

Week 2 Research

Maria Shevchuk

Everything CNN's

- Network structure
- Sigmoid Activation function
- CNN vs MLP
- Cost function
 - Network accuracy
 - Loss function
- Padding and step size

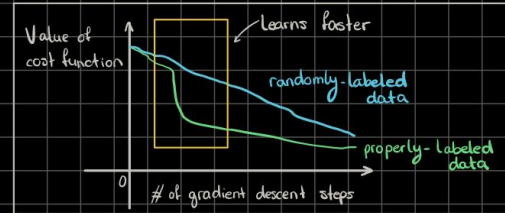
Cost function - average loss

compares expected activations with output

$$\text{Cost of } \boxed{3} = \begin{cases} (n_0 - 0.00)^2 \\ (n_1 - 0.00)^2 \\ (n_2 - 0.00)^2 \\ (n_3 - 1.00)^2 \\ (n_4 - 0.00)^2 \\ \vdots \\ (n_9 - 0.00)^2 \end{cases}$$

the more correct the output
→ the smaller the cost value!

the average cost (of all training data)
represents network accuracy



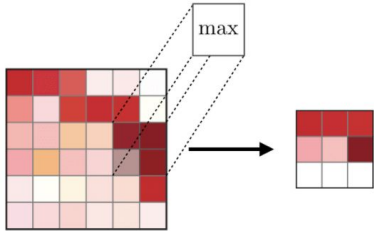
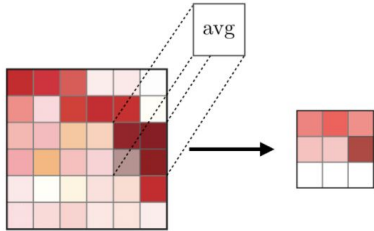
Loss function

- per single training example

$$E_{\text{total}} = \sum \frac{1}{2} (\text{target} - \text{output})^2$$

Pooling Layers

- Average pooling
- Max pooling
- Global pooling
- Keras

Type	Max pooling	Average pooling
Purpose	Each pooling operation selects the maximum value of the current view	Each pooling operation averages the values of the current view
Illustration		
Comments	<ul style="list-style-type: none"> • Preserves detected features • Most commonly used 	<ul style="list-style-type: none"> • Downsamples feature map • Used in LeNet

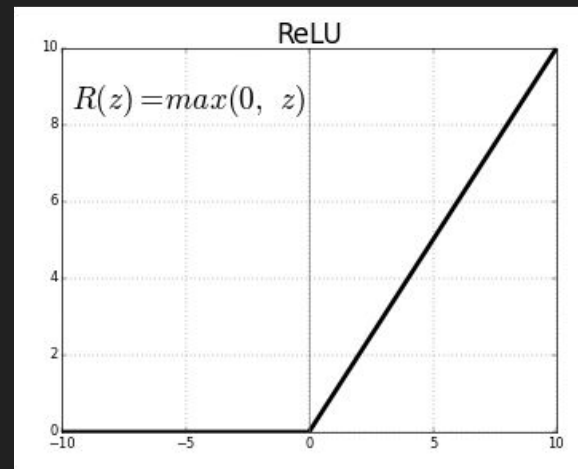
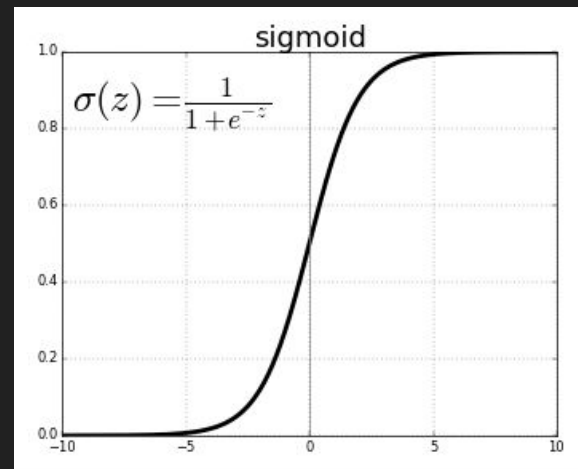
Pooling layer

Output dimentions:

$$\begin{array}{c}
 \text{feat. map height} \uparrow \\
 \frac{(n_h - f + 1)}{s} \times \frac{(n_w - f + 1)}{s} \times n_c \\
 \downarrow \quad \quad \quad \downarrow \quad \quad \quad \uparrow \\
 \text{stride length} \quad \quad \text{feat map width} \quad \quad \text{\# channels in feat. map}
 \end{array}$$

Activation functions

- Sigmoid
 - Binary output
 - Tend to vanish gradient
- ReLU
 - Simple math
 - Networks with Relu tend to show better convergence performance than sigmoid. ([Krizhevsky et al.](#))
 - Tend to blow up activation



$$a_{l+1} = \sigma(w_l a_l + b_l)$$

Sigmoid function $\sigma(x) = \frac{1}{1 + e^{-x}}$

$$a_o^{(i)} = \sigma(w_{0,0} a_o^{(o)} + w_{0,1} a_1^{(o)} + \dots + w_{0,n} a_n^{(o)} + b_o)$$
$$\sigma \left(\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_o^{(o)} \\ a_1^{(o)} \\ \vdots \\ a_n^{(o)} \end{bmatrix} + \begin{bmatrix} b_o \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \right)$$