23 - Open Problems

Ludwig Krippahl

Open Problems

Today, last lecture

- (Some) Open problems in deep learning:

- Automated Architecture Search
- Verification & Validation
- Training with small data sets
- Bridging the neuro-symbolic gap

9:45 Introduction to PyTorch

- Prof. Cláudia Soares, Zoom session
- Afterwards, questions about assignment 2

Automated Architecture Search

Automated Architecture Search

AutoML

- Automating machine learning:

- https://automl.github.io/auto-sklearn/master/
- Test different algorithms and parameters to optimize classification

- In classical ML additional problems of feature engineering and selection
- Deep learning should be better for this

- Deep neural networks are good at finding best features

- However...

Automated Architecture Search

Finding the best network

- A discontinuous optimization problem; needs a good strategy.

- Evolutionary methods, such as genetic algorithms
- Reinforcement learning:the agent performs a sequence of actions to build the network
- Bayesian optimization: maximizes a black box function by fitting estimates of its output
- E.g. Sequential Model-based Algorithm Configuration (SMAC), uses random forest to predict performance
- Was used to build MLP that perform better than human designed in some applications

Automated Architecture Search

Finding the best network

- A discontinuous optimization problem; needs a good strategy.
- Evaluating networks is expensive; needs speedup

- Train few epochs or on small subsets of data
- Extrapolate performance from first training epochs
- Inherit weights transforming the architecture in ways that preserve function
- E.g add a layer with identity operation
- Share weights between different models that are subsets of a large original trained model

Automated Architecture Search

Finding the best network

- A discontinuous optimization problem; needs a good strategy.
- Evaluating networks is expensive; needs speedup
- Choose the search space.

- We need to know what elements to use in order to search them
- Convolution, residual blocks, recurrent, attention and transformers...
- And it is a huge search space...

Validation and Verification

Validation and Verification

- Software V & V is important, especially in critical applications
  - Autonomous driving, medical diagnosis, credit risk prediction, ...
- Validation
  - Assessment of the conformity to the requirements

"Is the software being built correctly?"

- Verification
  - Assessment of the adequacy of the software to the use it will be put to

"Is the right software being built?"

Validation and Verification challenges in DNN

- Very large state-space for the data and network responses

- Difficult to estimate how DNN responds in anomalous situations (with fatal consequences)
- Possible solutions: probabilistic models, process control methods establishing safety limits

- Testing specifications

- Not easy to specify adequate tests for deep neural networks
- Genetic algorithms and other forms of test data generation are possible solutions

- Formal methods are used in software for critical applications

- Formal descriptions of algorithms and requirements enable automated proofs
- This is hard for software in general, and more so for neural networks

Dataset Size

Dataset Size

Lots of data

- DNN need large datasets. Or do they? (We don't...)
- Few-shot and one-shot learning

- DNN trained on large data sets, generally with metric learning
- Learn to separate different examples and put close together similar ones
- Can then be applied to different datasets, even with different classes

- Data augmentation, whenever possible
- Regularization: DNN can easily overfit but dropout and weight penalties can help mitigate

- The network can function as an ensemble

- The loss function:

- Cosine loss function seems to improve generalization with small data sets

The neuro-symbolic gap

Neuro-symbolic gap

- Neural networks are simultaneously very basic algorithms
  - Composition of products, sums and little else
- But very complex, like our brains
  - Composition of many many basic operations
- Our brains connect sub-symbolic representations with symbolic reasoning
  - We can describe and explain
  - We can use symbolic representations to guide parts of our network
  - For example: learn to pick out pictures of boats; then pick only yellow ones
- Can we do this with ANN?
  - Mapping from network activations to concepts, apparently yes
  - But the other way around? Can we talk to the networks in symbols?

Summary

Open Problems

Summary

- Automating network design
- Validating and Verifying
- Learning with fewer data
- Use symbolic information to guide networks

Next week:

- No lecture, just questions and revisions