Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2005-026
AIM-2005-012

April 13, 2005

Learning From Snapshot Examples

Jacob Beal

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Abstract

Examples are a powerful tool for teaching both humans and computers. In order to learn from examples, however, a student must first extract the examples from its stream of perception. Snapshot learning is a general approach to this problem, in which relevant samples of perception are used as examples. Learning from these examples can in turn improve the judgement of the snapshot mechanism, improving the quality of future examples. One way to implement snapshot learning is the Top-Cliff heuristic, which identifies relevant samples using a generalized notion of peaks. I apply snapshot learning with the Top-Cliff heuristic to solve a distributed learning problem and show that the resulting system learns rapidly and robustly, and can hallucinate useful examples in a perceptual stream from a teacherless system.[1]

1 Introduction

Examples are a powerful tool for teaching both humans and computers. In order to learn from examples, however, a student must be able to extract the examples from the stream of perception which the student experiences.

Consider a teacher explaining multiplication of negative numbers to a class full of students, as illustrated in Figure 1. The teacher starts by writing "5 X 7 = 35" on the blackboard, saying, "As you all know, multiplying two positive numbers makes a positive number." The teacher adds a minus in front of the 7, saying, "if we make one number negative, then the product is negative" and adds a minus in front of the 35. "But if both are negative" the teacher continues, adding a minus in front of the 5, "then the minus signs cancel and the product is positive." The teacher finishes, turning the -35 into +35, and the board reads "-5 X -7 = +35".

The teacher has given the students three examples, but the students have not necessarily received three examples. What the students have experienced is the teacher talking and writing on the blackboard for about a minute. In order to learn from the teacher's examples, the students must first find them and extract them from this experience — and getting it wrong can teach the student -5 X -7 = -35! A good teacher makes finding the examples easier, and a bad teacher may obscure the examples entirely.

When teaching computers with examples, we usually avoid this problem by providing the learner with input which contains explicit signals segmenting it into examples. This is often a reasonable assumption: it is natural for a medical diagnosis system, for example, to be presented with trivially distinguishable case histories.

In the absence of explicit signals, however, the learner needs to be able to discover examples for itself. Perception may be indirect (e.g. the teacher must use images and sound to convey math), there may be transients in the construction of the example (e.g. when the board says "5 X -7 = 35"), or there may not be a teacher at all!

[1]This work partially supported by NSF grant #6895292

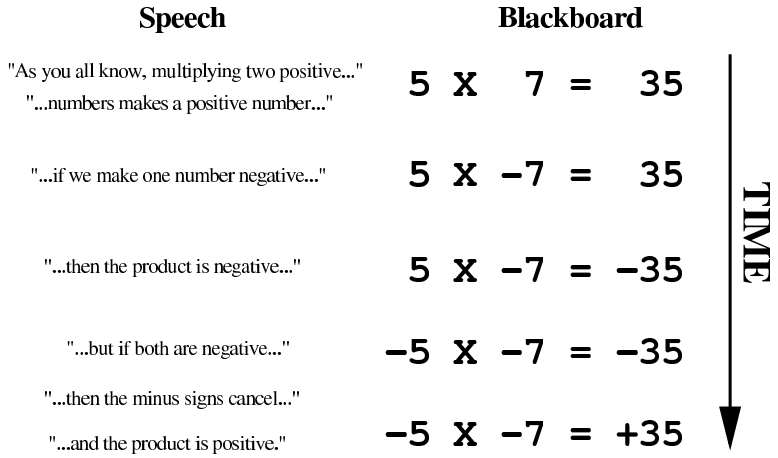


Figure 1: Three examples of multiplication, as seen by a student. The examples are spread over time and there are transitional states where the equation on the board is wrong. In order to learn correctly, the student must segment the sequence into the original three examples. This may become even harder with the addition of irrelevant information (e.g. "Johnny, stop that this instant!" or the beautiful day outside).

I propose a general approach in which examples are extracted by taking snapshots of relevant instants of perception, and one mechanism which implements the approach. Learning from these examples can in turn improve the performance of the snapshot mechanism. As evidence of the utility of this approach, I explain a learning problem that I solved using it, and demonstrate that the snapshot approach works predictably well under a wide range of conditions and parameters. I further demonstrate that when applied to a scenario which is not formulated as a sequence of examples, the snapshot approach can still extract examples which capture interesting features of the scenario.

2 Snapshot Learning

A *snapshot* is just what it sounds like: an instantaneous sample of perception. Figure 2 shows the general *snapshot learning* framework. Perceptual input streams continually on several *channels*; for example the system might have one channel carrying vision and another carrying sound. From this stream of input, the system extracts *targets* to learn about. The *snapshot mechanism* observes the input and measures how relevant the current perceptual sample is with respect to the current model of each target. When a perceptual sample is relevant enough for a particular target, the snapshot mechanism sends it to the *learner* as an example for that target. The learner then incrementally updates its model of the target using the new example. If the model is improved,

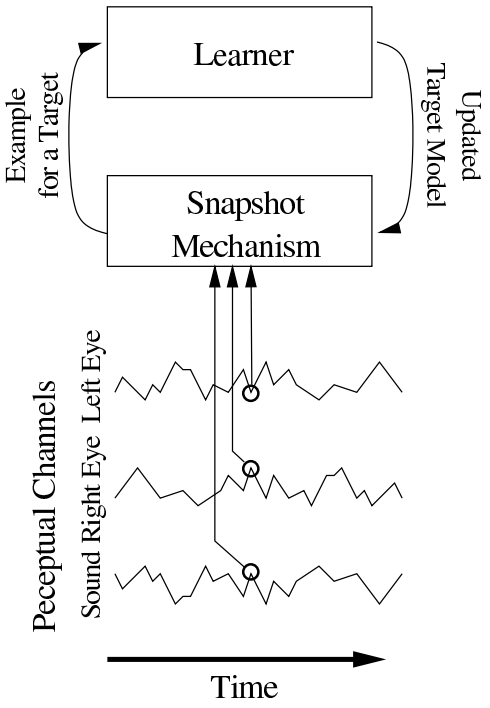


Figure 2: A *snapshot learning* system receives perceptual input on several *channels*. The snapshot mechanism identifies learning *targets* from the input stream. When a sample of perception is highly relevant to a particular target, the snapshot mechanism sends the sample to the learner as an example for that target. The learner then updates the target's model, improving in turn the snapshot mechanism's relevance estimates.

then the snapshot mechanism will be able to better estimate which samples are relevant, closing the loop on a beneficial cycle.

2.1 The Top-Cliff Heuristic

The Top-Cliff heuristic is one example of a snapshot mechanism. Top-Cliff assumes perceptual activity is sparse and, given a relevance metric for each channel, leverages that sparseness to choose a few selected samples for each target as "relevant enough" to be used as examples.

In essence, Top-Cliff is a peak detector. Rather than lumping relevance measures from each channel together into a single measure, though, Top-Cliff generalizes the idea of a peak to multiple dimensions, looking instead for moments when the relevance measures are about to fall off a multi-dimensional plateau. If the relevance measures were combined into a single measure, information such as which channels have zero relevance would be discarded.

Top-Cliff assumes two types of sparseness: across time and within samples. Sparseness across time means that any given perceptual feature is not present most of the time. Sparseness within samples means that in any given sample, most perceptual features will not be present. As a result, any given target can expect toq have many periods when there are no relevant perceptual features on any channel. For any given target, then, there will be a collection of easily separable relevant periods, even if the perceptual stream as a whole is not easily separable into examples.

If there are many relevant periods, then we can afford to be conservative and pick only the best moments from each relevant period; a few high-quality examples are likely to provide a much better basis for training than many examples of dubious quality.[2] Moreover, it is reasonable to assume that nearby moments in a relevant period are likely to be very similar, so taking many samples from any given relevant period is unlikely to add much new information and risks over-fitting to the situation.

If relevance was a differentiable function from a single channel of input, peaks in the relevance function would be a reasonable guess at the best moments in a relevant period. Since there may be many channels, and the relevance function may have plateaus of equal maximum value, I detect the onset of a top-cliff instead. I define a top-cliff to occur when:

- Some channel's relevance is decreasing. If nothing is going down, it isn't a cliff.
- No channel's relevance is increasing. If some channel is increasing, then it's a saddle, not a cliff.
- All *relevant channels* have increased since the last time they decreased. If some channel has already decreased, then this isn't the top cliff, but a lower one.

The last clause needs elaboration to precisely define *relevant channels*. The definition is motivated as follows: if we add a new channel which is never relevant, it should not affect the behavior of the system. However, if two channels are relevant and one drops to irrelevance, the other dropping to irrelevance is a lower cliff and shouldn't trigger a snapshot. Thus I define the set of *relevant channels* to be the set of channels which are either currently relevant, or which were relevant in the same period of relevance as one which is currently relevant. This small piece of memory allows the Top-Cliff heuristic to incrementally distinguish between these situations (Figure 3).

For discrete sampling of the channel, I approximate the derivative of relevance by comparing the most recent sample of the input with the previous sample. If a top-cliff is detected, then the previous sample, just before the cliff, is taken as the example.

For example, consider the behavior of the Top-Cliff heuristic on the sequence on three channels in Figure 4, showing relevance levels over time with respect to

[2]See, for example, Winston's near-miss learner.[12]

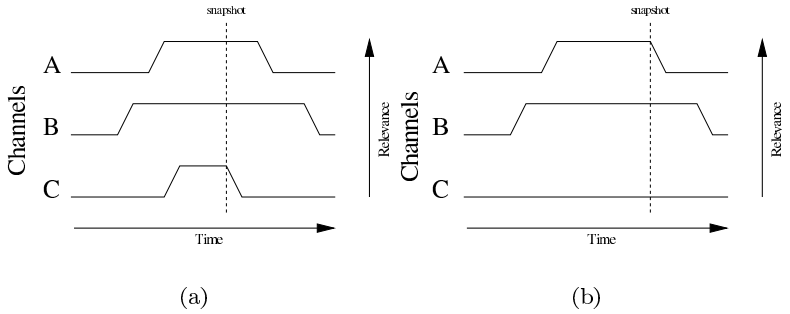


Figure 3: *Relevant channels* are not only the ones currently active, but also all channels whose most recent period of relevance overlapped those which are still relevant. This is needed to allow the system to distinguish between situations (a) and (b) once channel C is no longer relevant.

some target. At time 1, only channels A and C are relevant, and A is dropping, so it takes a snapshot. At time 2, A is dropping, but since B is rising at the same time, it is not a cliff and no snapshot happens. At times 3 and 4, a channel drops, but since A has not risen since the last time it dropped, this is not the top cliff and no snapshot happens. At time 5, channel A is still relevant, since it was previously active concurrently with channels B and C, so no snapshot occurs. At time 6, however, only A is relevant, so it takes a snapshot.

For channels with high-frequency noise, a naive implementation of the Top-Cliff heuristic may produce many spurious snapshots. In such cases filtering the relevance signal may reduce the problematic noise, allowing elimination of most spurious snapshots via thresholding or hysteresis.

2.2 Related Work

The vast majority of systems which learn from examples simply do not address the question of where the examples come from. It is assumed either to be simple, uninteresting, or dealt with by someone else.

Snapshot learning with the Top-Cliff heuristic can be viewed as solving a problem of unsupervised learning, since not even examples are provided to the system. Its two-stage design sets it apart from other unsupervised learning approaches, however, since the learning applied to the examples might be anything at all.

There are some related probablistic models in which similar information could be learned, but none with the same inherent inductive biases. These might be applied either to solve the problem as a whole, or to implement the snapshot mechanism, using a modelling framework such as dynamical Bayesian networks[10] or embedded hidden Markov models[11] and carrying out the learning via an algorithm like Expectation-Maximization[4].

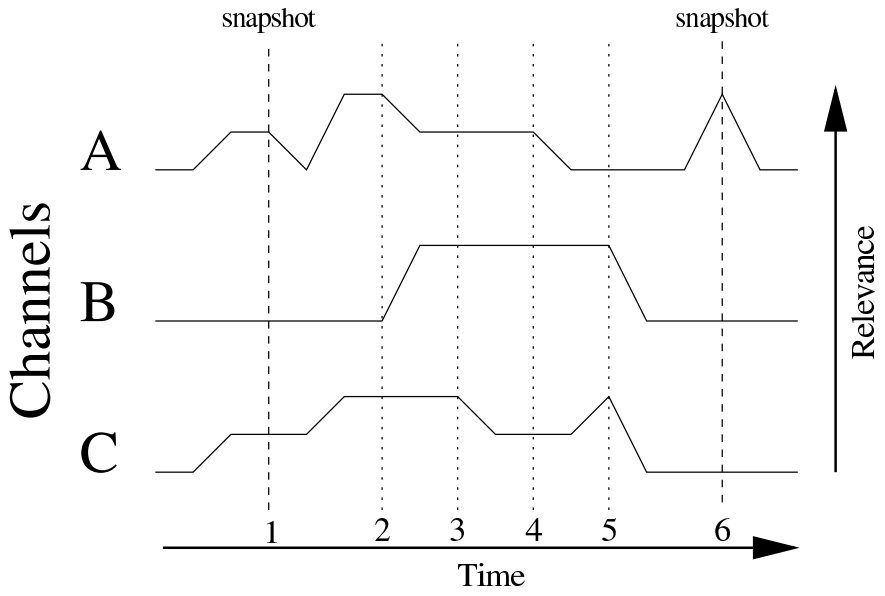


Figure 4: The Top-Cliff heuristic takes a snapshot when all relevant channels have risen since the last time they fell, no channel is rising, and some channel falls. At times 1 and 6, Top-Cliff is satisfied and takes a snapshot. At time 2, channel B is rising, so no snapshot happens. At times 3, 4 and 5, channel A is relevant and has not risen since the last time it fell, so no snapshot happens.

Solving the problem as a whole might be intractable with these techniques, depending on what sort of examples are in the input stream. If used to implement the snapshot mechanism, on the other hand, these approaches are subject to the same set of constraints which led me to develop the Top-Cliff heuristic, and an appropriate set of priors might yield equivalent behavior.

Further, snapshot learning with the Top-Cliff heuristic is fast, simple, and incremental, which many unsupervised learning techniques are not.

Segmenting a stream of input into useful chunks is a focus of much research in sensory domains such as computer vision (e.g. [9]) and speech processing (e.g. [1]). Snapshot learning offers a mechanism by which these methods might be composed with symbolic techniques to enable learning of more abstract relations.

3 Association of Desynchronized Binary Features

I have applied the snapshot learning model to the problem of learning associations between desynchronized binary features. This is the problem which motivated me to develop the snapshot learning model, as I discovered it was a subproblem in my investigation of a larger distributed learning problem:[3] because simultaneous observations of an event may propagate information through the network of learners with variable delays, each learner may see a highly distorted sequence of events.

Figure 5 shows the learning scenario. The world contains scenes composed of a set of binary *features*. Only a few of the possible features are present in any given scene, satisfying sparseness. The agent observes this world via several perceptual channels. Features from the world are conveyed to the agent on some subset of the channels as one or more binary *percepts* on each channel. For example, if an agent with vision, language, and touch channels is presented with a red ball, the "ball" feature might appear as two vision percepts, one language percept and two touch percepts, and the "red" feature as one vision and two language percepts. If all three types of input were instead carried on a single "perception" channel, then "ball" would be conveyed to the agent by five percepts in that channel and "red" by three. Finally, there is a desynchronizer on each channel which applies a delay function (possibly time-varying) to each percept.

Each percept is a learning target; the agent's goal is to discover, for each percept, the set of other percepts which correspond to the same underlying feature in the world. I apply the snapshot learning framework, using the Top-Cliff heuristic to choose snapshots and a form of Hebbian learning[6] to find associations. Hebbian learning is intentionally chosen for its extreme simplicity, since the problem itself is not difficult, given appropriate examples.

A target is modelled as a collection of possibly associated percepts with confidence ratings. Relevance is measured as the number of possible associates

[3] A generalization of communication bootstrapping[3, 2] to larger networks with looser synchronization.

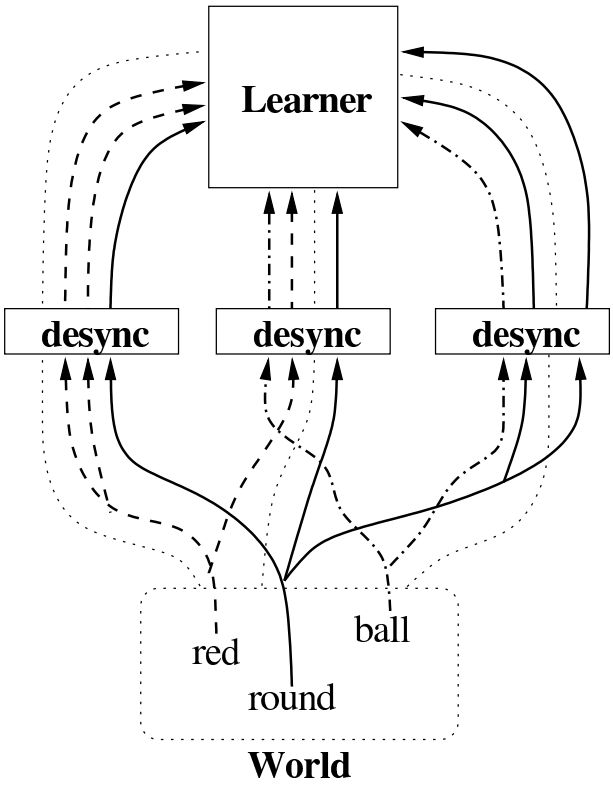


Figure 5: An agent is connected to the world by several channels, each with a desynchronizer which introduces arbitrary delays. Scenes in the world are sets of binary *features*, conveyed to the agent over some subset of the channels as binary *percepts*. For each percept, the goal is to determine which other percepts correspond to the same feature in the world.

present on a channel, except that the target percept is treated as being in its own separate channel, which has relevance 1 when the target percept is present and relevance 0 when it is absent; this extra channel is used to separate positive and negative learning. Each sample (and therefor each example) is the set of percepts currently active.

Given an example, the learner updates the confidence ratings of possible associates using a modified Hebbian learning rule.[6] If the target percept is present, then it is a positive learning update: all possible associates which are present are incremented C^+ units of confidence, and possible associates which are absent are decremented C^- units of confidence. If the target percept is absent, then it is a negative learning update: all possible associates which are present are decremented C^- units of confidence, and absent possible associates are unchanged (otherwise unrelated irrelevant percepts could reinforce each other). When a possible associate's confidence rating drops below a pruning threshold P, it is discarded from the list.

Possible associates are initially acquired by collecting all percepts present during the first period in which the target percept is present and assigning each an initial confidence rating C_0. If pruning ever results in there being no possible associates left, then it is assumed that something has gone wrong in the learning, and the target starts over again with a new initial collection. Since it is possible to miss some associates during the initial collection, percepts which are not possible associates attempt to enter the list, starting at a rating X below the pruning threshold P. If the confidence of one of these external percepts rises to P, then it becomes a possible associate, and if it drops below X it is once again discarded. This is made more difficult (reflecting doubt that latecomers are actually associates) by assigning an additional penalty C^x whenever an external percept is decremented by the Hebbian learning rule.

Finally, the channel from which the target is drawn is treated slightly differently than other channels. Percepts from that channel are likely to be useful in identifying negative examples, since they will generally be associated with percepts in the other channels. Accordingly, the thresholds for these percepts are doubled to slow the rate at which they are pruned or added.

By default, I set the increments $C^+ = C^- = C^x = 1$, the initial confidence $C_0 = 0$, the pruning threshold $P = -3$, and the external percept threshold $X = -10$. Note that because updates are additive, the behavior of the learner is unaffected by shifting or scaling these constants.

3.1 Learning From Examples

When shown a sequence of examples, the system takes snapshots that enable it to rapidly learn the correct set of associations. In this scenario, I used a sequence of examples constructed from a set of $N = 50$ independent binary features. These are perceived over two input channels, A and B, and with each feature corresponding to one binary percept on each channel, giving a total of 100 targets for learning.

For example, the feature "red" might be associated with some arbitrary

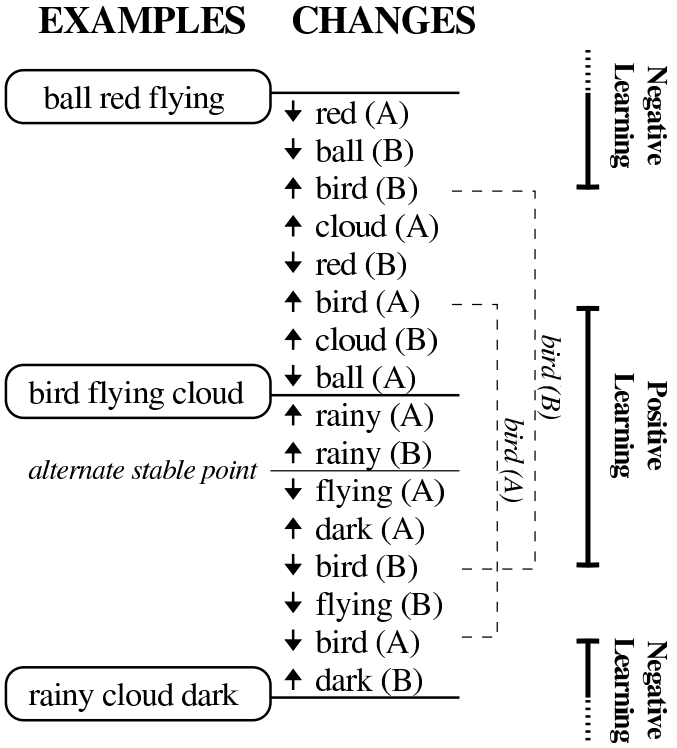


Figure 6: A sequence of examples is perceived as a sequence of percepts appearing and disappearing as one example fades into another. The stable points when a pure example is perceived are ideal moments for learning, but ambiguities prevent these from being recovered in general. Any given target, however, can be learned during the larger periods when the associated percepts are either all present or all absent.

percept `A-1609` in channel A and `B-3462` in channel B. Then when "red" is present in an example, `A-1609` and `B-3462` will be perceived by the system and when red is not present, neither `A-1609` nor `B-3462` will be perceived by the system. The learner's goal is to discover that the target `A-1609` is associated with the percept `B-3642` and vice versa.

Examples are constructed by drawing randomly from the set of features, with the size of each example randomly chosen between $E_{min} = 2$ and $E_{max} = 6$. Each example in turn is presented to the system, as shown in Figure 6. When an example is presented, the system's percepts change from the previous example to the current example one by one in a random order. Once all of the percepts that need to change have changed, the next example is presented.

The moment when one example has finished being presented but another has not yet started is a stable point, the only time when the system's perception

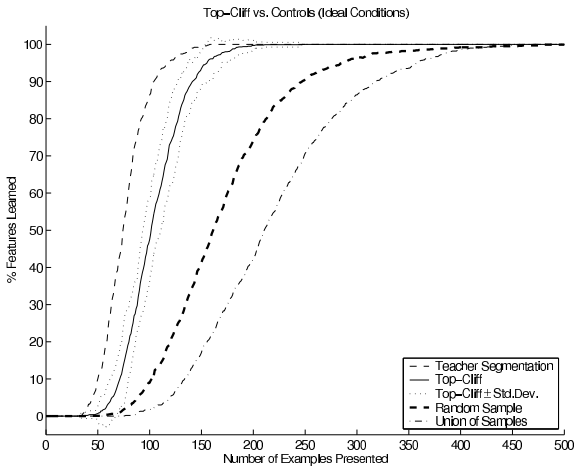


Figure 7: Top-Cliff versus controls solving a simple association problem (100 targets, 50 independent features on two channels, 2-6 features per example) under ideal conditions. The plot shows average behavior over 10 trials.

precisely represents the example and a snapshot at that time would convey the example to the learner perfectly. Although it is not generally possible to recover this stable point from the sequence of changes, any given target can be learned about during a larger period: a snapshot will yield a good example whenever the target's associated percepts are all present or all absent.

I compared learning from snapshots chosen by the Top-Cliff heuristic against three controls: teacher segmentation, random sample, and intersection of unions. Teacher segmentation is perfect example extraction, which magically knows each example the teacher shows, and applies it to all targets. Random sample and union of samples are naive methods which use only positive learning: union of samples forms examples by taking the union of all percepts seen during an interval in which the target percept is present, while random sample chooses one sample of percepts randomly from that interval.

I ran 10 trials of 1000 examples for each system, and compared the results. Unsurprisingly, Top-Cliff performs slightly worse than teacher segmentation, and significantly better random sample and union of samples (Figure 7). Although a precise analysis is difficult, Top-Cliff appears to perform worse that teacher segmentation mainly due to two factors: there are many more percepts identified as possible associates in the initial estimate, and when two examples in a row contain the same feature, its percepts do not drop to trigger learning between the examples. The worse performance of the controls under ideal conditions is partially due to their lack of negative learning, but also due to the lack of precision in selecting an example, which does not improve as the target model improves. Union of samples adds many extra percepts from the neighboring examples, making it harder to weed out the irrelevant percepts. Random

sample also ends up with more percepts from neighboring examples, but also often chooses bad examples, when not all of the associated percepts are present.

Snapshot learning with the Top-Cliff heuristic is not sensitive to the particulars of this scenario. I ran the experiments to verify this, varying system parameters and data sets and running 10 trials of 1000 examples for each set of conditions. For each experiment, I varied one parameter from the base configuration of N=50 features, 2 channels, 1 percept per feature per channel, 2-6 randomly selected features per presented example, random ordering of percept transitions between examples, pruning threshold P=-3, external percept threshold X=-10, and sampling at every transition. Representative results are shown in (Figure 8).

- **Varying pruning threshold P from -1 to -10.** Lowering the pruning threshold slows the learning rate proportionally, since it takes longer to discard the irrelevant percepts. A very high pruning threshold makes it likely that some relevant percepts will be pruned accidentally, however, slowing learning while they are recovered.

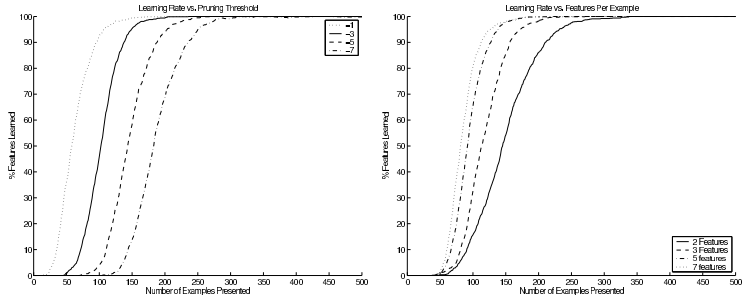
- **Varying external percept threshold X from -3 to -20.** Lowering the the external percept threshold proportionally increases the time to acquire percepts accidentally pruned or not initially considered relevant. When it is too close to P, however, it becomes easy for irrelevant percepts to accidentally become relevant, preventing a stable solution from being reached.

- **Varying expected number of features generated per example from 1 to 9, with variability between zero and ± 4 features per example.** The more features per example, the faster learning proceeds, until violation of the sparseness assumption starts to degrade the utility of each example.

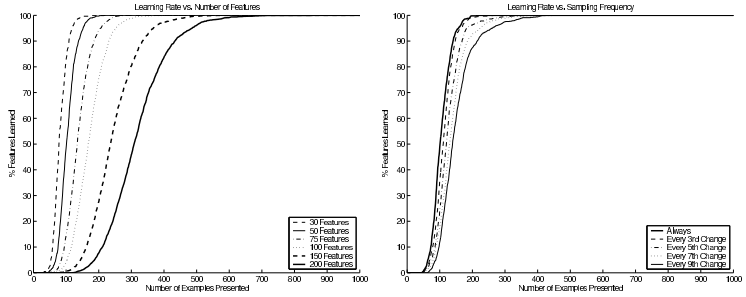
- **Varying number of features N from 20 to 200.** Raising the number of features lowers the learning rate proportionally, dependant on the likelihood of a given feature being in any particular example. When the number of features is too low, however, the sparseness assumption is violated and learning degrades.

- **Varying expected sample rate from every transition to every ninth transition, with variability between zero and ± 4 transitions per sample.** As the system samples its input less frequently, so that multiple transitions happen between samples, the learning rate slows slightly, due to a combination of some events becoming unobservable and the increasing likelihood that an irrelevant percept in a neighboring example will cause some channel to rise when otherwise a Top-Cliff would occur.

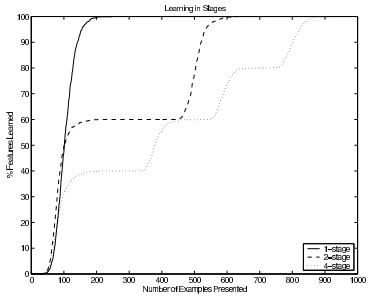
- **Postponing introduction of 1 to 3 groups of features, comprising between 20% and 60% of the total features.** If new learning targets



(a) Pruning Threshold (b) Features Per Example



(c) Number of Features (d) Sampling Frequency



(e) Learning in Stages

Figure 8: Snapshot learning with the Top-Cliff heuristic does not depend critically on any particular parameter value or arrangement of features and presented examples. Varying features, structure of presented examples, sampling, or pruning thresholds has a predictable impact on learning rate. Varying number of channels, number of percepts per feature, allocation of percepts across channels, or transition order has negligible impact.

are introduced to the system after some learning has already taken place, they are learned at the same rate as if they had been introduced at the beginning.

- **Using a deterministic order for percept transitions.** Using a deterministic order for transitions between examples rather than a random order has no significant effect on learning rate.

- **Varying the number of channels from 1-9, percepts per feature on each channel from 0-5, and proportion of channels with percepts for a feature from 22% to 100%.** Changing the number of channels, number of percepts per feature, or allocation of percepts between channels, has no significant effect on learning rate. These may compound the impact of adverse conditions, however.

Snapshot learning with the Top-Cliff heuristic degrades gracefully under adverse conditions (Figure 9). I evaluated the system against two types of degradation (10 trials of 1000 examples, using the same base configuration as before) and found it surprisingly resilient against both. For the first, I blurred examples with a blur factor b, such that any percept present in an example is added to the preceding example with probability b and to the following example with probability b. I applied this recursively, so a percept is smeared across k examples with probability b^k. Even 50% blurring, however, is unable to prevent learning, though it slows it down significantly. For the second, I introduced subtractive noise with a loss rate l, such that any percept which would otherwise be present is deleted with probability l. Still, more than a 30% loss is necessary before the system is unable to reliably learn all of the targets.

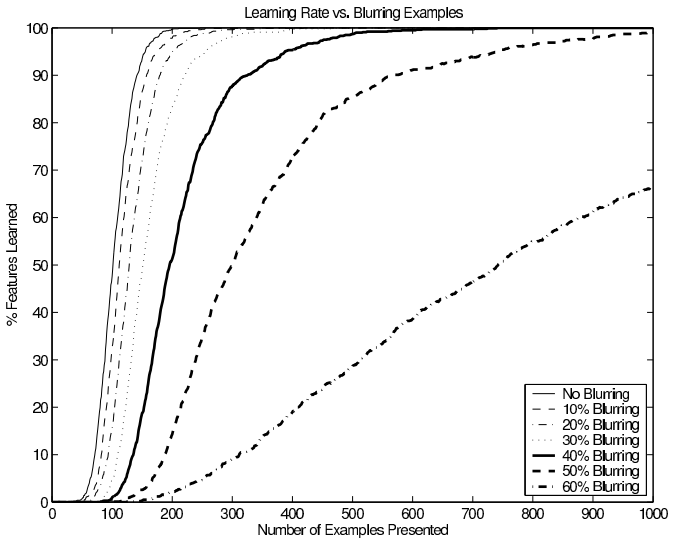
In the degraded scenarios, the naive controls performed dramatically worse than the Top-Cliff heuristic. Figure 10 shows some particularly dramatic conditions, in which the controls never come close to learning all the targets but Top-Cliff still does so fairly rapidly.

Taken together, these experiments suggest that snapshot learning using the Top-Cliff heuristic to select examples is an effective and resilient method for identifying examples.

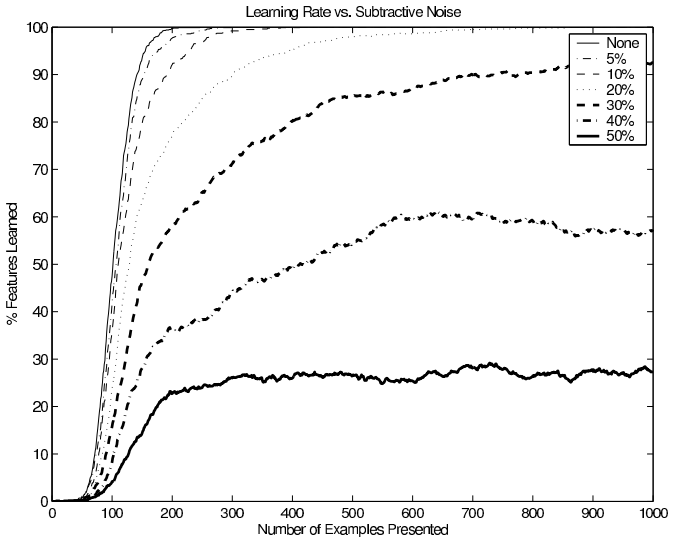
3.2 Learning Without a Teacher

What if there is no teacher supplying examples? Having a teacher means that the system's input is generated from a stream of instructional examples. What's important for learning, however, is not how the input is generated, but how it is perceived by the learner. If there is no teacher, but the input has learnable content that can be broken into examples, then learning can take place as before: the system will still identify learning targets and find snapshots to provide examples for learning about those targets.

I hypothesize that snapshot-learning's assumption that the world is presenting instructional examples will allow it to discover structure in the world even

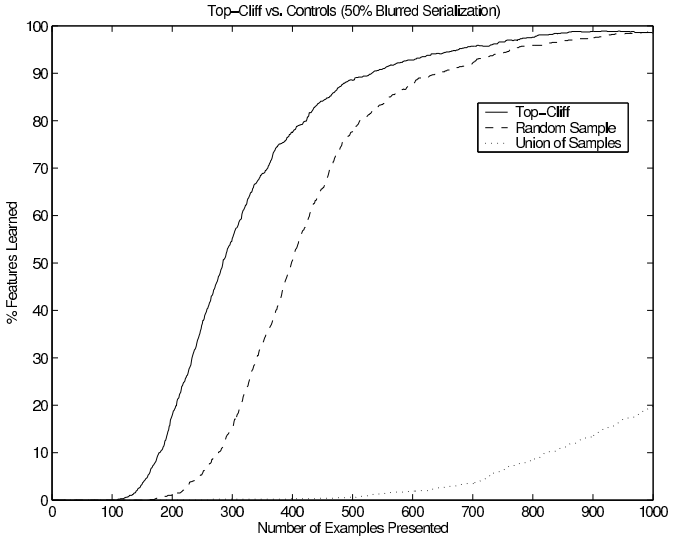


(a) Blurred Examples

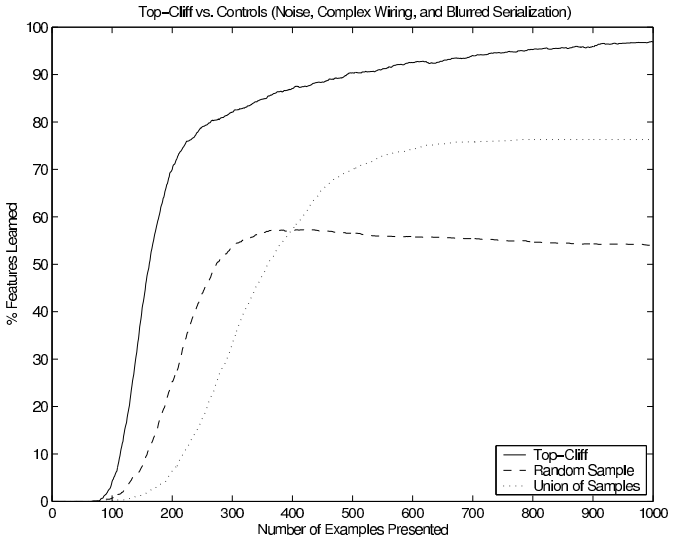


(b) Subtractive Noise

Figure 9: The Top-Cliff heuristic degrades gracefully under adverse conditions for learning. Very high rates of blurring or noise are necessary to prevent successful learning.



(a) 50% blur



(b) 10% noise, 20% blur, 4 percepts/feature across 2 of 3 ports

Figure 10: The Top-Cliff heuristic continues to learn when more complex interconnects, noise, and blurring together of examples prevents the naive controls from finding useful examples.

when the world is not presenting instructional examples, but merely behaving according to clear rules.

To test this hypothesis, I constructed a model of a four-way intersection with a stoplight to generate input for the same system of snapshot-learning using the Top-Cliff heuristic. The system's percepts are the current and previous set of observations from the model. If the hypothesis is correct, then the system should identify examples in its input and learn associations between previous observations and current observations that model the transitions of the stoplight.

The intersection model has five locations: North, South, East, West, and Center, and 11 distinguishable types of vehicles. Each time-step of the model is one second, and the stoplight has a sixty second light cycle: each side is green for 27 seconds, yellow for 3 seconds, then red for 30 seconds while the other side is green and yellow. Cars arrive randomly at any of the four cardinal directions with a mean time between arrivals of 5 seconds, randomly choosing one of the other three directions as an exit goal. Cars are either moving or stopped: a stopped car must spend one second changing state to moving before it can change location. A moving car which isn't blocked takes one second to change locations. Although cars arrive moving, they stop and queue up if there is a car already stopped in front of them or they cannot enter the intersection. Cars enter the intersection when the light is green (and sometimes when it is yellow) and there isn't a car in front of it in the intersection, or when turning right on red and there isn't a car in the center or to the left. Once in the intersection, a car can go to its goal unless it is turning left, and there is a car going the other way in the intersection which is not also turning left.

Percepts from the model arrive over six channels: one for each location, plus a channel for the light. The cardinal directions have percepts for the types of the first car in the queue and the exiting car (if there is one), the center has percepts for the types of whatever cars are there, and the light has percepts for the two active colors. Added to these percepts is a copy of the percepts from the previous second, tagged to be distinguishable from the current percepts. Transitions between sets of percepts take place as a sequence of changes in random order. Figure 11 shows a typical sequence of perceptual states, omitting the previous percepts for clarity.

I ran 10 trials of 1000 seconds each, encompassing $16\frac{2}{3}$ light cycles per trial. There were 122 learning targets (current and previous percepts for 11 vehicle types in each of 5 directions and six lights), for which the Top-Cliff heuristic identified an average of 1048 positive examples and 2105 negative examples per trial, an average of 25.8 examples per target. For the 12 light percept targets, an average of 23.5 positive examples and 4.5 negative examples were identified per trial.

On the basis of these examples, the system learns associated percepts for the light percepts which capture the transitions of the stoplight finite state machine. The cars, on the other hand, have harder to predict behavior and generally no clear association is learned. To evaluate what has been learned, I set a threshold of confidence ≥ 7, and consider only targets which have an associated percept with at least this confidence to have been effectively learned. In the 10 trials,

```
1. (L NS_GREEN) (L EW_RED) (C CONVERTIBLE) (E SEDAN) (W COMPACT)
2. (L NS_YELLOW) (L EW_RED) (N CONVERTIBLE) (E SEDAN) (W COMPACT)
3. (L NS_YELLOW) (L EW_RED) (E SEDAN) (W COMPACT)
4. (L NS_YELLOW) (L EW_RED) (E SEDAN) (W COMPACT)
5. (L NS_RED) (L EW_GREEN) (E SEDAN) (W COMPACT)
6. (L NS_RED) (L EW_GREEN) (E SEDAN) (W COMPACT)
7. (L NS_RED) (L EW_GREEN) (C SEDAN) (C COMPACT) (E MINIVAN) (W SEMI)
8. (L NS_RED) (L EW_GREEN) (C SEMI) (C SEDAN) (E COMPACT) (E MINIVAN)
9. (L NS_RED) (L EW_GREEN) (C SEDAN) (S SEMI) (E MINIVAN)
10. (L NS_RED) (L EW_GREEN) (S SEDAN) (E MINIVAN)
11. (L NS_RED) (L EW_GREEN) (C MINIVAN)
```

Figure 11: A sample perceptual sequence generated from the intersection model (omitting the copy of previous percepts). A convertible finishes driving South to North in the first two seconds, while a compact and a sedan wait at the red light. In the 5th second, the lights change, in the 6th the compact and sedan start moving, and in the 7th they enter the intersection while the cars behind in the queue move up. The compact drives straight through, exiting East, while the sedan is blocked from making its left turn by first the compact, then the semi following it. In the 9th second the sedan starts moving again and exits in the 10th second while the minivan queued up to enter the intersection starts moving again, finally entering the intersection in the 11th second.

there are 126 targets effectively learned, of which 118 are light percepts - the remaining 8 are uninteresting statements involving only cars (e.g. a hatchback in the center is associated with there previously being a hatchback in the center).

Except for two cases, the learned light percepts are associated with only other light percepts — the two exceptions contain a few car percepts which are almost at the pruning threshold. In all cases, the associated light percepts are either uniquely specify a transition (e.g. PREV_EW_GREEN is associated with PREV_NS_RED, EW_GREEN, and NS_RED) or, more frequently, are the union of two uniquely specified transitions (e.g. PREV_EW_GREEN is associated with PREV_NS_RED, EW_GREEN, NS_RED and EW_YELLOW) — Figure 12 shows the associations learned in a typical trial. In every one of the 10 trials, the learned transitions cover all 8 transitions of the stoplight finite state machine, most multiply: Figure 13 shows the finite state machine specified by the associations in Figure 12.

As predicted, the system has extracted useful examples, learned that the behavior of the light is not dependent on the behavior of the cars, and learned associations that model the behavior of the stoplight. This suggests that the hypothesis is correct, and that in fact it is reasonable to learn as though being presented with instructional examples, whether or not any teacher is actually doing so. One cautionary note: the learning exhibited in this case is fragile and does not recover from large mistakes. There are two stable types of associations, superimposed transitions and single transitions in which the light does

```
EW_GREEN = PREV_NS_RED, PREV_EW_GREEN, PREV_NS_YELLOW, NS_RED
EW_YELLOW = PREV_EW_YELLOW, NS_RED, PREV_EW_GREEN, PREV_NS_RED
EW_RED = NS_YELLOW, PREV_EW_RED, PREV_NS_GREEN, NS_GREEN
NS_GREEN = PREV_EW_RED, PREV_NS_GREEN, EW_RED, PREV_EW_YELLOW
NS_YELLOW = PREV_NS_YELLOW, EW_RED, PREV_NS_GREEN, PREV_EW_RED
NS_RED = PREV_NS_RED, PREV_EW_GREEN, EW_GREEN, PREV_NS_YELLOW
PREV_EW_GREEN = PREV_NS_RED, NS_RED, EW_GREEN
PREV_EW_YELLOW = PREV_NS_GREEN, PREV_NS_RED, NS_GREEN EW_RED
PREV_EW_RED = PREV_NS_YELLOW, NS_YELLOW, EW_RED, NS_GREEN, PREV_NS_GREEN
PREV_NS_GREEN = PREV_NS_YELLOW, NS_YELLOW, PREV_EW_RED, EW_RED, NS_GREEN
PREV_NS_YELLOW = EW_GREEN, NS_RED, PREV_EW_RED, NS_YELLOW
PREV_NS_RED = PREV_EW_RED, EW_RED, PREV_EW_YELLOW, NS_GREEN
```

Figure 12: Example of associations learned during one trial in which at least one associated percept has confidence ≥ 7. Collectively, these associations specify the finite state machine showed in Figure 13.

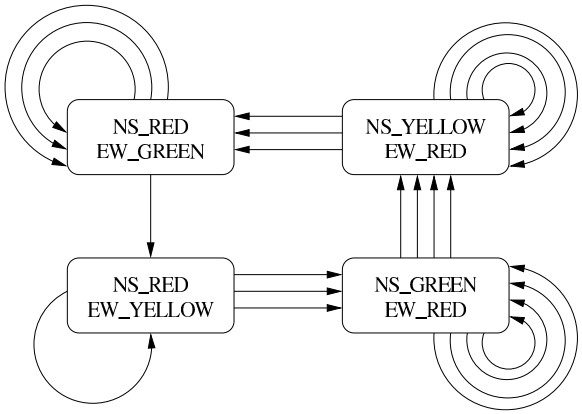


Figure 13: Finite state machine specified by the learned associations shown in Figure 12. Multiple arrows indicate multiple associations specifying a transition.

not change, and mutilating an association can cause a transition from superimposed to single, but not the reverse. This could probably be corrected, however, by biasing the system towards learning about cause and effect.

4 Conclusions

The snapshot learning framework transforms the problem of learning from streams of perceptual input into problems of extracting examples relevant to a learning target, and learning from those examples. Feedback between the learning targets and the snapshot mechanism improves the quality of snapshots as the quality of the target model improves, and allows fast, incremental learning.

Learning from examples is a well studied problem, while extracting relevant examples has been largely assumed away or studied in isolation, ignoring how to interface extracted models with different types of learning techniques. Furthermore, identifying good examples relevant to particular learning targets, as required by snapshot learning, may be a significantly simpler problem than identifying examples which are good in general. The Top-Cliff heuristic provides one mechanism for incrementally identifying auspicious snapshots by applying a generalized notion of peaks to relative measures of input relevance.

I applied the snapshot learning framework, using the Top-Cliff heuristic, to learn associations between desynchronized binary features, a problem caused by varying propagation times in a distributed learning network. The resulting system learns rapidly and is robust against changes to its parameters and adverse learning conditions such as large amounts of noise and blurred examples. Moreover, the system discovers examples even when none are provided, which can allow it to learn about the behavior it is observing: when exposed to a simulation of a four-way intersection, the system reliably learns associations which specify the finite state machine of the traffic light.

The results of these experiments suggest that the snapshot learning framework and the top-cliff heuristic are powerful and important tools for learning. The ability to extract discrete examples from a stream of perception may help bridge the gap between real-world data and machine learning techniques which demand carefully segmented and annotated examples.

More importantly, the ability to find useful examples in the absence of a teacher suggests that it may be possible deploy supervised learning methods to solve unsupervised learning problems. More analysis is necessary to clearly understand the properties and limitations of the snapshot learning framework and the top-cliff heuristic, and the framework needs to tested in application to other problems.

References

[1] Kannan Achan, Sam Roweis, and Brendan Frey. "A Segmental HMM for Speech Waveforms." Technical Report UTML-TR-2004-001, University of

Toronto, January 2004.

[2] Jacob Beal. "An Algorithm for Bootstrapping Communications" International Conference on Complex Systems (ICCS), June 2002.

[3] Jacob Beal. "Generating Communications Systems Through Shared Context" MIT AI Technical Report 2002-002, January, 2002.

[4] Arthur Dempster, Nan Laird, and Donald Rubin. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society*, Series B, 39(1):1-38, 1977.

[5] Zoubin Ghahramani. "Unsupervised Learning." In Bousquet, O., Raetsch, G. and von Luxburg, U. (eds) *Advanced Lectures on Machine Learning* LNAI 3176. Springer-Verlag, 2004.

[6] D.O. Hebb. *The Organization of Behaviour.* John Wiley & Sons, New York, 1949

[7] Simon Kirby. "Language evolution without natural selection: From vocabulary to syntax in a population of learners." Edinburgh Occasional Paper in Linguistics EOPL-98-1, 1998. University of Edinburgh Department of Linguistics.

[8] Simon Kirby. "Learning, Bottlenecks and the Evolution of Recursive Syntax." Linguistic Evolution through Language Acquisition: Formal and Computational Models edited by Ted Briscoe. Cambridge University Press, in Press.

[9] Marina Meila and Jianbo Shi. "Learning segmentation by random walks." In Neural Information Processing Systems 13, 2001.

[10] Kevin Murphy. "Dynamic Bayesian Networks: Representation, Inference and Learning." UC Berkeley, Computer Science Division, July 2002.

[11] Radford Neal, Matthew Beal and Sam Roweis. "Inferring State Sequences for Non-linear Systems with Embedded Hidden Markov Models." Neural Information Processing Systems 16 (NIPS'03). pp. 401-408, 2003.

[12] Patrick Winston. "Learning Structural Descriptions from Examples." MIT AI Technical Report 231, 1970.