Planning to Control Crowd-Sourced Workflows

Daniel S. Weld
University of Washington

Crowdsourcing

- Obtaining ideas / content by soliciting contributions from a large group of people
- Combine the efforts of volunteers/part-time workers (each contributing a small portion) which adds to a relatively large or significant result

Commonality:
Large Tasks from Micro-Contributions

- Challenges
  - Small work units
  - Reliability & skill of individual workers vary
- Therefore
  - Use workflow to aggregate results & ensure quality
  - Manage workers with (unreliable) workers

How Motivate People to Contribute?

- Community
- Self-Interest
- Fun
- Money

Ex: Iterative Improvement

(initial artifact) → improve → vote → output

[Little et al., 2010]
c2  success stories, not motivation

cse, 6/7/2013
Iterative Improvement

First version
A partial view of a pocket calculator together with some coins and a pen.

Version after 8 iterations
A CASIO multi-function, solar powered scientific calculator.
A blue ball point pen with a blue rubber grip and the tip extended.
Six British coins; two of £1 value, three of 20p value and one of 1p value.
Seems to be a theme illustration for a brochure or document cover treating finance - probably personal finance.

Workflow Control Problem

How many times?

A Familiar Diagram

Agent
Sensors

Percepts

Environment

Actuators

Actions

• Observe ballot

• Post IMPROVE job

• Post BALLOT job

• Return answer

World Representation

Artifact quality
Q ∈ [0, 1]

Problem difficulty
D ∈ [0, 1]
c1 say 'goal' seq process
cse, 6/7/2013
Belief States

Artifact quality
\[ Q \in [0, 1] \]

Approximate with Beta distribution, Truncated normal or Discretized approximation

Partially-Observable MDP

- **World State**: Quality of artifact(s), problem difficulty, ...
- **Belief State**: Probability distribution over world states
- **Actions**: Submit jobs to labor mkt & observe results
  - Eg, improve job prob distribution on new artifact
  - Eg, ballot job Bayesian update on quality
    EM update on difficulty, worker diligence
- **Objective**: Maximize \( E[R(w) - \Sigma c] \)

Outline

- Introduction
- Perception
- Controlling Iterative Improvement
- Controlling Taxonomy Generation
- Controlling Citizen Science
- POMDPs for the Masses

Interpreting Sensing Actions

Is this bird an Indigo Bunting?

- Yes
- No

Submit

Majority Voting

\[ \text{accuracy}_{w}(d) = \frac{1}{2}[1+(1-d)^{1/w}] \]

Assume: no malevolence

Probability of a Correct Answer

Assume: no malevolence

diligence

Majority vote of 8 Turkers better than expert labeling
Probability of a Correct Answer

\[
\text{accuracy}_w(d) = \frac{1}{2}(1+(1-d)^{1/\gamma})
\]

Assume: no malevolence

Unsupervised Learning

[Dawid and Skene, 1979; Whitehill et al, 2009; Lin et al 2012, etc]

- No labeled data
- Joint estimation of all parameters: Expectation Max.

Effect of Probabilistic Model

Reduces cost by 50%

Iterative Improvement POMDP

- **World State:** Quality of artifact(s), problem difficulty
- **Belief State:** Probability distribution over world states
- **Actions:** Submit jobs to labor mkt & observe results
  - Eg, improve job prob distribution on new artifact
  - Eg, ballot job Bayesian update on quality
    EM update on difficulty, worker diligence
- **Objective:** Maximize \( E[R(w) - \Sigma c] \)
c4 cut plate model
cse, 6/7/2013
Transition Model of Ballot Action

Prior(quality_{\alpha_1})  \rightarrow  \text{Worker votes that artifact 1 is better} \rightarrow  \text{Posterior(quality_{\alpha_1})}

Prior(quality_{\alpha_2})  \rightarrow  \text{Posterior(quality_{\alpha_2})}

Solving the POMDP

- Beliefs over continuous world state
- Compare algorithms
  - Fixed lookahead search with beta distributions for belief states
  - ADBS – approximate belief state with fixed discretization & solve resulting MDP with VI
  - UCT on discretized space

POMDP for Iterative Improvement

- Submit artifact
- \( Y \): improving?
- Better of \( \alpha \) and \( \alpha' \)
- Generate improved job
- Estimate quality of \( \alpha' \)
- More voting
- Make ballot job
- Update quality estimates
- N: improving?
- Better of \( \alpha \) and \( \alpha' \)
- Generate improved job
- Estimate quality of \( \alpha' \)
- More voting
- Make ballot job
- Update quality estimates
- Y: improving?
- Better of \( \alpha \) and \( \alpha' \)
- Generate improved job
- Estimate quality of \( \alpha' \)
- More voting
- Make ballot job
- Update quality estimates
- N: improving?
- Better of \( \alpha \) and \( \alpha' \)
- Generate improved job
- Estimate quality of \( \alpha' \)
- More voting
- Make ballot job
- Update quality estimates
POMDP for Iterative Improvement

submit architecture

initial architecture

[Diagram showing decision-making process with options for improve, estimate quality of \( \alpha \) and \( \alpha' \), make ballot job, update quality estimates, and choosing between \( \alpha \) and \( \alpha' \).]

better of \( \alpha \) and \( \alpha' \)

need improving?

estimate quality of \( \alpha \) and \( \alpha' \)

generate improve job

make ballot job

update quality estimates

better of \( \alpha \) and \( \alpha' \)

need improving?

estimate quality of \( \alpha \) and \( \alpha' \)

generate improve job

make ballot job

update quality estimates

better of \( \alpha \) and \( \alpha' \)

need improving?

estimate quality of \( \alpha \) and \( \alpha' \)

generate improve job

make ballot job

update quality estimates

better of \( \alpha \) and \( \alpha' \)

need improving?

estimate quality of \( \alpha \) and \( \alpha' \)

generate improve job

make ballot job

update quality estimates

better of \( \alpha \) and \( \alpha' \)

need improving?

estimate quality of \( \alpha \) and \( \alpha' \)

generate improve job

make ballot job

update quality estimates

better of \( \alpha \) and \( \alpha' \)

need improving?

estimate quality of \( \alpha \) and \( \alpha' \)

generate improve job

make ballot job

update quality estimates

better of \( \alpha \) and \( \alpha' \)
POMDP for Iterative Improvement

Comparison

40 images, same average cost

Controlling quality: POMDP 30% cheaper
Observation: Ballot Use

Outline

- Introduction
- Perception
- Controlling Iterative Improvement
- Controlling Taxonomy Generation
- Controlling Citizen Science
- POMDPs for the Masses

Image Data Sets

Q&A Site Responses

Iterative Improvement

Initial Approach 2: Category Comparison

Problems
1. The growing hierarchy becomes overwhelming
2. Workers confused

Lesson: Decompose the task into smaller steps

Problem
Without context it’s hard to judge relationships:
- flying vs. flights
- TSA liquids vs. removing liquids
- Packing vs. what to bring

Lesson: Don’t compare abstractions to abstractions
summarize lessons
introduce different kinds of tasks
it seems like only kind of workflow in talkd

design patterns
binary choice, multi-label
cse, 6/7/2013
Cascade Overview

Use the crowd to:
1. Generate category names
2. Select the best categories
3. Place the data into the best categories

Use machines to:
4. Infer global structure of categories

Input: 100 Random Colors

Step 1. Sample Data

Step 1. Generate Categories

For each color

Task

What category do you suggest for this color?

Crowd responses

green greenish Light pastel aqua lime

This generates an initial set of category names.

Step 2. Select Best Categories

For each color

Task

What is the best category for this color?

Crowd responses

Aqua: 1/5 Greenish: 4/5 Lime: 0/5 Pastel: 0/5

An early filter for spam and vague categories

Step 3. Label Data

For each color and category

Task

What categories does this belong to?

Crowd responses

Category

Votes

Green: 4/5 Greenish: 5/5 Yellow: 2/5 Pink: 0/5

This determines category membership.

Step 4. Global Structure Inference

Determine parent/child relations; eliminate duplicates.
Finally, ... Recurse

Blue:  
Light Blue:  
Green:  
Other:  

Evaluation

Categories Shared with Experts

Decision-Theoretic Control!

Why is Cascade Expensive?

3 crowd steps

Generate SelectBest Categorize

1 machine step

But do we really need all these votes?
What’s the best order to ask them?

Why is Cascade Expensive?

Generate SelectBest Categorize

Agent Belief State

• Probability distribution for which labels apply
  – If a worker says X is “professional athlete”
  – Then increase confidence that X is “person”
  – Prioritize asking if X is “football player”
  – Downgrade asking if X is “vehicle”
• Must learn label co-occurrence model during plan execution

Cycle

• Given new item
  – Do until confident:
    • Ask worker about most interesting label (VOI)
    • Probabilistic inference to update posteriors
  – Expectation maximization
    • Determine which labels apply
    • Update accuracy models for workers

Probabilistic Models

Independent
Joint (naive Bayes)
Ising (MRF / LKJ)
Savings

- 60/165 votes to reach same F score
- 74% savings

Outline

- Citizen Science

Hierarchical Task Networks

- Partially-ordered set of tasks  \( \Rightarrow \) Parallel execution
- Dynamic workflow switching [Lin AAAI12, UAI12]
- Recursive expansion
  - Preconditions & resources
  - Eg. availability of workers with required skills

CLOWDER

Other Future Work

1. Use Cascade to organize:
   1. jobs available on oDesk
   2. eGov’t suggestions
   3. Product reviews
   4. Free-form survey comments
2. Standardize “human task primitives” for future workflows
Related Work

Collaborative Taxonomies
- CardSorting
- Wikipedia

Crowdsourcing Workflows:
- TurKit [Little, UIST’10]
- Soylent [Bernstein, UIST’10]
- Mobi [Zhang, CHI’12]
- CrowdForge[Kittur, CHI’12]

Optimization of Workflows
- [Shahaf&Horvitz AAAI’10]
- [Kamar et al. AAMAS’12]

Conclusion

- Decision-theoretic methods for scaling crowdsourcing
  - POMDP model, EM & reinforcement learning
  - Objective: DT compiler to make techniques accessible

- Novel workflow for decentralized taxonomy construction
  - Global view from locally informed microwork

Barbados

World Domination?

Specifying a POMDP

- Actions
  - Worker tasks – user has to specify anyway
- Transition & Observation Probabilities
  - Learned from experience (vs PDDL)
- World State
  - POMDP specific, unintuitive \(\rightarrow\) templates
- Utility Function
  - Implicit, \(F(\text{world state})\) \(\rightarrow\) utility elicitation
- Control Guidance

RELATED WORK

- PPDDL, RDDL
- Alisp [Andre et al, 2002]
- A²BL [Simpkins et al, 2008]
- Adaptive Programs [Pinto et al, 2010]
Utility / Control Advice

• Users specify utility procedurally!
  – Utility is implicit in their control program
  – E.g., “majority vote of three people”
  – Vs “83% probability threshold”
  – Let alone utility curve

• Procedural behavior also good for roll-out policy

How Compose Utility?

• Want to encapsulate sub-behaviors

(define-behavior X ...
  (do A ...
  (choose-between B or C...
  (do iterative-improvement ...))

POAPS Primitives

A primitive is a ten-tuple \((D, R, \Omega, T, O, I, C, D_U, R_U, F)\), where:

- \(D = D^1 \times ... \times D^n\) is a set of domain states.
- \(R\) is a set of range states.
- \(\Omega\) is a set of observations.
- \(T : D \times R \rightarrow [0, 1]\) is a transition function.
- \(O : R \times \Omega \rightarrow [0, 1]\) is an observation function.
- \(I\) is an \(n\)-dimensional indicator vector indicating which of the \(D^i\) are observable.
- \(C : D \rightarrow \mathbb{R}^+\) is a cost function.
- \(D_U = D_U^1 \times ... \times D_U^n\) is a set of user domain states.
- \(R_U\) is a set of user range states.
- \(F : D_U \rightarrow R_U\) is a user function.

POAPS Language

(define (improve text)
  (choose
    (improve (c-imp text))
    text)))

Call by Poaps Value Semantics

Compile into HAM

Primitive: c-imp

- \(F(\alpha \in D_U) = \text{call} o\text{API}(\alpha)\)
- \(D_U, R_U\) is the set of all artifacts \(\alpha\).

- \(D = R = [0, 1]\)
- \(T(q \in D, q' \in R) = P(q'|q)\)
- \(I = (0)\)
- No observations/observation function

Step 1: Define a set of states \(S(p)\)

(define (improve text)
  (choose
    (improve (c-imp text))
    text)))

A state variable for every argument

Not Shown: State variables for called functions
(Recursive Definition)
Step 2: Construct a HAM

\[
\text{define (improve text)} \\
\text{(choose (improve (c-imp text)) text))}
\]

Step 3: Merge

- Final State Space: \( S(p) + \text{States of HAM} \)
- Actions – Given by HAM
- Transitions – Ensure “Call by Poaps value semantics”
- Observations – given by primitives
- Costs – given by primitives

Conclusion

- How bring RL to the masses?
- Compose dynamical systems?
- Specify & compose utility functions?
- RL algorithms?

```python
def iterativeCascade(tips):
catsAndMembers = []
tipsToRun = [tips]
while(len(tipsToRun) > 20):
    tipSubset = createTipSubset(tipsToRun)
catsAndMembersForSubset = cascade(tipSubset)
catsAndMembers = pruneTaxonomy(catsAndMembersForSubset)
    newCategories = catsAndMembersForSubset.getCategories()
catsAndMembers = newCategories + catsAndMembersForSubset
    catsAndMembers.add(catsAndMembersForSubset)
    newCategories = catsAndMembersForSubset.getCategories()
catsAndMembers = pruneTaxonomy(catsAndMembers)
tipsToRun.remove(tipSubset)
tipsToRun.add(tipSubset)
return generateTaxonomy(catsAndMembers)

def cascade(tips):
suggestedCategories = generateCategories(tips)
bestSuggestedCategories = getBestSuggestedCategories(tips, suggestedCategories)
allTipsWithCategories = createDictionaryTipsToCategories(tips, bestSuggestedCategories, k = 7)
catsAndMembersFirstPass = categorize(allTipsWithCategories)
catsAndMembersSecondPass = categorize(catsAndMembersFirstPass)
return catsAndMembersSecondPass
```

Existing Tools Inadequate

- No support for
  - Workflow planning & optimization
  - Worker modeling & parameter learning
  - Adaptive workflow execution
- Managing Turkers is like ... herding cats...

Clowder
Adaptive Programs

```plaintext
for (i = 1; i < N; i++) {
    c = randomContext();
    m = move.suggest(c);
    reward(payoff(c,m));
}
```

Primitive: c-imp

- $F(\alpha \in D_U) = \text{calltoAPI}(\alpha)$
- $D_U, R_U$ is the set of all artifacts $\alpha$.

- $D = R = [0, 1]$
- $T(q \in D, q' \in R) = P(q'|q)$
- $C = 0.05$
- No observations/observation function
- $I = (0)$

POAPS Primitives

A primitive is a ten-tuple: $(D, R, \Omega, F, I, C, D_U, R_U, F)$, where:
- $D = D_U \times \ldots \times D_U$ is a set of domain states.
- $R$ is a set of range states.
- $\Omega$ is a set of observations.
- $F : D \times R \rightarrow [0, 1]$ is a transition function.
- $O : R \times \Omega \rightarrow [0, 1]$ is an observation function.
- $I$ is a n-dimensional indicator vector indicating which of the $D$ are observable.
- $C : D \rightarrow R^+$ is a cost function.
- $D_U = D_U \times \ldots \times D_U$ is a set of user domain states.
- $R_U$ is a set of user range states.
- $F : D_U \rightarrow R_U$ is a user function.

POAPS LANGUAGE

```plaintext
(define (improve text)
    (choose
        (improve (c-imp text))
        text))
```

Call by Poaps Value Semantics
**COMPILATION**

- Input: program \( p \)

\[
\text{(define (improve text)}
\begin{align*}
 & \text{(choose}\n & \text{(improve (c-imp text))}\n & \text{text}))\n\end{align*}
\]

**Step 1: Define a set of states \( S(p) \)**

- A state variable for every argument

\[
\text{(define (improve text)}
\begin{align*}
 & \text{(choose}\n & \text{(improve (c-imp text))}\n & \text{text}))\n\end{align*}
\]

- A state variable for every sub-expression

Not Shown: State variables for called functions (Recursive Definition)

**Step 2: Construct a HAM**

\[
\text{(define (improve text)}
\begin{align*}
 & \text{(choose}\n & \text{(improve (c-imp text))}\n & \text{text}))\n\end{align*}
\]

**Step 3: Merge**

- Final State Space: \( S(p) + \) States of HAM
- Actions – Given by HAM
- Transitions – Ensure “Call by Poaps value semantics”
- Observations – given by primitives
- Costs – given by primitives

\[
\text{(define (vote q a0 a1 c0 c1)}
\begin{align*}
 & \text{(choose}\n & \text{(if (ask-crowd q a0 a1)}
 & \text{(vote q a0 a1 (+ c0 1) c1)}\n & \text{(vote q a0 a1 (+ c1 1))}\n & \text{(if (> c0 c1)}
 & \text{True}\n & \text{False)})})\n\end{align*}
\]

\[
\text{(define (it-i image worse-text better-text)}
\begin{align*}
 & \text{(choose}\n & \text{(it-i image better-text}\n & \text{(c-imp better-text))}\n & \text{(if (vote image better-text}\n & \text{worse-text 0 0)}\n & \text{(it-i image worse-text better-text)}\n & \text{(it-i image better-text worse-text))}\n & \text{better-text})})\n\end{align*}
\]