Homework 5

Luca Witte

2022-12-11

Task 1

Here we explore kernels as tools in data mining. Kernel methods are algorithms that access data only in terms of kernel functions. An important kernel method is the support-vector machine (SVM). The term *kernel* usually refers to the kernel function.

In SVM, we map the data ($x \in \mathcal{X}$) to a (ususally higher-dimensional) space $\mathcal{H}$ in which a hyperplane is used for classification:

$$\phi : \mathcal{X} \to \mathcal{H}$$

The kernel defines a dot product ($\langle x, y \rangle$) between the points in $\mathcal{H}$. The mapping of the data points in another space can be a costly process which can be avoided by using the *kernel trick*. A dot product $k(x, y) = \langle \phi(x),\ \phi(y) \rangle$ is evaluated by accessing the data only via the inner product.

Subtask 1.a

The polynomial kernel

We define $x = (x_1, x_2)$ and $x' = (x'_1, x'_2)$ in the input space. The polynomial kernel is defined as:

$$k(x, x') = (\langle x, x' \rangle +\ c)^p$$

With $p = 2$ and $c = 1$, we obtain:

$$\begin{aligned} k(x, x') &= (\langle x, x' \rangle +\ 1)^2 \\ &= (x_1 x'_1 + x_2 x'_2 +\ 1)^2 \\ &= (x_1 x'_1)^2 + (x_2 x'_2)^2 + 2(x_1 x'_1)(x_2 x'_2) + 2(x_1 x'_1) + 2(x_2 x'_2) + 1 \\ &= (x_1^2, x_2^2, \sqrt{2} x_1 x_2, \sqrt{2} x_1, \sqrt{2} x_2, 1) \cdot (x_1^{'2}, x_2^{'2}, \sqrt{2} x'_1 x'_2, \sqrt{2} x'_1, \sqrt{2} x'_2, 1) \end{aligned}$$

Therefore we can extract in the feature space:

$$\phi(x) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2, \sqrt{2} x_1, \sqrt{2} x_2, 1)$$

This indicates a six-dimensional feature space. As the last term is an identical constant in both $\phi(x)$ and $\phi(x')$, we can ignore this dimension.

Here we can also observe a simple example of how using a kernel reduces the complexity of a calculation. Instead of the fully expanded dot product given above, the calculation is reduced to the kernel function.

The Gaussian radial basis function kernel

The Gaussian radial basis function (RBF) kernel is another often used non-linear kernel. It is defined as:

$$k(x, x') = exp(-\frac{1}{2\sigma}||x - x'||^2)$$

As can be seen above, the Gaussian RBF kernel uses the squared Euclidean distance and can be interpreted like a similarity measure. Deriving the explicit function ϕ requires a Taylor series expansion of a natural exponential term. This expansion of the exponential function by definition has infinite length. Therefore, the ϕ of the Gaussian RBF kernel is mapping a vector into infinite dimensional space ($\phi : \mathbb{R}^n \to \mathbb{R}^\infty$). The full proof can be found in the literature, but is not supposed to be provided here.

Subtask 1.b

This question refers to the kernel trick mentioned above. The performance of kernel methods is very good due to the fact that the data does not have to be mapped to the feature space $\mathcal{H}$ explicitly. In high dimensional feature spaces, this process can be costly. A kernel instead returns $k(x, y) = \langle\phi(x),\ \phi(y)\rangle$ without the need to explicitly calculate $\phi(x)$ and $\phi(y)$. Without such an approach, an exact explicit transformation of a vector in infinite dimensions (as discussed above for the Gaussian RBF kernel) is by definition impossible.

SVM is a kernel method as it only accesses the input data via kernel functions, thereby not requiring explicit mapping to the feature space.

Task 2

The second and third part of the homework deal with constructing and validating kernels.

A similarity measure is a kernel if its Gram matrices are positive semi-definite. This means that for a set $\{x_1, ..., x_n\}$, the $n \times n$ Gram matrix $K_{ij} = k(x_i, x_j)$ satisfies the following condition for all $c_i \in \mathbb{R}$:

$$\sum_{i,j} c_i c_j K_{ij} \geq 0$$

Subtask 2.a

For $\mathbb{X} \subset \mathbb{R}^d$, the linear kernel $k(x, x') = \sum_{m=1}^{d} x_m x'_m = x^T x'$ is a kernel for all $n \in \mathbb{N}$ and all sets $\{x_1, ..., x_n\} \in \mathbb{R}^d$. To prove this, we can show that the Gram matrix is positive semi-definite:

$$For:\ k(x, x') = \langle x, x'\rangle = \sum_{m}^{d} x_m x'_m$$
$$we\ require:\ \sum_{i,j} c_i c_j K_{ij} \geq 0$$

We can use scalar multiplication of the dot product to derive:

$$\begin{aligned}\sum_{i,j} c_i c_j K_{ij} &= \sum_i \sum_j c_i c_j \sum_m^d x_m x'_m \\ &= \sum_i \sum_j \sum_m^d c_i x_m c_j x'_m \\ &= \sum_m^d (\sum_i c_i x_m)(\sum_j c_j x'_m)\end{aligned}$$

As x and x' belong to the same set:

$$\begin{aligned}\sum_{i,j} c_i c_j K_{ij} &= \sum_m^d (\sum_i c_i x_m)(\sum_j c_j x'_m) \\ &= \sum_m^d (\sum_i c_i x_m)^2 \\ with : & \\ & \sum_m^d (\sum_i c_i x_m)^2 \geq 0\end{aligned}$$

Subtask 2.b

In this subtask we prove that the dot product in any feature space is a kernel:

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

Again we can use the positive semi-definiteness of the Gram matrix and both x and x' belonging to the same set:

$$\begin{aligned}\sum_{i,j} c_i c_j K_{ij} &= \sum_i \sum_j c_i c_j \phi(x) \phi(x') \\ &= \sum_i \Big(c_i \phi(x)\Big)^2 \\ with : \sum_i \Big(c_i \phi(x)\Big)^2 &\geq 0\end{aligned}$$

Subtask 2.c

In this subtask, we are exploring the closure properties of kernels. For this, two valid kernels k_1 and k_2 with the Gram matrices K_1 and K_2 are defined.

To prove that a kernel can be constructed by addition of two valid kernels, we use again the property of positive semi-definiteness. For a vector $\vec{x}$ with at least one non-zero element, by definition the following statements for the Gram matrices hold true:

$$\vec{x}^T K_1 \vec{x} \geq 0$$
$$\vec{x}^T K_2 \vec{x} \geq 0$$

Therefore:

$$\vec{x}^T K_1 \vec{x} + \vec{x}^T K_2 \vec{x} \geq 0$$
$$\vec{x}^T (K_1 + K_2) \vec{x} \geq 0$$

Which shows that all Gram matrices are positive semi-definite, therefore $k_3(x, x') = k_1(x, x') + k_2(x, x')$ is a semi-definite kernel function.

Next, we show that multiplication of a valid kernel with a positive real scalar also results in a valid kernel. For $k_4(x, x') = \lambda k_1(x, x')$ with $\lambda \in \mathbb{R}^+$, we can show:

$$By\ definition : \vec{x}^T K_1 \vec{x} \geq 0$$
$$Therefore : \lambda \vec{x}^T K_1 \vec{x} \geq 0$$

Task 3

This task deals with the construction of a kernel. A kernel is constructed by either:

1. Define a mapping ϕ explicitely and/or
2. Using the closure properties of kernels

The closure properties of kernels are defined as follows:

- the sum of two valid kernel functions is also a kernel function (closed under addition)
- the product of a kernel and a positive scalar is also a kernel function
- the product of two valid kernel functions is also a kernel function
- the zero-extension of a valid kernel that is only defined on a specific set is also a valid kernel

Some of these properties have been explored above.

Subtask 3.a

In this task we are required to prove that the following kernel is valid:

$$k(x, x') = 6\langle x, x' \rangle^4 + 3 + x^T x' + exp(-\frac{1}{2\sigma}||x - x'||^2)$$

This can be approached using the property of kernels being closed under addition. Therefore, we require a separate proof of each term being a valid kernel.

- $6\langle x, x' \rangle^4$:

  This term is a product and therefore we can prove it to be a kernel by proving that each of the factors is a valid kernel.

  - The term 6 is a constant kernel as we can define the mapping $\phi(x) = 6$ explicitly
  - The term $\langle x, x' \rangle^4$ is a polynomial kernel with $c = 0$ and $d = 4$. The polynomial kernel is a known kernel.

- 3:

  As above, we can define a mapping $\phi(x) = 3$ explicitly to obtain a constant kernel.

- $x^T x'$:

  This is the linear kernel, which is a known kernel and was proven earlier.

- $exp(-\frac{1}{2\sigma}||x-x'||^2)$:

  This term is the Gaussian RBF kernel, which is known to be a kernel. Proofs can be found in the literature.

By showing that each term in the sum is a valid kernel, we prove that the kernel in subtask 3.a is also a kernel. This follows from the property of kernels being closed under addition.

Subtask 3.b

Here we construct a kernel to compare amino acid sequences. The background suggests that these sequences stem from human collagen proteins. Collagen is rich in glycine-XY (GXY) motifs, 3-mers that start with glycine and contain two non-glycine amino acids.

By defining a GXY similarity measure, we can compare the number of similar GXY motifs in two sequences. More explicitely, the measure compares all 3-mers between two sets that start with glycine and quantifies the number of perfect matches in the 3-mer sequences.

The input objects X and X' are amino acid sequences of arbitrary length. We decompose the objects into the sets S and S' containing all 3-mers (substrings of length 3). To create this set from a string in task 3.d, the following example R-code can be used:

```r
seq1 <- "GPAGFAGPPGDA"

begin <- 1
set1 <- NULL
for (i in c(1:(nchar(seq1)-2))){
  set1 <- c(set1, substring(seq1, begin, begin+2))
  begin <- begin + 1
}

print(set1)
```

```
##  [1] "GPA" "PAG" "AGF" "GFA" "FAG" "AGP" "GPP" "PPG" "PGD" "GDA"
```

We obtain the sets containing 3-mers: $S = \{s_1, ..., s_n\}$ and $S' = \{s'_1, ..., s'_m\}$. In this notation, n and m represents the number of 3-mers in each set. Following the R-convolution framework, we can define the GXY similarity measure as sum of 3-mer similarities:

$$k_{GXY}(X, X') = \sum_{s \in S, s' \in S'} k_{base}(s, s')$$

The double sum is required as we do not expect the sequences to be aligned. Therefore the calculation iterates through every possible pairing of 3-mers between the two sequences. This also makes the approach more robust to insertions and deletions.

We require the kernel k_{base} to return 0 if at least one 3-mer does not start with a G. If both 3-mers start with G, the number of perfect matches is returned. In the following notation, the amino acid positions of each 3-mer are indexed by $i \in \{1, 2, 3\}$. As an example, the second amino acid of the first 3-mer in S (GPA) is: $s_1[2] = P$.

$$k_{base}(s, s') = \begin{cases} 0 & \text{if } s[1] \neq G \text{ or } s'[1] \neq G \\ \sum_{i=1}^{3} \Big[s[i] = s'[i]\Big] & \text{otherwise} \end{cases}$$

The Iverson bracket notation (https://en.wikipedia.org/wiki/Iverson_bracket, 12.12.2022) is used to count perfectly matched elements in s and s'. It is a shorthand notation for:

$$\Big[s[i] = s'[i]\Big] = \begin{cases} 0 & if\ s[i] \neq s'[i] \\ 1 & if\ s[i] = s'[i] \end{cases}$$

What we do here, is defining a feature map which returns the value of shared amino acids under the condition that both sequences start with "G". Therefore, this is a modified version of the spectrum kernel.

The similarity measure $k_{GXY}(X, X')$ sums over all shared residues in GXY motifs.

An alternative description of the GXY-similarity measure can be given in pseudocode:

```
define S #Set of 3-mers
define S_prime #Set of 3-mers

for each s in S:
  for each s_prime in S_prime:
    if s[1] == G & s_prime[1] == G:
      for i in (1:3):
        if s[i] == s_prime[i]:
          Output += 1
```

Subtask 3.c

Here, the function defined in 3.b is used to evaluate similarities between short sample sequences:

1. X_1(COL1A1): GPAGFAGPPGDA
2. X_2(COL1A2): PRGDQGPVGRTG
3. X_3(GPR 143): GFPNFDVSVSDM

First the sets of 3-mers are defined:

- $S_1 = \{GPA, PAG, AGF, GFA, FAG, AGP, GPP, PPG, PGD, GDA\}$
- $S_2 = \{PRG, RGD, GDQ, DQG, QGP, GPV, PVG, VGR, GRT, RTG\}$
- $S_3 = \{GFP, FPN, PNF, NFD, FDV, DVS, VSV, SVS, VSD, SDM\}$

In the written calculation, we can reduce the complexity by only considering the 3-mers in each set that start with "G":

- $S_1 = \{GPA, GFA, GPP, GDA\}$
- $S_2 = \{GDQ, GPV, GRT\}$
- $S_3 = \{GFP\}$

For all other 3-mers, the kernel k_{base} is zero and therefore does not affect the GXY similarity measure.

First we calculate $k_{GXY}(X_1, X_2)$. For this we calculate k_{base} for every combination of trimers from S_1 and S_2:

$$\begin{aligned}
k_{base}(GPA, GDQ) &= 1\\
k_{base}(GPA, GPV) &= 2\\
k_{base}(GPA, GRT) &= 1\\
\\
k_{base}(GFA, GDQ) &= 1\\
k_{base}(GFA, GPV) &= 1\\
k_{base}(GFA, GRT) &= 1\\
\\
k_{base}(GPP, GDQ) &= 1\\
k_{base}(GPP, GPV) &= 2\\
k_{base}(GPP, GRT) &= 1\\
\\
k_{base}(GDA, GDQ) &= 2\\
k_{base}(GDA, GPV) &= 1\\
k_{base}(GDA, GRT) &= 1
\end{aligned}$$

with $k_{base} = 0$ for all other 3-mer pairs

$$\begin{aligned}
k_{GXY}(X_1, X_2) &= \sum_{s \in S_1, s' \in S_2} k_{base}(s, s')\\
&= 9 * 1 + 3 * 2\\
&= 15
\end{aligned}$$

The same way we can compare X_1 and X_3 to calculate $k_{GXY}(X_1, X_3)$:

$$\begin{aligned}
k_{base}(GPA, GFP) &= 1\\
\\
k_{base}(GFA, GFP) &= 2\\
\\
k_{base}(GPP, GFP) &= 2\\
\\
k_{base}(GDA, GFP) &= 1
\end{aligned}$$

with $k_{base} = 0$ for all other 3-mer pairs

$$\begin{aligned}
k_{GXY}(X_1, X_3) &= \sum_{s \in S_1, s' \in S_3} k_{base}(s, s')\\
&= 2 * 1 + 2 * 2\\
&= 6
\end{aligned}$$

It can be observed that sequences 1 and 2 have a higher frequency of GXY motifs and therefore score higher in the similarity measure than the comparison of sequences 1 and 3. Though there are no matches of full 3-mers, which does not necessarily indicate a close genetic relation of the sequences. Rather, this might imply a similar (helical) structure and biochemical function.

Subtask 3.d

The data used in 3.c was of equal length and a multiple of three. In this subtask the author discusses limitations of the presented approach and possible consequences of using arbitrary sequences.

Sequence length

In subtask 3.c we observed that a higher frequency of glycine increases the similarity score. Though not all glycine residues are part of a helical structure. Indeed, glycine constitutes around 5-7% of a species proteome (doi:10.1186/s13104-018-3221-0). Due to its small structure, it has an important role in protein folding. Therefore we can expect additional glycines (and thereby GXY motifs) to occur for increased sequence length in any arbitrary protein. Subsequently, increased sequence length will increase the returned score. While shared GXY motifs will still have a larger impact on the returned score, single glycins will increase the score even for non-helical structures.

Subsequently, longer sequences will score higher in the described similarity measure. Comparing X_1 to a short GXY-rich sequence (e.g. X_2) and a very long protein sequence with no helical structure might return higher scores for the longer sequence just because of the higher absolute number of glycine residues. Therefore, the author proposes to implement a normalization to sequence length or to restrict usage to sequences of equal lengths. Also, of course we require sequence lengths of at least three amino acids to obtain a non-empty set of 3-mers.

Further, in the naive implementation shown above, every 3-mer is compared to every 3-mer from the other set. This implies a complexity on the order of $\mathcal{O}(N^2)$. As longer sequences correspond to a larger number of 3-mers (N), we expect a quadratic increase in computation cost with an increase in length.

Lengths that are not multiples of 3

The presented approach is used to identify a specific three amino acid motif, therefore we investigated 3-mers. Length of the sub-sequences in which the input object is decomposed is a user-defined hyperparameter and can be adjusted to different use cases. In the described example, the sequences are not aligned prior to analysis and therefore we iterated over all possible pairs of subsequences. In such an approach, sequence length multiplicity does not matter, as we consider all possible substrings.

In other cases, the sequence might be required to be a multiple of a certain number. Genomic DNA comes to mind as an example, where omitting a single base might change the reading frame of a sequence. Therefore outcomes in an analysis that considers codon usage or otherwise relies on the triplets might be affected.