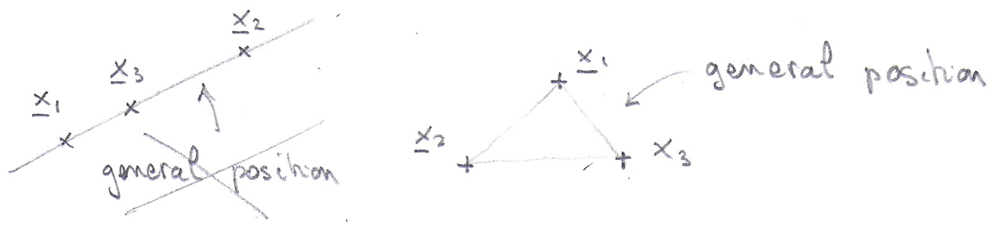
Perceptron capacity

capacity: Fraction of true/false assignments that are linearly separable given a data configuration in general position.

reminder: A configuration $(\underline{x}_1, \dots, \underline{x}_P) \in \mathbb{R}^N$ is in general position if there is no subset $J \subset \{1, .., P\}$, $|J| \leq N+1$ such that $\{\underline{x}_\mu\}_{\mu \in J}$ is linearly dependent.



sufficient condition: no hyperplane contains more than N points

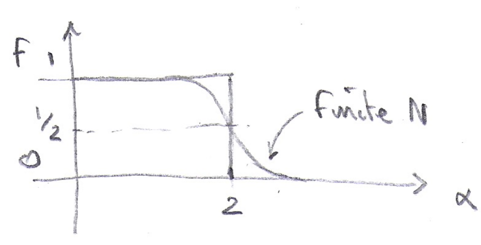
Cover's function counting theorem:

If $(\underline{x}_1, \dots, \underline{x}_P)$ is in general position, the number of linearly separable patterns (ie. the number of realizable dichotomies) is

$$C(P,N) = 2 \sum_{k=0}^{N-1} \binom{P-1}{k} \quad \text{where} \quad \binom{n}{m} = \frac{n!}{m!\,(n-m)!}$$

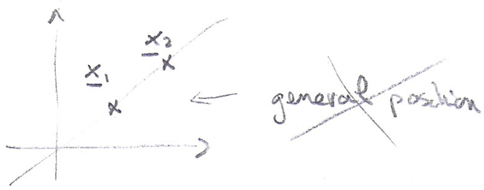
Capacity:
* $P \leq N$: $\quad F(P,N) = 2 \sum_{k=0}^{P-1} \binom{P-1}{k} / 2^P = 2^P / 2^P = 1$
* $P = 2N$: $\quad F(2N,N) = 2 \sum_{k=0}^{2N-1} \binom{2N-1}{k} / 2^{2N} = 2 \, \frac{1}{2} \, 2^{2N-1} / 2^N = 1/2$
* $P \gg N$: $\quad F(P,N) = A P^N / 2^P$ for some $A > 0$

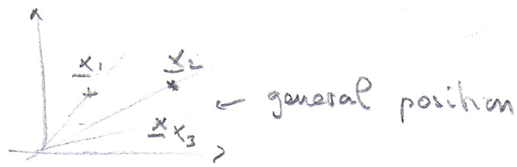
if $\alpha = \frac{P}{N} = \text{constant}$, $N \to +\infty$:



Setting: homogeneous threshold linear function $(\theta = 0)$
threshold

↳ general position: $\text{rank}\left\{\left[\underline{x}_\mu\right]_{\mu \in J, |J| = k}\right\} = \min(k, N)$ for all $J \subset \{1, \dots, P\}$.





Proof: idea → recurrence on the number of points P.

* Suppose $P, N \in \mathbb{N}^*$ and denote $C(P-1, N)$ the number of achievable dichotomies. Denote by: $\underline{w}_i$, $1 \leq i \leq C(P-1,N)$, the weight vector implementing the i-th dichotomy with labels: $y_{i,\mu} = \text{sgn}\left(\underline{w}_i^T \cdot \underline{x}_\mu\right)$ for all $\mu \in \{1, \dots, P-1\}$
* Consider a new point $\underline{x}_P$ such that the configuration remains in general position. Then $\underline{w}_i$ still defines a partition of the P points, that one which assigns labels $\{y_{i,1}; \dots; y_{i,P-1}; \text{sgn}(\underline{w}_i^T \cdot \underline{x}_P)\}$. Thus, $C(P,N) = C(P-1,N) + D$ where D is some possibly configuration-dependent non negative integer.
* The integer D actually counts the dichotomies over the $(P-1)$ points $\mu = 1, \dots, P-1$ for which the P-th point can be labelled true or false. This corresponds to configurations for which there is an hyperplane going through the P-th point and realizing the old dichotomy over $(P-1)$ points: if there is such an hyperplane, a slight change of angle categories the P-th point as true or false; if there is no such hyperplane

the P-th point always lies on the same side of an hyperplane realizing the old dichotomy over P-1 points

* Thus D is the number of dichotomies that can be realized among P points by an hyperplane going through one of the points. That point can be assumed to lie on one of the axis of $\mathbb{R}_n$. The problem is then reduced to finding the number of dichotomies for the remaining (P-1) points projected onto the (N-1) other axes, i.e. $C(P-1, N-1)$, as the points remain in general position once projected.

conclusion: $C(P,N) = C(P-1,N) + C(P-1,N-1)$

Iterating over P, we get $C(P,N) = \sum_{k=0}^{N-1} \binom{P-1}{k} C(1, N-k)$

Observing that there are only two possible labellings of a single point, $C(1,n) = 2$, $n \geq 1$, which yields the result.

	P=1	P=2	P=3	P=4	P=5
N=1	2	2	2	2	2
N=2	2	4	6	8	10
N=3	2	4	8	14	22
N=4	2	4	8	16	30

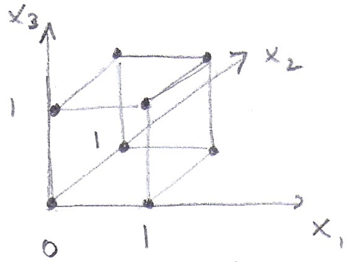
The above proof naturally generalizes to separating hypersurfaces obtained via high-dimensional embedding

$\phi : \underline{x} \longrightarrow (\phi_1(\underline{x}), \phi_2(\underline{x}), \ldots, \phi_d(\underline{x}))$, $d > N$.

This requires introducing the right notion of general position (see Cover)

Capacity for "special" configurations

Example of non-generic configurations.



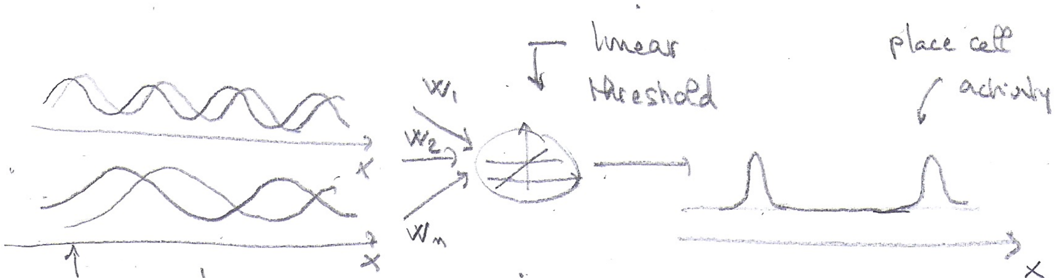
→ hypercube : set of boolean functions

$f : \{0,1\}^k \to \{0,1\}$

→ linear separability of boolean functions NP-hard problem.

Motivation: Place cells / Grid cells

Grid cell input = structured



various phases, discrete period

What is the capacity of place cells activity, as modeled by perceptron?

Discrete model of grid input: M modules of periods λ_i, $1 \leq i \leq M$ each with M grid cells

Ex: 2 modules : $\lambda_1 = 2$, $\lambda_3 = 3$

activity matrix → $A = \left|\begin{array}{cc|cc|cc} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right|$ ↑ cells

position → x

if the λ_i's are pairwise coprime there are $\prod_{i=1}^{M} \lambda_i$ unambiguously encoded position (Chinese remainder theorem).

Single position encoding

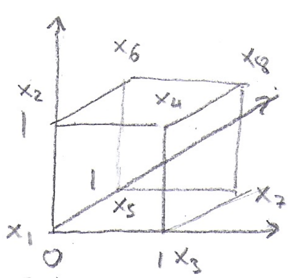
- The capacity of the grid cell encoding for single position readout can be measured via $L = \prod_{i=1}^{M} \lambda_i$, the total number of encoded position
- How optimal is it? Not much: it requires $\sum_{i=1}^{M} \lambda_i$ cells.
  A binary encoding strategy with $\sum_{i=1}^{M} \lambda_i$ cells would yield $2^{\sum_{i=1}^{M} \lambda_i}$ positions.

Hypothesis: The grid code allows for encoding of combination of positions.

$$
\begin{array}{cccccccc}
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\
\hline
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{array}
\;\underset{\text{hierarchy}}{\overset{\text{sparsity}}{\rightleftarrows}}\;
\begin{array}{cccccc}
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\hline
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1
\end{array}
\;\underset{\text{hierarchy}}{\overset{\text{sparsity}}{\rightleftarrows}}\;
\begin{array}{cccc}
x_1 & x_2 & x_3 & x_4 \\
\hline
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{array}
$$

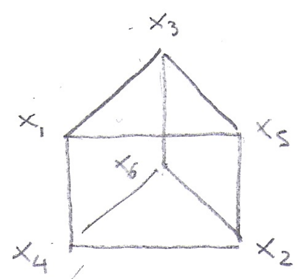
binary — coprime — naive

Different coding strategies



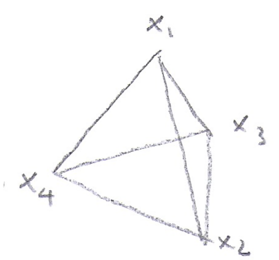
hypercube

x_1, x_2 not linearly separable



prism

= product of simplices



simplex

clear by permutation invariance

Different geometries.

Simplicial geometry: every combination of positions is linearly separable for defining a facet.

- Not true for coprime / gride code geometry

Simplicial decomposition

The convex hull generated by the columns of the activity matrix $A_{\underline{\lambda}}$, $\underline{\lambda} = \{\lambda_1, \dots, \lambda_M\}$, $\gcd(\lambda_i, \lambda_j) = 1$, $i \neq j$, defines d-dimensional polytope $H_{\underline{\lambda}}$ with $d = \sum_{i=1}^{M} \lambda_i - M$. Moreover

$H_{\underline{\lambda}} = \Delta^{\lambda_1} \times \dots \times \Delta^{\lambda_M}$, where Δ^{λ_i} denotes the $(\lambda_i - 1)$-simplex

Proof: By the Chinese remainder theorem, the columns of $A_{\underline{\lambda}}$ have the following product structure:

$$C_{\underline{\lambda}} = \{ \underline{c} = (\underline{c}', \underline{e}_i) \mid c' \in C_{\underline{\lambda}'}, 0 \leq i < \lambda_M \}$$

components corresponding to the M-th module

components corresponding to modules $1 \leq i \leq M-1$.

Indeed, $C_{\underline{\lambda}}$ comprises all the vectors that have a single one in each module $\underline{c} = (\underline{e}_{i_1} \mid \underline{e}_{i_2} \mid \dots \dots \mid \underline{e}_{i_M})$

vector with 0 components except in i_M of length λ_M.

$$H_{\underline{\lambda}} = \left\{ \sum_{i=1}^{\lambda_M} \sum_{j=1}^{L/\lambda_M} \alpha_{ij} (\underline{c}_j, \underline{e}_i) \;\middle|\; \sum_{i=1}^{\lambda_M} \sum_{j=1}^{L/\lambda_M} \alpha_{ij} = 1 \right\}$$

$$= \left\{ \left(\sum_{j=1}^{L/\lambda_M} \left(\sum_{i=1}^{\lambda_M} \alpha_{ij} \right) \underline{c}_j, \sum_{i=1}^{\lambda_M} \left(\sum_{j=1}^{L/\lambda_M} \alpha_{ij} \right) \underline{e}_i \right) \;\middle|\; \sum_{i=1}^{\lambda_M} \sum_{j=1}^{L/\lambda_M} \alpha_{ij} = 1 \right\}$$

$$= \left\{ \left(\sum_{j=1}^{L/\lambda_M} \beta_j \underline{c}_j, \sum_{i=1}^{\lambda_M} \gamma_i \underline{e}_i \right) \;\middle|\; \sum_{i=1} \beta_j = 1, \sum_{i=1} \gamma_i = 1 \right\}$$

$$= \left\{ (\underline{c}', \underline{\delta}) \mid c' \in H_{\underline{\lambda}'}, \delta \in \Delta^{\lambda_M} \right\} = H_{\underline{\lambda}'} \times \Delta^{\lambda_M}$$

Result follows by iterating.