Case-Based Reasoning in Transfer Learning

David W. Aha[1], Matthew Molineaux[2], and Gita Sukthankar[3]

[1]Navy Center for Applied Research in Artificial Intelligence;
Naval Research Laboratory (Code 5514); Washington, DC 20375
[2]Knexus Research Corporation; Springfield, VA 22153
[3]School of Electrical Engineering and Computer Science,
University of Central Florida, Orlando, FL 32816
david.aha@nrl.navy.mil | matthew.molineaux@knexusresearch.com | gitars@eecs.ucf.edu

Abstract. Positive transfer learning (TL) occurs when, after gaining experience from learning how to solve a (source) task, the same learner can exploit this experience to improve performance and/or learning on a different (target) task. TL methods are typically complex, and case-based reasoning can support them in multiple ways. We introduce a method for recognizing intent in a source task, and then applying that knowledge to improve the performance of a case-based reinforcement learner in a target task. We report on its ability to significantly outperform baseline approaches for a control task in a simulated game of American football. We also compare our approach to an alternative approach where source and target task learning occur concurrently, and discuss the tradeoffs between them.

1 Introduction

There is a huge body of research on the study of transfer in psychology and education (e.g., Thorndike & Woodworth, 1901; Perkins & Salomon, 1994; Bransford *et al*., 2000), among other disciplines. Transfer is, more specifically, a focus of some research related to case-based reasoning (CBR), namely psychologically plausible theories of analogical transfer (e.g., Gentner, 1993; Hinrichs & Forbus, 2007). Arguably, case-based reasoning *is* the study of computational models for knowledge transfer, which plays a central role in the crucial topic of lifelong learning. Given this, there are surprisingly few publications on this topic in the CBR literature today.

Recently, this changed due to an increased emphasis on the study of Transfer Learning (TL), motivated in part by a DARPA research program of the same name.[1] TL concerns the study of how learning to solve tasks in a *source* domain can impact an agent's ability to learn to solve tasks from a *target* domain, where the focus is on *positive* transfer (i.e., involving improvements on measures of task performance). For example, we would expect that knowledge obtained from learning how to play one real-time strategy game (e.g., AGE OF EMPIRES) may assist us in learning to play a related, but different, game in the same genre (e.g., EMPIRE EARTH).

[1] http://www.darpa.mil/ipto/programs/tl/docs/TL_Brief.ppt

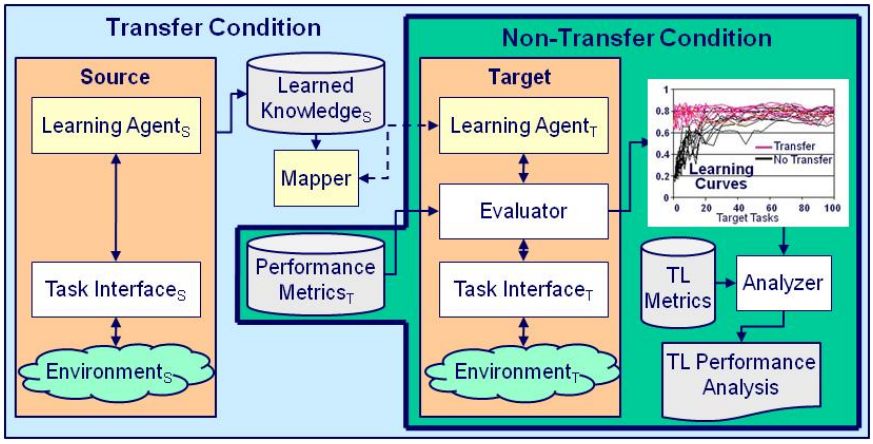

Figure 1: Transfer learning conditions, method, and evaluation strategy.

In this paper, we provide an overview of Transfer Learning and CBR techniques for achieving TL. We also introduce a method that uses *intent recognition* to create an abstraction of an actor's intentions to facilitate the TL process. This could enable the learner to reformulate problem instances involving the same actor. We conceptualize the actor's behavior as being governed by a set of hidden state variables that can be discovered through intent recognition and utilized by the transfer learner.

In the following sections, we discuss transfer learning, how CBR has served and can serve in some TL methods, introduce and evaluate a novel method that uses intent recognition to facilitate transfer learning for a case-based reinforcement learner performing a target task, and examine the tradeoffs of TL approaches versus a concurrent learning method (i.e., that gains experience from the source *and* target tasks, simultaneously) (Molineaux *et al*., 2009). Our study involves tasks drawn from an American Football simulator (RUSH 2008). We will conclude with future research suggestions at the intersection of CBR, TL, and plan recognition.

2 Transfer Learning and Case-Based Reasoning

Research on machine learning has traditionally focused on tasks in which examples are repeatedly and independently drawn from an identical distribution (i.i.d.) (Simon, 1983). This simplifying assumption is the basis of most ML research to date, as well as most research on case-based learning. In particular, it underlies the mathematical foundations of many existing supervised learning algorithms, and permits the study of their ability to *generalize* from training data.

This contrasts with reality; people frequently leverage their experience gained from one task to improve their performance on different, novel tasks. That is, they perform transfer learning:

> **Definition**: *Transfer learning* (TL) is the practice of recognizing and applying knowledge and skills learned from one or more previous tasks to more efficiently and/or effectively learn to solve novel tasks (in new domains).

Table 1: Summary of some related work on CBR and transfer learning.

Reference	CBR Use	TL Levels	TL Tasks		Environment Type
			Source	Target	
(von Hessling & Goel, ICCBR-05 WS)	Maps <world, decision tree policy> cases/models for Q fn. xfer in RL	2,5	Navigation in Civ. Turn-based Game	Same; Vary map	Turn-Based Strategy: Microworld
(Liu & Stone, AAAI-06)	Map QDBN task models: Modified SME for value fn. xfer in RL	4	KeepAway	Same; Vary # of units	Team Sports: RoboCup Soccer
(Sharma et al., IJCAI-07)	Map case models via Euclidean state sim. for Q fn. xfer for CBR/RL	1,4	Military control	Same; Vary # units or locations	Real-Time Strategy: MadRTS
(Klenk & Forbus, AAAI-07)	Map solutions via knowledge-intensive structure-mapping for CBR	1-6	Mechanics Problem Solving	Several variations!	Question answering: AP Physics
(Kuhlmann & Stone, ECML-07)	Map game description task models via graph isomorphism for RL	3-5	Board games (e.g., mini-chess)	Same; Vary board size, # pieces, etc	General Game Playing: GGP
This Paper	Map feature learned via intent recognition to learn Q fn. for CBR/RL	9	Identify Play of Defense	Control QB in Football Play	Team Sports: Rush 2008

Thus, the i.i.d. assumption does not (normally) hold in TL tasks. Yet general methods for TL hold the promise for being exceedingly useful; they could dramatically decrease the amount of training/cases required to achieve a given level of problem-solving competence in one domain by successfully employing knowledge obtained from solving problems for different, but related, tasks. This gap has motivated the development of computational models for TL, which has been the focus of recent workshops at NIPS-05, ICML-06, and AAAI-08[2], in addition to a large number of publications (e.g., Marx *et al*., 2005; Raina *et al*., 2006; Shapiro *et al*., 2008).

To clarify, Figure 1 summarizes a TL method and its evaluation process. The learning agents, tasks, performance metrics, and environments may all differ between the source and target domains. The key object handed off from source to target is the learned knowledge from the source domain. An evaluation of the process typically compares two conditions: the *transfer* condition (depicted by the entirety of Figure 1), in which case a *Mapper* is used to transform knowledge acquired from training on the source task to solve target tasks, and the *non-transfer* condition (depicted as a subsection in Figure 1), in which no such source task knowledge is leveraged. (The dotted line emanating from the Mapper indicates that it is referenced only by the transfer condition.) For these two conditions, two sets of learning curves are generated according to the target task's performance metric. These, along with a TL metric, are given to an analysis component for significance testing. We discuss examples of these for our study in Section 3.

TL problems vary widely, and a large variety of learning agents have been examined. For example, in some agents, only trivial mapping takes place, while in others mapping is a comprehensive, focal task. In most cases, the mapping is an abstraction designed to assist with relating source and target tasks.

While transfer has been long-studied in CBR (e.g., Goel & Bhatta, 2004), we review only a subset of recent relevant research to provide the context for our contribution. In this context, Table 1 summarizes and distinguishes five branches of prior research. In their TL processes, CBR is used for mapping, for learning in the target domain, or both. For example, Kuhlmann and Stone (2007) enumerate a constrained space of related game descriptions and store graph representations of them as cases. They then use graph isomorphism to map stored cases so as to reuse reinforcement learning (RL) value functions learned from solving source tasks (i.e., related game variants). Liu and Stone (2006) instead use a domain-tailored variant of the structure-mapping engine (SME) (Falkenhainer *et al*., 1989) to perform source-

[2] http://teamcore.usc.edu/taylorm/AAAI08TL/index.htm

target task mapping. Their cases encode agent actions in KeepAway soccer games as qualitative dynamic Bayesian networks. The focus is again on value function reuse. Von Hessling and Goel (2005) propose to learn abstractions of Q-learned policies (i.e., as decision trees) indexed by the environment's features. The policies are to be reused in similar environments. Sharma et al. (2007) focus on learning and reusing Q values for state-action pairs, where the feature vector states are case indices representing situations in a real-time strategy game. Their CBR/RL algorithm updates the Q values and eligibility traces of stored cases, where actions denote tactical decisions at the middle level of a three-level action hierarchy.

In stark contrast, Klenk and Forbus (2007) do not rely on RL. Instead, they employ SME in a context where cases are solutions to mechanics problems represented in predicate calculus. Given a new problem, they use the candidate inferences generated by SME to construct the necessary model (e.g., applicable equations and assumptions) to arrive at a solution.

Unlike these prior studies, we use an intent recognition approach in the source task to learn a crucial feature (i.e., the defensive team's play) for use in the target task (i.e., controlling the offensive team's Quarterback). The source and target tasks in previous work used similar control problems for both tasks, differing, for example, only in the number of agents on the playing field. One way to distinguish transfer learning tasks was suggested in the DARPA TL program, which proposed 11 categories (*levels*[1]) of TL tasks. While intended to be only an *initial* categorization scheme that ranges from simple *Memorizing* (i.e., generalization over i.i.d. examples) to complex *Differing* (i.e., distant-transfer) tasks, it was not subsequently refined by TL researchers. Still, it is useful for our needs: the prior work described above focused on tasks in which, for example, the set of components/units were modified, or the map/formulation differs among the source and target tasks. We instead focused on TL level 9, *Reformulating*, in which a representation transformation is needed to relate the two tasks. This is not to say that our task is more challenging, but rather that it involves different issues.

In addition, our task environment differs; it is a multiagent team sports simulator for American football, in which agents differ in their capabilities and pre-determined responsibilities, and the target task concerns controlling a specific agent.

We discuss related work on intent recognition for TL in Section 5. Due to page limitations, please see (Molineaux *et al*., 2009) for a discussion of related work on case-based reinforcement learning, which is not the focus of our current investigation.

3 Case Study: Intent Recognition for Transfer Learning

In this section, we introduce and evaluate a TL agent that performs intent recognition in a source task to improve the performance of a case-based reinforcement learner on a distinct target task. We describe the tasks, the learning algorithms, and their analyses. Our empirical study instantiates the methodology summarized in Figure 1.

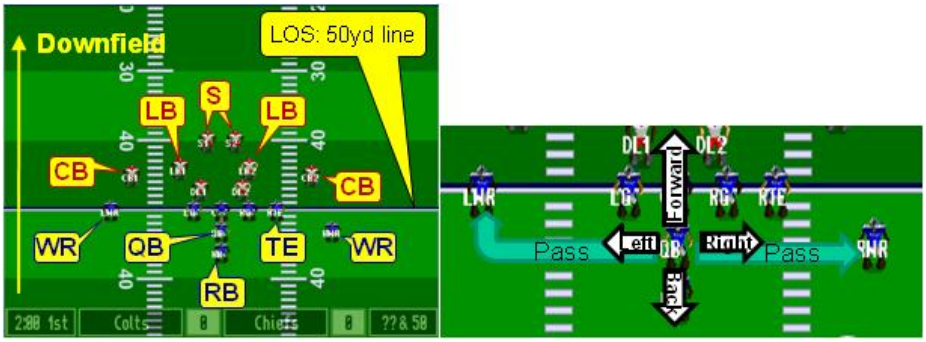

Figure 2: Left: The RUSH 2008 starting formation we used for the offense (**blue**) and defense (**red**), with some player/position annotations. Right: Six of the QB's eight possible actions. Not shown are **Throw RB** and **Noop.**

3.1 Environment, tasks, and state representations

Our source and target tasks both use the RUSH 2008[3] environment, which we adapted from the open-source RUSH 2005[4] simulator to support intelligent agent studies. RUSH simulates a simplified American football game on a small field (100x63 yards) where both the offensive and defensive teams have only eight players (see Figure 2). The source task concerns supervised intent recognition: learn to predict the play of the defensive team after observing the first few moments of the play. The defense always starts in the same formation and randomly selects one of 8 possible strategies appropriate for his formation. The defensive players act as follows:

Defensive lineman (DL) (2 players): These line up across the line of scrimmage (LOS) from the OL (see acronyms below) and try to tackle the ball handler.

Linebacker (LB) (2): Starting behind the DL, they will blitz the QB or guard a zone of the field or an *eligible* receiver (i.e., the RB, WR1, or WR2).

Cornerback (CB) (2): These line up across the LOS from the WRs and guard a player or a zone on the field.

Safety (S) (2): These begin 10 yards behind the LOS and provide pass coverage or chase offense players.

The target task, in which the identity of the defensive play is *not* revealed (i.e., is a hidden state variable), is to learn to control the quarterback's actions on repeated executions of a pass play, where the offensive players perform the following actions:

Quarterback (QB): Given the ball at the start of each play while standing 3 yards behind the center of the LOS, our QB agent decides whether and when to run (in one of four possible directions), stand, or throw (and to which receiver).

Running Back (RB): Starts 3 yards behind the QB, runs to an initial position 7 yards left and 4 yards downfield, then charges straight toward the goal line.

Wide Receiver #1 (WR1): Starts 16 yards to the left of the QB on the LOS, runs 5 yards downfield and turns right.

---

[3] http://www.knexusresearch.com/projects/rush/

[4] http://rush2005.sourceforge.net/

Wide Receiver #2 (WR2): Starts 16 yards to the right of the QB a few yards behind the LOS, runs 5 yards downfield, and waits.

Tight End (TE): Starts 8 yards to the right of the QB on the LOS and pass-blocks.

Offensive Linemen (OL): These 3 players begin on the LOS in front of the QB and pass-block (for the QB).

All players are given specific behavioral instructions (i.e., for the offense, a series of actions to execute) except for the QB, whose actions are controlled by our learning agent. The simulator is *stochastic*; each instructed player's actions are random within certain bounds (e.g., the RB will always go to the same initial position, but his path may vary). Each player has varying ability (defined on a 10-point scale) in the categories *power*, *speed*, and *skill*; these affect the ability to handle the ball, block, run, and tackle other players. The probability that a passed ball is caught is a function of the number and skills of defenders near the intended receiver (if any), the skills of the receiver, and the distance the ball is thrown.

RUSH uses a simplified physics; players and the ball each maintain a constant velocity while moving, except that the ball will accelerate downwards due to gravity. Objects are represented as rectangles that interact when they overlap (resulting in a catch, block, or tackle).

For the source task, we found that plays from a given starting formation are usually distinguishable after 3 time steps (Sukthankar *et al*., 2008). Given this, we use a feature vector representation of length 17 for source task examples. This includes 16 features for the 8 defensive players' displacement in two-dimensional space over the first 3 time steps, plus a label indicating the defensive play's name.

At the start of each play (for both tasks), the ball is placed at the center of the LOS along the 50 yard line. For the target task, the agent's *reward* is 1000 for a touchdown (i.e., a gain of at least 50 yards), -1000 for an interception or fumble, or is otherwise ten times the number of yards gained (e.g., 0 for an incomplete pass) when the play ends. A reward of 0 is received for all actions before the end of the play. Touchdowns (0.01%-0.2%), interceptions, and fumbles (combined: 1%-3%) rarely occur.

For the target task, our learning agent attempts to control the QB's actions so as to maximize total reward. The QB can perform one of eight actions (see Figure 2) at each time step. The first four (**Forward**, **Back**, **Left**, and **Right**) cause the QB to move in a certain direction for one time step. Three more cause the QB to pass to a receiver (who is running a pre-determined pass route): **Throw RB, Throw WR1,** and **Throw WR2**. Finally, **Noop** causes the QB to stand still. The QB may decide to run the football himself, and will choose actions until either he throws the ball, crosses into the end zone (i.e., scores a touchdown by gaining 50 yards from the LOS), or is tackled. If the QB passes, no more actions are taken, and the play finishes when an incompletion or interception occurs, or a successful receiver has been tackled or scores a touchdown.

The state representation in the target task contains only two features. The first denotes the defensive strategy (predicted after the third time step), while the second is the time step. For the *transfer* condition, the first feature's value will be predicted using the model transferred from learning on the source task. For the *non-transfer* condition, it will instead be randomly selected, chosen according to a uniform distribution over the defensive plays in use.

3.2 Learning algorithms

Many multi-class supervised learning algorithms can be used for the source task. We considered both support vector machines (SVMs) (Vapnik, 1998), due to their popularity and success, and the k-nearest neighbor classifier (using Euclidean distance), because it is a simple case-based algorithm. A soft-margin SVM for binary classification projects data points into a higher dimensional space, specified by a *kernel* function, and computes a maximum-margin hyperplane decision surface that most nearly separates two classes. Support vectors are those data points that lie closest to this decision surface; if these data points were removed from the training data, the decision surface would change. More formally, given a labeled training set $T = \{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\ldots,(\mathbf{x}_n,y_n)\}$ with feature vectors $\mathbf{x}_i \in \Re^n$ and class labels $y_i \in \{-1,1\}$, a soft margin SVM solves the following to find the maximal hyperplane that (most nearly) separates the two classes:

$$\min_{\mathbf{w},\mathbf{b},\xi} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i$$

constrained by:

$$\begin{aligned} y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + \mathbf{b}) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \end{aligned}$$

where $\mathbf{w}$ is the normal vector lying perpendicular to the hyperplane, $\mathbf{b}$ is a bias, ξ_i is a *slack* variable that measures the degree of misclassification of $\mathbf{x}_i$, C is a constant, and function $\phi(.)$ is represented by a kernel function $K(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i,\mathbf{x}_j)\phi(\mathbf{x}_i,\mathbf{x}_j)$. We use the radial basis function kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2}, \gamma > 0$$

To work on our 8-class problem, we employ a standard one-vs-one voting scheme where all 28 (i.e., 8*(8-1)/2) pair-wise binary classifiers are trained and the most popular class is selected. For our studies, we use the LIBSVM (Chang & Lin, 2001) implementation, and set C=1 and γ=0.008.

In the transfer condition, the model learned from the source task will be provided to the target task learner. The Mapper (Figure 1) will apply this model to the scenario data and output the predicted value of the first (of two) features.

Our recent focus has been on examining the behavior of novel case-based RL algorithms (Molineaux *et al*., 2008; 2009). Here we use a related algorithm, named *Case-Based Q-Lambda with Intent Recognition* (CBQL-IR), for the target learning task. Based on the Q(λ) algorithm (Sutton & Barto, 1998), it uses a set of case bases to approximate the standard RL Q function and the trained supervised algorithm from the source task to add opponent intent information to the state.

The Q function approximation maps state-action pairs to an estimate of the long-term reward for taking an action a in a state s. There is one Q_a *case base* in this set for each action $a \in \boldsymbol{A}$, where $\boldsymbol{A}$ is the set of 8 actions defined by the environment. Cases in Q_a are of the form $<s, v>$, where s is a feature vector describing the state and v is a real-valued estimate of the reward obtained by taking action a in state s, then pursuing the current policy until the task terminates. These case bases support a case-based problem solving process (López de Mantaras *et al.*, 2005). At the start of each experiment, each Q_a case base is initialized to the empty set. Cases are then added and

modified as new experiences are gathered, which provide new local estimates of the Q function.

At each time step, a state is observed by the agent, and an action is selected. With probability ε, a random action will be chosen (*exploration*). With probability 1-ε, CBQL-IR will predict the (estimated) best action to take (*exploitation*). To exploit, it *reuses* each Q_a case base by performing a locally-weighted regression using a Gaussian kernel on the *retrieved k* nearest neighbors of the current observed state *s*. (For faster retrieval, we use *kd*-trees to index cases.) Similarity is computed using a normalized Euclidean distance function. This produces an estimate of the value of taking action *a* in the current observed state *s*. CBQL-IR selects the action with the highest estimate, or a random action if any case base has fewer than *k* neighbors.

Once that action is executed, a reward *r* and a successor state *s'* are obtained from RUSH 2008. This reward is used to improve the estimate of the Q function. If the case is sufficiently novel (greater than a distance τ from its nearest neighbor) a new case is *retained* in Q_a with state *s* and $v = r + \gamma \max_{a \in A} Q_{a'}(s')$, where $Q_{a'}()$ denotes the current estimate for a state in Q_a and $0 \leq \gamma < 1$ is the discount factor. If the case is not sufficiently novel, the *k* nearest neighbors are *revised* according to the current learning rate α and their contribution β to the estimate of the state's value (determined by a normalization over the Gaussian kernel function, summing to 1). The solution value (i.e., reward estimate) of each case is updated using:

$$v = v + \alpha\beta[r + \gamma \max_{a \in A} Q_{a'}(s') - Q_a(s)].$$

Finally, the solution values of all cases updated earlier in the current trial are updated according to their λ-eligibility:

$$v = v + (\lambda\gamma)^t \alpha\beta[r + \gamma \max_{a \in A} Q_{a'}(s') - Q_a(s)],$$

where *t* is the number of steps between the earlier use and the current update, and $0 \leq \lambda < 1$ is the trace decay parameter.

3.3 Empirical evaluation

We hypothesized that *transferring an intent recognition model learned from the source task* (i.e., described in Section 3.1) *can significantly increase a learner's ability in the target task*. To investigate this, we applied the algorithms described in Section 3.2 for the transfer and non-transfer conditions. We also tested, as a baseline, a simple non-learning agent on the transfer task to provide additional insight on the difficulty of performing that task.

The methodology we use for testing is shown in Figure 1. We have described the learning agents, environments, the model learned from the source task, our source-target mapping, and the performance metrics. For our evaluation, we use the Lightweight Integration and Evaluation Toolkit (LIET) as the evaluator. That is, we use LIET, a variant of TIELT[5] (Molineaux & Aha, 2005), to integrate the learning agents with RUSH, and to conduct the evaluation.

For the source task, we varied the amount of training data (between 8 and 40 examples) given to the supervised learners; it wasn't clear, a priori, how much training would suffice to obtain good accuracy for predicting the defensive team's strategy. For the target task, we trained each condition for 100K trials. After every 250 training trials, the performance of the CBQL-IR agent was tested 10 times against

[5] http://www.tielt.org/

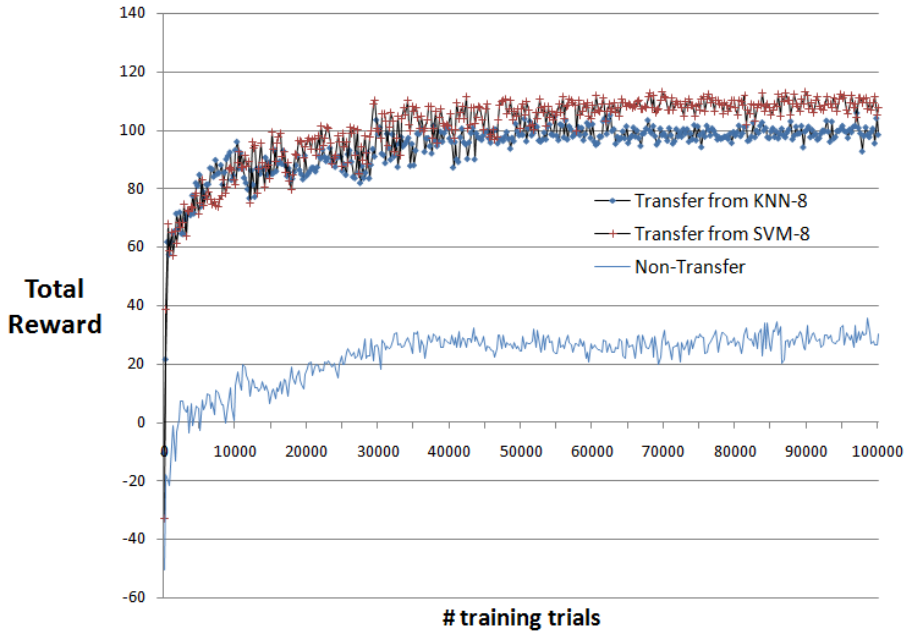

Figure 3: Target task performance on the Quarterback control task for the transfer condition (using either SVM or kNN to learn from the source task), and non-transfer condition's (Random's) play predictions.

each of the defensive strategies. The total reward from all 50 testing trials was averaged to obtain an estimate of average performance at each time step. This evaluation was conducted 20 times to ensure repeatability and obtain statistically significant measures of overall transfer performance.

We use three standard TL metrics to measure transfer performance. The first, *jump start*, measures the difference in performance between the transfer and non-transfer conditions before training begins on the target task. Next, *asymptotic gain* is the same measure, but applied to the end of the curves rather than the beginning. Finally, *k-step regret* measures the increase in performance over the entire learning period, defined as the integral difference between two learning curves, divided by the area of the bounding box that extends from the origin horizontally through the last trial of the curve (or the first trial at which they have both reached asymptotic performance) and vertically to the highest accuracy achieved by either averaged curve. Intuitively, this is a percentage of the maximum performance gain possible between an algorithm that always achieves worst-case performance and an algorithm that always achieves best-case performance.

For the analyzer, we used USC/ISI's implementation of randomized bootstrap sampling[6], which bins the two sets of curves and repeatedly draws a pseudosample (with replacement) for both sets. Applying a TL metric to each pseudosample yields a distribution, which is used to assess whether the two original sets differ significantly.

[6] http://eksl.isi.edu/cgi-bin/page.cgi?page=project-tl-evaluation.html

3.4 Results and analysis

Both supervised learners on the source task generalized well (i.e., recorded test accuracies above 95%) after training on only 1 example from each of the 8 defensive plays. Therefore, we used the trained SVM or k-nearest neighbor classifier with only 8 training examples to provide the transferred model for the transfer condition.

Figure 3 shows the performance on the target task, averaged over the 20 evaluations, for each of three conditions: transfer using a trained support vector machine, transfer using a k-nearest neighbor classifier, and a non-transfer version which predicts any one of the defensive strategies used in the target task with uniform probability. Transfer from 2 and 5 examples of each play yielded performance similar to the kNN and SVM curves depicted, so their performance is not shown. Also not shown is the average performance (over 10K trials) of the non-learning agent on the transfer task because its performance was so low (i.e., -61.1986).

As expected, both transfer variants significantly outperform the non-transfer variant of the agent in terms of both asymptotic advantage and k-step regret. Thus, we accept our hypothesis as stated in Section 3.3. The regret of the kNN variant over the non-transfer agent is 78.57 (measured over the first 50,000 trials), while the regret of the SVM variant over the non-transfer agent is 78.87 (measured over the first 70,000 trials). The asymptotic advantage of kNN over the non-transfer agent is 59.5; for SVM, it's 66.66. All of these measures are highly statistically significant with $p<0.0001$. For jump start, there is no statistical advantage between the three curves. This is because CBQL-IR acts randomly when there are no stored cases, as is true before any training occurs; so all three pursue the same policy before training. The differences between SVM and kNN are not statistically significant using either the k-step-regret or asymptotic advantage measures.

4 A Concurrent Learning Alternative

Transfer learning is intuitively appealing, cognitively inspired, and has led to a burst of research activity, much of which concerns new techniques involving hierarchical RL. However, transfer is not always cost-effective, and can sometimes result in decreased performance (i.e., *negative* transfer). Also, determining *what* to transfer is a research question itself (e.g., Rosenstein *et al.*, 2005; Stracuzzi, 2006). Yet seemingly overlooked in the literature is a more fundamental question: Should learning be performed separately on the source and target tasks, or can the knowledge learned on the source task be instead learned *during* the target task? What are the tradeoffs of doing this versus transfer learning? We found no investigations on this issue.

We briefly address this here, referring to the alternative as *concurrent learning*. In this approach, intent recognition takes place during (rather than prior to) learning to control the QB. As usual during the target task, the label of the defensive play is not given, and must be inferred. We model this as an online unsupervised learning task that clusters the observable movements of the defensive players into groups. The perceived movement $m \in \boldsymbol{M}$ for each defensive player is the direction that player is moving during a time step, which has nine possible values:

$\boldsymbol{M}$ = {*None, Forward, Left, Right, Back, Forward-Right, Forward-Left, Back-Right, Back-Left*}

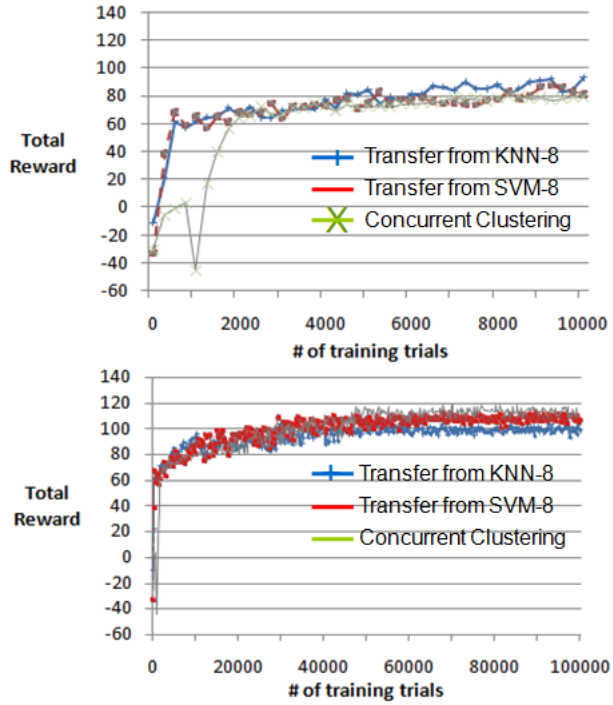

Figure 4: Target task performance for the SVM and kNN transfer conditions and the concurrent learning (clustering) algorithm (for 10K (top) and 100K (bottom) trials).

These directions are geocentric; *Forward* is always in the direction of play (downfield), and all other directions are equally spaced at 45° angles. Clustering is performed after the third time step of each play, at which time we observe the offsets of the players from their starting formation positions. Thus, 16 features are used to represent defensive plays (i.e., the two-dimensional offsets of each of 8 defensive players). For the first 1000 trials, examples were added to the batch to be clustered, but the predicted cluster (i.e., the recognized plan) was *not* used in action selection.

We used the Expectation-Maximization (EM) algorithm from the Weka[7] suite of machine learning software for clustering. EM iteratively chooses cluster centers and builds new clusters until the centers move only marginally between iterations. Also, it increases the number of clusters to discover until successive steps decrease the average log-likelihood of the correct clustering of all points. We selected EM after reviewing several other algorithms; the clusters it found correctly disambiguated the defensive plays over 99% of the time with less than 1000 examples.

Figure 4 compares the performance of the online clustering agent, which learns two tasks concurrently, with the SVM and kNN transfer agents introduced in Section 3. Performance differences are not statistically significant using any of our metrics over the full training period (100,000 trials). However, there is a significant benefit to performing transfer during early learning; transfer achieves good performance far more quickly. As shown, both TL algorithms learn quickly during the first 500 trials,

[7] http://www.cs.waikato.ac.nz/ml/weka/

reaching a total reward greater than 60 on average. The concurrent learning algorithm instead takes 2,000 trials to reach the same performance level.

Regret for the kNN TL agent versus the concurrent learning agent for the first 2,000 trials is 29.4, and for the SVM agent is 29.3. Both TL agents statistically outperform the online agent with $p<0.0001$ during this early stage of learning.

The concurrent learner's initial learning rate is low due to the time it must spend learning to recognize opponent behaviors, which the TL algorithms learned from the source task. Also, it discovered more clusters than the actual number of defensive plays used. Finally, early examples provided by the CBR/RL agent aren't helpful; they are either too short, or atypical, because that agent does not yet have a successful policy for controlling the QB, whose actions affect the play outcome and the actions of the defensive team. The benefit of TL is that the QB used in the target task is already an expert at "reading" the defense, and so good examples can be obtained starting with the first trial on the target task.

The benefits of TL versus concurrent learning should increase with task complexity because more complex tasks typically require more knowledge to learn, which is often easier to obtain via transfer from simpler tasks. However, there is a cost to TL, both in terms of engineering (in some cases, the source task may be in an entirely different domain) and in higher overall computational complexity. Practitioners interested in multi-task learning may benefit by instead using concurrent learning.

5 Discussion: Intent Recognition, TL, CBR, and Future work

In his seminal work on plan recognition, Kautz (1987) described his event hierarchy circumscription framework as a process for determining "which conclusions are absolutely justified on the basis of the observations, the recognizer's knowledge, and a number of explicit closed-world assumptions." In general, this union of observations, prior knowledge, and closed-world assumptions characterizes research efforts on *plan recognition*. Typically this prior knowledge is encapsulated into a plan library of generative models (either logical (Kautz, 1987) or probabilistic (Bui, 2002)) that the recognizer matches against streams of observations.

In *intent recognition*, the recognizer attempts to identify useful features relating to an actor's future actions (e.g., the next action to be performed) without an explicit representation of the plan generation model. Even without knowledge of the underlying generative model, being able to discriminate between different types of plans or actions sufficiently in advance can be extremely valuable (e.g., as shown in our study). For instance, in applications involving opponent or user modeling, having limited knowledge of the actor's intentions at an early stage can be more useful than having complete information about the entire plan later in the execution process.

Prior work on TL and plan recognition has focused on the problem of transferring recognition models between different actors. Liao et al. (2005) demonstrated a method for using data from other users to learn priors for a discriminative location-based activity recognition model; an alternate approach is to use prior knowledge to dictate the *structure* of the model rather than the parameters (Natarajan *et al*. 2007). A general issue is that plan generation models based on propositional representations (e.g., the basic HMM) do not generalize well across different users, so there has been work on inference methods for more expressive relational and hierarchical HMM

variants that have superior generalization properties (Natarajan *et al*. 2008; Blaylock & Allen, 2006). In our work, the source task (learning a discriminative model of the opponent's play) is quite dissimilar from the target task (learning an optimal single-agent play policy) and we do not address the problem of generalizing across actors.

While several researchers have addressed the topic of CBR and intent or plan recognition, none have proposed its use to facilitate transfer learning. For example, Fagan and Cunningham (2003) analyze a method for predicting a player's actions in a computer game, where the task was supervised learning rather than transfer learning. Kerkez and Cox (2003) represent plans as state-action sequences, index them using an abstract representation (i.e., the number of generalized predicates instantiated in a state), and analyze a case-based algorithm for action prediction for multiple domains. However, no transfer was performed.

Case-based methods that leverage knowledge of intentions and/or plans have significant potential for TL. We demonstrated a simple approach of this type where the beneficiary was a case-based reinforcement learner. Our future work includes investigating our techniques on more comprehensive tasks (e.g., learning to control all offensive players to win an entire football game), including those requiring transfer between different problem domains. More generally, CBR can also be used to map the learned knowledge, and in other roles. An excellent line of future research concerns the study of how case-based algorithms can support continuous learning and planning processes in environments with intentional agents where performance depends on the ability to master multiple tasks, and where reuse can significantly reduce the time required before competent performance emerges.

6 Summary

We introduced and evaluated a transfer learning strategy that uses intent recognition to assist a case-based reinforcement learner. It significantly improved task performance for an application involving the control of an agent in a multi-agent team sports environment. Our work is novel in its use of intent recognition for this purpose.

We also briefly examined the typically ignored issue of whether concurrent learning strategies should be considered as an alternative to transfer learning. We discussed some of their tradeoffs, but leave their formal analysis for future research.

Case-based reasoning can play significant roles in transfer learning, yet it has received only a limited amount of attention (e.g., Klenk & Forbus, 2007). This is surprising, given its potential as a focal process for mapping learned knowledge, and reducing overall learning time. In our future work, we will examine how case-based approaches can support lifelong learning in multi-task, multi-agent environments in which knowledge of intentions and (e.g., adversarial) plans (e.g., Sukthankar *et al*., 2008) can be leveraged to improve performance for decision support applications.

Acknowledgements

This research was supported by DARPA's Information Processing Techniques Office and the Naval Research Laboratory. Thanks also to Matthew Klenk for his suggestions on this paper.

References

Blaylock, N., & Allen, J. (2006). Fast hierarchical goal schema recognition. *Proceedings of the Twenty-First National Conference on Artificial Intelligence* (pp. 796-801). Boston, MA: AAAI Press.

Bui, H. (2002). A general model for online probabilistic plan recognition. *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence* (pp. 1309-1318). Acapulco, Mexico: Morgan Kaufmann.

Bransford, J.D., Brown, A.L., & Cocking, R.R. (Eds.) (2000). *How people learn: Brain, mind, experience, and school*. Washington, DC: National Academy Press.

Chang, C.-C., & Lin, C.-J. (2001). LIBSVM: A library for support vector machines. [http://www.csie.ntu.edu.tw/~cjlin/libsvm]

Fagan, M., & Cunningham, P. (2003). Case-based plan recognition in computer games. *Proceedings of the Fifth International Conference on Case-Based Reasoning* (pp. 161-170). Trondheim, Norway: Springer.

Falkenhainer, B., Forbus, K.D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, **41**(1), 1-63.

Gentner, D. (1993). The mechanisms of analogical learning. In B.G. Buchanan & D.C. Wilkins (Eds.) *Readings in knowledge acquisition and learning: Automating the construction and improvement of expert systems*. San Francisco, CA: Morgan Kaufmann.

Goel, A., & Bhatta, S. (2004). Design patterns: A unit of analogical transfer in creative design. *Advanced Engineering Informatics*, **18**(2), 85-94.

von Hessling, A., & Goel, A. (2005). Abstracting reusable cases from reinforcement learning. In D.W. Aha & D.C. Wilson (Eds.) (2005). *Computer gaming and simulation environments: Proceedings of the ICCBR Workshop.* In S. Bruninghaus (Ed.) *Workshop Proceedings of the Sixth ICCBR*. Chicago, IL: DePaul University.

Hinrichs, T., & Forbus, K. (2007). Analogical learning in a turn-based strategy game. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence* (pp. 853-858). Hyderabad, India: Professional Book Center.

Kautz, H. (1987). *A formal theory of plan recognition*. Doctoral dissertation: University of Rochester, Rochester, NY.

Kerkez, B., & Cox, M.T. (2003). Incremental case-based plan recognition with local predictions. *International Journal on Artificial Intelligence Tools*, **12**(4), 413-463.

Klenk, M., & Forbus, K.D. (2007). Measuring the level of transfer learning by an AP physics problem-solver. *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence* (pp. 446-451). Vancouver (BC), Canada: AAAI Press.

Kuhlmann, G., & Stone, P. (2007). Graph-based domain mapping for transfer learning in general games. *Proceedings of the Eighteenth European Conference on Machine Learning* (pp. 188-200). Warsaw, Poland: Springer.

Liao, L., Fox, D., & Kautz, H.A. (2005). Location-based activity recognition using relational Markov networks. *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence* (pp. 773-778). Edinburgh, Scotland: Professional Book Center.

Liu, Y., & Stone, P. (2006). Value-function-based transfer for reinforcement learning using structure mapping. *Proceedings of the Twenty-First National Conference on Artificial Intelligence* (pp. 415-420). Boston, MA: AAAI Press.

López de Mantaras, R, McSherry, D., Bridge, D.G., Leake, D.B., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K.D., Keane, M., Aamodt, A., & Watson, I.D. (2005). Retrieval, reuse, revision and retention in case-based reasoning. *Knowledge Engineering Review*, **20**(3), 215-240.

Marx, Z., Rosenstein, M.T., Kaelbling, L.P., & Dietterich, T.G. (2005). Transfer learning with an ensemble of background tasks. In D. Silver, G. Bakir, K. Bennett, R. Caruana, M. Pontil, S. Russell, & P. Tadepalli (Eds.) *Inductive Transfer: 10 Years Later: Papers from the NIPS Workshop*. Whistler (BC), Canada. [http://iitrl.acadiau.ca/itws05/]

Molineaux, M., & Aha, D.W. (2005). TIELT: A testbed for gaming environments. *Proceedings of the Twentieth National Conference on Artificial Intelligence* (pp. 1690-1691). Pittsburgh, PA: AAAI Press.

Molineaux, M., Aha, D.W., & Moore, P. (2008). Learning continuous action models in a real-time strategy environment. *Proceedings of the Twenty-First International FLAIRS Conference* (pp. 257-262). Coconut Grove, FL: AAAI Press.

Molineaux, M., Aha, D.W., & Sukthankar, G. (2009). Beating the defense: Using plan recognition to inform learning agents. To appear in *Proceedings of the Twenty-Second International FLAIRS Conference*. Sanibel Island, FL: AAAI Press.

Natarajan, S., Bui, H.H., Tadepalli, P., Kersting, K., & Wong, W.-K. (2008). Logical hierarchical hidden Markov models for modeling user activities. *Proceedings of the Eighteenth International Conference on Inductive Logic Programming* (pp.192-209). Prague, Czech Republic: Springer.

Natarajan, S., Tadepalli, P., & Fern, A. (2007). A relational hierarchical model for decision-theoretic assistance. *Proceedings of the Seventeenth International Conference on Inductive Logic Programming* (pp. 175-190). Corvallis, OR: Springer.

Perkins, D. N., & Salomon, G. (1994). Transfer of learning. In T. Husen & T. N. Postelwhite (Eds.). *International Handbook of Educational Research* (pp. 6452-6457). Oxford, UK: Pergamon Press.

Raina, R., Ng, A.Y., & Koller, D. (2006). Constructing informative priors using transfer learning. *Proceedings of the Twenty-Third International Conference on Machine Learning* (pp. 713-720). Pittsburgh, PA: ACM.

Rosenstein, M.T., Marx, Z., Kaelbling, L.P., & Dietterich, T.G. (2005). To transfer or not to transfer. In D. Silver, G. Bakir, K. Bennett, R. Caruana, M. Pontil, S. Russell, & P. Tadepalli (Eds.) *Inductive Transfer: 10 Years Later: Papers from the NIPS Workshop*. Whistler (BC), Canada. [http://iitrl.acadiau.ca/itws05/Papers/ITWS10-RosensteinM05_ITWS.pdf]

Shapiro, D., Könik, T., & O'Rorke, P. (2008). Achieving far transfer in an integrated cognitive architecture. *Proceedings of the Twenty-Third Conference on Artificial Intelligence* (pp. 1325-1330). Chicago, IL: AAAI Press.

Sharma, M., Holmes, M., Santamaria, J.C., Irani, A., Isbell J., C.L., & Ram, A. (2007). Transfer learning in real-time strategy games using hybrid CBR/RL. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence* (pp. 1041-1046). Hyderabad, India. [http://www.aaai.org/Papers/IJCAI/2007/IJCAI07-168.pdf]

Simon, H.A. (1983). Search and reasoning in problem solving. *Artificial Intelligence*, **21**, 7-29.

Stracuzzi, D. (2006). Memory organization and knowledge transfer. In B. Banerjee, Y. Liu, & G.M. Youngblood (Eds.) *Structural Knowledge Transfer for Machine Learning: Papers from the ICML Workshop.* Pittsburgh, PA.[http://orca.st.usm.edu/~banerjee/icmlws06/]

Sukthankar, G., Molineaux, M., & Aha, D.W. (2008). Recognizing and exploiting opponent intent in Rush Football (Technical Note AIC-09-062). Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Thorndike, E.L., & Woodworth, R.S. (1901). The influence of improvement in one mental function upon the efficiency of other functions (I). *Psychological Review*, **8**, 247-261.

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley & Sons.