[Paper review 43]

A Theoretically Grounded Application of Dropout in Recurrent Neural Networks

(Gal, Ghahramani, 2016)

[Contents]

1. Abstract

RNN's major difficulty :

tendency to overfit , **but dropout is shown to fail** on recurrent layers!

This paper, offers insights into the use of dropout with RNN models

Apply **VI based dropout techniques** in LSTM & GRU

assess it to language model & sentiment analysis task

2. Introduction

RNN

- sequence based models, key to NLP
- but overfit quickly

  (+ lack of regularization ..... difficult on small dataset)
- dropout has not been successful

with Bayesian view!

- **"Dropout = variational approximation to the posterior of BNN"** (Gal and Ghahramani)
- RNNs with weights, treated as **random variables**
- perform approximate VI in these probabilistic Bayesian models

  (called **Variational RNNs**)
- weights with mixture of Gaussians $\rightarrow$ tractable optimization objective

$\rightarrow$ Identical to performing a new variant of DROPOUT in the respective RNNs!

Dropout in RNN

- repeat the SAME dropout mask at each time step for inputs/outputs/recurrent layers

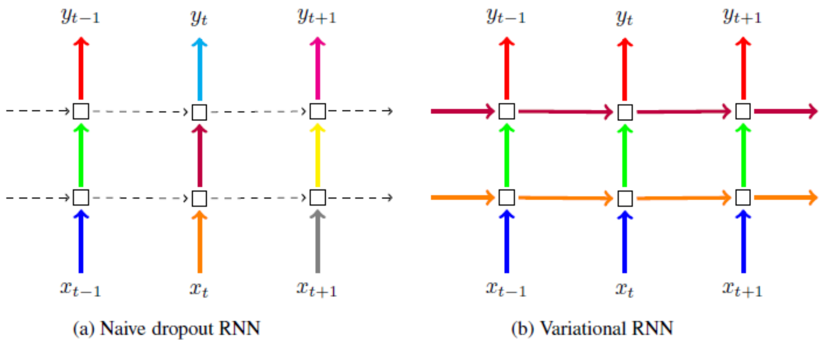

Figure 1: **Depiction of the dropout technique following our Bayesian interpretation (right) compared to the standard technique in the field (left).** Each square represents an RNN unit, with horizontal arrows representing time dependence (recurrent connections). Vertical arrows represent the input and output to each RNN unit. Coloured connections represent dropped-out inputs, with different colours corresponding to different dropout masks. Dashed lines correspond to standard connections with no dropout. Current techniques (naive dropout, left) use different masks at different time steps, with no dropout on the recurrent layers. The proposed technique (Variational RNN, right) uses the same dropout mask at each time step, including the recurrent layers.

.

3. Background

3-1. BNN

Softmax Likelihood (for classification)

- $p(y=d \mid \mathbf{x}, \boldsymbol{\omega})= \text{Categorical} \left(\exp \left(f_d^{\boldsymbol{\omega}}(\mathbf{x})\right) / \sum_{d^{\prime}} \exp \left(f_{d^{\prime}}^{\boldsymbol{\omega}}(\mathbf{x})\right)\right).$

Predictive distribution

- $p\left(\mathbf{y}^* \mid \mathbf{x}^*, \mathbf{X}, \mathbf{Y}\right)=\int p\left(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\omega}\right) p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y}) \mathrm{d} \boldsymbol{\omega}.$

Prior

- place a prior distn over NN's weight $\rightarrow$ BNN
- often place standard matrix Gaussian prior, $p\left(\mathbf{W}_i\right)=\mathcal{N}(\mathbf{0}, \mathbf{I})$.

3-2. Approximate VI in BNN

posterior is intractable, use VI

minimize KL-div, $\mathrm{KL}(q(\boldsymbol{\omega}) \| p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y})) \propto -\int q(\boldsymbol{\omega}) \log p(\mathbf{Y} \mid \mathbf{X}, \boldsymbol{\omega}) \mathrm{d} \boldsymbol{\omega}+\mathrm{KL}(q(\boldsymbol{\omega}) \| p(\boldsymbol{\omega}))$.

extend this to **probabilistic RNNs**

4. VI in RNNs

use simple RNN models for simplicity (LSTM, GRU)

(1) hidden state h_t :

- $\mathbf{h}_t=\mathrm{f}_{\mathbf{h}}\left(\mathbf{x}_t, \mathbf{h}_{t-1}\right)=\sigma\left(\mathbf{x}_t \mathbf{W}_{\mathbf{h}}+\mathbf{h}_{t-1} \mathbf{U}_{\mathbf{h}}+\mathbf{b}_{\mathbf{h}}\right)$

  where σ : non-linearity function

(2) output of model

- $\mathrm{f}_{\mathbf{y}}\left(\mathbf{h}_T\right)=\mathbf{h}_T \mathbf{W}_{\mathbf{y}}+\mathbf{b}_{\mathbf{y}}$.

RNN models (by (1) & (2)) :

- parameters : $\omega=\left\{\mathbf{W}_{\mathbf{h}}, \mathbf{U}_{\mathbf{h}}, \mathbf{b}_{\mathbf{h}}, \mathbf{W}_{\mathbf{y}}, \mathbf{b}_{\mathbf{y}}\right\}$

  (random variables, following Normal prior)

- using MC integration..

$$\begin{aligned} \int q(\boldsymbol{\omega}) \log p\left(\mathbf{y} \mid \mathbf{f}_{\mathbf{y}}^{\boldsymbol{\omega}}\left(\mathbf{h}_T\right)\right) \mathrm{d} \boldsymbol{\omega} &= \int q(\boldsymbol{\omega}) \log p\left(\mathbf{y} \mid \mathbf{f}_{\mathbf{y}}^{\boldsymbol{\omega}}\left(\mathbf{f}_{\mathbf{h}}^{\boldsymbol{\omega}}\left(\mathbf{x}_T, \mathbf{h}_{T-1}\right)\right)\right) \mathrm{d} \boldsymbol{\omega} \\ &= \int q(\boldsymbol{\omega}) \log p\left(\mathbf{y} \mid \mathbf{f}_{\mathbf{y}}^{\boldsymbol{\omega}}\left(\mathbf{f}_{\mathbf{h}}^{\boldsymbol{\omega}}\left(\mathbf{x}_T, \mathbf{f}_{\mathbf{h}}^{\boldsymbol{\omega}}\left(\ldots \mathbf{f}_{\mathbf{h}}^{\boldsymbol{\omega}}\left(\mathbf{x}_1, \mathbf{h}_0\right) \ldots\right)\right)\right)\right) \mathrm{d} \boldsymbol{\omega} \\ &\approx \log p\left(\mathrm{y} \mid \mathrm{f}_{\mathrm{y}}^{\widehat{\omega}}\left(\mathrm{f}_{\mathrm{h}}^{\widehat{\omega}}\left(\mathrm{x}_T, \mathrm{f}_{\mathrm{h}}^{\widehat{\omega}}\left(\ldots \mathrm{f}_{\mathrm{h}}^{\widehat{\omega}}\left(\mathrm{x}_1, \mathrm{~h}_0\right) \ldots\right)\right)\right)\right), \quad \widehat{\omega} \sim q(\omega) \end{aligned}$$

  .

  $\rightarrow$ unbiased estimator to each sum term

Objective function (minimize)

- $\mathcal{L} \approx-\sum_{i=1}^N \log p\left(\mathbf{y}_i \mid \mathbf{f}_{\mathbf{y}}^{\widehat{\omega}_i}\left(\mathrm{f}_{\mathbf{h}}^{\widehat{\omega}_i}\left(\mathbf{x}_{i, T}, \mathrm{f}_{\mathbf{h}}^{\widehat{\omega}_i}\left(\ldots \mathrm{f}_{\mathbf{h}}^{\widehat{\omega}_i}\left(\mathrm{x}_{i, 1}, \mathrm{~h}_0\right) \ldots\right)\right)\right)\right)+\mathrm{KL}(q(\omega) \| p(\omega))$.

For each sequence x_i, sample new realization $\widehat{\omega}_i=\left\{\widehat{\mathbf{W}}_{\mathbf{h}}^i, \widehat{\mathbf{U}}_{\mathbf{h}}^i, \widehat{\mathbf{b}}_{\mathbf{h}}^i, \widehat{\mathbf{W}}_{\mathbf{y}}^i, \widehat{\mathbf{b}}_{\mathbf{y}}^i\right\}$.

Approximating distribution :

$q\left(\mathbf{w}_k\right)=p \mathcal{N}\left(\mathbf{w}_k ; \mathbf{0}, \sigma^2 I\right)+(1-p) \mathcal{N}\left(\mathbf{w}_k ; \mathbf{m}_k, \sigma^2 I\right)$.

- m_k : variational parameters (of random weight matrices)
- p : dropout probability

$\rightarrow$ optimize over m_k!

(2nd term (KL-term) can be approximated as L_2 regularization)

Key point : **"SAME MASK is used through all time steps"**

Prediction :

- method 1) propagating the mean of each layer to next
  (= standard dropout approximation)
- method 2) approximating the posterior
  - approximate $p\left(\mathbf{y}^* \mid \mathbf{x}^*, \mathbf{X}, \mathbf{Y}\right)=\int p\left(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\omega}\right) p(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y}) \mathrm{d} \boldsymbol{\omega}$ with $q(w)$
  - Thus, $p\left(\mathbf{y}^* \mid \mathbf{x}^*, \mathbf{X}, \mathbf{Y}\right) \approx \int p\left(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\omega}\right) q(\boldsymbol{\omega}) \mathrm{d} \boldsymbol{\omega} \approx \frac{1}{K} \sum_{k=1}^K p\left(\mathbf{y}^* \mid \mathbf{x}^*, \widehat{\boldsymbol{\omega}}_k\right)$
    where $\widehat{\omega}_k \sim q(\omega)$

4-1. Implementation and Relation to Dropout in RNNs

Implementing **approximate inference**

= Implementing **dropout in RNNs** with **same network units dropped** at each time step

LSTM's 4 gates : input / forget / output / input modulation

2 notations

- 1) United-weights LSTM
- 2) Tied-weights LSTM

1) United-weights LSTM

$$
\begin{array}{ll}
\underline{\mathbf{i}}=\operatorname{sigm}\left(\mathbf{h}_{t-1} \mathbf{U}_i+\mathbf{x}_t \mathbf{W}_i\right) & \underline{\mathbf{f}}=\operatorname{sigm}\left(\mathbf{h}_{t-1} \mathbf{U}_f+\mathbf{x}_t \mathbf{W}_f\right) \\
\underline{\mathbf{o}}=\operatorname{sigm}\left(\mathbf{h}_{t-1} \mathbf{U}_o+\mathbf{x}_t \mathbf{W}_o\right) & \underline{\mathbf{g}}=\tanh \left(\mathbf{h}_{t-1} \mathbf{U}_g+\mathbf{x}_t \mathbf{W}_g\right) . \\
\mathbf{c}_t=\underline{\mathbf{f}} \circ \mathbf{c}_{t-1}+\underline{\mathbf{i}} \circ \underline{\mathbf{g}} & \mathbf{h}_t=\underline{\mathbf{o}} \circ \tanh \left(\mathbf{c}_t\right)
\end{array}
$$

2) Tied-weights LSTM

$$
\left(\begin{array}{c}
\underline{\mathbf{i}} \\
\underline{\mathbf{f}} \\
\underline{\mathbf{o}} \\
\underline{\mathbf{g}}
\end{array}\right)=\left(\begin{array}{c}
\operatorname{sigm} \\
\operatorname{sigm} \\
\operatorname{sigm} \\
\tanh
\end{array}\right)\left(\left(\begin{array}{c}
\mathbf{x}_t \\
\mathbf{h}_{t-1}
\end{array}\right) \cdot \mathbf{W}\right) .
$$

Even though, 1) and 2) result in the same deterministic model,

lead to different approximating distribution $q(w)$!

- for 1) .... could use different dropout masks for different gates
- for 2) ... place a distribution over the single matrix $\mathbf{W}$
  dropout variant with 2)

$$\begin{pmatrix} \underline{\mathbf{i}} \\ \underline{\mathbf{f}} \\ \underline{\mathbf{o}} \\ \underline{\mathbf{g}} \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} \left(\begin{pmatrix} \mathbf{x}_t \circ \mathbf{z}_\mathbf{x} \\ \mathbf{h}_{t-1} \circ \mathbf{z}_\mathbf{h} \end{pmatrix} \cdot \mathbf{W} \right).$$

Others

- Zaremba et al. [4] 's dropout variant

$$\begin{pmatrix} \underline{\mathrm{i}} \\ \underline{\mathbf{f}} \\ \underline{\mathbf{o}} \\ \underline{\mathrm{g}} \end{pmatrix} = \begin{pmatrix} \text{sig m} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} \left(\begin{pmatrix} \mathrm{x}_t \circ \mathbf{z}_\mathrm{x}^t \\ \mathbf{h}_{t-1} \end{pmatrix} \cdot \mathbf{W} \right)$$

- Moon et al.'s dropout variant

  replace $\mathbf{c}_t = \mathbf{c}_t \circ \mathbf{z}_\mathbf{c}$. in (1) United-weights LSTM

4-2. Word Embeddings Dropout

Continuous vs Discrete

- In continuous inputs..
  - apply dropout to input layer

    (= placing a distn over weight matrix)
- In discrete inputs...
  - seldom done...

Discrete Input

- product of **one-hot encoded vector** & **embedding matrix** = word embedding

  (embedding matrix : $\mathrm{W}_E \in \mathbb{R}^{V \times D}$)
- this parameter layer is largest layer in NLP, but often not regularized... :(
- Thus, apply dropout to the one-hot encoded vector!

  (= dropping words at random)

Dropping words?

- sample V Bernoulli r.v (what is V is too large...?)
- with sequence of length T, at most T embeddings could be dropped

  $\rightarrow$ first map the words to the word embedding, then zero-out word embeddings!

  (more efficient)

5. Conclusion

New technique for RNN regularization, **RNN dropout variant**