Deep learning of transformations

Roland Memisevic

University of Montreal, LISA lab

June 23, 2014

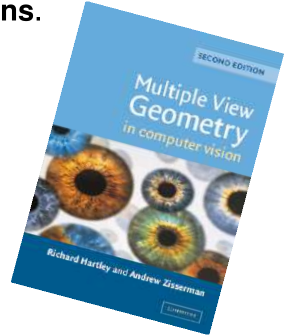
Beyond object recognition

- Geometry, stereo, structure-from-motion, motion understanding, activity analysis, tracking, optic flow, odometry, modeling articulation, modeling object relations, detailed scene understanding, analogy making, ...

Beyond object recognition

- Geometry, stereo, structure-from-motion, motion understanding, activity analysis, tracking, optic flow, odometry, modeling articulation, modeling object relations, detailed scene understanding, analogy making, ...

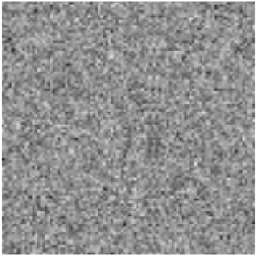
- To make progress with representation learning, it is necessary to represent **relations**.

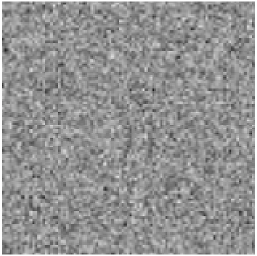


Some things are hard to infer from still images

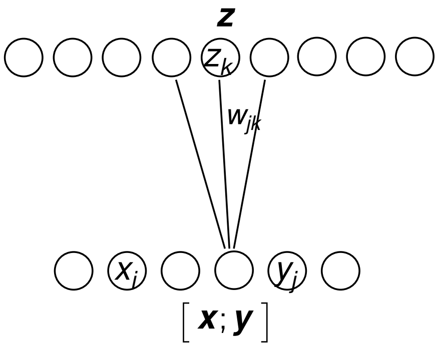
(Ayvaci, Soatto 2012)

Random dot stereograms



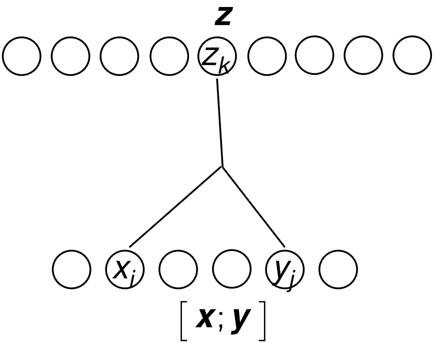


Learning relations by concatenating two inputs?



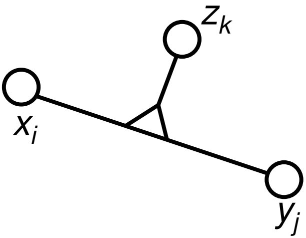
- Problem: This would make unit x_i conditionally independent of unit y_j, given $\boldsymbol{z}$.

Learning relations by concatenating two inputs?



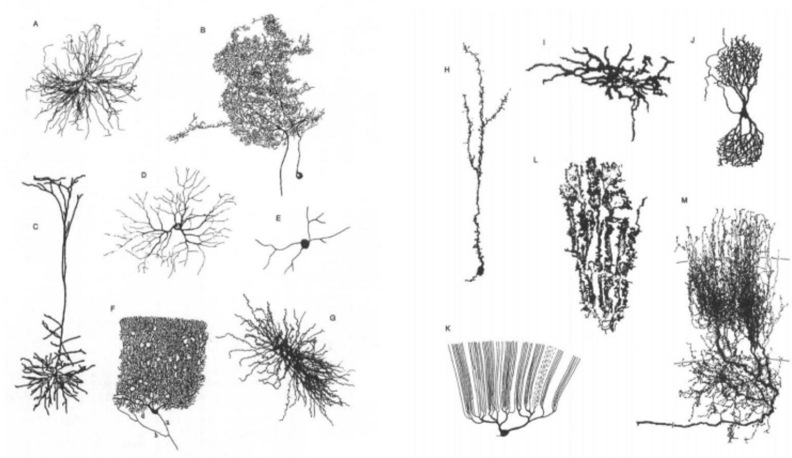
- Solution: Put x_i and y_j in a single clique.
- This will require "transistor neurons" that can do more than the usual weighted summation $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$.

Mapping units



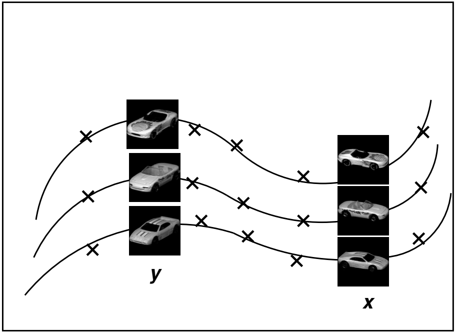
- (Hinton $\approx$ 1980), (v.d. Malsburg $\approx$ 1980)
- determine connection strength at run time
- blend in a sub-network dynamically
- route information (attention) (Olshausen 1994)
- closely related to motion energy models (Adelson, Bergen 1985)
- solve the binding problem (Smolensky 1990; Plate 1994)
- compute logical ANDs (Zetzsche $\approx$ 2000)
- add capacity within a single layer
- treat relations as first-class objects

$w^{\mathrm{T}}x$?



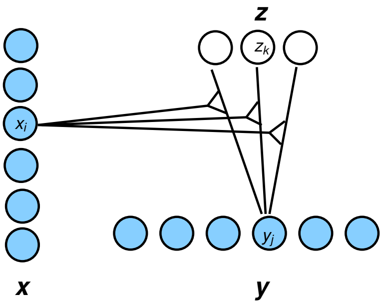
- Some neuroscientists believe that we will need to look beyond weighted summation to understand the brain.
- Mel, 1994

Relations as first-class objects



- If ***y*** is a transformed version of ***x***, then ***y*** will be on a **conditional manifold.**
- This suggsets learning a model for ***y***, while letting parameters be **a function of *x***.

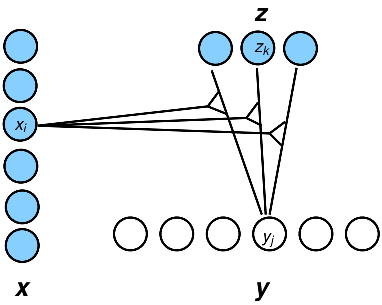
Bi-linear models



- Set $w_{jk}(\boldsymbol{x}) = \sum_i w_{\color{red}{\boldsymbol{i}}jk} x_i$:

$$z_k = h\Big(\sum_j w_{jk} y_j\Big) = h\Big(\sum_j \Big(\sum_i w_{ijk} x_i\Big) y_j\Big) = h\Big(\sum_{ij} w_{ijk} x_i y_j\Big)$$

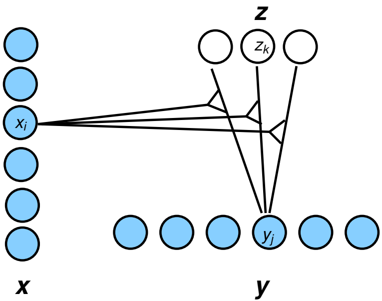
Bi-linear models



- Similar for $\boldsymbol{y}$:

$$y_j = \sum_k w_{jk} z_k = \sum_k \big(\sum_i w_{ijk} x_i \big) z_k = \sum_{ik} w_{ijk} x_i z_k$$

Learning

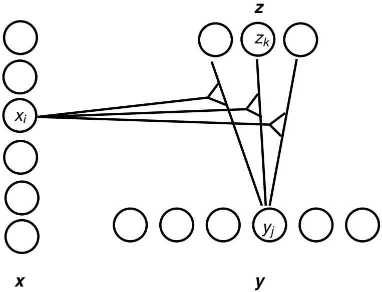


- For learning optimize conditional cost such as

$$\sum_j \Big(y_j - \sum_{ik} w_{ijk} x_i z_k(\boldsymbol{x}, \boldsymbol{y})\Big)^2$$

- (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007), (Memisevic, Hinton; 2007)

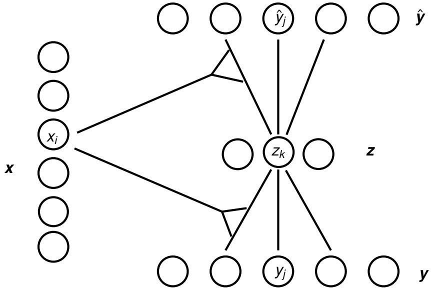
Gated boltzmann machine



$$E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{ijk} w_{ijk} x_i y_j z_k$$
$$p(\boldsymbol{y}, \boldsymbol{z} | \boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\right)$$
$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y},\boldsymbol{z}} \exp\left(E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\right)$$
$$p(z_k | \boldsymbol{x}, \boldsymbol{y}) = \mathrm{sigmoid}(\sum_{ij} W_{ijk} x_i y_j)$$
$$p(y_j | \boldsymbol{x}, \boldsymbol{z}) = \mathrm{sigmoid}(\sum_{ik} W_{ijk} x_i z_k)$$

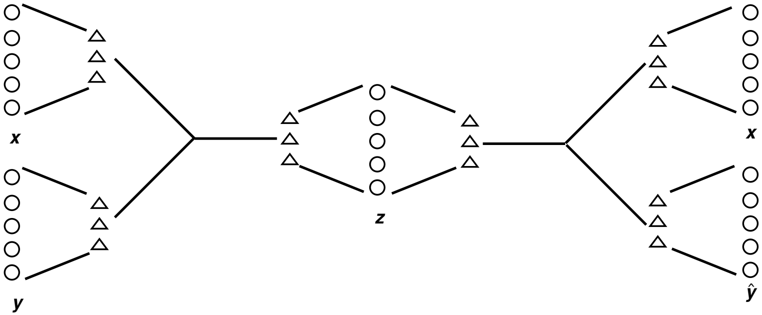
- (Memisevic, Hinton; 2007)

Gated autoencoder



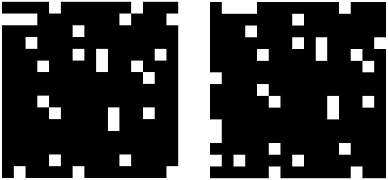
- Encoder and decoder weights become functions of $\boldsymbol{x}$.
- Train with back-prop (Memisevic, 2008)

Parameter factorization



- Projecting onto filters *first* allows us to use fewer products. (Memisevic, Hinton 2010), (Taylor et al 2009)
- This is equivalent to *factorizing* the three-way parameter tensor.

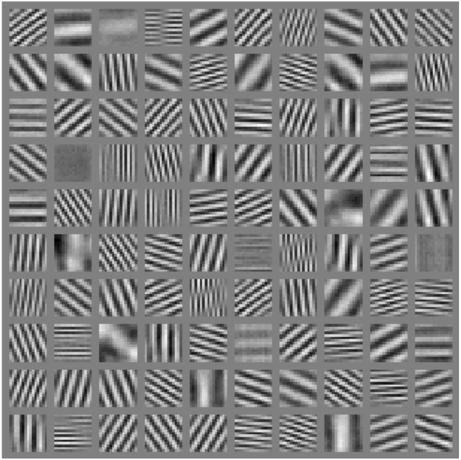
Learning relational features



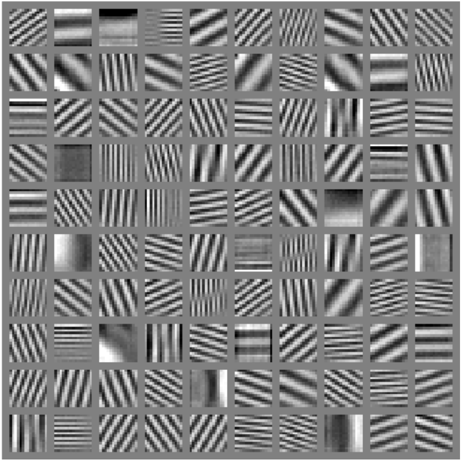
x *y*

- There is no structure in these images so vanilla feature learning won't work.

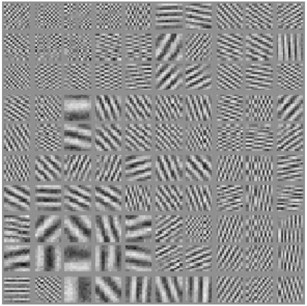
Input filters from a factored gating model

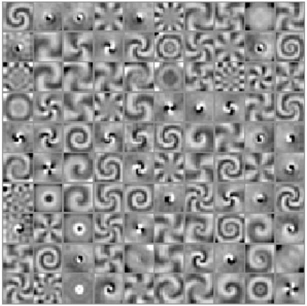


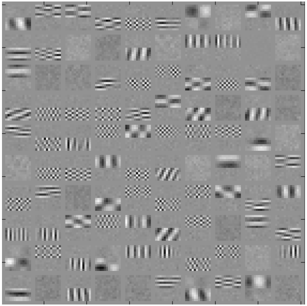
Output filters from a factored gating model

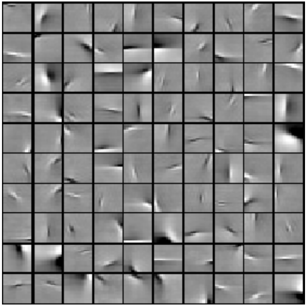


Learned filters

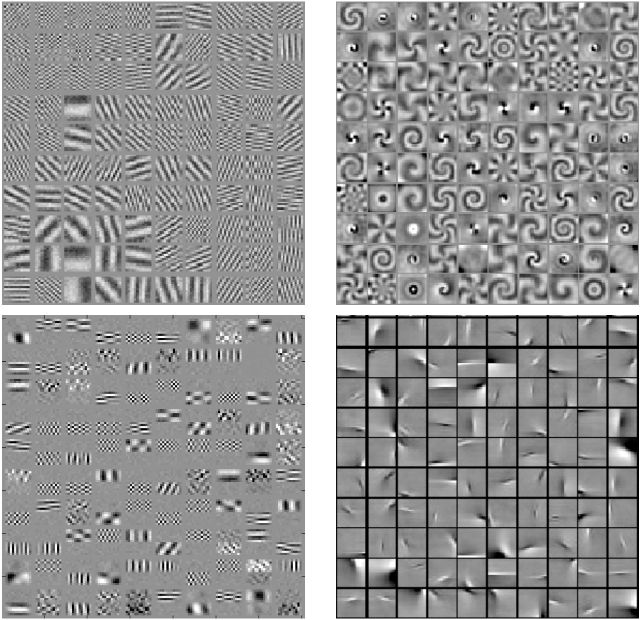








Learned filters

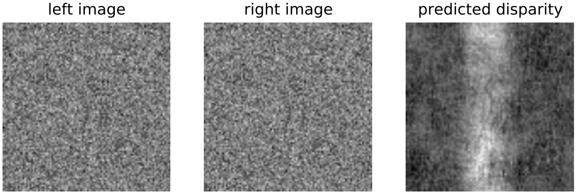


Face filters (Susskind et al. 2011)

Analogy making (infer ***z***, then compute ***y*** with new ***x*** clamped):

Applications of gating connections

- Activity recognition (Taylor et al., 2010), (Le, et al., 2011)
- Learning time series/MOCAP (Taylor et al., 2009)
- Learning depth cues, 3-D activity (SOTA) (Konda, 2013)
- Better generative models of images (Ranzato, et al., 2009)
- Invariance from video (Cadieu, Olshausen 2011), (Zou et al. 2012), (Memisevic, Exarchakis 2013)
- Simple analogy making (Memisevic, Hinton 2010), (Susskind, et al., 2011)



Some theoretical insights

(I) Orthogonal transformations decompose into 2-D rotations:

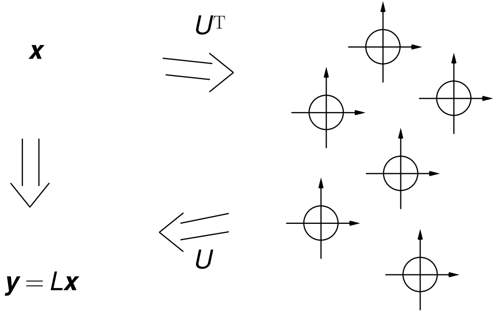
$$U^{\mathrm{T}} L U = \begin{bmatrix} R_1 & & \\ & \ddots & \\ & & R_k \end{bmatrix} \qquad R_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix}$$

- (Eigen-decomposition $L = UDU^{\mathrm{T}}$ has complex eigenvalues of length 1)

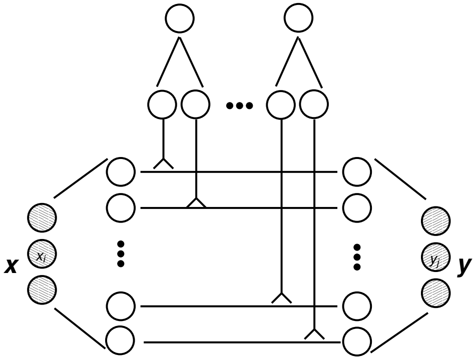
(II) Commuting transformations share an eigen-basis:

- They differ only with respect to the rotation-angle they apply in their eigenspace.

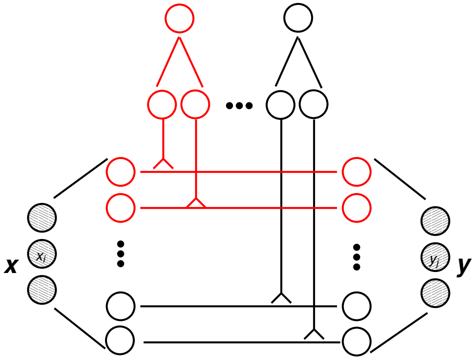
(I)+(II)



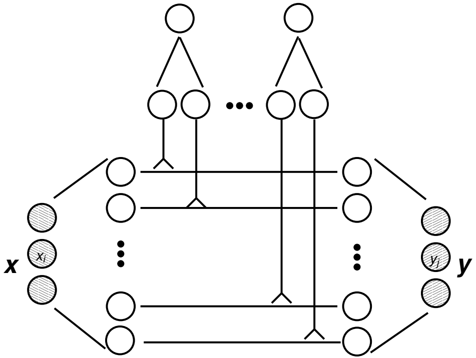
To detect the rotation angle, compute a 2-d inner product



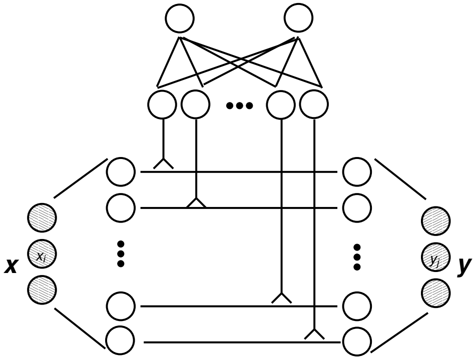
To detect the rotation angle, compute a 2-d inner product



To detect the rotation angle, compute a 2-d inner product

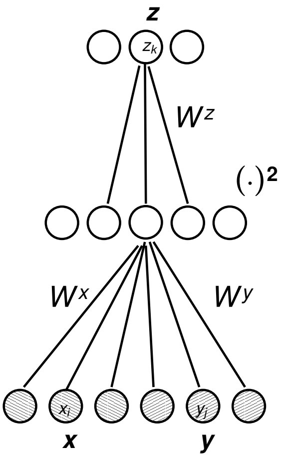


To detect the rotation angle, compute a 2-d inner product



Gating and square pooling

- (Adelson, Bergen 1985)
- ASSOM (Kohonen 1996)
- ISA (Hyvarinen 2000)
- PoT model (Welling et al. 2002)
- (Karklin Lewicki 2008)
- mcRBM (Ranzato et al. 2009)

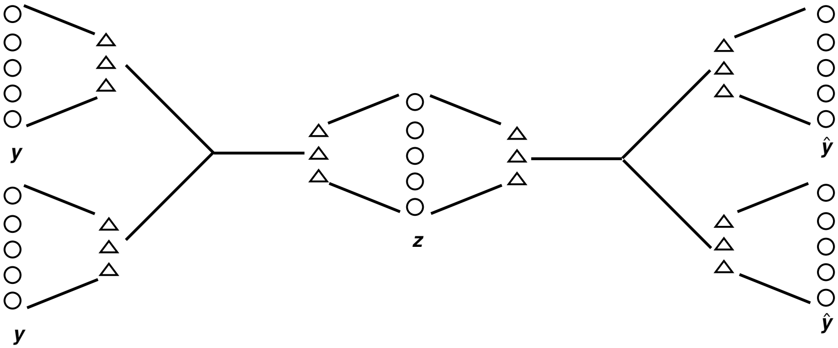


- The activity for hidden unit k:

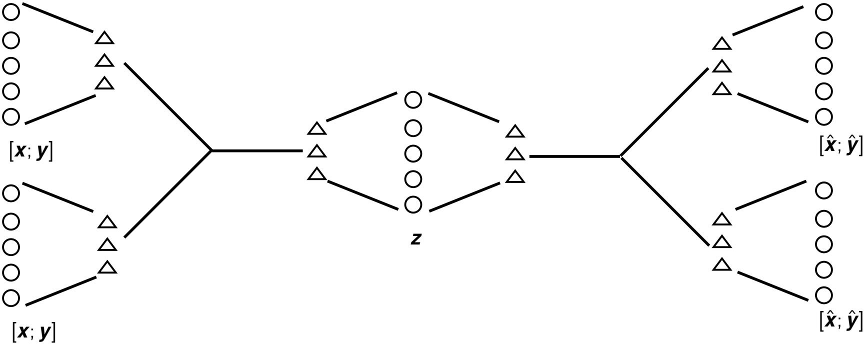
$$\sum_f W^z_{kf}\left(W^{x\,\mathrm{T}}_{\cdot f}\boldsymbol{x} + W^{y\,\mathrm{T}}_{\cdot f}\boldsymbol{y}\right)^2$$

$$= \sum_f W^z_{kf}\left(2(W^{x\,\mathrm{T}}_{\cdot f}\boldsymbol{x})(W^{y\,\mathrm{T}}_{\cdot f}\boldsymbol{y}) + (W^{x\,\mathrm{T}}_{\cdot f}\boldsymbol{x})^2 + (W^{y\,\mathrm{T}}_{\cdot f}\boldsymbol{y})^2\right)$$

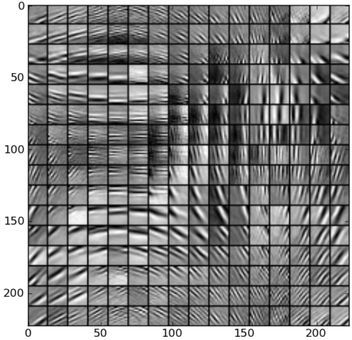
Square pooling via gating



Gating via square pooling via gating



Topographic filter maps



Directions

- Gating can solve different types of task with a single type of module.
- $\rightarrow$ Use gating to get more mileage out of local learning rules?
- Gating tends to orthonormalize weights.
- $\rightarrow$ Gating units in deep networks?
- $\rightarrow$ Gating units in recurrent networks?