Reinforcement learning of conditional computation policies for neural networks

Emmanuel Bengio, Pierre-Luc Bacon, Ryan Lowe
Joelle Pineau, Doina Precup

June 23, 2016

McGill

Motivation

- Using RL to take decisions inside deep architectures
- Running high capacity models on low-end devices
  - Phones, mobile devices, electric wheelchair, etc.
- Large networks, fast evaluation?
  - sparse models
  - pruning, quantization
  - lazy/conditional evaluations

Conditional Computation

- Learn a *gater* function
  $\rightarrow$ which parts of the model to evaluate
  $\rightarrow$ which parts are useful
- Learn the main model concurrently

Conditional Computation with RL

- Build model with parameters θ
- Divide parameters/computation into disjoint subsets of θ/H
- Learn a *gating policy* $\pi_\omega(x)$ with separate parameters ω
  $\rightarrow$stochastic policy with binary actions (on/off)
  $\rightarrow$one action per subset
  $\rightarrow$state space is x
- Learn the main model(θ) concurrently

The REINFORCE estimator

k-Bernoulli policy:

$$\sigma = \text{sigm}(W^{(\omega)}\mathbf{x} + b^{(\omega)})$$

$$u_i \sim \text{Bern}(\sigma_i)$$

$$\pi(\mathbf{u} \,|\, \mathbf{x}) = \prod_{i=1}^{k} \sigma_i^{u_i} (1 - \sigma_i)^{(1-u_i)}$$

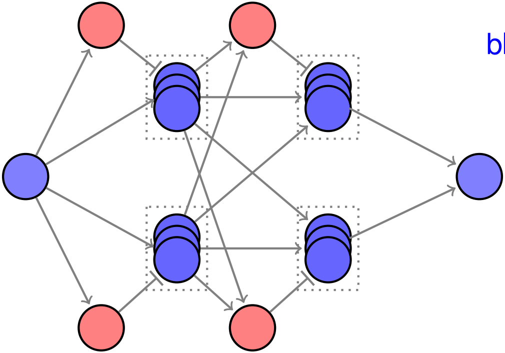
$$\nabla_\omega \mathcal{L} = \sum_{j}^{minibatch} (cost(\mathbf{x}_j) - b) \nabla_\omega \log \pi_\omega(\mathbf{u}_j | \mathbf{x}_j)$$

b is an exponential moving average of the costs.

Policy Regularization

- $L_b = \sum_j^n \|\mathbb{E}\{\sigma_j\} - \tau\|_2$
  each unit is τ in $\mathbb{E}$ over the $\mathbf{x}$s
- $L_e = \mathbb{E}\{\|(\frac{1}{n}\sum_j^n \sigma_j) - \tau\|_2\}$
  the mean of units is τ for some $\mathbf{x}$
- $L_v = -\sum_j^n \text{var}_i\{\sigma_{ij}\}$
  encourage input-dependent units

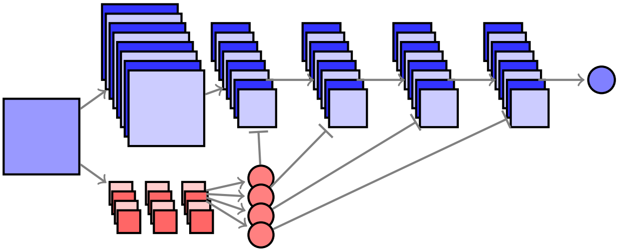
Fully-Connected Architecture



red: policy
blue: network

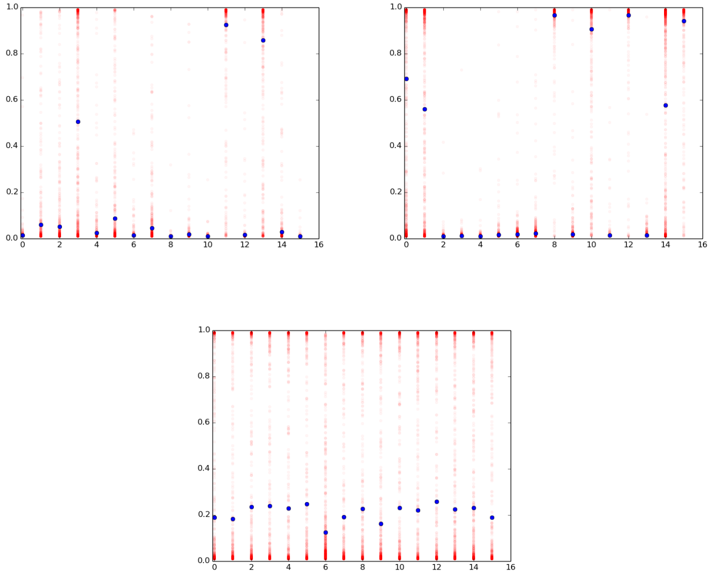
Convolutional Architecture

red: policy
blue: network



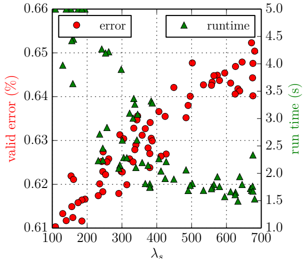
All kinds of parametrizations are possible

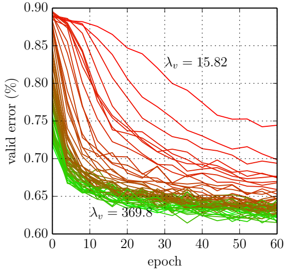
Policies (fully connected, MNIST)



Policy Regularization

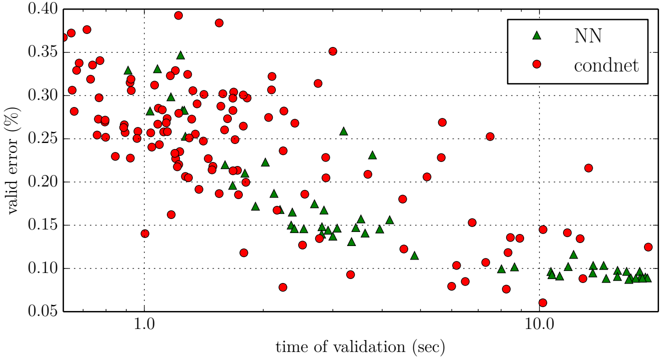
- $L_b = \sum_j^n \|\mathbb{E}\{\sigma_j\} - \tau\|_2$
  each unit is τ in $\mathbb{E}$ over the $\mathbf{x}$s
- $L_e = \mathbb{E}\{\|(\frac{1}{n}\sum_j^n \sigma_j) - \tau\|_2\}$
  the mean of units is τ for some $\mathbf{x}$
- $L_v = -\sum_j^n \text{var}_i\{\sigma_{ij}\}$
  encourage input-dependent units





Fully-Connected Results

- MNIST, CIFAR-10, SVHN
- Same or better accuracy than conventional NN
- up to 5× faster
- >25-50× less computations



Convnet Results

model	test error	N	τ	test time
conv-condnet	**.157**	4	0.5	1.03s
conv-condnet	.167	4	0.3	0.84s
conv-condnet	.176	4	0.2	0.66s
conv-condnet	.173	2	0.5	**0.58s**
conv-NN	.159	4	-	1.07s

CIFAR-10 results for conditional convnets

Conclusion

- It works!
- Similar accuracy, lower forward-pass time
- Much less computations being done
  (25% active nodes $\rightarrow$ ~6.25% computations, 10%$\rightarrow$1%)

- Does not scale very well (REINFORCE?)
- Hard to compete with dense computations
- Chicken and egg problem during learning