HW5

ALESSANDRO MANZOTTI

1. Problem 1

1.1. **1a.** For a psd matrix we have that $\mathbf{v}^t A\mathbf{v} \geq 0$. Now being A a symmetric matrix we can diagonalize it. For the spectral theorem its eigenvector $\mathbf{u}_i$ span the space. So let's write the vector $\mathbf{v}$ in that basis $\mathbf{v} = \sum_i \alpha_i \mathbf{u}_i$ where $\mathbf{u}_i$ is the eigevector of A of eigenvalue λ_i. So $\mathbf{v}^t A\mathbf{v} = \sum_i \alpha_i^2 \lambda_i \geq 0$ where λ_i are the eigenvalues of the matrix. So from this we have $\mathbf{v}^t A\mathbf{v} \geq 0$ iff $\lambda_i \geq 0$ since α_i^2 are positive numbers.

1.2. **2b.** By definition, if the kernel is positive the corresponding Gram matrix is a semidefinite positive matrix. The elements $k(x, x)$ for any x correspond to its diagonal elements. But for psd matrix all diagonal entries are ≥ 0. This can be seen from its definition using $\mathbf{e}_i^t A\mathbf{e}_i \geq 0$ where $\mathbf{e}_i$ is the unit vector with 1 in position i and zeros in all the others.

1.3. **2c.** Remember the connection between the psd definition of a function and the corresponding Gram matrix. Now consider two matrices:

$$\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix}$$

This is symmetric and all the entries are positive. So if this is a Gram matrix of a kernel k $k(x, x') > 0$. However this is not psd (for example its eigenvalues are not $\geq$0)

Now consider the Gram matrix

$$\begin{bmatrix} 1 & -1 \\ -1 & 4 \end{bmatrix}$$

this has $k(x_1, x_2) < 0$ but it can be shown to be psd (just with a general vector $\mathbf{v} = (x, y)$ and see the componenets appears only squared in $\mathbf{v}^t A\mathbf{v}$).

1.4. **2d.** Since k' is a kernel on $\mathcal{X}'$ we have $\sum_i \sum_j \alpha_i \alpha_j k(\phi(x), \phi(x'))$, indeed every elements of $\mathcal{X}'$ can be written using the function from $\mathcal{X}$.

Fix $s_1 ... s_n \in \mathcal{X}'$ and scalars $\alpha_1 ... \alpha_n$ and let $\{x_1 ... x_p\} = \{\psi(s_1)..\psi(s_n)\}$ with $p \leq n$. With $A_k = \{i : \psi(s_i) = x_k\}$ and $\beta_k = \sum_{i \in A_k} \alpha_i$.

Then $\sum_{i,j} \alpha_i \alpha_j K(\psi(s_i), \psi(s_j)) = \sum_{k,l} \sum_{i \in A_k} \sum_{j \in A_l} \alpha_i \alpha_j K(x_k, x_l) = \sum_{k,l} \beta_k \beta_l K(x_k, x_l) \geq 0$ since $K(x, x')$ is an kernel on $\mathcal{X}'$ and so $K(\psi(x), \psi(x'))$ is a kernel on $\mathcal{X}$

1.5. **2e.** The properties that $d(u, u') \geq 0$ with $= 0$ only if $u = u'$ and $d(u, u') = d(u', u)$ and $d(x, y) \leq d(x, z) + d(z, y)$(using Cauchy-Schwarz) comes directly from the properties of the norm on an Hilbert space.

We have $d(\phi(x), \phi(x'))^2 = \|\phi(x) - \phi(x')\|^2$. Because of the property of RKHS we have $\|\phi(x) - \phi(x')\|^2 = < \phi(x) - \phi(x'), \phi(x) - \phi(x') >$, by linearity $< \phi(x), \phi(x) > + < \phi(x'), \phi(x') > -2 < \phi(x), \phi(x') > = k(x, x) + k(x, x') - 2k(x, x')$.

So

$$d(\phi(x), \phi(x'))^2 = k(x, x) + k(x, x') - 2k(x, x') \tag{1}$$

In the same way

$$k(x, x') = (d(\phi(x), \phi(x'))^2 - d(\phi(x), \phi(x))^2 - d(\phi(x'), \phi(x'))^2)/2 \tag{2}$$

2. Problem 2

2.1. **2a.** Let K1, K2 be the kernel Gram matrices of k1 and k2 and these matrices are PSD on $\mathcal{X}$. Also let α be any vector. we have:

$$K = K1 + K2 \rightarrow \alpha^T K\alpha = \alpha^T K1\alpha + \alpha^T K2\alpha \geq 0$$

2.2. **2b.** This can be prove by simple concatenation: Let $X = (x_1, x_2)^T$, i.e. the concatenation of the x_1 vector with the x_2 vector. then: K(x, y) =

$$\begin{aligned} < X, X' > &= \left\langle \begin{pmatrix} x_1 \\ x_1' \end{pmatrix} \begin{pmatrix} x_2 \\ x_2' \end{pmatrix} \right\rangle \\ &=< x_1, x_1' > + < x_2, x_2' > \\ &= K1(x_1, x_1') + K2(x_2, x_2') \end{aligned}$$

2.3. **2c.** Let's define $u = \mathbf{x}/\|\mathbf{x}\|$. Then we can write the cosine kernel gram matrix as $Ki, j =< u_i, u_j >$ so it can be written as $A^T A$ where A is a $n \times n$ matrix whose kth column is the vector $\mathbf{u_k}$. It follows that

$$< Kx, x >=< A^t Ax, x >= \|Ax\| \geq 0 \tag{3}$$

whence K is positive semi-definite.

3. Problem 3

The vanilla k-nearest neighborhood is simply based on a definition of a distance in the features space. Once this has been set the closest points (in the training set) to a testing one can be found and a classification can be done using a majority vote.

We can implement a richer set of features by using a map $\phi : \chi \rightarrow H$. The only thing we need in the new Hilbert feature space is a notion of metric. Using results 1e we can see that the distance in the new space exists and can be computed by using the kernel in the starting feature space using

$$d(\phi(x), \phi(x')) = k(x, x) + k(x, x') - 2k(x, x') \tag{4}$$

A possible pseudo code could be

(1) Train on the training set: once we have defined the kernel we want we calcualte $k(x, x)$ for all the elements of the training set, for example $x^t x$ for a boring linear kernel
(2) to classify a point x' we compute $k(x, x')$ and k(x',x') and from this the distance $d(\phi(x), \phi(x')) = k(x, x) + k(x, x') - 2k(x, x')$.
(3) we can sort the distances, and issue a majority vote among the N(parameter chose by the users) closest point in the training set

4. Problem 4

4.1. **4a.** This kernel is based on a map ϕ_s that send the string (or a document) a into the number of time the substring s appears contiguously. "apple" for example contains twice the mono-string or letter p, once the length 2 string "le" and so on. So depending on the substring length it will be similar to words like "plea" that contains similar substring. For example, the $K_3('abccc','abc') = 1$ because they both contains 'abc'. This kernel is clearly semi-positive since the lowest value is zero where none of the words a or b contains the same substring of length $k\ \bar{s}$ and positive when we have a match in both.

Computing the kernel correspond in finding the number of time every substring of length k in appears in a multiply by the times it appears in b. So this is close to the string search problem. We start by

(1) computing all the substring of lenght k in the first element a with their occurrence number.
(2) search this substring in and the number of time it occurs in b.

A possible way to compute the kernel can be seen as follow. For simplicity of notation, we describe the binary-valued version of the feature map, though the count-valued version is similar. For each sequence x, collect the set of k-length subseqences into an array A_x and sort them. Now the inner product $K_k(x, y)$ can be computed in linear time as a function of length(x) + length(y). We need to scan through the 2 list and counting the number of same object. Thus the overall complexity of computing the kernel value is $O(n \log(n))$ in the length of the input sequences using this method.

Suffix tree can be shown to be even faster.

4.2. **4b.** This kernel is very similar to the previous one with the difference that now the substring does not need to be contiguous and they can be gappy inside the target documents or word. For example string 'The cat was chased by the fat dog' can be seen to contain, among others, the following gapped substrings of length 3 "tea" "ted" "dog".

This definition does not make a difference between occurrences that contain few gaps and those that contain several gaps, or the lengths thereof; all contribute to the feature value equally. For example, the substring 'tea' will have a high weight as it occurs many times in the text, although it never occurs with fewer than two gaps and the total length of gaps is at least three. At the same time, 'dog' will have much smaller weight although it occurs in the text without any gaps.

i) In this case k=1 so we are basically looking for single letter occurrence in the 2 strings.

I would probably take the string a and b and order them alphabetically at this point for every I can go through each list and count the number of occurrence of each letter and multiply them.

ii) In this case if I understand well we are looking for find a matching substring of length p in words that have a length p.

I would use a merge-sort algorithm to sort a and b. Then I can go through the list (for i from zero to p) return zero if $sorted(a)[i] \neq sorted(b)[i]$ or return one if I go through the list without returning before the end. This might be $O(p \log p)$

An $O(p)$ way could be to count the frequency of each character in the two strings and check if the two histograms match.

1. Create an array of letters in alphabet initialized with 0?s. 2. For first string increment count of character in count array. 3. For second string decrement the count of character from count array. 4. Repeat steps 2 and 3 till we reach end of any string. 5. Check if array contains only zero then strings are anagram otherwise not.

Something like

```
function are_anagrams(string1, string2)

    let counts = new int[26];

    for each char c in lower_case(string1)
        counts[(int)c]++

    for each char c in lower_case(string2)
        counts[(int)c]--

    for each int count in counts
        if count != 0
            return 0

    return 1
```

5. Problem 5

See below hand written.

5) LETS START BY INTRODUCING THE SLACK VARIABLES

$$\min_{f} \left[\frac{1}{m} \sum_i (1 - f(x_i))_{\geq 0} + \frac{1}{2C} \|f\|^2 \right] \longrightarrow$$

① $$\min_{f;\, \xi_{i \dots m}} \frac{1}{2C} \|f\|^2 + \frac{1}{m} \sum_i \xi_i \qquad \text{s.t} \quad y_i f(x_i) \geq 1 - \xi_i \qquad \xi_i > 0$$

NOW BECAUSE OF THE REPRESENTER THEOREM WE KNOW THAT THE SOLUTION WILL BE OF THE FORM $f(x) = \sum_i \gamma_i \, k(x_i; x)$

SO (MULTIPLYING ALL BY C, STILL THE SAME MAX)

② $$\min \frac{1}{2\cancel{C}} \sum_i \sum_j \gamma_i \gamma_j k(x_i; x_j) + \frac{C}{m} \sum_i \xi_i \qquad \text{s.t} \quad y_i \sum_i \gamma_i \, k(x_i; x) \geq 1 - \xi_i \quad \xi_i > 0$$

THIS IS BECAUSE $\langle f; f \rangle_H = \langle \sum \gamma_i K_{x_i}; \sum \gamma_j K_{x_j} \rangle = \sum_i \sum_j \gamma_i \gamma_j \langle K_{x_i}; K_{x_j} \rangle_H$

$$= \sum_i \sum_j K(x_i x_j) \rightarrow$$

WHERE WE USE LINEARITY AND THE DEFINITION OF A RKHS

FROM ② USING LAGRANGIAN MULTIPLIERS (β, α)

$$\mathcal{L}(\beta; \xi; \alpha; \gamma) = \frac{C}{m} \sum_i \xi_i + \frac{1}{2} \gamma^t K \gamma - \sum \alpha_i y_i \left(\sum_j \gamma_j k(x_i x_j) - 1 + \xi_i \right) - \sum_i \beta_i \xi_i$$

WITH $\beta > 0$

DUAL IS $\operatorname{ARGMAX}_{\alpha; \beta > 0} \inf_{\gamma \xi} \mathcal{L}(\beta; \xi; \alpha \gamma)$

1) $$\frac{\partial \mathcal{L}}{\partial \gamma} = 0 \rightarrow \gamma_i = \alpha_i y_i$$

2) $$\frac{\partial L}{\partial \xi_i} = 0 \qquad \frac{C}{m} - \alpha_i - \beta_i = 0 \rightarrow 0 \leq \alpha_i \leq C/m$$

PLAGGING $\beta_i = \frac{1}{m} - \alpha_i$ IN $\mathcal{L}$ WE HAVE

$$\arg\max \inf_{\gamma} \mathcal{L}(\gamma; \alpha) - \frac{1}{2} \gamma^t K \gamma + \sum_i \alpha_i \left(1 - y_i \sum_j k(x_i; x_j) \gamma_j \right)$$

INDEED THE ξ_i FACTOR CANCELS

NOW WE PLUG IN $\gamma_i = \alpha_i Y_i$ AND WE GET FINALLY

$$\boxed{\underset{\alpha \geq 0}{\arg\max}\ L(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum \alpha_i Y_i K(x_i ; x_J) \alpha_J Y_J}$$

$$\downarrow$$

$$\frac{1}{2} \alpha^t \operatorname{diag}(Y) K \operatorname{diag}(Y) \alpha$$