Intrinsic Plasticity
and
Batch Normalisation

Nolan Peter Shaw

2019-03-15

Agenda

- Introduce Intrinsic Plasticity (IP)
- Discuss the biological and computational benefits of IP

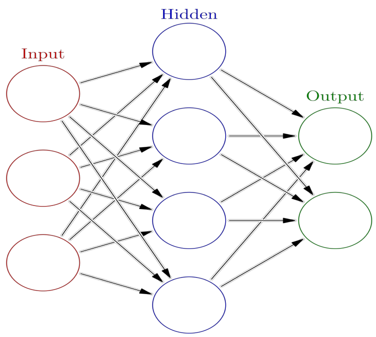
- Introduce batch normalisation (BN)
- Outline BN implementation

- Demonstrate the relation between IP and BN

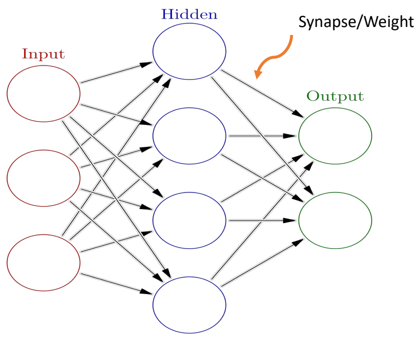
Section I: Intrinsic Plasticity

Computational Neuroscience

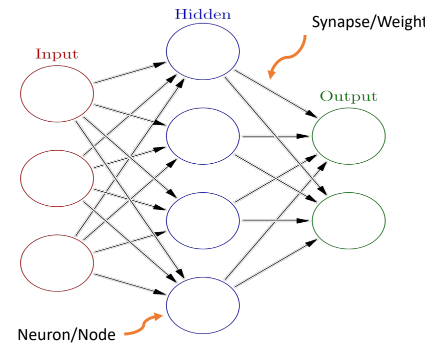
Synaptic vs Intrinsic Plasticity in the Brain



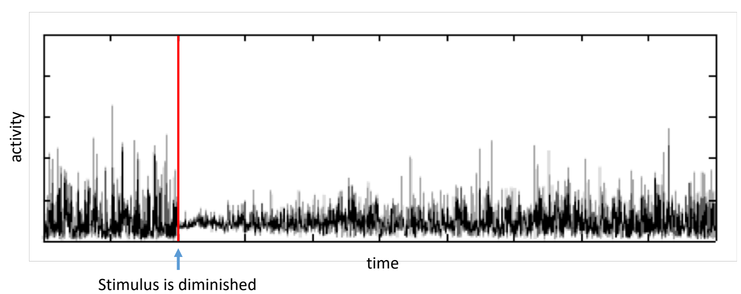
Synaptic vs Intrinsic Plasticity in the Brain



Synaptic vs Intrinsic Plasticity in the Brain

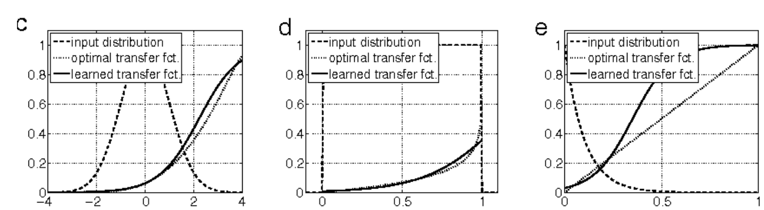


Intrinsic Plasticity



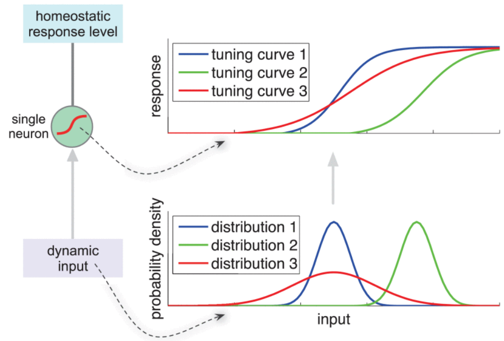
Biological Benefits

- Human brain consumes calories
- Cost of a 0/1 in an ANN is identical



Triesch, Jochen. (2005). A Gradient Rule for the Plasticity of a Neuron's Intrinsic Excitability. Artificial Neural Networks: Biological Inspirations ICANN 2005.

Computational Benefits



Bell AJ, Sejnowski TJ (November 1995). An information-maximization approach to blind separation and blind deconvolution. Neural Computation

Computational Benefits

Synaptic Plasticity

- Learn weights w.r.t. an error signal
- Minimise loss on some task

Intrinsic Plasticity

- Learn gains and biases w.r.t. local statistics
- Maximise information potential

Implementation in ANNs

Biology		**Artificial**
• Sensitivity	⟷	• Gain (Horizontal stretch)
• Threshold	⟷	• Bias (Horizontal translation)

Implementation in ANNs

Original Activation Function **Becomes**

$$y = \theta(x) \longrightarrow y = \theta(\alpha * x + k)$$

Implementation in ANNs

Original Activation Function **Becomes**

$$y = \theta(x) \longrightarrow y = \theta(\alpha * x + k)$$

Super simple (YAY!)

Implementation in ANNs

Update rules

Gain:

$$\Delta\alpha = \frac{1}{\alpha} - 2 * \mathbf{E}[\boldsymbol{xy}]$$

Bias:

$$\Delta k = -2 * \mathbf{E}[\boldsymbol{y}]$$

- Note that these update rules are still being studied and that there may be update rules that are better suited to learning

Issues

- Unstable
- $\mathbf{E}[\boldsymbol{uy}]$ may be ill-suited for adjusting the sensitivity/gain
- May homogenise inputs too much
- Competes with error-based learning of synaptic weights

Section II: Batch Normalisation

Machine Learning

The Problem

- The "shape" of inputs to a layer may be radically different from one input to the next
- This slows down learning as hidden layers are required to learn representations and distributions as well as perform computation

The Problem

- The "shape" of inputs to a layer may be radically different from one input to the next
- This slows down learning as hidden layers are required to learn representations and distributions as well as perform computation

The Solution

- Normalise all inputs w.r.t. their distribution
  - (infeasible to do for an entire dataset so treat each batch as a sample of the population)
- De-normalise w.r.t. error
  - (prevents homogeneity and preserves computational properties of neurons)

Visualising Batch Normalisation

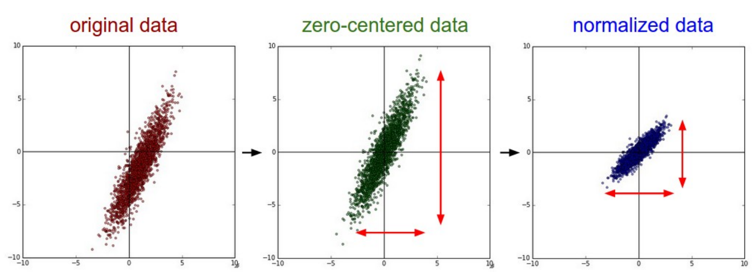


Image courtesy of: https://zaffnet.github.io/batch-normalization

Implementing Batch Normalisation

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

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

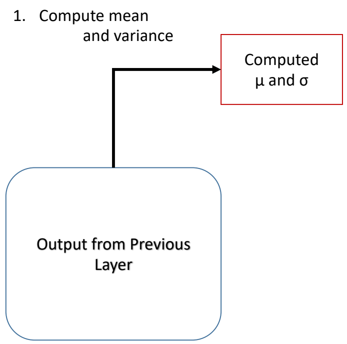
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

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

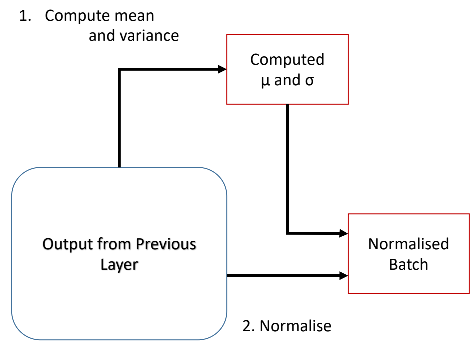
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

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

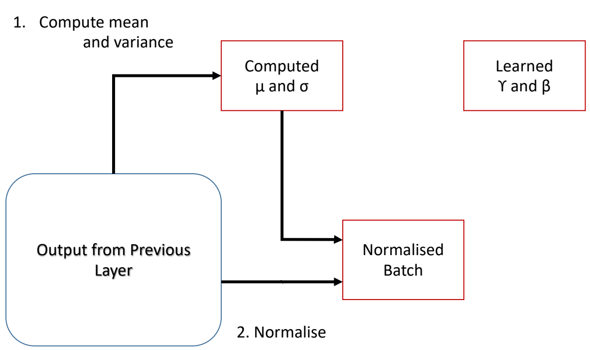
Step-by-step Walkthrough



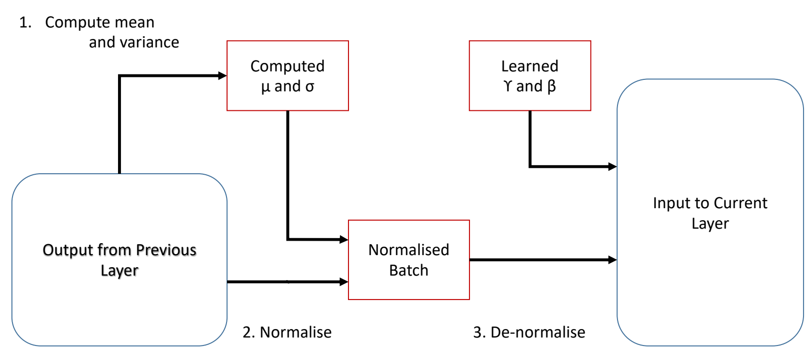
Step-by-step Walkthrough



Step-by-step Walkthrough



Step-by-step Walkthrough



Section III: Unifying IP and BN

(or: How I Stopped Caring About Big Data and Learned to Love the Brain)

Equivalence of the two models

Intrinsic Plasticity:

$$y = \theta(\alpha * x + k)$$

Equivalence of the two models

Intrinsic Plasticity:

$$y = \theta(\gamma\ * (\alpha * x + k) + \beta)$$

Equivalence of the two models

Intrinsic Plasticity:

$$y = \theta(\gamma\ * (\alpha * x + k) + \beta)$$

Batch Normalisation:

$$y = \theta(\gamma\ * \left(\frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}\right) + \beta)$$

Equivalence of the two models

Intrinsic Plasticity:

$$y = \theta(\gamma \ * (\alpha * x + k) + \beta)$$

Batch Normalisation:

$$y = \theta\left(\gamma \ * \left(\frac{1}{\sqrt{\sigma^2 + \epsilon}} * x + \frac{-\mu}{\sqrt{\sigma^2 + \epsilon}}\right) + \beta\right)$$

Equivalence of the two models

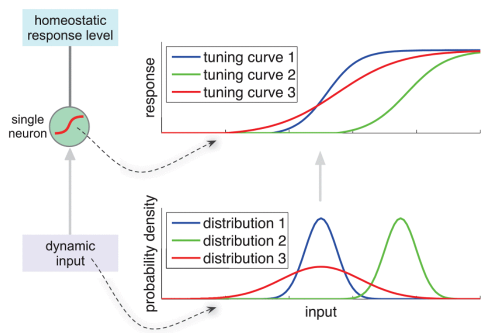
Intrinsic Plasticity:

$$y = \theta(\gamma \ * ({\color{red}\alpha} * x + {\color{green}k}) + \beta)$$

Batch Normalisation:

$$y = \theta\left(\gamma \ * \left({\color{red}\frac{1}{\sqrt{\sigma^2 + \epsilon}}} * x + {\color{green}\frac{-\mu}{\sqrt{\sigma^2 + \epsilon}}}\right) + \beta\right)$$

Relation to the Vanishing Gradient Problem



Thank you!