Deep Learning

Alexandre Allauzen, Michèle Sebag, Mathieu Labeau
CNRS & Université Paris-Sud

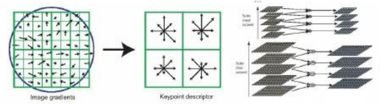
Jan. 17th, 2018

Credit for slides: Yoshua Bengio, Yann LeCun, Nick McClure, Victor Berger

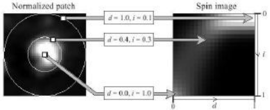
Deep Learning: what is new ?

Former state of the art

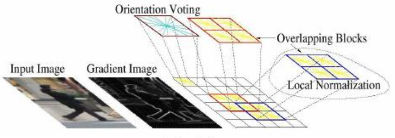
e.g. in computer vision



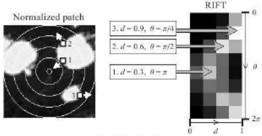
SIFT



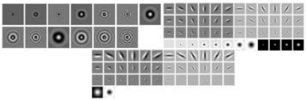
Spin image



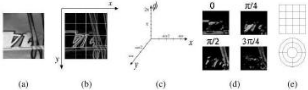
HoG



RIFT



Textons



GLOH

SIFT: scale invariant feature transform
HOG: histogram of oriented gradients

Textons: "vector quantized responses of a linear filter bank"

What is new, 2

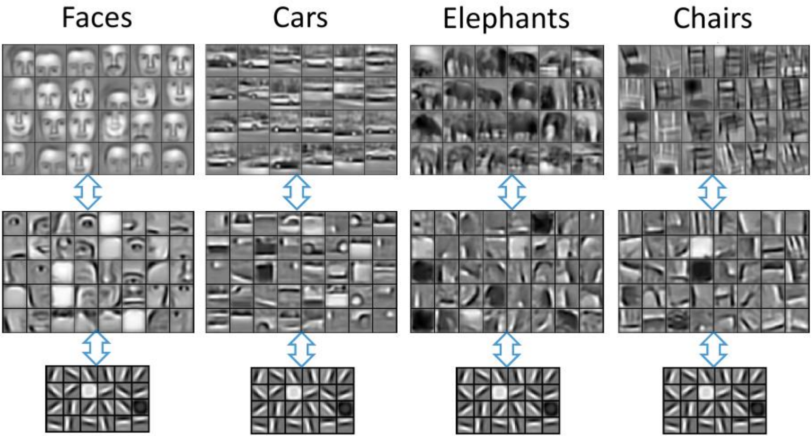
Traditional approach

→ Manually crafted features → Trainable classifier

Deep learning

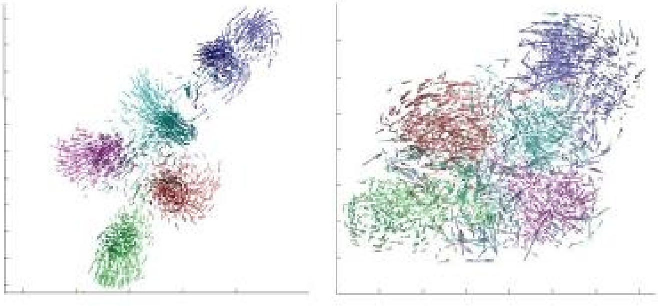
→ Trainable feature extractor → Trainable classifier

A new representation is learned



Good features ?

Good **Bad**



- Similar examples are close
- Dissimilar examples are farther away

Do we need labels ?

LeCun 2016

UL: Unsupervised
SL: Supervised
RL: Reinforcement

Auto-Encoders

Siamese Networks

Variational Auto-Encoders

Generative Adversarial Networks

Domain adaptation

Partial conclusions

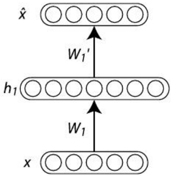
Auto-encoders

$$\mathcal{E} = \{(\mathbf{x}_i, y_i), x_i \in \mathbb{R}^d, y_i \in \mathbb{R}, i = 1 \dots n\}$$

$$\mathbf{x} \longrightarrow h_1 \longrightarrow \hat{\mathbf{x}}$$

- An auto-encoder:

$$\text{Find } W^* = \arg\min_W \left(\sum_i ||W^t o W(\mathbf{x}_i) - x_i||^2 \right)$$



(*) Instead of min squared error, use cross-entropy loss:

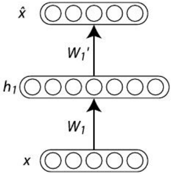
$$\sum_j \mathbf{x}_{i,j} \log \hat{\mathbf{x}}_{i,j} + (1 - \mathbf{x}_{i,j}) \log (1 - \hat{x}_{i,j})$$

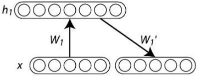
(**) Why W for encoding and W^t for decoding ?

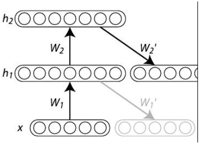
Auto-encoders were used (2006-2010) to initialize deep networks

First layer $\mathbf{x} \longrightarrow \mathbf{h}_1 \longrightarrow \hat{\mathbf{x}}$

Second layer $\mathbf{h}_1 \longrightarrow \mathbf{h}_2 \longrightarrow \hat{\mathbf{h}_1}$ same, replacing $\mathbf{x}$ with $\mathbf{h}_1$



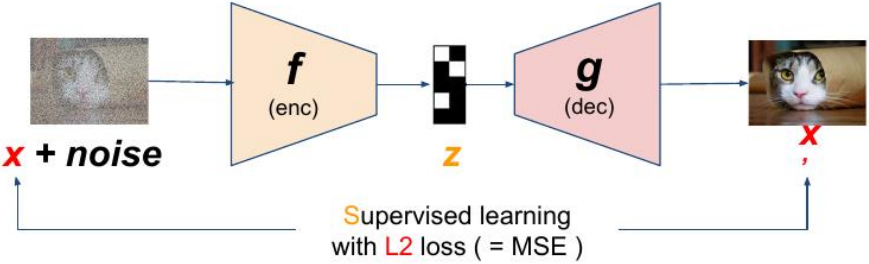




Denoising Auto-Encoders

Principle

- Add noise to **x**: input AE is $x + \epsilon$
- Recover **x**.



Auto-encoders and Principal Component Analysis

Assume

- A single layer
- Linear activation

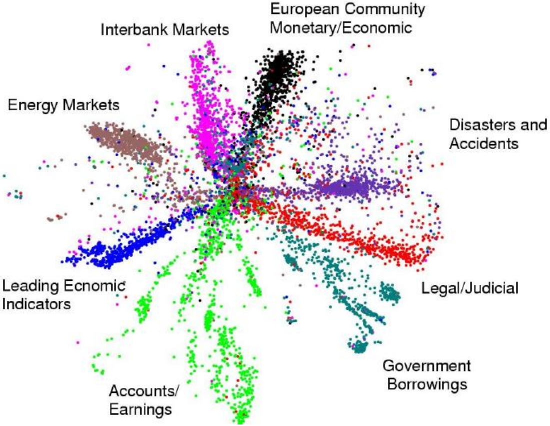
Then

- An auto-encoder with k hidden neurons $\approx$ first k eigenvectors of PCA

Why ?

Visualization with (non linear) Autoencoders

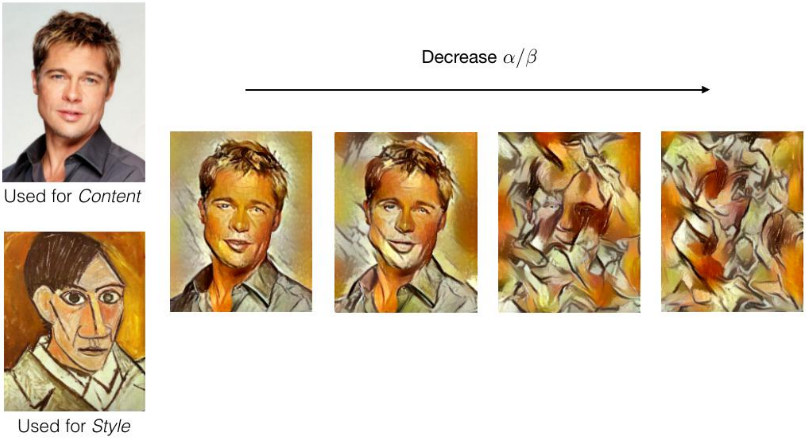
For $k = 2$,



more: t-SNE https://distill.pub/2016/misread-tsne/

Morphing of representations

Gatys et al. 15, 16

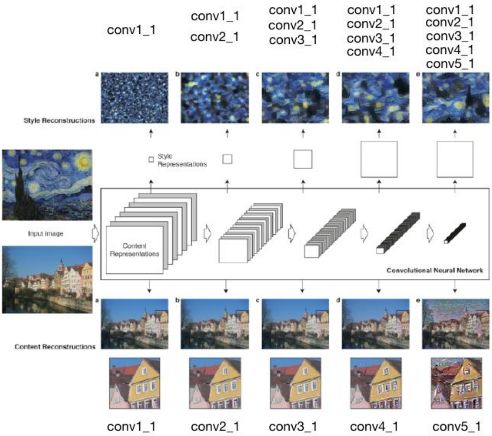


Use image $\mathbf{x}_0$ for content (AE ϕ), $\mathbf{x}_1$ for style (AEϕ')
Morphing: Find input image $\mathbf{x}$ minimizing

$$\alpha\|\phi(\mathbf{x}) - \phi(\mathbf{x}_0)\| + \beta\langle\phi'(\mathbf{x}), \phi'(\mathbf{x}_1)\rangle$$

Morphing of representations, 2

Gatys et al. 15, 16

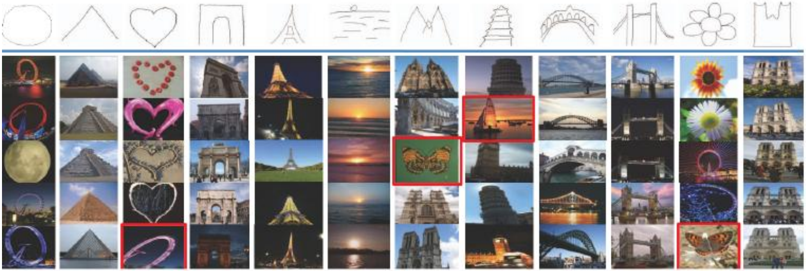


- Contents (bottom): convolutions with decreasing precision
- Style (top): correlations between the convol. features

Morphing of representations, 2

Gatys et al. 15, 16

Siamese Networks



Classes or similarities ?

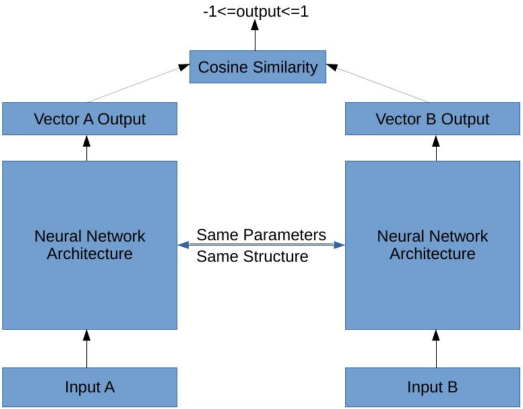
Siamese Networks

Bromley et al. 93

Principle

- Neural Networks can be used to define a latent representation
- Siamese: optimize the related metrics

Schema



Siamese Networks, 2

Data

$$\mathcal{E} = \{x_i \in \mathbb{R}^d, i \in [[1, n]]\}; \mathcal{S} = \{(x_{i,\ell}, x_{j_\ell}, c_\ell) \ s.t. \ c_\ell \in \{-1, 1\}, \ell \in [[1, L]]\}$$

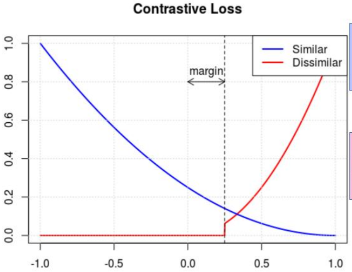
Experimental setting

- Often: few similar pairs; by default, pairs are dissimilar
- Subsample dissimilar pairs (optimal ratio between 2/1 ou 10/1)

Loss

Given similar and dissimilar pairs (E_+ and E_-)

$$\mathcal{L} = \sum_{(i,j) in E_+} L_+(i,j) + \sum_{(k,\ell) in E_-} L_-(k,\ell)$$



$$L_+ = \frac{1}{4} \cdot (1 - cosine(x_1, x_2))^2$$

$$L_- = \begin{cases} \text{cosine}(x_1, x_2)^2, & \text{if } x \geq \text{margin} \\ 0, & \text{otherwise} \end{cases}$$

Applications

- Signature recognition
- Image recognition, search
- Article, Title
- Filter out typos
- Recommandation

Siamese Networks

PROS

- Learn metrics, invariance operators
- Generalization beyond train data

CONS

- More computationally intensive
- More hyperparameters and fine-tuning, more training

Distribution estimation

Data

$$\mathcal{E} = \{x_1, \ldots, x_n, x_i \in \mathcal{X}\}$$

Goal

- Find a probability distribution that models the data

$$p_\theta : \mathcal{X} \mapsto [0,1] \;\; \text{s.t. } \theta = \arg\max \prod_i p_\theta(x_i)$$

$\equiv$ maximize the log likelihood of data

$$\arg\max \prod_i p_\theta(x_i) = \arg\max \sum_i log(p_\theta(x_i))$$

Gaussian case

$$\theta = (\mu, \sigma) \qquad p_\theta(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\}$$

Graphical models

Find hidden variables $\mathbf{z}$ s.t.

$$\mathbf{z} \mapsto \mathbf{x} \text{ i.e. } \quad \text{good} \quad p(\mathbf{x}|\mathbf{z})$$

Bayes relation

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z}|\mathbf{x}).p(\mathbf{x}) = p(\mathbf{x}|\mathbf{z}).p(\mathbf{z})$$

Hence

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z}).p(\mathbf{z})}{\int p(\mathbf{x}|\mathbf{z}).p(\mathbf{z})d\mathbf{z}}$$

Problem:

denominator computationally intractable...

State of art

- Monte-Carlo estimation
- Variational Inference: choose $\mathbf{z}$ well-behaved, and make $q(\mathbf{z})$ "close" to $p(\mathbf{z}|\mathbf{x})$.

Variational Inference

- Approximate $p(\mathbf{z}|\mathbf{x})$ by $q(\mathbf{z})$
- Minimize distance between both, using Kullback-Leibler divergence

Reminder

- information $(\mathbf{x}) = -log(p(\mathbf{x}))$
- entropy$(\mathbf{x}_1, \ldots \mathbf{x}_k) = -\sum_i p(\mathbf{x}_i) log(p(\mathbf{x}_i))$
- Kullback-Leibler divergence between distribution q and p

$$KL(q||p) = \sum q(\mathbf{x}) log \frac{q(\mathbf{x})}{p(\mathbf{x})}$$

Beware: not symmetrical, hence not a distance; plus numerical issues when supports are different

Variational inference

$$\text{Minimize } KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

Evidence Lower Bound (ELBO)

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) \quad = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

Evidence Lower Bound (ELBO)

$$
\begin{aligned}
KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) &= \int q(\mathbf{z}) log \frac{q(\mathbf{Z})}{p(\mathbf{Z}|\mathbf{X})} d\mathbf{z} \\
&= \int q(\mathbf{z}) log \frac{q(\mathbf{Z})p(\mathbf{X})}{p(\mathbf{Z},\mathbf{X})} d\mathbf{z}
\end{aligned}
$$

Evidence Lower Bound (ELBO)

$$
\begin{aligned}
KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) &= \int q(\mathbf{z}) log \frac{q(\mathbf{Z})}{p(\mathbf{Z}|\mathbf{X})} d\mathbf{z} \\
&= \int q(\mathbf{z}) log \frac{q(\mathbf{Z})p(\mathbf{X})}{p(\mathbf{Z},\mathbf{X})} d\mathbf{z} \\
&= \int q(\mathbf{z}) log \frac{q(\mathbf{Z})}{p(\mathbf{Z},\mathbf{X})} d\mathbf{z} + \int q(\mathbf{z}) log(p(\mathbf{x})) d\mathbf{z}
\end{aligned}
$$

Evidence Lower Bound (ELBO)

$$
\begin{aligned}
KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) &= \int q(\mathbf{z}) log \frac{q(\mathbf{Z})}{p(\mathbf{Z}|\mathbf{X})} d\mathbf{z} \\
&= \int q(\mathbf{z}) log \frac{q(\mathbf{Z})p(\mathbf{X})}{p(\mathbf{Z},\mathbf{X})} d\mathbf{z} \\
&= \int q(\mathbf{z}) log \frac{q(\mathbf{Z})}{p(\mathbf{Z},\mathbf{X})} d\mathbf{z} + \int q(\mathbf{z}) log(p(\mathbf{x})) d\mathbf{z} \\
&= \int q(\mathbf{z}) log \frac{q(\mathbf{Z})}{p(\mathbf{Z},\mathbf{X})} d\mathbf{z} + log(p(\mathbf{x}))
\end{aligned}
$$

Evidence Lower Bound (ELBO)

$$
\begin{aligned}
KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) &= \int q(\mathbf{z}) log \frac{q(\mathbf{Z})}{p(\mathbf{Z}|\mathbf{X})} d\mathbf{z} \\
&= \int q(\mathbf{z}) log \frac{q(\mathbf{Z})p(\mathbf{X})}{p(\mathbf{Z},\mathbf{X})} d\mathbf{z} \\
&= \int q(\mathbf{z}) log \frac{q(\mathbf{Z})}{p(\mathbf{Z},\mathbf{X})} d\mathbf{z} + \int q(\mathbf{z}) log(p(\mathbf{x})) d\mathbf{z} \\
&= \int q(\mathbf{z}) log \frac{q(\mathbf{Z})}{p(\mathbf{Z},\mathbf{X})} d\mathbf{z} + log(p(\mathbf{x})) \\
&= -\int q(\mathbf{z}) log \frac{p(\mathbf{Z},\mathbf{X})}{q(\mathbf{Z})} d\mathbf{z} + log(p(\mathbf{x}))
\end{aligned}
$$

Evidence Lower Bound, 2

Define

$$L(q(\mathbf{z})) = \int q(\mathbf{z}) log \frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z})} d\mathbf{z}$$

Last slide:

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = log(p(\mathbf{x})) - L(q(\mathbf{z})))$$

Hence

Minimize Kullback-Leibler divergence $\equiv$ Maximize $\mathbf{L(q(z))}$

Evidence Lower Bound, 3

More formula massaging

$$\begin{aligned} L(q(\mathbf{z})) &= \int q(\mathbf{z}) log \frac{p(\mathbf{Z}, \mathbf{X})}{q(\mathbf{Z})} d\mathbf{z} \\ &= \int q(\mathbf{z}) log \frac{p(\mathbf{Z}|\mathbf{X}) p(\mathbf{X})}{q(\mathbf{Z})} d\mathbf{z} \\ &= \int q(\mathbf{z}) log \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} d\mathbf{z} + \int q(\mathbf{z}) log p(\mathbf{x}) d\mathbf{z} \\ &= -KL(q(\mathbf{z}) || p(\mathbf{z}|x)) + \mathbb{E}_q[log(p(\mathbf{x}))] \end{aligned}$$

Finally

$$\text{Maximize } \mathbb{E}_q[log(p(\mathbf{x})) - KL(q(\mathbf{z}) || p(\mathbf{z}|x))$$

(make $p(\mathbf{x})$ great under q while minimizing the KL divergence between the two)

Where neural nets come in

Searching p and q

- We want $p(\mathbf{x}|\mathbf{z})$, we search $p(\mathbf{z}|x)$
- Let $p(\mathbf{z}|\mathbf{x})$ be defined as a neural net (encoder)
- We want it to be close to a well-behaved distribution $q(\mathbf{z})$

$$\text{Minimize } KL(q(\mathbf{z})||p(\mathbf{z}|x))$$

  with $q(\mathbf{z})$ for instance Gaussian.
- And from $\mathbf{z}$ we generate a distribution $p(\mathbf{x}|\mathbf{z})$ (defined as a neural net, "decoder")
- such that $p(\mathbf{x}|\mathbf{z})$ gives a high probability mass to our data (next slide)

$$\text{Maximize } \mathbb{E}_q[log(p(\mathbf{x}))]$$

Good news

All these criteria are differentiable: can be used to train the neural net.

The loss of the variational decoder

Continuous case

- $\mathbf{x} \mapsto \mathbf{z}$; Gaussian case, $\mathbf{z} \sim p(\mathbf{z}|\mathbf{x})$
- Now $\mathbf{z}$ is given as input to the decoder, generates $\hat{\mathbf{x}}$ (deterministic)
- $p(\mathbf{x}|\hat{\mathbf{x}}) = F(\exp\{-\|\mathbf{x} - \hat{\mathbf{x}}\|^2\})$
- ... back to the L_2 loss

Binary case

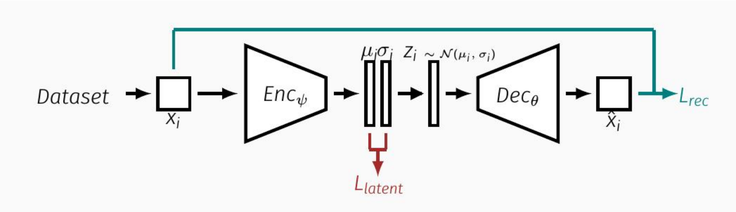
- Exercize: back to the cross-entropy loss

Variational auto-encoders

Kingma et al. 13

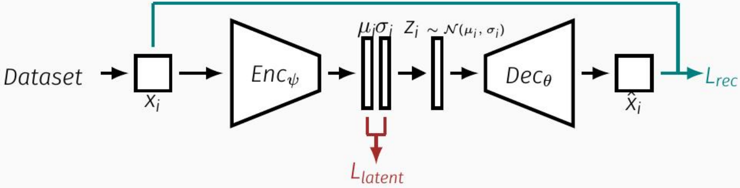
Position

- Like an auto-encoder (data fitting term) with a regularizer, the KL divergence between the distribution of the hidden variables $\mathbf{z}$ and the target distribution.
- Say the hidden variable follows a Gaussian distribution: $\mathbf{z} \sim \mathcal{N}(\mu, \Sigma)$
- Therefore, the encoder must compute the parameters μ and Σ



Variational auto-encoders, 2

Kingma et al. 13



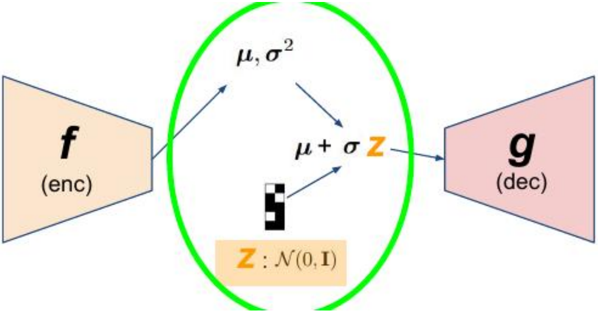
- encoding cost: $L_{latent} = \sum_i D_{KL}\left(Enc_\psi(x_i) \,||\, \mathcal{N}(0;1)\right)$
- reconstruction loss:

$$L_{rec} = \sum_i \mathbb{E}_{z \sim Enc_\psi(x_i)} \left[-\log p_{Dec_\theta(z)}(x_i)\right]$$
$$= \sum_i \mathbb{E}_{z \sim Enc_\psi(x_i)} ||Dec_\theta(z) - x_i||^2 + cst.$$

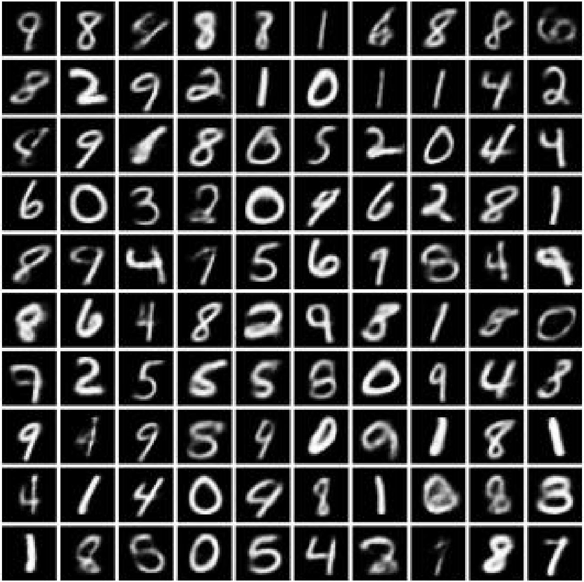
The reparameterization trick

Principle

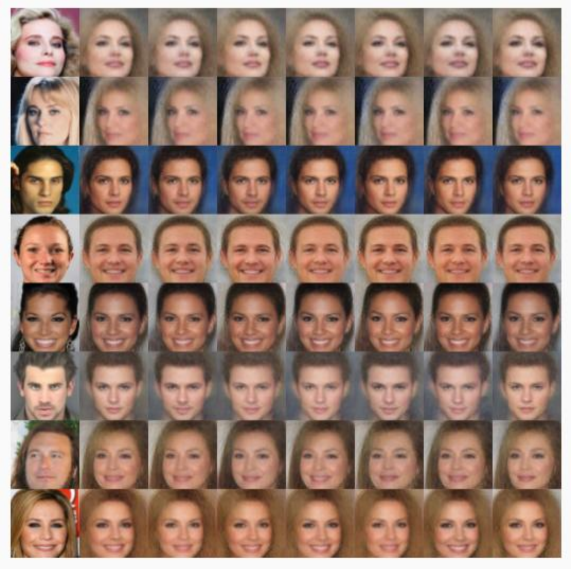
- Hidden layer: parameters of a distribution $\mathcal{N}(\mu, \sigma^2)$
- Distribution used to generate values $z = \mu + \sigma \times \mathcal{N}(0, 1)$
- Enables backprop; reduces variances of gradient



Examples



Examples



Also: https://www.youtube.com/watch?v=XNZIN7Jh3Sg

Generative Adversarial Networks

Goodfellow et al., 14

Goal: Find a generative model

- Classical: learn a distribution — hard
- Idea: replace a distribution evaluation by a 2-sample test

Principle

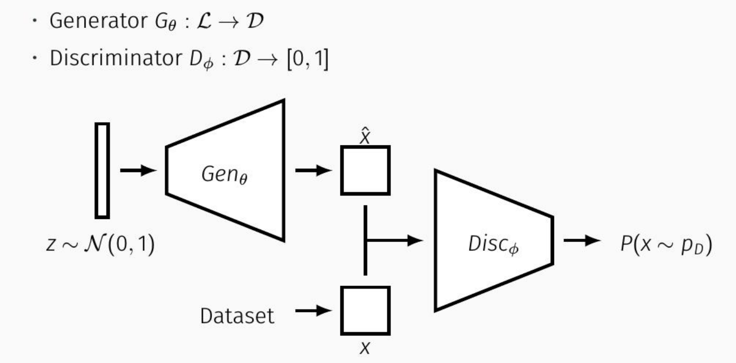
- Find a good generative model, s.t.
- Generated Sample **cannot be discriminated** from Initial Sample (not easy)

Principle

Goodfellow, 2017

Elements

- True samples (Real)
- A generator G (variational auto-encoder): generates from x (real) or from scratch (fake)
- A discriminator D: discriminates *fake* from others (*real* and *Real*)



Principle, 2

Goodfellow, 2017

Mechanism

- Alternate minimization
- Optimize D to tell fake from rest
- Optimize G to deceive D

Turing test

$$Min_G \ Max_D \mathbb{E}_{x \ in \ data} log(D(\mathbf{x}) + \mathbb{E}_{z \sim p_x(z)}[1 - D(z)]$$

Generative adversarial networks

Goodfellow, 2017

Generative adversarial networks, 2

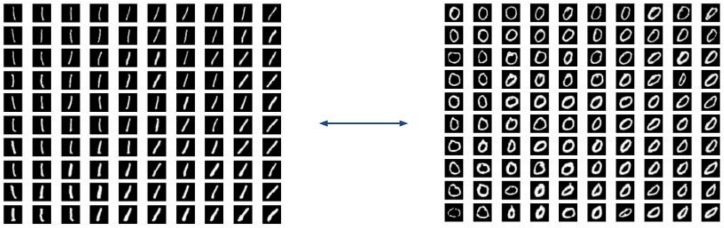
Goodfellow, 2017

Limitations

Training instable

co-evolution of Generator / Discriminator

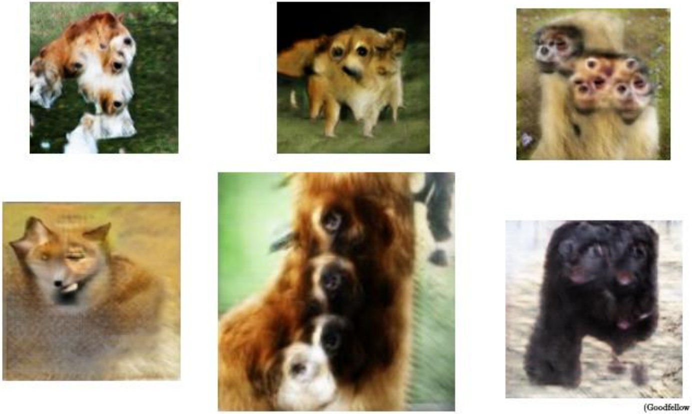
Mode collapsing



Generating monsters

Generative adversarial networks

Goodfellow, 2017



(Goodfellow 2016)

Domain adaptation

Context

- Testing Distribution $\neq$ Training Distribution
- Goal: learn from source distribution $\mathcal{D}_S$, apply on target distribution $\mathcal{D}_T$

What can change ?

- $p(\mathbf{x})$ distribution of instances
- $p(y|\mathbf{x})$ conditional distribution

Domain adaptation, 2

Motivations

- A source domain $\mathcal{D}_S$

$$\mathcal{E} = \{(\mathbf{x}_i, y_i), i = 1 \ldots n\}$$

- A target domain $\mathcal{D}_T$
  - Without labels, or
  - With few labels

Goal

Use source data to improve / speed-up learning on target data

Examples

- Opinion mining (movies vs books, hifi vs electric devices)
- Character recognition, different fonts

Formalization

Input

$$\mathcal{E} = \{(\mathbf{x}_i, y_i), x_i \in \mathcal{X} = \mathbb{R}^d, y_i \in \mathcal{Y} = \{-1, 1\}, i = 1 \dots n\}$$

$$\mathcal{E}' = \{(\mathbf{x}'_i), x_j \in \mathbb{R}^d, j = 1 \dots n'\}$$

Two distributions: $\mathcal{D}_S$ on $\mathcal{X} \times \{-1, 1\}$ and $\mathcal{D}_T$ on $\mathcal{X} \times \{-1, 1\}$.

Goal: build a classifier with low risk wrt $\mathcal{D}_T$.

$$R_T(h) = Pr_{\mathcal{D}_T}(h(x) \neq y$$

although we only know $\mathcal{D}_T^X$ (the projection of the target distribution on the instance space).

No magic !

Ben-David et al. 2006, 2010

1. $\mathcal{H}$ Divergence between $\mathcal{D}_S$ and $\mathcal{D}_T$ on $\mathcal{X}$

$$d_X(\mathcal{D}_S, \mathcal{D}_T) = 2 \sup_{h \in \mathcal{H}} \left| Pr_{\mathcal{D}_T}(h(x) = 1) - Pr_{\mathcal{D}_S}(h(x) = 1) \right|$$

This divergence is high if there exists h with value 1 on source and 0 on target (or vice versa).

2. Proposition

A good approximation of $\mathcal{H}$ divergence is

$$d_X(\widehat{\mathcal{D}_S, \mathcal{D}_T}) = 2 \left(1 - \min_h \left(\frac{1}{n} \sum_i 1_{h(x_i)=0} + \frac{1}{n'} \sum_j 1_{h(x'_j)=1} \right) \right)$$

The divergence can be approximated by the ability to empirically discriminate between source and target.

Bounding the domain adaptation risk

Ben-David et al. 2006, 2010

3. Theorem

With probability $1-\delta$, if d is the dimension of $\mathcal{H}$,

$$R_T(h) \leq \widehat{R_S(h)} + C\sqrt{\frac{4}{n}(d\log\frac{2}{d} + \log\frac{4}{\delta})} + \widehat{d_X} + \text{Best possible}$$

and

$$\text{Best possible} = \inf_h \left(R_S(h) + R_T(h)\right)$$

What we want (risk on h wrt $\mathcal{D}_T$) is bounded by:

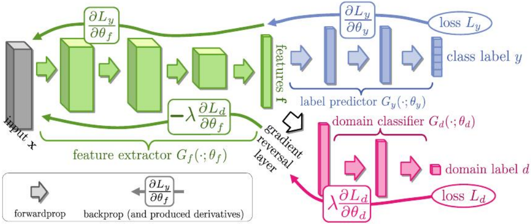
- empirical risk on S
- $+$ error related to possible overfitting
- $+$ min error one can achieve on both source and target distribution.

Domain Adaptation with Deep NN

Ganin et al. 2016

Two labels

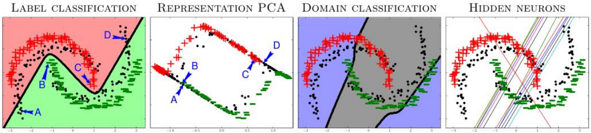
- $(\mathbf{x}, y)$: y is known if $\mathbf{x} \sim \mathcal{D}_S$ — source label
  the feature layers help to predict y
- $(\mathbf{x}, z)$: $z = 1$ if $\mathbf{x} \sim \mathcal{D}_S$ and $z = 0$ if $\mathbf{x} \sim \mathcal{D}_T$
  the feature layers **do not want help** to predict z



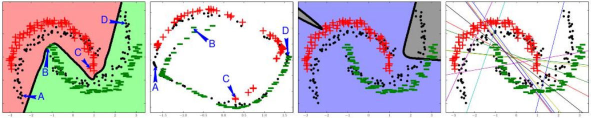
Algorithm

1. Green part + blue part: backpropagation of error wrt $(\mathbf{x}, y)$
2. Red part: backpropagation of error wrt $(\mathbf{x}, z)$
   followed by backpropagation of $-$ gradient of red error.

The intertwinning moons



(a) Standard NN. For the "domain classification", we use a *non adverserial* domain regressor on the hidden neurons learned by the Standard NN. (This is equivalent to run Algorithm 1, without Lines 22 and 31)

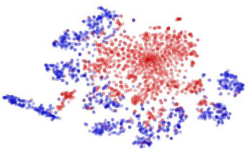


(b) DANN (Algorithm 1)

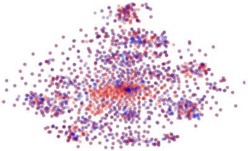
- left: the decision boundary
- 2nd left: apply PCA on the feature layer
- 3rd left: discrimination source vs target
- right: each line corresponds to hidden neuron = .5

An image domain adaptation

MNIST → MNIST-M: top feature extractor layer

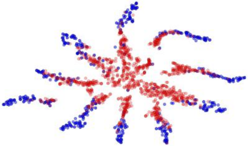


(a) Non-adapted

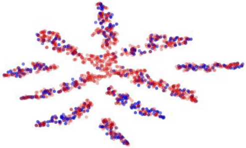


(b) Adapted

SYN NUMBERS → SVHN: last hidden layer of the label predictor



(a) Non-adapted



(b) Adapted

Auto-Encoders

Siamese Networks

Variational Auto-Encoders

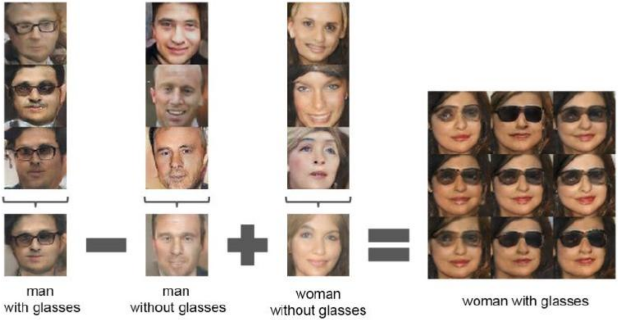
Generative Adversarial Networks

Domain adaptation

Partial conclusions

Take-home message

Representation – Computation – Concept



Questions

- What should be learned / given ? (innate / acquired)
- Data: how much ?
- Debiasing the data... Toward Fair AI.