Models of Cortical Maps

Lecture 18

Instructor: Anatoli Gorchetchnikov <anatoli@cns.bu.edu>
Competitive Learning

Second layer has neurons that respond to a combination of features in the first layer.

Competition between these nodes can ensure only one neuron as a winner (WTA) or a more distributed representation.

Only winning neurons update the weights.

Weights follow the presented input pattern:

\[ \dot{w}_{ij} = \eta x_i y_j - \alpha y_j w_{ij} \]

or

\[ \dot{w}_{ij} = (\eta x_i - \alpha w_{ij}) y_j \]
Competitive Learning

Competitive learning networks typically learn in an unsupervised fashion.

Specific examples include:

- von der Malsburg (1973) model of visual cortex
- Grossberg (1976) adaptive resonance theory (ART) and its precursors: instar and outstar
- Kohonen (1983) self-organizing feature maps (SOFM, SOM)
The RCF Network of Grossberg (1976)

\[ \frac{dy_j}{dt} = -Ay_j + \left( B - y_j \right) \left( f(y_j) + I_j \right) - \left( C + y_j \right) \sum_{k \neq j} f(y_k) \]

where \( I_j = \sum_i x_i w_{ij} \)

or \( \overrightarrow{x} \cdot \overrightarrow{w_j} = \| \overrightarrow{x} \| \| \overrightarrow{w_j} \| \cos \theta \)

So the winner will be the cell that has incoming weights closest to the current input pattern.

Note that when \( x \) and \( w \) are both normalized, \( \cos \theta \) is the only value that matters.
The RCF Network of *Grossberg* (1976)

\[
\frac{dy_j}{dt} = -Ay_j + (B - y_j) \left( f(y_j) + I_j \right) - \left( C + y_j \right) \sum_{k \neq j} f(y_k)
\]

Describes an unsupervised learning scheme that can be used to build “feature detectors” or perform categorization.

This paper presents many of the theoretical bases of adaptive resonance theory (ART)
Neural Feature Detectors

A neural feature detector is a (typically sensory) neuron that fires preferentially to inputs of a particular type. Orientation cells in visual cortex fire preferentially to lines of particular orientations:

Such a cell could be called a line orientation detector.

- The detector in the figure is “tuned” to vertical lines.

The pattern of firing as a function of some stimulus dimension (e.g., line orientation angle in the above figure) is called the “tuning curve”
Tuning vs. Categorizing

Although a feature detector cell is “tuned” to a particular orientation, it usually fires for other nearby orientations. Categorization, on the other hand, implies that a cell fires only for inputs in the category coded by the cell. This can be related to the difference between “choice” and “contrast enhancement” in an RCF:

In the competitive learning network of Grossberg (1976), choice leads to a categorization network, and contrast enhancement leads to feature detectors with broader tuning curves.
The Network of Grossberg (1976)

\[
\frac{dy_j}{dt} = -Ay_j + (B - y_j)(f(y_j) + I_j) - (C + y_j) \sum_{k \neq j} f(y_k)
\]

Network is a standard competitive learning network in which the F2 layer is a shunting recurrent competitive field (RCF).

The case where F2 is a “choice” RCF network (i.e., faster-than-linear signal function)

Hence, it looses the tuning curve bit
Consider what happens when an input pattern is presented to the F1 stage of the network.

Because the RCF at F2 is winner-take-all, the F2 cell with the largest input will win the competition.

All other cells will have their activities quenched.
The input to F2 cell will be a product of input pattern and the weight vector

\[ I_j = \sum_i x_i w_{ij} \]

In vector notation the input \( I_j \) to the F2 cell indexed by \( j \) is:

\[ \overrightarrow{x_i} \cdot \overrightarrow{w_j} = \|\overrightarrow{x_i}\| \|\overrightarrow{w_j}\| \cos \theta \]

Where \( \overrightarrow{x_i} \) is the vector of F1 cell activities, \( \overrightarrow{w_j} \) is the vector of weights projecting to F2 cell \( j \), and \( \theta \) is the angle between them.
Assume for now that the total activity at F1 is normalized, as are the weight vectors projecting to the F2 cells. In this case, the F2 cell with the most input will be the one whose afferent weight vector $\bar{w}_j$ is closest to parallel to the activity pattern at F1 $x_1$ and thus has the largest $\cos \theta$ term. For a 2-D input space (i.e., 2 F1 cells), the situation can be schematized as follows:
The Instar Learning Law

Grossberg (1976) studied the effects of using an “instar” learning law with Hebbian growth and post-synaptically gated decay in the F1 to F2 weights

\[ \dot{w}_{ij} = \eta x_i y_j - \alpha y_j w_{ij} \]

If F2 is winner-take-all, learning will only occur in the weight vector projecting to the F2 cell that won the competition.

This is competitive learning.

What happens to the afferent weight vector for the F2 cell that won the competition?

\[ \dot{w}_{ij} = \left( x_i - \frac{\alpha}{\eta} w_{ij} \right) \eta y_j \]
The weight vector becomes more like the current input vector.
That is, the afferent weight vector becomes more parallel to the current F1 STM pattern

This means that the winning F2 cell will become even more responsive to the current input pattern
Weight “Normalization” in Grossberg (1976)

If the F1 layer is a shunting competitive network with normalized activity, then the weight vector projecting to each F2 cell will be approximately normalized as well after learning has taken place

\[ \dot{w}_{ij} = \left( x_i - \frac{\alpha}{\eta} w_{ij} \right) \eta y_j \]

Thus, each component of the F2 cell’s weight vector tracks the corresponding component of the input vector at F1, and since the input vector is normalized, the weight vector becomes approximately normalized as well
“Coding” of the Input Space

The afferent weight vectors for the F2 cells span the input space.

When an input arrives at F1, the F2 cell whose weight vector is closest to parallel to the input vector will have the most input and win the competition at F2.

Thus, each F2 cell is tuned to (“codes”) a region of the input space roughly centered around its afferent weight vector:
We know that, for a single input pattern, the weight vector afferent to an F2 cell moves toward the input pattern vector. More general question: what happens to the network as a whole when presented with many input patterns? In particular, do the “features” (i.e., regions of input space) coded by the cells constantly change, or do they stabilize?

Grossberg (1976) noted that we cannot in general guarantee that:

– an initial partition of input space will be maintained through time, or
– a “stable” partition will ever form
Strange learning sequences might cause the constant recoding of F2 cell firing preferences:

Note that this picture is only true if there is only one winning cell
Sparse Patterns Theorem

However, we can guarantee that a partitioning into “sparse” classes (defined below) will persist: Sparse patterns theorem

Assume that you start out with a “sparse partition”; i.e., a partition such that the classifying weight vector for a given subset of input vectors is closer to all vectors in the subset than to any other vector from any other subset.
Then, with instar learning and a choice network at F2, you are guaranteed that the initial partitions will not be recoded, and the weight vectors will eventually enter the convex hull of the subset of input patterns that it codes.
Proof Of Convergence to Convex Hull

Grossberg (1976) provides an analytical proof

We will look at a simpler and more general “geometric” proof here (Guenther, 1992)

Key concept:
For an acute angle $ABC$, as $x$ is moved from $B$ toward $C$ along line segment $BC$, the distance between $A$ and $x$ initially decreases

This concept is applied twice in the following proof
To prove that the weight vector \( z \) will enter the convex hull of training inputs, we can show that the closest point on the convex hull, \( c_{\min} \), keeps getting closer to \( z \) during learning.

Consider the hyperplane \( H \) passing through \( c_{\min} \) and perpendicular to line segment \( zc_{\min} \).
First, note that all points in the convex hull must lie on $H$ or on the side of $H$ opposite $z$.

To see that this is the case, note that any point $c$ in the convex hull that might lie on the same side of $H$ as $z$ would form an acute angle $z c_{\min} c$, which means that some points between $c_{\min}$ and $c$ would be closer to $z$, thus violating the definition of $c_{\min}$.

Thus all training inputs lie on the side of $H$ opposite $z$ or on $H$ itself.
This means that the angle $c_{\min}z_c$ must be acute.

Thus incrementing the weight vector $z$ toward any training input $c$ from the convex hull will decrease the distance between $z$ and $c_{\min}$.

I.e., the weight vector constantly moves closer to the convex hull, and once in the hull it can never leave it.

More generally, constantly moving toward arbitrary points in a convex hull eventually leads to the convex hull.
Beyond Sparse Patterns

Similar to the instar case, when an outstar is presented with an equivalence class of border patterns that form a convex hull in some input space, the weight vector is guaranteed to enter the convex hull if enough training is applied.

In the general case, we cannot rely on inputs falling into sparse classes.

How, then, do we guarantee that cells in F2 won’t constantly be recoded?

This issue became the motivation for top-down feedback in ART, with top-down weights adjusted according to the outstar learning law.
The Model of *von der Malsburg* (1973)

This model was designed to explain data from single cell studies of visual cortex by *Hubel and Wiesel* (1962, 1963, 1968).

To do this, von der Malsburg showed that a self-organizing neural network utilizing a recurrent competitive field could develop cell properties that were very similar in certain aspects to those seen in visual cortex.

Furthermore, the network cells developed these properties simply by being exposed to stimulus bars of different orientations.

This network uses a different kind of RCF than what we have studied so far.
Question:
What is the simplest explanation for the retinotopic organization of visual cortex?
What about the topographic representation of line orientations in visual cortex?
Hubel and Wiesel originally proposed that genetic predetermination was the mechanism that led to the V1 cell properties they observed. Von der Malsburg pointed out that there are several problems with this view:

- An immense amount of genetic information would be required to tell all the branches of afferent axons which cortical neuron(s) they should contact.
- A rigid, genetically programmed circuit would not possess the plasticity seen in cortical cells (e.g., early visual experience drastically affects orientational selectivity).
- Self-organization is needed by higher-level brain areas that must deal with situations not foreseen by nature.

Therefore, von der Malsburg proposed a self-organizing neural network that explains the orientational selectivity seen in V1.
von der Malsburg’s Model

The model consists of two layers of neurons meant to correspond roughly to retina and visual cortex.

The cortical level is an on-center, off-surround recurrent competitive field, and the synapses between the retinal level and the cortical level are adaptive.

How is this different from the actual visual system circuitry? How is this different from Grossberg (1976)?
The Retinal Level

The “stimuli” presented to the network consist of patterns of activity across the retinal cells. There are 19 retinal cells in a hexagonal array:

Retinal cell activities are “all or nothing”, specified by large and small dots, respectively. This diagram shows a stimulus corresponding to a vertical bar.
Training Stimuli

Nine different input patterns were repeatedly presented, corresponding to nine different bar orientations on the retina.

The question of interest is what happens to the cortical level cell selectivities with repeated presentation of the stimuli.

Fig. 5. The standard set of stimuli used on the model "retina". Large and small dots represent active and non-active fibres respectively.
The Cortical Level

The competitive field (‘‘cortex’’) consists of 169 E-cells (excitatory) and 169 I-cells (inhibitory) in hexagonal arrays. Each dot in the array corresponds to one E-cell and one I-cell.

In the *von der Malsburg* figures, active (suprathreshold) E-cells are indicated by large dots, subthreshold cells by small dots. Activity of the I-cells is not indicated.
Each cortical E-cell receives inputs through adaptive weights from all of the retinal level cells.

The weights start out random; these are the only adaptive weights between network layers.

Cortical I-cells do not receive any inputs from retinal cells.

Question: What does this imply about the receptive fields of the network in terms of retinal locations?
Cortical Cell Interactions

The model contained three types of horizontal interactions between cortical cells: E-cell to E-cell, E-cell to I-cell, and I-cell to E-cell.

I-cells do not interact with each other.

All connections within the cortical level are non-adaptive (fixed weight).

(1) E-cell to E-cell connections \( (p_{ij}) \) –

E-cells excite their immediate neighbors in the array.
(2) E-cell to I-cell connections \((r_{ij})\) –

E-cells excite the corresponding I-cell and its immediate neighbors in the array

(3) I-cell to E-cell connections \((q_{ij})\) –

I-cells inhibit the next-to-immediate neighbor E-cells

Question: What is this connectivity meant to do?
These connections are meant to approximate a smoothly varying on-center, off-surround competitive network amongst the E-cells.
Cell Equations

The model has separate cell equations for the E-cells and I-cells due to the different connectivity of the two cell types (recall that I-cells receive no retinal input and no input from other I-cells).

The cell equations for the cortical cells are:

\[
\dot{E}_i = -aE_i + \sum_j p_{ji} f(E_j) + \sum_k s_{ki} R_k - \sum_l q_{li} f(I_l) \\
\dot{I}_i = -aI_i + \sum_j r_{ji} f(E_j)
\]

Note that these are additive (no implicit normalization)
Weight Change Equations

Learning on each trial is done in two steps:

1) Hebbian weight changes:
\[ s_{ij}^*(t + 1) = s_{ij}(t) + hR_i E_j \]

Where \( h \) is a learning rate parameter.

Question: What differential equation does this equation approximate?

2) Weight normalization:
\[ s_{ij}(t) = \frac{s_{ij}^*(t)}{\sum_i s_{ij}^*(t)} \]

This step can be thought of as maintaining a conservation of synaptic strength projecting to a cortical E-cell.

Question: Why is this step necessary?
Weight normalization is needed to keep the system stable. Two properties of the network would contribute to instability without weight normalization:

– The model’s Hebbian learning law favors weight increases, and
– The use of additive cell equations at the cortical level means that increasing the total cortical input will increase the total activity of the map.

Increasing the weights increases the total input to cortex which leads to larger cortical activity, which leads to even bigger weights, etc.
Is weight normalization biologically plausible?
Possible. For example, increasing the efficacy of some synapses could lead to decreases in efficacy of others by increasing the cell’s firing threshold.
Recall also the *Levy and Desmond* learning rules from an earlier lecture – physiological changes appear to limit the total synaptic strength of incoming weights to a postsynaptic cell.
Network Output Before Learning

Response of the cortical E-cells to each of the 9 stimulus patterns prior to any training (von der Malsburg, Figure 6).

Note tendency for active cells (large dots) to exist in clusters

Why is this?
Figure at right shows the network after 100 trials (i.e., 100 applications of all 9 training stimuli)

Now there is a much stronger tendency for clustering of active cells

Why do cells tend to cluster more after training than before training?
Prior to training, two somewhat opposing forces are acting on the cortical E-cells:

(1) Tendency to fire in clusters due to on-center, off-surround network structure
(2) Tendency to fire in random pattern due to random bottom-up inputs (recall that weights start out random)

After training, however, the randomness of the weights is replaced by weights that favor clustering
Therefore, both forces act to make the cells fire in clusters
Why do the weights favor clustering after training?
Need to look at the weight vectors…
Cortical E-Cell Response Curves

Bottom row indicates weight sizes, middle row indicates total bottom-up input to cell for each of the 9 stimuli, and top row indicates cell activity in steady state for a typical cortical E-cell.

Two things to note after 100 learning trials (right column):

- Strong preference for one orientation,
- Receptive field shape similar in some respects to those seen in V1 cells in vivo.

![Diagram showing receptive field organization, afferent excitation, and ES of the E-cell No. 70.](image)

Fig. 8. Receptive field organization, afferent excitation and ES of the E-cell No. 70. Its position on the cortex is in the seventh line top (see Fig. 6.1). For explanation see Fig. 7.
The “Preferred Orientation” of All Cortical E-cells

After only 20 trials, the network already shows some aspects of cortical cell firing.

At this stage there are still many cells that did not respond to any of the stimuli (no orientational preference).

Fig. 12. View onto the cortex. Each bar indicates the optimal orientation of the E-cell (for definition see text). Dots without a bar are cells which never reacted to the standard set of stimuli. Two bars indicate two separate sensitive regions.
After 100 trials, almost all cells are orientation-sensitive, and neighboring cells tend to prefer similar orientations.

These are the main aspects of the visual cortical data of *Hubel and Wiesel* (1962, 1963, 1968) that von der Malsburg set out to explain.

Fig. 13. View onto the cortex after 100 steps of learning.
Next Time

Kohonen’s SOM as yet another model of cortical maps
Comparison of the three models
Cohen-Grossberg theorem, its implications and applicability

Readings: