Neuroscience

Neuronal Dynamics. Real nervous systems can be understood as nonlinear dynamical systems. Their basic computational units are nonlinear dynamical systems themselves and exhibit a variety of dynamics [Izhikevich]. When coupled, novel dynamical phenomena emerge, e.g. synchronization of neural populations and oscillation, as shown in large scale simulations [Izhikevich]. Single neurons or populations can be characterized in terms of their attractor dynamics and phase transitions between different dynamical regimes.

Population Geometry. Neuronal activity lives in a high dimensional 'neural state space' where coordinates represent the activities of individual neurons. The availability of large scale recordings often preclude tuning-based or phase space analyses inconclusive or inexpressive. Neural activity often lies on lower-dimensional manifolds that are embedded in neural state space. Population geometry analyzes the topological structure of the point-cloud manifold to characterize the underlying computation implemented by the neural population. The intrinsic geometry of representation or computation can be uncovered by nonlinear dimensionality reduction techniques. Popular techniques such as Isomap [40], LLE [41], tSNE [42], MDS [43], PHATE [44] and UMAP [45] assume that the manifold structure is simple, whereas e.g. Spline Parameterization for Unsupervised Decoding (Chaudhuri et al. [46]), Manifold Inference from Neural Dynamics (Low et al. [47]) are more complicated yet do not suffer constrains by simplicity assumptions).
Computation and cognition are often characterized by geometric constraints imposed on the structure of the manifold, or by constraints imposed on the dynamics of the neural activity or probability flow on the manifold.

Neuromodulation. Across large parts of the nervous system, upstream or higher-order neuronal activity alters individual neural output and global network cluster activity in downstream populations by using neuromodulatory transmitters to shift relative synaptic weight contributions. Neuromodulation operates on multiple timescales to shape tonic and phasic activity, and induce as well as control neuroplasticity. Neuromodulatory activity often exerts downstream control by using temporally sparse and highly selective modulatory signals that transiently push downstream activity in a different dynamical regime until relaxation.

Bayesian Brain Hypothesis. Perception is not only a transduction of sensory states into internal representations, rather it is an inferential process that combines prior information about the causes of sensations with sensory stimuli. As such it is an active constructive process in which sensations are used to confirm or disconfirm hypotheses about how they were generated.
Bayesian inference thus requires a generative model, represented as a joint probability $P(x, y)$, where x are the hidden representations of environment states h and y are the observations. When the joint probability is decomposed into a prior $P(x)$ and a likelihood $P(x \mid y)$, the posterior probability over hidden states can be computed by Bayes rule.
The process of inference thus maximizes the negative log marginal probability of observations $-\log P(y)$, a measure of how surprising an observation is under the model P, constrained by the dissimilarity between prior and posterior in terms of the KL Divergence $\mathbb{E}_{P(x \mid y)}[\log P(x \mid y) - \log P(x)]$.
Bayesian inference minimizes the variational free energy, taking into account the prior plausibility of x and the uncertainty of the estimation. Due to the limited computational resources of real nervous systems, exact Bayesian inference is intractable, and with it the guarantee of (subjective) optimality.

Active Inference. A biological organism cannot only build an internal model of the latent causes x of its perceptions, but it can actively change the data source and the way in which data are generated, and can thus engage in reciprocal exchange with the environment. The theory of Active Inference states that perception and action have the same inferential nature, and both optimize the same objective: variational free energy (surprise, entropy, uncertainty, prediction error). It posits that the organism minimizes discrepancy between the world it perceives and its internal model.
Since model evidence and posterior probability are computationally intractable, Active Inference relies on a variational approximation, where model evidence is approximated by a variational free energy F and posterior probability approximated by an approximate posterior Q. The negative variational free energy is the evidence lower bound as a function of approximate posterior and evidence:

$$F[Q, y] = D_{\mathrm{KL}}[Q(x) \mid P(x \mid y)] - \log P(y)$$

The objective can be rewritten in terms of energy and entropy as $F[Q, y] = -\mathbb{E}_{Q(x)}[\log P(y, x)] - H[Q(x)]$, or in terms of complexity and accuracy:

$$F[Q, y] = D_{\mathrm{KL}}[Q(x) \mid P(x)] - \mathbb{E}_{Q(x)}[\log P(y \mid x)]$$

In this interpretation that the variational free energy optimizes for the least complex accurate explanation of the observed data in terms of the model P.

Artificial Intelligence

Parameterizing Nonlinear Dynamical Systems. Discretizing the dynamics equation in nonlinear dynamical systems yields the update equation $\mathrm{d}x = g(x, u)$. When the mapping g is parametrized by a sufficiently expressive artificial neural network, it is a universal finite state machine approximator called recurrent neural network (RNN), and can therefore in principle approximate any nonlinear neural computation.

Training Algorithms. The standard training algorithm for RNNs is backpropagation through time (BPTT), which works by storing past inputs u and states x in a register to compute the gradient with respect to the loss L. After updating the networks weights the register is cleared. State machine approximators are notoriously hard to train due to gradient instability introduced by high or low spectral radii of their recurrent weight matrices, costly storage and computational requirements, and vast state spaces induced by their recurrent architecture.
To overcome these architectural and algorithmic limitations, alternative approaches for training and parametrization have been proposed, such as real time recurrent learning (Williams and Zipser 1989) which is based on recursive gradient computation, or Hessian-free optimization (Sutskever 2013).
To circumvent the caveats that come with computing gradients along the path taken in state space, some approaches abandon optimizing recurrent weights altogether, most notably Fast Weight Programmers (Schmidhuber 1991) and echo state networks (ESN). In ESNs, the weights of the recurrent weight matrix, termed 'reservoir', are assigned before training and not altered after initialization. Instead, only the weights of the output layers are learnable, while the internal state of the system evolves according to the dynamics defined by the reservoir. The parameters of the reservoir are chosen such, that the spectral radius of the reservoir matrix is smaller than 1 in order to avoid chaotic behavior by design.
Other architectures, most prominently the Long Short-Term Memory network (LSTM) and the Gated Recurrent Unit as its direct descendant, target the gradient stability problem without abandoning BPTT: the core idea is to reparametrize conventional RNNs by computing the state update dx which is then added to x_{t-1} instead of mapping x_{t-1} directly to x_t. The reparametrization itself does not provide a representational advantage (Sutskever), yet results in less ill-behaved gradients.
When abstracting away architectural specifics, the linearized state update equation reads $x_t = g_1 x_{t-1} + g_2 \mathrm{d}x$, where the gates g_1 and g_2 as well as the update $\mathrm{d}x$ are nonlinear functions of x and u. The gates allow to read from and write to the internal state x by multiplication, while x is held fixed by default. The decomposition into multiplicative and additive elements is an early predecessor of a growing trend across the field to utilize multiplicative architectural elements. Examples of this can be found in artificial neuroplasticity (Miconi 2016), other gated

architectures (Gated MLP), dynamic routing (Capsule networks), and various attention-based models (NTM, Transformer).

Temporal Credit Assignment. The tremendous success of all aforementioned approaches rests on units with rate-coded output resulting in fully differentiable architectures, which renders end-to-end learning with backpropagation possible (Hinton 1986).
The only principled way these algorithms perform temporal credit assignment, i.e. identifying events or internal state transitions at earlier time points which are associated with later outcomes, is done by calculating the dependency of a state x_t on all earlier states $x_{0:t}$ in terms of partial derivatives.
Furthermore, current training algorithms are temporally dense (or sparse in unprincipled ways, e.g. gradient skipping), and as such potentially computationally wasteful, and inept for continual learning.
Real nervous systems are heavily constrained by limited computational resources and are therefore unlikely to implement or approximate backpropagation through time in order to track temporal dependencies and reason about counterfactuals in terms of derivatives, yet if so, highly selectively.

Reinforcement Learning. One of the key endeavours of reinforcement learning is to learn goal-directed sequential decision making in partially observable and possibly stochastic environments with sparse feedback.
The basic quantity that underpins reinforcement learning is the environment dynamics $P(s', r \mid s, a)$, where s are the observable environment states, r is the reward, and a is the action taken by a policy.
In all but a few cases the mapping P is not known, and information about the reward distribution has to be gathered by the agent interacting with the environment. Therefore, most flavors of reinforcement learning are essentially prospective in nature, in that information about the reward distribution on the environment is represented as an expectation value over future rewards (called value function v or action value function q), rather than retrospective collective statistics about the behavior of the distribution.
Since the environment dynamics and the reward distribution are not differentiable with respect to the reward r or the action a, and thus do not permit the propagation of gradient information, agents cannot be trained by algorithms developed for end-to-end differentiable problems.
Instead, a vast variety of algorithms has been developed to account for 1. the strong correlation between consecutive environment states, 2. the delayed consequences of earlier decisions, i.e. temporal credit assignment, and 3. for balancing locally best actions with exploring other parts of the state space not visited yet to find potentially better paths, i.e. exploration exploitation dichotomy.
For this purpose, modeling the policy as a probability distribution over actions can have desirable properties: action distributions can exhibit asymptotically deterministic behavior when needed, yet naturally implement heuristics derived from optimal exploratory behavior (Gershman 2019), such as Bayes optimality.
Reinforcement Learning and Control have deep connections to Probabilistic Inference (Levine 2018), and maximum entropy reinforcement learning was shown to be equivalent to structured variational inference, resulting in Soft Actor Critic (Haarnoja 2018) and a variational version of m_0 (Levine 2020).

Abstract

I'd argue that it is possible to train finite state machine approximators that approximate neural function by reframing learning a state transition function g in latent state space as an active inference or reinforcement learning problem.

Basic Terminology. One can reframe learning dynamics on low dimensional point-cloud manifolds as an active inference problem in latent state space as follows.
Denote the latent state as x, and the posterior over current state conditioned on past state as $P(x_t \mid x_{t-1})$. Substitute the action a by the index i indexing the state vector x or the latent state transition $\mathrm{d}x$. Replace the reward r by any performance metric y, e.g. the loss L, or the fitness F.
The performance metric can be interpreted as and observable signal from the 'environment'. Active inference is of course not limited to sensory data, but works with general observables, e.g. rewards r.

State Transitions in Latent Space. The distribution over sensory input becomes $P(L \mid x)$ and the posterior (i.e. a latent policy) becomes $P(\mathrm{d}x \mid x)$ or $P(i \mid x)$. To transition the state we need an update rule, in the simplest case $x + \mathrm{d}x$. This update rule is fully general since it is a parametrization of the general form of a stochastic ordinary differential equation. Updating the latent state thus becomes an active process executed by an internal agent that can be optimized by all available reinforcement learning algorithms.

Optimization. I think there is a number of ways how one can cast learning stochastic latent dynamics as an (active) inference and reinforcement learning problem.
One particularly appealing approach might be learning latent state dynamics by optimizing the variational free energy as

$$F[Q, L] = D_{\mathrm{KL}}[Q(\mathrm{d}x) \mid P(\mathrm{d}x)] - \mathbb{E}_{Q(\mathrm{d}x)}[\log P(-L \mid \mathrm{d}x_t, x_{t-1})],$$

where Q is a parametric approximation to the exact or optimal latent dynamics distribution P (time indices t and conditional dependencies on past states x_{t-1} omitted wherever possible for better readability).
The first term connects active inference in latent space to population geometry by implicitly constraining the geometry of the manifold on which the latent state lives, while the second term controls the optimality of the internal dynamics for optimizing the loss (or any other available performance metric).

Latent State Control. Navigating latent space becomes a control problem with an internal state controller Q_w as the transition policy with parameters w. There is a vast body of literature that explores this idea from different angles, I might summarize it in the future.
In contrast to conventional reinforcement learning problems, this approach does not suffer from two crucial disadvantages:
1. The reward distribution is known, at least to some degree. We argue that optimizing variational free energy equates learning a model of the 'internal environment' concurrent with a transition control policy by optimizing the expected $\log P(-L \mid \mathrm{d}x_t, x_{t-1})$. As opposed to reinforcement learning problems, this is possible because the loss distribution is differentiable with respect to $\mathrm{d}x$.
2. The joint environment dynamics $P(h_t, a_t, h_{t-1})$ are neither known nor differentiable with respect to the action a in conventional reinforcement learning, whereas the latent space dynamics $P(\mathrm{d}x_t, x_{t-1}, L_{t-1})$ have a well-defined gradient with respect to $\mathrm{d}x$ and x.
This begs the question if it is worthwhile making matters more complicated: one could simply use any customary Q-learning of policy gradient algorithm to train the internal state controller, but most of them a built around dealing with of non differentiability of the environment dynamics.

Schematically, I envision the following strategy:
1. Learn value functions of state transitions. This allows to compute expectations of future performance and transport value to deal with long term dependencies in a more efficient way then temporally dense gradient computations.
2. Search the latent space by using exploration strategies known from cognitive neuroscience and utilized in reinforcement learning.
3. Use of the existence of gradients of the latent space dynamics with respect to states and transitions to supplement conventional algorithms with gradient and environment dynamics that is usually inaccessible and has to be learned separately.

Control and Population Geometry. In contrast to reinforcement learning, there is no immutable 'ground truth' dynamics to be learned. This means that one essentially has to pick suitable prior ground truth dynamics. Typically, Gaussian priors (VAE, for sequential input Stochastic Latent Actor-Critic) are, used but we could utilize the KL Divergence term to incorporate prior knowledge about locally optimal behavior or desirable geometric structure on Q. Some ideas to be explored further include: $P(\mathrm{d}x)$ is

1. locally optimal, as found by exhaustive search over all available or the most promising set of transitions
2. obtained by a search procedure in trees (MCTS, interesting recent paper on planning in large decision trees under limited resources) or on graphs that represent a good transition strategy along the surface of the manifold
3. obtained by another exploration strategy used in reinforcement learning
4. parametrized by a value function v or action-value function q and learned alongside Q
5. carefully chosen to satisfy specific geometric properties known to result in 'useful' dynamics, e.g. properties of attractor manifold, transition sparsity, etc.

Nonlinear dynamical systems, latent state dynamics on low dimensional point–cloud manifolds. Manifold structure and p flow represent a distribution over state transitions dx with either states or a probabilistic representation of the state transitions on the surface of the manifold. The primary object of study is therefore a probability distribution p(x) over state transitions x which can be continuous or discrete, or a probability distribution p(i) over indices i that index states to which a certain state transition budget dx is 'allocated'.

Transition dynamics can be deterministic or stochastic, but probabilistic 1. more general case, 2. the lends itself to being interpreted and laid out in the framework of probabilistic inference and active inference. This interpretation in turn points directly to reinforcement learning by the work of Levine, and opens up the possibility of using heuristics from reinforcement learning for learning transition dynamics and searching of the manifold itself or the state space the manifold is embedded in. Opens up connection between the rich theory of nonlinear dimensionality reduction or neural population geometry, and learning state machine approximators, Leave backpropagation through time behind the only means to learn state machines, which essentially happens in a rather unstable way by retrospectively perturbing a path taken through state space in a local neighborhood around the path. Prospective evaluation of value, searching space by clever heuristics, counterfactual reasoning, model based or value function based lookaheads…

Skeleton for basic experiments with Reinforcement Learning applied to internal state dynamics.
The code contains custom environments inherited from OpenAI gym, custom optimizers, and custom state transition dynamics regulated by the Reservoir class which has no tunable parameters (hence the name 'reservoir', borrowed from Echo State Network Literature). It's possible to propagate gradient information through the reservoir by substituting activation_fn with a (tunable or fixed) network, but that's not intended right now because the transition dynamics should be entirely learnable by the actor.

Transitions computed by the actor (in actor-critic-like algorithms) or computed by the action-value function (in Q-learning-like approaches) are deterministic or stochastic.

The code here represents the very basics: The files p_dx, MLE_x, and MLE_dx contain runnable code with experiments. The agent learns to use a binary state switch by observing sequences of spikes, and to navigate simple mazes with and without walls.