Policy Shaping: Integrating Human Feedback with Reinforcement Learning

Shane Griffith
Department of Computer Science
Georgia Institute of Technology
sgriffith7@gatech.edu

Kaushik Subramanian
Department of Computer Science
Georgia Institute of Technology
ksubrama@cc.gatech.edu

Jonathan Scholz
Department of Computer Science
Georgia Institute of Technology
jkscholz@gatech.edu

Charles L. Isbell
School of Interactive Computing
Georgia Institute of Technology
isbell@cc.gatech.edu

Andrea Thomaz
School of Interactive Computing
Georgia Institute of Technology
athomaz@cc.gatech.edu

Abstract

A long term goal of Interactive Reinforcement Learning is to incorporate non-expert human feedback to solve complex tasks. Some state-of-the-art methods have approached this problem by mapping human information to rewards and values and iterating over them to compute better control policies. In this paper we argue for an alternate and more effective characterization of human feedback: Policy Shaping. We introduce **Advise**, a Bayesian approach that attempts to maximize the information gained from human feedback by utilizing it as direct policy labels.

We compare **Advise** to state-of-the-art approaches using a series of experiments. These experiments use two classic arcade games, together with feedback from a simulated human teacher, which allows us to systematically test performance under a variety of cases of infrequent and inconsistent feedback. We show that **Advise** has similar performance to the state of the art, but is more robust to a noisy signal from the human and fairs well with an inaccurate estimate of its single input parameter. With these advancements this paper may help to make learning from human feedback an increasingly viable option for intelligent systems.

Keywords: Interactive Reinforcement Learning, Human-Agent Interaction, Human-in-the-loop learning

1 Introduction

A long–term goal of machine learning is to create systems that can be interactively trained or guided by non-expert end-users. This paper focuses specifically on integrating human feedback with Reinforcement Learning. One way to address this problem is to treat human feedback as a shaping reward. Yet, recent papers have observed that a more effective use of human feedback is as direct information about policies [1, 2]. Most techniques for learning from human feedback still, however, convert feedback signals into a reward or a value. In this paper we introduce *Policy Shaping*, which formalizes the *meaning* of human feedback as policy feedback, and demonstrates how to use it directly as policy advice. We also introduce **Advise**, an algorithm for estimating a human's Bayes optimal feedback policy and a technique for combining this with the policy formed from Bayesian Q-Learning[1].

We validate our approach using a series of experiments. These experiments use a simulated human teacher and allow us to systematically test performance under a variety of cases of infrequent and inconsistent feedback. The results demonstrate two advantages of **Advise**: 1) it is comparable to or outperforms state of the art techniques for integrating human feedback with Reinforcement Learning; and 2) by formalizing human feedback, we avoid ad hoc parameter settings and make **Advise** robust to infrequent and inconsistent feedback.

2 Reinforcement Learning

Reinforcement Learning (RL) defines a class of algorithms for solving problems modeled as a Markov Decision Process (MDP). An MDP is specified by the tuple (S, A, T, R), which defines the set of possible world states, S, the set of actions available to the agent in each state, A, the transition function $T : S \times A \rightarrow \Pr[S]$, a reward function $R : S \times A \rightarrow \mathbb{R}$, and a discount factor $0 \leq \gamma \leq 1$. The goal of RL is to identify a policy, $\pi : S \rightarrow A$, that maximizes reward.

This paper used an implementation of the Bayesian Q-learning (BQL) RL algorithm [4]. BQL maintains parameters that specify a normal distribution with unknown mean and precision for each Q value, $Q[s, a]$, which represents an estimate of the long-term expected discounted reward for taking action a in state s. This representation approximates the agent's uncertainty in the optimality of each action, which makes the problem of optimizing the exploration/exploitation trade-off straightforward. Because the Normal-Gamma (NG) distribution is the conjugate prior for the normal distribution, the mean and the precision are estimated using a NG distribution with hyperparameters $\langle \mu_0^{s,a}, \lambda^{s,a}, \alpha^{s,a}, \beta^{s,a} \rangle$. These values are updated each time an agent performs an action a in state s, accumulates reward r, and transitions to a new state s'. Details on how these parameters are updated can be found in [4].

The NG distribution for each Q value can be used to estimate the probability that each action $a \in A_s$ in a state s is optimal, which defines a policy, π_R, used for action selection. The optimal action can be estimated by sampling each $\hat{Q}(s, a)$ and taking the max. A large number of samples can be used to approximate the probability an action is optimal by simply counting the number of times an action has the highest Q value [4].

3 Related Work

A key feature of RL is the use of a reward signal. The reward signal can be modified to suit the addition of a new information source (this is known as *reward shaping* [5]). This is the most common way human feedback has been applied to RL. However, several difficulties arise when integrating human feedback signals that may be infrequent, or occasionally inconsistent with the optimal policy–violating the necessary and sufficient condition that a shaping function be potential-based [5]. Another difficulty is the ambiguity of translating a statement like "yes, that's right" or "no, that's wrong" into a reward. Typically, past attempts have been a manual process, yielding *ad hoc* approximations for specific domains. Researchers have also extended reward shaping to account for idiosyncrasies in human input. For example, a drift parameter can account for the human tendency to give less feedback over time [6].

Advancements in recent work sidestep some of these issues by showing human feedback can instead be used as policy feedback. For example, Thomaz and Breazeal [1] added an *UNDO* function to the negative feedback signal, which forced an agent to backtrack to the previous state after its value update. Work by Knox and Stone [2, 7] has shown that a general improvement to learning from human feedback is possible if it is used to directly modify the action selection mechanism of the RL algorithm. Although both approaches use human feedback to modify an agent's exploration policy, they still treat human feedback as either a reward or a value (e.g., "right" becomes $+1$ and "wrong" -1). In our work, we assume human feedback is making a direct statement about the policy itself, rather than influencing the policy through a reward.

4 Policy Shaping

We use feedback labels directly to infer what the human believes is the optimal policy of action in the previous state. We assume a human providing feedback knows the right answer, but noise in the feedback channel introduces inconsistencies between what the human intends to communicate and what the agent observes. Thus, feedback is consistent, $\mathcal{C}$, with the optimal policy with probability $0 < \mathcal{C} < 1$. We also assume that a human watching an agent learn may not provide feedback after every single action, thus the likelihood, $\mathcal{L}$, of receiving feedback has probability $0 < \mathcal{L} < 1$. In the event feedback is received, it is meant as a comment on the optimality of the immediately preceeding action.

Although many different actions may be optimal in a given state, we will assume for this paper that the human knows only one optimal action, which is the one they intend to communicate. In that case, an action, a, is optimal in state s

[1] A longer version of this paper appears in NIPS 2013 [3].

if no other action is optimal. The probability s, a is optimal can be obtained by application of Bayes' rule in conjuction with the binomial distribution and enforcing independence conditions arising from our assumption that there is only one optimal action. This gives: $\mathcal{C}^{\Delta_{s,a}}(1-\mathcal{C})^{\sum_{j\neq a}\Delta_{s,j}}$, where $\Delta_{s,a}$ is the difference between the number of "right" and "wrong" labels. We take this equation to be the probability of performing s, a according to the feedback policy, π_F (i.e., the value of $\pi_F(s,a)$). This is the Bayes optimal feedback policy given the "right" and "wrong" labels seen, the value for $\mathcal{C}$, and that only one action is optimal per state.

Because the use of **Advise** assumes an underlying RL algorithm will also be used, the policies derived from multiple information sources must be reconciled. Before an agent is completely confident in either policy, it has to determine what action to perform using the policy information each provides. We combine the policies from multiple information sources by multiplying them together: $\pi \propto \pi_R \times \pi_F$. Multiplying distributions together is the Bayes optimal method for combining probabilities from (conditionally) independent sources [8]. Note that BQL can only approximately estimate the uncertainty that each action is optimal from MDP reward. Rather than use a different combination method to compensate for the fact that BQL converges too quickly, we introduced the exploration tuning parameter, θ, from [9], that can be manually tuned until BQL performs close to optimal.

5 Experimental Setup

We evaluate our approach using two game domains, Pac-Man and Frogger, with a simulated oracle. Pac-Man consists of a 5x5 grid world with two food pellets, one ghost, walls, and the Pac-Man avatar. The goal is to eat all the pellets while avoiding the ghost. Points are awarded for each pellet (+10) and winning the game (+500). Points are taken away as time passes (-1) and for losing the game (-500). The action set consisted of the four primary cartesian directions. The state representation included Pac-Man's position, the position and orientation of the ghost, and the presence of pellets.

Frogger consists of a 4x4 grid world with two moving cars, two water hazards, and the Frogger avatar. The goal is to cross the road without being run over or jumping into a water hazard. Each car drives one space per time step. The car placement and direction of motion is randomly determined at the start and does not change. As a car disappears off the end of the map it reemerges at the beginning of the road and continues to move in the same direction. The cars moved only in one direction, and they started out in random positions on the road. Each lane was limited to one car. Points are won for arriving at a safe spot on the far side (+500). Points are lost as time passes (-1), for being run over (-500), and for hopping into a water hazard (-500). The action set consisted of the four primary cartesian directions and a stay-in-place action. The state representation included frogger's position and the position of the two cars.

A simulated oracle was used in the place of human feedback, because this allows us to systematically vary the parameters of feedback likelihood, $\mathcal{L}$, and consistency, $\mathcal{C}$ and test different learning settings in which human feedback is less than ideal. The oracle was created manually by a human before the experiments by encoding the optimal action in each state. For states with multiple optimal actions, a small negative reward (-10) was added to the MDP reward of the extra optimal actions to preserve the assumption that only one action be optimal in each state.

6 Experiments

6.1 A Comparison to the State of the Art

In this evaluation we compare Policy Shaping with **Advise** to the more traditional Reward Shaping, as well as recent Interactive RL techniques. Knox and Stone [2, 7] tried eight different strategies for combining feedback with an environmental reward signal and they found that two strategies, *Action Biasing* and *Control Sharing*, consistently produced the best results. Both of these methods convert human feedback to a value but recognize that the information contained in that value is policy information.

Action Biasing, Control Sharing, and Reward Shaping can all be defined using the same set of parameters and variables. Positive and negative feedback is declared a reward r_h, and $-r_h$, respectively. A table of values, $H[s,a]$ stores the feedback signal for s, a. The value $\mathrm{B}[s,a]$ controls the influence of feedback on learning, and is incremented by a constant b when feedback is received for s, a, and is decayed by a constant d at all other time steps.

Action Biasing modifies the action selection of BQL to be $\mathrm{argmax}_a \hat{Q}(s,a) + \mathrm{B}[s,a] * H[s,a]$. Control Sharing defines a transition between π_R and a feedback policy as the probability $P(a = \mathrm{argmax}_a H[s,a]) = min(\mathrm{B}[s,a], 1.0)$[2]. Reward Shaping modifies the MDP reward function to be $R'(s,a) \leftarrow R(s,a) + \mathrm{B}[s,a] * H[s,a]$.

We compared the methods using four different combinations of feedback likelihood, $\mathcal{L}$, and consistency, $\mathcal{C}$, in Pac-Man and Frogger, for a total of eight experiments[3] [4]. Table 1 summarizes the quantitative results. In the ideal case of frequent and correct feedback ($\mathcal{L} = 1.0$; $\mathcal{C} = 1.0$), we see in Table 1 that **Advise** does much better than the other methods early

[2]Control Sharing interprets feedback as a reward, but it does not use that information, so is unaffected if its magnitude changes.

[3]We manually tuned all the parameters before the experiments to maximize MDP reward. BQL:$\langle \mu_0^{s,a} = 0, \lambda^{s,a} = 0.01, \alpha^{s,a} = 1000, \beta^{s,a} = 0.0000\rangle$, $\theta = 0.0001$ for Frogger, and $\theta = 0.5$ for Pac-Man. Discount factor: $\gamma = 0.99$. Action Biasing, Control Sharing, and Reward Shaping: $b = 1$, $d = 0.001$, for Action Biasing $r_h = 100$, and for Reward Shaping $r_h = 100$ in Pac-Man and $r_h = 1$ in Frogger.

[4]We used the conversion $r_h = 1$, 10, 100, or 1000 that maximized MDP reward in the ideal case to also evaluate the three cases of non-ideal feedback. We had to use $r_h = 1.0$ for Reward Shaping in frogger because the agent can end up in infinite loops when feedback is less than ideal. This was not a problem in Pac-Man because the ghost can force Pac-Man out of oscillatory behavior.

	Ideal Case		Reduced Consistency		Reduced Frequency		Moderate Case	
	($\mathcal{L}=1.0, \mathcal{C}=1.0$)		($\mathcal{L}=0.1, \mathcal{C}=1.0$)		($\mathcal{L}=1.0, \mathcal{C}=0.55$)		($\mathcal{L}=0.5, \mathcal{C}=0.8$)	
	Pac-Man	Frogger	Pac-Man	Frogger	Pac-Man	Frogger	Pac-Man	Frogger
BQL + Action Biasing	0.58 ± 0.02	0.16 ± 0.05	-0.33 ± 0.17	0.05 ± 0.06	0.16 ± 0.04	0.04 ± 0.06	$\mathbf{0.25 \pm 0.04}$	0.09 ± 0.06
BQL + Control Sharing	0.34 ± 0.03	0.07 ± 0.06	-2.87 ± 0.12	-0.32 ± 0.13	0.01 ± 0.12	0.02 ± 0.07	-0.18 ± 0.19	0.01 ± 0.07
BQL + Reward Shaping	0.54 ± 0.02	0.11 ± 0.07	-0.47 ± 0.30	0 ± 0.08	0.14 ± 0.04	0.03 ± 0.07	0.17 ± 0.12	0.05 ± 0.07
BQL + **Advise**	$\mathbf{0.77 \pm 0.02}$	$\mathbf{0.45 \pm 0.04}$	$\mathbf{-0.01 \pm 0.11}$	0.02 ± 0.07	$\mathbf{0.21 \pm 0.05}$	$\mathbf{0.16 \pm 0.06}$	0.13 ± 0.08	$\mathbf{0.22 \pm 0.06}$

Table 1: Comparing the learning rates of BQL + **Advise** to BQL + Action Biasing, BQL + Control Sharing, and BQL + Reward Shaping. Each entry represents the average and standard deviation of the cumulative reward in 300 episodes, expressed as the percent of the maximum possible cumulative reward for the domain with respect to the BQL baseline. Negative values indicate performance worse than BQL. Bold values indicate the best performance for that case.

in the learning process. A human reward that does not match both the feedback consistency and the domain may fail to eliminate unnecessary exploration and produce learning rates similar to or worse than RL on its own. **Advise** avoided these issues by not converting feedback into a reward.

The remaining results in Table 1 show performance for each of the non-ideal conditions that we tested: reduced feedback consistency ($\mathcal{L} = 1.0$; $\mathcal{C} = 0.55$), reduced frequency ($\mathcal{L} = 0.1$; $\mathcal{C} = 1.0$), and a case that we call moderate ($\mathcal{L} = 0.5$; $\mathcal{C} = 0.8$). Action Biasing and Reward Shaping performed comparably to **Advise** in two cases. Action Biasing does better than Advise in one case in part because the feedback likelihood is high enough to counter Action Biasing's overly influential feedback policy. This gives the agent an extra push toward the goal without becoming detrimental to learning (e.g., causing loops). In its current form, **Advise** makes no assumptions about the likelihood the human will provide feedback.

The results in Table 1 comprehensively show that **Advise** always performed at or above the BQL baseline, which indicates robustness to less than ideal feedback. In contrast, Action Biasing, Control Sharing, and Reward Shaping blocked learning progress in several cases with reduced consistency (column 3 has the most extreme example). Control Sharing performed worse than BQL in three cases. Action Biasing and Reward Shaping performed worse than BQL in one case.

Having a prior estimate of the feedback consistency, $\mathcal{C}$, allows **Advise** to balance what it learns from the human appropriately with its own learned policy. We could have provided the known value of $\mathcal{C}$ to the other methods, but doing so would not have helped set r_h, b, or d. These parameters had to be tuned since they only slightly correspond to $\mathcal{C}$. We manually selected their values with ideal feedback, and then used those same settings for the other cases. However, different values for r_h, b, and d may produce better results in the cases with reduced $\mathcal{L}$ or $\mathcal{C}$. We tested this next.

6.2 How The Reward Parameter Affects Action Biasing

Here, we test how Action Biasing performed with a range of values for r_h for the case of moderate feedback ($\mathcal{L} = 0.5$ and $\mathcal{C} = 0.8$), and for the case of reduced consistency ($\mathcal{L} = 1.0$ and $\mathcal{C} = 0.55$). Control Sharing was left out of this evaluation because changing r_h did not affect its learning rate. Reward Shaping was left out of this evaluation due to the problems mentioned in Section 6.1. The conversion from feedback into reward was set to either $r_h = 0$, 500, or 1000.

The results in Fig. 1 show that a large value for r_h is appropriate for more consistent feedback; a small value for r_h is best for reduced consistency. This is clear in Pac-Man when a reward of $r_h = 1000$ led to better-than-baseline learning performance in the moderate feedback case, but decreased learning rates dramatically below BQL in the reduced consistency case. In that case, the use of $r_h = 0$ produced the best results. Therefore, r_h depends on feedback consistency.

This experiment also shows that the best value for r_h is somewhat robust to a slightly reduced consistency. A value of either $r = 500$ or 1000, in addition to $r = 100$ (see Table 1), can produce good results with moderate feedback in both Pac-Man and Frogger. The use of a human influence parameter $\mathrm{B}[s, a]$ to modulate the value for r_h is presumably meant to help make Action Biasing more robust to reduced consistency. The value for $\mathrm{B}[s, a]$ is, however, increased by b whenever feedback is received, and reduced by d over time; b and d are more a function of the domain than the information in accumulated feedback. Our next experiment demonstrates why this is bad for IRL.

6.3 How Domain Size Affects Learning

Action Biasing, Control Sharing, and Reward Shaping use a 'human influence' parameter, $\mathrm{B}[s, a]$, that is a function of the domain size more than the amount of information in accumulated feedback. To show this we froze the parameters and evaluated the algorithms in a larger domain. Frogger was increased to a 6×6 grid with four cars. An oracle was created automatically by running BQL to 50,000 episodes 500 times, and then for each state choosing the action with the highest value. The oracle provided moderate feedback ($\mathcal{L} = 0.5$; $\mathcal{C} = 0.8$) for the 33360 different states identified in this process.

Our results (omitted due to space constraints) show that, whereas **Advise** performed roughly the same as in the smaller Frogger domain (see the last column in Table. 1), Action Biasing, Control Sharing, and Reward Shaping all had a negligible effect on learning, performing roughly the same as the BQL baseline. In order for those methods to perform as well as they did with the smaller version of Frogger, the value for $\mathrm{B}[s, a]$ needs to be set higher and decayed more slowly by manually finding new values for b and d. Thus, like r_h, the optimal values to b and d are dependent on both the domain and the quality of feedback. In contrast, the estimated feedback consistency, $\hat{\mathcal{C}}$, used by **Advise** only depends on the true feedback consistency, $\mathcal{C}$. For comparison, we next show how sensitive **Advise** is to a suboptimal estimate of $\mathcal{C}$.

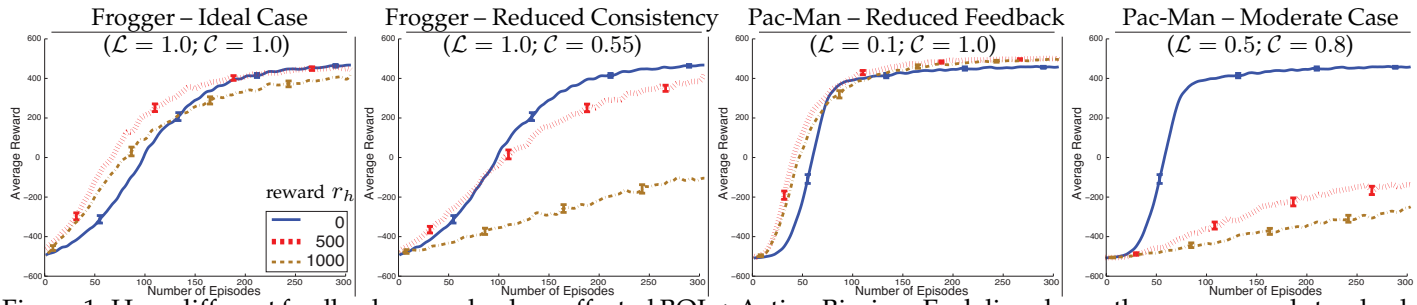

Figure 1: How different feedback reward values affected BQL + Action Biasing. Each line shows the average and standard error of 500 learning curves over a duration of 300 episodes. See the text for more details.

6.4 Using an Inaccurate Estimate of Feedback Consistency

Interactions with a real human will mean that in most cases **Advise** will not have an exact estimate, $\hat{\mathcal{C}}$, of the true feedback consistency, $\mathcal{C}$. It is presumably possible to identify a value for $\hat{\mathcal{C}}$ that is close to the true value. Any deviation from the true value, however, may be detrimental to learning. This experiment shows how an inaccurate estimate of $\mathcal{C}$ affected the learning rate of **Advise**. Feedback was generated with likelihood $\mathcal{L} = 0.5$ and a true consistency of $\mathcal{C} = 0.8$. The estimated consistency was either $\hat{\mathcal{C}} = 1.0$, 0.8, or 0.55.

Our results (omitted due to space constraints) show that in both Pac-Man and Frogger using $\hat{\mathcal{C}} = 0.55$ reduced the effectiveness of **Advise**. The learning curves are similar to the baseline learning curves because using an estimate of $\mathcal{C}$ near 0.5 is equivalent to not using feedback at all. In general, values for $\hat{\mathcal{C}}$ below $\mathcal{C}$ decreased the possible gains from feedback. In contrast, using an overestimate of $\mathcal{C}$ slightly boosted learning rates for these particular domains and case of feedback quality. In general, however, overestimating $\mathcal{C}$ can lead to a suboptimal policy especially if feedback is provided very infrequently. Therefore, it is desirable to use $\hat{\mathcal{C}}$ as close to its true value, $\mathcal{C}$, as possible.

7 Conclusion and Future Work

Overall, our experiments indicate that it is useful to interpret feedback is as a direct comment on the optimality of an action, without converting it into a reward or a value. **Advise** performed comparably to or better than tuned versions of Action Biasing, Control Sharing, and Reward Shaping. These methods are outperformed because they first convert feedback into a reward, which reduces the effectiveness of the information. Their performance also suffers because their use of 'human influence' parameters is disconnected from the amount of information in the accumulated feedback. In contrast, **Advise** has only one input parameter, which is independent of the domain, and can be used to calculate the exact amount of information in the accumulated feedback in each state. **Advise** combines the feedback policy with the RL policy using the right amount of influence. It also always utilizes information from both sources.

In conclusion, this paper defined the Policy Shaping paradigm for integrating feedback with Reinforcement Learning. We introduced **Advise**, which tries to maximize the utility of feedback using a bayesian approach to learning. **Advise** produced results on par with or better than the current state-of-the-art IRL techniques, showed where those approaches fail while **Advise** is unaffected, and it demonstrated robustness to infrequent and inconsistent feedback. We plan to extend our work by computing $\hat{\mathcal{C}}$ online as a human interacts with an agent, and addressing other aspects of human feedback like errors in credit assignment.

References

[1] A. L. Thomaz and C. Breazeal, "Teachable robots: Understanding human teaching behavior to build more effective robot learners," *Artificial Intelligence*, vol. 172, no. 6-7, pp. 716–737, 2008.

[2] W. B. Knox and P. Stone, "Combining manual feedback with subsequent MDP reward signals for reinforcement learning," in *AAMAS*, pp. 5–12, 2010.

[3] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. Thomaz, "Policy Shaping: Integrating Human Feedback with Reinforcement Learning," in *NIPS*, pp. 2625–2633, 2013.

[4] R. Dearden, N. Friedman, and S. Russell, "Bayesian Q-learning," in *AAAI*, pp. 761–768, 1998.

[5] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, pp. 341–348, 1999.

[6] C. L. Isbell, C. Shelton, M. Kearns, S. Singh, and P. Stone, "A social reinforcement learning agent," in *Agents*, pp. 377–384, 2001.

[7] W. B. Knox and P. Stone, "Reinforcement learning from simultaneous human and MDP reward," in *AAMAS*, pp. 475–482, 2012.

[8] C. Bailer-Jones and K. Smith, "Combining probabilities." GAIA-C8-TN-MPIA-CBJ-053, July 2011.

[9] T. Matthews, S. D. Ramchurn, and G. Chalkiadakis, "Competing with humans at fantasy football: Team formation in large partially-observable domains," in *AAAI*, pp. 1394–1400, 2012.