Batch Normalization

mingzailao

2016-9-11

Outline

1. Introduction
2. Towards Reducing Internal Covariate Shift
3. Normalization via Mini-Batch Statistics

SGD Introduction

SGD optimize function

$$\Theta = arg\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} l(x_i, \Theta) \tag{1}$$

- $x_{1...N}$: the training set
- $l(\cdot)$: loss function
- Θ : the parameters.

with SGD, the training proceeds in steps,and at each step we consider a minibatch $x_{1...m}$ of size m.

SGD workflow

How SGD works

The mini-batch is used to approximate the gradient of the loss function with respect to the parameters, by computing

$$\frac{1}{m}\sum_{i=1}^{m}\frac{\partial l(x_i,\Theta)}{\partial\theta} \tag{2}$$

Fixed distribution of inputs

Positive consequences

Consider a layer with a sigmoid activation function

$$z = g(Wu + b)$$

where

$$g(x) = \frac{1}{1 + exp(-x)}$$

As $|x|$ increases, $g'(x)$ tends to zero.

Fixed distribution of inputs

Positive cinsequences

This means that for all dimensions of $x = Wu + b$ except those with small absolute values, the gradient flowing down to u will vanish and the model will train slowly. If, however, we could ensure that the distribution of nonlinearity inputs remains more stable as the network trains, then the optimizer would be less likely to get stuck in the saturated regime, and the training would accelerate.

Internal Covariate Shift

Definition

The change in the distribution of network activations due to the change in network parameters during training.

The way to reducing the Internal Convariate Shift

By fixing the distribution of the layer inputs x as the training progresses, we expect to improve the training speed.

Example

Whiting

Normalization via Mini-Batch Statistics

From x to $\hat{x}$

For a layer with d-dimensional input $x = (x^{(1)}, x^{(2)}, \cdots, x^{(d)})$,we will normalize each dimension

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}} \tag{3}$$

Notes

Simply normalizing each input of a layer may change what the layer can represent.

Normalization via Mini-Batch Statistics

From $\hat{x}$ to y

To address the problem above, we make sure that the transformation inserted in the network can represent the identity transform.

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)} \tag{4}$$

- $\gamma^{(k)} = \sqrt{Var[x^{(k)}]}$
- $\beta^{(k)} = E[x^{(k)}]$

Normalization via Mini-Batch Statistics

Algorithm

- Input :Values of x over a mini-batch: $\mathcal{B} = \{x_{1..m}\}$; Parameters to be learned: γ, β
- Output :$\{y_i = BN_{\gamma,\beta}(x_i)\}$

1. $\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
2. $\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$
3. $\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$
4. $y_i \leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$

The BN transform can be added to a network to manipulate any activation.

Normalization via Mini-Batch Statistics

BP

$$
\begin{aligned}
\frac{\partial l}{\partial \hat{x}_i} &= \frac{\partial l}{\partial y_i} \cdot \gamma \\
\frac{\partial l}{\partial \sigma_{\mathcal{B}}^2} &= \sum_{i=1}^{m} \frac{\partial l}{\partial \hat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \left(-\frac{1}{2}\right)(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2} \\
\frac{\partial l}{\partial \mu_{\mathcal{B}}} &= \sum_{i=1}^{m} \frac{\partial l}{\partial \hat{x}_i} \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial l}{\partial \sigma_{\mathcal{B}}^2} \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m} \\
\frac{\partial l}{\partial x_i} &= \frac{\partial l}{\partial \hat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial l}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial l}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m} \\
\frac{\partial l}{\partial \gamma} &= \sum_{i=1}^{m} \frac{\partial l}{\partial y_i} \cdot \hat{x}_i \\
\frac{\partial l}{\partial \beta} &= \sum_{i=1}^{m} \frac{\partial l}{\partial y_i}
\end{aligned}
$$

Training and Inference with Batch- Normalized Networks

Algorithm

- Input : Network N with trainable parameters Θ; subset of activations $\{x^{(k)}\}_{k=1}^{K}$
- Output : Batch-normalized network for inference, N_{BN}^{inf}

1. $N_{BN}^{tr} \leftarrow N$
2. for k=1...K do:
   1. Add transformation $y^{(k)} = BN_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to N_{BN}^{tr}
   2. Modify each layer in N_{BN}^{tr} with input $x^{(k)}$ to take $y^{(k)}$ instead
3. Train N_{BN}^{tr} to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$

Training and Inference with Batch- Normalized Networks

Algorithm

1. $N_{BN}^{inf} \leftarrow N_{BN}^{tr}$
2. For k=1...K do
   1. // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}$, $\mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$ etc.
   2. Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$E[x] \leftarrow E_{\mathcal{B}}[\mu_{\mathcal{B}}]$$

$$Var[x] \leftarrow \frac{m}{m-1} E_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

   3. In N_{BN}^{inf}, replace the transform $y = BN_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{Var[x]+\epsilon}} \cdot x + (\beta - \frac{\gamma E[x]}{\sqrt{Var[x]+\epsilon}})$