Deep Learning of Representations: Looking Forward

Yoshua Bengio

Summary and Comments

Klemen Simonic

So Far ...

Structured Output:

- Binary and Multiclass
- Sequence
- Graphs

=> Complex Output

So Far ...

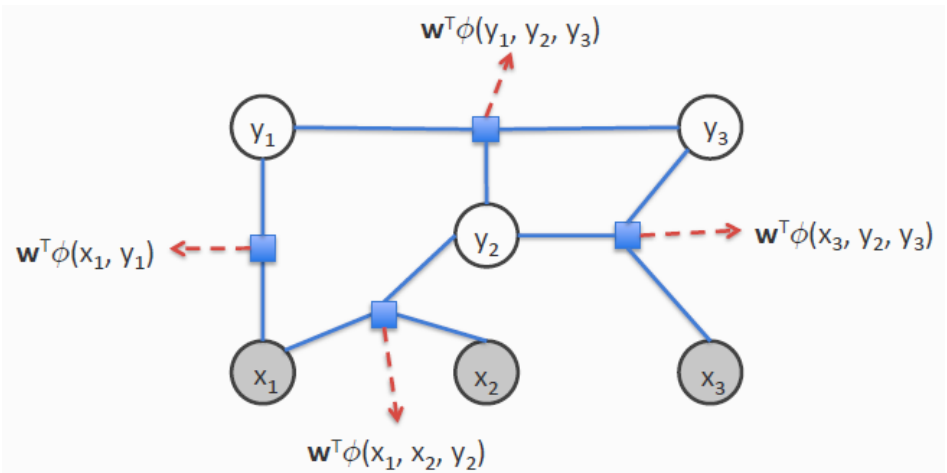
Learning Problem (1):

- Input: $\Phi(x, y)$
- Learn w to assign weights to each feature in $\Phi(x, y)$
- Score: $\langle w, \Phi(x, y) \rangle$

So Far ...

Learning Problem (2):

- Input: *Φ(x, y)*
- Decompose the *Φ(x, y)* and *w* into **manually** designed parts (factors)
- Learn *w* to assign weights to parts
- Score: summing all the parts



$$P(\mathbf{y}|\mathbf{x}) = \frac{\exp\left(\mathbf{w}^T\phi(\mathbf{x},\mathbf{y})\right)}{\sum_{\hat{\mathbf{y}}}\exp\left(\mathbf{w}^T\phi(\mathbf{x},\hat{\mathbf{y}})\right)}$$

$$\phi(\mathbf{x}, \mathbf{y}) = \phi(x_1, y_1) + \phi(y_1, y_2, y_3) + \phi(x_3, y_2, y_3) + \phi(x_1, x_2, y_2)$$

So Far ...

Provide:

=> Pretty simple input

=> Pretty simple learning

Want:

=> Very complex output

→←

Representation Learning

Can we learn $\Phi(x, y)$ for the a particular problem?

$\Phi(x, y)$:

- Consists of multiple levels
- Decomposes / factorizes the input appropriately

Not just $\Phi(x, y)$, but can we also learn the inference algorithm for final prediction?

Deep Learning Background

- Discovery of multiple levels of representations
- Big empirical success in AI applications
- Google, Microsoft, Apple, IBM, Baidu
- Applications:
  - Speech recognition
  - Image recognition
- Lots of attention in the news

Deep Learning Background

- ML methods “completely” depend on the choice of data representation (features)
- Most of the time spend on hand-crafted representation
- Unable to extract and organize the discriminative from the data

- Learning algorithms that can extract representation “automatically” => discover explanatory factors or features.

Overview of DL Algorithms

Automated discovery of abstraction

Why:

Higher level or more abstract representations tend to be more useful - represent the semantic content of the data, separated from low-level features of raw data.

Overview of DL Algorithms

Deep Supervised Nets:

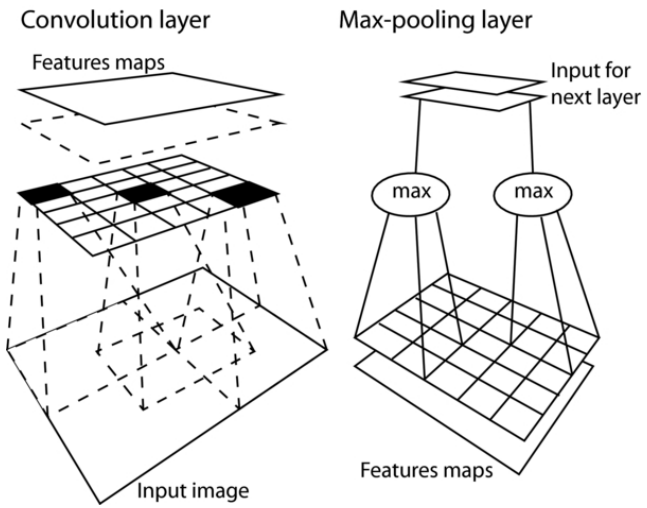
- Before 2006, training deep supervised networks was too difficult

  Breakthrough: unsupervised pre-training by RBMs
- Recently, proper initialization is enough

Overview of DL Algorithms

Convolutional Architectures:

- Alternates **convolutional layers** with **pooling layers**
- Units are associated with **spatial** and **temporal** positions
- Share parameters with other units of the same “type” located at different positions.



Overview of DL Algorithms

Dropouts:

- Injecting noise that prevents a too strong co-adaptation of hidden units
- Hidden units must compute a feature that will be useful even half of the other hidden units are stochastically turned off.
- Powerful regularizer

Overview of DL Algorithms

Layer-wise Pre-Training:

- Deep compositions of non-linearities can be very sensitive to initialization
- Greedy layer-wise pre-training: train one layer at the time, starting from lower layers.
- The new representations is reused for deeper layers.

Overview of DL Algorithms

Auto-encoders:

- Idea: learn a reconstruction function $r(x) = g(f(x))$
  f - encoding function
  g - decoding function
- Training criteria: $|| r(x) - x ||^2$
- Regularization, otherwise $r(x) = x$
- Much less hidden units than input / output => the hidden units learn the good basis for the data.

Scaling Computations

How do we scale to:

- Larger models?
- Huge datasets?

Why?:

- Richer models capture larger amount of information
- Many important AI problems to solve
- AI means to roughly understand the world around us at the same level of competence as humans

Scaling Computations

- Single computer capabilities are not sufficient
- Distributed computation engines
- GPUs are advancing
  (Nvidia Tesla K80, 24GB RAM, 2.91 Tflops double precision)

Scaling Computations

Parallel Updates: Asynchronous SGD:

- Train multiple versions of the model in parallel, each running on a different node and seeing different subsets of the data
- Asynchronous lock free mechanism, which keeps different version of the model not too far from each other.
- When scaling to large number of nodes, it becomes inefficient due to large exchange of parameters over the network.

Scaling Computations

Sparse Updates:

- For any mini-batch, there is only a small fraction of parameters that are updated.
- Smaller fraction of parameters need to be exchange between nodes.
- Obtained if the the gradient is very sparse.

Scaling Computations

Conditional Computation:

- Instead of dropping out paths independently and at random, drop them in a learned and optimized way.
- Decision trees: prediction time on the order of logarithm of the number of parameters.
- Idea: discard large amount of parameters.

=> Truly sparse activations.

Optimization

- Training deep neural networks involves a difficult optimization
- Not clear: local minima vs. ill-conditioning
- Training signal provided by backprop is sometimes to weak to properly train intermediate layers of deep network => pretraining (RBM).

Inference

Problems:

- Iterative procedures that can not be parallelized
- Potentially huge number of modes
  (highly multimodal distributions / functions)
- Mixing between modes (MCMC modes)

Inference

Mixing between modes is easier at higher levels of hierarchy:

Train the DNR and mix the modes:

1.) Run the MCMC in a high level representation space
2.) Project back in raw input space the obtained samples at that level

Distribution at higher level has more entropy (more uniform distribution) and better separated the underlying factors of variation.

Inference

Learning approximate inference:

Idea: Learn fast approximate inference + model

Traditional view of PGM:

- Separation between the modelling, optimization, inference and sampling.
- Rather, bring learning into inference and jointly learn the approximate inference and the generative model.

Inference

Avoiding inference and explicit marginalization over latent variables:

$$P(y|x) = \Sigma\, P(y|h)P(h|x)$$

- Avoid marginalization over latent (hidden) variables
- We directly approximate the function P(y|x)
- No need to for explicitly going through many configurations of the latent variables

Questions ???