Deep Learning

Batch Normalization

Prof. Dr. Christoph Lippert

Digital Health & Machine Learning

Let's review some of the practical challenges when training deep networks.

1. Often we standardize our input features to each have a mean of *zero* and variance of *one*.
2. Activations in intermediate layers of a deep network assume different orders of magnitude.
3. 100x differences between layers might require adjusting learning rates per layer (or even per node within a layer).
4. **Batch normalization** was introduced in 2015 bz Ioffe and Szegedy to avoid this drift in the distribution of activations to improve training of deep networks.

Batch Norm

The activation at a given layer is transformed from $\mathbf{x}$ to

$$\mathrm{BN}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \hat{\mu}}{\hat{\sigma}} + \beta$$

Here, $\hat{\mu}$ is the estimate of the mean and $\hat{\sigma}$ is the estimate of the ~~variance~~. standard deviation

- With large enough mini-batches, the approach proves effective and stable.[a]
- The result is that the activations are approximately re-scaled to zero mean and unit variance.
- We allow for a coordinate-wise scaling coefficient γ and an offset β

[a] Note that if our batch size was 1, we wouldn't be able to learn anything because during training, every hidden node would take value 0.

$$\mathrm{BN}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \hat{\mu}}{\hat{\sigma}} + \beta$$

Consequently, the activations for intermediate layers cannot diverge any longer: we are actively re-scaling them back to a given order of magnitude via μ and σ.

- This normalization allows us to be more aggressive in picking large learning rates.
- To address the fact that in some cases the activations may actually *need* to differ from standardized data, BN also introduces scaling coefficients γ and an offset β.

$$\mathrm{BN}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \hat{\mu}}{\hat{\sigma}} + \beta$$

In principle, we might want to use all of our training data to estimate the mean and variance.

- Activations for each example change each time we update our model.
- BN uses only the current mini-batch for estimating $\hat{\mu}$ and $\hat{\sigma}$.
- As we normalize based only on the *current batch* the procedure is called *batch normalization*
- To indicate which mini-batch $\mathcal{B}$ we draw this from, we denote the quantities with $\hat{\mu}_{\mathcal{B}}$ and $\hat{\sigma}_{\mathcal{B}}$.

$$\hat{\mu}_{\mathcal{B}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{\mathbf{x} \in \mathcal{B}} \mathbf{x} \quad \text{and} \quad \hat{\sigma}^2_{\mathcal{B}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{\mathbf{x} \in \mathcal{B}} (\mathbf{x} - \hat{\mu}_{\mathcal{B}})^2 + \epsilon$$

- constant $\epsilon > 0$ ensures that we never divide by zero.
- Variation in $\hat{\mu}_{\mathcal{B}}$ and $\hat{\sigma}_{\mathcal{B}}$ acts as a regularization, mitigating overfitting.
- Teye, Azizpour and Smith, 2018 and Luo et al, 2018 relate the properties of BN to Bayesian Priors and penalties respectively.

BN in Fully Connected Layers

Usually we apply the batch normalization layer between the affine transformation and the activation function in a fully-connected layer.
Denote by $\mathbf{u}$ the input and by $\mathbf{x} = \mathbf{W}\mathbf{u} + \mathbf{b}$ the output of the linear transform.

$$\mathbf{y} = \phi(\mathrm{BN}(\mathbf{x})) = \phi(\mathrm{BN}(\mathbf{W}\mathbf{u} + \mathbf{b}))$$

BN in Convolutional Layers

For convolutional layers, batch normalization occurs after the convolution computation and before the application of the activation function.
For multiple convolution channels, we need to carry out batch normalization for *each* of the outputs of these channels, and each channel has an independent scale $\sigma > 0$ and shift $\mu \in \mathbb{R}$.

- m examples in the mini-batch
- height p and width q of the convolution computation output
- We need to carry out batch normalization for $m \times p \times q$ elements in each channel simultaneously.
- we estimate μ and σ from the $m \times p \times q$ elements in this channel rather than one per pixel.

BN at Test time

- At prediction time, we are required to make one prediction at a time.
- The variability in μ and σ estimates, from a mini-batches are undesirable once we've trained the model.
- One way to mitigate this is to compute more stable estimates on a larger set for once (e.g. via a moving average) and then fix them at prediction time.
- Consequently, BN behaves differently during training and at test time.

Summary – Batch Norm

- During training, BN continuously adjusts the layer activations of the neural network using the mean and standard deviation of the mini-batch.
- BN for fully connected layers and convolutional layers are slightly different.
- Like a dropout layer, BN layers have different computation results in training and prediction.
- BN has regularizating effect.

[1] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.