Combining Supervised and Unsupervised Learning (and the Ladder Network)

Tapani Raiko

Aalto University

19 August 2015

Motivation

Deep learning today:

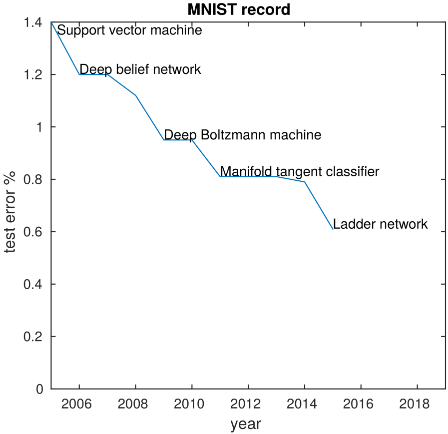
- Mostly about pure supervised learning
- Requires a lot of labeled data: expensive to collect

Deep learning in the future:

- Unsupervised, more human-like

"We expect unsupervised learning to become far more important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object."
–LeCun, Bengio, Hinton, Nature 2015

Motivation: Ladder network

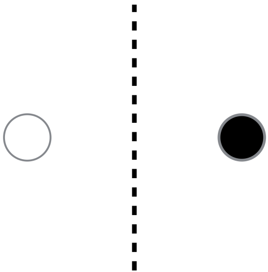


Yearly progress in permutation-invariant MNIST.

A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko. Semi-Supervised Learning with Ladder Network. ArXiv, July 2015.

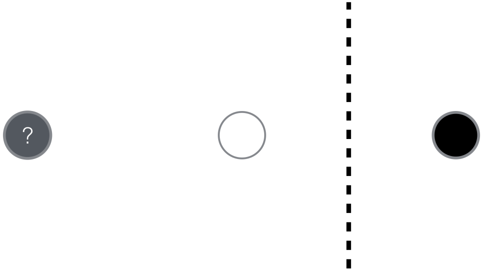
Table of Contents

Semisupervised learning



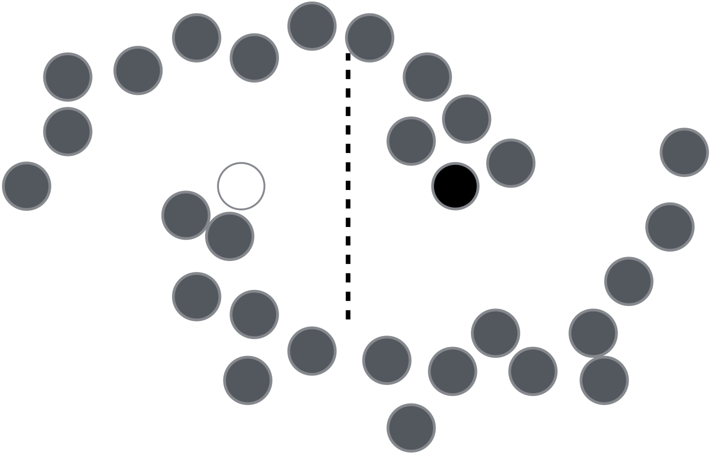
How can unlabeled data help in classification?
Example: Only two data points with labels.

Semisupervised learning



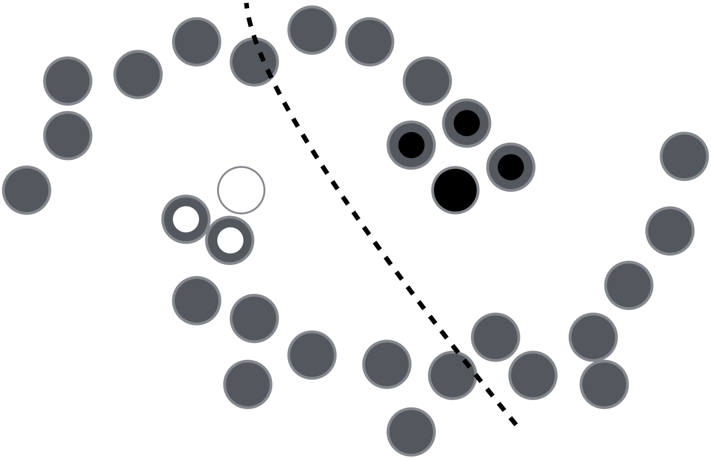
How would you label this point?

Semisupervised learning



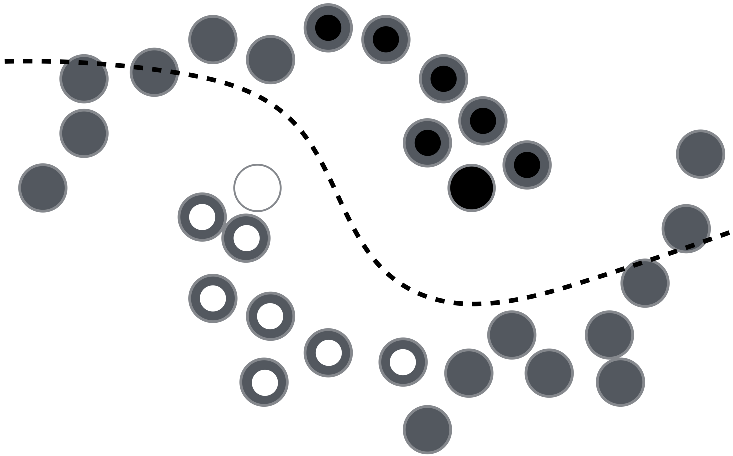
What if you see all the unlabeled data?

Semisupervised learning



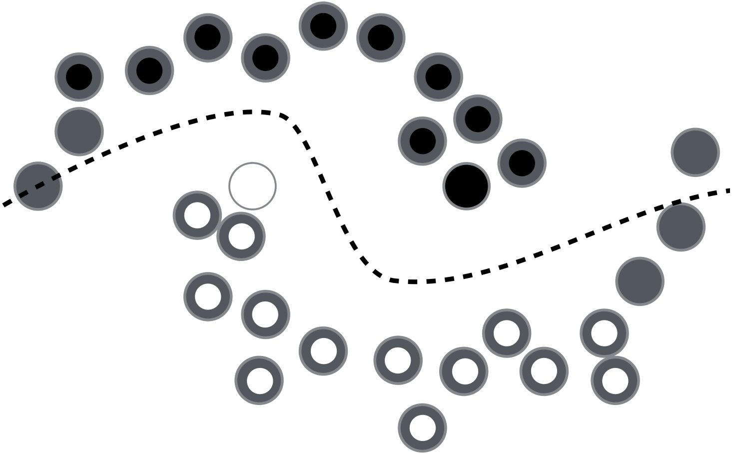
Labels are homogenous in densely populated space.

Semisupervised learning



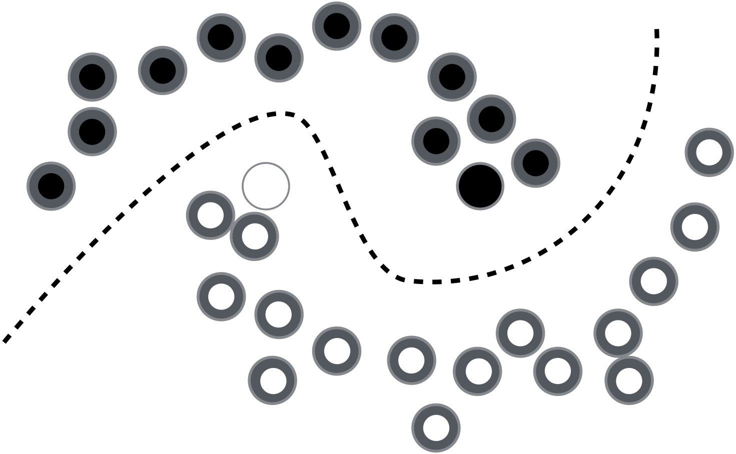
Labels are homogenous in densely populated space.

Semisupervised learning



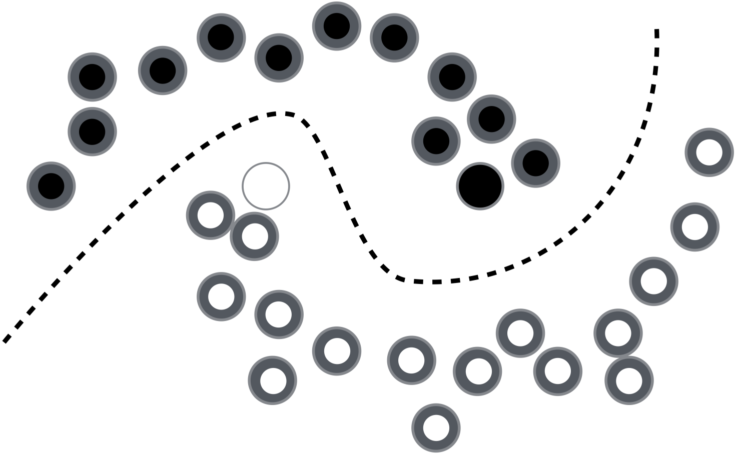
Labels are homogenous in densely populated space.

Semisupervised learning



Labels are homogenous in densely populated space.

Semisupervised learning



Labels are homogenous in densely populated space.

Semisupervised learning

Labeled data: $\{\mathbf{x}_t, y_t\}_{1 \leq t \leq N}$.
Unlabeled data: $\{\mathbf{x}_t\}_{N+1 \leq t \leq M}$.
Often labeled data is scarce, unlabeled data is plentiful: $N \ll M$.

Early works (McLachlan, 1975; Titterington et al., 1985) modelled $P(\mathbf{x}|y)$ as clusters.
Unlabeled data affects the shape and size of clusters.
Use Bayes theorem $P(y|\mathbf{x}) \propto P(\mathbf{x}|y)P(y)$ to classify.

Aalto University

How about $P(y|\mathbf{x})$ directly?

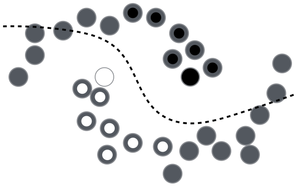
Modelling $P(\mathbf{x}|y)$ is inefficient when real task is $P(y|\mathbf{x})$.

Idea? Assign probabilistic labels $q(y_t) = P(y_t|\mathbf{x}_t)$ to unlabeled inputs $\mathbf{x}_t$, and train $P(y|\mathbf{x})$ with them. However, there is no effect as the gradient vanishes:

$$\mathbb{E}_{q(y)}\left[\frac{\partial}{\partial\boldsymbol{\theta}}\log P(y \mid \mathbf{x})\right] = \int q(y)\frac{\frac{\partial}{\partial\boldsymbol{\theta}}P(y \mid \mathbf{x})}{P(y \mid \mathbf{x})}\mathrm{d}y$$
$$= \frac{\partial}{\partial\boldsymbol{\theta}}\int P(y \mid \mathbf{x})\mathrm{d}y = \frac{\partial}{\partial\boldsymbol{\theta}}1 = 0.$$

There are ways to adjust the assigned labels $q(y_t)$ to make them count.

Adjusting assigned labels $q(y_t)$ (1/2)



Label propagation (Szummer and Jaakkola, 2003)

- Nearest neighbours tend to have the same label.
- Propagate labels to their neighbours and iterate.

Pseudo-labels (Lee, 2013)

- Round probabilistic labels $q(y_t)$ towards 0/1 gradually during training.

Adjusting assigned labels $q(y_t)$ (2/2)

Co-training (Blum and Mitchell, 1998)

- Assumes multiple views on $\mathbf{x}$, say $\mathbf{x} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)})$.
- Train a separate classifier $P(y \mid \mathbf{x}^{(j)})$ for each view.
- For unlabeled data, the true label is the same for each view.
- Combine individual $q^{(j)}(y_t)$ into a joint $q(y_t)$ and feed it as target to each classifier.

Part of **Ladder network** (Rasmus et al., 2015)

- Corrupt input $\mathbf{x}_t$ with noise to get $\tilde{\mathbf{x}}_t$.
- Train $P(y|\tilde{\mathbf{x}})$ with a target from clean $q(y_t) = P(y_t|\mathbf{x}_t)$.

Aalto University

Table of Contents

Unsupervised learning

Data is just $\mathbf{x}'$, not input-output pairs $\mathbf{x}, \mathbf{y}$.
Possible goals:

- Model $P(\mathbf{x}')$, or
- Representation $f : \mathbf{x}' \rightarrow \mathbf{h}$.

Comparisons to supervised learning $P(\mathbf{y}|\mathbf{x})$:

- See data as $\mathbf{x}' = \mathbf{y}$, model $P(\mathbf{y}|\mathbf{x} = \varnothing)$
- No right output $\mathbf{y}$ given, invent your own output $\mathbf{h}$
- Concatenate inputs and outputs to $\mathbf{x}' = [\mathbf{x}; \mathbf{y}]$, prepare to answer any query, including $P(\mathbf{y}|\mathbf{x})$.

From here on, data is just $\mathbf{x}$. Notation $\mathbf{x}'$ was used to avoid confusion.

Aalto University

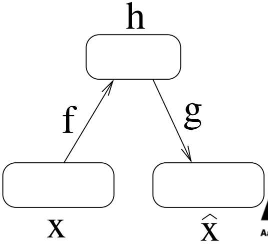
Approaches to unsupervised learning (1/2)

Besides kernel density estimation, virtually all unsupervised learning approaches use variables **h**.

- Discrete h (cluster index, hidden state of HMM, map unit of SOM)
- Binary vector **h** (most Boltzmann machines)
- Continuous vector **h** (PCA, ICA, NMF, sparse coding, autoencoders, state-space models, ...)

Vocabulary:

- Encoder function $f : \mathbf{x} \to \mathbf{h}$
- Decoder function $g : \mathbf{h} \to \hat{\mathbf{x}}$
- Reconstruction $\hat{\mathbf{x}}$



Approaches to unsupervised learning (2/2)

Often the encoder function $f : \mathbf{x} \rightarrow \mathbf{h}$ is implicit:

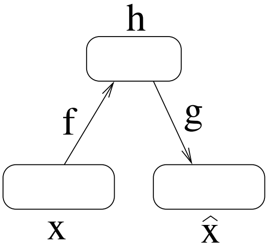
- Nearest cluster center $f(\mathbf{x}) = \arg\min_h \mathrm{D}(\mathbf{x}, \mathbf{c}_h)$
- Bayesian inference in a generative model, e.g. maximum a posteriori $f(\mathbf{x}) = \arg\max_{\mathbf{h}} P(\mathbf{x}|\mathbf{h})P(\mathbf{h})$

In complex models, exact inference is often impossible.
Approximate inference might hurt learning.

Autoencoders have an explicit encoder function $f(\cdot)$, which makes learning complex models easier:
Just backpropagation!

Aalto University

PCA as an autoencoder (1/2)



Assume linear encoder and decoder:
$f(\mathbf{x}) = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}$
$g(\mathbf{h}) = \mathbf{W}^{(2)}\mathbf{h} + \mathbf{b}^{(2)}$
PCA solution minimizes criterion $C = \mathbb{E}\left[\|\mathbf{x} - \hat{\mathbf{x}}\|^2\right]$.
Note: Solution is not unique, even if restricting $\mathbf{W}^{(2)} = \mathbf{W}^{(1)\top}$.

PCA as an autoencoder (2/2)

Just learning the identity mapping $g(f(\cdot)) = I(\cdot)$?
$\hat{\mathbf{x}} = g(f(\mathbf{x})) = \left(\mathbf{W}^{(2)}\mathbf{W}^{(1)}\right)\mathbf{x} + \left(\mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)}\right)$
We get $\hat{\mathbf{x}} = \mathbf{x}$ when $\mathbf{W}^{(2)} = (\mathbf{W}^{(1)})^{-1}$ and $\mathbf{b}^{(2)} = -\mathbf{W}^{(2)}\mathbf{b}^{(1)}$.
So any encoder with an invertible $\mathbf{W}^{(1)}$ is optimal.

How to make the autoencoding problem harder?

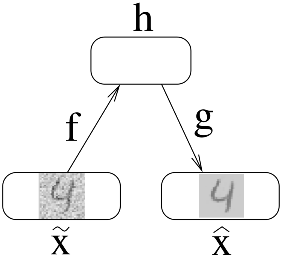
Regularized autoencoders

Regularization avoids learning the identity function:

- Bottleneck autoencoder (limit dimensionality of $\mathbf{h}$) (Bourlard and Kamp, 1988, Oja, 1991)
- Sparse autoencoder (penalize activations of $\mathbf{h}$) (Ranzato et al., 2006, Le et al., 2011)
- **Denoising autoencoder** (inject noise to input $\mathbf{x}$) (Vincent et al., 2008)
- Contractive autoencoder (penalize Jacobian of $f(\cdot)$) (Rifai et al., 2011)
- Sometimes also weight sharing $\mathbf{W}^{(2)} = \mathbf{W}^{(1)\top}$.

Aalto University

Denoising autoencoder (Vincent et al., 2008)



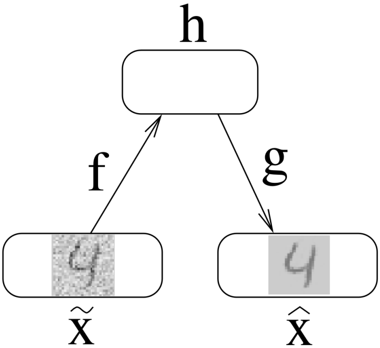
Feed corrupted inputs $\tilde{\mathbf{x}} \sim c(\tilde{\mathbf{x}}|\mathbf{x})$

- Additive noise $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}$ where e.g. $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$
- Salt noise $\tilde{\mathbf{x}} = \mathbf{m} \odot \mathbf{x}$ or $\tilde{x}_i = m_i x_i$
  where binary $m_i \sim \text{Bernoulli}(p)$
- Masking noise $\tilde{\mathbf{x}} = [\mathbf{m} \odot \mathbf{x}; \mathbf{m}]$

Train $\hat{\mathbf{x}} = g(f(\tilde{\mathbf{x}}))$ to minimize reconstruction error,
e.g. $C = \mathbb{E}\left[\|\hat{\mathbf{x}} - \mathbf{x}\|^2\right]$.

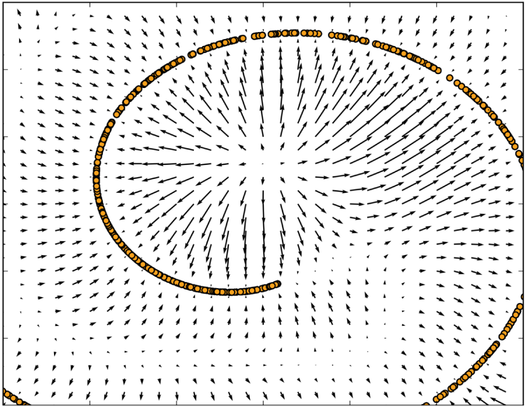
Aalto University

Denoising autoencoder



Basic encoder $\mathbf{h} = f(\tilde{\mathbf{x}}) = \Phi\left(\mathbf{W}^{(1)}\tilde{\mathbf{x}} + \mathbf{b}^{(1)}\right)$
and decoder $\hat{\mathbf{x}} = g(\mathbf{h}) = \mathbf{W}^{(2)}\mathbf{h} + \mathbf{b}^{(2)}$.
Deep autoencoder: both f and g multi-layered.

What does denoising autoencoder learn?



To point $g(f(\cdot))$ towards higher probability.

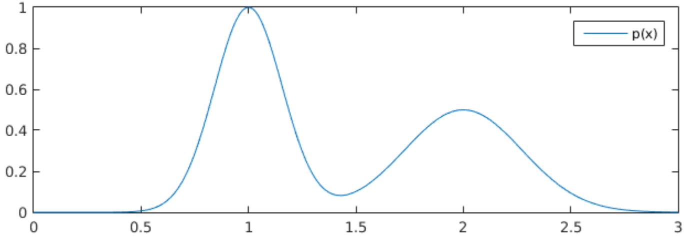
Image from (Alain and Bengio, 2014)

Table of Contents

Denoising versus probabilistic modelling

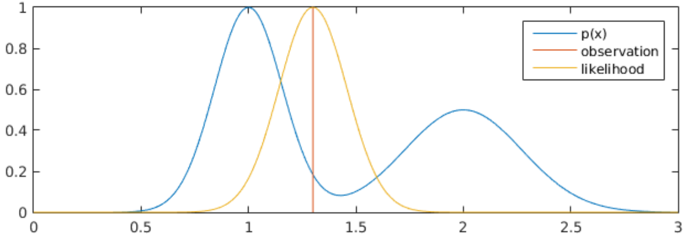
- We noted that denoising models are much easier to train than probabilistic models.
  Trainable by basic back-propagation.
- There is a strong connection between the two:
  Models can be converted into each other.

Probability to denoising



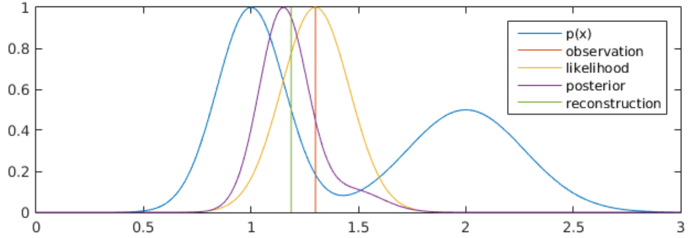
Given: Model $P(x)$ and observation $\tilde{x} = x + \mathrm{noise}$.
Noise distribution known.
Task: Find $\hat{x} = \arg\min \mathbb{E}_x \left[(x - \hat{x})^2 \right]$.
Solution: Compute the posterior $P(x \mid \tilde{x})$,
use its center of gravity as reconstruction $\hat{x}$.

Probability to denoising



Given: Model $P(x)$ and observation $\tilde{x} = x + \mathrm{noise}$.
Noise distribution known.
Task: Find $\hat{x} = \arg\min \mathbb{E}_x \left[(x - \hat{x})^2 \right]$.
Solution: Compute the posterior $P(x \mid \tilde{x})$,
use its center of gravity as reconstruction $\hat{x}$.

Probability to denoising



Given: Model $P(x)$ and observation $\tilde{x} = x + \mathrm{noise}$.

Noise distribution known.

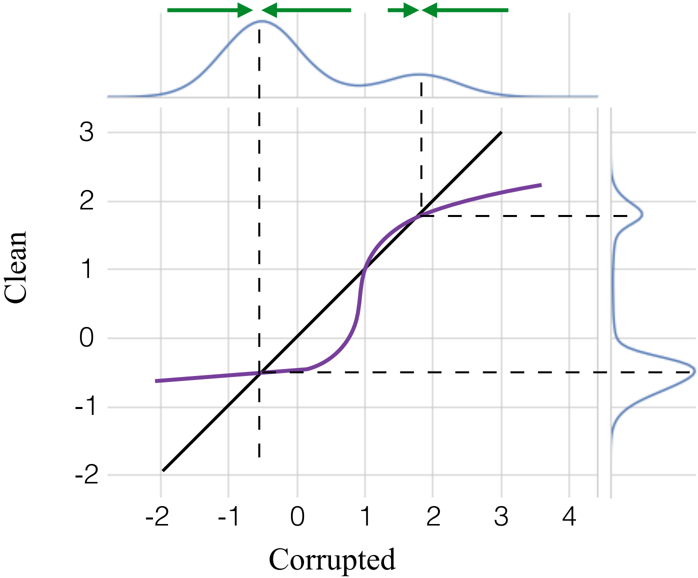
Task: Find $\hat{x} = \arg\min \mathbb{E}_x \left[(x - \hat{x})^2\right]$.

Solution: Compute the posterior $P(x \mid \tilde{x})$,

use its center of gravity as reconstruction $\hat{x}$.

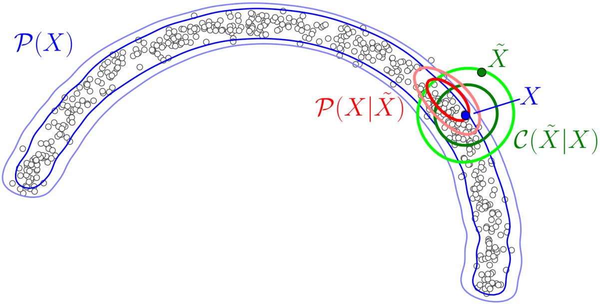
Aalto University

Probability to denoising



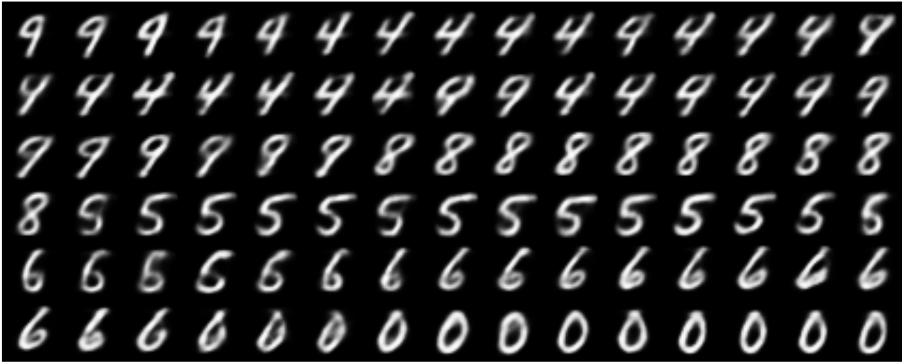
Denoising to probability

(Generative Stochastic Networks, Bengio et al., 2014)



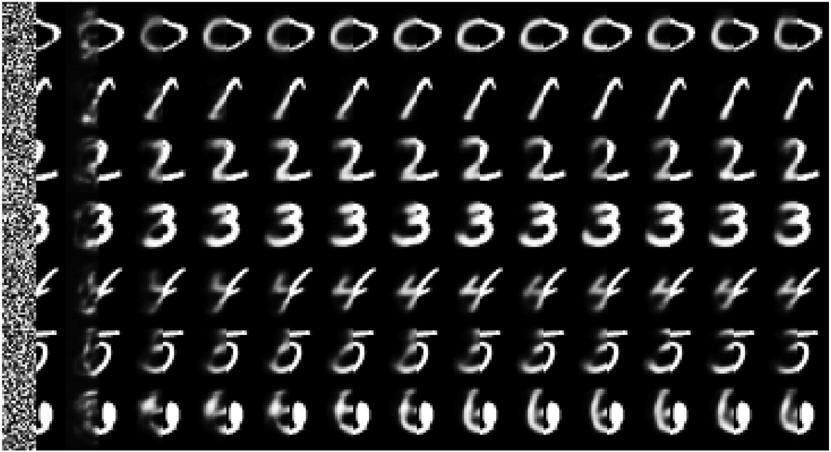
Markov chain alternating between corruption $C(\tilde{X}|X)$ and denoising $P(X|\tilde{X})$.
Theoretical result: Stationary distribution is $P(X)$.

Denoising to probability (Bengio et al., 2014)



Generating samples from the Markov chain.

Denoising to probability (Bengio et al., 2014)



Reconstructing the left half.

Table of Contents

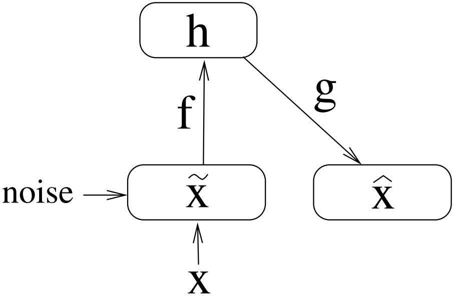
Aalto University

Layerwise pretraining

- Use unsupervised learning to construct representations layer by layer (Ballard, 1987).
- Breakthrough with Boltzmann machines (Hinton and Salakhutdinov 2006), starting deep learning boom.
- Presented here: Stacked denoising autoencoders

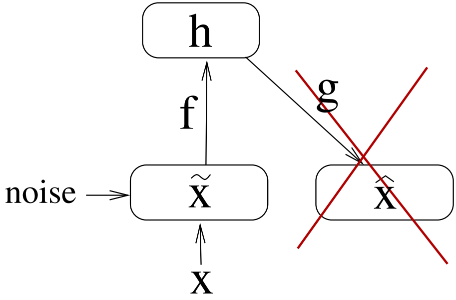
Aalto University

Layerwise pretraining



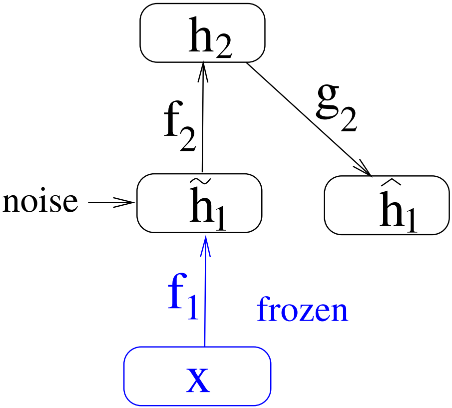
Phase 1: Denoising autoencoder.

Layerwise pretraining



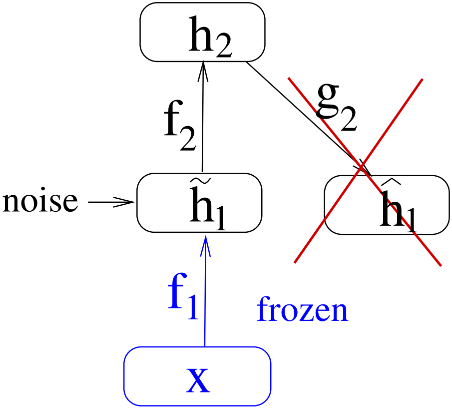
Toss away the decoder $g(\cdot)$.

Layerwise pretraining



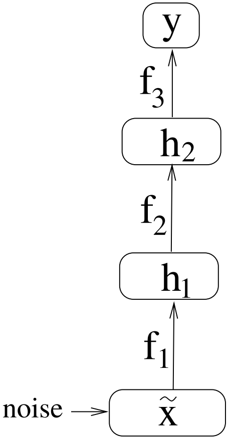
Phase 2: Stack another layer, keep the bottom fixed.

Layerwise pretraining



Toss away the second decoder $g_2(\cdot)$.

Supervised finetuning



Phase 3: Supervised finetuning with labels y.
Note: Encoder f of an autoencoder is the same mapping as used in supervised learning.

On details and invariance

What is average of images in the category Cat?
What is the average of Dog?

On details and invariance

Answer: both are just blurry blobs.

Autoencoder tries to learn a representation from which it can reconstruct the observations.

It cannot discard details: position, pose, lighting...

⇒ Not well compatible with supervised learning.

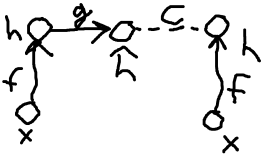
Table of Contents

Ladder network, main ideas

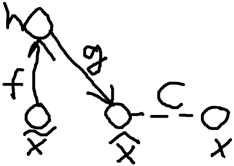
- Shortcut connections in an autoencoder network allow it to discard details.
- Learning in deep networks can be made efficient by spreading unsupervised learning targets all over the network.

Aalto University

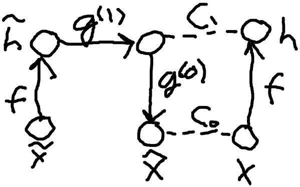
Combining DSS+DAE



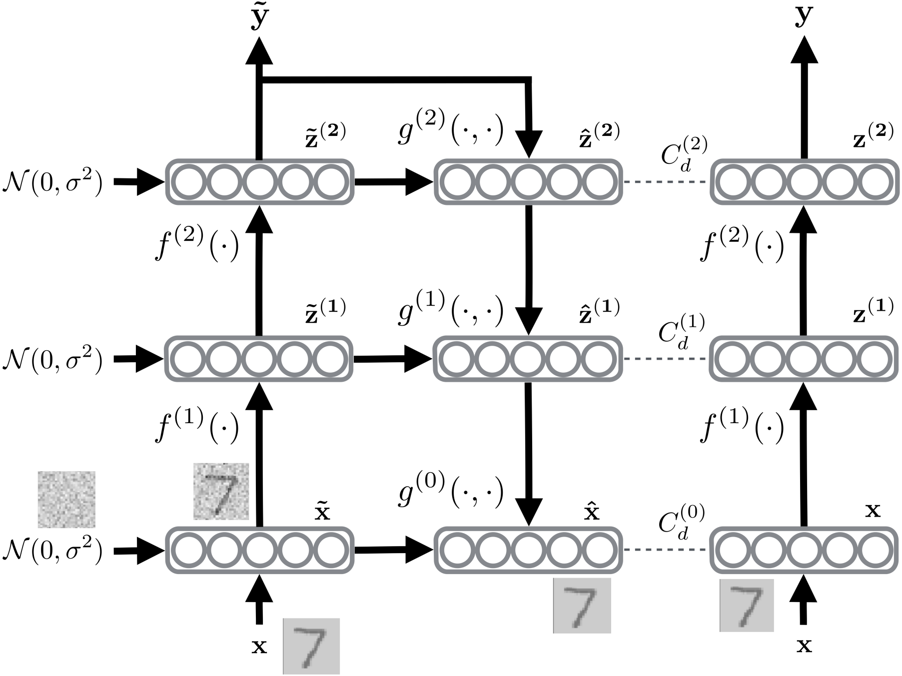
Denoising Source Separation
(Särelä and Valpola, 2005)



Denoising Autoencoder
(Vincent et al., 2008)

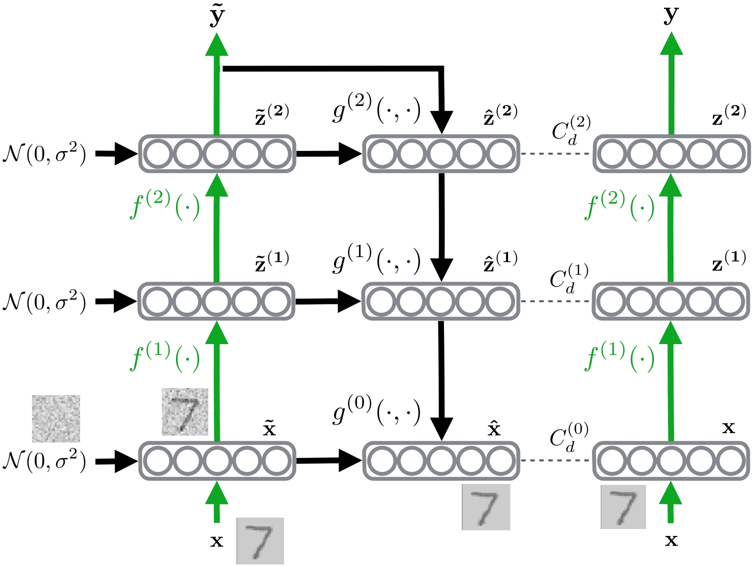


Ladder Network
(Valpola, 2015, Rasmus et al., 2015)

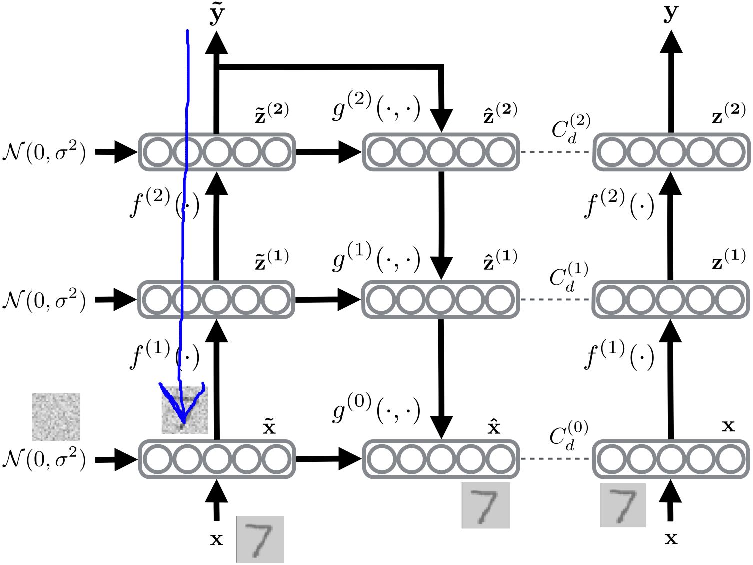


$\tilde{\mathbf{y}}$
$\mathbf{y}$
$\tilde{\mathbf{z}}^{(2)}$
$g^{(2)}(\cdot,\cdot)$
$\hat{\mathbf{z}}^{(2)}$
$C_d^{(2)}$
$\mathbf{z}^{(2)}$
$\mathcal{N}(0,\sigma^2)$
$f^{(2)}(\cdot)$
$f^{(2)}(\cdot)$
$\tilde{\mathbf{z}}^{(1)}$
$g^{(1)}(\cdot,\cdot)$
$\hat{\mathbf{z}}^{(1)}$
$C_d^{(1)}$
$\mathbf{z}^{(1)}$
$\mathcal{N}(0,\sigma^2)$
$f^{(1)}(\cdot)$
$f^{(1)}(\cdot)$
$g^{(0)}(\cdot,\cdot)$
$\tilde{\mathbf{x}}$
$\hat{\mathbf{x}}$
$C_d^{(0)}$
$\mathbf{x}$
$\mathcal{N}(0,\sigma^2)$
$\mathbf{x}$
$\mathbf{x}$

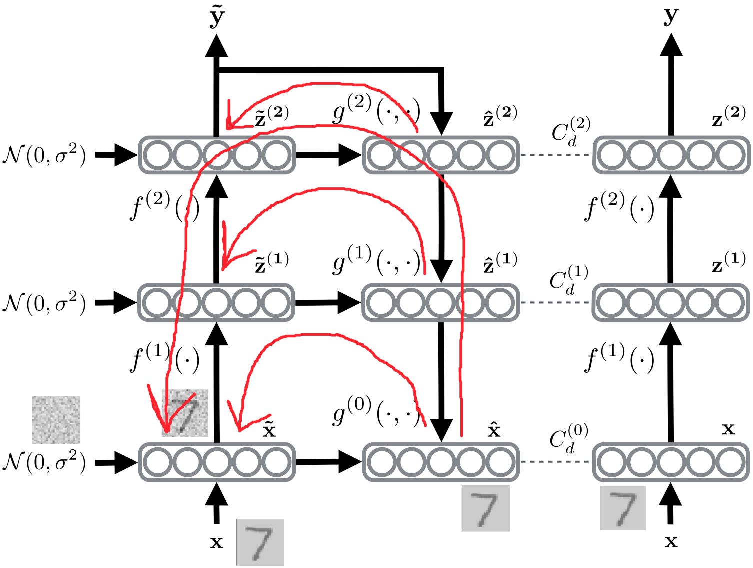
Same encoder $f(\cdot)$ used for corrupted and clean paths.



Supervised learning: Backprop from output $\tilde{\mathbf{y}}$.

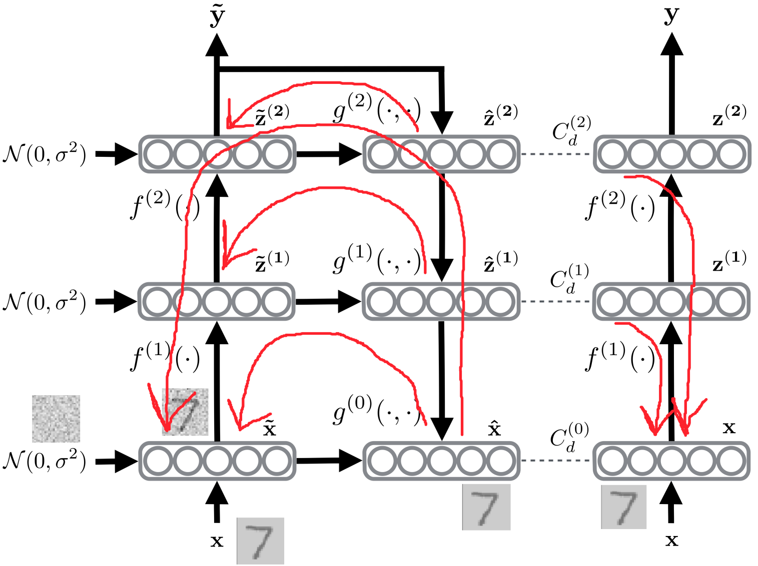


Unsupervised learning:
Several denoising autoencoders simultaneously.

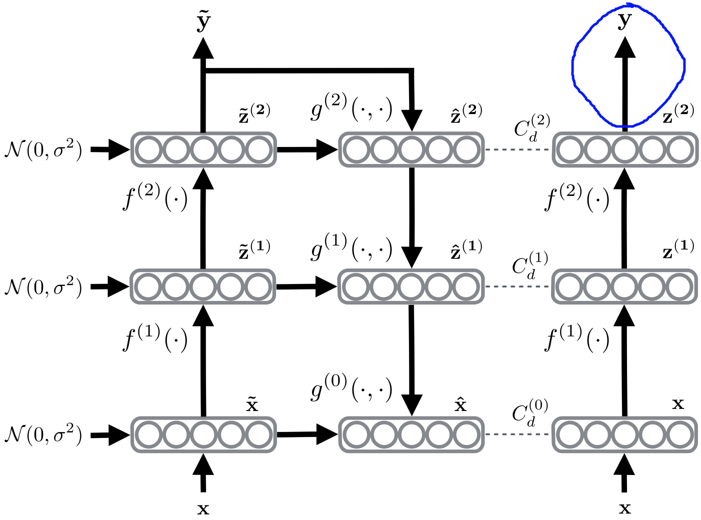


Aalto University

Unsupervised learning:
Produce robust representations (DSS aspect).



Read test output from the clean path. (Not used in training.)



Aalto University

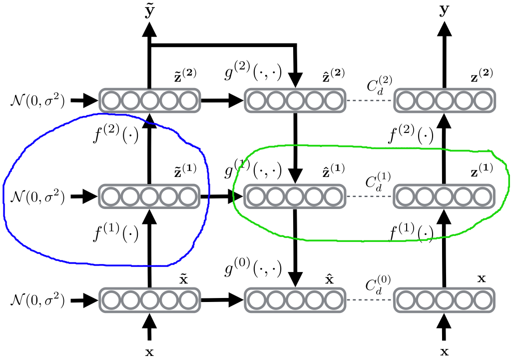
Training criterion

Only one phase of training: Minimize criterion C.

$$\mathrm{C} = -\log P(\tilde{\mathbf{y}} = \mathbf{y}_t | \mathbf{x}_t) + \sum_{l=0}^{L} \lambda_l \left\| \mathbf{z}^{(l)} - \hat{\mathbf{z}}_{\mathrm{BN}}^{(l)} \right\|^2$$

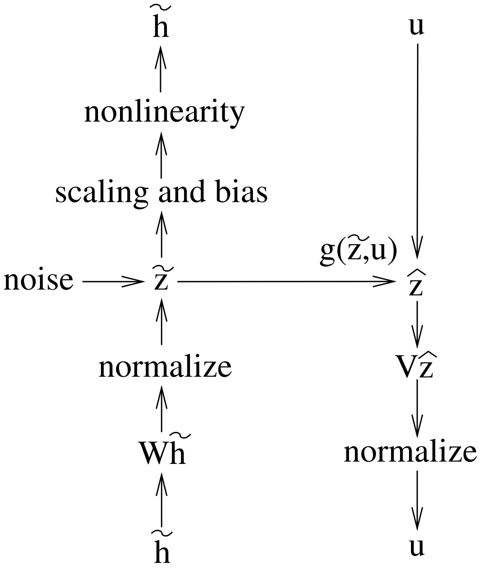
Aalto University

Scaling issues

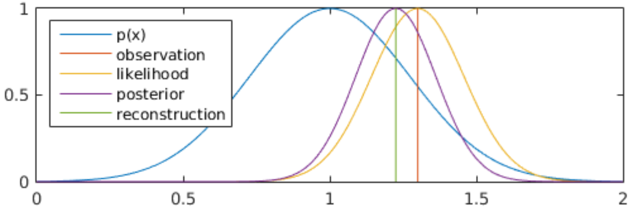


Issue 1: Doubling $\mathbf{W}^{(1)}$ and halving $\mathbf{W}^{(2)}$ decreases noise.
Issue 2: Collapsing $\mathbf{z}^{(1)} = \hat{\mathbf{z}}^{(1)} = 0$ eliminates cost $C^{(1)}$.
Solution: Batch normalization (Ioffe and Szegedy, 2015)

Some model details



Functional form of lateral connections?

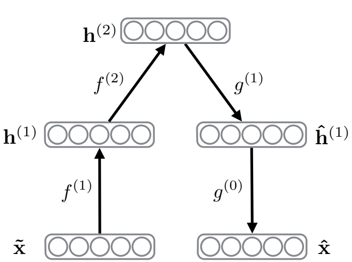


Gaussian model: $P(z) = \mathcal{N}(\mu, \sigma_p^2)$
Gaussian noise: $P(\tilde{z}|z) = \mathcal{N}(z, \sigma_n^2)$
Optimal denoising: $\hat{z} = \frac{\sigma_n^2}{\sigma_p^2+\sigma_n^2}\mu + \frac{\sigma_p^2}{\sigma_p^2+\sigma_n^2}\tilde{z}$
Top-down signal $\mathbf{u}$ corresponds to $P(z)$.
Modulating (multiplying, gating) $\tilde{z}$ corresponds to variance modelling.

Functional form of lateral connections?

How to interpret $\mathbf{u}$ modulating:
Does this detail in $\tilde{\mathbf{z}}$ fit in the big picture?
If yes, trust it at let it through to reconstruction $\hat{\mathbf{z}}$.
If not, filter it away as noise.

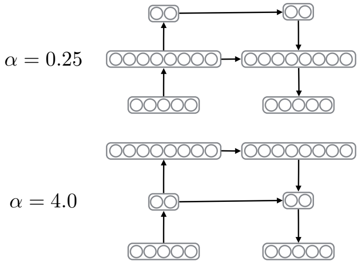
Analysis: Unsupervised learning

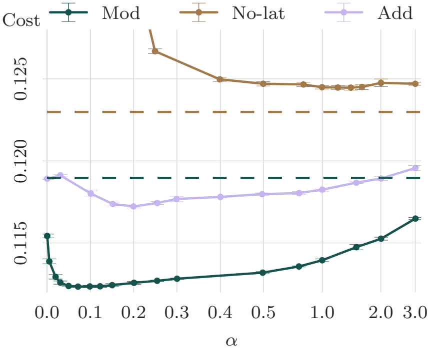


We compare deep denoising autoencoder and Ladder with additive or modulated lateral connections. Data is small natural image patches.

Analysis: Unsupervised learning

Denoising performance

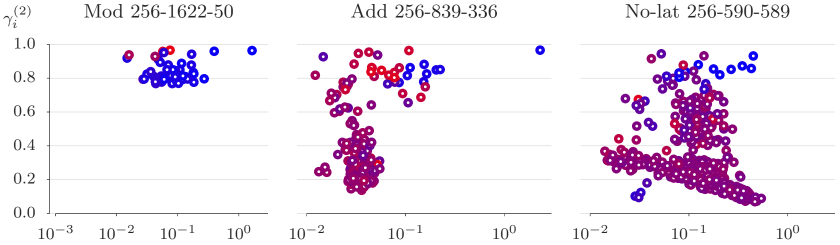




1 million parameters, vary sizes of layers.
Result: Modulated connections best.
Ladder needs fewer units on $\mathbf{h}^{(2)}$.

Analysis: Unsupervised learning

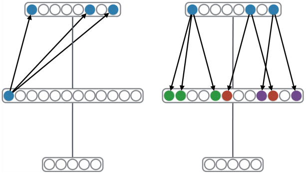
Translation invariance measure of units $\mathbf{h}^{(2)}$ as a function of unit significance.

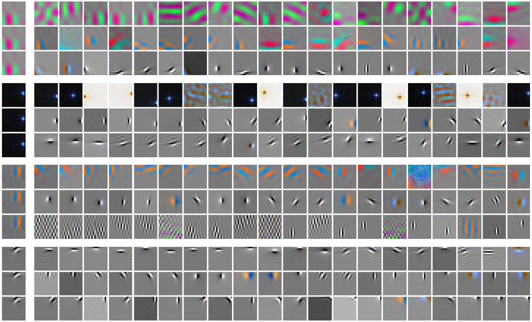


With modulated connections, all units become invariant.

Analysis: Unsupervised learning

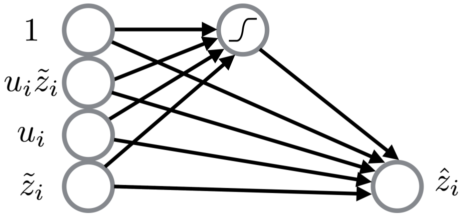
Learned pooling functions





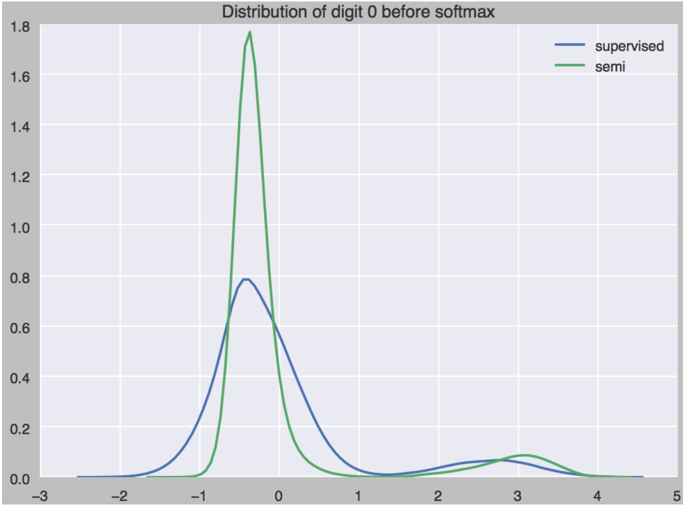
Each $\mathbf{h}^{(1)}$ unit belongs to several pooling groups.
Units $\mathbf{h}^{(2)}$ specialize to colour, orientation, location, ...

Small network for $\hat{z}_i = g(\tilde{z}_i, u_i)$



Each unit i has its own mini network with 9 parameters.
Few parameters compared to weight matrices.
Product $u_i \tilde{z}_i$ for modulating (variance modelling).
Nonlinearity for multimodal distributions.

Example of a multimodal distribution



Signal $z_0^{(L)}$ for digit 0 just before softmax.

Algorithm 1 Calculation of the output and cost function of the Ladder network

Require: $\mathbf{x}(n)$

  # Corrupted encoder and classifier

  $\tilde{\mathbf{h}}^{(0)} \leftarrow \tilde{\mathbf{z}}^{(0)} \leftarrow \mathbf{x}(n) + \texttt{noise}$

  for l = 1 **to** L **do**

    $\tilde{\mathbf{z}}^{(l)}_{\text{pre}} \leftarrow \mathbf{W}^{(l)} \tilde{\mathbf{h}}^{(l-1)}$

    $\tilde{\boldsymbol{\mu}}^{(l)} \leftarrow \texttt{batchmean}(\tilde{\mathbf{z}}^{(l)}_{\text{pre}})$

    $\tilde{\boldsymbol{\sigma}}^{(l)} \leftarrow \texttt{batchstd}(\tilde{\mathbf{z}}^{(l)}_{\text{pre}})$

    $\tilde{\mathbf{z}}^{(l)} \leftarrow \texttt{batchnorm}(\tilde{\mathbf{z}}^{(l)}_{\text{pre}}) + \texttt{noise}$

    $\tilde{\mathbf{h}}^{(l)} \leftarrow \texttt{activation}(\boldsymbol{\gamma}^{(l)} \odot (\tilde{\mathbf{z}}^{(l)} + \boldsymbol{\beta}^{(l)}))$

  end for

  $P(\tilde{\mathbf{y}} \mid \mathbf{x}) \leftarrow \tilde{\mathbf{h}}^{(L)}$

  # Clean encoder (for denoising targets)

  $\mathbf{h}^{(0)} \leftarrow \mathbf{z}^{(0)} \leftarrow \mathbf{x}(n)$

  for l = 1 **to** L **do**

    $\mathbf{z}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{W}^{(l)} \mathbf{h}^{(l-1)})$

    $\mathbf{h}^{(l)} \leftarrow \texttt{activation}(\boldsymbol{\gamma}^{(l)} \odot (\mathbf{z}^{(l)} + \boldsymbol{\beta}^{(l)}))$

  end for

  # Final classification:

  $P(\mathbf{y} \mid \mathbf{x}) \leftarrow \mathbf{h}^{(L)}$

  # Decoder and denoising

  for l = L **to** 0 **do**

    if l = L **then**

      $\mathbf{u}^{(L)} \leftarrow \texttt{batchnorm}(\tilde{\mathbf{h}}^{(L)})$

    else

      $\mathbf{u}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{V}^{(l)} \hat{\mathbf{z}}^{(l+1)})$

    end if

    $\forall i : \hat{z}^{(l)}_i \leftarrow g(\tilde{z}^{(l)}_i, u^{(l)}_i)$ # Eq. (1)

    $\forall i : \hat{z}^{(l)}_{i,\text{BN}} \leftarrow \frac{\hat{z}^{(l)}_i - \tilde{\mu}^{(l)}_i}{\tilde{\sigma}^{(l)}_i}$

  end for

  # Cost function C for training:

  $\text{C} \leftarrow 0$

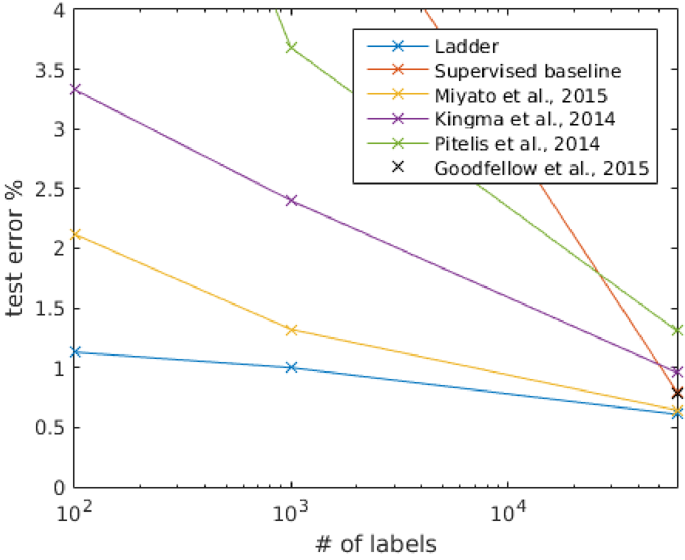
  if $t(n)$ **then**

    $\text{C} \leftarrow -\log P(\tilde{\mathbf{y}} = t(n) \mid \mathbf{x})$

  end if

  $\text{C} \leftarrow \text{C} + \sum_{l=0}^{L} \lambda_l \left\| \mathbf{z}^{(l)} - \hat{\mathbf{z}}^{(l)}_{\text{BN}} \right\|^2$

MNIST results



Thanks for listening!

Thanks to Antti Rasmus and Harri Valpola for some slides.

Aalto University

Possible exercises

- Follow the Theano tutorial on denoising autoencoders: `deeplearning.net/tutorial/dA.html`
- Examine and try the Ladder network code: `github.com/arasmus/ladder`