Multiview Feature Learning

Roland Memisevic

Frankfurt, Montreal

Tutorial at IPAM 2012

Outline

1. **Introduction**
   - Feature Learning
   - Correspondence in Computer Vision
   - Multiview feature learning
2. **Learning relational features**
   - Encoding relations
   - Learning
3. **Factorization, eigen-spaces and complex cells**
   - Factorization
   - Eigen-spaces, energy models, complex cells
4. **Applications and extensions**
   - Applications and extensions
   - Conclusions

Outline

1. **Introduction**
   - Feature Learning
   - Correspondence in Computer Vision
   - Multiview feature learning
2. **Learning relational features**
   - Encoding relations
   - Learning
3. **Factorization, eigen-spaces and complex cells**
   - Factorization
   - Eigen-spaces, energy models, complex cells
4. **Applications and extensions**
   - Applications and extensions
   - Conclusions

What this is about

- Extend feature learning to model *relations*.
- “mapping units”, “bi-linear models”, “energy-models”, “complex cells”, “spatio-temporal features”, “covariance features”, “bi-linear classification”, “quadrature features”, “gated Boltzmann machine”, “mcrbm”, ...
- **Feature learning beyond object recognition**

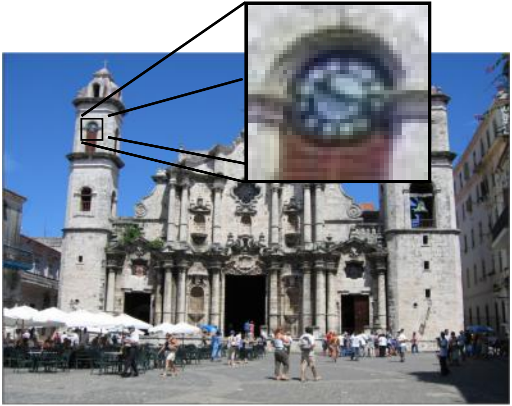
What this is about

- Extend feature learning to model *relations*.
- “mapping units”, “bi-linear models”, “energy-models”, “complex cells”, “spatio-temporal features”, “covariance features”, “bi-linear classification”, “quadrature features”, “gated Boltzmann machine”, “mcrbm”, ...
- **Feature learning beyond object recognition**

Local features for recognition

- Object recognition started to work very well.
- The main reason is the use of **local features.**

Local features for recognition



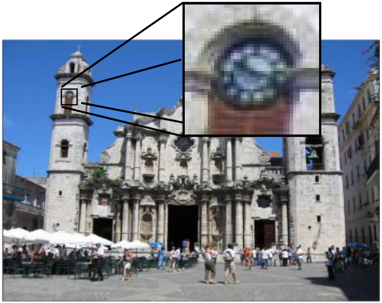
- Object recognition started to work very well.
- The main reason is the use of **local features.**

Bag-Of-Features

Bag-Of-Features

1. Find **interest points.**
2. Crop patches around interest points.
3. Represent each patch with a **sparse local descriptor** ("features").
4. **Add** all local descriptors to obtain a global descriptor for the image.

Bag-Of-Features



Bag-Of-Features

1. Find **interest points.**
2. Crop patches around interest points.
3. Represent each patch with a **sparse local descriptor** ("features").
4. **Add** all local descriptors to obtain a global descriptor for the image.

Bag-Of-Features

$$\longrightarrow \begin{pmatrix} f_1 \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ f_n \end{pmatrix}$$

Bag-Of-Features

1. Find **interest points**.
2. Crop patches around interest points.
3. Represent each patch with a **sparse local descriptor** (“features”).
4. **Add** all local descriptors to obtain a global descriptor for the image.

Bag-Of-Features

$$\begin{pmatrix} f_1^1 \\ \vdots \\ f_n^1 \end{pmatrix} + \cdots\cdots + \begin{pmatrix} f_1^M \\ \vdots \\ f_n^M \end{pmatrix}$$

Bag-Of-Features

1. Find **interest points**.
2. Crop patches around interest points.
3. Represent each patch with a **sparse local descriptor** (“features”).
4. **Add** all local descriptors to obtain a global descriptor for the image.

Bag-Of-Features

$$\begin{pmatrix} f_1^1 \\ \vdots \\ f_n^1 \end{pmatrix} + \cdots\cdots + \begin{pmatrix} f_1^M \\ \vdots \\ f_n^M \end{pmatrix}$$

Bag-Of-Features

1. Find **interest points**.
2. Crop patches around interest points.
3. Represent each patch with a **sparse local descriptor** (“features”).
4. **Add** all local descriptors to obtain a global descriptor for the image.

Convolutional

Convolutional

1. Crop patches along a regular grid (dense or not).
2. Represent each patch with a local descriptor.
3. **Concatenate** all descriptors in a very large vector.

Convolutional

Convolutional

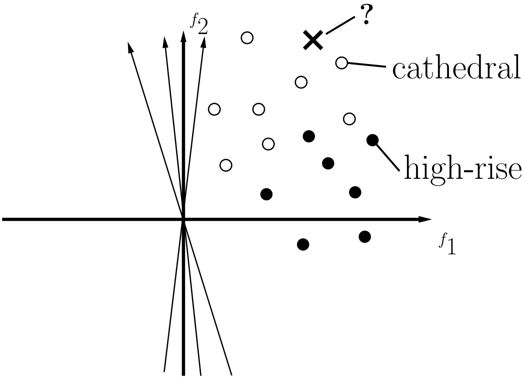
1. Crop patches along a regular grid (dense or not).
2. Represent each patch with a local descriptor.
3. **Concatenate** all descriptors in a very large vector.

Convolutional

Convolutional

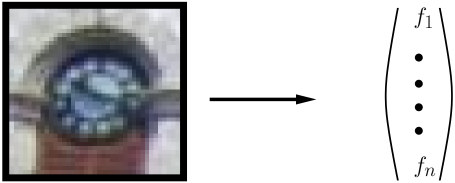
1. Crop patches along a regular grid (dense or not).
2. Represent each patch with a local descriptor.
3. **Concatenate** all descriptors in a very large vector.

Classification



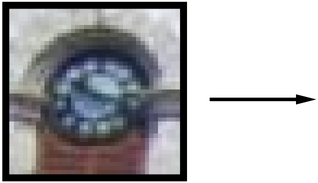
- After computing representations, use logistic regression, SVM, NN, ...
- There are various extensions, like fancy pooling, etc.

Extracting local features



- How to extract local features.
- Engineer them. SIFT, HOG, LBP, etc.
- *Learn* them from image data $\rightarrow$ **deep learning**

Extracting local features



$$\begin{pmatrix} f_1 \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ f_n \end{pmatrix}$$

- How to extract local features.
- Engineer them. SIFT, HOG, LBP, etc.
- *Learn* them from image data $\rightarrow$ **deep learning**

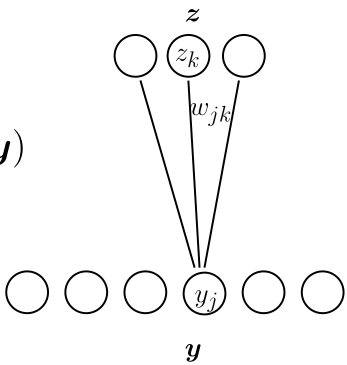
Extracting local features

$$\begin{pmatrix} f_1 \\ \vdots \\ \vdots \\ f_n \end{pmatrix}$$

- How to extract local features.
- Engineer them. SIFT, HOG, LBP, etc.
- *Learn* them from image data → **deep learning**

Feature learning

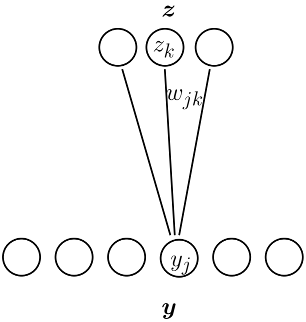
$$\begin{aligned} \boldsymbol{z}(\boldsymbol{y}) &= \mathrm{sigmoid}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{y}) \\ \boldsymbol{y}(\boldsymbol{z}) &= \boldsymbol{W}\boldsymbol{z} \end{aligned}$$



Feature learning

$$\boldsymbol{W} = \arg\min_{\boldsymbol{W}} \sum_{\alpha} \|\boldsymbol{y}^{\alpha} - \boldsymbol{y}(\boldsymbol{z}(\boldsymbol{y}^{\alpha}))\|^2$$

Feature learning models



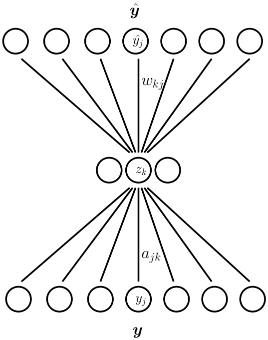
$$p(y_j|\boldsymbol{z}) = \text{sigmoid}\Big(\sum_k w_{jk} z_k\Big)$$

$$p(z_k|\boldsymbol{y}) = \text{sigmoid}\Big(\sum_j w_{jk} y_j\Big)$$

Restricted Boltzmann machine (RBM)

- $p(\boldsymbol{y}, \boldsymbol{z}) = \frac{1}{Z} \exp\left(\sum_{jk} w_{jk} y_j z_k\right)$
- Learning: Maximum likelihood/contrastive divergence.

Feature learning models



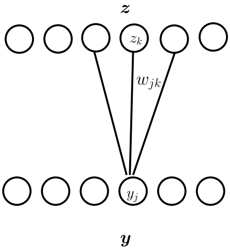
$$z_k = \text{sigmoid}\Big(\sum_j a_{jk} y_j\Big)$$

$$y_j = \sum_k w_{jk} z_k$$

Autoencoder

- Add **inference parameters.**
- Learning: Minimize reconstruction error.
- Add a sparsity penalty or *corrupt inputs during training* (Vincent et al., 2008).

Feature learning models



$$y_j = \sum_k w_{jk} z_k$$

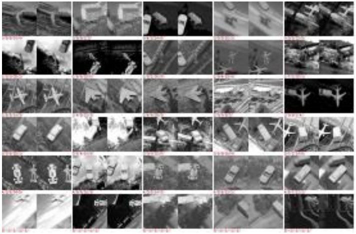
Independent Components Analysis (ICA)

- Learning: Make responses sparse, while keeping filters sensible

$$\min_W \|W^{\mathrm{T}}\boldsymbol{y}\|_1$$
$$\text{s.t.} \quad W^{\mathrm{T}}W = I$$

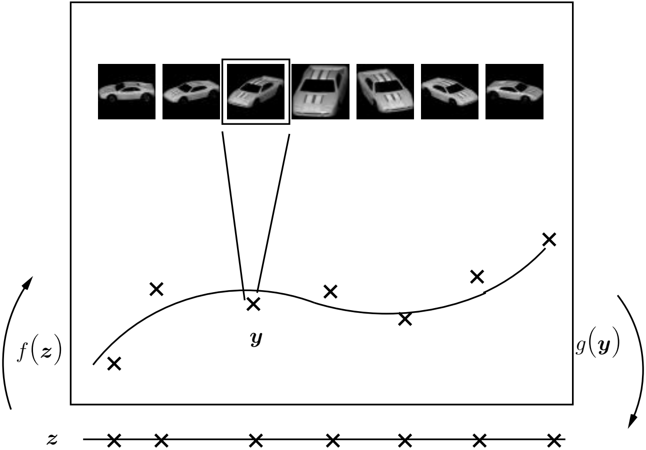
Feature Learning Works

(CIFAR)



(NORB)

Manifold perspective



Outline

Beyond object recognition

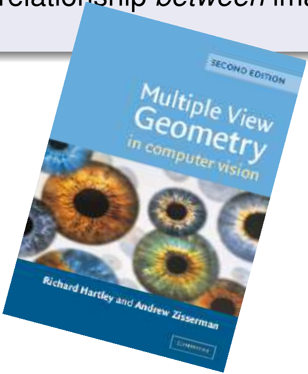
Can we do more with Feature Learning than recognize *things*?

- Brains can do much more than recognize objects.
- Many vision tasks go beyond object recognition.
- In surprisingly many of them, the relationship *between* images carries the relevant information.

Beyond object recognition

Can we do more with Feature Learning than recognize *things*?

- Brains can do much more than recognize objects.
- Many vision tasks go beyond object recognition.
- In surprisingly many of them, the relationship *between* images carries the relevant information.



Correspondences in Computer Vision

- **Correspondence** is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- Action Recognition
- Contours, Within-image structure

Correspondences in Computer Vision

- **Correspondence** is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- Action Recognition
- Contours, Within-image structure

Correspondences in Computer Vision

- **Correspondence** is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- Action Recognition
- Contours, Within-image structure

Correspondences in Computer Vision

- **Correspondence** is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- Action Recognition
- Contours, Within-image structure

Correspondences in Computer Vision

- **Correspondence** is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- Action Recognition
- Contours, Within-image structure

Correspondences in Computer Vision

- **Correspondence** is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- Action Recognition
- Contours, Within-image structure

Correspondences in Computer Vision

- **Correspondence** is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- Action Recognition
- Contours, Within-image structure

Correspondences in Computer Vision

- **Correspondence** is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- Action Recognition
- Contours, Within-image structure

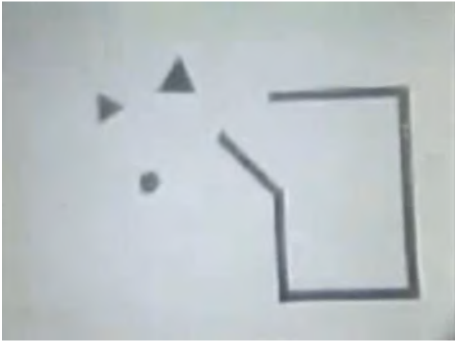
Correspondences in Computer Vision

- **Correspondence** is one of the most ubiquitous problems in Computer Vision.

Some correspondence tasks in Vision

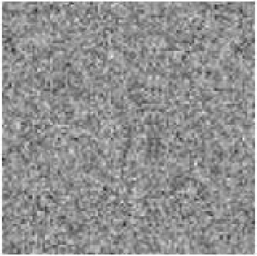
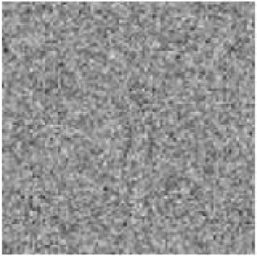
- Tracking
- Stereo
- Geometry
- Optical Flow
- Invariant Recognition
- Odometry
- Action Recognition
- Contours, Within-image structure

Heider and Simmel



- Adding frames is not just about adding proportionally more information.
- The relationships between frames contain additional information, that is not present in any single frame.
- See *Heider and Simmel, 1944:* Any single frame shows a bunch of geometric figures. The motions reveal the story.

Random dot stereograms

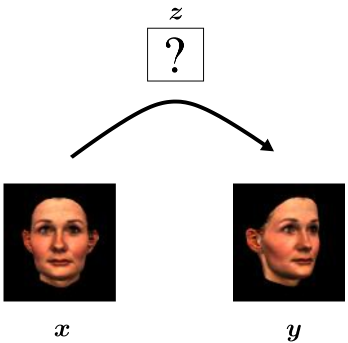
- You can see objects even when images contain *no* features.

Outline

1. **Introduction**
   - Feature Learning
   - Correspondence in Computer Vision
   - Multiview feature learning
2. **Learning relational features**
   - Encoding relations
   - Learning
3. **Factorization, eigen-spaces and complex cells**
   - Factorization
   - Eigen-spaces, energy models, complex cells
4. **Applications and extensions**
   - Applications and extensions
   - Conclusions

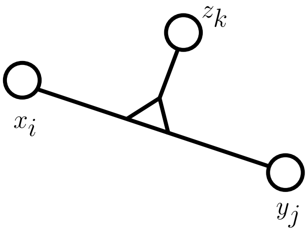
Learning features to model correspondences

- If *correspondences* matter in vision, **can we learn them**?



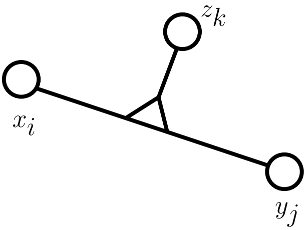
Learning features to model correspondences

- We can, if we let latent variables act like *gates*, that dynamically change the connections between fellow variables.



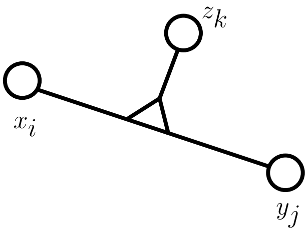
Learning features to model correspondences

- Learning and inference (slightly) different from learning without.
- We can set things up, such that inference is almost unchanged. Yet, the *meaning* of the latent variables will be entirely different.



Learning features to model correspondences

- Multiplicative interactions allow hidden variables to *blend in a whole "sub"-network*.
- This leads to a qualitatively quite different behaviour from the common, bi-partite feature learning models.



Multiplicative interactions

Brief history of gating

- "Mapping units" (Hinton; 1981), "dynamic mappings" (v.d. Malsburg; 1981)
- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, "Sigma-Pi-units"
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...

Multiplicative interactions

Brief history of gating

- "Mapping units" (Hinton; 1981), "dynamic mappings" (v.d. Malsburg; 1981)
- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, "Sigma-Pi-units"
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...

Multiplicative interactions

Brief history of gating

- “Mapping units” (Hinton; 1981), “dynamic mappings” (v.d. Malsburg; 1981)
- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, “Sigma-Pi-units”
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...

Multiplicative interactions

Brief history of gating

- “Mapping units” (Hinton; 1981), “dynamic mappings” (v.d. Malsburg; 1981)
- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, “Sigma-Pi-units”
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...

Multiplicative interactions

Brief history of gating

- "Mapping units" (Hinton; 1981), "dynamic mappings" (v.d. Malsburg; 1981)
- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, "Sigma-Pi-units"
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...

Multiplicative interactions

Brief history of gating

- "Mapping units" (Hinton; 1981), "dynamic mappings" (v.d. Malsburg; 1981)
- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, "Sigma-Pi-units"
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...

Multiplicative interactions

Brief history of gating

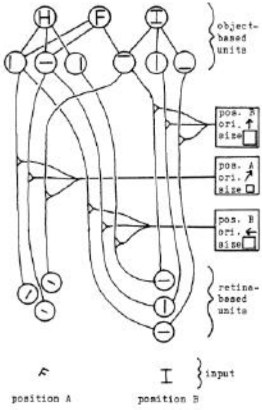
- "Mapping units" (Hinton; 1981), "dynamic mappings" (v.d. Malsburg; 1981)
- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, "Sigma-Pi-units"
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...

Multiplicative interactions

Brief history of gating

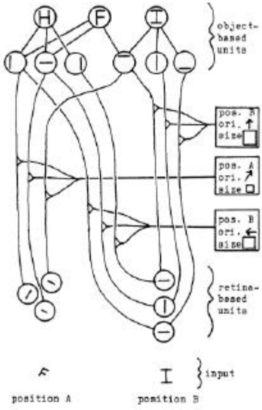
- “Mapping units” (Hinton; 1981), “dynamic mappings” (v.d. Malsburg; 1981)
- Binocular+Motion Energy models (Adelson, Bergen; 1985), (Ozhawa, DeAngelis, Freeman; 1990), (Fleet et al., 1994)
- Higher-order neural nets, “Sigma-Pi-units”
- Routing circuits (Olshausen; 1994)
- Bi-linear models (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007)
- Subspace SOM (Kohonen, 1996)
- ISA, topographic ICA (Hyvarinen, Hoyer; 2000), (Karklin, Lewicki; 2003): Higher-order within image structure
- (2006 –) GBM, mcRBM, GAE, convISA, applications...

Mapping units 1981



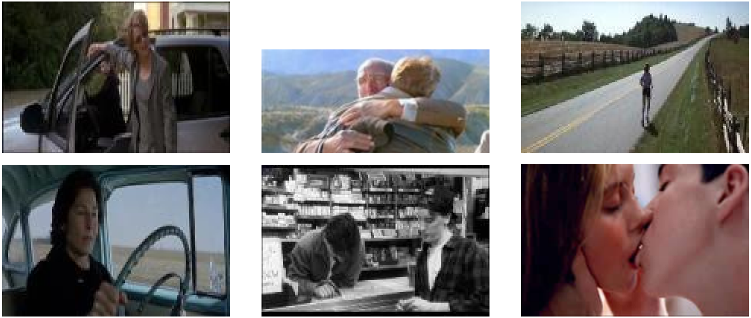
(Hinton, 1981)

Mapping units 1981



(Hinton, 1981)

Example application: Action recognition



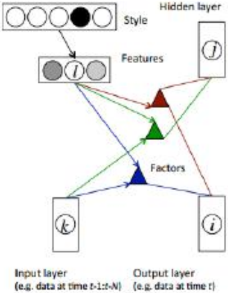
(Hollywood 2)

(Marszałek et al., 2009)

- Convolutional GBM (Taylor et al., 2010)
- hierarchical ISA (Le, et al., 2011)

Mocap

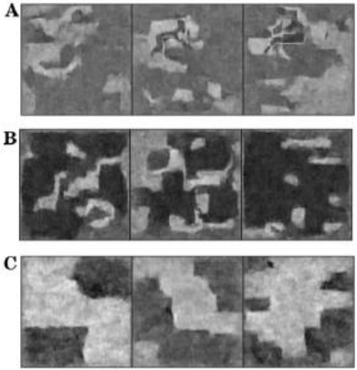
- (Taylor, Hinton; 2009), (Taylor, et al.; 2010)



Training	Test	Baseline	MoCorr [28]	GPLVM [13]	CMFA-VB [13]	CRBM	imCRBM-10
S1+S2+S3	S1	129.18±19.47	140.35	-	-	55.43±0.79	**54.27±0.49**
S1	S1		-	-	-	**48.75±3.72**	58.62±3.87
S1+S2+S3	S2	162.75±15.36	149.37	-	-	99.13±22.98	**69.28±3.30**
S2	S2		-	88.35±25.66	68.67±24.66	**47.43±2.86**	67.02±0.70
S1+S2+S3	S3	180.11±24.02	156.30	-	-	70.89±2.10	**43.40±4.12**
S3	S3		-	87.39±21.69	69.59±22.22	**49.81±2.19**	51.43±0.92

Gated MRFs

- (Ranzato et al., 2010)



Analogy making

Invariance

aperture feature similarities

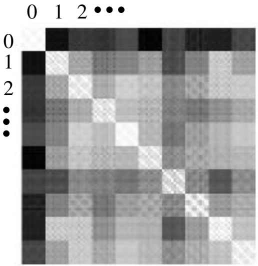
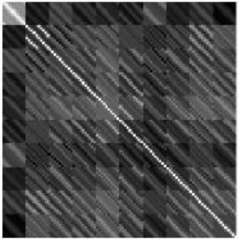


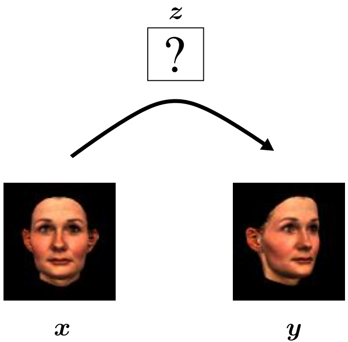
image similarities



Outline

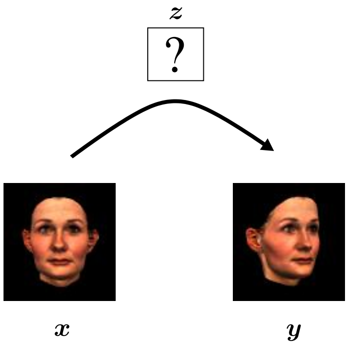
Outline

Sparse coding of images pairs?



- How to extend sparse coding to model relations?
- Sparse coding on the *concatenation*?

Sparse coding of images pairs?



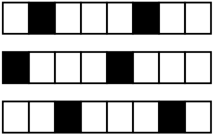
- How to extend sparse coding to model relations?
- Sparse coding on the *concatenation*?

Sparse coding on the concatenation ?

- A case study: Translations of binary, one-d images.
- Suppose images are random and can change in **one of three ways:**

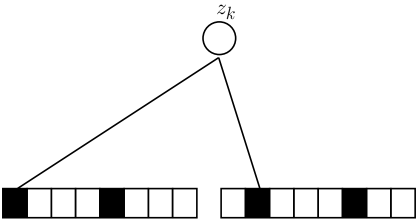
Example Image $\boldsymbol{x}$:

Possible Image $\boldsymbol{y}$:



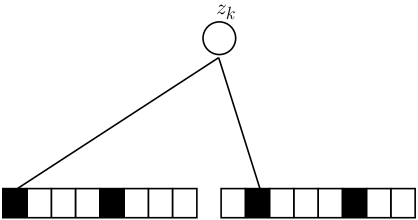
Sparse coding on the concatenation ?

- A hidden variable that collects evidence for a shift to the right.
- What if the images are random or noisy?
- Can we pool over more than one pixel?



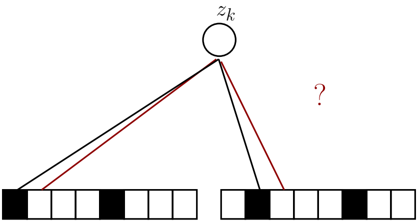
Sparse coding on the concatenation ?

- A hidden variable that collects evidence for a shift to the right.
- What if the images are random or noisy?
- Can we pool over more than one pixel?



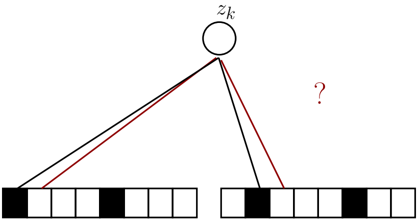
Sparse coding on the concatenation ?

- A hidden variable that collects evidence for a shift to the right.
- What if the images are random or noisy?
- Can we pool over more than one pixel?



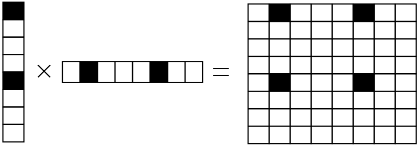
Sparse coding on the concatenation ?

- Obviously not, because now the hidden unit would get equally happy if it would see the non-shift (second pixel from the left).
- The problem: Hidden variables act like OR-gates, that accumulate evidence.

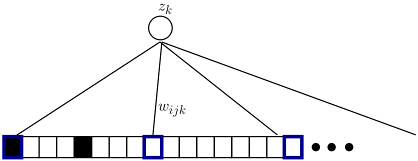


Cross-products

- Intuitively, what we need instead are logical ANDs, which can represent *coincidences* (eg. Zetzsche et al., 2003, 2005).
- This amounts to using the outer product $L := \mathrm{outer}(\boldsymbol{x}, \boldsymbol{y})$:

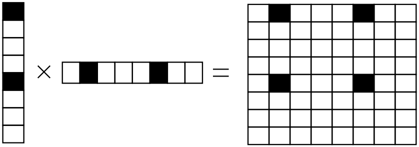


- We can unroll this matrix, and let this be the data:



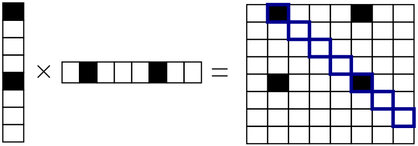
Cross-products

- Each component L_{ij} of the outer-product matrix will constitute evidence for exactly *one* type of shift.
- Hiddens pool over products of pixels.



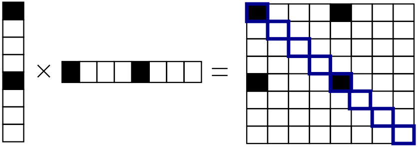
Cross-products

- Each component L_{ij} of the outer-product matrix will constitute evidence for exactly *one* type of shift.
- Hiddens pool over products of pixels.

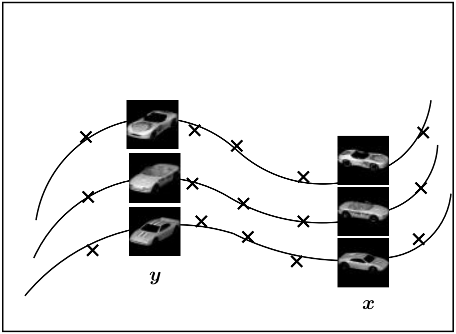


Cross-products

- Each component L_{ij} of the outer-product matrix will constitute evidence for exactly *one* type of shift.
- Hiddens pool over products of pixels.

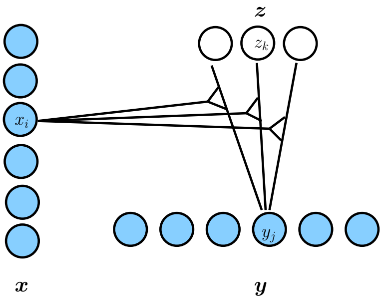


A different view: Families of manifolds



- Feature learning reveals the (local) manifold structure in data.
- When y is a transformed version of x, we can still think of y as being confined to a manifold, but it will be a **conditional manifold**.
- *Idea:* Learn a model for y, but let parameters be *a function of* x.

Conditional inference



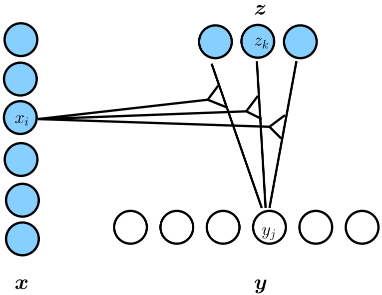
Inferring z

- If we use a linear function, $w_{jk}(\boldsymbol{x}) = \sum_i w_{ijk} x_i$, we get

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

- Inference via **bilinear** function of the inputs.

Conditional inference



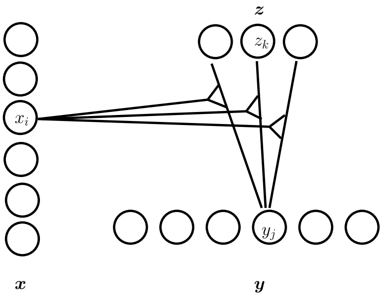
Inferring y

- To infer $\boldsymbol{y}$:

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

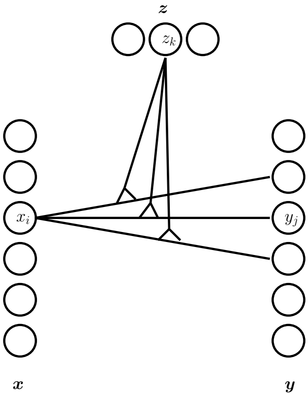
- Inference via **bilinear** function of $\boldsymbol{x}, \boldsymbol{z}$.

Input-modulated filters



- This is feature learning with input-dependent weights.
- Input pixels can vote for features in the output image.

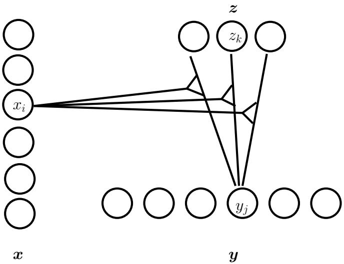
A different visualization



- A hidden can blend in one *slice* $W_{\cdot\cdot k}$ of the parameter tensor.
- A slice does linear regression in “pixel space”.
- So for binary hiddens, this is a **mixture of** 2^K **image warps.**

Outline

Learning is predictive coding



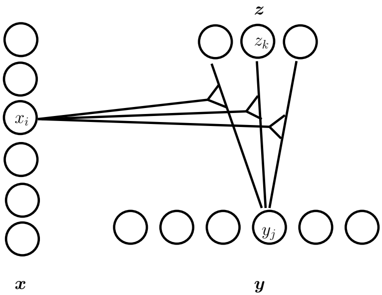
Predictive sparse coding

- The cost for a training pair $(\boldsymbol{x}, \boldsymbol{y})$ is:

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

- Training as usual: Infer z, update W. (Tenenbaum, Freeman; 2000), (Grimes, Rao; 2005), (Olshausen; 2007), (Memisevic, Hinton; 2007)

Example: Gated Boltzmann machine

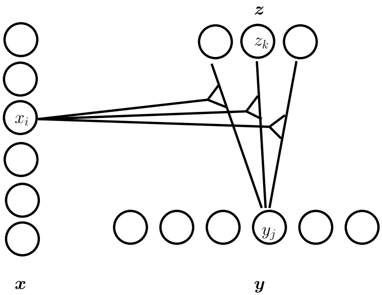


Three-way RBM (Memisevic, Hinton; 2007)

$$E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{ijk} w_{ijk} x_i y_j z_k$$

$$p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\big(E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big),\ Z(\boldsymbol{x}) = \sum_{\boldsymbol{y},\boldsymbol{z}} \exp\big(E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big)$$

Example: Gated Boltzmann machine

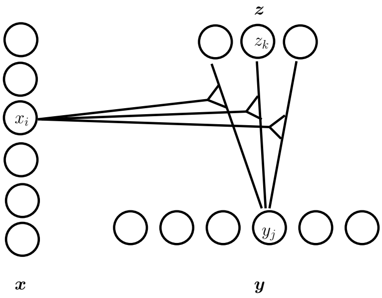


Three-way RBM (Memisevic, Hinton; 2007)

$$E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{ijk} w_{ijk} x_i y_j z_k$$

$$p(\boldsymbol{y}, \boldsymbol{z} | \boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\big(E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big), \; Z(\boldsymbol{x}) = \sum_{\boldsymbol{y},\boldsymbol{z}} \exp\big(E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big)$$

Example: Gated Boltzmann machine

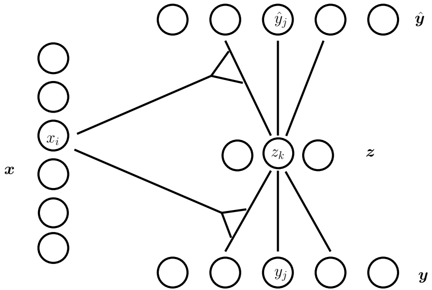


Three-way RBM (Memisevic, Hinton; 2007)

$$p(z_k|\boldsymbol{x}, \boldsymbol{y}) = \mathrm{sigmoid}(\sum_{ij} W_{ijk} x_i y_j)$$

$$p(y_j|\boldsymbol{x}, \boldsymbol{z}) = \mathrm{sigmoid}(\sum_{ik} W_{ijk} x_i z_k)$$

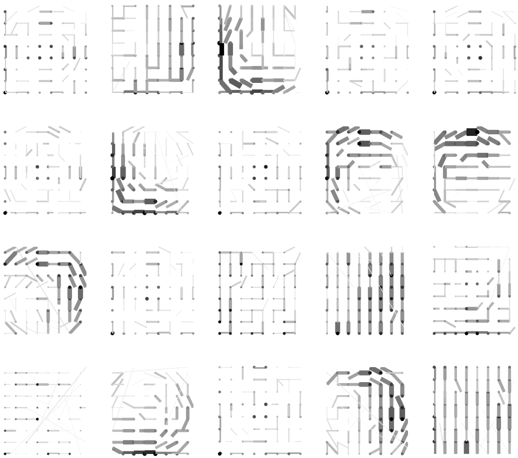
Example: Gated auto-encoder



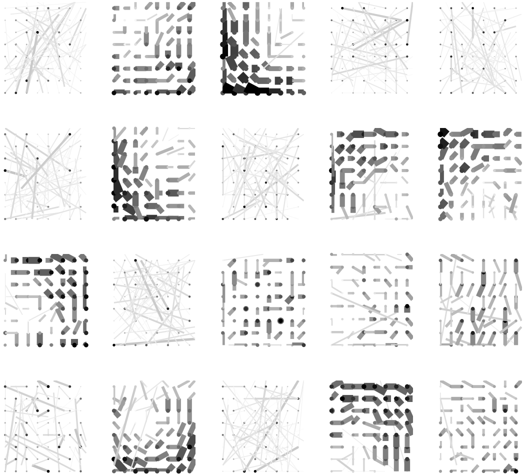
Gated autoencoders

- Turn encoder and decoder weights into functions of $\boldsymbol{x}$.
- Learning the same as in a standard auto-encoder for $\boldsymbol{y}$.
- The model is still a DAG, so back-prop works *exactly* like in a standard autoencoder. (Memisevic, 2011)

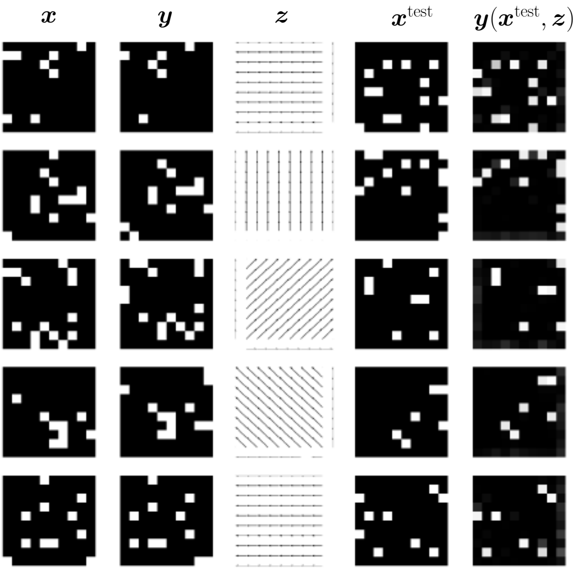
Toy example: Conditionally trained “Hidden flow-fields”



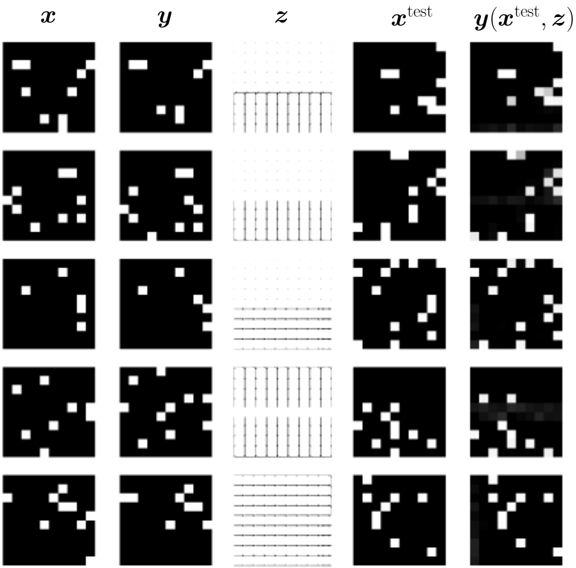
Toy example: Conditionally trained "Hidden flow-fields", inhibitory connections



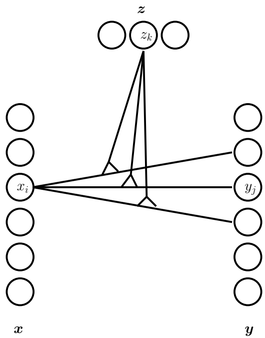
Toy example: Learning optical flow



"Combinatorial flowfields"

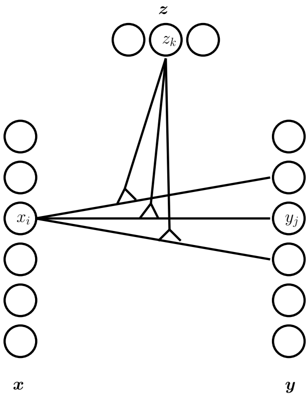


Joint training



- Conditional training makes it hard to answer questions like:
- "How likely are the given images transforms of one another?"
- To answer questions like these, we require a joint image model, $p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{z})$, given mapping units.

Joint training



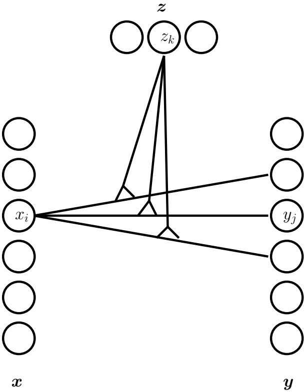
$$E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{ijk} w_{ijk} x_i y_j z_k$$

$$p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \frac{1}{Z} \exp\big(E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big)$$

$$Z = \sum_{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}} \exp\big(E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big)$$

- Use three-way sampling in a Gated Boltzmann Machine (Susskind et al., 2011).
- Can apply this to *matching* tasks (more later).

Joint training



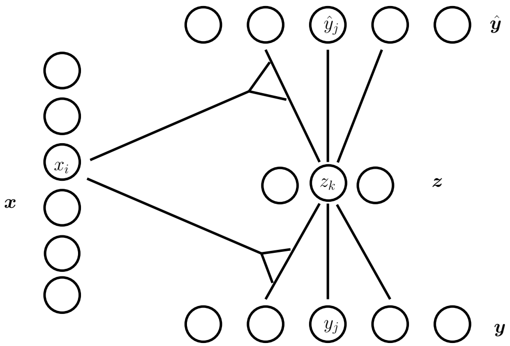
$$E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{ijk} w_{ijk} x_i y_j z_k$$

$$p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \frac{1}{Z} \exp\big(E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big)$$

$$Z = \sum_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}} \exp\big(E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big)$$

- Use three-way sampling in a Gated Boltzmann Machine (Susskind et al., 2011).
- Can apply this to *matching* tasks (more later).

Joint training

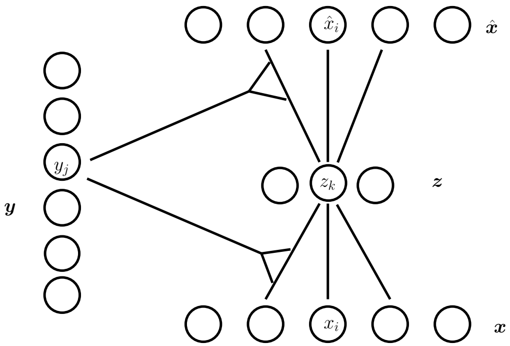


- For the autoencoder we can use a simple hack:
- Add up two conditional costs:

$$\sum_j \left(y_j - \sum_{ik} w_{ijk} x_i z_k\right)^2 + \sum_i \left(x_i - \sum_{jk} w_{ijk} y_j z_k\right)^2$$

- Force parameters to transform in both directions.

Joint training



- For the autoencoder we can use a simple hack:
- Add up two conditional costs:

$$\sum_j \left(y_j - \sum_{ik} w_{ijk} x_i z_k\right)^2 + \sum_i \left(x_i - \sum_{jk} w_{ijk} y_j z_k\right)^2$$

- Force parameters to transform in both directions.

Pool over products

Take-home message

To gather evidence for a transformation,
let hidden units compute the sum over products of input components.