Construction of Minimal n–2–n Encoders for Any n

D. S. Phatak[†] H. Choi and I. Koren

Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA 01003

(Neural Computation, vol. 5, no. 5, Sept. 1993, pp 783-794.)

ABSTRACT

The encoding problem [1] is an important canonical problem. It has widely been used as a benchmark. Here, we have analytically derived minimal–sized nets necessary and sufficient to solve encoding problems of arbitrary size. The proofs are constructive: we construct $n-2-n$ encoders and show that 2 hidden units are also necessary for $n > 2$. Moreover, the geometrical approach employed is general and has much wider applications. For example, this method has also helped us derive lower bounds on redundancy necessary for achieving complete fault tolerance [2, 3].

I Introduction

The encoding problem is an important canonical problem for neural networks [1]. In this problem, a set of orthogonal input patterns are mapped onto a set of orthogonal output patterns through a (small) set of hidden units. Typically, the inputs and outputs are assumed to be binary. There are n input units; n output units and m hidden units where $m \approx \log_2 n$. The hidden units are generally arranged in a single layer resulting in three layers of units. There are n input/output patterns. The hidden units are expected to form some sort of compact code for each of the patterns. Henceforth, we refer to an encoding problem of size n by the acronym $n \times n$ problem and a net for a problem of this size that has m hidden units as an $n - m - n$ encoding net.

The inputs and outputs of the units are continuous valued. That raises the question : are $\log_2 n$ hidden units necessary to solve an $n \times n$ problem ? If less units can do the job, what is the *minimum* number of units needed for an $n \times n$ encoding problem ?

We have analytically derived this minimum number of hidden units and established the capabilities of $n - m - n$ encoding nets. The next section describes the topology and states the assumptions. Section III presents and proves the results on the bounds and related parameters. The following sections present discussion and conclusion.

† D. S. Phatak is now with the EE Department, State Univ. of New York, Binghamton, NY 13790-6000

II Topology

The network is arranged into 3 layers as shown in Figure 1. Every unit in a layer feeds all other units in the next layer. There are no layer-skipping connections. Besides the incoming weights, each unit (in the hidden and output layers) has one more independently adjustable parameter, i.e., threshold or bias. The units are assumed to be sigmoidal and the output of the *i*th unit is given by

$$output_i = S(resultant\_input_i) \quad \text{where} \quad S(u) = \frac{1}{1+e^{-u}} \quad \text{and}$$
$$resultant\_input_i = netinput_i - bias_i \quad \text{and} \quad netinput_i = \sum_{j=1}^{r} w_{ij} o_j \tag{1}$$

Here, r is the number of units that feed unit i and w_{ij} is the weight of the link from unit j (sender) to i (receiver). The output is considered to be *on* or at logical level "**1**" if it is greater than or equal to 0.50 ; and *off* or at level "**0**" if it is less than 0.50 . The input patterns are the rows of $n \times n$ identity matrix. The target outputs are identical to the inputs, i.e., the hidden units are expected to simply replicate the input pattern onto the output layer. The hidden layer *encodes* each of the n patterns with $m < n$ units and the output layer *decodes* the compact codes developed by the hidden units back to original patterns.

III Results

With the above topology and assumptions we now proceed to state the following results.

Theorem 1 : *An encoding net with one single hidden unit (i.e., m = 1) can learn at most 2 × 2 encoding problem.*

Proof : That it can learn 1×1 and 2×2 problems can be demonstrated by giving an example. In Figure 2, a 2 – 1 – 2 net is illustrated along with all the weights and biases. Unit numbers are shown in parenthesis and the bias values are indicated inside the circles representing the units. Units 4 and 5 constitute the input layer and 1 and 2 belong to the output layer. It can be verified that $w_3 = w_4 = b_1 = b_2 = 5.0$; $w_1 = w_2 = 10.0$ along with the signs indicated in the figure lead to correct reproduction of the two input patterns (viz. $\{1,0\}$ and $\{0,1\}$) at the output layer. This is one of the infinitely many sets of weight and bias values that lead to correct outputs.

We now prove that it is impossible to reproduce 3×3 patterns using only one hidden unit. Here, the hidden unit must have 3 distinct outputs, one corresponding to each of the 3 input patterns, otherwise the output units can not distinguish between those patterns that map onto the same output value of the hidden unit. Denote the 3 distinct outputs of the hidden unit as o_1, o_2 and o_3 respectively, where without loss of generality, $o_1 > o_2 > o_3$. Let the weights from the hidden units to the output units be w_1, w_2 and w_3 and biases of the output units be θ_1, θ_2 and θ_3, respectively. Then, the resultant input to the *i*th output unit (denoted by y_i) is given by

$$y_i = w_i x - \theta_i \quad \text{where} \quad i = 1,2,3 \quad \text{and} \quad x = o_1, o_2, o_3 \tag{2}$$

Here, x denotes the output of hidden unit(s). Note that the functions

$$f_i(x) = S[y_i(x)] = \frac{1}{1+e^{-(w_i x - \theta_i)}} \quad \text{where} \quad i = 1,2,3 \quad \text{and} \quad x = o_1, o_2, o_3 \tag{3}$$

are *monotonic.* Without loss of generality, the input patterns are assumed to be {1,0,0}, {0,1,0}, and {0,0,1}. These same patterns should be reproduced at the output, which implies

$$f_1(o_1) = \text{“}\mathbf{1}\text{”}; \; f_1(o_2) = \text{“}\mathbf{0}\text{”}; \; f_1(o_3) = \text{“}\mathbf{0}\text{”}, \quad \text{i.e.,} \quad f_1(o_1) > 0.5; \; f_1(o_2) < 0.5; \; f_1(o_3) < 0.5 \tag{4}$$

$$f_2(o_1) = \text{“}\mathbf{0}\text{”}; \; f_2(o_2) = \text{“}\mathbf{1}\text{”}; \; f_2(o_3) = \text{“}\mathbf{0}\text{”}, \quad \text{i.e.,} \quad f_2(o_1) < 0.5; \; f_2(o_2) > 0.5; \; f_2(o_3) < 0.5 \tag{5}$$

$$f_3(o_1) = \text{“}\mathbf{0}\text{”}; \; f_3(o_2) = \text{“}\mathbf{0}\text{”}; \; f_3(o_3) = \text{“}\mathbf{1}\text{”}, \quad \text{i.e.,} \quad f_3(o_1) < 0.5; \; f_3(o_2) < 0.5; \; f_3(o_3) > 0.5 \tag{6}$$

From (3) it is seen that constraints (4) and (6) can be satisfied since they obey monotonicity. Constraints (5), however, cannot be satisfied since the function on the left–hand side is monotonic while the required outputs on the right–hand side are not monotonic. It can be verified that for any permutation of input patterns and output values, the constraints on one of the three units are impossible to satisfy since the inputs to that unit are *monotonic* but the target outputs are *not monotonic*. Thus the 3×3 problem cannot be solved by just one hidden unit.

The proof for the $n \times n$ sized problem with $n > 3$ is identical to the above proof for 3×3 case.

Q.E.D.

There is a geometrical interpretation of the above result which is illustrated in Figure 3. This interpretation is critical for the proof of the next theorem which establishes a bound for the general $n \times n$ problem. For a $2-1-2$ net, the output of the hidden unit corresponding to each of the (input) patterns can be represented by a point along 1 dimension or a line. Without loss of generality, Choose that line to be the x axis. Then, the output of the hidden unit corresponding to each of the 2 input patterns is a point between [0,1] on the x axis, as illustrated by points P_1 and P_2 in Figure 3. Because of the one–to–one mapping from the input patterns to the points representing the outputs of hidden units, the symbols P_1 and P_2 will also be used to refer to the *patterns*. The resultant input to the ith unit is given by equation (2), where $i = 1,2$ and w_i and θ_i are the weight and bias associated with the ith unit. Note that these equations represent straight lines (hyperplanes in general) in the x-y plane, as illustrated by lines l_1 and l_2 in Figure 3. Henceforth, we just use the labels 1 and 2 to refer to the output units as well as the corresponding lines (hyperplanes) implemented by the units. A point x_0 is considered to be on the *positive* side of the line $y = wx - \theta$ if $wx_0 - \theta > 0$; and on the *negative* side of the line if $wx_0 - \theta < 0$. For example, in Figure 3, all points (on the x axis) to the right of point Q are on the positive side of line l_1 and on the negative side of line l_2. The vertical distance P_1A between point P_1 and the line l_1 represents the *resultant_input* to output unit 1 for pattern P_1. Similarly, distance P_1B represents the *resultant_input* to unit 2 for pattern P_1. It is useful to think of *directed* distance from the points P_1, P_2 to lines l_1, l_2. If the direction is upwards (along $+y$ axis), then the corresponding *resultant_input* is positive (i.e., the output of the unit is "**1**"), while a downwards (along $-y$ axis) distance implies a negative *resultant_input* ("**0**" output). For the patterns (points) on the positive side of the line, the resultant input to the corresponding unit is positive and the unit output is *on* or "**1**". Conversely, a unit is *on* only if the pattern lies on the positive side of the line it implements. Similarly, a unit is *off* if and only if the pattern lies on the *negative* side of the line corresponding to the unit.

Learning implies finding weights and biases that satisfy the constraints

$$y_1(o_1) > 0 \ ; \ y_1(o_2) < 0 \ ; \ y_2(o_1) < 0 \ ; \ y_2(o_2) > 0 \tag{7}$$

The first two inequalities say that points P_1 and P_2 must be on positive and negative sides of line l_1, because unit 1 should be *on* for pattern 1 and *off* for pattern 2. The interpretation of the last two inequalities is similar. Together, the constraints imply that both lines l_1 and l_2 intersect the x axis between P_1 and P_2 and that one of them has a positive slope and the other has a negative slope. Figure 3 illustrates a case where the points P_1, P_2 and lines l_1, l_2 satisfy the above constraints. In this figure, both l_1 and l_2 intersect the x axis at the same point Q. In general, this may not be the case, as long as the constraints are satisfied.

In general, learning implies constraints similar to (7). The constraints are such that

1. An output unit is *on* for only one pattern. This means that the weight(s) and bias associated with that unit define a hyperplane which is such that only one of the points P_i is on its *positive* side, all others are on its *negative* side.

2. Each point P_i is such that for the corresponding input pattern, only one output unit is *on* and this unit stays *off* for all other input patterns. This means that each of the points P_i it is on the *positive* side of exactly one hyperplane and on the *negative* side of all others.

In Figure 3, P_1 is on *positive* side of only one line viz. l_1 and P_2 is on *positive* side of only one line viz. l_2 . Similarly line l_1 has only one point on its *positive* side viz. P_1 and line l_2 has only one point on its *positive* side viz. P_2.

For the $n \times n$ encoding problem, it may be expected that the minimum number of hidden units required is a function of n. Contrary to this expectation, however, it turns out that only 2 hidden units are sufficient to solve any $n \times n$ problem for arbitrarily large n.

Theorem 2 : *Only 2 hidden units are sufficient to encode and decode $n \times n$ patterns for any positive integer n.*

Proof : We prove this by a geometrical construction similar to the one illustrated above for the $2-1-2$ case. Here the network is $n-2-n$, i.e., there are n input units, 2 hidden units and n output units. For each input pattern, the hidden units develop outputs that can be represented by a *distinct* point in the *x*-*y* plane, where the x coordinate denotes the output of the 1st hidden unit and the y coordinate denotes the output of the 2nd hidden unit. These points are denoted by $P_i \ ; \ i = 1, 2..n$.

The hidden units feed all the output units. Let the weight associated with the link between hidden unit 1 and output unit i be denoted by w_i^1 . The weight from hidden unit 2 to output unit i is denoted by w_i^2. Let the bias of the output unit i be denoted by θ_i. Then, the resultant input to the ith output unit (denoted by z_i) is given by

$$z_i = w_i^1 x + w_i^2 y - \theta_i \quad \text{where} \quad i = 1, \cdots, n \quad \text{and} \quad (x, y) = (o_1^1, o_1^2), \cdots, (o_n^1, o_n^2) \tag{8}$$

Here, x and y correspond to the axes or dimensions representing the outputs of the hidden units, and z represents the dimension that corresponds to the $resultant\_input$ to the output units. These equations represent (hyper) planes in the 3-D space and that will henceforth be denoted by Π_i where $i = 1, \cdots, n$. These planes are the decision surfaces implemented by the corresponding units. We say that a point (x_0, y_0) is on the *positive* side of plane Π_i if

$$z_0 = w_i^1 x_0 + w_i^2 y_0 - \theta_i > 0 \tag{9}$$

and on the *negative* side if

$$z_0 = w_i^1 x_0 + w_i^2 y_0 - \theta_i < 0 \tag{10}$$

In order to map the input patterns onto the output patterns, the points P_k and the planes Π_i have to satisfy constraints similar to those listed above in the exposition on geometrical interpretation. Once again we observe that each plane Π_i defines the output of one of the units in the output layer, and each of the points P_k corresponds to a pattern. An output unit is *on* only for one of the n patterns and *off* for others. Similarly, each pattern has exactly one output unit *on* and all others *off*. These constraints can be geometrically interpreted as follows :

1. Each plane Π_i has only one point on its *positive* side, all other points are on its *negative* side.

2. Each point P_k is on the *positive* side of only one plane and on the *negative* side of all other planes.

If there exist points P_k and planes Π_i ; $i, k = 1, 2.., n$ that satisfy the above constraints, then they constitute a valid solution for the $n \times n$ problem using only 2 hidden units. Figure 4 shows the geometrical construction that proves the existence of such solution(s). It shows a $6-2-6$ case for the purpose of illustration, but the procedure can be applied to any $n-2-n$ problem.

As a first step toward the solution of the $n-2-n$ problem, a regular polygon of n sides is constructed in the *x-y* plane. This is illustrated by the hexagon with vertices (a,b,c,d,e,f) drawn in solid linestyle in Figure 4. Next, every edge is extended beyond the vertex up to a point where it meets the extension of some other edge of the polygon, so that (isoceles) triangles are obtained on the exterior of the original polygon, with the edges of the polygon as the bases of these triangles. This is illustrated by the shaded triangles in Figure 4. Now consider the original polygon as the base of a pyramid or a cross section of the pyramid along the *x-y* plane. The faces of the pyramid intersect at a point directly (vertically) below (along the $-z$ direction) the center of the circumcircle of the polygon. In Figure 4, for example, the center of the circumcircle is labeled as V. The vertex of the hexagonal pyramid lies directly (vertically) below the point V (i.e., on a line in the $-z$ direction, directed into the page from point V). The n faces of the pyramid define the n planes Π_i. The points P_k have to be located within the isoceles triangles on the exterior of the polygon in the *x*-*y* plane, in order to satisfy the two constraints mentioned above. One point is placed inside each triangle, as illustrated by points $P_1, \cdots, P_6$ inside the shaded triangles in Figure 4.

With this construction, each plane Π_i is such that only one point is on its *positive* side and all other points are on its *negative* side. For example, in Figure 4, the plane Π_1 passing through the vertex of the pyramid and edge *ab* is such that only one point, viz., P_1 is on its *positive* side while all others are on its *negative* side. Similarly, each point is on *positive* side of exactly one plane and *negative* side of all others. In Figure 4, for example, point P_2 is on the *positive* side of plane Π_2 only, and is on the *negative* side of all other planes.

Thus the points and planes satisfy all the above constraints and represent a valid solution. The outputs of all the units have to be in [0,1]. This means that the entire diagram should be within the unit square in the *x*-*y* plane, which is bounded by vertices (0,0), (0,1), (1,0), (1,1). This is always possible to do since the polygon can be shrunk to any desired size so that the entire diagram can fit inside the unit square. This proves that a solution (in fact infinitely many of them) always exists to the $n-2-n$ problem and can be obtained by the above construction.

Q.E.D.

IV Discussion

Above results hold for the complementary encoding problem (0's and 1's are interchanged) as well. For a complementary encoding problem, the vertex of the pyramid in the above construction lies directly (vertically) above the circumcenter V, which is in the *x*-*y* plane. Also note that the I/O patterns for the complementary encoding problem are *not* mutually orthogonal.

In the above construction, the points corresponding to the outputs of the hidden units must lie within the triangles formed on the edges of the polygon. Hence the area of the triangles is, in a crude sense, related to the probability of finding a valid solution. The larger the area, the higher is the probability that the gradient descent will latch on to a valid solution. Note that the outputs of the hidden units are confined to be between two circles, viz., an inner circle which touches (is tangent to) each edge of the polygon and an outer circle that passes through the tips of all the triangles on the exterior of the polygon. Both these circles are drawn in dotted linestyle in Figure 4. For a given *n*, the triangles have the largest area when the outer circle is as large as possible, i.e. it touches the edges of unit square in the $x-y$ plane. Hence the net is more likely to hit upon this solution. This is consistent with the observation that neural nets tend to stabilize at vertices or corners of the solution space.

As $n \rightarrow \infty$, the circles approach each other and in the limit they coincide. This means that the volume (area in this case) of the solution space approaches 0 and therefore, the probability that the search algorithm converges to a valid solution also approaches 0, as expected.

The distance (along the *z* direction) between the point P_r and the corresponding plane Π_r represents the resultant input to a unit. In the limit as $n \rightarrow \infty$, the points P_i approach planes Π_i and the vertical distance between the planes and the points approaches 0 as well. This means that the resultant inputs to the output units approaches 0. Hence the outputs of units that are *on* approach 0.5 from above, i.e., output values indicating a logical level $\text{“}\mathbf{1}\text{”} \rightarrow 0.5+$ and the outputs of the units that are *off* approach the limit 0.5 from the other side, i.e., logical $\text{“}\mathbf{0}\text{”} \rightarrow 0.5-$.

If the output tolerances are are specified (for example a “**1**” cannot be below 0.75 and a “**0**” cannot be above 0.25) then, in the above construction, it is possible to find out the maximum value of n that will deliver the outputs within the desired tolerances, for a given m. Conversely, given an n, the number of hidden units m required to deliver the outputs within the specified tolerance can be also calculated from the above construction.

If $n \leq 4$, the “allowable” regions for the points P_i are no longer triangles since the edges of a regular polygon with $n \leq 4$ sides when extended beyond the vertices, do not intersect the extensions of any of the other edges.

It should also be noted that in the above construction, the polygon need not be regular. If the polygon is not regular, however, some of the “allowable” areas shrink and others expand. Also, the planes Pi_i need not intersect at the same point or need not form a pyramid, as long the relative placement of the planes and the points satisfy the two constraints mentioned above.

The unbounded allowable areas for points P_i that arise due to $n \leq 4$ or due to irregularity of the underlying polygon, as well as the asymmetry in allowable areas that arises when the polygon is irregular is illustrated in Figure 5. Note that the construction remains the same in all these cases. The points P_i still have to be in the regions exterior to the polygon, and between the lines obtained by extending the edges of the polygon beyond the vertices. This is illustrated by the shaded regions in Figure 5. If the quadrilateral shown in Figure 5 was regular, i.e., it was a square, then all the “allowable” regions for points P_i would be identical in shape and unbounded on one side. Because the quadrilateral is irregular, some allowable regions have shrunk and others have grown. For example, the shaded region to the left of plane Π_2 has shrunk from a rectangular strip unbounded on left side, to a bounded and triangular region shown in the figure. Similarly the shaded region to the right of Π_4 has expanded from a rectangular strip to to an unbounded quadrilateral.

It seems that the symmetric solution is more fault–tolerant. The reasoning is as follows. The edges and planes of the polygon can be jiggled without changing the classification or logical output of the network. This corresponds to changing the weights and biases of the units represented by the planes. How much change is allowed in the weight and bias values depends on n and other factors. For the symmetric solution, it is evident that whatever tolerance applies to a point or a plane also applies to all other points or planes. In contrast, if the polygon is not regular or if the planes do not form a pyramid, then some points and planes must be confined to smaller tolerances (smaller than the corresponding one in the symmetric case) while others can have larger tolerance. The total amount of deviation allowed can be measured by the volume enclosed between the original positions of the planes and the extreme positions after large deviations in parameters (or faults), at which the solution (relative placement of planes and points) still satisfies the above constraints. It is conjectured that the total of such “fault–tolerance volumes” is maximum for the symmetric case, or in other words, a symmetric solution is more fault–tolerant.

V Conclusion

Bounds have been established for the solution of the encoding problem using a feedforward network with one layer of hidden units. Existence of solution(s) is demonstrated by constructive proofs, leading to the actual solutions. The discussion reveals interesting connections to limiting cases, fault tolerance, probability of finding a valid solution and other issues. The geometrical interpretation is general and applicable to other problems as well. For instance, this approach was employed in [2, 3]. to derive lower bounds on the redundancy necessary to achieve complete fault tolerance for all single faults. The encoding problem directly reflects on the ability of the net to develop distributed representations among the hidden units and map them back onto localized representations on the output units. These results will possibly help to define a meaningful measure of the distributedness of representations.

References

[1] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing, vol. 1 : Foundations*. MIT Press, 1986.

[2] D. S. Phatak and I. Koren, "Fault Tolerance of Feedforward Neural Nets for Classification Tasks," in *Proceedings of International Joint Conference on Neural Nets (IJCNN), Baltimore, MD*, vol. II, pp. II–386 – II–391, Jun. 1992.

[3] D. S. Phatak and I. Koren., "Complete and Partial Fault Tolerance of Feedforward Neural Nets," Tech. Rep. TR-92-CSE-26, Electrical and Computer Engineering Department, University of Massachusetts, Amherst, July 1992.

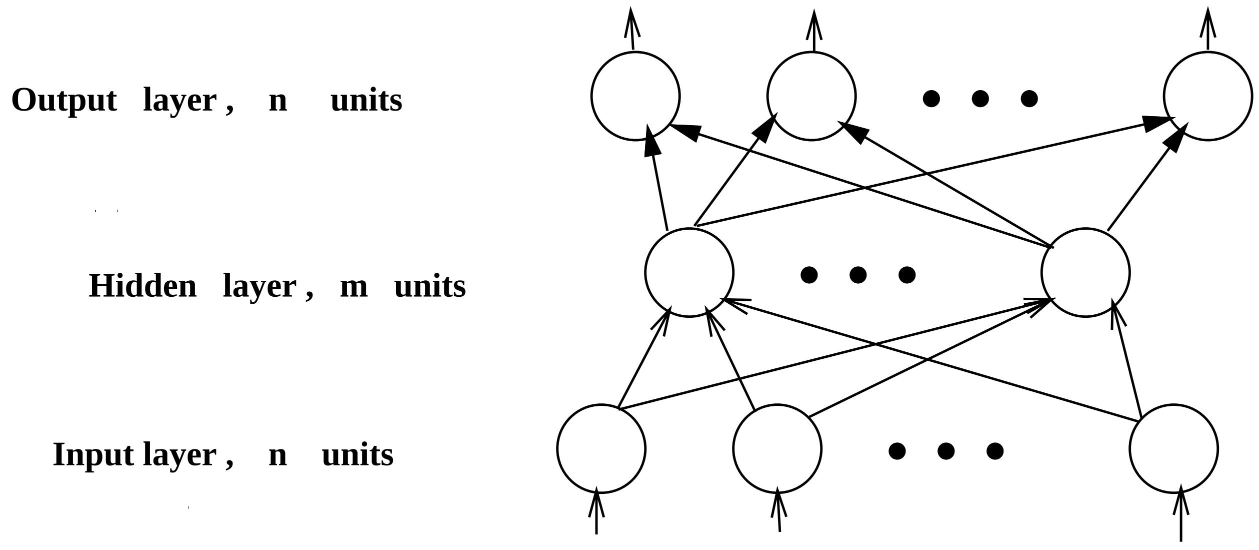

Figure 1 : An $n - m - n$ encoding net.

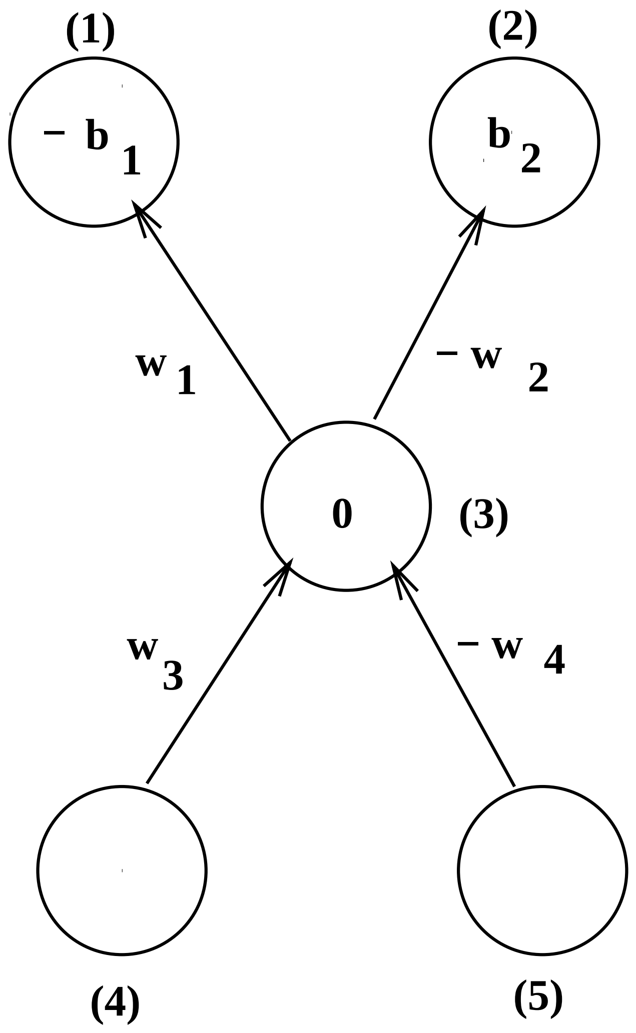

Figure 2 : A $2-1-2$ encoding net with weights and biases w_i and $b_i > 0$ for all i. Unit indices are shown in parenthesis.

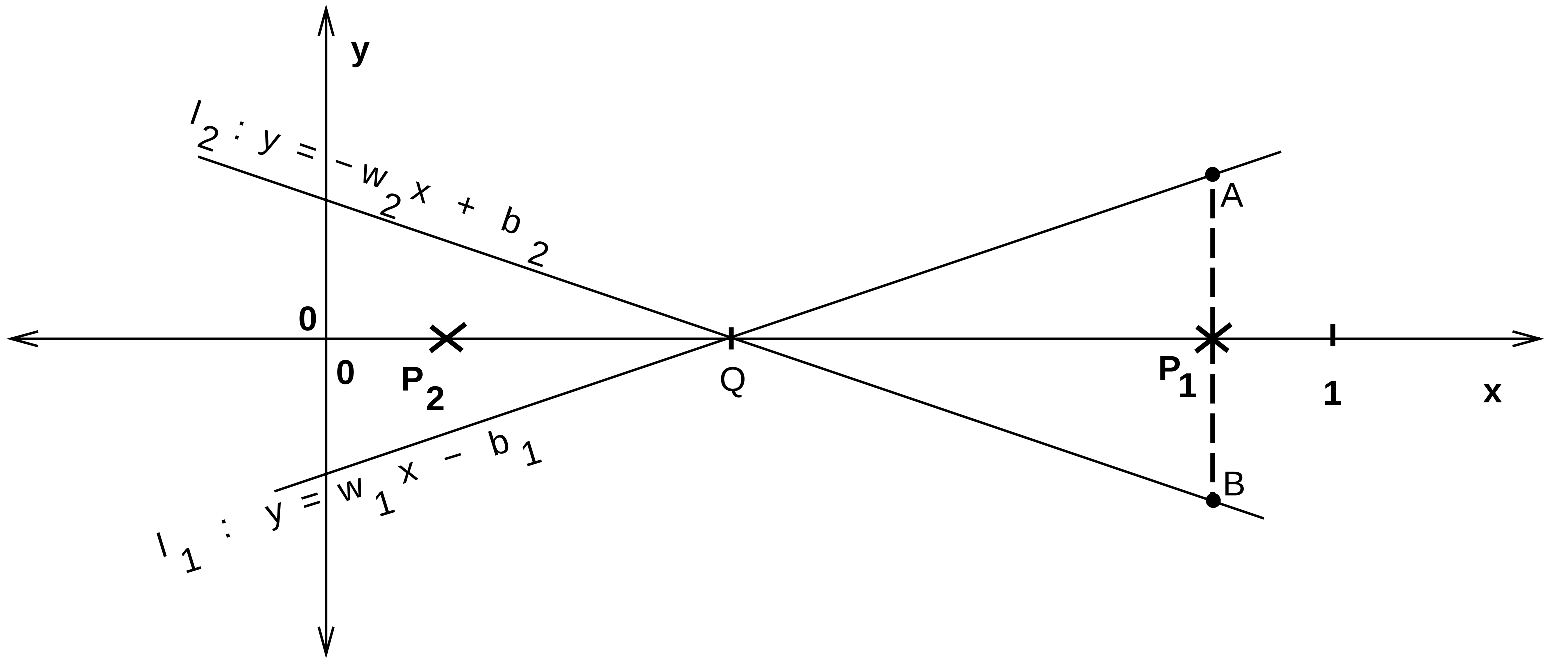

Figure 3 : A geometrical interpretation of the 2 – 1 – 2 encoding problem.

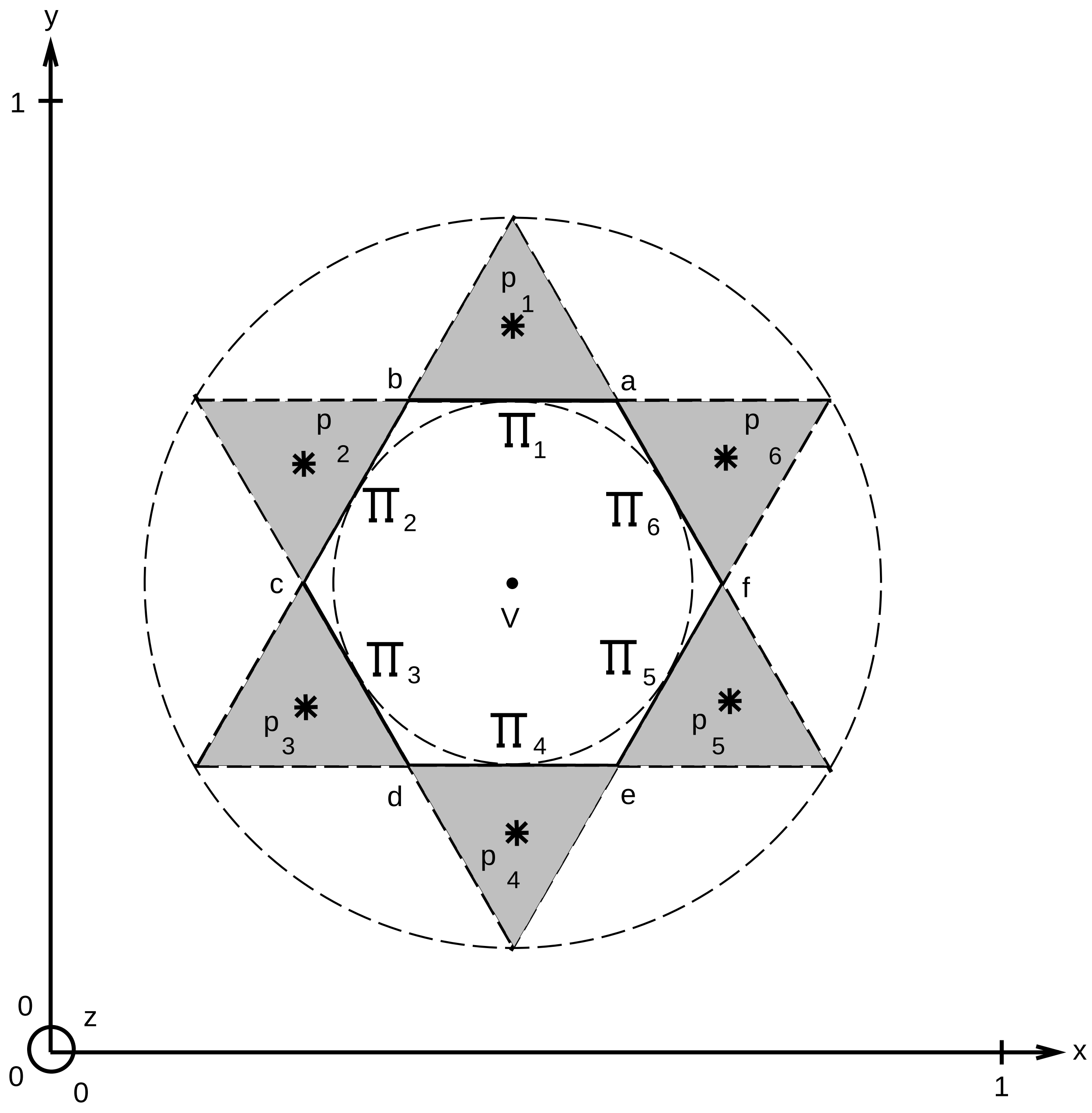

Figure 4 : The geometrical construction to obtain the weights and biases for a $6-2-6$ (or $n-2-n$ in general) encoding net.

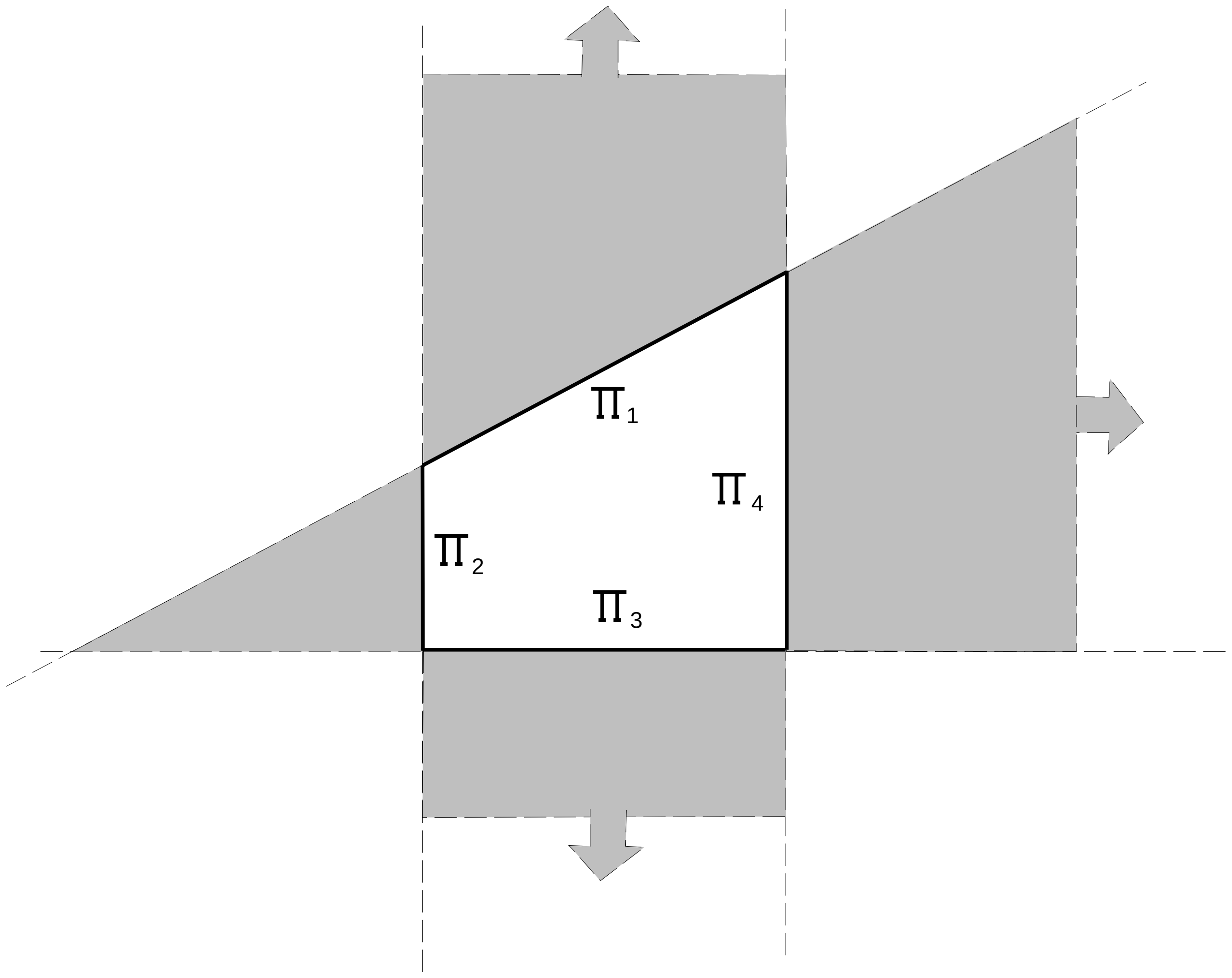

Figure 5 : The construction for the case when $n \leq 4$ and the polygon is not regular.