2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Group Induction

Alex Teichman and Sebastian Thrun
Stanford University

***Abstract*— Machine perception often requires a large amount of user-annotated data which is time-consuming, difficult, or expensive to collect. Perception systems should be easy to train by regular users, and this is currently far from the case.**

Our previous work, tracking-based semi-supervised learning [14], helped reduce the labeling burden by using tracking information to harvest new and useful training examples. However, [14] was designed for offline use; it assumed a fixed amount of unlabeled data and did not allow for corrections from users. In many practical robot perception scenarios we A) desire continuous learning over a long period of time, B) have a stream of unlabeled sensor data available rather than a fixed dataset, and C) are willing to periodically provide a small number of new training examples.

In light of this, we present *group induction*, a new mathematical framework that rigorously encodes the intuition of [14] in an alternating optimization problem similar to expectation maximization (EM), but with the assumption that the unlabeled data comes in groups of instances that share the same hidden label. The mathematics suggest several improvements to the original heuristic algorithm, and make clear how to handle user interaction and streams of unlabeled data. We evaluate group induction on a track classification task from natural street scenes, demonstrating its ability to learn continuously, adapt to user feedback, and accurately recognize objects of interest.

I. INTRODUCTION

It is not uncommon for perception systems to be trained on thousands or tens of thousands of labeled examples. At the more extreme end of the spectrum, previous supervised work on car, pedestrian, and bicyclist detection in autonomous driving was trained on 600 million labeled instances of objects [16]. Requiring thousands or more training examples makes scaling up these perception systems to larger numbers of object classes an onerous task, and puts the training of object recognition systems firmly out of reach of regular people.

For example, consider a farming robot designed for automated weeding. Each farm will have a different perception task due to differences in soil color, weed populations, crop types, crop age, and so on. Training a new perception system for each farm from scratch using fully-supervised techniques could easily be economically infeasible due to the costs of collecting large numbers of hand-labeled training examples. Typically, training examples for many different viewpoints, lighting conditions, and poses are necessary to produce a high accuracy perception system. However, a few examples provided by a user should be enough for the robot to recognize at least one object of interest, then look at it from different angles to automatically harvest new and useful training examples. It is this intuition that [14] harnessed, and

Fig. 1. We evaluate group induction on an object recognition task using Stanford's autonomous vehicle, Junior.

which we will continue to exploit in this paper. The goal is accurate perception systems that can be trained by non-experts using only tens of training examples. Towards this end, the primary contributions of this work are:

- a more rigorous mathematical framework for the intuition behind tracking-based semi-supervised learning, generic to any scenario in which the unlabeled data has group structure.
- extensions of this math for making use of streams of unlabeled data and occasional user input.
- a particular implementation of the above which we empirically demonstrate is practical for lifelong learning of object recognition systems for autonomous driving.

II. RELATED WORK

Group induction makes use of both labeled and unlabeled data and therefore falls into the category of semisupervised methods, for which a broad overview can be found in [20]. The abstract math is perhaps most related to the EM algorithm [4]; both are alternating optimization problems with training and hidden variable estimation steps, but group induction additionally assumes the group structure in the unlabeled data and uses hard rather than soft assignment.

Co-training [1] is a related semisupervised method which uses different "views" of the same instances. A different classifier for each view is maintained and used for labeling the training set of the other view. An example from a robotics context would be having a camera view and a pointcloud view of every training example; then the system could, in theory, be trained with camera images of cars, learn to recognize them in laser range finder data, then harvest camera image training examples of car headlights at night. This is similar to group induction in that groups of (two) unlabeled instances are assumed to have the same label, but different in that the two unlabeled instances live in different descriptor spaces. In group induction, we assume that all instances live in the same descriptor space.

Active learning involves a machine learning system asking a user for labels to particular examples. A broad overview can be found in [11]. We make use of a very simple version of active learning in which, when a user decides to provide feedback to the system, examples are sorted by confidence; useful examples can easily be found in low-confidence regions of the space. In robotics, active learning is used, for example, in [5] to reduce the labeling burden for autonomous navigation.

Other ways to reduce the labeling burden have also been explored in robotics. For example, in [8], domain adaptation is applied to use 3D models to improve object recognition in laser scans. Self-supervised learning is also common in terrain classification, as terrain labels can be found automatically using depth sensors [18, 2] or accelerometers [19].

A preliminary version of the work in this paper was presented at an RSS 2012 workshop as a presentation only. Slides can be found at [15]. The goals of online and active learning were the same, but group induction had not yet been developed.

III. GROUP INDUCTION

In this section, we introduce the abstract mathematics of group induction. In Section V-B, we will consider a concrete instantiation of this math for the autonomous driving problem.

Intuitively, group induction is a semi-supervised framework for harvesting new and useful training examples from unlabeled data by exploiting the fact that groups of unlabeled examples are known to share the same (hidden) label. Initially, a classifier is trained using only a small amount of user-annotated data. This initial classifier is used to search the unlabeled dataset for groups it can classify confidently. These groups get added to the training set, the classifier is updated using the new examples, and the process repeats. New and useful training examples - *i.e.* those that the classifier would otherwise classify incorrectly - can be added during this procedure because of the grouping in the unlabeled data. This intuition was introduced in [14]; what follows is the new formalization of this intuition.

Formally, group induction is the optimization problem

$$\underset{H,\ y_g \in \{-1,0,+1\} \forall g}{\text{minimize}} \quad \sum_i \ell(H(x_i), y_i) + \sum_g \sum_u \ell(H(x_{g,u}), y_g), \tag{1}$$

where H is a classifier, $x \in \mathbb{R}^N$ is a descriptor, $y \in \{-1, 0, +1\}$ is a class label, and ℓ is a loss function. The two terms correspond to supervised training data and unlabeled groups, respectively, with i ranging over the supervised data, g ranging over all groups, and u ranging over instances within a group.

Different machine learning models can be plugged in to this problem. For example, with logistic regression, $H(x) = w^T x$ and $\ell(H(x), y) = \log(1 + \exp(-yH(x)))$. Alternatively, with boosting, $H(x) = \sum_k h_k(x)$ is the strong classifier composed of many weak classifiers and $\ell(H(x), y) = \exp(-yH(x))$. Much of the following math remains the same or similar for different choices of ℓ, but we will use boosting and exponential loss from this point on. Finally, we present only the binary classification case; the extension to multiclass is straightforward.

The group induction problem as a whole is non-convex, but we can find a local minimum by alternating between an *induction* phase and a *training* phase, analogous to the expectation and maximization steps of EM [4].

In the induction phase, we hold H fixed and solve for each hidden group label y_g. This problem is separable across groups, so we can consider in isolation the problem

$$\underset{y \in \{-1,0,+1\}}{\text{minimize}} \quad \sum_u \exp(-yH(x_u)), \tag{2}$$

where unnecessary indexing and constant terms have been stripped. Drawing inspiration from self-paced learning [7], the $y = 0$ label is allowed so that only unlabeled groups which we are confident in will actually be used for training. This problem has an analytical solution; we simply evaluate the objective function for each of the possible labels and choose the minimizer. This is significantly more conservative from the induction heuristic in [14], which proposed a threshold on average group confidence $\frac{1}{U}\sum_{u=1}^{U} H(x_u)$. See Figure 2 for a concrete example of the induction criteria.

The training phase entails holding the group labels fixed while training the classifier. In the generic case, this reduces to the standard training objective function of $\sum_i \ell(H(x_i), y_i)$, where i ranges over all supervised instances as well as all instances in the inducted groups.

A. Worst-case classifier noise

In [14], one could adjust the induction speed with a threshold on average group confidence. Here, because the induction criteria has changed, that method no longer applies. Moreover, we would like to maintain the rigorousness of our optimization; if we were to simply decree that the induction phase must respect a threshold on average group confidence, we would no longer be optimizing the problem (1).

Instead, we can propose an alternative optimization problem which encodes the intuition of [14] in a rigorous way. The idea is simple: We assume that all classifier predictions are corrupted by a fixed amount of worst-case noise. The full optimization problem becomes

$$\underset{H,\ y_g \in \{-1,0,+1\} \forall g}{\text{minimize}} \quad \underset{\epsilon_i, \epsilon_{g,u} \in \pm\epsilon_{\max}\ \forall i,g,u}{\text{maximize}} \quad \sum_i \exp(-y_i[H(x_i) + \epsilon_i]) + \sum_g \sum_u \exp(-y_g[H(x_{g,u}) + \epsilon_{g,u}]).$$

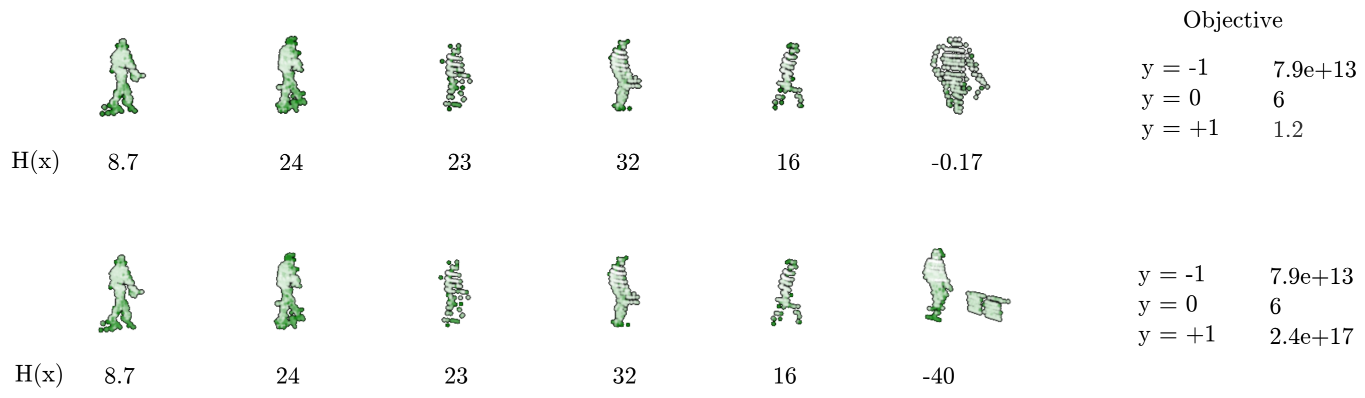

Fig. 2. Example of the induction phase for two hypothetical tracks, each of which defines a group of instances assumed to share the same label. Log odds predictions $H(x)$ for each frame were generated using a classifier trained with our method. The right column shows the objective function values corresponding to the optimization problem (2). In the top row, $y = +1$ minimizes the objective function so the track is inducted as a pedestrian. The result is that a useful training example - one for which the classifier predicts incorrectly - is added to the training set. In the bottom row, a transient under-segmentation looks very unlike a pedestrian to the classifier, preventing induction; $y = 0$ minimizes the objective function and so the group does not contribute during the training phase.

The optimization for the training phase reduces to

$$\begin{aligned}\underset{H}{\text{minimize}} \quad & \sum_i \exp(-y_i H(x_i) + \epsilon_{\text{max}}) \\ & + \sum_g \sum_u \exp(-y_g H(x_{g,u}) + \epsilon_{\text{max}}),\end{aligned}$$

and the worst-case errors drop out as a constant factor. We are left with boosting training as usual.

During the induction phase, analogous to (2), we have

$$\underset{y \in \{-1,0,+1\}}{\text{minimize}} \quad \underset{\epsilon_u \in \pm\epsilon_{\text{max}} \; \forall u}{\text{maximize}} \quad \sum_u \exp(-y[H(x_u) + \epsilon_u]). \quad (3)$$

As before, we can simply evaluate this sum for each possible y and choose the smallest. Setting $\epsilon_{\text{max}} > 0$ allows us to make the induction more conservative, analogous to increasing the average group confidence threshold in [14].

B. Active learning and retrospection

In practice, it is often desirable to have a user provide occasional input to guide the system to a good local minimum. This amounts to adding new examples to the supervised term of (1). These new examples could be acquired from any of a number of active learning methods.

When new annotations arrive, we start a new optimization problem that is similar to the old one, but with additional terms of supervised data. The naïve warm-start for this new optimization problem is to simply use the previous classifier and group labels untouched.

However, this can have undesirable effects when using a highly discriminative classifier. Consider a case in which several false positive inductions have been made and a user, watching the system make false positive classifications at runtime, provides new annotated examples to address the issue. Since the false positive inductions remain in the training set along with the new annotated examples, a highly discriminative classifier will learn to distinguish the two based on what seem to be minor and unnoticeable details.

What we actually want is to de-induct, *i.e.* set $y_g = 0$, just the false positive inductions. We refer to the process of using new annotated examples to de-induct certain groups as *retrospection*. Because we are choosing a warm start for a new optimization problem rather than actually taking a step during an optimization, we are free to apply heuristics that seem reasonable and effective.

One way to accomplish retrospection is to make use of the worst-case classifier noise formulation above: Do an induction step using $\epsilon_{\text{max}} = \max_i[-y_i H(x_i)]$, where i ranges over the new annotated examples. This provides a lower bound on the true value of the worst-case noise, and the result is that low confidence groups will be de-inducted and high confidence groups will remain. A user annotation which reveal a high confidence false positive will result in a large ϵ_{max} and therefore significant retrospection, whereas a low confidence false positive will result in only moderate retrospection. We also reset H, as otherwise at the next induction phase, it will tend to re-induct the false positive groups we were intending to be rid of in the first place.

C. Unlabeled data streams

When considering streams of unlabeled data, we need to amend the optimization problem (1) somewhat, as the sum over the unlabeled groups grows continuously. In practice, we cannot store all unlabeled groups, so we resort to the approximation of keeping a fixed-size set of groups that can fit in memory. Specifically *which* groups is motivated directly by the math: Choose those that contribute the most to the training phase objective function, after constant terms (*i.e.* groups with $y_g = 0$) are dropped. Intuitively, this corresponds to keeping the examples from which the classifier can learn the most.

We assume that all groups have a bounded number of instances and that unlabeled data arrives in *chunks* with a bounded number of groups. During the induction phase, one new chunk is added to the unsupervised term, induction is run as described above, and then the least useful groups are dropped until the unsupervised term is equal to its maximum size. In this way, the group induction system has an upper bound on its memory footprint and thus can be expected to run indefinitely even as new unlabeled data streams in.

IV. TRACK CLASSIFICATION

A. Problem overview

In this section, we briefly review the track classification task introduced in [16] and addressed in [12, 14]. Track classification is defined as a sub-task of an object recognition pipeline which includes segmentation, tracking, and track classification. Objects are first segmented and tracked without semantic labels, then tracks are classified as a whole to determine semantic label. In our case, raw data from a Velodyne HDL-64E is clustered using simple obstacle detection and connected components analysis, and then these clusters are tracked using Kalman filters [9, 10]. This segmentation and tracking stage has no knowledge of semantic label, but it is known that all frames in a track are likely to share the same label. Track classifications are determined by taking the average of the frame classifications.

We make use of the Stanford Track Collection (STC) for our evaluation. This is a large dataset designed for evaluating on the track classification problem, containing about one million labeled frames (*i.e.* single views of objects) across about fourteen thousand tracks collected from natural street scenes. Examples objects from the STC can be seen in Figure 3.

Segmentation and tracking errors are considered out of scope for the track classification task, as we need to avoid confounding these errors with classification errors. Thus, the test set does not include segmentation and tracking errors, though the unlabeled data we use for group induction does.

B. Parametric boosting classifier

Many choices of classifier and loss function would likely be effective for group induction on this task. For example, we expect that one could probably achieve similar results using the standard online learning approach of stochastic gradient descent training of a logistic regression model[1]. For completeness, however, we present our particular choice, based on boosting. A brief introduction to boosting using notation similar to that of this paper can be found in [14], and further details can be found in [6], among many others.

We refer to our method as *parametric boosting* because it has the form of a boosting classifier, but draws from a fixed pool of possible weak classifiers. This results in a fixed number of parameters that must be stored no matter how many new weak classifiers are added.

The specialized boosting classifier of [14] used hyperspheres as weak classifiers. Training continuously, as we will do with group induction, entails adding new weak classifiers continuously, and at runtime one must compute distances to all hyperspheres to make a prediction. This is a significant impediment to lifelong learning because classification time grows as training proceeds.

[1]with one caveat about descriptor space choice, discussed in Figure 4.

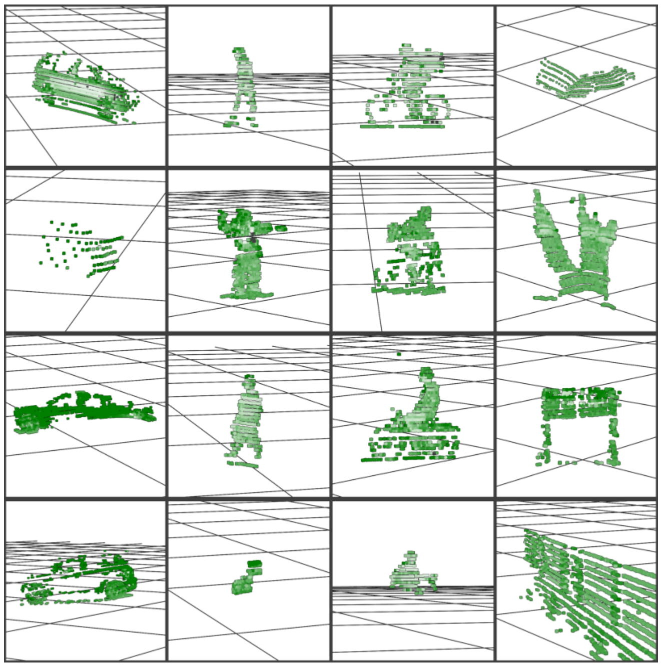
Fig. 3. Example objects from the Stanford Track Collection [17]. This dataset contains tracks of objects segmented using a standard depth-segmentation algorithm used in autonomous vehicles [9]. Each track forms a group of instances, all of which are assumed to share the same hidden label.

To avoid this slowdown, we first specify a simple descriptor transformation. For each descriptor element in the original space, we define a one dimensional grid, or array of bins. Each bin represents a binary feature in the transformed space. See Figure 4 for a visualization.

Then, we choose the form of our weak classifiers to be $h(x) = \alpha c(x)$, where $c : \mathbb{R}^n \rightarrow \{0, 1\}$ is a simple test and $\alpha \in \mathbb{R}$ is the response value for this weak classifier. In the case of [14], c encodes containment within a hypersphere, whereas in this paper it encodes containment in one of the aforementioned bins. The boosting training algorithm is similar to [14], but where we can now store one parameter per bin rather than per weak classifier. That is, multiple weak classifiers can live in the same bin, and their responses add.

The key property of this formulation is that we can express classification as a sum over the fixed number of bins rather than a sum over the growing number of weak classifiers. Formally, this is

$$H(x) = \sum_{k=1}^{K} h_k(x) = \sum_{b=1}^{B} z_b c_b(x),$$

where b is the bin index and z_b is the sum of the αs for all weak classifiers that live in bin b.

C. Descriptor spaces

We use the same 29 descriptor spaces as [14], all deriving from pointcloud data. These descriptors include oriented bounding box size, spin images, and HOG descriptors [3]

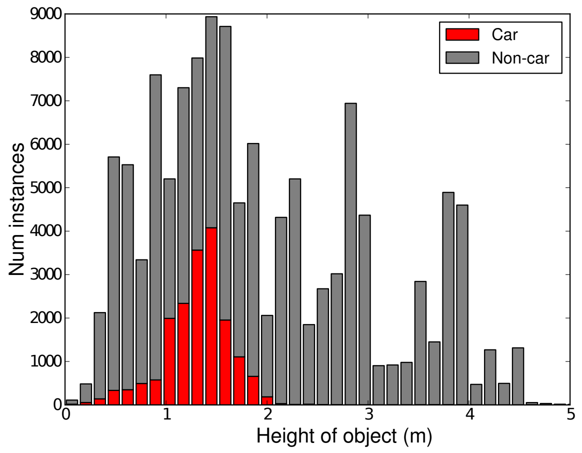

Fig. 4. Intuition for the grid feature transform we use. The bar plot shows histograms of positive and negative examples of cars vs descriptor element value. A naive application of the standard online learning technique of logistic regression assumes a linear relationship between each individual descriptor element and the log odds. That would mean assuming, for example, that as the height of an object increases, the log odds of being a car increases or decreases linearly; this is clearly wrong for our data. By instead having a binary variable per bin, we can learn that cars tend to live within a certain height range. The parametric boosting classifier of Section IV-B builds on this feature transform by using weak classifiers that make predictions for a single bin.

computed on virtual orthographic camera images. This defines an aggregate descriptor space of $\mathbb{R}^{4123}$. We then define two grids, one with 10 bins and one with 100 bins, across each element as described in the previous section. Bin width for a descriptor element is determined by computing the minimum and maximum values on a large unlabeled dataset.

V. EXPERIMENTS

A. *Fully-supervised baseline*

Before delving into group induction, we evaluate our parametric boosting classifier to ensure it can reach reasonable accuracy on a fully-supervised problem. Simultaneously, we provide an estimate of how many user-annotated examples are needed to reach a given level of accuracy.

In this experiment, we train using subsets of the ∼8000 tracks (∼700,000 frames) in the STC training set and evaluate on the ∼6000 tracks in the STC test set. As in [14], the training set is broken down into hand-labeled data $\mathbb{T}_0$ and automatically-labeled negative examples $\mathbb{B}_0$. The latter dataset is collected by driving in places without pedestrians, bicyclists, or cars and automatically labeling all segmented objects as negative examples of each of the three classes. This dataset does not require hand-labeling of individual tracks and is typically quite easy to collect, so we do not count it towards the number of hand-labeled examples.

To estimate accuracy as a function of hand-labeled data, we provide $\mathbb{T}_0$ to the algorithm one logfile at a time, repeating the experiment four times with different logfile orderings. For each evaluation on a subset of $\mathbb{T}_0$, the full background dataset $\mathbb{B}_0$ is also provided. Results can be seen in Figure 5. The x-axis only counts the subset of $\mathbb{T}_0$.

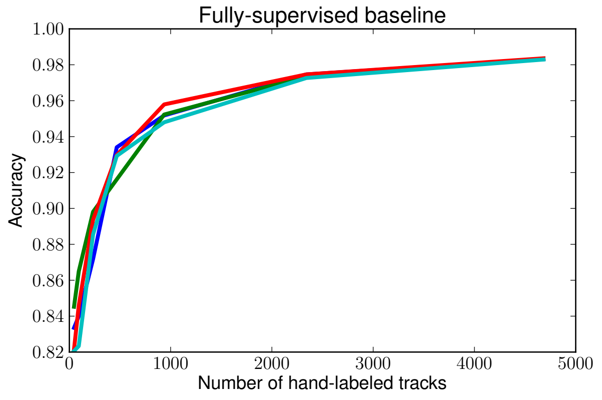

Fig. 5. Supervised accuracy of the parametric boosting classifier vs amount of training data provided. (Group induction is not used here.) Each line shows results for a different random ordering of the training data. Parametric boosting is similar to that used in previous work [14], but has the advantage of constant runtime - an essential property for lifelong learning. This experiment demonstrates that the new formulation can produce accuracy similar to the old method while delivering an asymptotic speed improvement. It also informs us of the number of training examples needed to achieve a given level of accuracy with supervised learning.

The STC dataset was sampled uniformly from the objects seen by our vehicle, so the proportion of background objects is quite high. Predicting $y = -1$ for all class problems results in accuracy of 81.8%. Previous work [14] achieved 98.7% correct, and the parametric boosting classifier of this paper gets to 98.3%. Thus, parametric boosting gives up a small amount of accuracy, but this is a small price to pay for constant runtime and thus the ability to run continuously.

B. *Group induction experiments*

We evaluate group induction for autonomous driving on the STC test set. Because negative examples of cars, pedestrians, and bicyclists are so easy to collect, we simplify the group induction problem by only allowing $y_g \in \{0, 1\}$ and assuming that a large dataset of negative examples is provided. For this, we use $\mathbb{B}_0$ from the previous section.

Our implementation otherwise follows that of Section III, with $\epsilon_{\text{max}} = 0$ and using the retrospection strategy as described in Section III-B. To ensure fixed memory footprint, we use the method described in Section III-C and limit group size to a maximum of 10. The latter is accomplished by simply breaking long unlabeled tracks into shorter ones. The group size of 10 is motivated by the use of exponential loss; long tracks are difficult to induct in the presence of segmentation and tracking noise, as visualized in Figure 2.

The number of unlabeled groups stored in memory is bounded by the approximation described in Section III-C, with the maximum number of unlabeled groups set to 20,000. This number is motivated by the fact that $\mathbb{B}_0$ contains about 150,000 frames, and so our number of inducted positive examples will be of the same order of magnitude.

We used $\mathbb{T}_0$ stripped of its labels along with several additional unlabeled datasets for the unlabeled data, totaling 2.3M frames and 46GB on disk. To simulate an infinite data stream with this finite dataset, during each induction phase a random logfile was loaded and added as a new unlabeled chunk.

New user annotations were added by classifying logs, sorting by confidence, and visually inspecting the low-confidence tracks - a simple kind of active learning. Each run of active learning resulted in new user annotations, and each took on the order of one to ten minutes of user time. Total computation time of the experiment was about 2 days.

Experimental results are presented in Figure 6. The key results are that

- it is possible to achieve good accuracy (98.1%) using on the order of ten to one hundred user-annotated examples for each class problem, plus the automatically labeled background dataset $\mathbb{B}_0$.
- supervised training on $\mathbb{B}_0$ and all user-annotated tracks shown in Figure 6 produces only 88.6% accuracy; group induction is necessary.
- the baseline experiment of Figure 5 suggests that in a fully-supervised context, about 4000 user-annotated examples (in addition to $\mathbb{B}_0$) would be necessary to reach the same level of accuracy.
- unlike [14], our method can learn continuously while maintaining constant classification speed.
- unlike [14], our method can adapt to small amounts of user feedback.

Moreover, the user annotations are dominated by 104 annotations of non-cars. In practice, this is an indication that one should simply use more automatically-labeled background data.

VI. CONCLUSIONS

In summary, we have presented a new mathematical framework for semisupervised learning, applicable any time one has access to groups of unlabeled instances with shared hidden labels. We have shown group induction can dramatically reduce the amount of user-annotated data necessary and that, unlike previous work, it can make use of streams of unlabeled data and user feedback. Group induction is potentially a viable path to lifelong learning perception systems that are trainable by regular users rather than machine learning experts.

Several primary avenues of future work are clear. First, in our experiments we have made use of an automatically-labeled background dataset and used a single-induction implementation, *i.e.* the system only inducts positive examples. While the background dataset is often easy to collect, it would be better if it were not required. Second, our segmentation and tracking solution is designed for street scenes, where objects tend to avoid under-segmentation with the environment. In less structured environments such as homes or offices, this kind of model-free segmentation and tracking is not generally available. Some initial work in this direction exists [13], but more work remains to be done before a sufficiently robust and real-time solution is available.

ACKNOWLEDGMENTS

Thanks to Neal Parikh and Jesse Levinson for helpful discussions and comments.

REFERENCES

[1] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Conference on Computational Learning Theory (COLT)*.

[2] Hendrik Dahlkamp, Adrian Kaehler, David Stavens, Sebastian Thrun, and Gary Bradski. Self-supervised monocular road detection in desert terrain. In *Robotics: Science and Systems*, 2006.

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. In *Journal of the Royal Statistical Society, Series B (Methodological)*, 1977.

[5] Cristian Dima and Martial Hebert. Active learning for outdoor obstacle detection. In *RSS*, 2005.

[6] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. 2000.

[7] M. Pawan Kumar, Ben Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NIPS*, 2010.

[8] Kevin Lai and Dieter Fox. 3D laser scan classification using web data and domain adaptation. In *RSS*, 2009.

[9] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, Michael Sokolsky, Ganymed Stanek, David Stavens, Alex Teichman, Moritz Werling, and Sebastian Thrun. Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium*, 2011.

[10] Mike Montemerlo. The DARPA Urban Challenge: Perspectives on urban autonomous driving. In *ICRA Workshop*, 2008.

[11] Burr Settles. Active learning literature survey. Tech Report 1648, University of Wisconsin–Madison, 2010.

[12] Alex Teichman and Sebastian Thrun. Tracking-based semi-supervised learning. In *Robotics: Science and Systems*, 2011.

[13] Alex Teichman and Sebastian Thrun. Learning to segment and track in RBGD. In *WAFR*, 2012.

[14] Alex Teichman and Sebastian Thrun. Tracking-based semi-supervised learning. In *International Journal of Robotics Research*, 2012.

[15] Alex Teichman and Sebastian Thrun. Online, semi-supervised learning for long-term interaction with object recognition systems. Invited talk at *RSS Workshop on Long-term Operation of Autonomous Robotic Systems in Changing Environments*, 2012.

[16] Alex Teichman, Jesse Levinson, and Sebastian Thrun. Towards 3D object recognition via classification of arbitrary object tracks. In *International Conference on Robotics and Automation*, 2011.

[17] Alex Teichman, Jesse Levinson, and Sebastian Thrun. The Stanford Track Collection, 2011. URL `http://cs.stanford.edu/people/teichman/stc`.

[18] Paul Vernaza, Ben Taskar, and Daniel D. Lee. Online, self-supervised terrain classification via discriminatively trained submodular markov random fields. In *ICRA*, 2008.

[19] Kai M. Wurm, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. In *ICRA*, 2009.

[20] Xiaojin Zhu. Semi-supervised learning literature survey. Tech Report 1530, University of Wisconsin–Madison, 2005.

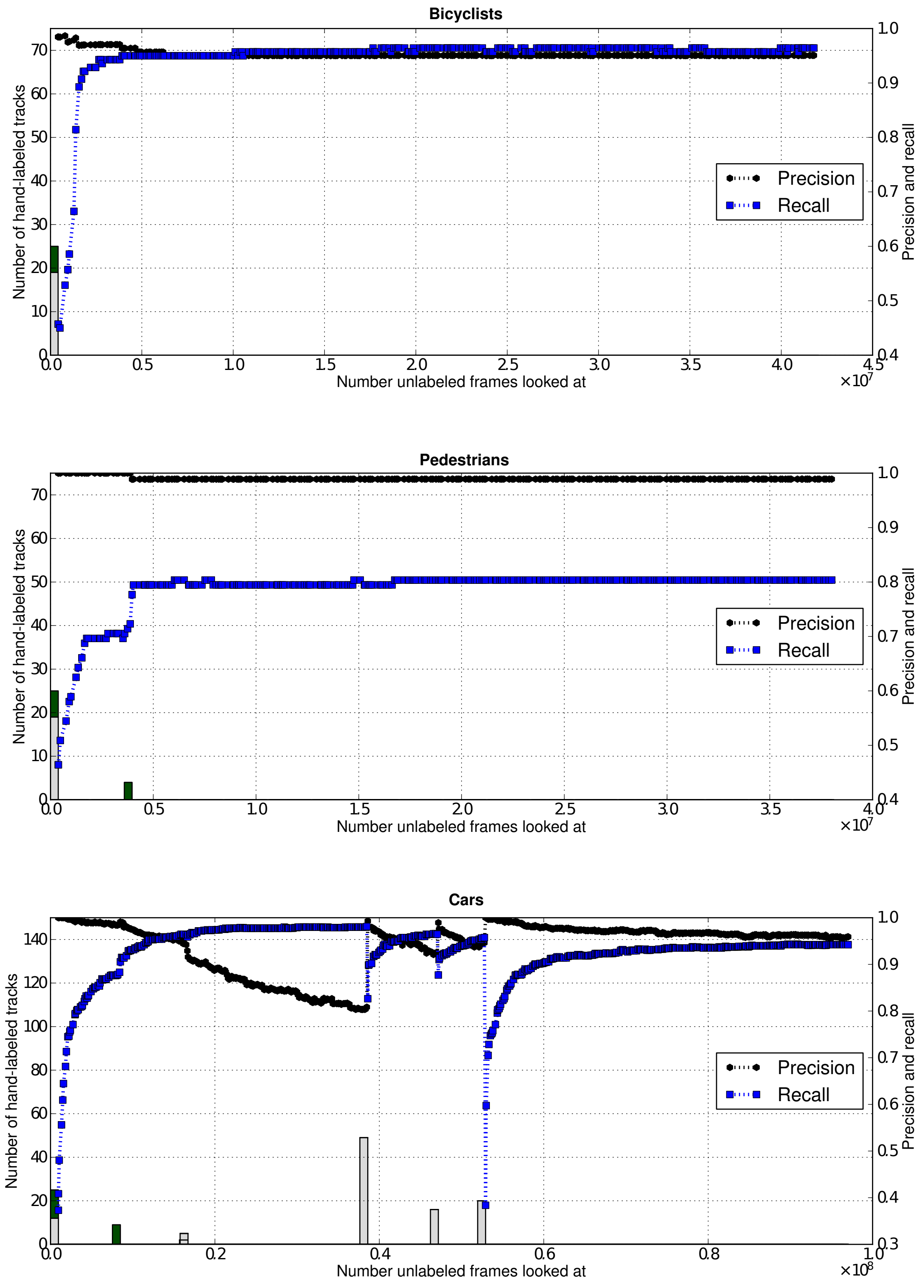

Fig. 6. Experimental results showing ability of group induction to adapt to user feedback and make use of large amounts of unlabeled data. Bar plots show user annotations, with positive examples in green and negative examples in gray. Notice, for example, the jump in recall when new positive examples of pedestrians are added. The retrospection strategy discussed in Section III-B is particularly evident in the final addition of negative examples to the car problem: Very confident false positive inductions were identified, leading to de-induction of most of the buffer. Performance then converges to a stable and accurate solution. Bicyclist and pedestrian plots are clipped to show detail in the early stages; they remain stable through the end. The final accuracy of group induction is 98.1%. In contrast, training on just the annotated examples from this run (*i.e.* without applying group induction) produces final accuracy of 88.6%. Predicting $y = -1$ for all class problems produces final accuracy of 81.8%. Amounts of user-annotated data in this figure can be fairly compared with those in the fully-supervised experiment of Figure 5.