Unsupervised learning of image transformations

Roland Memisevic, Geoffrey Hinton

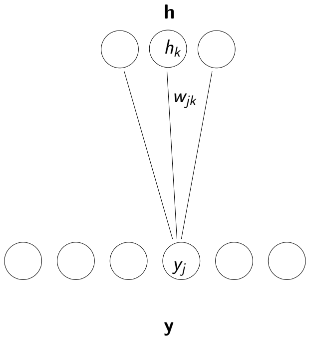
University of Toronto

March 19, 2007

'Real world data is not random'

- Use **unsupervised learning** to discover and exploit regularities in high-dimensional data.
- With the right *representation* many tasks become easier, or trivial. ("It's the feature, stupid!")
- Natural images.
- PCA, ICA, NMF, etc.
- What about *transformations* of natural images?
- Can we *learn* how images change?

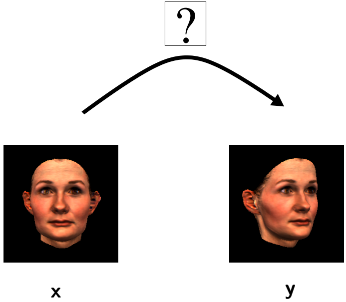
Bi-partite models



$$E(\mathbf{y}, \mathbf{h}) = -\sum_{jk} w_{jk} y_j h_k$$

- A common modeling framework for unsupervised learning:
- Let data and hidden variables populate a bi-partite network.
- Define $p(\mathbf{y}) = \sum_{\mathbf{h}} \frac{1}{Z} \exp(-E(\mathbf{y}, \mathbf{h}))$
- Think of $\mathbf{h}$ and $\mathbf{y}$ as binary vectors for now.

Learning data *transformations*



- How can we learn *transformations* of data?
- (...while keeping the statistics of the data in mind!)
- **Idea:** *Condition* on the input image.
- Hidden units become 'mapping' units that encode transformations.

Mapping units

- Make weights a function of the inputs: $w_{jk}(\mathbf{x}) = \sum_i w_{ijk} x_i$
- We get: $E(\mathbf{y}, \mathbf{h}; \mathbf{x}) = -\sum_{ijk} w_{ijk} x_i y_j h_k$
- Now model the **joint conditional** over data and hiddens as

$$p(\mathbf{y}, \mathbf{h}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp(-E(\mathbf{y}, \mathbf{h}; \mathbf{x}))$$

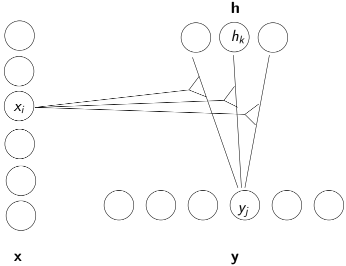
$$Z(\mathbf{x}) = \sum_{\mathbf{y},\mathbf{h}} \exp(-E(\mathbf{y}, \mathbf{h}; \mathbf{x}))$$

- From this we get the **conditional marginal:**

$$p(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{h}} p(\mathbf{y}, \mathbf{h}|\mathbf{x})$$

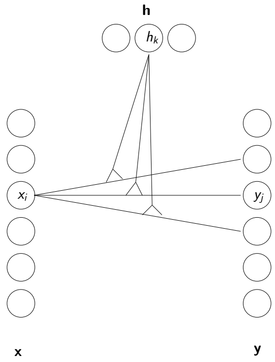
- (Hinton, Lang 1985), (He, Zemel 2004).

Two views: 1. Input-dependent filters



- Conditional RBM. *Potentials* depend on inputs:
- A CRF, that *learns* its features.
- Input-dependent filters.

Two views: 2. Modulated regression



- Modulated regression. Functions, modulated by hidden units.
- Exponential mixture of experts (with weight sharing).
- Mapping units provide a *factorial* code for transformations.

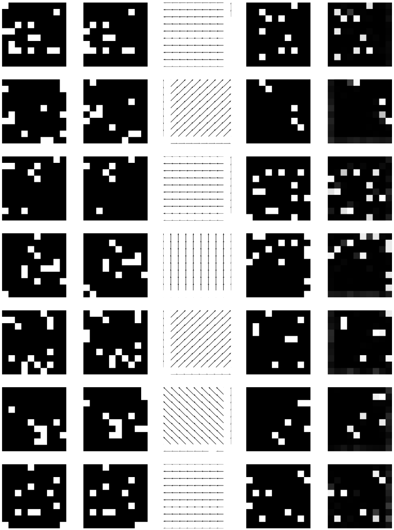
Training and inference

- For training, maximize $L(W) = \sum_\alpha \log p(\mathbf{y}^\alpha | \mathbf{x}^\alpha)$
- Gradient:

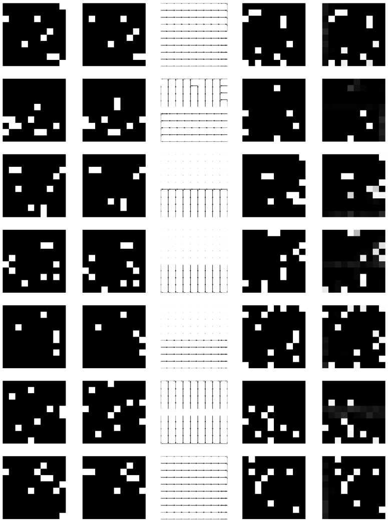
$$\frac{\partial L}{\partial W} = \sum_\alpha \left[\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{y}^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}) \frac{\partial E(\mathbf{h}, \mathbf{y}^{\boldsymbol{\alpha}})}{\partial W} - \sum_{\mathbf{h},\mathbf{y}} p(\mathbf{y}, \mathbf{h}|\mathbf{x}^{\boldsymbol{\alpha}}) \frac{\partial E(\mathbf{h}, \mathbf{y})}{\partial W} \right]$$

- Use contrastive divergence: "Don't learn anything we won't need at test time!"
- Inference: Compute $p(\mathbf{h}|\mathbf{y}, \mathbf{x})$. Easy, because of conditional independence.
- Optical flow is represented *implicitly* in the model (as a binary vector).
- But we can visualize what the model 'thinks' by plotting where it wants each pixel to go *the most*:

Example



Factorial flowfields

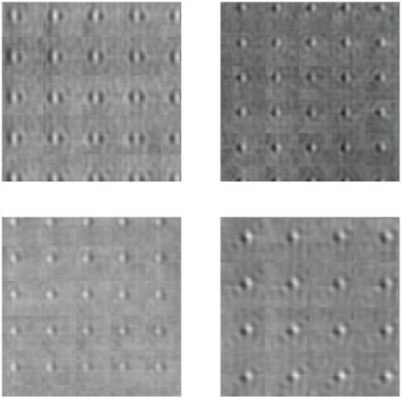


Exponential family

- Instead of using binary units, we can use any distribution from the **exponential family** as the conditionals for $\mathbf{h}$ and/or $\mathbf{y}$.
- Conditional analog of (Welling, et al. 2005).
- With Gaussian outputs, we get *structured output regression.*
- With Gaussian hiddens, we get a *'continuum-of-experts'* model.

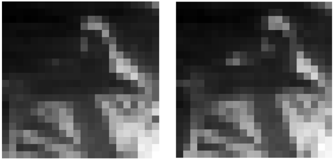
Flowfields on natural images

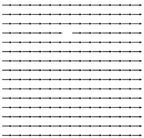
- Watching a few hours of television broadcast yields:



Pooled Reichardt detectors

- Hidden units learn *correlation patterns* in $\mathbf{x}$ and $\mathbf{y}$.
- Spatial pooling facilitates generalization and noise suppression:





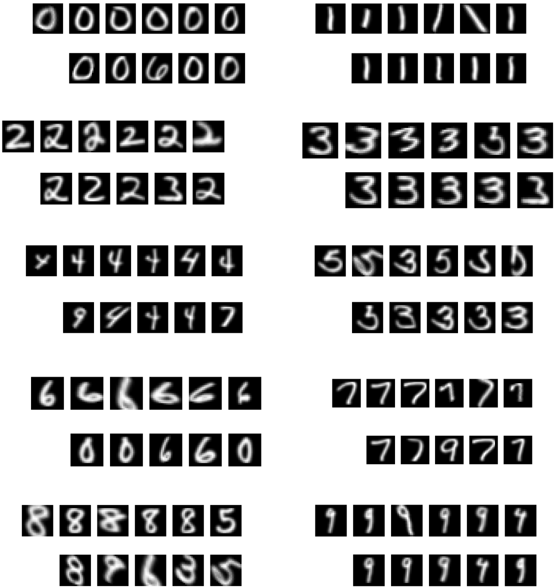
Dealing with large images

- Computational complexity: $\#\text{inputs} \times \#\text{hiddens} \times \#\text{outputs}$
- (Currently...) too large for big images.
- Also: Many transformation are local – they could occur anywhere in the image.
- Solution: Define the model as a 'product of patch-perts'...
- Local fields and weight-sharing.
- Neural network pre-mappings: Do dimensionality reduction first. (But *learn* the whole thing!)
- Define $w_{jk}(\mathbf{x}) = f_{jk}(\mathbf{x})$

Transformation distance

- After learning a transformation, we can define a metric that is invariant with respect to the transformation.
- To measure similarities between (new) cases $\mathbf{x}$ and $\mathbf{y}$:
  1. Set $\hat{\mathbf{h}} = \arg\max_{\mathbf{h}} p(\mathbf{h}|\mathbf{x}, \mathbf{y})$
  2. Set $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}, \hat{\mathbf{h}})$
  3. Define $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{y} - \hat{\mathbf{y}}\|$,
- One-step-reconstruction error. ('CD at test-time')
- No knowledge included!

Some nearest neighbors



k-nearest neighbors on affine digits

- After learning a metric, we can use it, for example, to do classification.

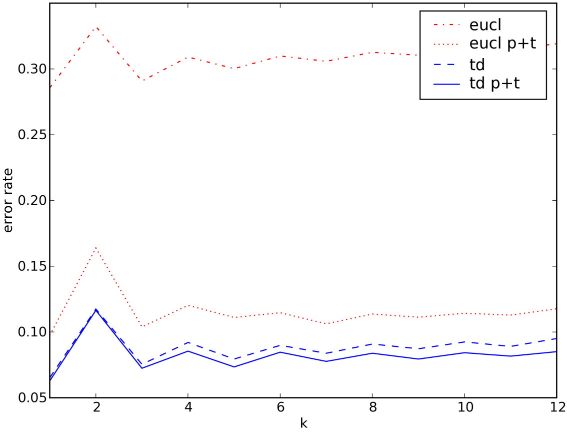
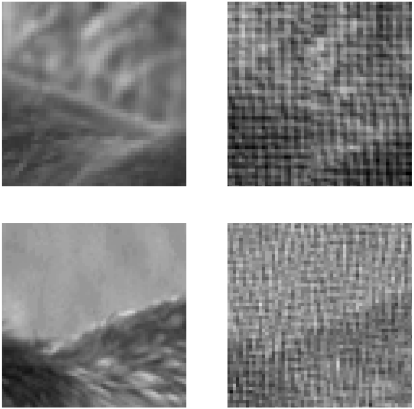


Image analogies

- After learning a transformation, we can ask the model to do 'the same' on a different image. (Hertzmann, et al., 2001)

Image analogies

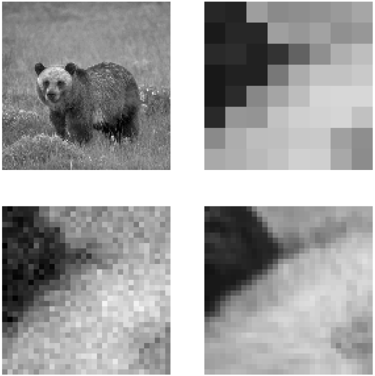


Discriminative de-noising

- If we just know how to *corrupt* images, we can **learn** to 'de-corrupt' them!

Discriminative super-resolution

- Another image-analogy problem:
- Given a **low-resolution** image, compute the corresponding **high-resolution** image.
- Use, for example, as a way to *zoom* into an image.



Discussion and further applications

- No kernels: Learn online, learn fast, learn on large data-sets.
- Several extensions exist: For example, 'Gated Autoencoders.'
- Transfer learning.
- Learning to cluster.
- Temporal de-noising.
- Video compression.
- Stereo, depth.