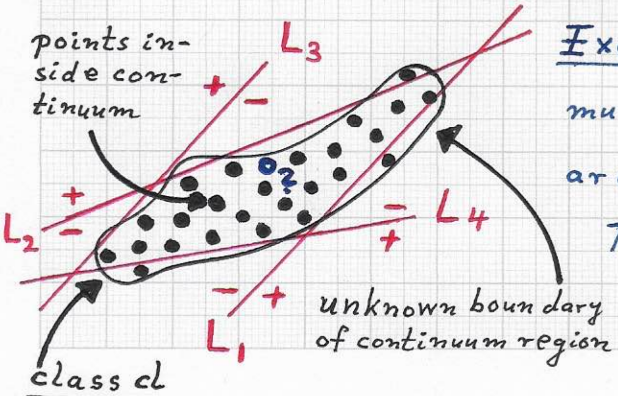
Stratovan

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks:... • **Note.** It is important to recognize the close relationships between **ALGEBRA**, e.g., linear equations, linear inequalities, linear systems and Boolean algebra; **GEOMETRY**, e.g., linear manifolds and linear/piecewise linear approximation; **LOGIC**, e.g., Boolean logic; **APPROXIMATION**, e.g., best approximation, piecewise linear data approximation and least squares methods; **COMBINATORICS**, e.g., computational complexity estimation; **STATISTICS**, e.g., statistical data characterization, statistics-based data processing and statistical evaluation of algorithms/software; **OPTIMIZATION**, e.g., optimal data/function approximation, optimal/minimal computational complexity and optimal/minimal computational complexity of logic circuits/networks/graphs; and **OBJECT AND MATERIAL DETECTION AND CLASSIFICATION**.

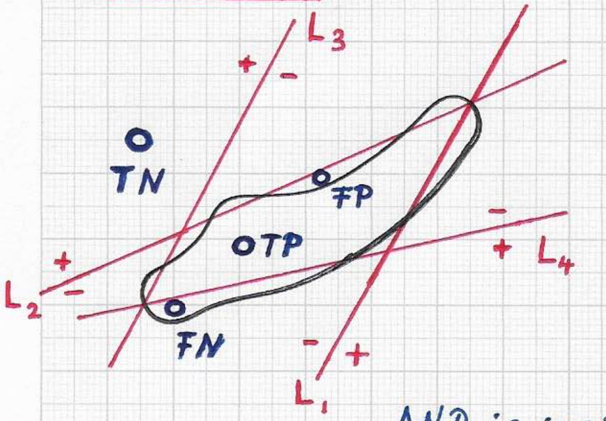


Example (left figure): The point ○ (?) must be classified. Sample points ● are given for some class-cl materials. The exact boundary of the class-cl continuum region is not known. This boundary is "optimally approximated" with oriented lines $L_1, \dots, L_4$.

...

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks:... Here, the class-cl continuum region is sketched as a NON-CONVEX region (unknown). The intersection of the four negative half-spaces defined by $L_1,\dots,L_4$ is a convex quadrilateral in the example. This convex quadrilateral is assumed to be the/an optimal approximation of the (unknown) non-convex continuum region. A point ○ (?) would be interpreted as a class-cl point, if it were lying in all four negative half-spaces associated with $L_1,\dots,L_4$. In other words, a point $\times$ is assumed to be inside or "very close to" (but outside) the class-cl continuum region when $L_1(\times) \le 0$ AND $L_2(\times) \le 0$ AND $L_3(\times) \le 0$ AND $L_4(\times) \le 0$. Using the meaning "Boolean variable" for each L_i, the point $\times$ must yield the value TRUE for the Boolean expression $L_1 L_2 L_3 L_4$ – and $\times$ will then be viewed as a class-cl point.

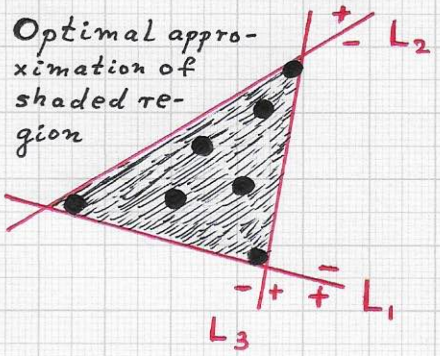


The left figure illustrates the four possible classification results for a point ○ when using the value of $L_1 L_2 L_3 L_4$ as basis for classification: Point ○ is inside (not inside) the continuum region AND is inside (not inside) the quadrilateral
⇒ Point ○ is a TP (TN); point ○ is inside (not inside) the continuum region AND is not inside (inside) the quadrilateral
⇒ Point ○ is an FN (FP). FN and FP cases must be minimal!

...

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks:... Considering the fact that the numbers of FN and FP cases must be minimal, the "geometrical optimization goal" is the optimal approximation of the continuum region representing TRUE (and being unknown) by a region(s) obtained by certain half-space intersections, defined via linear inequalities, leading to a minimal region-to-region approximation error. The left figure is a sketch of the best/optimal approximation problem and solution: The UNKNOWN shaded region must be represented optimally. One is given only a finite, discrete set of sample points ●. The goal is to compute and represent a boundary definition of the shaded region — via definition that is both NECESSARY AND SUFFICIENT. This definition should use a MINIMAL NUMBER of boundary-defining "building blocks" L_i — each one separating space into a negative and positive half-space — placed in a way that MINIMIZES THE ERROR between the exact, unknown shaded region and the region(s) defined by logic-based Boolean half-space expressions involving the "building blocks" L_i as Boolean variables.

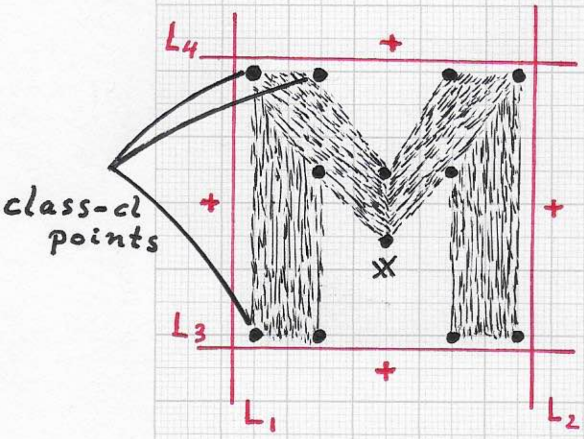


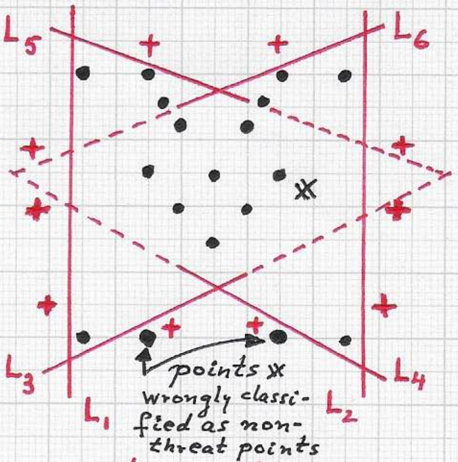
Optimal approximation of shaded region

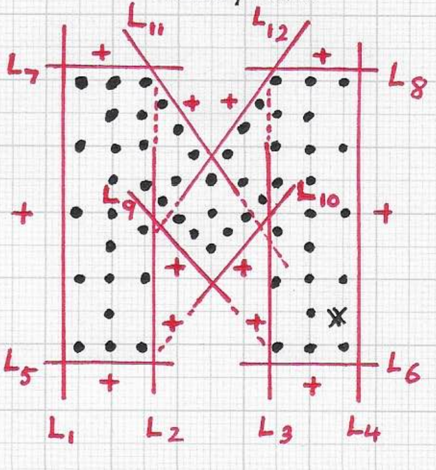
...

■ OBJECT AND MATERIAL EIGENFUNCTIONS — Cont'd.

• Laplacian eigenfunctions and neural networks: ...







Iterative refinement of sample point set and hierarchical multi-resolution representation of boundary. NON-CONVEX case.

The figures shown on this page illustrate important aspects one should consider carefully when designing an algorithm and data structure for the computation and — ideally — hierarchical, multi-resolution representation of the INSIDE region, i.e., the (unknown) shaded region ▩. Since the INSIDE region, in our application, is the region corresponding to RARELY OCCURRING threat materials, the designed multi-resolution (tree) representation should support the rapid, quick determination of non-threat materials — that represent the vast majority of materials to be classified as TNs. One can use a multitude of possibilities to define a/the region in space containing all (or "nearly all") class-cl points ●. Boolean definitions are:

- Top: $P(\times) = L_1 L_2 L_3 L_4$
- Middle: $P(\times) = L_1 L_2 L_3 L_5 \vee L_1 L_2 L_4 L_6$
- Bottom: $P(\times) = L_1 L_2 L_5 L_7 \vee L_3 L_4 L_6 L_8 \vee \bar{L}_2 L_9 L_{10} L_{11} \vee \bar{L}_3 L_9 L_{10} L_{12}$ ...

P: "predicate"

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks:... The example provided on the previous page is used to emphasize three important points to keep in mind:

- (Top) An initial convex region definition must be refined, subdivided to represent a non-convex region containing the points ✱.
- (Middle) It is possible that class-cl points ✱ lie outside the region defined as the class-cl region by a Boolean expression based on the set $\{L_i\}$, leading to FN classifications (two in the sketched scenario).
- (Bottom) One can construct a multitude of increasingly refined definitions of convex sub-regions for the overall class-cl region, using an increasing number of lines L_i. Further one can use a multitude of Boolean expressions, involving L_i as a Boolean variable, all representing the same overall class-cl region.

The evaluation of the final Boolean expression with Boolean variables L_i, $i=1,2,3,\ldots$, yields the desired PREDICATE for a point ✱: P(✱). P(✱) is TRUE when ✱ is interpreted as a class-cl point by the Boolean expression; P(✱) is FALSE when the Boolean expression yields FALSE.

•••