Exam (8 April 2021) Attempt review

Machine Learning 1 (Technische Universität Berlin)

Started on	Thursday, 8 April 2021, 8:30 AM
State	Finished
Completed on	Thursday, 8 April 2021, 10:30 AM
Time taken	1 hour 59 mins
Grade	**50.00** out of 100.00

Information

Eigenständigkeitserklärung / Declaration of Independence

By proceeding further, you confirm that you are completing the exam alone, without other resources than those that are authorized. Authorized resources include the ML1 course material, personal notes, online APIs documentation, and calculation/plotting tools.

Question **1**
Incorrect
Mark 0.00 out of 5.00

Which of the following is **True**: A Gaussian Process (GP):

- a. defines a multivariate Gaussian distribution over output variables, with covariance determined by input similarity.
- b. defines a multivariate Gaussian distribution over input variables, with covariance determined by output similarity.
- c. defines a multivariate distribution over output variables, with input drawn from a Gaussian distribution. ✖
- d. defines a multivariate Gaussian distribution over input variables.

Your answer is incorrect.

The correct answer is:
defines a multivariate Gaussian distribution over output variables, with covariance determined by input similarity.

Question **2**
Correct
Mark 5.00 out of 5.00

Which of the following is **True**: In learning theory, the VC (Vapnik-Chervonenkis) bound:

- a. Is an upper-bound to the generalization error of a trained ML classifier of any complexity.
- b. Is a lower-bound to the generalization error of a trained ML classifier of any complexity.
- c. Is an upper-bound to the generalization error of a trained ML classifier of limited complexity. ✔
- d. Is a lower-bound to the generalization error of a trained ML classifier of limited complexity.

Your answer is correct.

The correct answer is:
Is an upper-bound to the generalization error of a trained ML classifier of limited complexity.

Question **3**
Correct
Mark 5.00 out of 5.00

Which of the following is **True**: k-means:

- a. Is a supervised learning algorithm similar to k-nearest neighbors.
- b. Has a convex objective and always converges to the global optimum.
- c. Learns a solution that depends on the initialization. ✔
- d. Is a supervised learning algorithm for representation learning.

Your answer is correct.

The correct answer is:
Learns a solution that depends on the initialization.

Question **4**
Incorrect
Mark 0.00 out of 5.00

Which of the following is **True**: A Product of Experts:

- a. Is an extension of a mixture model where each mixture element is forced to be Gaussian.
- b. Is an extension of a mixture model where each mixture element can be Gaussian with non-isotropic covariance.
- c. Learns less local features than a mixture model.
- d. Is an extension of a mixture model where each mixture element can be non-Gaussian with isotropic covariance. ✘

RBM、PoE 专注于整体结构
k-means more localized

Your answer is incorrect.

The correct answer is:
Learns less local features than a mixture model.

Information

Assume you would like to build a neural network that implements some function $f : \mathbb{R}^d_+ \to \mathbb{R}$ mapping inputs assumed to be positive to a real-valued output. For this, you have at your disposal neurons of the type

$$a_j = \max\left(0, \sum_i a_i w_{ij} + b_j\right)$$

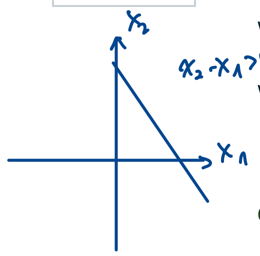
where $\sum_i$ sums over the indices of the incoming neurons, and zero otherwise. Denote by a_1 and a_2 the two input neurons (initialized to the value x_1 and x_2 respectively and which are always positive). Denote by a_3, a_4 the hidden neurons, and by a_5 the output neuron.

Question **5**
Complete
Mark 0.00 out of 5.00

Give the weights and biases associated to a neural network with the structure above and that implements the function $f(x) = \max(x_1, x_2)$.

w13 = 0.5, w23 = -0.5, b3 = 1
w14 = -0.5 , w24 = 0.5 , b4 = 1

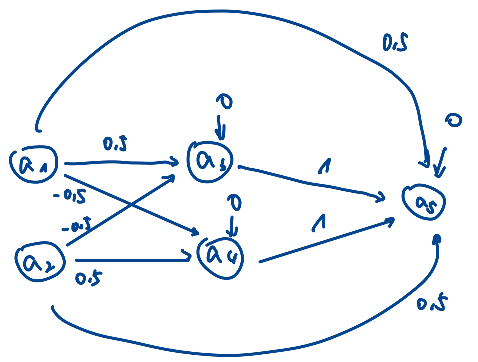
w35 = 1 , w45 = 1, b5 = (x1+x2)/2



$x_2 - x_1 > y$

$x_2 - x_1 < y$

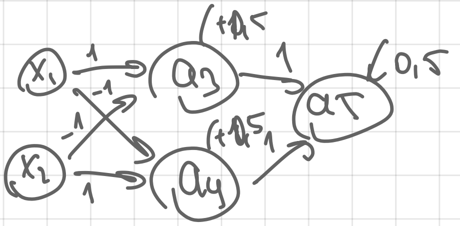
Comment: $x_1 - x_2 + y > 0$



$a_5 = \overbrace{(a_1 + a_2)}^{0.5} + \mathrm{Max}(0.5a_1 - 0.5a_2) + \mathrm{Max}(-0.5a_1 + 0.5a_2)$

$\frac{\partial a_5}{\partial a_1} = 0.5 + 0.5 \cdot 1_{a_1 > a_2} - 0.5\, 1_{a_1 < a_2}$

5)



$f: \mathbb{R}^d_+ \to \mathbb{R}$ $\quad f(x) = \max(x_1, x_2)$

$a_j = \max(0, \sum_i a_i w_{ij} + b_j)$

case: $x_1 < x_2$ in $(1, 2)$ $\quad f(x) = 2$

$$\begin{aligned} a_3 &= \max(0, x_1 w_{13} + x_2 w_{23} + b_3) \\ &= \max(0, 1 \cdot 1 + 2 \cdot (-1) + 0{,}5) \\ &= \max(0, 1 - 2 + 0{,}5) \\ &= \max(0, -0{,}5) \\ &= 0 \end{aligned}$$

$w_{13} = w_{24} = w_{35} = w_{45} = 1$

$w_{14} = w_{23} = -1$

$b_3 = b_4 = 0{,}5 = b_5$

$$\begin{aligned} a_4 &= \max(0, x_1 w_{14} + x_2 w_{24} + b_4) \\ &= \max(0, -1 + 2 + 0{,}5) \\ &= \max(0, 1{,}5) \\ &= 1{,}5 \end{aligned}$$

$$\begin{aligned} a_5 &= \max(0, a_3 + a_4 + b_5) \\ &= \max(0, 0 + 1{,}5 + b_5) \overset{!}{=} 2 \to b_5 = 0{,}5 \end{aligned}$$

case $x_1 > x_2$ in $(3, 1)$ $\to f(x) = 3$

$$\begin{aligned} a_3 &= \max(0, x_1 w_{13} + x_2 w_{23} + b_3) = \max(0, 3 \cdot 1 + 1 \cdot (-1) + 0{,}5) = \max(0, 3 - 1 + 0{,}5) \\ &= 2{,}5 \end{aligned}$$

$$a_4 = \max(0, x_1 w_{14} + x_2 w_{24} + b_4) = \max(0, -3 + 1 + 0{,}5) = \max(0, -1{,}5) = 0$$

$$a_5 = \max(0, a_3 + a_4 + b_5) = \max(0, 2{,}5 + 0 + b_5) \overset{!}{=} 3 \to b_5 = 0{,}5$$

6) each input dimension requires 2 neuron, so we need 2d hidden neurons. +1 output neuron

7) $$\frac{\partial a_5}{\partial x_1} = \frac{\partial a_5}{\partial z_5} \cdot \frac{\partial z_5}{\partial a_3} \cdot \frac{\partial a_3}{\partial z_3} \cdot \frac{\partial z_3}{\partial x_1} + \frac{\partial a_5}{\partial z_5} \cdot \frac{\partial z_5}{\partial a_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial x_1}$$

$$= w_{35} \cdot w_{13} + w_{45} \cdot w_{14} \quad \text{in } (2,3)$$

$$1 \cdot 1 + 1 \cdot (-1) = 0$$

Question **6**
Complete
Mark 5.00 out of 5.00

Explain what would be the minimum number of required hidden neurons if not taking two dimensions as input, but d dimensions, and replacing x_1, x_2 by $x_1, x_2, \dots, x_d$ in the formula above. In this exercise, you can consider architectures of any depth, where neurons can be connected in an arbitrary fashion, but where the nonlinear activation function is of the type mentioned above.

We would need d hidden neurons. Each input dimension requires one hidden neuron and every input variable is connected to each hidden layer. There is no need to build a more complex network with increasing dimensions.

d hidden neurons. fully connection between input layer and output layer

Comment:

Question **7**
Complete
Mark 5.00 out of 5.00

Assume you observe the data point $\boldsymbol{x} = (2, 3)$. Give the value of the partial derivative $\partial a_5 / \partial x_1$ for this data point.

0

Comment:

Information

A kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is positive semi-definite (PSD) if for any sequence of N data points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N \in \mathbb{R}^d$ and real-valued scalars $c_1, \dots, c_N$, the following inequality holds:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$$

In the ML1 course, various kernels have been shown to be PSD, for example, the linear kernel is PSD, a sum of two PSD kernels is PSD, a product of two PSD kernels is PSD, etc.

$$\sum_i \sum_j c_i c_j \left(\alpha + (1-\alpha)\langle x, x'\rangle\right)$$

$$\sum_i \sum_j c_i c_j \left(\alpha + (1-\alpha) \sum_k x_k x'_k\right)$$

$$\sum_i \sum_j c_i c_j \alpha + \sum_k (1-\alpha) \sum_i c_i x_k \sum_j c_j x'_k$$

Question **8**
Complete
Mark 5.00 out of 5.00

Consider the kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = \alpha + (1 - \alpha)\langle \boldsymbol{x}, \boldsymbol{x}' \rangle$$

$$\alpha \left(\sum_i c_i\right)^2 + (1-\alpha) \sum_k \left(\sum_i c_i x_k\right)^2 \geq 0$$

where α is a parameter of the kernel.

Give the conditions on the parameter α for which the kernel k is positive semi-definite.

$$\begin{cases} \alpha \geq 0 \\ 1-\alpha \geq 0 \end{cases}$$

$$0 \leq \alpha \leq 1$$

sum_i{ sum_j{ cicj(alpha+(1-alpha)<x,x'> }}

=alpha sum_i{ sum_j{ cicj}} + (1-alpha) sum_i{ sum_j{ cicjxixj}}

=alpha sum_i {(ci)^2} + (1-alpha) sum_i{(cixi)^2} >= 0

the term inside the sum are both quadratic hence ≥ 0. to ensure that the kernel is positive semi-definite, alpha needs to be between 0 and one such that 0 ≤ a ≤ 1 and 0 ≤ 1-a ≤ 1

Comment:

8) $\alpha \geq 0$ and $(1-\alpha) \geq 0 \rightarrow 1 \geq \alpha \rightarrow \boxed{0 \leq \alpha \leq 1}$

$$\sum_i \sum_j c_i c_j k(x_i, x_j) \geq 0$$

$$\alpha \left(\sum_i \sum_j c_i c_j\right) + (1-\alpha)\left(\sum_i \sum_j c_i c_j \langle x_i, x_j \rangle\right) \geq 0$$

$$\alpha \left(\sum_i c_i\right)^2 + (1-\alpha)\left(\sum_{i,j} c_i c_j \langle x_i, x_j \rangle\right) \geq 0$$

$\alpha = [0, 1]$

9) $$\|\phi(x) - \phi(x')\|^2 = (\phi(x) - \phi(x'))^T (\phi(x) - \phi(x'))$$
$$= \phi(x)^T\phi(x) - \phi(x)^T\phi(x') - \phi(x')^T\phi(x) + \phi(x')^T\phi(x')$$
$$= \phi(x)^T\phi(x) - 2\,\phi(x)^T\phi(x') + \phi(x')^T\phi(x')$$
$$= k(x, x) - 2k(x, x') + k(x', x')$$

$$\left.\begin{aligned} k(x,x) &= \alpha + (1-\alpha)\langle(1,0),(1,0)\rangle = \alpha + (1-\alpha)(1) = 1 \\ k(x,x') &= \alpha + (1-\alpha)\langle(1,0),(0,1)\rangle = \alpha \\ k(x',x') &= \alpha + (1-\alpha)\langle(0,1),(0,1)\rangle = 1 \end{aligned}\right\} \begin{aligned} &1 - 2\alpha + 1 \\ &= -2\alpha + 2 \\ &= 2(-\alpha + 1) \end{aligned}$$

10) $\alpha = 1 \rightarrow \phi(x) = \begin{pmatrix} \sqrt{\alpha} \\ \sqrt{1-\alpha}\, x \end{pmatrix}$

11) $\phi(x) = \begin{pmatrix} \sqrt{-1} \\ \sqrt{0}\, x \end{pmatrix}$ can't be in the feature space because $\sqrt{-1} \notin \mathbb{R}$ and $0 \leq \alpha \leq 1$

$$k(\mathbf{x}, \mathbf{x}') = \alpha + (1-\alpha)\langle \mathbf{x}, \mathbf{x}' \rangle$$

Condition of α for k to be PSD

That is

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \, k(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$

We first write

$$\begin{aligned} 0 &\leq \alpha\Big(\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \Big) + (1-\alpha)\Big(\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \, \langle \mathbf{x}_i, \mathbf{x}_j \rangle \Big) \\ &\leq \alpha\Big(\sum_{i=1}^{n} c_i \Big)^2 + (1-\alpha)\Big(\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \, \langle \mathbf{x}_i, \mathbf{x}_j \rangle \Big) \end{aligned}$$

Thus, $\alpha = [0, 1]$.

Squared Distance in feature space $\phi(\cdot)$ between $\mathbf{x} = (1, 0)$ and $\mathbf{x}' = (0, 1)$

As k satisfies the Mercer's condition (symmetric and PSD), we know that

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$$

To compute the squared distance in the feature space $\phi(\cdot)$, we first

$$\begin{aligned} \|\phi(\mathbf{x}) - \phi(\mathbf{x}')\|^2 &= \Big(\phi(\mathbf{x}) - \phi(\mathbf{x}')\Big)^T \Big(\phi(\mathbf{x}) - \phi(\mathbf{x}')\Big) \\ &= \phi(\mathbf{x})^T \phi(\mathbf{x}) - 2\phi(\mathbf{x})\phi(\mathbf{x}') + \phi(\mathbf{x}')^T \phi(\mathbf{x}') \\ &= k(\mathbf{x}, \mathbf{x}) - 2k(\mathbf{x}, \mathbf{x}') + k(\mathbf{x}', \mathbf{x}'). \end{aligned}$$

Evaluating the terms, we get

- $k(\mathbf{x}, \mathbf{x}) = \alpha + (1-\alpha)(1) = 1$
- $k(\mathbf{x}, \mathbf{x}') = \alpha + (1-\alpha)(0) = \alpha$
- $k(\mathbf{x}', \mathbf{x}') = \alpha + (1-\alpha)(1) = 1$

Thus,

$$\begin{aligned} \|\phi(\mathbf{x}) - \phi(\mathbf{x}')\|^2 &= 2(1) - 2(\alpha) \\ &= 2(1-\alpha) \end{aligned}$$

Explicit feature map when $d = 1$

$$\phi(x) = \begin{bmatrix} \sqrt{\alpha} \\ \sqrt{(1-\alpha)}\, x \end{bmatrix}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = \alpha + (1-\alpha)\langle \boldsymbol{x}, \boldsymbol{x}' \rangle$$

Question **9**
Complete
Mark 0.00 out of 5.00

A kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that is symmetric and positive semi-definite typically induces a feature map $\phi : \mathbb{R}^d \to \mathbb{R}^h$, that relates to the kernel as $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$.

Consider the kernel defined in the question above, and assume that the number of input dimensions is $d = 2$. *Compute* the square distance in feature space between the point $\boldsymbol{x} = (1, 0)$ and $\boldsymbol{x}' = (0, 1)$, i.e. compute

$$\|\phi((1,0)) - \phi((0,1))\|^2. = (\phi(x) - \phi(x'))^T(\phi(x) - \phi(x'))$$

$$= \phi^T(x)\phi(x) - 2\phi^T(x)\phi(x') + \phi^T(x')\phi(x')$$

$$= k(x,x) - 2k(x,x') + k(x',x')$$

$$= (1-\alpha)(\langle x,x \rangle - 2\langle x,x' \rangle + \langle x',x' \rangle)$$

$$= (1-\alpha)(1+1) = 2(1-\alpha)$$

(Hint: You may rewrite this quantity in terms of kernel computations).

|| alpha + (1-alpha)(1+0) - (alpha + (1-alpha)(0+1)) || = || 0 ||

0

Comment:

Question **10**
Complete
Mark 0.00 out of 5.00

Consider now the case where $d = 1$ (i.e. one-dimensional input data). *Give* an explicit feature map $\phi : \mathbb{R} \to \mathbb{R}^h$ associated to the kernel k.

$$\alpha + (1-\alpha)xx' = \langle \phi(x), \phi(x') \rangle$$

alpha +(1-alpha)x^2

$$\phi(x) = \begin{bmatrix} \sqrt{\alpha} \\ \sqrt{1-\alpha}\, x \end{bmatrix}$$

Comment:

Question **11**
Complete
Mark 3.00 out of 5.00

For the feature map you have found in the question above, *give* a point in the feature space $\mathbb{R}^h$ that has no pre-image in the input space, i.e. for which it is impossible to find a corresponding data point x.

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

-1

the above mentioned feature map cannot be < 0 since alpha +(1-alpha)x^2 since for every x and alpha where 0 < alpha < 1 the result is >= 0.

Comment:
answer is correct w.r.t to the answer provided for Q10, gives 3 points

$w_1^T x - 1 \geq 0$

$1 - w_1^T x \leq 0$ $\quad$ $1 + w_2^T x \leq 0$

Question **12**
Complete
Mark 2.50 out of 5.00

Consider the optimization problem

$$\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{x}\|^2 \quad \text{subject to} \quad \boldsymbol{w}_1^\top \boldsymbol{x} \geq 1 \quad \text{and} \quad \boldsymbol{w}_2^\top \boldsymbol{x} \leq -1$$

Viewing $\boldsymbol{x} \in \mathbb{R}^d$ as the input and $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^d$ as two feature detectors, *give* a high-level interpretation of such optimization problem, as well as an interpretation of Slater's conditions for duality.

这个优化问题的目标是**最小化** x 的范数，即找到**离原点最近**的可行解。约束条件要求 x **必须投影到两个方向**（w_1, w_2）**上，并满足一定的间隔**。这类似于**支持向量机（SVM）中的优化问题，其中** w_1, w_2 **可以视为特征检测器**。几何上，解 x^* 位于两个约束超平面之间，通常是边界上的某个点。

关于 **Slater 条件**，由于约束是线性的，并且两个超平面通常不会重合，所以**存在严格可行点**，即满足 $w_1^\top x > 1, w_2^\top x < -1$ 的 x。因此，Slater 条件成立，确保了强对偶性，我们可以使用拉格朗日对偶方法求解该问题。

The objective of this optimization problem is to **minimize** the norm of x, meaning we seek the **closest feasible solution to the origin**. The constraints require **x to project onto two directions (w_1, w_2) while maintaining a certain margin**. This is similar to **Support Vector Machines (SVMs)**, where w_1 and w_2 act as **feature detectors**. Geometrically, the optimal solution x^* lies between the two constraint hyperplanes, often on the boundary.

Regarding **Slater's condition**, since the constraints are linear and the two hyperplanes typically do not coincide, there exists a **strictly feasible point** satisfying $w_1^\top x > 1, w_2^\top x < -1$. Thus, Slater's condition holds, ensuring **strong duality**, and we can solve the problem using **Lagrangian duality**.

condition. If we can satisfy the constraints wi
be also written by its Lagrange dual formulati

Then the strong duality holds

Comment:

Question **13**
Complete
Mark 2.50 out of 5.00

Rewrite the optimization problem above in the form of

$$\max_{\boldsymbol{\alpha}} \left\{ \min_{\boldsymbol{x}} \{ \dots \} \right\}$$

where $\boldsymbol{x}$ and $\boldsymbol{\alpha}$ are the primal and dual optimization variables respectively.

argmax over alpha{argmin over x{0.5*||x||^2 +alpha*(|w'x|-1)}}

dual func: $g(x,\alpha) = \min_x L(x, \alpha_1, \alpha_2) = \frac{1}{2}\|x\|^2 + \alpha_1(1 - w_1^T x) + \alpha_2(w_2^T x + 1)$ $= \frac{1}{2}\|\alpha_1 w_1 + \alpha_2 w_2\|^2 + \alpha_1 + \alpha_2 - \alpha_1 w_1^T x + \alpha_2 w_2^T x^*$

$\frac{\partial L}{\partial x} = x - \alpha_1 w_1 + \alpha_2 w_2 = 0 \Rightarrow x^* = \alpha_1 w_1 + \alpha_2 w_2$

$\frac{\partial L}{\partial \alpha_1} = 0 \quad \frac{\partial L}{\partial \alpha_2} = 0$

optim Problem for dual problem: $\max_{\alpha_1, \alpha_2} g(x,\alpha) = \max_\alpha \min_x L(x, \alpha_1, \alpha_2)$

s.t. $\alpha_1 \geq 0 \quad \alpha_2 \geq 0$

Comment:

Question **14**
Complete
Mark 5.00 out of 10.00

Give the dual optimization problem associated to the primal optimization problem above. In particular, *state* the objective and the constraints on the dual optimization variables.

L = 0.5*||x||^2 +alpha(|w'x|-1) $\quad L = 0.5\|x\|^2 + \alpha(|w'x - 1|)$

derivative wrt w: x - alpha*w = 0 -> x = alpha*w

set in L: argmax over alpha{ 0.5*|| alpha*w||^2 +alpha(|w*alpha*w|-1) }

objective 0.5*|| alpha*w||^2 constraint |w*alpha*w| >= 1

Comment: $\frac{\partial L}{\partial x} \stackrel{!}{=} 0 \rightarrow x + \lambda w_1^T - \beta w_2^T \stackrel{!}{=} 0 \rightarrow x = -\lambda w_1^T + \beta w_2^T$

$\frac{\partial L}{\partial w} \stackrel{!}{=} 0 \rightarrow \lambda x - \beta x \stackrel{!}{=} 0 \rightarrow \lambda = \beta$

Question **15**

Complete

Mark 5.00 out of 5.00

For this specific problem, *explain* in one or two sentences when (or whether) the dual formulation should be preferred over the primal formulation.

Primal is preferred when N is very large(>> data points). Dual is better to use if the dimension D is very large.

Comment:

The **dual formulation** is preferred when the number of constraints is much smaller than the number of primal variables (i.e., when d is large). This is because solving the dual problem involves optimizing over a lower-dimensional space, making it computationally more efficient. However, if d is small or the number of constraints is large, solving the **primal problem** directly may be more straightforward.

当约束的数量远小于原始变量的数量时（即 whenis large），则首选对偶公式。这是因为解决对偶问题涉及在较低维空间上进行优化，从而提高计算效率。但是，如果很小或者 constraints 的数量很多，直接解决 theprimal problem 可能更直接。

Information

Consider a supervised dataset $(\boldsymbol{x}_1, t_1), \dots, (\boldsymbol{x}_N, t_N)$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $t_i \in \mathbb{R}$. The input data is stored in a matrix of size $N \times d$, and the corresponding outputs (targets) are stored in a vector of size N.

We consider a kernel-based regression model given by:

$$f(\boldsymbol{x}) = \frac{\sum_{i=1}^N k(\boldsymbol{x}, \boldsymbol{x}_i) \cdot t_i}{\sum_{i=1}^N k(\boldsymbol{x}, \boldsymbol{x}_i)}$$

and we would like to implement it.

Your implementation should take the form of a *function* that receives four arguments,

1. a matrix called Xtrain containing the training data points,
2. a corresponding vector of targets called Ttrain,
3. the hyperparameter(s) of the kernel,
4. a matrix of test points you would like to predict called Xtest.

Your function should return the vector of predicted values for points in Xtest. Your implementation should be *efficient*, i.e. make use of numpy/scipy vector and matrix operations when possible, and avoid redundant computations. *(Hint: you may make use of the function scipy.spatial.distance.cdist).*

Question **16**

Complete

Mark 7.00 out of 10.00

Implement the regression model defined above when the kernel function is given by $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma \cdot \|\boldsymbol{x} - \boldsymbol{x}'\|^2)$.

```
getKernel(X1,X2,gamma):
D2 = scipy.spatial.distance.cdist(X1,X2,"sqeuclidean")
K = numpy.exp(-gamma*D2)
return K



def regression(X1,X2,t):

return getKernel(X1,X2,gamma).dot(t)/getKernel(X1,X2,gamma)
```

Comment:

```
def regression(Xtrain,Ttrain,gamma,Xtest):
    K= getKernel(Xtest,Xtrain,gamma)    # (M,N)

    return K@Ttrain/np.sum(K,axis=1)
```

Question **17**
Complete
Mark 0.00 out of 10.00

Write a function that selects the best kernel hyperparameter for the model above. Use for this a holdout validation procedure where the training data is randomly split into 80% training and 20% validation, where candidate hyperparameters are spaced logarithmically between 10^{-5} and 10^{5}, and where the mean square error is used as a selection criterion.

```
Xtrain,Xtest = X[R[:len(R)//1.6]]*1,X[R[len(R)//1.6:]]*1 #80%
Ttrain,Ttest = T[R[:len(R)//0.8]]*1,T[R[len(R)//0.8:]]*1 #20%
from sklearn_model_selection import train_test_split
from sklearn.metrics import mean squared_error
X_train, X_val, T_train, T_val = train_test_split ( X, T, test_size =0,2)
gamma_values = np.logspace (-5,5, 20)
return min (gamma_values, key = lamda g: mse(T_val, regression(Xtrain, Ttrain, g, Xtest ) )
```

Comment:

```
def mse ( y_true, y_pred):
    return np.mean( (y_true - y_pred) ** 2 )
```

```
def train_val_split(X, T, test_size=0.2):
    N = len(X)
    idx = np.random.permutation(N)
    split = int(N * (1 - test_size))
    return X[idx[:split]], X[idx[split:]], T[idx[:split]], T[idx[split:]]
```