derivative.shape
    assert Ns1 == Ns2, \
        "Weights and stimulus are of incompatible shape: " + \
        "weights {0}, stimulus {1}".format(
            ktd.weights_mean.shape, discounted_time_derivative.shape
        )

    # Update the mean and covariance
    ktd.weights_mean = ktd.weights_mean # mean remains the same
    ktd.weights_cov  += ktd.weights_cov_noise

    # Algorithm 1 from Geist and Pietquin suggests that the predicted reward
    # is the expected reward over the weights
    return (ktd.weights_mean.T @ discounted_time_derivative).flat[0]

def update(
        ktd:KalmanTD,
        expected_reward,
        actual_reward,
        stimulus_features_new,
        stimulus_features_old
):
    """
    Update the weights, given the actual reward. We are taking
    expected_reward as an argument, because the weights_cov change in
    the mere process of calculating it.
    """

    discounted_time_derivative = _compute_discounted_time_derivative(
        ktd, stimulus_features_new, stimulus_features_old
    )
    delta = actual_reward - expected_reward

    # numer: vector of dimensions (num_stimulus_features, 1)
    numer = ktd.weights_cov @ discounted_time_derivative
    # denom: scalar
    denom = (discounted_time_derivative.T @ ktd.weights_cov \
             @ discounted_time_derivative).flat[0] \
```

```
            + ktd.sigma_r**2
    kalman_gain = numer / denom

    ktd.weights_mean += kalman_gain * delta
    ktd.weights_cov  -= kalman_gain @ discounted_time_derivative.T @ ktd.weights_cov
    # current kalman gain can only be computed after a round of predict
    ktd.old_kalman_gain = kalman_gain

    return

def default_kalman_td(num_stimulus_features):
    return KalmanTD(
        num_stimulus_features,
        tau=0.1,
        gamma=0.98,
        sigma_r=1,
        sigma_w=1
    )
```

3 SIMULATIONS

All the simulations below assume the following header:

```
from KalmanTD import *
import numpy as np
import matplotlib.pyplot as plt
```

The parameters used for all the simulations are from Gershman, 2015 and are given below:

$$\sigma_w^2 = 1, \sigma_r^2 = 1, \tau^2 = 0.01, \gamma = 0.98$$

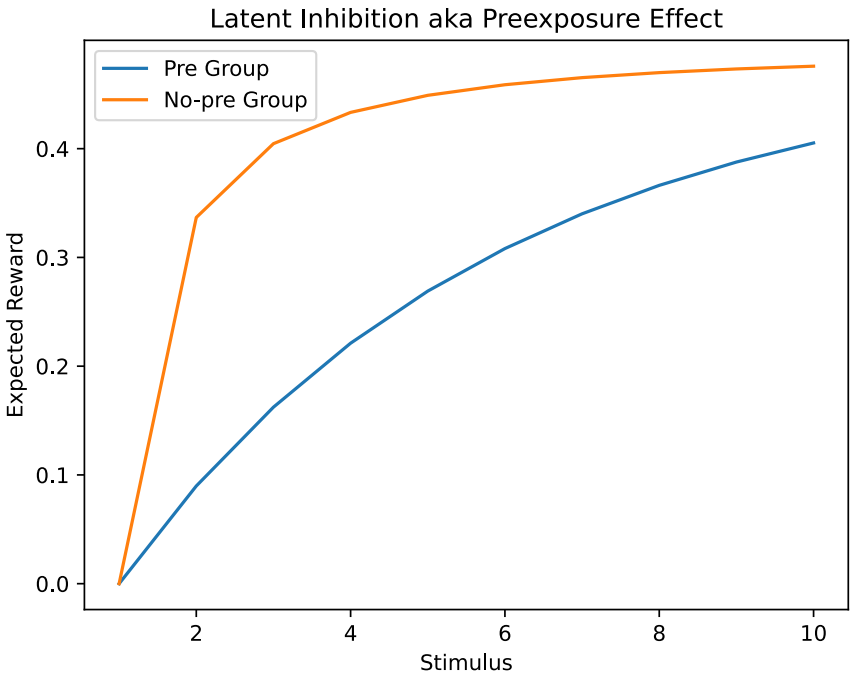
In the following subsections, we provide a brief description of each of the paradigms, along with the associated plots. The code corresponding to each of the simulations has been provided in `simulations.ipynb`.

The basic structure of simulations is to create a 3D array representing stimuli, and a 2D array representing the rewards. The first dimension for each of these is the time axis, thus each 2D sub-array along the outer dimension denotes a stimuli at a particular time step. This sub-array is actually a one-hot-encoded column vector, with first dimension denoting the various stimuli present in the paradigm, and the second dimension equal to 1. In a similar manner, the second dimension for the reward array is equal to 1.

Once the stimuli and rewards are created, one alternately calls `predict_reward` and `update` with the appropriate `KalmanTD` instance, tracking various variables (variance, covariance, kalman gain) one needs to track, and optionally presenting the test stimuli to obtain the expected reward.

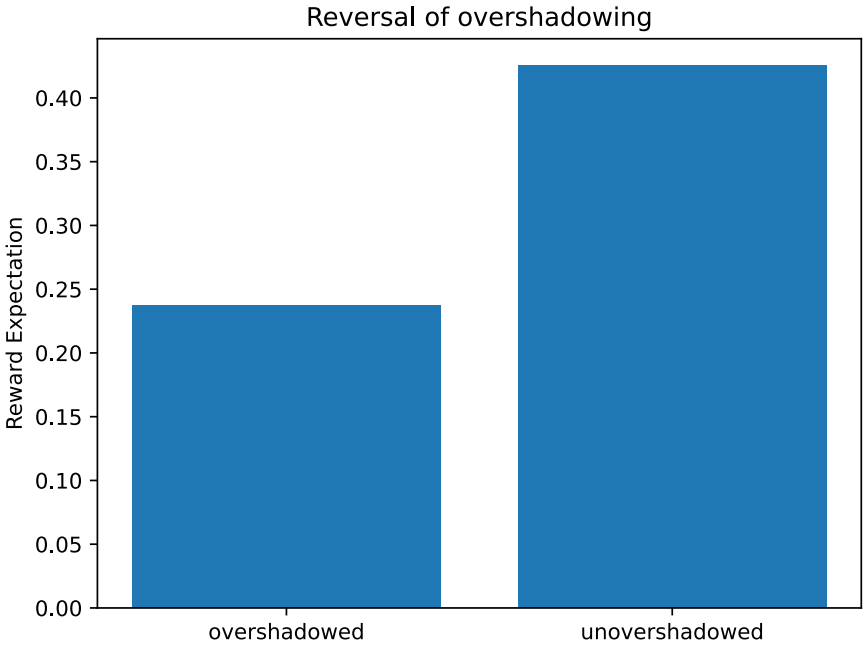
3.1 Latent Inhibition

This involves repeated exposure to the stimulus aka CS in the absence of the US aka reward, and results in a decrease in the learning rate when later the CS is presented in the absence of US.



3.2 Overshadowing

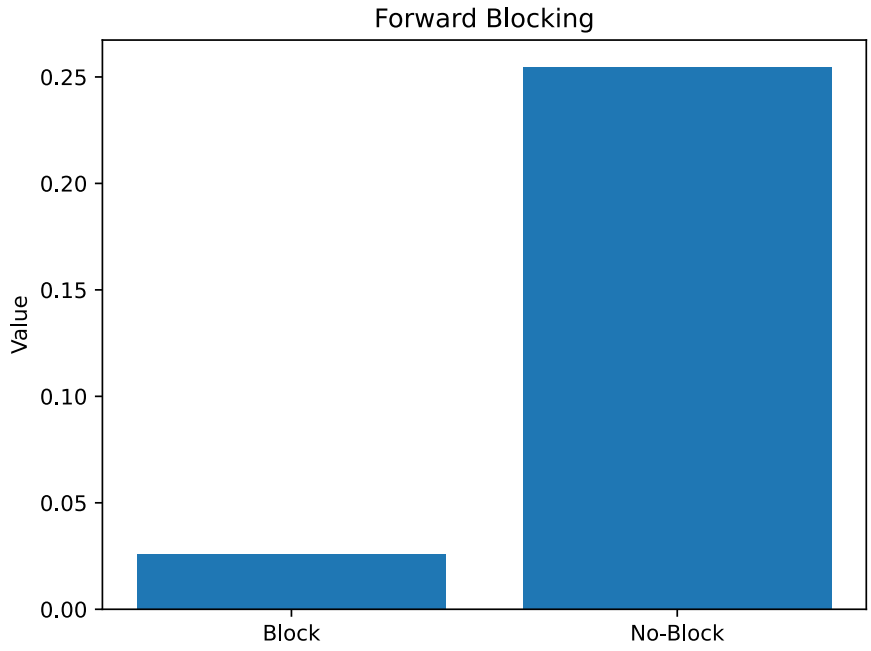
This involves two conditions: overshadowing and unovershadowing. In both conditions, the agent is exposed to 10 conditioning trials (AB -> +), while in the unovershadowing condition, the agent is additionally exposed to 10 extinction trials (A -> -). This is followed by a test of responding to B for both of the conditions.



3.3 Forward Blocking

When a stimulus A is paired with reward and then in a second phase the compound stimulus AB is paired with reward, then in a subsequent test of B alone, responding is found lower compared to a condition in which the first phase is omitted. This is called forward blocking.

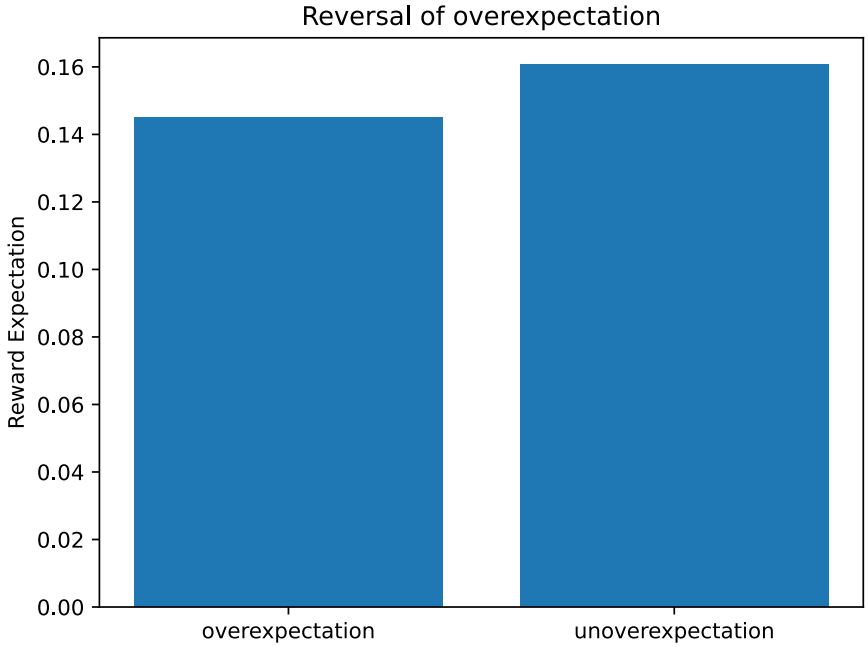
In current simulation, in the "blocking" condition, the agent was exposed to 10 conditioning trials (A -> +) followed by 10 compound conditioning trials (AB -> +) and a test of responding to B. In the "unblocking" condition, the agent was additionally exposed to 10 extinction trials (A -> -) between compound conditioning and test.



3.4 Overexpectation

This involved 10 conditioning trials for each stimulus (A -> + / B -> +). This was followed by 10 compound conditioning trials (AB -> +). The "unoverexpectation" condition involved an additional exposure to 10 extinction trials (A -> -) after the compound conditioning.

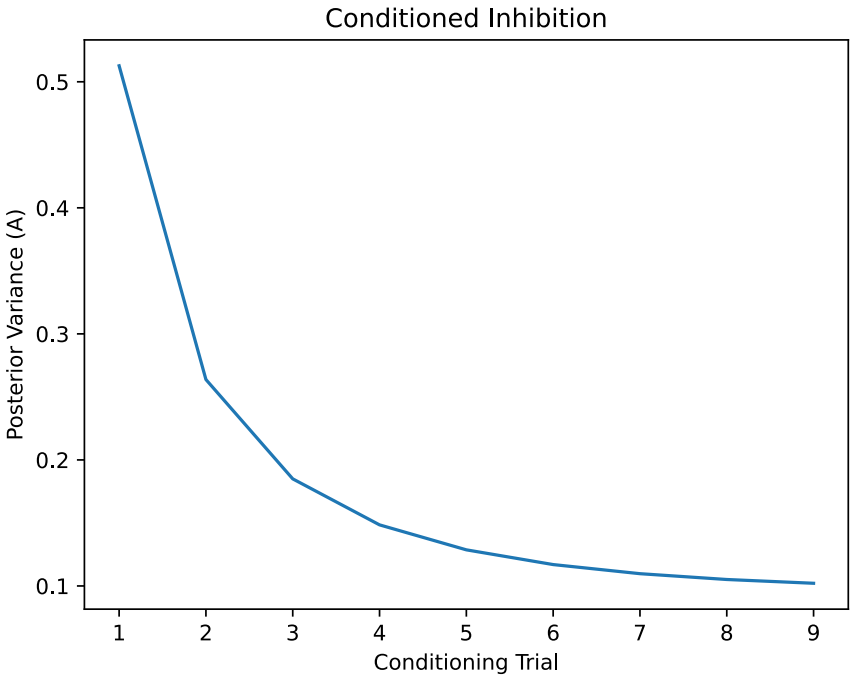
Response was tested for the stimulus B, and the KalmanTD prediction (corrborated empirically) is that the response for the unoverexpectation condition will be larger than the overexpectation condition.

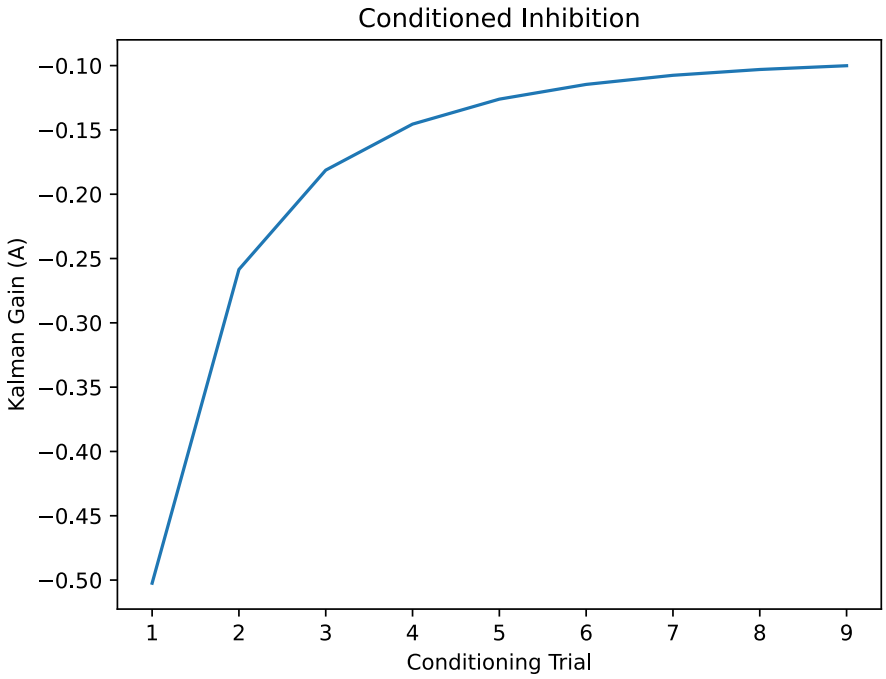


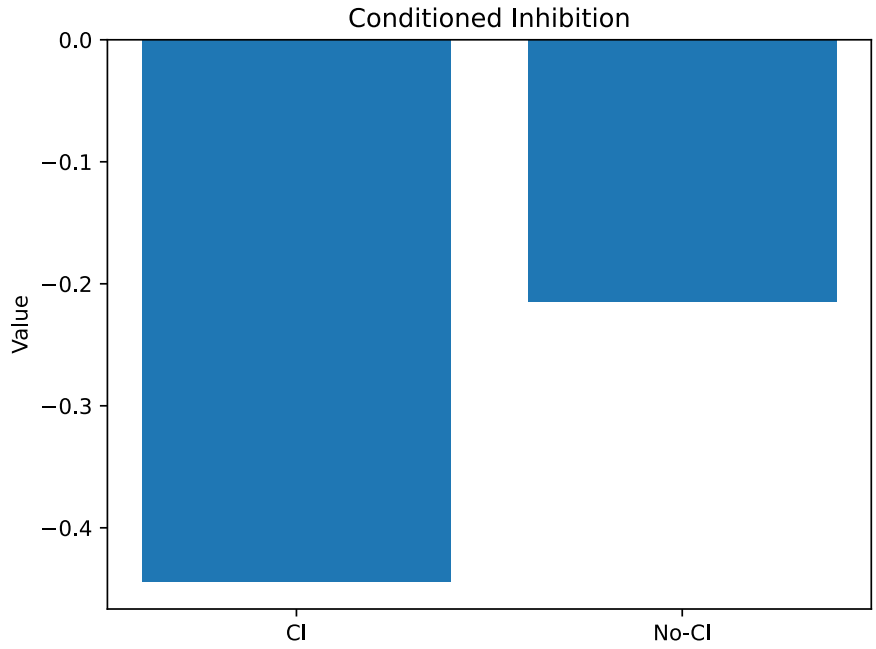
3.5 Conditioned Inhibition

To simulate negative (inhibitory) associative strength, in the conditioned inhibition paradigm, (A -> +) trials are interspersed with (AB -> -) trials, resulting in negative associative strength accruing to stimulus B

In current simulation, in the "inhibition" condition, the agent was exposed to 10 (A -> +) trials and 10 (AB -> -) trials, followed by a test of responding to B. In the "uninhibition" condition, the agent was additionally exposed to 10 extinction trials (A -> -) prior to test.

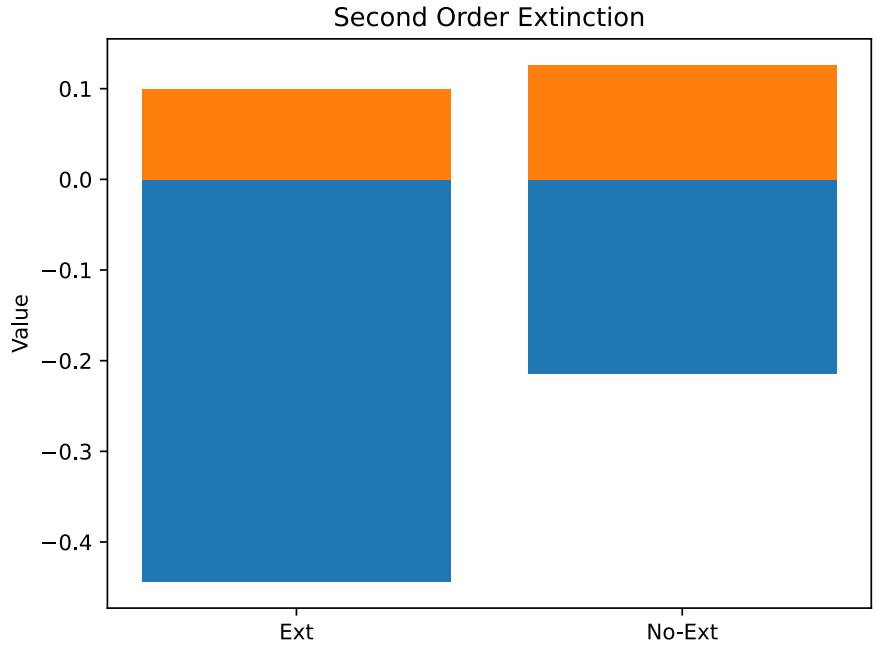



Conditioned Inhibition
Kalman Gain (A)
−0.10
−0.15
−0.20
−0.25
−0.30
−0.35
−0.40
−0.45
−0.50
1
2
3
4
5
6
7
8
9
Conditioning Trial


Conditioned Inhibition
Value
0.0
−0.1
−0.2
−0.3
−0.4
CI
No-CI

3.6 Second Order Extinction

This involves a first phase of pairing two stimuli in the absence of a US, while pairing one of the two stimuli with the US. This causes second order conditioning. In a second phase, the experimental condition involves extinction of the paired stimuli, while no such extinction trials are carried out for the control condition. The result is that extinction of the first order stimuli carries over to the second order stimuli.

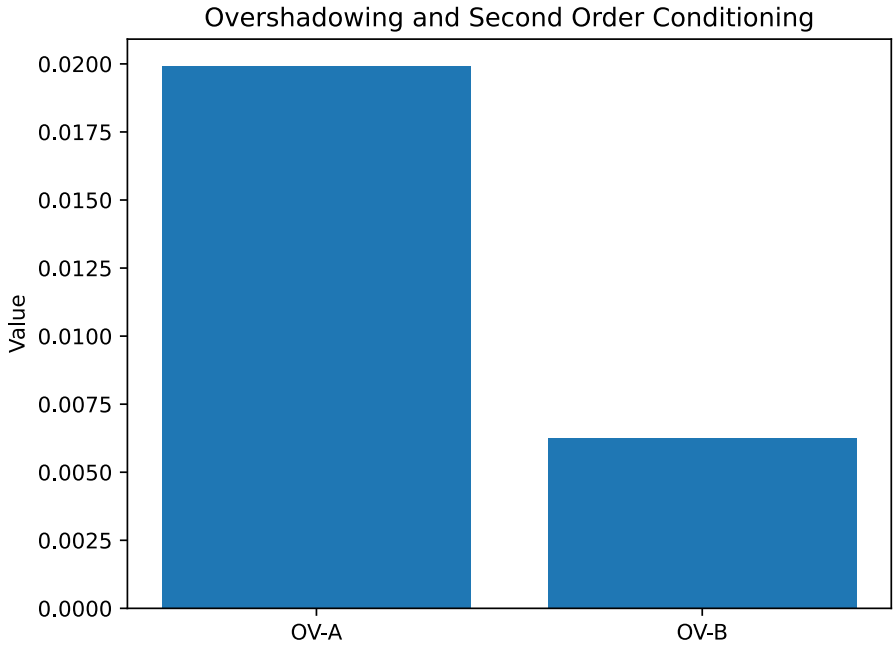


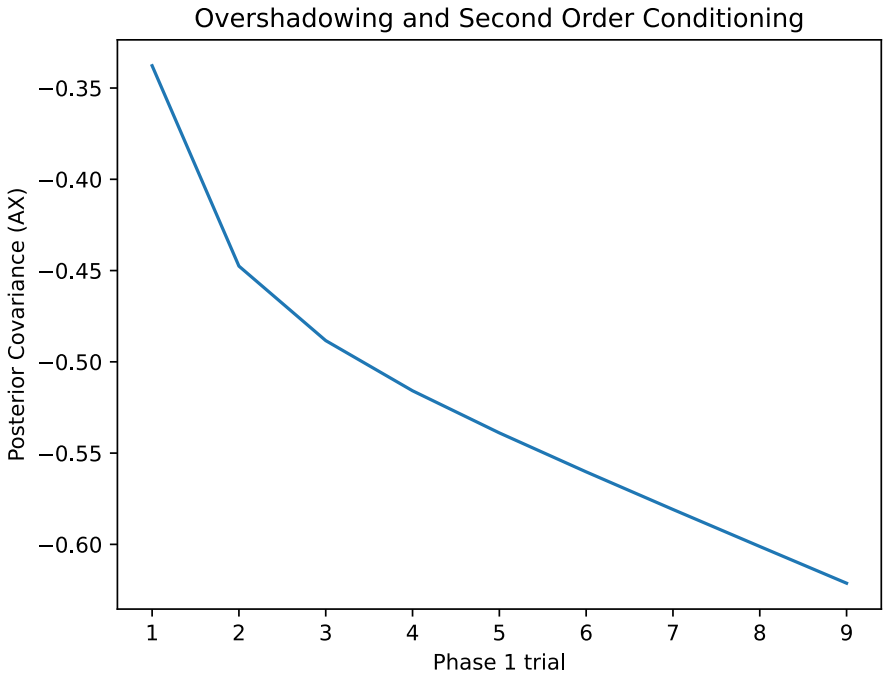
3.7 Overshadowing and second-order conditioning

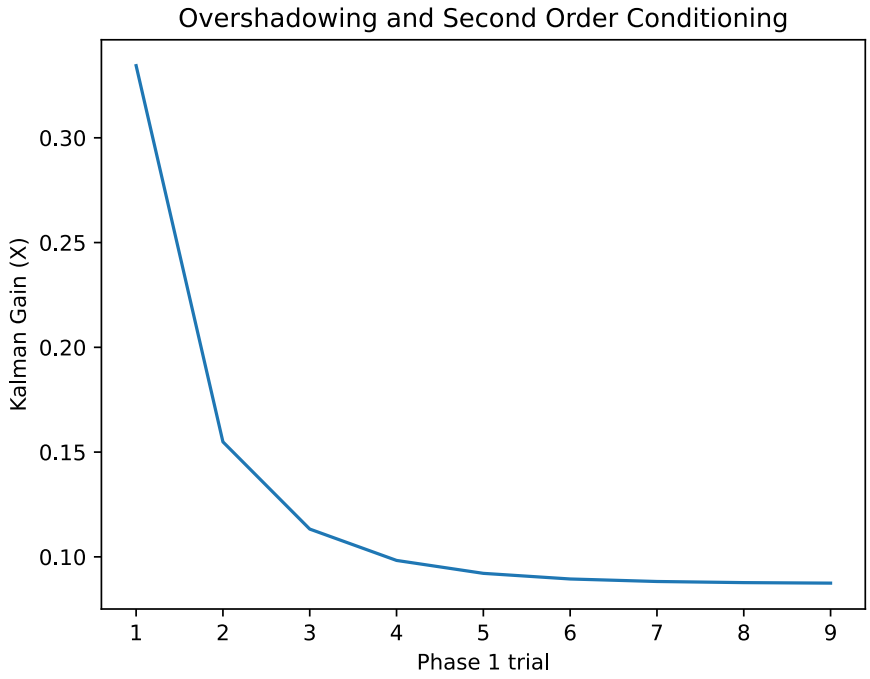
This paradigm considers a final test of responding to Z, after the following three phases:

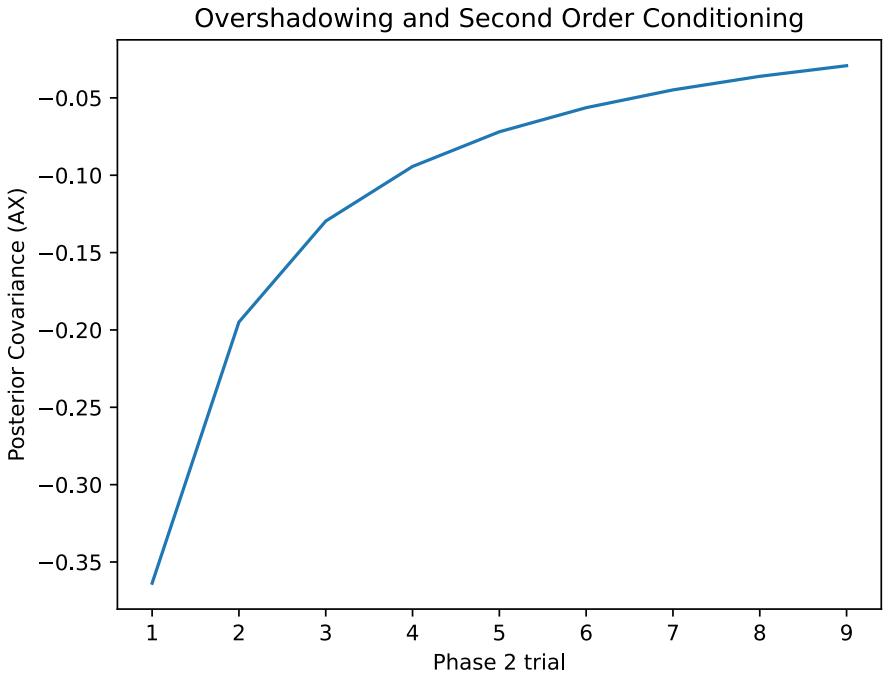
1. (AX -> + / BY -> +)
2. (A -> -) for the OV-A group, (B -> -) for the OVB group
3. (Z -> X)

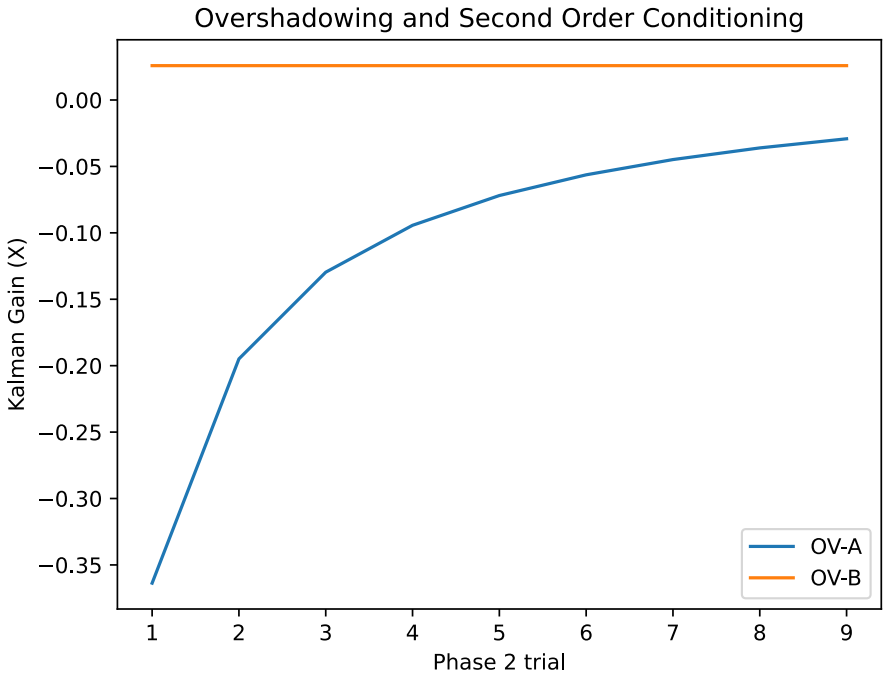
The (corroborated) expectation is that the responding as a result of the second order conditioning in the third step would be higher for the overshadowed stimulus whose counterpart was extinguished in step 2. Thus, responding would be higher for OV-A group than OV-B group.




Overshadowing and Second Order Conditioning
Posterior Covariance (AX)
−0.35
−0.40
−0.45
−0.50
−0.55
−0.60
1
2
3
4
5
6
7
8
9
Phase 1 trial

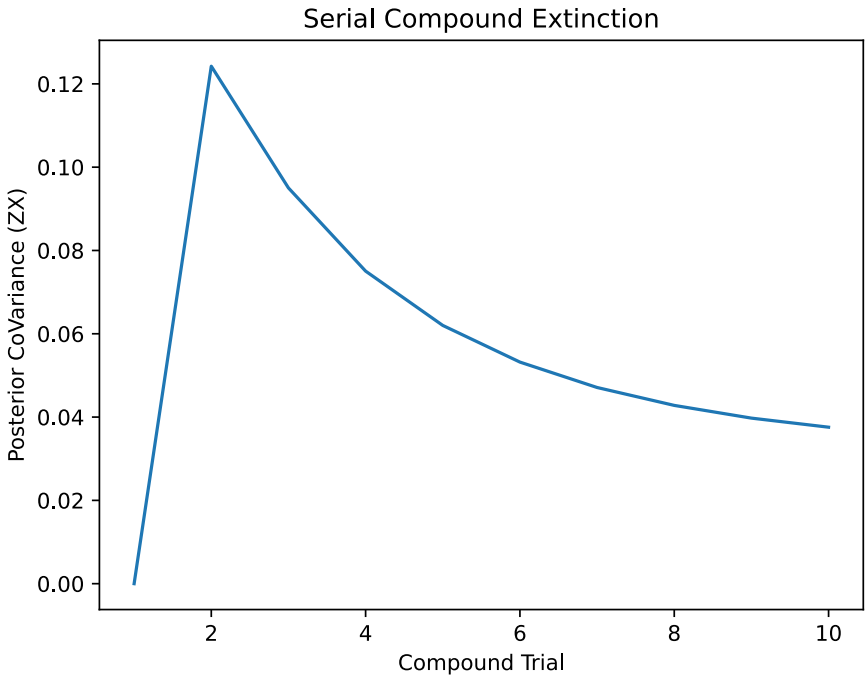

Overshadowing and Second Order Conditioning
Kalman Gain (X)
0.30
0.25
0.20
0.15
0.10
1
2
3
4
5
6
7
8
9
Phase 1 trial

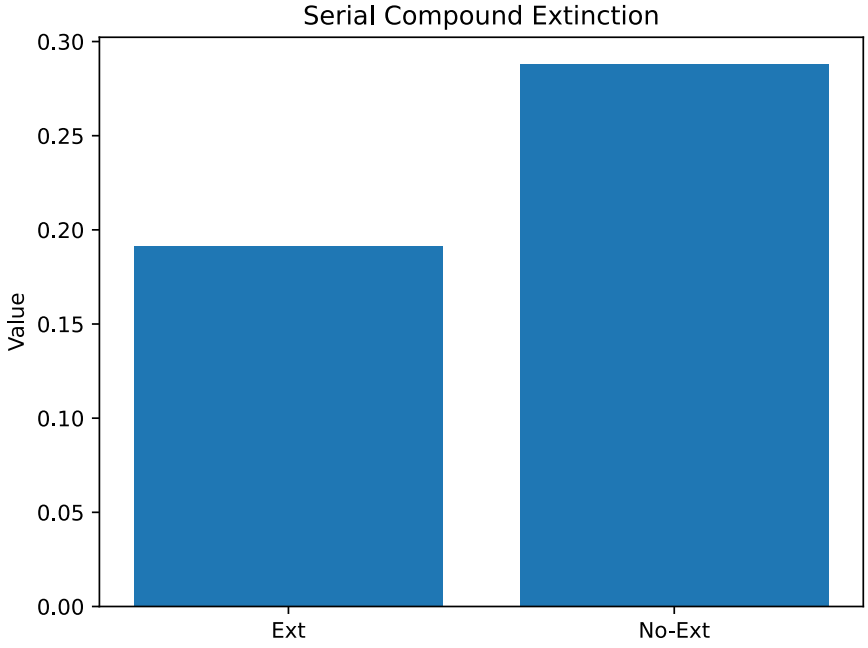

Overshadowing and Second Order Conditioning
Posterior Covariance (AX)
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
−0.35
1
2
3
4
5
6
7
8
9
Phase 2 trial


Overshadowing and Second Order Conditioning
Kalman Gain (X)
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
−0.35
OV-A
OV-B
1
2
3
4
5
6
7
8
9
Phase 2 trial

3.8 Serial compound extinction

This paradigm studies the effect of extinguishing stimulus X following serial compound training (Z->X->+). We note that this extinction treatment reduced the conditioned response to Z





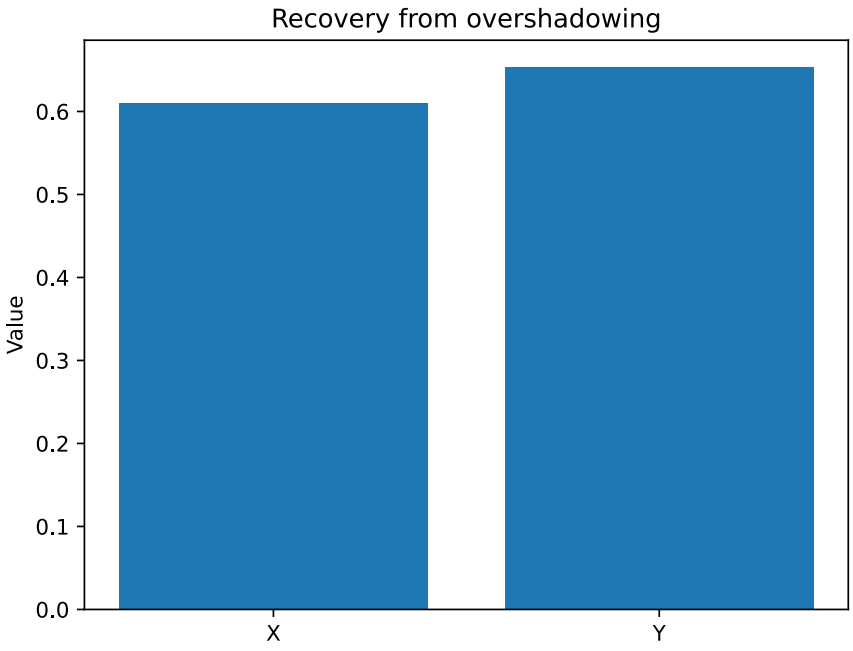
3.9 Recovery from overshadowing

This involves following a compound stimulus session by an extinction session for one component of the compound. The effect is that the responding increases for the remaining component of the compound.

Thus, this involves two phases:

1. (Z -> X -> +) for experimental and (B -> Y -> +) for control.
2. (Z -> -) for the experimental condition

(The plot given in Gershman, 2015 are actually incorrect, as can be verified from Shevill and Hall, 2004.)



4 REFERENCES

Geist, Matthieu and Olivier Pietquin (2014). "Kalman Temporal Differences". In: DOI: 10.48550/ARXIV.1406.3270. URL: https://arxiv.org/abs/1406.3270 (cit. on p. 2).

Gershman, Samuel J. (Nov. 2015). "A Unifying Probabilistic View of Associative Learning". In: *PLOS Computational Biology* 11.11. Ed. by Jörn Diedrichsen, e1004567. DOI: 10.1371/journal.pcbi.1004567. URL: https://doi.org/10.1371/journal.pcbi.1004567 (cit. on pp. 2, 4, 16).

Shevill, Ian and Geoffrey Hall (Oct. 2004). "Retrospective revaluation effects in the conditioned suppression procedure". In: *The Quarterly Journal of Experimental Psychology Section B* 57.4b, pp. 331–347. DOI: 10.1080/02724990344000178. URL: https://doi.org/10.1080/02724990344000178 (cit. on p. 16).