Contents

Workshop website (with videos): `https://simons.berkeley.edu/workshops/schedule/3750`

1 Geometry, Optimization and Generalization in Multilayer Networks (Nathan Srebro Bartom, TTI Chicago)

Abstract: What is it that enables learning with multi-layer networks? What makes it possible to optimize the error, despite the problem being hard in the worst case? What causes the network to generalize well despite the model class having extremely high capacity? In this talk I will explore these questions through experimentation, analogy to matrix factorization (including some new results on the energy landscape and implicit regularization in matrix factorization), and study of alternate geometries and optimization approaches.

1.1 Introduction

Let $\mathcal{H}_{G(V,E),\sigma}$ be $\{f_w(x)\}$ where f_w is the output of net with graph G and activation function σ.

Three things to understand:

1. capacity/generalization ability/sample complexity: This is well understood; it is $\widetilde{O}(|E|)$ (with some assumptions).

   (There is still some gap between linear and quadratic.)

2. Expressive power/approximation:

   We can represent any continuous function. This is useless because size of network increases exponentially with dimensionality.

   We can represent lots of interesting things naturally with small networks. Have a hierarchy of features...

   Any T time computable function can be presented with a network of size $O(T \ln T)$.

3. Computation/optimization: Even if function is exactly representable with single hidden layer with $\Theta(\ln d)$ units, with no noise, even allowing a much larger network when learning, no polytime algorithm always works. (Kearns-Valiant 94, Klivans Sherstov 06, Daniely Linial Shalev-Shwartz 14)

But there has been huge success in learning neural networks. What is the magic property that makes local search work? Being representable by a small neural net is not good enough! The main mystery is optimization.

For MNIST we increase the network size(by factors of 2) and see what happens to training and test error. (Train to convergence.) Training error decreases monotonically.

Under classic understanding of learning, initially error drops, but as we increase the hypothesis class/capacity, test error should increase (overfitting). But we observe something different: the training error continues dropping significantly, and the test error doesn't increase: the network never overfits. This is consistent with what people see in much bigger networks. We routinely fit networks with much more parameters than data points with little or no regularization.

Mysteries: what gives generalization ability? How can we get to good minima?

1.2 Matrix factorization

I wanted to work on deep learning and was working on matrix factorization... But matrix factorization is deep learning! With 2 layers and linear transfer.

$$y = U(Vx) = Wx$$

where $W = UV$. r hidden units corresponds to $\text{rank}(W) \leq r$.

This has many of the complexities of deep learning. It is the most complex deep learning model we actually understand.

We know how to work with not just the rank, but norm-bounded weights. Instead of $\text{rank}(W)$ consider

$$\|W\|_{\text{tr}} = \min_{W=UV} \|U\|_{Fr} \|V\|_{Fr}$$

or other factorization norms such as weighted trace norm, max-norm $1 \to \infty$, etc.

This is a convex relaxation on the rank so is easier to optimize. There is no spurious local minima in high enough dimension even when optimizing over U, V.

It's a better hypothesis class, better inductive bias. This is a richer and more expressive model, allowing unbounded number of factors, with bounded sum of "importance" of factors, not their number.

Minimize

$$W_k = \text{argmin}_{\text{rank}(W)=k, L(W)=0} \|W\|_{\text{tr}}.$$

This is similar to how we train matrix factorization models. As k increases, $\text{rank}(W_k)$ increases. But $\|W_k\|_{\text{tr}}$ decreases with k. If norm is a better inductive bias, then W_k generalizes better as k increases.

In practice, for many tasks (Netflix),

$$W_k = \text{argmin}_{\text{rank}(W)=k} L(W) + \lambda \|W\|_{\text{tr}}.$$

The test error of W_k monotonically decreases as k increases.

I'm arguing that rank is not the "real thing".

Is improved generalization of neural nets explained by decrease in norm? No, the norm increases. But this is the $\|W\|_F = \sqrt{\sum_e w(e)^2}$ norm, which is not a good norm.

If I scale layers by λ and $\frac{1}{\lambda}$, then the norm gives different answers even if the function is the same. We want measure of complexity that is invariant to rescaling. Use the path-norm

$$\phi(W) = \sqrt{\sum_{\text{path}} \prod_{e \in \text{path}} w(e)^2}$$

(calculated by dynamic programming). The path norm decreases as the number of hidden units increases.

Train real labels vs. wrong (jump) labels: As training size increases, norm for true labels flattens out, but norm for wrong labels increases.

How does path norm compare to test error? We screwed with the optimization algorithms, which still find global minimum, but have different path norm. (There are many ways of fitting data.) As path norm increases, test error increases.

Why does gradient descent work? We minimize unregularized error to convergence. In convex models, we understand how one-pass SGD, or SGD with early stopping, provides for implicit ℓ_2 regularization, but we are getting implicit regularization without early stopping.

Optimization is biasing us towards specific global optimum. What's the bias introduced by optimization, and can we get better bias by changing optimization?

Steepest descent is with respect to Euclidean geometry. We do path-SGD, approximate steepest descent on $\phi(W)$.

Optimization is better. Both find global optimium. Path-SGD finds a global optimum that generalizes better. We changed the implicit induced bias!

This is a very overconstrained problem. Local search guides us down towards global optimum. We'll get to some beach, the closest beach. The notion of "closest" goes back to the geometry we're using to guide the local search.

Mysteries of local search (gradient descent and relatives):

1. How come local search succeeds in finding global optimum of nonconvex function?
2. How does gradient descent bias us towards low norm solutions? What norm?

Our result: local search is sufficient [Bhojanapalli Neyshabur S NIPS16]

Where is implicit bias coming from? For least squares, we understand. Gradient descent converges to least norm solution.

Optimization and inductive bias (geometry/regularizer) are linked. They affect generalization (learning).

2 A Non-generative Framework and Convex Relaxations for Unsupervised Learning (Elad Hazan, Princeton University)

We'll describe a novel theoretical framework for unsupervised learning which is not based on generative assumptions. It is comparative, and allows to avoid known computational hardness results and improper algorithms based on convex relaxations. We show how several families of unsupervised learning models, which were previously only analyzed under probabilistic assumptions and are otherwise provably intractable, can be efficiently learned in our framework by convex optimization. These includes dictionary learning and learning of algebraic manifolds.

We have much more data that is unlabeled than labeled. Ex. wikipedia text contains 6b tokens, common crawl has 840b tokens, while Stanford question answering has 100k+ Q/A pairs. Compare all images on the web vs. ImageNet which has 1.2m.

A successful example is word embeddings trained on Wikipedia corpus.

Wikipedia: Unsupervised machine learning is the machine learning task of inferring a function to describe hidden structure from "unlabeled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabeled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm.

We give a proposed definition for evaluating the solution and convex relaxations for 3 problems.

Unsupervised learning has a lot of topics without a common foundation (clustering, EM, PCA, ICA), while supervised learning has a common learning theory as foundation.

We're missing improper learning. Learning a sparse linear separator is a nonconvex, hard problem.

1. The statistical approach is by reconstruction. Under statistical and structural assumptions, learn in poly-time.

2. Forget about the constraints. Learn a dense separator by minimizing l_1 (LASSO). Get comparable sample complexity and better generalization error.

Matrix prediction/recommendation systems

1. Low rank reconstruction (Candes, Recht,...) under satistical and structural (incoherence) assumptions, polytime.

2. Nuclear norm minimization (Srebro...) in polytime, with comparable sample complexity, and no incoherent assumptions.

The second approach (convex relaxations) doesn't exist in unsupervised learning! We'll give new results via convex relaxations.

2.1 Framework

Learn h from data $x \sim D$. D is unknown, arbitrary, get iid samples.

Hypothesis $H \ni h = (f, g)$ encoder-decoder,

$$l(h, x) = |g \circ f(x) - x| + \lambda |f(x)|,$$

$err(h) = \mathbb{E}_{x \sim D}[l(h, x)]$.

Goal: for all $\varepsilon, \delta > 0$ there exists an algorithms such that after seeing $m = \text{poly}(\varepsilon, \delta, \dim(H))$ example, finds h such that w.p. $1 - \delta$,

$$err(h) \leq \min_{h^* \in \mathcal{H}} err(h^*) + \varepsilon.$$

This has been called PAC rate distortion theory + minimum description length. Learn representation of data and use to solve any future (Lipschitz) supervised task.

1. compression: explicit succinct representation. Ensures low sample complexity for future supervised task.

2. reconstruction

2.2 PCA, Spectral autoencoders

Take data in d-dimensional space. Encode in lower-dimensional space.

$$B = \text{argmin}_{\text{rank}(B) \leq k} \mathbb{E} \|Bx - x\|_2^2 .$$

Think of this as linear encoding and decoding, $f(x) = Ax, g(y) = A^+ y$. Reconstruction optimization is

$$\min_{A \in \mathbb{R}^{k \times n}} \mathbb{E} |A^+ Ax - x|_2^2.$$

This is properly learnable with sample complexity $\frac{k}{\varepsilon^2}$.

Why don't we try to come up with more sophisticated encodings?

What if your data lies on some manifold, $Cx^{\otimes r} = 0$?

Encoding is $y = Ax^{\otimes 2}$ and decoding is $v_{\max}(A^+ y) \approx v_{\max}(A^+ Ax^{\otimes 2})$. [1]

We can write an optimization problem

$$\text{argmin}_{A \in \mathbb{R}^{k \times n^r}} |v_{\max}(A^+ Ax) - x|_2^2.$$

($r = 2$ above.) This class is improperly learnable. Use surrogate loss based on convex relaxation,

$$\text{argmin}_{R \in \mathbb{R}^{d^2 \times d^2}, |R|_* \leq k} \mathbb{E} |Rx^{\otimes 2} - x^{\otimes 2}|_{op}^2.$$

Intended solution is $R = A^+ A$. Optimization by non-smooth projection-free algorithm returns low rank solution R; factorization gives A.

[1] This is related to kernel PCA, which requires reconstruction of $x^{\otimes 2}$.

We need to show that sample complexity is comparable, and doesn't blow up.

Get: SA learnable with encoding length $\frac{k^4}{\varepsilon^4}$ for error ε.

(In supervised learning, good predictions means good generalization. What is the notion of prediction here? Reconstruction error. You can learn anything you could have learned before, so you do not lose in terms of predictive power.)

In PCA, what happens if your data does not lie in a low-dimensional subspace? It still works—you only compete with the best?

Is this the right notion of unsupervised learning? Fits many schemes that have been used. It doesn't capture everything; it's not clear it captures structure; but it does allow low sample complexity on future tasks.

We're trying to use techniques from supervised to unsupervised learning.
[2]

Reconstruction error is better for SA. Classification error improves in SA (with SVM on top) vs. KPCA, PCA.

2.3 Dictionary learning, linear representations

Dictionary learning: DL is improperly learnable discriminatively.

Word embeddings. We give convex relaxation for linear representations. Consider a distribution over tasks $D \sim \mu$. Each task is a distribution over same domain $(x, y) \sim D$.

Goal: learn represenation $f(x)$ such that classification over new representation is as good as it is on training tasks.

Theorem: there is efficient improper algorithm. By convex optimization, it is solvable efficiently.

3 Representations for language: from word embeddings to sentence meanings (Christopher Manning)

A basic—yet very successful—tool for modeling human language has been a new generation of distributed word representations: neural word embeddings. However, beyond just word meanings, we need to understand the meanings of larger pieces of text and the relationships between pieces of text, like questions and answers. Two requirements for that are good ways to understand the structure of human language utterances and ways to compose their meanings. Deep learning methods can help for both tasks. I will then look at methods for understanding the relationships between pieces of text, for tasks such as natural language inference, question answering, and machine translation. A key, still open, question raised by recent deep learning work for NLP is to what extent do we need explicit language and knowledge representations versus everything being latent in distributed representations. Put most controversially, that is the question of whether a bidirectional LSTM with attention is the answer for all language processing needs.

[2] LB: Two kinds of unsupervised learning. Normal: know how to do—compress information. Mythical: like humans, look at world and understand how it works. The difference between the two is not that clear.

3.1 What's special about human language?

1. Distinguishing human characteristic. (Other species have good vision systems, etc.)
2. The only hope for "explainable" intelligence. Pictures of neurons spiking does not explain what an AI is doing.
3. Communication was central to human development and dominance. This ability allowed teamwork, and allowed humans to be dominant without being particularly strong. When people developed writing, they could send ideas across time and space.
4. Language forms come with meanings. One word or phrase has meaning attached to it. How to make sense of it has been debated for many years.
5. Language is a social system. Computer scientists want a platonic view of language, but that isn't how language is; it's a social system that changes over times.
6. Constructed to convey speaker/writer's meaning—not just an environmental signal but a deliberate communication.
7. Uses encoding which little kids learn amazingly quickly.
8. Language is a discrete/symbolic categorical signaling system. There are very minor exceptions for expressive signaling (I looooove it). This is presumably because of greater signaling reliability. (cf. EE reasons for having discrete signaling systems.)

   Symbols are not just an invention of logic/classical AI!
9. Language symbols are encoded as a continuous communication signal in sound, gesture, and writing. Symbol is invariant across different encoding. You can convey the same symbols and meaning across the different encodings.

Traditionally, people have extended the symbolic system of language into the brain, "the language of thought." But a brain encoding appears to be a continuous pattern of activation, just like the signal to transmit language. Deep learning is exploring a continuous encoding of thought.

3.2 From symbolic to distributed word representations

Vast majority of rule-based and statistical NLP/IR (info retrieval) work regarded words as atomic symbols—a vector with one 1 and 0's everywhere else, a one-hot representation. These representations don't work very well.

Ex. if user searches for "Dell notebook battery size", we want to match "Dell laptop battery capacity." There's no notion of similarity.

There are many things you can do: query expansion with synonum dictionaries, separately learning word similarities...

A word representation that encodes similarity wins: less parameters to learn, more sharing of statistics.

"You shall know a word by the company it keeps." J.R. Firth 1957. This is one of the most successful ideas of modern NLP.

Define a model that predicts between a center word w_t and context words in terms of word vectors, $\mathbb{P}(\text{context}|w_t)$ which has loss function $J = 1 - \mathbb{P}(w_{-t}|w_t)$. Word2vec: Use skipgram model

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{v} \exp(u_w^T v_c)}.$$

Using this algorithm, a miracle occurs. If you repeat over backprop a million times, you get wonderful word representations that do a good job of capturing fine-grained directions and meaning.

Another approach is latent semantic analysis: singular value decomposition of (log) word-context matrix. $A = U\Sigma V^T$. Keep the largest orthogonal directions.

The miraculous property of having dimensions of meaning: linear vector differences have meaning. Ex. linear relationship between countries and capitals.

A decade earliers, psycho-linguistics: you can also take LSA model, with just-right scaling of word occurrences, you can also get linear relationships (Rohde, Gonnerman, Plaut, 2005).

What is the relationship between these 2 approaches? What if you have a word-count based model? Crucial insight: ratios of co-occurrence probabilities encode meaning components. Ex. look at $\frac{\mathbb{P}(x|\text{ice})}{\mathbb{P}(x|\text{steam})}$ for gas/solid.

How to capture ratios of co-occurrence probabilities as meaning components in word vector space?

$$w_x \cdot (w_a - w_b) = \ln \frac{\mathbb{P}(x|a)}{\mathbb{P}(x|b)}.$$

Ex. closeby words to "frog." Gender pairs, company-CEO, etc.

Using word vectors in other algorithms can make performance go up a few percent: good notion of word similarity generalizes better. But not everything can be represented with a linear representation.

SVD hurts, while modern neural representations do better from categorical CRF baseline.

GloVe model is a model that works on counts. It connects Count! and Predict! models.

3.3 BiLSTM hegemony

To a first approximation, the consensus in NLP in 2017 is: No matter what the task, throw a BiLSTM at it, with attention if you need information flow.

A RNN encoder-decoder network is

$$h_t = \tanh(W[x_t] + Hh_{t-1} + b).$$

This didn't work, but you can this picture, if you change the computation in the box to be a little more complicated (gated recurrent unit, LSTM). You can invent many variants. Essentially,

1. there is a piece that calculates candidate next hidden state representation,
2. Bernoulli variable "gates" control how much history is kept and how much input is attended to.

3. Summing previous and new candidate hidden states gives direct gradient flow and more effective memory.

Note: recurrent state mixes control and memory. This could be good (freedom to represent however it wants) or bad (get mush of things that are conceptually different).

We changed the units and make the net deeper. With some care this works well as a machine translation system. It has one big limitation. There is a bottleneck between the source to output sentence: the last state (vector) after reading the input. Also the model takes a long time to train—it needs to know what to remember.

1. Add attention: vector will be used to look at source time steps, with some weighting use these time steps to influence what's computed at the next time step.

2. Have another LSTM running backwards. Get a hidden state vector from both directions; concatenate.

[3]

Neural machine translation is now much better than the other translation algorithms. It's not just better at dealing with word similarities; it's better at syntax of things like getting the word order right.

Four big wins of neural MT:

1. End-to-end training: all parameters are simultaneously optimized to minimize a loss function on the network's input.

2. Distributed representations shared strength. Better exploitation of word and phrase similarities.

3. Better exploitation of context to translate more accurately.

4. More fluent text generation: deep learning text generation is much higher quality.

BiLSTMs and attention are not just for neural MT: part of speech tagging, named entity recognition, syntactic parsing (constituency and dependency), reading comprehension, question answering, text summarization.

CNN/Daily Mail: figure out missing word in tagline after reading article. Run BiLSTM on story. Use attention function: using question, for each position in passage, figure out how much to attend to it. Get $a_i = \text{softmax}_i(q^T W s\widetilde{p}_i)$.

This works incredibly well.

Recipe for getting work in NIPS: take a model, add epicycles and curlicues, fiddle until numbers go up, claim that the epicycles and curlicues made the numbers go up. If you can build a simpler model that does somewhere between 97% and 103% as well as the previous model, that should be good!

It's appalling that this model works so well. Contrast this from history of NLP.

Standard theory of natural language interpretation:

[3] English-French is easy. English-German is harder because German has different word order, finite verb auxiliary, real verb at end. LSTM are good at learning syntactic reordering better than any model we used use use.

1. from language expressions,
2. create syntax trees (deletion goes back to language expression),
3. do semantic interpretation to get logical formulas,
4. use semantic denotation to get models.

We'd like to get some of that into our language models, semantic interpretation of language.

How can we minimally know when larger language units are similar in meaning?

- Snowboarder is leaping over a mogul.
- A person on a snowboard jumps into the air.

It seems you have to go through the old NLP pipeline...

3.4 Choices for multi-word language representations

From bag of words to complex, esoteric representations...

There are 4 tools in your toolbox.

1. neural bag-of-words models: average/sum word vectors.

   You can improve effectiveness by putting output through 1+ fully connected layers (deep averaging networks).

   Doing this and nothing more does well for many tasks.

   For tasks more focused on semantic similarity, this works better than LSTM's, who are devoting too much attention to modeling language.

2. recurrent neural networks: use word order but cannot capture phrases without prefix context. Gated LSTM/GRU could separate meaning of constituent units, but it seems unlikely.

   They capture too much of last workds.

3. Convolutional neural network: compute vectors for every h-word phrase? This is not very linguistic, but you get everything.

   Ex. 1-d convolution neural network with max pooling and FC layer (Kim2014 CNN for sentence classification). Can use multile filter weights and multiple window sizes.

4. Data-dependent composition. You have to put parts together to get bigger parts.

   Visual scenes have parts too that go together into bigger parts.

   Noam Chomsky: Distinctive attribute in human languages is that they have recursion.

   Build tree-recursive neural networks. Look at particular sentence, build constituents.

   TreeRNN can capture constructions like "X but Y."

3.5 Conclusion

- Deep learning, distributed representations, end-to-end training, richer modeling of state has brought great gains to NLP.
- At the moment, it seems like we can't win, or only barely win by having more structure than a vector space mush, and converges to 0 as data goes large.
- I believe we need more structure and modularity for language, memory, knowledge, and planning; it'll just take some time.

  Trees seem like the minimal bit of extra structure. It'll take time for these models to be empirically successful.

Q: adversarial perturbation to input? A: Various rearrangings. It's a game for the general public. People publish the freaky things that happen with a NN. "Women women..." "Femme femme" "Woman woman man man..." This wouldn't have gone wrong in a last-century system! Although NN does much better overall, it has some interesting failure cases.

Q: What's something you need a LSTM for? A: There's no function LSTM can compute that you can't also compute by suitable basic RNN. In practice these models learn hugely better. For simple RNN, having the emphasized diagonal is a contingent fact that can easily die out; for these it's built into the architecture.

Q: written vs. spoken language? A: NN models win everywhere but more when language is more informal and less well-structured.

4 Looking for the Missing Signal (Leon Bottou, Facebook AI Research)

We know how to spot object in images, but we have to learn on more images than a human can see in a lifetime. We know how to translate text (somehow), but we have to learn it on more text than a human can read in a lifetime. We know how to learn playing Atari games, but we have to learn it by playing more games than any teenager can endure. The list is long.

We can of course try to pin this inefficiently to some properties of our algorithms. However we can also take the point of view that there is possibly a lot of signal in natural data that we simply do not exploit. I will report on two works in this direction. The first one establishes that something as simple as a collection of static images contains non trivial information about the causal relations between the objects they represent. The second one shows how an attempt to discover structure in observational data led to a clear improvement of Generative Adversarial Networks.

Motivation:

Don't believe the newspapers: Machine learning is not a success. We have to train on more images than a human can see, more text than a human can read, more games than a teenager can endure, etc.

What are we doing wrong? Are our learning algorithms so inefficient? Absence of prior? (Cramer-Rao) Does transfer learning help?

We take another viewpoint: is there more signal in data than we think? Look beyond correlations: at causation, complex moments. Typical supervised machine learning systems use $\mathbb{E}[\phi(x)]$, $\mathbb{E}[y]$, $\mathbb{E}[y\phi(x)]$, $\mathbb{E}[\phi(x,y)]$.

The parse tree is good not because it compresses, but because it allows us to manipulate the structure. It's not about the sentences you see, but all the sentences you can construction.

4.1 Causation and moments

Problem with causation is confounding. Suppose we have binary random variables, where Z has positive effect on X, Y, and X has slight negative effect on Z. If we only observe X, Y we see X and Y are positively correlated. This is the Simpson effect. Odd scatterplots hint at causation.

Ex. Hertzsprung-Russell diagram.

Reichenbach's principle. Either

1. A causes B
2. B causes A
3. A and B have a common cause C or
4. A and B are independent

Ex. additive noise $Y = \alpha X + \beta + (\text{noise})$.

Sharp corners (high moments) hint at causation. It's not apparent in means or covariances.

We featurize a scatterplot to infer causation. Use a neural causation classifier.

We don't have access to large causal direction datasets, but we can generate artificial scatterplots.

Generate artificial scatterplots $Y = f(X) + v(X)\varepsilon$. Take $k \sim U_{\{1,\dots,5\}}$, $r, s \sim U_{[0,5]}$. Draw f: cubic spline with random number of random knots. Draw noise, Gaussian with random variance $U_{[0,5]}$, cubic spline with random knots. Draw x_j, ε_j, and then compute $y_j = f(x_j) + v(x_j)\varepsilon_j$. Rescale. Generate training examples: $(\{(x_i, y_i)\}, 1)$, $(\{(y_i, x_i)\}, 0)$.

After training on artificial data, NCC achieves state-of-the-art performance on Tübingen cause-effect dataset, `https://webdav.tuebingen.mpg.de/cause-effect/` (79%).

This also works for detecting confounding variables. How to validate this? 2-dimensional scatterplots are limited...

Take something without a causal signal.

4.2 Finding causal signal in images

Ex. if you remove bridge from a picture, you also have to remove the cars.

All images are preprocessed using state-of-the-art pretrained CNN. Feature scores are often interpretable as features of the scene.

For each object categories we can define 2 sets of features.

1. Causal features: cause presence of object of interest.

2. Anticausal feature: caused by presence of object of interest.

Ex. if there is a causal relationship car→wheel, then we hope it's also visible in the scores.

Apply NCC to feature scores and object scores to identify top 1% causal and anticausal features for each of 20 categories.

What is causality, concretely? If I remove something, what else do I need to remove to get something consistent? If I remove the car, I need to remove the wheels. If I remove the car, I can keep the lamppost.

In computer vision, one is often interested in another way of splitting features:

1. object features: belong to object, most often activated inside bounding box (wheels, eyes)

2. context features: activated outside bounding box: road under car, shadow.

"Bags of visual words."

To empirically identify, black out the context, vs. black out the object.

Hypothesis: there exists observable statistical dependence between object and anticausal features. Statistical dependence between context fatures and causal features is nonexistent or weaker.

This implies: Image datasets carry observable statistical signal revealing asymmptric relationship between object categories that result from causal dispositions.

Top anticausal features have higher object scores (whether it is a feature or not, from knowing bounding boxes) for all 20 categories. There is no clear relation between top causal and context scores.

Effect disappears if we replace NCC by correlation coefficient between feature and category.

We've indirectly shown that high-order statistics can inform causation in the scenes. To our knowledge, no prior work has established or considered presence of such a signal.

4.3 On uses of Wasserstein(ish) distance

The notion of cliff: ex. a uniform distribution stops, and translates into a sharp edge in the scatterplot.

"Mythical unsupervised learning." Not about using unlabeled data to discover probability ratios, but using unlabeled data to discover (causal) generating mechanism.

Causal footprints include corners, cliffs, shocks, low-dimensional causl models.

Low-dimensional causal models: there is no density because it is supported on low-dimensional manifold. One way is to add noise.

$X \sim P_r$ is observed data. From $Z \sim P_z$, typically low-dimensional, transform by $G_\theta(Z)$ to get P_g. This has low-dimensional support, a cliff-shaped density.

1. TV distance.

2. KL divergence requires density.

3. JS divergence is symmetric and does not require density. Lose 1 bit of information by considering the mixture.

4. Earth-mover (EM) or Wasserstein-1 distance,

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|].$$

In generative adversarial network, the discriminator D_ϕ tries to say whether the sample comes from the real or generated distribution, $X \sim P_r$ or $G_\theta(z) \sim P_g$. Train D_ϕ to differentiate as well as possible, and generator to make this as hard as possible. Discriminator maximizes and generator minimizes

$$L(\phi, \theta) = \mathop{\mathbb{E}}_{x\sim\mathbb{P}_r} \ln D_\phi(x) + \mathop{\mathbb{E}}_{z\sim p_Z}[\ln(1 - D_\phi(g_\theta(z)))].$$

If you keep discriminator optimal, $\min_\theta L(\phi^*(\theta), \theta)$ minimizes $JS(P_r, P_g)$. Keeping generator optimal and maximizing $\max_\phi L(\phi, \theta^*(\phi))$ yields garbage. If you train discriminator too much, there is no gradient.

Alternative update is $\Delta\theta \propto \mathbb{E}_{z\sim p_Z}[\nabla_\theta \ln(D_\phi(g_\theta(z)))]$, which optimizes $KL(\mathbb{P}_\theta||\mathbb{P}_r) - 2JSD(\mathbb{P}_r||\mathbb{P}_\theta)$.

Consider distributions with low-dimensional support. Ex. uniform distributions supported on parallel line segments separated by θ. W is continuous in θ; the other metrics are not.

To optimize, use the dual formulation (Kantorovich duality),

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \max_{\|f\|_L \le 1} \mathop{\mathbb{E}}_{x\sim\mathbb{P}_r}[f(x)] - \mathop{\mathbb{E}}_{x\sim\mathbb{P}_\theta}[f(x)].$$

Parametrize $f(x)$, e.g. with neural networks. Enforce Lipschitz constraint for instance by aggressively clipping weights. Maintain $f(x)$ well-trained, and train $G_\theta(z)$ by back-prop through $f(x)$. There are no vanishing gradients.

Figure: For a GAN discriminator if the distribution the gradient vanishes; for WGAN, the gradient does not vanish.

We have $\nabla_\theta W(\mathbb{P}_r, \mathbb{P}_\theta) = -\mathbb{E}_{z\sim p(z)}[\nabla_\theta f(g_\theta(z))]$. If you manage to generalize on f, you get the gradient. [4]

I'm not sure there is a real manifold of data. I want manifold of generated data to be low.

WGAN loss correlates with sample quality: when Wasserstein distance goes down, quality goes up.

With a normal GAN, loss increases but image gets better.

WGAN is less sensitive to modeling choices. Without batch normalization WGAN still works, but GAN doesn't.

4.4 Conclusion

In search for lost signal:

[4] Wasserstein is defined on empirical distributions. Why not minimize on empirical distributions? Wasserstein between continuous distribution and discrete approximation has bad asymptotic properties.

- Causal signal in high moments
- Takes form of cliffs, corners, shocks, etc.
- Mythical unsupervised learning
- Weak distribution distances such as Wasserstein seem more able to catch it.
- This is the beginning.

MNIST digits: create 3-digit number. How many of 1000 numbers do you get? Normal GAN completely fails, it does lots of mode-dropping, gets < 15. Reverse KL, JSD has wrong sign. High price for incorrect, low price for learning something that isn't there.

How much is your model memorizing the data?

BR: conditional GAN's/regression: rather than make Gaussian noise into images, make images look like other images.

You can interpolate between bedrooms.

SBD: There's a huge gap between theory and practice. How can I generate something that looks like real life from few bits? Why would a person think this is a bedroom?

GAN's are a beauty contest now... It's not satisfying.

RS: Want: Given new bedroom scene, say it has high probability.

Noise modeling is bane of generative models.

Can you make generative model roughly invertible to get probabilities? I don't know what to make of the results. Until it's done in a way that's robust...

5 Representations of relationships in text and images (Hal Daumé III, University of Maryland)

I'll describe recent work on modeling complex relationships in a neural setting, based primarily on a combination of topic-model style dictionary learning (for interpretability) and recurrent neural networks (to capture the flow of time). This all ties in to the question of how to learn common-sense knowledge; for this, I'll talk first about understanding how relationships between humans evolve, learned from text alone; and then how this can be extended to multimodal (image and text) settings. Joint with many people, especially Snigdha Chaturvedi, Mohit Iyyer and Jordan Boyd-Graber.

Fundamental goal is deep understanding of text. "Arthur shall strike the blow that sets Lucy free."

BiLSTM hegemony does named-entity recognition, syntactic parsing, machine translation, factoid question answering (Bram Stoker wrote this (Dracula)).

Complex question answering: what does Lucy need to be freed from?

I'm not going to talk about learning through interaction (see webpage) or bag of n-gram and logistic regression. Everytimes there's a LSTM result, I try this as my go-to representation; often it does almost as well.

5.1 Relationships in narratives

I'll talk about a long project of trying to understand the nature of narratives, fictitious and real (newspapers, history texts). We'll try to do this in an unsupervised way. What is an objective for unsupervised learning? I want to produce something people can understand in a meaningful way. You're trying to take a huge amount of data and whittle it down to something a human can understand. (This is hard to quantify.)

I'll talk about relationships. If a digital assistant doesn't understand how humans interact, it'll be unnatural.

There's a reasonable amount of work on understanding narratives from the role that characters play. A lot of narratives are about relationships between multiple characters.

Ex. Calvin and Hobbes: not clear who's hero or antihero, it's just a story about how their relationships evolve over time.

1. Relationships are not static.
2. We don't know what kind of relationships exist in the world.

Ex. Tom Sawyer.

- Tom falls in love with Becky.
- Romance collapses.
- They become friends again.

At first we looked at it as binary classification—friends or enemies. More interesting than the progression of relationship over time is that you have effect: friend of friend is probably a friend; friend of enemy is probably enemy. Ex. Saddam Hussein is not friends with everyone else, and everyone else is friends.

Real-world relationships have multiple facets. Going back to Dracula,

- Arthur and Lucy get married. Marriage, joy, love.
- Lucy feels bad because she got bitten by a vampire. Sickness, sadness.
- Lucy dies. Relationship devolved.
- But she's not actually dead. Arthur has to kill Lucy. This is predominately murder, not so much murder going on.

Why should humanities scholars care? Distant reading (Moretti 2005) can help rapidly collect examples of specific relationship types. Can we detect positive or negative subtexts underrlying meals between 2 characters? (Dostoyevsky) It's highly indicative of what happens later! We can try to automate this.

We want to look at subtext, secondary things that are going on.

We have a dataset of character interactions from Project Gutenberg. Find spans of text with both characters. Assume books proceed linearly in time.

Model as dictionary learning over time. We want results we can show to people and have them understand.

Each span is probability distribution over states (violence, sadness, politics, love, suffering, etc.).

Relationship modeling network is recurrent autoencoder with dictionary learning.

1. Start with span of text mentioning 2 characters. Take word representations, and average them.
2. Mix with embedding for characters and books.
3. A recurrent connection copies some of the previous hidden state.
4. Use softmax to make hidden state a probability distribution over states. States are probability distribution over topics.
5. Multiply hidden state by dictionary matrix to obtain reconstruction of span vector
6. Make reconstructed vector close to the input span vector. (Minimize square error.)

Two things are going on: things about relationship (marriage, love), about one character (death). We tried enforcing sparsity: softmax and threshold, ReLU and renormalize, but it didn't help. Vectors did end up reasonably peaked. Dimension of word embeddings is 300. d_t (number of topics) is 10, 30, 50.

Baseline is hidden topic markov models (HTMM). Relationship modeling network:

- sadness: regretful rueful pity pained despondent.

HTMM picks out concrete things. Relationship modeling picks out more abstract things.

Evaluation: Word intrusion task. Show turker the words and a word that doesn't belong. If turker picks out the word then the topics are good. Precision is 0.7 to 0.8. Random guessing is $\frac{1}{6}$. The two variants of our model are markedly better than standard topic models and topic models over time.

You have to be careful about word frequency. Pick a random intruder about as frequent as other words in the topic.

Where in the model was there anything about relationships? If the relationship is important, affecting words appearing, needs to capture relationships to accurately capture. We're encouraging it by giving the characters it's looking at. (Look at neighborhood of characters.) There's no explicit cross product term.

(Remove recurrent connection does worse. There's 2 things that help: neural and temporal.)

We tried to evaluate. This is super-difficult. We pick the most likely topic at each moment in time and graph. Give people plot summaries and ask people which graph is better.

People prefer our things to the baseline things by 70%-30%. Reading is super-difficult.

We name the topics by looking at the list and picking the name.

We are moving in the direction of interpretability. The main point: We care a lot about showing the result to people and making sure they can make sense of it. How do people use this output in a useful way, whether it's a humanities task or something else.

5.2 Comics

We care about commonsense inferences. "Lex Luthor's limousine came late today. The driver had overslept."

Get from dataset of comic books. Comic books are well-studied in humanities literature.

"What is exploding, and why?" The roadblock. This is bad, a ghost will come out. It's not stated explicitly the relationship between panels. Closure is the process by which we connect panels together. Cf. jump cuts in movies.

We have 4000 comic books from digital comics museum. Extract 1.2 million panels with 2.5 million textboxes. NN do extraction and OCR. It's a fun dataset!

Cloze tasks for testing closure. "The Simons workshop has been interesting so far. I hope the rest of the week will be as (compelling)."

Predict dialogue in a panel given previous panels as context.

Use BiLSTMs. Take text panels (thought bubbles, dialogue boxes, OCR'd), run through LSTM, run panel through CNN. If no text box, there's no LSTM connection. Predict correct text.

There are image and text datasets: visual question answering. One frustrating thing is that you can ignore the image and do well with the text. "Is there a steam engine in the image?" If no, no one would have written the question, there are these biases!

Image-only is 48.7% accuracy, Text-only is 51.9% accuracy, image-text with no/full context is 59.6%, 63.4%. Human accuracy is 84%.

How to generate confused text? Hard configuration: pick distractors from adjacent pages from book. If you pick text from different comic books it can be pretty obvious (different styles of speaking). In the easy setting people are in the high 90's.

COMICS: new dataset and tasks, deep learning can learn some degree of closure, our best models lag behind humans despite lots of data.

5.3 Inductive bias in networks or data

As a NLP researcher, my job is to get inductive bias into ML systems. I don't know how to put connections in NNs for inductive biase.

I want to do machine translation as you're speaking rather than wait until the end. Get things in an order where it's easier to be monotone.

Reorder the output to be more like order of input.

Parse English side, rewrite in lots of ways and generate lots of training data.

Evaluate in 2 ways: how simultaneous in input (how much delay between word spoken and translation output) and quality. Evaluate against rewritten data. Model using rewritten data does better. Extra data is not perfect but makes better. I as language person can cast inductive bias and transform output to enforce invariants.

Implications for humanities: the most solid thing is a paper under submission with an English professor (Krause) looking at the question: in Dickens type novels, how does the relationship between characters map to socioeconomic status (rentor, rentee).

6 A formal framework for representation learning (Andrej Risteski, Princeton University)

We will introduce a formal framework for thinking about representation learning, endeavoring to capture its power in settings like semi-supervised and transfer learning. The framework involves modeling how the data was generated, and thus is related to previous Bayesian notions with some new twists.

In simple settings where it is possible to learn representations from unlabeled data, we show:

1. it can greatly reduce the need for labeled data (semisupervised learning)
2. it allows solving classification tasks when previous notions such as nearest neighbors or manifold learning either don't work or require too much data.

We also clarify two important settings—linear mixture models and loglinear models—where representation learning can be done under plausible assumptions (despite being NP-hard in the worst case).

Joint work with Sanjeev Arora.

I will

1. introduce formal framework
2. clarify in two settings: mixture and log-linear models
3. illustrate power of representation learning vs. nearest neighbor classifiers and reducing labeled data.

Why learn representations? What do we want out of our model? Our approach is motivated by variety of uses and successes.

- Unsupervised learning: discover structure
- Transfer learning: features for one task helps for others
- Semi-supervised learning: help do classification with smaller number of labeled samples.

I'll focus on the last. Our model is inspired by Bayesian considerations.

6.1 Formal model

Think of representation as high-level succinct representation of object. Map is intuitively many-to-one. Our proposal:

$$x = g(h, r)$$

where x is raw data, h is representation, and r is random seed. The distribution of $g(h, \cdot)$ is raw data manifestions of some representation.

We want benefit from passing to the representation: classification is easier in representation space. There is a classifer from representation space with low error.

Since $x = g(h, r)$, learning a representation is "inverting" g.

Definition 6.1: A function f is (β, γ)-valid encoder if wp $\geq \beta$,

$$\|f(x) - h\| \leq (1 - \gamma) \|h\|$$

So for α-Lipschitz classifier C, wp $\geq \beta$,

$$\|C(f(x)) - C(h)\|_\infty \leq (1 - \gamma)\alpha \|h\| .$$

This notion depends on what you're trying to classify. This is tied to a task.

Comparison with related notions:

- manifold learning: assume points lie on low-dim manifold. Sample complexity is exponential in dimension of manifold.
- kernel learning: We exhibit encoder examples involving thresholding; it is unclear if they correspond to a kernel.

Point: our notion is different.

Relationship to Bayesian modeling. From Bayesian perspective, we have posterior distribution over h.

Theorem 6.2. *If $(1 - \delta^2, \gamma)$-encoder exists, wp $1 - \delta$ over choice of x, posterior $h|x$ concentrates close to f*

$$\mathbb{P}_{h|x}[\|f(x) - h\| \leq (1 - \gamma) \|h\|] \geq 1 - \delta.$$

We can exhibit simple settings involving mixture models where

- encoders are efficiently learnable.
- encoders help for semi-supervised learning: nearest neighbors in representation space require few samples.

6.2 Recommendation systems

See many users and subset of movies they like. There isn't a numerical values. Then ask whether the user likes another movie.

We need to impose latent structure. Make a linear structure (mixture model). There are a bunch of genres, subsets of movies. A user is mixture of genres.

$$\mathbb{P}(\text{movie } i \text{ is emitted}) \propto (Ah)_i$$

where A is genre matrix and h is genre proportions. Encoder maps liked movies to latent genres h, and semi-supervised learning learns encoder first and then does classification knowing h.

Theorem 6.3 (Corollary of AGKM16). *If A is l_1 well-conditioned ($\|A(h_1 - h_2)\|_1 \gtrsim \|h_1 - h_2\|_\infty$), then there is good encoder learnable in time* poly (*movies*) *even if number of emitted movies is logarithmic in total number of movies.*

Classification tasks are the ones that use h.

6.3 Power from representations

We get power from these representations. Quantify by 2 phenomena:

1. Nearest neighbors: if $T \ll \sqrt{m}$, NN doesn't reflect latent genres. If $T \gg \sqrt{m}$ with natural genre assumptions NN works.

2. Semi-supervised learning: for constant number of genres, label $l(h) = \text{sign}(\langle w, 2h - \mathbb{1}\rangle)$,...

Similar results for log-linear models.

7 Semi-Random Units for Learning Neural Networks with Guarantees (Bo Xie, Georgia Institute of Technology)

Training neural networks is a difficult non-convex optimization problem with possibly numerous local optimal and saddle points. However, empirical evidence seems to suggest the effectiveness of simple gradient-based algorithms. In this work, we analyze the properties of stationary points for training one-hidden layer neural networks with ReLU activation functions and show that a stationary point implies a global optimum with high probability under some conditions on the neural weights. Moreover, we introduce semi-random units where the activation pattern is determined by a random projection of the input, and show that networks with these units are guaranteed to converge to global optimal with high probability.

Joint work with Yingyu Liang, Kenji Kawaguchi, Le Song.

Neural networks are successful in learning many nonlinear functions. Most are trained with simple SGD. This is a highly nonconvex objective function. Why does SGD work so well?

I'll give a first attempt to answer this question.

We look at 1 hidden layer NN with ReLU activation function.

$$f(x) = \sum_{k=1}^{n} v_k \sigma(w_k^T x).$$

Least-squares loss

$$L(f) = \frac{1}{2m} \sum_{l=1}^{m} (y_l - f(x_l))^2.$$

With some assumption on weights, any local min is global optimum.

Gradient wrt first layer weights is

$$((v_i\sigma'(w_i^T x_j)x_j)\cdot\frac{1}{m}(f(x_i)-y_i),$$

$\frac{\partial L}{\partial W} = Dr$. Key inequality is

$$\|r\| \le \frac{1}{s_m(D)}\left\|\frac{\partial L}{\partial W}\right\|.$$

where r is training error, s_m is min singular value. Directly analyze singular value of $G_n = \frac{D^T D}{n}$. This is a function of weights so difficult to analyze. Introduce intermediate variable that has uniform weights.

The intermediate variable is no longer a function of actual weights.

Decompose into 2 parts

$$\lambda_m(G_n) \ge \lambda_m(G) - \|G - G_n\|.$$

1. Spherical harmonic decomposition:

$$G_{ij} = \left(\frac{1}{2} - \frac{\cos^{-1}\langle x_i, x_j\rangle}{2\pi}\right)\langle x_i, x_j\rangle = \sum_{u=1}^{\infty}\gamma_u\phi_u(x_i)\phi_u(x_j).$$

   Whp, $\lambda_m(G) \ge m\gamma_m/2$.

2. $\|G - G_n\| \le O(\rho(L_2(W)))$ where

$$L_2(W)^2 = \frac{1}{n^2}\sum_{i,j=1}^{n} k(w_i, w_j)^2 - \mathop{\mathbb{E}}_{u,v}[k(u,v)^2].$$

where $k(x,y) = \frac{1}{2} - \frac{\cos^{-1}\langle x,y\rangle}{2\pi}$. Whp $s_m(D)^2 \ge nm\gamma_m/2 - cn\rho(L_2(W))$.

Simplified result: n, d large enough and weight discrepancy is small. Then whp

$$s_m(D)^2 \ge \Omega(m).$$

($n = \widetilde{\Omega}(1/\gamma_m)$, $d = \widetilde{\Omega}(1/\gamma_m)$, $L_2(W) = \widetilde{O}(n^{-\frac{1}{4}}d^{-\frac{1}{4}})$.)

"Certificate of global optimality."

Final error: for n, d large enough, W small weight discrepancy,

$$\frac{1}{2m}\sum_{l=1}^{m}(f(x_l)-y_l)^2 \le O\left(\left\|\frac{\partial L}{\partial W}\right\|^2\right).$$

(n and d are between $O(\sqrt{m})$ and $O(m)$. Most W have weight discrepancy small enough.)

Will output of gradient algorithm have small discrepancy?

Technical difficulty on ensuring small weight discrepancy. We describe semi-random unit to overcome this. Think of ReLU as having nonlinear part and linear part. We decouple and replace nonlinear part by random projection

$$\sigma(w^T x) = \mathbb{1}[r^T x > 0]w^T x$$

where r is random. For any hidden unit, I have one r. It's fixed for the unit at the beginning. Whp this will satisfy low weight discrepancy condition.

Semi-random units

- sits between fully-random features and fully-adjustable units.
- linear in parameters but nonlinear in input.
- guaranteed to converge to global optimum whp
- has universal approximation ability.

Experiment results: not as expressive as ReLU, but can increase number of units to get ReLU performance.

Width vs. depth: depth helps more.

Semirandom 4x matches performance of ReLU.

8 Intrinsic motivation and automatic curricula via asymmetric self-play (Rob Fergus, NYU)

RL typically requires a huge number of episodes, and often requires a supervision signal (reward) that is expensive to obtain.

Can we learn about environment in an unsupervised way? The assumption is that interaction with environment is cheap.

Agent plays a game where it challenges itself. A single physical agent has 2 separate minds: Alice proposes a task (by doing it), Bob completes the task.

Consider 2 classes of environment:

1. Actions are reversible within same time (reverse self-play)
2. Reset to initial state is allowed (repeat self-play)

Jointly train with self-play and target task.

For reverse self-play: take some actions; at some time take "STOP" action handing control to Bob. Bob's goal is to take reverse action.

For repeat self-play: take some actions, issue STOP. Reset and Bob tries to repeat.

Incentivize exploration (internal) through reward structure. Bob's reward is $R_b = -t_b$, time spent, with $t_{\max}$ if fail. Alice's reward is $R_a = \max(0, t_b - t_a)$. Make Bobh fail with less effort.

Alice's optimal behavior is to find simplest tasks that Bob cannot complete. This makes learning for Bob easy since the new task will be only just beyond his current capabilities.

Use use policy gradient. Parameterize policy functions by

$$a_A = f_A(s_t, s_0) \tag{1}$$
$$a_B = f_B(s'_t, s'_0) \tag{2}$$
$$a_{\text{target}} = f_B(s''_t, e) \tag{3}$$

where e is a dummy vector/task description.

The policy that Bob learns will be used for the target task.

We hope self-play will be useful for target. Try different, both discrete and continuous settings.

Under strong assumptions (tabular policies, finite state) Bob must learn all possible tasks (how to transition between any pair of states as efficiently as possible). If Bob fails on a certain task, then Alice would propose to increase reward. Then Bob sees the task and learns it to increase reward.

There might be tasks that Alice fails to propose. (Q: how do you know you will reach equilibrium?)

Related work: self-play (checkers, backgammon, Go, RoboSoccer), GAN's (think of Alice as generator of hard examples, Bob as discriminators), intrinsic motivation (they have reward for novelty of state, we learn to transition between pairs of states), robust adversarial reinforcement learning (adversarial perturbation to state).

Experiments:

- use Reinforce algorithm with learnt baseline and entropy regularization
- 2-layer NN model for Alice and Bob (separate)
- Train on 20% target task and 80% self-play
- discrete vs. continuous environments
- measure target task reward vs. number of target task episodes (self-play is free)

Toy example: long hallway.

MazeBase: LightKey task (2-D world)

Grid is procedurally generated. Toggle key to lock/unlock door, toggle light on/off (only switch is visible when light is off). Goal is to reach flag in other room. Alice takes more time steps as training goes on. This means Bob is solving the shorter paths.

Mountain car

- control car stuck in 1-D valley (need to build momentum by reversing)
- Sparse reward (+1 if reach left hill top)
- hard task because random exploration fails
- asymmetric environment, so do repeat self-play

Reinforce + self-play does as good as other exploration methods (TRPO + VIME, TRPO + SimHash). Reinforce alone fails.

Ex. Alice goes halfway up the wall.

Bob gets gradient both from self-play and target task. Alice only has to fool Bob.

Swimmer Gather:

- control worm with 2 flexible joints, swimming in 2d viscous fluid
- reward +1 for eating green apples and -1 for touching red bombs
- reverse self-play

To swim anywhere, coordinate over multiple time steps.

Alice may find one particular task and keeps giving it. It can concentrate in a single point. We want meta-exploration for Alice—have multiple Alices? Future work: mark target state, propose task by communication, propose a hypothesis for Bob to test.

We can control how similar the self-play and target episodes are: ex. whether the light is on. For repeat self-play, if light is off consistently during self-play it helps; for reverse self-play, if light is on consistently it helps.

9 Learning from Unlabeled Video (Kristen Grauman, University of Texas, Austin)

The status quo in visual recognition is to learn from batches of unrelated Web photos labeled by human annotators. Yet cognitive science tells us that perception develops in the context of acting and moving in the world—and without intensive supervision. How can unlabeled video augment computational visual learning? Ill describe our recent work exploring how a system can learn effective representations by watching unlabeled video. First we consider how the ego-motion signals accompanying a video provide a valuable cue during learning, allowing the system to internalize the link between "how I move" and "what I see". Building on this link, we explore end-to-end learning for active recognition: an agent learns how its motions will affect its recognition, and moves accordingly. Next, we explore how the temporal coherence of video permits new forms of invariant feature learning. Incorporating these ideas into various recognition tasks, we demonstrate the power in learning from ongoing, unlabeled visual observations—even overtaking traditional heavily supervised approaches in some cases.

`http://www.cs.utexas.edu/~grauman/research/pubs.html`

Take a scene and understand object categories, instances, things happening, etc. Look at benchmark datasets that feed the algorithms, BSD, Caltech 101/256, PASCAL, LabelMe, ImageNet, SUN, Places, MS COCO, Visual Genome.

How are we teaching these systems? How do our systems learn about the visual world today? Curate 1000+ images for a category to teach the concept. This is a limitation. It's expensive and restrictive in scope: the way we teach, and what we want them to do at test time.

The status quo is to learn from disembodied bag of labeled snapshots. Our goal is visual learning in the context of acting and moving in the world. We want to shift to this more embodied learning. Take account of continuous, long-running observations (video), multi-sensory input. This will be inexpensive and unrestricted in scope.

I'll talk about

1. learning representations tied to ego-motion
2. learning representations from unlabeled video
3. learning how to move and where to look

9.1 Learning representations tied to ego-motion

Kitten carousel experiment (Held, Hein 1963). Scientists study kittens' visual development. They are in dark every day except one hour, when they are on the carousel. The active kitten controls motion, while the passive kitten doesn't. Tha active kitten's vision develops normally, while the passive kitten doesn't. Key to perceptual development is self-generated motion and visual feedback.

How do we get this into a modern visual recognition system?

Teach computer system the connection: how I move (ego-motion) vs. how my visual surrounding change. It could be on glasses, a car, or handheld camera. Take unlabeled video and sensor data external to vision (accelerometer, GPS). Build representation prior to tackling a specific challenge.

Cast in terms of view prediction task, conditioned on ego-motion. Predict before making ego-motion what world will look like after making action. Fundamental vision cues give us hints.

What is the vision system forced to pick up on? Semantics—street intersection even if not observed. Geometry: occlusions. Grouping (other face of building).

Cast in terms of equivariant feature learning.

1. Invariant features are unresponsive to some classes of transformations $z(gx) \approx z(x)$.

2. Equivariant features are predictably responsive to some classes of transformations, through simple mappings (e.g., linear), $z(gx) \approx M_g z(x)$.

Invariance discards info, equivariance organizes info.

Pairs of frames related by similar ego-motion should be related by similar transformation.

Take unlabeled video and learn equivariant features, $z_\theta(gx) = M_g z_\theta(x)$. Use a pair of Siamese embeddings (identical convolutional NN's). An additional network takes care of transformation, M_g. Loss is

$$\|M_g z_\theta(x_i) - z_\theta(gx_i)\|_2 .$$

We discovered 6 ego-motions by clustering. We've played with continuous motions but not been successful yet. We do 2-D view prediction conditioned on ego-motion. You could try including depth information, etc.

You can jointly train recognition models, or do unsupervised learning first and then do classifications on the learned features.

Learn from unlabeled car video (KITTI). Exploit features for static scene classification.

Can we build up the fundamental vision cues on semantics, context, geometry when we have few labels for the target task?

Our results improve on baselines.

Next steps: one-shot shape reconstruction for feature learning. Predict entire 3-D object shape. Given one view from unknown viewing angle, reconstruct entire view grid (view from all angles).

9.2 Learning representations from unlabeled video

What if we have passively watched video? This is attractive because of the large volume of such video.

Prior work is slow feature analysis: high-level information changes slowly over time, most of time. Look at nearby frames. Learn feature map z such that

$$z(a) \approx z(b)$$

where a, b are adjacent frames.

We introduce steady feature analysis: do higher order temporal coherence. Change from a to b should be like change from b to c,

$$z(a) - z(b) \approx z(b) - z(c).$$

This is equivariance. Do Siamese triplet embeddings. Use contrastive loss: I want the correct ones to be close, closer than negative pairs (random, from the wrong ego-motion).

9.2.1 Pre-training a representation

Suppose I have a new task for which I have a few labels. If I have a related enough task, I can take a network, train it for the original task, and use it as a starting point for learning the new task (fine-tuning).

What if pre-training is watching the video?

Can we learn more from unlabeled video than related labeled images? Yes, unlabeled video (HMDB) does better.

Is it because there's more video, or because video is better?

9.3 Active recognition

Benchmark challenges limit our scope. We have passive, disembodied snapshots at test time, too. If we have active systems, have it move around and make decisions that way. This is active recognition: move around to recognize.

If you don't take static photos, you have motion blur, composition, etc. But you have the ability to move to change input. If you can't tell from one angle if something is bowl or mug, move to disambiguate. After perception, do action selection. Perceive again, do evidence fusion.

Each of these 3 modules are tackled independently. Now we have the opportunity to learn all these things jointly, end-to-end learning. Do multi-task training of active recognition components and look ahead.

Settings:

- 360 panorama, choose which glimpses to take.
- robot manipulate object, how to choose movement of hand
- ModelNet—artificial objects.

Ex. 3 views to decide on "plaza courtyard."

FusionSeg: pull objects out of video. Foreground segmentation out of video.

Challenge of viewing 360° videos: if you try to watch, you have to click around to swing the camera. How to find the right direction to watch?

Input is 360° video, output is natural looking regular video. We can learn what human-taken videos look like from YouTube. We'll select a trajectory of viewpoint that looks human-taken. Ex. track the human diver.

g's are local actions. Can you enforce a composability structure?

10 Representations in deep reinforcement learning (Peter Abbeel, Berkeley)

In RL, agent chooses action, environment changes state, and gives reward signal. Supervision is in terms of reward accumulated over time rather than on which actions to take. You can model robotics, dialogue, logistics, gameplay, anything where after acting the world changes state and agent acts in the changed world.

Before people used heavily engineered features; now there's neural nets.

There are many approaches. Two big approaches are policy optimization and dynamic programming (decompose into pieces, solve based on temporal decomposition).

I'll look at policy gradients (policy optimization), Q-learning (DP), and actor critic methods.

10.1 Classical RL

In policy optimization, $\pi_\theta(u|s)$, the policy, is parameterized by θ. Environment is stochastic, so we want

$$\max_\theta \mathbb{E}\left[\sum_{t=1}^{H} R(s_t)|\pi_\theta\right].$$

Stochastic policy can smooth out the optimization landscape.

Likelihood ratio policy gradient: Let τ be state-action sequence $s_0, u_0, \ldots, u_H$. Optimize

$$\sum_\tau P(\tau;\theta)R(\tau).$$

We don't need a dynamics model of the world. Take the gradient wrt θ:

$$\nabla_\theta U(\theta) = \sum_\tau \frac{P(\tau;\theta)}{P(\tau;\theta)}\nabla_\theta P(\tau;\theta)R(\tau) \tag{4}$$

$$= \sum_\tau P(\tau;\theta)\nabla_\theta \ln P(\tau;\theta)R(\tau) \tag{5}$$

$$\nabla_\theta U(\theta) \approx \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \ln P(\tau^{(i)};\theta)R(\tau^{(i)}). \tag{6}$$

We can estimate this by sampling. Point in direction to make the trajectory I just experienced more likely, by amount proportional to the reward.

Gradient increases probability of paths with positive R and decreases probability of paths with negative R. The likelihood ratio shifts mass on paths, and does not try to change the paths. Decompose path into states and actions

$$\nabla_\theta \ln P(\tau^{(i)};\theta) = \nabla_\theta \left[\sum_{t=0}^{H} \ln P(s_{t+1}^{(i)}|s_t^{(i)},u_t^{(i)}) + \sum_{t=0}^{H} ...\right] \tag{7}$$

$$\widehat{g} = \frac{1}{m}\sum_{k=1}^{m}\sum_{t=0}^{H-1} \nabla_\theta \ln \pi_\theta(u_t^{(k)}|s_t^{(k)}) R(\tau^{(k)}) \tag{8}$$

$$= \frac{1}{m}\sum_{k=1}^{m}\sum_{t=0}^{H-1} \nabla_\theta \ln \pi_\theta(u_t^{(k)}|s_t^{(k)}) \sum_{t'=t}^{H-1} r_{t'}^{(k)} \tag{9}$$

$$= \frac{1}{m}\sum_{k=1}^{m}\sum_{t=0}^{H-1} \nabla_\theta \ln \pi_\theta(u_t^{(k)}|s_t^{(k)}) \sum (r_{t'}^{(k)} - b(s_{t'}^{(k)})) \tag{10}$$

$$b(s_{t'}^{(k)}) = \frac{1}{m}\sum_{i=1}^{m}\sum_{t'=t}^{H-1} r_{t'}^{(k)}. \tag{11}$$

(= here means in expectation.)

Notes:

1. Original expression is high variance. We only need to look at reward in future.
2. Compare with average rather than push everything up

Make gradient update $\theta_{i+1} \leftarrow \theta_i + \alpha\widehat{g}$.

Can we do better. Use a function to represent b to generalize from past data. Introduce V^π and plug in for b. Bellman equation is

$$V^\pi(s) = \sum_u \pi(u|s) \sum_{s'} P(s'|s,u)[R(s,u,s') + \gamma V^\pi(s')]$$

(immediate reward plus reward after that). For large state spaces, do fitted V iteration,

$$\phi_{i+1} = \min_\phi \sum_{(s,u,s',r)} \left\| r + V_{\phi_i}^\pi(s') - V_\phi(s) \right\|_2^2 + \lambda \left\|\phi - \phi_i\right\|_2^2 .$$

If you have a rich representation this works well.

This gives Actor-Critic algorithm: policy gradient and fitted V iteration.

- Initialize π_{θ_0}, $V_{\phi_0}^\pi$.
- Collect data (s,u,s',r)
- Update ϕ_{i+1} as above.
- Update θ_{i+1} with $V_{\phi_i}^\pi(s_{t'}^{(k)})$ rather than with ϕ_{i+1}.

Generalized adventage estimation (Schulman 2016) also uses learned value function to better estimate first term $\sum r_{t'}^{(k)}$.

Exploration: force to have certain entropy. Exploration is small. Another approach is to insert exploration bonuses.

How much data to collect? It's not clear how much you need; it's more stable with more data. Stability has been more the focus than data efficiency.

Is it useful to batch data collection.

Initial premise: reset and re-rollout, does that give higher signal-to-noise ratio? I couldn't get that to work. Which state to roll-out from (information theoretic considerations, e.g., say one state is particularly interesting).

Won't work well if environment adapts/changes too quickly for you to keep up with. (Ex. adversaries adapt to you.)

Another approach is Q-learning: instead of representing value of state, represent value of (state, action). Actu u and then act optimally.

$$Q^*(s,u) = \sum_{s'} P(s'|s,u)[R(s,u,s') + \gamma \max_{w'} Q^*(s',u')].$$

Fitted Q-iteration:

1. Init Q_{ϕ_0}
2. Collect data (s,u,s',r)
3. Update $\phi_{i+1} \leftarrow \|r + \max_{u'} Q_{\phi_i}(s',u') - Q_\phi(s,u)\|_2^2 + \lambda \|\phi - \phi_i\|$.

Interleave collecting data and updating.

Double DQN: keep 2 Q functions. Compute argmax from 1, help other to not have that much drift.

You can use off-policy data. You don't have to just have to use Q-data from current policy, unlike Actor-Critic. It's more data efficient, you can replay all past data.

10.2 Different approaches/architectures

In DQN, use 3 conv nets and 2 fully-connected layers to play beyond human level.

TRPO+GAE: output probability distribution over actions. V_ϕ output 1 value to subtract out. It's easy to stabilize because you can put trust region on policy.

A3C: merge π_θ and V_ϕ. Have one extra output.

Dueling: Q-learning approach. What is in the Q-function? Differentiate between different actions. Q-value is dominated by state. Signal dominated by value of state. Separate this out. Fully connected to get V and advantage $Q - V$. Combining gives Q function. Zone in on value separately from differences between actions. Best performing algorithms use this. "Dueling" for attention of Q-function. Certain rescaling of gradients are the right thing to do.

TRPO+GAE can be used for continuous control.

Random exploration. Rewards are well-shaped—reward based on standing head height, moving north.

For real robots: Amount of data is impractical. We modify architecture. Bottleneck (attention) layer: look at conv responses; for each filter, check where it is most active and extracting coordinates. There are 92000 parameters. Have 1 controller that has access to state (learn quickly) and 1 controller with access to image (which has to agree).

Hanging coat hanger, hammering nail.

How generalizable are these policies? They are specific to task. How do you learn things that generalize beyond the skill?

You can have A3C with LSTM, auxiliary losses (learn Q-function for other objectives—you can in parallel train Q-functions with same trunk).

You can learn to navigate maze environment from first-person vision.

They store whatever is relevant to making decisions in future. Can you predict coordinates based on activations in LSTM? There are some results that these things happen.

Exploration through reward bonuses: give reward for seeing something you haven't seen before.

- VIME: Bayesian neural net (Houthooft 2016)

  Posterior over dynamics models changes. KL divergence is how informative what you saw.

- Pixel-CNN density estimation (Ostrovski 2017)

Steer towards things that are novel

Value interation network (VIN): A lot of problems are planning problem. It's not clear how to plan over raw sensory inputs. Put VI in module, obsevation feeds into it. Force observation processing to extract something like state so that planning against it does well. Prior on what you think should happen to learn more effectively.

First-person mapping and navigation with VIN.

Embed to control: LQR in middle.

Predictron (Silver 2016): solve search problem. Look at where I am. Predict reward in future. I don't have dynamics, reward model, but train to be good at predicting and plan against. Plan to hit billiards ball.

If you just learn dynamics model, it's easy to spend resources on wrong thing (ex. how leaves on tree move). Zone in rewards and values.

Modular networks: 2 robots, 2 tasks, a network associated with each robot, each task, and mix and match. Train to be good at being sequenced together.

Option-critic architecture. Critic computes value function. Low-level policy: 1 is active. High-level gates say which one is active. Choose which one goes next. Higher-level think about longer horizons.

Feudal networks: most promising results (Dayan Hinto 1993, Vezhnevets 2017). Tell someone else to do it. Hi level just say what outcome it wants. Decompose into hierarchy, don't need to reason forward more than 10 steps at each level. Manager has internal reward representation, goes into LSTM, feeds into goal state, project to worker space. Weighted sum of distributions over actions. Selects an action to sample. High level sets goals, low level

tries to achieve goals. Reward is how well you satisfy goal of level above you. What goals should you be setting? Naively the high level can run standart policy gradient. It's not aware of the effect of request. $\nabla g_t = A_t^M \nabla_\theta d_{\cos}(s_{t+1} - s_t, g_t(\theta))$. How good was what happened compared to average, set goals according to what just happened (rather than asked for)

10.3 Meta-learning

Wheere are the ImageNet features of motor control? ImageNet is large-scale, emphasizes diversity, evaluated on generalization. Problem:

- small-scale
- emphasizes mastery
- evaluated on performance

Can we emphasize generalization? We want a few gradient updates to solve a new task. We want a gradient on the gradient update. After gradient update we do really well (MAML),

$$\theta \leftarrow\!\shortmid \theta + \alpha \sum_i \nabla_\theta [\theta + \alpha \nabla_\theta R_i(\theta)].$$

Ex. after 1 gradient step, learn forward or backward motion!

Ex. omniglot few-shot classification.

RL2. Deep RL takes 40 days, human takes 2 hours. TRPO, DQN, A3C are fully general RL algorithms. MDP's in real world are a tiny subset of all MDPs that could be defined. Can we build a good prior for real-world MDPs, design algorithms that make use of such prior information?

Key idea: learn a fast RL algorithm that encodes this prior?

RL2 is trained on many different environments. It does as well as carefully designed bandit algorithms, Bayesian optimal algorithms, good strategies for mazes. Once trained, it learns to remember the maze when dropped in a new maze.

Deep RL code bases: rllab, rlpy, GPS.

11 Supersizing Self-Supervision: Learning Perception and Action without Human Supervision (Abhinav Gupta, Carnegie Mellon University)

In this talk, I will discuss how to learn representation for perception and action without using any manual supervision. First, I am going to discuss how we can learn ConvNets for vision in a completely unsupervised manner using auxiliary tasks. Specifically, I am going to demonstrate how spatial context in images and viewpoint changes in videos can be used to train visual representations. Next, I am going to talk about how we can use a robot to physically explore the world and learn visual representations for classification/recognition tasks. Finally, I am going to talk about how we can perform end-to-end learning for actions using self-supervision.

Human learning is lifelong, integrated.

Manual labeling is not scalable. Imagenet has collected 1 million bounded boxes over 5 years. Common sense (Cycorp) has 2 million facts. Facebook gets 400 million photos daily.

We talk about

- self-supervised systems. Get supervision from data on auxiliary tasks.
- physical exploration
- 300 million images

Deep learning is based on promise of learning from mega-scale data. But does more data actually help in representation learning. Maybe performance will plateau. Deep learning depends on GPUs, model size, and data. GPUs and model size have increased, but not data.

11.1 Self-supervised systems

In supervised learning, hope that in training classification, get useful features (parts, pose, materials, geometry, boundary...). Do we even need manual labels? If we can get rid of this, we can scale to billions of images.

Use context as supervision (Collobert and Weston 2008, Mikolov et al. 2013).

Train a CNN to predict patches in context.

Ex. predict toilet paper roll besides toilet. But this is hard: Predicting pixels is a high-dimension regression problem.

Easier: predict spatial layout—how 2 image patches are related to each other. Even though the task is nonsemantic, you have to go via a semantic representation!

Sample 2 patches, feed through Siamese network, and learn classifier on top that predicts the spatial layout. The top layer of the CNN should be semantic in nature. Indeed, nearest neighbors are from the same category.

We have to avoid trivial shortcuts, ex. matching boundaries. So include gap and jittering.

Network has learned to encode location inside image itself: ex. nearest neighbors are all from bottom right. Neural net can predict location in image by chromatic aberration. Crop color channels, or work on black and white images.

What is learned? Look at nearest image images. It's comparable to AlexNet.

How to evaluate: Use as pre-trained network for VOC object detection. Compare to ConvNet. AlexNet does 40.7% for no pretraining, 46.3% for our pretraining, 54.2% when using ImageNet labels.

Increase capacity and use VGG, which gets 42.4%, 61.7%, 68.6%.

Within image-context leads to across-image similarities. Significantly better than no pretraining.

Is there not enough supervision? It's hard to learn viewpoint invariance. Ex. a face upside down would make legs above.

There is one source where viewpoint invariance comes for free: videos. Use this data to learn viewpoint invariance between images. Do Siamese triplet: put first patch and last

patch of person/object in track so they are close to each other, and other is far (contrastive loss).

Space of negatives is huge, and might take lots of time for network to train. Use hard negative mining for triplet sampling. Select negative patches giving high loss to backprop. When just learning viewpoint invariances, VGG get 60%.

Given videos, find closest patches. Use transitivity, loop closure to find new invariances we should be learning.

Using both, get 63.2% for VOC 2008. B

11.2 Learning in humans

We throw things in air, put things in mouth... Build a simliar robot that.

- Planar grasping: randomly grasp object.
- Pushing
- Poking
- Pose and scale invariance (change viewpoint). This is hardest; loss in 7th layer.

For retrieval we beat AlexNet.

We have a drone trying to hit objects to get information (haptic feedback).

11.3 Promise of big data

Data does not seem to be growing, especially data. There is no real effor in making ImageNet 10x or 100x, why?

1. It is not about data anymore, it is all about model; we see plateauing effect.
2. It is not about general representation learning. We should collect task-specific data, COCO, 100k images for object identification and segmentation.

We revisit unreasonable effectiveness of data.

Network has been training 2.5 months on 50 GPUs (3 epchs).

Surprise: Performance between data and performance is linear. There is no plateauing effect at 300 million.

Data still has the biggest impact. Data is not saturating.

Future is ripe for unsupervised; data seems to be very helpful.

Discriminative models see to be better than generative models.

12 Failures of Gradient-based Deep Learning (Shai Shalev-Shwartz, Hebrew University of Jerusalem)

The empirical success of Deep Learning is stunning, and everyday we hear new success stories. However, for both theoreticians and practitioners, it is important to understand the limitations. We describe three families of problems for which existing deep learning algorithms fail. We illustrate practical cases in which these failures apply and provide a theoretical insight explaining the source of difficulty.

Joint work with Ohad Shamir and Shaked Shammah.

Simple problems where standard deep learning either does not work well (require prior knowledge, more than gradient update, decompose problem and add supervision) or does not work at all.

I will give intuition about when and why deep learning doesn't work.

12.1 Piecewise linear curves

Ex. paint edge of free space (where car can go).

Problem: train piecewise linear curve detector, $f = (f(0), \ldots, f(n-1))$ where $f(x) = \sum_{r=1}^{k} a_r [x - \theta_r]_+$, $\theta_r \in \{0, \ldots, n-1\}$.

Output curve paramteres $\{a_r, \theta_r\}_{r=1}^k$.

First try: deep autoencoder. Learn encoding network E_{w_1} and decoding network D_{w_2}. Squared loss is $(D_{w_2}(E_{w_1}(f)) - f)^2$. This is doable. It doesn't work so well.

Second try: pose as convex objective. Let $p \in \mathbb{R}^{n,n}$ be k-sparse vector whose k max/argmax elements are $\{a_r, \theta_r\}_{r=1}^k$. Then $f = Wp$, $W_{i,j} = [i - j + 1]_+$. Train a 1-layer fully connected network, $\min_U \mathbb{E}[(Uf - p)^2] = \mathbb{E}[(Uf - W^{-1}f)^2]$.

If works better than the autoencoder. It is convex and realizable but still doesn't work well. It will converge, but not after 50,000 iterations.

Convergence of SGD is governed by condition number of $W^T W$ which is large, grows with resolution,

$$\frac{\lambda_{\max}(W^T W)}{\lambda_{\min}(W^T W)} = \Omega(n^{3.5}).$$

Adagrad/Adam doesn't work because diagonal conditioning doesn't work.

3rd try: pose as convolution. $p = W^{-1}f$ where W^{-1} has 1, -2, 1 on main, -1, -2 diagonal. $W^{-1}f$ is 1D convolution of f with 2nd derivative filter. Train convnet. Condition number is $\Theta(n^3)$. Converges after 50,000 iterations but not 10,000. $\Theta(n^3)$ is disappointing for problem in $\mathbb{R}^3$.

4th try: convolution and preconditioning. Solve after 500 iterations.

Convnet allows for efficient preconditioning; we are estimating and manipulating 3×3 rather than $n \times n$ matrices.

Lesson: SGD might be extremely slow. Prior knowledge allows us to choose better architecture (not for expressivity, but for better geometry); choose better algorithm.

12.2 Flat activations

We have vanishing gradients due to saturating activation s (e.g. RNN's).

Problem: learn $x \mapsto u(\langle w^*, x\rangle)$ where u is fixed step function. Optimization problem:

$$\min_w \mathop{\mathbb{E}}_x[(u(N_w(x)) - u(w^{*T}x))^2].$$

Problem: $u'(z) = 0$ almost everywhere so we can't apply gradient-based methods.

Smooth approximation: replace u with $\widetilde{u}$. This is a headache.

Approach: end-to-end $\min_w \mathbb{E}_x[(N_w(x) - u(w^{*T}x))^2]$. Slow train and test time, and curve is not catured well.

Approach: multiclass, where $N_w(x)$ is the step that x belongs.

Different approach: descent but not with the gradient. "Forward only" backpropagation.

$$\min_w \mathop{\mathbb{E}}_x \left[\frac{1}{2}\left((u(w^Tx)) - u(w^{*T})\right)^2\right].$$

In the backward pass, think the layer is identity. Lesson: local search works, but not with the gradient.

Flat activation is desirable: you want a gate. Message of deep learning was to avoid flatness? Machine learning: you want flatness...

12.3 End-to-end training

We didn't do anything, it's all learned by itself.

2 approaches: end-to-end, backprop everything; think about your problem, decompose into subtasks, and supervise subtasks.

There are some problems where if you tackle end-to-end you fail.

Input is x, a k-tuple of images of random lines. $f_1(x)$ is whether slope is positive or negative. $f_2(x)$ is parity of slope signs. Goal: learn $f_2(f_1(x))$.

Architecture: concatenate LeNet and 2-layer ReLU.

End-to-end approach: Train on primary objective. First network looks at 3 images, gets one number per image (+/-). Numbers enter into 2nd network.

Decomposition approach: augment objective with loss specific to first net using per-image labels.

For $k = 1, 2$ both work, but for $k = 3$, composable approach works but end-to-end approach does as well as random chance.

In the decomposition, supervise more.

Similar experiment by Gulcehre, Bengio 2016. They suggest local minima problems; we show the problem is different.

SNR for random initialization: SNR of end-to-end for $k \geq 3$ is below recision of float32.

LB: zip codes: find zip code, split, read digits. Do these tasks separately. If after doing this you do fine tuning end-to-end it does better.

12.4 Learning many orthogonal functions

Let H be hypothesis class of orthonormal functions: for all $h, h' \in H$, $\mathbb{E}[h(x)h'(x)] = 0$.

Improper learning of H using gradient-based deep learning: learn w of architecture $p_w : X \to \mathbb{R}$. For every target $h \in H$, learning task is

$$\min_w F_h(w) := \mathop{\mathbb{E}}_x[l(p_w(x), h(x))].$$

Start with random w and update weights based on $\nabla F_h(w)$.

How much does $\nabla F_h(w)$ tell us about identity of h?

Theorem: for every w, there are many pairs $h, h' \in H$ such that $\mathbb{E}_x[h(x)h'(x)] = 0$ while

$$\|\nabla F_h(w) - \nabla F_{h'}(w)\|^2 = O\left(\frac{1}{|\mathcal{H}|}\right).$$

Number of such functions could be exponential in dimension.

Proof: show that if functions are orthonormal, then for every w,

$$\mathrm{Var}(H, F, w) := \mathop{\mathbb{E}}_h \left\| \nabla F_h(w) - \mathop{\mathbb{E}}_{h'} \nabla F_{h'}(w) \right\|^2 = O\left(\frac{1}{|\mathcal{H}|}\right).$$

We have

$$\nabla F_h(w) = \mathbb{E}_x[p_w(x)\nabla p_w(x)] - \mathbb{E}_x[h(x)\nabla p_w(x)] \tag{12}$$

where first term is independent of h. Fix j, let $g = \nabla_j p_w(x)$. Expand $g = \sum_{i=1}^{|H|} \langle h_i, g \rangle h_i$+(orthogonal component).

An example of such a class is the parity functions. There are 2^d orthonormal functions so there are many pairs $h, h' \in H$ orthogonal but $\|\nabla F_h(w) - \nabla F_{h'}(w)\|^2 = O(2^{-d})$. (Similar hardness can be shown by combining existing results.)

Illustration: linear-periodic functions, $F_h(w) = \mathbb{E}_x[(\cos(w^T x) - h(x))^2]$ for $h(x) = \cos([2, 2]^T x)$ in 2 dimensions. No local search works.

- Optimization can be difficult for geometric reasons other than local min/saddle points: condition number, flatness. Using bigger/deeper network doesn't always help.
- Remedies: prior knowledge can be important. Convolution can improve geometry, use "other than gradient" rule, decompose problem and add supervision.
- Understand limitations.

`http://www.arxiv.org/abs/1703.07950`

13 3/28/17 Panel

1. Good representation: what is it? How do we recognize a good one?

   EH: A good representation is one that is succinct. You can measure generalization. I agree there is something more but I don't know how to capture it.

   LB: Compression depends on the statistics of the data. Very often, you want different distributions for different tasks and distributions, and compression won't mean the same thing.

   What is a good abstraction in math? It's short but also provides you many means to work with. A good representation allows you do manipulations. Ex. parse tree. Do you want it to be compact, or allow you to manipulate the sentence? These are different concepts. In math, abstraction allows you to come up with new statement, things I haven't seen before.

   SSS: Represent data without knowing what comes after, what tasks I will do.

   RS: Ex. take mushroom features for medical images.

   Remove the word "representation." What you are learning is a hypothesis class. Fix H. Learn ϕ, implicitly defining H_ϕ, all functions in H applied to ϕ. The important thing is H_ϕ, not H.

   NS: I have to talk about doing a lot of tasks. If I have a single task, the best representation is just the answer. A good representation is just a hypothesis test that can cover all the tasks. It's hard to differentiate representation and multi-task learning.

   BSD: Representation is an algorithm. If my representation is nearest neighbor, my task is similarity. Think of representation learning is something which will be useful for your favorite aglorithm.

   MW: Look at learning problem in complexity theoretic sense. We run into hardness results quickly. The fascinating thing is that we actually have representations from neural networks (mushrooms to medical images, going between languages).

   NS: If we didn't have computational constraints, up to certain edge cases, you can just learn by ERM.

   LB: This is only a small part of learning.

   CM: I think we should go out into the scientists. Physicists would say something is a good/bad representation of something. Support natural generalization in problem.

   NS: Whole advantage of learning is that we don't have to make sense of representations. Once you do that you hurt the learning.

   EH: It's very much guided by learning by example. All we know how to do in Statistical LT is learning by example. But there are other types of learning. Representational learning might capture that.

   LB: Instead of talking, let's project examples on the board.

CM: How to reconcile the two? Any linguist would claim to know what a good representation is. It should allow you to capture, it's a good representation of any linguistic property you can then start looking at, control verb, etc. can be naturally explained in terms of the representation. You have all these natural questions you can ask, if representation allows you to represent it you have a good representations. They exist beforehand.

LB: To some extent they're wrong because instead of using their representation they should use a BiLSTM.

NS: If you just want to do a task, OK. But if you want to understand...

LB: Why is it different?

NS: Understanding gets in the way. Humans want to understand, machines want to do the task.

LB: What about AI?

RS: What about images?

AG: ImageNet is best representation, but it's just a good initialization, fine tuning changes all the layers. Invariants for every task is different. Ex. I may or may not want pose invariance. Representations are at odds with each other.

CM: Why are they at odds? Why not support both?

RB: May want to be expert at one. Depends whether you value generalization or performance.

AG: I care about performance.

MW: I didn't know there was something in common between Chinese and English. Some smart representation goes across languages.

AG: Language is different from vision. Language is designed by humans. Vision is something we are observing about the world.

NS: Training on mushrooms and doing better on medical images is multi-task learning. Ego-motion learning is multi-task learning.

KG: The surprise: it's important to start with it. It's a multi-task success without the multi-task present. It's surprising the generality that came out.

AG: We don't understand the fine tuning. There's evidence, if you take a network, not fine tuning, one might be better, but fine tuning, the other network might be better. I try these things and am surprised. How fine-tuning makes things better? I can just tell from practice what I'm seeing.

NS: Algorithm: start with initialization and fine-tuning. You get better results than starting from scratch. This is representation learning.

M: Fine-tuning comes from your brain.

AG: Fine-tuning: take labeled data for task, use loss function on that task to change your weights.

M: So fine-tuning is a specific process.

LB: Fine-tuning erases everything taught before.

NS: Not necessarily.

2. How to incorporate prior knowledge, symbolic reasoning, etc.?

   RS: In ImageNet thinking of the right representation didn't get people anywhere. Where is it useful?

   RF: Architecture is a huge prior, ex. locality. Can you automatically figure out architecture.

   SA: There was this idea in vision before...

   RS: Deformal parts models.

   RF: Stack idea in multiple stages, so you can look at large-scale features.

   SS: There are 2 aspects: expressivity and sample complexity. If you use prior knowledge, and it's correct, it helps sample complexity. It can help in terms of function approximation.

   End-to-end vs decomposable affects optimization, even if for expressivity you don't need it. It's not a matter of sample complexity, but of geometry.

   RS: It can be hard to figure out what the decomposition is.

   CM: Practical answer is, if you know some thing as a prior, the easiest way to take advantage of this is data augmentation (and subtract out anything irrelevant). That's one answer, I don't think it's the right answer. We want to take advantage of the structure of domains, I'm interested in doing that even if it doesn't always work.

   RS: Most papers: we go from X to Y, everyone's trying to do end-to-end.

   LB: If you try to do something more complicated it won't fit in the paper.

   Two ways to implement prior knowledge: in way overwritten by data, or in a way that's hard (ex. convnet). Prior knowledge is schematic. Data is high-dimensional. We should do it in way that can be overwritten. Instead of having loss forcing to take value you thought it's good, add another output layer connected to middle layer. Don't force the middle layer to be exactly what you say; instead, force the output that be that. This works better.

3. Bayesian methods for neural networks?

   DR: It's difficult to put a prior.

   RS: Bayesian neural networks try to incorporate prior knowledge, uncertainty in parameters. Do it softly, through other layers. Define multiple losses, have to tweak regularization... it's not kosher, lots of tweaking.

4. SA: Symbolic reasoning? There ought to be more, but it's not clear how to incorporate it.

LB: We would like representations that are subject to the same kind of algebraic operations. A network takes two concepts and puts them together to get one, and can disassociate. If you associate and disassociate, you should get back what you had.

Construct your network to have elementary algebraic properties you want it to have.

NS: Use algebraic structure of transformations...

LB: When you have ego-motion, you can have compositions that bring you to the same place. Train the network to satisfy this. Have the bricks for an algebraic system. It's not the full thing but a beginning.

SA: Is it a good research goal.

NS: Symmetry groups isn't symbolic reasoning.

LB: When you work on words you have symbols.

SA: But then you go immediately to a continuous representation. What's logic on that?

LB: In symbolic reasoning there are 2 words, symbolic, reasoning. Reasoning is already very important. We describe it in algebraic system. To capture this notion, we have to have some algebraic properties, find ways to use them inside these networks. The symboli part, I don't know.

SBD: Some kind of hierarchical learning. Logical reasoning is high-level skill, common to many tasks. More principled guiding learned from many tasks together, help in each specific tasks. It's some kind of meta-learning. On lower-level it's symbolic computations.

LB: Describe a task that needs 2 agent to cooperate (ex. encoder, decoder), and make the channel quite noisy. Something that explains the emergence of symbolic communication.

Semantic hashing.

RS: That was done to constrain communication channel.

LB: It became more discrete.

CM: It's interesting, human beings want to construct symbols for anything at all no matter how continuous it is, clouds, shapes, waves when you're surfing. Classes of things helps them think about things, remember.

LB: It helps them remember. It's about reliability.

CM: Categories influence your decisions.

?: How to learn from both data and symbolic knowledge—constraints about what is possible in the world, what is for sure. SS, do you find connections to tasks that correspond to the functions?

SS: They're not very natural. But think about scores in tennis game. You need to learn who won at the end. SBD conjectured it will be difficult to learn end-to-end.

SS: What if you take another boolean function at end. We were trying many different boolean functions. For some of them the training works end-to-end.

SBD: Sensitivity?

SS: Wrong answer. We think it's a matter of frequency. Lower frequency in Fourier domain are easy to learn, higher frequency is harder to learn. There are many other functions (besides parity) that are hard as well.

NS: In language, negation words like "except". It's not that that word has a vector, I need to negate some other word. If I'm just doing translation that's fine. If I want to do something deeper, QA... how do these pose challenges to systems that don't use symbolic reasoning?

CM: We can build neural distributed representation with more structure that handles them. They can't possibly be handled right with bag or words, RNN. You need to understand scope, etc. Gee, tree-structured neural networks, I can do it with that. People have looked at higher order tensor networks. You don't have to go the symbolic route.

This gets back to the first question. A paper argued against LSTM's. For a linguist, if you have negation and stuff under scope, that's an obvious fact that any representation should be able to capture and model, any representation that doesn't do that is immediately broken and not of interest no matter what's coming out. The paper (Tau Lindson): look at subject verb agreement, basic fact of many languages. If you split subject and verb with other stuff, human accuracy drops. The students who talk to the reporter has... Human performance is mid-90's. Any of LSTM models performances very quickly drops down to random as distance gets longers. LSTM's aren't capturing those kinds of facts well at all.

EH: You don't need to go to language to find this. Some we have the mechanism wrong, ex. adding numbers.

RF: Train on up to 10 digit numbers. It does beautifully on training set. As soon as you try on 100 digit it fails.

SA: Dawn's paper takes care of that in recursion.

NS: "Use more layers." Can we train these things effectively. LSTMs with 2–3 layers, you can have the power to capture negations (xors, etc.). Will you succeed in learning this from data?

LB: One thing that plays against you is that in the dataset most sentences don't need that. A small number do, with small gradient signal.

CM: On one hand, it's extremely surprising that multilayer LSTM can learn a lot of structure and manipulate it. They learn enough about phrases to move them around. They are learning a lot. On the other hand, they don't have the obvious generalizations you get from a tree-structured models. What happens when we add a few more levels of linguistic embedding, make phrases longer, performance tanks. No structural generalizations a linguist thinks is fundamental.

5. ?: Bias in training, test set?

   LB: This is an important question. We used to curate data and make sure it's nice. Now data is large, people don't look at it. Data can be biased. It's easy to learn just

that. Detecting data is biased is hard! Suppose you want to learn X to Y. Being robust to all changes to X, Y is impossible; what changes do you want to be robust to?

6. MW: I am concerned about societal limitations. Translations between arbitrary languages, driverless cars. This has effects on society, political mess, because people have their own groups, listen to YouTube, and not to anyone else.

   LSTMs might cause white-collar unemployment. Two coasts are doing machine learning, and the middle is lost.

   LB: Social implications are huge.

   MW: We keep on inventing with gusto.

   RF: I don't think people would argue the laundry machine is horrible.

   How to reduce working hours so many people work less rather than some people benefit and other people starve. Solution is hastening society change.

   ?: Bill Gates had solution of taxing robots.

   ?: Machine help without taking human out of loop.

   ?: Start with fixed algorithm. Humans craft features and then learn. Learn features. Does the whole thrust of representation learning mean we understand less of what's going on under hood, hand over more to machines we don't understand?

   LB: There are two ways to validate gizmos. One is to open it and makes sure you understand everything, prove the code. We can't validate systems like this. The second way is to look at behavior. How do we validate a human driver? We don't open the skull. Observe behavior. Our relationship to engineering is changing from being able to opening and understanding how it works. Opening ML is like opening a brain.

   EH: Turing test: evaluate something functionally. Bayesian tried to put interpretability. Deep machines work better. We don't understand why they work. We udnerstand Statistical LT. There's a difference between understanding what it does and why it works.

   SA: Social impact: will ML become a dirty word like nuclear energy?

   RS: Not there yet.

   ?: Additional impact. Earlier they weren't good enough to do things we relied on humans to do, ex. whether to give loan to someone. Now we have machines we believe can do these things better than we can.

   EH: Already happened, just now it's more.

   RF: Many of these methods are performing very poorly in terms of accuracy. It's not clear what it means to be accurate. Ex. credit scores go back 30 years. This is a different problem. In many cases, it's wrong to use any kind of statistical technical. Statistical techniques used are crude, not good in some sense. It's a different societal problems that has to be addressed. In cases where it is societally accepted, it is possible to evaluate externally.

EH: We're not politicians. Something people have be doing: look at fairness of ML, make sure there's not biased towards certain groups.

MT: When we teach classes, explain these impacts.

7. Changes to fundamental paradigm of ML? (iid data, labels, etc.)

14 Continuous state machines and grammars for linguistic structure prediction (Noah A. Smith, University of Washington)

Linguistic structure prediction infers abstract representations of text, like syntax trees and semantic graphs, enabling interpretation in applications like question answering, information extraction, and opinion analysis. This talk is about the latest family of methods for linguistic structure prediction, which make heavy use of representation learning via neural networks. I'll present these new methods as continuous generalizations of state machines and probabilistic grammars. I'll show how they've led to fast and accurate performance on several syntactic and semantic parsing problems.

Collaboration with group at CMU.

The big idea is that we would like to take natural language system and convert them into dependency trees, useful for downstream tasks. Trust the linguists! There's some consensus that building trees is useful.

The way we've been thinking about building this trees has evolved.

1. Compiling Comp Ling: practical weighted dynamic programming and the Dyna language, Eisner, Goldlust, S.

   Everything can be boiled down to DP!

   This gives polytime algorithms.

   We're still using DP. (Segmental recurrent neural networks, Kong et al. 2016.) At the very top do dynamic programming (HMM). Except now put neural network to calculate scores.

   I don't want this to die, because this allows you to study your model without studying the inference procedure.

   Great example for neural parsing, Kiperwasser and Goldberg, TACL 2016.

   Use representation learning (new), get embedding of parts, and put structured prediction on top.

   You need strong independence assumptions to make this work. We want richer features, interdependence!

2. Turbo parsers, dependency parsing by approximate variational inference. Martins, S, Xing, Aguiar, Figueiredo, EMNLP 2010.

   Use techniques inspired by graphical models. You don't get polytime guarantees, but you get certificates.

3. Transition-based dependency parsing with stack long short-term memory. Dyer, Ballesteros, Ling, Matthews, S.

   Move to a different paradigm. Forget about optimality, rely on representation learning. Make greedy decision and do linear-time fast NLP.

Greedy parsing with a stack is not new. (Nivre, Scholz 04, Henderson 04) Stack starts out empty. Shift words onto the stack, pointer advances. You can do things like

- reduce left: take 2 things off top, combine/attach, and put back on. (Stack can hold trees.)
- reduce right.

Stack represents state of processing. Looking at stack in buffer, we see where we are and what can happen in the future.

Here it's nondeterministic. In a data-driven way, look at current state and decide what to do next. At runtime it becomes deterministic, take the best choice. You can do search if you want. But greedy works pretty well. There are no guarantees.

Eventually you get the tree as the one thing left on the stack, and you're done.

In 2010 I wouldn't have wanted to do this—there are no guarantees, it's local greedy search, learning is funny: when you train—default way to train assumes you've done the right decisions in the past. These are very fast at runtime, good for real-world applications.

Here's the interesting twist. I'll give you a view of a recurrent neural network and say how to use it.

Think of a RNN as a stack, where you only push. At the end you have a state which encodes the whole sequence. There's a relationship and the list of things you've accumulated—I've embedded the stack into a vector.

In the stack RNN, you can pop! If you query after popping, you get the stack from before. Discreteness goes into construction of graph.

Popping something X_2 off stack means moving back. The state for X_3 comes off X_1.

There are 2 possibilities:

1. train in supervised fashion assume gold sequence of pushes and pops, there is 1 computation graph for each sequence at training.
2. If there's uncertainty to whether you're pushing or popping at different times, there's no way to straightforwardly do backprop.

Default is have supervised sequence of action, not managing uncertainty. There are variations where we start moving in direction of exploration; we haven't figured it out. We have fully supervised data here.

If you have a stack RNN, you can use it to implement a parser. For the parser, we only pop off the stack RNN. Have a stack and action history (reduce right, shift, shift... we never have to pop off this).

On my stack I have pieces of tree and need a way to represent them. Use recursive neural net. Assume we have a word vector for each, and label for each attachment. Three things that make up one attachment are fed through feed-forward neural net that assembles into a

single vector fed through RNN. Do this recursively to vectorize the tree. What's important is that each subtree has head-word that has most of information.

Learning is an area where more exciting things will happen. End-to-end supervised training maximizes conditional log-likelihood of trees given words,

Chen and Manning 2014 do well on English and Chinese.

14.1 Variations

As you hoist work on the representation learning, and your design of the structure of problem, maybe you can do more interesting sharing.

1. Characters, not words

2. Many languages, one parser:

   One way to do sharing is to learn many languages together. If you can parse well in one language, you can use to parse in another in language if they share something.

   Requirements are multilingual word representations (words that mean roughly similar things in 2 languages are mapped to similar vectors). We need language ID and single annotation scheme across languages (universal annotation).

   Apply same model, and feed in language as part of input. Train on 6–7 languages. Nothing else changes. Learn to share as much as you can across languages.

   Word representations are shared. Learn to share patterns across languages.

   MAnyLanguagesOnePArser does well. We have 1 language held out and test on it and do pretty well. We have hope for doing NLP in languages we don't have treebank data for!

   You need to have gotten word embeddings. Get word vectors for both languages. Use languages that maps some of words. Then do CCA.

3. Add semantics:

   People glaze over when I talk about syntax trees. It's a relatively small (intellectual) step to go to semantic dependencies between words. Extend to build semantic graphs.

   This is not a tree, there are many cycles. This is what people in NLP really want. You can design transition functions to have a stack LSTM build this, perhaps alongside the syntactic parse.

4. Ensemble

5. Distillation

6. Grammars:

   RNN grammars. This is more widely useful. Stack LSTM can be used to learn state representation in a generative grammar.

   Ex. (S (NP The hungry cat) (VP meows))

This is often based on context-free grammars, maybe probabilistic. Go one step further. Condition on not just parent but everything you've done so far. Everything you open up you put on the stack. At some point go back up to nonterminal and close.

It's a different set of operations and substructures but parallel to what we had before.

It's a generative model.

$$\max_{\theta} \sum_{n=1}^{N} \ln p_\theta(y^{(n)}|x^{(n)}) \tag{13}$$

$$\max_{\theta} \sum_{n=1}^{N} \ln p_\theta(y^{(n)}, x^{(n)}) \tag{14}$$

You can train it discriminately or generatively. For reasons we don't understand, training generatively does much better.

You need the stack, you don't need to represent the other things. (The stack built, words produced, and action history are redundant but not totally. Words and actions can be derived from the stack.)

Other details: no POS or word embeddings, importance sampling for inference, composition function. State of art for Chinese phrase-structures, strong language models. RNNG learns head rules, an important idea in syntax.

Representation learning is great. Inductive bias (set up problem based on stacks and grammars) is great. Classical symbolic structures (stacks, grammars) offer sensible inductive biase. We're moving back to using classic paradigm at top to get guarantees. Questions of model design, loss functions, regularization are still important.

When building generative/joint model, when you sample how sentences look like, they look beautiful! It's better than LSTM's. But there are several decades where people have tried to pack syntax into data, but then someone comes up with a model with more data...

Why a stack and not another data structure? You need to pop off but don't need to look at the other end. You can imagine building other such computational structures; we didn't need it for NLP.

15 Adversarial Perceptual Representation Learning Across Diverse Modalities and Domains (Trevor Darrell, UC Berkeley)

Learning of layered or "deep" representations has provided significant advances in computer vision in recent years, but has traditionally been limited to fully supervised settings with very large amounts of training data. New results in adversarial adaptive representation learning show how such methods can also excel when learning in sparse/weakly labeled settings across modalities and domains. I'll review state-of-the-art models for fully convolutional pixel-dense segmentation from weakly labeled input, and will discuss new methods for adapting models

to new domains with few or no target labels for categories of interest. As time permits, I'll present recent long-term recurrent network models that learn cross-modal description and explanation, visuomotor robotic policies that adapt to new domains, and deep autonomous driving policies that can be learned from heterogeneous large-scale dashcam video datasets.

15.1 Adversarial domain adaptation

Joint work with Eric Tzeng, Judy Hoffman.

Limits of deep learning: few-shot learning, having predictive uncertainty, learning from new domains.

Autonomous vehicle trained to detect pedestrians. If you train with enough data and deep architecture, it works well, but if you take to a new environment, mileage may vary.

Domain adaptation is a classic problem statement. We have source domain with labels, do some classification tasks. Now things change, I don't have an analytic model of how they change. We can have poor performace even with best, highest-capacity supervised learning algorithms.

I'm interested in the case where you have no labels in target domain; you can't do few-shot learning. I assume you don't have paired/aligned instance examples (Siamese network).

What can we do without labels in target domain or constraints about pairs mapping together? The methods converge on the method of minimizing discrepancy.

Distance between domains might dominate distance between classes.

What can you do? Assumption underlying minimizing discrepancy: make decisions in domain invariant to transformation.

We take a broad notion of what domains are: differend distributions, labeling functions, modalities, feature dimensions... We want a representation blind to this bias.

A series of methods. Let's align means of these distributions. At least optimize representation so that mean of two groups of points are the same.

State of art approach: Even if I try to train a high-capacity discriminative classifier it can't tell the difference. Simultaneously trying to learn a supervised classifier in source domain. Have additional domain. Have a representation where you can't tell the difference between the domains. Penalize inversely according to success.

You get representaiton where you can't separate domain but can separate classes.

Start off training a domain classifier to predict the label.

Have 2 more losses.

$$\mathcal{L}_D(x_S, x_T, \theta_r, \theta_D) = -\sum_d \mathbb{1}[y_D = d] \ln q_d \tag{15}$$

$$\mathcal{L}_{conf}(x_S, x_T, \theta_D, \theta_r) = -\sum_d \frac{1}{D} q_d \tag{16}$$

"confusion" loss. Iterate between these two like GAN's. This is called adversarial discriminative domain adaptation.

CoGAN: two GAN's with coupled learning but unshared weights works quite well in some cases.

For classification, you get better results with dicriminative loss/front-end. There is no sharing and GAN adversarial loss.

Enforce: you can't tell the difference between 2 domains. This is reasonable.

Precursor to these ideas: focus on dataset bias. The standard is MSCOCO. But it's just as much an arbitrary dataset as Caltech 101! If you can play "name the dataset" game, there's a problem! One set of techniques work on Caltech 101, UIUC,... They had researchers come up and say what comes from what dataset. The good researchers could tell. If you can do this, you've overfit to properties of these datasets.

You should play the dataset game and lose.

This is part of bigger family of multitask or transfer learning. Transfer learning is the same/orthogonal? Here we have the same tasks in 2 different places, maybe a different camera or sensing modality.

If I train on Caltech101 and on ImageNet, and it classifier makes a mistake, would you be able to tell if it's a mistake based on Caltech101 or ImageNet? Don't segregate your supervised data. To a first approximation, take all labels, pool together, and train jointly.

I'm not saying to only train on one domain, just asking what to do in new domain?

What if there are lots more buses in one and lots more cars in another? Datasets have to be aligned. Class imbalance is one aspect of shift. (? But classifier is also built on top of representation!)

CoGAN works well on small domain shifts, not on large.

You don't even necessarily have the same dimension or modality. Train on one sensor and test on another. This speaks to robustness on convnets. There's enough similarity that you get better-than-chance performance.

15.2 Learning end-to-end driving models from crowdsourced dash-cams

Autonomous driving paradigms:

1. Learn affordances to predict state; apply rules or learned classic controllers.
2. Abandon engineering principles, learn "end-to-end" policy.

   I don't believe I should have a manual, arbitrary bottleneck, I'd like to learn the whole thing end-to-end.

I believe in both approaches.

How can visual sensing be robust to new environments, how to learn generic driving policies from diverse data?

We have BDD (Berkeley deep-drive) data set.

AlexNet, etc. gives label for whole image. Let's do this at every pixel, In-domain fully suprevised FCN (fully convolutional network). This is only 2–3 times the computation.

Cityscapes: label every pixel in scene with category labels: blue pixels are cars, dark red are bicycles, pedestrians, etc. When trained and tested on the same datasets, the algorithms do very well.

Domain shift: train on Cityscapes, test on San Francisco Dashcam does well. Car flying in tree, mess in front of image.

Every image in Cityscape has logo. They were smart enough to exclude from loss, but network is messed up where the logo was!

Also there are no tunnels in CityScapes. It makes bogus predictions; I don't have deep models with predictive uncertainty.

Blue pill: let's learn end-to-end driving.

I'm not excited about end-to-end learning without lifetimes of driving on vehicles. Define self-driving as ego-motion prediction. Predict trajectory of vehicle. Add recurrence to fully convolutional model. Have a loss on future ego-motion and semantic segmentation.

Diversity of dataset.

We can also do SLAM. Off-the-shelf methods failed; we did semantic filtering to make it work.

Take held-out sequences and predict what humans will do. Cf. character model, it doesn't have model of human intentions.

15.3 Vision and language: learning to reason to answer and explain

Original domain adaptation papers came from language.

I'm interested in the fusion of language and vision. How to get explainable models, models that have and reason about compositionality.

We're interested in explainable AI: ex. going from images to captions. You can jointly train language and image models.

We're interested on text not just conditioned on image, but conditioned on image and task. Ex. I'd like the car to tell me why it turned left.

Two approaches.

1. Implicit: predict what story a human would have told. (Collect rationale test and train in fully supervised fashion.)

2. Explicit: unpack what specific model is doing.

Use an attentive model to do well. What was the model looking at when it gave the answer. "What is the woman feeding the giraffe?" "What color is the shirt?" "What is the hairstyle?" It changes attention based on question.

We have models that not only generate text, but also attention.

Neural modular networks: instead of single homogeneous network, have an explicitly compositional network to process images and answer a question. Create parse tree, routines, etc.

Compiling using parsers, answer specific questions. A neural modular network takes a sentence, compiles into parse tree, create neural models convert to semantic parse.

CLEVR, new dataset.

Learning to Reason. Take neural modular networks and learn end-to-end fashion the parsing. We're no longer using a side parser! It learns to reason by learning layout policy, and using it to answer question.

How many other things are of the same size as the green matte ball? It compiles plan: find something, relocate, count. It grounds that the thing it should find is the green matte ball. Count is grounded on "thing."

"Does the blue cylinder have the same material as the big block on the right side of the red metallic thing?"

16 Evaluating Neural Network Representations Against Human Cognition (Tom Griffiths, UC Berkeley)

How do cognitive scientists think about representation?

I'll focus on the particular problem of categorization: how to tell whether something is a dog or cat? How do people represent categories?

History of cognitive psychology:

1. Logic: At the beginning, people though we were using logic, cat is "small, furry, domestic, carnivore". Dogs also satisfy this, which means this particular rule is wrong, but there can still be some other rule that works. People had faith in this up to 1970's.

2. Eleanor Rosch showed that for many natural categories. Think in terms of family resemblance structure.

   How to represent this? How to represent categories in a way that captures family resemblance structure?

3. Prototypes: you have a prototype cat that you compare against. (Posner Keele 1968, Reed 1972)

   A few years later, people figured out another approach.

   People are better at classifying for things that look like the prototypes.

4. Exemplar: store every instance (exemplar) in memory. Compare new object to every instance of cat and dog you've seen. [5]

Prototype corresponds to linear classifier. Exemplar is equivalent to doing Bayesian classification using kernel density estimator.

Which gives best accounts of human behavior. People can learn really complex stuff.

Nosofsky and McKinley 1994. Two mixtures of several Gaussians. You can do classifications from exemplars, not prototypes. After 4000 trials, people could learn the complex boundaries. This supports the exemplar model.

What is similarity? A decreasing function of distance in psychological space. Roger Shepard did a lot of interesting work revealing representations people had. You can use this to figure out right similarity for domain, given people's similarity judgments.

[5] Is this equivalent because of representor theorem? If you allow rich enough feature space, this is equivalent. Do these predict differences of behavior, or just representational detail?

Multidimensional scaling: have people judge similarity between colors. Given similarity between points, reconstruct spatial representation of domain. Give you back a color wheel. It's the representation that makes sense psychologically.

This approach was popular but not universally accepted.

Amos Tversky did work on similarity. Do people's similarity judgments violate the distance axioms? He systematically showed violations of each axiom.

1. Ex. They are asymmetric S(NKorea, China)>S(China, NKorea), S(ellipse, circle)>S(circle, ellipse).

2. Triangle inequality. We can find similarity judgments that violate this: gas lamp, moon, soccer ball. Gas lamp is nothing like a soccer ball.

Density hypothesis: You're more likely to compare the outlier thing to prototypical thing.
Objectives.

1. Use psychological methods to investigate representations in neural networks.

   These were methods used by psychologists were developed for insight into learning system with opaque representations.

2. Explore corresp betweenhuman representations and those of neural networks

3. Leverage neural networks to get deeper insight into human cognitions.

Construct simple stimulus to get knowledge of features (ex. Gabor features). This is a long way from cats and dogs. Neural networks give us ways of classifying tigers, etc., and may give us better, more realistic test of models.

16.1 Images

Joint work with Peterson, Abbott, Battleday.

The canonical method is convolutional neural networks.

Take representations and ask how well hidden layer representations correspond to psychological representations. We use similarity judgments as tools.

Take similarity judgments from people and models (inner products in hidden layers) and see how well they match. Human judgments are pretty stable.

Collect pairwise similarity judgments for 120 images of animals using Amazon Mech Turk (71400 judgments).

We get pretty good similarity. Clusters of human judgments correspond to different taxonomic groups. Similar representations appear in the deep representations, but they're not pulling out clusters in the same kind of way.

Hierarchical clustering: deep representations preserve local structure but don't preserve high-level structure (classes of things).

Train on hierarchical structure, do you get something closer to human similarity judgments? What if you add in word similarity? Try to produce something closer to what people do.

Is there enough information in these reprsentations to already reconstruct what human representations are like?

At last layer, try to have a representation useful for many different classification tasks. Is this representation rich enough that we can use it to reconstruct human similarity judgments?

A basis for human representations: can we reweight representation to better capture human similarity judgments? Use classic psychological model (Tversky 1977)

$$S = FWF^T$$

This turns reweighting into a regression problem.

We get a big improvement in performance.

Hierarchical representation works on the transformed representations.

We've reproduced on many different domains: fruits, furniture, vegetables.

When we take those representations and apply the model to other images, can we predict human behavior on those images? Take another set of 120 images of animals that were't used in setting the transformations, get a similarity matrix, do clustering, use it to define 1–4 clusters, train humans to classify clusters, see how long it takes them to learn the clusters.

Can we use these representations as a way of conducting strong tests of psychological models with naturalistic stimuli? Use a dataset of 300,000 human judgments of whether a picture contains a bird or a plane (Project Superman). As computer vision problem, you're interested in producing ground truth. For humans, there are significant deviations. We're trying to predict human judgments.

With more complex stimuli, there's very little difference between performance of prototype and exemplar models. The reason is pretty clear: this is a situation where there is a complex stimulus, but you are constructing a complex representation where you can apply a simple classifier. A simple prototype will do well in this space.

By focusing on simple stimulus, the only way to produce complex behavior is complex boundaries, but for naturalistic images, we can have a complex representation where we don't need the complex classifier.

16.2 Text

I focus on vector space models.

Each word is point in vector space, you can linearly combine them, get out analogies.

Objections to spatial representations: Semantic association has same properties: asymmetry.

Word association: ex. planet→Earth. There are asymmetries, (beer—keg) $>$ P(keg—beer). There are also violation of triangle inequality: asteroid, belt, buckle. This is potentially problematic for vector space models.

But with more modern vector space models comes a more probabilistic view, $\mathbb{P}(w_2|w_w) \propto \exp(w_2^T w_1)$. This can predict human associations.

When estimated from small number of words, topic models still outperform vector space models. But vector space models can be estimated from very large corpora and then beat topic models.

Conditional probabilities computed from vector space models adequately capture asymmetries and violations of triangle inequality.

Think about the parallelogram model for analogies, going back to Rumelhard and Abrahamson (1973). Try to evaluate using human data. If you have an analogy of this form (man:woman::king:queen), you can find the fourth word by completing the parallelogram. They did MDS of human similarity judgments.

Previous evaluations have focused on automatically generated relational pairs. We wanted to generate a new database across a wide range of relations. Starting point is SemEval2012 task 2. Class inclusion, part-whole, similarity, contrast, attribute, non-attribute, case (soldier-gun), cause, space-time, reference. We ask people to generate more relations of similar type.

There are types where the linear transformation parallelogram does capture the relations. Other don't (contrast, similar). It's represented quite differently geometrically.

Have people generate ratings for these different types. Generate pairs of pairs and write similarity between pairs of pairs (who similar the relation is). How well can we predict those relation similarities? We find a clear variation between relation types on how well they are predicted. Similarity and contrast are relatively poorly predicted.

This reveal how 1 particular relation corresponds to 1 aspect of human judgment.

Do same objections apply in analogies? They are arguments against using a vector space model to capture everything.

- hairdresser:comb::pitcher:baseball
- pitcher:baseball::hairdresser:comb.

People think the first is better. (Baseball is though of as object in first, game in second.)

Collect judgments for 500 relational pairs in both directions. Effect of priming? What frame is activated?

Example: nurse : patient::mother : baby :: frog : tadpole. This is relational violation of triangle inequality.

lawyer:books::chemist:beakers::librarian:books. Encourages you to think about differences in which lawyers and librarians relate to books.

1. Off-the-shelf deep networks do surprisingly well, but miss important conceptual structure.

   Look at different tasks, training regimes?

2. Learned representations can be used to make high-fidelity predictors of similarity that can be used in models of human cognition.

3. Lots of open questions: training, data, architectures...

Q/A:

- Using other metrics? There is psych literature on L^1 vs. L^2.
- Things that predict better perform better. The higher in network, the better the correlations.

- Asymmetries and lack of triangle inequality are superficially similar to KL divergence. This is similar to density hypothesis.
- Similiarity could be a nonlinear function of the metrics.
- Analogies that don't give linear relationships. Some examples unfair? Linear when clear semantic relationship.

  Similarity is better captured by a constraint on distance between points.
- Text: "Belt" has 2 different signals. Image example more fundamental? "B" item in A:B::B:C changes the sense vs. the features.
- Can you use image similarity to improve accuracy?

  You get slight defecit in performance using human representation transformation. Original model overfitting tasks. Human representations also do many other tasks. I think there's places where it's relevant to have these transformations.

17 Representation learning for reading comprehension (Russ Salakhutdinov, Carnegie Mellon University)

In this talk I will focus on discussing deep learning models that can find semantically meaningful representations of words, learn to read documents and answer questions about their content. First, I will introduce the Gated-Attention (GA) Reader model, that integrates a multi-hop architecture with a novel attention mechanism, which is based on multiplicative interactions between the query embedding and the intermediate states of a recurrent neural network document reader. This enables the reader to build query-specific representations of tokens in the document for accurate answer selection. Second, I will next introduce a two-step learning system to question answering from unstructured text, consisting of a retrieval step and a reading comprehension step. Finally, I will discuss a fine-grained gating mechanism to dynamically combine word-level and character-level representations based on properties of the words. I will show that on several tasks, these models significantly improve upon many of the existing techniques.

Joint work with with Bhuwan Dhingra, Zhilin Yang, Yusuke Watanabe, Hanxiao Liu, Ye Yuan, Junjie Hu, and William W. Cohen

I'll show a trick to train LSTM's more efficiently.

17.1 Multiplicative and fine-grained attention

We have a who-did-what dataset. Most existing systems are far from human-level performance.

The cloze-style Q/A task involves tuples (d, q, a) where d is document, q is question over contents, and a is answer. The answer comes from fixed vocab A.

Task: given (d, q), find $a \in A$ answering q.

Half of work is from DeepMind. A lot of models rely on recurrent networks,

$$h_t = \phi(Uh_{t-1} + W_t + b),$$

where h_t is hidden state at time t, x_t is input at time t, and ϕ is nonlinearity.

We replace with elementwise multiplication

$$\phi(Uh \odot Wx + b)$$

or

$$\phi(\alpha \odot Uh \odot Wx + \beta_1 \odot Uh + \beta_2 \odot Wx + b).$$

This is easer to optimize.

If we train regular RNN, we get saturation effect. When you apply tanh, they go to sides, and it becomes hard to optimize. For the multiplicative model, it sits in the middle, in the linear regime. We have much faster convergence.

If I take a derivative with respect to W in the multiplicative system, x can have direct effect. (Hidden state can affect gradient.) The way gradients are flowing. Gating has big impact on performance.

Off-the-shelf GloVe words better than word2vec.

Use bi-directional LSTM/GRUs to encode both document and query.

- Use attention mechanism,

$$\alpha_i = \text{softmax}(Q^T d_i) \tag{17}$$
$$\widetilde{q}_i = Q\alpha_i \tag{18}$$
$$x_i = d_i \odot \widetilde{q}_i. \tag{19}$$

  Use elementwise multiplication operator to model interactions between d_i and $\widetilde{q}_i$.

- Multi-hop architecture.

- Create a softmax. Aggregation happens at output.

$$P(c|d,q) \propto \sum_{i \in \mathbb{I}(c,d)} s_i.$$

There's so much work done in this space. Gating mechanisms do reasonably well. The pace is very fast.

When we look at attention models, we sometimes see interesting things. On the x-axis is the query, on the y-axis is the document.

Words vs. characters:

- Word-level representations are good at learning the semantics of the tokens.

- Character-level representations are more suitable for modeling sub-word morphologies (cat, cats).

- Hybrid word-character models have been successful. A commonly used method is to concatenate these two representations.

Fine-grained gating:

$$h = g \odot c + (1 - g) \odot (Ew).$$

Gating happens as elementwise multiplication. If we use scalars it doesn't work as well. Here

$$g = \sigma(W_g v + b_g).$$

We use a bunch of prior knowledge to decide whether to use word or character.

High gate values correspond to character-level representations. Low gate values mean word-level representations. Low-frequency words and proper nouns use character-level representations.

17.2 Incorporating knowledge as explicit memory for RNN's

LAMBADA dataset: answer who. Coreference resolution is the standard technique. People in NLP are familiar with this. A lot of existing models don't take this into account.

Can we learn from data and incorporate coreference dependency parse, etc. other NLP info that's useful.

Introduce coreference links, hyper/hyponyny links like skip-connections. This is like having multiple RNN chains with skip-connections. Construct parse matrix. Construct in such a way so it's a RNN model with additional links. The directionality matters. Move forward and backwards. We can take the question into account.

Mathematically, putting a link means that at every hidden state, incorporate info from all links coming in.

It's an interesting and bad results. For babiQA, go from 75.2% to 91.3%. For others, we don't get significant gains.

Q: how many objects is Sandra carrying?

GA model defaults to 1. Adding structure helps.

17.3 Generating text

Leverage unlabeled text for question answering.

It's easy to find an answer. Use POS/NER/parsing to extract possible answer chunks. Anything can be answer. Assume answer is available.

Given an answer, can you generate a question for it?

We have labeled data (p, q, a) and unlabeled data p, a. Generate with seq2seq with copy mechanism. Maybe we can combin to train QA model. One piece that's difficult is to generate the question.

Some simple baselines work well.

1. Given answer, look at some surrounding context. This works remarkably well (in low data regime).

2. GAN: One objective function differentiates between true or fake question, the other tries to answer.

   One problem is that it's hard to generate good-quality questions, and harder to discriminate between real and fake.

We do something different, generative domain-adaptive nets.

Have a data tag that says whether data is coming from true distribution. Objective is not to generate coherent sentences, but whethever helps answer the questions.

We see gains in the low data regime. In high data regime, we still see edge but gap closes.

We started thinking about: is there a way to generate better looking questions?

We looked at VAE's. We have $z \sim N(0, I)$ and a code $c \sim \text{Bernoulli}(\mu)$.

We can fix z and change code to change sentiment, tense, etc.

We can fix the code and generate the latent state. Generate a sentence which is negative and past, etc. (Some sentences are new.) How do you evaluate quality? There's no good answer.

I can flip code and regenerate sentence: take a negative sentence and make it positive, etc.

Who-did-what dataset: humans cannot answer the question by looking at last sentence. Our model does better on these types of questions.

18 Tractable Learning in Structured Probability Space (Adnan Darwiche, UCLA)

Over the past few decades, various approaches have been introduced for learning probabilistic models, depending on whether the examples are labeled or unlabelled, and whether they are complete or incomplete. In this talk, I will introduce an orthogonal class of machine learning problems, which have not been treated as systematically before. In these problems, one has access to Boolean constraints that characterize examples which are known to be impossible (e.g., due to known domain physics). The task is then to learn a tractable probabilistic model over a structured space defined by the constraints.I will describe a new class of Arithmetic Circuits, the PSDD, for addressing this class of learning problems. The PSDD is based on advances from both machine learning and logical reasoning and can be learned under Boolean constraints. I will also provide a number of results on learning PSDDs. First, I will contrast PSDD learning with approaches that ignore known constraints, showing how it can learn more accurate models. Second, I will show that PSDDs can be utilized to learn, in a domain-independent manner, distributions over combinatorial objects, such as rankings, game traces and routes on a map. Third, I will show how PSDDs can be learned from a new type of datasets, in which examples are specified using arbitrary Boolean expressions. A number of case studies will be illustrated throughout the talk, including the unsupervised learning of preference rankings and the supervised learning of classifiers for routes and game traces.

We will be learning from prior knowledge use symbolic reasoning. We will identify 3 areas for symbolic reasoning and ML.

Joint work with van der Broock, et al.

18.1 Structured probability spaces

Consider a department with several (4) courses. We have prior knowledge: department requirements and prerequisites. I need to learn from both of these inputs.

Unstructured space has all $2^4 = 16$ elements. The prior knowledge says, e.g., 7/16 instantiations are impossible. We propose a statistical model which assigns 0 probability to instantiations that violate the constraints.

People deal with this all the time.

1. Ex. identify players in sports videos. Handcraft in the fact that a person can't be in 2 places at once.

2. Language: at least 1 verb in each sentence; if modifier is kept, subject also kept...stem

3. NASA Ames power system takes into account physics.

4. DeepMind reasoning about London Underground.

People deal with this by ignoring, handcraft models, use specialized distributions, find non-structured encoding, trying to learn constraints... There are no statistical ML boxes that take constraints as input!

How to systematically learn distributions over combinatorial objects?

18.2 Specification language: logic

Use boolean constraints.

For example, consider rankings. People have dedicated distributions. Introduce boolean variables A_{ij}, where A_{ij} says i is at position j. One constraint: each item i assigned unique position $\bigvee (A_{ij} \wedge (\bigwedge_{k \neq j} \neg A_{ik}))$.

We can also have a structured space for paths, graphs$\supseteq$trees$\supseteq$labeled trees$\supseteq$parse trees.

The next story has 2 components.

First convert into logical circuits. The circuit must satisfy some additional requirements. If I ask you: is this space empty, this is SAT, NP-complete. The first property will make this linear time. Counting is #P complete; the second property will make this linear time. The last property allows us to do learning. The secret to tractability is symbolic.

1. Decomposability: look at every $\wedge$ gate. Things leading into it are independent. Dynamic programming.

2. Determinism: for any input, no $\vee$ gate has more than one high (1) input. (Mutual exclusivity.)

3. Sentential decision diagram (SDD):

   This allows equivalence testing in poly time.

The computational bottleneck is in getting this circuit representation.

18.3 PSDD

Now we move to PSDD, probabilistic SDD. Induce a distribution. Do this by putting local distributions on OR gates. This is a distribution over structured space induced by the circuit.

How to find the probability of a given assignment? If the circuit evaluates to 0, it's outside the space. Otherwise, trace the high wire down and multiply the numbers you encounter.

These probabilities are normalized, even though we only ensured local normalization.

We can read probabilistic independences off the circuit structure.

MAP inference, conditional probabilities, and sampling are doable efficiently.

But I want to learn these things.

PSDDs are arithmetic circuits. These are called SPNs (sum-product networks).

18.4 Learning PSDDs

Parameters are interpretable. Subcircuits are interpretable, ex. student takes course L, student takes course P, probability of P given L. This is invariant to any change you make in the circuit.

We can do closed form max likelihood from complete data. Do one pass over data to estimate parameters.

What is the structured learning problem? You don't learn the circuit, you compile it from the constraints. Which circuit do you choose? It defines parameter space. We have only been using the naive way of learning by minimizing its size. It works well. Why are we getting away with minimizing the size?

Compile constraints to SDD by using SAT sovler (SDD library). Search for structure to fit data (ongoing work).

What else does this allow you to do? Comparing the naive learning approach with mixture-of-mallows (for permutations). Mallows if unimodal, we have to do 20 mixtures to get comparable results.

What happens if you ignore constraints? It's much worse when you don't have enough data; it starts catching up with more data but doesn't completely catch up.

Induce distribution over outcomes of game. Outputs of game are structured probability space. We did this for 3 types of games. We integrated with Naive Bayes. Each attribute is structured space.

Ex. Given outcomes, try to decide whether player is optimal, heuristic, or random. I can do naive Bayes when features are coming from structured space.

Quantify $X|C$ using PSDD.

18.5 Structured datasets and queries

A complete dataset: each row is a full instantiate. Incomplete: some values are missing.

I can deal a more general type of incomplete dataset, where each row is a boolean formula that is true (ex. $X = Y$). Conditioning on arbitrary boolean expression is not part of the ML repertoire. You need to efficiently condition on boolean expression. If you can compile in way that is compatible with the master circuit, you can condition in quadratic time.

A classic incomplete dataset: people have to partially fill in rows. We can have more general incomplete datasets, like partial rankings.

Ex. Movielens dataset.

The fact that we were just minimizing sizes of circuits: you can interpret circuit as doing recursive matrix decomposition over space. You are in a sense identifying features.

1. Statistical ML, probability
2. Symbolic AI, logic
3. Connectionism, deep.

PSDD is in intersection.

Version presented is parametric: fix circuit (and hence which parameters) before seeing data.

Now we're trying different circuit structures and regularization. It's tricky.

Logical assertions. How am I sure that will give boolean circuit? Use SDD library. There will always be a circuit, the question is how big.

What if I have incomplete knowledge of constraints. Once you have circuit, queries are cheap. Encoding part of game: domain-specific part.

Constraints might be loose; real solution might be much better. Can use put a deep NN inside, subject to these constraints? NN in structured space?

Can you do this for spanning trees? We can have routes under simple path assumption.

19 Spotlight Talk: Convolutional Dictionary Learning through Tensor Factorization (Furong Huang, UC Irvine)

Tensor methods have emerged as a powerful paradigm for consistent learning of many latent variable models such as topic models, independent component analysis and dictionary learning. Model parameters are estimated via CP decomposition of the observed higher order input moments. We extend tensor decomposition framework to models with invariances, such as convolutional dictionary models. Our tensor decomposition algorithm is based on the popular alternating least squares method, but with additional shift invariance constraints on the factors. We demonstrate that each ALS update can be computed efficiently using simple operations such as fast Fourier transforms and matrix multiplications. Our algorithm converges to models with better reconstruction error and is much faster, compared to the popular alternating minimization heuristic, where the filters and activation maps are alternately updated.

Joint work with Anima Anandkumar

Feature learning is the cornerstone of ML. Map words into vectors so machines will understand.

We want to find efficient representations based on sparsity, low dimensional structure, group invariance.

Is there a principled approach guaranteed to learn good representations?

Dictionary learning is a well-posed problem to find representations.

Posit that the observed signal is a linear combination of dictionary. Find hidden dictionary and feature representation $h^{(i)}$,

$$x^{(i)} = Ah^{(i)}.$$

For image DL thinks fhat two objects in the same location are more similar, rather than the same object in different location. This is missing spatial invariance.

We want spatial dictionary learning that has shift-invariant dictionary and location encoded representation. Convolutional model satisfies both requirements.

$$x^{(i)} = \sum_l f_l * w_l^{(i)}.$$

People use alternating minimization: gradient-based method is nonconvex and has spurious local optimal, There is additional symmetry, and can be prohibitively expensive.

Reformulate as multiplication of circular matrix: columns are filters and shifted version

$$x^{(i)} = \sum_l f_l * w_l^{(i)} = \sum_l Cir(f_l) w_l^{(i)}.$$

How do we do dictionary learning?

Use tensor decomposition for DL. Use inverse method of moments.

$$\mathbb{E}[x \otimes x] = \sum_l \sigma_l^2 A_l \otimes A_l.$$

Do eigenvalue decomposition on this matrix? Matrix decomposition recovers subspace, not actual model. You need to go to 3rd moments.

Ex. matrix orthogonal decompositions: if there is no eigenvalue gap there is no uniqueness of eigenvector decomposition.

For tensor, if you have orthogonal decomposition, even if you don't have eigenvalue gap, decomposition is unique.

$$T(I, I, u) = \sum_i \langle u, u_i \rangle \, u_i \otimes u_i.$$

(This is undercomplete.)

Tensor decomposition uniquely identifies A up to permutation.

We want to do spatial dictionary learning. There are some difficulties. If you don't consider this as a convolution, it's overcomplete, number of columns is greater than data dimension. There are additional symmetric solutions.

We have nice properties for this specific application. The circulant matrix is a simultaneously diagonalizable group. Circulant matrix is simultaneously diagonalizable. Diagonal matrix is Fourier transform of filters.

Circulant projection $\min_f \|M - cir(f)\|_F^2$ has closed form. Do tensor decomposition with circulant projection.

Naive loss is $\min_F \|\text{Cumulant} - \mathcal{F}\Lambda(\mathcal{F} \odot \mathcal{F})\|_F^2$. Do asymmetric relaxation $F\Lambda(H \odot G)^T$. We can solve for F by least squares.

Problem: pseudoinverse $(\text{Cumulant}(H \odot G)^T)^+$. Naive implementation is $O(n^6)$, n filter length, and unstable. Use simultaneous diagonalizability property. Use FFT. Get running time $O(\ln n + \ln L)$ per iteration with $O(L^2 n^3)$ threads.

Compared to AM, Tensor methods works better when number of samples is large; if you exploit maximal degree of parallelism, tensor is better even with stochastic AM.

Do synthetic experiments. Generate data based on spatial dictionary model.

AM means: alternate between filter and activation map in each sample.

Summary: method of moments for learning dictionary elements; invariances handled efficiently.

Future: broader family of invariances (rotation, scaling). Analyzing optimization landscape for invariant models via tensor methods. Understand shallow convolutional net better.

Q/A:

- Is convergence local/global, and how does it depend on initialization? We use AM as baseline. Problem is nonconvex. We don't have global convergence guarantee.

  FT are complex numbers, you have to analyze landscape in complex space.

- Initialize using samples, overparametrization? We picked the best random initialization.

20 How to escape saddle points efficiently (Praneeth Netrapalli, Microsoft Research India)

We show that a perturbed form of gradient descent converges to a second-order stationary point in a number iterations which depends only poly-logarithmically on dimension (i.e., it is almost "dimension-free"). The convergence rate of this procedure matches the well-known convergence rate of gradient descent to first-order stationary points, up to log factors. When all saddle points are non-degenerate, all second-order stationary points are local minima, and our result thus shows that perturbed gradient descent can escape saddle points almost for free.

Gradient descent for $\min_x f(x)$: for l-smooth functions, converges to points where gradient is close to 0: $\|\nabla f(x_t)\| < \varepsilon$ in $t = O\left(\frac{l(f(x_0)-f^*)}{\varepsilon^2}\right)$. Convertes to ε-first order stationary points. They can be local minima or saddle points.

They can converge to either but these are very different in practice. Local minima are often good, but saddle points are very poor compared to global minima.

Local minima are much more desirable to converge to than saddle points. Gradient descent can converge to saddle points. How to escape saddle points efficiently?

Ge2015: by adding noise, gradient descent escapes saddle points. This requires $\text{poly}(d)$ iterations. Gradient descent comes close, after $\text{poly}(d)$ iterations it escapes.

Can we do this more efficiently?

Hessian Lipschitz: $\|\nabla^2 f(x) - \nabla^2 f(y)\| \le \rho \|x - y\|$. 2nd order stationary point: $\|\nabla f(x)\| \le \varepsilon$ and $\lambda_{\min}(\nabla^2 f(x)) \ge -\sqrt{\rho\varepsilon}$. (Nesterov Polyak 2006)

Perturbed gradient descent finds ε-second order stationary point in $t = \widetilde{O}\left(\frac{\ell(f(x_0)-f^*)}{\varepsilon^2}\right)$. It converts to second order stationary point in essentially same amount of time as GD to first order stationary point.

Keep doing GD, and once in a while it does perturbation (when perturbation condition holds). It kicks in when gradient at current point is small and we haven't added noise for a while.

Proof idea: two cases.

1. $\|\nabla f(x_t)\| > \varepsilon$. Then descends.
2. $\|\nabla f(x_l)\| \leq \varepsilon$. There exists a descent direction.

   Let S be set of points around saddle point from where gradient descent does not escape saddle point. Key technical result: Vol(S) is small.

   Consider adding a negative eigendirection to get w. If u does not escape, w escapes. For every point there are many other points that escape. (Divide in lines; it intersects set in small segment.)

Gradient descent converges to first order stationary points, perturbed GD converges to second order stationary points, depends only logarithmically on dimension.

Further results using local structure:

- strict saddle property: every saddle point has strictly negative eigenvalue. Then all second-order stationary points are local minima.
- Local strong convexity. Get local geometric convergence once we are in neighborhood.

Open:

- is randomness in beginning sufficient?
- Do momentum methods help accelerate for nonconvex problems?
- Extensions to stochastic case.

21 Re-Thinking Representational Learning in Robotics and Music (Sham Kakade, University of Washington)

Let's look at representational learning in 2 domains, to build broadly applicable methods.

1. Control: what are the representations being learned by deep learning?
2. Music: In the low level, there are similarities to vision. In the mid level there are similarities to language. Use MusicNet, a new labeled dataset.

21.1 Control

MuJoCo is a realistic physics simulator popular in robotics. Swimmer, hopper, walker, cheetah, ant. You want these thing to move, stay upright, do it fast. Sometimes for planning for robotics they use the simulator to plan internally. There's another question of whether these tasks are good, but they have been studied a lot.

One way to attack MDP's is using gradient methods.

$$g = \mathbb{E}_r \left[\sum_{t=0}^{r} \nabla_\theta \ln \pi(a_t|s_t;\theta) \left(\sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} - b(s_t) \right) \right]$$

Compute unbiased estimates with Monte Carlo sampling. Typically you need variance reduction to get this to work.

People tend to use a certain preconditioner. A natural preconditioner is

$$F_\theta = \mathbb{E} \left[\sum_{t=1}^{T} \nabla_\theta \ln \pi(a_t|s_t;\theta) \nabla_\theta \ln(a_t|s_t;\theta)^T \right],$$

Fisher information. Think about it as poor man's Gaussian method. Then

$$\theta_{k+1} = \theta_k + \alpha \widehat{F}_{\theta_k}^{-1} \widehat{g}.$$

There are 2 state-of-the-art approaches.

- Trajectory optimization, Todorov 11.

  Monte Carlo rollouts and choose best action based on rollout. At test time you still need model and rollouts; it can be costly.

- Deep learning (Schulman et al. 15). Model free methods give compact policies.

  Optimization: use TRPO (similar to natural policy gradient).

  Representation: deep architectures.

We want to move to harder robotics tasks. Variance is an issue.

- Optimization: stick with natural gradient.
- States: about 10–30 dimensions. Relative joint angles, centers of mass.
- What policy parametrization should we try? Linear policy does as well as deep net! They aren't as hard as people think!

Optimally we'd like to decouple study of optimization and representation, but they're coupled together.

- Linear policy: $a_t \sim N(Ws_t + b, \Sigma)$ (need stochasticity to take natural gradients)

- RBF policy (random Fourier): $\sin\left(\frac{\sum_j P_{ij} s_t^{(j)}}{v} + \phi^{(i)}\right)$, $P_{ij} \sim N(0,1)$, $\phi^{(i)} \sim U[-\pi, \pi)$, $a_t \sim N(\widetilde{W} y_t + \widetilde{b}, \widetilde{\Sigma})$.

RBF is better than linear; they are competitive with NN. Number of parameters is proportional to training time. This set of benchmarks does not resolve the representational question.

How does stability compare?

These performance numbers aren't what we're really interested in. The hard part is getting off the ground; once you have in a stable state around the right answer, you can tweak.

(NN should be as good if everything is convex—but it's not.)

These all work, but we care about stability; do they work as well under perturbation? Everything breaks in presence of perturbations! For all of these tasks, none of them work after thwacking it. Ex. If we rotate the snake it doesn't know how to move. (After running for a while it forgets how to get up.) We need reward for getting up? Let's just try to initialize in diverse states.

Why are these brittle to perturbations? By design, agent gets to see data in only some parts of the state space. The learned policy is trajectory-centric/local.

Idea: remove termination and use diverse state distribution. A fraction start on the ground. RBF's are the best here. Linear isn't all that bad.

We're making a lot of progress; there are issues but the current benchmarks don't resolve these issues. We want not to do rollouts at deploy time.

21.2 Music

Large labeled datasets have been key to success.

In music labels would be which notes are played at which times.

Annotation is difficult for rich pieces (fast, rich, metric structure, polyphonic, multiple instruments). Score is crude labeled data. (Cf. ask people to read essays. What we want is at each time step, what is being played.) Do music to score alignment.

MusicNet is a new labelled dataset with > 30 hours of aligned polyphonic music, > 1 million temporal labels.

Music2Score alignment problem. Output: tuples of (start, end, instrument, note, measure, beat, note value).

We estimated error rate $< 4\%$ by sampling and noting errors.

If we just do a data-driven method what do we learn? Low-level features are spectrogram features. Now we need something like a language model to put these together.

Low-level is convnet. We haven't had success with RNN's, optimization is hard. Conditioning is poor, you have low-frequency to high-frequency filters.

Open questions: style transfer. Complete the picture/music.

22 Representation Learning of Grounded Language and Knowledge: with and without End-to-End Learning (Yejin Choi, University of Washington)

Representation learning through an end-to-end learning framework has shown increasingly strong results across many applications in NLP and computer vision, gradually removing the reliance on human engineered features. However, end-to-end learning is feasible only if high quality training data is available at scale, learning from scratch for each task is rather wasteful, and learned representation tends to focus on task-specific or dataset-specific patterns, rather than producing interpretable and generalizable knowledge about the world.

In contrast, humans learn a great deal about the world with and without end-to-end learning, and it is this rich background knowledge about the world that enables humans to navigate through complex unstructured environments and learn new tasks efficiently from only a handful of examples.

In this talk, I will present our recent efforts that investigate the feasibility of acquiring and representing trivial everyday knowledge about the world. In the first part, I will present our work that focuses on procedural language and knowledge in the cooking recipe domain, where procedural knowledge, e.g., "how to bake blueberry muffins", is implicit in the learned neural representation. In the second part, I will present a complementary approach that attempts to reverse engineer even such knowledge not explicit in language due to reporting bias—people rarely state the obvious, e.g., "my house is bigger than me"—by jointly reasoning about multiple related types of knowledge about actions and objects in the world.

Intelligent communication requires reading between the lines: understanding what is said and not said. Ex. "Cheeseburger stabbing" means someone stabbed someone else over a cheeseburger.

Ex. "bake for 30 minutes": we fill in the missing information (bake what? where?).

Types of knowledge:

- Propositional knowledge: encyclopedic, everyday (commonsense)
- Procedural knowledge.

  Compared to encyclopedic knowledge (under information extraction framework), little effort has been made in learning these other types of knowledge.

Representation learning of commonsense knowledge? Only one way to find out! What kind of knowledge is learnable? What paradigm makes sense (neural/symbolic, end-to-end or not)?

General theme: machine learned representation is better than human engineering features.

End-to-end is successful if a clean curated dataset exists at a large scale: SQuAD, sentence summarization. We can't do end-to-end for some tasks (ex. sonnet composition).

How about for learning background knowledge?

I want a robot to cook breakfast for me. Natural language instructions are grounded by nature. Accomplish a goal by interacting, biology wet lab instruction, How to do X (change engine oil).

Unique challenges of procedural language:

- traditional parsers have trouble with imperatives.
- Elided arguments are common. (Bake for 30 minutes.)
- Reference resolution—identities change over time. (Whisk eggs. Add flour. Fold sugar into wet mixture.)

22.1 Procedural language and knowledge

We want to produce action graphs: model flow of ingredients as DAG. "In a large bowl, stir together milk, egg, and oil. Add flour to the wet mixture." Wet mixture refers to stirred mixture. "Add flour."

1. Predicate argument structure with implicit arguments. This is semantic role labeling.
2. How entities are introduced and flow (reference resolution)
3. Discourse-level parsing.

This is unsupervised learning based on probabilistic models. We used EM for learning. Probability model has components encoding knowledge.

1. P(New composite entity—entities)

   P(dressing—oil, vinegar)>P(batter—oil, vinegar).

2. Reference to intermediate entity or initial reference?

   P(batter—initial reference)<P(flour—initial reference).

3. P(location—action)

   P(stove—bake)<P(oven—bake)

Learned good composite words for different ingredients: eggs (yolk, mixture, noodels, whites). Flour (flour, mixture, dough, batter, top, crust).

We want to translate unstructured recipe text into structured plan, make planning models, and generate new recipes (ex. look at available ingredients).

Can we simplify this? Action graphs are noisy.

Input: a title and list of ingredients.

Output: long text with many sentences. Goal: make dish specified by title with ingredients. Use encoder-decoder NN. Global coherence is missing.

We make a neural checklist model. "Garlic tomato salsa" with tomatoes, onions, garlic, salt. Build on GRU's. Encode title. Have hidden state switcher: contextually decide new ingredient (look at list of things not used yet), old ingredient, coreference to something produced, etc. After using new ingredient check off. Then move foward.

Everything has to be differentiable. It's more like a probabilistic checklist. Switcher is not discrete, but interpolation. Only 1 is generic model, other is based on attention mechanism. Attention over available ingredents, used ingredients. Attention accumulated over time.

Example: oven eggplant.

Sentences in this domain are pretty simple.

Ex. skillet chicken rice. RNN repeats. "Stir in rice." Checklist has cleaner global coherence. We're not modeling it in a precise way.

Human recipe is more complex.

Ex. chocolate covered potato chips. Baseline bakes an empty pan.

We deployed the same system for hotel dialogue.

What's missing in end-to-end:

- Deep-fried cauliflower, cauliflower, frying oil, sauce, salt, pepper.

  Cauliflower wasn't fried...

  Surface string-based models won't go very far. We need much better reasoning models.

22.1.1 Dynamic ??? networks

Cf. Chris Dyer, recurrent entity networks.

Parsing by simulation, analysis by synthesis.

- 385 actions from cooking recipes.
- 8 state change types: location, composition, cleanliness, cookedness, temperature, shape, orientation, accessibility.
- Ask AMT workers to assign state changes to actions.

Read in sentences. Have actions represented as vectors, ingredient list as list of vectors. Attend over actions and entities. Have attentino coefficients. Compose action representations and entity representations, learn to predict what happens to different entitites.

22.2 Propositional knowledge about objects, actions, and events

1. Are elephants bigger than butterflies?
2. A horse is eating. Is it standing or sitting?
3. What makes a wedding a wedding?
4. Verb physics (Forbes, Choi)

It's hard to learn from text; learn by combining images with text.

"I am usually larger than a chair/pen/stone/ball/towel." People don't say these things; how to do we learn?

The way people talk implies this knowledge: x threw y probably means means x is better than y, weighs more than y, and as a result, y will be moving faster than x.

Represent using VerbPhysics Frames. Squashed x with y: x is smaller, lighter than y, etc.

Reverse engineer commonsense knowledge by solving 2 puzzles simultaneously:

- learn VerbPhysics frames
- learn object-object relations

with respect to relative physical knowledge over 5 dimensions. Vision can't do weight, strength, flexibility.

Create inference network as factor graph.

Representation learning of commonsense knowledge?

I've been going back and forth between representation choice. I started with action graph, then used implicit representation in network. End-to-end is applicable in some cases where procedural knowledge aligns pretty well with what's in the surface string. End-to-end doesn't help as much if there's a big gap between language and knowledge. Reverse engineering certain types of commonsense may be feasible. Some of commonsense knowlege can be organized as lexical or frame knowledge.

Some part of model learned from skip-gram.

Generate the graph? Train to learn from automatically parsed action graphs. It learns patterns in meaningful way. When generate text with graph, overspecify?

Can we validate whether something makes sense without explicit graph representation? The way we prove whether we understood is to predict what happened to each entity at the end.

Cf. story understanding, do things to each other, change each other's feelings.

How often is the model copy? It doesn't happen very often when generation space is constrain: checklist tends to prevent RNN's from just copying and repeat.

Intersect FSA with RNN language model: sample get legal sonnet, but completely random. Intersect with LSTMs. There are many engineering details. It's hard to generate out of LSTM; we generate backwards. We use other data like song lyrics.

Ghazvininejad et. al.: poetry.

23 Unsupervised representation learning (Yann LeCun, New York University)

We can train a machine on lots of examples. But will it recognize tables, chairs, dogs, cars, and people it has never seen before?

It's amazing what we can do with supervised learning (convolutional neural nets).

Supervised and unsupervised learning will give the same features at the low-level. But the high-level features are task-dependent. Is there a way to learn task-independent high-level features?

People have trained on ImageNet, fine-tuned on X-ray images, etc. It kind-of works. There is something intrinsic to images that is relatively task-independent.

CNN are weakly biologically inspired.

There are two pathways in the human visual system: Identify objects; localize objects. This has also been done in neural nets. Objects are segmented and labeled. (Ex. counting sheep. You can do instance segmentation.)

Mask R-CNN, He, Gkioxari, Dollar, Girshick `http://www.arxiv.org/abs/1703.06870`: Produces object mask for each region of interest.

Supervised learning can take you quite far.

Obstacle to AI: how can machines acquire common sense? This is an old holy grail for AI.

Machines don't have common sense. They must learn how the world works by observation. Predictive/unsupervised learning is the missing link. This is the main obstacle.

What does the agent need to have internally?

- learn/understand how world works
- learn large amount of background knowledge
- perceive state of the world
- update and remember world state estimates
- reason and plan

Intelligence and common sense = perception + predictive model + memory + reasoning and planning.

We don't know how to do predictive modeling.

Winograd schema:

- "The trophy doesn't fit in the suitcase because it's too large/small."
- "Tom picked up his bag and left the room."

How do we get a machine to learn millions of commonsense facts? (Cf. large team typing facts, hoping that commonsense will emerge, Doug Lennett, Cyc.) Commonsense is the ability to fill in the blanks.

- Infer state of world from partial information.
- Infer future from past and present.
- Infer past events from present state.
- Fill in visual field at retinal blind spot.
- Fill in occluded images.
- Fill in missing segments in text, words in speech.
- Predict consequences

- Predict sequence of actions leading to a result.

Necessity of unsupervised learning/predictive learning.

"The brain has 10^{14} synapses and we only live for about 10^9 seconds. SO we have a lot more parameters than data. This motivates the idea that we must do a lot of unsupervised learning since the perceptual input is the only place we can get 10^5 bits of data a second." (Jeff Hinton)

1. Pure (model-free) reinforcement learning (cherry): Sample complexity sucks. Machine predicts a scalar reward given once in a while.

   (I will advocate model-based RL.)

2. Supervised learning (icing): Machine predicts category, 10-10000 bits per sample.

3. Unsupervised/predictive learning model (cake): Predicts any part of input for any observed part. Predict future frames in videos. Millions of bits per sample.

This is a black forest cake, cherry is not optional.

VizDoom Champion: Actor-Critic RL system from FAIR, winner of 2016 competition.

StarCraft: Brood War RL-based system for battle tactics. Define subgoals, strategy... It's hierarchical. No one knows how to do this properly.

Sutton's Dyna architecture: Try things in your head before acting. Integrated architecture for learning, planning, and reacting.

RL sucks, do model-based RL. "The main idea of Dyna is the old, commonsense idea that planning is 'trying things in your head,' using an internal model of the world (Craik, 1943; Dennett, 1978; Sutton & Barto, 1981). This suggests the existence of a more primitive process for trying things not in your head, but through direct interaction with the world. Reinforcement learning is the name we use for this more primitive, direct kind of trying, and Dyna is the extension of reinforcement learning to include a learned world model."

Classical model-based optimal control: Simulate the world (the plant) with initial control sequence. Adjust control sequence to optimize objective through gradient descent. Backprop through time was invented by control theorists in late 1950's (adjoint state method). Optimal control people invented backprop, they just didn't realize we could use it for learning.

23.1 Architecture of an AI system

1. Perception: estimates state of world.

2. Agent

3. Objective: measure agent's happiness from state.

Inside the agent:

1. World simulator: predict world state

2. Actor: generate action proposal

3. Critic: predict long-term value

We know how to build the critic, we don't know how to build the world simulator.

Use classical optimal control:

1. Find action sequence through optimization
2. Use sequence as target to train the actor.

You don't want to do optimization step every time. Instead of running simulator every time, train an actor to directly predict which action to take. Eventually actor will directly generate the correct sequence of actions.
[6]

Insurance pricing. Challenge: when the world is not easily explainable, Actor-critic finds stupid edge cases. This system is dangerously convenient but horrifying the real world.

You need lots of diverse example, covering space. In classical control, people build a model by hand. They have accurate dynamical model. In anything in the world there is stochasticity, it diverges quickly.

The biggest challenge is the edge cases. The insurance company doesn't know what happens when you drop prices by half. You need beyond what you've seen before.

Learned model of Billiard and planning. Forward model: entity RNN + interactions (Henaff, Whitney, LeCun 2017).

Classical optimal control, except model built by entity RNN.

How to learn models of world? PhysNet learns intuitive physics. (Lerer, Gross, Fergus, `http://www.arxiv.org/abs/1603.01312`) Predict what happens to tower of cubes. Train on lots of experiments. We can show real cubes in real video and it does decently.

There are fuzzy predictions. Network predicts average of possible futures. This is a problem we have to deal with.

23.2 Inferring state of world from text: entity RNN

Augment neural nets with memory module.

- Recurrent networks cannot remember things for very long.
- Need hippocampus (separate memory module).
  - Memory networks
  - stacked augmented recurrent neural net
  - neural turing machine
  - differentiable neural computer

[6] Humans do backward chaining. There's only rollout here?
Optimization is backward chain.
There is a more high-level process with subgoals missing from the picture.
Greg Wayne, Columbia, DeepMind came up with hierarchical system for generating action.
Dynamic programming with pruning, backtrack and stay in high-value states?

Differentiable memory is like a soft RAM circuit or soft hash table. It stores key-value pairs (K_i, V_i).

Get scores for keys, put though softmax to get coefficients of values, linearly combine.

You can use this for question-answering.

bAbI: 20 types of tasks. Entity RNN can solve all 20 types.

Maintain current estimate of state of world. Each module is recurrent net with memory. Ex. John went to kitchen. Each cell decides whether it's relevant, if so, update: "kitchen" cell updated with John, "John" cell updated with kitchen.

23.3 Unsupervised learning

Energy-based unsupervised learning: Learn energy function (or contrast function) that takes low values on data manifold and higher values everywhere else.

Ex. negative log desnity.

How to learn a function that takes a vector and says whether it looks like the training set or not? We need to train the machine to learn these contrast functions. Tweak parameters so output is small outside. How do we make sure the function takes high values outside the samples?

Cf. partition function problem: if you give high prob to some, have to give low prob to others.

Push down on energy of desired outputs, up on everything else. How to choose where to push up?

7+ strategies:

1. build machine so volume of low stuff is constant
2. push down on energy of data points, up everywhere else.
3. push down on energy of data points, up on chosen loactions
4. minimize gradient and maximize curvature around data points (score matching)
5. train dynamical system so dynamics goes to manifold
6. use regularizer that limits volume of space that has low energy
7. $\mathbb{E}(Y) = \|Y - G(Y)\|^2$, make $G(Y)$ as constant as possible.
8. adversarial training

23.4 Adversarial training

I think this is the answer. Train a generative model. Two ways to view: how to train generator? How to train contrastive function?

Ex. try to predict what happens in the next frame of video, where will the pen fall? I don't want to punish the machine for making the wrong choice. I need a cost function that is a contrast function between the data manifold and outside.

Use 2nd neural net, train to produce contrast function. Train discriminator to tell difference between real futures and generated futures. Generator trains to produce predictions that the discriminator can't tell are not real. Adjust parameters to minimize output. Two loss functions: one minimized by discriminator, one by generator.

Eventually generator moves towards low-energy areas.

Energy-based GAN (Zhao, Mathieu, LeCun, ICLR2017). Discriminator is auto-encoder. For various pairs of loss functions there is Nash equilibrium.

EBGAN: Side connections help decoder reconstruct. You can use discriminator, train unsupervised on MNIST, and fine-tune on labels to get $< 1\%$ error on MNIST.

23.5 Video prediction

Look at 4 frames in past and predict 2 frames in future. Training with L^2 get blurry prediction. With adversarial training, it's crisper.

It's captured some structure of the data. Is it the case that the internal representation is helpful for recognition?

It's not clear.

Video goes haywire after 10 frames.

Predict in feature space? Ex. segmentation for car camera. Can you predict segmentation maps in the future?

You need this because the alternative is to wait for crash to happen, and learn that you shouldn't do this.

Q/A:

- Relationship between real inference process (forward), policy knows intuitively without having to execute.

  Ex. chess grand master compiles tree-exploration and planning into pattern recognition. This is the secret of AlphaGo. First is convolutional net, predict where you should play. That's the intuitive part. Second part is tree-exploration, MCTS. You want both. The first one helps the second one; if the second one does better it can refine the first one.

  Ex. learning to ski. Every prediction you make results in falling, be careful. Compile into reactive moves.

- Best thing is to predict average, but average is not outcome itself. How do you avoid predicting the average?

  You can use adversarial training or simpler form of training. Things in street views are more predictable. It's when something unpredictable happens that you have to react.

  One way of building self-driving cars is not only having predictive models, but see when they fail. There are many problems with ML that could be moderated with predictive or generative models.

- Transfer learning. Tips on building architectures good for transfer learning?

  ResNet, Inception don't seem to be as good as VGG.

It used to be a subfield of machine learning. Since deep learning, it's been obvious that you can do transfer learning. Fine-tune, or train on tasks simultaneously.

People do this for translation. It works better for languages with small amount of data.

How do you train a self-driving car system in simulator and transfer into real world?

It's the input that's different.

24 Generalization and Equilibrium in Generative Adversarial Nets (GANs) (Sanjeev Arora, Princeton University)

This paper makes progress on several open the- oretical issues related to Generative Adversarial Networks. A definition is provided for what it means for the training to generalize, and it is shown that generalization is not guaranteed for the popular distances between distributions such as Jensen-Shannon or Wasserstein. We introduce a new metric called neural net distance for which generalization does occur. We also show that an approximate pure equilibrium in the 2- player game exists for a natural training objective (Wasserstein). Showing such a result has been an open problem (for any training objective).

Finally, the above theoretical ideas lead us to pro- pose a new training protocol, MIX+GAN, which can be combined with any existing method. We present experiments showing that it stabilizes and improves some existing methods.

Joint work with Rong Ge, Yingyu Liang, Tengyu Ma, Yi Zhang.

There hasn't been much theoretical analysis of GANs.

Take Gaussian noise, input through neural net (denoising autoencoder, variational autoencoder) and transform into image. The hope is that the process teaches us something about the structure of images. There is concrete evidence in MNIST results, etc.

Prologue: In 2013 I visited Google and talked to Geoff. I asked Geoff, "I understand anthropomorphic principle that vision is represented by neural net. Why do you think realistic distrbutions are expressible by a small shallow deep net?" "Neural nets are like universal basis that can approximate almost anything very efficiently." Cf. tarp drapes over any object.

Geoff's "neural net hypothesis."

This is inconsistent with curse of dimensionality. In d dimensions there are $\exp(d)$ directions whose pairwise angle is $> 60°$.

The number of distinct distributions is $> \exp(\exp(d))$ by discretizing. Counting arguments shows we will need neural nets of size $\exp(d)$ to represent some of these distributions.

I thought: Real-life distributions must be special in some way...

For the last 5 years, I though that you have to make assumptions about distributions; in this work I won't.

We're trying to train a generative model to convert noise into faces. Train by training a discriminator net D_v, to train between realistic and synthetic images. This is reminescent of pseudorandomness.

Objective is

$$\min_{u\in U}\max_{v\in V}\mathop{\mathbb{E}}_{x\sim D_{real}}[D_v(x)] - \mathop{\mathbb{E}}_{h}[D_v(G_u(h))],$$

Wasserstein GAN (Arjovsky et al. 17).

Repeat until convergence: backprop update on discriminator.

Think of GAN as 2-player game. Most theorists like the idea of GAN because it presses the right buttons: pseudorandomness, game theory. Generator plays a net, discriminator plays a net.

When to stop? A necessary stopping condition is equilibrium: payoff unchanged if we flip the order of moves.

In practice this is very unstable and it's believed there's no equilibrium.

1. I'll address generalization: suppose generator has won and discriminator has been left with no option but random guessing. Does this mean in some sense that the true distribution has been learnt?

   Past analyses: If discriminator capacity and number of samples are very large, then yes because Wasserstein distance is exactly the $\max_{v\in V}\mathbb{E}_{x\sim D}[D_v(x)] - \mathbb{E}_h[D_v(G_u(h))]$.

2. Equilibrium: does equilibrium exist in this 2-person game? A priori not guaranteed, ex. rock/paper/scissors. You can convert this to a GAN example.

Bad news: bounded capacity discriminators (with n parameters) are weak. Suppose I take a uniform distribution on $\frac{n\ln n}{\varepsilon^2}$ random samples from D_{real}.

Theorem 24.1. *If generator has capacity n its distinguishing probability between these two distributions is $< \varepsilon$.*

Proof: ε-net argument.

1. Still holds if many more samples available from D_{real}, including any number of held-out samples.

2. Suggests current GAN objectives may be unable to enforce sufficient diversity in generator's distribution.

Hope internal parameters give insight into images.

Small discriminator cannot distinguish between low and high entropy.

Take 1000 samples; you see repeated faces, this suggests by birthday paradox diversity is not high.

Partial good news: if number of samples is $> \frac{n\ln n}{\varepsilon^2}$ then performance on samples tracks (within ε) performance on full distribution. Generalization does happen with respect to "neural net distance",

$$\max_{D\in U}\left|\mathbb{E}_{x\in D_{real}}[D(x)] - \mathop{\mathbb{E}}_{x\in D_{synth}}[D(x)]\right|.$$

(Similar theorems in pseudorandomness.)

Generalization happens for NN distance.

1. Deep nets are Lipschitz with respect to trainable parameters: changing parameters by δ changes output by $< C\delta$ for some small C.

2. If number of paramteres is n, there are only $\exp(n/\varepsilon)$ fundamentally distinct deep nets (others are ε-close).

3. For any fixed discriminator, once we draw $> \ln n/\varepsilon^2$ samples, then with probability $\leq \exp(-n/\varepsilon)$ distincuishing ability is not within ε of distinguishing ability for full distribution.

By union bound, empirical NN distance on $n \ln n/\varepsilon^2$ samples tracks overall NN distance.

Suppose generator just won. If number of samples is somewhat more than trainable parameters of discriminator, then generator would win against all discriminators of full distribution.

But why should generator win in the first place?

Just assume backprop gives optimal discriminator.

Equilibrium means D gets max paoff from G among all discriminators in class, G ensures min payoff to D among all generators in class. Pure equilibrium may not exist, but we're hoping for it to exist with payoff $= 0$.

Thought experiment: instead of single generator net, what if we allow infinite mixture of generator nets?

Fact: these can represent D_{real} closely (ex. kernel density estimation).

Generalization result does not use capacity of generator.

What about finite mixtures of generator nets?

Theorem 24.2. *A mixture of $n \ln n/\varepsilon^2$ generator nets can produce D_{synth} that looks like (within ε) D_{real} to every deep net discriminator with n trainable parameters.*

von Neumann min-max theorem: There exists equilibrium if replace discriminator/generator by infinite mixture of discriminators/generators. By our recent observations, in such an equilibrium for the GAN game, generator wins (with payoff $= 0$).

There is a finite mixture which obtains ε-approximate equilibrium, D get $\geq V - \varepsilon$ and G gets $\leq V + \varepsilon$, where V is the payoff in von Neumann equilibrium. There exists ε-approximate equilibrium when we allow mixtures of size $n \ln n/\varepsilon^2$.

Summary: Infinite mixture is powerful, finite mixture is roughly similar, lose only ε.

Existence of approximate pure equilibrium (for Wasserstein objective): Take small mixture (with weights $w_1, \ldots$) approximate equilibrium. Show it can be simulated by a single deep net. Use some bits in white noise to implement selector circuit to choose generator. The number of parameters is $\widetilde{O}(n^2)$.

Empirics: The MIX+GAN protocol can be used to enhance GAN game for any existing architecture. Take any GAN code, apply sauce (mixture) on top. Player 1/2 is mixture of k discriminators/generators, ex. $k = 3$ to 5. Maintain separate weights for each component, update via backprop. Use entropy regularizer on weights to discourage collapsing.

Even with $k = 3$ quality is better. (CeleA Faces Dataset) Adding capacity is not trivial. [7]

[7] Generator could be nonparametric. Take and add noise to it. It doesn't train, it's pretrained with data, so in a way has infinite capacity, gets close to kernel approximation.

Memorization is not that easy. It's easier to memorize with a mixture of networks. Are you pushing towards memorization with more networks. Will backprop find it?

Inception score: epoch by epoch MIX+DCGAN does better (Inception score). It's not trivial to get improvements like this. I'm not convinced Inception score is a good measure.

Also do MIX+WGAN. We get much better objective.

Epilogue:

- Infinite mix of very simple generators closely represent any D_{real}.
- Reasonable size subsample of generators produce D_{synth} indistinguishable from D_{real} by any small net.
- D_{synth} should look like D_{real} to us if our visual system is small neural net.

LSH, look at WGAN output with nearest neighbor?

Give generator a bit more capacity, generator can win by sampling uniformly. That happens because it has more capacity. Generator capacity has to limited—how?

25 Provably Learning of Noisy-or Networks (Rong Ge, Duke University)

In this talk we discuss recent works on learning the single-layer noisy or network, which is a textbook example of a Bayes net, and used for example in the classic QMR-DT software for diagnosing which disease(s) a patient may have by observing the symptoms he/she exhibits. These networks are highly non-linear, as a result previous works on matrix/tensor decomposition cannot be applied directly. In this talk we show matrix/tensor decomposition techniques can still be adapted to give strong theoretical guarantees even for these nonlinear models.

Why are we interested?

One traditional idea in learning representations is to use latent variable models. If I estimate unknown parameters, they tell me about structure in data. I want to learn mean and covariance. For many of these simple latent variable models, we can learn by tensor decomposition, in polynomial in parameters and $\frac{1}{\varepsilon}$ time.

What happens when we try to generalize to slightly more complicated models?

These are more similar to nonlinear versions of the traditional models. It is very similar to RBM's. Hopefully understanding these models will tell us more about why deep representations are useful.

The same techniques don't apply immediately; these problems are harder to learn.

The previous problems we know how to learn are linear; these problems are nonlinear. What ideas can we use to learn these nonlinear latent variable models.

Disease-symptom networks: diseases are 0-1 variables. We have an edge from disease to symptom if it has some probability of causing the symptoms.

If we observe the sympotoms of many patients we want to learn how many diseases, and which symptoms they cause.

m disaeases d_i, independent wp p. Edge weight $\mathbb{P}(s_i = 0|d_j = 1) = \exp(-W_{ij})$—if patient has only one disease, probability diseases does not happen. In the 90's people tried to construct these graphs from expert knowledge. One famous dataset is the QMR-DT network, 570 dieases, 4k symptoms, 45k edges.

Suppose patient has 2 diseases. If they both have $50\% = 1-\exp(-W_{ij})$ chance of causing a symptom, each tries to cause the symptom independently, we observe if at least one disease causes the symptom. The probability we observe the symptom is 75%. "Noisy" means each tries to cause with some probabilitity; "or" means final value is an "or".

Theorem 25.1. *We give a polytime algorithm that recovers W with $O(\rho\sqrt{m})$ relative error in ℓ_2 norm in each column (disease).*

(Think $\rho = C/m$.)

This has fewer requirements on structure of network. The problem is that the polynomial is large. If there are additional nice properties, we get a faster algorithm, poly in $\frac{1}{\varepsilon}$.

Theorem 25.2. *If the network has nice combinatorial structure (true for QMR-DT), an algorithm to recover W with accuracy ε using* $\mathrm{poly}(n, m, \frac{1}{\varepsilon})$ *samples and running time.*

I make a comparison to topic models. A lot of our algorithms are inspired from algorithms for topic models. What are those algorithms and what do we need to do to apply similar algorithms?

- Topic is distribution over words. Multiple topics: words from mixture distribution.
- Disease is set of symptoms. Multiple diseases: symptoms from or.

Topic modeling is linear; noisy or is not.

Ex. consider document with 2 topics, patient with 2 diseases.

- Generate 30%, 70% words from topics 1, 2. Final doc is all of the words. $\mathbb{E}\,(\text{doc}) = 0.3t_1 + 0.7t_2$.
- Generate symptoms from diseases 1, 2. Final symptom is union of 2 sets. $\mathbb{E}$ patient is nonlinear expression.

Idea: linearize. Use PMI (pointwise mutual information),

$$PMI(s_i, s_j) = \ln\left(\frac{\mathbb{P}(s_i, s_j)}{\mathbb{P}(s_i)\mathbb{P}(s_j)}\right).$$

$PMI(x, y) > 0$ if x, y are positively correlated, and inversely.

If symptoms i, j share disease, then $PMI_{i,j} > 0$.

Claim 1.

$$PMI = \rho\sum_{k=1}^{m} F_k F_k^T + \rho^2(\cdots) \approx \rho\sum_{k=1}^{m} F_k F_k^T, \quad F_k = 1 - \exp(-W_k)$$

by Taylor expansion. Log transformation linearizes the product.

Get similar expression for PMI tensor $PMI3(s_i, s_j, s_k)$ measuring 3-wise correlation. Analogous to inclusion-exclusion formula. $PMI3 \approx \rho \sum_k F_k^{\otimes 3}$.

We get systematic error terms because of nonperfect linearization.

Topic models: if you can compute word-word correlation and 3-word correlation (low rank matrix, tensor), you can apply tensor decomposition to get topic matrix. We try to use the same approach. Hope is that we have PMI matrix, approximately low-rank, PMI tensor, and apply tensor decomposition to get W.

Challenge: we only have access to tensor plus systematic error. We need to handle it carefully.

Claim: $PMI \approx \rho FF^T + \rho^2 GG^T + ...$ where $F = 1 - \exp(-W)$, $G = 1 - \exp(-2W)$.

Recover span of F from PMI?

Matrix perturbation theorem (Davis-Kahan, Wedin) are not strong enough, give recovery error $\precsim \rho m$, vacuous since $\rho m \geq 1$.

Difficulty: F, G are not well-conditioned.

Key observation: F, G are very similar.

Intuition: more tolerant on large singular directions. Singular directions of G are aligned with singular directions of F!

Traditional theorems don't differentiate these cases.

Lemma: recovery error $\precsim \rho \max \frac{x^T GG^T x}{x^T (FF^T + \sigma_m(F)^2 I) x} =: \rho\tau$. On QMR-DT, $\tau \leq 6$. This is provably small constant for random sparse graphs. It suffices to get good approximation for span of F. We need to generalize to asymmetric matrices/tensors.

Summary:

- PMI approximately linearlizes log-linear model.
- Better matrix/tesnor perturbation results handle systematic error.
- Challenge: PMI tensor requires many samples.
- Next: use structure of disease/symptom graph to get faster algorithm.

Inspired by topic models. Compare topic models and noisy-or. Rows are words/symptoms. Columns are topics/diseases. An anchor symptom is a symptom that appears in only one disease. (An anchor word appears in only 1 topic.)

Only a subset of diseases have anchor symptoms. Idea: learn these diseases first and them remove. Then other symptoms which were not anchor before are anchor.

Actually, use sequential 2-anchor condition: if all diseases have not been recovered, there is a disease with at least 2 anchor symptoms. (In QMR-DT, 7 layers is enough.)

In previous works, sample complexity depends exponentially on T. Maybe we can learn diseases in first few layers.

Reduce noisy OR to symmetric NMF. $PMI \approx \rho FF^T$. This is symmetric nonnegative matrix factorization. We need to be careful with higher-order terms; I ignore this.

High-level algorithm: Repeat until all diseases learned.

1. Find all anchor symptoms.

   If 2 anchors correspond to same disease, rows in PMI matrix are duplicates. If we try to subtract this component, no entry should become negative.

2. Learn diseases with at least 2 anchors.

   We only need to learn a scaling. $F_p = \lambda PMI_j$, $PMI_{i,j} = \rho F_p(i) F_p(j)$.

3. Remove diseases from graph. subtract $F_p F_p^T$.

We run synthetic experiments.
Open:

- More practical for learning noisy OR (sample complexity)
- Better generative model for QMR-DT (why layered structure?)
- Learning more nonlinear models (RBM, deep belief networks, etc.)

Dependencies between diseases?

26 3/30/17 Panel

1. MT: Societal impact: Last time: answer was not our problem. What is our responsibility?

   YLC: Facebook, IBM, Google, Amazon, Apple have partnership for AI, joined by ACLU, etc. Discuss ethical and safety issues including societal impact. I'm most worried not about Terminator but about the accelerated progress of tech which increases wealth and income inequality, has already created a new gilded age. Marxism 2.0.

   First gilded age created Marxism 1.0 and split the world for 3/4 of a century.

   The issue is how political system deals with accelerated tech progress. Population trails behind. As tech develops, more and more are left behind and they're pissed. European countries are more prepared.

   MT: It's our responsibilty to have solutions.

   YLC: I'ts our reseponsibility to point the problems. It's arrogant to say we have political solutions.

   CP: People like Bertrand Russell did not shy away. It was an even more dangerous era. There were brilliant thinkers of the kind we admire from both sides—von Neumann and Bertrand Russell.

   YLC: Every economist I've heard says: something has to be done at political level.

   CM: Sometimes I feel like things are going to be really bad, other times I'm unconvinced the problems are worse than changes that have come before. There have been enormous changes, 90% to 3% in farming, number of people making garments. The fear is that it's going faster this time.

   People in AI suffer from thinking it's so special, but it's no so different from moving from people hand-sewing cloth to having machines doing that. The fact that we have

billionaires now is no different from Andrew Carnegie one century ago. Society will adapt better than some people are fearing.

People can find employment being artisans again because people look for human quality.

CP: Industrialization, beginning of information era. For the last 30 years there has been social consensus. The world is advancing and people want to get the fruits of that.

Back then there was the specter of communism, left-wing ideology that was flourishing. That's not the case now. As a result, you have the disenfranchised believing that the way to go is to let multibillionaires take directly the reins of power.

2. TM: Can we learn something from neuroscience, how the brain is doing representation learning?

   CP: The short answer is that it's depression how little we know about representation in the brain. No experimental field gallops faster than neuroscience, my impression is that the field is more and more confused.

   YLC: Largely the result of learning, no reason they are easy to understand. More interesting to think about how it learns. I've always been interested in how the brain learns, not how it works once it's learned. A lot of neuroscience has been on the second question. A lot of people in vision are interested, frustrated by NN, you don't need to understand vision anymore, spend time looking at how convnets work, what's the point?

   SA: Decoding fMRI: measure blood flow at 6000 places, make sense of it using word embeddings. There's signal like that.

   MT: There's no real notion of how to extract what's going on in the brain. Work on correlating connection patterns?

   CP: You can measure lots of things, infer synapses by correlating firing. I believe it's much harder to measure weights.

   MT: I meant bigger structure, see Marvin Minksy talks, pillars...

   CP: We understand more and more. Learning is an object of intense study. Three layers of Marr: algorithm, program, hardware. Learning is a fourth part. We have to understand how the brain is modified by experience.

   We now some of the neural circuits. We know a lot about the sensory, visual cortex.

3. ?: What are prospects for understanding errors and uncertainty in predictions by neural nets. Can we bound errors, have confidence intervals? Is this something we can ever do?

   YLC: Bayesian averaging: approximate by Gaussian depending on Hessian... Marginalize over distribution, get confidence intervals. It's not clear whether they're good.

   : You have to put in a lot of work.

   YLC: You can simplify with assumptions, diagonal Hessian, etc., don't lose too much.

MT: I'll take your neural net diagnosing disease and tell you what drug to take. What does it take for you to trust this?

CP: He wants to be the 10000th patient.

YLC: I'm not convinced there's a principled method better than a hack. Find the place on precision-recall curve...

How do you get reliable scores?

SA: I don't understand the answer for any traditional method.

MT: Sometimes Hoeffding tells me something.

: Is there a way to know whether you have confidence in your answer?

*: One issue is that we're focused so much on accuracy in our metrics. If we gave points for saying "I don't know"...

: Imagine an autonomous driving system stopping and alerting the driver when it doesn't know.

YLC: Have a watcher neural net, SafetyNet at NVidia.

4. Br: What's wrong with the bootstrap—resample from the training set, rerun convnet and see predictions it makes. We've talked about rep learning. Representation theory is well studied in math, represent algebraic structure. Should we use it as a guide?

   YLC: Resample training set and retrain. If you train large net it takes a long time.

   Br: If it's drugs I'm going to take...

   TM: Assume iid large sample, not theoretically justified.

   YLC: Restarting, splitting data set various ways, take samples around solution...

   CM: They work!

   YLC: If it's worth the trouble use them.

5. J: Meaning and symbols. What are the implications for philosophy and linguistics and vice versa? Are we getting closer to understanding to Wittgenstein meanings? Is there room for philosophy and linguistics and deep learning to come together, or are they different parts of the world that don't need to talk?

   CP: I'm surprised that more philosophers haven't put under scrutiny the field of science and tech.

   J: In philosophy there was a lot of head-butting, philosophy didn't think there was much to learn.

   CM: Interesting implications. Issues of meaning, how to find meaning in different ways relate directly to different things we're doing in deep learning, shaking different theories of meaning.

   There's a lot of space for more denotational theories of meaning, capturing those in deep learning systems. Symbols have been prominent in philosophizing, but more of an epiphenomenon in machine learning.

J: Platonic idea, word vector?

6. :Projection 5 years in the future, will deep learning still be front and center?

   YLC: I think we'll find something beyond deep learning, but gradients will stick with us.

   There are several aspects. One is gradient descent works, you can run a lot, even things that are complicated. You can make learning model by assembling modules. There are many instantiations.

   If you define as learning representations, that will stick with us.

   If you see as more philosophy/AI, replace symbols by vectors, I think that will stick, only way to have differentiability.

   I see it disappearing from front and center, being just a tool. We don't talk about computers anymore.

   CM: There will be a much bigger swing to something else, I don't know what it is. I feel I have this clear memory, back around 2008, everybody (but Yann) was so convinced that prob models were right, they were the solution being found because prob were obviously the one true path, you would have been laughed out of the room saying that in a few years everyone would be doing nn again. I assume the same thing will happen. In 10 years: people 10 years before were building these discriminative models with no idea what they were doing...

   SA: Life is simpler than complexity theorists thought. There could be alternative universe where there is a SAT-solver and you leverage that and do AI.

   YLC: The theory that everything is nearest neighbor and you just need a fast NN.

   SA: There's a clear reason why that can't work. If P=NP, you could use that to solve AI. There's an alternate reality where that could have happened.

   DS: It's a question all of us think a lot about every day. What is missing, what is deep learning not giving us. Is there a hope of getting there going down this path? An important part is how to get better abstraction. It doesn't have the properties we hope, will gradient descent give us that. I don't see evidence it will, but I don't see better solutions.

   First we see advancement at perception level, vision. I come from program synthesis, my goal is to teach computer to write code. Whether gradient descent will give us that is open question. From what we've seen so far, we have significant pieces missing, we just don't know what they are.

   YC: Many of us are interested in creating a model that can reason better than how things work today, seems to require background knowledge about world, it's unclear how we do that. One of 2 things can happen: assume what we work on today are applications that make sense as AI challenges. If we run out of applications and datasets we can do well, look at bigger challenges, entirely different classes of algorithms, ML approaches, could be inference coming back, that could introduce new class of directions. A new name.

At the moment people want to look at different problems compared to several years ago: reading comprehension, image captioning. If we keep doing it... I hope some reasoning will happen.

7. JeffB: Holy grail of real-world knowledge: grounded knowledge, model-based RL. Historically in AI people (ex. Cyc) wanted real world knowledge in some representation, laid out or learned automatically. TO the extent that we want these to mimic human ways, might be the extent we don't want these to be the real world. Humans are enamored with the fantastical, believe in deitic entity. At any point in time, most people are thinking of things that have nothing to do with real-world knowledge, reasoning. (Recent election) Is it the case that efforst to construct real-world model are doomed to failure because they don't fantasize the way a person does?

   YLC: Due to 2 things. Exploration-exploitation: exploration means imagining new things that are not necessarily efficient, possibly dangerous to individual, increases chance of population's survival. Ability to predict and act on world, to get to state you like, need causal inference, see what we can influence. That drives us to have a mechanism to establish causal relationships, and find it when there is none.

   J: When people using for financial, edge cases. If model is more imaginative and creative (kids coming up with ludicrous situations, things that will never happen).

   CP: A lot of fallacies are result of blind spot to understanding probability. What genius is is to see things beyond what is on the table, find associations that shouldn't be there. This is something that we should try to make our machines do.

   J: Every human is a genius in our ability to infer things about the real world. Is that a necessary component?

   DS: You can come up with deeper models to simulate the world.

   J: It has to be plausible fantasy. Errors GANs make are not plausible, whereas a person vanishing is plausible.

   SA: Going back to other questions, that feels more symbolic. Is that in the current models or not?

   J: Symbolic vs. continuous, orthogonal issue.

   SA: Can you do it with a vector.

   J: High-level representations are necessary.

   : Why do you think GANs will fantasize something will disappear?

   J: Is that something that needs to be in the training?

   The error: Every AI scientists things everybody thinks like them. It's not ths case that people are, think the same, but they functino perfectly well.

   SA: Is an MDP a reasonable model?

   YC: I don't think necessarily there should be one repository of commonsense knowlege. If we could randomize the way neural nets learn text, images, interact with world, have internal representation of what it's seen so far, in a way it's the case with robots that

learned from interaction, they all have different internal models because of different experiences. We could combine to make stronger, or keep them separate specializations. It's not clear there's one perfect model. Everyone has cognitive dissonance. We think we think but pattern match and believe something not entirely correct (if we sit down and think about it).

: I fantasize based on things I haven't seen, no one has disappeared in front of me but I imagine it. There has to be a mechanism.

8. Do you think you work on artificial intelligence vs. AI problems?

   MT: I quit music school to solve strong AI.

   YLC: I try to solve AI, but it's arrogant to say it's one step towards AI. In a paper you want quantifiable progress on a concrete problem. The paper trail is more applied. I want to get machines to learn forward models.

   You have to be scientific about it.

   CM: AI problems. I'm not directing myself to think about solving the whole AI, am grounded in human language.

   CP: I started grad school in 73, wanting to do AI, that was the first major AI winter, instead did theory. I'm very envious. We started with the question, will AI take over the world, take away everybody's job. How about our world: computer science department. Theoretician, others do graphics, databases, PL. Is AI taking over our world?

   DS: I didn't quit security but expanded research scope.

   YLC: Applications for Ph.D. 2/3 want to work on AI related fields.

   CP: In our department, 3/4. I remember when this happened in 1993. I was at Stanford, 80% of applications wanted to do AI. Discriminator we used was whether they talked about G'odel Escher Bach. If not admit them as theoretician.

   YLC: Derive knowlege from data in operational way. The amount of data is growing exponentially, as fast as communication bandwidth of network. Capacity of brain to analyze is not growing. It's important for society to turn data into knowledge. It could be a field that spins off from CS, the same way in 60's CS spun off from math and EE.

   In NYU, CS department is malignant tumor on math department. Now we see what AI/ML. Subfield of statistics, where you use computers instead of pen and papers. Vs. subfield of CS?

   CP: Now CS is a misnomer. It's obviously about so much more.

   JB: You're aware of the old adage, all problems solved by one additional level of indirection. Including problem of computer science itself. Write programs that write programs. Automatic program writers. Compilers back in 50's was viewed as AI. In some sense it's natural outgrowth of CS.

   YLC: When a field is claimed by many fields... We're borrowing methods from physics (prob models).

27 Learning Paraphrastic Representations of Natural Language Sentences (Kevin Gimpel, TTI Chicago)

I will discuss our experience in learning vector representations of sentences that reflect paraphrastic similarity. That is, if two sentences have similar meanings their vectors should have high cosine similarity. Our goal is to produce a function that can be used to embed any sentence for use in downstream tasks, analogous to how pretrained word embeddings are currently used by practitioners in a broad range of applications. I'll describe our experiments in which we train on large datasets of noisy paraphrase pairs and test our models on standard semantic similarity benchmarks. We consider a variety of functional architectures, including those based on averaging, long short-term memory, and convolutional networks. We find that simple architectures are easier to train and exhibit more stability when transferring to new domains, though they can be beaten by more powerful architectures when sufficient tuning and aggressive regularization are used.

This is work on learning to embed natural language sentences.

We all have an impression that word embeddings are very important. Pretrained word embeddings are really useful. They are a go-to technique for NLP. Use to initialize for your supervised task.

When it comes to longer things (phrases, sentences), there aren't pre-trained models.

Socher, Huang, Pennington, Ng, Manning (2011) introduce recursive neural net, composing based on syntactic parse. It's one of the early examples of getting a generic vector space model.

Another famous example is paragraph vectors, Le and Mikolov (2014). In your training set, learn representation not just for words but sentences or paragraphs; have a sentence predict its own words. At the test time you have to do the same procedure (gradient descent).

In neural machine translation, encode source sentence and then decode translation. (Sutskever, Vinyals, Le 2014, Cho et al. 2014)

RNN used to translate language can be used as sentence embedding model. They plotted the vectors to see what it's capturing.

Skip-thoughts: encode sentence, decode neighboring sentences. Condition on sentence vector in generating the words in neighboring sentences.

Nearest neighbors. RNN's are excited when you see the same sequence of categories of words: "I'm sure you'll have a glamorous evening, she said, giving an exaggerated wink." vs. "I'm really glad you came to the party tonight, he said, turning to her." There is both shallow and topical similarity.

LSTM autoencoders: encode sentence and decode same sentence. Hill, Cho, Korhonen (2016) made a LSTM denoising autoencoders—encode corrupted sentence and decode sentence.

I want to learn paraphrastic embeddings—sentences with the same meaning should be close. Applications:

- Multi-document summarization (see if sentence to add is already there).
- Automatic essay grading.

Evaluate by semantic textual similarity. Humans evaluate 1–5 how similar 2 sentences are. This is different from relatedness, topicalness; it's about similarity in meaning.

Evaluate on 22 datasets from many domains.

Would 2 sentences that mean the opposite be rated less similar than sentences about orthogonal things? Contradiction vs. unrelatedness—the dataset conflates these thigns. (Contradiction may be higher.) It may make sense to have 2 axes: contradiction vs. relatedness judgment.

See also Wieting, Bansal, G, Livescu (2016).

If we take pretrained word vectors and do averaging, it does better than the LSTM, etc. models! If you do fine-tuning, the other models can do better.

There are lots of freely and cheaply available data, like paraphrase databse (PPDB). Ex. "I can hardly hear you" and "you're breaking up." It's a bit noisy (arbitrary boundaries) but huge.

27.1 Learning

Goal: learn sentence embedding function $g_\theta(x)$. First, do averaging. For learning, use hinge loss

$$\min_\theta \sum_{(u,v)\in \text{Train}} [\Delta - \cos(g_\theta(u), g_\theta(v)) + \cos(g_\theta(u), g_\theta(t))]_+,$$

where $t = \text{argmax}_{s:\langle\cdot,s\rangle\in\text{batch},s\neq v} \cos(g_\theta(u), g_\theta(s))$ is a negative example. We do argmax over current mini-batch for efficiency. (Over the whole dataset, we will actually find true paraphrases.)

(What about adversarial models?)

We find sometimes that this objective can be too easy.

Regularize by penalizing squared L_2 distance to original.

We do 71% (average on PPDB), improvement over 65%. How about LSTM? Using the same learning framework, we get 52%. What's going on here? Why is the LSTM struggling so much?

Do in-domain evaluation on held-out annotated PPDB pairs. Then LSTM does a little bit better (61.3% vs. 60%). It is able to learn something. Word averaging is better at all sentence lengths in test data. Characteristics of phrase pairs themselves are the issue.

Why does the LSTM struggle on out-of-domain data?

Maybe the problem is the training data.

New data: sentence pairs automatically extracted by Coster and Kauchak (2011), English and simple English Wikipedia. Simple English is often longer and uses simpler words. Data is noisy.

Training on simple-standard Wikipedia, both get better, but LSTM improves more (averaging is still ahead).

Maybe LSTM is memorizing training sequences. We tried other ways to regularize. For people who cared about language this is a shocking regularization. We shuffled the sentences. The LSTM learns to not pay too much attention to the sequence but remember all words. We get gain of $\approx 6\%$. A critical point: you're just learning averaging. Combining with more

tricks (dropout) gets equal performance to average. (Dropout on whole words also helps.) Averaging over hidden states helps a bit more.

Word averaging underestimates similarity when there are multiword paraphrases. (Ex. imposed = being introduced by force.) Averaging doesn't pick up info on cooccurrences of words.

SA: do TF-IDF instead of averaging.

There are cases when LSTM overestimates similarity—with similar sequences of syntactic categories but different meanings. (birds $\neq$ dogs) Which differences in words lead to different meanings?

We can train as neural translation system. It seems to give reasonable results. One challenge is that anytime you have to generate words, it's harder to scale up. This way you never have to do softmax over large vocabulary, you can scale to larger datasets, training is fast. You get same results at 30,000 as 160,000 sentence pairs; for translation model you need more data.

In the dataset, contradiction is less common than unrelatedness.

Gated recurrent averaging network (GRAN): Inspired by the success of averaging and LSTM, we propose

$$a_t = x_t \odot \sigma(W_x x_t + W_h h_t + b), \quad g(x) = \frac{1}{|x|} \sum_t a_t.$$

This is more stable to train.

We can look at the magnitudes of the gates and look at which parts they are focusing on. Direct objects and objects of proposition phrases have larger weights than subjects. I think the reason is that looking at the sentence pairs, the new information resides more at the object than the subject level.

We see this pattern across different parts of speech.

Using unsupervised representations to initialize and regularize, we have gains.

New data: automatically translated bilingual sentence pairs.

Q/A:

Middle ground between averaging and sequence, it seems there are lots of ways to do it. w_{h-1} vs. average over previous or average over previous w_i's. How would those do? Try more variations...

28 Formation and Association of Symbolic Memories in the Brain (Christos Papadimitriou, UC Berkeley)

Joint work with Santosh Vempala and Wolfgang Maass.

Brain and computation: the great disconnects.

- Babies vs. computers.
- Clever algorithms vs. what happens in cortex.

- Deep nets vs. the brain.
- Understanding brain anatomy and function vs. understanding emergence of the mind.

  Despite insights into how the brain works, we don't understand how the mind emerges.

What are you going to bring to the question? Computation. How does one think computationally about the brain? This is a good question.

John von Neumann wrote a book in 1950. "[The way the brain works] may be characterized by less logical and arithmetical depth than we are normally used to."

Les Valiant on the Brain (1994): A computational theory of the Brain is possible and essential. The neuroidal model and vicinal algorithms. Random graph with language for operations, carefully designed so it can be implemented by neurals. This generates a genre of algorithms, vicinal algorithms.

David Marr (1945–1980), an amazing Turing-like figure. He came up with the 3-step program

1. specs: what the system does
2. algorithm
3. hardware

Can we use Marr's framework to identify the algorithm run by the brain?
Our approach/disposition:

- Start by admitting defeat: Expect large-scale algorithmic heterogeneity.

  There are thousands of problems to be solved, find them one by one. Use Marr's method but not in grand way.
- Small computational theories consistent with what we know from neuroscience.
- Start at boundary between "symbolic/subsymbolic" brain function.
- Once candidate: assembly of neurons in MTL

Experiment (Ison et al 2016).

Neuron lights up every time Eiffel tower is shown. Show Obama, etc., neuron doesn't light up. Photoshop Obama into Eiffel lights up. Neuron fires. Then show Eiffel tower, it fires. Show Obama, it fires.

This guy has learned that Obama has been to the Eiffel tower. That's learning.
These are the specs. What is the hardware, what is the algorithms?
Speculating on the hardware:

- recorded 100 out of 10^7 MTL neurons in every subject.
- So each memory is represented by assembly of many 10^4 to 10^5 neurons (Hebb 1949, Buzsaki 2003, 2010). This is the nearest we have to consensus in neuroscience.
- It is highly connected, therefore stable.

- It is somehow formed by sensory stimuli.
- Every time we think of this memory, all these neurons fire.
- Two memories can be associated by "seeping" into each other.

(What is you ask to imagine Eiffel tower?)

Algorithm?

- How are assemplies formed?
- How are they recalled?

Theory challenge: in a sparse graph, how can you select a dense induced subgraph? In theory we have been haunted by this problem for 50 years, the clique problem.

There's evidence that synapses are changing, new ones are formed. It's not unchanging. (We don't need to think about this.)

Everything is a hypothesis. We know this is necessary for the current theory of assemblies.

MTL has $\approx 10^7$ neurons. Stimulus stimulates $\approx 10^4$ neurons to fire (sensory cortex). 10^4 neurons in MTL fire as a result of zero-sum war.

Note that these are scattered neurons. These are extremely important, also philosophically. This is the first time it lost every connection to the world, become an abstract memory.

Metaphor (homology): olfaction in the mouse (Axel 2011). How olfactory is projected: In mouse's nose there is 3 cells that specialize. Once tarragon cell hits, neurons fire, a lot of activity happens in olfactory ball. There's a pea-size sphere that specializes in tarragon. Once it fires, there is a huge strong pathway, projects to prefrontal cortex, where it disperses. This is where you say "I like tarragon," where it becomes an idea.

From the discussion section: A few cells could receive enough input to fire. This small fraction generates sufficient recurrent excitation to recruit a larger population of neurons. Strong feedback inhibition would suppress further spiking. Some cells receive enough recurrent input to fire without receiving initial input.

MTL seems to be a random graph $G_{n,p}$, $p \approx 10^{-2}$.

We have to take plasticity into account: how neurons learn from experience. "Fire together, wire together." (Hebb 1949) Near synchronous firing increases synaptic weight.

How to verify such a theory? Reproduced in simulations by artificial neural nets.

Look at linearized model, without the threshold.

$$x_j(t+1) = s_j + \sum_{i \to j} x_i(t) w_{ij}(t) \tag{20}$$

$$w_{ij}(t+1) = w_{ij}(t)(1 + \beta x_i(t) x_j(t+1)) \tag{21}$$

Equilibrium equation is translation of Axel's prediction.

Theorem: linearize dynamics converges geometrically to

$$x_j = s_j + \alpha \sum_{i \to j} x_i^2,$$

be born rich or have rich friends. (In order to have a good chance to be an assembly, get stimulus or be preceded by neurons in the assembly.) These are small numbers; squares says if all predecessors are small, they don't help.

You're living in a sparse random graph soup. Excitation happens.

Quantitative narrative theorem:

1. Elite (high s_j) cells fire.
2. About half of elite cells fire again, and so do some other highly connected cells.
3. "Rich get stably rich" through plasticity.
4. Part of assembly may keep oscillating (common periods 2 and 3).

How can a set of neurons have connectivity? Associations?

Random graph theory does not seem to suffice.

Song et al. 2005: Some deviation from $G_{n,p}$: reciprocity and triangle completion. Once you have 2 edges, the probability the triangle is completed is much higher 10^{-1}.

This explains the mystery. Birthday paradox says there is at least one neuron connected to 2 sets, so the 2 sets have a lot of connectivity.

Theory challenge: how to select a dense induced subgraph?

- Recruit expressly highly connected nodes
- Plasticity
- Edge completion and birthday paradox.

As von Neumann predicted, the algorithm vanishes. Hardware and task have coevolved.

Another operation is bind, e.g. "give" is a "verb". It's not between assemblies but between assembly and brain area. Pointer assembly, surrogate for "give" is formed in "verb" area. Legenstein et al 2017.

Internal connectivity is immaterial. Operations are Join, Link $\neq$ association. It's formed by recruiting completely new cells. Through orchestrated precise setting of parameters (strengths, thresholds). Also see Predictive Join (P. Vempala COLT 2015).

The brain in context:

- Environment is paramount.
- Language: environment created by us a few thousand generations ago.
- Last-minute adaptation
- Hypothesis: evolved so as to exploit brain's strength
- Langauge optimized so babies can learn it.

Knowledge of language= grammar. Assemblies, association, bind (pjoin) seem ideally suited for implementing language in brain.

On learning representations of the world: Imagine a mouse running around. The mouse has in its brain a cell. Picture: Every dot is the place where a certain cell fires. It forms a triangular grid. This is representation learning in the real world.

Got Nobel prize for discovering.

This is learned representation of real world. How? For what purpose? See "Optimality of grid cells."

Conclusion: how does one think computationally of the brain? Assemblies. Experimental neuro provide specs. Algorithm often vanishes. Language!

QA:

Famous results: regularizing and normalizing even when adults around don't speak well (cf. creolizing pidgin). Is this evidence or challenge?

Randomness is just "we don't understand it"? If randomness is vehicle for saying we don't understand, we should definitely use it in the brain.

Cell firing pattern depends on geometry of rat's nest. It adjusts.

Binding operation: did you go through combinatorics. If you define in connectivity-based way you tear everything up?

29 Learning Representations for Active Vision (Bruno Olshausen, UC Berkeley)

The human visual system does not passively view the world, but actively moves its sensor array through eye, head and body movements. Here we explore the consequences of the active perception setting for learning efficient visual representations. This work focuses on two specific questions: 1) what is the optimal spatial layout of the image sampling lattice for visual search via eye movements? and 2) how should information be assimilated from multiple fixations in order to form a holistic scene representation that allows for visual reasoning about compositional structure of the scene? We answer these questions through the framework of end-to-end learning in a structured neural network trained to perform search and reasoning tasks. The derived models provide new insight into the neural representations necessary for an efficient, functional active perception system.

Synthesis of EE, CS, neuroscience.

Joint work with Alex Anderson, Brian Cheung, Eric Weiss, Pentti Kanerva.

What are the principles governing information processing in this system?

The eye is a 10 Mp camera that's always on. It's remarkable from an engineering perspective—a camera power-hungry. Optic nerve: 1Gb/sec datastream to the brain. Where does all this data go? It has to be assimilated on the fly. What parts are new, what have I seen before. If you don't do it, the data is gone, you don't get a second look.

This is done with 20 watts.

I'll talk about another remarkable property: the eye moves 3–5 times per second, mostly because of self-generated motions. The data is highly dynamic.

We can look at how things are done in biology, see things that are important that we've overlooked. We're not aware of eye movements. They're not random scanning, they're purposeful. You're mostly unaware of them. Yarbus, 1967. From the view of biology it's immesnsely important. Lots of animals use this.

Ex. jumping spider has 8 eyes, 2 high-resolution eyes, eyes at side are fish-lenses iwht wide field of vision. Between all the eyes it has 360° cover. It doesn't use a web; it hunts its prey. It uses low-res system to see things that are moving, then uses its high-res system. It can solve 3-D mazes.

1-day old: tubes for eyes moves back and forth in scanning pattern. Retina is not 2-D array, but 1-D photoreceptor. It has about as good spatial resolution as a cat. (And has behavior similar to cat.) It scans back and forth to form an image. (Bair, Olshausen, 1991)

1. It's not about the image. In computer vision, spend a lot of time on the image. There's 8 eyes. Sensors are just a set of measurements from the world. It has to build a model of the outside world! It's about forming internal representations.

2. There's no learning. It's born out of the box ready to go, optimized by evolution. Mammals are also this way. We need experience to fine-tune it, but it's not born with a pile of mush.

3 questions.

1. How do we see in the presence of fixational eye movements?

   This teaches an important lesson about how active vision works.

   Subject tries to hold eye still. There is still natural movement, but the subject doesn't notice the motion. Image of human retina looking at a "E". Track in real time. They have complete control on stimulus, look at perception at pixel-grain of retina. (Austin Roorda, UCB, adaptive optics. Use mirrors to correct.)

   Is this a bug or a feature?

   You can take the stimulus so that it moves with the retina. If you do that it fades after a second. If you paralyze your eyes, you go blind.

   They have AC response characteristic.

   The motion helps you. Look at perception under 3 conditions: natural, stabilized (move with retina, look at timescale $< 1s$, when it hasn't faded), incongruent (not related to motion of retina).

   Any motion (even incongruent) helps you see better.

   How does the brain do that, take all these spiking messages and make sense of them, build a model?

   We have to average over time, because we have a spike train we have to integrate.

   Simple averaging (integrating) is not sufficient. Movement blurs and destroys the pattern.

Why might movement help? All the photoreceptors are different types, L cones, etc., not packed regularly. You don't want to depend on where the image lands on the retina.

The problem from a math point of view: try to infer S,

$$I(x,t) = S(x - \Delta x(t)) + \varepsilon(x,t) \tag{22}$$
$$\Delta x(t) = \operatorname{argmin}_{\Delta x(t)} |I(x,t) - S(x - \Delta x(t))|^2 \tag{23}$$
$$S(x) = \int I(x + \Delta x(t))\, dt. \tag{24}$$

We don't know the movement. If we have it we can estimate the pattern. If we have the pattern, we can estimate the movement.

Build a probabilistic graphical model (Alex Anderson, Ph.D. thesis). Eye position→spikes, pattern→spikes.

$$\widehat{S} = \operatorname{argmax}_S \ln P(R|S)$$

Build EM algorithm. Given current estimate of position X update S, and vice versa.

Do gradient descent. These are better explained by neural computation. Somewhere in primary vision cortex, we need 2 populations of neurons.

Internal position estimate multiplicatively gates to get internal pattern estimate.

There is evidence that you build up stabilized representations. Cross-correlate current reprsentation with inputs, combine distributions of location and image. This can all be done in a neural circuit.

Demo: Irregular hexagonal sampling, E moves.

Does eye get no feedback about movement? There are signals about the microsaccades, but we're looking at drift. Neural correlates are unclear. If you introduce uncorrelated motion it still helps; the brain is not relying on external signals about motion.

Performance is better with motion (because of gaps in sensors—you get benefit from moving in and out of the photoreceptors. We haven't put in the heterogeneity yet).

$$\widetilde{A} \operatorname{argmax}_A \ln P(R|A) + \ln P(A).$$

We don't think the brain is building up a pixelated representation, it's building in a feature space. Don't infer any type of noise; instead impose a sparse prior on features. Learn a dictionary.

Incorporating a prior helps.

This is a microcosm version of the question of the big saccades.

2. What is the optimal spatial layout of the image sampling array?

   Consider a deep network solving the ImageNet challenge. A scene is not just one thing but a composition of things.

When we give neural network a task and label with a single word, it's an artificial task. You put perception through a linguistic bottleneck. You can write a novel about what you see in these images.

We don't take in scene all at once, we use attention to look at it piece by piece in a serial fashion.

People have built NN models of attention. (DRAW: recurrent neural network for image generation) Have a glimpse window. It trains how to move around the scene to answer the question.

Given you're going to move it around, you do you want to sample within the glimpse window?

Almost all animals have spatially inhomogeneous lattice (retinal ganglion cell sampling array), high-res in middle (fovea), falls off towards edges. Tradeoff between high-resolution and wide field of view? If I spread out uniformly, I have wide field of view but very low resolution (you can't even see the largest letter on the eye chart).

This is the intuitive argument. Can we show that quantitatively?

Cheung, Weiss, Olshausen, 2017. Learning the glimpse window sampling array. Network trained end-to-end to correctly classify digit in scene. To do this it must find a digit and move its gimpse window to that location. It learns how to control its attention.

Start out uniform, and adjust by learning. Train by gradient descent. Glimpse window chooses to move sample nodes, pack in middle with higher resolution. It shrinks size of dots in the middle.

Try datasets with translation, with translation and rotation.

Imagine an animal with a zoom lens. It chooses not to build a fovea.

Looking at sampling interval, get linear increase in eccentricity.

There is intimate connection between foveated sampling system, and movement. It doesn't make sense to build foveated system if you move it around. We can take info from such a system.

3. How is information integrated across multiple fixations (saccades)?

   We have a complete disconnect between perception of world and what's going on inside brain.

   Info coming in is replaced 3–5 times a second.

   Yet it appears to you like the world is stable. It seems like 1 world you see in high-res all the time.

   Where does this exist in the brain, the world-view part of the brain?

   No one has found a neural correlate, part of brain that does this. There's no clear explanation.

   In the meantime, we can try as modelers to do this. What would you do to use these fixations to build a model?

Two things must be encoded and combined at each fixation:

(a) Position at glimpse window.

(b) Contents of glimpse window.

We need to bind these two things together. A scene may then be represented as a superposition of such bindings.

Hyperdimensional computing: an introduction to computing in distributed representation with high-dimensional random vectors, Penti Kanerva. Tony Plate, Holographic reduced representations. (Read Ch. 6) Smolensky had a solution with outer product. It explodes dimensionality of space.

- binding without growing dimensionality
- fully distributed representation
- math framework for storing and recovering information: multiplication for binding, addition for combining, operators and inverses.

Network for binding and combining, Eric Weiss. DRAW is doing this somehow. How is this happening, what is the recurrent neural net doing? Take the location and encode as high-dimensional vector. Bind by elementwise multiplication. Add into short-term memory.

$$m_{t+1} = m_t + r_t \odot v_t$$

We want unified representation space for all info. One motivating idea is to address how to represent a lot of modalities in one workspace. Dimension is high enough. When you have high enough dimensionality, 2 random vectors are almost orthogonal. Number of nearly orthogonal vectors grows exponentially.

For example, $m = v_6 \odot r_{t=0} + v_5 \odot r_{t=1} + v_4 \odot r_{t=2}$. (cf. Russ S. is using this. Multiplicative things seem to help. Maybe this gives a theoretical understanding.)

Multiplying representations is useful.

We can start querying. Ex. where is the "5"? By (near) orthogonality, recover the position of 5, $r_{t=1}$.

Use uncertainty to drive where to move next. It looks in places where it has the most uncertainty.

Spatial reasoning: What is below a '2' and to the left of a '1'? Reframe in algebra to get the answer.

Main points: Active sensing. Binding is an important part; build into short-term memory.

Can you explain optical illusions in computational model? All illusions are on your brain; you see with you brain, not your eye. (Center surround fields?) Most are due to high-level filling-in phenomena.

Are there similar structures in other sensory inputs? In the auditory systems, they put sensors where the action is, 10-20kHz, for cats and dogs it's higher. For bats, 60kHz, a kind of "fovea" in that region. Electric fish has a "fovea" as well. Fingertips (Braille).

30 Resilient Representation and Provable Generalization (Dawn Song, UC Berkeley)

See §6 in notes here, `https://www.dropbox.com/s/d68tepx2x5h13at/representation.pdf?dl=0`.

In this talk, I'll present the challenges in today's deep learning approach for learning representations resilient against attacks. I will also explore the question of providing provable guarantees of generalization of a learned model. As a concrete example, I will present our recent work on using recursion to enable provablely perfect generalization in the domain of neural program architectures.

Current learned representations are not resilient against adversarial attacks. What properties should the learned representation have?

- Smoothness under semantic distance metric? It's a challenge to define what this means.

  (If we perturb in some other way, semantically the image could be different.)

- Detect when test distribution is different from training distribution. We want some kind of confidence bounds.

Do adversarial examples matter? Why should be care about resilient representations? Adversarial examples are just the tip of the iceberg, part of adversarial machine learning.

Attacker always follow footsteps of new tech development or sometimes leads it. Stakes are higher with AI. Security will be one of the biggest challenges. Security of learning systems: software, learning, and distributed levels. The first is well-understood.

Can we define better, new architectures with stronger (provable) generalization and security guarantees?

The example domain is neural program synthesis. Learn neural networks that perform certain programming tests, like addition. All previous approaches suffer from

1. generalization. Neural programmer interpreter (NPI), ICLR 2016 (best paper award) fails for large inputs.

2. no proof of generalization. Even if empirically you observe perfect generalization, you have no proof.

Using recursion, a problem is reduced to sub-problems, with base cases and reduction rules. Learn recursive neural programs!

In a recursive program, a function learns to call itself, reducing the problem to a smaller problem. The recursive neural program learns faster and generalizes better, and recursion enables proof of perfect generalization.

Prove perfect generalization of learned recursive program via verification procedure: Explicitly test on all possible base cases and reduction rules (verification set). Lessons learned:

- Program architecture impacts provability.

This is similar to our experience in the symbolic program domain. Well-designed program with good architectures are easier to prove properties of. Arbitrary programs are difficult to prove properties of.

- Caution for end-to-end monolithic NN: harder to train, generalize, interpret.
- Recursive modular neural architectures are easier to reason, prove, and generalize.
- Explore new architectures and approaches to enable strong generalization and security properties for broader tasks.

  Can we have provable guarantees for more systems?

We should evaluate systems under adversarial events, not just normal events.

In regression testing, run program on normal inputs, prevent normal users from encountering errors. In security testing, run abnormal/adversarial inputs, prevent attackers from finding exploitable errors.

Building large complex systems require compositional reasoning. How to do abstraction, compositional reasoning for non-symbolic programs? How to ensure good local decisions make good global decisions?

AI and security: AI in the presence of attacker. What are some greater responsibilities?

So far I've been talking about attackers attacking AI, causing the system to do the wrong things. We just need security for learning systems.

On the flip side, an attacker can misuse AI to attack other systems. We need security in other systems, which is a harder task.

Deep learning can empower bug finding and phishing attacks.

Cambridge Analytica contributed to success of Trump. Use AI for large-scale, automated, targeted manipulation.

There is asymmetry between attacks and defense. AI progress is uneven in different areas.

Dual problem: stronger AI makes attacks easier.

Compare to nuclear weapons. Treat can help and detection is possible. When attacker is misusing AI, this will be difficult. We have strong responsibilities. At the same time we develop AI techniques, think of techniques to prevent misuse of AI.

Security will be one of the biggest challenges in deploying AI.

QA:

Human psychophysics. If we had brain Jacobians, could we do the same thing against the brain. We can construct adversarial examples in human space. There will always adversarial examples. We call them optical illusions. We want the machine to only have human-type illusions. We want models to act more like humans.

Humans don't have this type of problems, so NNs do not have the desired properties human systems have. You can talk about adversarial examples in other settings.

31 Word Representation Learning without <unk> (closed-vocabulary) Assumptions (Chris Dyer, Carnegie Mellon University)

Languages synthesize, borrow, and coin new words. This observation is so uncontroversially robust that it is charaterized by empirical laws (Zipf's and Heap's Laws) about the distributions of words and word frequencies rather than by appeal to any particular linguistic theory. However, the first assumption made in most work on word representation learning and language modeling is that a language's vocabulary is fixed, with the (interesting!) long tail of forms replaced with an out-of-vocabulary token, <unk>. In this talk, I discuss the challenges of modeling the statistical facts of language more accurately, rather than the simplifying caracature of linguistic distributions that receives so much attention in the literature. I discuss existing models that relax the closed vocabulary assumption, how these perform, and how they still might be improved.

There is power-law distribution, Zipf's law. Rare types are common in language.

Lesson: You are throwing away a lot of the distribution when you "unkify".

People are throwing away more aggressively, ex. throwing away if appear < 5 times.

Heaps' law: you will always see new words.

Lesson: you cannot model language and close the vocabularity.

Without solving AI, can we do something better?

31.1 Form and function: Building word representations compositionally

Set of phonemes are closed.

Arbitrariness (car-c+b=bar), de Saussure, 1916. Different languages picked different words for "car".

At the other end of the spectrum:

Compositionality: look at meaning of expression as composition of parts. (Frege, 1892; Plato) (John dances - John + Mary = Mary dances)

It looks like: at lexical level, arbitrariness, at prepositional level, compositionality. Memorize in the first setting, generalize in the second setting.

Things aren't that simple. We have challenges:

1. Idioms: John saw the football/bucket. John kicked the football/bucket.

2. Morphology: We don't think that we have a lot of word formation processes but we do: cool, coooool, cooooooooool. cat+s=cats, bat+s=bats.

   This is a productive process: teach a kid a new word; kid can produce the plural.

Arbitrariness and compositionality exist at levels, although the tendency is smaller units are more arbitrary and larger units are more compositional. Good learners must be able to cope with both.

Our solution: use LSTMs (nominally a model with good generalization behavior) at lexical level where we usually use "memorization".

If you look at where deep learning models fail, it's when they memorize too much and fail to make the necessary abstractions.

We've never been great at working in languages other than English. Languages have different amounts of morphology.

1. Analytic: Mandarin, English. "cat"
2. Fusional, templatic: to the cat
3. Agglutinative: they caused it not to meow.
4. Polysynthetic: they would carry the cat on their shoulders.

There are pieces that fit together but different from words.

Memorization: take a word and project into 1-hot vector. Project into low-dimension space. This projection matrix is going to have a vector for every word in the language. Each is independent. Different words are independently parametrized as words that are similar.

On the other side, word with generalization model. (At first we didn't think this was sensible.) View "car" as sequence of characters and read as bidirectional LSTM's. Then combine to give vector. We get an effectively infinite matrix.

(You can read a sequence from left to right or right to left. Reading in both directions and combining makes the learning problem easier. In a lot of languages, syntactically interesting parts are prefixes and suffixes. You have gradients impinging on that part more easily.)

In both cases, train parameters using backprop.

Example: dependency parsing. "I saw her duck." (Words point to arguments.)

Have 2 parsers. Compare 2 ways of representing input: word lookup and CharLSTM. We are building tree up as systems of actions. Alternate between shifting words or combining words on stack to treelets.

Replace the first level of the network. Have lookups from table, or have single network giving vectors.

If you make small orthographic changes, does the CharLSTM understand the context? We've done analyses about what these models are learning.

If you make 1-character change, in some words it makes a lot of difference, in some words it doesn't? The evidence suggests this is happening.

Normally when you're reading a word you don't know, you go to dictionary, definition gives some representation. Just have a dictionary and combine synonyms? Dictionaries are complete? Wait till you work on Twitter! We were sick of having to translate. We found 500 ways of spelling "would".

I will come back to this question. If I say "Gorp" and have particular idea in mind, you have no idea what I mean. In general you need models that know what they know or don't know. Sometimes you can infer it's a noun and verb... If you're doing translation you're out of luck. Find a thousand instances where it's used.

For transition-based parsing, CharLSTM>word lookup! (What proportion of words are labeled with the right head?)

You don't need to know very much about meanings of words to do parsing.

You can think of creating interesting adversarial examples.

What about languages with richer lexicons, Turkish and Hungarian? In agglutinative languages, we see much bigger improvements. In fusional/templatic language, CharLSTM is somewhat better. In analytic languages, the models are roughly equivalent. (Chinese has little morphology.)

Also see correlation with how many unks do we see in test set (test set OOV rate).

What is really being learned by transition-based parsing?

- Is it fair to say we are learning anything beyond syntactic classes? (well-known to have morphological reflexes)
- Let's turn to even more basic task, language modeling. (Density estimation on sentences.)

Just change the way we represent the input.

We'll see a similar story. With English we get the same results. Fusional, agglutinative have improvement.

The number of parameters is an order of magnitude less in LSTM models. We should be cautious of parameter budgets in our word embedding models, think about more compressed representations.

Word similarites. Take a word and ask what the model thing the most similar words are. Find words with similar semantic and syntactic properties.

Build embedding for query word. For words in fixed vocab, see 5 nearest neighbors. "John" gets other names of English kings.

This is left-to-right language task.

Your query word can be a non-word (Jabberwocky test). Ex. Noahsire gives Nottinghamshire. phding gives mixing.

Word-color regression: How are colors used in advertising? Dataset of 2 million names colors, facebook for colors. Carrot, firw, Artemis dark, divinely pink, emo love. Train a character-level model to predict color based on name. To train, backpropagate. How to represent colors and measure color distances. RGB colorspace is convenient but perceptually nonuniform. Use Lab (CIELUV) instead. We have good models of what people can perceive in colorspace. A unit distance is supposed to be perceptually uniform.

We can predict color of "bacon lipstick," we can also go the other direction. We evaluated by giving people color Turing test. Model predicts a color. Which is a better color for the name? People like our colors better.

Summary:

- morphological regularities are exploited well. LSTMs are good at composition.
- Mapping between form and function is generally arbitrarily. LSTMs are good at memorizing.
- Lots of follow-up work. Convnets also work.

What about generalizing? Model characters, not words.

We don't lose very much. Best xentropy goes from 1.21 to 1.33. Adding latent discovery of word-like units, 1.24.

31.2 Word reuse: Open vocabulary neural language models

First we need one other observation (Kawakami, D, Blunsom, 2017). Rare words are repeated more often than they should—if you invent a word in an article you reuse it. Once a word has been used once, it's more likely to be used again. "The chance of two noriegas is close to $p/2$ than p^2."

Use caching models: don't just generate words as sequence of letters, also copy something I just generated. This is not new, but no one was using open vocabularies.

Open-vocab caching model has 2 parts.

1. word-level LSTM tracking.
2. feed in reprsentation of previous word.

Cache:

1. Every time we generate word, put in cache in empty slot s or least recently used slot, $k_s = h_t, m_s = w_t$.
2. If already a word there, take an average.

What's the probability of copying from cache? Attend over contents. Make a final decision: how to produce the next word—copy or generate? Condition on current state of LSTM.

Put pieces together: First decide probability of copying, generating. Probability of generating Pokemon from cache and from scratch; add together to get total probability.

On open-vocab, see significant results.

What's the incentive not to copy? Cache limited. There's a key associated with everything in cache. It might just be easier to spell out words one at a time. Copy things that are sensible to be copied, but otherwise it knows how language goes.

We have new dataset, same story. Names tend to get copied. Numbers don't tend to get repeated.

Statistical facts about language; Heaps' law and Zipf's law, church's law (rare are "bursty"). We should and now can account for both of these.

Have you tried spellings that are non-adversarial for humans, and fluent readers have no difficulty reading at the same pace. This models doesn't do this. You could train it. Is the goal to get something that works, or something human-like?

Both ar goals. I wanted to model marginal distribution of sentences (not a cognitive problem, it's an engineering task). We can ask where this comes from (being able to unscramble words). It's about generalizations that V1 makes when it learns to read that give

us insight into how reading works. We could make models like that. This was meant to solve engineering task.

If you start to think about adversarial reading, you would be concerned about that psychophysical fact. You can add this to training set or change regularization.

32 Performance Guarantees for Transferring Representations (Daniel McNamara, Australian National University)

A popular machine learning strategy is the transfer of a representation (i.e. a feature extraction function) learned on a source task to a target task. Examples include the re-use of neural network weights or word embeddings. Our work proposes sufficient conditions for the success of this approach. If the representation learned from the source task is fixed, we identify conditions on how the tasks relate to obtain an upper bound on target task risk via a VC dimension-based argument. We then consider using the representation from the source task to construct a prior, which is fine-tuned using target task data. We give a PAC-Bayes target task risk bound in this setting under suitable conditions. We show examples of our bounds using feedforward neural networks. Our results motivate a practical approach to weight transfer, which we validate with experiments.

Joint work with Nina Balcan, CMU.

I want to automatically create separate photo albums of my dog Rufus and cat Macy! Maybe use a neural net model? But my dog and cat are unique!

When does transfer help you?

We want to learn a task for which labelled data is scarce, but we have abundant data for another related task.

Ex. in computer vision, pre-train and fine-tune.

Let F be class of representations $f : X \to Z$, G a class of specialized classifiers $g : Z \to Y$, $g \in G$. , $H = \{h : \exists f \in F, g \in G, h = g \circ f\}$.

Learn $\widehat{f} : X \to Z$ from source task S. Can we restrict representations F when learning target task T? Use statistical learning theory to provide tighter risk upper bounds for T?

Idea: keep a neighborhood $\widehat{F}$ around $\widehat{f}$.

Try to learn $\widehat{g}_S \circ \widehat{f} \in H$ on S and extract $\widehat{f} \in F$. Then conduct empirical risk minimization over

$$G \circ \widehat{f} = \left\{g \circ \widehat{f} : g \in G\right\}$$

on T to get $\widehat{g}_T = \text{argmin}_{g \in G} \widehat{R}_T(g \circ \widehat{f})$.

Theorem 32.1 (Risk upper bound for fixed rep)**.** *Suppose we have a transfer function $\omega : \mathbb{R} \to \mathbb{R}$ and transferability property: if I do well on the first task, it gives a bound on how well I do on second,* $\min_{g \in G} R_T(g_T \circ \widehat{f}) \leq \omega(R_S(\widehat{g}_S \circ \widehat{f}))$*, then* $R_T(\widehat{g}_T \circ f) \leq \omega(\widehat{R}_s(\widehat{g}_S \circ \widehat{f}) + ...)$.

Bound is tighter than learning T from scratch and using VC dimension-based risk bound.

On the target task, fix the representation.

Ex. fix first level of weights in neural net.

Assume a shared representation exists, $\exists f \in F$, $g_S, g_T \in G$, $\max[R_S(g_s \circ f), R_T(g_T \circ f)] \leq \varepsilon$. But is the f we learned good?

Theorem 32.2 (ω for neural network, fixed representation)**.** $\omega(R) := cR + \varepsilon(1+c)$. *For all* $\widehat{g}_S \in G$, $\min_{g \in G} R_T(g \circ \widehat{f}) \leq \omega(R_S(\widehat{g}_S \circ \widehat{f}))$.

Representation fine-tuned using target task: PAC-Bayes result bounds generalization error using KL divergence between prior and posterior hypothesis. Want for all $\widehat{h}$, $KL(\widetilde{h}||\widetilde{h}_{\widehat{g_S} \circ \widehat{f}}) \leq \omega(R_S(\widehat{g}_S \circ \widehat{f}))$.

Theorem 32.3 (Risk upper bound with fine-tuning)**.** *Given this,* $R_T(\widetilde{h}) \leq \widehat{R}_T(\widehat{h}) + \cdots$

Modified regularization penalty: penalize for diverging from source task. It could vary for different layers. Ex. let higher layers vary more.

Set up task to test this out. Make MNIST into binary classification task, divide into source and target tasks.

MNIST: best results from using regularization penalty. Baseline: learning target task from scratch.

Newsgroups: low level groups already encoding disjunctions because of sparseness. Weights were less transferable.

Conclusion:

- Step towards theoretical foundation for transferring representations, both with and without fine-tuning.
- Theory motivates tranfer regularization penalty to prevent target task overfitting.