Supplementary Materials for: Moderate Levels of Activation Lead to Forgetting In the Think/No-Think Paradigm

Greg J. Detre[1]*, Annamalai Natarajan[1]*, Samuel J. Gershman[1], Kenneth A. Norman[1]

[1] Department of Psychology and Princeton Neuroscience Institute, Princeton University

* indicates equal contributions

Address for correspondence:

Kenneth A. Norman

Department of Psychology

Princeton University

Green Hall, Washington Road

Princeton, NJ 08540

E-mail: knorman@princeton.edu

Contents

Overview of Supplementary Materials

The supplementary materials contain three major sections. Section 1 contains technical details about the curve-fitting algorithm. Section 2 contains a description of how we generated simulated data to test the sensitivity and specificity of the curve-fitting algorithm, as well as the results of these simulated-data tests. Section 3 presents importance maps that illustrate which brain regions the classifier was relying on to detect face and scene activity.

For readers interested in replicating or extending our curve-fitting analyses, we have also prepared a fully documented, downloadable toolbox containing:

1. The Matlab routines that implement our curve-fitting algorithm

2. The data tables that were used as input to the curve-fitting algorithm, in order to generate Figures 8 and 9 in the main paper. These data tables contain, for each no-think item from each participant, the 12 classifier evidence values elicited by that item during the think/no-think phase, and a binary value indicating whether that item was remembered correctly on the final test (see Section 2.7.3 from the main paper).

The toolbox is called P-CIT ("Probabilistic Curve Induction and Testing") and it can be downloaded from http://code.google.com/p/p-cit-toolbox/ or from http://compmem.princeton.edu.

1 Curve-Fitting Algorithm Details

This section contains additional mathematical and technical details about the curve-fitting algorithm that were not included in the main paper. Importantly, this section is meant to be a companion piece to our discussion of the algorithm in the main paper, not a stand-alone algorithm description – readers should go through the high-level algorithm description in the main paper first, and then read this section afterwards.

1.1 Probabilistic model

1.1.1 Terminology

Let N be the number of item (word-image) pairs, each of which are presented K (12 repetitions for no-think trials, 6 repetitions for think trials) times during the experiment. Repetition k of pair n is associated with an fMRI-based classifier evidence variable x_{snk} for each participant s. During the final memory test, the word cue from each pair is presented, and the binary variable y_{sn} indicates whether participant s successfully ($y_{sn} = 1$) or unsuccessfully ($y_{sn} = 0$) remembered the associated image for pair n. We shall denote the complete dataset of S participants by $D = \{X, Y\}$.

1.1.2 Generative model

We assume that the probability of successful retrieval depends on the history of an item's activation. In particular, we posit the following logistic regression model:

$$P(y_{sn} = 1|\mathbf{x}_{sn}, \theta_s, \beta) = \frac{1}{1 + e^{-z_{sn}}}, \tag{1}$$

where z_{sn} is a sufficient statistic of the activation history:

$$z_{sn} = \beta_0 + \beta_1 \sum_{k=1}^{K} f(x_{snk}; \theta). \tag{2}$$

β_0 is an intercept parameter and β_1 is a slope parameter shared across participants (note: these z_{sn} values correspond to the "net effect" values described in Section 2.7.3 of the main paper). The plasticity function $f(x; \theta)$ belongs to a family of models $\mathcal{M}$ parameterized by θ.

We did not collect enough data from each participant to estimate the curve shape individually for each participant. As such, we used a model that assumes that the curve-defining parameters θ are shared across participants (refer to Section 1.2.1 for discussion of

curve parameters).

Let $\Theta_{\mathcal{M}}$ denote the parameter space such that $\theta \in \Theta_{\mathcal{M}}$. Our prior over parameters θ within $\Theta_{\mathcal{M}}$ is uniform. Let $|\Theta_{\mathcal{M}}|$ denote the volume (Lebesgue measure) of $\Theta_{\mathcal{M}}$. The prior P(θ) is given by:

$$P(\theta) = \frac{1}{|\Theta_{\mathcal{M}}|} \tag{3}$$

1.1.3 Inference

Each model family $\Theta_{\mathcal{M}}$ can be divided into two disjoint subsets, $\mathcal{H}_1$ and $\mathcal{H}_0$ (which we will refer to as "hypotheses"—i.e., theory-consistent and inconsistent, respectively). Let $\mathcal{H}_c$ denote the true hypothesis where $c \in \{0, 1\}$. Our goal ultimately is to infer the posterior over c, which is given by Bayes' rule:

$$P(c|D, \beta) \propto \int_{\theta} P(\theta|c) \prod_{s=1}^{S} \prod_{n=1}^{N} P(y_{sn}|\mathbf{x}_{sn}, \theta, \beta) d\theta. \tag{4}$$

There are two problems here. First, we do not know β, so we need to estimate it from the data. Second, because the integral in Eq. 4 is analytically intractable, we must resort to approximation. Both of these problems can be addressed using the expectation-maximization (EM) algorithm, which we describe next.

The expectation-maximization algorithm The EM algorithm, first introduced by Dempster et al. (1977), is a method for performing maximum-likelihood estimation in latent variable models. We follow the derivation of Neal and Hinton (1998), who showed that the EM algorithm can be understood as coordinate ascent on the functional $\mathcal{F}(\beta, Q) = \int_{\theta} Q(\theta|D) \log P(D, \theta|\beta) d\theta$ with respect to β and the approximate posterior $Q(\theta|D)$. This functional is a lower bound on the log marginal likelihood, $\log P(D|\beta)$. The EM algorithm

alternates between maximizing $\mathcal{F}(\beta, Q)$ with respect to β and Q. Letting j indicate the iteration:

$$\textbf{E-step} : Q^{j+1} \leftarrow \operatorname*{argmax}_{Q} \mathcal{F}(\beta^j, Q)$$

$$\textbf{M-step} : \beta^{j+1} \leftarrow \operatorname*{argmax}_{\beta} \mathcal{F}(\beta, Q^{j+1})$$

Alternating these steps repeatedly, $\mathcal{F}(\beta, Q)$ will converge to a local maximum. It can be shown that setting $Q^{j+1}(\theta|D) = P(\theta|D, \beta^j)$ maximizes $\mathcal{F}(\beta^j, Q)$ with respect to Q (Neal and Hinton, 1998). In the next sections, we describe how we implemented this algorithm for our generative model.

The E-step In the E-step, we use a Monte Carlo technique known as importance sampling (Gelman et al., 2004) to approximate the intractable posterior:

$$P(c = 1|D, \beta) \approx \sum_{m=1}^{M} w^{(m)} \mathcal{I}[\theta^{(m)} \in \mathcal{H}_1], \tag{5}$$

where $\theta^{(1:M)}$ are drawn from a proposal distribution $G(\theta)$, $\mathcal{I}[\theta \in \mathcal{H}_c] = 1$ if $\theta \in \mathcal{H}_c$, and 0 otherwise. The importance weights $w^{(1:M)}$ are given by:

$$w^{(m)} = \frac{\tilde{w}^{(m)}}{\sum_{j=1}^{M} \tilde{w}^{(j)}} \tag{6}$$

$$\tilde{w}^{(m)} = \frac{P(\mathbf{Y}|\mathbf{X}, \theta^{(m)}, \beta) P(\theta^{(m)})}{G(\theta^{(m)})} \tag{7}$$

The probability density functions are given by:

$$P(\mathbf{Y}|\mathbf{X},\theta,\beta) = \prod_{s=1}^{S}\prod_{n=1}^{N} P(y_{sn}|\mathbf{x}_{sn},\theta,\beta) \tag{8}$$

$$G(\theta) = \sum_{j=1}^{M} \bar{w}^{(j)} \mathcal{N}_t(\theta;\bar{\theta}^{(j)},\tau^2,a,b) \tag{9}$$

where $\bar{w}^{(1:M)}$ and $\bar{\theta}^{(1:M)}$ represent the importance weights and curve parameter samples (respectively) from the previous E-step, and $\mathcal{N}_t(\theta;\bar{\theta}^{(j)},\tau^2,a,b)$ computes the likelihood of θ under a truncated normal distribution with mean $\bar{\theta}^{(j)}$, variance τ^2 and bounds a, b. In other words, the proposal distribution $G(\theta)$ is constructed to be the approximate posterior from the previous iteration of the EM algorithm, smoothed by a truncated Gaussian kernel with variance τ^2. $P(\mathbf{Y}|\mathbf{X},\theta,\beta)$ is the likelihood of behavioral outcomes $\mathbf{Y}$ given plasticity curve parameters θ, classifier evidence values $\mathbf{X}$, and logistic function parameters β, where classifier evidence is used to predict behavioral outcomes according to the generative model described in Section 1.1.2 above.

For the first iteration of the EM procedure, proposals are drawn from the prior, $P(\theta)$, in which case $P(\theta^{(m)})$ and $G(\theta^{(m)})$ cancel out in the equation for $\tilde{w}^{(m)}$ above, and the importance weight $w^{(m)}$ is simply the likelihood normalized over samples. For subsequent iterations, to draw a sample from the proposal distribution $G(\theta)$, we first choose sample j from the previous set of samples with probability $\bar{w}^{(j)}$, and then add noise with variance τ^2 to each parameter in $\bar{\theta}^{(j)}$ from a truncated normal distribution bounded by a, b, where a and b are the minimum and maximum allowable values for that parameter (for horizontal parameters, the min and max are 0 and 1; for vertical parameters, the min and max are -1 and 1). This realizes an *adaptive* importance-sampling scheme, wherein the proposal distribution is focused on regions of high probability density.

In the adaptive importance-sampling approach, we sample the space around a good

curve hoping to find better curves. The τ parameter specifies the width of the space that is explored around each good curve. We tried running analyses with τ values ranging from 0.01 to 0.1. We found that, when τ was less than .05, the algorithm did not do an adequate job of exploring the parameter space, leading to unstable results (in particular, low values of τ led to *sample degeneracy*, where a small number of samples dominated the others; see Section 1.5). τ values of .05 and higher yielded results that were stable across runs of the algorithm and also identical across τ values (so long as $\tau \geq .05$). We set τ to .05 for the curve-fitting analyses reported in the main paper.

The M-step As a result of the E-step, the functional $\mathcal{F}(\beta, Q)$ is approximated by:

$$\begin{aligned}\mathcal{F}(\beta, Q) &\approx \sum_{m=1}^{M} w^{(m)} \log P(D, \theta^{(m)}|\beta) \\ &= \sum_{m=1}^{M} w^{(m)} \sum_{s=1}^{S} \sum_{n=1}^{N} \log P(y_{sn}|\mathbf{x}_{sn}, \theta_s^{(m)}, \beta) \end{aligned} \tag{10}$$

Differentiating this expression with respect to β, we use a gradient ascent algorithm (Matlab's $fminunc$) to find the β that maximizes it. [1] β_0 is initialized to zero and β_1 is initialized to 1.

Because the EM algorithm is designed only to find a local optimum of the objective function, we ran each analysis twice to get an estimate for the posterior probability (for the curve-fitting analyses described in the main paper, convergence across the two runs was excellent). For each run of the curve-fitting analysis, we let the EM process run for 20 iterations. We found empirically that the objective function tended to converge after 10-15 iterations – we selected 20 iterations based on our subjective sense that this was the point of diminishing returns (further iterations beyond 20 took extra time to run, but typically

[1] Technically, $fminunc$ is a gradient descent algorithm, so – instead of maximizing the objective function – we minimize -1 $\times$ the objective function.

did not yield improvements in the objective function).

1.1.4 Evaluation criteria

Weighted final curve and credible interval The approximate posterior can be used to estimate the plasticity curve:

$$\hat{f}(x;\theta) = \mathbb{E}[f(x;\theta)|D,\beta] = \int_{\theta} P(\theta|D,\beta) f(x;\theta) d\theta \approx \sum_{m=1}^{M} w^{(m)} f(x;\theta^{(m)}). \quad (11)$$

Credible intervals (see Gelman et al., 2004) are also readily obtained from this approximation by finding (for each x value) the range of y values that contain the middle γ percent of the weighted probability mass (i.e., such that $1-\gamma/2$ of the probability mass falls below this range, and $1-\gamma/2$ of the probability mass falls above this range). This interval can be interpreted to mean that the posterior probability that $f(x;\theta)$ will pass through the interval for some value of x is equal to γ.

1.2 Plasticity curve

1.2.1 Six parameter curve

Figure 1 (copied over from the main paper for convenience) illustrates our curve parameterization. There are four vertical parameters y_1, y_2, y_3, y_4 that determine the height of the points in the curve; there are also two horizontal parameters x_1 and x_2, where $x_1 < x_2$. The vertical parameters can take on values between [-1, 1] and the horizontal parameters can take on values between [0, 1]. Our curves are defined by the following four points: $(0, y_1)$, (x_1, y_2), (x_2, y_3), and $(1, y_4)$. Moving from left to right, we connect these points using three line segments to form a piecewise linear function.

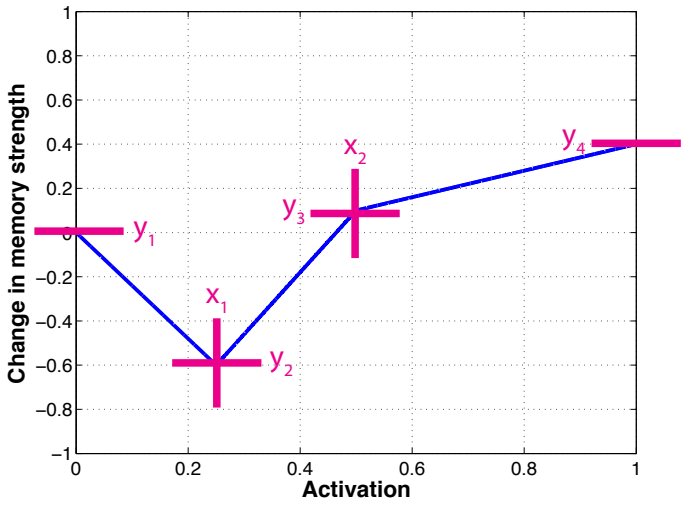

Figure 1: Illustration of piecewise-linear parameterized curve with six adjustable parameters

1.2.2 Criteria for theory consistency

Here we define the criteria that we used for specifying curves as theory-consistent (i.e., consistent with the nonmonotonic plasticity hypothesis) or theory inconsistent. A curve was considered to be theory consistent if – moving from left to right– one of the inner points dipped below the leftmost point (and below zero), and then the curve subsequently rose above zero. Since we have four vertical parameters, we could achieve this requirement in several ways – these ways ("branches") are defined below. Any curve that does not fall into one of these branches is, by default, theory inconsistent.

The different branches of theory-consistent curves can be defined in terms of the location of the *dip* in the curve (the point that anchors the weakening part of the curve) and the *rise* in the curve (the point that anchors the strengthening part of the curve). More formally,

- The *dip* in a theory-consistent curve is a point that is located horizontally between the left edge of the curve and the rise. Within this horizontal range, the dip is the lowest point on the curve; it also has to fall below zero on the y-axis.

- The *rise* in a theory-consistent curve is a point that is located to the right of the dip.

Within this horizontal range, the rise is the highest point on the curve; it also has to fall above zero on the y-axis.

Branch I: y_2 defines the dip and y_3 defines the rise.

1. -1 $\leq y_2 < 0$, y_2 is the dip so it must fall below zero
2. $0 < y_3 \leq 1$, y_3 is the rise so it must fall above zero
3. -1 $\leq y_4 \leq y_3$, y_4 can hold any value that is below the rise (y_3)
4. $y_2 < y_1 \leq 1$, y_1 can hold any value that is above the dip (y_2)

Branch II: y_2 defines the dip and y_4 defines the rise

1. -1 $\leq y_2 < 0$, y_2 is the dip so it must fall below zero
2. $0 < y_4 \leq 1$, y_4 is the rise so it must fall above zero
3. $y_2 \leq y_3 \leq y_4$, y_3 can hold any value between the dip (y_2) and the rise (y_4)
4. $y_2 < y_1 \leq 1$, y_1 can hold any value that is above the dip (y_2)

Branch III: y_3 defines the dip and y_4 defines the rise

1. -1 $\leq y_3 < 0$, y_3 is the dip so it must fall below zero
2. $0 < y_4 \leq 1$, y_4 is the rise so it must fall above zero
3. $y_3 < y_1 \leq 1$, y_1 can hold any value that is above the dip (y_3)
4. $y_3 \leq y_2 \leq 1$, y_2 can hold any value that is above the dip (y_3)

Notice the branches above only impose constraints on vertical parameters. If horizontal parameters are set to 1 or 0 then one or more of the line segments will be missing.

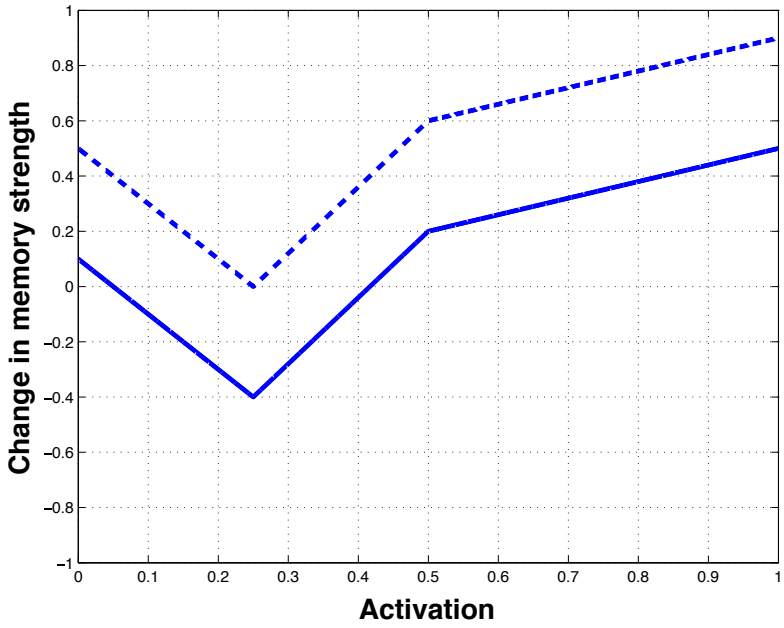

Figure 2: Two identically shaped plasticity curves displaced vertically from each other.

Irrespective of which branch above the curve falls into, the two conditions listed below will ensure that all three line segments exist.[2]

1. $x_1 \neq 0$ AND $x_1 \neq 1$

2. $x_2 \neq 0$ AND $x_2 \neq 1$

When evaluating curves for theory consistency we check if a curve satisfies (Branch I OR Branch II OR Branch III) AND $\{x_1, x_2\} \neq \{0, 1\}$

1.3 Anchoring the vertical position of the curve using baseline items

In the main paper, we described how we included baseline items (with net effect = 0) in the data table, and we mentioned that doing this helps to constrain the vertical placement of the curve; here, we explain why this is the case.

[2]If x_1 and x_2 have the exact same sampled value, the curve-evaluation code is set up to treat x_2 as sitting .0001 to the right of x_1, thereby ensuring that each x value will map onto a unique y value.

Consider the two curves in Figure 2. Because the curves have the exact same shape, the net effect scores for the upper curve will be equal to the the net effect scores for the lower curve plus a constant value (corresponding to the vertical displacement) – as such, the two curves will do an equally good job of explaining variance *within* the no-think condition. The only place where the curves differ is in their predictions about differences *across* conditions. Because the top curve is located entirely above the y = 0 line, it predicts an average net effect score for no-think items that is greater than 0; this implies (incorrectly) that no-think items should be remembered better than baseline items on average. The bottom curve predicts an average net effect score for no-think items that is lower – let us assume that it is less than zero; this implies (correctly) that the average level of recall for no-think items should be numerically lower than the average level of recall for baseline items.

In summary: When baseline items are included with net effect = 0, this "breaks the tie" between the upper and lower curves: Specifically, the lower curve will be assigned a higher importance weight because it gets the numerical ordering of the no-think and baseline conditions correct. Without baseline items, the algorithm has no way of preferring the lower curve over the upper curve based on the data alone; in this situation, the algorithm's choice of curves will be driven by other factors (e.g., our choice of the starting value for β_0).

1.4 Negative β_1 and flipping curve parameters

Our theory-consistency criteria were devised under the assumption that positive y-values on the plasticity curve indicate memory strengthening and negative y-values indicate memory weakening. This assumption is upheld when β_1 (the slope of the relationship between "net effect" values and behavioral memory outcomes in the logistic regression) is positive; however, when β_1 is negative, the opposite is true: Positive y-values on the plasticity curve indicate memory weakening and negative y-values indicate memory strengthening. To avoid this confusing situation, we took advantage of the fact that simultaneously flipping

the sampled curves (i.e., reflecting them around the x-axis) and flipping the sign of β_1 leaves the curve importance weights unchanged (intuitively, this is like taking a double negative; the two changes cancel out). Whenever the curve-fitting procedure settled on a negative β_1 value at the end of the adaptive importance-sampling procedure, we enacted this "double flip", leaving us with a positive β_1 value. We then applied our theory-consistency criteria to the (flipped) set of curves.

1.5 Symptoms of the curve-fitting procedure failing

In some circumstances, the curve-fitting procedure can fail to generate a meaningful estimate of the shape of the underlying plasticity curve. It is important to be able to recognize the symptoms of algorithm failure, so P(theory consistent) values from these defective runs can be disregarded. This section contains a listing of "failure symptoms" that are grounds for disregarding the output of the curve-fitting algorithm.

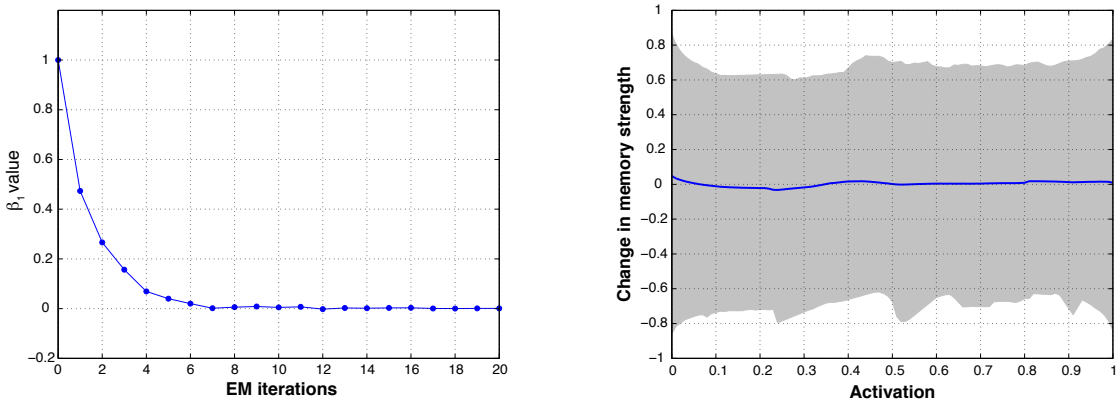

Figure 3: Illustration of a curve-fitting analysis run where β_1 goes to zero (left panel); the resulting weighted final curve is flat line with wide credible intervals (right panel).

- Low values of β_1: Very low values of β_1 indicate that the algorithm is having trouble finding a reliable relationship between neural measurements and behavior. Sometimes β_1 falls all the way down to zero during the curve-fitting process, indicating a complete

failure to relate neural measurements and behavior. When this occurs we observe a flat mean plasticity curve with very wide credible intervals (see Figure 3). In other situations, β_1 can fall to a value very close to zero but not all the way to zero. In this situation, even though none of the sampled curves are doing a good job of explaining behavior in the absolute sense, some of these curves might be doing a relatively better job of explaining the data. If the curves that are doing relatively better happen to be predominantly theory consistent or theory inconsistent, this can cause P(theory consistent) values to deviate from .5. In this case, even though the P(theory consistent) values are deviating from .5, they should be disregarded because the absolute goodness-of-fit values are so low. In general, for a particular dataset, the higher the β_1 value is, the more confident we can be that we are picking up on real signal about the relationship between brain activity and behavior. For our think/no-think analyses (including the simulated-data analyses discussed in Section 2 below), we used a heuristic of automatically disregarding P(theory consistent) estimates when $\beta_1 < .01$ at the end of the curve-fitting process. Note that β_1 values are affected by the scale of the data; as such, a fixed β_1 cutoff that is suitable for one dataset might not be suitable for other datasets. In more recent versions of our software, we have replaced this fixed β_1 cutoff with a statistical test of the null hypothesis that $\beta_1 = 0$. With this statistical test in place, we can use a heuristic of disregarding P(theory consistent) whenever β_1 does not significantly differ from zero.

- Sample degeneracy: In this setting, sample degeneracy refers to a situation where a small number of sampled curves account for a large proportion of the posterior probability mass. Refer to Figure 4 for an example of degeneracy. When degeneracy occurs, this indicates that the algorithm is failing to appropriately sample the full space of curves; this problem can sometimes be fixed by increasing the τ parameter that governs how new curves are generated (i.e., how different the new curves are from

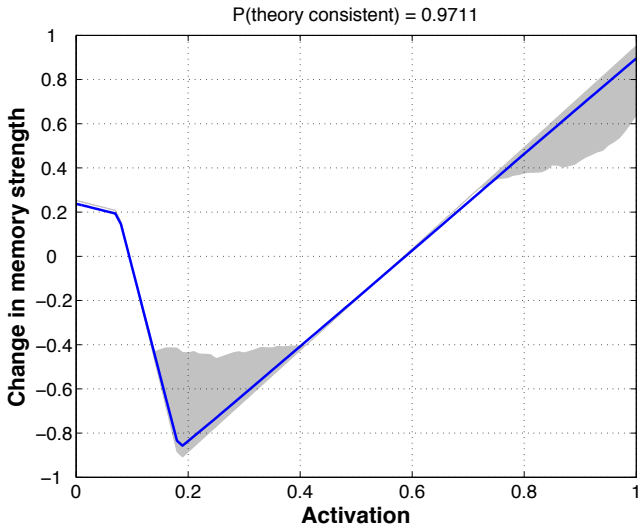

Figure 4: Traces of degeneracy: one sample has an importance weight of 0.9711 and is the driving force behind the final weighted curve. Due to the domination of one sample the credible intervals are very narrow.

previously sampled curves). In our think/no-think analyses, we used a heuristic of disregarding P(theory consistent) estimates when the largest sample weight exceeded .02.

- F-values: As noted in Section 1.1.3, in the M-step of the EM procedure we optimize the objective function $\mathcal{F}(\beta, Q)$ with respect to β using Matlab's $fminunc$ function. If the curve-fitting procedure is working properly (i.e., it is generating a good estimate of the posterior distribution over curves), then the values of the objective function computed by $fminunc$ (henceforth referred to as F-values) should exhibit an overall decrease as a function of expectation maximization (EM) iterations. When the curve fitting procedure fails to generate a good estimate of the posterior, the F-values can sometimes exhibit the opposite behavior. In our analyses, we labeled F-values as increasing if the EM iteration yielding the minimum F-value came before the EM iteration yielding the maximum F-value (e.g., if the minimum F-value was observed on iteration 2 and the maximum was observed on iteration 20); conversely, we labeled

F-values as decreasing if the EM iteration yielding the maximum F-value came before the EM iteration yielding the minimum F-value. Note that small increases in F-values (measured by taking the difference between the maximum and minimum F-values) do not necessarily indicate problems, but large increases indicate that the curve-fitting procedure is failing to converge on the best solution. The question of what counts as a "large" or "small" increase is a gray area. For our think/no-think analyses, we used a heuristic of disregarding P(theory consistent) estimates when F-values increased by 1 or more (note that F-values are reported on a log scale, so an F-value increase of 1 corresponds to an order-of-magnitude increase; see Figure 5 for examples of small and large F-value increases). However, this heuristic may not be appropriate for other datasets; because of the way that F-values are computed, larger datasets tend to generate larger F-values. Rather than looking at the absolute change in F-values, it may be more useful to look at the relative change in F-values (i.e., the difference between maximum and minimum F-values, divided by the minimum F-value). For example, one could use a heuristic of applying further scrutiny to model fits when the relative change in F-values exceeds .01. [3]

- Consistency across runs of the algorithm: When the algorithm is working properly, running the algorithm multiple times on the same dataset will yield highly consistent estimates of the posterior mean curve, the credible interval around the curve, and P(theory consistent). If you run the algorithm multiple times on the same dataset and you get notably different results across runs, this is a sign that the algorithm is not working properly (most likely because of sample degeneracy).

[3] If F-values are increasing but the data look good according to other data-quality metrics (i.e., β_1 values are high and there are no signs of sample degeneracy), the best way to address this situation is to increase the number of samples – this will have the effect of making the approximation of the posterior more accurate. In the limiting case (where the number of samples is infinite, so the approximation perfectly matches the posterior), the EM algorithm is guaranteed to yield decreasing F-values (Neal and Hinton, 1998).

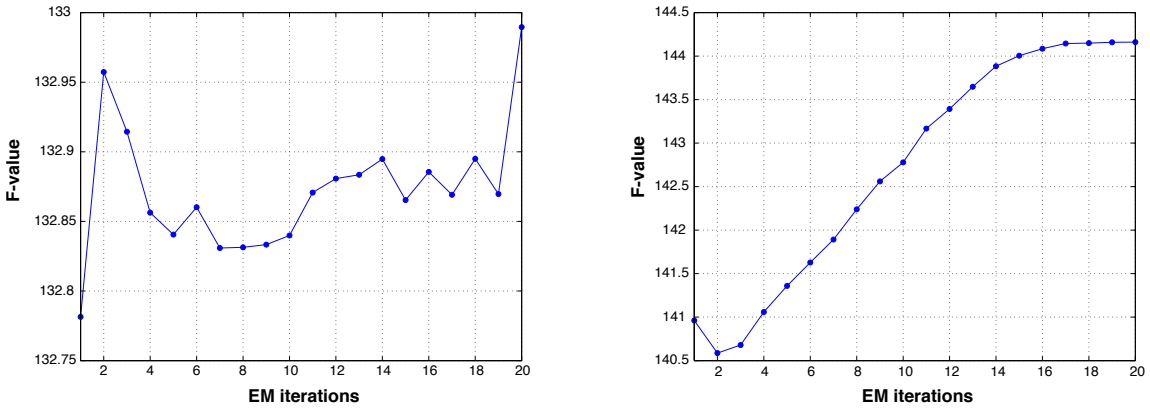

Figure 5: Left side: Illustration of small perturbations in F-values over EM iterations – these perturbations reflect natural variability across successive generations of samples and do not necessarily indicate that the algorithm is failing. Right side: Illustration of an increase in F-values spanning 4 orders of magnitude.

Importantly, none of the curve-fitting analyses described in the main paper exhibited the failure symptoms listed above – for all of the analyses described in the main paper, β_1 was around .5, F-values decreased over EM iterations, the maximum sample weight was less than .01, and P(theory consistent) values were highly consistent across runs of the algorithm.

2 Simulated data

To assess the sensitivity of our curve-fitting algorithm (i.e., how likely it is to detect theory consistency when it is actually present) and the specificity of the algorithm (i.e., how likely it is to report theory consistency when the underlying curve is theory inconsistent), we ran simulated-data analyses. Our simulated-data analysis procedure involved the following steps: First, we generated a ground-truth plasticity curve. Next, we generated simulated memory activation values and behavioral outcomes that were consistent with this curve. After generating these simulated data points, we added noise to the simulated activation

values. Finally, we fed the simulated data into the curve-fitting algorithm and generated a recovered curve. The sensitivity and specificity of the algorithm can be evaluated by comparing the recovered curve to the ground-truth curve.

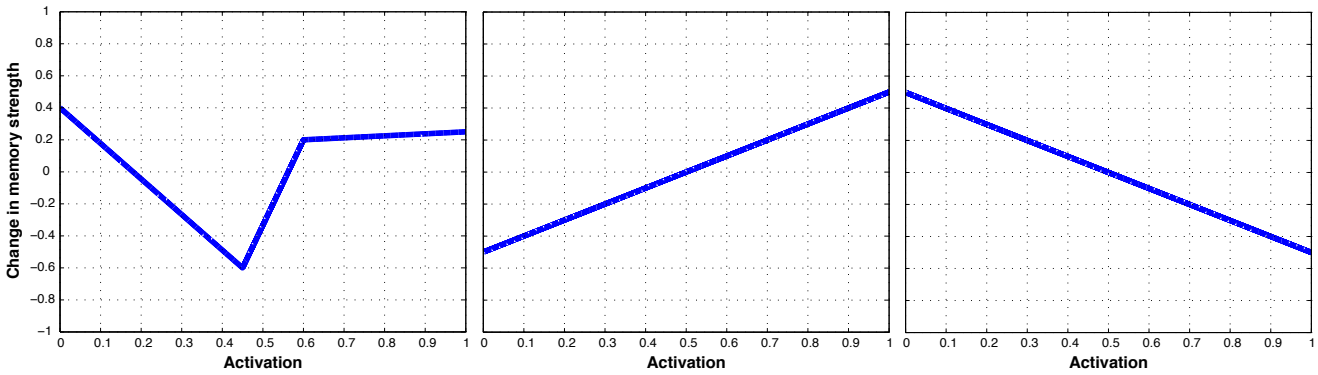

Figure 6: Ground-truth curves used in simulated-data analyses. From left to right: theory consistent, monotonically increasing, and monotonically decreasing ground-truth curves.

2.1 Ground-truth curves

The first step in the simulated-data analyses is to generate ground-truth curves. These curves are generated by specifying the six curve parameters. For more details on curve parameters refer to section 1.2.1. Our simulated-data analyses focused on three ground-truth curves: a theory-consistent curve, with a dip followed by a rise; a monotonically increasing (theory-inconsistent) curve; and a monotonically decreasing (theory-inconsistent) curve (see Figure 6).

2.2 Procedure for generating simulated data

To create a simulated dataset, we took the following steps: First, for each simulated participant, we sampled a "central value" for each simulated item from a uniform distribution between .15 and .85; then, for each simulated item, we generated individual-trial activation

values for that item by adding a uniformly sampled value between -.15 and .15 to the item's central value. Next, within each simulated participant, we rescaled the individual-trial activation values so the maximum value was 1 and the minimum value was zero. After generating these simulated activation values, the next step was to determine which of the simulated items would be successfully (vs. unsuccessfully) remembered. To do this, we used the ground-truth curve to compute a "net-effect" value for each item (by evaluating the ground-truth curve at x values corresponding to each of the individual-trial activation values for that item, and summing together the corresponding y values) – this net-effect value specifies the overall effect of the (simulated) no-think phase on memory for that item, according to the ground-truth curve. Next, within each simulated participant, we ranked the items according to their net-effect values, and did a median split. Items in the upper half of the net-effect distribution were assigned a behavioral memory outcome of 1 (indicating correct responding on the final memory test), and items in the lower half of the net-effect distribution were assigned a behavioral memory outcome of 0 (indicating incorrect responding on the final memory test). After assigning memory outcome values to each item, we went back and added noise to the activation values (from a truncated normal distribution with SD σ, bounded between zero and 1) – this added noise is meant to simulate measurement noise in the fMRI signal; the more noise there is, the harder it should be to recover the shape of the ground-truth curve.

We generated our simulated datasets to match the structure of our actual dataset; each simulated dataset was composed of 26 simulated participants, with 8 items per participant and 12 trials per no-think item. The simulated-data generation procedure is summarized in Algorithm 1. We ran the algorithm with varying levels of noise, ranging from σ = .001 to σ = 2.0. Refer to Figure 7 for an example of simulated data with noise for a single participant. The red and green dots refer to the remembered (green) and forgotten (red) items, eight (rows) in total. Each row has twelve dots corresponding to the twelve repetitions of that

Algorithm 1 Generate simulated data

participants ← 26
repetitions ← 12
items ← 8
for $s = 1$ to $participants$ **do**
  item centers ← Unif([0.15, 0.85], items)
  for $i = 1$ to $items$ **do**
    activations ← Unif([item centers(i)-0.15, item centers(i)+0.15], repetitions)
  end for
  scaled activations ← **scale data**(0, 1, activations)
  net effects ← **compute net effects**(ground-truth curve, scaled activations)
  for $i = 1$ to $items$ **do**
    if net effects(i) $>$ **median**(net effects) **then**
      memory ← 1
    else
      memory ← 0
    end if
  end for
  noisy activations ← scaled activations + $\mathcal{N}_t(\sigma, 0, 1)$
end for
return noisy activations, memory

item. The upper and lower sets of dots correspond to the same ground-truth curve but with 0.001 and 0.5 noise respectively. The simulated classifier data and memory performance data are both N-dimensional vectors where N = participants $\times$ items $\times$ repetitions.

2.3 Simulated-data results

For our simulated-data analyses, we generated five sets of simulated data points for each level of noise (.001 to .2; plus a "uniform" condition where activation values were replaced with uniform random noise) for each of the three ground-truth curves (theory consistent; monotonically increasing theory inconsistent, and monotonically decreasing theory inconsistent). For each set of simulated data points, we computed P(theory consistent). The results of these analyses are plotted in Figure 8. We evaluated each result according to whether the recovered P(theory consistent) value "matched" the ground-truth curve: i.e., for theory-

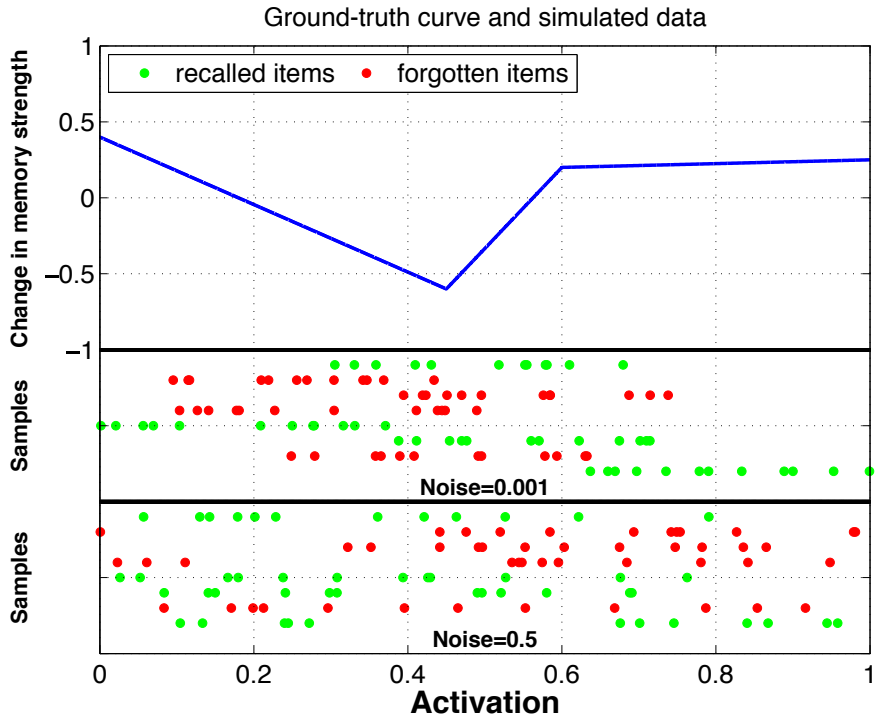

Figure 7: Upper panel: ground-truth curve; middle panel: simulated data with .001 noise added; lower panel: the same simulated data, with .05 noise added.

consistent ground-truth curves, was the recovered P(theory consistent) $>$ .5? Likewise, for theory-inconsistent ground-truth curves, was the recovered P(theory consistent) $<$ .5? Green points indicate simulated-data results that matched the ground-truth curve and red points indicate simulated-data results that mismatched the ground-truth curve. Points outlined in black indicate simulated-data runs that failed one of the data-quality checks outlined in Section 1.5.

In the lowest-noise condition, all of the simulated-data results matched the ground-truth curves (i.e., the balance of evidence was tilted toward theory consistency for theory-consistent ground-truth curves and it was tilted toward theory inconsistency for theory-inconsistent ground-truth curves). The algorithm was able to decisively label monotonically decreasing curves as being theory inconsistent, but it showed more uncertainty about the status of monotonically increasing curves. Overall, P(theory consistent) values tended to

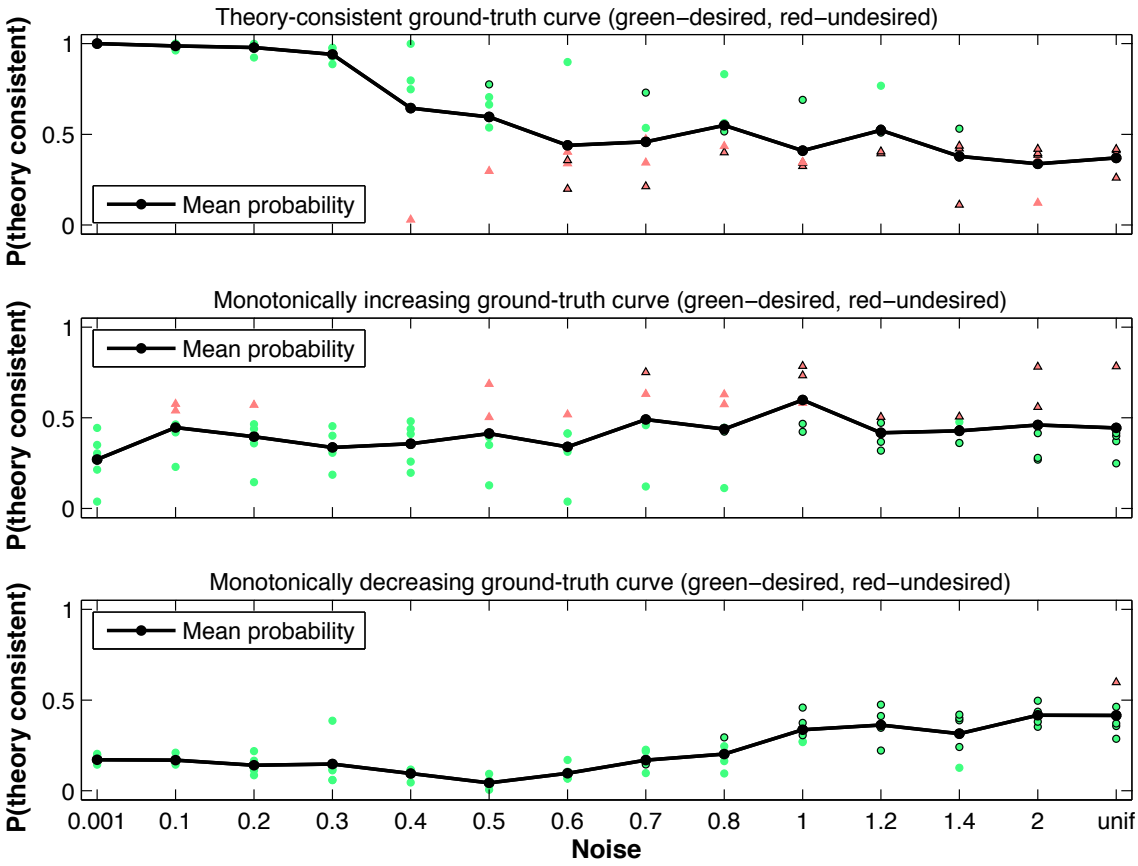

Figure 8: Results from simulated-data analyses where data were generated with a theory-consistent ground-truth curve (upper panel), a monotonically increasing ground-truth curve (middle panel), and a monotonically decreasing ground-truth curve (lower panel). Five simulated datasets were generated for each ground-truth curve and each level of added noise ("uniform" = activation values were replaced with uniform random noise). See text for explanation of green/red point colors and black outlines around the points.

converge toward .5 (indicating increasing uncertainty) as more noise was added.

Of particular concern is the issue of *false positives* – does the curve-fitting algorithm ever assign a high P(theory consistent) value to points derived from a theory-inconsistent ground-truth curve? The middle and lower panels of Figure 8 indicate that simulated data derived from theory-inconsistent ground-truth curves were sometimes assigned P(theory consistent) values greater than .5; this was especially true for simulated data derived from monotonically increasing ground-truth curves. The primary reason why these "false positives" occur is that theory-consistent curves and (theory-inconsistent) monotonically increasing curves can be very similar to one another; adding a tiny dip to the left-side of a monotonically increasing curve will make it theory consistent. Because these two families of curves are so similar, it is possible that – after adding noise – a simulated dataset generated using a theory-inconsistent ground-truth curve might be best explained using a theory-consistent plasticity curve. To assess whether this was occurring in our simulated data, we took every simulated-data run that yielded a "false positive" (i.e., theory consistency greater than .5, with a theory-inconsistent ground-truth curve) and we compared how well the simulated data were explained by the (theory-consistent) recovered mean curve versus the (theory-inconsistent) ground-truth curve. In every case but one, the recovered theory-consistent curve did a better job of explaining the data than the theory-inconsistent ground-truth curve. The other reason why these "false positives" occur is that – as described in Section 1.5 – the curve-fitting algorithm sometimes fails to converge on a meaningful solution when data are noisy (this was the case for the one run where the ground-truth curve fit the data better than the recovered curve).

Regardless of why these "false positives" occur, the fact that they do occur is potentially worrisome. In particular, the fact that P(theory consistent) values rose as high as .79 when we generated simulated data from a monotonically increasing ground-truth curve suggests that our real findings – e.g., our finding of P(theory consistent) = .76 for scene no-think

trials – could also have arisen from a monotonically increasing ground-truth curve.

Fortunately, there are several other measures that we can use to distinguish our actual-data results from the simulated-data "false positives" shown here. The first line of defense is to look at the data-quality diagnostics described in Section 1.5. As noted in Section 1.5, our actual scene, no-think results pass all of the diagnostics; by contrast, most of the simulated-data "false positives" fail these diagnostics (as indicated by black outlines around the points in Figure 8).

The second line of defense is to look at the results of the nonparametric statistical tests described in Section 2.7.6 of the main paper (in particular, the bootstrap resampling test). We took the most compelling "false positive" simulated-data run – i.e., the run with the highest P(theory consistent) value that also passed the data-quality diagnostics from Section 1.5 – and ran the bootstrap test on this simulated dataset. Although P(theory consistent) for this dataset was quite high (.69), the bootstrap test revealed that only 69% of the resampled datasets yielded P(theory consistent) values greater than .5. This can be compared with the actual scene, no-think data, where 95% of the resampled datasets yielded P(theory consistent) values greater than .5 (see Figure 11 in the main paper). We also ran the bootstrap test on two simulated-data runs from the monotonically increasing, noise = 1.0 condition than yielded very high P(theory consistent) values (.73 and .79, respectively; note that these runs failed the data-quality checks due to increasing F-values). Despite the high theory-consistency values, only 84% and 85% (respectively) of the resampled datasets yielded P(theory consistent) values greater than .5. The bootstrap test is a useful means of rooting out false positives because it measures the reliabilty of the results across participants. Intuitively, it is possible to get a few unusual participants who boost up the overall P(theory consistent) value due to chance, but it is much harder to get a U-shaped pattern that is reliable across the full set of participants due to chance.

In summary: Our simulated-data results show that high P(theory consistent) values can

sometimes occur given a theory-inconsistent ground-truth curve, but these "false positives" can be rooted out by looking at data-quality diagnostics and also by running bootstrap tests. For this reason, it is essential to run these additional diagnostics and statistical tests in order to properly interpret P(theory consistent) values.

3 Importance Maps

3.1 Methods

To gain insight into which brain regions were driving classifier performance, we constructed importance maps for the scene and face categories using the procedure described in McDuff et al. (2009). [4] This procedure identifies which voxels were most important in driving the classifiers output when each category (scene, face) was present during the functional localizer. For each category, we computed the average activation of each voxel during the scans that were associated with that category (after correcting for haemodynamic lag) during the functional localizer; note that voxel time courses were z-scored within runs, according to the procedure described in Section 2.4.2 of the main paper. Classifier β weights were obtained from our primary ridge-regression analysis (described in Section 2.5 of the main paper) – this analysis yields one set of β weights per category.

Voxels with a positive β weight and a positive z-scored average activation value for a category were assigned a positive importance weight for that category, with a value equal to the product of the weight and the activation value. Voxels with a negative β weight and a negative z-scored average activation value for a category were assigned a negative importance weight for that category, with a value equal to the product of the weight and the activation value. Voxels where the sign of the β weight differed from the sign of the

[4] Note that these importance maps have nothing to do with the importance-sampling procedure that we used for curve-fitting – the fact that the word "importance" has been used in the past to describe both procedures (even though they have nothing in common) is an unfortunate coincidence.

average activation value were assigned an importance value of zero. Note that, with these equations, both positive and negative importance values indicate a net positive contribution of that voxel to activating the category classifier (when that category is present). The absolute value of the importance score indicates the size of that voxel's contribution. The sign of the importance value indicates whether the voxel contributes via a characteristic deactivation that is picked up by the classifier (via a negative weight) or a characteristic activation that is picked up by the classifier (via a positive weight). We computed these importance values within each participant; we then warped the importance values into Talairach space and averaged together the participant-specific importance maps to get a group importance map. For further details regarding the logic on this procedure, please refer to McDuff et al. (2009). Note that importance maps do not provide a comprehensive treatment of where category-relevant information is located in the brain; as discussed by Norman et al. (2006), there are many reasons why informative voxels might be assigned zero weights. The importance maps are presented in Figure 9 for informational purposes only; no inferential statistics were computed based on these maps.

3.2 Results

Figure 9 shows the category-specific importance maps. Regions that contributed positively to detection of faces (as identified using AFNI's TT-atlas tool) included fusiform gyrus, declive, inferior occipital gyrus, middle occipital gyrus, Brodmann area 18, lingual gyrus, left culmen, and left Brodmann area 19. Regions that contributed negatively to detection of faces (i.e., through a characteristic deactivation relative to the mean when faces were present) included: left parahippocampal gyrus, and left Brodmann area 37. Regions that contributed positively to detection of scenes included fusiform gyrus, parahippocampal gyrus, culmen, declive, Brodmann areas 19, 20, 36 and 37, and lingual gyrus.

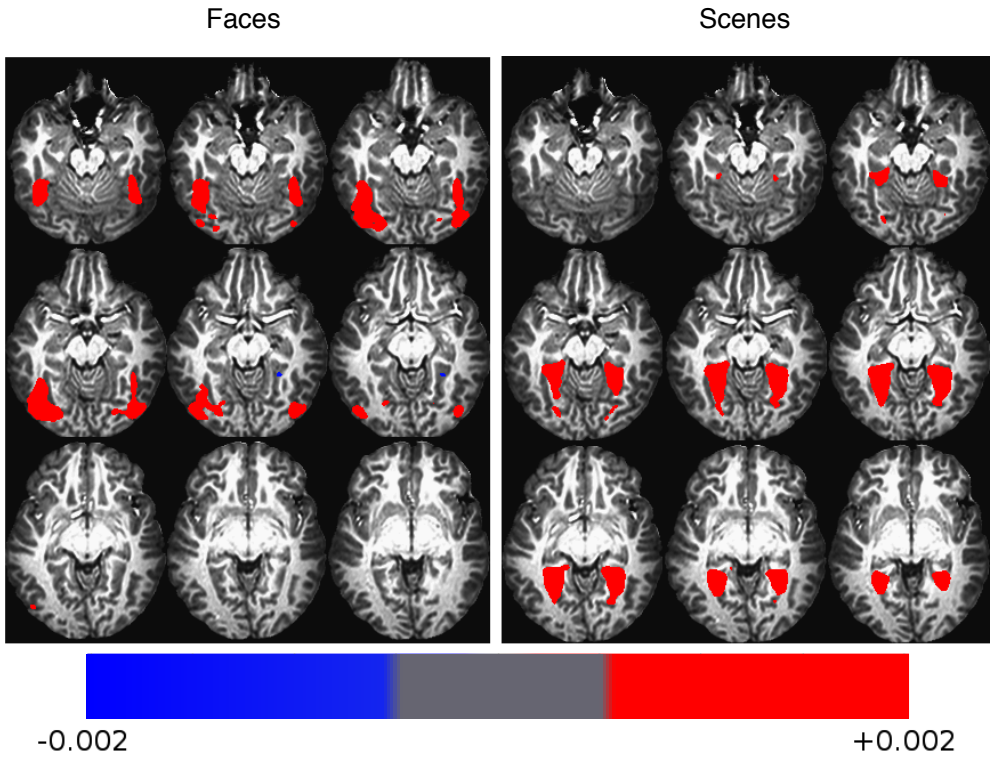

Figure 9: Importance maps showing which voxels were most important in driving the classifier's response to faces (left side) and scenes (right side) during the functional localizer. The montages were created in AFNI by loading the importance maps over the anatomical data from one participant. Both the overlay and the underlay are in a 1x1x1 mm resolution in Talairach space. Voxels with importance values greater than .0005 or less than -.0005 are shown in color. Voxels colored red have positive importance values (i.e., they contribute to category detection via a characteristic increase in activation); voxels colored blue have negative importance values (i.e., they contribute to category detection via a characteristic decrease in activation). Note that map colors directly depict importance values and do not indicate statistical significance. The slices in each 3 x 3 montage (going from top-left to bottom-right, row-wise) correspond to Z = [-19, -17, -15, -13, -11, -9, -7, -5, -3]mm.

References

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Gelman, A., Carlin, J., Stern, H., and Rubin, D. (2004). *Bayesian Data Analysis*. Chapman and Hall.

McDuff, S. G. R., Frankel, H. C., and Norman, K. A. (2009). Multivoxel pattern analysis reveals increased memory targeting and reduced use of retrieved details during single-agenda source monitoring. *Journal of Neuroscience*, 29(2):508–516.

Neal, R. and Hinton, G. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M., editor, *Learning in Graphical Models*, pages 355–368. MIT Press.

Norman, K. A., Polyn, S. M., Detre, G. J., and Haxby, J. V. (2006). Beyond mind-reading: Multi-voxel pattern analysis of fMRI data. *Trends in Cognitive Sciences*, 10(9):424–430.