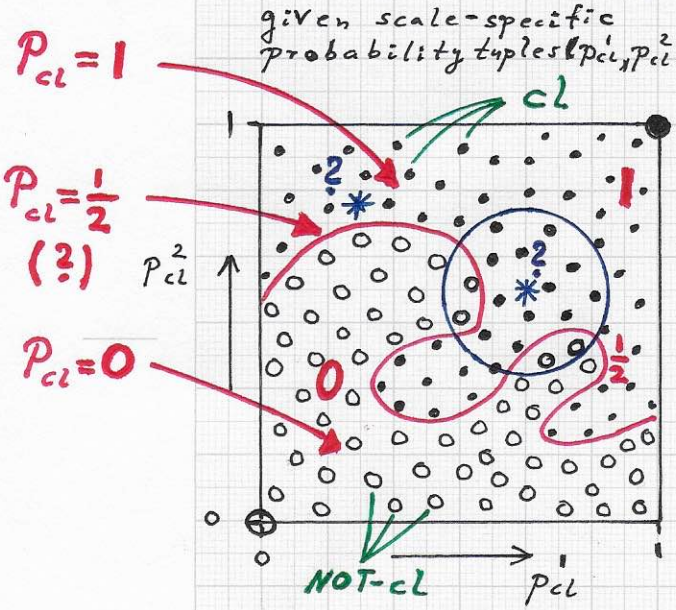
■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks:...

Simple scenario for the computation of a CLASS-specific probability value $P_{cl}(p'_{cl}, p^2_{cl})$, $P_{cl} \in [0,1]$, when class-labels "cl" and "NOT-cl" are attached to given scale-specific probability tuples (p'_{cl}, p^2_{cl}):



→ All samples '•' and 'o' have an associated scale-specific tuple (p'_{cl}, p^2_{cl}), p'_{cl} and $p^2_{cl} \in [0,1]$, and a DISCRETE, binary class label: 1 ⇒ sample belongs to class cl; 0 ⇒ sample does NOT belong to class cl.

→ The function/probability function $P_{cl}(p'_{cl}, p^2_{cl})$ must return a value that should be the "best possible value" one can estimate for any site * to make a classification decision: "* is/is NOT of class cl."

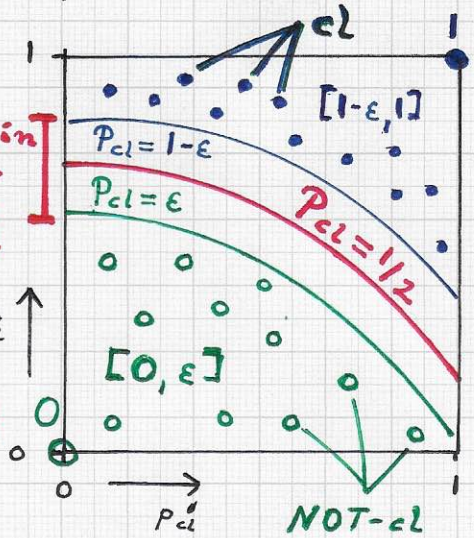
We must establish the relationship between the computation of a function value f at any location in a domain, using given function values f_i at certain sites x_i, and the estimation of a CLASS-specific probability value P_{cl}, using given scale-specific probability tuples $(p'_{cl}, p^2_{cl})_i$ with associated class labels '1' (tuple belongs to class cl) or '0' (tuple does not belong to class cl) – for example. This simple scenario is illustrated in the figure (left). Specifically, P_{cl} can be defined as a Shepard function that is used to calculate an overall, combined CLASS-specific P_{cl}-value for any tuple '*' (?) in the domain. As pointed out, one can use the exponents of the distance function of the Shepard formula to **INFLUENCE THE LOCATION OF THE "INTERFACE" $P_{cl}=\frac{1}{2}$ AND DEGREE OF "STEEPNESS" OF P_{cl} AT THE "INTERFACE" OF '0' AND '1'.**

...

■ OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

• Laplacian eigenfunctions and neural networks: ...

Characterizing the interface region in the domain of a CLASS-specific probability function P_{cl}:



uncertain transition region

make it small, use a "steep cliff"

A Shepard probability function P_{cl} subdivides the (p^1_{cl}, p^2_{cl}) domain in three regions based on its contours. In the illustrated scenario, a tuple (p^1_{cl}, p^2_{cl}) is a

- → cl-tuple $\Leftrightarrow P_{cl} \in [1-\varepsilon, 1]$;
- → NOT-cl-tuple $\Leftrightarrow P_{cl} \in [0, \varepsilon]$;
- → transition-region-tuple $\Leftrightarrow P_{cl} \in [\varepsilon, 1-\varepsilon]$.

One can view the contour for $P_{cl}=1/2$ as interface between "cl" and "NOT-cl" — and use the degrees of freedom/parameters of the Shepard function to place $P_{cl}=\frac{1}{2}$ optimally.

If the function P_{cl} was a "perfect probability function," behaving like a perfect binary classifier to identify "class-cl" or "NOT-class-cl" presence, it would only generate 1.0 or 0.0 as probability values, for any value of the tuple (p^1_{cl}, p^2_{cl}). Since we only have finite, discrete sample sets — sets of '•' (cl) and 'o' (NOT-cl) tuples (p^1_{cl}, p^2_{cl}) — we do not have knowledge of the exact boundary/interface location where cl and NOT-cl regions meet in the domain. In other words, we do not know the exact location of the interface where a perfect binary classifier must discontinuously change its value from 0 to 1 (or from 1 to 0). Therefore, a smooth (Shepard) function is used to calculate probability values in the interval [0, 1], and one assumes that the function's contour for the iso-value 1/2 is close to the unknown interface.

...

■ OBJECT AND MATERIALS EIGENFUNCTIONS – Cont'd.

- Laplacian eigenfunctions and neural networks:...
- **Note.** A CLASS-specific probability function P_{cl} can depend on different probabilities p_{cl}^{sc}, for different classes and scales. For example, P_{cl} can be defined via

$$P_{cl} = P_{cl}(p_{cl}^1, p_{cl}^2)$$

or

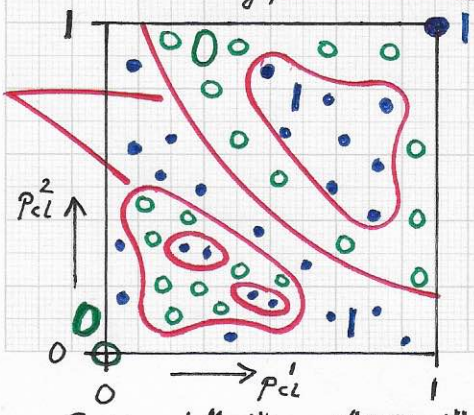
$$P_{cl} = P_{cl}(p_{cl}^1, p_{cl}^2, \dots, p_{cl}^H)$$

or

$$P_{cl} = P_{cl}(p_o^1, \dots, p_o^H, \dots p_c^1, \dots, p_c^H).$$

The dimension of P_{cl}'s domain and the selected specific variables p_{cl}^{sc} should be chosen in order to optimize classification performance and computational efficiency.

- Theoretically possible contour:



Several "cl" and "NOT-cl" regions induced by $P_{cl} = \frac{1}{2}$.

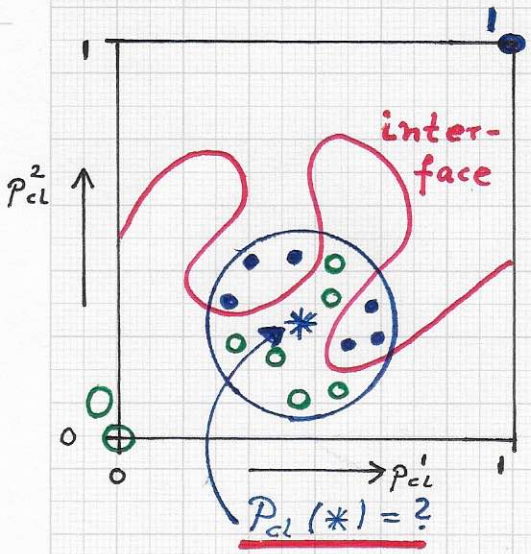
Of course, the value of P_{cl} should be precisely 1 (0) for the finite set of sites '•' ('o') with known, given classification label "cl" ("NOT-cl"). Concerning the optimization of a **Shepard probability** function – via the exponents of its distance functions – the goal is to make the transition region in the domain extremely small; this region with function values in the interval $[\varepsilon, 1-\varepsilon]$ should become very "narrow," effectively forcing the function's graph to exhibit a steep "1-0-cliff" and inducing a "quasi-discontinuity." In addition, the precise location of the contour 1/2 in the domain can and should be optimized. The figure (left) sketches a theoretically possible arrangement of "cl" ('•') and "NOT-cl" ('o') regions in the 2D domain of P_{cl}. Such a complex geometrical arrangement does not cause a problem, since it is only important that P_{cl} has the proper value.

...

■ OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

• Laplacian eigenfunctions and neural networks: ...

The domain of a P_{cl} function can maximally have $H \cdot (C+1)$ dimensions. It can be assumed that, generally, a much smaller number of the available p_{cl}^{sc} variables suffices for most classification purposes. Regardless, one must define a dimensionality-dependent value for the parameter k – the number of local p_{cl}^{sc} values in the domain of P_{cl} to compute P_{cl}'s value locally.



In this example, the 12 given sample tuples are the 12 closest, nearest sample tuples with labels 1 ('•') or 0 ('○'). The P_{cl} value of the unclassified '*' tuple is estimated by using only the 12 nearest classified data as argument values for a localized Shepard probability function as P_{cl}.

The following aspects are of great importance for the overall classification performance of a Shepard probability function and its efficient computation:

i) For the computation of a P_{cl}-value at an arbitrary location, only a "small" number of sample data should be used in a local neighborhood – to speed up the computation and eliminate completely the influence of "far-away data."

ii) Step i) must be supported by a spatial data structure that makes possible the most efficient search for the k closest sample data. For example, data could be organized via a BSP tree.

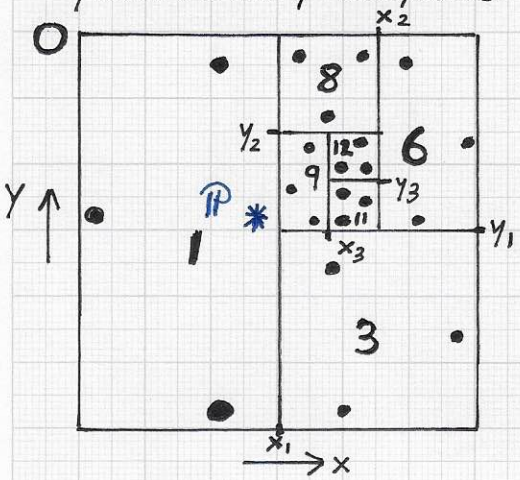
iii) The value of k must be chosen carefully, with the dimensionality of P_{cl}'s domain being particularly relevant.

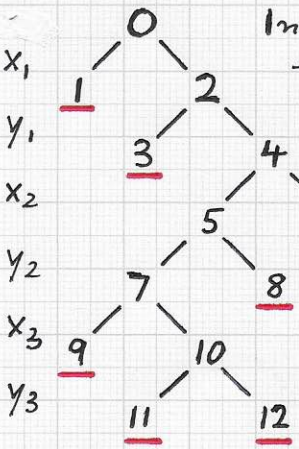
...

■ OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

• Laplacian eigenfunctions and neural networks: ...

2D space partition obtained by repeatedly splitting cells/rectangles at alternating x-midpoints and y-midpoints:





In this ideal scenario, an original square [0] is repeatedly split at alternating x- and y-values, i.e., x_1, y_1, x_2, y_2, x_3 and y_3. This partition was driven by the goal to have less than four points in the resulting rectangles. The binary tree encodes the progression of the sequence of alternating x- and y-splits.

Most importantly, one must use this tree to efficiently determine the k points '•' closest to point P '*'.

Concerning a spatial data structure to organize a set of p_{cl}^{se} values (becoming the coordinate values of point tuples) in such a way that the structure supports the highly efficient search for a point's k closest, nearest "neighbors," one can choose from various well-understood multi-dimensional data structures. The left figure illustrates the final partition of the unit square '0' one obtains when repeatedly splitting rectangles containing more than three points at constant x- and y-values (mid-values of edges). **In our application, such a multi-dimensional data structure is needed to perform the evaluation of a locally applied Shepard probability function - subject to establishing a dimensionality-dependent value for k, the number of closest samples.**

...