Modelling with Kernels based on Gower Similarity

Laureline Pinault[†], **Supervisor:** Lluis Belanche[‡]

[†]ENS de Lyon, France
[‡]UPC, Barcelona, Spain

August 28, 2015

Abstract

Contents

1 Introduction

With the optimization and the complexification of available computer power and tools, artificial intelligence is being more and more used. Machine learning, and classifiers, branches of this domain, are especially useful to analyse big databases and to find connections or categories that might be really hard for the human eye alone to see. For example, analyzing which zones of the brain activate when somebody is reading a particular word can allow the association of activation of precise zones in the brain with the reading and the understanding of a precise type of words –one of the most well-known experiment being non-living versus living, i.e. which zones activate in the brain when someone is reading the name of an object, and when he is reading the name of an animal.

Databases for these types of experience are usually very large, and consist of two things: observations, and targets. Observations are the set of what has been measured; in our example on the brain, it would be measures taken by electrodes on the brain of a patient, while he has to read words, some being in the living category and some in the non-living. An observation would then be a measure of the voltage of the electrodes at a given time. Targets, often called classes, are what correspond to a given observation; for example, if a patient is reading the word 'cat' at a given time, the observation taken at that time will correspond to the target 'living'.We can schematize it with this figure:

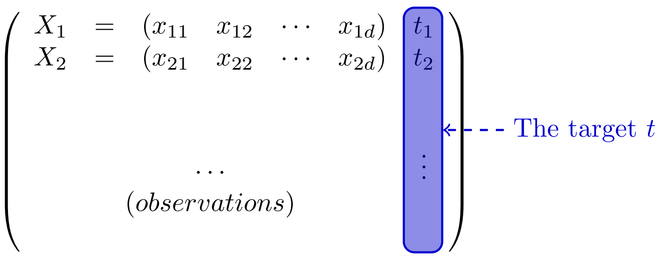

$X_1 = (x_{11} \quad x_{12} \quad \cdots \quad x_{1d})$
$X_2 = (x_{21} \quad x_{22} \quad \cdots \quad x_{2d})$
$\dots$
$(observations)$
t_1
t_2
$\vdots$
The target t

Figure 1: Architecture of data

The goal is to build a model to accurately predict the target for a given observation; when the target is numerical, it is called a regression and when it is categorical, a classification. In this document we mainly talk about classification but most of this work also apply to regression. Perfect prediction of targets from observations is usually near impossible because of three main causes; first, the universe of possible observations is far too large to be entirely computed and studied - otherwise a simple direct algorithm would suffice. Therefore, the data collected and studied are an incomplete and inaccurate representation of the possible observations. Secondly, the same observations could correspond to two different targets in our database but they of course would not be differenciable by a classifier; that's why, for an observation X, we don't have a function $t = f(X)$ but a joined distribution $\mathbb{P}(t, X) = \mathbb{P}(t|X) \cdot \mathbb{P}(X)$. If t is numerical, the model should be the expectation $t^*(X) = \int \mathbb{P}(t|X)dt$. If it is categorical, the model $t^*(X)$ should be the t such that $\mathbb{P}(t|X)$ is maximum. The error is then $1 - \mathbb{P}(t|X)$). Thirdly, a linear separation is usually impossible.

Traditionally, theory and algorithms of machine learning and statistics have been very well developed for the linear case $-i.e.$ when the data are separable by an hyperplane. Real world data analysis problem, on the other hand, often require nonlinear method to detect the kind of dependencies that allow successful prediction of interest.

Using a kernel function allows to embed the original space where the data are not linearly separable into a new space where, hopefully, they would be –this space is called the *feature space*. Mapping a problem into a space with a larger number of dimensions makes it more likely that the problem will become linearly separable. See Figure 2 below for an illustration.

The second idea behind the kernel trick is that it is usually very awkward and computationally expensive to work in a very high-dimensional space. However, if an algorithm only uses the dot products between points (which you can think of as distances), then you only have to work with a matrix of scalars. You can implicitly perform the calculations in the higher-dimensional space without ever actually having to do the mapping or handle the higher-dimensional data.

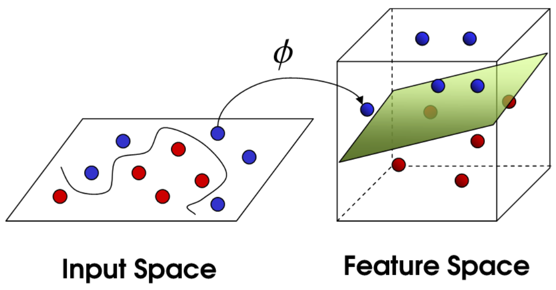


Figure 2: We map the input space into a higher dimensional space where the data are lieraly separable.

All of this theory, however, only work with homogeneous data: indeed heterogeneous data are hard to analyze. For example, let's say somebody is asking a bank for a loan. The bank wants to know whether or not they should loan the money to the customer, i.e. how likely the customer is to reimburse them. The variables the bank has to consider - and what the observations will consist of - are the duration of the loan, the loan amount, the profession of the customer, the purpose of the loan... Since the machine has to consider here both numerical and categorical data, this will of course induces a lack of precision when trying to fit all of the data into a homogeneous kernel.

With the development of the Internet, problems with more and more various and complex types of variables, such as trees - for example websites with a tree-like architecture - or pictures - for example recognizing what is represented on a picture - need to be resolved. This work aims to build a kernel able to work with problems with different data types. Following Gower's idea [Gow71], it is based on pre-existing kernels for each data type. We first develop kernels for several simple data types, thanks to similarity measures we prove to have the right properties. We then use these kernels to build the heterogeneous one described by Gower, and the areas of improvement available - and how to handle the missing values with our kernel. Finally, in the last sections we present the experiments we conducted and the results we obtained.

2 A Kernel for Heterogeneous Data

2.1 Kernel Functions, PSDness, and Links with Euclidean Distances

The goal of a kernel function being to embed the data in a space where they will be linearly separable, the ideal function puts together data with the same classes - also called targets. But we cannot use the classes themselves to construct the embedding because that's exactly what we are trying to guess when using our algorithm. Based on the idea that similar data should have the same class, a kernel function puts together what "looks alike". Basically, a kernel function is a measure of similarity of the input space. However, the opposite is not true since we also need the kernel function to correspond to a dot product in the feature space. This allows the use of linear machine learning methods based on dot products, without a high computational cost. We will now express more formally what a kernel function is and what needs a function to be a kernel. For more details on these aspects, see [HSS07].

Definition 2.1. Let $\mathcal{X}$ be a nonempty set. $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *kernel function* if it satifies

$$\forall x, x' \in \mathcal{X},\ k(x, x') = \langle \phi(x) | \phi(x') \rangle\,, \tag{1}$$

where ϕ maps $\mathcal{X}$ into some dot product space $\mathcal{H}$ called the *feature space.*

Theorem 2.1. *$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a* kernel function *if and only if $\forall n \in \mathbb{N}$, $\forall (x_i)_{i\leq n} \in \mathcal{X}^n$, the matrix $K = (k(x_i, x_j))_{i,j\leq n}$ is Positive Semi Definite –PSD –, meaning that $\forall c \in \mathbb{R}^n$, $c^T Kc \geq 0$.*

Proof. If k satisfies to 1, we can rewrite

$$c^T Kc = \sum_{i,j} c_i c_j k(x_i, x_j) = \sum_{i,j} c_i c_j \langle \phi(x_i) | \phi(x_j) \rangle = \left\langle \sum_i c_i \phi(x_i) \middle| \sum_j \phi(x_j) \right\rangle \geq 0$$

To prove the converse, we will construct explicitely the dot product space $\mathcal{H}$ –called the *reproducing kernel Hilbert space* –and the mapping ϕ.

We define a map from $\mathcal{X}$ into the space of functions mapping $\mathcal{X}$ into $\mathbb{R}$ denoted $\mathbb{R}^{\mathcal{X}}$: $\phi : \begin{cases} \mathcal{X} & \to \mathbb{R}^{\mathcal{X}} \\ x & \mapsto k(\cdot, x) \end{cases}$.

We then construct a dot product space containing the images of the inputs under ϕ by forming linear combination $f(\cdot) = \sum_{i=1}^n \alpha_i k(\cdot, x_i)$ and defining a dot product

$$\langle f | g \rangle = \left\langle \sum_{i=1}^n \alpha_i k(\cdot, x_i) \middle| \sum_{j=1}^n \beta_j k(\cdot, x'_j) \right\rangle = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \beta_j k(x_i, x'_j)$$

□

Below, we cites some properties of PSD matrices which will help showing PSDness properties of the kernels in our work.

Proposition 2.1 (Characterization of PSD Matrices)**.** *Let A be a real symmetric matrix. The following statements are equivalent:*

1. *A is PSD, i.e. for all real vector $\mathbf{x}$, $\mathbf{x}^t A\mathbf{x} \geq 0$.*
2. *All eigenvalues of A are non-negative.*
3. *$\det(A_I) \geq 0$ for all $I \subset \{1, \ldots, n\}$, where A_I denotes the restriction of A to the rows and columns indexed by I.*
4. *$A = Z^t Z$ for some real matrix Z.*

Corollary 2.1. *Let A be a real symmetric matrix. Then A is PSD if and only if all its sub-matrix A_I are PSD.*

Proposition 2.2 (Symmetric operations on PSD matrices)**.** *Symmetric operations on a matrix are:*

1. *Multiplying both the i-th row and i-th column by $\lambda \neq 0$.*
2. *Swapping the i-th and j-th column; and swapping the i-th and j-th row.*
3. *Adding $\lambda \times i$-th column to j-th column; and adding $\lambda \times i$-th row to j-th row.*

$A \cong B \Leftrightarrow B$ is obtained from A by zero or more symmetric matrix operations. Let A, B be real and symmetric. If $A \cong B$, then A is PSD if and only if B is PSD.

Proposition 2.3 (Closure operations on PSD matrices)**.** *Let A, B be real and symmetric. If both A and B are PSD, then:*

1. *$A + B$ is PSD.*
2. *$A * B$ is PSD, where $*$ designates the pointwise multiplication.*

Another interesting thing about PSD matrices is their link with euclidean distances –i.e. distances that can be represented in $\mathbb{R}^n$ for some n. Euclidean distances are interesting for vizualisation of data for example. Gower and Legendre explore this aspect in [GL86]. Among the properties they present, the following one will interest us for our work.

Proposition 2.4 ([GL86])**.** *Let $S = (s_{ij})_{i,j \leq n}$ be a similarity matrix such that $0 \leq s_{ij} \leq 1$ and $s_{ii} = 1$ for all $i, j \leq n$. Then S is P.S.D. implies that the dissimilarity matrix $D = (d_{ij})_{i,j \leq n} = (\sqrt{1 - s_{ij}})_{i,j \leq n}$ is Euclidean.*

Among the most widely used and well-known Kernels we find:

Lineal kernel $K(u, v) = \langle u|v \rangle$.

Polinomial kernel $K(u, v) = (\langle u|v \rangle + \gamma)^d$ with $\gamma \in \mathbb{R}$ and $d \in \mathbb{N}$ parameters.

Gaussian kernel also known as Radial Basis Function (RBF) kernel $K(u, v) = e^{-\frac{||u-v||^2}{\gamma^2}}$ with $\gamma \in \mathbb{R}$ parameter.

Sigmoid kernel $K(u, v) = \tanh(\alpha \langle u|v \rangle + r)$ for some (not every) $\alpha > 0$ and $r < 0$ parameters, where tanh is the hyperbolic tangent function.

The RBF is by far the most popular choice of kernel in Support Vector Machines. This is mainly because of its localized and finite response across the entire range of the real line; it also includes the polynomial Kernel as a limiting case. All these kernels assume and need the data set features to be continuous.

2.2 Kernels Based on Coefficients of Similarity for Various Data Types

In 1971 Gower proposed a kernel which generalized a coefficient of similarity for several types of variables [Gow71]. As the urge for treating more and more various data types (pictures, graphs, trees, ...) rises with the new sources of information, many researchers have been extending Gower's kernel to their own data types. Indeed, the strength of Gower's kernel is that it can be used to handle any kind of heterogeneous data with the only condition that you have a Kernel for each of your data types.

In this section we present kernels to treat the following simple data types:

Quantitative variables It is used for all observations for which the difference between two measures is meaningful. It includes continuous as well as integer measurements. For example, if you measure the length of a plant or the number of children per family, this would be the type of your variable.

Categorical ordinal variables It is used for all measurements which take value in an ordered set. The difference with the quantative variables is that the difference between two successive values is not a constant. Hence the difference of two values does not really make sense. For instance, the rank of runners is an ordinal variable because the first and the second runners might be very close but the third far away; however you only have their rankings, 1-2-3, in opposition to their times of arrival, which could have worked as quantitative variables. Another example is the pain scale, when someone is asked to rate his pain on a scale from 1 to 10. It is often interesting to rank these variables, since the difference between two distinct values does not matter, as previously said: the only matter of interest is the position of a variable in relation to the others. In particular the fractional ranking allows to take the number of ties per category in account. In this ranking the ties receive the same rank, the mean of what they would have had in an ordinal ranking - where ties are not allowed. For example, the sequence 1-1-1-1-2-2-2-4 is ranked as 2.5-2.5-2.5-2.5-6-6-6-8.

Categorical nominal variables It is used for all measurements which take value among mutual exclusive but non ordered categories. The nationality of someone or his eyes' color are categorical nominal for instance. Sometimes this measurement can take the value "others". It is used for instance in polls when someone's answer does not correspond to the possible choices. Then the comparison between two "others" is not considered as a match.

Categorical dichotomous variables It is a special case of nominal variables where the measurement can only take two values. They are labelled by "presence" and "absence" and the "absence" value corresponds to the value "others" in nominal variables. It is particularly used in the classification of species, when it measures the presence or absence of physical characters. For example, the homology problem in taxonomy consists on deciding whether or not a character occuring in one group of organisms also occur in another group. They are also sometimes called asymetric binary variables.

Continuous circular variables It is used for all measurements that can be represented on a circle. The most common example is the measure of an angle. They are expressed in radian.

Discrete circular variables It is used for all measurements in a finite set that are cyclic and evenly spaced. The most common uses are the days of the week and the months of the year. We call the cardinal of the set of values the number of levels of this variable.

Multichoice variables It is used for all measurements which can take several values among the universe of choices. For example, the answers to a question like "What did you dislike in your travel?" would correspond to this type of variable. The difference with the categorical nominal variables is that the categories are not mutually exclusives. We can see a multichoice variable as a set of dichotomous variables. Each possible choice of answer can be "Yes" - present - or "No".

Fuzzy numbers It is used for all measurements where there might be an incertitude. For example, a measure in physics is never exact; the length of an object will be 8cm $\pm$ 0.2cm, instead of precisely 8cm. Moreover, the imprecision may not be symmetrical. If we know a temperature captor of a washing machine can underestimate the temperature by 0.5 degrees, but only overestimate it by at most 0.2 degrees, the temperature measured could be $39 + 0.5 - 0.2$ degrees. We will represent such a number $m - s_1 + s_2$ by a measure function which would take the value 0 for $x \leq m - s_1$ and $x \geq m + s_2$, the value 1 in m and be linear between $m - s_1$ and m and m and $m + s_2$. Graphically, it looks like a triangle:

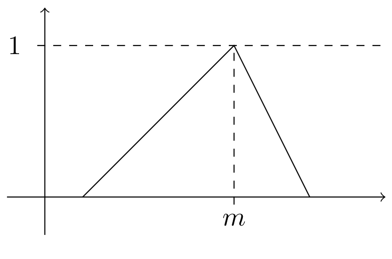


Figure 3: Graphical representation of a fuzzy number

For each of the previously cited data types we define a coefficient of similarity, called s_{ijk}. It measures the similarity between the observation x_i and x_j with respect to the variable k. The Table 1, shown below, sums up those coefficients, and another quantity called δ_{ijk} which indicates if the comparaison is valid for these specific observations on this specific variable. When nothing is specified, δ_{ijk} is equal to 1.

Each time, three criterion have been valorised for the choice of the coefficient of similarity. First, we wanted to use a coefficient both natural and significant. For this we looked for a coefficient already used in machine learning, even if not as a kernel. Secondly, we wanted this coefficient to be a kernel. Sadly, most intuitive ways of measuring similarity lead to associated distances, that do not respect the triangular inequality and are therefore not kernels. An illustration of this fact is represented in figure 4.

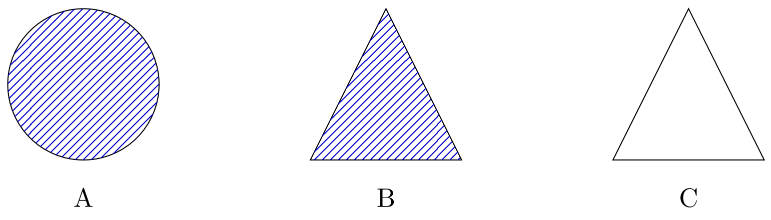


Figure 4: Counter-example of Triangle Inequality: A and B are similar in their shades, B and C are similar in their shape, but A and C are not similar at all.

Finally, the implementation of the coefficient also required some thought. For non-natural coefficients of similarity, references of uses have been put. In the case when there references do not include a proof of PSDness, some hints have been indicated.

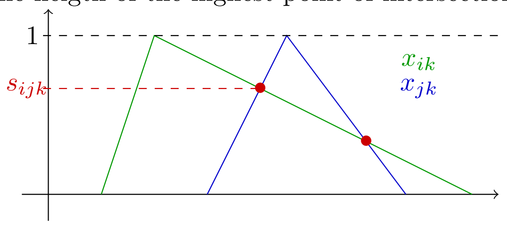
Data Type	**Coefficient of Similarity**	**Reference**	**Proof of PSDness**
Quantitative	$s_{ijk} = 1 - \frac{\lvert x_{ik}-x_{jk}\rvert}{R_k}$ where R_k is the range of the variable.	[Gow71]	See [Gow71].
Ordinal	$s_{ijk} = 1 - \frac{\lvert r_{ik}-r_{jk}\rvert}{R_k}$ where $\begin{cases} r_{ik} \text{ is the fractional of } x_{ik}. \\ R_k \text{ is the range of the variable.} \end{cases}$	[Pod99]	Particular case of the formula for quantitative variables.
Nominal	$\begin{cases} s_{ijk} = 1 \text{ ; } \delta_{ijk} = 1 & \text{if } x_{ik} = x_{jk} \neq \text{ others.} \\ s_{ijk} = 0 \text{ ; } \delta_{ijk} = 0 & \text{if } x_{ik} = x_{jk} = \text{ others.} \\ s_{ijk} = 0 \text{ ; } \delta_{ijk} = 1 & \text{if } x_{ik} \neq x_{jk}. \end{cases}$	[Gow71]	Very similar to the proof for dichotomous variables in [Gow71].
Dichotomous	$\begin{cases} s_{ijk} = 1 \text{ ; } \delta_{ijk} = 1 & \text{if } x_{ik} = x_{jk} = \text{ present.} \\ s_{ijk} = 0 \text{ ; } \delta_{ijk} = 0 & \text{if } x_{ik} = x_{jk} = \text{ absent.} \\ s_{ijk} = 0 \text{ ; } \delta_{ijk} = 1 & \text{if } x_{ik} \neq x_{jk}. \end{cases}$	[Gow71]	See [Gow71].
Continuous circular	$s_{ijk} = \frac{1+\cos(x_{ik}-x_{jk})}{2}$		$\cos(\alpha, \beta) = \frac{\alpha^T \beta}{\lvert\lvert\alpha\rvert\rvert\lvert\lvert\beta\rvert\rvert}$
Discrete circular	$s_{ijk} = \begin{cases} \frac{2}{n}\left(\left\lvert\lvert x_{ik}-x_{jk}\rvert - \frac{n}{2}\right\rvert\right) & \text{if } n \text{ is even.} \\ \frac{2}{n-1}\left(\left\lvert\lvert x_{ik}-x_{jk}\rvert - \frac{n}{2}\right\rvert - \frac{1}{2}\right) & \text{if } n \text{ is odd.} \end{cases}$ where n is the number of levels of the variable.	[PVD+09]	Computing the eigenvalues of the matrices, see Annex A.
Multichoice	$s_{ijk} = \frac{\sum_{p=1..n} s_{ijk_p}}{n}$ where $(k_p)_{p=1..n}$ are the dichotomous sub-variables of k	[Gow71]	Hierarchical caracters of [Gow71].
Fuzzy numbers	The heigth of the highest point of intersection 	[VMA00]	Particular case of Nonsingleton TSK Kernel on fuzzy sets –see [GHC14] –with $T = \min$.

Table 1: Coefficient of similarity for various data types

2.3 Combination of those Coefficient in an Heterogeneous Kernel

In [Gow71], Gower proposed to do the arithmetic mean of the kernels of each variable to obtain a global heterogeneous kernel. This gives the general formula:

$$S_{ij} = \frac{\sum\limits_{k=1}^{v} s_{ijk}\delta_{ijk}}{\sum\limits_{k=1}^{v} \delta_{ijk}}$$

Where:

$$\begin{cases} s_{ijk} \text{ is the coefficient of similarity between observations } i \text{ and } j \text{ with respect to variable } X_k. \\ \delta_{ijk} = \begin{cases} 1 \text{ if the comparison between observations } i \text{ and } j \text{ is valid for the variable } X_k. \\ 0 \text{ otherwise.} \end{cases} \end{cases}$$

As the space of PSD matrices is stable by addition and by multiplication by a scalar, this indeed yields a kernel function.

This construction of the coefficient of similarity is a linear combination. But the key idea of kernels is that they use their inner non-linearity to make linear separation methods work for non-linear cases. If we were to keep this formula, the only way to induce non-linearity would be to have a non-linear kernel for each type of variable. The alternative is of course to transform the formula, into a non-linear one, in order to improve the performances of the kernel.

This also gives another positive aspect: by introducing non-linearity in the combination of the sub-kernel, we create a new parameter, controlling this non-linerarity. Optimization on this parameter will allow to further improve the results.

We want to find a function g to apply to gower similarity - the similarity then becomes $g\left(\frac{\sum\limits_{k=1}^{v} s_{ijk}\delta_{ijk}}{\sum\limits_{k=1}^{v} \delta_{ijk}}\right)$ - such that g verifies three conditions:

- To be an increasing bijection from $[0,1]$ to $[0,1]$.
- To have a parameter we can optimize.
- To conserve the PSDness, in order for the new formula to be a kernel.

And of course we want it to be useful (i.e. to improve the results), even if this can't be known without experimental testing.

First candidate

This is a kernel quite similar to the RBF one. It comes from a tentative of generalization of RBF with an idea of using a generalized distance instead of the euclidean distance for numbers. We called this kernel "exponential Gower", symbolized expoG.

$$\text{expoG} = \exp\left(-\gamma D^2(X,Y)\right)$$

We can express the distance D with the kernel function as we would do with a scalar product: $D^2(X,Y) = ||\phi(X) - \phi(Y)||^2 = K(X,X) - 2K(X,Y) + K(Y,Y) = 2(1 - K(X,Y))$. This leads to:

$$\text{expoG} = \exp\left(\gamma(K(X,Y) - 1\right) = \frac{\exp\left(\gamma K(X,Y)\right)}{\exp\left(\gamma\right)}$$

The complete formula is given by:

$$\text{expoG} = \exp\left(\gamma(K(X,Y) - 1\right) = \exp\left(\gamma\left(\frac{\sum_{k=1}^{v} s_{ijk}\delta_{ijk}}{\sum_{k=1}^{v}\delta_{ijk}} - 1\right)\right). \tag{2}$$

It is a well known fact that the exponential function preserve the PSD-ness of a matrix. To prove it we can use the following theorem:

Theorem 2.2. *Let $A = (a_{ij})$ be a PSD matrix and f a Hadamard function. If f is an analytic function with positive radius of convergence $R > |aij| \forall i, j$ and all the coefficients in its power series expansion are non-negative, then the matrix $f(A)$ is PSD as proved in Horn and Johnson (1991).*

The power series expansion of the exponential function is $\sum_{k\in\mathbb{N}^*} \frac{1}{k} x^k$ and its radius convergence is infinite so it is PSD-conservative.

Second Candidate

We wanted to have a function which looked like a sigmoid with a parameter p to control the curvature of the function - see figure 5. However the formula we have is not exactly a sigmoid.

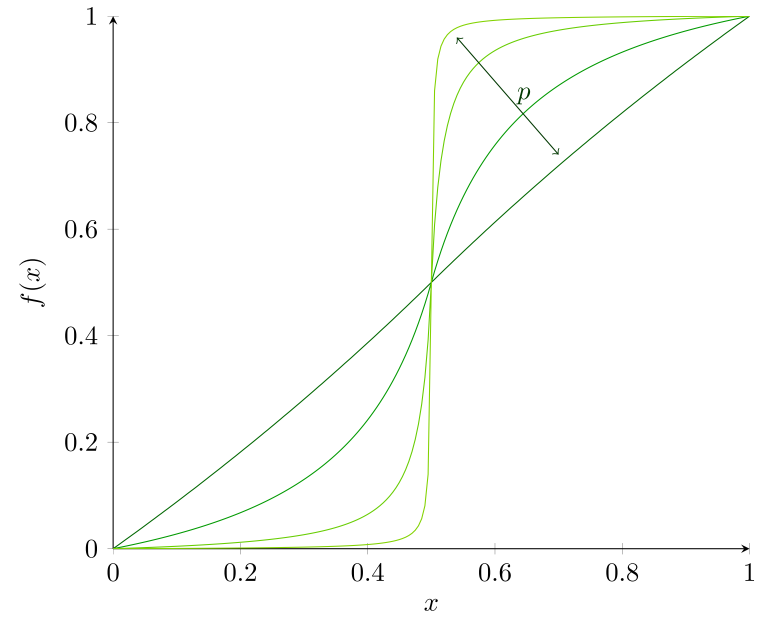


Figure 5: Representation of the sigmoid function we want to apply to the similarity

We define the function on $\left[0, \frac{1}{2}\right]$ and then by symmetry on $\left[\frac{1}{2}, 1\right]$. As we require the function to be continuous and derivable in $\frac{1}{2}$, this leads to the following formula:

$$f : x \mapsto \begin{cases} \frac{-p}{\left(x-\frac{1}{2}\right)-a(p)} - a(p) & \text{if } x \in \left[0, \frac{1}{2}\right] \\ \frac{-p}{\left(x-\frac{1}{2}\right)+a(p)} + a(p) + 1 & \text{if } x \in \left[\frac{1}{2}, 1\right] \end{cases} \text{ with } a(p) = -\frac{1}{4} + \frac{1}{2}\sqrt{\frac{1}{4} + 4p}\ ,\ p > 0$$

Note that the term $a(p)$ comes from the solution of the equation $a(p)^2 + a(p)/2 - p = 0$.

We do not know if this function is PSDness conservative, because we cannot use the same method used for the exponential function to prove that f is PSD conservative. Indeed, the power series expansion of $x \mapsto \frac{1}{a-x}$ is $\sum_{k\in\mathbb{N}} \frac{1}{a^{k+1}x^k}$ for $a \neq 0$ and its radius convergence is $]-|a|, |a|[$. For $x \in \left[0, \frac{1}{2}\right[$, we have $a = \frac{1}{2} + \frac{1}{4} - \frac{1}{2}\sqrt{\frac{1}{4} + 4p}$ which is greater than $\frac{1}{2}$ iff $p \leq 0$. And here $p > 0$.

The PSD-ness conservative property is often hard to prove for functions with a parameter, because it actually is a set of functions istead of a single function - one function for every value of p here. We wanted to wait for the experiments results to confirm whether the function was usable or not - meaning whether it could yields good enough results.

3 Missing Values

Missing information is an old issue in statistical analysis. For example, they are very common in Medicine and Engineering, where many variables come from on-line sensors or device measurements, or are simply too costly to be measured at the same rate as other variables (e.g., analytical tests). There are a lot of possible causes for the absence of a value, far too much to be listed exhaustively but here are some common ones:

- Technical limitations (e.g. sensors only working for given periods of time or malfunctioning);
- Measures costly to perform in time or money or involving destructive methods (e.g., data from car crash tests);
- Measures not done by unknown number of reasons, or in invalid conditions, or simply lost during transmission or storage;
- Senseless values related to other variables (e.g., number of pregnancies in male adults);
- Reluctance to supply the value (e.g., salaries, phone or credit card number, etc).

Missing information is difficult to handle, especially when the missing parts are of significant size. This can be fixed by either removing the entire variable or it can be “filled in” with the mean, median, nearest neighbour, or encoded by adding another input equal to one only if the value is absent and zero otherwise. Statistical approaches need to make parametric assumptions or model the input distribution itself.
There are two basic ways of dealing with missing data:

1. Complete the description of the object in a hopefully optimal way;
2. Extend the methods to be able to work with incomplete object descriptions.

The possibility of simply discarding the involved data can not be considered as a “method” and is also frustrating because of the time and money spent collecting the information. This can only be done if the number of missing values is very small or if they are heavily concentrated in some variables. In practice, it is not uncommon for missing values to be randomly distributed (that is, according to an unknown distribution but independently of the observed values) and hence, if their quantity is not negligible, it is likely to affect a significant number of examples (in the worst case scenario, just one missing value per example) so discarding them all is not a viable alternative.
The first method, called *imputation*, is often used when confronted with quantitative or categorical data. We can either take the median or the mean for quantitative variables, and the most represented category for categorical variable. But the more complex the data type becomes, the more difficult it is to compute an ”average value“.
Instead of filling in the values themselves, we can fill in the values of $k(x, \mathcal{X})$ and $k(\mathcal{X}, \mathcal{X})$ where x is a non-missing value and $\mathcal{X}$ represents a missing value. This has the important advantage that no information is needed from the training set to apply the method.

Gower's coefficient of similarity is often used for its way to handle missing values [PVD+09]. When comparing two observations, in the event of a missing value, Gower recommends to put the average of the similarity for those two observations on characters without missing values, in place of the missing one. We can simply express it in Gower's formula with the coefficient δ_{ijk} which indicates if the comparison is valid for these specific observations on this specific variable. We put δ_{ijk} to 0 if at least one of x_{ik} or x_{jk} is missing or if the comparaison is invalid due to some other reasons for dichotomous or nominal variable. Otherwise we put it to 1. The formula is still:

$$S_{ij} = \frac{\sum_{k=1}^{v} s_{ijk}\delta_{ijk}}{\sum_{k=1}^{v} \delta_{ijk}}$$

Where:

$$\begin{cases} s_{ijk} \text{ is the coefficient of similarity between observations } i \text{ and } j \text{ with respect to variable } X_k. \\ \delta_{ijk} = \begin{cases} 1 \text{ if the comparison between observations } i \text{ and } j \text{ is valid for the variable } X_k. \\ 0 \text{ otherwise.} \end{cases} \end{cases}$$

This method, despite being elegant, has two disavantages. The first and most important one is that the similarity we obtain is not a kernel –Gower gives an counter example in [Gow71]. The second lies in the meaning of the formula: it does not penalize at all the fact to have missing values for similar observations. The two disavantages are linked. We can intuitively understand the non-PSDness property by saying that it could break the triangular inequality: if two observations are very similar in all characteristics but one, they would be more similar by "going through" the same observation in which the different character would have be replaced by a missing value. This situation is illustrated Figure 6.

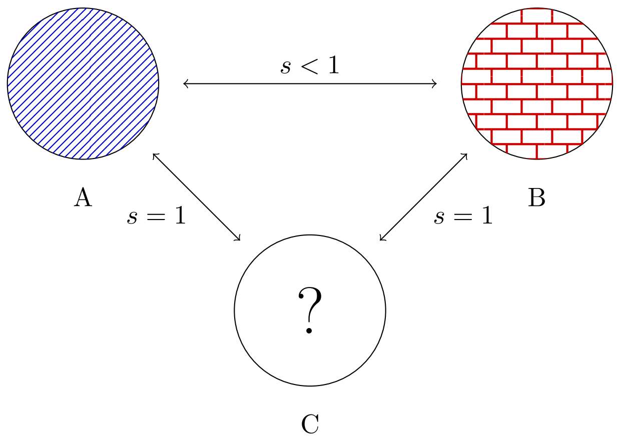


Figure 6: A visual intuition of the non-PSDness of Gower formula: A and B are similar in all but their shades. Therefore, if we "go through" C which is the same but with an an unknown shade, the similarity would be better than comparing directly A and B.

By exploring further –and more formally –this idea, we can get a theoretical superior bound on the value we can replace $K(x, \mathcal{X})$ and $K(\mathcal{X}, \mathcal{X})$ by.

As a corollary of the proposition 2.4, we have that if $(\sqrt{1 - s_{ij}})_{i,j \le n}$ is not a metric or is non-Euclidean, then $S = (s_{ij})_{i,j \le n}$ is not PSD.

Let's examine a case with n observations and d variables. We note x_{lt}, the value of the variable t for the observation l. Let S be a similarity matrix on these data such that $0 \le s_{ij} \le 1$ and $s_{ii} = 1$ for all $i, j \le n$.

Let's pick 3 observations, i, j and k.

$$\begin{array}{lllll} i & x_{i1} & x_{i2} & \dots & x_{id} \\ j & x_{j1} & x_{j2} & \dots & x_{jd} \\ k & x_{k1} & x_{k2} & \dots & x_{kd} \end{array}$$

Suppose that i, j and k are such that:

$$\begin{cases} x_{it} = x_{jt} = x_{kt} & \text{for } 1 \le t < d \\ x_{id} <> x_{jd} & \text{where } <> \text{ means "are opposed" (the similarity between them is 0)} \\ x_{kd} \text{ is missing} \end{cases}$$

Suppose that we want to replace the similarity of the comparaison with the missing value by $x \in [0;1]$. Then we have:

$$\begin{cases} s_{ij} = \frac{d-1}{d} \\ s_{ik} = s_{jk} = \frac{d-1+x}{d} \end{cases}$$

By the corollary, if $\sqrt{1-s_{ij}} > \sqrt{1-s_{ik}} + \sqrt{1-s_{jk}}$, then S is not PSD.

$$\begin{aligned} \sqrt{1-s_{ij}} &> \sqrt{1-s_{ik}} + \sqrt{1-s_{jk}} \\ \sqrt{1-\frac{d-1}{d}} &> 2\sqrt{1-\frac{d-1+x}{d}} \\ 1-\frac{d-1}{d} &> 4\left(1-\frac{d-1+x}{d}\right) \\ 1 &> 4-4x \\ x &> \frac{3}{4} \end{aligned}$$

So, if $x > \frac{3}{4}$, then S is not PSD.

This is only on an example. It is simple and show why Gower's formula does not work. The worst possible examples seem to be this one with an arbitrary number of missing values and the one where the similarity between the observations x_i and x_j is the worst possible –0, but the similarity between x_i and x_k and between x_j and x_k is the best possible, given that the similarity between i and j is 0. The calculations are a bit more complicated, and the results depends on the number of missing values and the number of variables, but the lowest bound is always $\frac{3}{4}$.

A way to get a tighter bound that might depend on the dimension would be to try to express directly on 3×3 sub-matrices the condition of PSDness.

If we summarize, if we try to replace the value of the comparison with a missing value by $x > \frac{3}{4}$, then the resulting matrix can be non-PSD. If we try to replace it by $x = 0$, but keeping the fact that $K(x_i, x_i) = 1$ for any observation x_i it is easy to show that the matrix will conserve its PSDness property. However if we try to replace it by $0 < x \leq \frac{3}{4}$ the PSDness of the matrix is unknown.

Another way of dealing with this problem could be to replace the comparison by $x = 0$ because we know it works. But, as it underestimates the similarity, we then rise the value of each similarity between two observations proportionally with the number of missing values. As said above, we want a function f depending of a parameter α such that the similarity between two observations x_i and x_j can be expressed as

$$S_{ij} = f_{\alpha ij}\left(\frac{\sum_{k=1}^{v} s_{ijk}\delta_{ijk}}{\sum_{k=1}^{v} \delta_{ijk}}:\right)$$

Where αij is the number of missing values when comparing x_i and x_j.

The difficulty lies in finding a function conserving the PSDness even when applied with a different parameter for every element. A possibility could be to use the symmetric operation on lines and columns that conserve PSDness, but it would be too costly to compute because we would need to remember every sub-kernel matrix.

4 Experiments

4.1 Experimental Frame: Support Vector Machines

Support vector machines, or SVMs in short, are a supervised form of machine learning. Basically, the idea is to build a machine, that we train on data similar to the data we want to classify, but where we already know its classes. The goal is to use the information learned during the training phase to be able to accurately predict the classes of the measurements when we apply the machine to unknown data. This usually gives very efficient, both in time and in its results, classifiers to navigate among very large databases.

This section aims to present briefly how the SVMs work. For more details, see [Bur98], [MMR+01] or [HSS07].

SVMs are classifiers designed for binary-classification problems –i.e. when there are only two classes, assuming the data are linearly separable. Say we have some observations $(x_i)_{i\leq l}$ and their associated targets $t_i \in \{-1, 1\}$; we want to separe the positive and negative classes by an hyperplane. It generally exists an infinity of hyperplanes separating the two classes. The SVM algorithm works on the idea that the hyperplane which separes the best the data is the one maximizing the distance between the hyperplane and the closest elements of each class –this distance is called the *margin*, as illustrated in the Figure 7. This idea of margin comes from the theory of minimization of the structural risk (Vapnik, 1979).

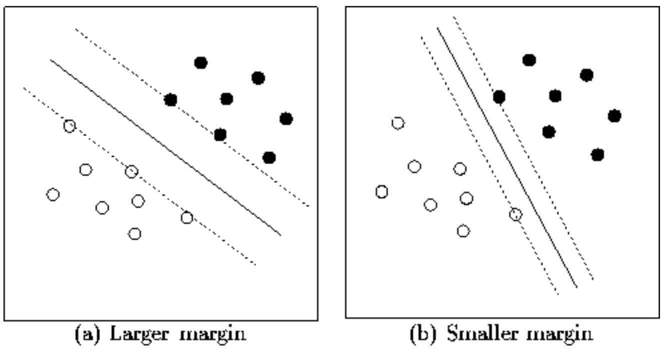


Figure 7: Two hyperplanes separating the same data, one with a large margin (a) and one with a small one (b).

More formally, we will call ω the normal vector of the optimal separating hyperplane –OSH –and b its offset, as shown in the Figure 8. The margin is then $\frac{2}{||\omega||}$. Finding this hyperplane consists in solving the following optimization problem:

$$\min_{\omega,b} \langle\omega|\omega\rangle \text{ subject to: } t_i(\langle\omega|x_i\rangle + b) \geq 1 \text{ , } i = 1..l \tag{3}$$

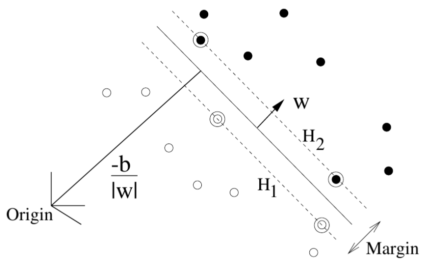


Figure 8: Optimal separating hyperplane in the linear separable case.

It is the phase of training. The model or decision function we obtain then consists to know whether we are on the positive or negative side of the hyperplane: $f(x) = \text{sign}(\langle\omega|x\rangle + b)$.

In order to relax the margins constraints for the non-separable case, we introduce the slack variables $(\xi_i)_{i=1..l}$ and the penalty parameter of error C, which allow for some outsiders as shown in the Figure 9. The optimization problem to solve becomes then:

$$\min_{\omega,b,\xi} \langle\omega|\omega\rangle + C\sum_{i=1}^{l}\xi_i \text{ subject to: } t_i(\langle\omega|x_i\rangle + b) \geq 1 - \xi_i \ , \ i = 1..l \ , \ \xi_i \geq 0 \tag{4}$$

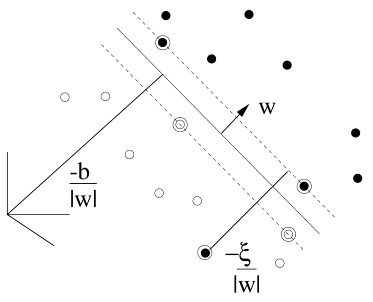


Figure 9: Optimal separating hyperplane in the linear non-separable case.

The decision function can be expressed exactly as in the separable case. This leads to a soft margin SVM called the C-SVM.

When the data are not linearly separable or located in an input space whith no inner product structure, we map them into a feature space where we can use the SVM algorithm, thanks to a kernel function. However if the only way to access the feature space is via dot-products computed by the kernel, we can not solve (3) or (4) directly since ω lies in the feature space. But it turns out that we can get rid of the explicit usage of ω by forming the dual optimization problem. Introducing Lagrange multipliers $(\alpha_i)_{i=1..l}$, the Lagrangian for (3) is:

$$L(\omega, b, \alpha) = \frac{1}{2}||\omega||^2 - \sum_{i=1}^{l}\alpha_i(t_i(\langle\omega|x_i\rangle + b) - 1)$$

We want to minimize it with respect to ω and b and to maximize it with respect to α, which lead to the following optimization problem:

$$\max_{\alpha}\sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j t_i t_j \langle x_i|x_j\rangle \text{ subject to: } \begin{cases} \alpha_i \geq 0 & , \ i = 1..l \\ \sum_{i=1}^{l}\alpha_i t_i = 0 & , \ i = 1..l \end{cases} \tag{5}$$

By solving this new optimization problem, one obtains the coefficient α_i, needed to express the ω solving (4): $\omega = \sum_{i \text{ st } x_i \in \text{SV}} t_i\alpha_i x_i$. This leads to the decision function:

$$f(x) = \text{sign}\left(\sum_{i \text{ st } x_i \in \text{SV}} t_i\alpha_i \langle x_i|x\rangle + b\right)$$

From introducing furthermore the slack-variables ξ_i, one gets a box constraints that limits the size of the Lagrange multipliers: $0 \leq \alpha_i \leq C \ , \ i = 1..l$.

As we now have a SVM algorithm expressed only with inner products, we can apply the kernel trick and replace all inner products by calls to the kernel function K.

The optimization problem becomes:

$$\max_{\alpha}\sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j t_i t_j K(x_i, x_j) \text{ subject to: } \begin{cases} 0 \leq \alpha_i \leq C & , \ i = 1..l \\ \sum_{i=1}^{l}\alpha_i t_i = 0 & , \ i = 1..l \end{cases} \tag{6}$$

The decision function becomes:

$$f(x) = \text{sign}\left(\sum_{i \text{ st } x_i \in \text{SV}} t_i \alpha_i K(x_i, x) + b\right)$$

For most of the points, the multiplier α_i is equal to 0, which means that the point is outside the margin area. The points for which $0 < \alpha_i < C$ are those which are exactly on the margin –they are called the support vector, they are the one we need to remember in the model. Finally the points for which $\alpha_i = C$ are the points that are inside the margin area or even in the wrong side of the hyperplane.

There also exists a variant with a different parameter error, called ν-SVM. Its advantage is that the new parameter has a clearer interpretation than simply "the smaller, the smoother".

To build a classifier when there is only one class, the idea is to compute an hyperplane in the feature space that maximize the distance to the origin while allowing for some outliers.

To build a classifier for more than two classes, there are two methods. Say we want to build a classifier for k classes, we can either:

- Build $\frac{k(k-1)}{2}$ binary classifiers, one for each pair of classes.
- Build k classifiers, one for each class.

The first method has a higher complexity because there is more training, but it yields better results. This is the one we used during our experiments.

There also exists some variant of SVMs capable of doing regression –i.e. when the target is numerical.

The great advantages exhibited by SVMs can be summarized as:

- SVMs are based on a solid theory and on the principle of minimization of the structural risk.
- SVMs do not have local optima and have few parameters to fit, usually C and the kernel's.
- The SVMs are resistent to overfitting (although not immune); overfitting is more probable when the parameter C is larger than necessary.
- The input vectors need not be real numbers; they can be anything as long as one can devise a kernel.

Perhaps the biggest limitation of the SVM and on other kernel-based methods lies in the choice of a proper Kernel for a given problem. Once the Kernel is fixed, SVM classifiers have only one user-chosen parameter (the error penalty C), but the Kernel is a very big rug under which to sweep parameters. A second limitation are computational speed and size, both in training and testing phases, specially when the number of SVs is big.

4.2 Selection of Data Bases

The ideal testing ecosystem would have been to generate synthetical data to test our kernels and SVMs. Real data come with its own problems that interfere with the possible issues with our software:

- It is quite hard to find various heterogeneous data since most problems only consist of variables either nominals, ordinals or quantitatives. There are very few problems with various types being significatively present in them. If there is only one variable of a given type, changing the way we treat it will not yield results different enough to be able to conclude on the method used for this particular data.
- It can be hard to distinguish between an aberration coming from the data and a bug caused by the software; in other words, it can sometimes be hard to say if an issue is due to a malfunction of the algorithm or because real world data tend to have absurd things sometimes.

- We can't control the data, meaning part of the data might be irrelevant to our work, that we don't know if the relation is simple and complex... In particular we don't really know the type of the variables or what the proper way to treat them is.

Synthetical data, where everything is known about it, would have been great to test the software, before starting real world testing to demonstrate its usefulness and compare its performances to other softwares. However, it is both not an easy task and time-consuming to generate useful synthetical data; indeed, you can't generate everything at random if you want to learn something from the data. A relation between observations and targets is needed, and this is especially hard to generate when the result needs to be heterogeneous data.

For these reasons, we used real databases for all our experiments. The databases all come from the UCI website https://archive.ics.uci.edu/ml/datasets/. We tried to have various databases –different subjects, different data types, some homogeneous datasets and some heterogeneous, some large and some small; but all having a significant amount of each of the variables we wanted to test for this dataset and no missing values.

The databased we retained are:

Acute The data consist of diagnosis of two diseases of the urinary system, the acute inflammations of urinary bladder and the acute nephritises. The variables are some symptoms of the patient such as the lumbar pain or the occurence of nausea. The targets are the diagnosis –whether the patient has contracted the disease or not. The data was created by a medical expert.

It is a small dataset that contains mainly dichotomous variables (almost homogeneous dataset). A particularity is that there is two possible binary targets. The data are available at https://archive.ics.uci.edu/ml/datasets/Acute+Inflammations.

Car Evaluation (CEv) The data consist of an evaluation of cars by customers based on criterions such as the price or the capacity. The rating goes from unacceptable to very good.

It is a large dataset that contains only ordinal variables (homogeneous dataset). There are 4 classes and they are strongly unbalanced. The data are available at https://archive.ics.uci.edu/ml/datasets/Car+Evaluation.

Contraceptive Method Choice (CMC) The data consist of a survey on contraceptive methods in Indonesia. The variables are the answers of the candidates to the survey, such as their religion or their standard of living. The target is the method of contraception they use, none, short-term or long-term.

It is a rather large dataset that contains mainly ordinal, nominal and quantitative variables (heterogeneous dataset). There are 3 classes. The data are available at https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice.

German Credit Data (GCD) The data consist of judgements on whether a customer is solvable for a credit or not. The variables are some information such as the age of the customer, or the presence of guarantors. The target is to rank the customer as bad or good for the bank.

It is a rather large dataset that contains ordinal, nominal and quantitative variables (heterogeneous dataset). There are 2 classes which are unbalanced. The data are available at https://archive.ics.uci.edu/ml/datasets/German+Credit+Data

Teaching Assistant Evaluation (TAE) The data consist of evaluations of teaching performances over three regular semesters and two summer semesters of 151 teaching assistant assignments at the Statistics Department of the University of Wisconsin-Madison. The variables are diverse attributes such as the size of the class or the english level of the teacher. The target is a rating between "low", "medium" and "high".

It is a very small dataset that contain mainly nominal variables (almost homogeneous dataset). There are 3 classes. The data are available at https://archive.ics.uci.edu/ml/datasets/Teaching+Assistant+Evaluation

4.3 Experimental Setup

In this section we describe the protocol we used to conduct each of our experiments. We will use the following definition:

Definition 4.1. Let S be a set. Separating S in two stratified sets means that the proportion of each class in S is the same in both subsets.

For each kernel and each function we iterated a standard procedure to have an average accuracy. This standard procedure includes a stratified separation, a scaling, an optimization of parameters and finally the training and prediction themselves. It is summarized in the Algorithm 1.

Algorithm 1 METHOD FOR CONDUCTING THE EXPERIMENTS

```
1: for each data set and each kernel do
2:     for $i = 1$ to 50 do
3:         Separe at random the set between two stratified disjoint sets: a training with $2/3$ of data and
           a testing set with $1/3$ of data.
4:         Scale the training set and the testing set with the same parameters given by the training set.
5:         On the training test set, optimize the paremeters of the kernel and the $C$-svm.
6:         Train a model with those parameters, on the training set.
7:         Use the model to predict the values for the testing set.
8:         Compute the number of right answer over the number of tests –this is called the accuracy.
```

The scaling step corresponds to a standardization –substraction of the mean and division by the standard deviation –for quantitative variables and to a fractional ranking for ordinal variables. The standardization of quantitative variables is done to supress scale effects whereas the fractional ranking of ordinal variables gives a stronger similarity property in the ordinal kernel because it depends on the ties. It is important to note that the scaling of the testing set has to be done with the same parameters as the training set. For example, during the standardization of a quantitative variable we will always use the mean and the standard deviation we find in the training sample.

The optimization of the parameters is done for both the kernel parameter(s) and the SVM parameter C. These parameters need to be optimized to predict as well as possible the testing data, but this optimization is done without knowing the data –meaning that only the information from the training set is taken into account for the optimization.

The most often used method is the grid-search with validation by cross-validation accuracy. The principle is to sample a grid of possible values parameters and to test the validity of the obtained model by n-fold cross validation on the training set. The n-fold cross validation consists in separating the training set in n sub-sets, and then for every combination of $n-1$ sub-sets, to train the model on the $n-1$ sub-sets and predicting on the last one. The cross-validation accuracy is the average of the n accuracies obtained when predicting. The model giving the higher cross-validation accuracy is selected and the traning and testing themselves can begin. Usually n is set to 5 or 10. The big drawback of this method is its high complexity. Indeed, for each combination of parameters, we have to train n models.

To save some computational time we used a heuristic presented in [WW09] to optimize the kernel parameter, and then used cross-validation to optimize only the parameter C. The idea is that we can evaluate the performance of a kernel without training any SVMs by looking at some characteristics of the feature space the kernel defines. Here the characteristic we are interested in is the distance between the classes in the feature space. We evaluate it thanks to a distance index called δ. More exactly, this index will compute the distance between the mean of the classes, to be less sensitive to the noise.

More formally, if we have two classes, X_+ and X_-, in the feature space:

$$\delta = d(\widehat{x_+}, \widehat{x_-}) = d\left(\frac{\sum_{x_+ \in X_+} x_+}{|X_+|}, \frac{\sum_{x_- \in X_-} x_-}{|X_-|}\right)$$

Where d stands for the distance.

As computations in the feature space are costly, we compute the distance thanks to the kernel function:

$$d(\phi(x),\phi(y)) = \sqrt{||\phi(x)-\phi(y)||^2} = \sqrt{\phi(x)\cdot\phi(x) - 2\phi(x)\cdot\phi(y) + \phi(y)\cdot\phi(y)} = \sqrt{K(x,x)+K(y,y)-2K(x,y)}$$

So:

$$\begin{aligned}\delta &= \sqrt{\left\|\frac{\sum_{x_+\in X_+}\phi(x_+)}{|X_+|} - \frac{\sum_{x_-\in X_-}\phi(x_-)}{|X_-|}\right\|^2}\\ &= \sqrt{\frac{\sum_{x_{+1},x_{+2}\in X_+}K(x_{+1},x_{+2})}{|X_+|^2} + \frac{\sum_{x_{-1},x_{-2}\in X_-}K(x_{-1},x_{-2})}{|X_-|^2} - \frac{\sum_{x_+\in X_+,x_-\in X_-}K(x_+,x_-)}{|X_+||X_-|}}\end{aligned}$$

The kernel which gives rise to the higher index distance is selectionned. If there are more than two classes, we compute the index for each pair of classes and compute the average.

[WW09] tests this heuristic only on the RBF kernel. Yet as it is based on a distance index in the feature space, its efficiency should not depend on the particular kernel we use. Hence this heuristic is used for finding the parameter of both the Exponential-Gower and the Sigmoid-Gower kernels.

In our experiments we sample the kernel parameter in $[2^{-20}, 2^{-19}, ..., 2^{20}]$ and the SVM parameter C in $[2^{-7}, 2^{-6}, ..., 2^{7}]$, which seems to be standard [WW09].

4.4 Details of implementation

We used the software LibSVM [CL11]. Each new kernel –Linear Gower, Exponential Gower and Sigoid Gower –have been implemented in the software. As the software was only used with homogeneous numeric kernels, a new file format was also defined. It is composed of a header where the user can precise the data type of each of his variables among quantitative, ordinal, nominal, dichotomous, continuous circular, discrete circular, multichoice and fuzzy numbers.

The software is divided in three parts: SVM-GOWERSCALE, SVM-TRAIN and SVM-PREDICT. The SVM-GOWERSCALE corresponds to both the scaling operations and a part of the training which needs to remember some additionnal data on the training set. For instance, the computation and division of the data by the range for quantitative and ordinal data is done during this step. It is a mandatory step for a good use of Gower's based kernels. Then the SVM-TRAIN can take the sequel and built a model which will be used by SVM-PREDICT.

When implementing, several questions arised. Among them was the question of what to do when the value of a quantitative variable in the testing sample is outside the range of the training sample. Some implementation such as the one in R augment the range by 10% and hope that it will not overcome the new boundaries. We decided to clip all the values outside the range to the boudaries of the training sample. There is a similar problem for the fractional ranking of ordinal variables. An ordinal in the testing set can either has been present or not in the training set. In the first case there is no problem, but in the second case we have to decide which value to attribute. If it is smaller than any of the ordinal in the training set, it will have the value 0, meaning that it is smaller to any other. Similarly, if it is greater than any ordinal in the training set it would get the value number-of-observations-for-this-variable+1, which is the value it would had get if it had been on the training set. To cope with those two rules, the ranks is number-of-observations-for-this-variable+2. Finally, if the variable does not appear in the training set but lies in the range of the training set, we interpolate its value for fractional ranking as the middle of the values for the precedent and the following ordinals present in the training sample. Indeed even if the value is closer from one than another, the only information we get from the ordinal is that it lies between this one and this one.

All the code is present at `https://github.com/eliuranel/libsvm`. For more details, please read the README.

5 Results

We first launched some experiments with the Linear-Gower kernel. The main goal was to see how it worked if we changed the way some classes of variables were considered. For instance we can treat the dichotomous variables as dichotomous or as nominal variables. The ordinal variables can also be treated as ordinal or as nominal.

For the Acute database, the dichotomous one, the accuracy obtained is 100% for every cases, for both diagnosis. So we cannot conclude a lot of things from it - the database was a bit too easy to classify and even non-optimal algorithms performed perfectly here. For the duality ordinal/nominal, the results are summarized in the Table 2. Overall the results are very close whether we treat ordinal variables as nominal variables or as ordinal. When dealing with real world data, it is quite hard to guess what hides between the "ordinal variables" being used; indeed, it can correspond to a lot of various things that sometimes are better understood by the machine when treated as nominal variables, and sometimes as ordinal. The most significative difference in our results was with the GCD database, where we have a slightly better mean and a better median in the NOMI accuracy than in the ORDI one - and also a smaller standard deviation, meaning the results tend to be overall better, even if not by a huge margin. We might improve further the results by testing to see which variables would be better analyzed as ordinal variables, and which would be better analyzed as nominal variables, and compiling a mixed accuracy. We can conclude that it could be interesting to try to use different ways of treating variables with Gower's kernels, even if for this duality, on these example, the results are very similar.

For the following results, we also optimized on the data types of the variables –i.e. treating ordinal as ordinal or nominal for instance –but we only present the best result.

	NOMI Accuracy (%)			ORDI Accuracy (%)		
Database	median	average	standard deviation	median	average	standard deviation
CEv	**93.6**	**93.5**	1.00	93.4	93.4	0.85
CMC	54.3	54.4	1.53	**55.0**	**54.8**	1.71
GCD	**75.4**	**75.1**	**1.55**	74.8	74.8	1.93
TAE	60.2	60.8	5.82	—	—	—

Table 2: Comparison of the predicting accuracies of the Linear-Gower kernel for several databases when treating ordinal variables as nominal ones (NOMI Accuracy) or as ordinal ones (ORDI Accuracy),when it applies.

The second set of experiments we launched was for testing the two non-linear possible improvements presented in Section 2.3: the Exponential-Gower and the Sigmoid-Gower kernels. As we wanted to see the effect of the non-linearity, we used them with the default kernel parameters which are 1 for the Exponential-Gower kernel and 0.1 for the Sigmoid-Gower kernel. The C parameter of the SVM is obtained by cross-validation, as always.

The results of this set of experiments are summarized in the Table 3. We can observe that the Exponential-Gower kernel indeed always gives better results than the Linear-Gower kernel whereas the Sigmoid-Gower kernel always gives worst results than the Linear-Gower kernel. This is expected for the Exponential-Gower kernel, but this is quite surprising for the other one since the idea behind the construction of the Sigmoid-Gower kernel was to improve the linear one...We do not know at the time of writing yet the causes of this surprising result - it might have been a bad implementation of the algorithm instead of a theoretical issue but this hasn't been shown yet in a definite way. We then conducted further experiments on the Exponential-Gower.

The last experiment set compares the performances of our final heterogeneous kernel based on Gower's similarity –the Exponential-Gower kernel with an optimization of the kernel parameter –with the performances of the RBF kernel which is kind of the state of the art in this domain. The RBF kernel which has been used is the one impemented in LibSVM. The optimization of the RBF parameter is done with the same method as the Exponential-Gower one –to know, the heuristic presented Section 4.3.

The results are summarized in Table 4. We can observe that the Exponential-Gower kernel gives significatively better results - even if the margin is not huge. For every database tested, our kernel works much better than the RBF one, reliably producing results 2 or 3% better than what the RBF kernel yields, while at the same time reducing the standard deviation. This shows that even if improvement

	Exponential Gower Accuracy (%)			Sigmoid Accuracy (%)		
Database	median	average	standard deviation	median	average	standard deviation
Acute	100	100	0.00	100	100	0.00
CEv	98.4	98.3	0.60	91.5	91.4	1.11
CMC	54.5	54.6	1.77	45.3	45.3	3.01
GCD	75.7	75.6	1.67	70.9	71.2	1.70
TAE	59.2	60.0	5.94	42.9	43.0	5.65

Table 3: Comparison of the predicting accuracies of the Exponential-Gower and the Sigmoid-Gower kernels for several data bases with default kernel parameters.

is probably still possible, our construction works and seems to be reliably better than what is currently used - the RBF kernel.

	Exponential Gower Accuracy (%)			RBF Accuracy (%)		
Database	median	average	standard deviation	median	average	standard deviation
Acute	100	100	0.00	100	100	0.00
CEv	**99.7**	**99.5**	**0.40**	98.4	98.4	0.62
CMC	**54.1**	**54.3**	**1.59**	52.7	53.0	1.70
GCD	**75.5**	**75.4**	**1.47**	72.7	72.6	1.95
TAE	**61.2**	**61.9**	**6.36**	57.1	55.8	6.38

Table 4: Comparison of the predicting accuracies of the Exponential-Gower and the RBF kernels for several data bases, both with parameters optimized thanks to the distance-index based heuristic.

6 Conclusion

On the end, even if we only tested the kernel we designed on a small range of it is designed for, we obtain really promising results: they are competitive and slightly better than what we obtain on the same databases with the same protocol with the RBF kernel.

In future work, we hope to prove that with more complex data structures and kernels specifically designed for those data structures, and integrated into the Exponential-Gower kernel, it could really outperform RBF by a huge margin.

Moreover, we only tested this kernel for classification with C-SVMs, but it can also be used for a lot of other kernel-based learning algorithms such as kernel fisher discriminant analysis and kernel principal component analysis.

Another area where this kernel can be further improves is with other features available for all kernels such as weights. The function would becomes then:

$$\exp\left(\gamma\left(\frac{\displaystyle\sum_{k=1}^{v} s_{ijk}\delta_{ijk}w_k}{\displaystyle\sum_{k=1}^{v}\delta_{ijk}w_k} - 1\right)\right)$$

Acknoledgments

I'd like to thank my advisor Lluis Belanche for guiding me during all my internship and providing me answers during the long hours of our meetings. It was great to discover this domain and work on it.

References

[Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[GHC14] Jorge Guevara, Roberto Hirata, and Stéphane Canu. Positive definite kernel functions on fuzzy sets. In *IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2014, Beijing, China, July 6-11, 2014*, pages 439–446, 2014.

[GL86] J. C. Gower and P. Legendre. Metric and euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3(1):5–48, March 1986.

[Gow71] J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27(4):857–871, December 1971.

[HSS07] Thomas Hofmann, Bernhard Scholkopf, and Alexander J. Smola. Kernel methods in machine learning. Published online at arXiv.org http://arxiv.org/abs/math/0701907v2., 2007. To appear in Annals of Statistics, 2008.

[MMR$^+$01] K.-R. Müller, S. Mika, G. Rätsch, S. Tsuda, and B Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–202, 2001.

[Pod99] János Podani. Extending gower's general coefficient of similarity to ordinal characters. *Taxon*, 48(2):pp. 331–340, 1999.

[PVD$^+$09] S. Pavoine, J. Vallet, A.-B. Dufour, S. Gachet, and H. Daniel. On the challenge of treating various types of variables: application for improving the measurement of functional diversity. *Oikos*, 118(3):391–402, 2009.

[VMA00] Julio J. Valdés, Lluís A. Belanche Muñoz, and René Alquézar. Fuzzy heterogeneous neurons for imprecise classification problems. *Int. J. Intell. Syst.*, 15(3):265–276, 2000.

[WW09] Kuo-Ping Wu and Sheng-De Wang. Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space. *Pattern Recognition*, 42(5):710–717, 2009.

A Proof of the PSDness of the similarity for discrete circular variables

Definition A.1. Let X_k be a discrete circular variable with n levels. Let x_{ik} (resp. x_{jk}) be the value taken by observation i (resp j). Then the similarity between observations i and j with respect to variable X_k, s_{ijk}, is defined by:

$$\boxed{s_{ijk} = \begin{cases} \frac{2}{n}\left(\left||x_{ik} - x_{jk}| - \frac{n}{2}\right|\right) & \text{if } n \text{ is even.} \\ \frac{2}{n-1}\left(\left||x_{ik} - x_{jk}| - \frac{n}{2}\right| - \frac{1}{2}\right) & \text{if } n \text{ is odd.} \end{cases}}$$

Claim A.1. *If the similarity matrix obtained for the particular observation vector* $\mathbf{x} = (1, 2, \ldots, n)$ *is PSD, then the similarity matrix obtained for any observation vector* $\mathbf{x}'$ *is PSD.*

Proof. Let assume that the similarity matrix obtained for the particular observation vector $\mathbf{x} = (1, 2, \ldots, n)$ is PSD. We want to show that, for all observation vector $\mathbf{x}'$, the similarity matrix we obtain is PSD.

First, we can observe that we can remove the duplicates of $\mathbf{x}'$. Indeed, if we substract the row and the column of one of the duplicate to the other ones, we don't change the PSDness of the matrix (see Lemma **??**) and we have 0-rows and 0-columns for all the duplicates. Which mean that the matrix for $\mathbf{x}'$ is PSD if and only if the matrix for $\mathbf{x}''$ is PSD where $\mathbf{x}''$ is the same vector as $\mathbf{x}'$ but without the duplicates.

Secondly, as the matrix obtained for the observation vector $\mathbf{x}''$ is a submatrix of the matrix obtained for the observation vector $\mathbf{x}$ (in the sense that it can be obtain by removing some rows and the corresponding columns), it is also PSD (see Corollary 2.1). □

Claim A.2. *Let* $m = \lfloor \frac{n}{2} \rfloor$. *Then the matrix* S *obtained for the particular observation* $(1, 2, 3, \ldots, n)$ *is*

$$S = \frac{1}{m}\begin{pmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{pmatrix} \quad \textit{with} \begin{cases} c_0 = m \\ c_k = c_{n-k} = m - k = \lfloor \frac{n}{2} \rfloor \textit{ for } k = 1..m \end{cases}$$

Lemma A.1. $\forall n \in \mathbb{N}$, *the matrix* S *defined above is PSD.*

Proof. To show that S is PSD, we will show that all its eigenvalues are non-negative. As S is a circulant matrix[1], we can write its eigenvalues as:

$$\lambda_j = c_0 + \sum_{k=1}^{n-1} c_k \omega_j^{n-k} \text{ where } \begin{cases} j = 0..n-1 \\ \omega_j = e^{\frac{2i\pi j}{n}} \text{ are the } n^{th} \text{ root of unity.} \end{cases}$$

We will treat separately the case when j equals 0 and the general case.

When $j = 0$, we have:

$$\lambda_0 = c_0 + \sum_{k=1}^{n-1} c_k = m + 2\sum_{k=1}^{m}(m-k) = m + 2\sum_{k=0}^{m-1} k = m + m(m-1) = m^2 \geq 0.$$

[1]For more details, see https://en.wikipedia.org/wiki/Circulant_matrix.

When $j = 1..n-1$, we have:

$$\begin{aligned}
\lambda_j &= c_0 + \sum_{k=1}^{n-1} c_k \omega_j^{n-k} \\
&= m + \sum_{k=1}^{m} (m-k)\left(\omega_j^k + \omega_j^{n-k}\right) \\
&= m + \sum_{k=1}^{m} (m-k) 2 \cos\left(\frac{2j\pi}{n} k\right) \\
&= m + 2\left[m \sum_{k=1}^{m} \cos(\alpha k) - \sum_{k=1}^{m} k \cos(\alpha k)\right] \text{ where } \alpha = \frac{2j\pi}{n} \ (\neq 0[2\pi])
\end{aligned}$$

$\sum_{k=1}^{m} \cos(\alpha k)$ and $\sum_{k=1}^{m} k\cos(\alpha k)$ are two classic trigonometric sums. When $\alpha \neq 0[2\pi]$:

$$\begin{aligned}
\sum_{k=1}^{m} \cos(\alpha k) &= \mathbb{R}\mathrm{e}\left(\sum_{k=1}^{m} \left(e^{i\alpha}\right)^k\right) \\
&= \mathbb{R}\mathrm{e}\left(e^{i\alpha} \frac{1-e^{i\alpha m}}{1-e^{i\alpha}}\right) \\
&= \mathbb{R}\mathrm{e}\left(e^{i\alpha} \frac{e^{\frac{i\alpha m}{2}}}{e^{\frac{i\alpha}{2}}} \frac{e^{\frac{i\alpha m}{2}} - e^{\frac{-i\alpha m}{2}}}{e^{\frac{i\alpha}{2}} - e^{\frac{-i\alpha}{2}}}\right) \\
&= \mathbb{R}\mathrm{e}\left(e^{\frac{i\alpha(m+1)}{2}} \frac{\sin\left(\frac{\alpha m}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)}\right) \\
&= \cos\left(\frac{\alpha(m+1)}{2}\right) \frac{\sin\left(\frac{\alpha m}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)} \\
\sum_{k=1}^{m} k\cos(\alpha k) &= \mathbb{R}\mathrm{e}\left(\frac{1}{i}\frac{d}{dx}\left(\sum_{k=1}^{m}\left(e^{ix}\right)^k\right)(\alpha)\right) \\
&= \mathbb{R}\mathrm{e}\left(\frac{1}{i}\frac{d}{dx}\left(e^{\frac{ix(m+1)}{2}} \frac{\sin\left(\frac{xm}{2}\right)}{\sin\left(\frac{x}{2}\right)}\right)(\alpha)\right) \\
&= \mathbb{R}\mathrm{e}\left(\frac{1}{i}\left[i\frac{m+1}{2} e^{i\alpha\frac{m+1}{2}} \frac{\sin\left(\frac{\alpha m}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)} + e^{i\alpha\frac{m+1}{2}} \frac{\frac{m}{2}\cos\left(\frac{\alpha m}{2}\right)\sin\left(\frac{\alpha}{2}\right) - \frac{1}{2}\cos\left(\frac{\alpha}{2}\right)\sin\left(\frac{\alpha m}{2}\right)}{\sin^2\left(\frac{\alpha}{2}\right)}\right]\right) \\
&= \mathbb{R}\mathrm{e}\left(\frac{e^{i\alpha\frac{m+1}{2}}}{\sin^2\left(\frac{\alpha}{2}\right)}\left[\frac{m+1}{2}\sin\left(\frac{\alpha m}{2}\right)\sin\left(\frac{\alpha}{2}\right) + \frac{1}{i}\frac{m}{2}\cos\left(\frac{\alpha m}{2}\right)\sin\left(\frac{\alpha}{2}\right) - \frac{1}{i}\frac{1}{2}\sin\left(\frac{\alpha m}{2}\right)\cos\left(\frac{\alpha}{2}\right)\right]\right) \\
&= \frac{m+1}{2}\cos\left(\frac{\alpha(m+1)}{2}\right)\frac{\sin\left(\frac{\alpha m}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)} + \frac{m}{2}\sin\left(\frac{\alpha(m+1)}{2}\right)\frac{\cos\left(\frac{\alpha m}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)} \\
&- \frac{1}{2}\sin\left(\frac{\alpha(m+1)}{2}\right)\frac{\cos\left(\frac{\alpha}{2}\right)\sin\left(\frac{\alpha m}{2}\right)}{\sin^2\left(\frac{\alpha}{2}\right)}
\end{aligned}$$

So,

$$\begin{aligned}
\lambda_j &= m + 2\left[\frac{m-1}{2}\cos\left(\frac{\alpha(m+1)}{2}\right)\frac{\sin\left(\frac{\alpha m}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)} - \frac{m}{2}\sin\left(\frac{\alpha(m+1)}{2}\right)\frac{\cos\left(\frac{\alpha m}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)}\right. \\
&\quad \left.+\frac{1}{2}\sin\left(\frac{\alpha(m+1)}{2}\right)\frac{\cos\left(\frac{\alpha}{2}\right)\sin\left(\frac{\alpha m}{2}\right)}{\sin^2\left(\frac{\alpha}{2}\right)}\right] \\
&= m + \frac{m}{\sin\left(\frac{\alpha}{2}\right)}\left[\cos\left(\frac{\alpha(m+1)}{2}\right)\sin\left(\frac{\alpha m}{2}\right) - \sin\left(\frac{\alpha(m+1)}{2}\right)\cos\left(\frac{\alpha m}{2}\right)\right] \\
&\quad + \frac{\sin\left(\frac{\alpha m}{2}\right)}{\sin^2\left(\frac{\alpha}{2}\right)}\left[\sin\left(\frac{\alpha(m+1)}{2}\right)\cos\left(\frac{\alpha}{2}\right) - \cos\left(\frac{\alpha(m+1)}{2}\right)\sin\left(\frac{\alpha}{2}\right)\right] \\
&= m + m\frac{\sin\left(-\frac{\alpha}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)} + \frac{\sin\left(\frac{\alpha m}{2}\right)}{\sin^2\left(\frac{\alpha}{2}\right)}\sin\left(\frac{\alpha m}{2}\right) \\
&= \frac{\sin^2\left(\frac{\alpha m}{2}\right)}{\sin^2\left(\frac{\alpha}{2}\right)} \geq 0
\end{aligned}$$

$\square$