Recurrent Neural Networks

Kate Farrahi

ECS Southampton

March 7, 2019

Batch Normalisation

- ▶ Batch normalization (BN) is a technique for improving the performance of ANNs
- ▶ It improves the stability of ANNs by adjusting and scaling the activations
- ▶ BN makes your ANN more robust to the choice of hyperparameters (larger range of parameters that will work well)
- ▶ It was introduced by Sergey Ioffe and Christian Szegedy in 2015 [1]

---

[1] https://arxiv.org/pdf/1502.03167.pdf

Batch Normalisation

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

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

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

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

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

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

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

Batch Normalisation

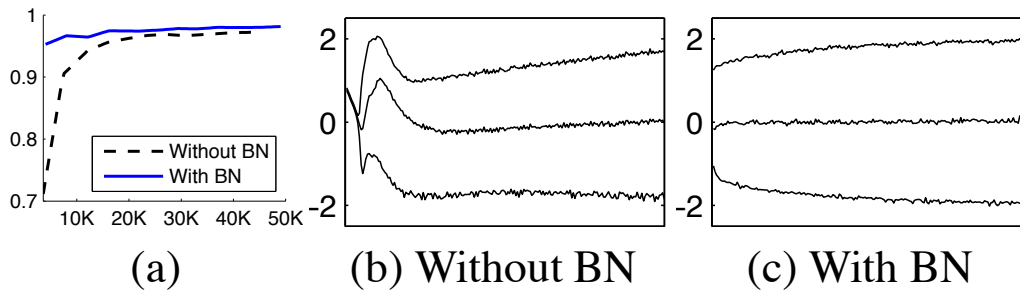(a)   (b) Without BN   (c) With BN

Figure 1: (a) *The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and achieve higher accuracy. (b, c) The evolution of input distributions to a typical sigmoid, over the course of training, shown as $\{15, 50, 85\}th$ percentiles. Batch Normalization makes the distribution more stable and reduces the internal covariate shift.*

Batch Normalisation
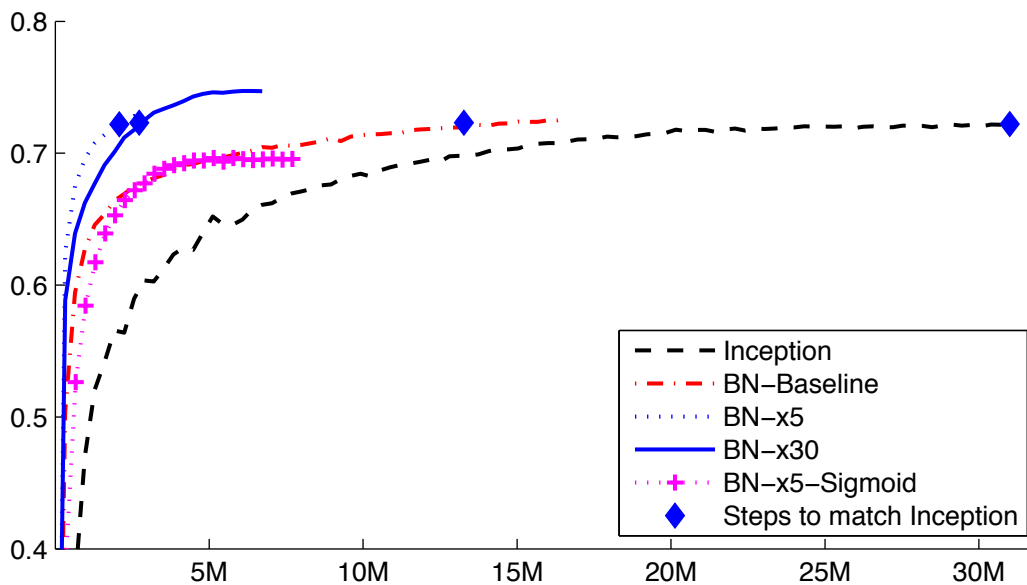

Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

Recurrent Neural Networks - Motivation

x:	Kate	Farrahi	and	Jonathon	Hare	teach	deep	learning
y:	1	1	0	1	1	0	0	0

Recurrent Neural Networks - Motivation

$$x: \quad x^{<1>} \quad x^{<2>} \quad ... \quad x^{<t>} \quad ... \quad x^{<T_x>}$$

x: Kate   Farrahi   ...   Hare   ...   learning

$$y: \quad y^{<1>} \quad y^{<2>} \quad ... \quad y^{<t>} \quad ... \quad y^{<T_y>}$$

y: 1   1   ...   1   ...   0

In this example, $T_x = T_y = 8$ but T_x and T_y can be different.

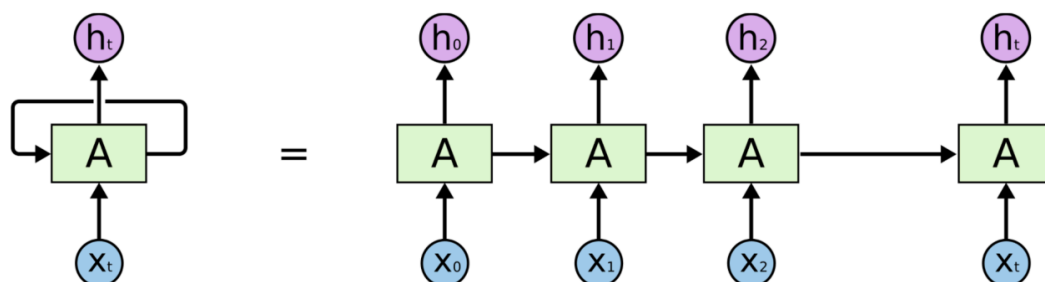# One Hot Encoding

How can we represent individual words?

Why Not a Standard Feed Forward Network?

- ▶ For a task such as "Named Entity Recognition" a feed forward network would have several disadvantages
- ▶ The inputs and outputs may have varying lengths
- ▶ The features wouldn't be shared across different positions in the network

Recurrent Neural Networks

3

- ▶ RNNs are a family of ANNs for processing sequential data
- ▶ RNNs have directed cycles in their computational graphs
- ▶ They can have complicated dynamics, difficult to train
- ▶ They are more biologically realistic

[3]Image taken from https://towardsdatascience.com

Recurrent Neural Networks

RNNs combine two properties which make them very powerful.

1. Distributed hidden state that allows them to store a lot of information about the past efficiently. This is because several different units can be active at once, allowing them to remember several things at once.

2. Non-linear dynamics that allows them to update their hidden state in complicated ways. They can however have complicated dynamics, making them difficult to train

Backpropagation Through Time (BPTT)

---
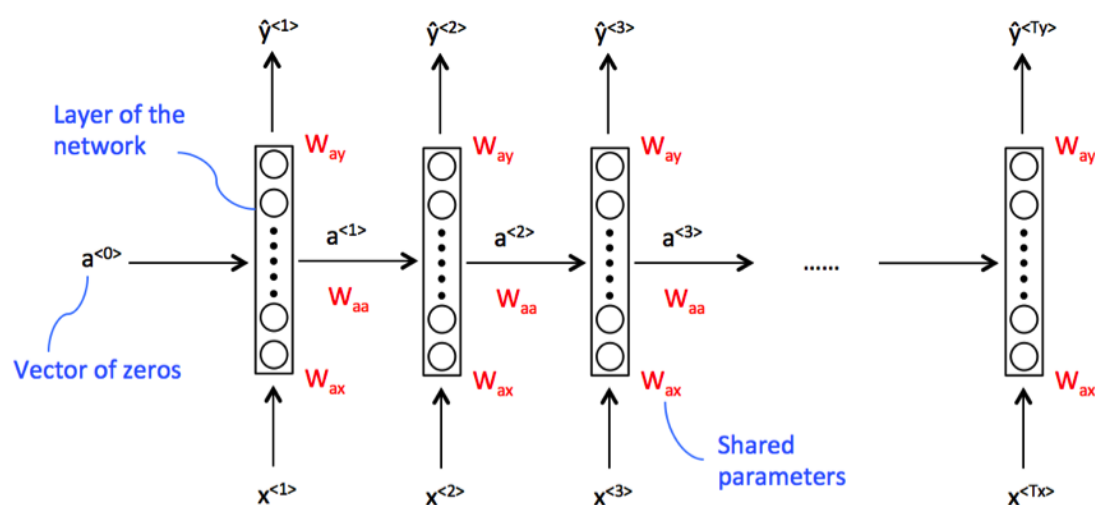
[4]Image taken from Andrew Ng

BPTT - Forward Pass

$$
\begin{aligned}
a^{<t>} &= g\left(w_{aa}a^{<t-1>} + w_{ax}x^{<t>} + b_a\right) & (1) \\
\hat{y}^{<t>} &= g\left(w_{ya}a^{<t>} + b_y\right) & (2) \\
\mathcal{L}^{<t>} &= -y^{<t>}log(\hat{y}^{<t>}) - (1 - y^{<t>})log(1 - \hat{y}^{<t>}) & (3) \\
\mathcal{L} &= \sum_{t=1}^{T_y} \mathcal{L}^{<t>} & (4)
\end{aligned}
$$

BPTT - Backwards Pass

$$
\begin{aligned}
\frac{\partial \mathcal{L}^{<3>}}{\partial w_{ya}} &= \frac{\partial \mathcal{L}^{<3>}}{\partial \hat{y}^{<3>}} \frac{\partial \hat{y}^{<3>}}{\partial w_{ya}} & (5) \\
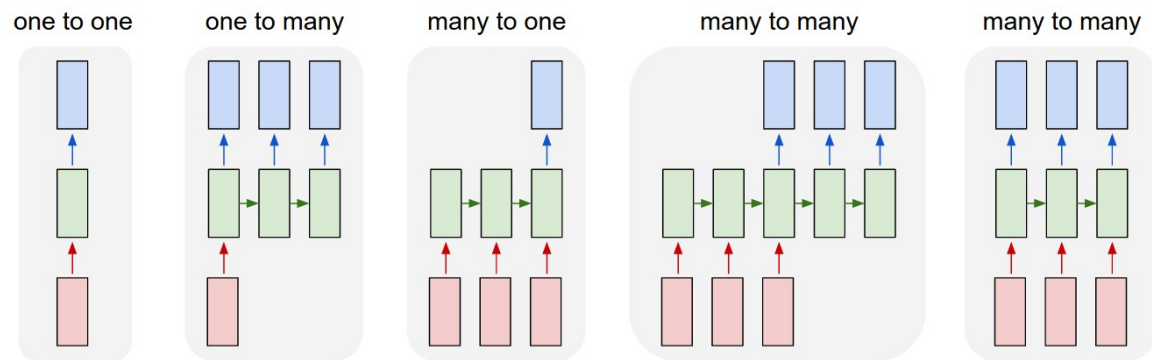\frac{\partial \mathcal{L}^{<3>}}{\partial w_{aa}} &= \frac{\partial \mathcal{L}^{<3>}}{\partial \hat{y}^{<3>}} \frac{\partial \hat{y}^{<3>}}{\partial a^{<3>}} \frac{\partial \hat{a}^{<3>}}{\partial w_{aa}} & (6) \\
& & (7)
\end{aligned}
$$

$$
\begin{aligned}
\text{Recall } a^{<3>} &= g\left(w_{aa}a^{<2>} + w_{ax}x^{<3>} + b_a\right) \\
\frac{\partial \mathcal{L}^{<3>}}{\partial w_{aa}} &= \frac{\partial \mathcal{L}^{<3>}}{\partial \hat{y}^{<3>}} \frac{\partial \hat{y}^{<3>}}{\partial a^{<3>}} \frac{\partial a^{<3>}}{\partial a^{<2>}} \frac{\partial a^{<2>}}{\partial a^{<1>}} \frac{\partial a^{<1>}}{\partial w_{aa}} & (8)
\end{aligned}
$$

Recurrent Neural Networks

one to one   one to many   many to one   many to many   many to many

5

---

[5]http://karpathy.github.io/2015/05/21/rnn-effectiveness/