THEORY

Where state (s) includes a system's perceptions (p), thoughts (t), and actions chosen given current perceptions and thoughts, (a), ($\{p, t, a\} \subseteq s$) the optimal action to perform (a') and predicted state (s') can be determined given all previous states.

$$previous\ states \rightarrow optimal\ action$$

$$previous\ states \rightarrow predicted\ state$$

Since an LSTM-based prediction function can inherently be trained to map a stream of input to a single appropriate output, action selection and prediction functions can be modeled as

$$a'(s_t) \rightarrow s_t.a$$

$$s'(s_t) \rightarrow s_{t+1}$$

Inside a' and s'

While mathematically distinct functions, a' and s' both share many internal parameters. a' overrides whatever actions previously belonged to a state. Then s' takes that state with the action a' selected and predicts the outcome of that state. Essentially:

$$s'(s_t) \cong s'\big(s_t - s_t.a + a'(s_t)\big)$$

This is not precisely the case as explained later but meets the strict mathematical requirements for maximization so there exists a $\frac{\partial s'}{\partial a'}$ to adjust $a'_{w,b}$ appropriately

If the predictor is trained that states where a_{good} is selected precede states where a high user supplied reward ($s.p.r = r_{high}$) is given ($s'(\{a = a_{good}\}) \rightarrow \{p = \{r = r_{high}\}\}$), then reward can be maximized by adjusting the action selection function's internal parameters (in the case of a neural network, it's weights and biases ($a'_{w,b}$) so that a' is biased more towards selected actions that s' predicts will yield highest possible reward:

$$\max_{a'_{w,b}} s'(s_0).\mathrm{p.r}$$

If s' correlated both a_{good1} and a_{good2} with higher reward, maximizing reward of the predicted state given an initial state by adjusting action selection would cause a_{good1} and a_{good2} to both be chosen more often. Thus, **action selection can be optimized to select actions that yield an increase in a given resulting state parameter**.

State prediction, however, is not limited to the immediate next state. Chains of s' can predict many states into the future

$$s'\Big(s'\big(s'(s_0)\big)\Big) \rightarrow s_3$$

Average or sum future states can then be calculated

$$s_{simple\ sum}(s) = s + s_{simple\ sum}(s'(s))$$

Then, instead of simply optimizing a' to select actions with a higher immediate reward, **maximizing the sum of future state rewards can enable farsighted optimization**

$$\max_{a'_{w,b}} \mathrm{s_{simple\ sum}}(s_0).\mathrm{p.r}$$

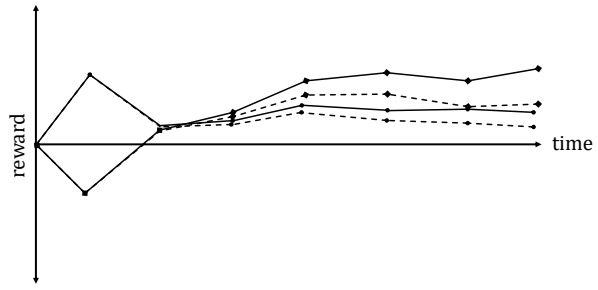
How farsighted?

If even one remotely probable state with any nonzero element s_{sum}, being a self-referencing function, will contain infinity. Nothing can be done to maximize these tensors; one does not speak of maximizing infinity. Even if the true integral of all future states were finite ($\lim_{t_{max}\rightarrow\infty} \int_{t_0}^{t_{max}} s_t\, dt < \infty$), in all practicality, no algorithm could directly evaluate the self-referencing nature of s_{sum} as a single tensor. To solve this, s_{sum} is only evaluated until hyperparameter f_n iterations.

Realistically though, the rewards or losses an agent may experience far off into the future are not as definite as those predicted sooner. To account for this, a scalar farsighted factor f selected between zero and unity is multiplied by the predicted state at every iteration to more reasonably predict the future sum of all states.

$$s_{sum} = s + f s_{sum}\big(s'(s)\big)$$

This, though, does not mean that a' would tend to select actions with immediate gratification over long-term benefit. If sufficiently close to unity, brief decrease in the parameter sought to be optimized may allow for it to increase significantly more than the path of least resistance would. Even in cases where $s_{simple\ sum}(s_0) > s_{sum}(s_0)$, optimizing for s_{sum} with the farsighted factor may ultimately bring greater long-term reward. The following diagram illustrates this:

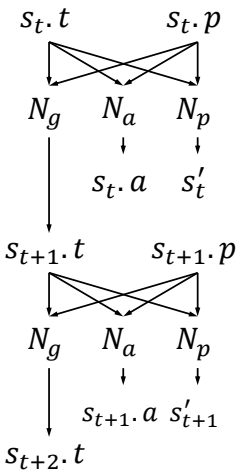


This graph plots immediate reward at each moment in time for two courses of action in circle and diamond plots. The farsighted reward ($reward \times f^t$) is shown in dotted lines. Had a' been optimized for simply greatest sum of solid plotted points, it would gain an immediate reward as pictured, but overall gain less than if it was optimized to consider the farsighted factor.

Implementation

This action selection and optimization system is implemented in a flat, sparse, LSTM neural network pretrained to (1) maximize reward and (2) train a' with present and past observation data.

Four sections of the neural network input and output mapping emerge: perceptions, thoughts, actions, and predictions. The network host considers the neurons as participating in either: general thinking (N_g), action selection (N_a), or prediction (N_p) and computes their activations in that order. Activation values from general thoughts are fed from the previous layer into the next layer. Activation values for action selection are interpreted and executed by the network host. Activation values for prediction are only computed when needed, likely only in maximizing a reward.



As a result of its flat structure, if connection patterns emerge comparable to a one-hidden-layer perceptron network between the perceptions and action selection, it will take at least two iterations of activation computation to solve N_a for a new perception.

Backpropagation, when used, assigns a δ to each input given δ's and weights connecting to each output just as a fully connected multilayer perceptron would except each 'layer' in this network is a recorded state of the δ's, weights, and activation values for an arbitrarily chosen hyperparameter number of states to save N_s. Selecting to record eight past states at any given time ($N_s = 8$) may make the network's 'intelligence' gain by backpropagation comparable to that of an eight-layer network.