Carleton UNIVERSITY

Canada's Capital University

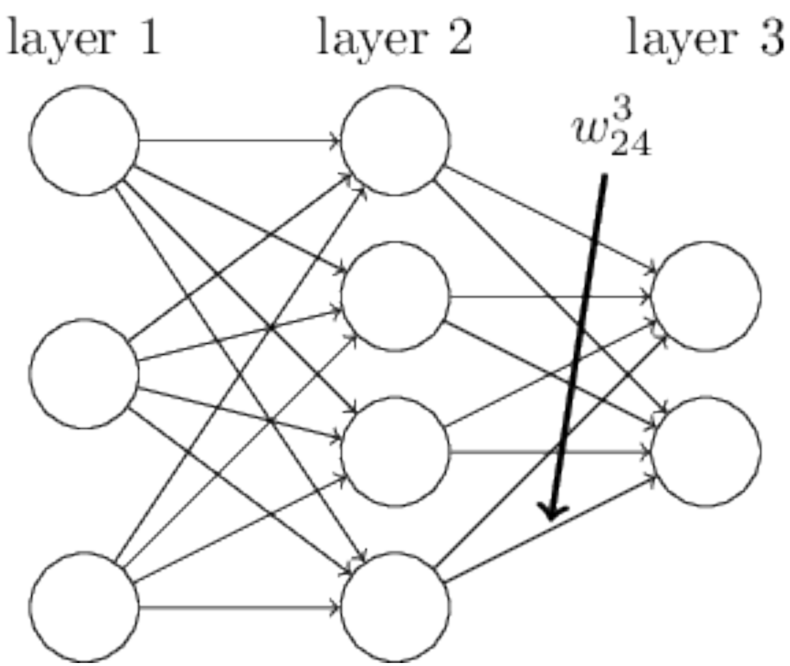
Backpropagation and Gradient Descent

Brian Carignan, Dec 5 2016

Overview

- **Notation/background**
  - Neural networks
  - Activation functions
  - Vectorization
  - Cost functions
- **Introduction**
- **Algorithm Overview**
- **Four fundamental equations**
  - Definitions (all 4) and proofs (1 and 2)
- **Example from thesis related work**

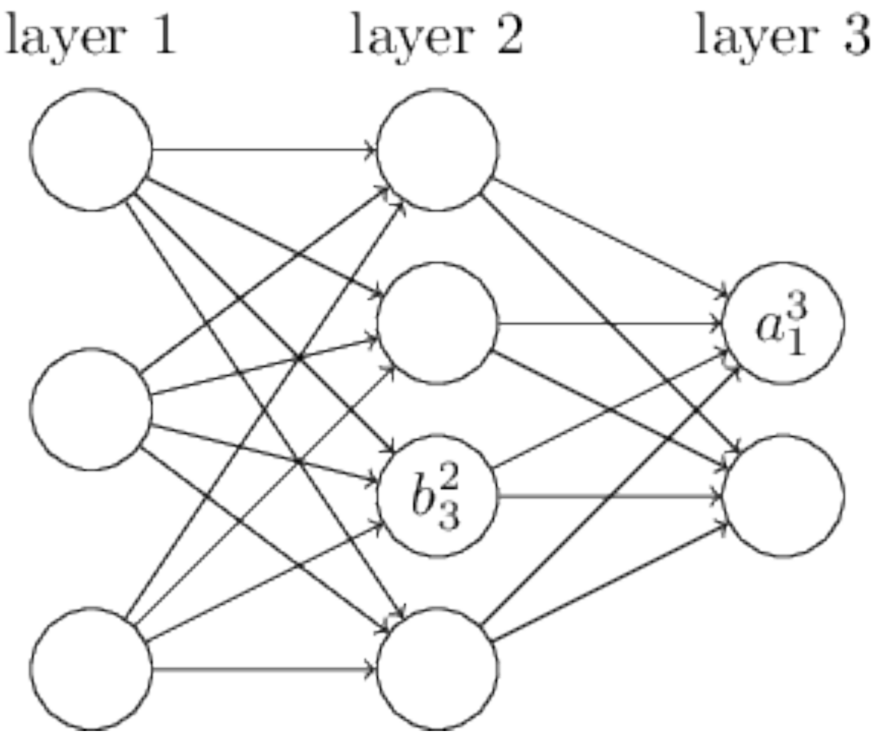
Neural Networks 1



w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

Neural Networks 2

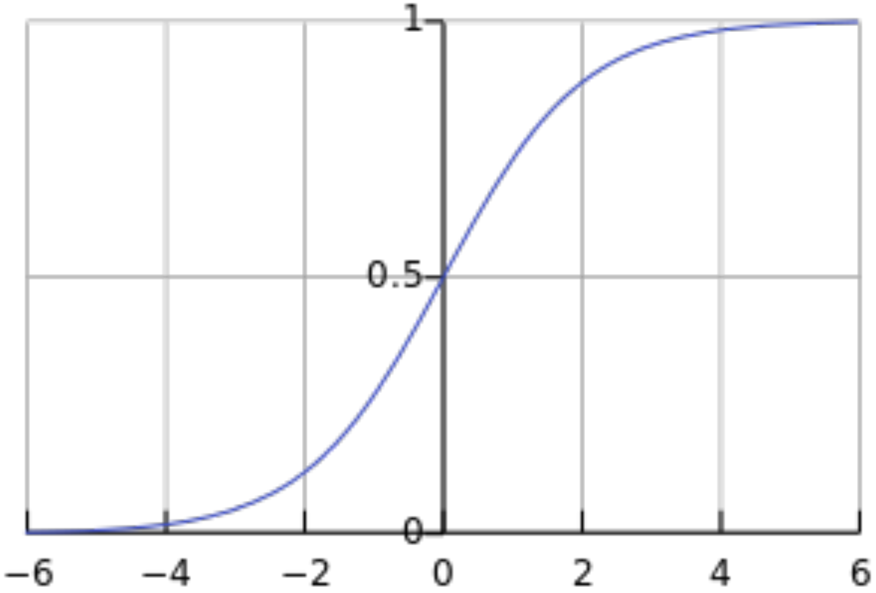
- **a – Activation of a neuron is related to the activations in the previous layer**
- **b – bias of a neuron**



$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$$

Activation Functions

- **Similar to an ON/ OFF switch**
- **Required properties**
  - Nonlinear
  - Continuously differentiable



$$f(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{d}{dx} f(x) = f(x)(1 - f(x))$$

Vectorization

- **Represent each layer as a vector**
  - Simplifies notation
  - Leads to faster computation by exploiting vector math
- **z – weighted input vector**

$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right)$$

$$a^l = \sigma(w^l a^{l-1} + b^l)$$

$$a^l = \sigma(z^l)$$

Cost Function

- **Objective Function**
- **Optimization Problem**
- **Assumptions**
  - Can average over C_x
  - Function of the outputs
- x – individual training examples (fixed)

- **Example:**

$$C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2$$

$$C = \frac{1}{n} \sum_x C_x$$

$$C = C(a^L)$$

$$C = \frac{1}{2}\|y - a^L\|^2 = \frac{1}{2} \sum_j (y_j - a^L_j)^2$$

Introduction

- **Backpropagation**
  - Backward propagation of errors
  - Calculate gradients
  - One way to train neural networks
- **Gradient Descent**
  - Optimization method
  - Finds a local minimum
  - Takes steps proportional to -gradient at current point

Algorithm Overview

1. **Input a set of training examples**

2. **For each training example x:** Set the corresponding input activation $a^{x,1}$, and perform the following steps:

   - **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^{x,l} = w^l a^{x,l-1} + b^l$ and $a^{x,l} = \sigma(z^{x,l})$.

   - **Output error $\delta^{x,L}$:** Compute the vector $\delta^{x,L} = \nabla_a C_x \odot \sigma'(z^{x,L})$.

   - **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^{x,l} = ((w^{l+1})^T \delta^{x,l+1}) \odot \sigma'(z^{x,l})$.

3. **Gradient descent:** For each $l = L, L-1, \ldots, 2$ update the weights according to the rule $w^l \rightarrow w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$, and the biases according to the rule $b^l \rightarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$.

Equation 1

- **Definition of error:** $$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$$

An equation for the error in the output layer, δ^L: The components of δ^L are given by

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L). \quad \text{(BP1)}$$

$$\delta^L = \nabla_a C \odot \sigma'(z^L). \quad \text{(BP1a)}$$

Equation 2

An equation for the error δ^l in terms of the error in the next layer, δ^{l+1}: In particular

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l), \qquad \text{(BP2)}$$

- **Key difference**
  - Transpose of weight matrix
- **Pushes error backwards**

Equation 3

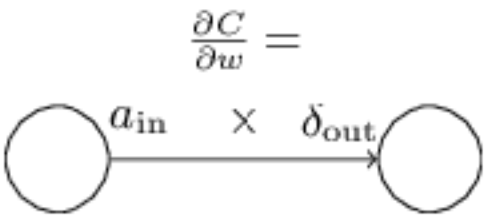
An equation for the rate of change of the cost with respect to any bias in the network: In particular:

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j. \tag{BP3}$$

- **Note that previous equations computed error**

Equation 4

An equation for the rate of change of the cost with respect to any weight in the network: In particular:



$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j. \qquad \text{(BP4)}$$

- **Describes learning rate**
- **General insights**
  - Slow learning when:
  - Input activation approaches 0
  - Output activation approaches 0 or 1 (from derivative of sigmoid)

Proof – Equation 1

- **Steps**
  1. Definition of error
  2. Chain rule
  3. k=j
  4. BP1 (components)

$$\delta_j^L = \frac{\partial C}{\partial z_j^L}$$

$$= \sum_k \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L}$$

$$= \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L}$$

$$= \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)$$

Proof – Equation 2

- **Steps**
  1. Definition of error
  2. Chain rule
  3. Substitute definition of error
  4. Derivative of weighted input vector
  5. BP2 (components)

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}$$

$$= \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l}$$

$$= \sum_k \frac{\partial z_k^{l+1}}{\partial z_j^l} \delta_k^{l+1},$$

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \sigma'(z_j^l)$$

$$\delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(z_j^l)$$

- **Recall:**

$$z_k^{l+1} = \sum_j w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_j w_{kj}^{l+1} \sigma(z_j^l) + b_k^{l+1}$$

Example – Thesis Related Work

Algorithm 1 Learning TransE

input Training set $S = \{(h, \ell, t)\}$, entities and rel. sets E and L, margin γ, embeddings dim. k.

initialize $\boldsymbol{\ell} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each $\ell \in L$

$\quad\quad \boldsymbol{\ell} \leftarrow \boldsymbol{\ell} / \|\boldsymbol{\ell}\|$ for each $\ell \in L$

$\quad\quad \mathbf{e} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each entity $e \in E$

loop

$\quad \mathbf{e} \leftarrow \mathbf{e} / \|\mathbf{e}\|$ for each entity $e \in E$

$\quad S_{batch} \leftarrow \text{sample}(S, b)$ // sample a minibatch of size b

$\quad T_{batch} \leftarrow \emptyset$ // initialize the set of pairs of triplets

$\quad$ **for** $(h, \ell, t) \in S_{batch}$ **do**

$\quad\quad (h', \ell, t') \leftarrow \text{sample}(S'_{(h,\ell,t)})$ // sample a corrupted triplet

$\quad\quad T_{batch} \leftarrow T_{batch} \cup \{((h, \ell, t), (h', \ell, t'))\}$

$\quad$ **end for**

$\quad$ Update embeddings w.r.t. $$\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla \left[\gamma + d(\boldsymbol{h} + \boldsymbol{\ell}, \boldsymbol{t}) - d(\boldsymbol{h'} + \boldsymbol{\ell}, \boldsymbol{t'})\right]_+$$

end loop

Carleton UNIVERSITY
Canada's Capital University

References

- Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, (2015)
- Bordes et al. “Translating embeddings for modeling multi-relational data”, NIPS'13, (2013)