

Vectorized backprop

September 10, 2017

Contents

1	Notations	2
2	Loss and activation functions	2
3	Vectorized forward pass	3
4	Vectorized backward pass	3
5	Alternative vectorized backward pass	5
6	Optimization	6
6.1	Basic gradient descent	6
6.2	Initialization	6
6.3	Normalization	7
6.4	Xavier Initialization	7
6.5	RMSPProp	7
6.6	RMSPProp	7
7	Regularization	7
7.1	L2/L1	7
7.2	Dropout	8
7.3	Early stopping	8

1 Notations

1. n_x - dimensionality of input data
2. superscript $[l]$ denotes l -th layer (input layer is $[0]$, output layer - N)
3. superscript (i) denotes i -th example
4. $n^{[l]}$ - nb of neurons in the l -th layer, $n^{[0]} = n_x$
5. m - batch size
6. X - (n_x, m) -matrix of input data. each row is a single training example
7. $Z^{[l]} = (z_k^{[l](i)})_{i,k}$: dimension - $(n^{[l]}, m)$ (different examples are stacked as columns).
This is the matrix of inputs of the l -th layer. $z_k^{[l](i)}$ is the input of the k -th neuron in the l -th layer for the i -th training example
8. $A^{[l]} = (a_k^{[l](i)})_{i,k}$: dimension - $(n^{[l]}, m)$ (different examples are stacked as columns).
This the matrix of output activations of the l -th layer. $a_k^{[l](i)}$ is the output of the k -th neuron in the l -th layer for the i -th training example
9. $W^{[l]}$: dimension - $(n^{[l]}, n^{[l-1]})$ -matrix. The weights between the $l - 1$ -st and the l -th layers. $W_{(i,j)}^{[l]}$ is the weight of the arc from the j -th neuron in the $l - 1$ -st layer to the i -th neuron in the l -th layer
10. N - output layer.
11. L - loss function, $L^{(i)}$ is the loss function for i -th example
12. $J = \frac{1}{m} \sum_{i=1}^m L^{(i)}$ - batch loss function
13. \bar{L} - $(1, m)$ vector of the $L^{(i)}$ (again different example are stacked as columns)
14. $\delta^{[l]} = dZ^{[l]} = [d\bar{L}/dZ^{[l]}]^T$. Dimension - $(n^{[l]}, m)$ matrix. Partial derivative of the $L^{(i)}$'s wrt the $[l]$ -th layer inputs $Z^{[l]}$.
15. $dA^{[l]} = [d\bar{L}/dA^{[l]}]^T$ - $(n^{[l]}, m)$. Dimension - $(n^{[l]}, m)$. Partial derivative of the $L^{(i)}$'s wrt the the $[l]$ -th layer outputs $A^{[l]}$.
16. $db^{[l]} = [dJ/db^{[l]}]^T$. Dimension - $(n^{[l]}, 1)$. Partial derivative of the batch loss wrt the biases.
17. $dW^{[l]} = [dJ/dW^{[l]}]$. Dimension - $(n^{[l]}, n^{[l-1]})$. Partial derivative of the batch loss wrt the arc weights.
18. $g : R \rightarrow R$ - activation function

2 Loss and activation functions

Currently we will use the cross-entropy loss function:

$$L^{(i)} = - \sum_{k=1}^{n^{[N]}} y_k^{(i)} \log \hat{y}_k^{(i)}$$

In the binary case it simplifies to:

$$L^{(i)} = -y^{(i)} \log \hat{y}^{(i)} - (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

The total batch loss that is optimized is:

$$J = \frac{1}{m} \sum_{i=1}^m L^{(i)}$$

Sigmoid function:

1. Sigmoid:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}, \quad \sigma'(z) = \sigma(z)(1 - \sigma(z))$$

2. tanh

$$\tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}, \quad \tanh'(z) = 1 - \tanh^2(z)$$

3. ReLU

$$r(z) = \max(0, z), \quad r'(z) = I_{[0, \infty)}(z)$$

4. leaky ReLU

$$r(z) = \max(\epsilon z, z), \quad r'(z) = I_{[0, \infty)}(z) + \epsilon I_{(-\infty, 0)}(z)$$

3 Vectorized forward pass

The calculation is as follows:

Start: $X = A^{[0]}$

Recursion:

1. $Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$ in dimensions: $(n^{[l]}, m) = (n^{[l]}, n^{[l-1]})(n^{[l-1]}, m) + (n^{[l]}, m)$

where $b^{[l]}$ is broadcasted from $(n^{[l]}, 1)$ to $(n^{[l]}, m)$

2. $A^{[l]} = g(Z^{[l]})$ where g is applied component-wise

4 Vectorized backward pass

Remark: $\frac{dZ^{[l+1]}}{dA^{[l]}}$ is technically a $(n^{[l+1]} \cdot m, n^{[l]} \cdot m)$ with

$$\frac{dZ^{[l+1](k)_i}}{dA^{[l](s)_j}} = W^{[l+1]}(i, j) \delta_{ks}$$

so effectively we can keep only the diagonal $k = s$ elements and rearrange it in a $(n^{[l+1]}, n^{[l]})$ matrix:

$$\frac{dZ^{[l+1]}}{dA^{[l]}}(i, j) := W^{[l+1]}(i, j)$$

1. Derivative of the vectorized/stacked single losses \bar{L} wrt to output activation $A^{[N]}$: Dimension - $(n^{[N]}, m)$ (often $(1, m)$) For the binary cross entropy loss:

$$(d\bar{L}/dA^{[N]})_{(i,j)}^T = dL^{(j)}/dA_i^{[N](j)} = -\frac{y^{(j)}}{a^{[N](j)}} + \frac{1 - y^{(j)}}{1 - a^{[N](j)}}$$

2. Derivative of the vectorized/stacked single losses \bar{L} wrt $Z^{[l]}$: Notation - $\delta^{[l]}$: dimension - $(n^{[l]}, m)$

$$\delta^{[l]} = \left(\frac{d\bar{L}}{dZ^{[l]}} \right)^T$$

The chain rule:

$$\frac{d\bar{L}}{dZ^{[l]}} = \frac{d\bar{L}}{dZ^{[l+1]}} \frac{dZ^{[l+1]}}{dA^{[l]}} \frac{dA^{[l]}}{dZ^{[l]}}$$

and after transposing and reducing the last multiplication with a diagonal matrix to component-wise multiplication with a broadcasted vector we get

$$\left(\frac{d\bar{L}}{dZ^{[l]}} \right)^T = \delta^{[l]} = (W^{[l+1]})^T \cdot \delta^{[l+1]} \odot g'(Z^{[l]})$$

with dimensions

$$(n^{[l]}, m) = (n^{[l]}, n^{[l+1]}) \cdot (n^{[l+1]}, m) \odot (n^{[l]}, m)$$

3. Derivative of the vectorized/stacked single losses \bar{L} wrt $A^{[l]}$: Notation - $dA^{[l]}$: dimension - $(n^{[l]}, m)$

$$dA^{[l]} = \left(\frac{d\bar{L}}{dA^{[l]}} \right)^T$$

The chain rule:

$$\frac{d\bar{L}}{dZ^{[l]}} = \frac{d\bar{L}}{dA^{[l]}} \frac{dA^{[l]}}{dZ^{[l]}}$$

and after transposing and reducing the last multiplication with a diagonal matrix to component-wise multiplication with a broadcasted vector we get

$$\delta^{[l]} = \left(\frac{d\bar{L}}{dA^{[l]}} \right)^T \odot g'(Z^{[l]}) = dA^{[l]} \odot g'(Z^{[l]})$$

with dimensions

$$(n^{[l]}, m) = (n^{[l]}, m) \odot (n^{[l]}, m)$$

4. Derivative of the batch loss J wrt $b^{[l]}$: Notation - $db^{[l]}$: dimension - $(n^{[l]}, 1)$

$$db^{[l]} = \left(\frac{dJ}{db^{[l]}} \right)^T$$

The chain rule:

$$\frac{dJ}{db^{[l]}} = \frac{dJ}{dZ^{[l]}} \frac{dZ^{[l]}}{db^{[l]}}$$

and after transposing and noting that the last term is the identity matrix we get:

$$db^{[l]} = \frac{1}{m} \sum_{i=1}^m \delta^{[l](i)} = \frac{1}{m} \text{np.sum}(\delta^{[l]}, \text{axis}=1, \text{keepdim}=\text{True})$$

with dimensions

$$(n^{[l]}, 1) = (n^{[l]}, 1) + (n^{[l]}, 1) + \dots$$

5. Derivative of the batch loss J wrt $W^{[l]}$: Notation - $dW^{[l]}$: Dimension - $(n^{[l]}, n^{[l-1]})$

$$dW^{[l]} = \frac{dJ}{dW^{[l]}}$$

The chain rule:

$$\frac{dJ}{dW^{[l]}} = \frac{dJ}{dZ^{[l]}} \frac{dZ^{[l]}}{dW^{[l]}}$$

and hence:

$$\begin{aligned} (dW^{[l]})_{(i,j)} &= \frac{1}{m} \sum_{k=1}^m \sum_{s=1}^{n^{[l]}} s = 1^{n^{[l]}} \frac{dL^{(k)}}{dZ_s^{[l](k)}} \frac{dZ_s^{[l](k)}}{dW_{(i,j)}^{[l]}} \\ &= \frac{1}{m} \sum_{k=1}^m \sum_{s=1}^{n^{[l]}} \frac{dL^{(k)}}{dZ_s^{[l](k)}} \delta_{s,i} a_j^{[l-1](k)} \\ &= \frac{1}{m} \sum_{k=1}^m \delta_i^{[l](k)} a_j^{[l-1](k)} = \frac{1}{m} (\delta^{[l]} (A^{[l-1]})^T)_{i,j} \end{aligned}$$

that is

$$dW^{[l]} = \frac{1}{m} \delta^{[l]} (A^{[l-1]})^T$$

with dimensions

$$(n^{[l]}, n^{[l-1]}) = (n^{[l]}, m) \cdot (m, n^{[l-1]})$$

backprop:

1. Compute $dA^{[N]}$ - transposed derivative of stacked loss wrt output layer. Dimension - $(n^{[N]}, m)$
2. Compute $dZ^{[N]} = dA^{[N]} \odot g'(Z^{[N]})$ - transposed derivative of stacked loss wrt input of the last layer. Dimension - $(n^{[N]}, m)$
3. Recursion1: $dZ^{[l]} = (W^{[l+1]})^T dZ^{[l+1]} \odot g'(Z^{[l]})$. Dimension - $(n^{[l]}, m)$
4. Produce layer $[l]$ derivatives wrt to the batch loss J :
5. $db^{[l]} = \frac{1}{m} \text{np.sum}(dZ^{[l]}, \text{axis}=1, \text{keepdim}=\text{True})$. Dimension - $(n^{[l]}, 1)$
6. $dW^{[l]} = \frac{1}{m} dZ^{[l]} (A^{[l-1]})^T$. Dimension - $(n^{[l]}, m) \cdot (m, n^{[l-1]}) = (n^{[l]}, n^{[l-1]})$

5 Alternative vectorized backward pass

This is based on the following slightly different but equivalent recursive calculation: 1A. As before we have $dZ^{[l]} = dA^{[l]} \odot g'(Z^{[l]})$

2A. Derivative of the vectorized/stacked single losses \bar{L} wrt $A^{[l]}$: Notation - $dA^{[l]}$: dimension - $(n^{[l]}, m)$

$$dA^{[l]} = \left(\frac{d\bar{L}}{dA^{[l]}} \right)^T$$

The chain rule:

$$\frac{d\bar{L}}{dA^{[l]}} = \frac{d\bar{L}}{dZ^{[l+1]}} \frac{dZ^{[l+1]}}{dA^{[l]}}$$

and after transposing and recalling the remark in the beginning of the previous section stating that

$$\frac{dZ^{[l+1]}}{dA^{[l]}}(i, j) := W^{[l]}(i, j)$$

we get

$$dA^{[l]} = ((W^{[l+1]})^T \cdot dZ^{[l+1]})$$

with dimensions

$$(n^{[l]}, m) = (n^{[l]}, n^{[l+1]}) \cdot (n^{[l+1]}, m)$$

3A. Finally the expressions for $dW^{[l]}$ and $db^{[l]}$ remain as in the previous section

The corresponding backprop algorithm is:

backprop A:

1. Compute $dA^{[N]}$ - transposed derivative of stacked loss wrt output layer. Dimension - $(n^{[N]}, m)$
2. Compute $(dZ^{[N]})^T = dA^{[N]} \odot g'(Z^{[N]})$ - transposed derivative of stacked loss wrt input of the last layer. Dimension - $(n^{[N]}, m)$
3. Backwards recursion
 - (a) $dA^{[l]} = ((W^{[l+1]})^T \cdot dZ^{[l+1]})$. Dimension - $(n^{[l]}, m)$
 - (b) $dZ^{[l]} = dA^{[l]} \odot g'(Z^{[l]})$. Dimension - $(n^{[l]}, m)$
4. Produce layer $[l]$ derivatives wrt to the batch loss J :
 - (a) $db^{[l]} = \frac{1}{m} \text{np.sum}(dZ^{[l]}, \text{axis}=1, \text{keepdim}=\text{True})$. Dimension - $(n^{[l]}, 1)$
 - (b) $dW^{[l]} = \frac{1}{m} dZ^{[l]} (A^{[l-1]})^T$. Dimension - $(n^{[l]}, m) \cdot (m, n^{[l-1]}) = (n^{[l]}, n^{[l-1]})$

6 Optimization

6.1 Basic gradient descent

The learning rate α . The update of weights:

$$\begin{aligned} W^{[l]} &:= W^{[l]} - \alpha dW^{[l]} \\ b^{[l]} &:= b^{[l]} - \alpha db^{[l]} \end{aligned}$$

6.2 Initialization

Initialization helps speed up the learning process and avoid vanishing/exploding gradients

6.3 Normalization

To speed-up learning normalize the feature matrix X :

$$\begin{aligned}\mu &:= \frac{1}{m} \sum_{i=1}^m X^{(i)} = \frac{1}{m} \text{np.sum}(X, \text{axis}=1, \text{keepdim}=\text{True}) \\ X &:= X - \mu \\ \sigma^2 &:= \frac{1}{m} \sum_{i=1}^m X^{(i)} \odot X^{(i)} = \frac{1}{m} \text{np.sum}(X \odot X, \text{axis}=1, \text{keepdim}=\text{True}) \\ X &:= X / \sigma^2\end{aligned}$$

6.4 Xavier Initialization

To avoid vanishing/exploding gradients one should initialize the weights should be initialized in such a manner that the fan-in into a neuron should have variance 1. The fan-in into a neuron in the l -th layer has $n^{[l-1]}$ components. Therefore the Xavier initialization is (for tanh or sigmoid activations)

$$W^{[l]} = \text{np.rand.randn}((n^{[l]}, n^{[l-1]})) * 1 / \text{np.sqrt}(n^{[l-1]})$$

For ReLU layers it is better to use

$$W^{[l]} = \text{np.rand.randn}((n^{[l]}, n^{[l-1]})) * \text{np.sqrt}(2/n^{[l-1]})$$

The Bengio et al. initialization

$$W^{[l]} = \text{np.rand.randn}((n^{[l]}, n^{[l-1]})) * \text{np.sqrt}(2/n^{[l-1]} + n^{[l]})$$

6.5 RMSProp

6.6 RMSProp

7 Regularization

7.1 L2/L1

The total batch loss without regularization is:

$$J = \frac{1}{m} \sum_{i=1}^m L^{(i)}$$

For the L2 regularization the objective function is modified as follows:

$$J^{L2} = J + \frac{\lambda}{2m} \sum_{k=1}^L \|W^{[k]}\|_F^2 = \frac{1}{m} \sum_{i=1}^m L^{(i)} + \frac{\lambda}{2m} \sum_{k=1}^L \|W^{[k]}\|_F^2$$

where the squared Frobenius matrix norm is

$$\|W^{[k]}\|_F^2 = \sum_{i=1}^{n^{[k-1]}} \sum_{j=1}^{n^{[k]}} (W_{ij}^{[k]})^2$$

The gradient is then:

$$dW^{[l]} := dJ^{L2}/dW^{[l]} = dJ/dW^{[l]} + \frac{\lambda}{m}W^{[l]}$$

and the update:

$$W^{[l]} := W^{[l]} - \alpha dW^{[l]} = (1 - \frac{\alpha\lambda}{m})dW^{[l]} - \alpha dJ/dW^{[l]}$$

which is a minor modification of the basic un-regularized backprop update

7.2 Dropout

7.3 Early stopping