Convolution kernels

Predicting Structured Data seminar

Esa Pitkänen

Department of Computer Science, University of Helsinki

Outline

- Kernels
- Convolution kernels
  - Decompositions
  - R-convolution
  - Examples: radial basis kernel, subtree kernels
- Fisher kernels (time permitting)

References

- David Haussler: Convolution Kernels on Discrete Structures. Technical Report, UCSC-CRL-99-10, 1999.
- John Shawe-Taylor and Nello Cristianini: Kernel Methods for Pattern Analysis. Cambridge University Press, 2004.
- J.P. Vert, K. Tsuda, and B. Scholkopf: A primer on kernel methods. Kernel Methods in Computational Biology, MIT Press, 2004.

Convolution kernels: overview

- *Convolution kernel* is a general approach for defining kernels for structured data
- Structured object can be decomposed into component objects
- Components have kernels defined for them
- Convolution kernel: use component kernels to measure similarity over all possible decompositions

Convolution...

In general, *convolution* is the integral

$$(f * g)(t) = \int_a^b f(\tau)g(t-\tau)d\tau$$

which can be interpreted as “sliding” the reverse of g over f.

Convolution in the context of convolution kernels: summing over the possible ways of decomposing the structured object.

...Kernels

A *kernel* is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that is

- symmetric, $K(x, y) = K(y, x)$, and
- positive (semi)definite,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$$

for any $n > 0$, any $x_i, \ldots, x_n \in \mathcal{X}$ and any real numbers $c_1, \ldots, c_n$.

Properties of kernels

- The dot product of feature vectors is a kernel...

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

- ..for each kernel there is a feature space.

Properties of kernels

- Dimensionality of the feature space does not affect the computation
- We do not have to know the feature space in order to be able to do data analysis with the kernel

Closure properties

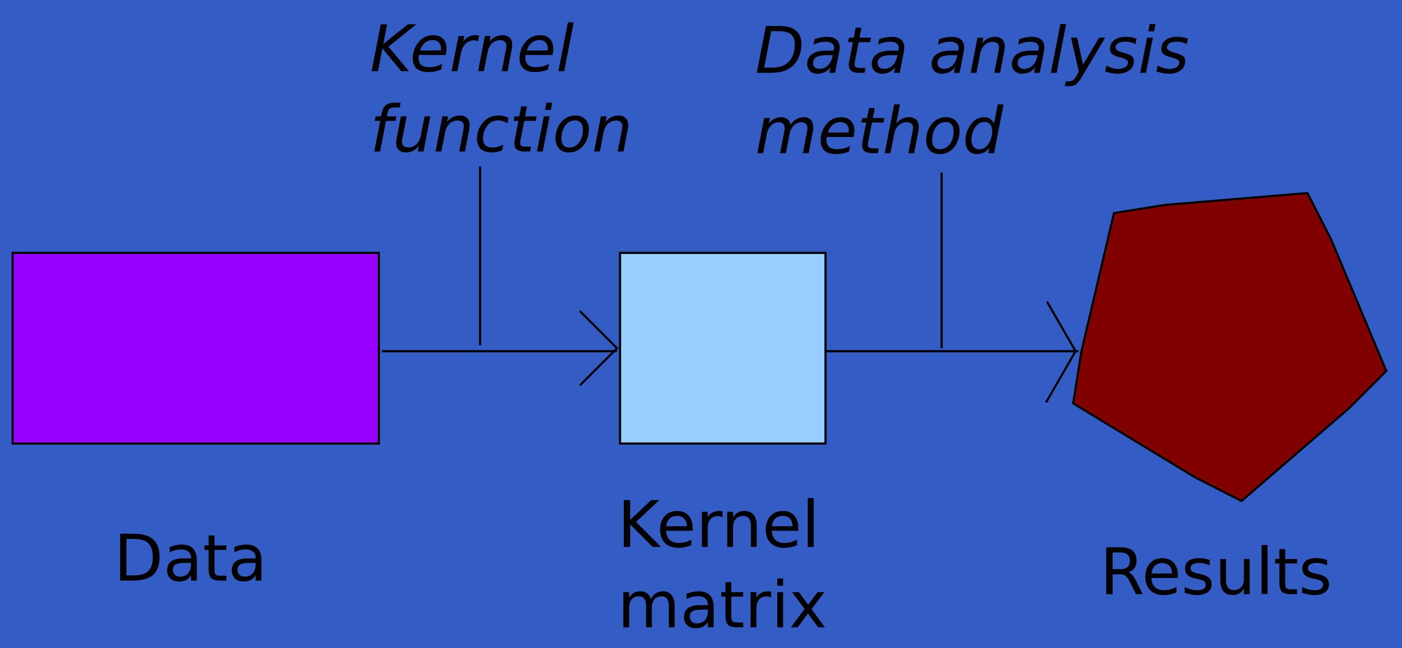
More complex kernels can be derived from simpler kernels using for instance the following closure properties.

Let K_1 and K_2 be kernels over $\mathcal{X} \times \mathcal{X}$, $a \in \mathbb{R}^+$.

- $K(x, y) = K_1(x, y) + K_2(x, y)$
- $K(x, y) = aK_1(x, y)$
- $K(x, y) = K_1(x, y)K_2(x, y)$

What are kernels good for then?

Kernels unify the interface between data and data analysis methods.



Structured data

What do we mean by *structured data*?

Data can be decomposed into smaller parts.

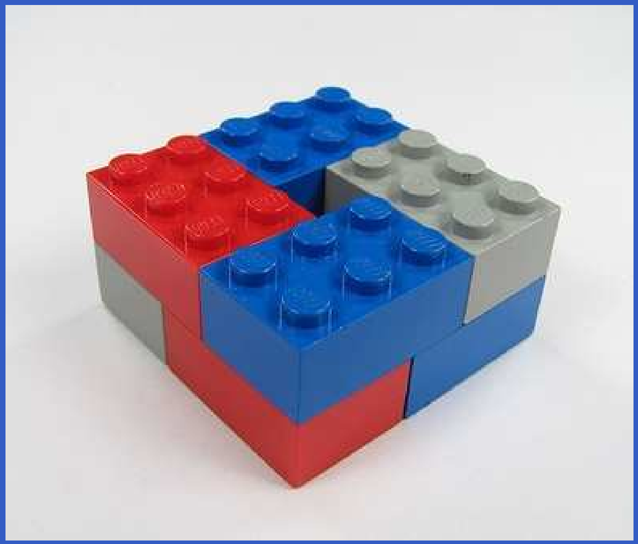
Examples of structured data

Strings

abracadabra

Trees

Legos



Examples of decompositions

Concatenations of strings

abracadabra:

a + bracadabra

ab + racadabra

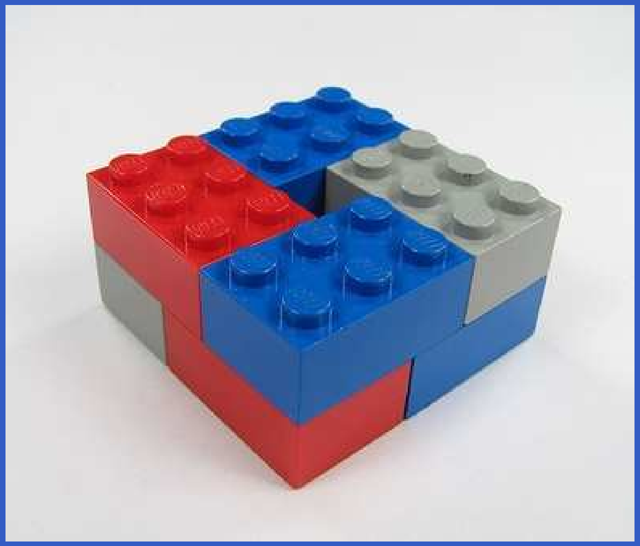
abr + acadabra

abra + cadabra

. . .

Subtrees

Legos (?)



Formalising decompositions

Let $x \in X$ be a structured object with “parts” $\vec{x} = (x_1, \ldots, x_D)$.

Each part x_d is in the set X_d.

The notion of “x_d is a part of x” is formalised by the relation R:
$R(\vec{x}, x)$ is true iff $x_1, \ldots, x_D$ are parts of x.

Formalising decompositions

Function $R^{-1}(x)$ gives the possible decompositions of x:

$$R^{-1}(x) = \{\vec{x} : R(\vec{x}, x)\}$$

Example decompositions

Example of R: String concatenations

abracadabra:
a + bracadabra
ab + racadabra
abr + acadabra
abra + cadabra
...

$$R((a, bracadabra), abracadabra)$$
$$R((ab, racadabra), abracadabra)$$
$$R((abr, acadabra), abracadabra)$$
$$R((abra, cadabra), abracadabra)$$
$$\ldots$$

$$R^{-1}(abracadabra) = \{(a, bracadabra), (ab, racadabra), \ldots\}$$

Example decompositions

Example of R: trees and subtrees

Let $\mathcal{X}$ be the set of all D-degree ordered and rooted trees. Now we can define relation R for example as follows.

$R(\vec{x}, x)$ iff $x_1, \ldots, x_D$ are the D subtrees of the root of the tree $x \in \mathcal{X}$.

Note that in this case there is only one possible decomposition and hence $|R^{-1}(x)| = 1$.

R-convolution

Let $x, y \in \mathcal{X}$ be two structured objects.

- Possible decompositions given by $R^{-1}(x)$ and $R^{-1}(y)$.

Assume that we have a kernel K_d for each "part" $x_d, y_d \in \mathcal{X}_D$.

R-convolution

Similarity between x and y can now be given in terms of the kernels K_d as the following *R-convolution* of $K_1, \ldots, K_D$,

$$K(x, y) = \sum_{\vec{x} \in R^{-1}(x), \vec{y} \in R^{-1}(y)} \prod_{d=1}^{D} K_d(x_d, y_d)$$

Note that K is symmetric.

R-convolution is a kernel

To see that

$$K(x,y) = \sum_{\vec{x} \in R^{-1}(x), \vec{y} \in R^{-1}(y)} \prod_{d=1}^{D} K_d(x_d, y_d),$$

is a kernel, we can use the closure properties of kernels for the “product” part:

- Product is a kernel: follows from $K(x,y) = K_1(x,y)K_2(x,y)$
- Sum is a kernel: **does not** follow from the sum closure - more work is needed (skipped for now)

Example: Radial basis kernels

Consider base kernels of type

$$K_d(x, y) = \exp\left(-\frac{(x_d - y_d)^2}{2\sigma^2}\right)$$

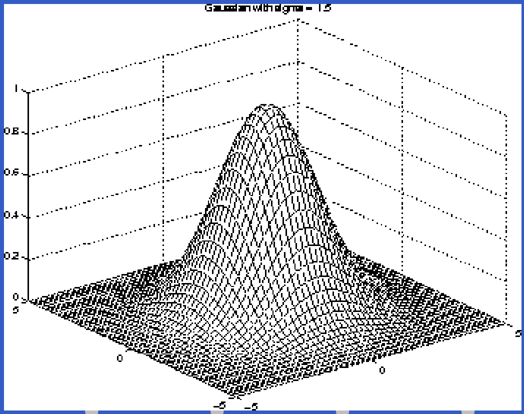
and the relation $R = \{(\vec{x}, x) : \vec{x} = x\}$. Because $R^{-1}(x)$ contains only one element, R-convolution kernel simplifies into

$$K(x, y) = \prod_{d=1}^{D} K_d(x_d, y_d).$$

Example: Radial basis kernels

$$K(x,y) = \prod_{d=1}^{D} K_d(x_d, y_d) = \prod_{d=1}^{D} \exp\left(-\frac{(x_d - y_d)^2}{2\sigma^2}\right) =$$

$$\exp\left(-\frac{\sum_{d=1}^{D}(x_d - y_d)^2}{2\sigma^2}\right) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right).$$



Example: co-rooted subtree kernel

A *complete subtree* $\tau(v)$ of a tree T at a node v is the tree obtained by taking the node v together with all nodes and edges reachable from v.

A *co-rooted subtree* of the tree T is obtained by removing a sequence of complete subtrees from the tree.

Example: co-rooted subtree kernel

Let us consider the following feature mapping ϕ for a tree T.

$$\phi_S(T) = \begin{cases} 1 & \text{if } S \text{ is a co-rooted subtree of } T \\ 0 & \text{otherwise,} \end{cases}$$

and the kernel

$$K(T_1, T_2) = \langle \phi(T_1), \phi(T_2) \rangle = \sum_{S \in \mathcal{T}} \phi_S(T_1) \phi_S(T_2)$$

where $\mathcal{T}$ is the set of all trees.

Example: co-rooted subtree kernel

To come up with the co-rooted subtree kernel, we can set our R as

$$R(S,T) \text{ iff } S \text{ is a co-rooted subtree of } T.$$

and kernel $K_d(S_1, S_2) = 1$ iff $S_1 = S_2$, and 0 otherwise. Now the kernel K would be

$$K(x,y) = \sum_{S_1 \in R^{-1}(x), S_2 \in R^{-1}(y)} K_d(S_1, S_2).$$

Note that we omit the product as $D = 1$.

Fisher kernel

A Fisher kernel works on a single generative model.

The idea is to evaluate how the model parameters should be changed to accomodate a (new) data point into the model.

Fisher kernel measures the similarity in parameter change between two data points.

Likelihood of the data

Let the likelihood of a data item x with respect to the model $m(\theta)$ for parameters θ be

$$\mathcal{L}_\theta(x) = P(x|\theta) = P_\theta(x).$$

We assume that the parameters θ have been determined.

The log-likelihood of x is $\log \mathcal{L}_\theta(x)$.

Fisher score of the data

Fisher score of x is the vector gradient of the log-likelihood

$$g(\theta, x) = \nabla_\theta \log \mathcal{L}_\theta(x) = \left(\frac{\partial \log \mathcal{L}_\theta(x)}{\partial \theta_i} \right)_{i=1}^{N}.$$

Fisher kernel

Since Fisher score is a vector, we can define the the (practical) Fisher kernel simply as

$$K(x, y) = g(\theta, x)' g(\theta, y).$$

1-dimensional Gaussian Fisher kernel

Let likelihood of the data to be defined by a Gaussian,

$$P(x|\theta) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Now the model parameters are $\theta = (\mu, \sigma)$.

The log-likelihood of a point x is

$$\log \mathcal{L}_{(\mu,\sigma)}(x) = -\frac{(x-\mu)^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma).$$

1-dimensional Gaussian Fisher kernel

We get Fisher scores by taking partial derivates,

$$g(\theta, x) = \left(\frac{x-\mu}{\sigma^2}, \frac{(x-\mu)^2}{\sigma^3} - \frac{1}{2\sigma} \right),$$

with the corresponding Fisher kernel is
$K(x, y) = g(\theta, x)' g(\theta, y)$

Structural data vs Fisher kernels

In string or tree kernels seen before the similarity is computed with respect to the similarity of components.

In Fisher kernel, the connection between the structure of objects and similarity between them is indirect.

- Objects with similar structures would cause similar changes in model parameters, and hence higher Fisher kernel values.

Markov strings & Fisher kernels

Fisher kernels can be defined for structured data as well, as long as there is an underlying generative model.

For instance, consider strings of length n, with probability $p_{u \to \sigma}$ that character σ follows the string u in the string.

Length of string u is fixed to k.

Likelihood of the string

Likelihood of the string s is then

$$\prod_{j=1}^{n-k} p_{s(j:j+k-1)\to s_{j+k}}$$

Assuming that

$$\sum_{\sigma\in\Sigma} p_{u\to\sigma} = 1, \text{ for all } u \in \Sigma^k.$$

and that the first k symbols are fixed, we get a distribution over the possible strings.

Deriving the Fisher score

The model is parametrised with arbitrary real values $\theta = (a_{u \to \sigma})_{u \in \Sigma^k, \sigma \in \Sigma}$,

$$p_{u \to \sigma} = \frac{a_{u \to \sigma}}{\sum_{\sigma \in \Sigma} a_{u \to \sigma}}.$$

The Fisher score is then

$$g(\theta, s)_{u \to \sigma} = \frac{\partial \log \mathcal{L}_\theta(s)}{\partial a_{u \to \sigma}}$$

Deriving the Fisher score

$$\begin{aligned} g(\theta, s)_{u\to\sigma} &= \frac{\partial \log \mathcal{L}_\theta(s)}{\partial a_{u\to\sigma}} \\ &= \sum_{j=1}^{n-k} \frac{\partial \log p_{s(j:j+k-1)\to s_{j+k}}}{\partial a_{u\to\sigma}} \end{aligned}$$

Deriving the Fisher score

$$g(\theta, s)_{u\to\sigma} = \sum_{j=1}^{n-k} \frac{\partial \log p_{s(j:j+k-1)\to s_{j+k}}}{\partial a_{u\to\sigma}}$$

By considering parameter setting where $a_{u\to\sigma} = p_{u\to\sigma} = |\Sigma|^{-1}$ this simplifies to

$$\begin{aligned} g(\theta, s)_{u\to\sigma} &= |\{(v_1, v_2) : s = v_1 u\sigma v_2\}||\Sigma| - \\ &\quad |\{(v_1, v_2) : s = v_1 u v_2\}|. \end{aligned}$$

Summary

- Convolution kernels give a general framework for defining kernels for structured data
- May lead into impractical kernels
- Much design details hidden in choice of component (base) kernels and relation R.
- Fisher kernels use the log-likelihood gradients as a feature vector