How we went from graphical models to Deep Learning and what changed

Christian Wolf
Decision making

\[ \text{dog, cat, bike, avocado, blender, …} \]

\[ \{0, 1, \ldots 26, 27, 28, \ldots, 98, 99, \ldots\} \]

\[ \text{Left, right, ahead, back, pick up, …} \]

Motor commands

“\text{A blue parrot with a yellow belly sitting on a branch in a forest}”
We would like to **learn** to predict a value $y$ from observed input $x$

$$y = h(x, \theta)$$

Handcrafted from domain knowledge

Learned from data or interactions

Fully handcrafted

Fully Learned
Deep neural networks
The 3 fundamental problems of ML

1. Expressivity
   – What is the complexity of functions my model can represent?

2. Trainability
   – How easy is it to fit my model to my training data?

3. Generalization
   – Does my model generalize to unseen data?
   – In presence of shifts in distributions?

(After Eric Jang & Jascha Sohl-Dickstein)
Illustration: model fitting and generalization

How do we choose model complexity?

1.1. Example: Polynomial Curve Fitting

Figure 1.4 Plots of polynomials having various orders $M$, shown as red curves, fitted to the data set shown in Figure 1.2.

The root mean square (RMS) error defined by

$$E_{\text{RMS}} = \sqrt{\frac{1}{N} E(w^\star)}$$

in which the division by $N$ allows us to compare different sizes of data sets on an equal footing, and the square root ensures that $E_{\text{RMS}}$ is measured on the same scale (and in the same units) as the target variable $t$.

Graphs of the training and test set RMS errors are shown, for various values of $M$, in Figure 1.5. The test set error is a measure of how well we are doing in predicting the values of $t$ for new data observations of $x$.

We note from Figure 1.5 that small values of $M$ give relatively large values of the test set error, and this can be attributed to the fact that the corresponding polynomials are rather inflexible and are incapable of capturing the oscillations in the function $\sin(2\pi x)$.

Values of $M$ in the range $3 \leq M \leq 8$ give small values for the test set error, and these also give reasonable representations of the generating function $\sin(2\pi x)$, as can be seen, for the case of $M = 3$, from Figure 1.4.

[C. Bishop, Pattern recognition and Machine learning, 2006]
[Yosinski, Clune, Bengio, Lipson, "How transferable are features in deep neural networks?", 2014]
Yosinski, Clune, Bengio, Lipson, "How transferable are features in deep neural networks?", 2014

Numerical/optimization issues (Positive or negative)
High dimensional input and/or output

Sequences

Images and other 2D grids

Kinematic trees

Multi-label Problems
How did we address this in the past?

\[ y = h(x, \theta) \]

Large numbers of variables, observations, parameters:

\[ \{y_1, y_2, \cdots, y_N\} = h(x_1, x_2, \cdots, x_N, w_1, w_2, \cdots, w_N) \]

Probabilistic Graphical Models

\[ p(x, y|\theta) \quad p(y|x, \theta) \]

Generative Models

Discriminative models
Hidden Markov Models:

\[ p(y, x) = p(y_1) \prod_{i=1}^{N-1} p(y_i | y_{i-1}) \prod_{i=1}^{N} p(x_i | y_i) \]

Bayesian Networks

\[ p(v) = \prod_{i=1}^{N} p(v_i | v_{\text{parents}(i)}) \]

Markov Random Fields:

\[ p(y, x) = \frac{1}{Z} \left\{ - \sum_{i} U(x_i, y_i) - \lambda \sum_{i \sim j} B(x_i, x_j) \right\} \]

Conditional Random Fields:

\[ p(y|x) = \frac{1}{Z(x)} \left\{ - \sum_{c} f_c(y_c, x) \right\} \]
RBM – Restricted Boltzmann machines

HMM – Hidden Markov Models

Bayesian Networks (BN)

Dynamic Bayesian Networks (DBN)

Pairwise Markov chains (PMC)

MRF – Markov Random Fields

CRF – Conditional Random Fields

HCRF – Hidden Conditional Random Fields

CRF – Conditional Random Fields (Structure linéaire)
Back to neural networks

400x400 pixels

500 units

Flattened into 160,000 values

500 * 160,000 = 80 Million parameter (for a single layer)
We need inductive bias

Parameters for one part of the image do not generalize to other parts of the image.
A linear and shift invariant operator is equivalent to a convolution with the impulse response of the operator!

\[
[\phi(f)](x, y) = \phi \left( \sum_{m=-M/2}^{M/2} \sum_{n=-N/2}^{N/2} f(m, n) \phi^{m,n,p} \right) (x, y)
\]

\[
= \sum_{m=-M/2}^{M/2} \sum_{n=-N/2}^{N/2} f(m, n) \phi^{m,n,A} (x, y)
\]

\[
= \sum_{m=-M/2}^{M/2} \sum_{n=-N/2}^{N/2} f(m, n) \phi^{m,n,S} (x, y)
\]

\[
= \sum_{m=-M/2}^{M/2} \sum_{n=-N/2}^{N/2} f(m, n) \phi^{m,n,h} (x, y)
\]

\[
= \sum_{m'=-M/2}^{M/2} \sum_{n'=-N/2}^{N/2} f(x - m', y - n') \phi^{m',n'} (x, y)
\]

[\text{Dirac at position } m,n, \text{ Linearity, Shift operator, Impulse response}]

\[\text{Jähne, 1997}\]
Breakthrough: convolutions introduce an inductive bias for imaging/vision applications.
Recurrent neural networks (RNNs)

Prediction: feed-forward computation in a DAG. No optimization is needed.
GMs vs. NNs

Graphical models:
- state is stochastic
- make it easier for experienced practitioners to model known relationships between data.
- optimization required for prediction
- specific structures allow to obtain global optima with message passing (chains and trees) or graph cuts (submodular potentials) etc.

Neural networks:
- state is deterministic
- complex models with componential hidden states
- higher order interactions are easier to handle
- no optimization during prediction

Is it better to get a global min/max of a simple model or a feed-forward prediction for a high-capacity model trained on a large amount of data?
What stays the same:
- We still model expressivity: inductive biases and interactions between variables
- We still solve optimization problems
- We still care about generalization

What changed:
- Our models are far more expressive
- Most (all?) of the fitting/optimization is done during training
- Fitting at runtime has been replaced by highly parametrized feed-forward calculations
- We care less about optimality of the optimization process
We went from

Global optimization and global coherence for a single instance / image

to

Feed forward calculations with high-capacity models fitted on many images

---

Yann LeCun
@y_lecun

Replying to @earmymturns @kchonyc @yoavgo

Global top-down inference will come back. But I haven't met a top-down inference that cannot be approximated efficiently by a suitably designed recurrent net.

1:37 PM - 13 Oct 2018
Before: image segmentation

Markov Random Fields / Conditional Random Fields

\[ p(x, y) = \frac{1}{Z} \exp \left\{ - \sum_{i} U(x_i, y_i) - \lambda \sum_{i \sim j} B(x_i, x_j) \right\} \]

[Geman and Geman, PAMI 1984]
[Kolmogorov and Zabih, PAMI 2004]
Now: interactions through bottleneck layers
Before: Deformable Parts Models

- Model an object/human/activity as a collection of local parts
- Optimize over (latent) local part positions

\[
\sum_{i=0}^{n} F_i' \cdot \phi(H, p_i) - \sum_{i=1}^{n} d_i \cdot \phi_d(dx_i, dy_i) + b,
\]

Local appearance  Deformation

[Felzenszwalb et al., PAMI 2010]
Now: no decomposition into parts

[Erhan et al., 2014] « Multibox »
[Redmon et al., 2016] « YOLO »
[Erhan et al., 2016] « SSD »
Now: attention models

[Durand, Mordan, Thome, Cord, CVPR 2017 ]
Now: attention models

Part positions are predicted during test time and never questioned.

[Baradel, Wolf, Mille, Taylor, CVPR 2018]
Now: attention models

Unsupervised pretraining

3D Global model: Inflated Resnet 50

RGB input video

Discriminative

Pose prediction

Pose attraction

Discriminative

Spatial Attention process

External memory

Soft-assignment of glimpses over workers

[Baradel, Wolf, Mille, Taylor, CVPR 2018]
Relational Reasoning

[Baradel, Neverova, Wolf, Mille, Mori, ECCV 2018]
Goal: learning a joint function or distribution over $x_i$

$$g(x_1, x_2, \ldots, x_N)$$

Classical solution: graphical model / energy function

$$p(x) = \frac{1}{Z} \exp \left\{ - \sum_i U(x_i) - \alpha \sum_{i\sim j} B(x_i, x_j) \right\}$$
Relational Reasoning

\[ g(x_1, x_2, \ldots, x_N) = \max(h(x_1), h(x_2), \ldots, h(x_N)) \]

“PointNet” [Qi, Su, Mo, Guibas, CVPR 2017]
Defined over points of a point cloud

\[ g(x_1, x_2, \ldots, x_N) = \sum_{i,j} h(x_i, x_j) \]

“Relational Reasoning” [Santoro et al., NIPS 2017]
Defined over feature map cells
\((x_i, x_j)\)
Object level Visual Reasoning

\[ g_t = \sum_{j,k} h_\theta(o_{t'}^j, o_t^k) \]

[Baradel, Neverova, Wolf, Mille, Mori, ECCV 2018]
Learned interactions

Class: person-book interaction

[Baradel, Neverova, Wolf, Mille, Mori, ECCV 2018]
Conclusion

Problems to solve:
- Expressivity: model symmetries and invariances in your data
- Trainability
- Generalization

Coming back: Geometry, Causality, Uncertainty, Explainability