Neural networks' architectures

$\vec{x}$ - input , y - output

$f(\vec{x}) \approx y$

Ex. 1. Regression $y \in \mathbb{R}$ $\quad f(\vec{x}) \in \mathbb{R}$

$$(f(\vec{x}) - y)^2 \to \min$$

2. Classification $y \in \{1, \dots, c\}$, $f(\vec{x}) \in \mathbb{R}^c$

$\vec{p} = f(\vec{x})$, $\quad \vec{p} = \{p_i\}_{i=1}^{c}$, $\quad p_i \geq 0 \quad \sum p_i = 1$

$$y = c \to \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} = \vec{y}$$

$- \sum y_i \log p_i \to \min$ - cross-entropy

$f(\vec{x}) \in \mathcal{F}$

$$\mathcal{F} = \{ f(\vec{x}; \vec{\theta}) = f_{\vec{\theta}}(\vec{x}) = f(\vec{x} \mid \vec{\theta}), \vec{\theta} \in \Theta \subseteq \mathbb{R}^p \}$$

$f_{\vec{\theta}}(\vec{x}) = \vec{\theta}^T \cdot \vec{x}$ $\quad$ I layer, $\{\vec{\theta}_i\}_{i=1}^{c}$, $c \cdot d$ parameters

logit

$f_{\vec{\theta}}(\vec{x}) = \text{softmax}(\{\vec{\theta}_i^T \vec{x}\})$, $\quad i = 1 \dots c$

$$\vec{p} = \text{softmax}(\vec{\ell}) = \left\{ \frac{\exp(\ell_i)}{\sum_{j=1}^{c} \exp(\ell_j)} \right\}$$

II layer, nonlinear layer, no parameters

$\vec{\theta}_1^T \vec{x} < \vec{\theta}_2^T \vec{x} \qquad 0 < (\vec{\theta}_2 - \vec{\theta}_1)^T \vec{x}$

hyperplane

Multilayer (fully-connected) neural network

$$\sigma(t) = \frac{1}{1+\exp(-t)}$$

Ex. 2 layer neural network

I layer $\quad \{(\vec{\theta}_i^1)^T \vec{x}\} \to \Theta_1 \vec{x} \to \vec{\ell}_1 \to \sigma(\vec{\ell}_1) \to \vec{z}_1$

II layer $\quad \{(\vec{\theta}_i^2)^T \vec{z}_1\} \to \Theta_2 \vec{z}_1 \to \vec{\ell}_2 \to \sigma(\vec{\ell}_2) \to \vec{z}_2$

$\vdots$

k layer $\quad \{(\vec{\theta}_i^k)^T \vec{z}_{k-1}\} = \Theta_k \vec{z}_{k-1} \to \vec{\ell}_k \to \sigma(\vec{\ell}_k) \to \vec{z}_k$

softmax $(\vec{\ell}_k) \to \vec{p}$

[θ - theta $\qquad t \quad T$

Θ - capital theta $\qquad T \quad \top$

$L(\Theta) = \frac{1}{m} \sum_{i=1}^{m} CE(\vec{p}_i, y_i) \to \min_{\Theta}$, $\Theta = \{\Theta_1, \dots, \Theta_k\}$

$\frac{\partial L(\Theta)}{\partial \Theta}$ → apply a variant of gradient descent

Backpropagation to calculate derivatives

f - Big enough? - optimization - good or not?

Universal approximation theorem

$k \geq 2$ - $|g(\vec{x}) - f_{NN}(\vec{x})| < \varepsilon \quad \forall \vec{x} \in X \subset \mathbb{R}^d$

$d_1 = \dim(\vec{z}_1)$ $\quad \Theta_1 \in \mathbb{R}^{d_1 \times d}$

$\vec{z}_k$ - a representation of an object at layer k

$\vec{z}_0 = \vec{x}$

$\mathrm{softmax}(\Theta_2 \, \sigma(\Theta_1 \vec{x}))$

$d_0 = d$

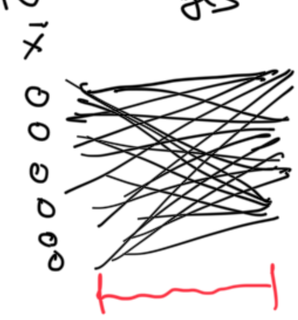
$p = \sum_{i=1}^{k} d_i d_{i-1} = k \cdot d^2$, m - a sample size

$\vec{x}$ - an image $32 \times 32 = 1024 = d_0$

$d_1 \approx d_0 \quad d_1 = d_0 \quad .. \quad d_k = d_{k-1}$

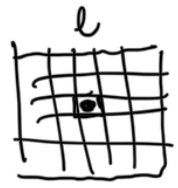
$\boxed{p \leq m}$ ImageNet 10^7 images

$10\,p < m$



Convolutional Neural Networks

→ $\vec{z}_k \xrightarrow{FC} \vec{p}$



Filter

$z_{ij} = \sum_c \sum_{k,l=-2}^{2} \underbrace{x^c_{(i+k)(j+l)}}_{x_{ij}} \left[\Theta^{ij}_{klc}\right]$

$k = l = 0 \qquad l^2$

$\underbrace{l^2 \cdot hw} \qquad \underbrace{l^2 \cdot d} < d^2$

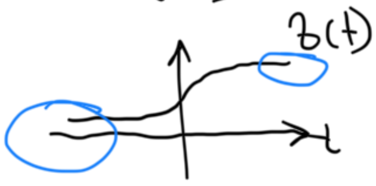
$\underbrace{l^2}$

$z_{ij} = \sum x^c_{(i+k)(j+l)} \cdot \Theta_{klc}$

$- k \cdot l^2$

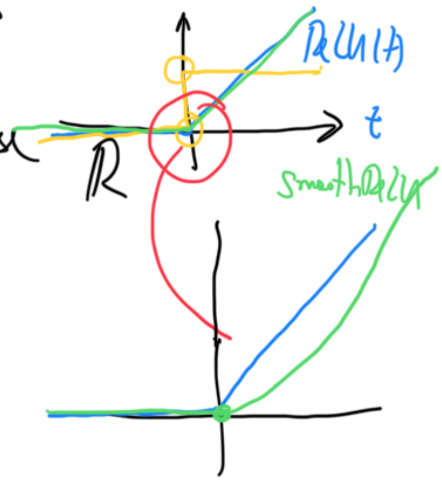
Ex. $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ $z_{ij} = x_{ij}$ $\quad \begin{bmatrix} 1/9 & 1/9 & \dots \\ \dots & \dots & \dots \\ \dots & \dots & 1/9 \end{bmatrix}$ $z_{ij} = \sum x_{(i+k)(j+l)} \cdot \frac{1}{9}$

$\sigma(t)$



ReLU(t) - rectified linear unit

$$\mathrm{ReLU}(t) = \begin{cases} t, & t \geq 0 \\ 0, & \text{otherwise} \end{cases}$$



$k = 50$

$$\vec{z}_1 = \mathrm{ReLU}(F(\vec{x}))$$

$$\vec{z}_1 = \mathrm{ReLU}(F(\vec{x}) + \vec{x})$$

1) skip connection

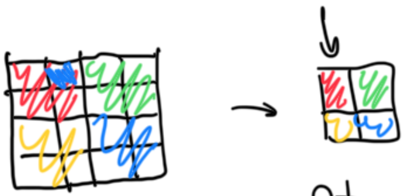
2014 AlexNet $\dfrac{8-10}{50}$

$$\frac{\partial}{\partial \theta} g_3(g_2(g_1(\vec{x}, \theta))) = \frac{\partial g_3}{\partial g_2} \cdot \frac{\partial g_2}{\partial g_1} \cdot \frac{\partial g_1}{\partial \theta}$$

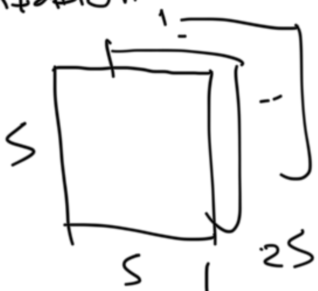
2) Pooling layers

dropout ~ Bayesian regularization

t



$\dfrac{\partial t}{\partial \theta}$



$625 \xrightarrow{FC} \xrightarrow{FC} \vec{p}$