Lecture 8, Perceptorn II

Haim Sompolinsky, MCB 131, 18 February, 2010

1 The Capacity of the Perceptron

Cover's Function Counting Theorem (Cover 1965):

Theorem: Let $\left\{x_1, ..\vec{x}^P\right\}$ be vectors in R^N, that are in general position. Then the number of possible dichotomies that can defined on them by means of a hyperplane passing through the origin is:

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

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

Note that if $P \leq N$ then it is natural to expect the vectors to not be linearly dependent – which means that they are in general position and Cover's Theorem holds for them. However, if $P > N$ then the input vectors must be linearly dependent. Even so, Cover's theorem holds if they fulfill the weaker demand that they be in general position, which is obeyed by all generic generators of inpu vectors. Thus Cover's theorem is widely applicable.

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

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

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

2. for $P = 2N$,

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

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.

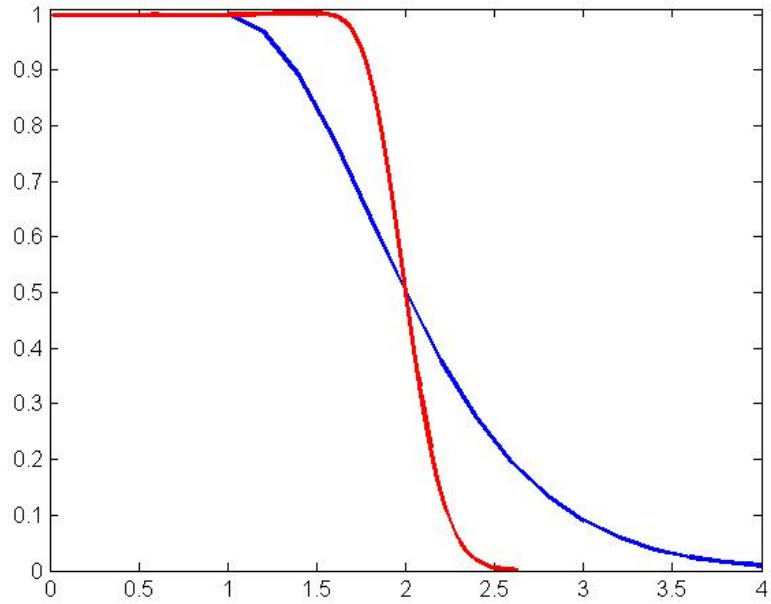

Figure 1 : *Fraction of linearly separable dichotomies vs.* P/N *for* $N = 5$ *and* 65

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.

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 the network 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 this W, we get a dichotomy over the set of $P + 1$ inputs:

$$\big(sign(Wx^1), ..., sign(Wx^{P+1})\big) = (b_1, ..., b_P, sign(Wx^{P+1})) \tag{4}$$

Thus for every dichotomy over P points there is a 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, we have seen that $(b_1, ..., b_P, sign(Wx^{P+1}))$ is possible (for every $(b_1, ..., b_P)$). Perhaps the additional dichotomy $(b_1, ..., b_P, -sign(Wx^{P+1}))$ (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

$$C(P + 1, N) = C(P, N) + D \tag{5}$$

Our goal is to find D, the number of additional dichotomies, by with we will find a recursive formula for $C(P, N)$.

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 (i.e. in the weights W which define 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).

Figure 2 : Geometry of Cover's Theorem

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 dichotomoy, 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, of course). 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

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

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,

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

$$= 2 \sum_{k=0}^{N-1} \binom{P-1}{k} + 2 \sum_{k=0}^{N-1} \binom{P-1}{k-1} = 2 \sum_{k=0}^{N-1} \binom{P}{k} \tag{7}$$

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

2 The Perceptron Learning Algorithm

Our goal now is to train a perceptron in order to correctly classify a set of points. In other words, given $\{x^1, ..., x^P\}$, and a set of correct labels $\{y_0^1, ..., y_0^P\}$, we want to find a set of weights W so that $sign(W^T x^\mu) = y_0^\mu$ for every i. This is of course not possible if the data is not linearly separable, so we will assume that it is, i.e. there exists a set of weights W^* so that $sign(W^{*T} x^\mu) = y_0^\mu$. We now need an actual algorithm to find a set of weights that correctly separates the data (which may not be equal to W^*, if there are many hyperplanes that all correctly separate the data).

The learning algorithm for the Perceptron is incremental. It is assumed to begin with an arbitrary set of weights W^0. At the n^{th} step we begin with the weights arrived at in the previous step, W^{n-1}, and a current example (x^n, y_0^n). The examples are presented one by one, in turn. Once we have passed through the

entire set of examples, we will start again from the first. Thus the examples will be shown in the following order: $1, 2, ..., P, 1, 2, ..., P, 1, 2, ...$

If $y_0^n(W^{n-1T}x^n) > 0$, then the algorithm classifies the new example correctly. Set $W^n = W^{n-1}$and move to the next step.

If $y_0^n(W^{n-1T}x^n) < 0$, then there is a mistake in the classification and $\vec{W}$ is adjusted in the following manner: $W^n = W^{n-1} + \eta y_0^n x^n$

The change in the weights can be written as:

$$W^n = W^{n-1} + \eta y_0^n x^n \Theta(-y_0^n W^{n-1T} x^n) \tag{8}$$

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

As mentioned before, the training examples are shown one by one and repeated from the beginning when we finish with the Pth example. Every time a complete cycle is finished from the first til last example, we count the number of errors during that cycle (and hence the number of adjustments to W). If there was no error, then the algorithm halts, and all the training examples are classified correctly. Thankfully, we have the following theorem, that guarantees that the algorithm works if the data is linearly separable:

Perceptron Convergence Theorem:

If P examples are given that are linearly separable, then the Perceptron Learning Algorithm will halt after a finite number of iterations to a vector W of connections that will divide the examples without errors, i.e. $sign(W^T x^\mu) = y_0^\mu$ for every μ.

Note: it is true that the number of iterations is finite, but it is usually larger than P because each example needs to be processed more than once.

Proof:

We note by W^* a weight vector that correctly separates the training examples: $\forall \mu, y_0^\mu = sign(W^{*T} x^\mu)$. 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 $A(n)$:

$$A(n) \equiv \cos\theta(n) = \frac{W^{nT} W^*}{\|W^n\| \, \|W^*\|} \tag{9}$$

The basic idea behind the proof is that if the algorithm were to go on forever the RHS of 9 is greater than 1 for large values of n. This is because W^n makes a baised random walk during learning. Because the bias is in the direction of W^* 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 only as $\sqrt{n}$. Since the RHS of 9 cannot be larger than 1, n cannot be large. We now proceed to the formal proof.

Define

$$\delta^\mu = \frac{y_0^\mu W^{*T} x^\mu}{\|W^*\|} \tag{10}$$

δ^μ is the (signed) Euclidean distance of the point x^μ from the plane W^*. It is nowadays called the margin of this example from the separating plane. Note that the numerator is positive since all point is correctly classified, and thus δ^μ is positive. Define

$$\delta^* = \min_\mu \delta^\mu > 0 \tag{11}$$

this signifies the minimal margin relative to the separation hyperplane, W^*. Another property of the training set, is $D \equiv max_\mu ||x^\mu||$, the size of the largest training example.

We will consider all time steps where an update occurs, so that the weights are updated. At the n$^{\text{th}}$update step we have $W^n - W^{n-1} \equiv \Delta W^n = \eta y_0^n x^n$ by the definition of the algorithm. Now, if there is an actual update then there was an error, i.e. $y_0^n W^{n-1T} x^n < 0$, and this has the consequence that

$$W^{n-1T} \Delta W^n < 0 \tag{12}$$

In words, if a correction was made then the projection of W^{n-1}on the corrected vector will be negative. We will use the two inequalities above.

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

$$\begin{aligned} W^{nT} W^* &= W^{n-1T} W^* + \Delta W^{nT} W^* \\ &= W^{n-1T} W^* + \eta y_0^n W^{*T} x^n = W^{n-1T} W^* + \eta \delta^\mu \|W^*\| \end{aligned}$$

$$W^{nT} W^* \geq W^{n-1T} W^* + \eta \delta^* \|W^*\| \tag{13}$$

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

$$W^{nT} W^* \geq W^{0T} W^* + n\eta \delta^* \|W^*\| \tag{14}$$

Hence for large values of n we obtain,

$$W^{nT} W^* \geq \eta n \delta^* \|W^*\| \tag{15}$$

Examining **the denominator** (just the $\|W^n\|$ component, because $\|W^*\|$doesn't change), we can see that:

$$\|W^n\|^2 = \left\|W^{n-1} + \Delta W^n\right\|^2 \tag{16}$$

$$= \left\|W^{n-1}\right\|^2 + \eta^2 \|x^n\|^2 + 2(W^{n-1T} \Delta W^n) \tag{17}$$

$$\leq \left\|W^{n-1}\right\|^2 + \eta^2 D^2 \tag{18}$$

where $D \equiv max_\mu ||x^\mu||$ as defined before, and $W^{n-1T}\Delta W^n < 0$ was shown earlier. Applying the inequality n times, we get

$$\|W^n\|^2 \leq \left\|W^0\right\|^2 + n\eta^2 D^2. \tag{19}$$

Thus for large n,

$$\|W^n\| \leq \sqrt{n}\eta D \tag{20}$$

Putting it all together: for large enough n,

$$A(n) \equiv \frac{W^{nT}W^*}{\|W^n\|\,\|W^*\|} \geq \frac{\eta n\delta^*\,\|W^*\|}{\sqrt{n}\eta D\,\|W^*\|} = \frac{\delta^*}{D}\sqrt{n} \tag{21}$$

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

Section 3 was not discussed in class.

3 *More Realistic Perceptron Models

Actual neurons have a threshold, below which they do not respond, and above which they respond in a uniform manner. Recall that we stated that the threshold parameter θ could be dealt with by moving from $(w_1, ..., w_N)$ to $(w_1, ..., w_N, \theta)$, while also changing the inputs from $(x_1, ..., x_N)$ to $(x_1, ..., x_N, -1)$. The perceptron convergence theorem applies to this problem: we add a -1 to the end of each training example, and learn $N + 1$ parameters normally.

However, in real neurons the threshold is an unchanging property; hence seeing it as a learnable parameter is problematic. What would happen if the threshold θ were fixed, instead of learnable, in the perceptron? As is obvious from the proof of the perceptron learning algorithm, $||\vec{W}^n||$ rises to infinity. Thus in the sum $\vec{W}^n\vec{x} - \theta$, the threshold becomes neglible. Learning with a threshold becomes the same as learning without one (although it may take longer in order for the weight vector to become large enough).

This leads us to a further objection, that allowing the synaptic weights to grow arbitrarily is also unrealistic: in real neurons, there are limitations on the efficacy of synapses. Furthermore, we do not just allow the weights to grow without limit - this is the normal behavior of the perceptron learning algorithm.

Hence we would like to consider a more realistic model, solving the two problems just raised: (1) the model has a fixed threshold θ, and (2) synaptic weights are limited in size. (Note that once synaptic weights are bounded, the threshold becomes relevant, since we cannot just raise $\vec{W}$ in order to marginalize it, as

mentioned before.) A further requirement for the sake of realism that we will consider is (3) allowing synapses to be excitatory or inhibitory (i.e. learning cannot alternate a synapse between these two).

Senn and Fusi (2004) consider such a model for a binary perceptron returning one of the values $0, 1$ (this is more convenient for a model in which synapses are either excitatory or inhibitory). Synaptic weights are assumed in the range $W_i \in [0, 1]$. All learnable synapses are excitatory; a single (constant) inhibitory input $g_I > 0$ is also assumed, the strength of inhibition is proportional to the sum of all the excitatory inputs (implementable by having an inhibitory interneuron synapsed by all excitatory neurons). The model is given by

$$y = \Theta\left(\frac{1}{N}\vec{W}^t\vec{x} - \frac{1}{N}g_I(1, ..., 1)\vec{x} - \theta\right) = \Theta\left(\frac{1}{N}\left(\vec{W} - g_I(1, ..., 1)\right)^t\vec{x} - \theta\right)$$

(the factor $\frac{1}{N}$ allows us, for convenience, to keep the weights W_i in the range [0,1] and not $[0, \frac{1}{N}]$). Note that, effectively, the synaptics weights are $W_i - g_I$, which allows the implementation of 'negative synapses'.

Their learning rule is similar to the perceptron learning algorithm, but contains a mechanism to keep synaptic weights in the range [0,1]. As in the perceptron learning algorithm, we only update if there is an error. The update is then given by:

$$y_0^n = 1 \quad \rightarrow \quad W_i^{n+1} = W_i^n + \eta x_i^n(1 - W_i^n) \tag{22}$$

$$y_0^n = 0 \quad \rightarrow \quad W_i^{n+1} = W_i^n - \eta x_i^n W_i^n \tag{23}$$

Notice how synaptic updates become progressively smaller as we near the limits of the range $[0, 1]$.

Using this update rule, Senn and Fusi prove that any linearly separable set of patterns is learnable, given three conditions:

1. Global inhibition exists, i.e. $g_I > 0$.
2. The learning rate η is small enough.
3. The threshold θ is small enough.

Note that these conditions are, of course, not needed with the original perceptron learning algorithm. Their necessity stems from the features of the more realistic model: global inhibition is needed in order to simulate negative synapses (required, e.g., if a pattern $x_i \equiv 1$ should be classified by 0); the learning rate and threshold need to be small enough, because bounding the synapses leads us to possibly working with minute distinctions in the range we do allow.

Bibliography:

Senn, W. & Fusi, S. Learning only when necessary: better memories of correlated patterns in networks with bounded synapses. Submitted to Neural Computation.

4 Training vs. Generalization

1. The notion of generalization. The lack of sufficient data. The tradeoff between capacity and generalization. Within a given architecture we would like to find a learning alg that maximizes our chances for a good generalization.

2. In the perceptron context we would like to find out the best optimal separating hyperplane.

3. This problem is tied closely with the limitation of linear separability. We could always recode the inputs into a new high dimensional representation.

However, we will face two problems:

a. Aggrevate the problem of generalization since we increase the system capacity.

b. Make processing harder by need to perform may more computation.