Batch Normalization

ZZL

2017 年 7 月 25 号

Introduction

在训练深层神经网络中，在每一层 layer 中，数据的分布随着上一层 layer 参数的变化而发生变化。每次训练，参数都需要适应新的数据分布，影响了训练的速度。由于这种性质，使得我们不得不采用较低的学习率和仔细的初始化参数。本文提出了一种方法来克服这样的缺点。

1. 深层网络中的缺陷

在本文中，我们用批量梯度下降算法来跟新参数。

$\mathrm{L}=F_2(F_1(u,\emptyset_1),\emptyset_2)$ 是一个两层神经网络，F_1、F_2表示每层网络的运算，$\emptyset_1$、$\emptyset_1$表示每层的参数。

我们将$F_1(u,\emptyset_1)$表示为 x，是F_2层的输入，那么有$\mathrm{L}=F_2(x,\emptyset_2)$。

利用批量梯度下降算法更新$\emptyset_2$，$\emptyset_2=\emptyset_2-\frac{\alpha}{m}\sum_{i=1}^{m}\frac{\partial F_2(x_i,\emptyset_2)}{\partial\emptyset_2}$（$\alpha$是学习率，m 是每批数据的数量）。X 的分布，随着$\emptyset_1$的变化而变化，$\emptyset_2$为了适应输入数据分布的变化而变化，影响了训练的速度。

Layer 中激活函数的特性也会影响网络的训练。一个 layer 的公式表示$\mathrm{z}=\mathrm{f}(\mathrm{WX}+\mathrm{b})$,f 是激活函数，W、b 是参数。如果 f 是 sigmoid 函数，它的倒数在 WX+b 的绝对值较大时趋近于 0。随着训练次数增加，W、b 也在变化，在训练后期 WX+b 绝对值较大，使得训练速率下降，这种情况随着层数的增加而变得严重，也就是梯度消失。有人提出用 ReLU(x) = max(x,0)，激活函数，但它需要仔细的初始化参数，设置较小的学习率。如果我们能保证数据分布是一个稳定的状态，那么我们就可能回避这些问题，加速训练。

2. reduce Covariate Shift

如果数据为白化数据，那么网络训练将会加快[2]。白化指的是（1）减少数据特征之间的关联性（2）使特征协方差矩阵为单位阵。

白化的目的是使特征之间独立，便于后续特征分量提取。但白化需要计算数据的协方差矩阵（比如有基于 PCA 的白化），计算量大，本文不采用。本文是一种简化的方法，大大的减少了计算量，并取得不错的效果。

3. Normalization

对输入数据$\mathrm{x}=(x^{(1)}x^{(2)}\cdots x^{(3)})$的每一次做如下处理

$$\widehat{x}^{(k)}=\frac{x^{(k)}-\mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

这样处理后，数据是均值为 0，方差为 1。但这样的变换会丢失数据的特性，我们对每个

维度加上一组参数$\gamma^{(k)}, \beta^{(k)}$，对数据做放缩旋转变换，使数据尽可能保留更多的特性。具体转换公式如下：$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)}$。

下图是 BN 对批量数据转换的计算流程：

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

网络损失用 L 表示，利用链式法则求出每个参数的导数，再用反向传播算法更新参数即可。

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2+\epsilon}}\right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i-\mu_{\mathcal{B}})}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2+\epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i-\mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

4. Batch Normalization network

我们取多次训练过程中，批量数据均值的期望 E[X]和方差$E_\beta[\sigma_\beta^2]$的期望作为网络预测过程中的参数。预测过程中，一次只有一个数据，那么对数据的 Normalization 处理就依靠 E[X]和$E_\beta[\sigma_\beta^2]$。

$$\widehat{x} = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}}$$

这是对输入数据 x 的 Normalization 处理，这里$\mathrm{Var}[x] = \frac{m}{m-1} \cdot \mathrm{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$（这里是样本方差得无偏估计，具体见附录）。M 是每批数据的个数。

网络训练流程如下：

Input: Network N with trainable parameters Θ;
subset of activations $\{x^{(k)}\}_{k=1}^{K}$

Output: Batch-normalized network for inference, $N_{\mathrm{BN}}^{\mathrm{inf}}$

1: $N_{\mathrm{BN}}^{\mathrm{tr}} \leftarrow N$ // Training BN network

2: **for** $k = 1 \ldots K$ **do**

3: Add transformation $y^{(k)} = \mathrm{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\mathrm{BN}}^{\mathrm{tr}}$ (Alg. 1)

4: Modify each layer in $N_{\mathrm{BN}}^{\mathrm{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead

5: **end for**

6: Train $N_{\mathrm{BN}}^{\mathrm{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$

7: $N_{\mathrm{BN}}^{\mathrm{inf}} \leftarrow N_{\mathrm{BN}}^{\mathrm{tr}}$ // Inference BN network with frozen
// parameters

8: **for** $k = 1 \ldots K$ **do**

9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.

10: Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\mathrm{E}[x] \leftarrow \mathrm{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\mathrm{Var}[x] \leftarrow \frac{m}{m-1}\mathrm{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

11: In $N_{\mathrm{BN}}^{\mathrm{inf}}$, replace the transform $y = \mathrm{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\mathrm{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x]+\epsilon}}\right)$

12: **end for**

Algorithm 2: Training a Batch-Normalized Network

5. Batch-Normalized Convolutional Networks

$z = f(WX + b)$,X 是网络的输入，W、b 是参数，f 是非线性函数。下面的问题是，我们讲 Normalized 操作 X 上，还是 WX+b 后的数据上。文章中说 WX+b 更可能是对称非稀疏的，也更“高斯”[3]，所以操作放在 WX+b 后。因为做 Normalized，所以这里不需要偏执 b 公式为$z = f(\mathrm{BN}(WX))$。

对于 CNN，这里的参数γ、β采用共享权重的机制，即每个 filter 内的数据采用同一组参数。

6．总结

对于一般的深层网络，大的学习率会放大参数的数值，会导致模型爆炸（目前不懂）。对于 BN net 来说，不存在该问题。

比如讲参数 W 放大 a 倍，BN(Wu)=BN((aw)u)，通过简单求导计算，我们可以得出如下

$$\frac{\partial \mathrm{BN}((aW)\mathrm{u})}{\partial \mathrm{u}}=\frac{\partial \mathrm{BN}(W\mathrm{u})}{\partial \mathrm{u}}$$

结论 $\frac{\partial \mathrm{BN}((aW)\mathrm{u})}{\partial (aW)}=\frac{1}{a}\cdot\frac{\partial \mathrm{BN}(W\mathrm{u})}{\partial W}$。参数增大使得梯度变小，不会产生模型爆炸的问题。

BN 网络可以使用大的学习率，加速了网络的训练，并且训练过程通常不需要 dropout 这些的小 trick。

思考

就我个人而言，我本科接触得绝大部分都是 P 问题，思维得训练方向也是寻找最优可行解。现在遇到得问题大多是 NP 问题，由于思维惯性，我会潜意识得去掉某些方法，而有些文章里面得方法类型恰恰是我潜意识会直接丢弃的类型。比如某些细小领域的开山之作，会有较强的假设，有很多不合理，但它的作用是踏出了第一步，对后来者有启发意义。现在我自己需要做的是去除思维惯性，扩大思维面。

参考文献

[1] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]//International Conference on Machine Learning. 2015: 448-456.
[2] Wiesler S, Ney H. A convergence analysis of log-linear training[C]//Advances in Neural Information Processing Systems. 2011: 657-665.
[3] Hyvärinen A, Oja E. Independent component analysis: algorithms and applications[J]. Neural networks, 2000, 13(4): 411-430.

附录

$$
\begin{aligned}
E(S_1^2) &= \frac{1}{n}\sum_{i=1}^{n} E\left((X_i-\overline{X})^2\right) = \frac{1}{n}E\left(\sum_{i=1}^{n}(X_i-\mu+\mu-\overline{X})^2\right) \\
&= \frac{1}{n}E\left(\sum_{i=1}^{n}\left((X_i-\mu)^2-2(X_i-\mu)(\overline{X}-\mu)+(\overline{X}-\mu)^2\right)\right) \\
&= \frac{1}{n}E\left(\sum_{i=1}^{n}(X_i-\mu)^2-2\sum_{i=1}^{n}(X_i-\mu)(\overline{X}-\mu)+n(\overline{X}-\mu)^2\right) \\
&= \frac{1}{n}E\left(\sum_{i=1}^{n}(X_i-\mu)^2-2n(\overline{X}-\mu)(\overline{X}-\mu)+n(\overline{X}-\mu)^2\right) \\
&= \frac{1}{n}E\left(\sum_{i=1}^{n}(X_i-\mu)^2-n(\overline{X}-\mu)^2\right) \\
&= \frac{1}{n}\left(\sum_{i=1}^{n}E\left((X_i-\mu)^2\right)-nE\left((\overline{X}-\mu)^2\right)\right) \\
&= \frac{1}{n}\left(nVar(X)-nVar(\overline{X})\right) \\
&= Var(X)-Var(\overline{X}) = \sigma^2-\frac{\sigma^2}{n} = \frac{n-1}{n}\,\sigma^2,
\end{aligned}
$$