Contents

Workshop website (with videos): `https://simons.berkeley.edu/workshops/schedule/3749`

1 Machine Learning from Verbal User Instruction (Tom Mitchell, Carnegie Mellon University)

Abstract: Unlike traditional machine learning methods, humans often learn from natural language instruction. As users become increasingly accustomed to interacting with computer devices using speech, their interest in instructing these devices in natural language is likely to grow.

We present our Learning by Instruction Agent (LIA), an intelligent personal agent that users can teach to perform new action sequences to achieve new commands, using solely natural language interaction. LIA uses a CCG semantic parser to ground the semantics of each command in terms of primitive executable procedures defined in terms of the sensors and effectors of the agent. When LIA is given a natural language command that does not understand, it prompts the user to explain how to achieve the command through a sequence of steps, also specified in natural language. As a result of the instruction episode, LIA learns two types of knowledge: (1) a new procedure defined in terms of previously understood commands, and (2) an extension to its natural language lexicon that enables it to understand and execute the new command across multiple contexts (e.g., having been taught how to "forward an email to Alice," LIA can correctly interpret the command "forward this email to Bob." This talk will present LIA, plus ongoing research to extend it to include demonstration-based learning, and theoretical questions raised by such learning from instruction.

Background reading: `http://ai2-website.s3.amazonaws.com/publications/LearnByInst.pdf`

To think about the future of machine learning, look at how humans learn that machines don't. ML has been overfocused on one paradigm.

Suppose there is a sensor-effector system (like a phone) you would like to teach through human instruction. Sensors include mike, camera, GPS, web pages, incoming message, news... Effectors include vibrate, speak word, display message, seach web, invoke app...

If you could teach your phone things, what would you teach? Most things we say now are instantaneous: what is the weather?

We can have conditional strategies monitored all the time. Ex. Google has: Whenever I have to catch a flight, alert me to head to the airport so that I'll arrive an hour before flight time.

But there are lots of other things I want:

- Whenever it snows at night, set my alarm 30 minutes earlier.
- If somebody sends me a picture containing my mom, email it to her.
- Whenever a student sends email about a homework question, forward it to the right teaching assistant.

These are all in the sensor-effectuator closer of the phone: depending on things the phone can sense and actions the phone can do.

We assume

- set of sensors, effectors
- primitive language capability (language lexicon able to invoke each sensor, effector)
- general learning mechanisms.

We want the system to learn

- langauge: lexicon entries to understand new phrases. ("drop a note to Bill") The system might not know the phrase at first, but it should learn it.

- Capabilities: new sensors and effectors defined in terms of existing.
- Representations: new concepts and instances linked to instance.

  Ex. define concept "workshop". One instance is Interactive Learning 2017.

  This becomes another sensor. "Whenever I'm attending the workshop,..."

I'll show a prototype which allows teaching new commands.

1.1 Teaching new commands

Philosophy:

- Teaching the agent should be like teaching another human: it will remember.
- Anyone should be able to teach. Anyone can become a programmer!
- Verbal instruction and demonstration
- Anytime.

If the system doesn't know what something means, "I don't know how to do that, can you teach me what it means?"

- "Tell Tom to come to the meeting a few minutes early."
- Instruct by breaking into substeps.
- "I'm trying to generalize to other similar commands."
- "Tell Bob to bring some coffee." This is now comprehendable!

The system learned new language competence, and learned a procedure.

What it means to understand a command means to create the code to make the command execute.

The generalization comes out of inherent structure of the semantic parsing, and some heuristics to make it work as well as possible.

If the generalization was not perfect, how to correct? No good answer yet...

This system right now deals when unknown semantics have to do with phrasings and verbs (tell). It didn't know "tell" before; now it can show the semantics of "tell".

Meta-conversations: remember that time when I told you to tell Bob to bring coffee...

CCG (combinatory categorical grammar) parsing maps natural language to logical forms. "Prepare an email to Bill" becomes (doSeq (createEmail) (set recipient Bill)).

Parsing steps combine sentence parts both syntactically and semantically: Adjective Noun is a NP, where the logical form of the NP is Adjective(Noun).

"Set the subject to time to go."

- subject: FieldName, becomes (lambda x (getMutableFieldByFIeldName x)).

- time to go: StringV
- to: absorb phrase to right PP_StringV.
- End: (setFieldFromString (getMutableFieldByFieldName subject) (stringValue "time to go")).

Grammar rules can be ambiguous: we say this "can" be parsed a certain way rather than it will.

There are 2 things to learn: Learn new parsing rules and right control strategies to control the parse. Ex. "set" makes it more likely to parse the RHS as a string.

In practice, the number of potential parses increases rapidly as vocabulary.

We learn loglinearly and use beam search, keep 15 most valued parses. If you make complex sentences, the beam is not wide enough.

In the example, the algorithm learned the semantics of "tell".

Generalization algorithm: parameterize any text span whose semantic parse is a subtree of the full sentence semantic parse (replace by variable, ex. arg0="Tom", arg1="I am late").

We don't have things like yet: "Inform" is a synonym of "tell".

Everything is symbolic so far. Logical forms do have to be executable. SA: Embeddings could inform probabilities.

What about sending email vs. text? (Synonyms in action space.) It doesn't have a notion of goals, subgoals, planning. This would be a much richer substrate.

We ran experiments on Mechanical Turk. Create a new email and set the recipient to the current email's sender. Set recipient to Mom. I can't. Set recipient to Mom's email. This is a clarification that teaches the machine.

Subject taught agent new commands, saved on average 52 subcommands. Only 41% completed the task.

No cumulation in experiment. People can have own dialects (ex. send email vs. text). How about customizing from fully loaded sytem?

1.2 Teaching conditional strategies to automatically invoke commands

If (sensor value) then (command).

1. If I get new email from...: this is easily defined in language, "Test whether value of the from: field equals..."

   Declarative.

2. Concept definable as active sensing procedure. "If it snows at night..." Open weather app, read current conditions, equal to snow?

   Procedural.

3. Concept not definable in closed form, or procedurally. "If I get spam email..." No closed declarative or procedural definition. Generalize from examples with user instruction.

Typical user instruction is "it's probably spam if it's trying to sell me something."

The meaning has to be grounded in observable properties in email. Our approach is to jointly learn meaning of these natural language statements, (boolean) values for individual emails, and a classifier based on these values plus observed features.

MTurkers generated emails in the following classes.: contact, employee, event... Other MTurkers provided advice on classifying them.

The F1 score of the learned classifier is higher with sentences provided by users.

1.3 Teaching by demonstration

SUGILITE: Creating multimodal smartphone automation by demonstration.

Ex. "Order a Cappucino." Demonstrate using any third party apps. Determine parameter using voice command. Analyze app's user interface. Click on behalf of user.

1.4 Future work

Future work:

- Integrate verbal instruction with demonstration
- Share learning across agent (phones)
- Incremental refinement ("if it snows" to "if it snows lots")
- mixed initiative teachine dialogs
- If (sensor) then (effector) because (goal). This gives us a way to teach the system reasoning!
- Combine statistical, verbal, demonstration, and experiments.

Theory needed:

- Sample complexity impact from different types of instructions.

  Communcation is not necessarily a label, but another type of instruction: demonstrations with or without commentary, suggested new features...

  There is an effectively infinite set of features defined by a grammar. In ML we usually assume we have a nice set of features.

- Theory of agent architectures: what are sufficient initial capabilities to assure learnability of everything in the sensor-effector closure?

  What is the value of a good curriculum?

  What are sufficient conditions to assure non-damaging learning?

In one sentence: because now computers can understand what we're saying to some degree, there is this huge new underexplored area of verbal user instruction. Opening it up we get into interesting practical and theoretical questions, ex. incompleteness of verbal instruction.

Android platform easier than iOS. Toby Li: track every keystroke on every application.

Inverse question: how machine can teach humans to teach, interact in way it can understand. Student saying "that wasn't very good". I didn't understand your command because... there are many Bobs, I couldn't parse this last part...

Speech recognition is a bottleneck. We use google speech. One problem is that you get back the whole things; we want to track in real time the utterance. There are other speech systems at CMU (finer grained) but they are not as globally competent. Is there a way to make speech recognition better with grammar/parsing?

Gather vague and informative signals? Attention, etc. OpenSmile.

Part of the experiment: given human understanding, can they break it into pieces? Maybe it doesn't have to work as well as human understanding.

2 Machine teaching in interactive learning, Jerry Zhu, University of Wisconsin-Madison

Abstract: If machine learning is to discover knowledge from data, then machine teaching is an inverse problem to pass the knowledge on. More precisely, given a learning algorithm and a target model, the goal of machine teaching is to construct an optimal (e.g. the smallest) training set from which the algorithm will learn the target model. I will discuss several aspects of machine teaching that connect to interactive machine learning.

What do we really want from interactivity?

Imagine we're doing binary classification, $x \in [0,1]$, label $y \in \{-1,1\}$, the hypothesis space is threshold classifiers $\mathcal{H} = \{\theta \in [0,1] : \widehat{y} = \mathbb{1}_{x \geq \theta}\}$.

Contrast 3 types of teaching. Assume noiseless.

1. PAC (passive) learning: sample iid. A learner gets a batch of iid samples. With large probability

$$|\widehat{\theta} - \theta^*| = O(n^{-1}) \leq \varepsilon \tag{1}$$
$$n \geq O(\varepsilon^{-1}). \tag{2}$$

   We can do much better with active learning.

2. Active learning: the learner picks query x; human oracle answers $y = \theta^*(x)$. Do binary search,

$$|\widehat{\theta} - \theta^*| = O(2^{-n}) \leq \varepsilon \tag{3}$$
$$n \geq O(\lg(\varepsilon^{-1})). \tag{4}$$

3. An ideal human teacher: Teacher knows the learner and designs an optimal training set. $n = 2$ for all $\varepsilon > 0$ by giving 2 opposite items ε away from each other.

1. Machine teaching: what can we expect from an ideal teacher?
2. The real world is not ideal.

2.1 Humans are teachers, not annotators. What can an ideal teacher do?

We make strong assumptions.

1. Teacher knows target model $\theta^* \in \mathcal{H}$.
2. Teacher can give training set D but not θ^* to learner.
   - Constructive teaching (can lie) $D \in \mathbb{D} = \bigcup_{n=1}^{\infty} (X \times Y)^n$.
   - Constructive teaching (honest) $D \in \mathbb{D} = (\bigcup_{n=1}^{\infty} X^n, Y = \theta^*(X))$.
   - Pool-based teaching $D \in \mathbb{D} = 2^{\{(x_i, y_i)\}_{1:N}}$. Can only select examples from a pool.
3. Teacher knows learning algorithm/estimator/student $A : \mathbb{D} \to 2^{\mathcal{H}}$.
   - Ex. version space learner $A(D) = \{\theta \in \mathcal{H} : \theta(x_i) = y_i, 1 \le i \le n\}$.
   - Ex. regularized empirical risk minimizer $A(D) = \operatorname{argmin}_\theta \sum_{i=1}^n \ell(\theta, x_i, y_i) + \lambda \|\theta\|$.

The teacher is solving a problem in the space of training sets. Objective is size of training set.

$$\min_{D \in \mathbb{D}, \theta^* \in A(D)} \|D\|_0 .$$

This is like the "inverse" of machine learning, find something in $A^{-1}(\{\theta^*\})$. An alternative view is from coding: the message is θ^*, the decoder is A, the language is $\mathbb{D}$. It's a strange decoder, maps the training set to the model.

A special case is classical optimal teaching (Goldman, Kearns 95). Further restrictions: A is a version space learner, often for $X, \mathbb{D}, \mathcal{H}$ finite.

The teaching dimension is

$$\min_{D \in \mathbb{D}, \theta^* \in A(D)} \|D\|_0 \tag{5}$$

$$TD(\mathcal{H}) := \sup_{\theta^* \in \mathcal{H}} TD(\theta^*) \tag{6}$$

I'll move to other definitions depending on the learning algorithm.

Example 2.1: Think of hard-margin SVM. We want to teach a target model—a hyperplane. The teacher can do constructive teaching (pick arbitrary items). What's the smallest training set?

You can do it with 2. Place the examples such that the hyperplane is the perpendicular bisector of the hyperplane. So $TD = 2$ and $VC = d + 1$.

Example 2.2: Teach a d-dimension Gaussian to the maximum likelihood estimator.

In d dimension, choose tetrahedron vertices, $TD = d + 1$.

To teach linear learners (ridge regression, soft SVM, logistic regression), $A(D) = \operatorname{argmin}_{\theta \in \mathbb{R}^d} \sum_{i=1}^{n} \ell(\theta^T x_i$ $\frac{\lambda}{2} \|\theta\|_2^2$.

TD depends on the loss function. For just learning the boundary, TD dimension is 1 for squared, hinge, and logistic loss: you can teach by a single example even with one positive example. This is because the student has a regularizer, which acts as a negative pull from the training example.

For ridge regression, the singleton set is

$$x_1 = a\theta^*, \quad y_1 = \frac{\lambda + \|x_1\|^2}{a}.$$

You can take any $a \neq 0$. Ex. to teach $\lambda = 1$ student the target $\theta^* = 1$, the teacher lies: $x_1 = 1, y_1 = 2$. The teacher needs to lie because the student is shrinking the estimate. Cancel out the regularizer.

The student doesn't know it's being taught; it just follows its algorithm. (What if the student expects to be taught?)

TD is like the "speed of light". Unavoidable effort in interactive machine learning: $n \geq TD$. Ideal teacher achieves $n = TD$. This can be much faster than active learning (ex. 2 vs. $\lg\left(\frac{1}{\varepsilon}\right)$). We must allow teacher-initiated items, unlike active learning. (You can't allow the computer to pick the items.)

We can also define learner risk, teacher effort, constraints (minimize subject to tolerance, or subject to budget). There is a Lagrangian form.

Suppose the teacher is restricted to a pool of items. Let the pool be n uniform items from $[-1, 1]$. Take the innermost pair and learner puts the boundary in the middle. Giving the whole pool achieves $\frac{1}{n}$. Because the learner is special, we can achieve $\frac{1}{n^2}$. Pick the most symmetric pair! Whp,

$$|\widehat{\theta} - \theta^*| = O(n^{-2}).$$

This is not training set reduction nor sample compression.

This is boring.

2.2 Most humans are not ideal teachers

I will continue using the 1-D example.

How do real humans teach? (Cover story: they choose their own θ^*. We told them what the learner is doing; we're not sure they understood it.) (One failure mode is when the teacher doesn't know what the student is doing.) (Instruction: please give the smallest training set.) People seemed to try to cover the hypothesis space.

1. 30% move linearly in space.
2. 40% crazy.
3. Few people gave 2 examples. Often they start with 2 examples, but add more to make sure.

The mixed-initiative algorithm: from $i = 1$ to TD, let human give (x, y) pair, to allow ideal human teachers. Afterwards the computer takes control and runs active learning starting from D until completion. (Good teachers should be allowed to do their thing. If the human isn't good at teaching, running active learning is good.)

Examples of teachers:

1. Seed teacher: provides one point per positive region.

   The concept class of interval can be hard to learn because it can be narrow. (You need random search. A seed teacher warm-starts the active learner, and reduces to binary search.)

2. Naive teacher: can be arbitrarily bad. (Ex. only give positive examples.)

Fallback guarantee: waste at most TD examples. $TD \leq AL$, so you get factor of 2 at most.

1. Human teachers: a significant fraction were unable to teach the concept (ex. only provide positive examples).

2. Active learning: 14 by binary.

3. Mixed initiative: preserve smart teachers, force those who can't succeed to be around 14–16.

Two ideas to help:

1. Control with mixed-initiative learning.

2. Educate with analogues: automatically generated optimal training sets for arbitrary $\theta' \in \mathcal{H}$: If threshold was \$19000, show \$19000 acceptable, \$19001 unacceptable.

3. Translate the teacher.

   Ex. Don't know the teacher's regularization constant. If human assumes wrong λ^w, then the learner learns wrong θ.

   If translator-in-the-middle knows λ^w, λ^*, it can translates examples $\widetilde{x} = \frac{xy}{x^2+\lambda^w}$.

   The translator is the meta-algorithm.

   This is not realistic, but may be useful when we know roughly what the teacher is teaching (ex. mixture of Gaussians), while in reality the student is running something else (ex. SVM).

`http://pages.cs.wisc.edu/~jerryzhu/machineteaching/`

Recruit people with teaching experience? We asked if they were teachers or parents, but didn't see correlation. Tasks may be too artificial.

Role of adaptivity?

Imperfect memory?

Build from psychology: models human might assume of human student?

3 Interactively Learning Robot Objective Functions (Anca Dragan, UC Berkeley)

Abstract: Inverse Reinforcement Learning enables robots to learn objective functions by observing expert behavior, but implicitly assumes that the robot is a passive observer and that the person is an uninterested expert. In this talk, I will share our recent works in making this learning process interactive, where humans teach rather than provide expert demonstrations, and robots act to actively gather information to showcase what they've learned.

Formally specifying objective functions is hard. Ex. asking a vacuum to suck dust: the robot pours dust on the floor and sucks it.

How do we get a robot to learn objective functions that are implicit in people's heads but hard to write down?

A robot car can drive according to passenger preferences, and predict how other drivers will drive.

The framework is inverse reinforcement learning. The person implicitly knows parameters of θ but can't explicate or write it down, but can demonstrate behavior according to θ:

$$\mathbb{P}(u|\theta, x_0) \propto e^{U_\theta(x_0,u)}.$$

The action can be the person acting or operating the robot.

Arobot can take this evidence to update its belief about θ.

$$b'(\theta) \propto P(u|\theta, x_0)b(\theta)$$

This is implicitly making 2 assumptions we don't want to make.

1. The robot is a passive observer during the learning phase.

   Robots are actuated agents. Can we leverage this to improve learning?

   Desiderata 1: Robots should leverage their action to improve learning.

2. We're asking people to behave like uninterested experts, optimally with respect to θ.

   This is like the machine having to learn gymnastics from learning Gabby Douglas (Olympic level); instead it's easier to learn from a coach.

We formulate learning as a cooperative 2-player game.

This gives the person an interest in the robot's learning.

In cooperative inverse RL, the human has an objective function $U_\theta(x_0, u_R, u_H)$ and the robot has an objective function $U_R(x_0, u_R, u_H)$. Go a step further and equate the utility function. The robot acts to improve estimation and the human has an incentive for the robot to improve its estimation, learn the right θ.

The challenge is tractability: how to solve this?

Inverse RL is one approximation: fix the human's policy to expert demonstration π_H. The robot assumes no more human actions (it won't get more information) and chooses the best response $\pi_R = br(\pi_H)$.

Setting it up as the game allows us to think about other approximations that take us closer to the desiderata.

First: how to model as a game.

In autonomous driving, estimate θ because we assume the person acts according to it. It's important to be able to do this as an ongoing interaction because people drive in different ways: aggressive, distracted, defensive, attentive.

All users react in almost the same way. If a robot just does inverse RL, it doesn't get a lot of information. If the robot tries to merge left, it allows the other person to go first. This is typical behavior; they're defensive. That's not what humans do. I have to stick myself in front of people and force them to allow me in.

"Google's driverless cars run into problem: cars with drivers." The human drivers kept inching forward looking for the advantage—paralyzing Google's robot.

We thought: what if instead of just observing actions, allow the robot to act, and the human's actions depend:

$$\mathbb{P}(u_H|u_R, \theta, x_0) \propto e^{U_\theta(x_0, u_R, u_H)}.$$

Incentivize information gain. Robot updates belief $b'(\theta) \propto P(u_H|u_R, \theta, x_0)b(\theta)$ and optimizes

$$u_R^* = \text{argmax}_{u_R} \mathop{\mathbb{E}}_{\theta}[H(b) - H(b')] = \text{argmin}_{u_R} \int H(b')b(\theta)\, d\theta..$$

Choose actions to aid learning. Trade off exploitation and information gain. Learning more about the other person helps make better decisions.

This is a POMDP but we can't solve them well in continuous state and action space.

This is used in navigation, localization, etc. We're doing it not over physical state but human internal state.

Simplification: Assume optimal observation (rather than draw from distribution)

$$u_H = \text{argmax}\, U_\theta(x_0, u_R, u_H).$$

where $b'(\theta) \propto \mathbb{P}(u_H|u_R, \theta, x_0)b(\theta)$. Use quasi-Newton method. Use implicit differentiation.

Replace integral by sum,

$$\text{argmin}_{u_R} \sum_\theta H(b')b(\theta).$$

Offline do clustering of user types. Identify type of user online.

When merging, the robot car nudges in. If the person is distracted, go back. If the person is attentive, merge. At an intersection, inch forward. See one behavior from distracted (continue) and another from attentive users (stop). If distracted, backs up.

Data from a simulator.

Information gathering improves estimation.

In order to interact properly, we can't treat them as obstacles; we have to interact with them.

The robot autonomously generates strategies that we normally hand-code.

Right-of-way? We haven't put in those constraint.

How does the robot do with a copy of itself? That would be different.

There's another side of learning how the passenger wants to drive; make queries to the expert. Generate trajectories, which would you prefer?

$$\text{argmax}_{x_0, u_R^1, u_R^2} \mathop{\mathbb{E}}_{\theta}[H(b) - H(b')].$$

The person (teacher) tries to figure out how the robot is learning, and responds differently from just an expert demonstration:

$$\pi_H \neq br(br(\pi_H)).$$

Ex. Robot doesn't know the relative value of peaks of the optimization function. Expert goes straight to the closest peak. The best response visits both, gives robot evidence that the other peak is also good.

There's a sweet spot: if the person is optimal, there's little you can do. If there's enough flexibility, people can take interesting teaching actions.

Looking at the problem this way speeds up learning.

Safety vs. information? Adapt tradeoff with safety over time. Reason about value of information you gain.

Other relaxations? Game makes a lot of assumptions about person having perfect knowledge. This is not the case. Person has own beliefs. I don't think iterating on best responses buys us that much.

4 Words, Pictures, and Common Sense: Learning by playing (Devi Parikh, Georgia Institute of Technology)

Abstract: Wouldn't it be nice if machines could understand content in images and communicate this understanding as effectively as humans? Such technology would be immensely powerful, be it for aiding a visually-impaired user navigate a world built by the sighted, assisting an analyst in extracting relevant information from a surveillance feed, educating a child playing a game on a touch screen, providing information to a spectator at an art gallery, or interacting with a robot. As computer vision and natural language processing techniques are maturing, we are closer to achieving this dream than we have ever been.

In this talk, I will present two ongoing thrusts in my lab that push the boundaries of AI capabilities at the intersection of vision, language, and commonsense reasoning.

Visual Question Answering (VQA): I will describe the task, our dataset (the largest and most complex of its kind), our model, and ongoing work for free-form and open-ended Visual Question Answering (VQA). Given an image and a natural language question about the image (e.g., "What kind of store is this?", "How many people are waiting in the queue?", "Is it safe to cross the street?"), the machines task is to automatically produce an accurate natural language answer ("bakery", "5", "Yes"). We have collected and recently released a dataset containing ¿250,000 images, ¿760,000 questions, and ≈ 10 million answers. Our dataset is enabling the next generation of AI systems, often based on deep learning techniques, for

understanding images and language, and performing complex reasoning; in our lab and the community at large.

Learning Common Sense Through Visual Abstraction: Common sense is a key ingredient in building intelligent machines that make "human-like" decisions when performing tasks – be it automatically answering natural language questions, or understanding images and videos. How can machines learn this common sense? While some of this knowledge is explicitly stated in human-generated text (books, articles, blogs, etc.), much of this knowledge is unwritten. While unwritten, it is not unseen! The visual world around us is full of structure bound by commonsense laws. But machines today cannot learn common sense directly by observing our visual world because they cannot accurately perform detailed visual recognition in images and videos. This leads to a chicken-and-egg problem: we would like to learn common sense to allow machines to understand images accurately, but in order to learn common sense, we need accurate image parsing. We argue that the solution is to give up on photorealism. We propose to leverage abstract scenes – cartoon scenes made from clip art by crowd sourced humans – to teach our machines common sense.

4.1 Common sense from cartoons

Classical computer vision: describe what is going on in images.

Hard: "A man is *rescued* from his truck that is hanging *dangerously* from a bridge."

This requires commonsense reasoning.

Where do we gather this common sense from? We can use all the text on the web.

There is reporting bias in what people write about online. Ex. people inhale 6 times more often than they exhale, and get murdered 17 times as often. People have heads to gallbladders 1085:1.

Is there a way to watch the world around us and use the structure to learn commonsense knowledge. Ex. two people conversing in front of a blackboard vs. two people standing in front of a blackboard. Gaze helps us differentiate.

This is hard:

- lack visual density: hard to find two images where just gaze differs.
- Annotations are expensive
- Why need annotations? Computer vision doesn't work well enough.

There is a chicken-and-egg problem: learn common sense vs. improve image understanding

Do we need photorealism? No, it lies in semantic content of scene. We introduce Mike and Jenny with variety of objects. They can have different expressions and poses. We can get people to create visual data for us.

"Mike fights off a bear by giving him a hotdog while Jenny runs away."

People make different images: consistent: Mike facing bear with hot dog, Jenny running in opposite direction.

We took 1000 classes, 10 MTurkers for each.

Each image is trivially annotated. Which visual features are important for semantic meaning? Which words correlate with specific visual features?

We have full control over density of sampling.

We can take an input description (fresh sentence, not in training set), turn them into tuples, and automatically generate images. (Zitnick, Parikh, Vanderwende, ICCV2013)

Based on sentences, update on potential of how likely objects are likely to be there. Ex. "raining" reduces probability of sun relative to prior.

Goal is to use this abstract world to learn commonsense knowledge that generalizes to realistic examples.

Another example (fine-grained interactions): Learning what it looks like to interact with each other. Ex. "Person 1 is dancing with Person 2", walking with, holding hands with, talking with, jumping over, etc. We keep track of all intermediate stages but haven't used it.

Could we use this as training data to help learn real-life interactions (zero-shot learning). We can do statistically significant above chance. You would get similar performance if you train on 2–3 real images vs. if you train on 50 generated images.

This is just a 2-D canvas, while real images have all of that. If you want to improve this, make 3-D canvas.

We initialize poses randomly, hope that people go to the local minimum rather than all converge to the same thing. For Mike and Jenny, we give access to a randomly generated sample of clip art.

4.2 Visual question answering

What color are her eyes? What is the mustache made of? Is this a vegetarian pizza? Recent years: improve 55 to 68%. Human accuracy is 83%.

There is a heavy language bias. "Do you see a...?" is right 87% of the time. When collecting the dataset, people are asking while looking at the image. People ask "Is there a clock?" more often when there is a clock. This hindered progress on binary question.

Sometimes there is a bias towards no. "Should this person be doing...?" The framing of the question gives away information about what the answer is! Problem with using as a benchmark!

We want to rectify this, by removing language priors.

"Is there a place to sit other than the floor?" Modify the image as little as possible to make it true. Now we have 2 complementary scenes which are globally very similar but differ in the answer. Guessing can't do well here.

Only if model answers correctly for both in pair does it get a point. Answering based on question alone you get 0 accuracy. Training on holistic features gets 3.2%. Training on balanced set gets 23.13%. Making the model more sophisticated (parsing, explicit alignment), this attention-based model gets 9.84%, 34.73% in the unbalanced/balanced case.

Here everything has (primary, verb, secondary object) structure. Attention model is learned from training data. It decides how sharp the attention map should be.

Summary:

- Learn by playing

- Fully annotated visual data.
- Allow full control over distribution and density of data. You're not at the mercy of the distribution of Google, flickr...

4.3 Reasoning by imagination

We have text where it helps to imagine the scene behind the text.

- man holds meal
- tree grows in table.

Fill-in-the-blank: Mike is having lunch when he sees a bear. (Mike tries to hide.)

Visual paraphrasing: Are these two descriptions describing the same scene.

- Jenny was going to throw her pie at Mike
- Jenny is very angry. Jenny is holding a pie.

Ex. For multiple choice, generate scene from each answer. Given text with correct answer and scene should score higher than incorrect ones. What's relevant are semantics. This helps.

Ex. Plausibility of "tree grows in table." Find support for description in images *and* database of text.

You should be able to learn from text embeddings, so similarity in new representation space matches.

4.4 Understanding visual humor

People rate humor.

Ask maching to: Recognize humor. Add humor. Remove humor.

Ask people to create funny and nonfunny scenes. What part of scene is funny? Replace with soemthing else so it stops being funny. Which objects are contributing to humor? Replace with something semantically meaningful. Ex. replace with butterflies... Indoor: potted plant.

Test: which scene is less funny.

Turing test: human's funny version vs. algorithm's funny version. Algorithm wins 28% of the time. When there is a lot of something the machine thinks it's funny.

Often we weren't laughing, so we ask them why it's funny. "This terrified woman's home is being invaded by mice while the cat sleeps."

Visual abstraction for...

- studying mappings between images and text.
- zero-shot learning
- studying image memorability, specificity, visual humor
- learn common sense

- Rich annotation modality: ask for descriptions, ask for scenes, show scene and ask for modifications, perturb scene and ask for descriptions.

Study high-level image understanding tasks without waiting for lower-level vision tasks to be resolved.

Further push learning by playing modality.

In images, "look at" and "eat" are close in visual space. These are not close in word embedding space! Word2vec is now informed by this! (Note: not symmetric.)

5 Stochastic variance reduction methods for policy evaluation (Lihong Li)

Abstract: Policy evaluation is a crucial step in many reinforcement-learning problems, which estimates a value function that predicts long-term value of states for a given policy. In this talk, we present stochastic variance reduction algorithms that learn value functions from a fixed dataset, which is shown to have (i) guaranteed linear convergence rate, and (ii) linear complexity (in both sample size and feature dimension), under the condition of linear function approximation and possibly off-policy learning as well as eligibility traces. In particular, we transform the policy evaluation problem into an empirical (quadratic) saddle-point problem and apply stochastic variance reduction methods in the primal-dual space. Interestingly, the algorithms converge linearly even when the quadratic saddle-point problem has only strong concavity but no strong convexity. Numerical experiments on random MDPs and on Mountain Car demonstrate improved performance of our algorithms.

This is not common in RL work because it uses recent advances in optimization. The techniques can be very powerful.

A robot takes actions a_t to affect the world and gets immediate reward r_t.

The value of a state is the expected long-term return from state s,

$$V^\pi(s) := \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t | s_1 = s\right].$$

In the end we care about policy optimization.

A crucial step is to evaluate a fixed policy (policy evaluation).

For convenience, drop π and a for the rest of the talk. We focus on PE with linear function approximation, and a batch setting where data is fixed (experience replay).

We showed first-order algorithms with linear convergence: total time complexity $O\left(nd \ln\left(\frac{1}{\varepsilon}\right)\right)$ where n is sample size, d is dimension, and ε is optimization precision. Previous complexity is $O(nd^2)$ or $O(n/\varepsilon)$.

Key technical ingredients are

- Saddle-point reformulation of policy evaluation. (Usually it's described as an fixed point of an iteration.) This seems to make it more complicated, but we can apply optimization.

- Stochastic variance reduction based on SDVRG and SAGA.

- Eigenvalue analysis of corresponding matrices.

All results can be extended to eligibility traces and off-policy RL.

5.1 Problem setup

We model as Markov reward process (there is no action) $M = \langle S, P, R\rangle$: Markov chain with reward function. S is finite set of states, $\mathbb{P}(s'|s)$ is transition probability, $R(s)$ is expected one-step reward.

Value funciton is $V(s) = \mathbb{E}[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots | s_1 = s]$. Bellman equation says

$$V = R + \gamma PV.$$

V is a fixed point to the linear operator $X \mapsto R + \gamma PX$.

In linear value function approximation, we want an approximation

$$\widehat{V}(s) = \phi(s)^T\theta$$

(or $\hat{V} = \Phi\theta$). Here $\phi(s)$ is a d-dimensional feature vector and $\theta \in \mathbb{R}^d$ has parameters to optimize.

The objective function: Data is a trajectory of n steps, $D = (s_1, r_1, \ldots, r_n, s_{n+1})$ with $\phi_t = \phi(s_t)$. Training target is not given, unlike regression problemn.

We use empirical MSPBE (mean square projected Bellman error)

$$\left\|\widehat{V} - \Pi(R + \gamma P\widehat{V})\right\|_\Xi^2.$$

Precisely...

$$\theta^* = \text{argmin}_\theta \frac{1}{2}\|A\theta - b\|_{C^{-1}}^2 + \frac{\rho}{2}\|\theta\|^2 \tag{7}$$

$$A_t = \phi_t(\phi_t - \phi_{t+1})^T \tag{8}$$

$$A = \frac{1}{n}\sum_t A_t \tag{9}$$

$$b_t = r_t\phi_t \tag{10}$$

$$b = \frac{1}{n}\sum_t b_t \tag{11}$$

$$C_t = \phi_t\phi_t^T \tag{12}$$

$$C = \frac{1}{n}\sum_t C_t. \tag{13}$$

Important is that we have an objective function.

5.2 Saddle-point formulation and algorithms

The objective does not have the sum-over-data structure. It's not straightforward to decompose into individual examples. Direct optimization is not easy, often requiring $O(d^2)$ complexity. GTD2 (Sutton et al 2009) has $O(d)$ complexity but convergence is slow.

The conjugate function of $f(w) = \frac{1}{2}\|w\|_C^2$ is

$$f^*(u) := \sum(u^T w - f(w)) = \frac{1}{2}\|u\|_{C^{-1}}^2 .$$

Rewrite MSPBE as

$$\min_\theta \left[\frac{1}{2}\|A\theta - b\|_{C^{-1}}^2 + \frac{\rho}{2}\|\theta\|^2\right] = \min_\theta \max_w \underbrace{\left[w^T(b - A\theta) - \frac{1}{2}\|w\|_C^2 + \frac{\rho}{2}\|\theta\|^2\right]}_{L(\theta,w)} \tag{14}$$

L is convex-concave, so the we can find the minimax. We can decompose

$$L(\theta, w) = \frac{1}{n}\sum_t L_t(\theta, w).$$

Use existing optimization, like primal dual batch gradient.

Initialize (θ, w). For $m = 1 : M$, update by gradient

$$\begin{pmatrix}\theta \\ w\end{pmatrix} - = \begin{pmatrix}\sigma_\theta & \\ & \sigma_w\end{pmatrix} B(\theta, w) \tag{15}$$

$$B(\theta, w) := \begin{pmatrix}\nabla_\theta L \\ -\nabla'_w L\end{pmatrix} = \begin{pmatrix}\rho\theta - A^T w \\ A\theta - b + Cw\end{pmatrix}. \tag{16}$$

Per-iteration cost is $O(nd)$. This enjoys linear convergence.

But is it not practical because it is linear in n.

We can do a stochastic version. Randomly sample a data point and compute a stochastic gradient.

$$\begin{pmatrix}\theta \\ w\end{pmatrix} - = \begin{pmatrix}\sigma_\theta & \\ & \sigma_w\end{pmatrix} B_{t_m}(\theta, w) \tag{17}$$

$$t \sim U[1, \ldots, n] \tag{18}$$

$$B(\theta, w) := \begin{pmatrix}\nabla_\theta L_t \\ -\nabla'_w L_t\end{pmatrix} = \begin{pmatrix}\rho\theta - A_t^T w \\ A\theta - b_t + C_t w\end{pmatrix}. \tag{19}$$

This recovers GTD2. Per-iteration cost is $O(d)$ but convergence is sublinear. The problem is that B_{t_m} has high variance. We reduce the gradient by SVRG

$$\begin{pmatrix}\theta \\ w\end{pmatrix} - = \begin{pmatrix}\sigma_\theta & \\ & \sigma_w\end{pmatrix} B_{t_m}(\theta, w, \widetilde{\theta}, \widetilde{w}) \tag{20}$$

$$t \sim U[1, \ldots, n] \tag{21}$$

$$B_t(\theta, w, \widetilde{\theta}, \widetilde{w}) = B_t(\theta, w) - B_t(\widetilde{\theta}, \widetilde{w}) + B(\widetilde{\theta}, \widetilde{w}). \tag{22}$$

We need to maintain $\{\widetilde{\theta}, \widetilde{w}, B(\widetilde{\theta}, \widetilde{w})\}$, update periodically. In the long run, $B_t(\theta, w, \widetilde{\theta}, \widetilde{w}) \approx B(\widetilde{\theta}, \widetilde{w})$. Per-batch is $O(d)$. onvergence is linear.

SAGA also reduces variance. Idea is similar but details are different. It doesn't need periodic batch graident updates, but has to maintain gradient for each datum (extra space $O(n)$).

Extend to off-policy PE (when data sampled from different distribution) and PE with eligibility traces (smooth the gap between MSPBE and MSE). The difference is how (A, b, C) are formed.

5.3 Complexity bounds

Summary:

1. LSTD (least squares temporal difference) $O(nd^2)$ (exact matrix inversion)
2. GTD2 $O\left(\frac{d\kappa_1}{\varepsilon}\right)$.
3. PDBG $O\left(nd\kappa_2 \ln\left(\frac{1}{\varepsilon}\right)\right)$.
4. SVRG/SAGA $O\left(nd\left(1 + \frac{\kappa_3}{\ln}\left(\frac{1}{\varepsilon}\right)\right)\right)$].

The κ_i are algorithm-specific condition numbers.

$L(w, \theta) = w^T(b - A\theta) - \frac{1}{2}\|w\|_C^2 + \frac{\rho}{2}\|\theta\|^2$. Step sizes are $(\sigma_\theta, \sigma_w$. Let $\beta = \frac{\sigma_w}{\sigma_\theta}$. Optimality conditions give

$$\Delta_{m+1} = (I - \sigma_\theta G)\Delta_m \tag{23}$$

$$\Delta_m := \begin{pmatrix} \theta_m - \theta^* \\ \beta^{-0.54}(w_m - w^*) \end{pmatrix} \tag{24}$$

$$G = \begin{pmatrix} \rho I & -\sqrt{\beta}A^T \\ \sqrt{\beta}A & \beta C \end{pmatrix}. \tag{25}$$

Eigendecomp $G = Q\Lambda Q^{-1}$ is diagable for large enough β. Get

$$\Delta_{m+1} = (I - \sigma_\theta Q\Lambda Q^{-1})\Delta_m \tag{26}$$

$$Q^{-1}\Delta_{m+1} = (I - \sigma\Lambda)Q^{-1}\Delta_m. \tag{27}$$

Capture convergence by potential function $P_m := \|Q^{-1}\Delta_m\|^2$,

$$P_{m+1} \leq \|I - \sigma_\theta\Lambda\|^2 P_m.$$

Proper σ_θ gives exponential decay of P_m. This gives linear convergence of θ_m to θ^*.

$$\|\theta_m - \theta^*\| \leq \|Q\|^2 \left\|Q^{-1}\Delta_m\right\|^2 = O(2^{-m}).$$

We have to bound eigenvalues of Q.

5.4 Experiments

We compare

1. TD (experience replay)
2. GTD (experience replay)
3. PDBG
4. SVRG/SAGA
5. LSTD

We experiment on random MDPs (400 states, 200 actions, $\gamma = 0.95$, features $d = 200$, $n = 100000$) and mountain car (2d, 3 actions, $\gamma = 0.9$).

Previous work:

- stochastic variance reduction for convex optimization,
- saddle-point optimization (Balamurugan, Bach 2016).

  Require proximal mappings that are expensive to compute in PE, and strongly convex-concave objectives (we only require strong concavity in dual).
- Gradient-based TD (Sutton et al. 2009). This is derived from very different principles and have slow convergence.
- Incremental LSTD, unknown convergence rate.

5.5 Conclusion

We give a saddle-point formulation of batch policy evaluation, first-order algorithms with linear convergence rate, and have promising experimental results on benchmarks.

Future directions

- extend to nonlinear value-function approximation
- extend to control case (policy optimization)
- application of modern optimization techniques to RL.

6 Discussion 2/13/17

EBrunskill: We want to bridge different fields. RL and active learning communities are largely disjoint but they tackle common problems.

SDasgupta: Traditional divide between learning concepts (what is a giraffe) and skills (ride bicycle). Are there math formalisms which allow us to unite the two? This is why control-related work and conceptual active-learning work have been done separately.

JZhu: Relation between optimum teaching and RL: how do you teach a RL agent? If the learner is not a simple batch classifier, but something more complex, how does that change the problem formulation and solution?

R: Minimal contrast pair, smallest change to picture to change classification cf. learning with 2 examples.

JZ: What kind of learner takes that as important teaching examples?

SD: Representation teaching. Teaching doesn't have to be limited to examples and labels. Everyone understands what those mean. Features and explanations: one party doesn't understand the other any longer.

TMitchell: Tabular rasa assumptions in theory. Zhu's talk assumed the learner was a tabular rasa learner. How to frame theoretical questions that can be studied in this messy world where there are people. Tabular rasa is one of the big question marks: can we frame theory questions that are close enough to reality?

JZ: Baby step to weaken assumption is to give uncertainty to human student.

SD: Learning theory has focused on a single concept and the learner doesn't need to know anything else. In principle we can imagine a hierarchy, DAG of concepts; the learner is somewhere along the way. Teacher should understand what learner already has. Should be quite easy to formulate math model where there are many concepts and the learner knows some.

EB: cf. education, knowledge graph.

Issue when doing one-shot, minimal learning (esp. in interactions with human) is very little data. having a whole lot of experience. Policies of behavior are great when you have experience and terrible when you don't. There is disconnect between minimizing amount of data needed and having enough of it to optimize policy.

JZ: Human is not a great teacher.

When we explain to someone else, we make commonsense assumptions. We expect that you will spread info to rest of state space where it's applicable.

JZ: Educate the human teacher.

R: "Make a PB and J sandwich" game.

EB: cf. picking up dust.

SD: We have rich spaces of concept classes, how difficult we expect the problem to be. What are categories of agents we can learn to control easily, slightly more complicated... that would give a nice handle on how much linguistic communication is needed to be able to control these agents.

TM: Think of a policy of robot, say, actions in discrete space. It is a function. In that sense, there is no difference even though it feels like there are.

EB: Policy search: often formulate in terms of VC dimensions. dimension. There may be other forms.

SBen-David: View as hierarchies of agents. Hierarchies of behavior?

EB: cf. Anca. Control community. What assumptions do we make? Stochastic, 1-step approximation. Types of approximations.

SD: What about agnostic case? In the teaching cases, it seems more reasonable to assume there is a perfect concept. We are perfect categorizers. Of course there must be a perfect categorizer for zebra, antelope, because we do it.

: Can still have model misspecification problem.

JZ: Ideal teacher knows the world is complex, but I know you're limited (ex. linear classifier), and help you get the linear separated. Meta-level teacher: I want to tell you to expand your hypothesis space, can't just use linear classifier.

List of topics

- RL +/or AL
- Learning concepts vs. learning skills
- Optimum teaching and RL
- Representation teaching
- Theoretical framing, questions for nontabular rasa setting
- Min data needed for RL vs. typical data requirement
- Categoris of agents
- Imperfect concept classification, model misspecification