Chapter theory 1 solution

Artem Puzanov

2016-06-04

1 Sigmoid neurons simulating perceptrons

1.1 part I

Intuition: Each neuron has a binary activation function. Multiplication of weights and biases by a positive constant is identical to multiplying an inequality by a constant. Such a multiplication does not change the inequality, therefore all perceptron values shall stay the same.

Mathematical notation:

- a_j^l - value of activation function of j-th neuron in l-th layer of our network
- C - the multiplier constant
- w_{ji} - weight of connection from i-th neuron from the $(l-1)$-th layer to the j-th neuron in the l-th layer
- b_{jl} - bias from j-th neuron in the l-th layer
- x_i - output from the i-th neuron in the $(l-1)$-th layer

$$a_j^l = \begin{cases} 1 & \text{if } \sum_i w_{ji}x_i + b_j > 0 \\ 0 & \text{if } \sum_i w_{ji}x_i + b_j \leq 0 \end{cases}$$

For each neuron, we have three possible cases:

1. $\sum_i w_{ji}x_i + b_j > 0$
2. $\sum_i w_{ij}x_i + b_j = 0$
3. $\sum_i w_{ji}x_i + b_j < 0$

Given $C > 0$, it's easy to see that multiplication by C changes none of the conditions

Now, there is a catch: x_i is actually a_i^{l-1}. However, as the choice of the l was non specific, we can apply the same logic for a_i^{l-1}. By moving backwards through $(l-1)$, $(l-2)$, $(l-3)$ and so on, we can see that no a is changed. Therefore there are no changes in the network activation values, which is what we wanted to prove.

1.2 part II

Intuition: The activation function of the sigmoid neuron limits either at 0 or at 1 when z is multiplied by a constant. Limit of the particular neuron depends on whether $wx + b > 0$ or $wx + b < 0$

Mathematical notaion

- σ - activation function of sigmoid neuron
- a_j^l - value of activation function of j-th neuron in l-th layer of our network
- C - the multiplier constant
- w_{ji} - weight of connection from i-th neuron from the $(l-1)$-th layer to the j-th neuron in the l-th layer
- b_{jl} - bias from j-th neuron in the l-th layer
- x_i - output from the i-th neuron in the $(l-1)$-th layer

$$z_j^l = \sum_i w_{ji} x_i + b_j$$

z_j^l here is a value of linear part of the sigmoid activation funciton for j-th neuron in the l-th layer. Multiplication by C gives us:

$$Cz_j^l = C \sum_i w_{ji} x_i + Cb_j$$

Let's take arbtirary neuron a_j^l.

There are three cases:

1. $z_j^l > 0$
2. $z_j^l < 0$
3. $z_j^l = 0$

Case number 1: For perceptron, if $z > 0$ than $Cz > 0$ and $a = 1$. For sigmoid neuron, if $z > 0$:

$$\lim_{C \to \infty} \sigma(Cz) = \lim_{C \to \infty} 1/(1 + e^{-Cz}) \to 1$$

Case number 2: For perceptron, if $z < 0$ than $Cz < 0$ and $a = 0$. For sigmoid neuron, if $z < 0$:

$$\lim_{C \to \infty} \sigma(Cz) = \lim_{C \to \infty} 1/(1 + e^{-Cz}) \to 0$$

Therefore if $C \to \infty$ than for all $z < 0$ or $z > 0$ sigmoid neurons emulate perceptrons. If neural network with sigmoid neurons has no neurons for which $z = 0$ than for $C \to \infty$ it fully emulates perceptron network.

Case number 3: If $z = 0$ than $a = 0$. For sigmoid neuron if $z = 0$:

$$\sigma(Cz) = 1/(1 + e^{-Cz}) = 1/(1 + e^0) = 1/2$$

Values for perceptron and sigmoid neuron differ, therefore sigmoid neurons do not emulate perceptrons

2 Bitwise representation of neural network

Assumption: It is not clear what is the specific cost function, and whether or not we can choose activation function for the ouput layer as we wish. Therefore, we shall make some assumptions

- the output layer activation function is sigmoid
- the active neurons in the output layer should have value ≥ 0.99
- the non-active neurons in the output layer should have value ≤ 0.01

Intuition: The main idea is to encode (via weights) output from 10 neurons into 4 neurons. The trick is that our encoding table is essentially our desired weight matrix from $L - 1$ layer to L layer.

Mathematical notation: First, let's choose our encoding schema. We need to represent 10 values, with 4 bits. Now $C_4^1 + C_4^2 = 10$, so by activating 1 or 2 bits, we can exactly encode the required number of states Here is the full encoding (mapping) matrix:

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

First column represents encoding of number 0, second column represents number 1 and so on, up to 9.

The activation values should form a similiar matrix, but for each 0 there should be a value of $leq0.01$, and for each 1 there should be a value of ≥ 0.99. Let's call this matrix M (mapping).

Now, all possible values of $(L - 1)$ - th layer form a diagonal matrix

$$X = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,0} \\ a_{2,1} & a_{2,2} & a2,3 & \cdots & a_{2,10} \\ a_{3,1} & 0.01 & a_{3,3} & \cdots & 0.01 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{10,1} & a_{10,2} & a_{10,3} \cdots & \cdots & a_{10,10} \end{pmatrix}$$

where $a_{i,i} >= 0.99$ and $a_{i,i \neq j} <= 0.01$

In vector form, our activation equation should be:

$$M = \sigma(WX + b)$$

where b is biases vector, W is weights matrix, and σ is sigmoid activation function.

It's easy to see that there are two distinct values in the M matrix: weight for active neuron and weight for non-active neuron. Let's call active neuron weight w^{L-1}_{active} and non-active neuron weight $w^{L-1}_{non-active}$. In the output layer, active neuron activation value shall be:

$$\sigma(w^{L-1}_{active} * a_{i,i} + 3w^{L-1}_{non-active} * a_{i,i\neq j} + b_j)$$

non-active neuron activation value shall be:

$$\sigma(4w^{L-1}_{non-active} * a_{i,i\neq j} + b_j)$$

To guarantee that active neuron activation value is ≥ 0.99 and non-active neron activation value is ≤ 0.01, we shall assume that $a_{i,i} = 0.99$ and $a_{i,j\neq i} = 0.01$

Applying that to these equations, we get

$$\begin{cases} 1/(1 + e^{w^{L-1}_{active} * 0.99 + 3w^{L-1}_{non-active} + b}) >= 0.99 \\ 1/(1 + e^{4w^{L-1}_{non-active} + b}) <= 0.01 \end{cases}$$

Any solution for this system of inqeualities shall be required weights and biases.

3 Gradient descent proof

3.1 part I

Intuition The logic behind this assertion is simple: we have to move in the (anti)direction of the gradient. Moving in the direction of the gradient means that inner product of $\Delta\nu$ and ∇C. What does Cauchy Shwartz has to do with this? Well, it tells us about upper limit of length of inner product! So all we have to do is choose such a $\Delta\nu$, as to achieve upper limit of the inner product.

Mathematical notation By Cauchy Shwartz theorem:

$$|\langle \nabla C, \Delta\nu \rangle| <= ||\nabla C|| * ||\Delta\nu||$$

Given that $||\Delta\nu|| = \epsilon$

$$|\langle \nabla C, \Delta\nu \rangle| <= ||\nabla C|| * \epsilon$$

In order to achieve equality we need to make $\Delta\nu = \epsilon\nabla C/||\nabla C||$ Let's not forget that we want to decrease function, not increase it, therefore:

$$\Delta\nu = -\epsilon\nabla C/||\nabla C||$$

That is what we wanted to prove.

A note: we could have actually solved this via Lagrangian multiplier, without usage of Cauchy-Shwartz inequality.

3.2 part II

I won't do mathematical notation here, no reason to do so.

Intuition What happens is that there is that we don't have to balance several variables under one limit of ϵ. In geometrical terms, we basically find a line tangential to the current point on the curve. The larger the smaller the angle (to our function's value), the larger steps we take. This essentialy means that we just move against the derivative of our one variable, each step size normilized by $||\nabla C||$.

4 Online learning vs batch learning

Online learning has three essential qualities: speed, equentiality, gradient estimation precision

1. speed - for each step of parameter adaptation, only one gradient calculation is made
2. sequentiality - our particular path/parameters can heavily depend on the order of observations that we feed to the algorithm.
3. gradient estimation precision - for every gradient step, we base our estimation of gradient on just 1 observation

If I'm to choose one advantage and one disadvatange, I would go for speed as an advantage, and gradient estimation precision as a disadvantage.