model w/ param $\vec{w}$ fit to data D

$$\underbrace{P(\vec{w}|D)}_{\text{posterior}} = \frac{P(D|\vec{w})}{P(D)} \underbrace{P(\vec{w})}_{\text{prior}}$$

MAP: find $\vec{w}^*$ that maximizes $P(\vec{w}|D)$

Bayesian: find $P(\vec{w}|D) \Rightarrow$ estimate param. uncertainty

example: problem #2 on PS #1 (hinczlab.org/phys414)

machine learning classification

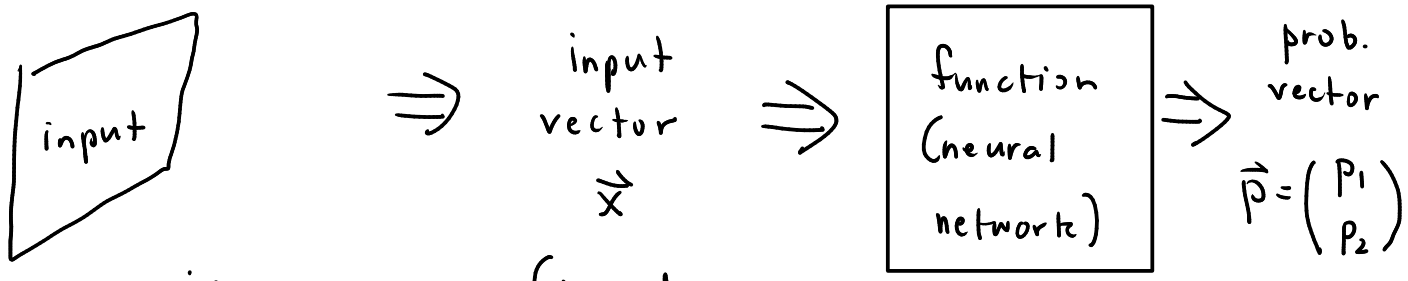


image (i.e. $100 \times 100 \times 3$)

(i.e. dim $= 30{,}000$)

p_i = prob. that input is category i

2 categories (i.e. cat or dog)

neural network: dence neural net (DNN)

or multilayer perceptron

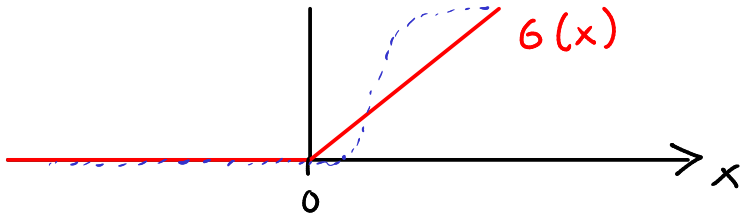
Series of operations ("layers") (transf. of input from previous oper.)

nonlinear "activation" func.

first layer: $\sigma\left(A^{(1)}\vec{x} + \vec{b}^{(1)}\right) = \vec{z}^{(1)}$ output of first layer

($A^{(1)}$: matrix (could be rectang.); $\vec{b}^{(1)}$: vector)

example of $\sigma(x) = \max(0, x)$



$$\sigma(\vec{x}) \equiv \left(\sigma(x_1), \sigma(x_2), \ldots\right)$$

next layer: $\vec{z}^{(2)} = \sigma\left(A^{(2)}\vec{z}^{(1)} + \vec{b}^{(2)}\right)$

. . .

$$\vec{z}^{(n)} = \sigma\left(\underbrace{A^{(n)}}_{2 \times M}\vec{z}^{(n-1)} + \underbrace{\vec{b}^{(n)}}_{2 \text{ dim.}}\right)$$

$\vec{z}^{(n)}$: same dim as # categ.

params of model: all the elements of $A^{(i)}, \vec{b}^{(i)}$ $i = 1, \ldots, n$ } put them in vec. $\vec{w}$

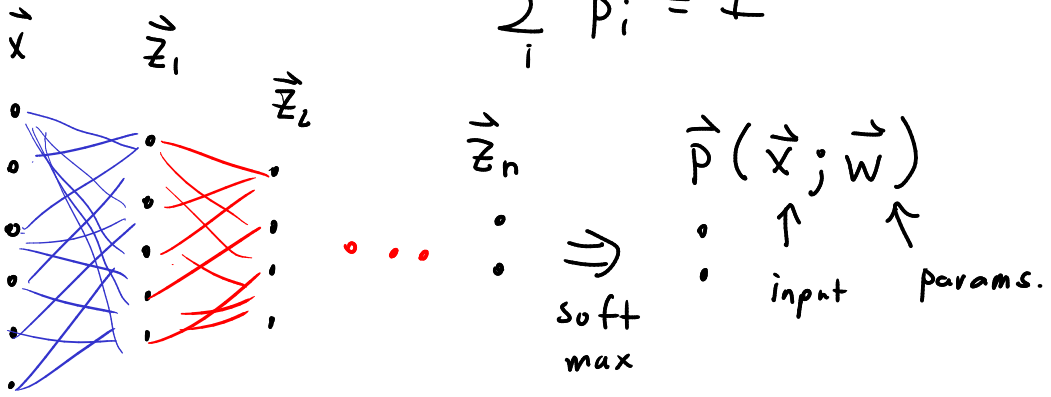
last transform: softmax converts $\vec{z}^{(n)}$ to a prob.

$$p_i = \frac{e^{-z_i^{(n)}}}{e^{-z_1^{(n)}} + e^{-z_2^{(n)}} + \dots}$$

$z_i^{(n)}$ is ith comp. of $\vec{z}^{(n)}$

guarantees that: $0 \le p_i \le 1$

$$\sum_i p_i = 1$$

$\vec{x}$: 

universal approx. theorems:

for DNNs if #layers & size of layers is large enough

$\Rightarrow$ you can approx. any func. that takes input from a finite domain

Bayesian interpretation:

dataset $\mathcal{D}$

"image"

$\vec{x}^{(1)}$

$\vdots$

$\vec{x}^{(N)}$

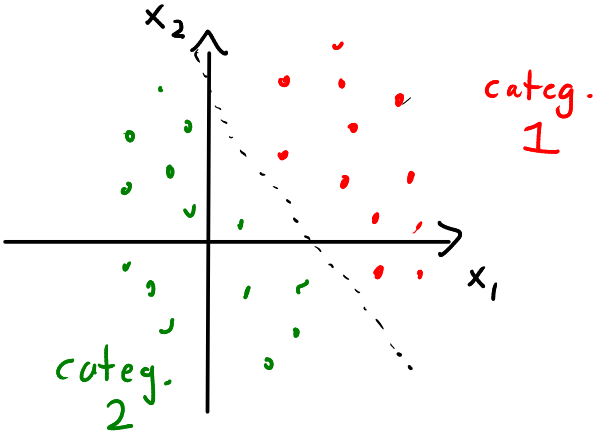
label:

$$\ell_1 = \begin{cases} 1 & \text{"cat"} \\ 2 & \text{"dog"} \end{cases}$$

$\vdots$

ℓ_N

$$\mathcal{D} = \{ (\vec{x}^{(i)}, l_i) \mid i = 1, \ldots, N \}$$



interested in $P(\vec{w} \mid \mathcal{D})$

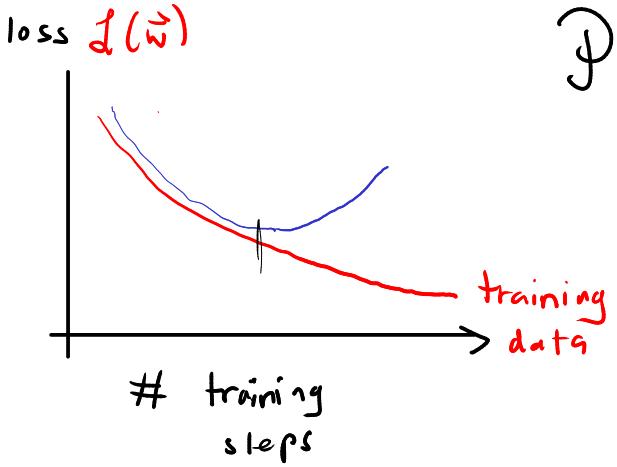
$\Rightarrow$ want to max. w/ respect to $\vec{w}$

"training" the network

$\vec{x}^{(i)} = (x_1^{(i)}, x_2^{(i)})$

first step: calculate

$$P(\mathcal{D} \mid \vec{w})$$



$$P(\mathcal{D} \mid \vec{w}) = \prod_{i=1} p_{l_i}(\vec{x}^{(i)}; \vec{w}) \leq 1$$

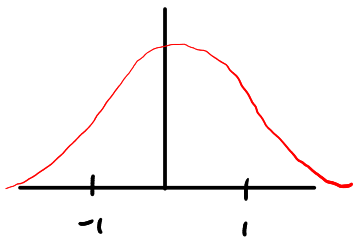
p_{l_i}: prob. net spits out correct label l_i given input $\vec{x}^{(i)}$

$P(\mathcal{D} \mid \vec{w}) = 1$ in ideal case of perfectly trained netw.

maximize $P(\vec{w} \mid \mathcal{D}) = \dfrac{P(\mathcal{D} \mid \vec{w}) \overbrace{P(\vec{w})}^{\text{prior}}}{P(\mathcal{D})}$

choose $P(\vec{w})$ for "regularization":

assume net params are not too big in magnitude to prevent blow-ups



$$P(w_i) = \frac{1}{\sqrt{2\pi}} e^{-w_i^2/2}$$ Gaussian w/ std. 1

$$P(\vec{w}) = \prod_i P(w_i)$$

trick: to maximize $P(\vec{w} \mid D)$

take $-\log$ of it & minimize

loss func. $\mathcal{L}(\vec{w}) = -\log P(\vec{w} \mid D)$

$$= -\log P(D \mid \vec{w}) - \log P(\vec{w}) + \log P(D)$$

$\log P(D)$: const.

x_1	x_2	label
-0.83	.51	1
⋮		