How BECCA works

Becca 0.5.0

Brandon Rohrer, March 2014

Becca's long term goals are captured in its name: brain-emulating cognition and control architecture. This is a detailed description of the architecture and the algorithms that underlie it. It is a summary of the python code. If any discrepancies exist, the code is the authoritative source. This description omits some of the details of implementation and computational bookkeeping that aren't essential to understanding its principles of operation.

Becca belongs to a class of machine learning algorithms called reinforcement learners. The distinguishing characteristic of a reinforcement learning agent is that it receives an explicit reward signal, which it tries to maximize. (Figure 1)

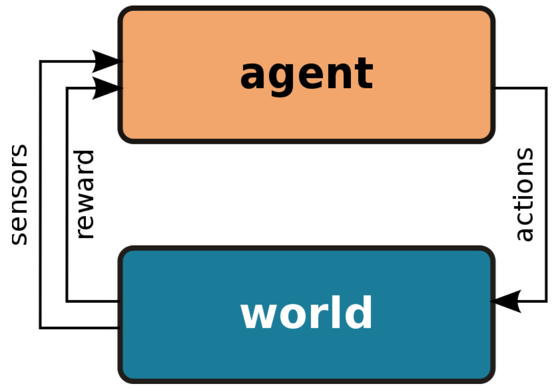

Figure 1: A reinforcement learning agent.

To be more precise, Becca consists of both an unsupervised learning algorithm that extracts features from the data, and a reinforcement learning algorithm that learns which sequences of features tend to result in positive outcomes. (Figure 2)

Becca is designed so that it doesn't know (or need to know) anything about the world it's operating in. This is so that it can be used as the learning portion of any embodied agent, a general purpose robot brain.

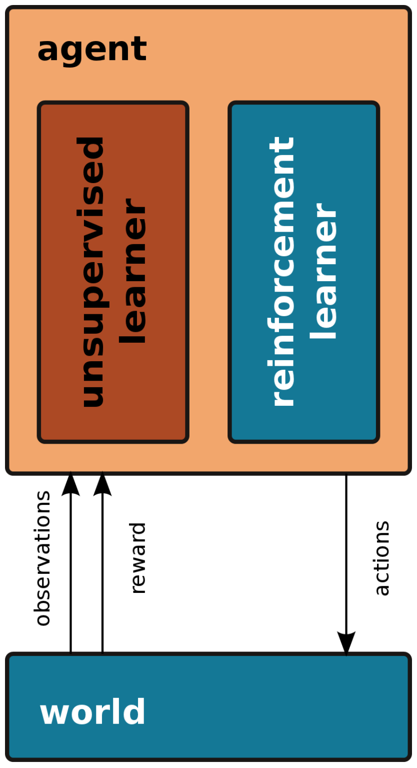

Figure 2: Becca's agent contains both an unsupervised feature extractor and a reinforcement learner.

AGENT

The agent is, in essence, the brain. Everything else—the hardware, the sensors, and the rest of the universe—is included in the world. Even any system-specific preprocessing of sensor inputs and post-processing of actions are part of the world. The agent handles only sensor and reward inputs and produces action outputs. It does this in discrete time steps.

The agent contains one or more blocks in a hierarchical arrangement and a central hub that is connected to

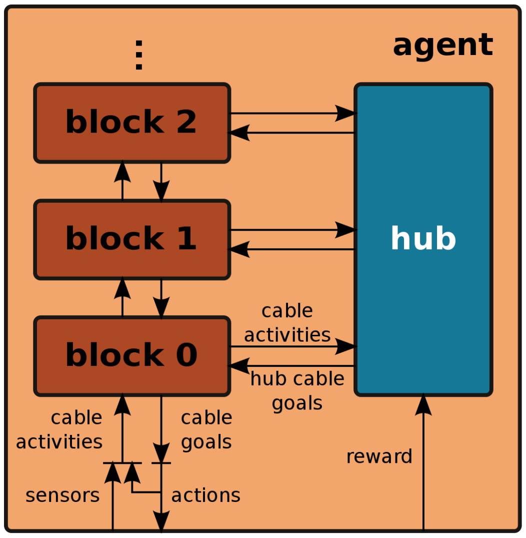

Figure 3: The agent consists of a hierarchy of blocks (the unsupervised learner), all of which are connected to a central hub (the reinforcement learner).

them all. (Figure 3) The blocks build the sensory information into spatio-temporal features whose complexity and extent grows greater in each subsequent block. The first block takes an array of cable activities as inputs at each time step, which are composed of the latest sensor activities and a copy of the actions from the previous time step. Here, cables are a physical metaphor for channels or lines that carry a signal. Cable activities are the set of signal values on an array of cables at one point in time. Each block takes a set of cable activities as inputs and produces a set of cable activities as outputs. They are cascaded, such that the cable activity outputs of one block are also the inputs to the next.

At each time step, the cable activities are propagated upward through the tower of blocks. Then, copies of the cable activity inputs to each block are sent to the hub. These, together with the reward, are used to select a hub cable goal for one of the blocks. This centrally-selected goal is analogous to deliberate behavior in humans. The blocks then propagate this goal downward through the tower, together with any internal goals (similar to reactions or auto-pilot behavior in humans) that they generate. A subset of the cable goals that emerge from the lowest block correspond to an array of actions. These are passed back to the world. They are they only decisions or influence that the agent has on the world around it.

BLOCK

Blocks are so named because they are core structural elements of the architecture and naturally form a hierarchy or tower, in which each is supported by the one below. Each block contains some cogs and a ziptie, a cable-clustering element. (Figure 4)

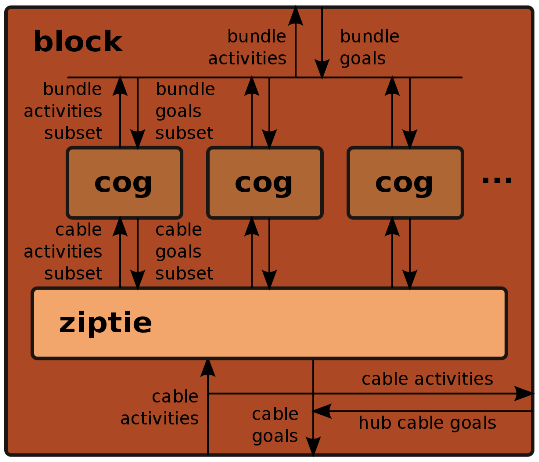

Figure 4: Each block consists of a ziptie cable clusterer and a number of cogs, all operating in parallel.

The block takes a set of cable activities as inputs. The ziptie organizes them into groups over time and assigns each group of cables to a single cog. Cogs are where cable activities are combined together across cables and over time to build spatio-temporal features. They take arrays of cable activities as inputs and combine those into bundles. Their outputs are the activities in each of those bundles. The bundle activities of all the cogs are combined together into a single array to form the output of the block.

The cable activities are normalized such that they fall on an interval from zero (the lowest cable activity ever experienced on that cable) to one (the highest). This helps keep every bock insensitive to the range of cable activities it takes in, including sensor values from the world.

COG

Like their mechanical counterparts, cogs are relatively simple devices whose true value is realized when they are used in parallel with a number of their fellows. Each cog consists of a daisychain and a ziptie. (Figure 5)

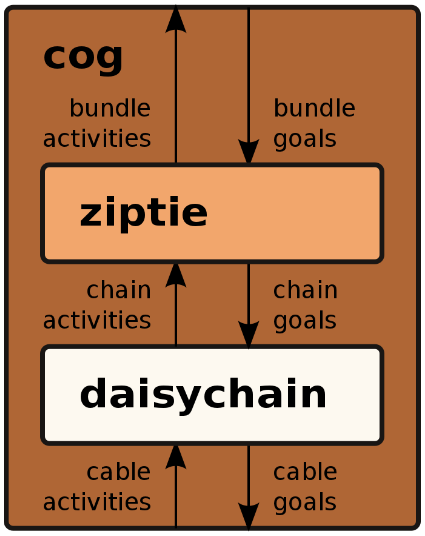

Figure 5: Each cog contains a daisychain and a ziptie.

The daisychain identifies the relative prevalence of all possible cable activity sequences from one time step to the next, i.e. chains. The ziptie clusters commonly co-occurring chains into bundles. A bundle of chains is a pattern of signals that covers multiple cables and multiple time steps. Another way of expressing this is to say that is has both spatial and temporal extent or is a spatio-temporal pattern. The bundles that the cog generates are data-driven features. Taken in aggregate, all the bundles from all the cogs form a feature set that Becca can use to represent its relationship to its world and to choose its future actions.

ZIPTIE

Ziptie is a clustering algorithm used throughout Becca. In contrast to many unsupervised learning algorithms which cluster individual data points, ziptie clusters signals or time series of data points. This is analogous to bundling groups of cables with zipties. Like all algorithms in Becca, it is incremental, performing its functions iteratively at each time step. Ziptie identifies cables whose activities tend to be high at the same time. Although at any given moment the cable activities may be very different, over time they will tend to be co-active.

The ziptie updates and maintains a map from cables to bundles. (Figure 6) This map dictates how cable activities are translated into bundle activities and how bundle goals are translated into cable goals.

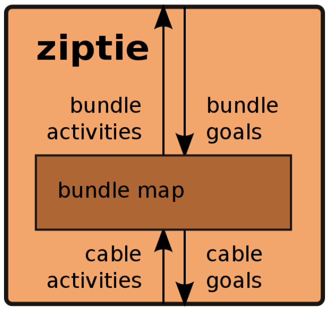

Figure 6: A ziptie clusters its cables together into bundles.

The ziptie groups cables into bundles based on how often the cables' activity levels tend to be simultaneously high. Any cable can be a member of one bundle, many bundles, or no bundles. Membership is binary (all or nothing), but there is no reason the algorithm couldn't be extended to represent partial membership.

Bundle activity

Bundle activity is a weighted combination of its constituent cable activities. Some inhibition occurs between bundles that share cables. Initial bundle activities are calculated by taking the generalized mean of the cable activities with a negative exponent. The generalized mean, M_p, is given by:

$$M_p(x) = \frac{1}{n}\left(\sum_{i=1}^{n} x_i^p\right)^{\frac{1}{p}}$$

For $p = 1$, M_p is the arithmetic mean. For $\lim_{p\to 0}$ it is the geometric mean. For $\lim_{p\to\infty}$ it gives the maximum value. And for $\lim_{p\to-\infty}$ it gives the minimum.

A negative exponent weights the lower cable activities more heavily. Ziptie makes a first pass at each bundle's activity .

$$\begin{aligned} a &= M_m(c) \\ a &: \text{initial bundle activity} \\ c &: \text{activities of cables in the bundle} \\ m &: \text{generalized mean exponent} = -4 \end{aligned}$$

If a cable contributes to multiple bundles, the activity it contributes to each will be somewhat inhibited.

$$\begin{aligned} d &= c\left(\frac{a}{b}\right)^k \\ a &: \text{initial bundle activity} \\ b &: \text{highest initial bundle activity} \\ c &: \text{original cable activity} \\ d &: \text{inhibited cable activity} \\ k &: \text{inhibition exponent} = 6 \end{aligned}$$

The final bundle activity is calculated in the same way as the initial bundle activity, except that it uses the inhibited cable activities. A single cable ends up contributing different activation levels to the different bundles of which it's a member.

A cable can also have a residual non-bundle activity, signifying that it has signal energy that has not been applied to activating any bundle.

$$\begin{aligned} f &= \max\left(0, c - \sum d\right) \\ c &: \text{original cable activity} \\ d &: \text{inhibited cable activity for each bundle} \\ f &: \text{non-bundle activity} \end{aligned}$$

Bundle nucleation

Non-bundle cable activity is accumulated over time. If it accumulates to a high enough level, that signals the ziptie that a new bundle needs to be created.

$$\begin{aligned} g &= g + (f(1-g) - cgp)q \\ c &: \text{cable activity} \\ f &: \text{non-bundle activity} \\ g &: \text{nucleation energy} \\ p &: \text{energy decay rate} = .01 \\ q &: \text{nucleation energy rate} = .0001 \end{aligned}$$

A cable's nucleation energy exceeds a threshold (.05), then it nucleates a new bundle, that is, it becomes the sole member of a new bundle.

Bundle agglomeration

Bundles accumulate agglomeration energy from cables whose non-bundle activity is correlated with their own.

$$\begin{aligned} h &= fa \\ a &: \text{bundle activity} \\ f &: \text{non-bundle activity} \\ h &: \text{co-activity} \end{aligned}$$

Each cable's non-bundle activity is distributed to agglomeration energy with each bundle proportionally to their co-activities.

$$\begin{aligned} s &= s + f\frac{a}{\sum a}(h(1-s) - csp)t \\ s &: \text{agglomeration energy between} \\ & \quad \text{a cable and a bundle} \\ f &: \text{non-bundle activity} \\ a &: \text{bundle activity} \\ \sum a &: \text{sum of all bundle activities} \\ h &: \text{co-activity} \\ c &: \text{cable activity} \\ p &: \text{energy decay rate} = .01 \\ t &: \text{agglomeration energy rate} = .01 \end{aligned}$$

When the agglomeration energy exceeds a threshold (.05), the cable is added to the bundle.

Cable goals

Bundle goals are translated back into cable goals as they propagate down the tower of blocks. For each cable, the goals for the bundles it contributes to are summed in such a way that the magnitude of the sum is never greater than one.

$$\begin{aligned} u &= S(v) \\ u &: \text{cable goal} \\ v &: \text{bundle goals} \end{aligned}$$

The bounded sum function, S, never produces a total greater than one, given that its arguments are all less than one.

$$\begin{aligned} S(x) &= A\left(\sum B(x)\right) \\ A(x) &= 1 - \frac{1}{x+1} \\ B(x) &= \frac{x}{1-x} - 1 \end{aligned}$$

DAISYCHAIN

Dasiychain is an incremental algorithm that estimates the probability of one cable being active following another. It represents this as a conditional probability: given that one cable is active what is the expected activity of a second cable in the next time step. (Figure 7) High expected chain activities indicate sequences of cable activities that co-occur regularly. They identify temporal structure in the data.

Expected chain activities are similar to transition probabilities in Markov models. The difference is that in a Markov model, only one state can be occupied at each time step. This is analogous to just one cable being active. In a daisychain, many cables can be completely or partially active at once. As a result, transition probabilities can sum to much more than one.

Expected chain activity

A temporal sequence of one cable being active, followed by another, is a chain. The activity of a chain is given

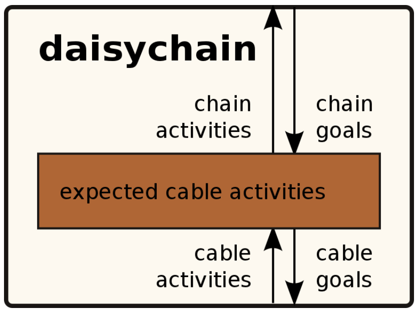

Figure 7: A daisychain identifies commonly occurring cable sequences, or chains.

by the product of the two cable activities involved.

$$c = ab$$

c	:	chain activity
a	:	cable activity
b	:	another cable activity, from the previous time step

A leaky accumulation of the activity on each cable and on each chain is also maintained.

$$d = d + c - \frac{1}{df}$$

d	:	accumulated chain activity
c	:	chain activity
f	:	aging time constant $= 10^6$

$$g = g + b - \frac{1}{gk}$$

g	:	accumulated cable activity
b	:	previous cable activity
k	:	chain update rate $= .1$

The expected chain activities are maintained and updated based on the current chain activities.

$$h = h + (c - h)b\left(\frac{1-k}{g} + k\right)$$

h	:	expected chain activities
c	:	chain activity
b	:	previous cable activity
k	:	chain update rate $= .1$
g	:	accumulated cable activity

Expected chain activity deviation

In addition, the expected *deviation* from the expected chain activities are maintained and updated based on the difference between the current and expected chain activities.

$$m = m + (|c - h| - m)b\left(\frac{1-k}{g} + k\right)$$

m	:	expected chain activity deviation
c	:	chain activity
h	:	expected chain activities
b	:	previous cable activity
k	:	chain update rate $= .1$
g	:	accumulated cable activity

Predictions

The temporal structure captured in the expected chain activities provide a basis for making short-term predictions. The reaction is the predicted next set of cable activities.

$$n = \frac{\sum ha}{\sum a}$$

n	:	predicted cable activity
h	:	all expected chain activities that include the cable activity being predicted
a	:	current cable activities

The most recently observed cable activities can be compared to those that would have been predicted from the previous cable activities to find surprising events.

$$p = \frac{\sum \frac{a|a-n|}{m}}{\sum \frac{a}{m}}$$

p	:	surprise
a	:	current cable activities
n	:	predicted cable activity, based on previous cable activity
m	:	expected chain activity deviation

Cable goals

As chain goals are propagated down through the daisychain, they are combined to form the cable goals. Each cable goal is a weighted, bounded sum of all the goals of the chains it belongs to.

$$s = S(qa + n)$$

s	:	cable goal
S	:	bounded sum operator
q	:	chain goals
a	:	current cable activities
n	:	predicted cable activity

HUB

The hub is where reinforcement learning is implemented and decisions are made. It takes in copies of all blocks' input cable activities, as well as the reward. It maintains an estimate of expected reward for every cable activity sequence, or chain, similar to the daisychain's estimate of future cable activity for every cable activity chain within a block. The hub also keeps a running total of each chain's activity. (Figure 8)

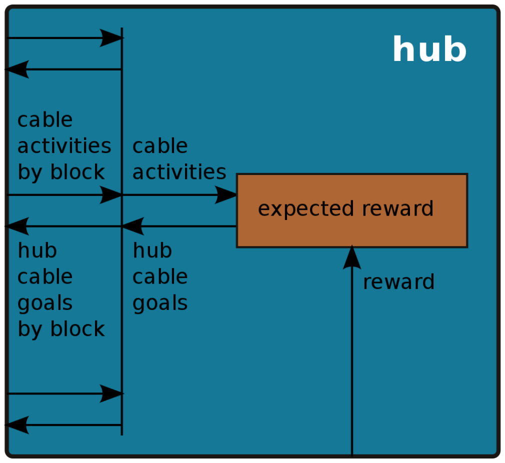

Figure 8: The hub uses cable activities and reward information to choose goals.

The reward signal is normalized such that it falls on an interval from zero (the lowest reward ever experienced) to one (the highest). This way, BECCA is insensitive to the range of reward values covered by the world. It is one less assumption that the agent needs to make. The difference in reward between one time step and the next is actually the quantity that the hub estimates. In this way it detects *changes* in reward, rather than constant values. The chains that consistently result in changes in reward are identified and exploited in planning.

The hub gathers cable activities from all the blocks. Using the current set of cable activities, it chooses a cable goal at each time step. It then associates changes in reward with each activity-goal pair. This is where the hub differs from daisychain. The transition chains in daisychain are from cable activity to next cable activity, rather than from cable activity to next cable goal. Like ziptie, the hub does this incrementally, adjusting the estimate by a small amount each time. It also creates a running sum of each chain's activity history, which slowly decays over time. This calculation is very similar to the one used in ziptie.

$$c = ab$$

c	:	chain activity
a	:	cable activity
b	:	cable goal

$$d = d + c - \frac{1}{df}$$

d	:	accumulated chain activity
c	:	chain activity
f	:	aging time constant $= 10^5$

Reward estimation

In order to provide some ability to associate transitions with delayed rewards, a reward trace is calculated. It is a summation of reward, decayed over time.

$$q = \sum_{t=0}^{m} p_t (1-n)^t$$

q	:	full reward
t	:	time steps into the future
m	:	trace length $= 10$
p	:	change in reward
p_t	:	change in reward t time steps into the future
n	:	reward trace decay rate $= .3$

The expected change in reward associated with each chain is updated using the instantaneous and cumulative chain activities.

$$r = r + (q-r)c\left(\frac{1-k}{g} + k\right)$$

r	:	expected change in reward
q	:	observed change in reward
c	:	chain activity
k	:	chain update rate $= .01$
g	:	accumulated chain activity

Uncertainty-driven exploration

The uncertainty in each of these reward estimates is also estimated. This is done using the accumulated chain activity for each estimate. The reasoning behind this is that more chain activity represents more sampling of the process, which in turn yields a higher confidence estimate. Estimates with a high level of uncertainty show where exploration would most likely improve the model. The uncertainty is used to scale additive random noise to the reward estimate, encouraging exploration in those areas.

$$s = r + N\left(0, \frac{x}{g+1}\right)$$

s	:	estimated reward value at this time step
r	:	expected change in reward
$N(a,b)$	:	The Normal distribution with a mean of a and a standard deviation of b
x	:	exploration constant $= .1$
g	:	accumulated chain activity

Goal selection

The estimated reward value for all the chains are weighted by the current cable activities. The winning cable goal is the one with the highest combined minimum and maximum values across all chains. The hub then sets that goal in the appropriate block.

$$w = \text{argmax}(\max(cs) + \min(cs))$$

w	:	goal cable
s	:	estimated reward value at this time step
c	:	previous cable activity

To ensure that the effects of intentionally-selected goals don't warp estimates, they are masked from the cable activities of the next time step. The activity of those cables is set to zero.

SUMMARY

Figure 9 shows the entire architecture at a glance. It's a bit unwieldy in this form, but having walked through all the pieces, it's useful to see how they all fit together. One of the agent's greatest strengths is that it doesn't start off knowing or believing anything about the world it's connected to. There are only three arrows that connect the two: the sensors and reward flowing in and the actions flowing out. It was designed this way so that Becca could be used for a brain in a robot of any type, physical or virtual, localized or distributed, commercial or experimental, rigid or flexible, mobile or fixed, walking, rolling, or articulated. For that matter it can be used to learn and control any system that has sensors and can take commands: stock trading, HVAC, retail shipping and receiving, or a self-driving automobile.

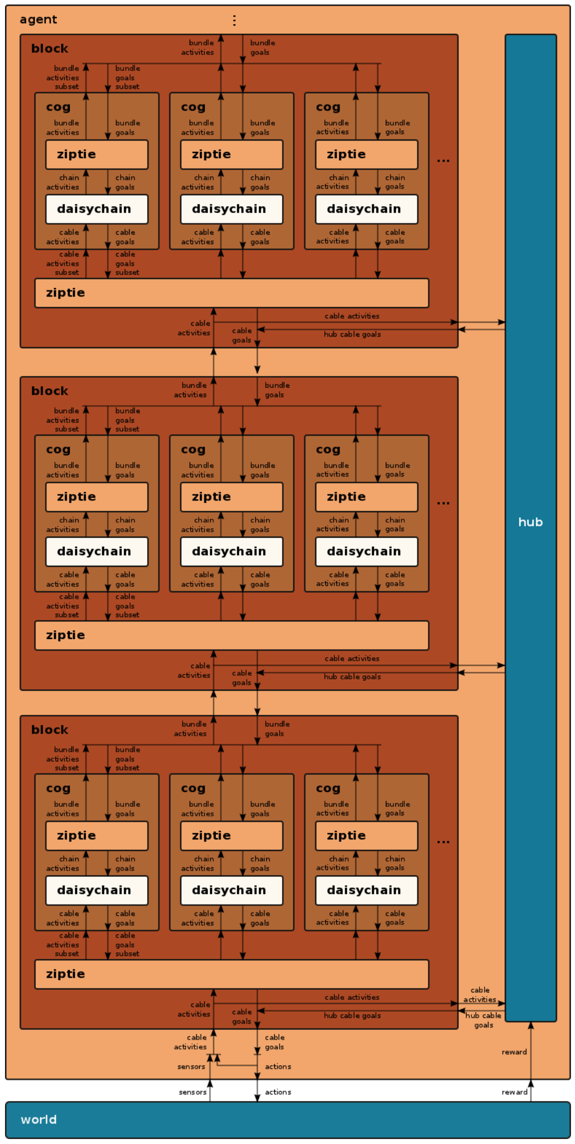

Figure 9: Becca, in detail.

Becca