статья

> Kernel methods

>> Матрица грамма

>> Kernel machines

>> Представление ядра через скалярное произведение

> Iterative methods and pre-conditioning

>> Gram matrix evaluation and matrix operations can be time expensive

>> there are matrix-free methods:

>>> conjugate gradient

>>> power iteration

>> Three common kernel machines that can use matrix free methods

>>> Gaussian Processes; Gaussian processes for machine learning (2016), The MIT Press

>>> SVMs

>>> Kernel PCA, which can be solved with power iteration.

>> pre-conditioning matrix

> random partition.

>> defenition

>> Well-known Bayesian models on random partitions:

>>> Chinese Restaurant Process (Aldous, 1985)

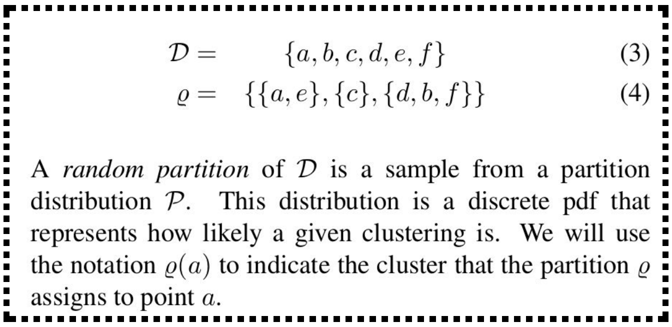
>>> the Pitman-Yor Process (Pitman & Yor, 1997)

>>> the Mondrian Process (Daniel M. Roy, 2009)

> Defining distributions with programs: **Thus any program that takes a datset as input and outputs a random partition defines a partition distribution.**

Random Partition Kernels

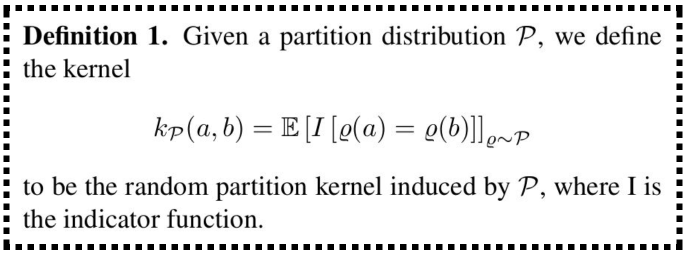
> **Defenitions**



$$\mathcal{D} = \{a, b, c, d, e, f\} \tag{3}$$

$$\varrho = \{\{a, e\}, \{c\}, \{d, b, f\}\} \tag{4}$$

A *random partition* of $\mathcal{D}$ is a sample from a partition distribution $\mathcal{P}$. This distribution is a discrete pdf that represents how likely a given clustering is. We will use the notation $\varrho(a)$ to indicate the cluster that the partition ϱ assigns to point a.

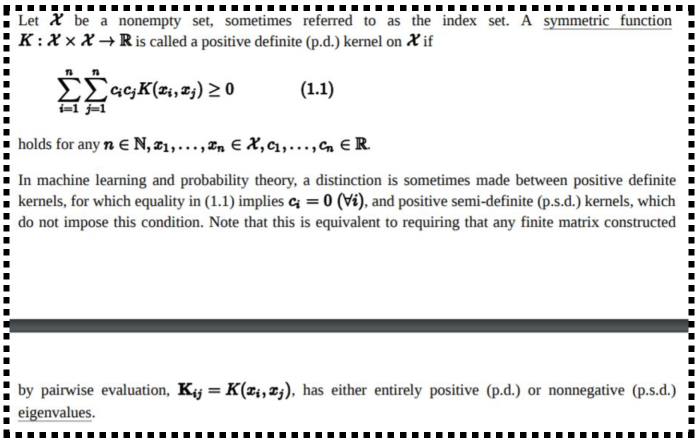


Definition 1. Given a partition distribution $\mathcal{P}$, we define the kernel

$$k_{\mathcal{P}}(a, b) = \mathbb{E}\left[I\left[\varrho(a) = \varrho(b)\right]\right]_{\varrho \sim \mathcal{P}}$$

to be the random partition kernel induced by $\mathcal{P}$, where I is the indicator function.

>> PSD kernel - Positive-definite kernel



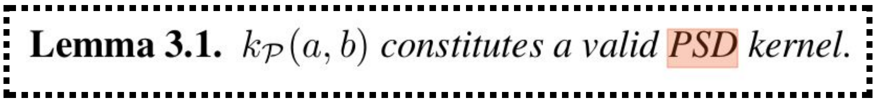
Let $\mathcal{X}$ be a nonempty set, sometimes referred to as the index set. A symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive definite (p.d.) kernel on $\mathcal{X}$ if

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0 \tag{1.1}$$

holds for any $n \in \mathbb{N}, x_1, \dots, x_n \in \mathcal{X}, c_1, \dots, c_n \in \mathbb{R}$.

In machine learning and probability theory, a distinction is sometimes made between positive definite kernels, for which equality in (1.1) implies $c_i = 0$ $(\forall i)$, and positive semi-definite (p.s.d.) kernels, which do not impose this condition. Note that this is equivalent to requiring that any finite matrix constructed

by pairwise evaluation, $\mathbf{K}_{ij} = K(x_i, x_j)$, has either entirely positive (p.d.) or nonnegative (p.s.d.) eigenvalues.

> 

Lemma 3.1. *$k_{\mathcal{P}}(a, b)$ constitutes a valid PSD kernel.*

> m - approximate Random Partition Kernel

Definition 2. The m-approximate Random Partition Kernel is the fraction of times that ϱ assigns a and b to the same cluster over m samples.

$$k_{\mathcal{P}}(a,b) \approx \tilde{k}_{\mathcal{P}}(a,b) = \frac{1}{m}\sum_{\varrho \sim \mathcal{P}}^{m} k_{\varrho}(a,b)$$

> and it's variance

Lemma 3.2. *If the samples from $\mathcal{P}$ are independent then the bound on the variance of the approximation to $k_{\mathcal{P}}(a,b)$ is $O\left(\frac{1}{m}\right)$.*

> The Colonel's Cocktail Party

This process of evaluating the kernel.

> Efficient evaluation

$$(\mathbf{K}_{\varrho}\vec{v})_i = \sum_{j \in \varrho(i)} \vec{v}_j$$

It follows that the kernel matrix for an **m-approximate Random Partition** Kernel requires mN space and 2mN operations for a matrix-vector product.

>> pre-conditioning matrix: Where σ is set as a small constant to ensure the matrix is invertible.

$$\mathbf{B} = m\sum_{\varrho \in \mathcal{P}}^{m} (\mathbf{K}_{\varrho} + \sigma\mathbf{I})^{-1}$$

invertible. Due to the simple form of the individual cluster matrices, we can compute $(\mathbf{K}_{\varrho} + \sigma\mathbf{I})^{-1}\vec{v}$ in only $2N$ operations and $\mathbf{B}\vec{v}$ in $2mN$ operations using the analytic form in Eq 10.

$$\left((\mathbf{K}_{\varrho} + \sigma\mathbf{I})^{-1}\vec{v}\right)_i = \frac{1}{\sigma}\vec{v}_i - \frac{1}{|\varrho(i)| + \sigma}\sum_{j \in \varrho(i)} \vec{v}_j \quad (10)$$

This small multiplicative overhead to the iterative solver greatly reduces the condition number and the number of iterations required for a solution in most cases.

> Example random partitions:

>> STOCHASTIC CLUSTERING ALGORITHMS

>> ENSEMBLED TREE METHODS

> Data-defined partitions:

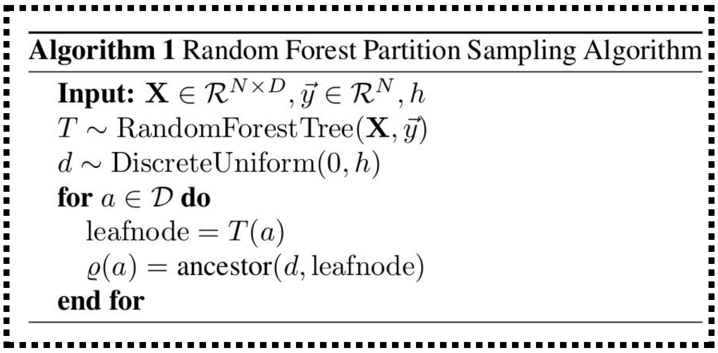
>> COLLABORATIVE FILTERING

>> TEXT ANALYSIS

> Exmaple Kernels

>> RANDOM FOREST KERNEL

- алгоритм основан на рандомном выборе высоты и дерева из RF



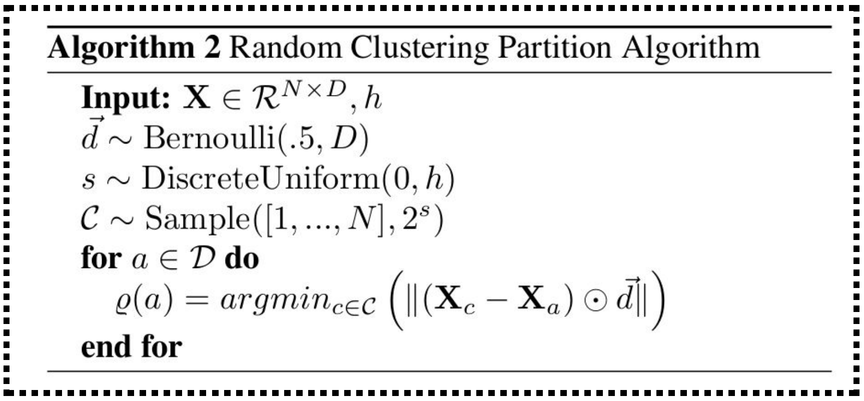
Algorithm 1 Random Forest Partition Sampling Algorithm

```
Input: $\mathbf{X} \in \mathcal{R}^{N \times D}, \vec{y} \in \mathcal{R}^N, h$
$T \sim \text{RandomForestTree}(\mathbf{X}, \vec{y})$
$d \sim \text{DiscreteUniform}(0, h)$
for $a \in \mathcal{D}$ do
    $\text{leafnode} = T(a)$
    $\varrho(a) = \text{ancestor}(d, \text{leafnode})$
end for
```

- **When used in Gaussian Processes, it is a prior over piece-wise constant functions.** This is true of all Random Partition Kernels
- **The GP prior is also non-stationary.** A property of real-world datasets that is often not reflected in analytically derived kernels is non-stationarity; where the kernel depends on more than just the distance between two points.
- **Such kernel has no hyper-parameters.** there is no costly hyper-parameter optimization stage. This normal extra “training” phase in GPs and SVMs is replaced by the training of the random forests, which in most cases would be more computationally efficient for large datasets.
- **It is a supervised kernel.** While this could reasonably lead to overfitting for regression/classification, the experiments do not show any evidence for this.

>> Random Clustering Partition Algorithm

- First, sample a subset of the dimensions, then select a random number of cluster centers. Assign every point to the cluster associated with its nearest center measured only on the subsampled dimensions.



Algorithm 2 Random Clustering Partition Algorithm

```
Input: $\mathbf{X} \in \mathcal{R}^{N \times D}, h$
$\vec{d} \sim \text{Bernoulli}(.5, D)$
$s \sim \text{DiscreteUniform}(0, h)$
$\mathcal{C} \sim \text{Sample}([1, ..., N], 2^s)$
for $a \in \mathcal{D}$ do
    $\varrho(a) = argmin_{c \in \mathcal{C}} \left( \|(\mathbf{X}_c - \mathbf{X}_a) \odot \vec{d}\| \right)$
end for
```

- This cluster scheme was constructed largely for it’s simplicity and speed, scaling as O(N).
- For Gaussian Processes it results in a distribution over piece-wise constant functions that is non-stationary, with smoother variances than the Random Forest kernel.
- It is also worth noting that when being used for classification/regression the Fast Cluster kernel can easily be used in a “semi-supervised” manner, by training it on a larger set of X than is used for the final learning task.

> Experiments

>> We compare the results of using the Random Forest kernel against the Linear kernel and the Radial Basis Function with and without Automatic Relevance Detection using standard GP training methodology on 6 real-world regression problems from the UCI Machine Learning repository.

>> The evaluation criteria that is most important for a Gaussian Process is the test log-likelihood, as this directly shows how well the posterior distribution has been learnt. It factors in not only how well the test data has been predicted, but also the predicted variances and covariances of the test data.

>> As GPs are often used simply as a point estimate regression method, it is still important to assess the kernels performance with respect to MSE. the Random Forest kernel predicts better in the majority of cases, though the improvement is not as pronounced as the improvement in log-likelihood. This shows that while the the resulting joint predictive posterior for the standard kernels can be very poorly fit, they may still result in accurate posterior mean predictions.

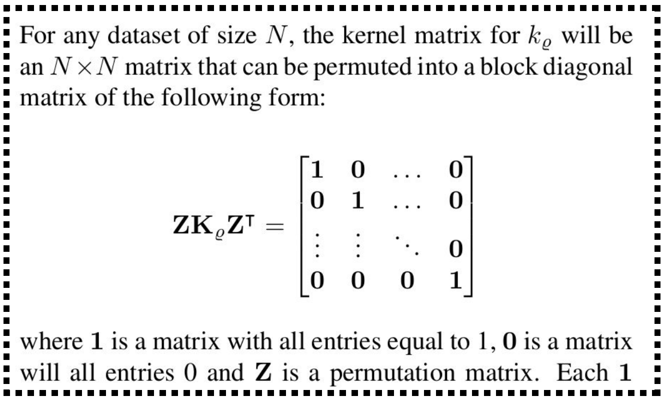
> **Conclusion**

We have presented the Random Partition Kernel, a novel method for constructing effective kernels based on a connection between kernels and random partitions. The example kernels constructed using this method, the Random Forest kernel and Fast Cluster kernel, both show excellent performance in modelling real-world regression datasets with Gaussian Processes, substantially outperforming other kenels.

> Некоторые мои вопросы:

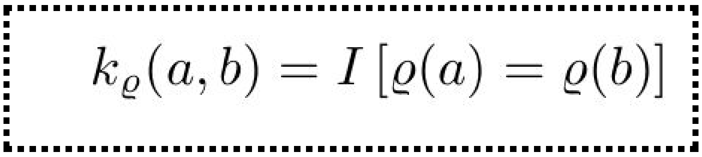
1) Если Random Partition Kernel это ядро, то в каком оно признаковом пространстве?

2) В доказательстве они переходили к виду матрицы ядра, почему она так выглядит?


For any dataset of size N, the kernel matrix for k_ϱ will be an $N \times N$ matrix that can be permuted into a block diagonal matrix of the following form:

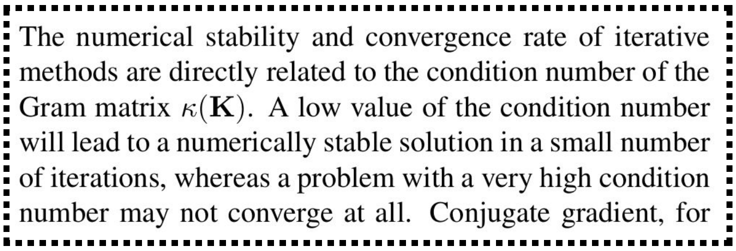
$$\mathbf{Z}\mathbf{K}_\varrho\mathbf{Z}^\intercal = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

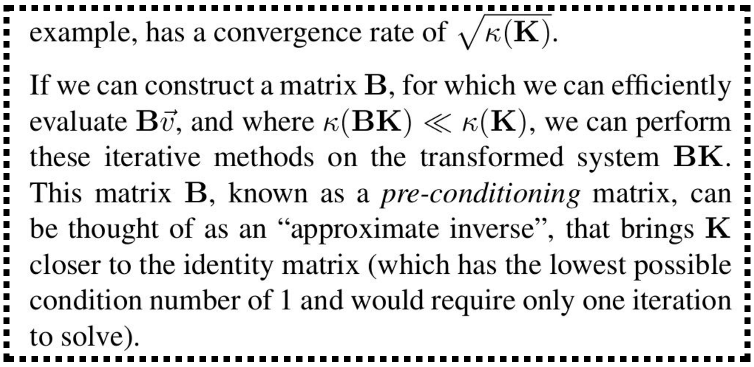
where $\mathbf{1}$ is a matrix with all entries equal to 1, $\mathbf{0}$ is a matrix will all entries 0 and $\mathbf{Z}$ is a permutation matrix. Each $\mathbf{1}$


$$k_\varrho(a,b) = I\left[\varrho(a) = \varrho(b)\right]$$

Хотя вроде понятно, векторы просто переставили так чтоб они шли подряд по кластерам

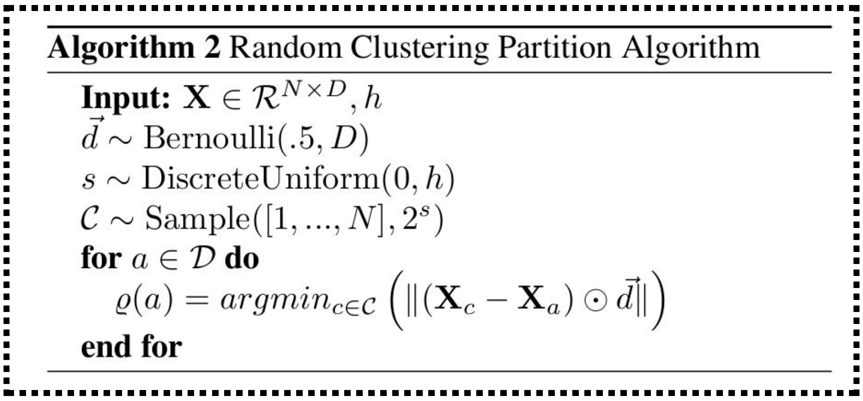
3) Какие есть методы для обнавления/ обучения ядра? и зачем нужна **pre-conditioning matrix**?


The numerical stability and convergence rate of iterative methods are directly related to the condition number of the Gram matrix $\kappa(\mathbf{K})$. A low value of the condition number will lead to a numerically stable solution in a small number of iterations, whereas a problem with a very high condition number may not converge at all. Conjugate gradient, for


example, has a convergence rate of $\sqrt{\kappa(\mathbf{K})}$.

If we can construct a matrix $\mathbf{B}$, for which we can efficiently evaluate $\mathbf{B}\vec{v}$, and where $\kappa(\mathbf{BK}) \ll \kappa(\mathbf{K})$, we can perform these iterative methods on the transformed system $\mathbf{BK}$. This matrix $\mathbf{B}$, known as a *pre-conditioning* matrix, can be thought of as an "approximate inverse", that brings $\mathbf{K}$ closer to the identity matrix (which has the lowest possible condition number of 1 and would require only one iteration to solve).

4) Как в Random Clustering Partition Algorithm считается Xc, если пока что мы насэмплировали только номера кластеров а не их параметры?


Algorithm 2 Random Clustering Partition Algorithm

```
Input: $\mathbf{X} \in \mathcal{R}^{N \times D}, h$
$\vec{d} \sim$ Bernoulli$(.5, D)$
$s \sim$ DiscreteUniform$(0, h)$
$\mathcal{C} \sim$ Sample$([1, ..., N], 2^s)$
for $a \in \mathcal{D}$ do
    $\varrho(a) = argmin_{c \in \mathcal{C}}\left(\|(\mathbf{X}_c - \mathbf{X}_a) \odot \vec{d}\|\right)$
end for
```