Multitask Learning over Shared Subspaces

Nicholas Menghi, Kemal Kacar, Will Penny*

School of Psychology, University of East Anglia, Norwich Research Park, Norwich, Norfolk, NR4 7TJ, UK

* w.penny@uea.ac.uk

Abstract

This paper uses constructs from the field of multitask machine learning to define pairs of learning tasks that either shared or did not share a common subspace. Human subjects then learnt these tasks using a feedback-based approach. We found, as hypothesised, that subject performance was significantly higher on the second task if it shared the same subspace as the first, an advantage that played out most strongly at the beginning of the second task. Additionally, accuracy was positively correlated over subjects learning same-subspace tasks but was not correlated for those learning different-subspace tasks. These results, and other aspects of learning dynamics, were compared to the behaviour of a Neural Network model trained using sequential Bayesian inference. Human performance was found to be consistent with a Soft Parameter Sharing variant of this model that constrained representations to be similar among tasks but only when this aided learning. We propose that the concept of shared subspaces provides a useful framework for the experimental study of human multitask and transfer learning.

Author summary

How does knowledge gained from previous experience affect learning of new tasks ? This question of "Transfer Learning" has been addressed by teachers, psychologists, and more recently by researchers in the fields of neural networks and machine learning. Leveraging constructs from machine learning, we designed pairs of learning tasks that either shared or did not share a common subspace. We compared the dynamics of transfer learning in humans with those of a multitask neural network model, finding that human performance was consistent with a soft parameter sharing variant of the model. Learning was boosted in the early stages of the second task if the same subspace was shared between tasks. Additionally, accuracy between tasks was positively correlated but only when they shared the same subspace. Our results highlight the roles of subspaces, showing how they could act as a learning boost if shared, and be detrimental if not.

Introduction

Recent advances in machine learning have delivered human-like levels of performance across a variety of domains from speech and image recognition [1] to language understanding [2] and game-playing [3]. These advances have been achieved, in the main, using neural network models with very large numbers (e.g. millions) of

parameters that are estimated using very large numbers (e.g. millions) of data points. The requirement for such a huge amount of training data places limits on the tasks that can be learnt and is at odds with much of the psychology literature on human learning which suggests that concepts can be learnt using very few examples. One way of achieving such "data-efficient" learning is to leverage information learnt on one task to more efficiently learn another. Two subfields of machine learning have been using this approach, one is known as Multitask Learning [4, 5] the other as Transfer Learning [6, 7]. This paper uses constructs from the machine learning literature on Multitask Learning to better understand how humans learn across multiple tasks.

Our starting point is the original Multitask Learning learning architecture proposed by Caruana et al. [4] in which generalisation across tasks is achieved using shared parameters. This architecture comprises a feature module, which can be shared across tasks, and an output module which is task-specific. In the original "hard-parameter sharing" architecture [5] the parameters defining the feature model are identical across tasks. Mathematically, this feature model defines a subspace that is shared across tasks. The idea that shared subspaces are useful for learning over multiple tasks has previously been highlighted, for example, under the term "structure learning" [8].

This paper uses an experimental design in which participants learn a pair of tasks that either do or do not share a common subspace. We investigate how learning proceeds with the hypothesis that learning will be facilitated for tasks that share a common subspace. Facilitation of learning could be manifested as faster and/or more accurate learning. We restrict ourselves to linear subspaces so that the shared features are a reduced-dimension linear projection of the input space, leaving nonlinear subspaces to subsequent experiments.

In additional modelling work we make use of a second construct from the Multitask Learning literature - that of "soft-parameter sharing" [5]. Here, a second task does not share exactly the same feature model, but parameters determining the features are constrained to be similar. We use a Bayesian estimation algorithm, similar to the Elastic Weight Consolidation approach [9], in which the prior over feature parameters for a second task is given by the posterior over feature parameters from the first. This implements a soft constraint that can be overwritten in the face of evidence to the contrary. We provide a novel contribution to the modelling literature by using a Bayesian learning approach that is implemented on mini-batches of data. This can produce learning dynamics both within and between tasks, and the model predicts facilitation of learning (or "positive transfer" [10]) in tasks that share a common subspace (perhaps to a lesser degree than hard parameter sharing) but gradual recovery from "subspace interference" (or "negative transfer" [10]) for tasks in different subspaces (whereas no such recovery would be observed for hard-parameter sharing). We compare these simulation results to empirical findings.

Overall, the paper presents a novel experimental task, empirical results on behavioural data, theoretical results from computer simulation, highlights similarities between them, and discusses ideas for future work in this area. We propose that the concept of shared subspaces provides a useful framework for the experimental study of human multitask and transfer learning.

Materials and methods

Our model-based analysis (see below) is described mathematically using the notation defined here. We use $\sigma_k()$ to denote the kth output of the softmax function

$$\sigma_k(x) = \frac{\exp(x_k)}{\sum_{i=1}^{K} \exp(x_i)} \tag{1}$$

where x is a K-element vector. We use $\mathsf{N}(x; m, C)$ to denote a multivariate Gaussian density over x with mean m and covariance C. The transpose of vector x is written x^T. 1_{RK} denotes an R-by-K matrix of ones. The delta function $\delta(a, b)$ takes the value 1 if $a = b$ and zero otherwise.

Participants

A total of ninety-six volunteers from the University of East Anglia (mean age = 19.90, SD = 1.36, 17 male) participated in the experiment. Data from seven participants became unavailable due to computer network synchronization errors. A further nine participants were discarded because they performed below chance level in both tasks. We performed our analysis on the remaining sample of 80 participants (mean age = 19.80, SD = 1.34, 13 male). All of them were naive to the purpose of the experiment. At the end of the experiment, participants received course credits for their participation. Written informed consent was obtained from all participants.

Apparatus and Stimuli

The experiment was performed in a dimly lit room with participants seated 60 cm away from a computer display with their head supported by a chin-rest. Stimuli were presented on a 23-inch HP Elite Display 240c monitor using the Psychophysics Toolbox [11] for Matlab (Mathworks) running on Windows 7.

Two virtual "pies" (1° x 1° visual angle) were displayed at 1° from the central fixation point. Each pie was divided into six slices with from one up to five slices that could be filled with red colour, making a total of twenty-five combinations. The slices of the two pies were filled in a mirrored way as shown in Fig 1. The stimuli were presented on a dark grey background.

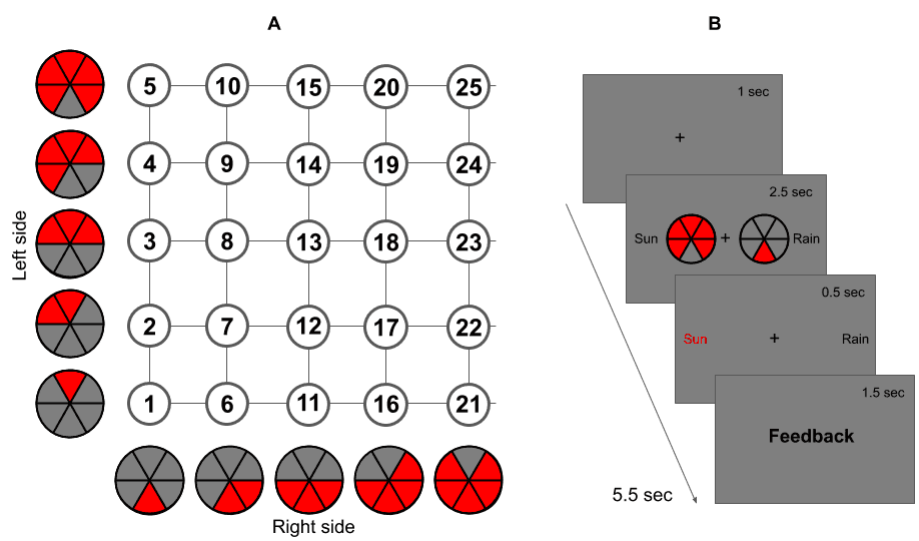

Fig 1. (A) Experimental Stimuli.(B) Trial Structure. (A) Each pie on the left/right side was combined with each pie on the right/left side, creating 25 potential configurations. (B) Each trial started with a fixation cross. Afterwards, two pies appeared and participants had up to 2.5 sec to respond. Confirmation of the choice was then given and feedback was provided.

Procedure

As we can see in Fig 1 each trial started with a black fixation cross presented at the center of the screen for an interval of 1000 ms. Afterwards, the stimuli appeared and stayed on screen for 2500 ms maximum or until a response was made. Responses were made on a standard keyboard, the "g" indicated sun/heads prediction in task1/2 and "j" indicated rain/tails in task 1/2. Responses not given within the required time constitute "missed trials". Right after the button press, confirmation of the choice was given for 500 ms. Finally, feedback was provided, saying "correct" if the prediction was correct, "incorrect" if it was not and "too slow" for a missed trial.

The experiment took about one hour to complete and was composed of two tasks, comprising 250 trials each (10 repetitions per configuration) divided into 5 blocks, each of 50 trials. For the first task subjects had to make sun/rain decisions, as in the classic Weather Prediction Task [12], and for the second task they made heads/tails decisions. The mappings from stimulus to reward (correct/incorrect) were specified as described in the following section.

At the end of each task, we probed participants knowledge. We first asked them to describe the way they approached the task. We then gave them a list of six strategies (where only one was correct) and asked them to tick the one that resembled the most the one they used. Finally, we presented them with a timeline of the task asking to mark the point in time in which they started using that strategy.

Stimulus-Reward Maps

Four different Stimulus-Reward Maps, or "Value Functions", were used over the course of the experiment (but only two per subject), as shown in Fig 2. These were generated using log-quadratic (Sub1, Add1) or log-linear (Sub2, Add2) mappings as follows. For the log-quadratic maps (Sub1, Add1), the probabilistic structure was specified by making the log-odds of the outcome a quadratic function of stimulus characteristics. Flipping the sign of a single parameter in this mapping changes the Sub1 map to the Add1 map. That is

$$\begin{aligned} \log\left[\frac{p(y_t=1)}{p(y_t=0)}\right] &= (u_t-\mu)^T W (u_t-\mu) + w_0 \qquad (2) \\ W &= 2.4 \times \begin{bmatrix} -0.71 & w_d \\ w_d & -0.71 \end{bmatrix} \\ \mu &= [3,3]^T \\ w_0 &= 4 \end{aligned}$$

where $w_d = -0.71$ produces the Sub1 map and $w_d = 0.71$ produces the Add1 map. If, for each cue, subjects choose the option with the highest probability, then the correct classification rate would be 95 per cent. This is the maximum possible for the Sub1 and Add1 tasks.

We also defined tasks using a log-linear model which can produce, for example, the Sub2 map shown in Figure 2. Although generated from different models (log-linear versus log-quadratic), from an MTL perspective this task is similar to the Sub1 task in that the relevant feature for both tasks is $x = u_2 - u_1$ ie. subtraction. The Add2 map was similarly defined. The maximum performance levels for the Sub2 and Add2 maps were both 93 per cent.

Additionally, these maps could be approximately described using the following rules: Sub1 - "Choose Sun if the difference in pie slices is zero"; Sub2 - "Choose Heads if there more are slices on the left than right"; Add1 - "Choose Sun if the sum of slices makes a full pie"; Add2 - "Choose Heads if the sum of slices is greater than six".

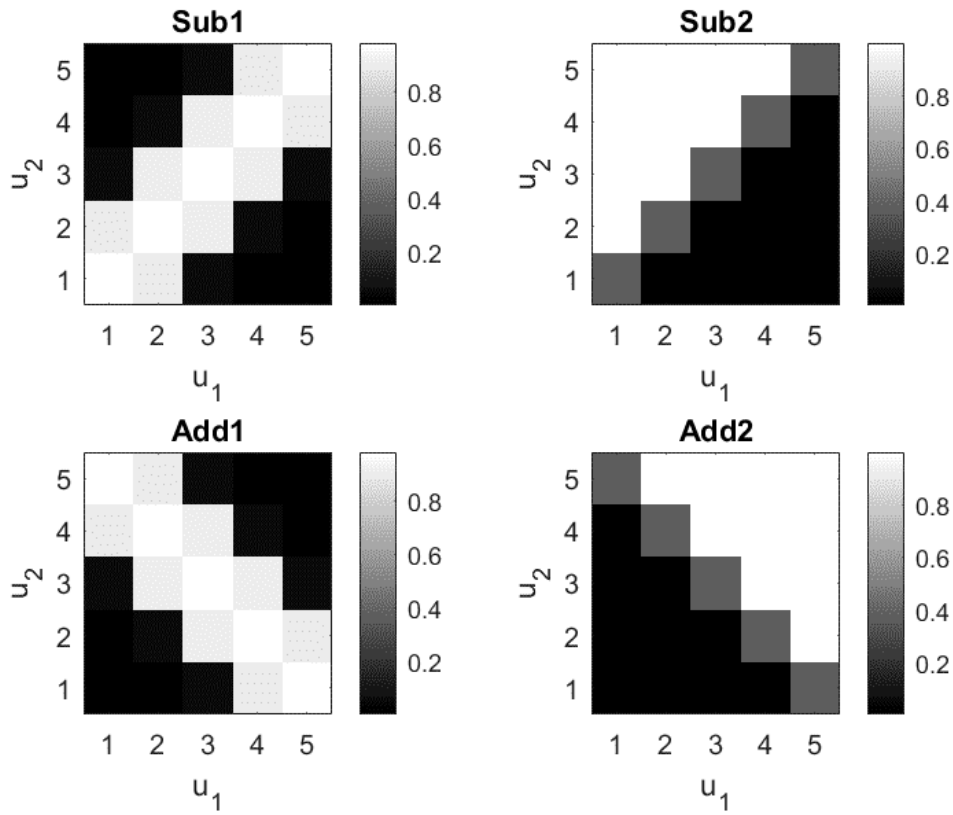

Fig 2. Stimulus-Reward Maps. Each gray scale image plots the reward probability (given button press "g") as a function of stimulus, u. The variables u_1 and u_2 denote the number of slices in the right and left pie stimuli. Each task can be implemented using two-stages of processing. For example, for the Sub1 and Sub2 maps the first stage requires extraction of a feature, $x = u_1 - u_2$. For Add1 and Add2 the required feature is $x = u_1 + u_2$. Tasks which use the same stimulus to feature space function (ie. subtraction or addition) are said to share the same subspace.

Experimental Design

Each participant did two tasks, in the first one they had to learn the association between stimuli and weather outcome (sun or rain); in the second one they had to learn the association between stimuli and a coin toss outcome (heads or tails). The stimulus to outcome mapping in task 1 was specified by either the Sub1 or Add1 map. Task 2 was specified by either the Sub2 or Add2 map. Participants were assigned to either a "Same-Subspace" (Same) or "Different-Subspace" (Diff) group according to the logic of table 1. There are 20 subjects per "condition" and 40 subjects per group.

Table 1. Subjects and Groups.

Condition	Task 1	Task 2	Subspace
1	Add1	Add2	Same
2	Sub1	Sub2	Same
3	Add1	Sub2	Diff
4	Sub1	Add2	Diff

Subjects were assigned to one of Same or Different Subspace Groups in a between-subjects design. Each of the Same/Different groups comprises data from two conditions e.g. data from the same subspace group is from both Add1-Add2 and Sub1-Sub2 conditions.

Additionally, orthogonal subgroups of participants had a minimum 12 seconds break between one learning block and another whereas another orthogonal subgroup had minimum 120 seconds break between one block and another, in a two-by-two between-subject design (with factors of subspace and break-length). However, the break-length factor is ignored in the data analyses presented in this paper.

Given that participants are required to make Sun/Rain decisions and learn incrementally via feedback, Task 1 is reminiscent of the classic Weather Prediction Task (WPT) [13, 14]. However, a major difference is that in our tasks there is a hidden structure in the stimulus-reward mappings that can be discovered by subjects. Further, Task 1 is also similar to the Configural and Elemental Learning tasks defined by Duncan et al. [15], with elemental tasks containing a hidden structure (the log-odds of an outcome being a linearly separable function of stimuli). However, the hidden structure we have specified is a linear subspace lying within a non-linear (quadratic) mapping. Task 1 also shares similarities with the Feature-based Multi-Armed Bandit (FMAB) task of Stojic et al. [16] in that the reward probability is a function of bivariate stimuli. However, FMAB uses a linear function and participants make a multi-way (rather than binary) decision on each trial.

Neural Network Model

This section describes a Neural Network model that we hope provides insight into some of the computational processes that may be engaged when solving Multitask learning problems. Learning in this model uses a Bayesian estimation algorithm, similar to the Elastic Weight Consolidation approach [9], in which the prior over feature parameters for a second task is given by the posterior over feature parameters from the first. Such a Bayesian learning approach to neural network training was first proposed by Mackay [17]. We consider two variants of this model - Hard Parameter Sharing (HPS) in which the parameters are constrained to be identical and Soft Parameter Sharing (SPS) which implements a soft constraint that can be overwritten in the face of evidence to the contrary [5]. A novel aspect of this modelling work is that we implement Bayesian learning over mini-batches of trials, allowing the model to predict learning dynamics at the time scale of tens of trials. The neural network models are exposed to exactly the same stimulus and stimulus-reward maps provided to experimental participants, and in the results section we compare simulations from these models with empirical findings.

In the machine learning literature, Multitask Learning means training a neural network simultaneously on data with multiple output labels but where the inputs are of the same type, for example, learning to detect multiple types of object from the same visual images [4, 7]. Whereas, Transfer Learning means training a network sequentially on data from task A and then task B, but only tuning the final layer or layers using data from task B [6, 7]. We have designed our neural network model to accommodate both types of learning (using a mini-batch buffer to potentially store trials from multiple tasks) although our empirical data is from a transfer task.

Value Network

The stimulus-reward maps, or "value functions" (value being expected reward), are approximated using a MultiTask Learning (MTL) neural network model

$$\begin{aligned} \pi_t^n &= \sigma(v_t^n) \\ v_t^n &= W_t^n h_t \\ h_{tp} &= \phi_p(x_t) \\ x_t &= A_t u_t \end{aligned} \tag{3}$$

where π_t^n is a $[K \times 1]$ probability vector, v_t^n is a $[K \times 1]$ latent vector, h_t is $[P \times 1]$ vector of hidden unit activations, x_t is an $[F \times 1]$ vector of "feature" units, and u_t is a $[D \times 1]$ input vector. The MTL architecture in this paper is restricted to linear feature mappings using an $F \times D$ feature matrix A_t. We make this simplification as linear subspaces allow for a reasonable variety of experimental paradigms to be modelled, including the current experiment. More generally, one would follow the machine learning literature and allow nonlinear mappings e.g. using networks of Sigmoid or Rectified Linear Units [6].

The overall network is trained, by updating A_t and W_t^n, so that π_{tk}^n approximates the expected reward, or value, obtained under action k. The overall architecture is shown in Fig 3. We refer to the mapping from u_t to x_t as the "Feature Mapping" and from x_t to π_t as the "Output Mapping". The Output Mapping contains a $[K \times P]$ weight matrix, W_t^n for task n. In Caruana's [4] original multitask learning method the feature mapping is constrained to be identical across tasks. This is now known as "Hard Parameter Sharing (HPS)" [5], and in our implementation of it we constrain A_t to be identical across tasks.

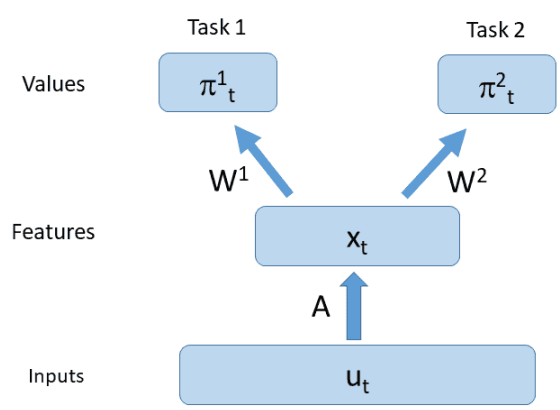

Fig 3. Hard Parameter Sharing (HPS) architecture for Value Approximation. We refer to the mapping from u_t to x_t as the "Feature Mapping" and from x_t to π_t as the "Output Mapping". Output mappings are task-specific and, for this HPS architecture, feature mappings are identical across tasks.

We define the mapping from x_t to h_t using a nonlinear transform specified using Radial Basis Functions (RBFs)

$$\phi_p(x_t) = \exp\left[-0.5\beta_p(x_t - m_p)^T(x_t - m_p)\right] \tag{4}$$

for $p = 1..(P-1)$ with basis centres m_p and precisions β_p and $\phi_P = 1$. This final ϕ_P value allows the corresponding regression coefficient in the output mapping W to contain an offset variable. In this paper, the RBF basis centres and widths are fixed throughout learning (see below). More generally, however, various strategies exist to construct and modify them during learning [18]. Overall, the output mapping (from x_t to π_t) is equivalent to a Radial Basis Function (RBF) classifier.

The Softmax mapping from v_t^n to π_t^n ensures that the output of the value network is between 0 and 1. Use of this Softmax function constrains the expected reward for $k = 1$ to be one minus the expected reward for $k = 2$, which is the case for the empirical tasks. This constraint could be relaxed, for example for FMAB tasks, by setting $\pi_{tk}^n = s(v_{tk}^n)$ where $s()$ is the logistic sigmoid function.

More generally, the Output Mapping could be implemented using other approaches instead of the RBF, for example, networks of Sigmoidal or Rectified Linear Units. One reason we have chosen it, however, is that if we set $A_t = I$, then the whole value function map reduces to an RBF, which provides a more standard approach for Value Function estimation [19]. In the results section we compare the Multitask Neural Net with this more standard online RBF approach (see below). With a large number of basis functions (equal to the number of unique stimulus configurations) the Neural Net further reduces to a Configural Reinforcement Learning algorithm (see e.g. [15]).

Decision Making

Action d on trial t for task n is taken with probability

$$\begin{aligned} q_{td}^n &\equiv p(a_t = d | s_t = n) \\ &= \sigma_d(v_t^n) \end{aligned} \tag{5}$$

where v_t^n is the value vector under task n at time t. Here we assume that the task variable $s_t = n$ is known (i.e. agent performs task s_t on trial t). That is, we have no task ambiguity. We also write the above as $q_t^n = \sigma(v_t^n)$. The agent then takes action $a_t = d$ by sampling from the multinomial distribution defined by q_t^n.

Offline Learning

We update model parameters not after each trial, but rather after a "mini-batch" or "block" of training trials. This might cover a specific interval of interest and could reflect e.g. the block structure of an experiment. In this paper, a mini-batch corresponds to a block of 50 learning trials. We define the jth block of training data, R_j, to comprise the input and task variables along with the decisions made by an agent and the rewards received. We write this as $R_j = \{r_t, a_t, s_t, u_t\}$ for $t \in \tau_j$ where τ_j is the set of all trials in the jth block. Here, the agent used to make decisions within a block is not necessarily the same agent that is being trained. This therefore allows for Off-Policy learning as described in the Reinforcement Learning literature [19].

Sequential Bayesian Inference

In this paper, once a block of training data has been used for offline learning it is then discarded. To make best use of this data we use sequential Bayesian inference so that information is efficiently propagated from one block to the next. We define Y_j to denote all blocks of data up to and including block j. That is $Y_j = \{R_1, R_2, ..., R_j\}$.

This paper considers Bayesian learning over feature parameters A only i.e. not over task parameters W. This is sufficient for demonstrating the points we wish to make in the paper but could be generalised in future work. Bayesian estimation of A proceeds over blocks such that the prior over A is updated to a posterior using Bayes rule

$$p(A|Y_j) = \frac{p(R_j|A)p(A|Y_{j-1})}{p(R_j)} \tag{6}$$

We use a Gaussian prior over A that factorises over rows

$$\begin{aligned} p(A|Y_{j-1}) &= \prod_{f=1}^{F} p(a_f|Y_{j-1}) \\ p(a_f|Y_{j-1}) &= \mathsf{N}(a_f; m_{jf}, \Lambda_{jf}^{-1}) \end{aligned} \tag{7}$$

where a_{jf}^T is the fth row of A, m_{jf} is the prior mean and Λ_{jf} the prior precision. The prior mean and precision are set equal to the posterior mean and precision from the previous episode (see equation in supplemetary material 1). For the first learning episode on the first task the prior is initialised to have a zero mean, $m_0 = 0_D$, and prior precision of $\Lambda_0 = \rho I_D$. In the experiments below we set $\rho = 1$.

The Neural Network is then trained on a mini-batch of data to find the Maximum A Posteriori (MAP) solution using the MTL-Offline algorithm as described in Appendix A. Generically, a factorised Laplace approximation (with factorisation over rows of A as in the prior) is then used to compute a Gaussian approximation to the posterior density (the simulations in this paper, however, only consider A matrices with a single row). We then use a Sequential Bayesian Learning approach in which this posterior becomes the prior for the next learning episode.

If we were working with linear Gaussian models then Sequential Bayesian Learning over J mini-batches would be exactly equivalent to Bayesian learning from a single batch (comprising all exemplars) [20]. However, as we are using a Laplace approximation in a nonlinear model, its an empirical matter as to whether this procedure works well.

Model Likelihood And Joint

Let r_t be a Bernoulli reward signal received after taking action $a_t = k$. The model-based estimated reward probability is then

$$\begin{aligned} \pi_{tk}^n &\equiv p(r_t = 1|a_t = k) \\ &= \sigma_k(v_t^n) \end{aligned} \tag{8}$$

The log likelihood of reward is

$$L_t = r_t \log \pi_{tk}^n + (1 - r_t) \log(1 - \pi_{tk}^n) \tag{9}$$

where $n = s_t$ is the selected task and $k = a_t$ is the selected action. We also refer to this quantity as the sample likelihood as it is based on a single data sample.

The MTL-Offline algorithm defined in this paper (see Appendix A) learns from a data set stored in a memory buffer. This buffer contains all inputs observed, task variables specified, decisions made and rewards received over a given set of trials, $R_j = \{u_t, s_t, a_t, r_t\}$. The overall likelihood to be maximised in MTL-Offline is

$$\begin{aligned} L^j(A, W) &\equiv \log p(R_j|A) \\ &= \sum_{t \in \tau_j} L_t \end{aligned} \tag{10}$$

We can then define the log joint density as

$$\begin{aligned} J(A,W) &= L^j(A,W) + \log p(A|Y_j) \\ \log p(A|Y_j) &= -\frac{FD}{2}\log 2\pi + \frac{1}{2}\sum \log|\Lambda_{jf}| - \frac{1}{2}\sum_f (a_f - m_{jf})^T \Lambda_{jf}(a_f - m_{jf}) \end{aligned} \tag{11}$$

with $p(A)$ being the prior over feature parameters (see below). Bayesian learning from data set R_j can then proceed by ascending the gradient of the log joint to reach a local maximum of the posterior density. Inclusion of the prior term ensures that parameter estimates are constrained to be similar to values found useful for previous blocks of data.

This can be made more explicit by inspecting the equation for the gradient in the MTL-Offline algorithm (see Appendix A for definition of all terms)

$$\begin{aligned} \frac{dJ}{dA} &= \sum_{t\in\tau}(r_t - \pi_{td})G_t^n + \Psi_j \\ \Psi_{jf} &= -\Lambda_{jf}(a_f - m_{jf}) \end{aligned} \tag{12}$$

where the fth row of Ψ_j is given by Ψ_{jf}^T. This drives the weights for feature f towards the prior mean for that episode, m_{jf}. Importantly, the prior precision Λ_{jf} controls the strength of this effect, and this quantity increases in proportion to the number of data samples so far observed (in sequential Bayesian Learning for linear Gaussian models the posterior precision equals the prior precision plus the data precision and therefore always increases - see equation in Appendix A). This leads to the desirable property that the feature matrix converges to a high precision solution.

Our offline training algorithm uses an adaptive step-size gradient ascent method. Additionally, because the likelihood landscape contains multiple local maxima with highly suboptimal solutions, this is embedded within a multi-start optimisation procedure (see Appendix A for full details).

Soft Parameter Sharing

A Soft Parameter Sharing (SPS) approach is shown in Fig 4. This is identical to the HPS approach except that the feature mapping is also task specific

$$x_t^n = A_t^n u_t \tag{13}$$

But the rows of A^n are constrained to be similar across tasks. For the fth row of A the prior distribution at the start of a new task is

$$p(a_f^n) = \mathsf{N}(a_f^n; m_{jf}, \Lambda_{jf}^{-1}) \tag{14}$$

where m_j and Λ_j are set equal to the posterior distribution from the previous task. By default, this allows representations learnt for one task to be used in another task thus facilitating faster and/or more accurate learning. However, if learning the new task (task n) does not go well, SPS will dilute this prior by reducing the prior precision. More specifically, if the multistart optimisation procedure does not find a solution with sufficiently high training accuracy (see Appendix A) the prior precision is updated as

$$\Lambda_{jf} = \alpha_a \Lambda_{jf} + (1-\alpha_a)\rho I_D \tag{15}$$

and another multistart optimisation is initiated. This update converges to ρI_D at a rate determined by α_a. For the results in this paper we used $\alpha_a = 0.1$

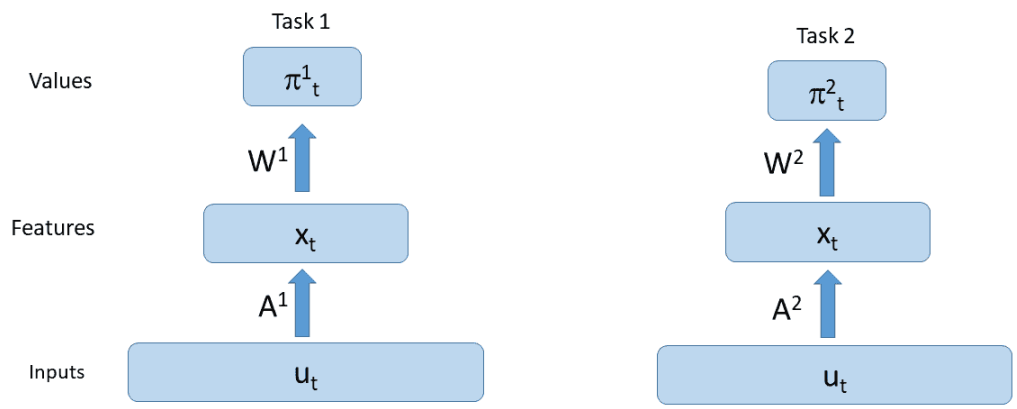

Fig 4. Soft Parameter Sharing (HPS) architecture for Value Approximation. We refer to the mapping from u_t to x_t as the "Feature Mapping" and from x_t to π_t as the "Output Mapping". Output mappings are task-specific and feature mappings are constrained to be similar across tasks by setting the prior distribution over A^2 to be the posterior from task 1, $p(A^2) = p(A^1|Task1)$. However, this prior can be diluted if learning the second task does not go well.

RBF Model

For comparison with the above approach we also implement an online learning system using just RBFs. We define online learning to be a process in which parameters of a model are updated after each trial. All trial-specific information such as the cue identity, action taken and reward received are then discarded.

We implemented RBF learning within the context of MTL by specifying the feature layer to be the identity matrix, $A = I_D$. Let $(w^n_{tk})^T$ be the kth row of matrix W^n_t. A gradient-based update rule for maximising the sample likelihood for the task layer is given by

$$w^n_{t+1,k} = w^n_{tk} + \alpha_w(r_t - \pi_{td})z_t \quad (16)$$

$$z_t = \frac{(\delta_{dk} - \pi_{tk})}{(1 - \pi_{td})} h_t \quad (17)$$

where the action taken on trial t was $a_t = d$, and the task was $s_t = n$. The quantity $(r_t - \pi_{td})$ is the Reward Prediction Error (RPE) and we use a learning rate of $\alpha_w = 0.3$. Nine basis functions were used spanning the two-dimensional input space with centres in fixed positions $([1,1],[1,3],[1,5],[3,1],[3,3],[3,5],[5,1],[5,3],[5,5])$ and precisions set to $\beta = 1.22$.

Results

In what follows, repeated-measures ANOVA p-values are corrected using the Greenhouse-Geisser method.

Empirical Results

Our main hypothesis is that learning will be facilitated for tasks that share a common subspace where facilitation of learning could be manifested as faster and/or more accurate learning. These directional hypotheses motivate the use of the one-sided p-values described below.

Effect of Subspace

Participants performed better when the second task was in the same rather than different subspace as the first (See Fig 5 right panel). A one tail, independent sample t-test revealed the effect to be significant ($t(78) = 1.955$, $p = 0.027$) (two tail, $p = 0.055$). Importantly, this is not caused by a preceding difference in performance (one tail independent measure t-test, $t(78) = 0.255$, $p = 0.400$, left panel). This result is in support of the main hypothesis of the paper - that learning is facilitated for tasks that share a common subspace, where the facilitation here manifests in higher accuracy.

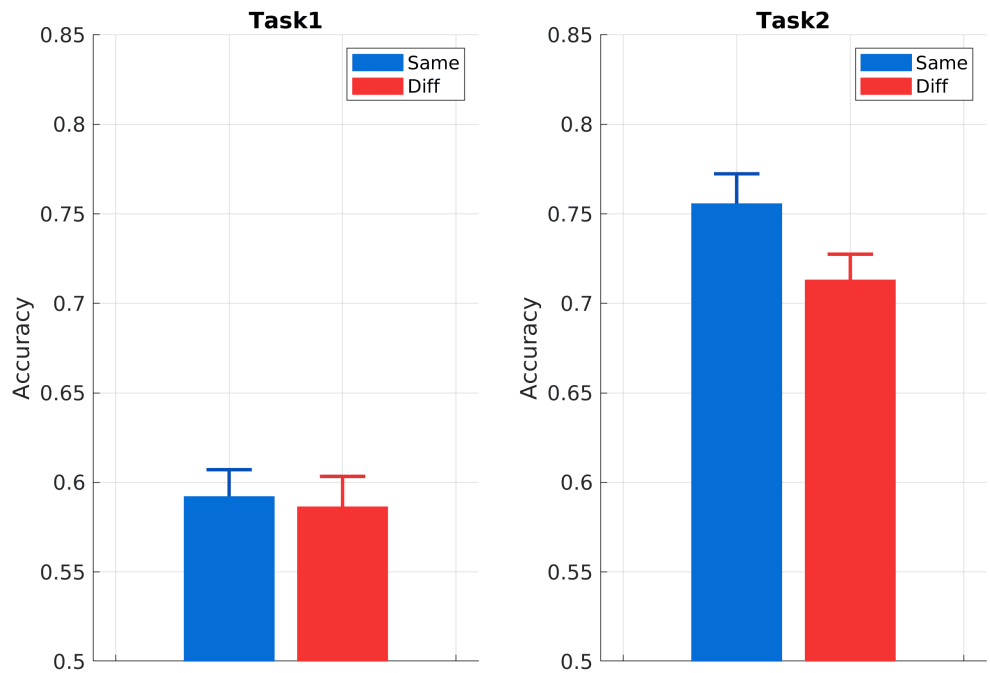

Fig 5. Effect of Subspace. The plots show average accuracy for learners in the same and different groups for task 1 (left panel) and task 2 (right panel). Subjects performed better in the second task when it shared the same subspace as the first. There was no significant difference in task 1 accuracy.

Moreover, for subjects in which tasks 1 and 2 shared the same subspace, task performance was significantly positively correlated ($r = 0.42, p = 0.007$) whereas for subjects in which the tasks were in a different subspace there was no such correlation ($r = -0.09, p = 0.591$). This is illustrated in Fig 6.

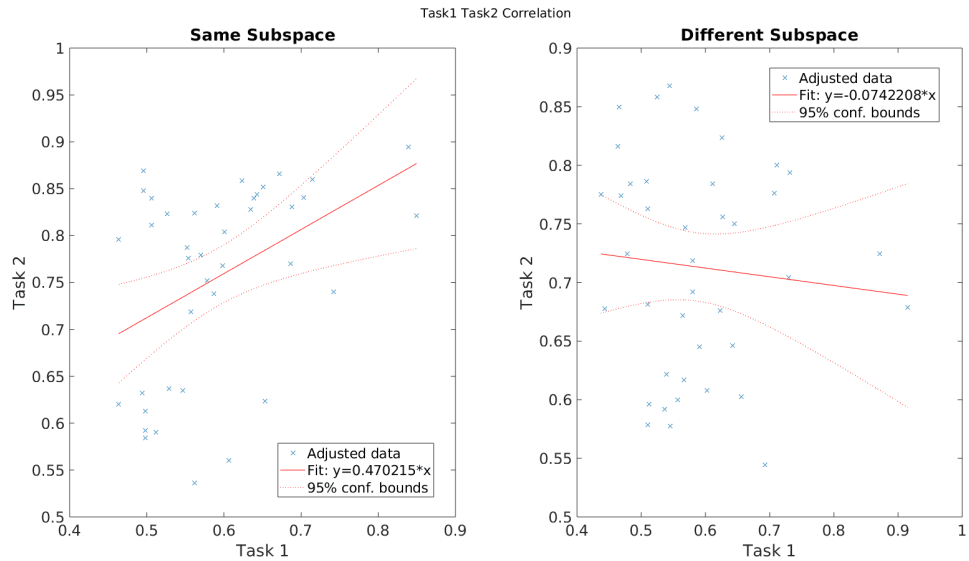

Fig 6. Correlation in Task Performance For subjects in which tasks 1 and 2 shared the same subspace (left panel), task performance was significantly positively correlated whereas for subjects in which the tasks were in a different subspace (right panel) there was no such correlation. Each cross specifies the performance level (accuracy) for an individual subject.

Subspace by Block Interaction

In order to further explore this difference we tested whether the advantage given by subspace repetition decreased over time in the second task. As illustrated in Fig 7 subspace repetition gave an advantage in the early blocks of the experiment, an advantage that decreased over blocks. We ran a two-way mixed design ANOVA with dependent variable accuracy and independent factors of (i) blocks and (ii) subspace. This revealed only the blocks main effect with two-sided p-values (Blocks, $F(4,152) = 32.055$, $p < 0.001$; Subspace, $F(1,38) = 3.873$, $p = 0.053$; interaction, $F(4) = 2.105$, $p = 0.080$). However, a one-sided test for the main effect of subspace, that is testing for an increase in accuracy for same versus different subspace gives $p = 0.027$, which is the same result as above (effect of subspace). Additionally, a one-sided test for the negative interaction, that tests for reductions in the benefit over same versus different subspace learners over time, gives $p = 0.04$. This result is again in support of the main hypothesis of the paper - that learning is facilitated for tasks that share a common subspace, where the facilitation here manifests as faster learning.

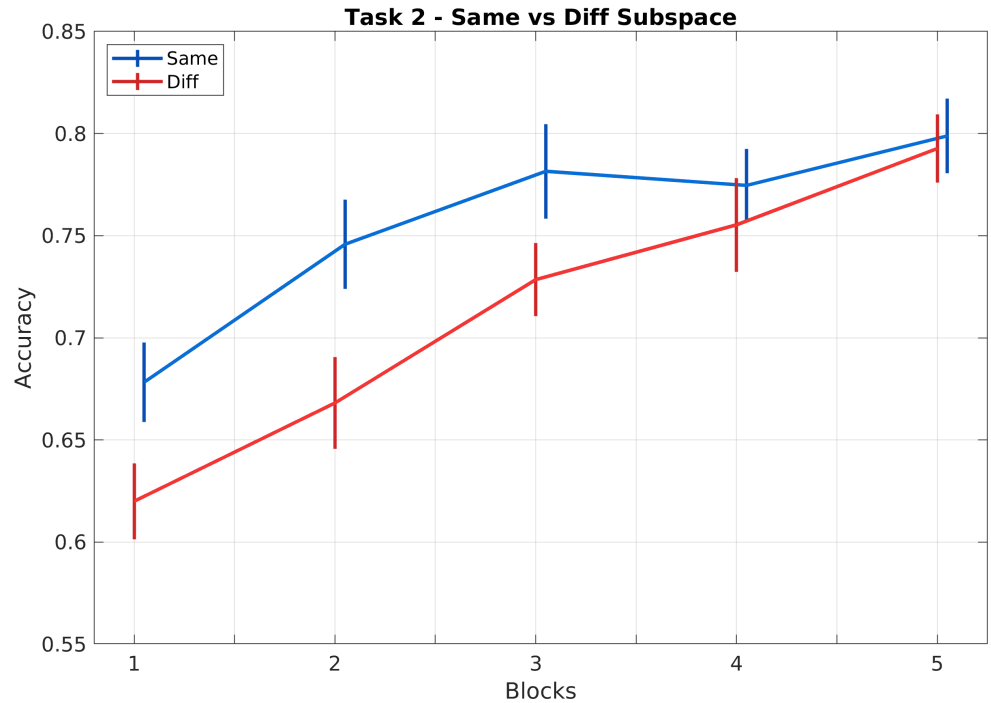

Fig 7. Subspace by Block Interaction The accuracy advantage enjoyed by learners in the same subspace group was significantly larger for early versus late blocks of task 2. This shows that learning is faster for tasks that share a common subspace.

Effect of Learner

We further reasoned that the advantage given by the subspace repetition should be greater for participants who performed better in the first task and hence had a better knowledge of the task. We therefore split participants using a median split that were in the same-subspace group into two sub-groups based on their performance in the last block of task 1: good and bad performers. We then analyzed their performance in task 2.

As shown in Fig 8 left panel, accuracy in task 2 was higher for good versus bad task1-performers. A one tail independent sample t-test revealed the effect to be significant ($t(38) = 2.676, p = 0.005$). Importantly, we repeated the same procedure for participants who were in the different-subspace group, here showing no significant effect of Task-1 Performance (one tail independent measure t-test, $t(38) = -0.065$, $p = 0.526$). Together these results are in support of the main hypothesis of the paper - that learning is facilitated for tasks that share a common subspace.

To check if there was an interaction between performers and subspace, we perfomed a two-way mixed design ANOVA with dependent variable accuracy and independent factors of (i) performer and (ii) subspace. This revealed performer and subspace main effects (Performers, F(1,38) = 4.808, $p = 0.034$; Subpsace, F(1,38) = 4.817, $p = 0.034$; interaction, F(1,38) = 3.178, $p = 0.083$). A one-sided test for the interaction between subspace and performers, testing for an benefit in accuracy for good performers in the same subspace versus different subspace, would give $p = 0.041$.

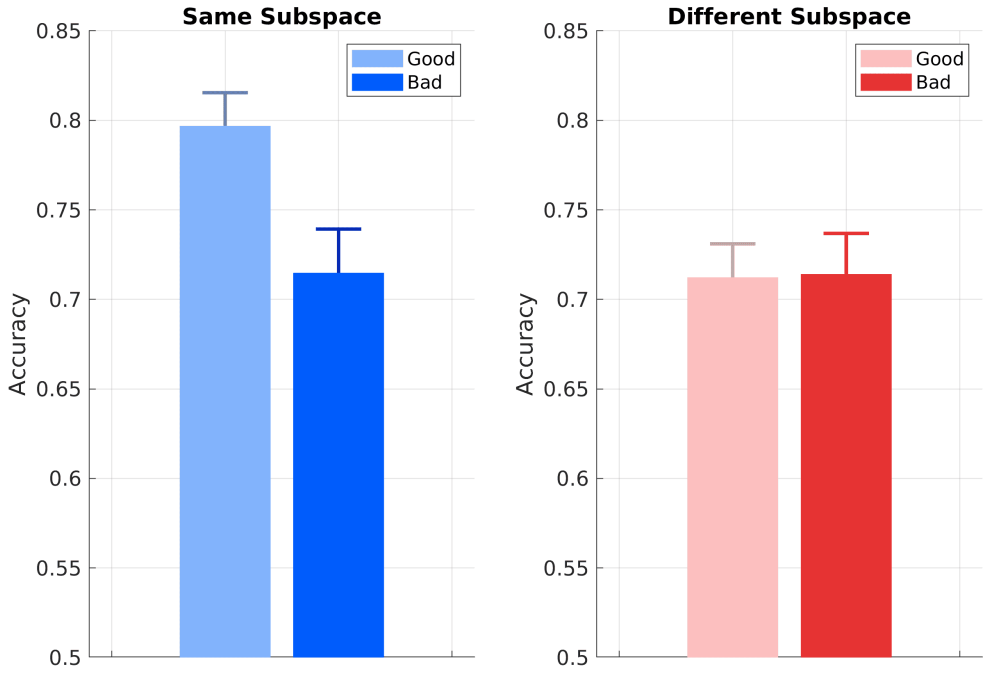

Fig 8. Effect of Learner Accuracy in task 2 was significantly higher for good versus bad learners on task 1, but only for subjects in the same-subspace group. The plots show average accuracy in task 2 for same-subspace (left panel) and different-subspace (right panel) groups.

Same-Subspace Learner by Block Interaction

In order to further explore this result we investigated if and how this advantage played out over time during the second task. As we can see in Fig 9 the advantage appears to increase overtime. A two-way mixed design ANOVA with dependent variable accuracy, and independent factors of (i) blocks and (i) performance revealed both main effects and the interaction (Blocks, $F(4,72) = 13.829$, $p < 0.001$; Performance, $F(1,18) = 7.153$, $p = 0.010$; Interaction, $F(4,72) = 3.100$, $p = 0.017$). The significant interaction means that the enhancement effect depends on block, with the figure showing the enhancement is not yet apparent in the first block of task 2.

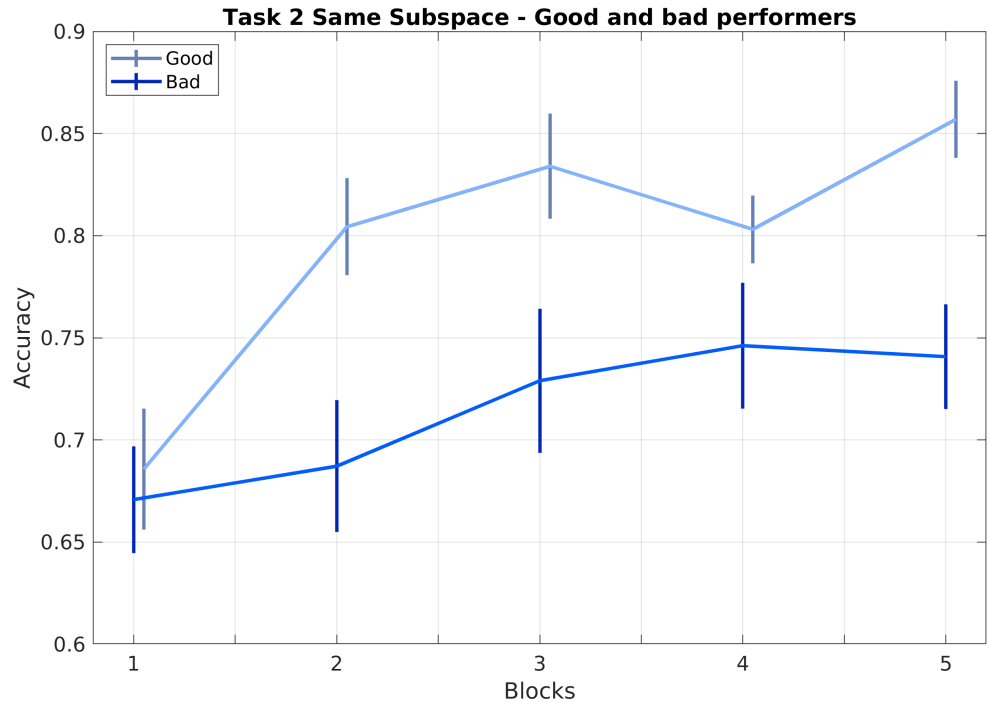

Fig 9. Same-Subspace Learner by Block Interaction. The plot shows average accuracy in task 2 for subjects in the Same-Subspace Group. Performance is enhanced for good versus bad learners on task 1, in blocks 2 to 5.

In the Appendix B, we repeat this analysis but for the different-subspace group, finding an interference effect for good versus bad learners that reduces over time, but only at a marginal level of significance.

Neural Network Simulations

Here we report simulation results for Neural Net models by specifying Task 1 as the Sub1 map, and Task 2 as either the Sub2 or Add2 maps, thus covering the same-subspace and different-subspace conditions. We ran 20 simulations of each type giving N=40 simulations in all. Results are qualitatively similar if we specify Task 1 as the Add1 map. The same software was used to implement both HPS and SPS approaches, with the prior precisions over A adapted in SPS as described above, but fixed to arbitrarily large values for HPS (thus constraining $A^2 = A^1$). In both approaches the task parameter W^2 are updated when learning task 2. In what follows, the accuracy of neural net decisions is assessed using the "average reward", which is computed as follows. First, we compute the log likelihood of obtaining reward given the neural networks decision (using equation 9), and average this over the known true reward probability for that stimulus (see above section on Stimulus-Reward Maps). We then average again over all trials in the block and exponentiate.

Fig 10 plots task accuracy (average reward) time series over 5 blocks of task 1 and 5 blocks of task 2 for task 1 being "Sub1" and task 2 being "Sub2" ie. the second task is in the same subspace as the first. Fig 11 plots the same thing but for when task 2 was "Add2" ie. the second task is in a different subspace. Both SPS and HPS do well on the same subspace task, better than the RBF net, as they make use of the representation

learnt during the first task. Only SPS does well on the different subspace task. HPS performs poorly as it uses the wrong representation. SPS is able to change the representation it uses by diluting the prior. RBF performs at the same lower level (though it outperforms SPS initially). Importantly, further analysis of SPS simulated data shows that it produces a similar pattern of statistical test results as previously described for the empirical data.

First, we ran a two-way mixed model ANOVA with dependent variable being task2 accuracy and independent factors of (i) subspace and (ii) blocks (repeated measure). This revealed a main effect of subspace with accuracy being significantly higher for the same subspace condition (mean same = 0.742, mean difference = 0.684, F(1,38)=11.99, p=0.001). It also revealed a main effect of block (F(4,152)=148.2, $p < 0.001$) and an interaction between subspace and block (F(4,152)=18.68, $p < 0.001$). For equivalent effects in human data, see "Effect of Subspace" and "Subspace by Block Interaction" sections under "Empirical Results" above. As with the empirical data, this interaction was driven by better same-subspace performance on early blocks and equivalent performance on later blocks (see also Figs 10 and 11).

Second, we identified learners as being either "good" or "bad" based on a median split of accuracy in the last block of the first task (as we did with the empirical data). We then ran a two-way mixed model ANOVA for the same subspace simulations with dependent variable being task2 accuracy and indepenent factors of (i) learner (good/bad) and (ii) blocks (repeated measure). This revealed a main effect of learner with performance accuracy being significantly higher higher for good versus bad learners (mean good = 0.774, mean bad = 0.711, F(1,18)=30.6, $p < 0.001$). It also revealed a main effect of block (F(4,72)=86.3, $p < 0.001$) and an interaction between learner and block (F(4,72)=3.66, $p = 0.022$). For equivalent effects in empirical data, see "Effect of Learner" and "Same-Subspace Learner by Block Interaction" sections under "Empirical Results" above. The pattern underlying this interaction was similar to the empirical data with good learners showing higher accuracy except on the first block. We then repeated this analysis but with data from the different subspace simulation. This revealed a main effect of learner with performance accuracy being significantly worse for good versus bad learners (mean good = 0.653, mean bad = 0.714, F(1,18)=5.70, $p = 0.028$). It also revealed a main effect of block (F(4,72)=91.2, $p < 10^{-7}$). But the interaction between learner and block was not significant (F(4,72)=1.95, $p = 0.15$ - all of the p-values we quote are two-tailed, so even a one-sided test would not be significant.) For equivalent effects in empirical data (which are only marginally significant), see Appendix B (section "Different-Subspace Learner by Block Interaction").

Third, we tested for correlations between performance in task 1 and task 2. We found a significantly positive correlation for same-subspace data ($r = 0.80, p < 0.001$) and a significantly negative correlation for different-subspace data ($r = -0.47, p = 0.037$). See Appendix B for equivalent effects in human data.

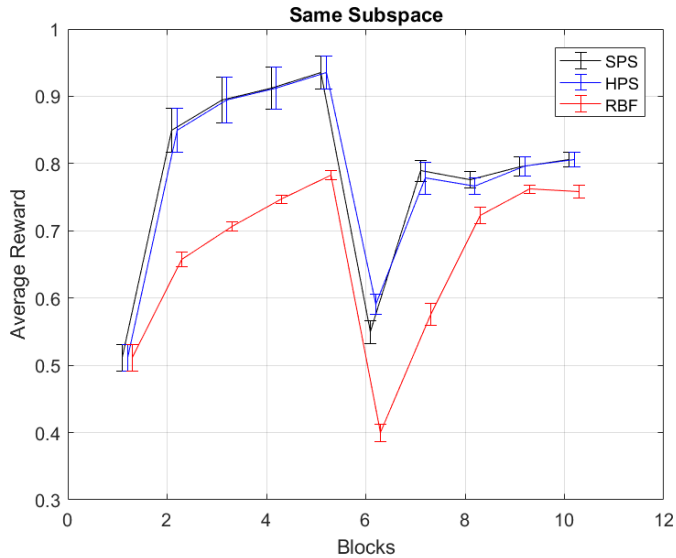

Fig 10. Same Subspace time series plot. Blocks 1 to 5 are for task 1 and blocks 6 to 10 for task 2. There are 50 trials in each block. The legend indicates the performance of SPS (Soft Parameter Sharing), HPS (Hard Parameter Sharing) and RBF (Radial Basis Function) neural network models. The lines and error bars depict means and standard errors of the means computed over 20 simulation runs.

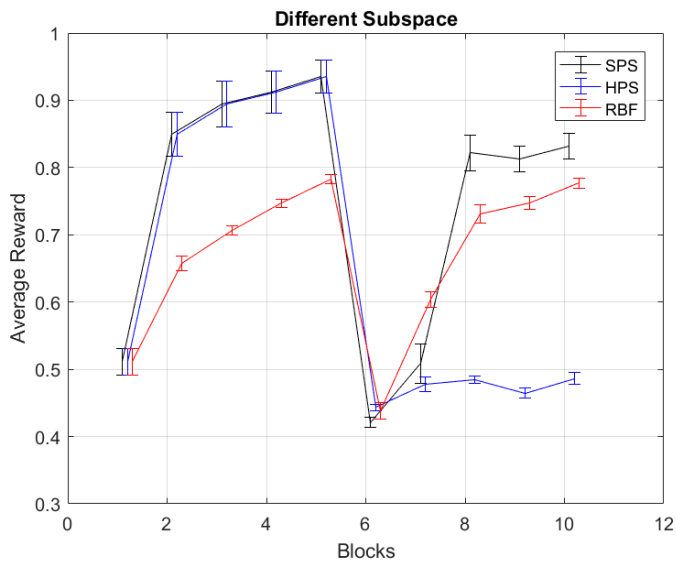

Fig 11. Different Subspace time series plot. Blocks 1 to 5 are for task 1 and blocks 6 to 10 for task 2. There are 50 trials in each block. The legend indicates the performance of SPS (Soft Parameter Sharing), HPS (Hard Parameter Sharing) and RBF (Radial Basis Function) neural network models. The lines and error bars depict means and standard errors of the means computed over 20 simulation runs.

Appendix A contains similar statistical analyses for the HPS and RBF models. For the HPS model there was a main effect of subspace, and an interaction between

subspace and block. For the same-subspace subjects, there was an effect of learner and an interaction between learner and block. But for different-subspace subjects there was no effect of learner, nor interaction with block. The missing learner effect (reflected by the poor learning in task 2 shown in Fig 11) is what differentiates HPS from SPS. For the RBF model, there is no effect of subspace nor interaction with block, and no effects of learner (or interactions between learner and block) for either same or different subspace data.

Discussion

We found evidence in support of our main hypothesis that learning would be facilitated (positive transfer) for tasks that share a common subspace. This was apparent from a number of statistical tests. First, in accuracy being higher on the second task if it shared the same subspace as the first ('effect of subspace'). Second, in accuracy being higher on the second task as a function of accuracy on the first, but only if tasks shared the same subspace ('correlation between tasks'). Further, the accuracy advantage enjoyed by same-subspace learners was more evident in early rather than late blocks of the second task ('subspace by block interaction').

Similar significant effects were found after categorising subjects as good versus bad learners depending on their performance in the last block of the first task - good versus bad same-subspace learners were better on the second task ('effect of learner'), an advantage that only appeared after the first block of the second task ('same-subspace learner by block interaction'). Conversely, good versus bad different-subspace learners were worse at task two to begin with (negative transfer) but better by the end ('different subspace learner by block interaction') - although this latter effect was only marginally significant.

We found a similar pattern of test results from simulations of the Soft Parameter Sharing Neural Net model. The similarities were in the main effect of subspace, correlation between tasks, subspace by block interaction, effect of learner and learner by block interactions. Results for the Hard Parameter Sharing variant were similar but with the important exception that there was no learner effect for different-subspace data. This clearly makes sense as the HPS model was unable to learn task two if the underlying map contained a different subspace. For the RBF model there was no effect of subspace nor interaction with block, and no effects of learner. Overall, these results show the SPS neural net model to be consistent with human behaviour, and the HPS and RBF models to be inconsistent.

Subspaces or Features ?

The rhetoric in this paper is perhaps slightly overblown in relation to the empirical work. Yes, we have found a performance advantage if the second task is in the same subspace as the first, but this subspace is defined by a single linear feature (difference or sum of number of pie slices). To infer that it is a subspace that is important we really need to set up tasks with multivariate subspaces and, to be fully generic, nonlinear multivariate subspaces. The challenge then is to define tasks with these more general properties that can be learnt by participants in the brief time they have available for psychology experiments.

Rule-based Learning

Subjects who performed the Sub1 and Sub2 or Add1 and Add2 tasks did better (than those who performed Sub1 and Add2 or Add1 and Sub2), but was this really because

the tasks were in the "same subspace". Are there not other similarities among these tasks ? For example, that both required the same logical operation or rule-based operation as an intermediate step ? This speaks to a body of work in rule-based learning. One approach to this topic is the "Rational-Rules (RR)" model [21] which formalizes a statistical learner that operates over the space of Boolean propositional logic expressions e.g. "A or B", "A and B", "A or (B and C)". In an fMRI study, Ballard et al. [22] found that the pattern of striatal responses was more consistent with prediction errors derived from a such rule-learning model than a Reinforcement Learning model. We accept therefore that there could be an ambiguity in interpretation here and that resolution of this issue requires further empirical work, perhaps with experiments using nonlinear and/or multivariate subspaces that are not readily expressible using rational rules.

Declarative Learning

In additional statistical analysis presented in the Appendix B we show that subjects who were able to declare a correct rule-based strategy also showed a stronger subspace effect. We also show, however, that subjects who performed better in the first task also showed a stronger subspace effect. Further analysis then showed these two effects to be collinear (as those who declared a correct rule-based strategy also performed better on the first task). Therefore, with the current data, we are unable to infer which of these factors (declarative learning or accurate learning) drives the subspace effect. Again, further experiments are required perhaps using nonlinear and/or multivariate subspaces.

Creation or Selection of Representations ?

Are new representations created i.e. features learnt ? Or, are pre-existing representations prioritized as potentially useful and selected from, as proposed by Collins and Koechlin [23]. For example, there may be representations in brain regions encoding for numerosity [24]that already encode differences and sums over numbers of items. An additional component in the model proposed in [23] is a process that creates new stimulus-response mappings from old ones. It could be that the offline learning algorithm we have described in this paper, or some similar process, plays this creative role.

Transfer of Learning

The study of transfer learning has a long history in psychology [10], and more recently in the fields of cognitive training and cognitive neuroscience [25]. A key qualitative concept here is the notion of near versus far transfer where distance reflects how similar the different learning contexts are. This may naturally map onto the probabilistic and quantitative measures defined under Soft Parameter Sharing e.g. the probability density of task-two feature parameters under the task-one prior. Noack et al. [26] propose a theory-driven approach to studying transfer effects in cognitive training research. They argue that data should be analysed within the context of theoretically motivated (using hierachical cognitive process models) and/or latent factor analysis methods, so that inferences can be made at the level of latent processes. The work in this paper concurs with this latent and hierarchical perspective, but whereas Noack et al. deconstruct existing batteries of cognitive tasks, our goal is to design new tasks with better defined relationships among latent and observed variables.

Machine Learning

The starting point of this paper was to leverage recent conceptual and algorithmic progress in machine learning to define new experimental psychology tasks and computational models, with the longer term goal of better understanding human multitask and transfer learning. To do this we made use of the concepts of Hard and Soft Parameter Sharing. This literature, however, is rich with other quantitative ideas about how to define relationships among tasks which could inform the design of future experiments. These include, for example, "sluice" and "cross-stitch" networks [5]which automatically infer how to share subspaces at multiple hierarchical levels and across multiple tasks.

Acknowledgments

We thank Francesco Silvestrin for piloting the experiment and John Spencer for his insights for the discussion.

References

1. LeCun Y, Bengio Y, Hinton G. Deep Learning. Nature. 2015;521:436–444.
2. Radford A, Narasimham K, Salimans T, Sutskever I. Improving Language Understanding by Generative Pre-Training. Open AI; 2018.
3. Silver D, Hassabis D. Mastering the game of Go with deep neural networks and tree search. Nature. 2016;529:484–489.
4. Caruana R. Multitask learning. Autonomous agents and multi-agent systems. 1998;27:95–133.
5. Ruder S. An overview of multitask learning in deep neural networks. BioArchive; 2017.
6. Goodfellow I, Bengio Y, Courville A. Deep Learning. MIT Press; 2016.
7. Ng A. Deep Learning Specialization. https://www.coursera.org/specializations; 2017.
8. Braun D, Mehring C, Wolpert D. Structure Learning in Action. Behavioural Brain Research. 2010;206:157–165.
9. Kirkpatrick J. Overcoming catastrophic forgetting in neural networks. PNAS. 2018;114:3521–3526.
10. Perkins D, Salomon G. Transfer of Learning. In: International Encylopedia of Education. Pergamon Press; 1992.
11. Brainard DH. The psychophysics toolbox. Spatial Vision. 1997;10:433–436.
12. Knowlton J B, Squire R L, Gluck A M. Probabilistic Classification Learning in Amnesia. Learning & Memory. 1994;1:106–120.
13. Knowlton B, Squire L, Gluck M. Probabilistic Category Learning in Amnesia. Learning and Memory. 1994;1:106–120.
14. Poldrack A R, Paré-Blagoev A R, Shohamy D, Moyano C J, Myers C, Gluck Al M. Interactive memory system in the human brain. Nature. 2001;414:546–550.

15. Duncan K, Doll B, Daw N, Shohamy D. More Than the Sum of Its parts: A role for the Hippocampus in Configural Reinforcement Learning. Neuron. 2018;98:1–13.

16. Stojic H, Schulz E, Analytis P, Speekenbrink M. It's new, but is it good? How generalization and uncertainty guide the exploration of novel options. Journal of Experimental Psychology. 2020;In Press.

17. Mackay DJC. A practical Bayesian framework for backprop networks. Neural Computation. 1992;4(3):448–472.

18. Pinto R, Engel P. A fast incremental Gaussian mixture model. PLoS One. 2015;10:e0139931.

19. Sutton R, Barto A. Reinforcement Learning: An Introduction. Cambridge: MIT Press; 1998.

20. Bishop CM. Pattern Recognition and Machine Learning. New York: Springer; 2006.

21. Goodman N, Tenenbaum J, Feldman J, Griffiths T. A rational analysis of rule-based concept learning. Cognitive Science. 2018;32:108–154.

22. Ballard I, Miller M E, Piantadosi T S, Goodman D N. Beyond Reward Prediction Errors: Human Striatum Updates Rule Values During Learning. Cerebral Cortex. 2018;28:3965–3975.

23. Collins A, Koechlin E. Reasoning, Learning, and Creativity: Frontal Lobe Function and Human Decision-Making. PLoS Biology. 2012;10:e1001293.

24. Harvey B, Dumoulin S. A network of topographic numerosity maps in human association cortex. Nature Human Behaviour. 2017;1:0036.

25. Schubert T, Strobach T, Karbach J. New directions in cognitive training: on methods, transfer, and application. Psychological Research. 2014;78:749–755.

26. Noack H, Lovden M, Schmiedek F. On the validity and generality of transfer effects in cognitive training research. Psychological Research. 2014;78:773–789.

27. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical Recipes in C (Second Edition). Cambridge: Cambridge; 1992.

28. Nabney I. NETLAB: Algorithms for Pattern Recognition. Springer; 2003.

29. Daw N. Trial by trial data analysis using computational models. In: Delgado M, Phelps E, Robbins T, editors. Decision Making, Affect and Learning: Attention and Performance XXIII. Oxford University Press; 2011.

30. Foong A, Turner R. In-Between Uncertainty in Bayesian Neural Networks. UCL; 2019.

A Neural Network Model

A.1 MTL-Offline

Offline learning ("MTL-Offline") follows the gradient of the log joint

$$\begin{aligned}
\frac{dJ}{dw_k^n} &= \sum_{t\in\tau}(r_t - \pi_{td})z_t \\
z_t &= \frac{(\delta_{dk} - \pi_{tk})}{(1-\pi_{td})}h_t \\
\frac{dJ}{dA} &= \sum_{t\in\tau}(r_t - \pi_{td})G_t^n + \Psi_j \\
\Psi_{jf} &= -\Lambda_{jf}(a_f - m_{jf}) \\
G_t^n &= \sum_{k=1}^{K}\frac{(\delta_{dk} - \pi_{tk})}{(1-\pi_{td})}g_{tk}^n u_t^T \\
g_{tk}^n &= \sum_{p=1}^{P} w_{tk}^n(p)h_{tp}\beta_p(m_p - x_t)
\end{aligned} \tag{18}$$

where $n = s_t$ is the selected task, $d = a_t$ is the selected action and w_k^n is the kth row of W. The quantity $(r_t - \pi_{td})$ is the Reward Prediction Error (RPE). The quantity g_{tk}^n is an $F \times 1$ vector, u_t^T is a $1 \times D$ so that what is added to A_t is of the appropriate dimensionality (i.e. $F \times D$). For the implementation in this paper, rather than using fixed step size updates we use a line search algorithm [27] based on the above gradients. Specifically, on the ith batch update we use

$$\begin{aligned}
w_{i+1,k}^n &= w_{i,k}^n + \alpha_w \frac{dJ}{dw_k^n} \\
A_{i+1} &= A_i + \alpha_a \frac{dJ}{dA}
\end{aligned} \tag{19}$$

but where optimal values for α_w and α_a are found using a single-variable bounded nonlinear function minimisation (implemented using `fminbnd.m` in matlab with step sizes bounded between 0 and 1). Alternative algorithms that would possess similar convergence properties include e.g. Iteratively Re-weighted Least Squares for W and Pseudo-Newton updates for A [28].

All entries in the feature matrix were initialised by sampling from the prior. Additionally, we found that the posterior landscape contains local maxima. We therefore implemented a multi-start optimisation procedure in which optimisation was re-initialised (with a different sample from the prior) until a satisfactory solution was found. This was defined as a solution with an Average Trial Likelihood (ATL) of at least $\gamma_{ATL} = 0.70$. ATL is computed by dividing the batch log likelihood (equation 10) by the number of trials and then exponentiating [29] and is also equivalent to the probability of being correct. If no such solution was found with a maximum of $S_{max} = 9$ starts the best solution was returned. To encourage diversity in the initialisations of the multistart algorithm every row of A was orthogonalised to the previous initialisation of that row (this offers only limited efficacy for two dimensional input spaces).

For the experiments in this paper, the feature subspace was defined using a single feature, $F = 1$. For the RBF mapping (from x to h), eleven basis functions were used spanning a one-dimensional space with centres in fixed positions ($[-10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10]$) and precisions set to $\beta = 1.5$.

A.2 Posterior Distribution

We compute the posterior distribution over A given data, Y_j, from blocks 1 to j. We use an approximate posterior based on a factorised Laplace approximation such that the posterior is assumed to factorise over rows of A and to be Gaussian within rows

$$\begin{aligned} p(A|Y_j) &= \prod_{f=1}^{F} p(a_f|Y_j) \\ p(a_f|D_j) &= \mathsf{N}(a_f; m_f, C_f) \end{aligned} \tag{20}$$

where a_f^T is a $1 \times D$ vector corresponding to the fth row of A, the posterior mean m_f is the estimate of a_f found using Offline Learning. From the Laplace approximation we have

$$\begin{aligned} C_f &= \Lambda_f^{-1} \\ \Lambda_f &\approx -H_f + \Lambda_{jf} \\ H_f(i,i') &= \frac{d^2 L^j(A,W)}{da_f(i)da_f(i')} \end{aligned} \tag{21}$$

where $L^j(A,W)$ is the Log-Likelihood of data in the jth block, H_f is the Hessian matrix for row f which we compute using central differences, and the posterior precision, Λ_f, is the sum of the data precision and prior precision. A full Laplace approximation would scale quadratically with FD whereas this factorised Laplace approximation scales only quadratically with D. For applications with large D, this may nevertheless be too computationally demanding. In these cases one can instead use diagonal [9] or outer-product Hessians [30].

A.3 Statistical Results for HPS and RBF networks

A.3.1 Subspace-by-Block Effects

We first report two-way mixed model ANOVAs with dependent variable being task2 accuracy and independent factors of (i) subspace and (ii) blocks (repeated measure).

For HPS, this revealed a main effect of subspace with accuracy being significantly higher for the same subspace condition (mean same = 0.740, mean different = $0.467, F(1,38) = 848.3, p < 10^{-26}$). It also revealed a main effect of block (F(4,152)=35.6, $p < 10^{-6}$) and an interaction between subspace and block (F(4,152)=19.85, $p < 10^{-4}$).

For RBF, this did not reveal a main effect of subspace with accuracy being almost identical in subspace conditions (mean same = 0.654, mean difference = 0.657, F(1,38)=0.255, p=0.614). It revealed a main effect of block (F(4,152)=284.5, $p < 10^{-18}$) but no interaction between subspace and block (F(4,152)=0.559, $p = 0.459$).

A.3.2 Same Subspace: Learner by Block ANOVA

We then ran a two-way mixed model ANOVA for the same subspace simulations with dependent variable being task2 accuracy and indepenent factors of (i) learner (good/bad) and (ii) blocks (repeated measure).

For HPS this revealed a main effect of learner with performance accuracy being significantly higher higher for good versus bad learners (mean good = 0.770, mean bad = 0.71, F(1,18)=27.4, $p < 10^{-4}$). It also revealed a main effect of block (F(4,72)=40.7, $p < 10^{-19}$) and an interaction between learner and block (F(4,72)=4.13, $p = 0.016$).

For RBF this did not reveal a main effect of learner with performance accuracy being similar for good versus bad learners (mean good = 0.655, mean bad = 0.653,

F(1,18)=0.071, $p = 0.793$). It revealed a main effect of block (F(4,72)=134.8, $p < 10^{-21}$) but no interaction between learner and block (F(4,72)=0.721, $p = 0.525$).

A.3.3 Different Subspace: Learner by Block ANOVA

We then repeated this last analysis but with data from the different subspace simulation.

For HPS this did not reveal a main effect of learner with performance accuracy being similar for good versus bad learners (mean good = 0.471, mean bad = 0.464, F(1,18)=1.55, $p = 0.229$). Neither did it reveal a main effect of block (F(4,72)=2.79, $p = 0.062$) or interaction (F(4,72)=0.169, $p = 0.885$).

For RBF this did not reveal a main effect of learner with performance accuracy being similar for good versus bad learners (mean good = 0.657, mean bad = 0.0.657, F(1,18)=0.002, $p = 0.967$). It revealed a main effect of block (F(4,72)=146.7, $p < 10^{-28}$) but no interaction between learner and block (F(4,72)=0.639, $p = 0.615$).

B Further Analyses of Behavioural Data

B.1 Different-Subspace Learner by Block Interaction

As illustrated in Fig 12 we can see how being a good performer creates an interference effect that reduces over time. A two-way mixed design ANOVA revealed only a significant effect of blocks (Blocks, F(4,72) = 10.415, p <0.001; Performance, F(1,18) = 0.210, p = 0.650; Interaction, F(4,72) = 1.987, p = 0.099). A one-sided interaction test, however, that is testing for increases in performance for good versus bad learners over blocks, would be on the border of significance ($p = 0.05$).

Fig 12. Different-Subspace Learner by Block Interaction. Good learners in task 1 are initially at a disadvantage as compared to bad learners, but end up performing better.

B.2 Within- versus Between-Block Learning

To identify at which time scale it would be most appropriate to set up the computational model we tested to see how much learning takes place within versus between blocks as subjects learn the first task.

We first look at learning within blocks by comparing the average accuracy increase (averaged over all 80 subjects) from the first 25 to the last 25 trials. This increase was significant (one-sided) for all five learning blocks (paired t-tests, average increase and two-sided p-values for Block 1: Mean=0.038, p=0.029; Block 2: Mean=-0.036, p=0.021; Block 3: Mean=0.051, p=0.00004; Block 4: Mean=0.023, p=0.100; Block 5: Mean=0.033, p=0.029). In the discussion we speculate that the significant decrease in block 2 may be due to a change in strategy e.g. from a configural to a rule- or subspace-based strategy.

We now look at learning between blocks by comparing the average accuracy increase (proportion correctly classified) from one block of trials to the next. This increase was significant for the first 3 transitions between blocks (paired t-tests, average increase and two-sided p-values for Block 1 to Block 2: Mean=0.033, p=0.021; Block 2 to Block 3: Mean=0.031, p=0.028; Block 3 to Block 4: Mean=0.038, p=0.001; Block 4 to Block 5: Mean=-0.003, p=0.824). Not that no significant learning takes place in the last transition.

We then looked to see if the increase between blocks was significantly greater than within - it was not (paired t-test, Mean Between = 0.025, Mean Within = 0.0218, p=0.17). Finally, we compared the first 25 trials of one block to the last 25 trials of the previous block to find out if there was an increase during the break between blocks. This difference, Mean Between = 0.0065, was less than Mean Within = 0.0218 (paired t-test, p=0.17) so there is no benefit of the break (if anything there's a startup cost).

Overall, this analysis shows that learning takes place within blocks and that the learning measured between blocks likely reflects the learning with blocks but at a longer time scale (there is no beneficial effect of breaks).

B.3 Self-Reports

We first define a declarative learner as a subject who correctly declared their strategy for the first task. This was based on the participant's response to the open-ended question posed after the first task.

A two-way between-subjects ANOVA with factors of declaration (declared/not declared) and subspace (same different) showed a main effect of subspace (F(1,76)=7.66,p=0.007), no main effect of declaration (F(1,76)=2.7, p=0.104) and a significant declaration by subspace interaction (F(1,76)=7.51, p=0.008). This is illustrated in Fig 13. Note that the size of the subspace effect in the declarative group is 12 per cent, much larger than the subspace effect computed over all subjects (5 per cent - see main text). For the declarative learners in the same subspace group (N=14) the mean acccuracy in task 2 is 82 per cent, whereas for declarative learners in the different subspace group (N=15) the mean acccuracy in task 2 is only 70 per cent. For the non-declarative learners the corresponding figures are both 72 per cent.

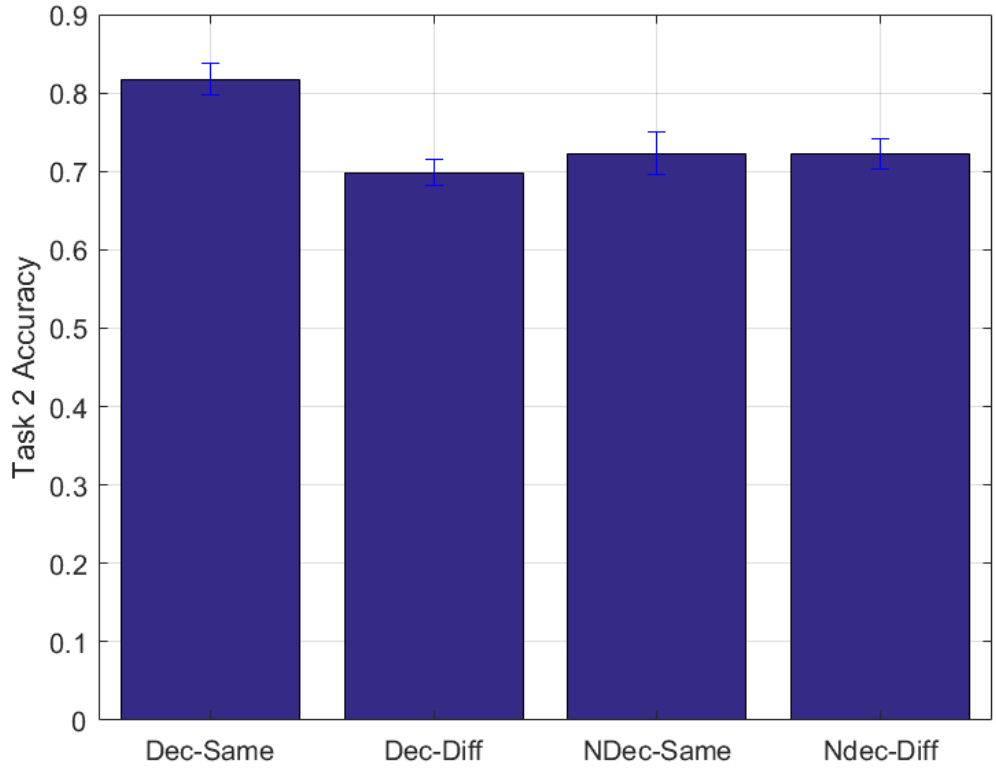

Fig 13. Declaration by Subspace Interaction. The left two bars are for declarative learners and the right two for non-declarative.

However, we cannot conclude from this analysis that the subspace effect is driven by declarative learners. This is because declarative learners also performed well on the first task. On the first task overall, average correct rates were 66 and 55 per cent for participants who declared or could not declare their strategy ($t(78) = 5.45, p < 10^{-6}$). On the last block of the first task, these correct rates were 76 and 56 per cent ($t(78) = 6.22, p < 10^{-7}$). We then defined a "good learner" as being in the group of those 29 subjects with highest performance in the last block of the first task (we chose 29 subjects to match the number of declarative learners). By this definition 19 of the 29 good learners also declared. A two-way between-subjects ANOVA with factors of learner (good/bad) and subspace (same different) showed a main effect of subspace (F(1,76)=5.46,p=0.022), a main effect of learner (F(1,76)=6.32, p=0.014) and a significant learner by subspace interaction (F(1,76)=4.01, p=0.049). This is illustrated in Fig 14. For good learners in the same subspace group (N=16) the mean acccuracy in task 2 is 81 per cent, whereas for good learners in the different subspace group (N=13) the mean acccuracy in task 2 is only 72 per cent. For bad learners the corresponding figures are 72 and 71 per cent.

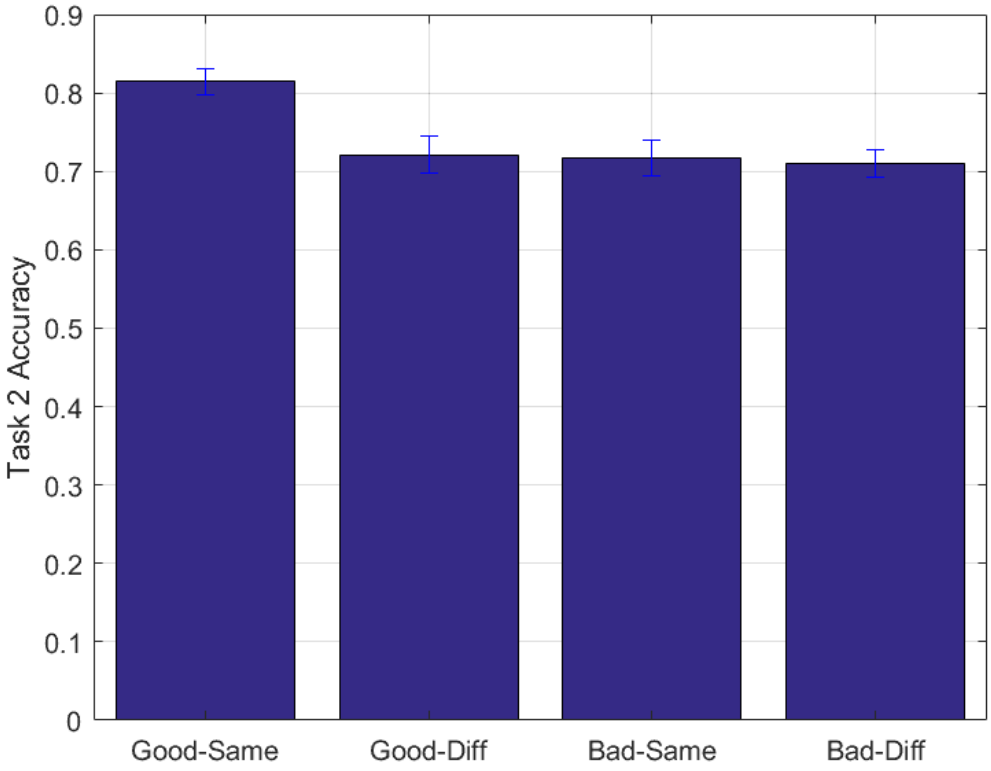

Fig 14. Learner by Subspace Interaction. The left two bars are for good learners and the right two for bad learners.

But, again, we can't conclude that being a "good learner" drives the subspace effect as many of these subjects (19/29) also declared. To shed further light on this matter we ran a 3-way between-subjects ANOVA with factors of subspace, declaration and learner, and neither of the above two-way interactions (declaration by subspace and learner by subspace) were significant. We therefore infer these effects to be collinear and, with the present data, we cannot tell which of these factors is driving the subspace effect.