Understanding Predictive Coding as a Second-Order Trust-Region Method

Francesco Innocenti [1] **Ryan Singh** [1] **Christopher L. Buckley** [1 2]

Abstract

Predictive coding (PC) is a brain-inspired local learning algorithm that has recently been suggested to provide advantages over backpropagation (BP) in biologically relevant scenarios. While theoretical work has mainly focused on the conditions under which PC can approximate or equal BP, how PC in its "natural regime" differs from BP is less understood. Here we develop a theory of PC as an adaptive trust-region (TR) method that uses second-order information. We show that the weight update of PC can be interpreted as shifting BP's loss gradient towards a TR direction found by the PC inference dynamics. Our theory suggests that PC should escape saddle points faster than BP, a prediction which we prove in a shallow linear model and support with experiments on deep networks. This work lays a theoretical foundation for understanding other suggested benefits of PC.

1. Introduction

Predictive coding (PC) is a long-standing theory of cortical function belonging to the wider class of bio-inspired local learning algorithms (e.g. Lillicrap et al. 2016; Scellier & Bengio 2017; Ororbia & Mali 2019; Meulemans et al. 2020; Payeur et al. 2021; Dellaferrera & Kreiman 2022; Hinton 2022) which has recently been gaining traction as a biologically plausible alternative to backpropagation (Millidge et al., 2021; 2022a). The basic premise of PC is that the brain is constantly trying to minimise prediction errors.

In recent years, there has been considerable effort attempting to relate PC to BP. Starting with Whittington & Bogacz (2017) showing that PC can approximate the gradients computed by BP on multi-layer perceptrons (MLPs) when the influence of the prior is upweighted relative to the observations, Millidge et al. (2022d) generalised this result to arbitrary computational graphs including convolutional and recurrent neural networks. A variation of PC, in which weights are updated at precisely timed inference steps, was later shown to be equivalent to BP on MLPs (Song et al., 2020), a result further generalised by Salvatori et al. (2021) and Rosenbaum (2022). Finally, Millidge et al. (2022b) unified these and other approximation results as certain equilibrium properties of energy-based models (EBMs).

On the other hand, the ways in which PC, in its natural regime, differ from BP are much less understood. Song et al. (2022) proposed that PC, and EBMs more generally, implement a fundamentally different principle of credit assignment called "prospective configuration". According to this principle, neurons first change their activity to better predict the output target and then update their weights to consolidate that activity pattern. Based on a wide range of empirical results, Song et al. (2022) suggested that PC can outperform BP in biologically realistic tasks including online and continual learning.

Recent work has started to provide a theoretical basis for this principle. For example, Millidge et al. (2022c) showed (i) that in the linear case the PC inference equilibrium can be interpreted as an average of BP's feedforward pass values and the local targets computed by target propagation (TP; Meulemans et al. 2020), and (ii) that any critical point of the PC energy function is also a critical point of the BP loss. In the online (mini-batches of size one) case, Alonso et al. (2022) proved that PC approximates implicit gradient descent under specific rescalings of the layer activations. While we were writing this paper, Alonso et al. (2023) further showed that when this approximation holds, PC is sensitive to Hessian information for small learning rates. Nevertheless, the fundamental relationship between PC and BP still remains to be fully elucidated.

Here, we show that PC can be usefully understood as a form of *adaptive trust-region* (TR) *algorithm that exploits second-order information*. In particular, we show that the inference stage of PC can be thought of as solving a TR problem on the BP loss using a trust region defined by the Fisher information of the generative model. The PC weight update can then be interpreted as shifting the loss gradient

[1]School of Engineering and Informatics, University of Sussex, Brighton, UK [2]VERSES Research Lab, Los Angeles, California, USA. Correspondence to: Francesco Innocenti <F.Innocenti@sussex.ac.uk>.

In *ICML Workshop on Localized Learning (LLW)*, Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).

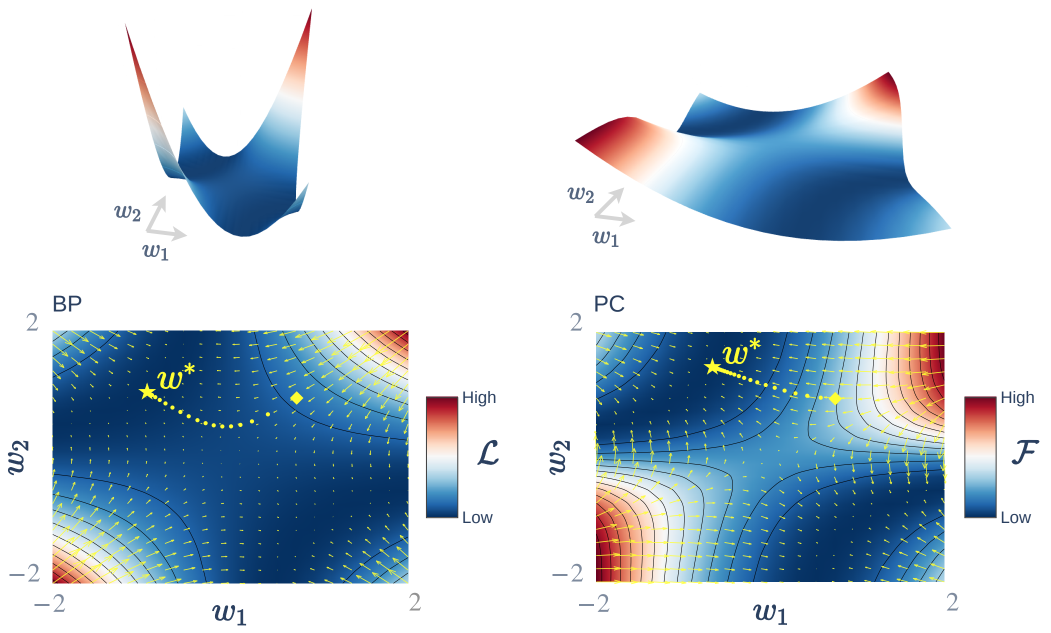

Figure 1. **Landscape geometry and gradient descent dynamics of BP vs PC on a toy network.** Training loss and energy landscapes of an example 1MLP trained with BP (*left*) and PC (*right*), plotted both as surfaces (*top*) and contours with superimposed gradient fields (*bottom*). Surfaces are plotted at the same scale for comparison, and vector fields are standardised for visualisation. The energy landscape of PC is plotted at the inference equilibrium $\mathcal{F}^*$ (see Figure 8 for a visualisation of the landscape inference dynamics).

computed by BP towards the TR inference solution. Our theory suggests that PC should escape saddles faster than BP, a well-known property of TR methods (Conn et al., 2000; Dauphin et al., 2014; Yuan, 2015; Levy, 2016; Murray et al., 2019). We confirm this prediction in a toy model (Section 3) and provide supporting experiments on deep networks (Section 5).

The rest of the paper is structured as follows. After some relevant background on PC and TR methods (Section 2), we build intuition for the differences between PC and BP by studying a toy network (Section 3). We then present our theoretical analysis of PC as a TR method (Section 4), followed by some experiments consistent with our theory (Section 5). We conclude with a discussion of our results as well as directions for future research (Section 6). Theorems, derivations and experiment details are all presented in Appendix A.

2. Preliminaries

2.1. Notation

Matrices and vectors are denoted with capitals A and small bold characters $\mathbf{v}$, respectively. All vectors are column vectors $[\cdot] \in \mathbb{R}^{n\times 1}$. The gradient and Hessian of any twice differentiable objective function $f : \mathbb{R}^d \to \mathbb{R}$ w.r.t. $\mathbf{x}$ are denoted as $\nabla_{\mathbf{x}} f$ and $\nabla^2_{\mathbf{x}} f$, respectively. We will often abbreviate them as $g_f(\mathbf{x})$ and $H_f(\mathbf{x})$, omitting the independent variable when it is clear from the context. The largest and smallest eigenvalues of the Hessian are $\lambda_{max}(H_f)$ and $\lambda_{min}(H_f)$. We denote any critical point of f where $\nabla_{\mathbf{x}} f = 0$ as $\mathbf{x}^*$.

2.2. Predictive coding networks (PCNs)

PCNs are energy-based models that implement a hierarchical Gaussian generative model of the data (Millidge et al., 2021). Each layer (and neuron within a layer) of a PCN can change its activity $\{\mathbf{z}^{(\ell)}\}_{\ell=0}^L$ and weights $\{W^{(\ell)}\}_{\ell=1}^L$ to minimise its local prediction errors. More formally, PCNs minimise an energy function, called the free energy, that can be reduced to a sum of squares across layers

$$\mathcal{F} = \sum_{\ell=1}^{L} \frac{1}{2} \Pi^{(\ell)} \left(\mathbf{z}^{(\ell)} - \phi^{(\ell)} \left(W^{(\ell)} \mathbf{z}^{(\ell-1)} \right) \right)^2 \quad (1)$$

where $\Pi^{(\ell)}$ are layer-wise precision (or inverse covariance) matrices, and $\phi^{(\ell)}$ is some activation function (e.g. ReLU).

Here we consider PCNs trained in a so-called "discriminative" direction which have been shown to recapitulate the performance of BP on small-to-medium machine learning tasks (Millidge et al., 2022a) and suggested to provide additional benefits in more biologically realistic settings (Song et al., 2022), although at a much higher computational inference cost. To train a discriminative PCN, the bottom layer is clamped to some data (e.g. labels), $\mathbf{z}^{(L)} = \mathbf{x}$, while the top layer is fixed to some "prior" (e.g. images), $\mathbf{z}^{(0)} = \mathbf{y}$. The energy (Eq. 1) is minimised in two phases, first w.r.t. the neural activities (inference) and then w.r.t. the weights (learning)

$$\textit{Inference:} \quad \Delta \mathbf{z}^{(\ell)} = -\eta \frac{\partial \mathcal{F}}{\partial \mathbf{z}^{(\ell)}} \tag{2}$$

$$\textit{Learning:} \quad \Delta W^{(\ell)} = -\alpha \frac{\partial \mathcal{F}}{\partial W^{(\ell)}} \tag{3}$$

where η, α are the respective step sizes. In practice, at every step of training, this minimisation is performed by running the inference dynamics to equilibrium $\Delta \mathbf{z}^{(\ell)} \approx 0$, followed by a single weight (e.g. GD) update $\partial \mathcal{F}/\partial W^{(\ell)}|_{\mathbf{z}^*}$. This reflects the intuition that the neural (activity) dynamics operate at a faster timescale than the synaptic (weight) dynamics.

2.3. Trust region (TR) methods

TR methods are often introduced as alternatives to "line-search" algorithms. Whereas line-search techniques such as GD first determine a direction and then a step size (or learning rate), TR methods begin by selecting a step (or region, known as the "trust region") and then optimise for the optimal direction within that region. More formally, given an objective $f(\theta_t)$ we want to minimise, a general TR problem (Conn et al., 2000; Dauphin et al., 2014; Yuan, 2015) can be formulated as

$$\Delta\theta = \underset{\Delta\theta}{\operatorname{argmin}}\, \tilde{f}(\theta_t) \quad \text{s.t.} \quad \Delta\theta^T A \Delta\theta \leq p \tag{4}$$

where $\tilde{f}(\theta_t)$ indicates different Taylor approximations of the objective, and A is some positive-definite matrix defining the norm or geometry of the trust region bounded by some radius p. Specific TR algorithms can therefore be derived by (i) different approximations $\tilde{f}(\theta_t)$, (ii) different geometries induced by A, and by (iii) whether A depends on the current state of the parameters θ_t and is therefore in some sense "adaptive".

Indeed, line-search methods can be seen as special cases of TR problems (Conn et al., 2000). For example, GD can be derived as a TR problem (Eq. 4) assuming a linear approximation of the objective $\tilde{f}(\theta) = f(\theta) + g^T \Delta\theta$ and an Euclidean geometry (or ℓ_2 penalty) given by $A = I$. Solving for the optimal parameter change gives the GD update $\Delta\theta^* = -\alpha g$ where the global learning rate is related to the trust region size $\alpha = \sqrt{p}/||g||$. Note that this formulation also makes explicit that "vanilla" GD is a non-adaptive algorithm (unless some learning rate schedule α_t is employed). Similarly, a damped or trust-region Newton (TRN) method can be obtained by using a quadratic approximation $\tilde{f}(\theta) = f(\theta) + g^T\Delta\theta + \Delta\theta^T H \Delta\theta$, leading to the update $\Delta\theta^* = -(H + \frac{1}{\alpha}I)^{-1}g$.

3. A Toy Model

Here we study a toy model, an MLP with a single linear hidden unit (1MLP) $f(x) = w_2 w_1 x$, which allows us to compare BP and PC exactly. An example of the landscape geometry and gradient descent (GD) dynamics of the 1MLP weights trained by BP and PC is shown in Figure 1 (see Appendix A.1 for details). For BP the landscape is simply the loss landscape, while for PC the landscape is the energy landscape *at the inference equilibrium*.

Even in this minimal setting, we can observe marked qualitative and quantitative differences between the two algorithms. In particular, PC seems to evade the saddle, taking a more direct path to the closest manifold of solutions. This is reflected in the geometry of the equilibrated energy landscape, which shows both a flatter "trap" direction leading to the saddle and a more negatively curved "escape" direction leading to a valley of solutions. Indeed, we provide proof that PC will escape this saddle faster than BP for any non-degenerate problem (Theorem A.3).

More generally, the (stochastic) gradient field of PC seems to be better aligned with the solutions than that of BP. As shown in Figure 2, on average the PC update points much closer and reliably than BP to the optimal direction (i.e. towards the closest solution).

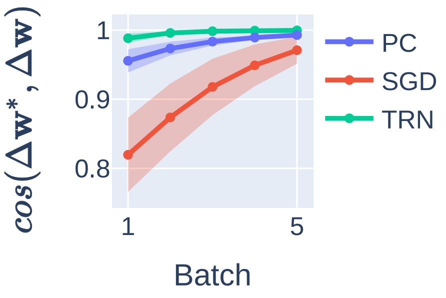

Figure 2. **PC weight update direction is closer to optimal than BP on 1MLPs.** For the first 5 training batches, we plot the mean cosine similarity between the optimal weight direction $\Delta\mathbf{w}^*$ and the update $\Delta\mathbf{w}$ computed by PC (i) $-\nabla_{\mathbf{w}}\mathcal{F}^*$, (ii) BP with SGD, $-\nabla_{\mathbf{w}}\mathcal{L}$, and a (iii) trust-region Newton (TRN) method $-(H + \lambda I)^{-1}\nabla_{\mathbf{w}}\mathcal{L}$ with $\lambda = 2$. Shaded regions indicate the standard error of the mean (SEM) across 10 random weight initialisations.

A second observation is the apparent slowdown of PC near a minimum. In the 1MLP case, we prove that this is because the manifold of minima of the equilibrated energy is *flatter* than that of the loss (Theorem A.4). This also means that during training PC will be more robust to weight perturbations near a minimum (see Figure 6), which could be important in more biological, online settings.

Thus, in this toy example, we show that PC inference effectively reshapes the geometry of the weight landscape such

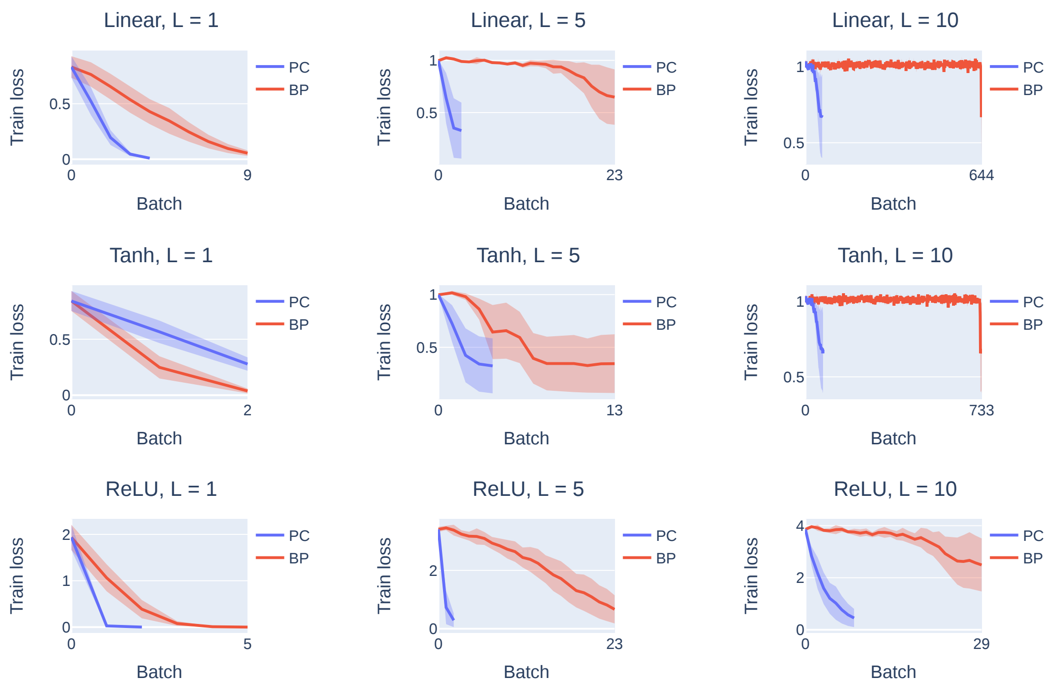

Figure 3. **PC can train deeper chains significantly faster than BP.** Mean training loss of 1D networks (deep chains) trained with BP and PC (see Appendix A.4 for details). Rows and columns indicate different activation functions (Linear, Tanh and ReLU) and depths or number of hidden layers $L = \{1, 5, 10\}$, respectively. Each network type was optimised for learning rate, and training was stopped when the loss stopped decreasing. Shaded regions represent SEM across 3 different initialisations.

that GD (i) escapes the saddle faster and (ii) takes longer to converge close to a minimum while being more robust to perturbations.

4. PC as a Trust-Region Method

Here we show that the inference stage of PC (Eq. 2) solves a TR problem (Eq. 4) on the BP loss in activity space, while the learning stage (Eq. 3) essentially uses the TR solution to shift the GD direction. To make this connection, we perform a second-order Taylor expansion of the free energy (Eq. 1) centred around the feedforward pass values $\mathbf{z}_t$ (see Appendix A.3 for a full derivation):

$$\mathcal{F}(\mathbf{z}) = \mathcal{L}(\mathbf{z}_t) + g_{\mathcal{L}}(\mathbf{z}_t)^T \Delta\mathbf{z} + \frac{1}{2}\Delta\mathbf{z}^T \mathcal{I}(\mathbf{z}_t)\Delta\mathbf{z} + \mathcal{O}(\Delta\mathbf{z}^3) \quad (5)$$

where $\Delta\mathbf{z} = (\mathbf{z} - \mathbf{z}_t)$, $g_{\mathcal{L}}$ is the gradient of the loss, and $\mathcal{I}(\mathbf{z}_t)$ is the Fisher information of the target given by the generative model $p(\mathbf{y}|\mathbf{z})$. This approximation allows us to characterise how (to second order) the PC energy diverges from the BP loss during inference. We observe that Eq. 5 defines a TR problem (Eq. 4) in *activity space* with a linear approximation of the loss plus an *adaptive, second-order* geometry given by $A = \mathcal{I}(\mathbf{z}_t)$. The solution to this TR problem (Eq. 5) is given by

$$\mathbf{z}^* \approx \mathbf{z}_t - \mathcal{I}(\mathbf{z}_t)^{-1} g_{\mathcal{L}}(\mathbf{z}_t) \quad (6)$$

How does this TR solution found by the inference dynamics impact the weight update of PC and so its learning dynamics? Recall that in PC the weights are updated after the activities have (approximately) converged (Section 2.2). We therefore calculate the weight gradient of the energy at the inference solution (see Appendix A.3)

$$\underbrace{\frac{\partial \mathcal{F}}{\partial W}\bigg|_{\mathbf{z}^*}}_{\text{PC direction}} \approx \underbrace{\frac{\partial \mathbf{z_t}}{\partial W}\mathcal{I}(\mathbf{z}_t)^{-1} g_{\mathcal{L}}(\mathbf{z}_t) +}_{\text{TR direction}} \underbrace{g_{\mathcal{L}}(W)}_{\text{BP direction}} \quad (7)$$

where $g_{\mathcal{L}}(W)$ is the loss gradient w.r.t. the weights, and $\frac{\partial \mathbf{z_t}}{\partial W}$ is a change of coordinates from activity to weight space. Thus, we see that the gradient on the equilibrated energy effectively shifts the GD direction of the loss gradient in the direction of the TR inference solution mapped back into weight space. We can then think of the equilibrated energy landscape $\mathcal{F}^*$ as a more "trustworthy" landscape—a landscape which should be easier to gradient descend—than the loss landscape $\mathcal{L}$ when $\mathcal{I}(\mathbf{z}_t)$ provides useful information.

We can gain insight into these GD dynamics (Eq. 7) by considering the contribution of the Fisher information $\mathcal{I}(\mathbf{z}_t)$.

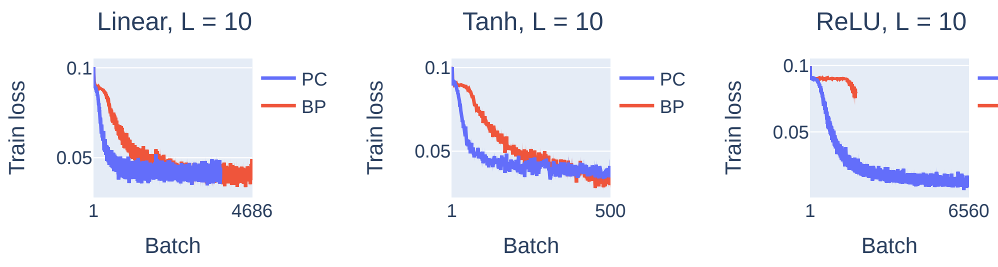

Figure 4. **Faster convergence of PC in deep and wide networks trained on MNIST.** Mean training loss of deep ($L = 10$) and wide ($n = 500$) networks trained to classify MNIST for 3 random initialisation (see Appendix A.4 for details). SEMs are not visible.

For example, in directions of high Fisher information or model curvature—corresponding to directions of high latent variance—the PC weight update will be biased towards the TR solution. TR methods are known to be better at escaping saddles (Conn et al., 2000; Dauphin et al., 2014; Yuan, 2015; Levy, 2016; Murray et al., 2019), which is exactly what we observe in the 1MLP case (Section 3). Indeed, we find that the weight direction taken by PC is much closer to that of a TRN method than BP with GD (see Figure 2). In areas of low Fisher information, on the other hand, PC will tend to look more, but not exactly, like GD, since the curvature will not be zero (unless we are at a critical point where the gradient also vanishes). This is what we seem to observe in the 1MLP case near a minimum, where the model curvature does not seem to provide useful information and slows down convergence. Our theory, then, qualitatively recapitulates the landscape geometry and GD dynamics of PC in the 1MLP case (Section 3).

5. Experiments

Here we report some experiments consistent with the hypothesis, proved for 1MLPs (Theorem A.3) and suggested by our analysis of PC as a TR method (Section 4), that PC escapes saddles faster than BP.

5.1. Deep chains

As a first step, we compared the loss dynamics of BP vs PC on "deep chains" $f(x) = w_L\phi_L(\dots\phi_1(w_1x))$ trained on toy regression tasks (see Appendix A.4 for details). These univariate or one-dimensional networks are the ideal minimal case to test the hypothesis that PC escapes saddles faster than BP since for certain activation functions the number of saddles is exponentially related to the number of hidden layers. Specifically, chains with invertible activations such as linear or Tanh are invariant to $\prod_{\ell=1}^{L} 2^{n_\ell} = 2^L$ "sign-flip", saddle-inducing symmetries (Chen et al., 1993; Bishop & Nasrabadi, 2006), as $n_\ell = 1$ for all layers. Therefore, since (S)GD is known to slow down near saddles (Dauphin et al., 2014; Du et al., 2017; Jin et al., 2021), we should expect BP to slow down with depth, while PC should converge more quickly if it indeed avoids saddles faster.

ReLU breaks this type of weight symmetry (while preserving positive rescalings) and so its associated saddles, but at the cost of introducing flat regions in the landscape. Because curvature information can be useful in such regions, and because PC seems to use second-order information, we also tested ReLU chains. Following previous work (Alonso et al., 2022; Song et al., 2022), we performed a learning rate grid search for each experiment to ensure that any differences were not due to inherently different learning rates between PC and BP (see Appendix A.4). We plot the loss dynamics during training rather than testing because we are interested in optimisation rather than generalisation. Nevertheless, the results do not significantly differ and we report the test losses in Appendix A.4.

Confirming our main prediction, we find that PC can train deeper chains with saddle-inducing activations significantly faster than BP (Figure 3). For linear and Tanh activations, we observe that BP's convergence with SGD significantly slows down with more layers. Indeed, we see the emergence of phase transitions with increased depth, a phenomenon observed in the loss dynamics of deep linear networks (Saxe et al., 2013). We also find significant speed benefits of PC in "saddle-breaking" ReLU chains, suggesting that PC uses second-order information to evade flat regions. Finally, we note that both BP and PC were unable to train very deep chains ($L = 15$), likely due to vanishing/exploding gradients.

5.2. Deep and wide networks

Next, we compared PC and BP on wide, as well as deep, networks $f(\mathbf{x}) = W^{(L)}\phi^{(L)}(\dots\phi^{(1)}(W^{(1)}\mathbf{x}))$. Wide networks introduce even more saddles due to the permutation symmetries between hidden units (Bishop & Nasrabadi, 2006; Brea et al., 2019; Simsek et al., 2021). Swapping any two neurons in same layer (or their incoming and outgoing weights) does not change the output. A layer with n hidden units has $n!$ permutation symmetries, leading a network to have at least a total of $\prod_{\ell=1}^{L} n_\ell! 2^{n_\ell}$ symmetries and associated saddles. We trained 10-layer networks with constant width $n_1 = \cdots = n_L = 500$ to classify MNIST digits (see Appendix A.5) and find speed-ups for PC similar to those observed in deep chains for all of the above activation functions (Figure 4).

6. Discussion

In summary, we showed that PC can be cast as an adaptive trust-region method that exploits second-order information. Our theory suggested that PC should escape saddle points faster than BP, a prediction which we verified in a toy model and supported with experiments on deep networks. This is consistent with previously reported speed-ups of PC over BP (Song et al., 2020; Alonso et al., 2022). For example, Song et al. (2020) found that PC converged significantly faster than BP on a 15-layer, LeakyReLU network of width 64 trained on Fashion-MNIST with Adam. Using batch size 1, Alonso et al. (2022) found significant speed-ups of PC compared to BP for relatively shallower ($L = 3$) and wider ($n = 1024$) ReLU networks trained to classify and reconstruct CIFAR-10. Our theory provides an explanation of these results in terms of faster saddle escaping.

More generally, together these results suggest that the second-order information used by PC contains information about the curvature of the loss landscape. Indeed, Alonso et al. (2023) showed that PC approximates TRN in the case of batch size 1. However, our theory is independent of batch size, and the empirical results suggest that PC exploits second-order information for large mini-batches too. In future work, we aim to better understand the nature of the second-order information used by PC and investigate its convergence benefits in more depth.

While we have shown the potential benefits of PC's inference scheme, its computational cost remains the major limitation of PC, making it orders of magnitude more expensive than BP. Indeed, our results explain this high inference cost by revealing the implicit computation and inversion of a Fisher matrix. However, promising amortised PC schemes have been developed (Tschantz et al., 2022), and future work should investigate whether the benefits of iterative inference can be retained with amortisation.

Although we did not explore this, our theory can recover previous approximation results to BP and TP relying on the ratio of bottom-up vs top-down information (Whittington & Bogacz, 2017; Millidge et al., 2022c). In particular, varying layer-wise precisions (see Section 2.2) can be seen as adjusting different axes of the trust region or, equivalently, per-parameter learning rates (see Figure 9 for an illustration). Indeed, because of the duality between TR and line-search methods (Conn et al., 2000), our theory admits an alternative interpretation of PC as an *adaptive gradient method* (AGM), conceptually similar to state-of-the-art optimisers like Adam (Kingma & Ba, 2014). Notably, AGMs have also been shown to escape saddle points faster than standard SGD (Staib et al., 2019). Studying the relationship between PC and AGMs is therefore an interesting future research direction.

A recent paper by Pogodin et al. (2023) suggests that our theory could be potentially tested against biological data. The authors showed that under certain assumptions the geometry of weight updates can be inferred from the weight distributions and suggested that an Euclidean geometry, as defined by standard GD, is inconsistent with the empirically observed log-normal distributions of synaptic weights. This is in line with our result that PC uses a non-Euclidean (natural) geometry with the Fisher information as the metric. To distinguish between different non-Euclidean geometries, however, experimental data both before and after learning is needed, since different geometries can lead to the same post-learning distribution depending on the pre-learning distribution (Pogodin et al., 2023).

Related, our study has implications for whether the brain may approximate GD. It seems to be widely accepted that the brain estimates gradients on some objective or loss function (Marblestone et al., 2016; Richards et al., 2019; Lillicrap et al., 2020; Hennig et al., 2021; Richards & Kording, 2023), and much (if not most) work trying develop bio-plausible algorithms assumes (either explicitly or implicitly) that the aim is to approximate BP and GD. Richards & Kording (2023) suggest that this claim could be experimentally tested by looking at how synaptic changes following learning on some task correlate with the true gradient of some loss for that task. Whether or not PC is a good model of learning in the brain, our results show that *first-order, gradient updates on a sum of local losses* (in this case the free energy) *can look like second-order updates on a global loss*[1]. This raises the possibility that the brain could use curvature information of the loss by still doing GD, just on a sum of local objectives. If so, synaptic changes may not correlate with the loss gradient and should also be compared with second-order updates.

Finally, our theory can be seen as an important step in providing a more solid theoretical footing to the principle of "prospective configuration" (Song et al., 2020) and its associated empirical benefits. We are excited by the possibility of extending this framework to explain, and perhaps uncover, other advantages and disadvantages of PC, such as robustness to small batch sizes and reduced weight interference.

[1]Furthermore, our results suggest that GD on a natural geometry approximates TRN on an Euclidean geometry.

Code availability

Code to reproduce all results and plots will be made publicly available on GitHub upon publication of this work.

Acknowledgements

We would like to thank Dhruva V. Raman for helpful discussions in the early stages of this project. F. I. is funded by the Sussex Neuroscience 4-year PhD Programme. R. S. was supported by the Leverhulme Trust through the be.AI Doctoral Scholarship Programme in biomimetic embodied AI. C. L. B. was partially supported by the European Innovation Council (EIC) Pathfinder Challenges, Project METATOOL with Grant Agreement (ID: 101070940).

References

Alonso, N., Millidge, B., Krichmar, J., and Neftci, E. O. A theoretical framework for inference learning. *Advances in Neural Information Processing Systems*, 35:37335–37348, 2022.

Alonso, N., Krichmar, J., and Neftci, E. Understanding and improving optimization in predictive coding networks. *arXiv preprint arXiv:2305.13562*, 2023.

Anandkumar, A. and Ge, R. Efficient approaches for escaping higher order saddle points in non-convex optimization. In *Conference on learning theory*, pp. 81–102. PMLR, 2016.

Bishop, C. M. and Nasrabadi, N. M. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Brea, J., Simsek, B., Illing, B., and Gerstner, W. Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *arXiv preprint arXiv:1907.02911*, 2019.

Chen, A. M., Lu, H.-m., and Hecht-Nielsen, R. On the geometry of feedforward neural network error surfaces. *Neural computation*, 5(6):910–927, 1993.

Conn, A. R., Gould, N. I., and Toint, P. L. *Trust region methods*. SIAM, 2000.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in neural information processing systems*, 27, 2014.

Dellaferrera, G. and Kreiman, G. Error-driven input modulation: solving the credit assignment problem without a backward pass. In *International Conference on Machine Learning*, pp. 4937–4955. PMLR, 2022.

Du, S. S., Jin, C., Lee, J. D., Jordan, M. I., Singh, A., and Poczos, B. Gradient descent can take exponential time to escape saddle points. *Advances in neural information processing systems*, 30, 2017.

Ge, R., Huang, F., Jin, C., and Yuan, Y. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on learning theory*, pp. 797–842. PMLR, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Hennig, J. A., Oby, E. R., Losey, D. M., Batista, A. P., Byron, M. Y., and Chase, S. M. How learning unfolds in the brain: toward an optimization view. *Neuron*, 109(23): 3720–3735, 2021.

Hinton, G. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. How to escape saddle points efficiently. In *International conference on machine learning*, pp. 1724–1732. PMLR, 2017.

Jin, C., Netrapalli, P., Ge, R., Kakade, S. M., and Jordan, M. I. On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points. *Journal of the ACM (JACM)*, 68(2):1–29, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. Gradient descent only converges to minimizers. In *Conference on learning theory*, pp. 1246–1257. PMLR, 2016.

Levy, K. Y. The power of normalization: Faster evasion of saddle points. *arXiv preprint arXiv:1611.04831*, 2016.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):13276, 2016.

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., and Hinton, G. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

Marblestone, A. H., Wayne, G., and Kording, K. P. Toward an integration of deep learning and neuroscience. *Frontiers in computational neuroscience*, 10:94, 2016.

Meulemans, A., Carzaniga, F., Suykens, J., Sacramento, J., and Grewe, B. F. A theoretical framework for target propagation. *Advances in Neural Information Processing Systems*, 33:20024–20036, 2020.

Millidge, B., Seth, A., and Buckley, C. L. Predictive coding: a theoretical and experimental review. *arXiv preprint arXiv:2107.12979*, 2021.

Millidge, B., Salvatori, T., Song, Y., Bogacz, R., and Lukasiewicz, T. Predictive coding: towards a future of deep learning beyond backpropagation? *arXiv preprint arXiv:2202.09467*, 2022a.

Millidge, B., Song, Y., Salvatori, T., Lukasiewicz, T., and Bogacz, R. Backpropagation at the infinitesimal inference limit of energy-based models: Unifying predictive coding, equilibrium propagation, and contrastive hebbian learning. *arXiv preprint arXiv:2206.02629*, 2022b.

Millidge, B., Song, Y., Salvatori, T., Lukasiewicz, T., and Bogacz, R. A theoretical framework for inference and learning in predictive coding networks. *arXiv preprint arXiv:2207.12316*, 2022c.

Millidge, B., Tschantz, A., and Buckley, C. L. Predictive coding approximates backprop along arbitrary computation graphs. *Neural Computation*, 34(6):1329–1368, 2022d.

Murray, R., Swenson, B., and Kar, S. Revisiting normalized gradient descent: Fast evasion of saddle points. *IEEE Transactions on Automatic Control*, 64(11):4818–4824, 2019.

Ororbia, A. G. and Mali, A. Biologically motivated algorithms for propagating local target representations. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4651–4658, 2019.

Payeur, A., Guerguiev, J., Zenke, F., Richards, B. A., and Naud, R. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature neuroscience*, 24(7):1010–1019, 2021.

Pogodin, R., Cornford, J., Ghosh, A., Gidel, G., Lajoie, G., and Richards, B. Synaptic weight distributions depend on the geometry of plasticity. *arXiv preprint arXiv:2305.19394*, 2023.

Richards, B. A. and Kording, K. P. The study of plasticity has always been about gradients. *The Journal of Physiology*, 2023.

Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R. P., de Berker, A., Ganguli, S., et al. A deep learning framework for neuroscience. *Nature neuroscience*, 22(11): 1761–1770, 2019.

Rosenbaum, R. On the relationship between predictive coding and backpropagation. *Plos one*, 17(3):e0266102, 2022.

Salvatori, T., Song, Y., Lukasiewicz, T., Bogacz, R., and Xu, Z. Predictive coding can do exact backpropagation on convolutional and recurrent neural networks. *arXiv preprint arXiv:2103.03725*, 2021.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Scellier, B. and Bengio, Y. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

Simsek, B., Ged, F., Jacot, A., Spadaro, F., Hongler, C., Gerstner, W., and Brea, J. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In *International Conference on Machine Learning*, pp. 9722–9732. PMLR, 2021.

Song, Y., Lukasiewicz, T., Xu, Z., and Bogacz, R. Can the brain do backpropagation?—exact implementation of backpropagation in predictive coding networks. *Advances in neural information processing systems*, 33: 22566–22579, 2020.

Song, Y., Millidge, B., Salvatori, T., Lukasiewicz, T., Xu, Z., and Bogacz, R. Inferring neural activity before plasticity: A foundation for learning beyond backpropagation. *bioRxiv*, pp. 2022–05, 2022.

Staib, M., Reddi, S., Kale, S., Kumar, S., and Sra, S. Escaping saddle points with adaptive gradient methods. In *International Conference on Machine Learning*, pp. 5956–5965. PMLR, 2019.

Tschantz, A., Millidge, B., Seth, A. K., and Buckley, C. L. Hybrid predictive coding: Inferring, fast and slow. *arXiv preprint arXiv:2204.02169*, 2022.

Whittington, J. C. and Bogacz, R. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262, 2017.

Yuan, Y.-x. Recent advances in trust region algorithms. *Mathematical Programming*, 151:249–281, 2015.

A. Appendix

A.1. Toy model experiments

1MLPs were trained with BP and PC to predict a simple linear function $y = -x$ where $x \sim \mathcal{N}(1, 0.1)$. We used a uniform weight initialisation $\mathbf{w} \sim \mathcal{U}(-1, 1)$ and SGD with batch size 64 and learning rate $\alpha = 0.2$ to clearly visualise the algorithms' learning trajectory. Training was stopped when the test loss reached a small tolerance $\mathcal{L}_{test} < 0.001$. For PC, standard Euler integration was used to solve the inference dynamics (Eq. 2), with a feedforward pass initialisation, step size $\eta = 0.1$, and $T = 20$ inference iterations (which were sufficient to reach equilibrium). Precisions were set to one, $\Pi^{(1)} = \Pi^{(2)} = 1$.

In Figure 2, we computed the cosine similarity between the optimal weight direction $\Delta\mathbf{w}^* = (w_1^* - w_1, w_2^* - w_2)$ and the algorithms' GD update at a given point $\Delta\mathbf{w} = -\nabla_\mathbf{w} f$

$$cos(\Delta\mathbf{w}^*, \Delta\mathbf{w}) = \frac{\langle \Delta\mathbf{w}^*, \Delta\mathbf{w} \rangle}{\|\Delta\mathbf{w}^*\| \|\Delta\mathbf{w}\|} \tag{8}$$

which is simply a normalised dot product. To calculate the optimal direction, at each training mini-batch we solved for the shortest (Euclidean) distance from the current iterate $\mathbf{w} = (w_1, w_2)$ to the manifold of solutions $\mathbf{w}^* = (w_1^*, \frac{y}{w_1^* x}) = (w_1^*, -\frac{1}{w_1^*})$

$$D = \sqrt{\left(-\frac{1}{w_1^*} - w_2\right)^2 + (w_1^* - w_1)^2} \tag{9}$$

To minimise this distance, we set the partial derivative of the distance w.r.t. the optimal weight w_1^* to zero

$$\frac{\partial D}{\partial w_1^*} = \frac{(w_1^*)^4 - (w_1^*)^3 w_1 - w_1^* w_2 - 1}{(w_1^*)^3 \sqrt{\left(-\frac{1}{w_1^*} - w_2\right)^2 + (w_1^* - w_1)^2}} = 0 \tag{10}$$

Finding the roots of this derivative means solving for the quartic polynomial in the numerator, for which we used numpy.

A.2. Toy model proofs

Here we present our two theorems on 1MLPs, showing (i) that PC escapes the saddle point induced by the symmetry in the 1MLP weights faster than BP, and (ii) that the 1MLP mimina of the equilibrated energy are flatter that those of the loss.

Definition A.1. *1MLP problem.* We define a 1MLP problem as any non-degenerate linear function of the form $y = mx, x, y \neq 0$ that can in principle be learned by a 1MLP $f(x) = w_2 w_1 x$ where x, y indicate the input and output to the network, respectively.

Definition A.2. *(Strict) saddle.* A critical point $\mathbf{w}^*$ of $f(\mathbf{w})$ where $\nabla f(\mathbf{w}^*) = 0$ is a saddle if the Hessian at that point has at least one positive and one negative eigenvalue, $\lambda_{max}(\nabla^2 f(\mathbf{w}^*)) > 0$, $\lambda_{min}(\nabla^2 f(\mathbf{w}^*)) < 0$. In the literature, these critical points are known as strict or non-degenerate saddles (Ge et al., 2015; Anandkumar & Ge, 2016; Jin et al., 2017).

Consider the BP mean squared error loss and PC energy (Eq. 1) associated with a 1MLP problem (Definition A.1)

$$\mathcal{L} = \frac{1}{2}(y - w_2 w_1 x)^2 \tag{11}$$

$$\mathcal{F} = \frac{1}{2}(z - w_1 x)^2 + \frac{1}{2}(y - w_2 z)^2 \tag{12}$$

where z indicates the value of the hidden unit or latent in PC (which is free to vary). Without loss of generality, we assume a single input-output pair. Note that we can change the sign of the weights without changing the objectives, $f(\mathbf{w}) = f(-\mathbf{w})$. This is known as a "sign-flip symmetry" and induces a saddle in the weight landscape (Chen et al., 1993; Bishop & Nasrabadi, 2006). Now recall that we are interested in how PC inference (Eq. 2) affects the weight update (Eq. 3). In the linear case, we can analytically solve for the inference equilibrium $\frac{\partial \mathcal{F}}{\partial z} = 0, z^* = \frac{w_1 x + w_2 y}{1 + w_2^2}$ and evaluate the energy at this fixed point

$$\mathcal{F}^* = \frac{\mathcal{L}}{1 + w_2^2} \tag{13}$$

The origin $\mathbf{w}^* = (0, 0)$ is critical point of both the loss and the equilibrated energy since their gradient is zero, $\nabla_\mathbf{w} \mathcal{L}(\mathbf{w}^*) = \nabla_\mathbf{w} \mathcal{F}^*(\mathbf{w}^*) = \mathbf{0}$. To confirm that this point is a saddle (Definition A.2), we look at the Hessians

$$H_\mathcal{L}(\mathbf{w}^*) = \begin{bmatrix} 0 & -xy \\ -xy & 0 \end{bmatrix} \tag{14}$$

$$H_{\mathcal{F}^*}(\mathbf{w}^*) = \begin{bmatrix} 0 & -xy \\ -xy & -y^2 \end{bmatrix} \tag{15}$$

and see that indeed they both have positive and negative eigenvalues $\lambda(H_\mathcal{L}) = \pm xy$, $\lambda(H_{\mathcal{F}^*}) = \frac{1}{2}(-y^2 \pm y\sqrt{4x^2 + y^2})$. Crucially, however, the eigenvalues of the energy are smaller than those of the loss

$$\begin{cases} \lambda_{max}(H_{\mathcal{F}^*}) < \lambda_{max}(H_\mathcal{L}) \\ \lambda_{min}(H_{\mathcal{F}^*}) < \lambda_{min}(H_\mathcal{L}) \end{cases} \tag{16}$$

which can be shown by using the fact that the square root of a sum is always smaller than the sum of the square roots, $\sqrt{a^2 + b^2} < \sqrt{a^2} + \sqrt{b^2}$ for $a, b \neq 0$. This result is sufficient to prove that PC will escape the saddle faster than BP, since the near-saddle (S)GD dynamics are controlled by the local curvature. To see this, consider a second-order Taylor expansion of objective f around the saddle

$$f(\mathbf{w}^* + \Delta\mathbf{w}) \approx f(\mathbf{w}^*) + \frac{1}{2}\Delta\mathbf{w}^T H_f \Delta\mathbf{w} \tag{17}$$

where the gradient vanishes. As shown by Lee et al. (2016), taking a gradient descent step of size α from this approximation leads to the following recursive update

$$\begin{aligned}\mathbf{w}_{t+1} &= (I - \alpha H_f)^{t+1}\mathbf{w}_0 \\ &= \sum_{i=1}^{n_w}(1-\alpha\lambda_i)^{t+1}\langle e_i \mathbf{w}_0\rangle e_i \end{aligned} \tag{18}$$

where $\mathbf{w}_0 = (\mathbf{w}^* + \Delta\mathbf{w})$, $n_w = 2$ is the number of parameters, and $\{\lambda_i\}_i^{n_w}$ are the Hessian eigenvalues with corresponding eigenvectors $\{e_i\}_i^{n_w}$. We see that (S)GD will be attracted to, and repelled from, the saddle depending on the degree of curvature along those directions. Because the equilibrated energy has smaller Hessian eigenvalues than the loss at the saddle (Eq. 16), PC will be simultaneously less attracted to and more repelled from it than BP. In dynamical systems terms, the energy saddle turns out to be more "unstable"—and therefore easier to escape—than the loss saddle.

Theorem A.3. *Given any 1MLP problem (Definition A.1) which induces a saddle (Definition A.2) at the origin in weight space, (S)GD on the equilibrated PC energy (Eq. 13) will escape the saddle faster than on the quadratic BP loss (Eq. 11).*

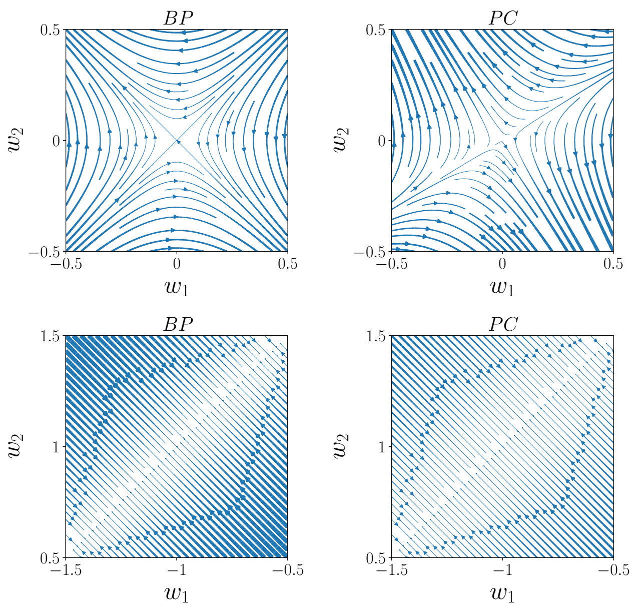

Figure 5. **Gradient flow of BP vs PC near different critical points on a toy network.** Continuous-time GD dynamics in the vicinity of the saddle (*top*) and an example minimum (*bottom*) of a 1MLP trained with BP and PC on the same regression problem illustrated in Figure 1. Comparing with the discrete and stochastic gradient fields (Figure 1), we observe that the continuous dynamics are a good approximation.

We can also see this by taking the continuous limit of the near-saddle GD dynamics $\alpha \to 0$ (Eq. 18, Figure 5), leading to the linear ODE system

$$\dot{\mathbf{w}}(t) = -H_f\mathbf{w}(t) \tag{19}$$

with solution $\mathbf{w}(t) = Qe^{\Lambda t}Q^T\mathbf{w}(0)$ and initial condition $\mathbf{w}(0) = (\mathbf{w}^* + \Delta\mathbf{w})$.

Using the same approach, we can also show that any 1MLP global minimum[2] of the equilibriated energy is *flatter* than any corresponding minimum of the loss. Formally, the Hessian eigenvalues of equilibrated energy will also be smaller than those of the loss at any minimum. Because 1MLPs already pose an overparameterised (underdetermined) problem, there is no unique solution but rather a manifold. That is, for any value of one weight, there exists only one optimal value of the other, e.g. $\mathbf{w}^* = (\frac{y}{w_2 x}, w_2)$. These are also all critical points of both the loss and energy, since their gradient is zero $\nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^*) = \nabla_\mathbf{w}\mathcal{F}^*(\mathbf{w}^*) = \mathbf{0}$. To verify that this is a manifold of minima, as before we look at the Hessian and see that they both have one zero eigenvalue $\lambda_{min}(H_\mathcal{L}) = \lambda_{min}(H_{\mathcal{F}^*}) = 0$ and one positive eigenvalue $\lambda_{max}(H_\mathcal{L}) = \frac{w_2^4x^2+y^2}{w_2^2}$ and $\lambda_{max}(H_{\mathcal{F}^*}) = \frac{w_2^4x^2+y^2}{w_2^2(1+w_2^2)}$. It is straightforward to see that the positive curvature of the energy is smaller than that of the loss, $\lambda_{max}(H_{\mathcal{F}^*}) < \lambda_{max}(H_\mathcal{L})$.

Theorem A.4. *Given any 1MLP problem (Definition A.1), the minima of the equilibrated PC energy (Eq. 13) are flatter than the corresponding minima of the quadratic BP loss (Eq. 11).*

Performing the same GD analysis as above (Eqs. 17, 18) around this manifold of minima leads to the conclusion that GD will converge slower than BP in the vicinity of a minimum but also be more robust to random weight perturbations where the local approximation holds (Figure 6). As before we can make a similar argument for the continuous case, illustrated in Figure 5.

A.3. Derivations of theoretical results

Free energy expansion. Recall the free energy is the sum of local prediction errors

$$\mathcal{F} = \frac{1}{2}(\mathbf{y} - \mathbf{z}^{(L)})^2 + \sum_{\ell=1}^{L-1}\frac{1}{2}\Pi^{(\ell)}\left(\mathbf{z}^{(\ell)} - \phi^{(\ell)}(W^{(\ell)}\mathbf{z}^{(\ell-1)})\right)^2 \tag{20}$$

Let the feedforward activations be defined as $\{\mathbf{z}_t^{(\ell)} = \phi^{(\ell)}(\dots\phi^{(1)}(W^{(1)}\mathbf{z}_t^{(0)}))\}_{\ell=1}^L$ and further we define the difference between $\Delta\mathbf{z} = (\mathbf{z} - \mathbf{z_t})$. Performing a Taylor

[2]It is easy to show that these minima are global since saddles are the only other type of critical point in this problem.

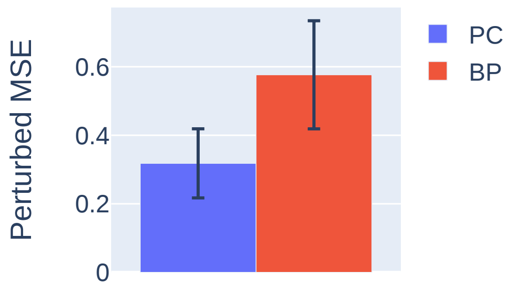

Figure 6. **PC is more robust to near-minimum weight perturbations than BP on toy network.** Mean squared error (MSE) between output target and weight-perturbed prediction $(y - \hat{y})^2$ of BP and PC trained on the same 1MLP problem illustrated in Figure 1. Weights were perturbed with i.i.d. Gaussian noise $\xi \sim \mathcal{N}(0, 0.5)$. Error bars indicate SEM across 10 different seeds.

expansion we see,

$$\mathcal{F}(\mathbf{z}) = \mathcal{F}(\mathbf{z_t}) + J_{\mathcal{F}}^T(\mathbf{z_t})\Delta\mathbf{z} + \frac{1}{2}\Delta\mathbf{z}^T H_{\mathcal{F}}(\mathbf{z_t})\Delta\mathbf{z} + \mathcal{O}(\Delta\mathbf{z}^3) \tag{21}$$

and observe the following: $\mathcal{F}(\mathbf{z_t}) = \mathcal{L}(\mathbf{z_t})$, $J_{\mathcal{F}}^T(\mathbf{z_t}) = g_{\mathcal{L}}(\mathbf{z_t})$, since in both cases the terms in the sum collapse at the feedforward values. Further, $H_{\mathcal{F}}(\mathbf{z_t}) \approx -\frac{\partial^2 \mathbb{E}_{y,x} \ln p(y,\mathbf{z},x)}{\partial \mathbf{z}^2} |_{\mathbf{z_t}} = \mathcal{I}(\mathbf{z_t})$ can be seen as the Fisher information of the feedforward values, w.r.t. to the model p. Hence:

$$\mathcal{F}(\mathbf{z}) = \mathcal{L}(\mathbf{z_t}) + g_{\mathcal{L}}^T(\mathbf{z_t})\Delta\mathbf{z} + \frac{1}{2}\Delta\mathbf{z}^T \mathcal{I}(\mathbf{z_t})\Delta\mathbf{z} + \mathcal{O}(\Delta\mathbf{z}^3) \tag{22}$$

Approximate inference solution. If we assume $O(\Delta\mathbf{z}^3)$ is a small contribution, we can approximate the inference equilibrium by finding the stationary point of the second-order expansion, yielding

$$\mathbf{z}^* \approx \mathbf{z_t} - \mathcal{I}(\mathbf{z_t})^{-1} g_{\mathcal{L}}(\mathbf{z_t}) \tag{23}$$

Approximate weight update. After the activities converge (at inference equilibrium), PC takes a gradient step on the free energy (Section 2.2). In order to find this we first calculate $\frac{\partial \mathcal{F}}{\partial W} = \frac{\partial \mathbf{z_t}}{\partial W}\frac{\partial \mathcal{F}}{\partial \mathbf{z_t}}$:

$$\frac{\partial \mathcal{F}}{\partial W} = \frac{\partial \mathbf{z_t}}{\partial W}\left[-\Delta\mathbf{z} - \mathcal{I}(\mathbf{z_t})^T\Delta\mathbf{z} + O(\Delta\mathbf{z}^2)\right] \tag{24}$$

Finally, plugging in the equilibrium value $\mathbf{z}^*$

$$\left.\frac{\partial \mathcal{F}}{\partial W}\right|_{\mathbf{z}^*} \approx \frac{\partial \mathbf{z_t}}{\partial W}\left[\mathcal{I}(\mathbf{z_t})^{-1} g_{\mathcal{L}}(\mathbf{z_t}) + g_{\mathcal{L}}(\mathbf{z_t})\right] \approx \frac{\partial \mathbf{z_t}}{\partial W}\mathcal{I}(\mathbf{z_t})^{-1} g_{\mathcal{L}}(\mathbf{z_t}) + g_{\mathcal{L}}(W) \tag{25}$$

A.4. Deep chain experiments

We trained deep chains using SGD with batch size 64. To control for the learning rate α, we peformed a grid search over $lrs = \{1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 1e^{-0}\}$ and compared the loss dynamics for the learning rate with the lowest training loss for each algorithm. Linear and Tanh chains were trained on the same regression task used for the toy models, $y = -x$ with $x \sim \mathcal{N}(1, 0.1)$, and initialised with PyTorch default's He initialisation (He et al., 2015). ReLU chains were instead trained to predict a positive linear function $y = 2x$ to avoid mapping to zero. For the same reason, weights were initialised from a uniform distribution with positive interval $\mathbf{w} \sim \mathcal{U}(0.5, 1)$.

We recorded training and test loss on every batch from initialisation and stopped training if either (i) the training loss on the current batch was smaller than $\mathcal{L}_{train} < 0.01$, (ii) the average training loss (estimated every 500 batches) did not decrease, or (iii) the loss diverged to infinity (typically because of high learning rates). For PC, we used an inference schedule similar to that of Song et al. (2022), halving the step size $dt = 0.1$ up to two times with maximum $T = 500$ training iterations.

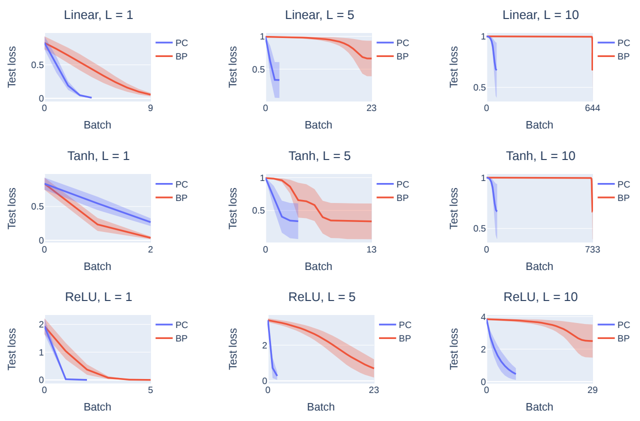

Figure 7. **Mean test losses for the deep chain experiments described in Section 5.**

A.5. Experiments on deep and wide networks

10-layer networks of width $n = 500$ were trained on MNIST with SGD, batch size 64, and the same learning rate grid search used for deep chains. As standard, the MNIST images were normalised. Training was stopped if the training loss did not decrease from the previous epoch or diverged to infinity. For PC, all details were the same as deep chains (Appendix A.4) except for $T = 1000$ maximum inference iterations, used to ensure that any failure to train highly overparameterised networks was not due to insufficient inference.

A.6. Supplementary figures

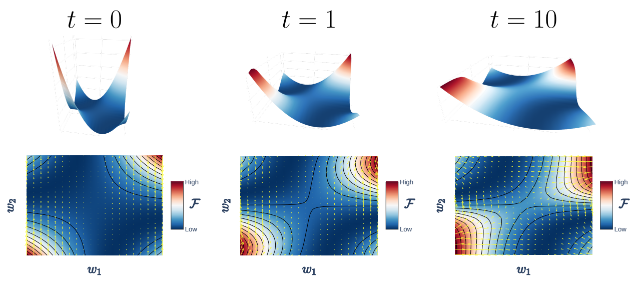

Figure 8. **Inference dynamics of PC energy landscape of a toy network.** Evolution of the free energy landscape as a function of the 1MLP weights over inference, plotted at initialisation ($t = 0$), the first inference step ($t = 1$), and equilibrium ($t = 10$) for the same problem illustrated in Figure 1.

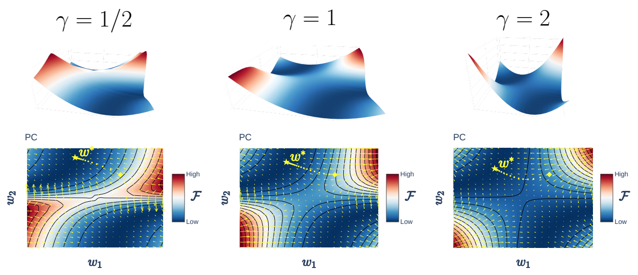

Figure 9. **Equilibrated PC energy landscape as a function of the ratio of bottom-up vs top-down information $\gamma = \Pi_1/\Pi_2$ in a toy network.** Varying γ can be seen as adjusting the size of the trust region or per-parameter learning rates. Increasing the relative influence of the input ($\gamma = 2$) recovers BP, while increasing that of the output ($\gamma = 1/2$) recovers TP.