Introduction: The Perceptron

Haim Sompolinsky, MIT

October 4, 2013

1 Perceptron Architecture

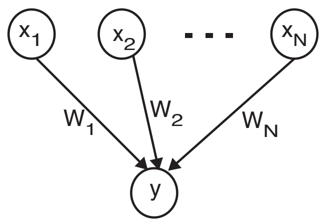
The simplest type of perceptron has a single layer of weights connecting the inputs and output.

Formally, the perceptron is defined by $y = sign(\sum_{i=1}^{N} w_i x_i - \theta)$ or

$$y = sign(w^T x - \theta) \tag{1}$$

where w is the weight vector and θ is the threshold. Unless otherwise stated, we will ignore the threshold in the analysis of the perceptron (and other topics), because we can instead view the threshold as an additional synaptic weight, which is given the constant input -1. This is so because $w^T x - \theta = [w^T, -1] \begin{bmatrix} x \\ \theta \end{bmatrix}$.

Figure 1: Perceptron Network



Perceptron's decision surface. It is easy to visualize the action of the perceptron in geometric terms because w and x have the same dimensionality, N.

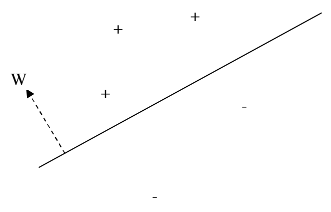


Figure 2 shows the surface in the input space, that divide the input space into two classes, according to their label. This is an example of a decision surface of a machine that outputs dichotomies. In this case, it is just a hyperplane drawn in input space and passes through the origin. The alignment of the hyperplane is perpendicular to the vector w . We will some time identify the plane by its associated weight vector w.

Any set of labled points that can be separated by a hyperplane (through the origin) is said to be a linearly separable dichotomy. However, there are many interesting simple dichotomies that cannot be realized by a perceptron; these are non-linearly separable problem. A famous example is the XOR.

2 Perceptron's Capacity: Cover Counting Theorem

Before we discuss learning in the context of a perceptron, it is interesting to try to quantify its complexity. This raises the general question how do we quantify the complexity of a given archtecture, or its capacity to realize a set of input-output functions, in our case-dichotomies. A straightforward simple answer would be to count the number of different functions that this architecture can implement. However, this measure is meaningless if the inputs come from some continuous space, as in this case, every arbitrarily small changes in the decision boundary will correspond to a new dichotomy.

One way to overcome this difficulty is to constrain the input to a fixed finite set of P input vectors, $\{x^1, ..., x^P\}$. We can now attempt to count the number of functions (dichotomies) that can be implemented by a given network architecture on this fixed set of inputs.

We will now apply this idea in the case of the perceptron. Before doing so, there is another potential complication which is that in general, the number of dichotomies that can be realized by a given architecture may depend also on the particular set of inputs. Fortunately enough this is not the case in the Perceptron. In fact, the number of linearly realizable dichotomies on a set of points depend only on a mild condition, known as general position.

General position is the extension of the concept of linear dependence. We can demand that the input set is linearly independent since P may be larger than N. General position is the condition that $\{x^1, ..., x^P\}$ has *no subset of size less than$N$* that is linearly dependent. General position is a necessary condition for linear separably. It is clear that if for instance, two oppositely labeled points lie on a line (emanating from the origin) they cannot be a plane intersecting the origin that separates them. On the other hand, this is a very mild condition that is obeyed by any examples generated by $P(x)$ which varies smoothly in R^N, and in high dimension, it is obeyed with probability close to 1 even when the inputs are discrete valued. The calculation of the number of linearly realized dichotomies is given by the following theorem.

Cover's Function Counting Theorem (Cover 1966):

Theorem: Let $\left\{x_1, ...x^P\right\}$ be vectors in R^N, that are in general position. Then the number of distinct dichotomies applied to these points that can be realized by a plane through the origin is:

$$C(P,N) = 2\sum_{k=0}^{N-1}\binom{P-1}{k} \tag{2}$$

Reminder:for $m \leq n$, $\binom{n}{m} = \frac{n!}{m!(n-m)!}$. For $m < n$, we define $\binom{n}{m} = 0$.

Before we prove the theorem let us discuss some of its consequences.

Some consequences of Cover's theorem:

1. For $P \leq N$ the above sum is limited by $P-1$, so it can be rewritten as

$$C(P,N) = 2\sum_{k=0}^{P-1}\binom{P-1}{k} = 2(1+1)^{P-1} = 2^P \tag{3}$$

(using the familiar binomial expansion). In other words, *all* possible dichotomies can be realized.

2. for $P = 2N$,

$$C(2N,N) = 2\sum_{k=0}^{N-1}\binom{2N-1}{k} = 2\frac{1}{2}2^{2N-1} = 2^{P-1} \tag{4}$$

This is so because the expression for C contains exactly one half of the full binomial expansion of $(1+1)^{2N-1}$, and we know that this expansion is symmetrical (for example, the first few (odd) binomial expansions are $(1,1)$, $(1,3,3,1)$, and $(1,5,10,10,5,1)$. We conclude that in this case exactly half of all possible dichotomies are able to be realized.

3. For $P \gg N$, it is easy to see that $C(P,N) \sim AP^N$ for some $A > 0$.

When P is large compared to N the number of dichotomies that can be implemented still grows with P, but the number of dichotomies that can be implemented in proportion to the number of possible dichotomies shrinks, since the total number of possible dichotomies is 2^P. Figure 4 shows this phenomena, in logarithmic scale: for small P, the number of dichotomies is maximal, i.e. 2^P, and thus linear in the logarithmic scale. When P becomes large, the number of possible dichotomies becomes only polynomial, and hence logarithmic in the logarithmic scale.

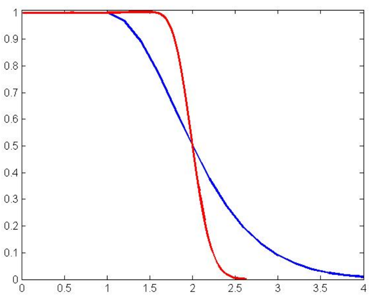


Figure 3 : *Fraction of linearly separable dichotomies vs.* P/N *for* $N = 5$ *and* 65

4. Another way to look at the behavior of $C(P,N)$ is to allow both P and N to approach ∞, but to keep them proportional, i.e. $\frac{P}{N} = \alpha$. In this case, as $N \to \infty$, $C(P,N)$ vs. $\frac{P}{N}$ becomes a step function, as shown in figure 5. Note that the shape of the step function must of course still obey the properties that we found before, in particular $\frac{C(2N,N)}{2^N} = 0.5$. This is in fact the value around which the step occurs, i.e. below the critical value of $\frac{P}{N} = 2$ we see that virtually all of the dichotomies are possible, and above that value virtually none are possible.

2.1 Proof of Cover's Theorem:

Start with P points in general position. We now assume that there are $C(P, N)$ dichotomies possible on them, and ask how many dichotomies are possible if another point (in general position) is added, i.e. what is the value of $C(P+1, N)$. In this way we will set up a recursive expression for $C(P, N)$.

Let $(b_1, ..., b_P)$ be a dichotomy realizable by a hyperplane over the set of P inputs – in other words, $b_i \in \{-1, +1\}$ for every $i = 1..P$, and there is a set of weights w so that for each of them $\big(sign(w^T x^1), ..., sign(w^T x^P)\big) = (b_1, ..., b_P)$. Using one such w, we get a dichotomy over the set of $P + 1$ inputs:

$$\big(sign(w^T x^1), ..., sign(w^T x^{P+1})\big) = (b_1, ..., b_P, sign(w^T x^{P+1})) \tag{5}$$

Thus for every linearly realized dichotomy over P points there is at least one linearly realized dichotomy over $P + 1$ points. Note that different dichotomies over P points define different dichotomies over $P + 1$ points, since they differ somewhere in the first P co-ordinates.

Now, potentially, the additional dichotomy $(b_1, ..., b_P, -sign(w^T x^{P+1}))$ (reversing the sign of the last co-ordinate) is also possible, by some other set of weights? In this way $C(P + 1, N)$ can be higher than $C(P, N)$. Let us write

$$C(P + 1, N) = C(P, N) + D \tag{6}$$

Our goal is to find D, the number of additional dichotomies.

Let us assume that one of the weight vectors W that generates $(b_1, ..., b_P)$ passes directly through x^{P+1}, as shown in the figure. In this case, it is clear that by slight changes in the angle of the hyperplane we will be able to move the hyperplane slightly to this side or the other of x^{P+1} - thus getting a value of either $+1$ or -1 on it. Thus in this case, *both* $(b_1, ..., b_P, +1)$ and $(b_1, ..., b_P, -1)$ are possible, and there is one additional possible dichotomy beyond $C(P, N)$ (that is counted in D). On the other hand, if no hyperplane that passes through x^{P+1} (and generates $(b_1, ..., b_P)$ on the first P vectors) exists, then it means that the point lies in one side of all the planes that generate the old dichotomy, hence we will *not* be able to achieve both dichotomies, unlike before. We have thus seen that D is the number of those dichotomies over P points that are realized by a hyperplane that *passes through a certain fixed point* x^{P+1} (which is in general position with the other points). [Show Figure]. Now, by forcing the hyperplane to pass through a certain fixed point, we are in fact moving the problem to one in $N - 1$ dimensions, instead of N. This can be understood if the point is on the x axis, for example - then the hyperplane has $N - 1$ axes left to "work with" (if the point is not on the x axis, then rotate the axes of the space around to get the point on the x axis, and this of course has no effect on the geometry of the problem). In conclusion, $D = C(P, N - 1)$, and the recursion formula is

$$C(P + 1, N) = C(P, N) + C(P, N - 1) \tag{7}$$

We now prove the theorem by induction. Let us assume that

$$C(P,N) = 2\sum_{k=0}^{N-1}\binom{P-1}{k}$$

holds up to P and N [Note that it trivially holds for $P = 1$ and all N, since it gives $C(1,N) = 2$ as expected, since one point in N dimensions can be dichotomized with the two labels by a hyperplane]. Then,

$$\begin{aligned} C(P+1,N) &= 2\sum_{k=0}^{N-1}\binom{P-1}{k} + 2\sum_{k=0}^{N-2}\binom{P-1}{k} \\ &= 2\sum_{k=0}^{N-1}\binom{P-1}{k} + 2\sum_{k=0}^{N-1}\binom{P-1}{k-1} = 2\sum_{k=0}^{N-1}\binom{P}{k} \end{aligned} \tag{8}$$

where we have used $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ [and used the convention that $\binom{n}{k} = 0$ for $k < 0$] .

3 The Perceptron Learning Algorithm

Given a set of input vectors$\{x^1, ..., x^P\}$, and a set of desired labels $\{y_0^1, ..., y_0^P\}$, we want to find an algorithm that, starting from some initial weight vector, it will modify it on the basis of the examples ultimately yielding set of weights w that classify correctly all the examples,

$$sign(w^T x^\mu) = y_0^\mu, \; \forall\mu \tag{9}$$

The famous Perceptron Learning Algorithm that is described achieves this goal. The PLA is incremental. Examples are presented one by one at each time step, and a weight update rule is applied. Once all examples are presented the algorithms cycles again through all examples, until convergence.

The PLA begins with an arbitrary set of weights w^0. At each subsequent step, call it the n$^{\text{th}}$ step, we have the weights arrived at in the previous step, w^{n-1}, and a new example (x^n, y_0^n). The weights are then updated according to the following rules: If

$$y_0^n(w^{n-1T}x^n) > 0 \tag{10}$$

implying that the current weight vector classifies correctly the new example, then

$$w^n = w^{n-1} \tag{11}$$

and the algorithm moves to the next step. If on the other hand,

$$y_0^n(w^{n-1T}x^n) < 0 \tag{12}$$

implying that error has occurred, then,

$$w^n = w^{n-1} + \eta y_0^n x^n \tag{13}$$

These rules can be summrized as

$$w^n = w^{n-1} + \eta y_0^n x^n \Theta(-y_0^n w^{n-1T} x^n) \tag{14}$$

where $\Theta(x)$ is a step function, returning 0 for $x < 0$ and 1 for $x > 0$.

The algorithm halts when all the examples are classified correctly by w . If there was no error, then the algorithm halts, and all the training examples are classified correctly. Whether the algorithm finds w that realizes the task depends first of all on whether the data is linearly separable. Assuming that it is, then the theorem discussed below, ensures that the PLA will find a weight vector that correctly classifies the data (although the solution may not coincide with w^*as the solution will not be unique).

Perceptron Convergence Theorem:

For any finite set of linearly separable labeled examples, the Perceptron Learning Algorithm will halt after a finite number of iterations.

In other words, after a finite number of iterations, the algorithm yields a vector w that classifies perfectly all the examples.

Note: Although the number of iterations is finite, it is usually larger than the size of the training set, because each example needs to be processed more than once.

Proof:

We note by W^* a 'teacher' weight vector that correctly separates the training examples, i.e., it obeys

$$y_0^\mu w^{*T} x^\mu > 0, \forall mu \tag{15}$$

Let w^n be the 'student's' weight vector at the n-th time step. It forms an angle $\theta(n)$ with w^*, which we will denote $A(n)$:

$$c(n) \equiv \cos\theta(n) = \frac{w^{nT} w^*}{\|w^n\| \, \|w^*\|} \tag{16}$$

The basic idea behind the proof is that if the algorithm were to go on forever the RHS of 16 is become greater than 1 for large values of n. Think about the dynamics of w as mimicking some sort of random walk, where some of the steps move it in the w^* directions, some away from it. At the same time, its norm is also changing. If it was indeed a truly unbiased random walk then the norm (denominator) would grow typically as $\sqrt{n}$. The numerator will have a fluctuating sign but by central limit theorem, its typical values are around $\sqrt{n}$.

However, as we will show, w^n makes a baised random walk during learning, in the direction of direction of w^* , hence the numerator grows linearly with n. On the other hand, because the change in w^n is biased away from the current weight vector w^{n-1},the term $\|w^n\|$ in the denominator grows at most only as $\sqrt{n}$. Since the RHS of 16 cannot be larger than 1, n cannot be large.

We now proceed to the formal proof.

Margins: An important quantity is

$$\delta^\mu = \frac{y_0^\mu w^{*T} x^\mu}{\|w^*\|} \tag{17}$$

δ^μ is the (signed) Euclidean distance of the point x^μ from the plane w^*. In modern learning theory it is called *the margin* of this example relative to the separating plane, w^*. Note that δ^μ is is strictly positive since all points are correctly classified, hence we can define

$$\delta^* = \min_\mu \delta^\mu > 0 \tag{18}$$

which is the minimal margin relative to the separation hyperplane, w^*. [Note that in the definition of the margin we could also divide by the norm of x^μ since only the angle between x^μ and w matters].

For simplicity we will count a learning step only those in which the weights are updated. At the n$^{\text{th}}$update step we have

$$w^n - w^{n-1} \equiv \Delta w^n = \eta y_0^n x^n \tag{19}$$

If there is an actual update at this step then there was an error, i.e. $y_0^n w^{n-1T} x^n < 0$, and this has the consequence that

$$w^{n-1T} \Delta w^n < 0 \tag{20}$$

In words, if a correction was made then the projection of w^{n-1}on the corrected vector will be negative. The two key inequalities are Eqs. 18 and 20.

We will now investigate **the numerator** of $c(n)$:

$$w^{nT} w^* = w^{n-1T} w^* + \Delta w^{nT} w^* \tag{21}$$

$$= w^{n-1T} w^* + \eta y_0^n w^{*T} x^n = w^{n-1T} w^* + \eta \delta^\mu \left\| w^* \right\| \tag{22}$$

$$w^{nT} w^* \geq w^{n-1T} w^* + \eta \delta^* \left\| w^* \right\| \tag{23}$$

since $\delta^* \leq \delta^\mu$. We can then apply this inequality n times, starting from w^0, to get

$$w^{nT} w^* \geq w^{0T} w^* + n\eta\delta^* \left\| w^* \right\| \tag{24}$$

Hence for large values of n we obtain,

$$W^{nT} W^* \geq \eta n \delta^* \left\| W^* \right\| \tag{25}$$

Examining **the denominator** (just the $\|w^n\|$ component, because $\|w^*\|$doesn't change), we obtain,

$$\begin{aligned} \left\| w^n \right\|^2 &= \left\| w^{n-1} + \Delta w^n \right\|^2 & (26) \\ &= \left\| w^{n-1} \right\|^2 + \eta^2 \left\| x^n \right\|^2 + 2(w^{n-1T} \Delta w^n) & (27) \\ &\leq \left\| w^{n-1} \right\|^2 + \eta^2 D^2 & (28) \end{aligned}$$

where $D \equiv max_\mu ||x^\mu||$. Applying the inequality n times, we get

$$\left\| w^n \right\|^2 \leq \left\| w^0 \right\|^2 + n\eta^2 D^2. \tag{29}$$

Thus for large n,

$$\left\| w^n \right\| \leq \sqrt{n}\eta D \tag{30}$$

Putting it all together: for large enough n,

$$c(n) \geq \frac{\eta n \delta^* \left\| w^* \right\|}{\sqrt{n}\eta D \left\| w^* \right\|} = \frac{\delta^*}{D}\sqrt{n} \tag{31}$$

Thus if $n \to \infty$ then $A(n)$ will become larger than one, and we arrive at a contradiction since $c(n) = cos(\theta)$. thus proving that after a finite number of corrections the algorithm will halt.

3.1 Convergence time of the PLA

From the convergence proof we can also derive a bound on the convergence of the PLA. Let us define

$$\delta_{max} = \arg\max_{w^*} \delta^* \tag{32}$$

Then,

$$n \leq \frac{D^2}{\delta_{max}^2} \tag{33}$$

which highlights the important role of the margin (or rather than maximum margin) in the performance of the perceptron.

Indeed, the convergence time of the PLA can be in worst cases can be extremely long since there might be two examples with opposite lables but very close to each othe which will make δ_{max}very small. However, the typical behavior can be evaluated from the following scenario. Imagine the inputs are generated from a gaussian distribution where each component is $\langle x_i^\mu \rangle = 0$ and variance $1/N$. so that the norm of the inputs is 1 on average. Then, the probaility of a random example to have a small (relative to σ) margin δ^μ is roughly $\delta^\mu/\sigma = \delta^\mu\sqrt{N}$. Thus, the probability that there will be one example with margin δ is $Prob(\delta) \approx P\delta\sqrt{N}$. So the minimal margin is given by the condition $Prob(\delta) = 1$, from which we obtain

$$\delta \approx \frac{1}{P\sqrt{N}},\ n \approx P^2 N \tag{34}$$