Summer school on semisupervised learning
Variational learning part 2
Auto-encoders for un- and semisupervised learning

Ole Winther

Dept for Applied Mathematics and Computer Science
Technical University of Denmark (DTU)

August 8, 2016

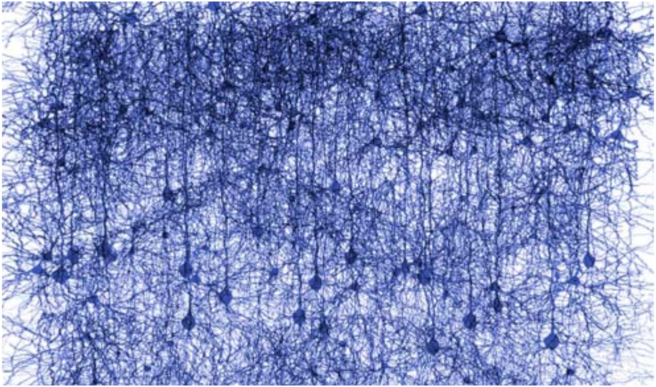
Objectives of talk

- What is unsupervised learning and
- why is it actually much more important than supervised learning.
- Auto-encoders for
- unsupervised and
- semisupervised learning

Part 1:
Unsupervised learning

Neural networks (NNs)

- Feedforward neural networks (FFNNs)
- Convolutional neural networks (CNNs)
- Recurrent Neural Networks (RNNs)
- Auto-encoders (AE)



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

Many thanks to Tapani Raiko for sharing slides!

Generative models for complex data

Unsupervised learning

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

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

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.

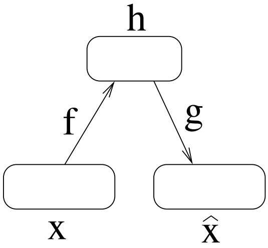
Approaches to unsupervised learning

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

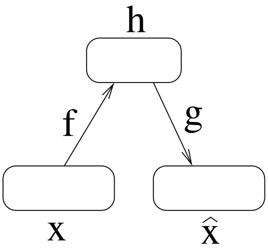
- Discrete h (cluster index, hidden state of HMM, map unit of SOM)
- Binary vector **h** (most Boltzmann machines)
- Continuous vector **h** (PCA, ICA, NMF, sparse coding, autoencoders, state-space models, ...)

Vocabulary:

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



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?

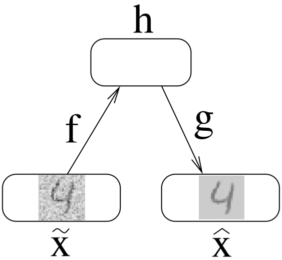
Part 2:
Auto-encoders

Regularized autoencoders

Regularization avoids learning the identity function:

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

Denoising autoencoder (Vincent et al., 2008)

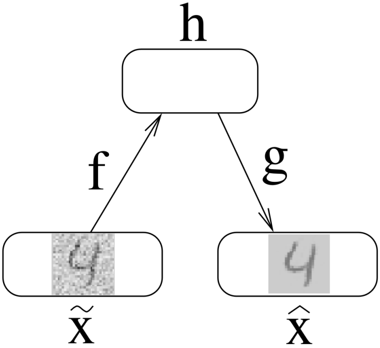


Feed corrupted inputs $\tilde{\mathbf{x}} \sim c(\tilde{\mathbf{x}}|\mathbf{x})$

- Additive noise $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$ 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 $\tilde{\mathbf{x}} = [\mathbf{m} \odot \mathbf{x}; \mathbf{m}]$

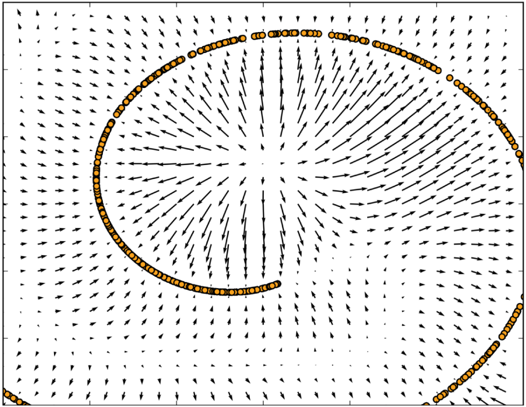
Train $\hat{\mathbf{x}} = g(f(\tilde{\mathbf{x}}))$ to minimize reconstruction error,
e.g. $C = \mathbb{E}\left[\|\hat{\mathbf{x}} - \mathbf{x}\|^2\right]$.

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.

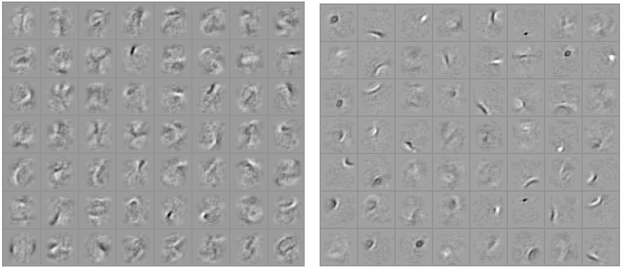
What does denoising autoencoder learn?



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

Image from (Alain and Bengio, 2014)

Comparison to training a classifier

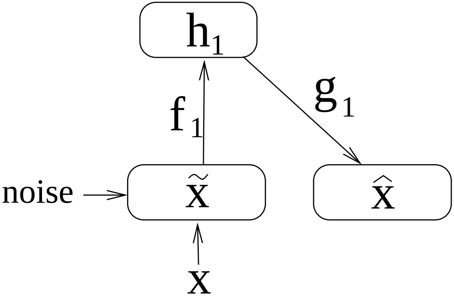


- Training a classifier (left), when you get the labels right, learning stops. $\Rightarrow$ Learned parameters are based on the information in labels:
  Less than #examples$\times$#classes bits.
- Training a denoising autoencoder (right), outputs are richer: #examples$\times$#dimensions.

Layerwise pretraining

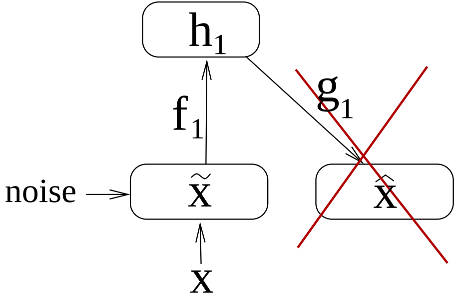
- Use unsupervised learning to construct representations layer by layer (Ballard, 1987).
- Breakthrough with Boltzmann machines (Hinton and Salakhutdinov 2006), starting deep learning boom.
- Presented here: Stacked denoising autoencoders

Layerwise pretraining



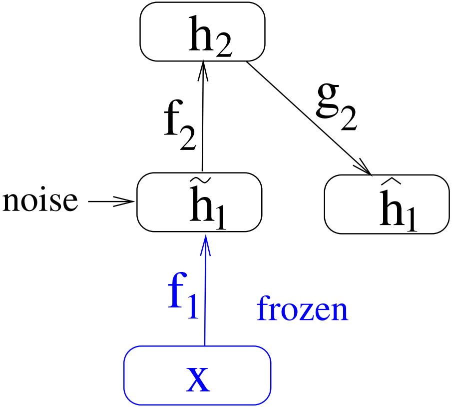
Phase 1: Denoising autoencoder.

Layerwise pretraining



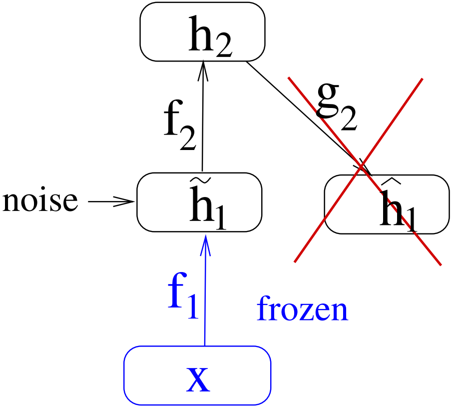
Toss away the decoder $g(\cdot)$.

Layerwise pretraining



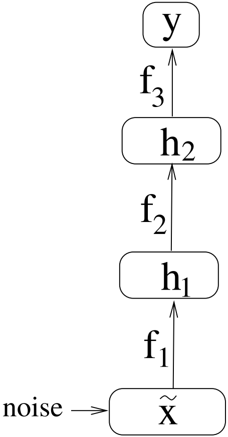
Phase 2: Stack another layer, keep the bottom fixed.

Layerwise pretraining



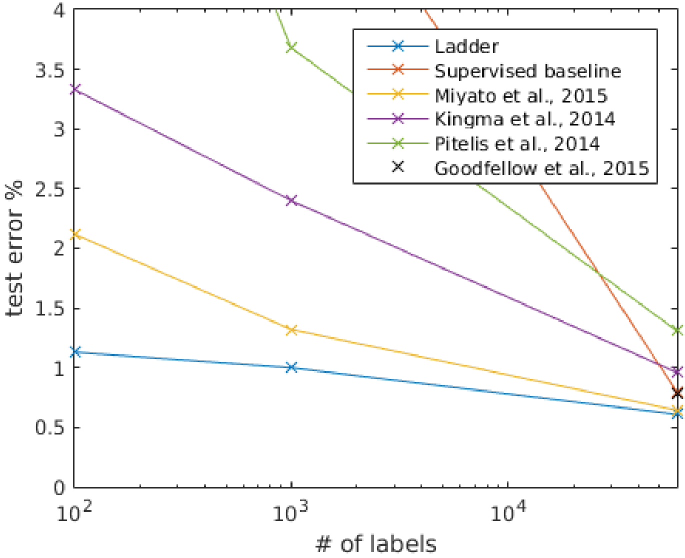
Toss away the second decoder $g_2(\cdot)$.

Supervised finetuning



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

Semi-supervised MNIST results



Part 4:

Deep generative models for un- and semi-supervised learning

Generative models for complex data

Latent variable model

- Latent variable model:

$$p_\theta(x,z) = \underbrace{p_\theta(x|z)}_{\text{FFNN}} \underbrace{p(z)}_{\mathcal{N}(z|0,I)}$$

Latent variable model

- Latent variable model:

$$p_\theta(x, z) = \underbrace{p_\theta(x|z)}_{\text{FFNN}} \underbrace{p(z)}_{\mathcal{N}(z|0,I)}$$

- Inference

$$z \sim p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

- Likelihood

$$p(x) = \int p(x|z)p(z)dz$$

- Learning

$$\theta_{\text{ML}} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log p_\theta(x_i)$$

- Difficult computations!

Deep variational auto-encoders

- Kingma+Welling 2013, Rezende et al 2013:
- Decoder: (example continuous observations)

$$p_\theta(x|z) = \mathcal{N}(x|\underbrace{\mu_\theta(z)}_{\text{FFNN}}, \underbrace{\mathrm{diag}(\sigma^2_\theta(z))}_{\text{FFNN}})$$

Deep variational auto-encoders

- Kingma+Welling 2013, Rezende et al 2013:
- Decoder: (example continuous observations)

$$p_\theta(x|z) = \mathcal{N}(x|\underbrace{\mu_\theta(z)}_{\text{FFNN}}, \underbrace{\mathrm{diag}(\sigma^2_\theta(z))}_{\text{FFNN}})$$

- Encoder

$$q_\phi(z|x) = \mathcal{N}(z|\underbrace{\mu_\phi(x)}_{\text{FFNN}}, \underbrace{\mathrm{diag}(\sigma^2_\phi(x))}_{\text{FFNN}})$$

Deep variational auto-encoders

- Kingma+Welling 2013, Rezende et al 2013:
- Decoder: (example continuous observations)

$$p_\theta(x|z) = \mathcal{N}(x|\underbrace{\mu_\theta(z)}_{\text{FFNN}}, \underbrace{\mathrm{diag}(\sigma^2_\theta(z))}_{\text{FFNN}})$$

- Encoder

$$q_\phi(z|x) = \mathcal{N}(z|\underbrace{\mu_\phi(x)}_{\text{FFNN}}, \underbrace{\mathrm{diag}(\sigma^2_\phi(x))}_{\text{FFNN}})$$

- Variational objective - optimize $\sum_i \mathcal{L}_{\theta,\phi}(x_i)$ wrt θ, ϕ:

$$\log p_\theta(x) \geq \mathcal{L}_{\theta,\phi}(x) = \int q_\phi(z|x) \log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} dz$$

$$= \underbrace{\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]}_{\mathbb{E}_q[\text{log likelihood}]} + \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p(z)}{q_\phi(z|x)}\right]}_{\text{regularization}}$$

Deep variational auto-encoders

- Variational objective - optimize $\sum_i \mathcal{L}_{\theta,\phi}(x_i)$ wrt θ, ϕ:

$$\log p_\theta(x) \geq \mathcal{L}_{\theta,\phi}(x) = \int q_\phi(z|x) \log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}\, dz$$

- Handle integration by sampling

$$\mathcal{L}_{\theta,\phi}(x) \approx \frac{1}{R}\sum_{r=1}^{R} \log \frac{p_\theta(x|z^r)p(z^r)}{q_\phi(z^r|x)},$$

$$z^r = \mu_\phi(x) + \sigma_\phi(x) \otimes \epsilon^r$$

Deep variational auto-encoders

- Variational objective - optimize $\sum_i \mathcal{L}_{\theta,\phi}(x_i)$ wrt θ, ϕ:

$$\log p_\theta(x) \geq \mathcal{L}_{\theta,\phi}(x) = \int q_\phi(z|x) \log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \, dz$$

- Handle integration by sampling

$$\mathcal{L}_{\theta,\phi}(x) \approx \frac{1}{R} \sum_{r=1}^{R} \log \frac{p_\theta(x|z^r)p(z^r)}{q_\phi(z^r|x)},$$

$$z^r = \mu_\phi(x) + \sigma_\phi(x) \otimes \epsilon^r$$

- Auto-encoder: map "$x \to z \to x$"
- This is a variational auto-encoder (VAE):

$$\text{Encoder} \quad z = \mu_\phi(x) + \sigma_\phi(x) \otimes \epsilon$$

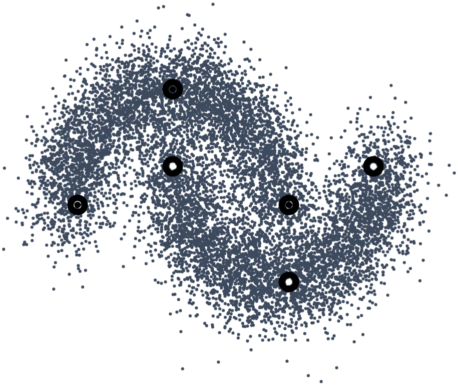
$$\text{Decoder} \quad p_\theta(x|z) = \mathcal{N}(x|\mu_\theta(z), \text{diag}(\sigma_\theta^2(z)))$$

Generative models for complex data

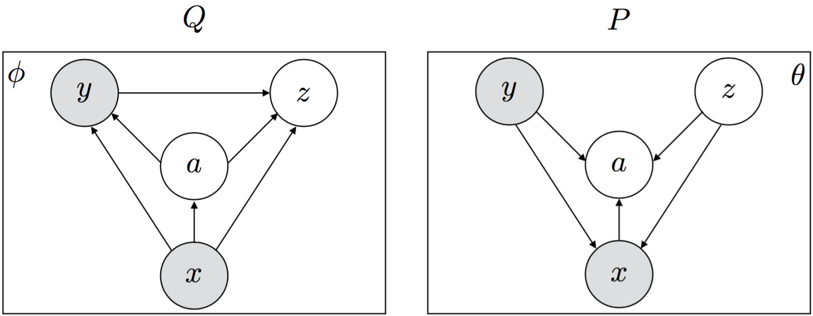
- Once model is trained we can make it dream up new digits by:

$$z \sim \mathcal{N}(z|0,1)$$
$$x \sim \mathcal{N}(x|\underbrace{\mu_\theta(z)}_{\text{FFNN}}, \underbrace{\mathrm{diag}(\sigma_\theta^2(z))}_{\text{FFNN}})$$

Semi-supervised learning



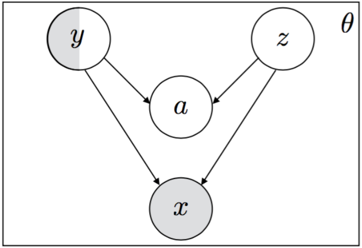
Auxiliary variable model for labeled data



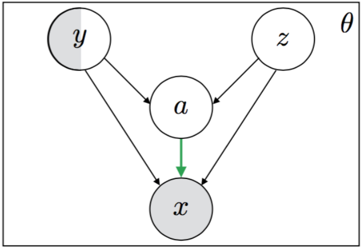
L Maaloe et al, ICML, 2016.

Skip deep generative model

P ADGM

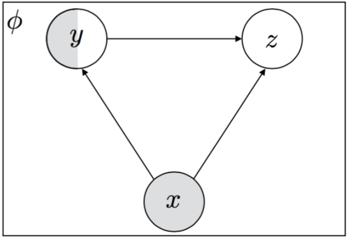


P SDGM

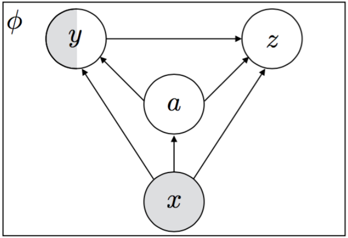


Kingma et al M1 and M2

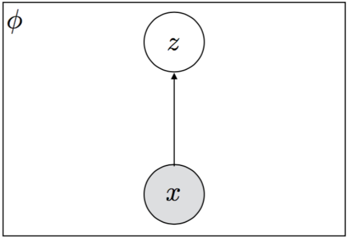
Q M2 (Kingma et al., 2014)



Q ADGM/SDGM



Q M1 (Kingma et al., 2014)



Semi-supervised benchmarks

	MNIST 100 labels	SVHN 1000 labels	NORB 1000 labels
M1+TSVM (Kingma et al., 2014)	11.82% (±0.25)	55.33% (±0.11)	18.79% (±0.05)
M1+M2 (Kingma et al., 2014)	3.33% (±0.14)	36.02% (±0.10)	
VAT (Miyato, 2015)	2.12 %	24.63 %	9.88 %
Ladder Network (Rasmus et al., 2015)	1.06% (±0.37)		
Auxiliary Deep Generative Model	**0.96% (±0.02)**	22.86 %	10.06% (±0.05)
Skip Deep Generative Model	1.32% (±0.07)	**16.61% (±0.24)**	**9.40% (±0.04)**

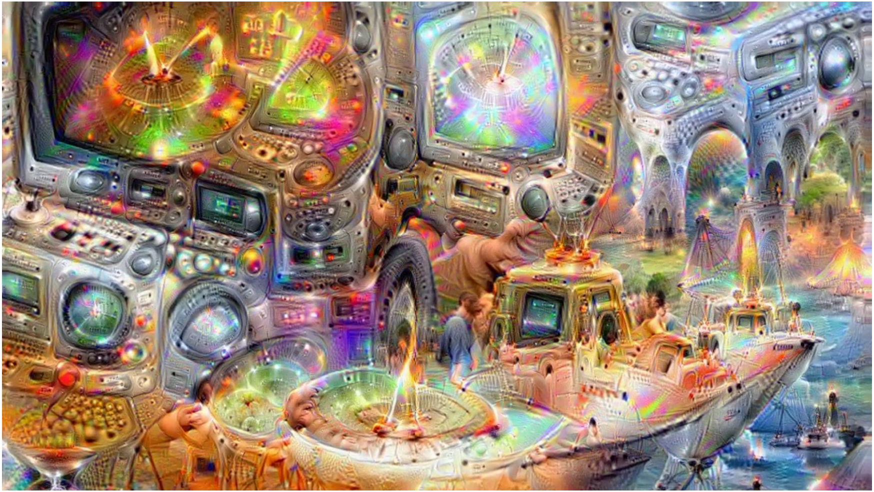
- Potential extensions: warm-up, batch normalisation, convolutional layers, combining with other improvements.
- The saga continues Salisman et al, Improved Techniques for Training GANs, 2016.

References

- Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. Biological cybernetics, 59(4-5), 291–294.
- E. Oja, Data Compression, Feature Extraction, and Autoassociation in Feedforward Neural Networks. In Proc. ICANN-91, Helsinki, Finland, June 1991, pp. 737–745, Elsevier.
- M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. "Efficient learning of sparse representations with an energy-based model." Proceedings of NIPS 2006.
- Le, Q. V., Karpenko, A., Ngiam, J., and Ng, A. Y. (2011). ICA with reconstruction cost for efficient overcomplete feature learning. In Advances in Neural Information Processing Systems (pp. 1017-1025).
- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR, 11, 3371–3408.
- Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In Proceedings of the 28th International Conference on Machine Learning (ICML-11) (pp. 833–840).
- Ballard, D. H. (1987). Modular learning in neural networks. In Proc. AAAI, pages 279–284.
- Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.
- Y. Bengio, E. Thibodeau-Laufer, G. Alain, and J. Yosinski. Deep generative stochastic networks trainable by backprop. Technical Report arXiv:1306.1091, 2014
- A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko. Semi-Supervised Learning with Ladder Network. In NIPS 2015.
- Särelä, J. and Valpola, H. (2005). Denoising source separation. JMLR, 6, 233–272.

- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR, 11, 3371–3408.

- H. Valpola. From neural PCA to deep unsupervised learning. arXiv preprint arXiv:1411.7783 (2014).
- Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167



Thanks!
Ole Winther