Stochastic Computation Graphs: optimization and applications in NLP

Maksim Kretov

MIPT, Deep learning lab & iPavlov.ai

24 Nov 2017

Outline

Table of Contents

Computation Graphs in ML

Naive deterministic view

Given: Data i.i.d. X (inputs), targets T (hidden variables).
Goal: Design policy $f = f_\theta(x) : X \to T$.
Algorithm: Optimize f_θ w.r.t. user-specified loss function L. It embeds all specificity of task, including decision design (being risk-averse, etc).

Computation Graph

Function f. No obvious need to introduce stochastic nodes in the graph. All is deterministic.

$$\theta, X \longrightarrow \boxed{f} \longrightarrow \boxed{L} \longleftarrow T$$

Computation Graphs in ML

Probabilistic view

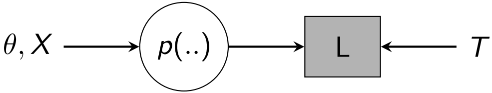
Given: Data i.i.d. X (inputs), targets T (hidden variables).
Goal: Design policy $f = f_\theta(x) : X \to T$.
Algorithm: Two-step process:

1. Infer $p(X, T|\theta)$
2. Use $p(X, T|\theta)$ and L to form decision function

Let's ignore p.2 from now on.

Computation Graph

Includes joint $p(X, T|\theta)$ or conditional parametric distribution.
Still no stochastic nodes, let's dive deeper.



Computation Graphs in ML

Probabilistic view: Maximum Likelihood Estimation

Ignore priors on θ, point estimate.

$$\theta^{MLE} = argmax_{\theta} p(X, T|\theta)$$

Discriminative approach

Don't model $P(X)$, consider X as 'fixed'.

$$\theta^{MLE} = argmax_{\theta} p(T|X, \theta)$$

Computation Graph

Let's now focus on $p(T|X, \theta)$.

Computation Graphs in ML

MLE, discriminative approach

Modelling $p(T|X,\theta)$. Let's use i.i.d. assumption:

$$\log p(T|X,\theta) = \log \prod_{i=1}^{N} p(t_i|x_i,\theta) = \sum_{i=1}^{N} \log p(t_i|x_i,\theta)$$

Introducing latent variables

Omitting i hereinafter to unclutter notation:

$$\sum \log p(t|x,\theta) = \sum \log \int p(t|x,z,\theta)p(z|x,\theta)dz \geq$$

$$\geq \sum \mathbb{E}_{z\sim p(z|x,\theta)}[\log p(t|x,z,\theta)] = \sum \mathcal{L}(x,t,\theta)$$

Computation Graphs in ML

MLE, discriminative approach, latent variables

Optimization of $p(T|X,\theta)$ w.r.t. θ:

$$\log p(T|X,\theta) = \sum \log p(t|x,\theta) \geq \sum \mathbb{E}_{z\sim p(z|x,\theta)}[\log p(t|x,z,\theta)]$$

Stochastic Computation Graph (SCG)

We received expression of the following form:

$$S(x,\theta) = \mathbb{E}_{z\sim p(z|x,\theta)}[f(x,z,\theta)]$$

Such quite general form we are going to attribute to SCG.
$\Rightarrow$ Lower bound for likelihood can be viewed as SCG with reward being equal log likelihood of target label (MLE framework).

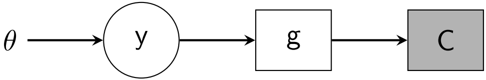
Stochastic Computation Graphs

Directed acyclic graphs that include both deterministic functions and conditional probability distributions.

Definitions

$\mathcal{C}$ – set of cost nodes, $\mathcal{S}$ – set of stochastic nodes.
$\theta \prec w$ means there is a path from θ to w.
$\theta \prec^D w$ means there is a path from θ to w and every such path contains only deterministic nodes.
DEPS_w – set of stochastic and input nodes that deterministically influences w.

Example



$S = \mathbb{E}_{p(y)}[C(g(y))]$

Stochastic Computation Graphs

Directed acyclic graphs that include both deterministic functions and conditional probability distributions.

Definitions

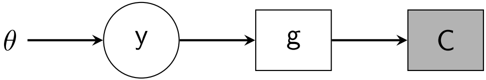
$\mathcal{C}$ – set of cost nodes, $\mathcal{S}$ – set of stochastic nodes.

$\theta \prec w$ means there is a path from θ to w.

$\theta \prec^{D} w$ means there is a path from θ to w and every such path contains only deterministic nodes.

DEPS_w – set of stochastic and input nodes that deterministically influences w.

Example



$S = \mathbb{E}_{p(y)}[C(g(y))]$

Stochastic Computation Graphs

Directed acyclic graphs that include both deterministic functions and conditional probability distributions.

Definitions

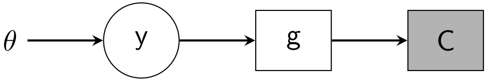
$\mathcal{C}$ – set of cost nodes, $\mathcal{S}$ – set of stochastic nodes.

$\theta \prec w$ means there is a path from θ to w.

$\theta \prec^D w$ means there is a path from θ to w and every such path contains only deterministic nodes.

DEPS_w – set of stochastic and input nodes that deterministically influences w.

Example



$S = \mathbb{E}_{p(y)}[C(g(y))]$

Stochastic Computation Graphs

Directed acyclic graphs that include both deterministic functions and conditional probability distributions.

Definitions

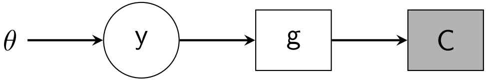
$\mathcal{C}$ – set of cost nodes, $\mathcal{S}$ – set of stochastic nodes.

$\theta \prec w$ means there is a path from θ to w.

$\theta \prec^D w$ means there is a path from θ to w and every such path contains only deterministic nodes.

DEPS_w – set of stochastic and input nodes that deterministically influences w.

Example



$S = \mathbb{E}_{p(y)}[C(g(y))]$

Stochastic Computation Graphs

Directed acyclic graphs that include both deterministic functions and conditional probability distributions.

Definitions

$\mathcal{C}$ – set of cost nodes, $\mathcal{S}$ – set of stochastic nodes.

$\theta \prec w$ means there is a path from θ to w.

$\theta \prec^D w$ means there is a path from θ to w and every such path contains only deterministic nodes.

DEPS_w – set of stochastic and input nodes that deterministically influences w.

Example

$$\theta \longrightarrow y \longrightarrow g \longrightarrow C$$

$S = \mathbb{E}_{p(y)}[C(g(y))]$

Stochastic Computation Graphs

Why view task in terms of SCG?

1. SCG representation is rich
2. Calculating unbiased estimates of gradient for arbitrary SCG is easy

Gradient of SCG

$$\frac{\partial}{\partial\theta}\mathbb{E}\left[\sum_{c\in\mathcal{C}} c\right] = \mathbb{E}\left[\sum_{w\in\mathcal{S},\theta\prec^D w}\left(\frac{\partial}{\partial\theta}\log p(w|\mathrm{DEPS}_w)\right)\sum_{c\in\mathcal{C},w\prec c} c + \sum_{c\in\mathcal{C},\theta\prec^D c}\frac{\partial}{\partial\theta}c(\mathrm{DEPS}_c)\right]$$

Stochastic Computation Graphs

Why view task in terms of SCG?

1. SCG representation is rich
2. Calculating unbiased estimates of gradient for arbitrary SCG is easy

Gradient of SCG

$$\frac{\partial}{\partial\theta}\mathbb{E}\left[\sum_{c\in\mathcal{C}} c\right] = \mathbb{E}\left[\sum_{w\in\mathcal{S},\theta\prec^D w}\left(\frac{\partial}{\partial\theta}\log p(w|\mathrm{DEPS}_w)\right)\sum_{c\in\mathcal{C},w\prec c} c + \sum_{c\in\mathcal{C},\theta\prec^D c}\frac{\partial}{\partial\theta}c(\mathrm{DEPS}_c)\right]$$

Stochastic Computation Graphs

Why view task in terms of SCG?

1. SCG representation is rich
2. Calculating unbiased estimates of gradient for arbitrary SCG is easy

Gradient of SCG

$$\frac{\partial}{\partial\theta}\mathbb{E}\left[\sum_{c\in\mathcal{C}} c\right] = \mathbb{E}\left[\sum_{w\in\mathcal{S},\theta\prec^D w}\left(\frac{\partial}{\partial\theta}\log p(w|\mathrm{DEPS}_w)\right)\sum_{c\in\mathcal{C},w\prec c} c + \sum_{c\in\mathcal{C},\theta\prec^D c}\frac{\partial}{\partial\theta}c(\mathrm{DEPS}_c)\right]$$

SCG reduction, example N1

MLE, discriminative classifier, latent variables

Original formulation:

$$\log \int p(t|x, z, \theta)p(z|x, \theta)dz$$

Lower bound formulation

$$\mathbb{E}_{z \sim p(z|x,\theta)}[\log p(t|x, z, \theta)]$$

Second one seems like SCG-type expression.

SCG reduction, example N1

MLE, discriminative classifier, latent variables

MLE: SCG reduction

1. Need to optimize $\log p(t|x, \theta)$
2. Build SCG with inputs x, intermediate stochastic nodes z
3. Set cost node $c = \log$ and output node is distribution over t
4. PROFIT!

Final expression for SCG will have the same form, so is the gradient.

Discussion

Benefits of simplification

- Easier to build complex models

SCG reduction, example N2

Example from DeepBayes school, lecture of Sergey Bartunov.

Reinforcement learning: MDP as a probabilistic model

Latent variables are $z = (s_{1:T}, a_{1:T})$. Observed variables are auxiliary variables $R_{1:T}$, such that:

$$p(R_{1:T}) = \prod_{i=1}^{T} exp(\alpha r_t)$$

Goal is to optimize marginal likelihood $p(R_{1:T})$ what is equivalent to maximization of sum of rewards.

SCG reduction, example N2

Reinforcement learning: MDP as a probabilistic model

Lower bound for new policy q_π and some initial policy p_{π_0}:

$$\mathcal{L}(q_\pi, p_{\pi_0}) = \mathbb{E}_{q_\pi(s_{1:T}, a_{1:T})}[\alpha \sum_{t=1}^{T} r_t] - \mathrm{KL}(q_\pi(s_{1:T}, a_{1:T}) || p_{\pi_0}(s_{1:T}, a_{1:T}))$$

Gradient of LB with uniform prior

R_t are sum of future rewards starting from step t.

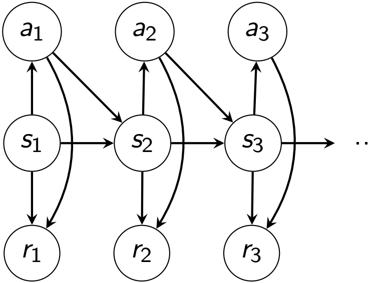
$$\nabla_\pi \mathcal{L}(q_\pi, p_{\pi_0}) = \mathbb{E}_{q_\pi(s_{1:T}, a_{1:T})}\left[\alpha \sum_{t=1}^{T} R_t \nabla_\pi \log \pi(a_t|s_t) + \nabla_\pi \mathcal{H}(q)\right]$$

This is just REINFORCE rule with new entropy term.

SCG reduction, example N2

Just derived (almost) REINFORCE rule using LV model.

Reinforcement learning: SCG reduction



Simplified reasoning: view MDP as SCG with r_t being cost nodes.

SCG reduction, example N2

Reinforcement learning: SCG reduction

We obtain formula for expected cost by definition of SCG:

$$J(\theta) = \mathbb{E}_{s_{1:T}, a_{1:T} \sim \pi_\theta}[\sum_{t=1}^{T} r_t]$$

Gradient w.r.t. policy parameters θ is just an application of general theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_{1:T}, a_{1:T} \sim \pi_\theta}[\sum_{t=1}^{T} R_t \nabla_\theta \log \pi_\theta(a_t | s_t)]$$

Derivation is virtually non-existent, we just applied the theorem.
The only difference is regularization entropy term.

SCG reduction: conclusions p.1

MLE

By doing two 'intuitive' choices:

- Cost $c = \log(x)$
- Output of SCG is distribution over target labels

We obtained same equations as if we were reasoning in terms of LV models and deriving LB for marginal distribution.

RL

We derived formula for the gradient virtually 'for free', in contrast with more complicated reasoning using LV models. Difference is KL term.

SCG reduction: conclusions p.2

Thinking in terms of SCG. For discussion:

Advantage

Easier to derive gradient estimator for complex models.

Disadvantage

Lack of interpretation.

Probabilistic vs SCG reasoning

Data $\rightarrow$ LV model $\rightarrow$ ELBO $\rightarrow$ SCG $\rightarrow$ optimization
Data $\rightarrow$ SCG $\rightarrow$ optimization

Subjective observation: SCG – LV = AE – RBM

Table of Contents

Overview

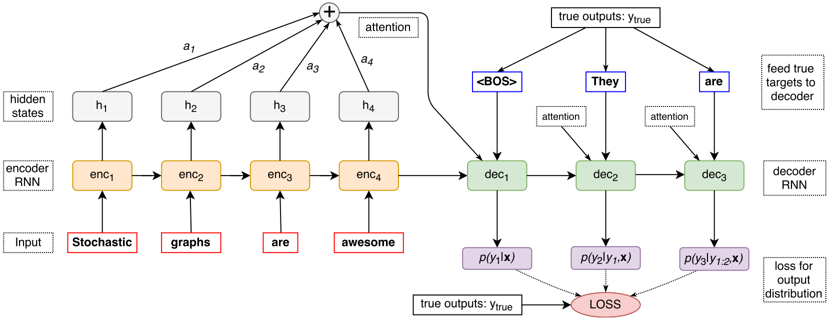
Once a plan gets too complex, everything can go wrong

So..

1. Look at the task
2. Find SCG there
3. Change SCG and (or) its optimization techniques

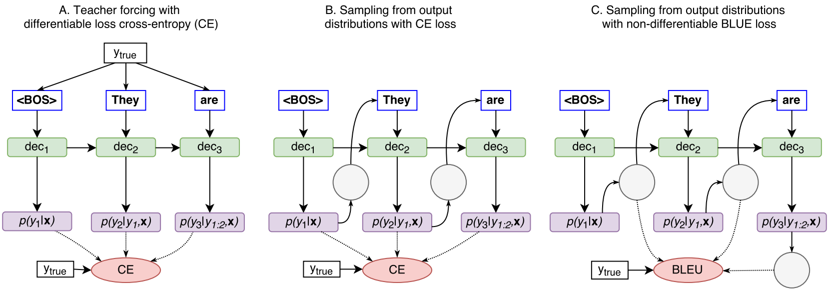
Optimization of seq2seq model

Using stochastic computation graphs formalism for optimization of sequence-to-sequence model.



Optimization of seq2seq model

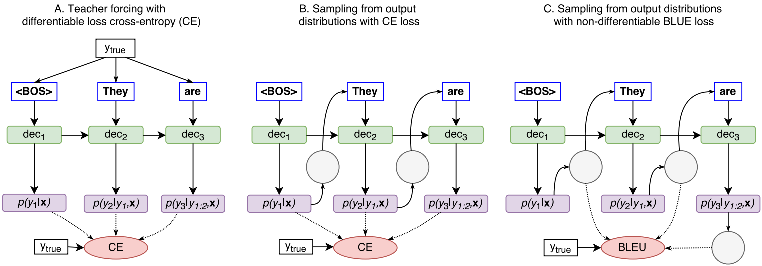
Variants of decoder: just different SCG graphs.



Optimization of seq2seq model

Table 1: Experimental results for MT. Either differentiable loss function Cross-Entropy (CE) was used or we directly optimized the target BLEU metric (last line).

	CE loss		BLEU	
	train	eval (best)	train	eval (best)
Teacher-forcing (Figure 2A, CE opt.)	5.96 ± 0.03	6.44 ± 0.08	31.8 ± 4.9	19.6 ± 3.6
Feed samples (Figure 2B, CE opt.):				
naive gradient	3.41 ± 0.17	4.20 ± 0.10	20.1 ± 2.7	9.8 ± 1.3
full gradient, control variates	3.45 ± 0.23	4.24 ± 0.08	12.4 ± 1.7	8.0 ± 0.7
full gradient, Gumbel reparam.	3.43 ± 0.16	4.22 ± 0.07	16.8 ± 1.5	8.7 ± 0.3
full gradient (Figure 2C, direct BLEU opt.)	-	-	24.6 ± 0.1	22.0 ± 0.2



Optimization of seq2seq model

Conclusions

1. Cross-entropy optimization
   - Feeding model with its own samples $\Rightarrow$ model learns to ignore decoder inputs regardless of the optimization technique used.
   - Teacher-forced model has lower entropy of output distributions in contrast to model learned with sampling.
2. Direct BLEU optimization
   - Model learns to pay attention to decoder inputs.
   - Once we addressed loss-evaluation mismatch, results are higher.
3. SCG approach
   - Easy to simulate different approaches (sampling, scheduled sampling, teacher forcing) within single framework.
   - There is no need to fit the task into RL framework in order to use REINFORCE rule.
   - PyTorch provides convenient tools for working with SCG.

Optimization of seq2seq model

What did we do? Just followed the plan:

1. Collected from the literature different optimization approaches for seq2seq model: teacher forcing, sampling.
2. Viewed them as different SCGs.
3. Played with optimization techniques.

Differentiable lower bound for BLEU score

BLEU: counting n-grams

Given distributions:
Output of model p_x of size $[seq\_len \times vocab\_size]$
Item from training data p_y of size $[seq\_len \times vocab\_size]$.

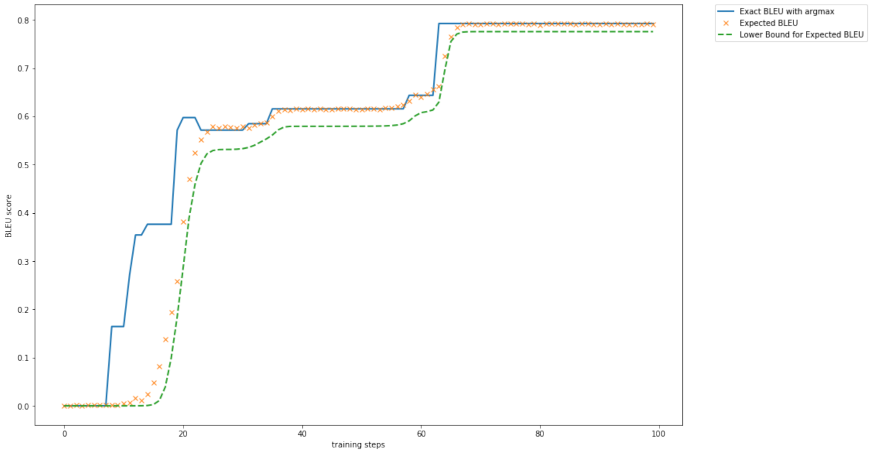
For unigrams:

$$\mathrm{BLEU}_1 = \mathbb{E}_{p_x,p_y}[\mathrm{BLEU}_1(x,y)] \sim \mathbb{E}_{p_x,p_y}[\sum_i O_1^i\Big)]$$

$$\mathbb{E}_{p_x,p_y}[O_1^i] \geq \sum_n p_x^{in} \cdot \min(1, \frac{\sum_j y_{jn}}{1+\sum_{k\neq i} p_x^{kn}}) = f(p_x, p_y)$$

Differentiable approximation of expected BLEU score.

Differentiable lower bound for BLEU score

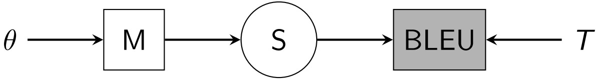


Differentiable lower bound for BLEU score

What did we do?

Before

BLEU is non-differentiable $\Rightarrow$ optimize expectation using samples.



After

Calculate expectation analytically using parameters of distribution $\Rightarrow$ no need in sampling.

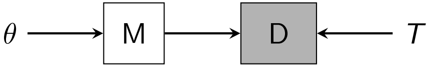


Table of Contents

Future directions

Better optimization of existing SCG

1. Generalization of TRPO and other RL techniques
2. Efficient control variates

Building equivalent SCG which is easier to optimize

1. Reparameterization trick (and ultimate reparameterization trick)
2. Replace stochastic nodes with deterministic ones (learn network to calculate expectation using parameters of distribution)

Goal: instrument for optimization of very complex SCG.
Then use it in challenging task where discrete actions are essential.
E.g. consider calculating gradient of SCG as different SCG and optimize it.