Two approaches for kernel construction

Leo Lahti

April 6, 2004

Articles

- Marginalized kernels for biological sequences (Tsuda et al.: Bioinformatics 18 suppl., 2002.)
- Covariance kernels from Bayesian generative models (Seeger: NIPS 14, 2002)

Overview

- Motivation
- Marginalized kernels
- Covariance kernels
- Examples
- Conclusions

Motivation

- kernel can be used to encode distance information between objects
- useful distance depends on the problem

$\rightarrow$ how to construct a suitable kernel for a given problem?

Two approaches for kernel construction from redundant data are presented.

1. Marginalized kernels

Designing kernels from *latent variable models*

Construction steps:

1. construct a *joint kernel* $K_z(z, z')$ for the whole model with visible and hidden variables

2. take expectation of the joint kernel over hidden variables

Marginalized kernel

Using above steps we get *marginalized kernel*

$$K(x, x') = \sum_{h,h'} p(h|x)p(h'|x)K_z(z, z').$$

- posteriors $p(h|x)$ and $p(h'|x)$ are unknown in general and have to be estimated
- suitable joint kernel depends on the problem

Marginalized kernels - an example

Sequence comparisons can be performed using the *count kernel*:

$$K(x, x') = \sum_k c_k(x) c_k(x'),$$

$$c_k(x) = \frac{1}{m} \sum_i I(x_i = k)$$

- $c_k(x)$ is the percentage of occurences of symbol k in sequence x.
- count kernel is succesfully used in text processing literature
- for biological sequences we need also context information

Constructing a joint kernel

- DNA structure can be represented by four symbols (A,C,G,T nucleotides)
- DNA is divided in coding and non-coding regions
- this information is typically hidden for genomic sequences

Let us study genomic sequence x and indicator h with 1 and 2 denoting coding and non-coding region for the corresponding nucleotide:

$$\mathbf{h} = 122122122$$
$$\mathbf{x} = \text{ACGGTTCAA}$$

Constructing a joint kernel

- denote z=(x,h) as a 'joint variable'
- represent the joint kernel for visible nucleotide and hidden context information as

$$K_z(z, z') = \sum_k \sum_l c_{kl}(z) c_{kl}(z'),$$

$$c_{kl}(z) = \frac{1}{m} \sum_i I(x_i = k, h_i = l).$$

- this has the same form as the usual count kernel.

Marginalizing the joint kernel

Marginalizing the joint kernel over hidden variables we have

$$K(x,x') = \sum_{\mathbf{h},\mathbf{h}'} p(\mathbf{h}|x)p(\mathbf{h}'|x)K_z(z,z') = \sum_{k,l} \gamma_{kl}(x)\gamma_{kl}(x'),$$

where

$$\gamma_{kl}(x) = \frac{1}{m}\sum_{i}\sum_{h_i} p(h_i|x)I(x_i = k, h_i = l).$$

- unknown $p(h_i|x)$ can be straightforwardly estimated using HMM
- suitable HMMs are readily available for biological sequences

Fisher kernel

Fisher kernel is defined by

$$K_f(x, x') = s(x, \theta')^T Z^{-1}(\theta') s(x, \theta'),$$

with Fisher score

$$s(x, \theta') = \nabla_\theta logp(x|\theta')$$

and Fisher information matrix

$$Z(\theta') = E[s(x, \theta') s(x, \theta')^T |\theta'].$$

Connection to Fisher kernel

Let us adopt a latent variable model $p(x|\theta) = \sum_h p(x, h|\theta)$.

The Fisher score now takes the form

$$\begin{aligned} s(x, \theta') &= \nabla_\theta log p(x|\theta') \\ &= \nabla_\theta log \sum_h p(x, h|\theta) \\ &= \sum_h p(h|x, \theta') \nabla_\theta log p(x, h|\theta'). \end{aligned}$$

The corresponding Fisher kernel is

$$\begin{aligned} K_f(x, x') &= s(x, \theta')^T Z(\theta')^{-1}(\theta') s(x, \theta') \\ &= \nabla_\theta log p(x|\theta')^T Z(\theta')^{-1} \nabla_\theta log p(x|\theta') \\ &= \sum_{h,h'} p(h|x, \theta') p(h'|x', \theta') K_z(z, z') \end{aligned}$$

Connection to Fisher kernel

We notice that for the Fisher kernel, the joint kernel is described as

$$K_z(z, z') = \nabla_\theta logp(x, h|\theta')^T Z(\theta')^{-1} \nabla_\theta logp(x', h'|\theta').$$

$\rightarrow$ Fisher kernel is a special case of marginalized kernels, with the above joint kernel.

Evaluation of marginalized kernels

- the joint kernel should be designed for the given purpose.
- joint kernel and probabilistic model $p(x|\theta)$ can be completely separated. This allows utilizing higher order information with a first order HMM.
- useful when context information is crucial
- MCK performed better in bacterial classification in comparison with the Fisher kernel

2. Mutual information for learning covariance kernels

Mutual information (MI) for learning covariance kernels from unlabeled data

- kernel should encode our notion of similarity
- clusters in probability distribution $P(x)$ contain similar samples and samples in different clusters are dissimilar
- mutual information is one way to represent such similarity
- mutual information can be computed for unlabeled data

$\rightarrow$ let's derive a kernel using mutual information

Deriving MI-kernel

Construction steps:

1. choose model family $\{P(\mathbf{x}|\theta)\}$ and a prior distribution $P(\theta)$ for the parameters θ

2. fit generative models to unlabeled data D_u:

$$P(\theta|D_u) \propto P(D_u|\theta)P(\theta),$$

3. relying too much on unlabeled information may result in the lack of robustness: adjust the effect of unlabeled data in the kernel learning process using *model-trust scaling*

$$P_{med}(\theta|\lambda) \propto P(D_u|\theta)^{\lambda/n}P(\theta), \lambda \in [0, n],$$

4. build covariance kernel using this posterior information

Definitions

Let us define joint distribution

$$Q(x_1, x_2) := \int P_{med}(\theta) P(x_1|\theta) P(x_2|\theta) d\theta,$$

where $P_{med}(\theta)$ is a *mediator distribution*.

From the joint distribution we derive *MI-score*:

$$I(x_1, x_2) = log \frac{Q(x_1, x_2)}{Q(x_1) Q(x_2)},$$

where

$$Q(x) = \int Q(x, x') dx'.$$

However, this is not *positive definite* as is required for a proper kernel.

MI-kernel by exponential embedding

To get a positive definite kernel, use exponential embedding:

$$
\begin{aligned}
K(x_1, x_2) &= exp(-\frac{1}{2}[I(x_1, x_1) + I(x_2, x_2)] + I(x_1, x_2)) \\
&= \frac{Q(x_1, x_2)}{\sqrt{Q(x_1, x_1)Q(x_2, x_2)}}.
\end{aligned}
$$

This is called *MI kernel*.

Conclusions

- it is not easy to encode our notion of similarity
- this is problem especially when data is only partially known
- kernel design is not a straightforward task
- kernel learning approaches may be useful