18/03/2025

Definition of the functional Risk for a function $f : X \to Y$ (Y = space of labels; X = "space of examples perceptual info")

it can also happen that $Y \subseteq X$ in some problems.

X = "space of handwritten characters"

Y = {0, 1, 2, ... 9}

$f \times id\,(x, y) = (f(x), y)$

Functional Risk

$$L(f) = E\, \ell \circ (f \times id) = \int_{X \times Y} \ell\,(f(x), y)\; d\pi(x, y)$$

If you know the probability distributions (or measure) of your data, then what L measures is the mean error that the model $f : X \to Y$ does.

↓

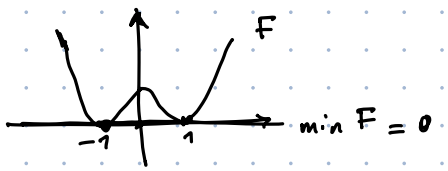
with respect to that distribution (π)

(*)

- Unless π is given (which in practice never happens) you cannot compute L and therefore you can't even try to minimize it.

(*) What is the goal of SML is to find the best model $f : X \to Y$, where best means a function that "minimize" L

- $f^* \in \operatorname{argmin} L(f)$. ←



$\operatorname{argmin} F = \{-1, 1\}$.

$F(-1) = F(1) = 0$

$F : H \to \mathbb{R}$ $\quad \min_{h \in H} F(h) = m$

m: $F(h) \geq m \quad \forall\, h \in H$

$\operatorname{argmin} F = \{ h \in H : F(h) = \min F \}$.

Instead of L we can write and optimize the Empirical Risk

$$L_N(f) = \frac{1}{N} \sum_{i=1}^{N} \ell\,(f(x_i), y_i) \qquad \{(x_i, y_i)\} = D$$

And the assumption is that (x_i, y_i) $i = 1, \ldots, N$ are independent samples from the same distribution π.

The other assumption that we tipically do is that instead of minimizing L_N on a general functional space we assume that f belongs to a parametric family of functions $\mathcal{F}$.

If you want you can think of $\mathcal{F}$ as a family of NN that share some fixed architecture.

So at this point you can define

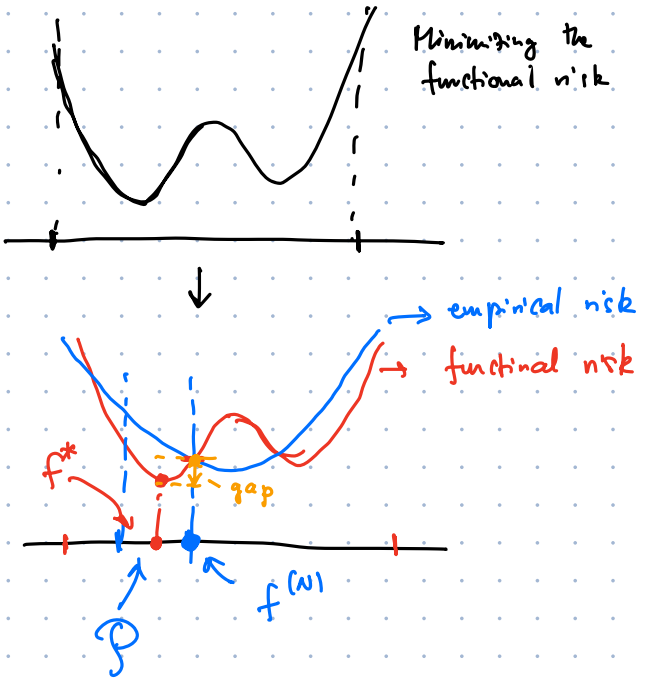
$$f^{(N)} \in \operatorname{argmin}\{ L_N(f) : f \in \mathcal{F} \}.$$

↑ This is what is actually computed by optimization algorithms.

But what we wanted to do is instead compute $\inf\{L(f) : f \in \operatorname{dom}(L)\}$.

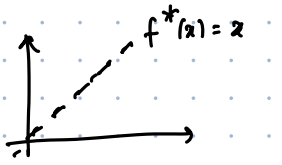
$$\text{gap} = L(f^{(N)}) - \inf\{L(f) : f \in \operatorname{dom}(L)\}$$

$$= \underbrace{\left(L(f^{(N)}) - \inf\{L(f) : f \in \mathcal{F}\}\right)}_{①} + \underbrace{\left(\inf\{L(f) : f \in \mathcal{F}\} - \inf\{L(f) : f \in \operatorname{dom}(L)\}\right)}_{②}$$



Ex. (Continuing the exercise of lecture 1).

$$L(f) = \frac{1}{\pi} \int_{\mathbb{R}^2} (f(x) - y)^2 \, e^{-(x-y)^2} e^{-x^2} \, dx \, dy.$$



We found a candidate to be a minimum $f^*(x) = x$.

$$L(f^*) = \frac{1}{2}$$

$$f(x) = e^{e^x}$$

We have to prove that $L(f) \geq \frac{1}{2} \quad \forall f \in \operatorname{dom}(L)$

$$L(f) = \frac{1}{\pi} \int_{\mathbb{R}^2} \underbrace{\left(f^2(x) - 2y f(x) + y^2\right)}_{① \quad ② \quad ③} e^{-(x-y)^2} e^{-x^2} \, dx \, dy$$

$$= \frac{1}{\pi} \int_{\mathbb{R}^2} f^2(x)\, e^{-(x-y)^2} e^{-x^2}\, dx\,dy - \frac{2}{\pi} \int_{\mathbb{R}^2} f(x)\, y\, e^{-(x-y)^2} e^{-x^2}\, dx\,dy$$

① = 0

$$+ \frac{1}{\pi} \int_{\mathbb{R}} y^2 e^{-(x-y)^2} e^{-x^2}$$

③

① ≥ 0 ←

$$② = -\frac{2}{\pi} \int_{\mathbb{R}} f(x)\, e^{-x^2} \left(\int_{\mathbb{R}} y\, e^{-(x-y)^2}\, dy \right) dx$$

$$= -\frac{2}{\pi} \int_{\mathbb{R}} f(x)\, e^{-x^2}\, x\sqrt{\pi}\, dx = -\frac{2}{\sqrt{\pi}} \int_{\mathbb{R}} f(x)\, x\, e^{-x^2}\, dx.$$

$y - x = z \quad y = x + z$

$$\int_{\mathbb{R}} y\, e^{-(x-y)^2}\, dy = \int_{\mathbb{R}} (x+z)\, e^{-z^2}\, dz = x\sqrt{\pi}$$

$$③ = \frac{1}{\pi} \int_{\mathbb{R}} y^2 e^{-(x-y)^2} e^{-x^2} = \frac{1}{2}$$

$$L(f) \geq \underbrace{-\frac{2}{\sqrt{\pi}} \int_{\mathbb{R}} f(x)\, x\, e^{-x^2}\, dx}_{?} + \frac{1}{2} \geq \frac{1}{2}$$

$$\frac{1}{\sqrt{\pi}} \int_{\mathbb{R}} f^2(x)\, e^{-x^2}\, dx - \frac{2}{\sqrt{\pi}} \int_{\mathbb{R}} f(x)\, x\, e^{-x^2}\, dx$$

$$= + \frac{1}{\sqrt{\pi}} \int_{\mathbb{R}} \underbrace{\left(f^2(x) - 2x f(x) \right)}_{\geq 0} e^{-x^2}\, dx + \frac{1}{2}$$

0

ATTEPT 2.

$f(x) = x$

$$\frac{1}{\pi} \int_{\mathbb{R}^2} (f(x) - y)^2\, e^{-(x-y)^2} e^{-x^2}\, dx\,dy$$

$$= \frac{1}{\pi} \int_{\mathbb{R}^2} \left(f(x) - x + x - y \right)^2 e^{-(x-y)^2} e^{-x^2}\, dx\,dy$$

$$= \frac{1}{\pi} \int_{\mathbb{R}^2} \left[(f(x) - x)^2 + (x-y)^2 + 2(x-y)(f(x) - x) \right] e^{-(x-y)^2} e^{-x^2}\, dx\,dy$$

$$= \frac{1}{\pi}\int_{\mathbb{R}^2} (f(x)-x)^2 e^{-(x-y)^2} e^{-x^2}\,dx\,dy \quad (1) \quad + \frac{1}{\pi}\int_{\mathbb{R}^2} (x-y)^2 e^{-(x-y)^2} e^{-x^2}\,dx\,dy \quad (2)$$

$$+ \frac{2}{\pi}\int_{\mathbb{R}^2} (x-y)(f(x)-x) e^{-(x-y)^2} e^{-x^2}\,dx\,dy \quad (3)$$

(1) ≥ 0 Because the integrand is positive.

$y - x = t$

(2) $$\frac{1}{\pi}\int_{\mathbb{R}^2} (x-y)^2 e^{-(x-y)^2} e^{-x^2}\,dx\,dy = \frac{1}{\pi}\int_{\mathbb{R}} e^{-x^2}\left(\int_{\mathbb{R}} (x-y)^2 e^{-(x-y)^2}\,dy\right)dx$$

$$= \frac{1}{\pi}\int_{\mathbb{R}} e^{-x^2}\left(\int_{\mathbb{R}} t^2 e^{-t^2}\,dt\right)dx = \frac{1}{\pi}\int e^{-x^2}\frac{\sqrt{\pi}}{2}\,dx = \frac{\pi}{2\pi} = \frac{1}{2}.$$

(3) $$\frac{2}{\pi}\int_{\mathbb{R}} (f(x)-x) e^{-x^2}\left(\int_{\mathbb{R}} (x-y) e^{-(x-y)^2}\,dy\right)dx$$

$$\int_{\mathbb{R}} t\, e^{-t^2}\,dt = 0.$$

← even function

↑ odd function

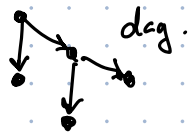
(1) + (2) + (3) $\geq 0 + \frac{1}{2} + 0 = \frac{1}{2} \Rightarrow L(f) \geq \frac{1}{2} \quad \forall f \in \mathrm{dom}(L)$

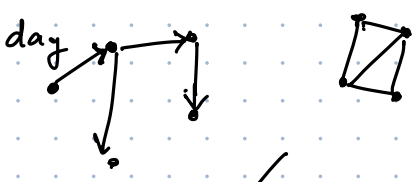
Since we know that $L(f^*) = \frac{1}{2}$ with $f^*(x) = x$, then f^* is a minimizer of L and $\frac{1}{2}$ is the minimum.

FEED FORWARD NEURAL NETWORKS.

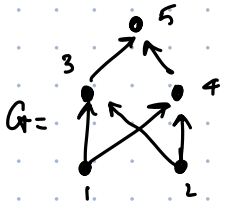
A feed forward neural network is a parametric function (its parameters are called "weights") whose computational structure is based on a dag.

$\varphi : \mathbb{R}^p \to \mathbb{R}^d$





1. You have a dag $G = (V, A)$



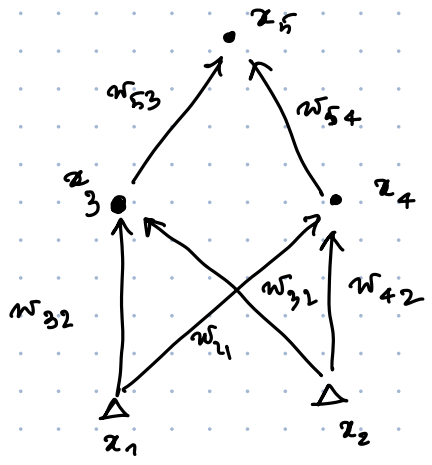
$|\{j : pa(j) = \emptyset\}| = p$

$|\{j : ch(j) = \emptyset\}| = d.$

2. You have a map that associate to each vertex $i \in V$ $\rightarrow x_i \in \mathbb{R}$

x_i is called "output of neuron i"

$$A \ni (j \to i) \longmapsto w_{ij} \in \mathbb{R}$$



$pa(1) = \phi$

$pa(2) = \phi$

$pa(3) = \{1, 2\}$

$pa(4) = \{1, 2\}$

$pa(5) = \{3, 4\}$

3. I give the following rule on how to compute the outputs

$$x_i = \sigma\left(\sum_{j \in pa(i)} w_{ij}\, x_j \right) \qquad \forall i \in V : \ pa(i) \text{ is non-empty.}$$

$$pa(i) = \{ j \in V : j \to i \in A \}$$

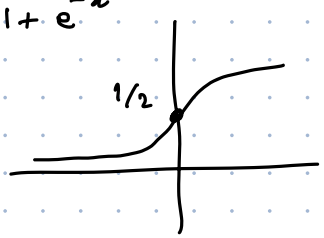
$$x_i = u_i \quad \text{if} \quad pa(i) = \phi .$$

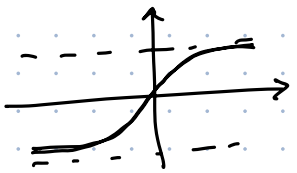
We call $u = (u_1, \dots, u_p)$ the input to the neural network.

$\sigma : \mathbb{R} \to \mathbb{R}$ is called "activation function" and:

1. Hystorically σ is a bounded increasing function (sigmoidal-like) $\sigma' > 0$

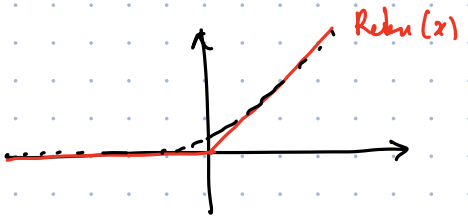
$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad \text{sigmoid}$$





$$\sigma(x) = \tanh(x)$$

2. More recently another common activation function is the ReLu positive part



$$\mathrm{Relu}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

$$(x)^+ \quad , \quad x \cdot [x \geq 0]$$

Example 1

$$x_1 = u_1$$

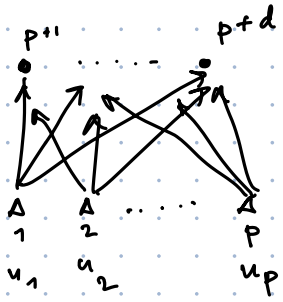
$$x_2 = \sigma(w x_1) = \sigma(w u_1)$$

$$f(u; w) = \sigma(w u)$$

$$u \longmapsto f(u; w)$$

↖ parameter or weight of the network.

Example 2



if $i \geq p+1$

$$x_i = \sigma\left(\sum_{j=1}^{p} w_{ij}\, x_j\right) \qquad j \in \{1, \dots, p\}$$

$$= \sigma\left(\sum_{j=1}^{p} w_{ij}\, u_j\right) \qquad (*)$$

$$W \in \mathbb{R}^{d \times p}$$

$$\sum_{j=1}^{p} w_{ij}\, u_j = W u$$

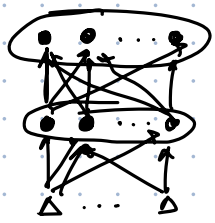
Therefore (*) is equivalent to

$$x = \vec{\sigma}(W u) \qquad \vec{\sigma}(x_1, \dots, x_n) = (\sigma(x_1), \dots, \sigma(x_n))$$

$$f(u; W) = \vec{\sigma}(W u) \qquad f(\cdot\,; w) : \mathbb{R}^p \to \mathbb{R}^d$$

Example 3.

Multi Layer Perceptron



$\vec{\sigma}(W_2 x^1)$

$\vec{\sigma}(W_1 u)$

$$f(u, w) = \vec{\sigma}(W_2\, \vec{\sigma}(W_1 u))$$

$$W_2 W_1 u$$

$$\underbrace{\qquad}_{M}$$

Backpropagation PDP book Geoffry Hinton

Suppose you want to compute the gradient of a function $\varphi \in C^1(\mathbb{R}^p; \mathbb{R})$

$$\nabla\varphi \approx \begin{pmatrix} \frac{\varphi(w_1+h, w_2, \dots w_p) - \varphi(w_1, w_2, \dots w_p)}{h} \\ \frac{\varphi(w_1, w_2+h, \dots w_p) - \varphi(w_1, w_2, \dots w_p)}{h} \\ \vdots \\ \frac{\varphi(w_1, w_2, \dots, w_p+h) - \varphi(w_1, \dots, w_p)}{h} \end{pmatrix}$$

$\varphi: \mathbb{R}^p \to \mathbb{R}$

Computing φ is $O(p)$

In general computing $\nabla\varphi$ numerically requires $O(p^2)$ which is a problem

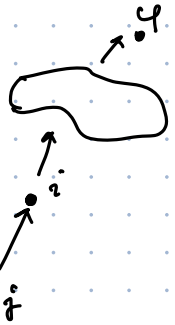
Relatively modern NN has 10^8 parameters, so to compute gradients we would require do do $O(10^{16})$ operations, which is computationally unfeasible.

(RNN)

It turns out that, instead if φ is not a generic map but it is a FNN you can improve the $O(p^2)$ bound and reduce it to $O(p)$ which is optimal. $[\Omega(p)]$

Suppose for simplicity that $\varphi: \mathbb{R}^p \to \mathbb{R}$

The parameters of φ are w_{ij} for $j \to i \in A$



$$\frac{\partial\varphi}{\partial w_{ij}} = \underbrace{\frac{\partial\varphi}{\partial a_i}}_{\delta_i} \frac{\partial a_i}{\partial w_{ij}}$$

δ_i - error of neuron i

$$\frac{\partial a_i}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_{k \in pa(i)} w_{ik} x_k$$

$$= \sum_{k \in pa(i)} \delta_{kj} x_k = x_j$$

$$a_i = \sum_{j \in pa(i)} w_{ij} x_j$$

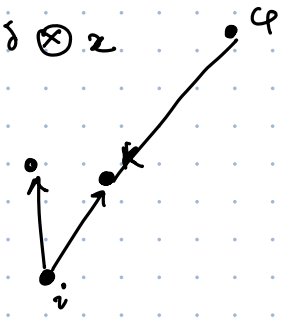
$$\delta_{kj} = \begin{cases} 1 & \text{if } k = j \\ 0 & \text{otherwise} \end{cases}$$

Kronecker δ Symbol

So we found that

$$\frac{\partial \varphi}{\partial w_{ij}} = \delta_i z_j$$

$\nabla \varphi = \delta \otimes z$



- $\delta_i = \frac{\partial \varphi}{\partial a_i} = \sum_{k \in ch(i)} \frac{\partial \varphi}{\partial a_k} \frac{\partial a_k}{\partial a_i}$

$$= \sum_{k \in ch(i)} \delta_k \frac{\partial a_k}{\partial a_i}$$

$z_i = \sigma\left(\underbrace{\sum_{j \in pa(i)} w_{ij} z_j}_{a_i}\right)$

$z_i = \sigma(a_i)$

$$a_k = \sum_{m \in pa(k)} w_{km} z_m = \sum_{m \in pa(k)} w_{km} \sigma(a_m)$$

$$\frac{\partial a_k}{\partial a_i} = \sum_{m \in pa(k)} w_{km} \sigma'(a_m) \delta_{mi} = w_{ki} \sigma'(a_i)$$

$$\delta_i = \sum_{k \in ch(i)} \delta_k w_{ki} \sigma'(a_i)$$

$$\boxed{\delta_i = \sigma'(a_i) \sum_{k \in ch(i)} \delta_k w_{ki}}$$

When i is an output neuron $\boxed{\delta_i = \frac{\partial \varphi}{\partial a_i} = \sigma'(a_i)}$

$x_i = \sigma\left(\sum w_{ij} x_j\right) \rightarrow x = \sigma(Wz)$ $O(p)$

$\delta_i = \sigma'(a_i) \sum \delta_k w_{ki} \rightarrow \delta \propto W'\delta$ $O(p)$ W' is transpose