 mark.sprevak@ed.ac.uk

Predictive coding III: The algorithmic level

Mark Sprevak
University of Edinburgh

19 October 2021

1 Introduction

At Marr's computational level, predictive coding suggests that the task facing the brain is to minimise its long-term, prediction-weighted sensory prediction error. This article focuses on the algorithm and representations by which the brain attempts to accomplish this task – in other words, what predictive coding says at Marr's algorithmic level. In principle, a huge range of possible algorithms could be used to minimise sensory prediction error. Furthermore, nothing has been assumed about the brain's *success rate* in solving this computational problem. There is no commitment within the predictive coding programme to the algorithm for minimising sensory prediction error being entirely reliable or always guaranteed to solve the problem in every case. This leaves a very wide field of possible candidates.

Among the algorithms that could conceivably be used to minimise sensory prediction error include various versions of approximate Bayesian inference – sampling-based techniques such as Monte Carlo simulation as well as the variational Bayesian methods favoured by advocates of predictive coding. It also includes non-Bayesian methods that, while not optimal in the respects favoured by Bayesians, are nevertheless capable of producing effective results in many circumstances. This latter includes a vast and diverse range of algorithms spanning from simple regression methods and likelihood maximisation to sophisticated forms of reinforcement learning and supervised learning. Even a large enough look-up table could, in principle,

be used to allow the brain to minimise sensory prediction error.[1]

Predictive coders tend to have relatively specific proposals about how the brain minimises its sensory prediction errors. Details vary between these proposals – currently, there is no agreed algorithm for predictive coding – but certain broad motifs concerning that algorithm tend to be repeated. These include the idea that the algorithm should involve: (i) a *multi-layered, hierarchically structured artificial neural network*; (ii) a repeated *duplex arrangement of prediction and error units* within each layer; (iii) each layer functioning so as to *minimise prediction error about the state of the layer below*. Some of the formal states of this algorithm may also be interpreted, in a further semantic step, as *probabilistic representations* and the manner in which they are transformed as a version of *approximate Bayesian inference*.

These points will be unpacked in more detail below. In Section 2, I describe how a single layer of the artificial neural network works. In Section 3, I describe how multiple layers connect to form a larger, hierarchically structured artificial neural network. Section 4 describes how the formal states of the artificial neural network can be given a semantic interpretation that allows the network to be viewed as performing approximate Bayesian inference. Section 5 describes how the algorithm can be used to model cognitive processes outside perceptual inference and perceptual learning – for example, motor control. Section 6 provides a brief conclusion and review.

2 Predictive coding in one layer

Predictive coding's algorithm involves an artificial neural network with multiple layers. This section focuses on a single layer. For the sake of simplicity, I describe the bottom-most layer – the one closest to sensory input. The workings of this layer are also deliberately simplified, for ease of exposition. More sophisticated versions are sketched in Section 2.5. In Section 3, I describe how multiple layers, each with a structure identical to that described in this section, are composed to form a hierarchy.[2]

2.1 The computational task

The computational task of a single layer in predictive coding is assumed to be the same as that of the entire system: namely, to minimise its sensory prediction error.

[1] Maloney and Mamassian (2009) describe how table lookup can produce similar results to Bayesian inference.

[2] My account of predictive coding's algorithm is based on those of Bogacz (2017); Friston (2005); Rao and Ballard (1999); Spratling (2017).

Each layer takes its 'sensory' input from below. The bottom-most layer takes, as its input, the signals delivered by the brain's external sensory organs. The computational goal of a layer is to converge on (i) a set of *prediction values* and (ii) a *generative model* that, when appropriately combined, reconstruct (i.e. predict) its sensory input as closely as possible. The *algorithm* by which the layer does this consists in two types of stepwise operation that occur on different time scales: (i) updating the activation values of the artificial neural network ('inference'); (ii) updating the connection weights of the artificial neural network ('learning').

Before describing the steps involved in inference and learning in quantitative terms, it is necessary to first formalise the computational task of a layer.[3] For the sake of simplicity, we will ignore the effects of both precision-weighting and long-term averaging of the error. Assume that the task of a layer is to *minimise its current sensory prediction error*. How might that problem be characterised in formal, mathematical terms? We might say that there are m numerical values, x_1 to x_m, which we label the 'sensory inputs'. These numbers might correspond to the magnitude of physical activities in the brain's sensory organs, e.g. the firing rates of individual sensory receptors. However, from the point of view of the algorithmic-level description, the particular details of the physical implementation are intentionally ignored or bracketed.[4] The x_i are simply notional values to be estimated or predicted. A layer's task is to estimate these x_i values ('sensory input') as accurately as possible using another set of numerical values, y_1 to y_n, (which we will call the 'prediction values') and a matrix of numerical weights, $w_{0,0}$ to $w_{m,n}$ (which we will call the 'generative model').

Let us call a layer's estimate of its sensory input, x_i, based on its prediction values and its generative model, its 'prediction', r_i. In the simplest version of a predictive coding algorithm, the system will generate these predictions, r_i, using a *linear* generative model: it estimates the x_i using a weighted sum of y_j values, where the weight of each prediction value is determined by the corresponding entry of the generative model, $w_{i,j}$. The function of the generative model is to modulate how much the value of each y_j contributes to the estimate of each x_i. Formally, a linear model estimates the x_i values in the following way:

$$r_1 = w_{0,0}y_0 + w_{0,1}y_1 + \ldots + w_{0,n}y_n$$

$$r_2 = w_{1,0}y_0 + w_{1,1}y_1 + \ldots + w_{1,n}y_n$$

[3] See the discussion of formal versus informal computational-level descriptions in Sprevak (forthcoming[b]), Section 3.

[4] For proposals about how the x_i map onto physical activities in the brain, see Sprevak (forthcoming[c]), Section 4.

$$\vdots$$

$$r_m = w_{m,0}y_0 + w_{m,1}y_1 + \ldots + w_{m,n}y_n$$

Or, in vector notation, $\mathbf{r} = \mathbf{W}\mathbf{y}$, where $\mathbf{r}$ is the layer's prediction, $\mathbf{W}$ is the generative model, and $\mathbf{y}$ are the prediction values that, when combined with elements of $\mathbf{W}$, generate the prediction.[5] The computational task of a single layer is thus to find $\mathbf{y}$ and $\mathbf{W}$ values that produce an $\mathbf{r}$ that matches the actual input, $\mathbf{x}$, as closely as possible.

The measure of by how much a layer misses its goal is the 'sensory prediction error', $\mathbf{e} = \mathbf{x} - \mathbf{r}$.[6] Sensory prediction error, $\mathbf{e}$, is an m-dimensional vector. The task of the layer is to minimise this vector. In order to do this, one needs a measure of 'how much' prediction error there is in $\mathbf{e}$. Typically, this figure is assumed to be the sum of the squares of the values in $\mathbf{e}$. A layer's computational task is, therefore, to find $\mathbf{y}$ and $\mathbf{W}$ that minimise the sum of the squares of the prediction errors over its sensory inputs, i.e. find $\mathbf{y}$ and $\mathbf{W}$ that, when combined to produce a prediction, $\mathbf{r}$, minimise $\sum_i (x_i - r_i)^2$.[7]

2.2 The inference algorithm

So far, we have only described a layer's computational task. We have not said *how* a layer should go about finding a combination of $\mathbf{y}$ and $\mathbf{W}$ that minimises its sensory prediction error. What would such an algorithm look like?

Generally speaking, an algorithm is a series of simple, rule-governed steps that can, in principle, be mechanised. An algorithm for solving the task is shown in Figure 1. The algorithm does not take the form of a flowchart or a sequence of explicit instructions (e.g. 'if A, then B'). Instead, it takes the form of an *artificial neural network* (ANN). The ANN has 'error' units (e_1 to e_m) and 'prediction' units (y_1 to y_n). The prediction and error units are linked by a series of excitatory and inhibitory 'connections'. The numerical 'activation level' associated with each unit in the ANN is the value of the variable of the same name, e_i or y_j. The 'strength' associated with each connection between the units is the weight of the corresponding element, $w_{i,j}$, of the matrix $\mathbf{W}$. The excitatory and inhibitory connections of the ANN are arranged so as to be of equal and opposite weight: the excitatory connection between e_i and y_j of weight $w_{i,j}$, is paired by an inhibitory connection between them, running in

[5] The $\mathbf{y}$ values are sometimes called 'coefficients' of the model.

[6] The $\mathbf{e}$ vector is also known as the 'residual error'.

[7] This is equivalent to optimising for minimal mean-squared error, $\sum_i (x_i - r_i)^2/m$, see Sprevak (forthcoming[a]), Section 4.

the opposite direction, of weight $-w_{i,j}$.[8]

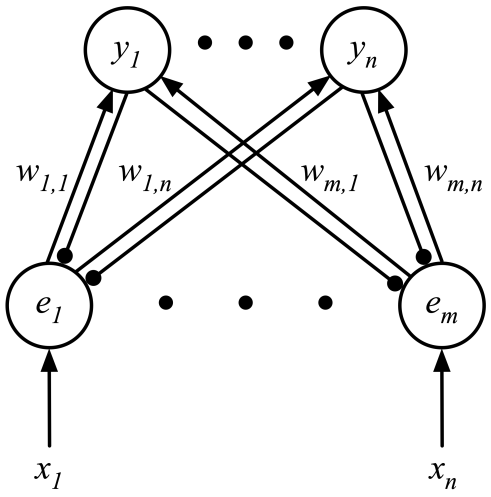

Figure 1: A single layer in predictive coding (diagram adapted from Spratling, 2017, p. 95; Harpur, 1997, p. 44)

Unlike real neurons, the outputs of the individual units of the ANN in a linear predictive coding model are assumed to be a linear function of their inputs. Their numerical activation level is the sum of their connected inputs weighted by the connection strength for that input, $y_j = \sum_i w_{i,j} e_i$.[9] This rule, called the 'activation function', specifies how the activation level of a unit depends on the activation levels of its incoming connected units and the weights of those connections.[10]

When the algorithm runs, the sequence of addition and multiplication operations specified by the activation function are applied in turn to every unit of the network (y_i and e_i) to update its activation level, and then the entire process is repeated. Over time, the y_i and e_i values will progressively change. The y_i values may eventually settle into a stable set of numbers or they may cycle between different sets of numbers. It is possible to prove that if an ANN with the aforementioned activation function and topology is run on some fixed input **x**, the activation level of the **y** units in

[8]For an introduction to artificial neural networks, see Bechtel and Abrahamsen (2002); Clark (2014), Ch. 4.

[9]Real neurons are highly non-linear. They change their response profile to issue a spike if input activation breaches a certain threshold. This is followed by a refractory period when they yield little or no output irrespective of their input.

[10]Error units are governed by an activation function of the same kind, i.e. $e_i = \sum_j -w_{i,j} y_j$.

the network will tend to converge on values that accomplish the task described in Section 2.1. In other words, the activation values of the **y** units will gradually vary – in a way that is wholly determined by the activation function and the structure of the network – towards a new set of values that tend to minimise the prediction error (i.e. that minimise the sum of the squares of the components of **e**).[11] If the **x** input values subsequently change – i.e. if the sensory input changes – then the prediction values **y** will change too in an effort to keep up.

If the steps of the algorithm that governs the ANN were unfolded and written out as a list of individual instructions, we would see that what is described here is really just shorthand for a very long sequence of elementary numerical operations – repeated additions and multiplications that progressively modify the e_i and y_j values – to converge on the **y** that will minimise the sum of the squares of the **e** values for a given **x** and **W**. An ANN with all its units and connections is no more than a statement of a series of additions and multiplications that should be applied in response to – or, given that the ANN commences with certain **y** values, *in anticipation of* – any input.

2.3 The learning algorithm

ANNs can run in two modes: *inference* or *learning*. What has been described so far is the network running in inference mode. In inference mode, connection weights, **W**, are assumed to be fixed and the **y** activation levels are varied – by repeated application of the activation function – to minimise the prediction error. In learning mode, **x** inputs and **y** prediction values are assumed to be fixed, and the connection weights, **W**, are varied – by repeated application of a 'learning rule' – to minimise the prediction error over those **x** and **y** values. When engaged in inference, the computational system varies its *prediction values* (**y**) to try to make its prediction (**Wy**) approximate the actual sensory input **x**. When engaged in learning, the system varies its *generative model* (**W**) to try to make its prediction (**Wy**) approximate the sensory input **x**. Advocates of predictive coding are generally keen to stress that for both inference and learning the computational task is the same: namely, to minimise sensory prediction error.[12]

An ANN cannot run in both inference and learning modes at the same time. Attempting to simultaneously vary both **y** and **W** to minimise prediction error would turn the error-minimisation task into an ill-posed problem. If an ANN is searching for **y** values using the rules described in Section 2.2, some set of weights **W** need to be assumed; if it is searching for **W** values to minimise its prediction errors, some

[11] See Harpur (1997), Section 4.2. The proof involves showing that the described ANN would perform a version of minimisation by gradient descent on the prediction-error value.

[12] For example, see Friston (2005), pp. 815, 821.

set of **x** and **y** values need to be assumed. If neither factor is held fixed, the system would not know whether or by how much to change its prediction values, **y**, or its predictive model, **W**, in response to a prediction error.

Advocates of predictive coding deal with this problem by assuming that the ANN performs inference and learning on different timescales. Over 'short' timescales (assumed to be of the order of hundreds of milliseconds for the brain), the ANN runs in inference mode, settling into the **y** values that minimise prediction error for some given **x** and **W**. Over 'long' timescales (assumed to be of the order of seconds, minutes, days, or years for the brain), many **x**, **y** values are assumed and a **W** (generative model) is sought that minimises prediction error over those pairs. If one sets aside any assumptions about physical implementation and considers what is being proposed only within the idealised world of numerical algorithms, the idea is that the learning rule should be applied to a network's connection weights once its prediction values **y** have settled into a relatively stable state after one or more episodes of inference have run to completion for a given set of **x** values. It is assumed that episodes of inference will in general complete comparatively quickly, and that updating of connection weights (learning) can occur in the pauses between inference.

Predictive coding proposes that the network uses a *Hebbian learning rule*. This specifies that, at each computational step during learning, the ANN should change a connection's weight in proportion to the current activation levels of the two units that it connects. The rule provides a step-by-step mathematical procedure by which to change connection weights, $\delta w_{i,j} = \eta e_i y_j$, where $\delta w_{i,j}$ is how much a connection weight $w_{i,j}$ should change each step, e_i and y_i are the activation levels of the error and prediction units linked by the connection, and η is a constant value that sets how rapidly connection weights change during learning (the 'learning rate' for the network). It is possible to show that if a Hebbian learning rule is applied to an ANN shown in Figure 1, the network will tend to converge on connection weights **W** that minimise prediction error over its past inputs and predictions.[13] Hebbian learning represents a different approach to learning to the 'backpropagation' rules that are currently popular in the AI community and which are utilised to great success by deep learning systems (LeCun, Bengio and Hinton, 2015). A Hebbian learning rule is regarded as attractive in this context because it makes learning depend only on local interactions between units, which is regarded as a more realistic model of the

[13]See Harpur (1997), Section 4.7. The proof again involves showing the ANN would perform a version of gradient descent on its prediction-error value – this time gradient descent over the space of possible connection weights, rather than over the space of possible activation values.

forms of plasticity found in the brain (Bogacz, 2017, p. 199).[14]

As with the activation function that governs inference, the Hebbian rule that governs learning should be viewed as a series of elementary mathematical steps. It reduces learning to a long sequence of numerical operations. Learning for predictive coding consists – at least at the algorithmic level – in a series of additions and multiplications on the elements of **W**. It is worth noting that inference and learning do not reduce to *exactly* the same computational task on this model. The sensory prediction errors minimised during inference are the *current* sensory prediction errors. The sensory prediction errors minimised during learning are *past* sensory prediction errors averaged over past inputs and their associated prediction values. Neither quantity maps in any obvious way onto the not-fully-articulated, prospective measure of long-term sensory prediction error favoured by predictive coding at Marr's computational level (see Sprevak, forthcoming[b], Section 5).

2.4 Precision weighting

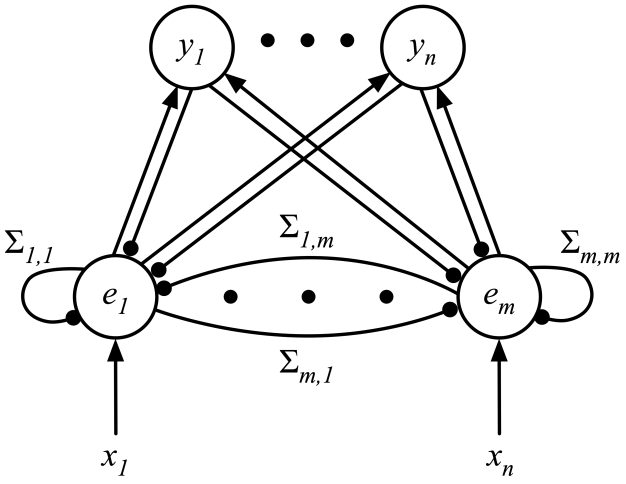

Figure 2: A single layer with inhibitory connections to allow precision weighting of the error signals.

In the current model, each error unit e_i has as much influence as any other during both inference and learning. There is no precision weighting of prediction errors.

[14] Backpropagation rules are commonly used to train systems in supervised learning; Hebbian learning rules are generally seen as better suited to model unsupervised forms of learning (Krotov and Hopfield, 2019).

The standard way to add precision weighting to the model is to modify the ANN to add *lateral and intrinsic inhibitory connections* between error units. These allow an error unit to inhibit both itself and its peers (Feldman and Friston, 2010, pp. 1–2; Friston, 2005, p. 823). Inter-error connections selectively dampen some error units, giving them outsize importance relative to their companions. The inhibitory connections between error units effectively control the 'gain' of each error unit.[15] Being the target of strong inhibitory connections can turn an error unit down; being the target of weak inhibitory connections tends to turn an error unit up. The weights of these new inhibitory connections, $\Sigma_{u,v}$, are captured by a matrix, $\mathbf{\Sigma}$, called the 'precision matrix'.[16]

The precision matrix, $\mathbf{\Sigma}$, should not be confused with the generative model, $\mathbf{W}$. The generative model, $\mathbf{W}$, records how much each prediction value contributes to the sensory prediction, $\mathbf{r}$; the precision matrix, $\mathbf{\Sigma}$, records how much each component of the sensory prediction error, $\mathbf{e}$, is dampened. Introducing weighted connections between error units effectively creates a new degree of freedom for how sensory prediction error might be minimised when the ANN runs. Potentially, a prediction error can be reduced by changing the prediction values ($\mathbf{y}$), changing the generative model ($\mathbf{W}$), or by changing the precision weighting over the error signal ($\mathbf{\Sigma}$) to selectively dampen certain components of the signal. Minimising sensory prediction error thus involves, not just changes to two parameters, $\mathbf{y}$, $\mathbf{W}$, but potentially changes to three parameters, $\mathbf{y}$, $\mathbf{W}$, $\mathbf{\Sigma}$. The individual steps that constitute inference (which governs changes to $\mathbf{y}$ for fixed $\mathbf{W}$) and learning (which governs changes to $\mathbf{W}$ for fixed $\mathbf{y}$) are widely known and typically take a form similar to those described above. However, at the algorithmic level it is not immediately obvious which step-by-step procedure should govern changes to $\mathbf{\Sigma}$.

Bogacz (2017) suggests that the algorithm for changing $\mathbf{\Sigma}$ should be a Hebbian learning rule of the same kind as that which determines the changes to $\mathbf{W}$ (pp. 206–208). The virtue of this proposal is that it makes it straightforward to incorporate changes to $\mathbf{\Sigma}$ into predictive coding's algorithm: connection weights, $\Sigma_{u,v}$, should be updated according to the same rule, and at the same time, as connection weights for the generative model, $w_{i,j}$. However, treating connection weights for the generative model and those for precision weighting of the error signal in the same way suggests that any change to precision weighting will be a relatively 'slow' process, one that unfolds in the same gradual, incremental way as learning. To this end, Bogacz observes that the precision matrix should be viewed as relatively stable quantity that, like the generative model, is acquired and maintained over one's lifetime;

[15] Equivalently: it allows certain prediction values or model parameters to held more 'confidently' by the system than others because they are less likely to be changed by activity flowing into their corresponding error units.

[16] The inverse of the $\mathbf{\Sigma}$ matrix is sometimes called the 'covariance matrix'.

it is not a volatile parameter, like $\mathbf{y}$, that may change rapidly and drastically on short timescales (p. 202). This does not, however, sit well with what predictive coders say about the psychological function of precision weighting. They associate precision weighting with the allocation of the subject's attention. This appears to place changes to precision weighting, $\boldsymbol{\Sigma}$, on the 'short' timescale of inference rather than the 'long' timescale of learning. Precision weighting should be capable of rapidly and dramatically reshaping the flow of information across the system to produce the psychological and behavioural effects observed in, for example, task switching or attentional capture.[17] It is hard to see how these kinds of quick, transformative changes in the ANN can occur if changes to precision weighting are restricted to taking place at timescales much longer than those of inference. In other words, it is not clear how a slow, gradual, Hebbian-based learning rule could be responsible for them.[18]

Kanai et al. (2015) sketch a different procedure by which changes to $\boldsymbol{\Sigma}$ might be produced. They claim that changes to precision weighting are determined extrinsically, by a second ANN, which they provisionally locate inside the pulvinar nuclei of the thalamus (see Sprevak, forthcoming[c], Section 6). Like with the present ANN – which they claim is physically implemented in the neocortex – this secondary ANN may be interpreted as performing a variety of message-passing subjective Bayesian inference (see Section 5). In distinction to the primary ANN however, the secondary network is arranged to perform a 'second-order' inference *about* the precision of (inverse of the variance of the degree of belief in) the error signals in the first ANN. The units of the two ANNs are joined by a series of connections such that the second ANN receives both predictions and prediction errors from the primary ANN, and its second-order predictions about precision weightings influence the $\boldsymbol{\Sigma}$ values of the first ANN. Hohwy (2012) provides an informal description of how this arrangement is supposed to work (pp. 3–4). However, the precise details of the numerical algorithm – the specific individual mathematical steps taken by the second ANN to modify the connection weights $\boldsymbol{\Sigma}$ of the first ANN – are still not clear or widely agreed.

2.5 Extending the algorithm

The algorithm sketched so far should be understood as only a basic skeleton that may be extended in any number of different ways. What we have described could be viewed as the most simple example of predictive coding. We assumed that each layer is governed by a linear generative model and that this model has a

[17] See Clark (2016), pp. 146–151; Friston (2003), p. 1345.

[18] An additional puzzle is how shifts in attention could be under volitional control if changes to $\boldsymbol{\Sigma}$ are determined by the rules of Hebbian learning.

fixed number of parameters ($\frac{nm}{2}$).[19] There is no scope for a layer to depart from a linear generative model, grow or shrink the number of parameters (by creating or removing a prediction unit or connection), change its activation function, break the symmetrical arrangement of equal and opposite weighted connections between prediction and error units, swap the learning rule, or otherwise depart from the many assumptions hard-wired into the algorithm. It is natural to wonder whether some of these assumptions regarding how a layer tries to predict its input should be loosened or modified.

It is common to suggest that the simple algorithm for a single layer should be modified to introduce some non-linearity into the generative model. Friston argues that the brain approximates sensory input using a non-linear generative model in which prediction units, y_i, have a non-linear influence on the error units below them, e_i.[20] This means that changes in the activation level of error units during inference need not be proportional to changes in the activation level of their connected inputs – the input an error unit receives may depend on both lateral interactions between prediction units and on a non-linear function of their individual activation levels.

Bogacz (2017) discusses several ways in which such a non-linearity might be built into the steps taken by the ANN. These include modifying the activation function of prediction or error units, or inserting additional artificial 'inter-neurons' with non-linear activation functions between the existing prediction and error units (p. 203). The exact nature of the non-linear function that should affect the outputs of the prediction units is also unclear. Friston interprets the connections between prediction and error units as encoding a generative probabilistic model of the sensory input (see Section 5). The required non-linear function could therefore be assumed to correspond to whatever would be necessary to encode the non-linear aspects of that probabilistic model.

A predictive coding algorithm need not be restricted to using a generative model with a set number of parameters. FitzGerald, Dolan and Friston (2014) suggest that the predictive coding algorithm engages, not only in inference and learning, but also in *model comparison*. Model comparison is usually regarded as a step that occurs at a level of abstraction above learning. Whereas learning assumes the system is using a generative model with a fixed number of parameters and it aims to optimise the values of those parameters relative to some objective function (e.g. to minimise prediction error), model comparison aims to find the *type* of generative model to subject to learning (e.g. would a model with N or $N+1$ parameters do a better job?). The predictive coding algorithm described above does not attempt to do any model

[19] Half the $n \times m$ elements of $\mathbf{W}$ are fixed because of the assumption made about reciprocal connections between prediction and error units, $w_{i,j} = -w_{j,i}$.

[20] Friston (2005), p. 823; Friston (2009), Box 3 on p. 297.

comparison. It is restricted to using a generative model with a fixed structure and with $\frac{nm}{2}$ parameters. FitzGerald, Dolan and Friston (2014) propose that the brain does something akin to model comparison, albeit slightly more sophisticated, called 'Bayesian model averaging'.

Bayesian model averaging assumes that the system has not one, but many generative models and that it has a subjective probability distribution over those generative models reflecting its prior credences in those models. The agent then applies Bayes' rule to calculate a posterior probability distribution over all its generative models given the observed data (Bishop, 2006, pp. 161–165). At this point, the agent could conceivably elect to adopt just one generative model – perhaps the one with the highest subjective probability given the data, the 'maximum a posteriori' model.[21] Alternatively – and this is what FitzGerald, Dolan and Friston (2014) suggest – the agent may continue to operate in a Bayesian fashion and entertain a full posterior subjective probability distribution over all its generative models and deploy this full posterior distribution in inference, updating it and revising its degree of belief in those models as more data comes in.[22] The predictions that the agent generates on such a scheme would be the average of the predictions of all models weighted by the system's posterior subjective probability in each model given the observed data.[23] FitzGerald, Dolan and Friston (2014) propose that this model averaging process occurs, not inside a single layer of predictive coding's ANN, but in the interaction between discrete layers, with the predictions from different generative models being weighted by their different top-down influences on lower layers (ibid., p. 3). The precise implementation of this, both in terms of the specific steps that an ANN

[21]This would be a form of Bayesian model selection.

[22]Note that Bayesian conditionalisation introduces a 'bias' towards simpler, more constrained models. Models with fewer parameters will effectively receive a bonus during Bayesian conditionalisation – a higher probability in the agent's posterior distribution – even if the agent was indifferent between those models before. Roughly speaking, this is because a model that makes no assumptions about the specific value of some additional parameter should, everything else being equal, be assigned more subjective probability than one that makes exactly the same assumptions *and* an extra assumption about that additional parameter's specific value. Bayesian model comparison (and model averaging) will thus tend to drive an agent towards models with fewer free parameters. (See MacKay, 2003, pp. 343–351 for a full explanation of how this Bayesian "Occam's razor" works.) FitzGerald, Dolan and Friston (2014) argue that this feature provides an explanation of how predictive coding can allow the brain to optimise for both *simplicity* as well as predictive *accuracy* of its generative model (c.f. Sprevak, forthcoming[a], Section 2).

[23]FitzGerald, Dolan and Friston (2014) claim that the computational task of Bayesian model averaging also entails minimising variational free energy, and hence (granted certain additional assumptions) minimising sensory prediction error. Therefore, like inference and learning, Bayesian model averaging falls under predictive coding's single computational-level task description of minimising sensory prediction error (pp. 2–3, Appendix A3–A5). See Friston and Stephan (2007), pp. 434–435.

should take and its physical basis remains, however, somewhat unclear.[24]

Clark (2016) suggests that the number of parameters in the generative model might change over time because the connections between prediction and error units may be subject to some 'pruning' algorithm (p. 272). A pruning algorithm is a procedure that would systematically remove connections between prediction and error units that are, according to the algorithm, deemed to be redundant. Many possible pruning algorithms exist for ANNs.[25] These algorithms normally operate on a timescale that is longer than both those of inference and learning, attempting to remove connections after one or more episodes of learning are complete. It is not clear which pruning technique should apply to predictive coding's ANN, the timescale it should run on, and how its operation would fit with FitzGerald, Dolan and Friston (2014)'s proposal about Bayesian model comparison and Bayesian model averaging.

These suggestions are indicative of just a few of the ways in which the basic algorithm of predictive coding might be changed or elaborated. The research programme of predictive coding takes the basic ANN described in the preceding sections as a starting point and develops it in different ways.

3 An algorithm, not an implementation

It is important to stress that what has been described is an *algorithm*. The ANN is sometimes called an 'implementation' of predictive coding.[26] This terminology is potentially misleading as there are no assumptions about physical implementation built into the model. There is no assumption, for example, that the units depicted in Figure 1 are physical neurons, that their connections correspond to synapses, or that their activation levels map in any straightforward way to neural firing rates. What has been described is a numerical procedure – a step-by-step method for modifying *numbers*. It accomplishes a task defined solely in numerical terms: to find values of $\mathbf{y}$ and $\mathbf{W}$ that minimise $\sum_i (x_i - r_i)^2$ for a sequence of given $\mathbf{x}$ (that we have labelled, without further justification, 'sensory input').

Despite its name, an artificial neural network is no more neural than any other algorithm, such as QuickSort, the Newton–Raphson algorithm for finding roots of real-valued functions, or the Runge–Kutta algorithm for finding solutions to ordinary differential equations. The prediction and error units and connections shown in Figure 1 may, in some loose sense, be suggestive of neural structures in the brain.

[24] See the brief discussion of this in FitzGerald, Dolan and Friston (2014), Appendix A5.

[25] For a survey, see Blalock et al. (2020).

[26] Friston (2005), pp. 822–823; Friston (2010), p. 132; Rao and Ballard (1999), p. 86; Spratling (2017), pp. 93–94.

But it should be clear that what is proposed here is only a numerical method for solving a numerical problem – effectively, a long sequence of elementary additions and multiplications that would enable someone without insight or ingenuity to estimate one set of values ($\mathbf{x}$) with another set of values ($\mathbf{y}$, $\mathbf{W}$, $\mathbf{\Sigma}$). These operations could easily be implemented in any number of pieces of hardware, including very un-brain-like hardware, such as an electronic PC.

What advocates of predictive coding are likely to mean when they describe the ANN as an implementation of predictive coding is that the algorithm in question is *biologically plausible* – it is the kind of algorithm that *could* be implemented in a brain. It is based around numerical states and procedures – a network-like structure, vector-based computational states, and Hebbian learning – that lend themselves to neural implementation.[27] However, the possibility of neural implementation does not mean that what has been described *is* a neural implementation. What is proposed at the algorithmic level by predictive coding is a sequence of abstract numerical operations to accomplish a computational task that has been characterised as a numerical problem. Precisely how these operations map onto the hardware of the brain, or onto the informal task description of a real-world agent minimising its sensory prediction error, is a separate issue and one that we will turn to later (see Sprevak, forthcoming[c]).

4 Hierarchical structure

A single layer attempts to predict its sensory input using a model, $\mathbf{W}$, each of whose variables, y_i, makes an independent contribution to the prediction. This kind of model makes sense if each variable of the model aims to track an independent feature of the input. However, many worldly features that are useful to track for prediction are not structured in this way. Often they stand in a hierarchical relationship to each other. For example, the *general category* of an encountered object (e.g. living thing, inanimate object) constrains the *specific type* of object it is (human being, cat, dog), which further constrains its *identity* (your father, Tiddles the cat). These are not three independent features in the world, but ones that stand in a well-defined relationship to each other. They should not be modelled by three variables with no inherent structure, but by variables that somehow capture the hierarchical

[27] See discussion of this point in Friston (2005), p. 823; Friston (2010), p. 130; Bogacz (2017), pp. 199, 209. Spratling (2017) raises doubts about neural plausibility of the version of the predictive coding algorithm described here as it assumes prediction and error units can take negative values as activation levels (pp. 94–95).

relationship between their corresponding features.[28]

A rich source of hierarchical relations between worldly features derives from features that occur on different spatial and temporal scales. The brain tracks sensory information at multiple spatiotemporal levels and information at one spatiotemporal scale helps constrain the brain's hypotheses at another.[29] The visual system, for example, tracks large-scale, 'gist'-like features – e.g. whether it is facing a jungle, desert, or underwater scene – as well as small-scale, 'detail'-like features – e.g. whether a particular patch of colour is green, red, or blue. The external features that generate our sensory input are structured in a hierarchical fashion over multiple spatiotemporal scales. A plausible strategy for an accurate predictive model would be to aim to mirror that structure inside the model. Rao and Ballard summarise the rationale as follows:

> The underlying assumption here is that the external environment generates natural signals hierarchically via interacting hidden physical causes (object attributes such as shape, texture and luminance) at multiple spatial and temporal scales. The goal of a visual system then becomes optimally estimating these hidden causes at each scale for each input image and, on a longer time scale, learning the parameters governing the hierarchical generative model. (Rao and Ballard, 1999, p. 80)[30]

Despite their focus on causal structure, there is nothing specifically causal about the point Rao and Ballard make about the benefits of using a hierarchical model for prediction. *Any* hierarchical relationship between external properties in the world – be that a hierarchy induced by a causal structure or one that arises for any other reason, e.g. due to the relationship between determinables and determinates or between types and tokens – might profitably be modelled by a model that reproduces the hierarchy amongst the variables that track them.[31]

[28] See Gelman and Hill (2007), pp. 3–8 for discussion of the benefits of using hierarchical models for prediction. See Mumford (1992) for an informal description of a hierarchical form of predictive coding in which the brain tries to match pattern-recognition 'templates' at various levels in its processing. Lee and Mumford (2003) give a more detailed hierarchical Bayesian version of the model for the visual system, albeit one that uses a different algorithm for inference (particle filtering) to that typically associated with predictive coding (variational inference using message passing).

[29] Hochstein and Ahissar (2002); Kadar and Ben-Shahar (2012).

[30] See also Friston, Kilner and Harrison (2006), p. 70.

[31] Building a hierarchical model of the world is *one* strategy to achieve accurate predictions. This is not to say that a predictive model that fails to encode the true hierarchical (causal or other) relationships between hidden features in the world might not be good enough at prediction for many practical purposes. See Cisek and Kalaska (2010) on the benefits of simple, 'pragmatic' representations and Clark (2016), Ch. 8 on 'frugal' forms of prediction.

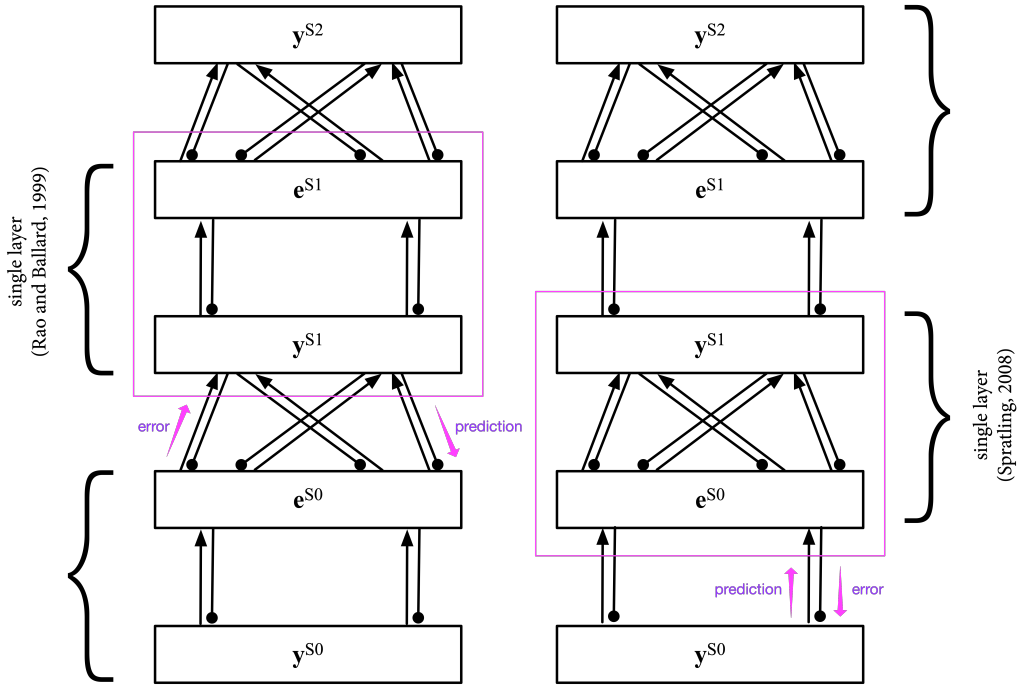

Figure 3: Hierarchical predictive coding according to two different schemes for dividing the ANN into layers. Rao and Ballard's scheme is shown on the left-hand side; Spratling's is shown on the right-hand side. Single prediction and error units inside each layer (circles in the previous diagram) are omitted for simplicity (adapted from Spratling, 2017, p. 95)

The predictive coding algorithm encodes relationships between variables using a hierarchy of inter-linked generative models. It consists of a stack of multiple ANNs, each identical to that described in Section 2, connected to each other. Each error unit of a layer (e_i) connects to a prediction unit in the layer below (y_i). So defined, the computational task of a single layer of the ANN is to predict the activation levels of the prediction units inside the layer below (the output of that lower layer serves as its 'sensory input'). Activation levels of prediction units in layers that are increasingly distant from sensory input will tend to track more abstract, high-level features in the sensory input.[32] The bottom-most layer has no prediction units below it; its inputs are clamped to some externally supplied sensory input, $\mathbf{x}$, as shown in Figure 1. The entire stack of layers will adjust the activation levels of prediction units, y_i, during inference to minimise sensory prediction errors, e_i, via gradient descent, with each layer operating to minimise the prediction error about the prediction values of the layer below. When the network is allowed to run, activation levels will evolve to minimise prediction error concerning the external sensory signal, $\mathbf{x}$, consistent with minimising prediction error at each level of the model's hierarchy.

From the point of view of a single layer, the primary change from the model previously described is the introduction of new connections between layers (denoted by vertical lines in Figure 3). These connections allow the predictions at one level of the predictive hierarchy to influence (via the intervening error units) the predictions at other levels. Unlike the weighted connections inside a layer, which are used to encode the generative model, these new connections between layers are of a constant, equal weight and they are not subject to modification by learning. Each error unit is assumed to connect to exactly one prediction unit in the layer below – i.e. each component of $\mathbf{e}^{Si}$ is connected, by reciprocal excitatory and inhibitory connections, to a single component of $\mathbf{y}^{Si}$. Given this arrangement, each error unit will measure the *prediction error* for its counterpart prediction unit in the layer below (or, for the bottom-most layer, the error in predicting the corresponding component of the sensory input, $\mathbf{x}$).

To see why the new connections do this, consider the *S*1 error units in Figure 3. Their activation levels are a function of just two factors: (i) excitatory inputs they receive from prediction units in the layer below ($\mathbf{y}^{S1}$); (ii) inhibitory inputs they receive from the layer's own prediction units filtered through the connection weights of the model ($\mathbf{W}\mathbf{y}^{S2}$). The net activation level of the *S*1 error units is thus:

$$\mathbf{e}^{S1} = \mathbf{y}^{S1} - \mathbf{W}\mathbf{y}^{S2}$$

The error units $\mathbf{e}^{S1}$ signal the difference between (i) the actual activation level of

[32] See the results of simulations run by Rao and Ballard (1999), p. 84.

the prediction units inside the layer below ($\mathbf{y}^{S1}$) and (ii) a 'prediction' about those levels generated by the prediction units from above ($\mathbf{W}\mathbf{y}^{S2}$). If these two quantities were to match exactly, then the $\mathbf{e}^{S1}$ units would be silent. If the discrepancy between these two quantities were to increase, so would the activation level of the $\mathbf{e}^{S1}$ units. The error units provide a measure of *prediction error* – the prediction values minus the layer's estimate of those prediction values based on its generative model.[33]

Figure 3 omits the inhibitory connections, $\boldsymbol{\Sigma}$, between error units that encode precision weighting. However, these can be straightforwardly incorporated into the ANN. With respect to the equation above, this would involve introducing a normalisation term that modulates the response of the error units in proportion to their relative precision weighting. If the inhibitory connections, $\boldsymbol{\Sigma}_{S1}$, were added, the error units in *S*1 would signal *precision-weighted prediction error*, $\mathbf{e}^{S1} = \boldsymbol{\Sigma}_{S1}^{-1}(\mathbf{y}^{S1} - \mathbf{W}\mathbf{y}^{S2})$.[34]

The hierarchical ANN for predictive coding is governed by exactly the same rules for inference and learning as those for the single layer described in Section 2. The activation function and the Hebbian learning rule are applied to every unit and weighted connection in the network as inference and learning proceed. The generative models encoded in the connections linking prediction and error units, since they are used at the same time, may be viewed as components of one giant, overarching, hierarchical generative model. Like with other ANNs, the pictured ANN is no more than a compact, finite mathematical formalism that prescribes a long sequence of additions and multiplications. What advocates of predictive coding mean by proposing this model is that this sequence would take anyone that follows it from one set of numerical values $\mathbf{x}$, to a series of 'prediction value' vectors ($\mathbf{y}^{Si}$) and 'generative model' matrices ($\mathbf{W}^{Si}$), such that when the prediction values are combined with the generative model according to the rules above, the $\mathbf{x}$ are reconstructed as accurately as possible, given that counts as 'accuracy' is modulated by a further set of values, the 'precision weightings' ($\boldsymbol{\Sigma}_{Si}$).

The formal model described so far is based on Spratling's (2017) account of hierarchical predictive coding. According to Spratling, a *single layer* of the hierarchical ANN is a set of prediction and error units connected by weighted connections (i.e. $\mathbf{y}^{Si+1}$ and $\mathbf{e}^{Si}$ form a layer). The function of a layer of the hierarchy, so defined, is to *predict the activation levels of the prediction units in the layer below*. Its success or failure in this is tracked by the activation levels of the layer's own error units. Given this assumption, the connections *inside* a layer have a relatively complicated structure – their weights encode a generative model and are subject to change during learning. In contrast, the connections *between* layers have a relatively simple structure and are

[33] See Bogacz (2017), p. 201; Friston (2003), pp. 1343; Friston (2005), p. 821; Spratling (2017), p. 94.
[34] See Bogacz (2017), pp. 202, 204–208.

not subject to change by learning. The function of the connections between layers is to *compute the prediction error*, as described by the equation above. Consequently on Spratling's view, the outputs of a 'layer' are: (i) its prediction values, y_i, which are passed *up* the hierarchy to the layer above; (ii) its prediction error values, e_i, which are passed *down* the hierarchy to the layer below. In short, error signals will flow *downwards* and prediction values will flow *upwards* between layers.

The hierarchical model first proposed by Rao and Ballard (1999) and subsequently developed by Friston (2005), Friston (2008), and Bogacz (2017) operates with a different conception of a hierarchical layer. For Spratling a 'layer' means a set of prediction and error units connected by weighted connections. For Rao and Ballard, it means a set of prediction and error units that stand in one-to-one relationship to each other (i.e. $\mathbf{y}^{Si}$ and $\mathbf{e}^{Si}$ form a layer). The function of a layer of the hierarchy, so defined, is to *compute the prediction error*. The error units subtract the prediction made by the layer above from the prediction values held inside the layer, as described by the equation above. Given this assumption, the connections *inside* a layer have a relatively simple structure and are not subject to change by learning. In contrast, the connections *between* layers have a relatively complicated structure – their weights encode a generative model and are subject to change during learning. On Rao and Ballard's view, the outputs of a 'layer' are: (i) its prediction values, y_i, which are passed *down* the hierarchy to the layer below; (ii) its prediction error values, e_i, which are passed *up* the hierarchy to the layer above. In short, error signals will flow *upwards* and prediction values will flow *downwards* between layers.

It is important to emphasise that these two proposals, considered purely as numerical methods, are identical. They agree about which mathematical steps should be taken, when they should taken, and in what order. Their point of disagreement concerns only how to *label* features of the ANN as 'layers'. Which method of labelling one favours makes no difference to the step-by-step operation of the abstract algorithm. This difference does, however, have consequences when it comes to making claims about the neural implementation of the algorithm. These claims involve mapping labelled parts of the algorithm (e.g. layers) onto discrete brain structures (e.g. cortical areas). We will explore how these two notions of layer affect claims about neural implementation in Sprevak (forthcoming[c]), Section 3.

5 Representing probabilistic guesses

In the preceding sections, I described the predictive coding algorithm as a numerical method that takes a set of numbers ('sensory input') and finds two sets of numbers ('prediction values' and 'generative model parameters') that, if combined, would reconstruct the first set as accurately as possible relative to some agreed (precision

weighted) error measure. However, these formal steps are frequently accompanied by an *interpretation* that connects the numbers to features that matter to the brain in cognition. The numerical values are meant to be understood not simply as abstract numbers; they should also be linked in some way to the informally characterised task that the brain faces of minimising its sensory prediction error. Interpretations have already been hinted at by the labels we assigned to these values ('sensory input', 'prediction', etc.), but there is more to say here.

One might distinguish between interpretations of predictive coding's algorithm that are representation-*light* – that do not assume that the algorithm is a process defined over representations – and those that are representation-*heavy* – that do assume that the algorithm is a process defined over representations.

According to a representation-*light* interpretation of the algorithm, the numbers correspond to certain physical quantities of functional significance to the brain in cognition. Precisely which physical quantities would be specified by what predictive coding says at the implementation level (Sprevak, forthcoming[c]). One might, for example, suggest that the magnitude of the **x** values measures the firing rate of certain neurons at the sensory periphery; the magnitude of the **y** values measures of the firing rate of certain populations of cortical neurons; and the magnitude of the **W** values measures the strength of synaptic connections between those populations of cortical neurons. Given this interpretation, the algorithm describes not just abstract relationships between numbers, but also how cortical neurons cooperate to suppress incoming sensory signals.

If one adopts this interpretation, then when one says that the brain issues a 'prediction', there need be no suggestion that the brain *represents* that sensory signal, that it has a hypothesis 'about' it. The interpretation does not commit to the algorithm involving content-bearing, semantically evaluable states – 'representations' as normally conceived. A 'prediction' on this reading is not like a weather report that provides a prediction *about* tomorrow's weather. Instead, it would 'predict' in the same sense that the seat of your chair 'predicts' the downward force exerted by your body: by cancelling it out, or quenching it. The predictive coding algorithm describes how certain physical magnitudes in the brain (quantified by **y** values) combine with other physical magnitudes (quantified by **W** values), to cancel out further physical magnitudes near the sensory boundary (quantified by the **x** values). 'Prediction' should be understood as a process of brute matching of incoming physical signals at sensory neurons. If a prediction is 'successful', that would mean that the brain's internally generated physical signals successfully counterbalance those at

the sensory boundary caused by the world.[35] Precisely which physical quantities are in balance – which aspects of brain function map onto which numerical values of the algorithm – will depend on what is said by predictive coding at Marr's implementation level. However, the basic idea – which might be fleshed out in many different ways – is that the numbers and mathematical operations in the algorithm should be understood as useful ways of describing the *dynamics of physical brain activity*. Such an interpretation of predictive coding's algorithm is notably silent about whether any representations are manipulated during that physical process.[36]

A representation-*heavy* reading assumes that the algorithm describes, *not only* the physical dynamics of brain activity, but *also* how representations are processed during cognition. In this case, a 'prediction' should be understood, at least roughly, on the model of a weather report. A prediction is associated with both a *physical magnitude* (as specified by the account of implementation) and a *semantic content*. It makes sense to ask what a prediction is 'about', what it refers to in the world. The numerical states and operations of the algorithm should be understood as describing not only physical dynamics, but also a kind of semantically rich *inference*. Representations are manipulated according to steps determined by the algorithm and that inference may be evaluated as conforming (or not) to various norms. It is not unusual for computer algorithms to be interpreted in this way. A chess-playing algorithm, for example, is often interpreted as requiring changes not only in the physical states of any machine that implements it, but also the manipulation of *representations* of chess pieces and chess positions in such a way that conforms to the norms that govern the game of chess.[37]

The dominant representation-heavy interpretation of the predictive coding algorithm treats that algorithm as describing a form of *probabilistic inference* (Bogacz, 2017; Friston, 2005; Friston, 2009; Friston, 2010). On this view, the representations being manipulated are subjective probabilities (probabilistic representations) and the process described by the algorithm is a form of approximate Bayesian inference. The key assumption of this interpretation is that the numerical *activation levels* and *weights* of the ANN units and connections encode the *sufficient statistics of subjective probability distributions*. These subjective probability distributions are the brain's

[35] Mumford (1992) describes how the brain strives to match incoming sensory stimulation – to inhibit the sensory neurons to the right degree to counteract the excitatory stimulation they receive from the outside world. A 'perfect' prediction means that the relevant sensory neurons would be silent (p. 247).

[36] Hohwy (2013) gives a wonderful illustration of a representation-light reading of the predictive coding algorithm by describing a Rube Goldberg-esque machine that 'predicts' – without in any obvious fashion using representations – leaks in a water dam by progressively moving a hierarchy of arms, cogs, wheels to insert plugs into spots where leaks have tended to appear (pp. 62–63).

[37] See Baker (1985), pp. 6–7.

subpersonal guesses or hypotheses about the values of various hidden variables in the environment (e.g. shapes, sizes, locations, and identities of environmental objects). These subjective probability distributions are combined during inference – according to the step-by-step rules described by the algorithm that progressively modify the values that correspond to their sufficient statistics – to generate a prediction about the sensory signal. Just as a classical computer algorithm – that applies symbolic rules to symbolic expressions – provides a way to automate deductive inference, so an ANN – that applies simple mathematical operations to numbers – provides a way to automate probabilistic inference. Given the right interpretation, an ANN with the appropriate structure can be viewed as a probabilistic inference engine.[38]

One of the main assumptions adopted by this representation-heavy interpretation of predictive coding is that the brain's *marginal* subjective probability distributions – its subpersonal guesses marginalised over specific variables – are always Gaussian, and so they can be fully characterised by just two numerical values, a mean and variance. The mean is assumed to be encoded by the activation level of an individual prediction unit, y_i. The variance is assumed to be encoded in the weight of the intrinsic inhibitory connection of the prediction unit's corresponding error unit, $\Sigma_{i,i}$. The covariance between marginal distributions (y_i, y_j) is encoded by the weights of lateral connections between their corresponding error units, $\Sigma_{i,j}$. The conditional probabilities that determine how marginal distributions are combined in inference are encoded by the connection weights between prediction and error units, $w_{i,j}$. If one interprets the numbers in the ANN in this fashion, the step-by-step operations of the ANN's algorithm can be shown to implement a message-passing form of Bayesian inference over a graphical probabilistic model. Predictions (**Wy** values) in higher layers of the ANN's hierarchy can be shown to act as priors on subjective probabilities (**y**) in lower layers.[39]

We saw in Section 2.5 that there is no single, agreed algorithm for predictive coding. Different proposals might be developed about the precise structure of the ANN, the activation functions of its units, and the learning rule that governs its weighted connections. There is also scope for different proposals about how to interpret an ANN's numerical elements as encoding subjective probability distributions, and

[38]The notion that numerical activation values and connection weights inside an ANN can be interpreted as subjective probabilities and the rules of the ANN as entailing that the network performs some form of Bayesian (or some other kind of) probabilistic inference is not new. For discussion of the employment of ANNs as probabilistic inference engines, see Hinton and Sejnowski (1983); Hinton and Sejnowski (1986); MacKay (2003), Ch. 41; McClelland (1998); McClelland (2013). For discussion of the general idea of algorithms as ways to automate semantic inference – automatic formal systems as 'semantic engines' – see Dennett (1987); Haugeland (1981).

[39]See Bogacz (2017), pp. 199–202; Friston (2005), 821–822 for worked examples.

consequently about exactly what kind of probabilistic inference the ANN should be interpreted as performing. There is widespread agreement that predictive coding's ANN should be interpreted as performing some form of *message-passing* probabilistic inference over a probabilistic graphical network.[40] The nodes, edges, and numerical values of that graphical probabilistic model should correspond in some (ideally simple!) way to the structure of the ANN and its processing. But exactly how the probabilistic graphical model maps onto the structure of the ANN, and exactly which message-passing algorithm is performed, is not clear. The substance of a representation-heavy interpretation will therefore depend on both the details of the ANN in question and on the exact scheme for how the numerical values of ANN are mapped onto subjective probability distributions. As with representation-light options, there is scope here for different ideas. Friston, Parr and de Vries (2017) and Parr et al. (2019) review a range of options, including algorithms that perform variational message passing (a message-passing version of variational Bayes) and belief propagation (a message-passing form of exact Bayesian inference).

On a representation-light interpretation, the ANN describes the physical dynamics of the brain as it inhibits incoming sensory signals. On a representation-heavy interpretation, the ANN describes, in addition, a form of probabilistic reasoning over subpersonal hypotheses entertained by the brain about its environment. Predictive coders often move freely between representation-light and representation-heavy formulations of their algorithm. It is fair to say that it is an open question whether predictive coding *should* be interpreted in a representation-heavy way or not. Complicating the issue is a lack of clarity about how *any* ascription of subpersonal subjective probabilities to the brain should be understood in cognitive neuroscience – whether such probabilistic representations are 'really there' in the brain or whether talk of them is just a useful *façon de parler* for us to interpret and group together neural activity.[41] These foundational issues about neural representation are however, to an extent, orthogonal to the specifics of predictive coding's algorithmic-level proposal. That proposal is, in principle, compatible with a range of different views regarding how we understand representation in the brain, including viewing it as performing a highly specific form of subjective Bayesian inference or remaining neutral about whether it performs any semantically rich inference at all.

[40]For a general introduction to message-passing forms of probabilistic inference and graphical probabilistic models and, see Bishop (2006), Ch. 8; MacKay (2003), Chs. 16, 26; Russell and Norvig (2010), Ch. 16.

[41]See Colombo and Seriès (2012); Colombo, Elkin and Hartmann (2018); Jones and Love (2011); Rescorla (2016).

6 Going beyond perception

The algorithm described in this article was originally proposed as a model of perceptual classification in the early visual system (Rao and Ballard, 1999). Understood this way, the algorithm takes raw sensory data as input and classifies these data into basic visual features based on a multi-layered artificial neural network. Subsequently, the predictive coding research programme has gone on to suggest that this algorithm is of much wider significance to neural function. It should be treated, not just as a model of perceptual classification in the early visual system, but potentially as a model of *all* cognitive processing. In its most ambitious form, the claim is that a single, giant ANN, like that pictured in Figure 3, is the computational engine behind all aspects of cognition (Friston, 2010, p. 130). Under Friston's representation-heavy interpretation of the ANN, this would mean that every cognitive process could be viewed as a Bayesian inference over a single, hierarchical, probabilistic generative model (Clark, 2013, pp. 194, 198).

In Sprevak (forthcoming[b]), we saw that one respect in which predictive coding offers a grand, unified theory of cognition is that it claims that a single computational *task* is faced by the brain in every domain of cognition – namely, the task of minimising sensory prediction error. Advocates of predictive coding are often also attracted to the idea that predictive coding offers a grand, unified theory of cognition in another, quite separate, respect. This is that a single type of abstract computational *method* is used by the brain to solve its task – namely, some (more sophisticated) version of the ANN pictured above. On this view, cognition would not only have a single, unified *objective*; the brain would also attempt to reach that objective by a single *means*.

This goal of providing a process-level unification of cognition is clearly aspirational. As we saw in Section 2.5, important elements of predictive coding's algorithmic-level theory remain to be spelled out, and this could be done in many different ways. Furthermore, anyone who wishes to defend the process-level unification would need to show that their algorithmic-level claim successfully models, not only cognitive processes inside the early visual system, but all aspects of cognition (including motor control, decision making, causal inference, executive function, and so on). Whether predictive coding can succeed – at the algorithmic level – as a grand, unified theory depends on the extent it can successfully be applied to cognitive processes outside early vision.

The challenge is structurally similar to that faced by predictive coding at Marr's computational level, as discussed in Section 7 of Sprevak (forthcoming[b]). In that case, the problem was to show that a single *task description* (originally proposed for early vision) can and should describe every problem the brain faces in cognition.

In the present case, the problem is to show that a single *algorithm* (also originally proposed for early vision) can and should describe every cognitive process. Neither challenge is likely to admit of quick or easy resolution; both should be viewed as issues that set the long-term agenda of a research programme. The two challenges are also distinct. Even if the computational-level claim were established – if it were shown that every task the brain faces can and should be described as minimising sensory prediction error – it would still be an open question whether the brain uses the same method to tackle that task in every case. A brain could conceivably use a range of computational techniques to minimise its sensory prediction errors. The ANN might be representative of only one of these. Advocates of predictive coding – even if they agree about the universal scope of the computational-level claim – may disagree about how far, and to exactly which cases, its algorithmic-level proposal should be applied outside the original domain of processing in early vision.[42]

As with its computational-level proposal, predictive coding needs to demonstrate both the *empirical adequacy* and *explanatory superiority* of its algorithmic-level proposal relative to other algorithmic-level proposals. It needs to show that cognitive processing outside early vision – e.g. in motor control, decision making, planning, and so on – can and should be treated as following the steps of the ANN above. The next section describes how predictive coding applies the hierarchical ANN to motor control. More difficult for predictive coding to accommodate at the algorithmic level are *high-level* cognitive processes such as logical reasoning, causal reasoning, inductive inference, long-term planning, and executive control. It is simply not obvious how to get the ANN above (or indeed any ANN) to simulate these processes in a way that achieves human-like levels of performance.[43]

6.1 Example of motor control

At first glance, motor control might seem an unlikely candidate for explanation in terms of the operation of the ANN. The pictured ANN for predictive coding has 'sensory inputs', but no outward-facing connections labelled 'motor outputs'. One might assume that the ANN sits exclusively on the 'perception' side of cognition and that it needs to be connected to some separate computational system that deals with 'motor control'. However, advocates of predictive coding have offered an ingenious proposal for how a single ANN could simultaneously govern both perception and

[42]Clark (2016), Ch. 8 suggests that the brain uses a diverse range of computational methods to minimise its sensory prediction error, including 'quicker, dirtier, more "embodied" ' strategies than the hierarchical algorithm described above (p. 268). He does suggest, however, that the hierarchical ANN is not simply one technique among others, but that it plays a special, structuring role in training, coordinating, and recruiting all the other processes (pp. 252–260).

[43]For discussion of challenges in dealing with modelling these cases, see Clark (2016), pp. 299–300; Roskies and Wood (2017); Williams (2018).

motor control.

This proposal is based around the idea that proprioceptive sensory inputs are connected to muscles via classical reflex arcs. When prediction errors cause these reflex arcs to fire, they automatically produce movement. According to the model, activity in the ANN follows the rules of inference and learning previously described and unfolds with the objective of minimising sensory prediction error. That sensory prediction error will be the precision-weighted average of every component of prediction error across all the cognitive system's sensory channels (visual, auditory, interoceptive, proprioceptive, and so on).[44] In the proposed model of motor control, the proprioceptive input channels play a special role. If the computational system makes a prediction regarding the position of its limbs (e.g. that its left hand is holding a glass of water), then that prediction will have proprioceptive sensory consequences. If the predicted proprioceptive sensory consequences do not match the actual proprioceptive input signals (perhaps because the fingers are not touching the glass), then there will be proprioceptive sensory prediction errors. If the proprioceptive prediction error units are active (and not dialled down by a low precision weighting), then the attached muscular reflex arcs will fire, causing the corresponding muscles to which they are connected to contract. Those muscles will cause the fingers around the glass to close, which will reduce the proprioceptive prediction error, making the previously false proprioceptive prediction true, reducing the signal for that reflex arc to fire further. In this way, sensory prediction errors concerning proprioceptive inputs have the ability to bring about motor action. In this context, a sensory prediction concerning proprioception may function, not as a passive prediction of incoming signals, but like a motor command.[45]

Many questions remain about how this is supposed to work. An important set of worries surround how one should understand the *direction of fit* of predictions inside the ANN. A cognitive agent is normally assumed to have two functionally distinct types of internal state. These may be called *belief-like* and *desire-like* state (or, on a probabilistic model, *credences* and *utilities*).[46] Roughly speaking, the former type of state keeps track of how the world is for the agent; the latter records what the agent aims for in its actions. According to the proposal above, both types of function

[44]Given that the single ANN attempts to minimise sensory prediction error averaged across all sensory channels, one should expect that inside the ANN many prediction values (unit activation levels) and generative model parameters (connection weights) will have a multi-modal character. Advocates of predictive coding suggest that even at early stages within the sensory periphery, one should expect multi-modal processing in the brain (see Clark, 2016, p. 121).

[45]See Adams, Shipp and Friston (2013) for the model. See Clark (2016), Ch. 4 for a helpful informal summary. Friston, FitzGerald et al. (2017) give a worked example that simulates a range of neural responses.

[46]See Russell and Norvig (2010), pp. 50–54.

state map onto a single kind of formal state in the ANN. Predictions serve *both* as the ANN's estimation of how the world is *and* also how it desires the world to be. In the example just described, the ANN predicts that it is holding a glass of water. It maintains this false prediction instead of revising it in light of the countervailing sensory evidence. Because the prediction is clamped, it can be used to drive reflex arcs which activate motor neurons so that the agent will grasp the glass. In a different scenario, the ANN might issue the same prediction – that it is holding a glass of water – but revise that prediction in light of its sensory evidence to reflect the reality that it is not currently holding the glass. In the first case, the prediction has a desire-like direction of fit – it is held fixed and not revised even if the world does not conform. In the second case, the prediction has a belief-like direction of fit – it aims to track the world and should be updated based on incoming sensory signals (either by changing the prediction values or by changing the parameters of its generative model).

How does this dual role for prediction fit into the algorithm described above? What makes a state of the ANN have a belief-like character rather than a desire-like one? In other words, how does the ANN know when to *hold its prediction values and generative model fixed* to effect motor control versus when to *revise its prediction values or generative model* to improve the accuracy of its model? Clark (2016) suggests that there is a delicate balance between the two modes of minimising sensory prediction error during cognitive processing (p. 124). But how does that balance operate? On the basis of what rule does the formal system 'know' whether a prediction should be treated in one way rather than the other?[47] The predictive coding literature offers two main answers to this question, neither of which is entirely satisfactory.

The first suggestion is that the *type* of sensory input explains the direction of fit of the corresponding predictions (Clark, 2016, p. 123; Friston, Mattout and Kilner, 2011). Sensory predictions regarding *proprioceptive* inputs play a special functional role – they have a desire-like direction of fit – because they are connected to motion-causing reflex arcs. There are no counterparts of these connections for other sensory channels – there are no muscular reflex arcs connected to the receptors for vision. Prediction errors for proprioception are thus uniquely wired to have behavioural consequences. This gives predictions for proprioceptive sensory inputs an inherently desire-like character.

While the presence of reflex arcs is a necessary feature of the proposed model of motor control, it is hard to see how it can explain the direction of fit of the ANN's predictions. For sometimes proprioception is used, like vision, in a passive way to acquire sensory information – in such a case, proprioceptive prediction errors

[47]For further discussion of this and associated problems, see Klein (2018); Shea (2013).

result in revisions to the model. When one puts one's hands in one's pocket to feel the shape of an unknown coin to determine whether it is a fifty-pence piece, one's proprioceptive channels are used (primarily) to provide information to update one's model. On the basis of proprioceptive prediction errors, one might adjust one's predictive model to conclude that the coin inside one's pocket is a pound coin, not a fifty-pence piece. In this case, predictions about proprioceptive inputs have a belief-like direction of fit. They are not used in a (vain) desire-like way to force the coin into the shape of a fifty-pence piece in order to make that prediction true. But if proprioceptive sensory inputs may be used in a passive fashion to revise the model, then it cannot be simply the proprioceptive nature of a sensory input that determines whether the associated predictions have a desire-like direction of fit.

The second suggestion – intended to complement the first – is that the degree of *precision weighting* of an error signal explains the direction of fit of its associated predictions (Brown et al., 2013; Clark, 2016, pp. 215–216; Friston, 2009, pp. 299–300; Hohwy, 2013, p. 83). A high degree of precision weighting for a proprioceptive signal indicates that a prediction error is more 'important' than others for the computational system. Desire-like states should have a high degree of precision weighting in order to motivate the system to act on them, and to drown out other prediction errors that might drive behaviour. On this view, the graded distribution of precision weights across the ANN could been seen as reflecting a distribution of more or less belief-like and desire-like states across the system's predictions.

While a high degree of precision weighting is again a necessary feature of the model, it is hard to see how that increased degree of precision weighting could determine direction of fit. For the *importance* of an error (how much the system should prioritise correcting for it relative to other errors), and *manner* that the system should go about correcting for it (changing the model or changing the world) seem to be two separate considerations that can, and often do, come apart. In principle, a cognitive agent might place supreme importance on correcting for a proprioceptive prediction error – it may believe that its future existence depends on it predicting what is in its pocket ('Is it a piece of putty shaped as a cube or shaped as a ball?'). But no matter how highly it rates correcting for that error (e.g. gaining certainty that it is shaped like a cube), it is still an open question whether the system reduces that proprioceptive prediction error by revising the model or by changing the world. It might attempt to gain certainty about the shape of the putty by trying to estimate the current shape as accurately as possible or by forcing the putty into some arbitrary shape that it has chosen to predict. A proprioceptive prediction error – no matter what its degree of precision weighting – might be reduced in either way.

It is currently unclear how a predictive coding algorithm should encode direction of fit. Friston, Schwartenbeck et al. (2013) suggest that one should 'recode' utilities

(which have a desire-like fit) as probabilities.[48] This would allow algorithms for decision making – including those that guide motor control – to be interpreted as a pure 'inference' over probabilities. All aspects of decision making could be viewed as instances of approximate Bayesian inference over a single type of formal state (over probabilities). Classical models of decision making tend to start from the assumption that decision making is a process that operates over two formally distinct states (credences and utilities). The objective of the process is typically characterised as that of maximising expected subjective utility. Bayesian inference counts as only one element in this process – that concerning belief fixation or belief update. It needs to be supplemented by, or embedded into, a larger account which explains how the credences arrived at by Bayesian inference are combined with utilities to generate action.[49]

While it is formally possible to recode utilities as probabilities, it is hard to see how it solves the problem described above. The cognitive system would still need to keep track of whether any particular probability so defined is a belief-like one (a credence) or a desire-like one (a recoded utility). The two have distinct functional roles within the cognitive economy of the agent and they differ in ways that go beyond any formal similarities. A cognitive agent might have a low credence in a specific outcome to which it also assigns a high utility (e.g. purchasing a winning lottery ticket); a high credence in an outcome to which it assigns a low utility (e.g. purchasing a losing lottery ticket); a high credence in an outcome to which it assigns high utility (e.g. purchasing a winning ticket to a lottery that it has secretly rigged); and a low credence in an outcome to which it assigns a low utility (e.g. purchasing a losing ticket to a lottery that it has secretly rigged). Regardless of how one formalises the relevant quantities (as probabilities or not), two independent degrees of freedom are needed to capture the difference between how the system thinks the world is and how it wishes the world to be. If both quantities are defined as probabilities, the system would need to traffic in two, materially distinct kinds of probability. But exactly how these two kinds of probability should be encoded in the numerical states of the ANN, and how the ANN should decide whether to minimise error via active inference (motor activity) or passive inference (belief revision) are unclear.

Explaining direction of fit may require a full algorithmic-level account of the high-level cognitive processes that lie between perception and motor control. One of

[48] More precisely, utilities should be encoded as log likelihoods. The utility of an outcome given a predictive model, $U(o \mid m)$, should be represented by the log of some prior probability of the outcome conditional on the model, $\ln P(o \mid m)$. Encoding utilities as log probabilities preserves the structure of the agent's utility function, and so can preserve the underlying logic of decision making (see Henriksen, 2020).

[49] For a summary of classical approaches to decision making, see Russell and Norvig (2010), Ch. 16.

the intended functions of these processes is to allow belief-like and desire-like states to combine in a rational way to produce action. These processes should take into account a wide, potentially unbounded, range of epistemic and pragmatic considerations to determine the most appropriate action given the context and whether that involves a motor response or a change to belief. However, a detailed algorithmic-level account of high-level central reasoning processes is exactly what ANN-based models currently do not provide.

7 Conclusion

This paper has focused on what predictive coding says at Marr's algorithmic level. The proposed algorithm consists in (i) a multi-layered, hierarchically structured artificial neural network; (ii) a repeated duplex arrangement of prediction and error units within each layer; (iii) each layer functioning so as to minimise prediction error about the state of the layer below. Numerical states of the network may be interpreted, in a further semantic step, as probabilistic representations and the manner in which they are transformed as a version of approximate Bayesian inference. The algorithm described in this paper is representative only of a deliberately simplified version of predictive coding. One should regard it as just a stepping stone on the way to a more elaborate proposal. However, it illustrates certain broad motifs – such as a hierarchically structured ANN, a functional structure of repeated prediction and error units – that one might expect to find in some future, more sophisticated algorithmic-level proposal.

Rao and Ballard (1999) proposed that a hierarchical predictive coding algorithm should be used to model neural responses in the early visual system. Subsequently, the algorithm has been claimed to have much wider applications. In its boldest form, predictive coding claims that a hierarchical predictive coding algorithm governs *all* aspects of neural cognitive processing. During cognition – which according to predictive coding, aims to minimise sensory prediction error – the brain should be assumed to be implementing a giant, hierarchically structured ANN.

As with predictive coding's computational-level proposal, it is natural to wonder what would happen if one were to trim the ambitions of predictive coding's algorithmic-level claim. Perhaps the ANN describes some computational methods used by the brain, but not all. As we saw with the computational-level claim, the extent to which an advocate of predictive coding chooses to limit the scope of their claim will proportionately decrease the view's coding entitlement to provide a grand, unifying theory of cognition. Nevertheless, even if it falls short of being a truly universal theory of all cognitive processes, if it were to accurately model *many* cognitive processes, or if it were to identify important *features* shared by

aspects of cognition that are traditionally thought of as separate and unrelated (e.g. perception and motor control), it could still justify a claim to provide a unification at the algorithmic level, albeit not a version of unification that treats every cognitive process as an instance of the same algorithm.[50] Precisely how much of cognition the algorithm might unify, and the exact form that unification will take, is presently unclear.

Bibliography

Adams, R. A., S. Shipp and K. Friston (2013). "Predictions not commands: active inference in the motor system". In: *Brain Structure & Function* 218, pp. 611–643.

Baker, L. R. (1985). "A farewell to functionalism". In: *Philosophical Studies* 48, pp. 1–13.

Bechtel, W. and A. Abrahamsen (2002). *Connectionism and the Mind: An Introduction to Parallel Processing in Networks*. 2nd ed. Oxford: Oxford University Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York, NY: Springer.

Blalock, D., J. J. G. Ortiz, J. Frankle and J. Guttag (2020). "What is the state of neural network pruning?" arXiv:2003.03033.

Bogacz, R. (2017). "A tutorial on the free-energy framework for modelling perception and learning". In: *Journal of Mathematical Psychology* 76.198–211.

Brown, H., R. A. Adams, I. Parees, M. Edwards and K. Friston (2013). "Active inference, sensory attenuation and illusions". In: *Cognitive Processing* 14, pp. 411–427.

Cisek, P. and J. F. Kalaska (2010). "Neural mechanisms for interacting with a world full of action choices". In: *Annual Review of Neuroscience* 33, pp. 269–298.

Clark, A. (2013). "Whatever next? Predictive brains, situated agents, and the future of cognitive science". In: *Behavioral and Brain Sciences* 36, pp. 181–253.

— (2014). *Mindware: An Introduction to Cognitive Science*. 2nd ed. Oxford: Oxford University Press.

— (2016). *Surfing Uncertainty: Prediction, Action, and the Embodied Mind*. Oxford: Oxford University Press.

[50] See discussion in Clark (2013), p. 200; Clark (2016), pp. 297–300, and versions of qualified unification that an algorithmic-level model might provide discussed in Danks (2014), pp. 175–204.

Colombo, M., L. Elkin and S. Hartmann (2018). "Being realist about Bayes, and the predictive processing theory of mind". In: *The British Journal for the Philosophy of Science*. DOI: 10.1093/bjps/axy059.

Colombo, M. and P. Seriès (2012). "Bayes on the brain—On Bayesian modelling in neuroscience". In: *The British Journal for the Philosophy of Science* 63, pp. 697–723.

Danks, D. (2014). *Unifying the Mind: Cognitive Representations as Graphical Models*. Cambridge, MA: MIT Press.

Dennett, D. C. (1987). "Three kinds of intentional psychology". In: *The Intentional Stance*. Cambridge, MA: MIT Press. Chap. 3.

Feldman, H. and K. Friston (2010). "Attention, uncertainty, and free-energy". In: *Frontiers in Human Neuroscience* 4, pp. 1–23.

FitzGerald, T., R. J. Dolan and K. Friston (2014). "Model averaging, optimal inference, and habit formation". In: *Frontiers in Human Neuroscience* 8, pp. 1–11.

Friston, K. (2003). "Learning and inference in the brain". In: *Neural Networks* 16, pp. 1325–1352.

— (2005). "A theory of cortical responses". In: *Philosophical Transactions of the Royal Society of London, Series B* 360, pp. 815–836.

— (2008). "Hierarchical models in the brain". In: *PLoS Computational Biology* 4, e1000211.

— (2009). "The free-energy principle: a rough guide to the brain?" In: *Trends in Cognitive Sciences* 13, pp. 293–301.

— (2010). "The free-energy principle: A unified brain theory?" In: *Nature Reviews Neuroscience* 11, pp. 127–138.

Friston, K., T. FitzGerald, F. Rigoli, P. Schwartenbeck and G. Pezzulo (2017). "Active inference: A process theory". In: *Neural Computation* 29, pp. 1–49.

Friston, K., J. Kilner and L. Harrison (2006). "A free energy principle for the brain". In: *Journal of Physiology (Paris)* 100, pp. 70–87.

Friston, K., J. Mattout and J. Kilner (2011). "Action understanding and active inference". In: *Biological Cybernetics* 104, pp. 137–160.

Friston, K., T. Parr and B. de Vries (2017). "The graphical brain: Belief propagation and active inference". In: *Network Neuroscience* 1, pp. 381–414.

Friston, K., P. Schwartenbeck, T. FitzGerald, M. Moutoussis, T. Behrens and R. J. Dolan (2013). "The anatomy of choice: active inference and agency". In: *Frontiers in Human Neuroscience* 7, p. 598.

Friston, K. and K. E. Stephan (2007). "Free-energy and the brain". In: *Synthese* 159, pp. 417–458.

Gelman, A. and J. Hill (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge: Cambridge University Press.

Harpur, G. F. (1997). "Low entropy coding with unsupervised neural networks". PhD thesis. University of Cambridge.

Haugeland, J. (1981). "Semantic Engines: An Introduction to Mind Design". In: *Mind Design*. Ed. by J. Haugeland. Cambridge, MA: MIT Press, pp. 1–34.

Henriksen, M. (2020). "Variational free energy and economics optimizing with biases and bounded rationality". In: *Frontiers in Psychology* 11.549187.

Hinton, G. E. and T. J. Sejnowski (1983). "Optimal perceptual inference". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* 448.

— (1986). "Learning and Relearning in Boltzmann Machines". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Ed. by D. E. Rumelhart, J. McClelland and the PDP Research Group. Vol. 1. Cambridge, MA: MIT Press, pp. 282–317.

Hochstein, S. and M. Ahissar (2002). "View from the top: Hierarchies and reverse hierarchies in the visual system". In: *Neuron* 36, pp. 791–804.

Hohwy, J. (2012). "Attention and conscious perception in the hypothesis testing brain". In: *Frontiers in Psychology* 3, pp. 1–14.

— (2013). *The Predictive Mind*. Oxford: Oxford University Press.

Jones, M. and B. C. Love (2011). "Bayesian Fundamentalism or Enlightenment? On the explanatory status and theoretical contributions of Bayesian models of cognition". In: *Behavioral and Brain Sciences* 34, pp. 169–231.

Kadar, I. and O. Ben-Shahar (2012). "A perceptual paradigm and psychophysical evidence for hierarchy in scene gist processing". In: *Journal of Vision* 12, pp. 1–17.

Kanai, R., Y. Komura, S. Shipp and K. Friston (2015). "Cerebral hierarchies: predictive processing, precision and the pulvinar". In: *Philosophical Transactions of the Royal Society of London, Series B* 370, p. 20140169.

Klein, C. (2018). "What do predictive coders want?" In: *Synthese* 195, pp. 2541–2557.

Krotov, D. and J. J. Hopfield (2019). "Unsupervised learning by competing hidden units". In: *Proceedings of the National Academy of Sciences* 116, pp. 7723–7731.

LeCun, Y., Y. Bengio and G. E. Hinton (2015). "Deep learning". In: *Nature* 521, pp. 436–444.

Lee, T. S. and D. Mumford (2003). "Hierarchical Bayesian inference in the visual cortex". In: *Journal of the Optical Society of America* 20, pp. 1434–1448.

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge: Cambridge University Press.

Maloney, L. T. and P. Mamassian (2009). "Bayesian decision theory as a model of human visual perception: Testing Bayesian transfer". In: *Visual Neuroscience* 26, pp. 147–155.

McClelland, J. L. (1998). "Connectionist models of Bayesian infererence". In: *Rational Models of Cognition*. Ed. by M. Oaksford and N. Chater. Oxford University Press, pp. 21–53.

— (2013). "Integrating probabilistic models of perception and interactive neural networks: A historical and tutorial review". In: *Frontiers in Psychology* 4, p. 503.

Mumford, D. (1992). "On the computational architecture of the neocortex: II The role of cortico-cortico loops". In: *Biological Cybernetics* 66, pp. 241–251.

Parr, T., D. Markovic, S. J. Kiebel and K. Friston (2019). "Neuronal message passing using mean-field, Bethe, and marginal approximations". In: *Scientific Reports* 9, p. 1889.

Rao, R. P. N. and D. H. Ballard (1999). "Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects". In: *Nature Neuroscience* 2, pp. 79–87.

Rescorla, M. (2016). "Bayesian sensorimotor psychology". In: *Mind and Language* 31, pp. 3–36.

Roskies, A. L. and C. C. Wood (2017). "Catching the prediction wave in brain science". In: *Analysis* 77, pp. 848–857.

Russell, S. and P. Norvig (2010). *Artificial Intelligence: A Modern Approach*. 3rd ed. Upper Saddle River, NJ: Pearson.

Shea, N. (2013). "Perception versus action: The computations may be the same but the direction of fit differs". In: *Behavioral and Brain Sciences* 36, pp. 228–229.

Spratling, M. W. (2017). "A review of predictive coding algorithms". In: *Brain and Cognition* 112, pp. 92–97.

Sprevak, M. (forthcoming[a]). “Predictive coding I: Introduction”. In: *TBC*.

— (forthcoming[b]). “Predictive coding II: The computational level”. In: *TBC*.

— (forthcoming[c]). “Predictive coding IV: The implementation level”. In: *TBC*.

Williams, D. (2018). “Predictive coding and thought”. In: *Synthese*. DOI: 10.1007/s11229-018-1768-x.