JMLR: Workshop and Conference Proceedings 24:143–153, 2012 10th European Workshop on Reinforcement Learning

An investigation of imitation learning algorithms for structured prediction

Andreas Vlachos ANDREAS.VLACHOS@CL.CAM.AC.UK
Computer Laboratory, University of Cambridge, UK.

Editor: Marc Peter Deisenroth, Csaba Szepesvári, Jan Peters

Abstract

In the imitation learning paradigm algorithms learn from expert demonstrations in order to become able to accomplish a particular task. Daumé III et al. [2009] framed structured prediction in this paradigm and developed the search-based structured prediction algorithm (SEARN) which has been applied successfully to various natural language processing tasks with state-of-the-art performance. Recently, Ross et al. [2011] proposed the dataset aggregation algorithm (DAGGER) and compared it with SEARN in sequential prediction tasks. In this paper, we compare these two algorithms in the context of a more complex structured prediction task, namely biomedical event extraction. We demonstrate that DAGGER has more stable performance and faster learning than SEARN, and that these advantages are more pronounced in the parameter-free versions of the algorithms.

Keywords: Real-world Applications, Imitation Learning, Natural Language Processing, Structured Prediction.

1. Introduction

Imitation learning algorithms aim at learning controllers from demonstrations by human experts [Schaal, 1999; Abbeel, 2008; Syed, 2010]. Unlike standard reinforcement learning algorithms [Sutton and Barto, 1996], they do not require the specification of a reward function by the practitioner. Instead, the algorithm observes a human expert perform a series of actions to accomplish the task in question and learns a policy that "imitates" the expert with the purpose of generalizing to unseen data. These actions have dependencies between them, since earlier ones affect the input to the following ones and the algorithm needs to handle the discrepancy between the actions of the expert in the demonstration during training and the actions predicted by the learned controller during testing. Imitation learning algorithms have been applied successfully to a variety of domains and tasks including autonomous helicopter flight [Coates et al., 2008] and statistical dialog management [Syed and Schapire, 2007].

For structured prediction tasks in natural language processing (NLP) the output space of an instance is a structured group of labels [Smith, 2011]. For example, in part-of-speech tagging the output for a sentence is a sequence of part-of-speech tags, or in handwriting recognition the output is a sequence of characters. Structures can often be more complex than sequences, for example in syntactic dependency parsing the output space for a sentence is a set of labeled edges spanning the words. While learning methods such as Conditional Random Fields [Lafferty et al., 2001] and Markov Logic Networks [Domingos and Lowd,

2009] have been developed for structured prediction, they are usually tailored to particular structures so that they can perform parameter learning and inference efficiently. As a result, a common approach that performs well in practice is to decompose structured prediction into multiple classification tasks, in which each label of the structured output is predicted by a classifier.

Under this prism, learning for structured prediction can be viewed as learning a controller whose actions are to output each of the labels of the structured output. Similar to the controllers in reinforcement learning, these actions have dependencies between them (e.g. in part-of-speech tagging, determiners are commonly followed by nouns instead of verbs, or in handwriting recognition some character sequences are more likely than others) which must be taken into account in order to achieve good performance. The training signal commonly provided is a set of labeled instances produced by a human expert, which is akin to the human demonstrations in the imitation learning paradigm. Daumé III et al. [2009] proposed an imitation learning algorithm[1], search-based structured prediction (SEARN), that reduces the problem of learning a model for structured prediction into learning a set of classifiers. This reduction enables SEARN to tackle structured prediction tasks with complex output spaces, and it has been applied successfully to a variety of tasks including summarization [Daumé III et al., 2009] and biomedical event extraction [Vlachos and Craven, 2011].

In this work, we investigate a novel imitation learning algorithm proposed by Ross et al. [2011], dataset aggregation (DAGGER) that also reduces the problem of learning structured prediction to classification learning. It was compared to SEARN on learning video game-playing agents and handwriting recognition and was shown to be more stable and have faster learning while achieving state-of-the-art performance.

In this paper we make the following contributions. We present SEARN and DAGGER in a unified description, highlighting the connections between imitation learning and structured prediction. We then compare them in the context of biomedical event extraction [Kim et al., 2011], a task in which the structure of the output is more complex than a sequence of labels, and confirm the aforementioned advantages of DAGGER over SEARN which are more pronounced in the parameter-free versions of the algorithms. Furthermore, we explore the effect of the learning rate on the balance between precision and recall achieved by the algorithms. We believe that these contributions are relevant to applications of imitation learning algorithms to other structured prediction tasks, as well as to the development and evaluation of imitation learning algorithms.

2. Imitation learning algorithms for structured prediction

SEARN and DAGGER form the structured output prediction of an instance s as a sequence of T actions $\hat{y}_{1:T}$ made by a learned policy H. Each action $\hat{y}_t$ can use features from s and all previous actions $\hat{y}_{1:t-1}$, thus exploiting possible dependencies. The number of actions taken for an instance is not defined in advance but it depends on the actions chosen.

Algorithm 1 presents the training procedure for SEARN and DAGGER. Both algorithms require a set of labeled training instances $\mathcal{S}$ and a loss function ℓ that compares structured

1. SEARN infers rewards using the loss function and the labeled instances, therefore it is better described as an apprenticeship learning algorithm [Syed, 2010]. Note though that this distinction between imitation and apprenticeship learning is not consistent among authors [Abbeel, 2008].

Algorithm 1: Imitation learning training

Input: training data $\mathcal{S}$, *expert policy* $\pi^\star$, loss function ℓ, learning rate β, CSC learner *CSCL*
Output: policy H_N

1 Examples $E = \emptyset$
2 **for** $i = 1$ **to** N **do**
3 $p = (1-\beta)^{i-1}$
4 current policy $\pi = p\pi^\star + (1-p)H_{i-1}$
5 **if** SEARN **then**
6 Examples $E = \emptyset$
7 **for** s ***in*** S **do**
8 Predict $\pi(s) = \hat{y}_{1:T}$
9 **for** $\hat{y}_t$ ***in*** $\pi(s)$ **do**
10 Extract features $\Phi_t = f(s, \hat{y}_{1:t-1})$
11 **foreach** *possible action* y_t^j **do**
12 **if** SEARN **then**
13 Predict $y\prime_{t+1:T} = \pi(s|\hat{y}_{1:t-1}, y_t^j)$
14 **else**
15 Predict $y\prime_{t+1:T} = \pi^\star(s|\hat{y}_{1:t-1}, y_t^j)$
16 Estimate $c_t^j = \ell(\hat{y}_{1:t-1}, y_t^j, y\prime_{t+1:T})$
17 Add (Φ_t, c_t) to E
18 Learn a policy $h_i = CSCL(E)$
19 **if** SEARN **then**
20 $H_i = \beta \sum_{j=1}^{i} \frac{(1-\beta)^{i-j}}{1-(1-\beta)^i} h_j$
21 **else**
22 $H_i = h_i$

output predictions of instances in $\mathcal{S}$ against the correct output for them. In addition, an *expert policy* $\pi^\star$ must be specified which is a function that returns the optimal action $\hat{y}_t$ for the instances in the training data, which is akin to an expert demonstrating the task. An action is optimal when it minimizes the loss over the instance given the previous actions $\hat{y}_{1:t-1}$ assuming that all future actions $\hat{y}_{t+1:T}$ are also optimal. $\pi^\star$ is typically derived from the labeled training instances (e.g. in handwriting recognition $\pi^\star$ returns the correct character for each position) and it must be able to deal with mistaken $\hat{y}_{1:t-1}$. Both algorithms output a learned policy H that is a classifier, which unlike the expert policy $\pi^\star$, it can generalize to unseen data. Finally, the learning rate β and a cost sensitive classification (CSC) learner (*CSCL*) must be provided. In CSC each training instance has a vector of misclassification costs associated with it, thus rendering some mistakes on some instances to be more expensive than others [Domingos, 1999].

Each training iteration of both algorithms begins by setting the probability p (line 3) of using $\pi^\star$ in the current policy π. In the first iteration only $\pi^\star$ is used but in later iterations

π becomes stochastic as for each action we use $\pi^\star$ with probability p and the learned policy from the previous iteration h_{i-1} with probability $1-p$ (line 4). Then π is used to predict each instance s in the training data (line 8). For each s and each action $\hat{y}_t$, a CSC example is generated (lines 10-17). The features Φ_t are extracted from s and the previous actions $\hat{y}_{1:t-1}$ (line 10) and are desinged to be good predictors of the current action $\hat{y}_t$. For example, in part-of-speech tagging, commonly used features include the word whose part of speech label we are predicting, the words preceding it and following it, as well as the label predicted for the previous word. The cost for each possible action y_t^j is estimated by predicting the remaining actions $y\prime_{t+1:T}$ in s using either π or $\pi^\star$ (line 13 or 15) and calculating the loss incurred given that action w.r.t. the correct output using ℓ (line 16). The features for each timestep together with the costs for each possible action at that timestep (Φ_t, c_t) form one CSC example (line 17). The CSC examples obtained from all the training instances are used by a CSC learning algorithm to learn a policy h_i (line 18) which is combined with the previously learned ones to form the new policy H_i.

In each iteration, the algorithm predicts the instances in the training data and estimates the cost of each action. This procedure is commonly referred to as inverse reinforcement learning [Abbeel and Ng, 2004], since unlike standard reinforcement learning, an expert policy is given but we try to learn the reward for each action (cost). Note that the learned policy from the previous iteration is used in generating the CSC examples in predicting each training data instance (line), as well as estimating the cost for each action in the case of SEARN (line 13). The degree to which it is used depends on the probability p set in the beginning of each iteration. By gradually decreasing the use of the expert policy in the current policy, both algorithms adapt the learned policy to its own predictions.

The main algorithmic difference between SEARN and DAGGER is in the learning of the classifiers h_i in each iteration and in combining them into a policy H_i. Under SEARN, each h_i is learned using only the CSC examples generated in iteration i (line 6) and is then combined with the classifiers learned in the previous iterations $h_{1:i-1}$ according to the learning rate β (line 20). On the other hand, DAGGER learns h_i using CSC examples from iterations $1:i$ and uses it as the learned policy H_i (line 22). Thus DAGGER can combine the training signal obtained from all iterations more flexibly, which results in faster learning and more stable performance compared to SEARN.

Another difference between the two algorithms is that DAGGER uses the expert policy $\pi^\star$ to predict the remaining actions in $y\prime_{t+1:T}$.[2] This approach to costing had been proposed by Daumé III et al. [2009] in the context of SEARN, referred to as optimal approximation.

The learning rate β determines how fast the current policy π moves away from $\pi^\star$. A special case is obtained when $\beta = 1$, also referred to as pure policy iteration or parameter-free. In this case, $\pi^\star$ is used only in the first iteration to reproduce the correct output and π only uses only the learned policy from the previous iteration H_{i-1}. Furthermore, the classifier combination under SEARN becomes the same as the one of DAGGER, i.e. only the most recently learned classifier is used. In this setting the algorithms cannot query $\pi^\star$ after the first iteration, thus $\pi^\star$ does not need to handle mistakes in previous actions since all actions are optimal in the first iteration. Furthermore, the predictions in lines 8 and 12 become deterministic. However, relying only on the learned policy for action prediction

2. This fact was pointed out by a reviewer as it was not mentioned by Ross et al. [2011].

beyond the first iteration (note that the cost estimation still uses the correct output via the loss function) renders the learning harder as the algorithms are given less supervision.

3. Tackling biomedical event extraction with imitation learning

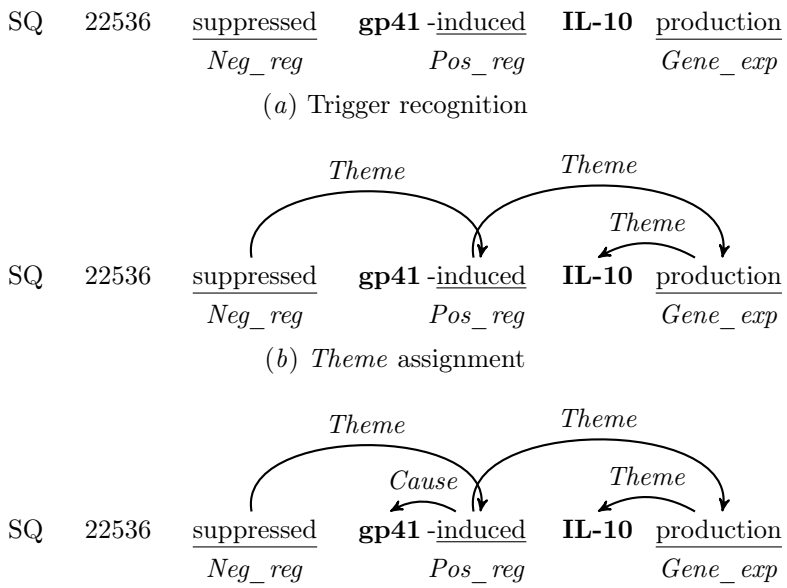
Vlachos and Craven [2012] developed a biomedical event extraction approach trained with Searn that achieved the second-best reported performance on the data from the recent BioNLP 2011 shared task (BioNLP11ST) [Kim et al., 2011]. Therefore, we decided to use the BioNLP11ST setup to compare empirically Searn and DAgger. In this section we describe briefly the task and the approach, but the interested reader is referred to the article by Vlachos and Craven [2012] for more details.

The term biomedical event extraction is used to refer to the task of extracting descriptions of actions and relations among one or more entities from the biomedical literature. In BioNLP11ST each event consists of a trigger and one or more arguments, the latter being proteins or other events. Protein names are annotated in advance and any token in a sentence can be a trigger for one of the nine event types. Depending on their event types, triggers are assigned theme and cause arguments. In an example demonstrating the complexity of the task, given the passage "...SQ 22536 suppressed gp41-induced IL-10 production in monocytes", systems should extract the three appropriately nested events listed in Figure 3.1(*d*). Performance is measured using Recall, Precision and F-score over complete events, i.e. the trigger, the event type and the arguments all must be correct in order to obtain a true positive.

In our approach, we treat each sentence independently and decompose event extraction in four stages: trigger recognition, theme assignment, cause assignment and event construction (Fig. 3.1).[3] Apart from the last one which is rule-based, each stage has its own module to perform the classification needed. The basic features used are extracted from the lemmatization and the syntactic parse of the sentence. Furthermore, we extract structural features for each action from the previous ones, for example the trigger recognition label of the previous token is used as a feature to predict the label for the current token.

The loss function sums the number of false positive and false negative events, following the task evaluation. The expert policy for a sentence is derived from the correct events contained in the training data and returns the action that minimizes the loss over the sentence given the previous actions and assuming that all future actions are optimal. In the first iteration it returns the actions required to reproduce the correct events in the sentence. In subsequent iterations, in order to deal with mistakes in previous actions, it avoids assigning arguments to incorrectly tagged triggers and avoids using incorrect events as arguments of other events. It it important to note that the same events can be obtained using different sequences of actions and that the correct events in the training data specify only one such sequence. For example in Figure 3.1, token "SQ" could have been tagged as a trigger without assigning any arguments to it and we ould still obtain the correct events. In order to resolve this ambiguity, we restrict the expert policy to return only the the triggers that are necessary to produce the correct events.

3. While different task decompositions are possible, this 4-stage decomposition is the most commonly used one among the systems participating in BioNLP11ST.



(*a*) Trigger recognition

(*b*) *Theme* assignment

(*c*) *Cause* assignment

ID	type	*Trigger*	*Theme*	*Cause*
E1	*Neg_reg*	suppressed	*E2*	
E2	*Pos_reg*	induced	*E3*	gp41
E3	*Gene_exp*	production	IL-10	

(*d*) Event construction

Figure 3.1: The stages of our biomedical event extraction system.

The classifiers learned for trigger recognition, theme assignment and cause assignment, are unlikely to be able to replicate to the expert policy due to the difficulty of the tasks. In such cases, the optimal approximation method for costing (line 15 in Alg. 1) is unlikely to estimate the cost of each action correctly, since the actions predicted by the learned policy are likely to differ from the ones returned by the expert policy. Therefore, in our experiments we use the SEARN-style cost estimation (line 13 in Alg. 1) for both SEARN and DAGGER. In order to restrict the effects of the stochasticity of this approach, we use the *focused costing* method [Vlachos and Craven, 2011], in which the cost estimation for an action takes into account only the part of the output graph connected with that action, thus limiting the part of the structured output considered for this purpose.

The structure of the output space for the biomedical event extraction decomposition described above is a sequence of tags defining triggers and their event types, combined with a directed acyclic graph in which vertices correspond to triggers or proteins and edges represent argument assignments. Thus it is more complex than the handwriting recognition task that was used in the comparison performed by Ross et al. [2011] which is a sequential tagging task. Also, it is not amenable to several commonly used structured prediction methods which rely on the output structure being a sequence or a tree in order to perform inference efficiently. Furthermore, incorrect actions can prohibit other correct actions from

being taken, e.g. if a token is incorrectly predicted not to be a trigger, it is impossible to assign arguments to it. Again, this is unlike sequential tagging tasks, where an incorrect prediction of a token does not prohibit the correct prediction of the remaining ones. Both algorithms under comparison use the learned policies from previous iterations in order to generate training examples (line 9 in Alg 1), therefore incorrectly predicted actions can inhibit the algorithm from reaching regions of the output space that could provide useful CSC examples. Thus, the learning rate which determines how frequently the expert policy is queried is likely to be more important for biomedical event extraction than it was in the comparisons of Ross et al. [2011].

4. Experiments

In our experiments we run Searn and DAgger for 12 training iterations and perform CSC learning using the online passive-aggressive (PA) algorithm [Crammer et al., 2006]. BioNLP11ST comprises three datasets – training, development and test – which consist of five full articles each and 800, 150 and 260 abstracts respectively. We extract features from the output of the syntactic parser by McClosky [2010] as provided by the shared task organizers [Stenetorp et al., 2011]. The use of this publicly available resource allows for easy replication of our experiments. While the correct output is provided for the training and development datasets, evaluation on the test dataset is only possible once per day via a webserver in order to maintain the fairness of comparisons between systems. However, since we focus on comparing the algorithms in terms of stability and learning speed, we report results on the development set.

Initially we compare Searn and DAgger with learning rate equal to one. Figure 4.1(a) shows that the performance of DAgger peaks after 6 iterations beyond which it remains stable. On the other hand, the performance of Searn oscillates between high recall/low precision and low recall/high precision iterations (Figures 4.1(b) and 4.1(c)). In particular, in the first iteration Searn learns theme and cause assignment components given correctly identified triggers only (the expert policy only returns those), but the trigger recognition component learned returns many incorrect ones, thus resulting in high recall and low precision. This behaviour is reversed in the second iteration, in which the theme and cause assignment components are learned so that they can accommodate for incorrectly recognized triggers, but at the same time the trigger recognition component becomes extremely conservative. This pattern holds for subsequent iterations, albeit progressively less pronounced. In contrast, DAgger can combine training signal from both iterations, thus its performance improves faster and avoids such oscillating behaviour. The unstable behaviour of Searn affects the training time as well, since in the high recall/low precision iterations the algorithm needs to consider many more actions during the cost-sensitive example generation steps (lines 8-17 in Alg. 1).

Figure 4.1(b) shows a substantial drop in recall for both algorithms in the second iteration. This is due to the learned policy in the first iteration being unable to replicate the correct output without using the expert policy. This issue did not emerge in the experiments of Ross et al. [2011], but as explained in Section 3, event extraction is likely to be affected by it. Using slower learning rates ameliorates this problem (Figure 4.1(e)) and renders Searn

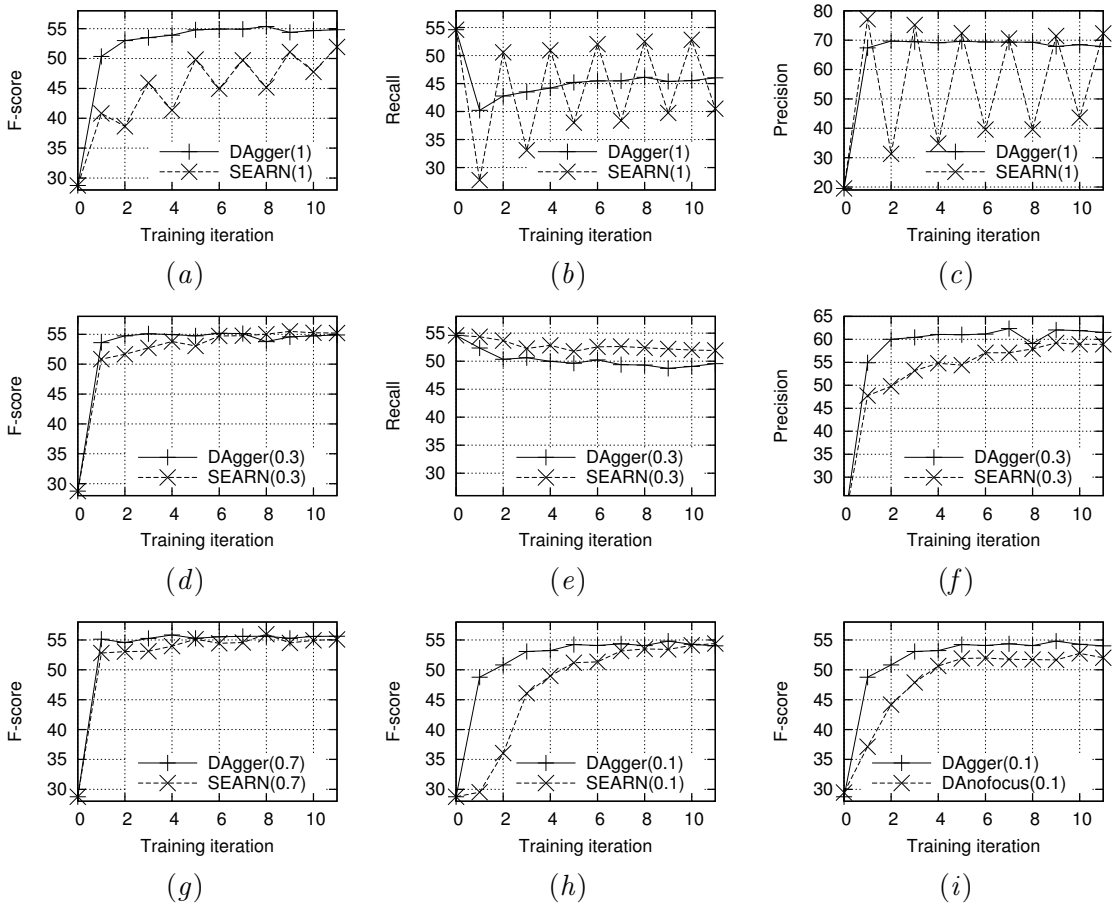


Figure 4.1: Development dataset results for DAGGER(β) and SEARN(β) with various learning rates.

more stable, but DAGGER still learns faster for a range of learning rates (0.7, 0.3 and 0.1 in Figures 4.1(d), 4.1(g) and 4.1(h) respectively).

Even though slower learning rates improve the performance for both SEARN and DAGGER, the improvement for the latter is not as dramatic (about 1.5 points in F-score). As discussed in Sec. 2, when $\beta < 1$ action costing becomes stochastic, which can result in unreliable estimates. In our experiments we used the *focused costing* approach proposed by Vlachos and Craven [2011], who reported that it improved the performance of SEARN by 4 F-score points. In Figure 4.1(i) we compared the performance of DAGGER with and without focused costing and we show that even though focused costing results in faster learning, the difference in terms of F-score is smaller, approximately 2 points. In other words, DAGGER is more robust w.r.t. the choice of action costing method. Also note that using DAGGER with $\beta = 1$ avoids introducing stochasticity in action costing, while unlike SEARN it remains stable, thus it is more likely to be applicable to other structured prediction tasks.

5. Conclusions - Future work

In this paper we compared two imitation learning algorithms for structured prediction, Searn and DAgger. We presented them in a unified description and evaluated them on biomedical event extraction. We found that DAgger is more stable and learns faster, while being more robust with respect to the choice of learning rates and action costing. These advantages are more pronounced in the parameter-free versions of the algorithms which avoid stochastic cost estimates and need simpler expert policy definitions. Finally, we assessed the effect of the learning rate in complex structured prediction tasks in which mistaken predictions can inhibit imitation learning algorithms from exploring useful parts of the training data. Our contributions should be relevant to applications of imitation learning to other structured prediction tasks.

In future work, we will apply imitation learning algorithms to other complex structured prediction tasks. Furthermore, we would like to explore and compare against other structured prediction frameworks such as structured prediction cascades [Weiss and Taskar, 2010] and output space search [Doppa et al., 2012] that also rely on reductions of structured prediction learning to simpler problems.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 270019 (SpaceBook project www.spacebook-project.eu).

References

Pieter Abbeel. *Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control*. PhD thesis, Department of Computer Science, Stanford University, 2008.

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning*, pages 46–53, 2004.

Adam Coates, Pieter Abbeel, and Andrew Y. Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th International Conference on Machine learning*, pages 144–151, 2008.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine Learning*, 75:297–325, 2009.

Pedro Domingos. Metacost: a general method for making classifiers cost-sensitive. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, pages 155–164. Association for Computing Machinery, 1999.

Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool Publishers, 1st edition, 2009. ISBN 1598296922, 9781598296921.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. Output space search for structured prediction. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Jin-Dong Kim, Yue Wang, Toshihisa Takagi, and Akinori Yonezawa. Overview of the Genia Event task in BioNLP Shared Task 2011. In *Proceedings of the BioNLP 2011 Workshop Companion Volume for Shared Task*, pages 7–15, 2011.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of 18th International Conference in Machine Learning*, pages 282–289. Morgan Kaufmann Publishers Inc., 2001.

David McClosky. *Any domain parsing: Automatic domain adaptation for natural language parsing*. PhD thesis, Department of Computer Science, Brown University, 2010.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *14th International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, June 1999.

Noah A. Smith. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, May 2011.

Pontus Stenetorp, Goran Topić, Sampo Pyysalo, Tomoko Ohta, Jin-Dong Kim, and Jun'ichi Tsujii. BioNLP Shared Task 2011: Supporting Resources. In *Proceedings of the BioNLP 2011 Workshop Companion Volume for Shared Task*, 2011.

Richard Sutton and Andrew Barto. *Reinforcement learning: An introduction*. MIT press, 1996.

Umar Syed. *Reinforcement learning without rewards*. PhD thesis, Department of Computer Science, Princeton University, 2010.

Umar Syed and Robert E. Schapire. Imitation learning with a value-based prior. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pages 384–391, 2007.

Andreas Vlachos and Mark Craven. Search-based structured prediction applied to biomedical event extraction. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 49–57. Association for Computational Linguistics, 2011.

Andreas Vlachos and Mark Craven. Biomedical event extraction from abstracts and full papers using search-based structured prediction. *BMC Bioinformatics*, 13(suppl. 11):S5, 2012.

David Weiss and Ben Taskar. Structured prediction cascades. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.