Convolutional and residual neural networks

An overview

Simon Jacquet

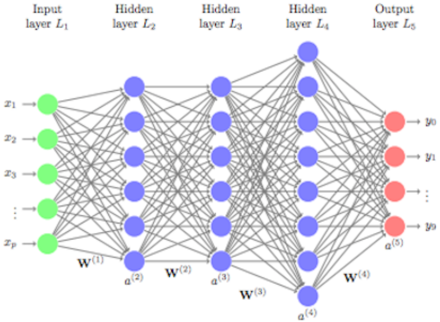
Faculty of Computer Science
Unamur

March 1st 2022

Content

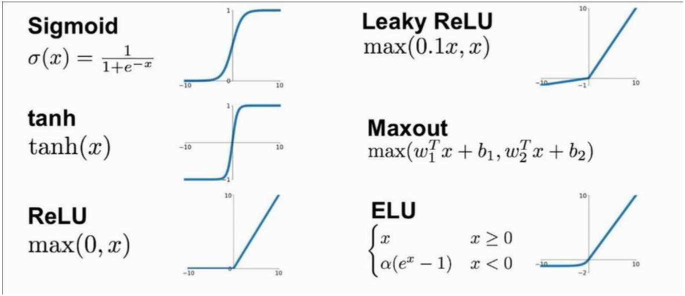
- Neural networks
  - Structure
  - Activation functions
  - Loss function
- Convolutional neural network (CNN or ConvNet)
  - Convolution
  - Pooling
  - Fully connected layers
  - Dropout
- Residual neural network (ResNet)
  - ResNet 18
  - Basic block
  - Batch normalization

Neural networks



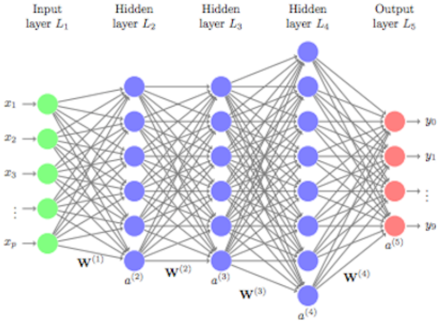
- Where $a^{(l)} = \sigma^{(l)}(\boldsymbol{W}^{(l-1)} a^{(l-1)} + b^{(l-1)})$, with:
  - l, the layer number
  - $\sigma^{(}l)$, the activation function at layer l
  - $\boldsymbol{W}^{(}l)$, the weights between layers l and $l+1$
  - $b^{(l)}$, the bias

Neural networks: activation function



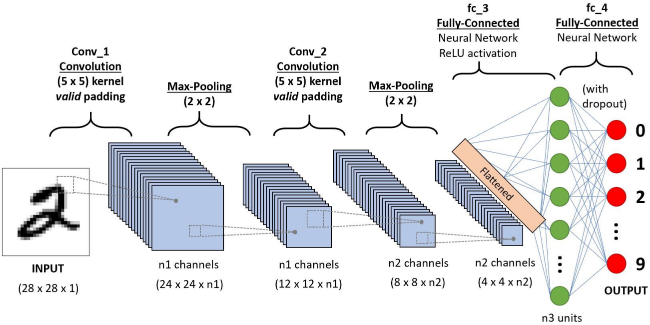
- There are many activation functions
- The two most common are the sigmoid and ReLU (Rectified Linear Unit)
- In deep learning, the sigmoid is typically used in the last layer and ReLU everywhere else

Neural networks: loss function



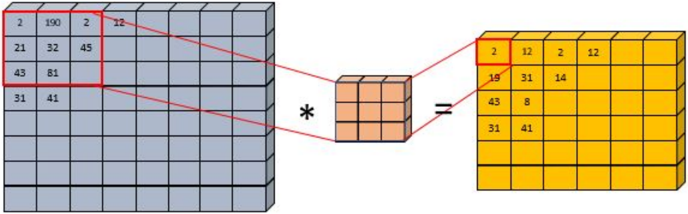
- Loss function: $L_{CE}(x) = -\sum_{i=1}^{m} t_i \log(y_i)$ with
  - y the predicted output of x,
  - t the true output of x
  - CE for cross entropy

Convolutional neural networks (CNNs or ConvNets)



- New concepts:
  - Convolution
  - Pooling
  - Fully connected layers
  - Dropout (for better training)

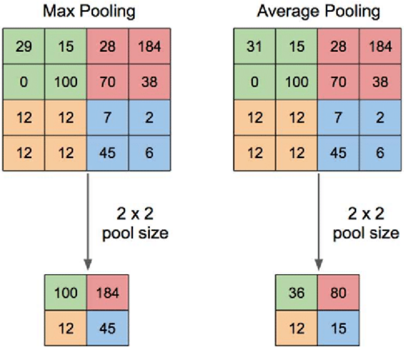
ConvNets: Convolution



- $x * k = y$, with
  - $*$, the convolution symbol,
  - x, the input image,
  - k, the kernel,
  - y the output image
- $y[i,j] = \sum_{i'=0, j'=0}^{i'<k_1, j'<k_2} x[i+i', j+j']\ k[i', j']$
- Padding:
  - VALID: y keeps the size of x, with added zeros
  - SAME: y has a smaller size than x

UNIVERSITÉ DE NAMUR

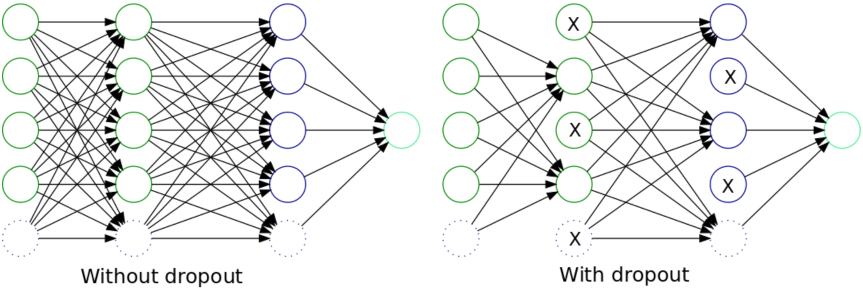
ConvNets: Pooling



- Reduce the dimensions of the feature maps [1]
- Summarises the features present in a region of the feature map generated by a convolution layer [1]

[1] https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

ConvNets: Dropout



- Parameter: d, the drop rate
- $\hat{a}_i^{(l)} = \begin{cases} 0, & \text{with probability } d \\ a_i^{(l)}, & \text{otherwise} \end{cases}$
- Happens only during training
- Reduce overfitting by preventing complex co-adaptations on training data.

ConvNets for text: embedding

Raw data	John likes to watch movies.
Preprocessed data	['john', 'likes', 'to', 'watch', 'movies']
Vocabulary *V*	{'bread', 'dislike', 'john', 'likes', 'mary', 'movies', 'watch', 'to', 'too'}

Feature vector *x*

'john'	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
'likes'	0	0	0	1	0	0	0	0	0
'to'	0	0	0	0	0	0	0	1	0
'watch'	0	0	0	0	0	0	1	0	0
'movies'	0	0	0	0	0	1	0	0	0

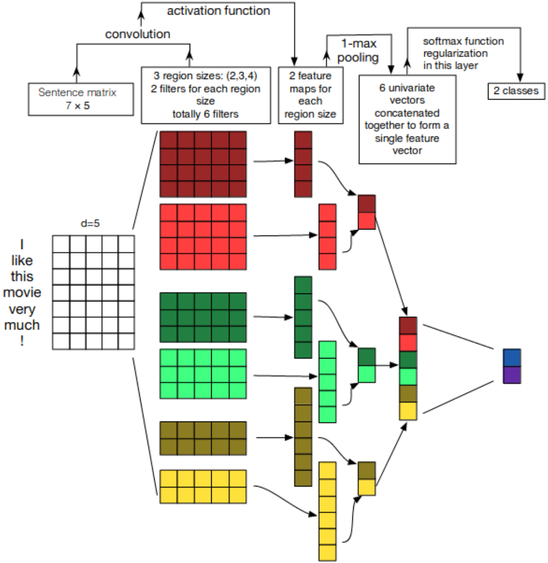
x after padding (*L=6*)

'john'	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
'likes'	0	0	0	1	0	0	0	0	0
'to'	0	0	0	0	0	0	0	1	0
'watch'	0	0	0	0	0	0	1	0	0
'movies'	0	0	0	0	0	1	0	0	0
''	0	0	0	0	0	0	0	0	0

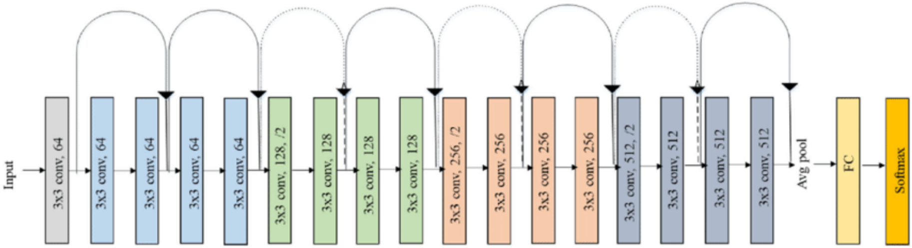
x after embedding (*d=2*)

'john'	**0.8**	**0.7**
'likes'	0.9	-0.1
'to'	0.1	0.2
'watch'	-0.6	-0.5
'movies'	-0.7	-0.3
''	0	0

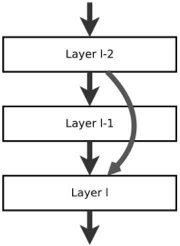
ConvNets for text: example of architecture



ResNet 18



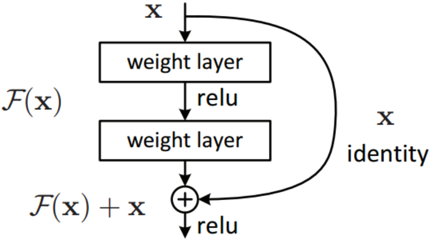
- Utilize skip connections
- Typically, two or three layer skips
- Uses batch normalization
  - ☞ method used to make artificial neural networks faster and more stable



ResNet 18: Details

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

ResNet: Basic block



- Reasons:
  - Avoid the problem of vanishing gradients
    - ☞ Gradients are proportional to the partial derivative wrt. the current weight and can quickly vanish
    - ☞ Residual neural networks provide 'highways' for the gradient
  - Address the Degradation problem
    - ☞ with increasing network depth, accuracy gets saturated and then degrades quickly

ResNet: Batch normalization

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

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

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

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

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

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

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

- Method: normalization of the layers' inputs by re-centering and re-scaling
- Goal: make artificial neural networks faster and more stable