Deep generative models

Johan Suykens

KU Leuven, ESAT-STADIUS
Kasteelpark Arenberg 10
B-3001 Leuven (Heverlee), Belgium
Email: johan.suykens@esat.kuleuven.be
http://www.esat.kuleuven.be/stadius

Lecture 11

Overview

- Restricted Boltzmann machines (RBM)
- Deep Boltzmann machines
- Generative adversarial networks (GAN)

Boltzmann Machines (1)

- Minimize **energy function**:

$$E = -\sum_{i<j} w_{ij} s_i s_j + \sum_i \theta_i s_i$$

w_{ij} connection strength between units i and j; θ_i are thresholds;
$s_i = 1$ if unit i is on, and 0 otherwise.

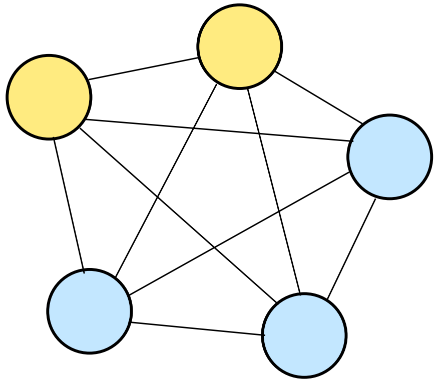
- Note [Hertz et al. 1991]: This is an energy function as in the Hopfield model, but one uses **stochastic units** (which also introduces the notion of temperature, related to magnetic materials and Ising models in physics). The deterministic dynamics are replaced by a stochastic rule $s_i = 1$ with probability $g(a_i)$ with a sigmoid function (related to Glauber dynamics)

$$g(a) = \frac{1}{1 + \exp(-2\beta\, a)} \quad \text{with } \beta = \frac{1}{k_B T}$$

with Boltzmann constant k_B and absolute temperate T and magnetic field $a_i = \sum_j w_{ij} s_j + a_{\text{ext}}$ with a_{ext} an external field.

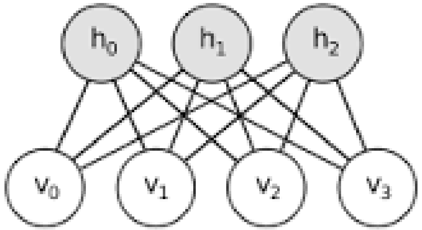
Boltzmann Machines (2)

- One has the property $\frac{P_\alpha}{P_\beta} = \exp(-(E_\alpha - E_\beta)/T)$ for states α, β. P_α probability being in α-th state with energy E_α [Ackley et al. 1985].
- One considers a network consisting of **visible units** (which are inputs and outputs) and **hidden units** (indicated in different colors in the figure).



Training of a Boltzmann Machine in this way is known to be very difficult. This led to the study of Restricted Boltzmann Machines (RBM) (which is also related to harmoniums proposed by [Smolensky 1986]).

Restricted Boltzmann Machines (RBM)



- Markov random field, bipartite graph, stochastic binary units
  Layer of visible units v and layer of hidden units h
  No hidden-to-hidden connections
- Energy:

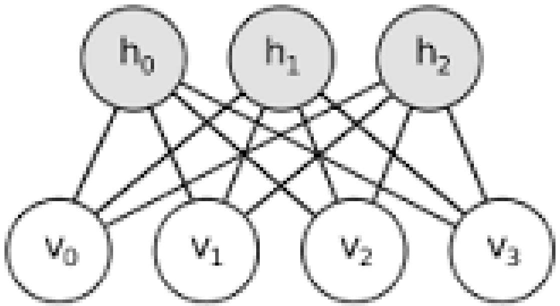
$$E(v,h;\theta) = -v^T W h - b^T v - a^T h \quad \text{with} \quad \theta = \{W, b, a\}$$

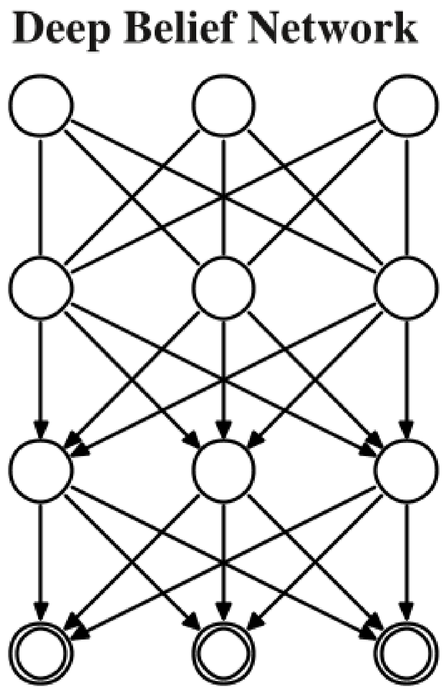
Joint distribution:

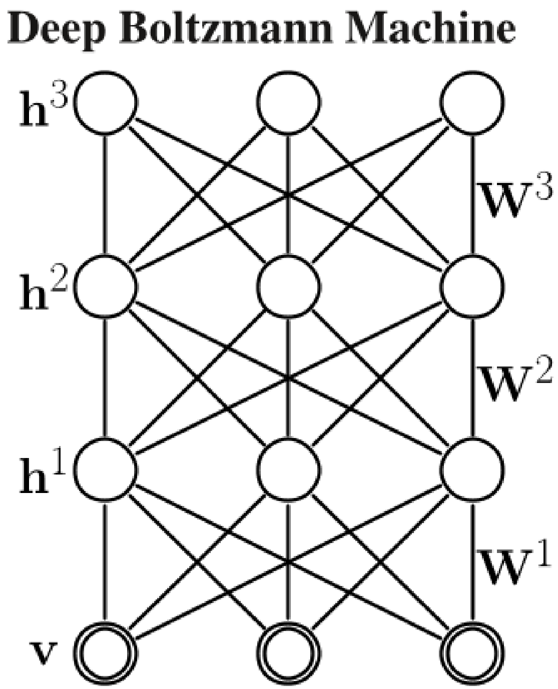
$$P(v,h;\theta) = \frac{1}{Z(\theta)} \exp(-E(v,h;\theta))$$

with partition function $Z(\theta) = \sum_v \sum_h \exp(-E(v,h;\theta))$
[Hinton, Osindero, Teh, Neural Computation 2006]

RBM and deep learning





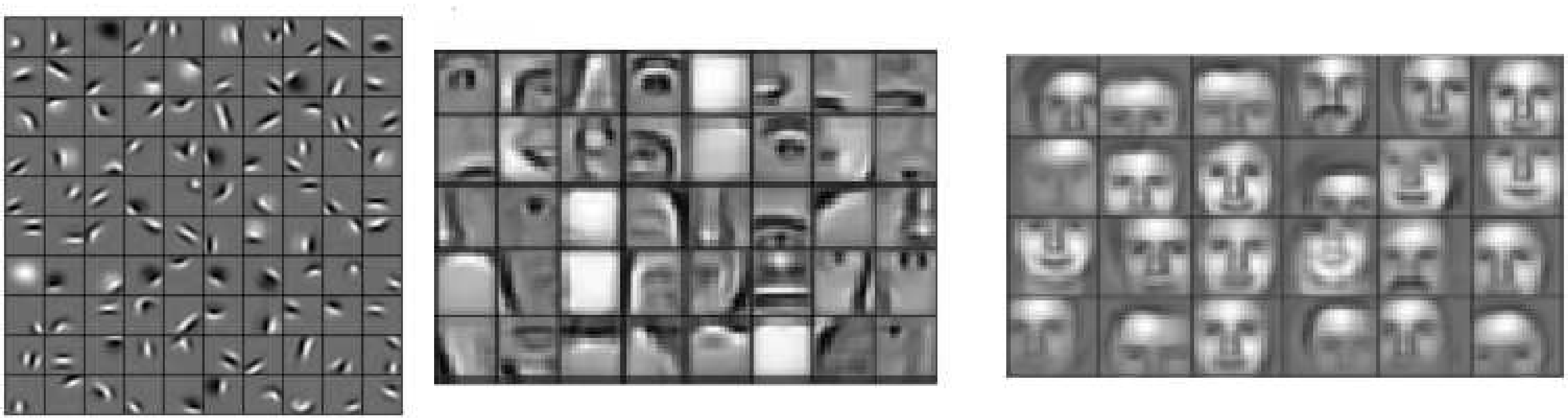


$p(v, h)$

$p(v, h^1, h^2, h^3, ...)$

[Hinton et al., 2006; Salakhutdinov, 2015]

Convolutional Deep Belief Networks



Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks [Lee et al. 2011]

Energy function

- RBM:

$$E = -v^T W h$$

- Deep Boltzmann machine (two layers):

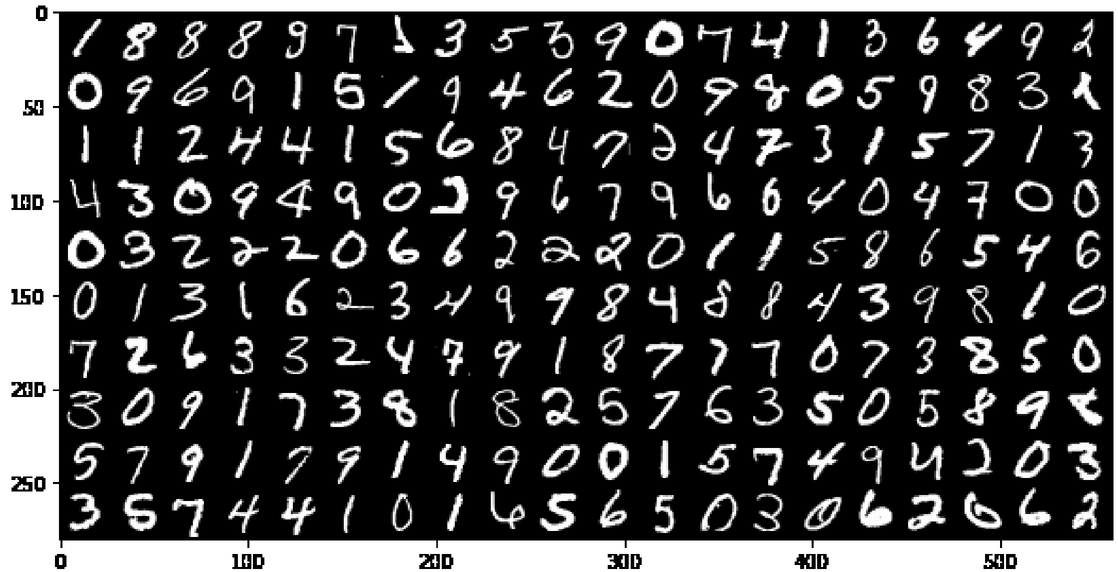
$$E = -v^T W^1 h^1 - {h^1}^T W^2 h^2$$

- Deep Boltzmann machine (three layers):

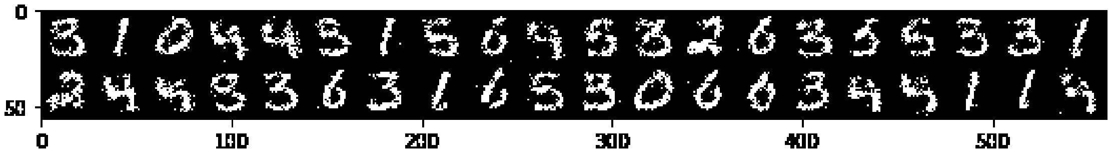
$$E = -v^T W^1 h^1 - {h^1}^T W^2 h^2 - {h^2}^T W^3 h^3$$

RBM: example on MNIST

MNIST training data:



Generating new images:



source: https://www.kaggle.com/nicw102168/restricted-boltzmann-machine-rbm-on-mnist

RBM training (1)

Thanks to the special bipartite structure, explicit **marginalization** is possible:

$$P(v;\theta) = \frac{1}{Z(\theta)} \sum_h \exp(-E(v,h;\theta)) = \frac{1}{Z(\theta)} \exp(b^T v) \prod_j (1+\exp(a_j + \sum_i W_{ij} v_j))$$

with $v_i \in \{0,1\}$, $h_i \in \{0,1\}$.

Conditional distributions:

$$P(h|v;\theta) = \prod_j p(h_j|v) \text{ with } p(h_j = 1|v) = \sigma(\sum_i W_{ij} v_i + a_j)$$

and

$$P(v|h;\theta) = \prod_i p(v_i|h) \text{ with } p(v_i = 1|h) = \sigma(\sum_j W_{ij} h_j + b_i)$$

with σ the sigmoid activation.

RBM training (2)

Given observations $\{v_n\}_{n=1}^{N}$, the **derivative of the log-likelihood** is

$$\begin{array}{rcl}
\frac{1}{N}\sum_n \frac{\partial \log P(v_n;\theta)}{\partial W_{ij}} & = & \mathbb{E}_{P_{\text{data}}}[v_i h_j] - \mathbb{E}_{P_{\text{model}}}[v_i h_j] \\
\frac{1}{N}\sum_n \frac{\partial \log P(v_n;\theta)}{\partial a_j} & = & \mathbb{E}_{P_{\text{data}}}[h_j] - \mathbb{E}_{P_{\text{model}}}[h_j] \\
\frac{1}{N}\sum_n \frac{\partial \log P(v_n;\theta)}{\partial b_i} & = & \mathbb{E}_{P_{\text{data}}}[v_i] - \mathbb{E}_{P_{\text{model}}}[v_i]
\end{array}$$

with

- **Data-dependent expectation** $\mathbb{E}_{P_{\text{data}}}[\cdot]$ (*form of Hebbian learning*):
  an expectation with respect to the data distribution $P_{\text{data}}(h, v; \theta) = P(h|v;\theta)P_{\text{data}}(v)$ with $P_{\text{data}}(v) = \frac{1}{N}\sum_n \delta(v - v_n)$ the empirical distribution.

- **Model's expectation** $\mathbb{E}_{P_{\text{model}}}[\cdot]$ (*unlearning*):
  an expectation with respect to the distribution defined by the model $P(v, h; \theta) = \frac{1}{Z(\theta)} \exp(-E(v, h; \theta))$.

RBM training (3)

Exact maximum likelihood learning is intractable (due to computation of $\mathbb{E}_{P_{\mathrm{model}}}[\cdot]$). In practice, **Contrastive Divergence** (CD) algorithm [Hinton 2002]:

$$\Delta W = \alpha(\mathbb{E}_{P_{\mathrm{data}}}[vh^T] - \mathbb{E}_{P_T}[vh^T])$$

with α learning rate and P_T a distribution defined by running a Gibbs chain initialized at the data for T full steps ($T = 1$, i.e. CD1 often in practice).

CD1 scheme:

1. Start Gibbs sampler $v^{(1)} := v_n$ and generate $h^{(1)} \sim P(h|v^{(1)})$
2. After obtaining $h^{(1)}$, generate $v^{(2)} \sim P(v|h^{(1)})$ (called fantasy data)
3. After obtaining $v^{(2)}$, generate $h^{(2)} \sim P(h|v^{(2)})$

with

$$\Delta W \propto (v_n {h^{(1)}}^T - v^{(2)} {h^{(2)}}^T)$$

Deep Boltzmann machine training (1)

Consider 3-layer Deep BM with **energy function** [Salakhutdinov 2015]:

$$E(v, h^1, h^2, h^3; \theta) = -v^T W^1 h^1 - {h^1}^T W^2 h^2 - {h^2}^T W^3 h^3$$

with unknown model parameters $\theta = \{W^1, W^2, W^3\}$.

The model assigns the following probability to a visible vector v:

$$P(v; \theta) = \frac{1}{Z(\theta)} \sum_{h^1, h^2, h^3} \exp(-E(v, h^1, h^2, h^3; \theta))$$

Deep Boltzmann machine training (2)

For training:

$$
\begin{aligned}
\frac{\partial \log P(v;\theta)}{\partial W^1} &= \mathbb{E}_{P_{\text{data}}}[v{h^1}^T] - \mathbb{E}_{P_{\text{model}}}[v{h^1}^T] \\
\frac{\partial \log P(v;\theta)}{\partial W^2} &= \mathbb{E}_{P_{\text{data}}}[h^1{h^2}^T] - \mathbb{E}_{P_{\text{model}}}[h^1{h^2}^T] \\
\frac{\partial \log P(v;\theta)}{\partial W^3} &= \mathbb{E}_{P_{\text{data}}}[h^2{h^3}^T] - \mathbb{E}_{P_{\text{model}}}[h^2{h^3}^T]
\end{aligned}
$$

Problem: the conditional distribution over the states of the hidden variables conditioned on the data is **no longer factorial**. For simplicity and speed one can **assume and impose a fully factorized distribution**, corresponding to a naive mean-field approximation [Salakhutdinov 2015].

Multimodal Deep Boltzmann Machine

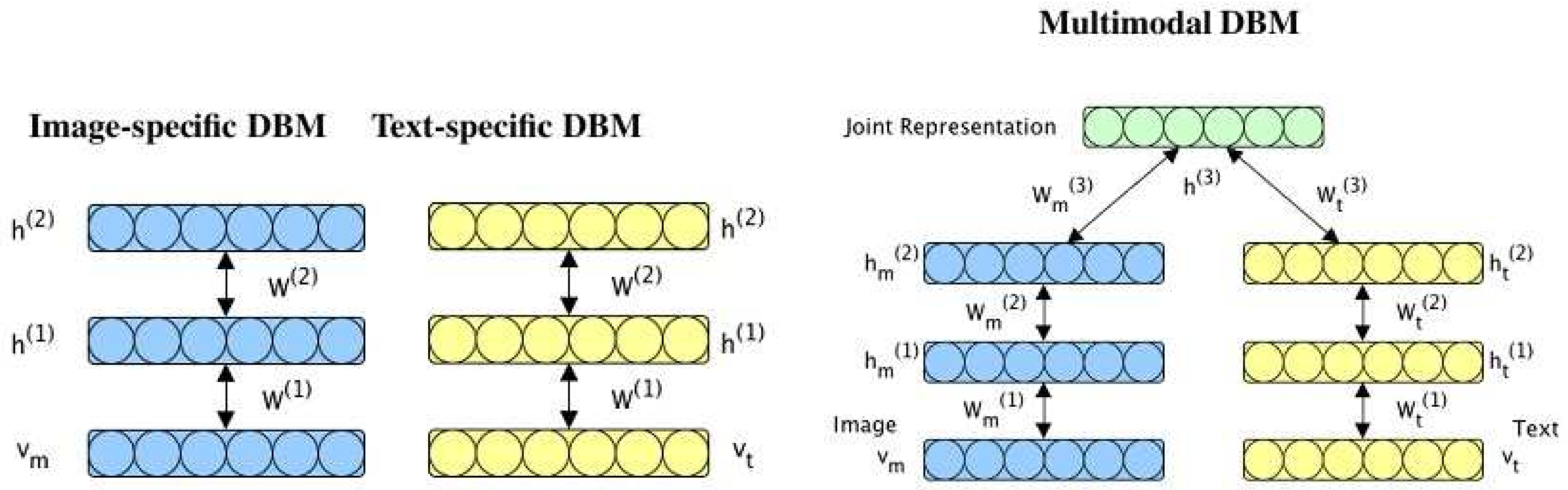


Figure 2: **Left:** Image-specific two-layer DBM that uses a Gaussian model to model the distribution over real-valued image features. **Middle:** Text-specific two-layer DBM that uses a Replicated Softmax model to model its distribution over the word count vectors. **Right:** A Multimodal DBM that models the joint distribution over image and text inputs.

From [Srivastava & Salakhutdinov 2014]

Wasserstein training of RBM (optimal transport) (1)

Wasserstein Distance: Consider a joint probability measure π with marginals p and q. The Wasserstein distance between p and q is defined as

$$W(p,q) = \min_{\pi \in \Pi(p,q)} \mathbb{E}_\pi d(x,x')$$

with $d(x,x')$ a distance metric.

Kontorovich duality: For p,q discrete distributions, the Wasserstein distance can be written as a linear program, with transport plan π:

$$\begin{array}{rll} W(p,q) & = \min_\pi & \sum_{x,x'} \pi(x,x') d(x,x') \\ & s.t. & \sum_{x'} \pi(x,x') = p(x), \sum_x \pi(x,x') = q(x') \\ & & \pi(x,x') \geq 0 \end{array}$$

The dual problem is

$$\begin{array}{rll} W(p,q) & = \max_{\alpha,\beta} & \sum_x \alpha(x) p(x) + \sum_{x'} \beta(x') q(x') \\ & s.t. & \alpha(x) + \beta(x') \leq d(x,x') \end{array}$$

Wasserstein training of RBM (optimal transport) (2)

γ-smoothed Wasserstein Distance:

$$W_\gamma(p,q) = \min_{\pi \in \Pi(p,q)} \mathbb{E}_\pi d(x,x') - \gamma H(\pi)$$

with $H(\pi)$ the **Shannon entropy** of π and $\Pi(p,q)$ the set of joint distributions with marginals $p(x)$, $q(x')$.

γ-smoothed Wasserstein Distance as optimization problem:

$$\begin{array}{rll} W_\gamma(p,q) & = \min_\pi & \sum_{x,x'} \pi(x,x')(d(x,x') + \gamma \log \pi(x,x')) \\ & s.t. & \sum_{x'} \pi(x,x') = p(x), \sum_x \pi(x,x') = q(x') \end{array}$$

From the Lagrangian one obtains:

$$W_\gamma(p,q) = \sum_x \alpha^\star p(x) + \sum_{x'} \beta^\star(x')q(x') - \gamma \sum_{x,x'} \exp(\frac{1}{\gamma}(\alpha^\star(x) + \beta^\star(x') - d(x,x')) - 1)$$

with $\alpha^\star = \gamma \log u(x)$, $\beta^\star(x') = \gamma \log v(x')$, $K(x,x') = \exp(-\frac{1}{\gamma} d(x,x') - 1)$, $u(x) \sum_{x'} K(x,x')v(x') = p(x)$, $\sum_x u(x)K(x,x')v(x') = q(x')$.

Wasserstein training of RBM (optimal transport) (3)

Take now p as the model distribution p_θ;
Take q as the data distribution $\hat{p} = \sum_i \frac{1}{N}\delta_{x_i}$ with data points x_i.

Use of γ-**smoothed Wasserstein Distance for training**:

$$\begin{aligned}\frac{\partial W_\gamma(p_\theta,\hat{p})}{\partial\theta} &= \sum_x \alpha^\star(x)\frac{\partial p_\theta(x)}{\partial\theta} \\ &= \mathbb{E}_{x\sim p_{\theta(x)}}[\alpha^\star(x)\frac{\partial \log p_\theta(x)}{\partial\theta}]\end{aligned}$$

In practice replace p_θ by a **sampling approximation** $\tilde{p}_\theta = \sum_{j=1}^{\tilde{N}} \frac{1}{\tilde{N}}\delta_{\tilde{x}_j}$. This yields

$$\frac{\partial W_\gamma(p_\theta,\hat{p})}{\partial\theta} \simeq \frac{1}{\tilde{N}}\sum_{j=1}^{\tilde{N}} \alpha^\star(\tilde{x}_j)\frac{\partial \log p_\theta(\tilde{x}_j)}{\partial\theta}$$

In the **RBM case** (taking visible $v = x$): $p_\theta(x) = \sum_h p_\theta(x,h)$ with

$$p_\theta(x,h) = \frac{1}{Z(\theta)}\exp(v^T W h + b^T x + a^T h)$$

A connection between RBM and Kernel PCA

Consider objective [Suykens 2017] (with feature map φ applied to x_i)

$$J(h_i \in \mathbb{R}^s, W) = -\sum_{i=1}^{N} \varphi(x_i)^T W h_i + \frac{\lambda}{2} \sum_{i=1}^{N} h_i^T h_i + \frac{1}{2} \mathrm{Tr}(W^T W)$$

Stationary points of J:

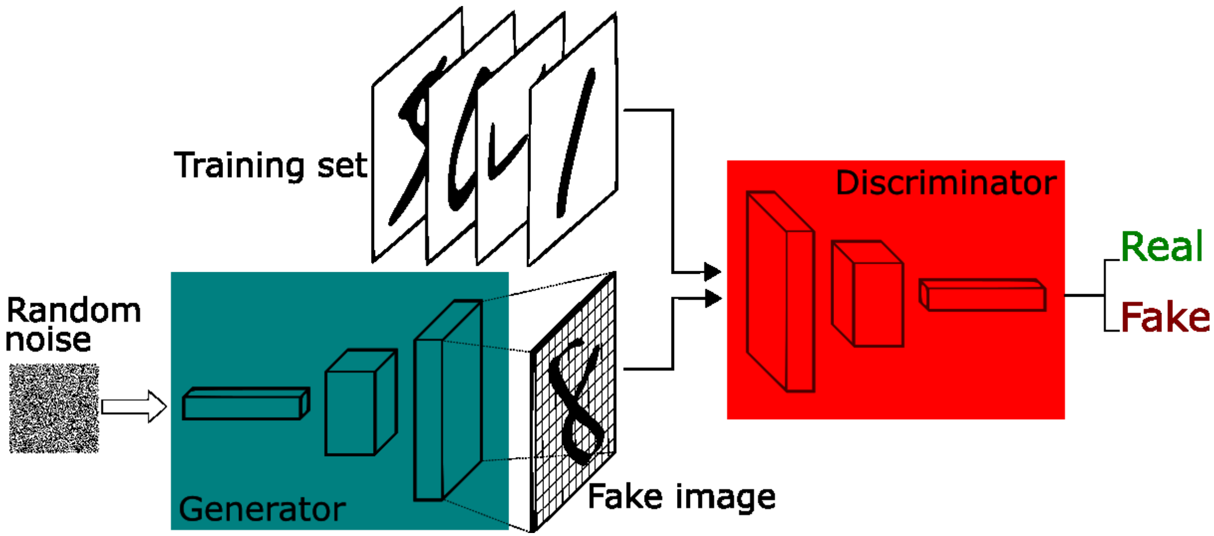
$$\begin{cases} \frac{\partial J}{\partial h_i} = 0 \quad \Rightarrow \quad W^T \varphi(x_i) = \lambda h_i, \ \forall i \\ \frac{\partial J}{\partial W} = 0 \quad \Rightarrow \quad W = \sum_i \varphi(x_i) h_i^T \end{cases}$$

Elimination of W yields Kernel PCA: $K H^T = H^T \Lambda$
with $H = [h_1 ... h_N] \in \mathbb{R}^{s \times N}$ and $\Lambda = \mathrm{diag}\{\lambda_1, ..., \lambda_s\}$ with $s \leq N$.

Generative Adversarial Network (GAN)

Generative Adversarial Network (GAN) [Goodfellow et al., 2014]
Training of two competing models in a zero-sum game:

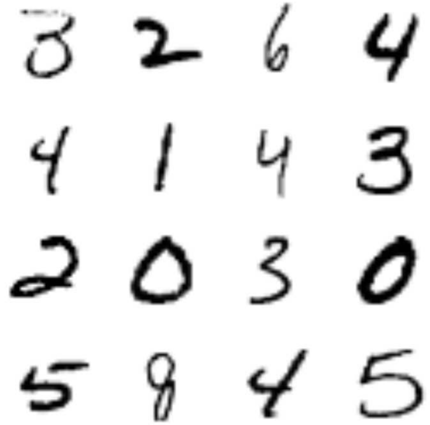
(Generator)	generate fake output examples from random noise
(Discriminator)	discriminate between fake examples and real examples.



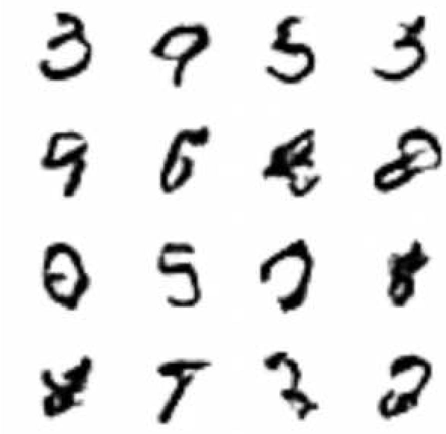
source: https://deeplearning4j.org/generative-adversarial-network

GAN: example on MNIST

MNIST training data:



GAN generated examples:



source: https://www.kdnuggets.com/2016/07/mnist-generative-adversarial-model-keras.html

GAN - Zero-sum game (1)

Game theoretic scenario: generator competes against an adversary.
https://www.deeplearningbook.org/contents/generative_models.html [Goodfellow et al. 2014, 2016]

The Generator network (G) produces samples $x = G(z; \theta^{(G)})$.
The Discriminator network (D) attempts to distinguish between samples drawn from the training data and samples drawn from the generator and gives a probability value $D(x; \theta^{(D)})$, indicating the probability that x is a real training example rather than a fake sample drawn from the model.

Zero-sum game: function $v(\theta^{(G)}, \theta^{(D)})$ determines the payoff of the discriminator. The generator receives $-v(\theta^{(G)}, \theta^{(D)})$ as its payoff. During learning each player attempts to maximize its own payoff, so that at convergence

$$G^* = \arg \min_G \max_D v(G, D)$$

Default choice, with parameter vectors $\theta^{(G)}$, $\theta^{(D)}$ of G, D:

$$\begin{aligned} v(\theta^{(G)}, \theta^{(D)}) &= \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{x \sim p_{\text{model}}} \log(1 - D(x)) \\ &= \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z \sim p_z} \log(1 - D(G(z))) \end{aligned}$$

GAN - Zero-sum game (2)

- Unfortunately, learning in GANs can be difficult in practice when G and D are represented by neural networks and $\max_D v(G, D)$ is not convex.

- The traditional ***minimax*** **GAN** with

$$J^{(G)}(G) = \mathbb{E}_{z \sim p_z} \log(1 - D(G(z)))$$

  suffers from vanishing gradients in the areas where $D(x)$ is flat (note: equilibria in zero-sum games are saddle points).

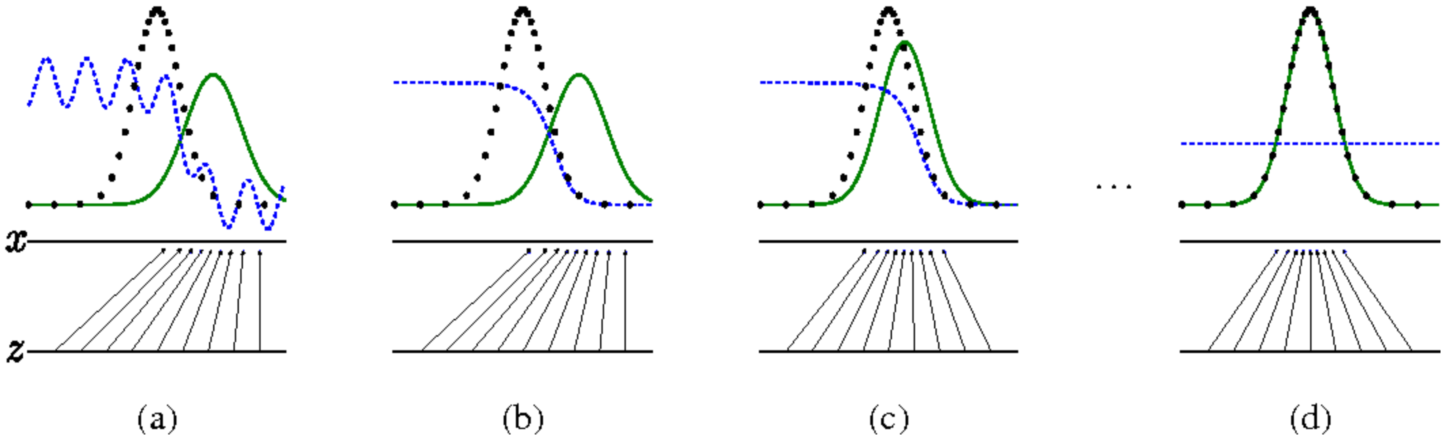
  Instead, the ***non-saturating*** **GAN** [Fedus et al. 2018] can be used

$$J^{(G)}(G) = -\mathbb{E}_{z \sim p_z} \log D(G(z))$$

- A main motivation for the design of GANs is that the learning process requires neither approximate inference nor approximation of a partition function gradient.

GAN - Illustration of working principle

black: data generating distribution, **blue**: discriminative distribution (D), **green**: generative distribution (G)



(a) adversarial pair near convergence: p_G is similar to p_{data} and D is partially accurate;
(b) In the inner loop of the algorithm, D is trained to discriminate fake samples from data, converging to $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_G(x)}$;
(c) After an update to G, gradient of D has guided $G(z)$ to flow to regions that are more likely to be classified as data;
(d) After several steps of training, one cannot further improve because $p_G = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions ($D(x) = \frac{1}{2}$).

From [Goodfellow et al. 2014]

GAN - Minibatch training

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
  - Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
  - Update the generator by descending its stochastic gradient:

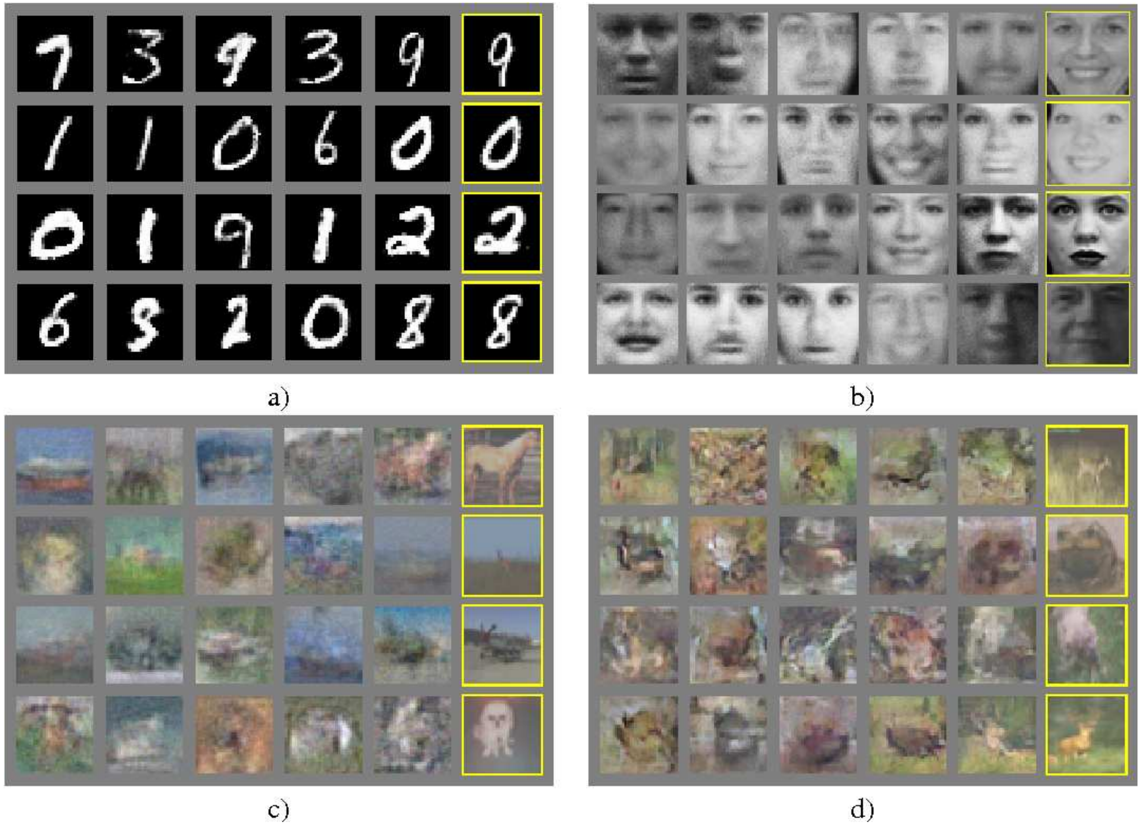
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

From [Goodfellow et al. 2014]

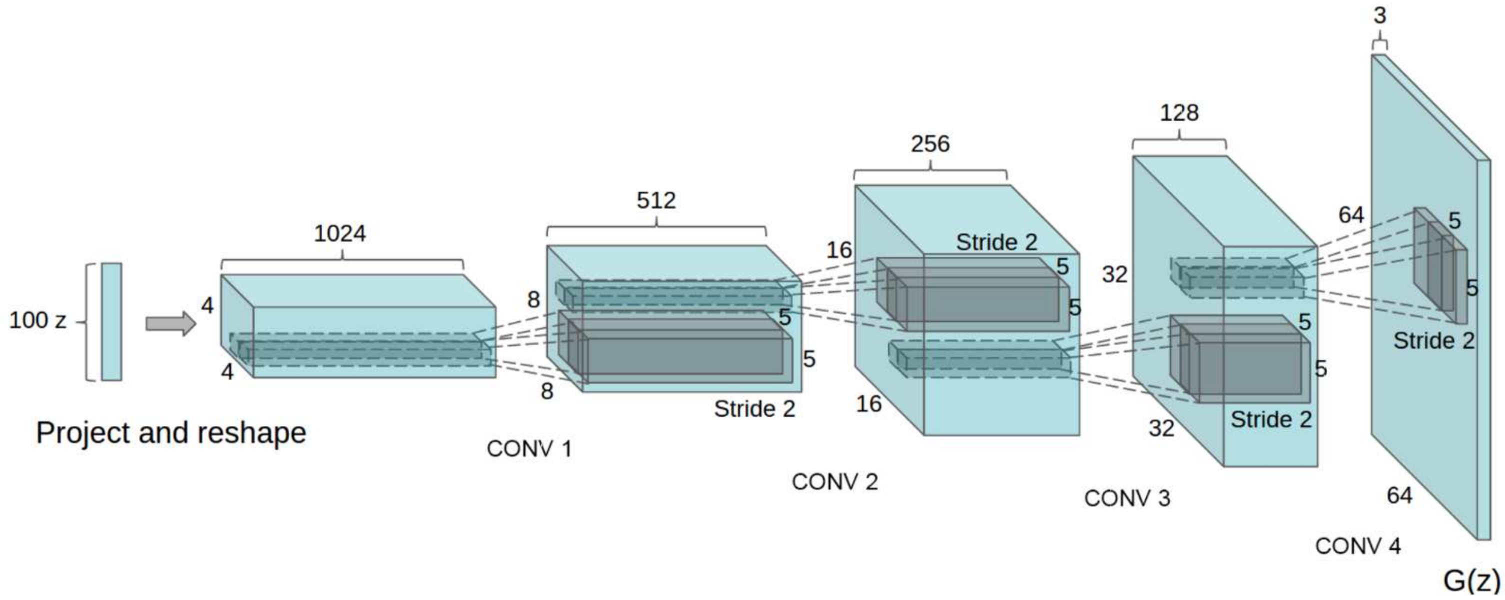
GAN - Samples from the model



Samples from the model are shown [a) MNIST, b) Toronto Face Dataset, c) CIFAR-10 (fully connected model), d) CIFAR-10 (convolutional D and deconvolutional G)].

Last column: **nearest training example** of the neighboring sample, in order to demonstrate that the model has not memorized the training set. [Goodfellow et al. 2014]

Deep convolutional GAN (DCGAN)



DCGAN generator used for Large-scale Scene Understanding. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions then convert this high level representation into a 64×64 pixel image. No fully connected or pooling layers are used.

[Radford et al. 2015]

Wasserstein GAN

In Wasserstein GAN (WGAN) [Arjovsky et al. 2017] the discriminator emits an unconstrained real number rather than a probability. The cost function for the WGAN omits the log-sigmoid functions used in the original GAN.

Cost function of the discriminator D and generator G:

$$\begin{aligned} W^{(D)}(D,G) &= \mathbb{E}_{x\sim p_{\text{data}}}[D(x)] - \mathbb{E}_{z\sim p_z}[D(G(z))] \\ W^{(G)}(D,G) &= -W^{(D)}(D,G) \end{aligned}$$

When the discriminator is Lipschitz smooth, this approach approximately minimizes the **Wasserstein distance** between p_{data} and p_{model} (this can be enforced by clipping the weights of D).

Note: Another approach is MMD GANs [Binkowski et al. 2018] where a *kernel-based method* is used in the Maximum Mean Discrepancy (MMD) between samples related to two probability measures.

[Fedus et al. 2017]

References

- I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016 (www.deeplearningbook.org)
- Y. Bengio, Learning deep architectures for AI, *Foundations and trends in Machine Learning*, 2(1): 1-127, 2009
- D.H. Ackley, G.E. Hinton, T.J. Sejnowski, A learning algorithm for Boltzmann machines, *Cognitive Science*, 9(1): 147-169, 1985.
- J. Hertz, A. Krogh, R.G. Palmer, *Introduction to the theory of neural computation*, Santa Fe Institute, Lecture Notes Vol I, Addison-Wesley, 1991.
- J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences USA*, 79, 2554-2558, 1982.
- P. Smolensky, Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart & J. L. McClelland (Eds.), Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1: Foundations. New York: McGraw-Hill, 1986.
- A. Fischer, C. Igel, Training restricted Boltzmann machines: an introduction, *Pattern Recognition*, 47, 25-39, 2014.

- G.E. Hinton, What kind of graphical model is the brain? In Proc. 19th International Joint Conference on Artificial Intelligence (pp. 1765-1775). San Francisco: Morgan Kaufmann, 2005.
- G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, *Neural Computation*, 18, 1527-1554, 2006.
- H. Larochelle, Y. Bengio, Classification using discriminative restricted Boltzmann machines. In Proceedings of the 25th International Conference on Machine Learning. New York: ACM, 2008.
- Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F.-J. Huang, A tutorial on energy-based learning. In G. Bakir, T. Hofmann, B. Scholkopf, A. Smola, B. Taskar (Eds.), Predicting structured data. Cambridge, MA: MIT Press, 2006.
- R. Salakhutdinov, G.E. Hinton, Deep Boltzmann machines, *PMLR*, 5, 448-455, 2009.
- R. Salakhutdinov, Learning deep generative models, *Annu. Rev. Stat. Appl.*, 2, 361-385, 2015
- N. Srivastava, R. Salakhutdinov, Multimodal learning with deep Boltzmann machines, *Journal of Machine Learning Research*, 15, 2949-2980, 2014.
- H. Lee, R. Grosse, R. Ranganath, A.Y. Ng, Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks, *Communication of the ACM*, Vol. 54, No. 10, pp. 95-103, 2011
- M. Welling, M. Rosen-Zvi, G.E. Hinton, Exponential family harmoniums with an application to information retrieval. In L. K. Saul, Y. Weiss, L. Bottou (Eds.),

Advances in neural information processing systems, 17. Cambridge, MA: MIT Press, 2004.

- J.A.K. Suykens, Deep Restricted Kernel Machines Using Conjugate Feature Duality, *Neural Computation*, Vol. 29, No. 8, pp. 2123-2163, August 2017.
- G. Montavon, K.R. Muller, M. Cuturi, Wasserstein Training of Restricted Boltzmann Machines, NIPS 2016, pp. 3718-3726
- C. Wang, T. Tong, Y. Zou, Wasserstein Training of Deep Boltzmann Machines.
- S. Theodoridis, *Machine learning: A Bayesian and Optimization Perspective*, Academic Press, 2015.
- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative Adversarial Networks, 2014, arXiv:1406.2661
- I. Goodfellow, NIPS 2016 Tutorial: Generative Adversarial Networks, arXiv:1701.00160
- W. Fedus, M. Rosca, B. Lakshminarayanan, A. M. Dai, S. Mohamed, I. Goodfellow, Many Paths to Equilibrium: GANs Do Not Need to Decrease a Divergence At Every Step, 2017, arXiv:1710.08446
- A. Radford, L. Metz, S. Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, arXiv:1511.06434
- M. Arjovsky, S. Chintala, L. Bottou, Wasserstein GAN, 2017, arXiv:1701.07875
- M. Binkowski, D.J. Sutherland, M. Arbel, A. Gretton, Demystifying MMD GANs, 2018, arXiv:1801.01401