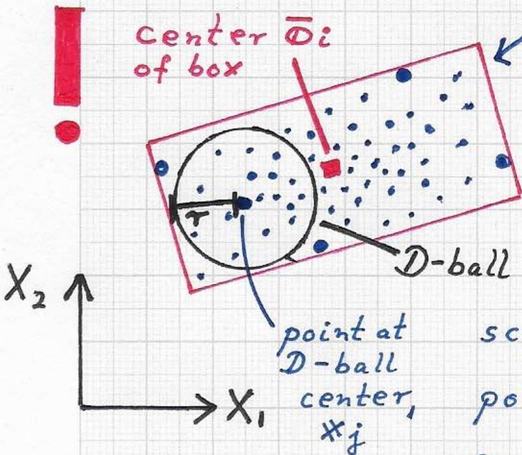
Stratovan

OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

• Laplacian eigenfunctions and neural networks:... **Local probability density function.**

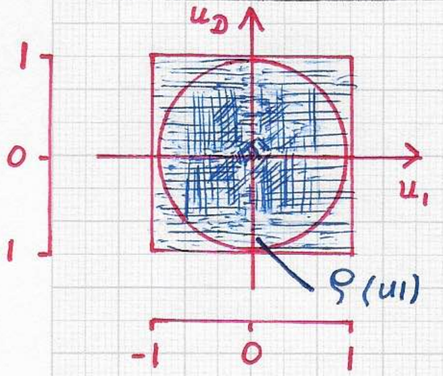


center $\bar{\Phi}_i$ of box
r
D-ball
X_2
X_1
point at D-ball center, $\mathbf{x}_j$

One must construct a "good" probability function for each point subset bounded by a minimal-size box, oriented based on the (covariance) eigendirections as described. First, one must estimate a point density value $\varrho_j = \varrho_i(\mathbf{x}_j)$ for each point $\mathbf{x}_j$ in the box. The left figure illustrates the concept of using a D-ball for this purpose. This D-ball has $\mathbf{x}_j$ as its center and has radius r. The total number of points inside the D-ball divided by the D-ball's volume can serve as the discrete density estimate ϱ_j. Having estimated density estimates for all points $\mathbf{x}_j$ ('•') inside the box, one can use all or some of these points $\mathbf{x}_j$ with densities ϱ_j to construct a probability density function $\varrho_i(\mathbf{x})$ for this box, i.e., box i. Second, one computes the parameter values of a "properly chosen" density model function. Ideally, the model function has only a small number of parameters; is efficiently and stably computable; and has a low approximation error. Considering the fact that $\varrho_i(\mathbf{x})$ describes the overall density function $\varrho(\mathbf{x}) = \sum_i w_i(\mathbf{x})\,\varrho_i(\mathbf{x})$ only locally, in box i, reminds us that the point density in box i cannot be expected to satisfy the characteristics of a "standard model." •••

■ OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

• Laplacian eigenfunctions and neural networks:...



The left figure sketches the local probability density function ρ that one must construct for a point subset inside its local minimal bounding box. After mapping all subset points to the corresponding $\mathbb{u}$-box, i.e., $[-1,1]^D$, one can perform the approximation calculations defining $\rho = \rho(\mathbb{u}) = \rho_i(\mathbb{u})$ for the i^{th} point subset. The hyper-cubes $[-1,1]^D$ defining the relevant $\mathbb{u}$-domains have volumes $2^1, 2^2, 2^3, 2^4$ etc. for dimensionality values $D = 1, 2, 3, 4$ etc. Inside a hyper-cube, we know estimated probability density values $\rho_j = \rho_j(\mathbb{u}_j)$. We must determine a "best approximation" $\rho(\mathbb{u})$ that minimizes the sum of the squared difference (error) values between the given density values (value **estimates**) and the values of the approximation $\rho(\mathbb{u})$ at the same $\mathbb{u}$-locations. The model function used to calcutate the needed approximation $\rho(\mathbb{u})$ must involve a sufficiently large number of parameters (degree of freedom) to approximate the given values ρ_j well; at the same time, it is also necessary – considering computational and data storage complexity – that the number of parameters of $\rho(\mathbb{u})$ does "not increase **too** quickly" with increasing value of D.

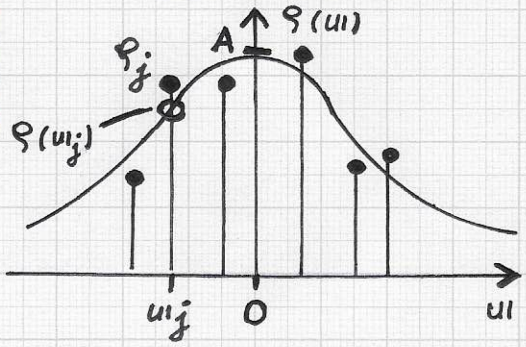
...

Bernd Hamann 5/22/2023

Stratovan

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks:...



The left figure is a rather abstract sketch of the approximating function ρ we must determine for the set $\{(\mathbb{u}_j, \rho_j)\}$. The figure shows the difference between a **given** probability density value (estimate) ρ_j and the approximated value $\rho(\mathbb{u}_j)$ = 'o' that we must minimize, in an integrated, summed-up manner for the set of all ρ_j-values. Using an exponentially decreasing model function is one viable – but not necessarily the best – choice to model the given data. For example, when considering all possible quadratic terms in the (negative) exponent of an exponential model function in the D-variate case, one obtains:

$D=1$: $\rho(u_1) = A\exp(-B_2 u_1^2)$

$D=2$: $\rho(u_1,u_2) = A\exp(-(B_{2,0}u_1^2 + B_{1,1}u_1u_2 + B_{0,2}u_2^2))$

$D=3$: $\rho(u_1,u_2,u_3) = A\exp(-(B_{2,0,0}u_1^2 + B_{0,2,0}u_2^2 + B_{0,0,2}u_3^2 + B_{1,1,0}u_1u_2 + B_{1,0,1}u_1u_3 + B_{0,1,1}u_2u_3))$

$$\underline{\mathbf{D}:} \qquad \underline{\rho(u_1,u_2,\dots,u_D)} = A\exp - \sum_{\substack{i_1+\dots+i_D=D \\ \wedge\, i_1\geq 0 \wedge \dots \wedge i_D \geq 0}} B_{i_1,\dots,i_D}\, u_1^{i_1}\cdots u_D^{i_D}$$

$$\hat{=}\ \underline{\underline{\rho(\mathbb{u}) = A\exp - \sum_{\|\mathbb{i}\|=D} B_{\mathbb{i}}\, \mathbb{u}^{\mathbb{i}}}} \quad (\text{"multi-index notation"})$$

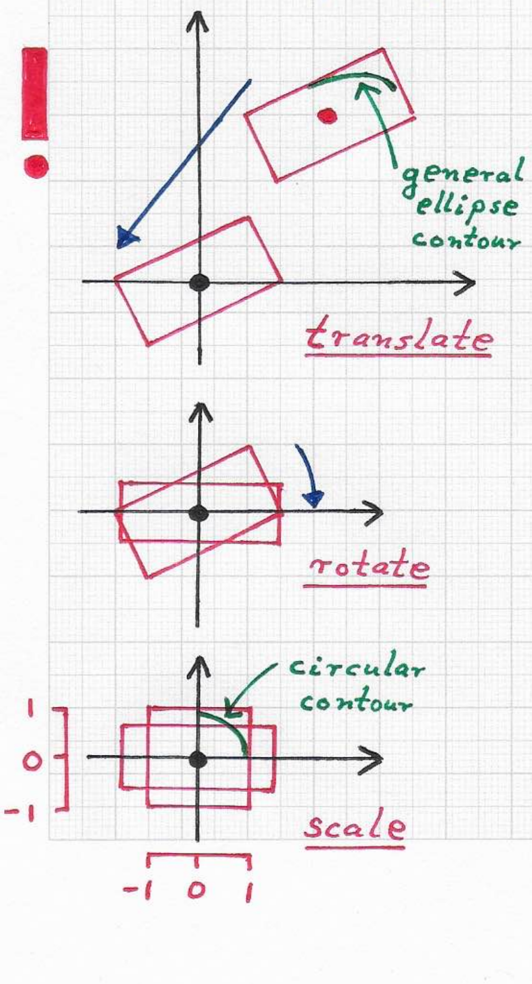
We must understand the computational complexity of $\rho(\mathbb{u})$, for computing and storing its coefficients. •••

OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks: ... The computational complexity of the calculation and storage of $\varrho(\omega)$ is defined by the number of coefficients in $A\exp(-\sum_i B_i \omega^i)$. This number is $1+1=2$ for $D=1$; $1+3=4$ for $D=2$; $1+6=7$ for $D=3$; $1+10=11$ for $D=4$ etc. For the general D-dimensional case, this number is

$$1+\binom{D+1}{2} = 1+\frac{(D+1)!}{2!\,(D-1)!} = 1+\tfrac{1}{2}D(D+1) = 1+\frac{D^2+D}{2}.$$

Thus, computational complexity scales quadratically in terms of D. As a consequence, the D-value must be small, i.e., $D \ll 10$. The image sequence shown in the left figure reminds us that the given point (sub-) set for which we must compute a least-squares-based probability function $\varrho(\omega)$ has undergone the sketched sequence/concatenation of linear transformations (translation, rotation and scaling), mapping the points involved to the "normalized" hyper-cube domain $[-1,1]^D$, based on the covariance analysis of the points. The details of this concatenation are provided on pp. 23–24 (4/14-15/2023).



• • •

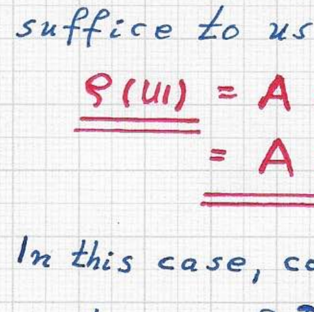
OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks:... The fact that a point (sub-) set has already been mapped via the linear transformations that consider point distribution characteristics should – generally – allow us to greatly reduce the computational complexity of the probability density function ϱ. **(i)** If a point (sub-) set had undergone only the translation and rotation steps, it should, generally, no longer be necessary to use "mixed" quadratic terms. In this case, it should suffice to use a quadratic model function of the form

$$\varrho(\mathbb{u}) = A \exp\left(-(B_1 u_1^2 + B_2 u_2^2 + \ldots + B_D u_D^2)\right)$$
$$= A \exp -\Sigma_{i=1}^{D} u_i^2 .$$

In this case, computational complexity scales only linearly in terms of D. **(ii)** If a point (sub-) set had undergone the translation, rotation and scaling steps, it should, generally, suffice to use only one quadratic term in the exponent. In this case, merely the squared ridial distance from the origin $\mathbb{u} = \mathbb{0}$ would appear in the exponent, leading to

$$\varrho(\mathbb{u}) = A \exp\left(-B(u_1^2 + u_2^2 + \ldots + u_D^2)\right) = A \exp(-Br^2).$$



$\varrho(\mathbb{u}) =$
$A \exp(-B\|\vec{u}\|^2)$
$= A \exp(-Br^2)$

In this case, computational complexity is constant, and only the values of A and B are needed per point sub-set representing a class.

• **Note.** Only computation of least-squares errors provides data-specific insight.

•••