Parameter estimation for discrete determinantal structures

Dissertation submitted for the degree of
MASTER OF SCIENCE IN INTERDISCIPLINARY MATHEMATICS

Johannes Müller

June 18, 2018

UNIVERSITY OF WARWICK

DEPARTMENT OF MATHEMATICS

Acknowledgements

I would like to thank the dreamteam for being such an eggcelent friend-group and for all the pun we had during the last year.

Abstract

Determinantal point processes are random subsets that exhibit a diversifying behaviour in the sense that the randomly selected points tend to be not similar in some way. This repellent structure first arrose in theortical physics and pure mathematics, but they have recently been used to model a variety of many real world scenarios in a machine learning setup. We aim to give an overview over the main ideas of this approach which is easily accessible even without prior knowledge in the area of machine learning and sometimes omit technical calculations in order to keep the focus on the concepts.

Contents

Chapter I

Introduction and motivating examples

I.1 Motivation

I.2 Previous work

I.3 Aim and outline of the dissertation

Chapter II

Determinantal points processes: Basic notions and properties

II.1 Historical remarks

II.2 Definitions and properties

Let in the following $\mathcal{Y}$ be a finite set, which we call the *ground set* and $N := |\mathcal{Y}|$. A *point process* on $\mathcal{Y}$ is a random subset of $\mathcal{Y}$, i.e. a random variable with values in the powerset $2^{\mathcal{Y}}$. Usually we will identify this random variable with its law $\mathbb{P}$ and thus refer to probability measures $\mathbb{P}$ on $2^{\mathcal{Y}}$ as point processes and will not distinguish those objects. Let in the following $\mathbf{Y}$ denote a random subset drawn according to $\mathbb{P}$. We call $\mathbb{P}$ a *determinantal point process*, or in short a DPP, if we have

$$\mathbb{P}(A \subseteq \mathbf{Y}) = \det(K_A) \quad \text{for all } A \subseteq \mathcal{Y} \tag{2.1}$$

where K is a symmetric matrix indexed by the elements in $\mathcal{Y}$ and K_A denotes the restriction of the matrix K to indices in A. We call K the *marginal kernel* of the DPP and note immediately that

cite or explain?

K is necessarily non negative definite. Further it can be shown (cf. page 3 in [Borodin, 2009])

have a look at this and maybe explain it!

that also the complement of a DPP is a DPP with marginal kernel $I - K$ where I is the identity matrix, i.e.

$$\mathbb{P}(A \subseteq \mathbf{Y}^c) = \det(I_A - K_A).$$

Thus we conclude $I - K \geq 0$ and obtain $0 \leq K \leq I$. This actually turns out to be sufficient

why? explain!

for K to define a DPP through (2.1) (cf. [Kulesza et al., 2012]). We call the elements of $\mathcal{Y}$ *items* and by choosing $A = \{i\}$ and $A = \{i, j\}$ for $i, j \in \mathcal{Y}$ and using (2.1) we obtain the probabilities of their occurrence

$$\begin{aligned} \mathbb{P}(i \in \mathbf{Y}) &= K_{ii} \quad \text{and} \\ \mathbb{P}(i, j \in \mathbf{Y}) &= K_{ii}K_{jj} - K_{ij}^2 = \mathbb{P}(i \in \mathbf{Y}) \cdot \mathbb{P}(j \in \mathbf{Y}) - K_{ij}^2, \end{aligned} \tag{2.2}$$

Thus the appearances of the two items i and j are always negatively correlated. This negative correlation is exactly what causes the diversifying behaviour of determinantal point processes. In practice one usually models the negative correlations to be high between items that are similar in some notion. For example in a spatial setting being similar could mean being close together and in this case the selected items would tend to be not very close together. This is repulsive

insert picture!

behaviour can be seen in Figure.

In this light the fact that also $\mathbf{Y}^c$ exhibits negative correlations becomes less surprising. Since the set $\mathbf{Y}$ tends to spread out due to the repulsion in (2.2), the complement, which is nothing but the gaps that are left after eliminating the elements in $\mathbf{Y}$, tend to show a non clustering behaviour as well.

L-ensembles

Let us now introduce an important subclass of DPPs, namely the ones where not only the marginal probabilities can be expressed through a suitable kernel, but also the elementary probabilities. Because this will be convenient for us in the sequel, we will restrict ourselves to this case from now on. If we have even $K < I$, then we define the *elementary kernel*

really? what about sampling?

$$L := K(I - K)^{-1} \tag{2.3}$$

which specifies the elementary probabilities since one can check

$$\mathbb{P}(A = \mathbf{Y}) = \frac{\det(L_A)}{\det(I + L)} \quad \text{for all } A \subseteq \mathcal{Y}. \tag{2.4}$$

Conversely for any $L \geq 0$ a DPP can be defined via (2.2) and the corresponding marginal kernel is given by the inversion of (2.3)

$$K = L(I + L)^{-1}.$$

We call DPPs which arise this way L *ensembles*.

The quality diversity decomposition

Note that any symmetric, positive semidefinite matrix L can be written as a Gram matrix

$$L = B^T B$$

where $B \in \mathbb{R}^{D \times N}$ whenever D is larger than the rank $\mathrm{rk}(L)$ of L. For example one could take the spectral decomposition $L = U^T C U$ of L and set $B := \sqrt{C} U$ and eventually drop some zero rows from $\sqrt{C}$. Let B_i denote the i-th column of B and write it as the product $q_i \cdot \phi_i$ where $q_i \geq 0$ and $\phi_i \in \mathbb{R}^D$ such that $\|\phi_i\| = 1$. This yields the representation

$$L_{ij} = q_i \phi_i^T \phi_j q_j =: q_i S_{ij} q_j$$

and we call q_i the *quality* of the item $i \in \mathcal{Y}$ and ϕ_i the *diversity feature vector* of i and S the *similarity matrix*. Since we will use this decomposition multiple times, we fix its properties.

Proposition 2.1 (Quality diversity parametrisation). *Let $D \in \mathbb{N}$ and let $\mathbb{S}_D$ denote the sqhere in $\mathbb{R}^D$. Further let $\mathbb{R}^{N \times N}_{sym,+}$ be the set of symmetric positive semidefinite $N \times N$ matrices. The quality diversity parametrisation is a continuous and surjective mapping*

its not a bijection!!!

$$\Psi \colon \mathbb{R}^N_+ \times \mathbb{S}^N_D \to \left\{ L \in \mathbb{R}^{N \times N}_{sym,+} \mid \mathrm{rk}(L) \leq D \right\}, \quad (q, \phi) \mapsto \left(q_i \phi_i^T \phi_j q_j \right)_{1 \leq i,j \leq N}.$$

Remark 2.2. (*i*) In the case $D = N$ the quality diversity decomposition gives a parametrisation of the whole symmetric positive definite $N \times N$ matrices.

(*ii*) Note that this parametrisation is not unique, i.e. Ψ is not injective. For example the identity matrix I can be parametrised by any orthonormal system ϕ and $q = (1, \ldots, 1)^T$.

(*iii*) One can without any problems consider diversity features ϕ_i in an abstract Hilbert space $\mathcal{H}$. However we will not need this in the remainder and thus restrict ourselves to the easier case Euklidean diversity features.

(*iv*) We call every preimage (q, S) of L under Ψ *quality diversity decomposition* of L.

The quality diversity decomposition will provide some useful expressions. For example the elementary probabilities take the form

$$\mathbb{P}(A = \mathbf{Y}) \propto \det\big((B^T B)_A\big) = \left(\prod_{i \in A} q_i^2\right) \cdot \det(S_A) \quad \text{for all } A \subseteq \mathcal{Y}. \tag{2.5}$$

An intuitive understanding of the quality diversity decomposition will play a central role in the modelling process of real world phenomena through DPPs. To get this we can think of $q_i \geq 0$ as a measure of how important or high in quality the item is and the diversity feature vector $\phi_i \in \mathbb{R}^D$ can be thought of as some kind of state vector that consists of internal quantities that describe the item i in some way. Further we interpret the scalar product $\phi_i^T \phi_j \in [0, 1]$ as a measure of similarity between the items i and j which justifies the name similarity matrix for S. Note that if i and j are perfectly similar or antisimilar, i.e. $\phi_i^T \phi_j = \pm 1$, then they can not occur at the same time, since

$$\mathbb{P}(i, j \in \mathbf{Y}) = \det \begin{pmatrix} 1 & \pm 1 \\ \pm 1 & 1 \end{pmatrix} = 0.$$

If we identify i with the vector $B_i = q_i \phi_i \in \mathbb{R}^D$, we can obtain a geometric interpretation of (2.5) since $\det\big((B^T B)_A\big)$ is the volume that is spanned by the columns $B_i, i \in A$, which is visualised in II.1. This volume increases if the lengths of the edges that correspond to the quality increase and decrease when the similarity feature vectors point into more similar directions.

Since we will use one form of diversity features multiple times, we will now give a short general formulation of it. Let $\mathcal{R} = \{r_1, \ldots, r_D\}$ be a finite set which we will call the *reference set* and its elements the *reference points*. Further let

$$d : \mathcal{Y} \times \mathcal{R} \to \mathbb{R}_+, \quad f : \mathbb{R}_+ \to \mathbb{R}$$

mappings. Usually $d(i, r)$ will be interpreted as a measure of similarity between an item $i \in \mathcal{Y}$ and a reference point $r \in \mathcal{R}$ and will typically be given by a metric on a larger space that contains both $\mathcal{Y}$ and $\mathcal{R}$. One can now model $\phi_i \in \mathbb{R}^{\mathcal{R}}$ via

$$(\phi_i)_r \propto f\big(d(i, r)\big) \quad \text{for } r \in \mathcal{R}$$

Thus the diversity feature vector ϕ_i stores how similar the item i is to all reference points and the scalar product $\phi_i^T \phi_j$ will be close to one, if the items i and j have approximately the same degrees of similarity to the reference points. It shall be noted that the choice of the D, the number of reference points bounds the rank of the kernel L and therefore of the largest subset that occurs with positive probability. Indeed we have $\mathrm{rk}(L) \leq D$ and for $A \subseteq \mathcal{Y}$ with more than D elements $\det(L_A) = 0$ and therefore $\mathbb{P}(A) = 0$.

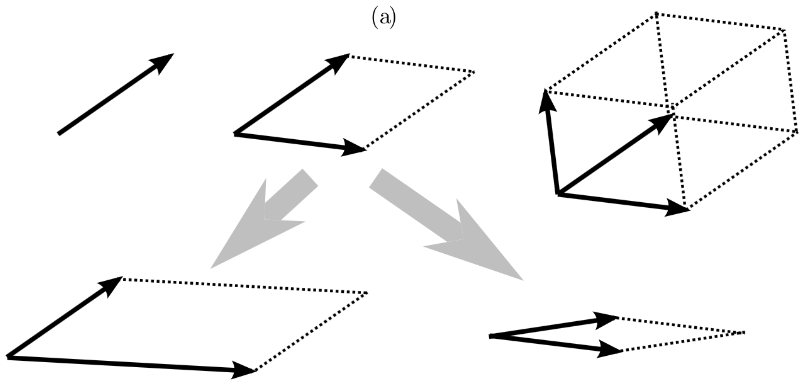

Figure II.1.: Taken from [Kulesza et al., 2012]; the first line (a) illustrates the volumes spanned by vectors, and in the second line it can be seen how this volume increases if the length – associated with the quality – increases (b) and decreases if they become more similar in direction which we interpret as two items becoming more similar (c)

In the fourth chapter we will see that there is a natural choice for the mapping d in most cases, at least in the ones where $\mathcal{Y}$ consists of points in a metric space. On the other hand the choice of f is crucial since it determines the structure and strength of the repulsion.

One last property of DPPs that we shall mention is the fact that the negative correlations of the DPP posses a transient property in the sense, that if i and j and j and k are similar, then i and k are also similar. This is due to the fact

$$\left\|\phi_i - \phi_j\right\|^2 = \|\phi_i\|^2 + \left\|\phi_j\right\|^2 - 2\phi_i^T\phi_j = 2(1 - \phi_i^T\phi_j)$$

and thus

$$\sqrt{1 - \phi_i^T\phi_k} = \frac{1}{2}\|\phi_i - \phi_k\| \leq \frac{1}{2}\left(\left\|\phi_i - \phi_j\right\| + \left\|\phi_j - \phi_k\right\|\right) = \sqrt{1 - \phi_i^T\phi_j} + \sqrt{1 - \phi_j^T\phi_k}.$$

reformulate that part!

II.3 Variations of DPPs

Conditional DPPs

A *conditional DPP* is a collection of DPPs indexed by $X \in \mathcal{X}$, where X is called the *input* of the conditional DPP. Thus for every $X \in \mathcal{X}$ we get a finite set $\mathcal{Y}(X)$ and a determinantal point process $\mathbb{P}(\cdot \mid X)$ on $\mathcal{Y}(X)$ which is given by the elementary kernel $L(X)$, i.e.

$$\mathbb{P}(A|X) \propto \det\left(L_A(X)\right) \quad \text{for all } A \subseteq \mathcal{Y}(X).$$

Further we denote the quality and diversity features of the conditional DPP by $q_i(X)$ and $\phi_i(X)$ respectively.

It is not immediately clear why one would want to model a family of DPPs as a conditional DPP rather than as seperate DPPs. The reason for this is that one wants to estimate the kernels $L(X)$

for every $X \in \mathcal{X}$. However if we would do this naively we would need to observe each of the DPPs $\mathbb{P}(\cdot \mid X)$ individually which is often not possible. Thus one hopes to not only memorise the kernels $L(X)$ for every single input $X \in \mathcal{X}$ but rather to learn the mapping L that assigns every input X its elementary kernel $L(X)$. If one achieved this task, one would be able to simulate and predict a DPP that one has not observed so far just by the knowledge about some DPPs that belong to the same conditional DPP. Of course this can only work if we assume some regularity or a certain structure of the function L which we will do in the third chapter where we put those consideration into a precise framework.

Fixed size or k-DPPs

Structured DPPs

Say something about number of parameters

We call a DPP *structured DPP* or short sDPP if the ground set is the cartesian product of some other set $\mathcal{M}$, which we will call the *set of parts*, i.e. if we have

$$\mathcal{Y} = \mathcal{M}^R = \left\{y_i = (y_i^r)_{r=1,\dots,R} \mid i = 1, \dots, N\right\}$$

where R is a natural number, $M = |\mathcal{M}|$ and $N = M^R$. The quality diversity decomposition of L take the form

$$L_{ij} = q(y_i)\phi(y_i)^T\phi(y_j)q(y_j)$$

and since $N = M^R$ is typically very big, it is impractical to define or store the quality and diversity features for every item $y_i \in \mathcal{Y}$. To deal with this problem we will assume that they admit factorisations and are thus a combination of only a few qualities and diversities.

More precisely we call $F \subseteq 2^{\{1,\dots,R\}}$ a *set of factorisations* and for a *factor* $\alpha \in F$, y_α denotes the subtupel of $y \in \mathcal{Y}$ that is indexed by α. Further we will work with the decompositions

$$\begin{aligned} q(y) &= \prod_{\alpha \in F} q_\alpha(y_\alpha) \\ \phi(y) &= \sum_{\alpha \in F} \phi_\alpha(y_\alpha) \end{aligned} \tag{2.6}$$

for a suitable set of factorisations F and qualities and diversities q_α and ϕ_α for $\alpha \in F$. Note that so far this is neither a restriction of generality – we could simply choose $F = \big\{\{1, \dots, R\}\big\}$ – nor a simplification – in that case we have the exact same number of qualities and diversities. However we are interested in the case where F consists only of small subsets of $\{1, \dots, R\}$. For example suppose that F is the set of all subsets with one or two elements, then we only have

$$R \cdot M + \binom{R}{2} \cdot M^2 = O(R^2M^2)$$

quality and diversity features instead of

$$M^R = O(M^R).$$

This reduction of variables will make modelling, storing and estimating them feasible again in a lot of cases where naive approaches are foredoomed because of their shear size.

II.4 The magic properties of DPPs

II.5 The mode problem

II.6 Calculations

(*i*) Complement of DPPs

(*ii*) Explain why every suitably definite matrix is a marginal kernel

(*iii*) Expression of elementary probabilities

Chapter III

Learning setups

III.1 What does learning mean and why is it interesting?

III.2 Kernel reconstruction using principle minors

In this section we want to see how we can estimate the marginal kernel from an increasing number of observations $\mathbf{Y}_1, \ldots, \mathbf{Y}_n \subseteq \mathcal{Y}$ that are distributed according to $\mathbb{P}$. For this we will sketch the procedure in [Urschel et al., 2017]. Let $\hat{\mathbb{P}}_n$ be the *empirical measure*

$$\hat{\mathbb{P}}_n := \frac{1}{n} \sum_{i=1}^{n} \delta_{\mathbf{Y}_i}.$$

The interest in those lies in the fact that they quite natural estimates for the actual underlying distribution. More precisely they are *unbiased estimators* for $\mathbb{P}$, i.e. they agree in expectation with $\mathbb{P}$. This can be seen by evaluating it at $A \subseteq \mathcal{Y}$

$$\mathbb{E}_{\mathbb{P}}\big[\hat{\mathbb{P}}_n(A)\big] = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\mathbb{P}}\big[\delta_{\mathbf{Y}_i}(A)\big] = \mathbb{P}(A).$$

cite, explain in more detail

explain consistency

And even stronger by the strong law of large numbers they converge to $\mathbb{P}$ almost surely if the sequence $(Y_k)_{k \in \mathbb{N}}$ of observations is independent. Therefore we can consistently estimate all principle minors of K, since

$$\hat{\mathbb{P}}_n(A \subseteq \mathbf{Y}) \xrightarrow{n \to \infty} \mathbb{P}(A \subseteq \mathbf{Y}) = \det(K_A) \quad \text{almost surely.}$$

Thus the question naturally arrises whether we can reconstruct the kernel K from the knowledge of all of its principle minors, which we will address in the next section.

Principle minor assignment problem

This is known as the *principle minor assignment problem* and has been studied extensively (cf. [Griffin and Tsatsomeros, 2006] and [Urschel et al., 2017]) and an computationally efficient algorithm has been proposed for the problem in [Rising et al., 2015]. It is in fact possible to retain the matrix from its principle minors up to an equivalence relation which identifies matrices with each other, that have the same principle minors. Obviously this is sufficent for the task of learning a DPP, because those matrices are exactly those who give rise to the same point process. To see roughly how this reconstruction works we note that the diagonal is given by

$$K_{ii} = \det(K_{\{i\}})$$

and the absolute value of the off diagonal can be obtained through

$$K_{ij}^2 = K_{ii}K_{ii} - \det\left(K_{\{i,j\}}\right).$$

The reconstruction of the signs of the entries K_{ij} turns out to be the main difficulty, but this can be done analysing the cycles of the adjacency graph G_K corresponding to K. The adjacency graph has $\mathcal{Y}$ as its vertex set and the set of edges consists of the pairs $\{i, j\}$ such that $K_{ij} \neq 0$. The reconstruction now relies on the analysis of the cycles of this graph and it has been shown, that one only needs to know all the principle minors up to the order of the cycle sparsity of G_K (cf. [Urschel et al., 2017]). Following this method it is possible to compute estimators $\hat{K}_n$ of K in polynomial time and give a bound on the speed of convergence in some suitable metric.

see whether this proof can be done in a simplified way without considering the sparsity l

state and explain the result

is this estimator unbiased? well $\hat{\mathbb{P}}_n$ is unbiased

III.3 Maximum likelihood estimation using optimisation techniques

The method of maximum likelihood estimation or short MLE is a very well established procedure to estimate parameters. The philosophy of MLE is that one selects the parameter under which the given data would be the most likely to be observed and to motivate this in more detail we roughly follow the corresponding section in [Rice, 2006].

For example if we consider random variables $X_1, \dots, X_n$ with a joint density $f(x_1, \dots, x_n, \theta)$ and we want to estimate the parameter θ based on a sample $x_1, \dots, x_n$ of our random variables. Then we would want to select the parameter θ that maximises the density $f(x_1, \dots, x_n, \theta)$. If additionally the random variables are indepent and identically distributed, their joint density factorises and thus we obtain

$$f(x_1, \dots, x_n, \theta) = \prod_{i=1}^{n} f(x_i, \theta)$$

where $f(x, \theta)$ is the density of the X_i. In practice it is often easier to maximise the logarithm of the density

$$\mathcal{L}(\theta) = \log\bigl(f(x_1, \dots, x_n, \theta)\bigr) = \sum_{i=1}^{n} \log\bigl(f(x_i, \theta)\bigr)$$

since this transforms the product over functions into a sum. However this is clearly equivalent to maximising the density since the logarithm is strictly monotone. We call the function $\mathcal{L}$ the *log likelihood function* and we denote its domain which is just the set of all parameters we wish to consider by Θ.

Kernel estimation

Assume again that we have a set of observations $Y_1, \dots, Y_n \subseteq \mathcal{Y}$ drawn independently and according to the DPP $\mathbb{P}$. Now we want to find the maximum likelihood estimator in the set $\mathbb{R}^{N\times N}_{\text{sym},+}$ of all symmetric and positive semidefinite $N \times N$ matrices. The log likelihood function is now given by

rewrite it in a MLE rather than ML fashion

$$\mathcal{L} \colon \mathbb{R}^{N\times N}_{\text{sym},+} \to [-\infty, 0] \qquad L \mapsto \log\left(\prod_{i=1}^{n} \mathbb{P}_L(Y_i)\right).$$

Using (2.4) we get the expression

$$\mathcal{L}(L) = \sum_{i=1}^{n} \log\left(\det(L_{Y_i})\right) - n\log\left(\det(L+I)\right)$$

We note that $\mathcal{L}$ is smooth and the gradient of this can be explicitly expressed, at least on the domain $\{\mathcal{L} > -\infty\}$. This is due to the fact that the determinants of the submatrices are polynomials in the entries of L and the composition of those with the smooth function $\log\colon (0,\infty) \to \mathbb{R}$ stays smooth. This property allows the use of gradient methods but they face the problem that the loss function is non concave and thus those algorithms will generally not converge to a global maximiser. To see that the lof li

Learning the quality

Let now $\{Y_t\}_{t=1,\dots,T}$ be a training set where $Y_t \subseteq \mathcal{Y}$ for every $t = 1,\dots,T$. Unlike earlier we will not try to estimate the whole kernel L but only the qualities q_i of the items $i \in \mathcal{Y}$. More precisely we can parametrise the positive definite symmetric matrices L using the quality diversity decomposition, i.e. we consider the bijection

whats the domain?

$$(q,S) \mapsto L \quad \text{where } L_{ij} = q_i S_{ij} q_j.$$

Now we fix a similarity kernel S_0, that usually comes from modelling, and only try to estimate the quality $q \in \mathbb{R}_+^N$. This means that we optimise the likelihood function over a smaller set of kernels, namely the ones that arrise from (q, S_0) for $q \in \mathbb{R}_+^N$ and thus the maximal likelihood that can be achieved will be lower compared to the general kernel estimation.

$$q_i(X) = g(f_i(X)), \quad \phi_i(X) = G(f_i(X))$$

where $f_i(X) \in \mathcal{Z}$ is being modelled and $g\colon \mathcal{Z} \to [0,\infty)$ and $G\colon \mathcal{Z} \to \mathbb{R}^D$ will be learned based on the observations. We will assume that $\mathcal{Z}$ is a subset of a vector space and therefore we call $f_i(X)$ the *feature vector*. If it is possible to estimate the quality and diversity as above, we would be able to sample from every DPP $\mathbb{P}(\cdot \mid X)$ and even from those that we haven't observed so far – just by the knowledge about DPPs with a similar structure.

Let us again illustrate this procedure in the example of the human point selection and we will restrict ourselves to learn the function g that determines the quality function, we might have a reason to be absolutely sure that we have modelled the diversity features $\phi_i(X)$ perfectly, so there is no need to learn, i.e. optimise them any further. However we are not convinced any more that humans really do not prefer some points over others – maybe we have the feeling that they lean more towards the points located in the center of the square. Therefore it is natural to assume that the quality, which is nothing but the popularity of a point, depends on the distance to the centre point of the square $m = (1/2, 1/2)$, i.e.

$$q_i(n) = g(\|i - m\|) = g(f_i(n))$$

where we want to learn g with respect to some loss function over a given family $\mathcal{F}$ of functions.

To put this back into the general setting we note that $g \in \mathcal{F}$ gives rise to a different conditional DPP which we will denote by $\mathbb{P}_g(\cdot \mid X)$. Just like in the case of simple DPPs we will work with the negative of the log likelihood function

$$\mathcal{L}(g) := -\log\left(\prod_{t=1}^{T} \mathbb{P}_g(Y_t \mid X_t)\right)$$

and seek a minimiser of the loss function $\mathcal{L}$. Thus we obtain an optimisation problem over a family of functions and in practice it is convenient to restrict ourselves to a parametric family

$$\mathcal{F} = \left\{ g_\theta \mid \theta \in U \subseteq \mathbb{R}^M \right\}.$$

In this case we write $\mathbb{P}_\theta(\cdot \mid X)$ for the conditional DPP that is induced by g_θ and the kernels become functions of θ and thus we write $L(\theta; X)$ and $K(\theta; X)$ for the kernel associated with the parameter θ. In analogue fashion we denote the loss function by

$$\mathcal{L}(g_\theta) = \mathcal{L}(\theta) = -\sum_{t=1}^{T} \log\left(\mathbb{P}_\theta(Y_t \mid X_t)\right).$$

Properties of the loss function $\mathcal{L}$

We want to see how the log likelihood approach naturally leads to a log linear model in θ for the quality features if one wants to obtain a convex loss function. Of course the motivation for a convex loss function is given by the nice properties of convex optimisation tasks described earlier. In order to see in which cases the loss function is convex, we use (**??**) to obtain

$$\begin{aligned} -\log\left(\mathbb{P}_\theta(Y_t \mid X_t)\right) &= -\log(\det(L_Y(\theta; X))) + \log\Big(\det(L(\theta; X_t) + I)\Big) \\ &= -2 \cdot \sum_{i \in Y_t} \log\left(g_\theta(f_i(X_t))\right) - \log\Big(\det\left(S_{Y_t}(X_t)\right)\Big) \\ &\quad + \log\left(\sum_{A \subseteq \mathcal{Y}(X_t)} \left(\prod_{i \in A} g_\theta(f_i(X_t))^2\right) \det(S_A(X_t))\right). \end{aligned} \tag{3.1}$$

This expression is well defined in $[0, \infty]$ if we adapt the common convention $\det(S_\emptyset(X)) = 1$. In order to give some criteria for the convexity and coercivity of the loss function, we say that a function f *log concave*, *log convex* or *logarithmically (affine) linear* if $\log(f)$ has the respective property.

Proposition 3.1 (Coercivity and convexity of the loss function). (*i*) *The rate function is coercive for all possible training sets if and only if*

$$\mathbb{P}_\theta(Y \mid X) \xrightarrow{|\theta| \to \infty} 0 \quad \textit{for all } Y \subseteq \mathcal{Y}(X) \textit{ and } X \in \mathcal{X}. \tag{3.2}$$

(*ii*) *The rate function is convex for all possible training sets if $g_\theta(f_i(X_t))$ is log concave in θ for all $i \in \mathcal{Y}(X)$, $X \in \mathcal{X}$ and if*

$$\prod_{i \in B} g_\theta^2(f_i(X_t))$$

is log convex in θ for all $B \subseteq \mathcal{Y}(X))$ and $X \in \mathcal{X}$.

(*iii*) *The conditions in* (*ii*) *are satisfied if and only if $g_\theta(f_i(X))$ is logarithmically affine linear in θ for every $i \in \mathcal{Y}(X)$ and $X \in \mathcal{X}$.*

Proof. (*i*) It is clear that under (3.2) we have

$$\exp\left(-\mathcal{L}(\theta)\right) = \prod_{t=1}^{T} \mathbb{P}_\theta(Y_t \mid X_t) \xrightarrow{|\theta|\to\infty} 0$$

for every possible training set and thus $\mathcal{L}$ is coercive. If on the other hand $\mathcal{L}$ is coercive for every training set we could also choose (Y, X) arbitrary as our training set and immediately obtain (3.2).

(*ii*) This condition for the convexity of the loss function can be directly derived from the fact that linear combination of log convex functions are log convex and formula (3.1).

(*iii*) If $g_\theta(f_i(X))$ is logarithmically affine linear, then it is also log convex and

$$\log\left(\prod_{i\in B} g_\theta^2\left(f_i(X)\right)\right) = 2\sum_{i\in B}\log\left(g_\theta(f_i(X))\right)$$

is convex. On the other side if (*ii*) holds, then all functions $\log\left(g_\theta(f_i(X))\right)$ are concave and $\sum\limits_{i\in B}\log\left(g_\theta(f_i(X))\right)$ is convex and thus $\log\left(g_\theta(f_i(X))\right)$ has to be affine linear.

□

The result above shows that logarithmically affine linear models are the natural fit for the parametric family $\mathcal{F}$ that we want to optimise over. However they can be easily transformed into log linear models through a simple parameter shift if we assume $f_i(X) \neq 0$ and thus we can assume without loss of generality that the functions g_θ have the form

$$g_\theta(f_i(X)) = \exp\left(\frac{1}{2}\theta^T f_i(X)\right) \quad \text{for all } i \in \mathcal{Y}(X) \text{ and } X \in \mathcal{X}.$$

This structure can be used to derive some explicit expression for this case. Of course this log linear model is only well defined if the feature space $\mathcal{Z}$ is a subset of $\mathbb{R}^M$ which we will assume from now on. We note that this is no restriction if we assume a log linear model, because otherwise we could just replace the feature functions f_i by the log linearity constants $\hat{f}_i(X) \in \mathbb{R}^M$. First we can apply the explicit structure to the elementary probabilities and get

$$\mathbb{P}_\theta(A \mid X) \propto \exp\left(\theta^T f_A(X)\right)\det(S_A(X))$$

where $f_A(X) := \sum_{i\in A} f_i(X)$. Using this we get that the single summands of the loss function are equal to

$$-\theta^T f_Y(X) - \det(S_Y(X)) + \log\left(\sum_{A\subseteq\mathcal{Y}(X)} \exp\left(\theta^T f_A(X)\right)\det(S_A(X))\right) \tag{3.3}$$

Since a lot of numerical optimisation algorithms depend on the gradient of the function, it is worth noting that an explicit expression for the gradient of the loss function $\mathcal{L}$ can be derived from this formula, since differentiating (3.3) with respect to θ gives

$$\begin{aligned}
-f_Y(X) + \frac{\sum_{A\subseteq\mathcal{Y}(X)} f_A(X) L_A(\theta; X)}{\sum_{A\subseteq\mathcal{Y}(X)} L_A(\theta; X)} &= -f_Y(X) + \sum_{A\subseteq\mathcal{Y}(X)} f_A(X)\mathbb{P}_\theta(A \mid X) \\
&= -f_Y(X) + \sum_{i\in\mathcal{Y}(X)} f_i(X) \sum_{i\in A\subseteq\mathcal{Y}(X)} \mathbb{P}_\theta(A \mid X) \\
&= -f_Y(X) + \sum_{i\in\mathcal{Y}(X)} f_i(X)\mathbb{P}_\theta(i \in \mathbf{Y} \mid X) \\
&= -f_Y(X) + \sum_{i\in\mathcal{Y}(X)} f_i(X) K_{ii}(\theta; X).
\end{aligned} \tag{3.4}$$

The later expression of this gradient has the advantage that it can be efficiently computed in contrary to the evaluation of the exponentially large sum in the first line.

Obviously the loss function is not coercive in general, since for $f_i(X) = 0$ the probability $\mathbb{P}_\theta(\{i\} \mid X)$ is constant in θ. However it is not straight forward whether it becomes coercive under the assumption $f_i(X) > 0$ entrywise for every $i \in \mathcal{Y}(X)$ and $X \in \mathcal{X}$ and this could be investigated further.

Estimating the mixture coefficients of k-DPPs

Learning kernels of conditional DPPs

III.4 A Bayesian approach to the kernel estimation

Chapter IV

Toy examples and experiments

IV.1 Minimal example?

IV.2 Points on the line

IV.3 Points in the square

IV.4 Toy example for quality learning

Chapter V

Summary and conclusion

Chapter A

Generated code

All my coding was done in R and I will provide the code for sampling, my examples and also the learning algorithm of my toy example here. During my coding I mostly followesd Google's R Style Guide (https://google.github.io/styleguide/Rguide.xml).

A.1 Sampling algorithm

```
# Implementation of the sampling algorithm as a function

SamplingDPP <- function (lambda, eigenvectors) {
  # First part of the algorithm, doing the selection of the eigenvectors
  N = length(lambda)
  J <- runif(N) <= lambda/(1 + lambda)
  k <- sum(J)
  V <- matrix(eigenvectors[, J], nrow=N)
  Y <- rep(0, k)

  # Second part of the algorithm, the big while loop
  while (k > 0) {
    # Calculating the weights and selecting an item i according to them
    wghts <- k^(-1) * rowSums(V^2)
    i <- sample(N, 1, prob=wghts)
    Y[k] <- i
    if (k == 1) break

    # Projecting e_i onto the span of V
    help <- V %*% V[i, ]
    help <- sum(help^2)^(-1/2) * help

    # Projecting the elements of V onto the subspace orthogonal to help
    V <- V - help %*% t(t(V) %*% help)

    # Orthonormalize V and set near zero entries to zero
    V[abs(V) < 10^(-9)] <- 0
    j <- 1
    while(j <= k) {
      help2 <- rep(0, N)
     m <- 1
        while (m <= j - 1) {
        help2 <- help2 + sum(V[, j] * V[, m]) * V[, m]
```

```
        m <- m + 1
      }
      V[, j] <- V[, j] - help2
      if (sum(V[, j]^2) > 0) {
        V[, j] <- sum(V[, j]^2)^(-1/2) * V[, j]
      }
      j <- j + 1
    }
    V[abs(V) < 10^(-9)] <- 0

    # Selecting a linear independent set in V
    k <- k - 1
    q <- qr(V)
    V <- matrix(V[, q$pivot[seq(k)]], ncol=k)
  }
  return(Y)
}
```

A.2 Points on the line

```
# NEEDS: sampling algorithm

# In this example we sample points on a (discrete) line according to a DPP
# We model L directly and via the quality-diversity decomposition. We plot and
# compare the patterns to uncorrelated points i.e. to a Poisson point process.

# Minimal example ____________________________________________________________
n <- 3
L <- matrix(c(2,1,0,1,2,0,0,0,2), nrow=n)

# Points on a line ___________________________________________________________
n <- 100
L <- rep(0, n^2)
for (i in 1:n) {
  for (j in 1:n) {
    L[(i - 1) * n + j] <- dnorm((i-j) * n^(-1/4))
  }
}
L <- matrix(L, nrow=n)

# Modelling phi and q ________________________________________________________
# Points on the line.
m <- 99  # 29
n <- m + 1
q <- rep(10, n)  # 0-1 sequences: rep(10^2, n)
phi <- rep(0, n^2)
for (i in 1:n) {
  for (j in 1:n) {
    phi[(i - 1) * n + j] <- dnorm((i - j) / 10)  # 0-1 sequences: devide by 2
  }
}
phi <- matrix(phi, ncol=n)

# Log linear quality for the points on the line ______________________________
m <- 99
n <- m + 1
```

```
q <- rep(0, n)
for (i in 1:n) {
  q[i] <- 10^2 * sqrt(m) * exp(-0.2 * abs(i - 50.5))
}
phi <- rep(0, n^2)
for (i in 1:n) {
  for (j in 1:n) {
    phi[(i - 1) * n + j] <- dnorm(2 * (i - j) / sqrt(m))
  }
}
phi <- matrix(phi, ncol=n)

# General part, define L ________________________________________________________
for (i in 1:n) {
  phi[, i] <- sum(phi[, i]^2)^(-1/2) * phi[, i]
}
S <- t(phi) %*% phi
time <- proc.time()
L <- t(q * S) * q
proc.time() - time

# Compute the eigendecomposition, set near zero eigenvalues to zero and
# set up poisson point process with same expected cardinality _________________
time <- proc.time()
edc <- eigen(L)
lambda <- edc$values
lambda[lambda < 10^(-9)] <- 0
mean <- sum(lambda / (1 + lambda))
eigenvectors <- edc$vectors
lambda2 <- rep(mean / n / (1 - mean / n), n)
eigenvectors2 <- diag(rep(1, n))
proc.time() - time

# Sample and plot things ________________________________________________________
# Minimal example

# 0-1 sequences
x <- sort(SamplingDPP(lambda, eigenvectors))
as.integer(1:n %in% x)
y <- sort(SamplingDPP(lambda2, eigenvectors2))
as.integer(1:n %in% y)

# Sample from both point processes and plot the points on the line
pointsDPP <- SamplingDPP(lambda, eigenvectors)
pointsPoisson <- SamplingDPP(lambda2, eigenvectors2)
plot(rep(1, length(pointsDPP)), pointsDPP,
     ylim=c(1, n), xlim=c(.4, 3.2), xaxt='n', ylab="Points", xlab="")
points(rep(2, length(pointsPoisson)), pointsPoisson, pch=5)
legend("topright", inset=.05, legend=c("DPP", "Poisson"), pch=c(1, 5))

# Remove all objects apart from functions
rm(list = setdiff(ls(), lsf.str()))
```

A.3 Points in the square

```
# NEEDS: sampling algorithm
```

```r
# In this example we sample points on a two dimensional grid according to a DPP
# We model L directly and via the quality-diversity decomposition including
# different dimensions D for the feature vectors phi. We plot and compare the
# patterns to uncorrelated points i.e. to a Poisson point process.

# Define the coordinates of a point ___________________________________________
CoordinatesNew <- function(i, n) {
  y1 <- floor((i - 1) / (n + 1))
  x1 <- i - 1 - (n + 1) * y1
  return (t(matrix(c(x1, y1)/n, nrow=length(i))))
}
DistanceNew <- function (i, j, n, d) {
  return (sqrt(colSums((CoordinatesNew(i, n) - CoordinatesNew(j, d))^2)))
}

# Direct modelling of L _______________________________________________________
m <- 19
n <- (m + 1)^2
L <- rep(0, n^2)
for (i in 1:n) {
  for (j in 1:n) {
    L[(i - 1) * n + j] = n^2 * dnorm(Distance(i, j, m))
  }
}
L <- matrix(L, nrow=n)

# Modelling phi and q _________________________________________________________
# Points in the square.
m <- 19
n <- (m + 1)^2
q <- rep(sqrt(m), n)
x <- ceiling(1:n^2 / n)
y <- rep(1:n, n)
time <- proc.time()
phi <- dnorm(sqrt(m) *matrix(DistanceNew(x, y, m, m), n))
proc.time() - time

# Quality diversity decomposition with small D ________________________________
d <- 25
q <- rep(10^5 * sqrt(m), n)
x <- ceiling(1:(n*d) / d)
y <- rep(1:d, n)
time <- proc.time()
phi <- dnorm(2 * sqrt(m) * matrix(DistanceNew(x, y, m, sqrt(d) - 1), ncol=n))
proc.time() - time

# Log linear quality for the points in the square ____________________________
m <- 39
n <- (m + 1)^2
q <- exp(-6 * DistanceNew(rep(5, n), 1:n, 2, m) + log(sqrt(m)))
x <- ceiling(1:n^2 / n)
y <- rep(1:n, n)
time <- proc.time()
phi <- dnorm(2 * sqrt(m) * matrix(DistanceNew(x, y, m, m), n))
proc.time() - time
```

```
# General part, defining L ____________________________________________________
# d <- length(phi) / n
for (i in 1:n) {
  phi[, i] <- sum(phi[, i]^2)^(-1/2) * phi[, i]
}
S <- t(phi) %*% phi
# B <- t(phi) * q
time <- proc.time()
L <- t(t(q * S) * q)  # B %*% t(B)
proc.time() - time

# Compute the eigendecomposition, set near zero eigenvalues to zero and
# set up poisson point process with same expected cardinality _________________
time <- proc.time()
edc <- eigen(L)
lambda <- edc$values
lambda[abs(lambda) < 10^(-9)] <- 0
mean <- sum(lambda / (1 + lambda))
eigenvectors <- edc$vectors
lambda2 <- rep(mean / n / (1 - mean / n), n)
eigenvectors2 <- diag(rep(1, n))
proc.time() - time

# Sample from both point processes and plot the points in the square __________
# par(mfrow = c(1,1))
time <- proc.time()
dataDPP <- sort(SamplingDPP(lambda, eigenvectors))
pointsDPP <- t(CoordinatesNew(dataDPP, m))
plot(pointsDPP, xlim=0:1, ylim=0:1, xlab="", ylab="", xaxt='n', yaxt='n', asp=1)
proc.time() - time
dataPoisson <- sort(SamplingDPP(lambda2, eigenvectors2))
pointsPoisson <- t(CoordinatesNew(dataPoisson, m))
plot(pointsPoisson, xlim=0:1, ylim=0:1, xlab="", ylab="",
                                        xaxt='n', yaxt='n', asp=1)

# Remove all objects apart from functions
rm(list = setdiff(ls(), lsf.str()))
```

A.4 Toy learning example

```
# NEEDS: Sampling algorithm, declaration of the points in the square
# TODO: Maybe do the gradient descent directly over the representation
# od the gradient

# With this toy example we aim to perform the first learning of paramters
# associated to a kernel of a DPP. More precisely we will generate our own
# data of points on a two dimensional grid with a log linear quality model
# and aim to estimate the log linearity parameter.

# Generation of data
time <- proc.time()
T <- 30
data <- rep(list(0), T)
for (i in 1:T) {
  data[[i]] <- sort(SamplingDPP(lambda, eigenvectors))
}
proc.time() - time
```

```
# Define the quality q, L, the feature sum and the loss in dependency of the
# parameter theta
Quality <- function(theta) {
  return(exp(theta[1] * DistanceNew(rep(5, n), 1:n, 2, m) + theta[2]))
}
LFunction <- function(theta) {
  return(t(t(Quality(theta) * S) * Quality(theta)))
}
Feature <- function(A) {
  # return(sum(DistanceNew(rep(5, length(A)), A, 2, m)))
  return(c(sum(DistanceNew(rep(5, length(A)), A, 2, m)), length(A)))
}
Loss <- function(theta) {
  T <- length(data)
  # Sum this over all data entries
  x <- 0
  for (i in 1:T) {
    A <- data[[i]]
    x <- x + 2 * sum(theta * Feature(A)) + log(det(matrix(S[A, A], length(A))))
  }
  return(- x + T * log(det(diag(rep(1, n)) + LFunction(theta))))
}

# Parameter estimations
time <- proc.time()
sol <- nlm(Loss, c(-3, 0))
proc.time() - time
sol$estimate

# Remove all objects apart from functions
rm(list = setdiff(ls(), lsf.str()))
```

Bibliography

[Affandi et al., 2014] Affandi, R. H., Fox, E., Adams, R., and Taskar, B. (2014). Learning the parameters of determinantal point process kernels. In *International Conference on Machine Learning*, pages 1224–1232.

[Benard and Macchi, 1973] Benard, C. and Macchi, O. (1973). Detection and“emission”processes of quantum particles in a“chaotic state”. *Journal of Mathematical Physics*, 14(2):155–167.

[Borodin, 2009] Borodin, A. (2009). Determinantal point processes. *arXiv preprint arXiv:0911.1153*.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[Griffin and Tsatsomeros, 2006] Griffin, K. and Tsatsomeros, M. J. (2006). Principal minors, part ii: The principal minor assignment problem. *Linear Algebra and its applications*, 419(1):125–171.

[Higham, 1990] Higham, N. J. (1990). Exploiting fast matrix multiplication within the level 3 blas. *ACM Transactions on Mathematical Software (TOMS)*, 16(4):352–368.

[Kulesza et al., 2012] Kulesza, A., Taskar, B., et al. (2012). Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286.

[Magen and Zouzias, 2008] Magen, A. and Zouzias, A. (2008). Near optimal dimensionality reductions that preserve volumes. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 523–534. Springer.

[Rice, 2006] Rice, J. (2006). *Mathematical statistics and data analysis*. Nelson Education.

[Rising et al., 2015] Rising, J., Kulesza, A., and Taskar, B. (2015). An efficient algorithm for the symmetric principal minor assignment problem. *Linear Algebra and its Applications*, 473:126–144.

[Samuel, 1959] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229.

[Urschel et al., 2017] Urschel, J., Brunel, V.-E., Moitra, A., and Rigollet, P. (2017). Learning determinantal point processes with moments and cycles. *arXiv preprint arXiv:1703.00539*.