My First Thesis

Smart Things as Said by Me

Daan van Stigt
Institute for Logic, Language and Computation (ILLC)

Contents

My First Thesis

ABSTRACT

This is the best thesis. I have all the best words. Also, do check out my code. You will be blown away. Such wow. Much amazings.

1

Introduction

I investigate the question: What are effective ways of incorporating syntactic structure into neural language models?

Central in this research is a class of neural language models that explicitly model the hierarchical syntactic structure in addition to the sequence of words **Buys+2015:neural-gen-dep**; **Buys+2018**; Dyer *et al.*, 2016. These models merge algorithms from transition based parsing ('shift-reduce parsing') adapted for joint (generative) modeling, with (recurrent) neural networks that parametrize the transition model.

The syntactic structure that decorates the words can be latent, and marginalized over, or can be given explicitly, for example as the prediction of an external parser. Although these are fundamentally joint model, they can be evaluated as regular language models (modeling only words) by (approximate) marginalization of the syntactic structure. In the case of the RNNG (Dyer *et al.*, 2016), exact marginalization is intractable due to the parametrization of the statistical model, but importance sampling provides an effective approximate method. An externally trained discriminative parser is used to obtain proposal samples. Other models provide exact marginalization, but this typically comes at the cost of a less expressive parametrization, for example one in which the features cannot be structure-dependent (**Buys+2018**).

I study the RNNG and investigate:

The impact of the proposal samples on the approximate marginalization I propose a new discriminative chart-based neural parser that is trained with a global, Conditional Random Field (CRF), objective. The parser is an adaptation of the minimal neural parser proposed in (**Stern+2017:Minimal**) which is trained with a margin-based objective. This contrast with the typical choice for a transition-based parser as proposal (a discrminatively trained RNNG). The rationale in this research is that we posit that a globally trained model is a better proposal distribution than a locally trained transition based model. A global model has ready access to competing analyses that can be structurally dissimilar but close in probability, whereas we hypothesize that a locally trained model is prone to produce locally corrupted structures that are nearby in transition-space. To promote more diverse samples, the transition distributions are flattened, causing as a downside for the model to visit. This is a general challenge for greedy transition based models that is typically answered to train dynamic oracles (**Goldberg+2013:dynamic**) (also called 'exploration' ((**Ballesteros+2016:exploration**; **Stern+2017:Minimal**)), instances of imitation learning (**Vlachos**); Eisner et al. 2012)), a direction which we do not consider in this research.

Semi-supervised training by including unlabeled data A major drawback of these syntactic language models is that they require annotated data to be trained, and preciously little of such data exists. To make these joint models competitive language models they need to make use of the vast amounts of unlabeled data that exists. We extend the training to the unsupervised domain by optimizing a variational lower bound on the marginal probabilities that jointly optimizes the parameters of proposal model ('posterior' in this framework) with the joint model. We obtain gradients for this objective using the score function estimator (Fu 2006), also known as REINFORCE (Williams 1992), which is widely used in the field of deep reinforcement learning, and we introduce an effective baseline based on argmax decoding (Rennie et al. 2017), which significantly reduces the variance in this optimization procedure. Our CRF parser particularly excels in the role of posterior thanks the independence assumptions that allow for efficient exact computation of key quantities: the entropy term in the lower bound can be computed exactly using Inside-Outside algorithm, removing one source of variance from the gradient estimation, and the argmax decoding can be performed exactly thanks to Viterbi, making the argmax baseline even more effective.

Alternative, simpler, models Multitask learning of a neural language model with a syntactic side objective is a competitive and robust alternative method to infuse neural language models with syntactic knowledge. Training the syntactic model on data that mixes gold trees with predicted 'silver' trees for unlabeled data is a competitive and robust alternative to fully principled semi-supervised learning. We consider these alternatives in order to quantify significance of the latent structure, and the semisupervised training on the other hand, as measured by some external performance metric. We propose a simple multitask neural language model that predicts labeled spans from the RNN hidden states, using a feature function identical identical to that used in the CRF parser. A similar strategy has recently proposed in work on semantic parsing and is called a 'syntactic scaffold' (Swayamdipta et al. 2018).

Targeted syntactic evaluation TBA

2

Background

../src/bibliography.bib

2.1 Syntax

- Some generic stuff on syntax and constituency in natural language
- Reference **Carnie2010:constituent**; **Everaert+2015:structures** or something?

2.2 Parsing

- treebanks
- CFGs, and CNF form (show graph of binarizing CFG)
- parsing methods: global chart based, local transition based
- dynamic programming inference vc search heuristics
- modelling types: generative, discriminative, log-linear, count-based, feature-based, neural network features

2.3 Neural networks

Introduce all the neural network business used

- Feedfoward
- RNN and LSTM etc.
- SGD optimization

2.4 Language models

- n-grams, (recurrent) neural network language models
-

2.5 Multitask learning

- Give formal description of multitask learning
- Name some examples of multitask learning **Zhang+2016:multitask**; **Goldberg+2016:multitask**; **Swayamdipta+2018:scaffold**
- In our case of language modelling with syntactic side-objective: $\mathcal{L}(x, y) = \log p_\theta(x) + \log q_\lambda(y|x)$

3

RNNG

4

CRF parser

../src/bibliography.bib

In this chapter I introduce a novel chart-based neural parser that is trained with a global, Conditional Random Field (CRF). This parser provides an alternative proposal model for the apprimate marginalization in the generative RNNG.

The parser is discriminative, chart-based neural parser that is trained with a global, Conditional Random Field (CRF), objective. The parser is an adaptation of the chart-based parser proposed in **Stern+2017:Minimal** which is trained with a margin-based objective.

This contrast with the typical choice for a transition-based parser as proposal (a discrminatively trained RNNG). The rationale in this research is that we posit that a globally trained model is a better proposal distribution than a locally trained transition based model.

A global model has ready access to competing analyses that can be structurally dissimilar but close in probability, whereas we hypothesize that a locally trained model is prone to produce locally corrupted structures that are nearby in transition-space.

To promote more diverse samples, the transition distributions are flattened, causing as a downside for the model to visit .

This is a general challenge for greedy transition based models that is typically answered to train dynamic oracles (**Goldberg+2013:dynamic**) (also called 'exploration' ((**Ballesteros+2016:exploration**) (**Stern+2017:Mini** instances of imitation learning (Vlachos 2012; Eisner et al. 2012)), a direction which we do not consider in this research.

4.1 Model

In this section I describe the probabilistic model of the parser.

1. Introduce probabilistic model, reference CRF background in apendix
2. Desribe how this is a generalization from (**Stern+2017:Minimal**)

,

4.1.1 Motivation

- Key point to make: this model regards a constituency tree as a collection of *labeled spans* over a sentence.
- Earlier models (log-linear and neural) regard a constituency tree as a collection of *context-free grammar rules* over a sentence (**Klein+2003**).
- A model over *rules* needs to put more expressiveness in the output space: lexicalize the rules **Collins2003** break the grammar's independence assumptions by annotating the rule with parent and sibling labels (**Klein+2003**);)
- This rich feature representations to parameterize local potentials that interact with a comparatively unconstrained structured decoder.

4.1.2 Features

In this section I describe how the scores are computed with bidirectional LSTM features.

- Mention: LSTM features
- Give formal expression for LSTM minus features + Feedforward scoring function

4.2 Inference

In this section we describe the solution to three related problems:

- Find the best parse $\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$
- Compute the normalizer $Z(\mathbf{x}) = \sum_{\curvearrowright \in \mathcal{Y}(\mathbf{x})} \Psi(x, y)$
- Compute the entropy conditioned on $\mathbf{x}$, $H(\mathbf{y}|\mathbf{x})$

All three problems can be solved with a different instance of the same two algorithm: the inside algorithm and the outside algorithm.

4.2.1 Inside recursion

In this derivation we follow Michael Collins notes on the Inside-Outside Algorithm.[1]

Let a sentence be $x_1, \dots, x_n$, where each x_i is a word. We are given a CFG (N, Σ, R, S) in Chomsky normal form. Let ψ be a function that maps any rule production $r \in R$ of the form $\langle A \to B\ C, i, k, j\rangle$ or $\langle A, i, i+1\rangle$ to a value $\psi(r) \geq 0$. Let a tree T be a set of such rules r with the only constraint that these rules make up a tree.

Following the minimal span parser we define ψ as

$$\log \psi(A \to B\ C, i, k, j) \triangleq s_{label}(i, j, A) + s_{span}(i, j) \tag{4.1}$$

$$\text{and} \tag{4.2}$$

$$\log \psi(A, i, i+1) \triangleq s_{label}(i, i+1, A) + s_{span}(i, i+1), \tag{4.3}$$

[1] http://www.cs.columbia.edu/~mcollins/io.pdf

and thus the potential of a tree as

$$\log \Psi(T) = \sum_{r \in T} \log \psi(r) \tag{4.4}$$

$$= \sum_{\langle A,i,j\rangle \in T} s_{label}(i,j,A) + s_{span}(i,j), \tag{4.5}$$

$$\tag{4.6}$$

Note that the potential function as defined in 4.1 disregards most of the information in a binary rule. In particular we see that B, C and k, the labels and split-point of the children, are discarded.

Now note that equation corresponds exactly to the second formula in section 3 of the minimal span-based parser paper

$$s_{tree}(T) = \sum_{(\ell(i,j)) \in T} [s_{label}(i,j,\ell) + s_{span}(i,j)]. \tag{4.7}$$

which is how I derived that 4.1 is the correct formula for the rule score.

We obtain our CRF objective when we normalize this score globally

$$P(T) = \frac{\prod_{r \in T} \psi(r)}{\sum_{T' \in \mathcal{T}} \prod_{r' \in T'} \psi(r')} \tag{4.8}$$

$$\tag{4.9}$$

or equivalently

$$\log P(T) = \sum_{r \in T} \log \psi(r) - \log \sum_{T \in \mathcal{T}} \prod_{r \in T} \psi(r) \tag{4.10}$$

$$\tag{4.11}$$

From the aforementioned notes we get the following general result for the inside value α. For all $A \in N$, for all $0 \le i < n$

$$\alpha(A,i,i+1) = \psi(A,i,i+1) \tag{4.12}$$

and for all (i,j) such that $1 \le i < j \le n$:

$$\alpha(A,i,j) = \sum_{A \to BC} \sum_{k=i+1}^{j-1} \psi(A \to B\ C, i,k,j) \cdot \alpha(B,i,k) \cdot \alpha(C,k,j) \tag{4.13}$$

Note that we are considering a CFG in which the rule set is complete, i.e.

$$\langle A \rightarrow B\ C \rangle \in R \text{ for each } (A, B, C) \in N^3, \tag{4.14}$$

and recall that the labels B and C do not appear in the scoring functions in 4.1. These facts will allow us to simplify the expression in formula 4.13 as

$$\begin{aligned}
\alpha(A,i,j) &= \sum_{B\in N}\sum_{C\in N}\sum_{k=i+1}^{j-1} \tilde{s}_{label}(i,j,A)\cdot\tilde{s}_{span}(i,j)\alpha(B,i,k)\cdot\alpha(C,k,j) && \text{(4.15a)}\\
&= \tilde{s}_{label}(i,j,A)\cdot\tilde{s}_{span}(i,j)\sum_{k=i+1}^{j-1}\sum_{B\in N}\alpha(B,i,k)\cdot\sum_{C\in N}\alpha(C,k,j) && \text{(4.15b)}\\
&= \tilde{s}_{label}(i,j,A)\cdot\tilde{s}_{span}(i,j)\sum_{k=i+1}^{j-1} S(i,k)\cdot S(k,j) && \text{(4.15c)}
\end{aligned}$$

where we've introduced a number of notational abbreviations

$$\begin{aligned}
\tilde{s}_{label}(i,j,A) &= \exp(s_{label}(i,j,A)) && \text{(4.16)}\\
\tilde{s}_{span}(i,j) &= \exp(s_{span}(i,j)) && \text{(4.17)}\\
S(i,j) &= \sum_{A\in N}\alpha(A,i,j) && \text{(4.18)}
\end{aligned}$$

Note that this is the exact same formula as **??**.

From equation 4.15c we can deduce that we in fact do even need to store the values $\alpha(i,j,A)$ but that it suffices to only store the marginalized values $S(i,j)$. In this case, the recursion simplifies even further:

$$\begin{aligned}
S(i,j) &= \sum_{A\in N}\alpha(A,i,j) && \text{(4.19a)}\\
&= \sum_{A\in N}\tilde{s}_{label}(i,j,A)\cdot\tilde{s}_{span}(i,j)\sum_{k=i+1}^{j-1} S(i,k)\cdot S(k,j) && \text{(4.19b)}\\
&= \left[\sum_{A\in N}\tilde{s}_{label}(i,j,A)\cdot\tilde{s}_{span}(i,j)\right]\left[\sum_{k=i+1}^{j-1} S(i,k)\cdot S(k,j)\right] && \text{(4.19c)}
\end{aligned}$$

where we put explicit brackets to emphasize that independence of the subproblems of labeling and splitting. We can now recognize this as the 'inside' equivalent of the expression from the paper[2]

$$s_{best}(i,j) = \max_{\ell}[s_{label}(i,j,\ell)] + \max_{k}[s_{split}(i,k,j)]. \tag{4.20}$$

The recursions are the same; the semirings are different. The viterbi recursion given above is in the ViterbiSemiring, which uses the max operator as $\oplus$; the inside recursion given in 4.15c has standard addition $(+)$ instead.

4.2.2 Outside recursion

$$\begin{aligned}
\beta(A,i,j) = &\sum_{B\to CA\in R}\sum_{k=1}^{i-1}\psi(B\to CA,k,i-1,j)\cdot\alpha(C,k,i-1)\cdot\beta(B,k,j)\\
&+\sum_{B\to AC\in R}\sum_{k=j+1}^{n}\psi(B\to A,C,i,j,k)\cdot\alpha(C,j+1,k)\cdot\beta(B,i,k)\\
=&\sum_{B\in N}\sum_{C\in N}\sum_{k=1}^{i-1}\psi(B,k,j)\cdot\alpha(C,k,i-1)\cdot\beta(B,k,j)\\
&+\sum_{B\in N}\sum_{C\in N}\sum_{k=j+1}^{n}\psi(B,i,k)\cdot\alpha(C,j+1,k)\cdot\beta(B,i,k)\\
=&\sum_{k=1}^{i-1}\left[\sum_{B\in N}\psi(B,k,j)\cdot\beta(B,k,j)\right]\cdot\left[\sum_{C\in N}\alpha(C,k,i-1)\right]\\
&+\sum_{k=j+1}^{n}\left[\sum_{B\in N}\psi(B,i,k)\cdot\beta(B,i,k)\right]\cdot\left[\sum_{C\in N}\alpha(C,j+1,k)\right]\\
=&\sum_{k=1}^{i-1}S'(k,j)\cdot S(k,i-1)+\sum_{k=j+1}^{n}S'(i,k)\cdot S(j+1,k)
\end{aligned}$$

[2] I believe there is actually an error in this equation: it should read $s_{label}(i,j,\ell) + s_{span}(i,j)$ instead of just $s_{label}(i,j,\ell)$. This is implied by the score for a single node, which is given by equation 4.1, taken directly from the paper.

where

$$S(i,j) = \sum_{A \in N} \alpha(A,i,j)$$
$$S'(i,j) = \sum_{A \in N} \psi(A,i,j)\beta(A,i,j)$$

5

Semisupervised learning of RNNG

Throughout these note we write x for a sentence, y for a (latent) constituency tree, and $\mathcal{Y}(x)$ for the *yield* of x, that is, all trees that can be assigned to x. Furthermore, let $\mathbb{L}$ be a set of pairs (x, y) of sentences with gold trees, and let $\mathbb{U}$ be a set of unlabeled sentences x.

We define the following semi-supervised objective

$$\mathcal{J} \triangleq \mathcal{J}_{\mathcal{S}} + \alpha \mathcal{J}_{\mathcal{U}}, \tag{5.1}$$

were $\mathcal{J}_{\mathcal{S}}$ is the supervised objective optimized over $\mathbb{L}$ and $\mathcal{J}_{\mathcal{U}}$ the unsupervised objective optimized over $\mathbb{U}$. We introduce $\alpha \in \mathbb{R}$ as an arbitrary scalar controlling the contribution of the unsupervised objective.

5.1 Supervised objective

Let $p_\theta(x, y)$ be parametrized by a Generative RNNG (Dyer *et al.*, 2016). Then our supervised objective is

$$\mathcal{J}_{\mathcal{S}} \triangleq \sum_{(x,y) \in \mathbb{L}} \log p_\theta(x, y) \tag{5.2}$$

$$\tag{5.3}$$

This objective is optimized as usual using stochastic gradient estimates:

$$\nabla_\theta \mathcal{J}_\mathcal{S} \approx \frac{|\mathbb{L}|}{n} \sum_{i=1}^{n} \nabla_\theta \log p_\theta(x^{(i)}, y^{(i)}), \tag{5.4}$$

$$\tag{5.5}$$

where $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$ is a mini-batch sampled uniformly from $\mathbb{L}$. To compute $\nabla_\theta \log p_\theta(x^{(i)}, y^{(i)})$ we rely on automatic differentiation (**Baydin+2015:AD**).

5.2 Unsupervised objective

Consider the following objective to be maximized:

$$\mathcal{J}_\mathcal{U} \triangleq \sum_{x \in \mathbb{U}} \log p(x) \tag{5.6a}$$

$$= \sum_{x \in \mathbb{U}} \log \sum_{y \in \mathcal{Y}(x)} p_\theta(x, y) \tag{5.6b}$$

This is a language modelling objective, in which we treat y as latent, and $p_\theta(x, y)$ is a generative RNNG. A consequence the independence assumptions of the RNNG (or better: lack thereof) is that the sum over trees y is no longer tractable. To optimize this objective we must fall back on approximate methods.

Variational approximation We optimize the objective using variational inference (Blei *et al.*, 2016). We introduce a posterior $q_\lambda(y|x)$ parametrised by λ and use Jensen's inequality to derive a variational lower bound:

$$\log p(x) = \log \sum_{y \in \mathcal{Y}(x)} q_\lambda(y|x) \frac{p_\theta(x, y)}{q_\lambda(y|x)} \tag{5.7a}$$

$$= \log \mathbb{E}_{q_\lambda} \left[\frac{p_\theta(x, y)}{q_\lambda(y|x)} \right] \tag{5.7b}$$

$$\geq \mathbb{E}_{q_\lambda} \left[\log \frac{p_\theta(x, y)}{q_\lambda(y|x)} \right] \tag{5.7c}$$

$$= \mathbb{E}_{q_\lambda} \big[\log p_\theta(x, y) - \log q_\lambda(y|x) \big] \triangleq \mathcal{L}(\theta, \lambda) \tag{5.7d}$$

The only requirement for q_λ is that

$$p(x, y) > 0 \Rightarrow q(y|x) > 0 \qquad \text{for all } x \text{ and } y \in \mathcal{Y}(x). \tag{5.8}$$

Any discriminatively trained parser fulfills this requirement.

The lower bound $\mathcal{L}(\theta, \lambda)$ will be optimized by gradient optimization, which means we will need to take the gradients $\nabla_\theta \mathcal{L}(\theta, \lambda)$ and $\nabla_\lambda \mathcal{L}(\theta, \lambda)$.

Gradients of joint parameters The first gradient is easy and permits a straightforward Monte-Carlo estimate:

$$\begin{aligned}
\nabla_\theta \mathcal{L}(\theta, \lambda) &= \nabla_\theta \mathbb{E}_{q_\lambda}\big[\log p_\theta(x, y) - \log q_\lambda(y|x)\big] && (5.9a)\\
&= \mathbb{E}_{q_\lambda}\big[\nabla_\theta \log p_\theta(x, y) - \nabla_\theta \log q_\lambda(y|x)\big] && (5.9b)\\
&= \mathbb{E}_{q_\lambda}\big[\nabla_\theta \log p_\theta(x, y)\big] && (5.9c)\\
&\approx \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \log p_\theta(x, y^{(i)}) && (5.9d)
\end{aligned}$$

where $y_i \sim q_\lambda(y|x)$ for $i = 1, \ldots, n$. We can move the gradient inside the expectation (second equality) because q does not depend on θ.

Gradients of posterior parameters The second gradient is harder and requires us to rewrite the objective:

$$
\begin{aligned}
\nabla_\lambda \mathcal{L}(\theta, \lambda) &= \nabla_\lambda \mathbb{E}_{q_\lambda}\left[\log p_\theta(x,y) - \log q_\lambda(y|x)\right] && (5.10a)\\
&= \nabla_\lambda \sum_y \{q_\lambda(y|x)\log p_\theta(x,y) - q_\lambda(y|x)\log q_\lambda(y|x)\} && (5.10b)\\
&= \sum_y \{\nabla_\lambda q_\lambda(y|x)\log p_\theta(x,y) && (5.10c)\\
&\qquad - \nabla_\lambda q_\lambda(y|x)\log q_\lambda(y|x) && (5.10d)\\
&\qquad - q_\lambda(y|x)\nabla_\lambda \log q_\lambda(y|x)\} && (5.10e)\\
&= \sum_y \{\nabla_\lambda q_\lambda(y|x)\log p_\theta(x,y) - \nabla_\lambda q_\lambda(y|x)\log q_\lambda(y|x)\} && (5.10f)\\
&= \sum_y \{(\log p_\theta(x,y) - \log q_\lambda(y|x))\nabla_\lambda q_\lambda(y|x)\} && (5.10g)\\
&= \sum_y \{(\log p_\theta(x,y) - \log q_\lambda(y|x))q_\lambda(y|x)\nabla_\lambda \log q_\lambda(y|x)\} && (5.10h)\\
&= \mathbb{E}_{q_\lambda}\left[(\log p_\theta(x,y) - \log q_\lambda(y|x))\nabla_\lambda \log q_\lambda(y|x)\right\} && (5.10i)\\
&= \mathbb{E}_{q_\lambda}\left[l(x,y)\nabla_\lambda \log q_\lambda(y|x)\right] && (5.10j)
\end{aligned}
$$

Where we've defined a learning signal $l(x,y) \triangleq \log p_\theta(x,y) - \log q_\lambda(y|x)$.

In this derivation we used the identity

$$\nabla_\lambda \log q_\lambda(y|x) = \frac{\nabla_\lambda q_\lambda(y|x)}{q_\lambda(y|x)} \tag{5.11}$$

or equivalently

$$\nabla_\lambda q_\lambda(y|x) = q_\lambda(y|x)\nabla_\lambda \log q_\lambda(y|x) \tag{5.12}$$

and the fact that

$$\begin{aligned}\sum_y q_\lambda(y|x)\nabla_\lambda \log q_\lambda(y|x) &= \sum_y q_\lambda(y|x)\frac{\nabla_\lambda q_\lambda(y|x)}{q_\lambda(y|x)} && (5.13a)\\ &= \sum_y \nabla_\lambda q_\lambda(y|x) && (5.13b)\\ &= \nabla_\lambda \sum_y q_\lambda(y|x) && (5.13c)\\ &= \nabla_\lambda 1 && (5.13d)\\ &= 0. && (5.13e)\\ & && (5.13f)\end{aligned}$$

This rewritten objective permits a straightforward MC estimate:

$$\mathbb{E}_{q_\lambda}\left[l(x,y)\nabla_\lambda \log q_\lambda(y|x)\right] \approx \frac{1}{n}\sum_{i=1}^{n} l(x,y^{(i)})\nabla_\lambda \log q_\lambda(x|y^{(i)}) \quad (5.14)$$

where $y_i \sim q_\lambda(y|x)$ for $i = 1, \ldots, n$.

This estimator has been derived (in slightly different forms) in among others (Williams, 1992), (Paisley *et al.*, 2012), (Mnih and Gregor, 2014), (Ranganath *et al.*, 2014), and (**Miao+16:LLVAE**) and is known as the REINFORCE estimator (Williams, 1992), or score function estimator (after the score function $\nabla_\theta \log p_\theta(x)$) (Fu, 2006).

5.3 Optimization

We use automatic differentiation (**Baydin+2015:AD**) to obtain all our gradient estimates.

To get the gradients in formula 5.14 we rewrite it in the form of a surrogate objective (Schulman *et al.*, 2015)

$$L(\lambda) = \frac{1}{n}\sum_{i=1}^{n} \log q_\lambda(x|y^{(i)})\textsc{blockgrad}(l(x,y^{(i)})) \quad (5.15)$$

where BLOCKGRAD is function that ‘detaches’ a parametrized function from the computation graph effectively turning it into a scalar. That is, loosely speaking

$$\textsc{blockgrad}(f_\theta(x)) = f(x) \quad (5.16)$$

such that

$$\nabla_\theta \text{BLOCKGRAD}(f_\theta(x)) = \nabla_\theta f(x) \tag{5.17}$$
$$= 0 \tag{5.18}$$

Then differentiation of L gives us the unbiased estimator

$$\nabla_\lambda L(\lambda) = \frac{1}{n} \sum_{i=1}^{n} l(x, y^{(i)}) \nabla_\lambda \log q_\lambda(x|y^{(i)}) \tag{5.19}$$

5.4 Variance reduction

We have derived an estimator for the gradient of the posterior parameters in the unsupervised objective. This estimator is unbiased, but is known to have high variance, often too much to be useful (Paisley *et al.*, 2012). Two effective methods to counter this are control variates and baselines (Ross, 2006).

Variance of estimator First, let's analyze the variance of our estimator. Note that our expectation is of the general form

$$\mu \triangleq \mathbb{E}[f(X)] \tag{5.20}$$

and that we estimate this quantity by generating n independent samples $X_1, \ldots, X_n \sim P(X)$ and computing

$$\hat{\mu} \triangleq \frac{1}{n} \sum_{i=1}^{n} f(X_i). \tag{5.21}$$

This is an unbiased estimator for μ with error

$$\text{MSE} = \mathbb{E}[(\mu - \hat{\mu})^2] = \text{Var}[\hat{\mu}] = \frac{\text{Var}[\hat{\mu}]}{n}, \tag{5.22}$$

which means that the error is of the order

$$\mu - \hat{\mu} \sim \sqrt{\frac{\text{Var}[\hat{\mu}]}{n}} \tag{5.23}$$

and reducing it linearly requires a quadratic number of samples.

In our particular case, the function f is

$$f_{X=x}(Y) \triangleq l(X, Y) \nabla_\lambda \log q_\lambda(Y|X = x) \tag{5.24}$$

where we have made explicit that y is the random variable, and x is given.

Control variates Consider a function $\phi(X)$ with known expectation

$$\mu_\phi \triangleq \mathbb{E}[\phi(X)] \tag{5.25}$$

Then we can define a new function $\hat{f}$ such that

$$\hat{f}(X) \triangleq f(X) - \phi(X) + \mu_\phi. \tag{5.26}$$

This function is also an estimator for μ, since

$$\begin{aligned}\mathbb{E}[\hat{f}(X)] &= \mathbb{E}[f(X)] - \mu_\phi + \mu_\phi && (5.27)\\ &= \mathbb{E}[f(X)], && (5.28)\end{aligned}$$

and a computation shows that the variance of the new function is

$$\begin{aligned}\mathrm{Var}[\hat{f}(X)] &= \mathbb{E}[(f(X) - \phi(X) + \mu_\phi) - \mu)^2]\\ &= \mathbb{E}[(f(X) - \phi(X) + \mu_\phi)^2] - 2\mathbb{E}[(f(X) - \phi(X) + \mu_\phi)\mu] + \mathbb{E}[\mu^2]\\ &= \mathbb{E}[(f(X) - \phi(X) + \mu_\phi)^2] - 2\mathbb{E}[(f(X) - \phi(X) + \mu_\phi)]\mu + \mu^2\\ &= \mathbb{E}[(f(X) - \phi(X) + \mu_\phi)^2] - 2\mu^2 + \mu^2\\ &= \mathbb{E}[f(X)^2 + \phi(X)^2 + \mu_\phi^2 - 2f(X)\phi(X) + 2f(X)\mu_\phi - 2\phi(X)\mu_\phi] - \mu^2\\ &= \mathbb{E}[f(X)^2] - \mathbb{E}[f(X)]^2\\ &\quad - 2(\mathbb{E}[f(X)\phi(X)] - \mathbb{E}[f(X)]\mathbb{E}[\phi(X)])\\ &\quad + \mathbb{E}[\phi(X)^2] - \mathbb{E}[\phi(X)]^2\\ &= \mathrm{Var}[f(X)] - 2\ \mathrm{Cov}[f(X), \phi(X)] + \mathrm{Var}[\phi(X)]\end{aligned}$$

This means we can get a reduction in variance whenever

$$\mathrm{Cov}[f(X), \phi(X)] > \frac{1}{2}\ \mathrm{Var}[\phi(X)]. \tag{5.29}$$

The function ϕ is called a *control variate*—it allows us to control the variance of f.

From the equality above we can see that this will be the case whenever $f(X)$ and $\phi(X)$ are strongly correlated. Our choice of control variate will be made with the that in mind. Furthermore, $\mathbb{E}[\phi(X)]$ must be known. What is an optimal control variate? Typically a control variate of the form $a\phi$ is chosen with fixed, and a is optimized to maximize the correlation. This brings us to the generic formulation of a control variate:

$$\hat{f}(X) \triangleq f(X) - a(\phi(X) - \mathbb{E}[\phi(X)]) \tag{5.30}$$

with variance

$$\mathrm{Var}[\hat{f}(X)] = \mathrm{Var}[f(X)] - 2a\ \mathrm{Cov}[f(X), \phi(X)] + a^2\ \mathrm{Var}[\phi(X)] \tag{5.31}$$
$$\tag{5.32}$$

We take a derivative of this with respect to a

$$\frac{\partial}{\partial a}\mathrm{Var}[\hat{f}(X)] = -2\ \mathrm{Cov}[f(X), \phi(X)] + 2a\ \mathrm{Var}[\phi(X)] \tag{5.33}$$

Setting this to zero and solving for a we obtain the optimal choice for a

$$a = \frac{\mathrm{Cov}[f(X), \phi(X)]}{\mathrm{Var}[\phi(X)]}. \tag{5.34}$$

Plugging in this solution into the expression for $\mathrm{Var}[\hat{f}(X)]$ and dividing by $\mathrm{Var}[f(X)]$ we get

$$\frac{\mathrm{Var}[\hat{f}(X)]}{\mathrm{Var}[f(X)]} = 1 - \frac{\mathrm{Cov}[f(X), \phi(X)]}{\mathrm{Var}[f(X)]\ \mathrm{Var}[\phi(X)]} \tag{5.35}$$
$$= 1 - \mathrm{corr}^2[f(X), \phi(X)], \tag{5.36}$$

which shows that given this choice of a the reduction in variance is directly determined by the correlation between $f(X)$ and $\phi(X)$.

Bringing this all together, we let our new estimator be

$$\mathbb{E}[f(X)] = \mathbb{E}[\hat{f}(X)] \approx \frac{1}{n}\sum_{i=1}^{n}[f(X_i) - a\phi(X_i)] - \mu_\phi \tag{5.37}$$

Example (Ross, 2006) Suppose we want to use simulation to determine

$$\mathbb{E}[f(X)] = \mathbb{E}[e^X] = \int_0^1 e^x dx = e - 1 \tag{5.38}$$

with $X \sim \mathcal{U}(0,1)$. A natural control variate to use in this case is the random variable X itself: $\phi(X) \triangleq X$. We thus define the new estimator

$$\hat{f}(X) = f(X) - \phi(X) + \mathbb{E}[\phi(X)] \tag{5.39}$$
$$= e^X - X + \frac{1}{2}. \tag{5.40}$$

To compute the decrease in variance with this new estimator, we first note that

$$\begin{aligned}
\operatorname{Cov}(e^X, X) &= \mathbb{E}[Xe^X] - \mathbb{E}[X]\mathbb{E}[e^X] \\
&= \int_0^1 xe^x dx - \frac{e-1}{2} \\
&= 1 - \frac{e-1}{2} \approx 0.14086 \\
\operatorname{Var}[e^X] &= \mathbb{E}[e^{2X}] - (\mathbb{E}[e^X])^2 \\
&= \int_0^1 e^{2x} dx - (1-e^x)^2 \\
&= \frac{e^2-1}{2} - (1-e^x)^2 \approx 0.2420 \\
\operatorname{Var}[X] &= \mathbb{E}[X^2] - (\mathbb{E}[X])^2 \\
&= \int_0^1 x^2 dx - \frac{1}{4} \\
&= \frac{1}{3} - \frac{1}{4} = \frac{1}{12}.
\end{aligned}$$

When we choose a as in formula 5.34 we can use formula 5.35 to compute that

$$\begin{aligned}
\frac{\operatorname{Var}[\hat{f}(X)]}{\operatorname{Var}[f(X)]} &= 1 - \frac{(0.14086)^2}{\frac{0.2420}{12}} && (5.41) \\
&\approx 0.0161. && (5.42)
\end{aligned}$$

This is a reduction of 98.4 percent! A simulation illustrates what this looks like in practice with . . . samples:

6

Multitask learning

7

Targeted syntactic evaluation

8

Conclusion

Acknowledgements

The author is grateful to the author, for he is his own eternal flame of inspiration.

Appendices

A

Figures

References

Blei, D. M., A. Kucukelbir, and J. D. McAuliffe (2016). “Variational Inference: A Review for Statisticians”. *ArXiv e-prints*. Jan.

Dyer, C., A. Kuncoro, M. Ballesteros, and N. A. Smith (2016). “Recurrent Neural Network Grammars”. *CoRR*. abs/1602.07776. arXiv: 1602.07776. URL: http://arxiv.org/abs/1602.07776.

Fu, M. C. (2006). “Gradient Estimation”. In: *Handbooks in Operations Research and Management Science (Volume 13)*. Ed. by B. L. N. Edited by Shane G. Henderson. Elsevier. Chap. 19. 575–616.

Mnih, A. and K. Gregor (2014). “Neural Variational Inference and Learning in Belief Networks”. In: *ICML*.

Paisley, J., D. Blei, and M. Jordan (2012). “Variational Bayesian Inference with Stochastic Search”. In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. Ed. by J. Langford and J. Pineau. *ICML ’12*. New York, NY, USA: Omnipress. 1367–1374.

Ranganath, R., S. Gerrish, and D. Blei (2014). “Black Box Variational Inference”. In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. Ed. by S. Kaski and J. Corander. Vol. 33. *Proceedings of Machine Learning Research*. Reykjavik, Iceland: PMLR. 814–822. URL: http://proceedings.mlr.press/v33/ranganath14.html.

Ross, S. M. (2006). *Simulation, Fourth Edition.* Orlando, FL, USA: Academic Press, Inc.

Schulman, J., N. Heess, T. Weber, and P. Abbeel (2015). “Gradient Estimation Using Stochastic Computation Graphs”. In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc. 3528–3536.

Williams, R. J. (1992). “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning”. *Mach. Learn.* 8(3-4): 229–256.