# Machine Learning. Homework 2.

Sofiia Petryshyn

APPS UCU, CS 2020

## 1 Create kernels

**a)**
**For each of these functions, specify whether it is a kernel or not. If you think the**
**function is a kernel, prove it. If you think not, give one prove example.**

**Kernel functions:**
a) $K(x,z) = K_1(x,z) + K_2(x,z)$ is a kernel.
b) $K(x,z) = K_1(x,z) - K_2(x,z)$ is not a kernel.
c) $K(x,z) = a \cdot K_1(x,z)$ is a kernel.
d) $K(x,z) = -\alpha \cdot K_1(x,z)$ is not a kernel.
e) $K(x,z) = K_1(a \cdot x, b \cdot z)$ is not a kernel.
f) $K(x,z) = K_1(x,z) \cdot K_2(x,z)$ is a kernel.
g) $K(x,z) = f(x) \cdot f(z)$ is a kernel.
h) $K(x,z) = K_3(\phi(x), \phi(z))$ is a kernel.
i) $K(x,z) = p(K_1(x,z))$ is a kernel.
j) $K(x,z) = \alpha \cdot (K_1(x,z)) - \beta \cdot (K_2(x,z))$ is not a kernel.
k) $K(x,z) = -\alpha \cdot (K_1(x,z)) - \beta \cdot (K_2(x,z))$ is not a kernel.
According to the Closure properties here is a proof to a, c, f, g, h:
**a.** Recall that K is positive semi-definite if and only if $\alpha^{'} \cdot K \cdot \alpha) >= 0$, for all $\alpha$.

$$\alpha^{'} \cdot (K_1 + K_2) \cdot \alpha = \alpha^{'} \cdot K_1 \cdot \alpha + \alpha^{'} \cdot K_2 \cdot \alpha) >= 0$$

, so $K_1 + K_2$ is positive semi-defined and symmetric, so that valid to be a kernel.
**c.** $\alpha^{'} \cdot a \cdot (K_1) \cdot \alpha = a \cdot \alpha^{'} \cdot (K_1) \cdot \alpha >= 0$
**f.** Let $K = K_1 K_2$.
A tensor product of two semi-definite matrices is positive semi-definite the matrix of multiplication of two kernels is known as Schur product = H. The H is a principal submatrix of K, if we prove that H is positive semi-deinite, then K is positive semi-definite.

We know that for any $\alpha \in R^l$ there is corresponding $\beta \in R^{l^2}$, so $\alpha^{'} \cdot (K) \cdot \alpha = \beta^{'} \cdot (K) \cdot \beta >= 0$, so H is positive semi-definite, as follows K as well, so K is a kernel.

**g.** Consider we have a 1 dim feature map $g : x- > f(x) \in R$, then $K(x, z)$ is the corresponding kernel for this mapping, then K is a valid kernel.

**h.** From the task we know that $K_3$ is a kernel of the vectors in $R^d$, and we know that the function $\phi : R^n - > R^d$. So if we set $\phi$ points into the kernel $K_3$, then $K_3$ is a kernel.

Proof of **i**:
As $p(x)$ is. a polinomial with positive coefficients, then we have summation and multiplication of values.

- First of all, we have the addition of the kernels and we have to prove that it is a valid kernel.

Recall that K is positive semi-definite if and only if $\alpha^{'} \cdot K \cdot \alpha) >= 0$, for all $\alpha$.

$$\alpha^{'} \cdot (K_1 + K_2) \cdot \alpha = \alpha^{'} \cdot K_1 \cdot \alpha + \alpha^{'} \cdot K_2 \cdot \alpha) >= 0$$

, so $K_1 + K_2$ is positive semi-defined and symmetric, so that valid to be a kernel.

- Second thing, is to prove that $K(x, z) = a \cdot K_1(x, z)$ is a kernel.
$\alpha^{'} \cdot a \cdot (K_1) \cdot \alpha = a \cdot \alpha^{'} \cdot (K_1) \cdot \alpha >= 0$

- And the third situation that we can have is kernels multiplication.

Let $K = K_1 K_2$.
A tensor product of two semi-definite matrices is positive semi-definite the matrix of multiplication of two kernels is known as Schur product = H. The H is a principal submatrix of K, if we prove that H is positive semi-deinite, then K is positive semi-definite.
We know that for any $\alpha \in R^l$ there is corresponding $\beta \in R^{l^2}$, so $\alpha^{'} \cdot (K) \cdot \alpha = \beta^{'} \cdot (K) \cdot \beta >= 0$, so H is positive semi-definite, as follows K as well, so K is a kernel.

Contr-argument to b, d, e, j, k: **b.** Recall that K is positive semi-definite if and only if $\alpha^{'} \cdot K \cdot \alpha >= 0$, for all $\alpha$.

$$\alpha^{'} \cdot (K_1 - K_2) \cdot \alpha = \alpha^{'} \cdot K_1 \cdot \alpha - \alpha^{'} \cdot K_2 \cdot \alpha) >= 0$$

, if $K_2 > K_1$, then $\alpha^{'} \cdot (K_1 - K_2) \cdot \alpha = \alpha^{'} \cdot K_1 \cdot \alpha - \alpha^{'} \cdot K_2 \cdot \alpha) < 0$, so $K$ is not positive semi-definite. That means that K is not a kernel.
**d.** As $K_1$ is a kernel, then is it a positive semi-definite matrix, so from the problem we know that $\alpha$ is a non-negative number, if we apply minus in front of the equation, we'll get a non-positive definite matrix. So K is not a kernel.
**e.** As we have a valid kernel $K_1(x,z)$, it is symmetric and positive semi-definite. If we multiply function elements by a non-negative scalar we are making changes in terms of matrix values, so we are not sure anymore if the matrix is semi-positive and symmetric, so $K = K_1 a \cdot x, b \cdot y$ is not a kernel.
**j.** If $K_1(x,z) < \frac{\beta}{\alpha} \cdot K_2(x,z)$, then
$\alpha^{'} \cdot (K_1 - \frac{\beta}{\alpha} \cdot K_2) \cdot \alpha = \alpha^{'} \cdot K_1 \cdot \alpha - \alpha^{'} \cdot \frac{\beta}{\alpha} \cdot K_2 \cdot \alpha < 0$, so $K$ is not positive semi-definite. That means that K is not a kernel. **k.**
$K(x,z) = -\alpha \cdot (K_1(x,z)) - \beta \cdot (K_2(x,z)) = -(\alpha \cdot (K_1(x,z)) + \beta \cdot (K_2(x,z))) < 0$, so in any case $K$ is not positive semi-definite. That means that K is not a kernel.

# 2 Complex projections of signs

**a)**
**If the projections from x to $\phi_1(x)$ and from $\phi_1(x)$ to $\phi_2(x)$ are realized by a linear kernel function, what will be the kernel function that realizes the projection from x to $\phi_2(\phi_1(x))$?**

Consider a mapping kernel function $\phi_1$ operation from an input space, to which x belongs to another space to which $\phi_1$ belongs. $\phi_1 : x \implies \phi_1(x)$.

Typical linear kernel is $k(x,z) = \phi_1(x)^T \cdot \phi_1(z)$.

If we have $x \implies \phi_2(\phi_1(x))$, then we are supposed to have $\phi_1 : x \implies \phi_1(x)$, then $K_1(x,z) = K_1(\phi_1(x), \phi_1(z)) = \phi_1(x)^T \phi_1(z) = (x)^T(z)$;
$\phi_1(x) \implies \phi_2(\phi_1(x))$
$K_2(\phi_2(\phi_1(x)), \phi_2(\phi_1(z))) = \phi_2(\phi_1(x))^T \phi_2(\phi_1(z)) = \phi_1(x)^T \phi_1(z)$.
To transpose from x to $\phi_2(\phi_1(x))$ we want to do:

$$(x \implies \phi_2(\phi_1(x)) =$$
$$= (x \implies \phi_1(x)) \implies \phi_2(\phi_1(x)) =$$

Define $K_2$ in terms of $K_1$, here we have: $= K_2(K_1(\phi_1(x), \phi_1(z))) =$
As $K_1 = \phi_1(x)^T \phi_1(z)$ and $K_2 = \phi_1(x)^T \phi_1(z)$, then $K_2$ in terms of $K_1$ is a simple $K_2$.

**b)**
**If the projections from x to $\phi_1(x)$ and from $\phi_1(x)$ to $\phi_2(x)$ are realized by a polynomial kernel function, what will be the kernel function that realizes the projection from x to $\phi_2(\phi_1(x))$?**
Consider a mapping kernel function $\phi_1$ operation from an input space, to

which x belongs to another space to which $\phi_1$ belongs. $\phi_1 : x \implies \phi_1(x)$.

Typical polynomial kernel is $k_{d,c}(x,z) = (\phi_1(x)^T \cdot \phi_1(z) + c)^d$.

1. $\phi_1 : x \implies \phi_1(x)$
$K_1(\phi_1(x), \phi_1(z)) = (\phi_1(x)^T \phi_1(z) + c_1)^{d_1}$

2. $\phi_2 : \phi_1(x) \implies \phi_2(\phi_1(x))$
$K_2(\phi_2(\phi_1(x)), \phi_2(\phi_1(z))) = K_2(\phi_2 \circ \phi_1(x), \phi_2 \circ \phi_1(z)) =$
$= ((\phi_2 \circ \phi_1)(x)^T (\phi_2 \circ \phi_1)(z) + c_2)^{d_2}$

3. $\phi_3 : (x \implies \phi_1(x)) \implies \phi_2(\phi_1(x))$
Here we will describe $K_2$ in terms of $K_1$, so we will represent $\phi_1$ & $\phi_2$ by $K_1$:

$$\text{Kernel 1: } K_1 = (\phi_1)(x)^T \phi_1(z) + c_1)^{d_1}, \text{ let } d_1 = 2$$
$$K_1 = \phi_1(x)^2 \phi_1(z)^2 + 2c_1 \phi_1(x)^T \phi_1(z) + c_1^2$$
$$\text{Kernel 2: } K_2 = (\phi_2(\phi_1)(x))^T \phi_2(\phi_1(z)) + c_2)^{d_2}, \text{ let } d_2 = 2$$
$$K_2 = \phi_2(\phi_1(x))^2 \phi_2(\phi_1(z))^2 + 2c_2 \phi_2(\phi_1(x))^T \phi_2(\phi_1(z)) + c_2^2$$
Represent $\phi_1(x)$ i terms of kernel 1:
$$\phi_1(x)^T \phi_1(x) \cdot \phi_1(z)^T \phi_1(z) \phi_1(x)^T \phi_1(x) + 2c_1 \cdot \phi_1(x)^T \phi_1(z) \phi_1(x)^T \phi_1(x) = K_1^* - c_1^*$$
, where $K_1^*$ & $c_1^*$ the constants are divided by $\phi_1(x)^T \phi_1(x)$
$$\phi_1(z)^2 + 2c_1 \phi_1(z) \phi_1(x) = K_1^* - c_1^*$$
$$\phi_1(x) = 2c_1(\phi_1(z) K_1^* - 1\phi_1(z) - \phi(z) c_1^2)$$
Let do the same for $\phi_1(z)$:
$$\phi_1(z)^T = 2c_1(\phi(x)^T K_1^* - \phi_1(x)^T c_1^2 - 1\phi_1(x))$$
$$\phi_1(z) = 2c_1(\phi(x) K_1^{(*)(T)} - \phi_1(x) c_1^{(2)(T)} - 1\phi_1(x)^T)$$
Now we can rewrite $K_2$ in terms of $K_1$:
$$K_2 = \phi_2(2c_1(\phi_1(z) K_1^* - 1\phi_1(z) - \phi(z) c_1^2))^2 \cdot \phi_2(2c_1(\phi(x) K_1^{(*)(T)} - \phi_1(x) c_1^{(2)(T)} - 1\phi_1(x)^T)) + 2c_2 \phi_2(2c_1(\phi_1(z) K_1^* - 1\phi_1(z) - \phi(z) c_1^2)) \phi_2(2c_1(\phi(x) K_1^{(*)(T)} - \phi_1(x) c_1^{(2)(T)} - 1\phi_1(x)^T)) + c_2^2$$

**c)**
**If the projections from x to $\phi_1(x)$ and from $\phi_1(x)$ to $\phi_2(x)$ are realized by the radial basis function kernel $\exp^{-\epsilon(x-z)^2}$, what will be the kernel function that realizes the projection from x to $\phi_2(\phi_1(x))$?**

1. $\phi_1 : x \implies \phi_1(x)$
$K_1(\phi_1(x), \phi_1(z)) = \exp^{-\epsilon(\phi_1(x) - \phi_1(z))^2}$

2. $\phi_2 : \phi_1(x) \implies \phi_2(\phi_1(x))$
$K_2(\phi_2(\phi_1(x)), \phi_2(\phi_1(z))) = K_2(\phi_2 \circ \phi_1(x), \phi_2 \circ \phi_1(z)) =$
$= \exp^{-\epsilon(\phi_2(\phi_1(x)) - \phi_2(\phi_1(z))^2}$

3. $\phi_3 : (x \implies \phi_1(x)) \implies \phi_2(\phi_1(x))$

$$ln(K_1) = -\epsilon(\phi_1(x) - \phi_2(z))^2$$
$$\sqrt{ln(K_1) - \epsilon} = \phi_1(x) - \phi_1(z)$$
Here we find $\phi_1(x)$ : $\phi_1(x) = \sqrt{ln(K_1) - \epsilon} + \phi_1(z)$ Here we find $\phi_1(z)$ :
$\phi_1(z) = \phi_1(x) - \sqrt{ln(K_1) - \epsilon}$ Now we can rewrite $K_2$:
$$K_2 = \exp^{-\epsilon(\phi_2(\sqrt{ln(K_1)-\epsilon}+\phi_1(z))-\phi_2(\phi_1(x)-\sqrt{ln(K_1)-\epsilon}))^2}$$

$$K_2(K_1(x,z)) = K_2 \circ K_1 = \exp^{-\epsilon(\phi_2(\phi_1(x))-\phi_2(\phi_1(z))^2} \circ \exp^{-\epsilon(\phi_1(x)-\phi_1(z))^2}$$

# 3 Neural networks

Consider a neural network with the activation function of the sigmoid on the hidden layer:

$$a_h(x) = g(x) = \frac{1}{1+\exp^{-x}}$$

Show that there is a neural network with an activating function of the hyperbolic tangent, which calculates the same function as the first network.

$$a_h(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

We will show that $\tanh isrescaledsigmoidfunction : \tanh(x) = \sinh(x)\cosh(x)$

$$\sinh(x) = \frac{1}{2}(e^x - e^{-x}) \text{ and } \cosh(x) = \frac{1}{2}(e^x + e^{-x})$$

$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} =$
$\frac{\frac{1}{2}(e^x - e^{-x})}{\frac{1}{2}(e^x + e^{-x})} =$
$\frac{(e^x - e^{-x})e^{-x}}{(e^x + e^{-x})e^{-x}} =$
$\frac{(1-e^{-2x})}{(1+e^(-2x))} =$
$= \frac{(1+1-1-e^{-2x})}{(1+e^(-2x))} =$
$= \frac{2-(1-e^{-2x})}{(1+e^(-2x))} =$
$= \frac{2}{(1+e^(-2x))} - 1 =$
$= 2 \cdot \frac{1}{(1+e^(-2x))} - 1 =$

So, as $sigma(x) = \frac{1}{1+e^{-x}}$, then $sigma(2x) = \frac{1}{1+e^{-2x}}$. So,
$\tanh(x) = 2 \cdot \frac{1}{(1+e^(-2x))} - 1 = 2sigma(2x) - 1$

Conclusion: *tanh* is a rescaled *sigmoid* function. So acctually they give the same result over the function calculation.
Let us take a function that has two inputs and one hidden layer $x_1$ and $x_2$. In

neural net first we do a summation and then activation function.
Let us see an example: function: XOR, input: $x_1$, $x_2$, two hidden neurons, one hidden layer and one output.
Let us suppose that $\theta_1$ and $\theta_2$ are the parameters, so let's set $\theta_1 = \theta_2 = 20$ and set the bias = 10, then $\sigma(20x_1 + 20x_2 - 10) = \frac{1}{1+e^{-20x_1-20x_2+10}}$.
Input $x_1$ and $x_2$: $x_1 = 0$ and $x_2 = 0$:
$\sigma(20 * 0 + 20 * 0 - 10) = \sigma(-10) = \frac{1}{1+e^(10)} = 0$ $x_1 = 0$ and $x_2 = 0$:
$\tanh(20 * 0 + 20 * 0 - 10) = \tanh(-10) = 2 * \sigma(-10) - 1 = 2 * \frac{1}{1+e^(10)} - 1 = 0 - 1 = -1$.
As we know, tanh scale is [-1; 1] and $\sigma \in [-1; 1]$, so value -1 for tanh and 0 for $\sigma$ is the same.
Next, we can prove this wiyth the math induction.