TD 2 : SVM, optimization and Neural Nets

Ex1 : Positive definite (p.d) kernels

Notes :

- A kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is said to be p.d kernel if for any $\{x_1, \dots, x_m\} \subseteq \mathcal{X}$ the matrice $K = [K(x_i, x_j)]_{ij} \in \mathbb{R}^{m \times m}$ is p.d
- K is a p.d. kernel if one of the following 2 equivalent conditions holds :
  (i) the eigenvalues of K are non-negative ;
  (ii) $\forall a \in \mathbb{R}^m$, $a^T K a = \sum_{i,j} a_i a_j K(x_i, x_j) \geq 0$

prop: if K_i p.d. kernel and $d_i \geq 0$ $\forall i \in \mathbb{N}$

(i) $d_1 K_1 + d_2 K_2$ and $K_1 K_2$ p.d kernels
(ii) $\lim_n K_n$ is p.d. kernel if the limit exists
(iii) $K_1(x, y) = K_2(\phi(x), \phi(y))$ p.d. kernel

$\Big\} \Rightarrow \sum_{n \geq 0} d_n K^n$ p.d. kernel (power series)

proof:

(i) Trivial for $d_1 K_1 + d_2 K_2$. For the other, it suffices to show that 2 Hermitian matrices A and B are p.d, so is their component-wise product
(ii) Just notice that the non-negativity in the definition holds also for the limit
(iii) Trivial

prop: (i) $K = \exp(K_1)$ p.d. kernel
(ii) $K(x, y) = e^{-\|x - y\|^2 / 2\sigma^2}$ p.d kernel

proof:

(i) $\exp(K_1) = \sum_{n \geq 0} \frac{1}{n!} K_1^n$ p.d. K
(ii) see course

Thm: (Aronszajn, 1950) K p.d kernel iff $\exists$ $\mathcal{H}$ hilbert space and mapping $\phi : \mathcal{X} \to \mathcal{H}$ s.t.

$$\forall x, x' \in \mathcal{X} \quad K(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$$

1) $\forall x,y \in \mathbb{R}$, $\begin{cases} K_1(x,y) = 10^{xy} \\ K_2(x,y) = 10^{x+y} \end{cases}$

$K_1(x,y) = 10^{xy} = e^{xy\ln(10)}$ with $\mathcal{X} = \mathbb{R}$ is a exponential of a scaled (by factor $\ln(10) > 0$) linear kernel

$\Rightarrow$ p.d. kernel

- $x = (x_1, \dots, x_n) \in \mathbb{R}^n$ and $a = (a_1, \dots, a_n) \in \mathbb{R}^n$,

$$a^T K_2 a = \sum_{i=1}^n \sum_{j=1}^n a_i a_j K_2(x_i, x_j) = \sum_{i=1}^n \sum_{j=1}^n a_i a_j 10^{x_i + x_j}$$

$$= \sum_{i=1}^n a_i 10^{x_i} \cdot \sum_{j=1}^n a_j 10^{x_j} = \left(\sum_{i=1}^n a_i 10^{x_i}\right)^2 \geq 0$$

$\Rightarrow$ p.d. kernel

2) $x,y \in [0,1[$, $K_3(x,y) = -\log(1-xy) = \sum_{n \geq 1} \frac{(xy)^n}{n} \Rightarrow K_3$ p.d. kernel

3) an example before : $\tilde{K}(x,y) = \min(x,y)$ with $\mathcal{X} = \mathbb{R}_+$

Let us define $\phi : \mathbb{R} \to L^2(\mathbb{R}_+)$

$$x \mapsto \mathbb{1}_{[0,x]}$$

$\forall a,b \in \mathbb{R}^+$

$$\left.\begin{array}{l} \int_0^\infty \phi(a) = \int_0^a \mathbb{1} = a \\ [0,a] \cap [0,b] = [0, \min(a,b)] \\ \phi(\min(a,b)) = \phi(a)\,\phi(b) \end{array}\right\} \Rightarrow \min(a,b) = \int_0^{+\infty} \mathbb{1}_{[0,a]} \mathbb{1}_{[0,b]}$$

$$= \langle \phi(a), \phi(b) \rangle_{L^2(\mathbb{R}_+)}$$

Thanks to Aronszajn (1950) Thm, $\tilde{K}$ p.d kernel

exercise : $\max(x,y)$ with $\mathcal{X} = \mathbb{R}_+$ p.d Kernel ?

Back to the question :

$$K_4(x,y) = \min\{f(x)g(y), f(y)g(x)\} = \min\left\{\frac{f(x)}{g(x)}, \frac{f(y)}{g(y)}\right\} g(x)g(y)$$

By using $\phi_4(x) = \mathbb{1}_{\left[0, \frac{f(x)}{g(x)}\right]}$,

$$\min\left\{\frac{f(x)}{g(x)}, \frac{f(y)}{g(y)}\right\} = \int \mathbb{1}_{\left[0,\frac{f(x)}{g(x)}\right]} \mathbb{1}_{\left[0,\frac{f(y)}{g(y)}\right]} = \langle \phi_4(x), \phi_4(y) \rangle_{L^2(\mathbb{R}_+)}$$

$\left.\begin{array}{l} \min\left\{\frac{f(x)}{g(x)}, \frac{f(y)}{g(y)}\right\} \text{ is a p.d Kernel} \\ \text{Moreover, } g(x)g(y) \text{ is a p.d kernel (trivial)} \end{array}\right\} \Rightarrow K_4$ p.d Kernel

4) a) Let S be a sequence of words on a finite alphabet A

For example $A = \{A, B, C, \dots, Z\}$

$$x = \text{CGGSLIAMMWFGV}$$

Subchains of length 5 in x $= (\bullet, \bullet, \dots, \bullet)$

Let $\phi_u(x)$ denote the number of subchains u in x

Let us consider $k(x,y) = \#$ subchains of length 5 that x and $y \in S$ share

$\Rightarrow$ $k(x,y) = \sum_{u \in A^5} \phi_u(x)\phi_u(y)$ which is clearly a p.d kernel :

$$\sum_{i,j=1}^{m} a_i a_j k(x_i, x_j) = \sum_{i,j=1}^{m} a_i a_j \sum_{u \in A^5} \phi_u(x_i)\phi_u(x_j)$$

$$= \sum_{u} \left(\sum_{i} a_i \phi_u(x_i)\right)^2 \geq 0$$

b) Left as an exercise

Ex 2 : Computations of RKHS

Definition of reproducing kernel (r.k) and RKHS (Reproducing Kernel Hilbert Space)

$K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a r.k of a Hilbert space $\mathcal{H}_0 \subset \mathbb{R}^{\mathcal{X}}$ iF

(i) $\mathcal{H}_0$ contains all functions of the form

$$\forall x \in \mathcal{X}, \quad K_x : t \mapsto K(x,t)$$

(ii) $\forall x \in \mathcal{X}, h \in \mathcal{H}_0$ the reproducing property holds:

$$h(x) = \langle h, K_x \rangle_{\mathcal{H}_0}$$

IF a r.k exists, then $\mathcal{H}_0$ is called RKHS

In particular, $\forall x, y \in \mathcal{X}$, $K(x,y) = \langle K_x, K_y \rangle_{\mathcal{H}_0}$

Thm (see course):

For any set $\mathcal{X}$, a function $K : \mathcal{X} \times \mathcal{X}$ is a p.d iff it is a r.k

1)
$$\begin{cases} K_5(x,y) = (xy+1)^2 \\ K_6(x,y) = (xy-1)^2 \\ K_7(x,y) = K_5(x,y) + K_6(x,y) \\ \quad = (xy+1)^2 + (xy-1)^2 \\ \quad = 2((xy)^2+1) \end{cases}$$
which are clearly p.d. kernels.

These are special cases of the polynomial kernel of degree p:

$$K_{poly}(x,y) = (\langle x,y \rangle_{\mathbb{R}^d} + c)^p$$

Let us find its RKHS $\mathcal{H}$ for $p=2$ and $c=0$

(Left as an exercise : - what is the RKHS of the general K_{poly} ?
- what about K_5, K_6 and K_7)

- step 1 : Look for an inner-product

$$\begin{aligned} K_{poly}(x,y) &= x^T y \, x^T y \\ &= \text{Trace}(x^T y \, x^T y) \\ &= \text{Trace}(x x^T y y^T) \\ &= \langle x x^T, y y^T \rangle_F \end{aligned}$$

rks: $Tr(AB) = Tr(BA)$, $Tr(A^t) = Tr(A)$

where F is the Frobeinus norm for matrices $\mathbb{R}^{d \times d}$

- step 2 : propose a candidate RKHS

We know that $\mathcal{H}$ contains all the functions

$$f(x) = \sum_i a_i K(x_i, x) = \sum_i a_i \langle x_i x_i^T, x x^T \rangle_F = \langle \sum_i a_i x_i x_i^T, x x^T \rangle_F$$

Any symmetric matrix in $\mathbb{R}^{d \times d}$ may be decomposed as $\sum_i a_i x_i x_i^T$

$\Rightarrow$ Candidate RKHS $\mathcal{H}$: the set of quadratic functions

$$f_S(x) = \langle S, xx^T \rangle_F = x^T S x \quad \text{for } S \in S^{d \times d}$$

where $S^{d \times d}$ is the set of symmetric matrices in $\mathbb{R}^{d \times d}$, endowed with the inner product $\langle f_{S_1}, f_{S_2} \rangle_{\mathcal{H}} = \langle S_1, S_2 \rangle_F$

- step 3 : check that the candidate is a Hilbert Space
  Here it is trivial because $\mathcal{H}$ is an Euclidean space

- step 4 : check that $\mathcal{H}$ is the RKHS

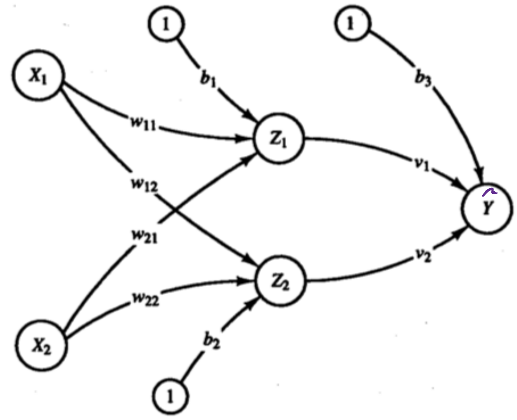
  (i) $\{ K_x : t \mapsto K(x,t) = \langle xx^T, tt^t \rangle_F \} \subset \mathcal{H}$

  (ii) Let us consider $f_S \in \mathcal{H}$ and $x \in \mathcal{X}$. Then,

$$f_S(x) = \langle S, xx^T \rangle_F = \langle f_S, f_{xx^T} \rangle_{\mathcal{H}} = \langle f_S, K_x \rangle_{\mathcal{H}}$$

2) see ex 1 for the p.d kernels property.

Ex4 : Neural Nets



back prop

Algorithm for this architecture :

step 0 : initialize weights

step 1 : while "stopping criterion is false"
do step 2-6

(2) For each training step :

feed forward / forward propagation

(3) compute output of each hidden unit :

activation function

$$z\text{-}in_1 = b_1 + x_1 w_{11} + x_2 w_{21} \Rightarrow z_1 = \sigma(z\text{-}in_1)$$
$$z\text{-}in_2 = b_2 + x_1 w_{12} + x_2 w_{22} \Rightarrow z_2 = \sigma(z\text{-}in_2)$$

(4) compute output of net

$$y\text{-}in = b_3 + z_1 v_1 + z_2 v_2 \Rightarrow \hat{y} = \sigma(y\text{-}in)$$

back propagation

(5) compute error and weights / biaises correction terms :

$$\delta = (y - \hat{y})\,\sigma'(y\text{-}in)$$ (error on net output)

$$\Delta v_j = \alpha \cdot \delta \cdot z_j \quad \text{and} \quad \Delta b_3 = \alpha \cdot \delta$$ (weights correction term)

$$\delta_j = \delta\_in_j \cdot \sigma'(z\text{-}in_j) \quad \text{with} \quad \delta\text{-}in_j = \delta \cdot v_j$$ (error on each hidden unit)

$$\Delta w_{ij} = \alpha \cdot \delta_j \cdot x_i \quad \text{and} \quad \Delta b_j = \alpha \cdot \delta_j$$ (weights correction term)

(6) updates weights and biaises :

$$v_j^{new} \leftarrow v_j + \Delta v_j \quad \text{and} \quad b_3^{new} \leftarrow b_3 + \Delta b_3$$
$$w_{ij}^{new} \leftarrow w_{ij} + \Delta w_{ij} \quad \text{and} \quad b_j^{new} \leftarrow b_j + \Delta b_j$$

(gradient descent)

rks :

- Let us define a loss function $\ell = \frac{1}{2}(\hat{y} - y)^2$,

$$\frac{\partial \ell}{\partial \hat{y}} = (\hat{y} - y) \; ; \; \frac{\partial \ell}{\partial v_j} = \frac{\partial \ell}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial y\text{-}in} \cdot \frac{\partial y\text{-}in}{\partial v_j} = \underbrace{(\hat{y} - y)\,\sigma'(y\text{-}in)}_{-\delta}\, z_j$$

1)

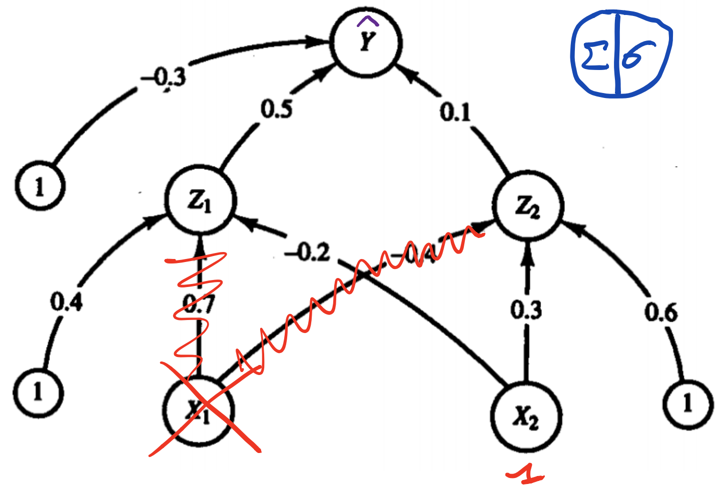
$(X_1, X_2) = (0,1)$

$Y = 1$

step size : $\alpha = 0.25$

activation function : $\sigma(x) = \dfrac{1}{1+e^{-x}}$

new weights ?



$$z\text{-}in_1 = 0.4 + (-0.2) = 0.2 \Rightarrow z_1 = \frac{1}{1+e^{-0.2}} \approx 0.55$$

$$z\text{-}in_2 = 0.6 + 0.3 = 0.9 \Rightarrow z_2 = \frac{1}{1+e^{-0.9}} \approx 0.71$$

$$y\_in = -0.3 + 0.5 \cdot z_1 + 0.1 z_2 \approx 0.05 \Rightarrow \hat{y} \approx \frac{1}{1+e^{-0.05}} \approx 0.51$$

BP $\delta \approx (1-0.51)\,\sigma'(0.05) \approx 0.12$

$$\left(\sigma'(x) = \sigma(x)(1-\sigma(x)) = \frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right) \right)$$

$$\Delta v_1 \approx 0.25 \times 0.12 \times 0.55 \approx 0.018$$

$$\Delta v_2 \approx 0.25 \times 0.12 \times 0.71 \approx 0.02$$

$$\Delta b_3 \approx 0.25 \times 0.12 \approx 0.03$$

$$\delta_1 = \delta \cdot v_1 \cdot \sigma'(z\_in_1) \approx 0.12 \times 0.5 \times \sigma'(0.2) \approx 0.02$$

$$\delta_2 = \delta \cdot v_2 \cdot \sigma'(z\text{-}in_2) \approx 0.12 \times 0.1 \times \sigma'(0.9) \approx 0.003$$

$$\Delta w_{11} = \Delta w_{12} = 0 \qquad \Delta w_{21} \simeq 0.05$$

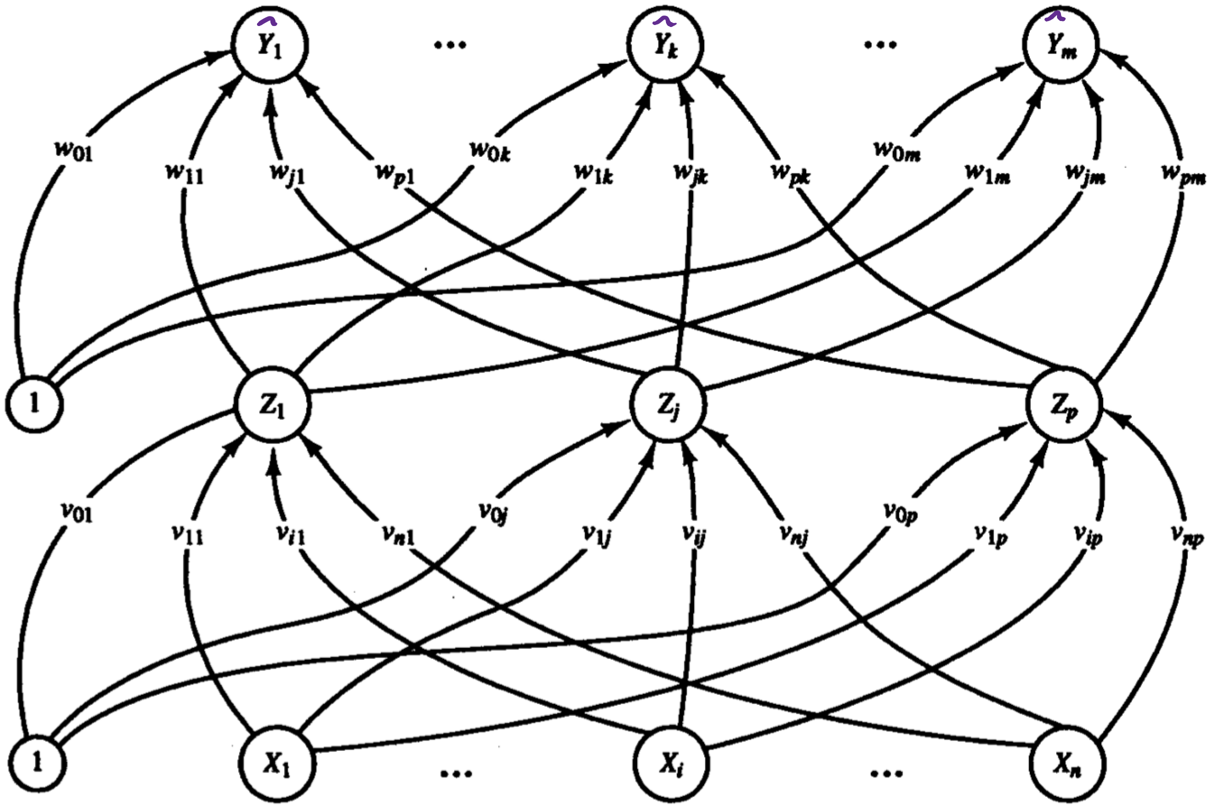
$$\Delta w_{22} \approx 0.00075$$

$v_1 \simeq 0.5 + 0.018 \simeq 0.517$ $\quad v_2 \simeq 0.1 + 0.02 \simeq 0.12$

$b_3 \approx -0.3 + 0.03 \approx -0.27$ $\quad$ ... until stopping criterion

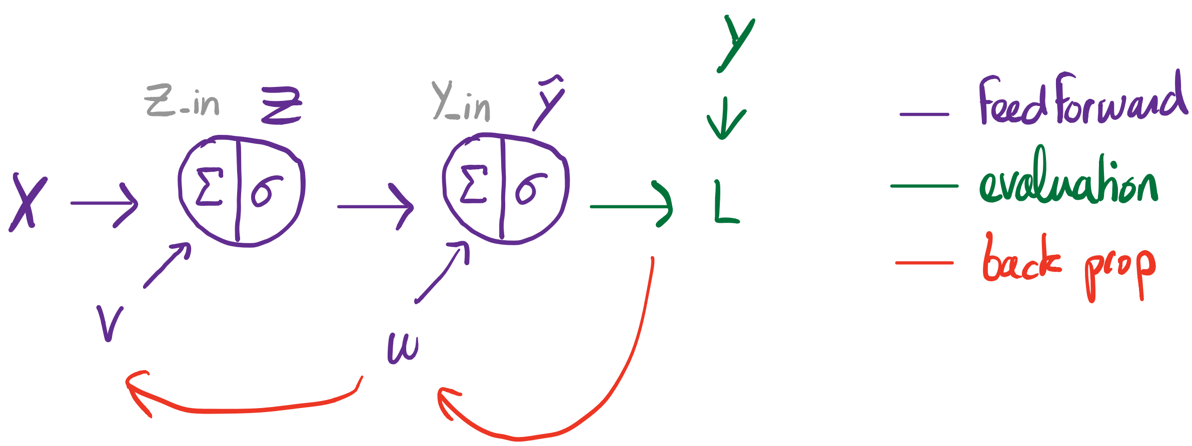
Exercice : implement with python feedforward and backprop answering question 1, 2 and 3

An overview



Graphs taken from "Fundamentals of Neural Networks" by L. Fausett

Let us simplify the graph :



- update its weights by gradient descent : $\begin{cases} w_{ij} \leftarrow w_{ij} - \alpha \dfrac{\partial L}{\partial w_{ij}} \\ v_{ij} \leftarrow v_{ij} - \alpha \dfrac{\partial L}{\partial v_{ij}} \end{cases}$
- compute $\begin{cases} \dfrac{\partial L}{\partial w_{ij}} \\ \dfrac{\partial L}{\partial v_{ij}} \end{cases}$ by chain-rule