Learning spatial invariance with the trace rule in non-uniform distributions

Jasmin Leveillé\textsuperscript{1} and Thomas Hannagan\textsuperscript{2}

\textsuperscript{1}Center for Adaptive Systems
Department of Cognitive and Neural Systems
Boston University
677 Beacon Street, Boston, MA, 02215, USA

\textsuperscript{2}Laboratoire de Psychologie Cognitive, CNRS
- Aix-Marseille University
3, place Victor Hugo, 13331 Marseille, France

Keywords: Visual recognition, Location invariance, Trace rule, Convolutional networks

Abstract

Convolutional models of object recognition achieve invariance to spatial transformations largely because of the use of a suitably defined pooling operator. This operator typically takes the form of a max or average function defined across units tuned to the same feature. As a model of the brain’s ventral pathway, where computations are carried out via weighted synaptic connections, such pooling can only lead to spatial invariance if the weights that connect similarly tuned units to a given pooling unit are of approximately equal strengths. How identical weights
can be learned in the face of non-uniformly distributed data remains unclear. In this article, we show how various versions of the “trace learning rule” can help solve this problem. This allows us in turn to explain previously published results and to make recommendations as to the optimal rule for invariance learning.

1 Introduction

In computer vision and computational neuroscience alike, state-of-the-art models of object recognition achieve invariance by constraining the learning to occur on a network architecture that implements a succession of feature detection and pooling stages (LeCun et al., 1998; Serre et al., 2007; Pinto et al., 2011; Cadieu et al., 2007). The pooling stages use max or average operators that pool across units tuned to the same feature but differing by some transformation. All models postulate in some form or another the presence of identical weights between tuned cells and pooling cells. It is sufficiently obvious why implementing the average operator should require identical weights (see for instance Boureau et al. (2008)), but this is less straightforward in the case of the max operator. Models that use a max operator (Fukushima, 1980; Riesenhuber & Poggio, 1999; Serre et al., 2007; Huang et al., 2011; Masquelier et al., 2007) implicitly assume weights that tend to be identical between tuned units and pooling units, since the implementation of a max or softmax function requires either an unweighted pooling operation, or an unweighted normalization, which are functionally equivalent to weighted computations with identical weights. The max operator could also be implemented as a two-stage process consisting of local winner-take-all competition among neighboring tuned cells - somewhat as proposed in Yu et al. (2002) - followed by a weighted projection to a pooling unit via a weight vector composed of identical coefficients.1

Because cells in the human visual system are initially not tuned to complex features and synapses are thought to be heterogeneously weighted, it becomes

1Implementations of the max operator using spiking networks assume instead identical membrane conductances from the tuned cells to pooling cells (Knoblich et al., 2007).
necessary to explain how these hypothesized identical pooling weights could develop, an issue that is bypassed in convolutional networks and appears to have been otherwise widely ignored. Some recent computational studies have provided empirical demonstrations that in a non-convolutional, feedforward network, the supervised backpropagation algorithm indeed learns to achieve location invariance by inducing identical weights for similarly tuned units at different locations (Hannagan et al., 2011). In general however, it is unclear how identical weights could be learned, especially in an unsupervised setting and considering that feature detector units are likely to be non-uniformly active.

Fig. 1 illustrates this last point. Gabor filters in four orientations were applied to up to 2000 frames of a natural video sequence (Betsch et al., 2004). The frequency at which a given unit had the highest response among a randomly chosen 2x2 spatial neighborhood of units was counted over a predefined time interval (varying between 4 and 76 seconds). The selection of the maximal response is meant here to approximate the effect of a winner-take-all mechanism. Normalizing the frequency across units of the same orientation but at different locations over the 2x2 spatial neighborhood yields a probability that a given unit is maximally active given that its orientation is the winning one. Computing the standard deviation $\sigma$ between the probabilities hence obtained across units of the same orientation gives a simple measure of uniformity of activity for each orientation separately. Note that if all units within the pool were active with the same probability, $\sigma$ would be equal to 0. As Fig. 1 clearly shows, even over relatively long periods of time (> 68 seconds), $\sigma$ is greater than zero, implying that units are unevenly activated.

Figure 1 thus provides good evidence that early competitive processes in the visual system will produce non-uniform feature distributions when subjected to natural input statistics. In this case, the usual Hebbian learning rule with weight normalization fails to produce uniform weights as it converges instead to the first eigenvector of the data covariance, at least when inputs are centered around zero (Oja, 1982) (see also Figs. 3 and 4).
Figure 1: **Non-uniformity in natural movies.** Each curve denotes the inhomogeneity - measured as the standard deviation $\sigma$ - in the response distribution of an array of Gabor feature detectors tuned to one of four orientations and located in a 2x2 spatial neighborhood. Activations were computed in response to up to 2000 frames of a natural video sequence collected at 25Hz (Betsch et al., 2004), for a total of 10 repetitions. The fact that $\sigma > 0$ indicates that despite being tuned to the same feature, spatially offset units are clearly unevenly activated, especially over short time spans. Error bars represent standard error.
There is, however, compelling empirical evidence that unsupervised learning of spatially invariant operators is possible using a temporal variation of Hebbian learning. This so-called “Trace rule” operates for instance in the Visnet model (Wallis & Rolls, 1997), where it is thought to extract temporal correlations during training, somehow allowing units to display increasing invariance properties as one progresses up the hierarchy. Although the Trace rule has been shown in simulations to lead to invariance for location, view and rotation, its properties are not at all understood. Particularly puzzling are the VisNet simulations of Rolls & Stringer (2001), which show that different variants of the Trace rule lead to various levels of invariance as measured by information-theoretic means.

This paper extends the results of Rolls & Stringer (2001) to the case of more plausible, non-uniform input distributions. Our main contribution is to explain the behaviour of the trace rule within a principled formalism. We show that the Trace rule achieves invariance by uniformizing the weights coming from similarly tuned units in the same pooling neighborhood. We therefore provide a principled account of how pooling may be learned in the brain and offer guidelines for improving invariance learning in computer simulations.

2 New results on the Trace rule

Let $x_t$ be an $n$-dimensional vector representing the activity at time $t$ of a pool of tuned units projecting to the pooling cell $y$. The units that compose $x$ are usually referred to as simple cells, or S-cells (Fukushima, 1980). Cell $y$’s net input is computed as the standard inner product:

$$y_t = w^T x_t, \quad (1)$$

where $w$ is a synaptic weight vector and $T$ denotes the transpose operator. The trace of cell $y$ is usually defined as the low-pass filter:

$$\bar{y}_t = (1 - \eta)y_t + \eta \bar{y}_{t-1}, \quad (2)$$
where \( \eta \) is a value between 0 and 1 (typically 0.8 in VisNet simulations Rolls \& Stringer (2001)). The trace learning rule is in turn defined as a Hebbian update step using the trace instead of the net input:

\[
\Delta w = \lambda \bar{y} x_t,
\]

where \( \lambda \) is a small learning rate. To avoid weights from blowing up, the learning step is usually followed by a normalization step. Although initially introduced in the context of reinforcement learning (Sutton \& Barto, 1981), the trace rule was first used to learn spatial invariance from exposure to smoothly varying patterns in Foldiak (1991). The rationale behind this use of the trace rule is that the invariant components in the sensory input typically vary at a slower time-scale than neural events, hence higher-order learning should also make use of slowly varying neural signals. A more precise way to understand what the trace rule does is to insert Eq. 2 and Eq. 1 in Eq. 3, and to compute the expectation assuming a small learning rate \( \lambda \):

\[
E[\Delta w] = E[(1 - \eta)x_t x_t^T]w + \eta E[x_t \bar{y}_{t-1}]
\]

\[
\approx E[(1 - \eta)x_t x_t^T]w + \eta E[x_t E[x_t^T]w]
\]

\[
\approx E[(1 - \eta)x_t x_t^T]w + \eta E[x_t]E[x_t^T]w
\]

The first approximation is due to the fact that for small learning rates, \( E[y] \approx w^T E[x] \), and the second approximation to the fact that, over stationary signals, \( E[x_{t-1}] \approx E[x_t] \) (Cooper et al., 2004; Sprekeler et al., 2007). The main point of the formulation in Eq. 4 is to disentangle the effects on learning of the instantaneous net input and of slow-varying signals. Despite that the approximations remove any explicit dependence on temporal order, Eq. 4 can still be used to understand trace rule learning - at least in competitive networks such as Visnet and the networks used in the simulations reported here - based on the following argument. First, it should be noted that, in such competitive networks, a postsynaptic unit’s weights changes significantly only if that unit wins the competition among its neighbors over a few time-steps, during which time the trace keeps receiving positive input \((1 - \eta)y_t\). Provided that inputs change smoothly, there will be a transient period
of time during which the winning unit is active and its trace leads to a change in activity. The intuition behind the approximation in Eq. 4 is thus to look at the distribution of inputs over all such transient periods of activity only (cutting out the long periods where the unit is presented an input for which it is less selective than its neighbors and thus does not change its weights much). The implication would be that, for unit $y$ to develop translation invariance to an input pattern, this pattern must be presented with approximately equal probability at each of its presynaptic sites during these periods of transient activity. Eq. 4 also happens to be similar to the gradient that maximizes an objective function, introduced in Maurer (2006), to learn a combination of fast and slow features. Specifically, Maurer (2006) considers maximizing the objective:

$$L(P) = \frac{1}{2} \mathbb{E} \left[ (1 - \eta) \| PX_t \|^2 - \eta \| P\dot{X}_t \| \right]$$

where $\dot{X}_t = X_t - X_{t-1}$ and $P$ is the projection operator. In the case of a single linear neuron, $P$ corresponds to the neuron’s synaptic weight vector $w^T$, and Eq. 5 reduces to:

$$L(P) = \frac{1}{2} \mathbb{E} \left[ (1 - \eta) y_t^2 - \eta (y_t - y_{t-1})^2 \right]$$

Maximizing Eq. 6 can be accomplished in part by minimizing the second term. This, in turn, is similar to maximizing the variance of the low-pass filtered postsynaptic signal (Sprekeler et al., 2007):

$$L(P) \approx \frac{1}{2} \mathbb{E} \left[ (1 - \eta) y_t^2 + \eta (f \circ y)^2 \right]$$

where $\circ$ denotes the convolution operator. Filter $f$ in Eq. 7 needs to be a low-pass filter in order for slow features to be learned. Computing the stochastic gradient of Eq. 7 at time $t$, one obtains:

$$\frac{\partial L(P)_t}{\partial w} = (1 - \eta) y_t x_t + \eta (f \circ y) f \circ x$$

As shown in Sprekeler et al. (2007), a reduced form of slow feature learning is still possible by replacing the filter $f$ in the second convolution by the $\delta$ function, since the first convolution indirectly preserves some temporal structure in the input. Further changing $f \circ y$ to the trace rule notation $\tilde{y}_t$, substituting $w^T x_t$ for $y_t$ and taking the expectation leads to Eq. 4. Since the matrix $\mathbb{E}[(1 - \eta)x_t x_t^T] + \eta \mathbb{E}[x_t] \mathbb{E}[x_t]$
Table 1: Variants of the trace rule studied here. Each equation is labeled in the left column according to its corresponding number in Rolls & Stringer (2001). The corresponding equation is shown in the middle column, together with the corresponding value of $\alpha$ in Eq. 9.

<table>
<thead>
<tr>
<th>Equation</th>
<th>$\Delta w$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS1</td>
<td>$\bar{y}_t x_t$</td>
<td>0.2</td>
</tr>
<tr>
<td>RS4</td>
<td>$\bar{y}_{t-1} x_t$</td>
<td>0</td>
</tr>
<tr>
<td>RS8ii</td>
<td>$(\beta \bar{y}_t - y_t)x_t$ where $\beta = 2.5$</td>
<td>-0.33</td>
</tr>
<tr>
<td>RS10</td>
<td>$(\beta \bar{y}_{t-1} - y_t)x_t$ where $\beta = 4.9$</td>
<td>-0.25</td>
</tr>
</tbody>
</table>

is positive semidefinite, the weight vector in Eq. 4 evolves in the direction of its principal eigenvector. Recalling that $\mathbb{E}[x_t x_t^T] = \text{Cov}(x) + \mathbb{E}[x_t] \mathbb{E}[x_t^T]$, Eq. 4 can be further simplified as follows:

$$\mathbb{E}[\Delta w] \approx (1 - \eta) [\text{Cov}(x) + \mathbb{E}[x_t] \mathbb{E}[x_t^T]] w + \eta \mathbb{E}[x_t] \mathbb{E}[x_t^T] w$$

$$= (1 - \eta) \text{Cov}(x) w + (1 - \eta) \mathbb{E}[x_t] \mathbb{E}[x_t^T] w + \eta \mathbb{E}[x_t] \mathbb{E}[x_t^T] w$$

$$= \alpha \text{Cov}(x) w + \mathbb{E}[x] \mathbb{E}[x_t^T] w, \quad (9)$$

where $\alpha = (1 - \eta)$ here.

### 3 A reinterpretation of various trace rules

Here we show how the derivations above shed light on the results obtained by Rolls & Stringer (2001) with the four variants of the trace rule in Table 1. We use the fact that each of the four variants can be rearranged so as to fit Eq. 9 up to a multiplicative constant, from which it is possible to compare how they relate to the value of factor $\alpha$.

RS1 is the original trace rule (Eq. 3). RS4 is the trace rule using the trace from the previous time-step. RS8ii can be derived from RS4 and assuming an additional parameter $\beta$. RS10 is the same as RS8ii but using the trace from the previous time-step. Simulations in Rolls & Stringer (2001) showed that RS1 was noticeably worse at learning invariance than RS4 which was in turn slightly worse than either RS8ii or RS10. The non-negative factor $\alpha$ for RS1 and RS4 can be
easily interpreted as saying that the input covariance is learned less than the slow-varying features (Maurer, 2006). Why the negative values of $\alpha$, in the cases of RS8ii and RS10, would lead to improved learning is harder to explain. Learning should yield a weight vector $w$ that is parallel to the principal eigenvector of the positive semidefinite matrix $\alpha \text{Cov}(x) + \mathbb{E}[x_t]\mathbb{E}[x_t^T]$. Hence, a negative $\alpha$ would imply that $w$ is perturbed away from the principal eigenvector of the covariance matrix $\text{Cov}(x)$. This is unlike Hebbian learning, in which $w$ becomes parallel to the principal eigenvector of the covariance matrix. The results in the next section show that removing part of the input covariance is indeed a way to uniformize weights in the case of non-uniform data.

4 Using the trace rule on natural movies

Before showing this, however, simulations were conducted to extend the findings reported in Rolls & Stringer (2001) to natural movies. A two-layer network was used for that purpose in order to disentangle the contribution of the trace rules to invariance learning from the other particularities of the Visnet model architecture. The first layer consists of a spatial array of log-Gabor feature detectors tuned to four orientations $(0, \pi/4, \pi/2, 3\pi/4)$ and at a spatial frequency of 1.5 pixels per cycle. Competitive winner-take-all (WTA) was performed on both the input and output layers by simply detecting the maximally active unit. Upon winning a competition, input units emitted a transient pulse of magnitude 1 and remaining units were reset to 0. The input to the postsynaptic layer was thus similar to a spike train of 0’s and 1’s. Pooling was learned in the second layer, where each unit pooled the activity coming from all units within a $2 \times 2$ neighborhood by computing a dot-product of its 16-dimensional input vector ($4$ orientations $\times 2 \times 2$) with its weight vector. To speed up simulations, output units were simulated at a single spatial location (i.e. a single hypercolumn) chosen at random. The output layer consisted of a total of nine units. As in VisNet simulations, all simulations reported here were conducted with $\eta = 0.8$, the learning rate was set to $\lambda = 0.01$, and the L2-norm was used to normalize the weights to 1 at each iteration. Learning was done with each of the four learning rules in Table 1. Video frames at a
Figure 2: Learning uniform weights from natural video. Results are shown in the following order: (a) Hebb’s rule, (b) RS1, (c) RS4, (d) RS8ii and (e) RS10. As noted in the introduction, Hebb’s rule leads to the most uneven weights. RS4-RS10 are clearly superior to RS1. RS8ii in particular appears mildly superior to RS4.

resolution of 320x240 were extracted from the data of Betsch et al. (2004) was used for these simulations. After learning (for approximately 100,000 iterations), for each postsynaptic unit, the standard deviation coefficient \( \sigma \) was computed across weights for the orientation that had in average the highest weights (to ensure that invariance was measured with respect to the feature to which the pooling had learned selectivity). Weights with the lowest variance are necessarily closest to being uniform. For comparison purposes, learning experiments were also conducted with Hebb’s rule. The distribution of standard deviations hence obtained over 10 repetitions are shown in Fig. 2 for each learning rule.

Except for the fact that RS10 is not clearly better than RS4, the results in Fig. 2 are consistent with those of Rolls & Stringer (2001). These results alone are
encouraging as they replicate, on natural video, the pattern of findings obtained in VisNet simulations which have been performed so far only on simplified inputs. Nevertheless, it is still difficult from natural video simulations to study how non-uniformity in the data distribution affects invariance learning.

5 Controlled non-uniform input distributions

In order to focus on that specific issue, additional simulations were conducted using a specially handcrafted data distribution. Let \( x_t \) be the response of a presynaptic winner-take-all population at time \( t \) and of dimension \( n \). By virtue of WTA dynamics, all presynaptic sites are zero except for one unit, whose index \( k \) in the population is determined by an exponential distribution:

\[
    f(k; \rho) = \frac{1}{\rho} e^{-\frac{1}{\rho}k},
\]

for \( 1 \leq k \leq n \). The degree of uniformity of the distribution can be conveniently controlled by varying the rate parameter \( \rho \): assuming a fixed \( n \), a smaller \( \rho \) means that presynaptic units are active in roughly the same proportion. Learning experiments with this controlled data were performed with a single pooling unit, assuming again a \( 2 \times 2 \) pooling neighborhood and 4 orientations. Presynaptic units sensitive to one specific orientation, but at different spatial locations, were activated according to Eq. 10. Units representing other orientations at all spatial locations were randomly activated with a 10 percent probability, to ensure some robustness to noise in otherwise tightly controlled simulations. This simulation protocol is meant to approximate the repeated presentation of smoothly varying sequences of similarly oriented inputs, which occur in real video (Betsch et al., 2004), to which the pooling unit has already developed some selectivity. Simulation experiments of 8000 iterations each were repeated 25 times. The standard deviation computed on the synaptic weights for the subset of units in the selected orientation are displayed in Fig. 3 for varying levels of \( 1/\rho \).

Clearly, all versions of the trace rule reduce variability in the synaptic weights compared to Hebb's rule. For moderately non-uniform data, RS1 is noticeably weaker than the other trace variants. Rules RS8ii and RS10 also appear relatively
Figure 3: Effect of increasing uniformity on learning. Controlled simulations again confirm that RS1 is the least efficient trace rule compared to the other trace rules. These results are also compatible with those of Rolls & Stringer (2001) that RS8ii and RS10 are better than RS4. Note that this is true in particular in a restricted range of non-uniformity, shown enclosed in the thick rectangle. Error bars indicate standard error.

better than RS4. For highly peaked data \(1/\rho \to 0\), all rules fail, which is not surprising as in this case, some of the input features are barely ever active. For uniform data \(1/\rho \to \infty\), the various trace rules perform somewhat more similarly. This can be understood by considering that in that case, the eigenvectors of both \(\text{Cov}(x)\) and \(E[x]E[x]^T\) are unit vectors (when considering only the units to which the pooling cells is sensitive), such that the value of parameter \(\eta\) doesn’t have as strong an impact on learning.

Figure 4 shows the evolution of variance over time for the different learning rules for \(1/\rho = 6\) (corresponding to the middle of the rectangle in Figure 3). Again, all trace variants clearly outperform Hebb’s rule, and the ordering of various traces again reflects that found in the previous results. Figure 4 also shows that learning is stable over time in all cases, although an analytical demonstration of stability is still lacking.
Figure 4: Temporal evolution of learning at a specific level of non-uniformity (1/\(\rho\) = 6). Error bars indicate standard error.

6 Deriving optimal trace learning parameters

We now use our analytic expression of the Trace rule to derive optimal learning parameters. From Eq. 9, it is possible to compute an estimate for the \(\alpha\) that will maximally uniformize weights by solving the following convex optimization problem:

\[
\min_{\alpha} 1 - \eta \frac{1}{2} \sum_{ij} (\alpha c_{ij} + \pi_{ij} - \sigma)^2, \tag{11}
\]

where \(c_{ij}\) stands for element (i,j) in \(\text{Cov}(x)\), \(\pi_{ij}\) stands for element (i,j) in \(E[x]E[x]^T\), and \(\sigma\) is an arbitrary constant. The solution to Eq. 11 is:

\[
\alpha^* = -\frac{\sum_{ij} (\pi_{ij} - \sigma) c_{ij}}{\sum_{ij} c_{ij}^2} \tag{12}
\]

From Eq. 12, in the case of uniform distribution and suitably chosen \(\sigma\), \(\pi_{ij} = \sigma\), and the optimal mixing coefficient becomes 0. The best way to learn an invariant pooling operator from uniform data is thus to use Trace rule RS4. A partial confirmation of this result is shown in Fig. 3, where RS4 surpasses RS10 for sufficiently uniform data.

Figure 5 shows values of \(\alpha^*\) computed for various levels of non-uniformity. As before, random binary data vectors (1000 training vectors of dimension 16) were generated from an exponential distribution for various values of the rate parameter \(\rho\). Clearly, for all levels of non-uniformity considered, \(\alpha^*\) is always non-positive.
Figure 5: Effect of increasing uniformity. For non-uniform data, removing a portion of the covariance ($\alpha < 0$) contributes to uniformizing weights, thereby explaining why RS8ii and RS10 perform better than the other trace rules. For uniform data, the optimal $\alpha$ tends to 0. The original trace rule RS1 is well beyond the optimal solutions.

The more non-uniform the distribution, the more negative $\alpha^*$ becomes. As the data becomes more uniformly distributed, $\alpha^*$ tends to 0. Note that this result does not vary much with respect to the precise value of $\sigma$ (to verify this, values of $\sigma$ between .1 and 100 were tried and yielded the same results). For comparison purposes, the $\alpha$-values for the learning rules considered above are overlayed on the Figure. Note that the best performing rules on natural video simulations (rules 8ii and 10) fall in the leftward portion of the curve, whereas rule 4 lies along the right portion, as uniformity increases. Subtracting a portion of the covariance matrix is thus a good way to mitigate the non-uniformity of the data. This can be compared to rule 1, the typical trace rule, which clearly lies above the region of optimality.

7 Discussion

This work is motivated by the goal of better understanding how the trace rule leads to spatial invariance. We have shown in particular that it is useful to decouple
the effects of instantaneous "Hebbian" learning from that of trace learning. In this respect, our analysis complements that of Sprekeler et al. (2007), Maurer (2006) and Wiskott (2002). Our results are also more generally relevant to the study of pooling in convolutional networks (Boureau et al., 2008; Caponnetto et al., 2007).

We have highlighted the role of non-uniformity in the data distribution, which clearly impacts the ability to learn uniform weight kernels. Although it is widely accepted that some form of Hebbian learning occurs in the brain, these results show that Hebb’s rule is not adequate to learn kernels that can subtend spatial invariance in networks that abide to the constraints of realistic inputs, locality and synaptic, linearly weighted transmission of activity.

This work points to new relationships between classes of object recognition models. Convolutional networks are a particular case of symmetry networks, i.e. neural networks in which some connection weights are equal to others, or in other words, networks that are invariant under certain permutations among their weights (Shawe-Taylor, 1993). The symmetries in convolutional networks are built-in both at the level of feature maps (enforcing that units tuned to the same feature under some transformation should have identical sampling weights) and at the level of pooling units (via the pooling operator which as we have seen require identical weights amongst pooled units). From the point of view of cognitive neuroscience, the impressive success of convolutional networks should only make it more pressing to answer the question of how the symmetries they postulate could emerge in the brain. There have been very few but notable attempts in this direction (Webber, 2000), the present article being an attempt that places the temporal nature of the unsupervised learning rule at the focus of investigation. Specifically, if the Trace rule indeed makes weights from similarly tuned units more identical, then Max-based convolutional networks such as HMax that do not use competition between units could be more related to non-convolutional, but competitive networks like Visnet than was previously thought. This is because when combined with the competition in a pool of units, identical pooling weights in Visnet would essentially implement a Max pooling operator as in HMax.
We have also provided an explanation for why different versions of the trace rule show different performance levels: because they handle non-uniformity differently. To our knowledge, this is the first time the results of Rolls & Stringer (2001) are given a formal explanation. We have confirmed that their results hold in two-layer simulations with both natural video and a handcrafted dataset. In general, we predict that in most situations, trace rules that yield negative $\alpha$ coefficients in Eq. 9 will perform favorably compared to those that yield a positive $\alpha$. It would be particularly useful to determine a value for this coefficient based on the non-uniformity measured in natural videos. Although the differences in performance observed here are relatively small, we conjecture that when used within a deep learning architecture (as in Visnet) the benefits of reducing the variance across weights gradually leads to greater invariance.

8 Conclusion

In this paper we provide an explanation for why different versions of the Trace rule lead to different levels of invariance. Our analysis emphasizes, for the first time, the role of non-uniformity in the responses of feature detectors in natural settings on the development of pooling operators. We have developed a theoretical framework and simulation paradigm from which it is easy to understand the role of non-uniformity, and to predict which parameter sets should lead to the best performance depending on the distributions of activity in feature detectors.

9 Acknowledgments

The second author was supported by ERC research grant #230313.

References


10 Appendix

This appendix shows how the values for RS4, RS8ii and RS10 in Table 1 are computed. Starting with RS8ii, using the approximation of Eq. 9, we observe that:

\[
E[\Delta w] = \mathbb{E}[(\beta \bar{y}_t - y_t)x_t]
\]

\[
\approx \beta[(1 - \eta)\text{Cov}(x) + \mathbb{E}[x_t]\mathbb{E}[x^T_t]]w - [\text{Cov}(x) + \mathbb{E}[x_t]\mathbb{E}[x^T_t]]w
\]

\[
= (\beta - 1)\left[\frac{\beta(1 - \eta) - 1}{\beta - 1}\right]\text{Cov}(x)w + \mathbb{E}[x_t]\mathbb{E}[x^T_t]w
\]

\[
\approx \frac{\beta(1 - \eta) - 1}{\beta - 1}\text{Cov}(x)w + \mathbb{E}[x_t]\mathbb{E}[x^T_t]w
\]

(13)

Given a fixed constant of \(\eta = 0.8\), the value in Table 1 for RS8ii is obtained by replacing \(\beta = 2.5\).

The value for RS4 is similarly computed using the fact, shown in (Rolls & Stringer, 2001), that RS4 can be converted to Eq. 13 up to a multiplicative constant, and where \(\beta = 1/(1 - \eta) = 5\). In this case, the first term in Eq. 14 is
cancelled out.

Finally, for RS10 we note that:

\[
E[\Delta w] = \mathbb{E}[(\beta \bar{y}_{t-1} - y_t)x_t]
\]

\[
= \mathbb{E}[\left(\frac{\beta}{\eta} (\bar{y}_t - (1 - \eta)y_t) - y_t\right)x_t]
\]

\[
= \frac{\beta}{\eta} \mathbb{E}[\bar{y}_tx_t] - \mathbb{E}[\left(\frac{\beta(1-\eta)}{\eta} + 1\right)y_tx_t]
\]

\[
\approx \frac{\beta}{\eta}[(1 - \eta)Cov(x) + \mathbb{E}[x_t]\mathbb{E}[x^T_t]]w - (1 + \frac{\beta(1-\eta)}{\eta})(Cov(x) + \mathbb{E}[x_t]\mathbb{E}[x^T_t])w
\]

\[
= (\beta - 1)\frac{-1}{(\beta-1)} Cov(x)w + \mathbb{E}[x_t]\mathbb{E}[x^T_t]w
\]

\[
\approx \frac{-1}{(\beta-1)} Cov(x)w + \mathbb{E}[x_t]\mathbb{E}[x^T_t]w
\]  

(15)

The value for RS10 in Table 1 is then obtained when setting $\beta = 4.9$. 

20