Normalization tricks

Xingdong Zuo*

Definition 0.1. *Covariate shift refers to the change of input distribution while the conditional distribution of outputs remains unchanged.*

The behavior of algorithms can change when input distribution changes. For example, each layer in neural networks is forced to continuously adapt to its changing input because the distribution of activations from previous layer changes.

Normalizing the inputs to be zero mean and unit variance can help to accelerate training (before nonlinearity).

Batch normalization

Algorithm 1: Batch normalization

input : Mini-batch $\mathcal{B} = \{x_1, \dots, x_m\}$; trainable parameters γ, β
output : $y_i, \forall i = 1, \dots, m$
begin
$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$
$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \varepsilon}}$
$y_i \leftarrow \gamma \hat{x}_i + \beta$
end

Limitation of batch normalization

- Batch size dependent: small batch size increase variance of estimating statistics.
- Non-trivial to use for RNN: each time step has different statistics. More problematic for varied length sequences.

Layer normalization

Layer normalization normalizes the inputs over the feature dimension instead of batch dimension.

Let $X \in \mathbb{R}^{m \times n}$ be an input data with batch size m and feature size n. The estimation of statistics becomes following instead:

$$\mu_i \leftarrow \frac{1}{n}\sum_{j=1}^{n} x_{ij}$$

$$\sigma_i^2 \leftarrow \frac{1}{n}\sum_{j=1}^{n}(x_{ij} - \mu_i)^2$$

$$\hat{x}_{ij} \leftarrow \frac{x_{ij} - \mu_i}{\sqrt{\sigma_i^2 + \varepsilon}}$$

Properties

*November 5, 2018

- Independent of batch size, useful when batch size is one
- Each training example has its own statistics
- All neurons in a layer share the same statistics
- Useful for RNN with varied sequence length
- Not working well for CNN

Layer normalization is used in LSTM in the following way:

$$\begin{pmatrix} \mathbf{f}_t \\ \mathbf{i}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = LN(\mathbf{W}_h \mathbf{h}_{t-1}; \boldsymbol{\gamma}_1, \boldsymbol{\beta}_1) + LN(\mathbf{W}_x \mathbf{x}_t; \boldsymbol{\gamma}_2, \boldsymbol{\beta}_2) + \mathbf{b}$$

$$\mathbf{c}_t = \sigma(\mathbf{f}_t) \odot \mathbf{c}_{t-1} + \sigma(\mathbf{i}_t) \odot \tanh(\mathbf{g}_t)$$

$$\mathbf{h}_t = \sigma(\mathbf{o}_t) \odot \tanh(LN(\mathbf{c}_t; \boldsymbol{\gamma}_3, \boldsymbol{\beta}_3))$$

where $LN(\cdot; \boldsymbol{\gamma}, \boldsymbol{\beta})$ is the layer normalization operation with trainable parameters $\boldsymbol{\gamma}, \boldsymbol{\beta}$.