

DIFFERENTIABLE COMBINATORIAL LOSSES THROUGH GENERALIZED GRADIENTS OF LINEAR PROGRAMS

SUPPLEMENTARY MATERIAL

Anonymous authors

Paper under double-blind review

A DETAILS OF THE EXPERIMENTAL SETUP

A.1 DIFFERENTIATING OVER BIPARTITE MATCHING FOR WEAKLY-SUPERVISED LEARNING

In experiments involving weakly-supervised learning for image recognition, we used the ResNet18 architecture (Zagoruyko & Komodakis, 2016) involving residual, convolutional blocks. We used the CIFAR100 benchmark image dataset, with the standard training-testing split, as the source of images and labels. Training set batch size was set to 384 elements. Each batch of samples is divided into bags that specify an independent weighted bipartite matching problem; that is, for bag size $b = 32$, we are solving $384/b = 12$ weighted matching problems, one for each of the 32 bags. We trained all the networks using a cyclic learning rate with a maximum of 0.1 for 80 epochs, following by another cycle with maximum learning rate of 0.05 for 40 epochs.

The PyTorch code for the differentiable bipartite matching combinatorial layer is presented below. The layer accepts as input an $n \times b^2$ matrix, corresponding to a batch of n independent bipartite matching problems to solve, each of the size $b \times b$.

Listing 1: PyTorch Code for the Differentiable Bipartite Matching Layer

```

import numpy as np;
import torch
# assignment problem solver: https://github.com/cheind/py-lapsolver
from lapsolver import solve_dense

class bipartiteMatchingLayer(torch.autograd.Function):

    @staticmethod
    def forward(ctx, costs, b):
        np_costs = costs.detach().cpu().numpy()
        (n, b2) = costs.shape;
        allSolutions = np.zeros((n, b2), dtype=np.float32);
        for i in range(0, n):
            matchingCostMatrix = np.reshape(np_costs[i, :], (b, b))
            rowids, colids = solve_dense(matchingCostMatrix);
            matchingSolution = np.zeros((b, b), dtype=np.float32);
            matchingSolution[rowids, colids] = 1.0;
            allSolutions[i, :] = np.reshape(matchingSolution, [1, b2]);
        ctx.allSolutions = torch.from_numpy(allSolutions)
            .float().to(costs.device);
        return torch.sum(torch.mul(costs, ctx.allSolutions), axis=1);

    @staticmethod
    def backward(ctx, grad_output):
        return torch.mul(ctx.allSolutions, grad_output.unsqueeze(1)), None;

```

A.2 DIFFERENTIATING OVER GLOBAL SEQUENCE ALIGNMENT FOR SENTENCE-LEVEL LOSS IN SEQUENCE-TO-SEQUENCE MODELS

In experiments involving global sequence alignment in sequence-to-sequence models, we used an encoder-decoder sequence-to-sequence architecture with bidirectional forward-backward RNN encoder and an attention-based RNN decoder (Luong et al., 2015), as implemented in PyTorch-Text (Hu et al., 2018). In both the encoder and the decoder we used 256 units and dropout rate of 0.2. We used batch size of 128, and learning rate of 0.001 with Adam optimizer.

In the RNN, in both softmax and Gumbel-softmax, we use annealing of the temperature parameter τ . As it has been used previously (Kusner & Hernández-Lobato, 2016), we start with a high value of the temperature parameter τ and reduced it as training progresses. Specifically, we started with $\tau = 5$, reduced it by 0.5 in each epoch until value of 1.0 is reached, and then kept it fixed at 1.0.

In evaluating the combinatorial GSA loss, we used text summarization task involving the GIGAWORD dataset (Graff & Cieri, 2003) as an example of a sequence-to-sequence problem. We used the same preprocessing as (Chen et al., 2019), that is, we used 200K examples in training. We used the validation set to select the best model epoch, and reported results on a separate test set.

REFERENCES

- Liquan Chen, Yizhe Zhang, Ruiyi Zhang, Chenyang Tao, Zhe Gan, Haichao Zhang, Bai Li, Dinghan Shen, Changyou Chen, and Lawrence Carin. Improving sequence-to-sequence learning via optimal transport. In *International Conference on Learning Representations*, pp. arXiv:1901.06283, 2019.
- David Graff and C Cieri. English Gigaword corpus. *Linguistic Data Consortium*, 2003.
- Zhiting Hu, Haoran Shi, Bowen Tan, Wentao Wang, Zichao Yang, Tiancheng Zhao, Junxian He, Lianhui Qin, Di Wang, et al. Texar: A modularized, versatile, and extensible toolkit for text generation. *arXiv preprint arXiv:1809.00794*, 2018.
- Matt J Kusner and José Miguel Hernández-Lobato. GANs for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*, 2016.
- Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412—1421, 2015.
- Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *International Conference on Learning Representations ICLR’17*. *arXiv:1605.07146*, 2016.