ε.δε.μ²
ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ & ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Boltzmann Machine
Gibbs Sampling & Bayesian Statistics
Boltzmann Learning Rule, Maximum Likelihood Principle
Generative Models, Generative Adversarial Network (GAN)

Prof. Vasilis Maglaris

maglaris@netmode.ntua.gr

www.netmode.ntua.gr

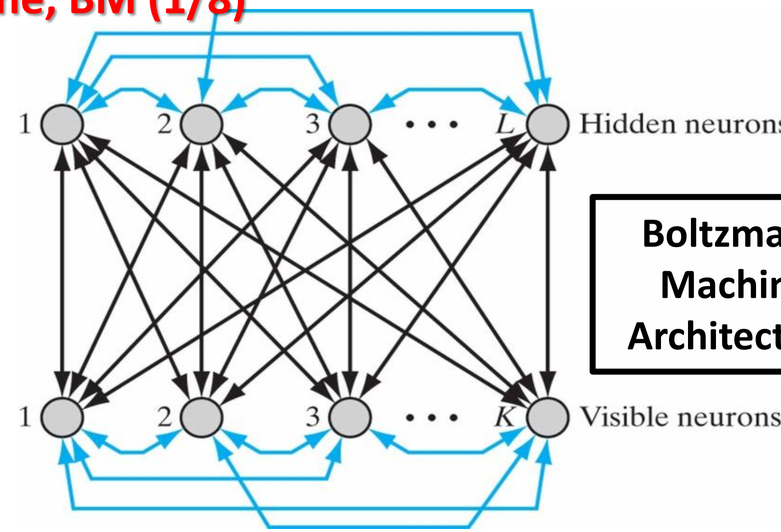
Room 002, New ECE Building

Tuesday March 18, 2025

Boltzmann Machine, BM (1/8)

(1985, ***Geoffrey Hinton*** & ***Terry Sejnowski***)

Goal: Approximation of deficient **input sample vectors** (e.g. image ***pattern completion***) by generation of **output vector estimates**, statistically conforming to ***unlabeled*** training sample



Boltzmann Machine Architecture

A ***Boltzmann Machine*** (**BM**) is a ***Stochastic Recurrent Network*** with 2 layers of neurons:

- K **Visible,** L **Hidden** binary state ***Stochastic Neurons***, with state probabilities assigned via unsupervised learning
- **Symmetric Synapses** $i \rightarrow j$: $w_{ji} = w_{ij}, w_{ii} = 0$ amongst **all** neurons

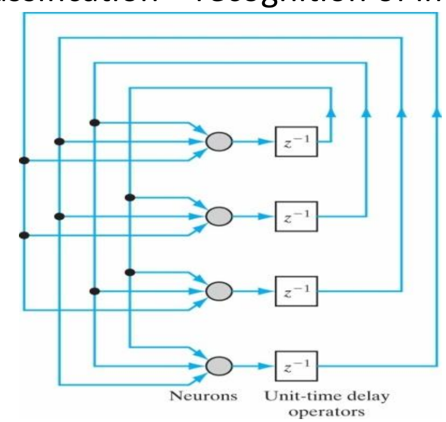
The **BM** converges via ***unsupervised*** learning to ***Markov Random Field*** "thermal" equilibrium:

- Binary **training vectors** are clamped to ***Visible Nodes***; via a **gradient ascent algorithm** synaptic weights converge and final states of **both** ***Visible*** & ***Hidden Neurons*** are determined
- A new **input** vector (**test**) is inserted In ***Visible Nodes***. The **BM** generates via ***Gibbs sampling*** its **output** image as an update in the ***Visible Nodes***, statistically conforming to training sample vectors

Hopfield Recurrent Neural Netwotk

(1982, ***John Hopfield***)

Binary non-stochastic neurons with recurrent synapses, threshold activation and ***Hebbian supervised learning*** to determine $w_{ji} = w_{ij}, w_{ii} = 0$ in (local) minimum of ***system energy***. Application in pattern classification – recognition of images



Boltzmann Machine, BM (2/8)

Gibbs Sampling - Bayesian Statistics

(*Jeff Miller*, Duke University 2015: https://jwmi.github.io/BMS/chapter6-gibbs-sampling.pdf)

- **Sampling**:
  - Selection of a ***Sample***, a subset of a ***Sample Space*** with elements conforming to statistics inferred from the superset sample space
  - Processing of ***Sample Elements*** of the ***Sample*** for statistical analysis that can be generalized to the superset sample space (polls, client preferences…)

- **Sampling for Multi-Dimensional Elements, Sample Vectors: *Gibbs Sampling***
  - A version of the ***Metropolis*** algorithm to generate and trace (*correlated*) trajectories of multi-dimensional sample vectors via ***Markov Chain Monte Carlo*** (***MCMC***) simulations
  - An Example of **Dimensionality 2**: Generation of random pairs $\mathbf{X} = [X\ Y]^{\mathrm{T}}$ with ***Joint Probabilities*** $P(X,Y)$ based on sampling of ***single*** random variables according to ***Conditional Probabilities*** (from ***Bayesian*** statistics):
  $$x(n) \sim P(X|y(n-1) \text{ και } y(n) \sim P(Y|x(n))$$
  - Generalization for ***K* Dimensions**: Sampling of $\mathbf{X} = [X_1\ X_2\ \cdots X_K\ ]^{\mathrm{T}}$

    $n = 0$: Arbitrary initialization $\mathbf{x}(0) = [x_1(0)\ x_2(0)\ \ \ldots\ x_K(0)]^{\mathrm{T}}$

    $n \to n+1$: For $i = 1, \ldots, K$ generate random variable $x_i(n+1)$ with probability
    $$P[X_i(n+1)|\{x_1(n+1)\ \ldots\ x_{i-1}(n+1)\ x_{i+1}\ (n)\ \ \ldots\ x_K(n)\}]$$
    The condition relies on recently recorded coordinate values and excludes $X_i(n)$

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

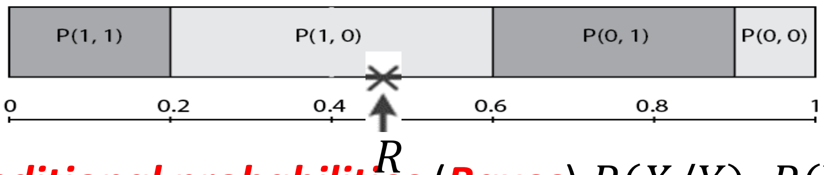
Boltzmann Machine, BM (3/8)

Example of Gibbs Sampling for 2 Dimensions

(***Enes Zvornicanin***, 2024 https://www.baeldung.com/cs/gibbs-sampling)

- Given a ***Sample Space*** of **Random Variable pairs** $[X\ Y]^{\mathrm{T}}$ with binary **values** $x, y \in \{0,1\}$
- The ***joint probabilities*** $P(X,Y)$ are given as:
  - $P(X=1, Y=1) = 0.2, \ \ P(X=0, Y=1) = 0.3$
  - $P(X=1, Y=0) = 0.4, \ \ P(X=0, Y=0) = 0.1$
- Generate n pairs: $[x(n)\ y(n)]^{\mathrm{T}} \sim P(X,Y)$ via ***direct*** or ***Gibbs*** sampling
- **Direct Sampling**: Use ***joint probabilities*** $P(X,Y)$

  Repeated ***Monte Carlo*** (***MC***) trials to generate uniformly distributed (pseudo)random number $0 < R \leq 1$ placed within 4 intervals . e.g. if $R = 0.5$ the trial generated $[1\ 0]^{\mathrm{T}}$



- **Gibbs Sampling:** Use ***conditional probabilities*** (***Bayes***) $P(X/Y)$, $P(Y/X)$ to avoid splitting the $\{0,1\}$ range to $\gg 2$ intervals and bypass involved ***MC*** comparisons
  - $P(X=1|Y=1) = \frac{P(X=1,Y=1)}{P(Y=1)} = \frac{P(X=1,Y=1)}{P(X=1,Y=1)+P(X=0,Y=1)} = \frac{0.2}{0.2+0.3} = 0.4$
  - $P(X=1|Y=0) = 0.8, P(X=0|Y=0) = 0.2, P(Y=1|X=1) = 0.333$
  - $P(Y=0|X=1) = 0.666, P(Y=1|X=0) = 0.75, P(Y=0|X=0) = 0.25$

Gibbs Sampling Algorithm

- Initial Step: $x(0) = 1$ $\quad$ $y(0) = 0$
- Step 1: $x(1) \sim P(X|y(0))$ $\quad$ $y(1) \sim P(Y|x(1))$
- Step 2: $x(2) \sim P(X|y(1))$ $\quad$ $y(2) \sim P(Y|x(2))$
- Step n: $x(n) \sim P(X|y(n-1))$ $\quad$ $y(n) \sim P(Y|x(n))$

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Boltzmann Machine, BM (4/8)

Learning Phases of Boltzmann Machine

- **Learning** is accomplished in two phases of the ***Boltzmann Learning Rule***
- **Positive Phase**: The input vectors of the ***Training Sample*** $\mathcal{T}$ are ***clamped*** to states ± 1 of the K ***visible neurons*** based on values of their features. ***Synaptic weights*** between all $L + K$ neurons are iteratively determined leading to target ***Gibbs equilibrium*** using the ***maximum likelihood principle***. During the positive phase, the **BM** encodes in its L ***hidden neurons*** higher order statistical properties (e.g. correlations) with ***marginal distribution*** under the clamped condition of its K visible neurons
- **Negative Phase**: In this subsequent phase all neurons (***visible*** and ***hidden***) interact freely, with no clamping to the training vectors in $\mathcal{T}$. Synaptic weights are iteratively determined leading all **BM** neurons to ***Gibbs equilibrium***. The final states of the visible neurons generate the ***output vector***, an image of the initial input vector as a ***new sample element*** with limiting feature probabilities consistent with those of $\mathcal{T}$
- **Algorithm Complexity**: It usually requires a very large number of hidden neurons (***hyperparameter*** $L \gg K$) to encode statistical properties of polymorphic samples, and may need a considerable number of iterations for convergence of $w_{ij} = w_{ji}$, $w_{ii} = 0$ amongst all $L + K$ neurons
- **Analogy with Neurophysiology**: Synapses between nodes of similar state tend to be tightly connected (***Hebbian*** Learning). *Positive Phase* $\sim$ Active Cerebral operation; *Negative Phase* $\sim$ Processing during sleeping time of signals acquired while awake (???)

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Boltzmann Machine, BM (5/8)

Definitions

- **Network State**: Random vector $\mathbf{X} \to \mathbf{x} = [x_1 \; x_2 \; \dots \; x_K \; \dots \; x_m]^{\mathrm{T}}, m = L + K$
  $x_i \in \{-1,1\} \triangleq \{OFF, ON\}$ with x_i the state of ***stochastic*** neuron i

- **State of K Visible & L Hidden Neurons**: $\mathbf{X}_\alpha \to \mathbf{x}_\alpha, \; \mathbf{X}_\beta \to \mathbf{x}_\beta, \; \mathbf{x} = (\mathbf{x}_\alpha, \mathbf{x}_\beta)$

- **Synaptic Weights** $i \to j$: $w_{ji} = w_{ij}, w_{ii} = 0$ (a possible external ***bias*** to neuron j is assumed to emanate from a fictitious node 0 in ON state with synaptic weight w_{j0})

- **Energy of BM**: $E(\mathbf{x}) \triangleq -\frac{1}{2}\sum_i \sum_{j \neq i} w_{ji} x_i x_j$ for $\mathbf{x}$ with coordinates $x_i \in \{-1,1\}$
  (***thermodynamic*** analogy)

- **Thermal Equilibrium Probabilities**: $P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \exp\left(-\frac{E(\mathbf{x})}{T}\right)$
  Gibbs/Boltzmann distribution depending on parameters w_{ij} in $E(\mathbf{x})$

- **State of K Visible Neurons**: $\mathbf{X}_\alpha \to \mathbf{x}_\alpha = [x_1 \; x_2 \; \dots \; x_i \; \dots \; x_K]^{\mathrm{T}}$
  Binary coordinates x_i correspond to ***features*** of input/output vectors. The probability that neuron i is ON equals to $P(x_i = 1)$

https://www.cs.toronto.edu/~hinton/csc321/readings/boltz321.pdf

https://youtu.be/5jaBneYd5Ig

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Boltzmann Machine, BM (6/8)

- **Definition of Events for m Dimensional Vector Sample Space**:

For sample vector $[X_1 = x_1 \;\; X_2 = x_2 \;\; \dots \;\; X_j = x_j \;\; \dots \;\; X_m = x_m]^{\mathrm{T}}$ we define the sets (***events***)

A: $(X_j = x_j)$, B: $(X_1 = x_1, \dots, X_{j-1} = x_{j-1}, X_{j+1} = x_{j+1}, \dots, X_m = x_m)$ and

C: $(X_1 = x_1, \dots, X_j = x_j, \dots, X_m = x_m)$ the joint event of A and B

In ***thermal equilibrium*** and for X_j generated via ***Gibbs sampling*** we obtain:

$$P(C) = P(A, B) = \frac{1}{Z}\exp\left(\frac{1}{2T}\sum_i \sum_{j \neq i} w_{ji} x_i x_j\right)$$

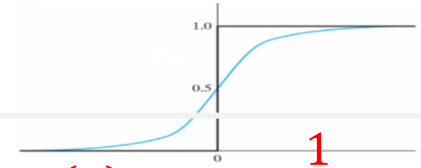
$$P(B) = \sum_A P(A, B) = \frac{1}{Z}\sum_{x_j} \exp\left(\frac{1}{2T}\sum_{i \neq j} \sum_j w_{ji} x_i x_j\right)$$

- **State Transition Conditional Probabilities with Parameters w_{ji}** :

With x_i, x_j values at ± 1 the conditional probability $P(X_j = x_j | B)$ takes a simplified form:

$$P(X_j = x | B) = P(A|B) = \frac{P(A, B)}{P(B)} = \frac{1}{1 + \exp\left(-\frac{x_j}{T}\sum_{i \neq j} w_{ji} x_i\right)}$$

Sigmoid Function



$$\varphi(v) = \frac{1}{1 + \exp(-v)}$$

The joint probability $P(A, B)$ results from ***Gibbs sampling*** starting at $\mathbf{x}(0)$, with $\mathbf{x}(n) \to \mathbf{x}(n+1)$ transitions based on the most recent $x_i(n)$ values and with $T \to 0$ (***Simulated Annealing***)

$$P(X_j = x | B) = P(X_j = x | \{X_1 = x_1, \dots, X_{j-1} = x_{j-1}, X_{j+1} = x_{j+1}, \dots, X_m = x_m\})$$

$$= \varphi\left(\frac{x}{T}\sum_{i=1, i \neq j}^{m} w_{ji} x_i\right)$$

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Boltzmann Machine, BM (7/8)

Boltzmann Learning Rule: Maximum Likelihood or Log-Likelihood Principles

The state vector $\mathbf{x}$ consists of two subsets: $\mathbf{x}_\alpha$ and $\mathbf{x}_\beta$ that reach ***Gibbs*** thermal equilibrium. The **BM *learning*** proceeds in two successive **phases**:

- **Positive Phase** with ***visible neurons*** clamped to input vectors of the training sample $\mathcal{T}$
- **Negative Phase** with ***all*** neurons interacting freely with no outside interference

Synaptic weights w_{ji} (elements of matrix $\mathbf{w}$ of the entire **BM**) lead to limiting equilibrium probabilities ***Gibbs*** of ***visible neurons*** $P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)$ for the sample $\mathcal{T}$. With many elements in $\mathcal{T}$ we can assume that vectors $\mathbf{X}_\alpha$ are ***independent*** random vectors with total equilibrium probability equal to the ***factorial distribution*** (product):

$$\prod_{\mathbf{x}_\alpha \in \mathcal{T}} P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)$$

Sample element probabilities $P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)$ contain all subsets $\mathbf{x}_\beta$ in vectors $\mathbf{x} = (\mathbf{x}_\alpha, \mathbf{x}_\beta)$. Thus:

$$P(\mathbf{X}_\alpha = \mathbf{x}_\alpha) = \frac{1}{Z}\sum_{\mathbf{x}_\beta} \exp\left(-\frac{E(\mathbf{x})}{T}\right), \;\; Z = \sum_{\mathbf{x}} \exp\left(-\frac{E(\mathbf{x})}{T}\right), E(\mathbf{x}) \triangleq -\frac{1}{2}\sum_i \sum_{j \neq i} w_{ji} x_i x_j$$

(Z involves normalization over ***all*** combined states $\mathbf{x}$)

The logarithm $L(\mathbf{w})$ of the ***factorial distribution*** is the ***Log-Likelihood*** function:

$$L(\mathbf{w}) = \log \prod_{\mathbf{x}_\alpha \in \mathcal{T}} P(\mathbf{X}_\alpha = \mathbf{x}_\alpha) = \sum_{\mathbf{x}_\alpha \in \mathcal{T}} \log P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)$$

Determining $\mathbf{w}$ is equivalent to maximizing $L(\mathbf{w})$ with respect to parameters w_{ji} (***maximum-likelihood principle***)

Boltzmann Machine, BM (8/8)

Boltzmann Learning Rule: Maximum Likelihood or Log-Likelihood Principles (*cont.*)

The ***Log-Likelihood*** $L(\mathbf{w})$ is given by:

$$L(\mathbf{w}) = \sum_{\mathbf{x}_\alpha \in \mathcal{T}} \left(\log \sum_{\mathbf{x}_\beta} \exp\left(-\frac{E(\mathbf{x})}{T}\right) - \log \sum_{\mathbf{x}} \exp\left(-\frac{E(\mathbf{x})}{T}\right)\right), \; E(\mathbf{x}) = -\frac{1}{2} \sum_i \sum_{j \neq i} w_{ji} x_i x_j$$

The ***Boltzmann Learning Rule*** maximizes the objective $L(\mathbf{w})$ as a function of all synaptic weights by moving towards directions of ***gradient ascent***:

$$\frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \frac{1}{T} \sum_{\mathbf{x}_\alpha \in \mathcal{T}} \left(\sum_{\mathbf{x}_\beta} P(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha) x_j x_i - \sum_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}) x_j x_i \right) \triangleq \frac{1}{T}\left(\rho_{ji}^{+} - \rho_{ji}^{-}\right)$$

ρ_{ji}^{+} denotes the average ***firing rate*** or the ***correlation*** between states of neurons $j \leftrightarrow i$ in the **Positive Phase** while ρ_{ji}^{-} the ***correlation*** between states of neurons $j \leftrightarrow i$ in the **Negative Phase**

The algorithm proceeds to weight updates in a ***batch mode***, with all training sample elements of $\mathcal{T}$ considered in each iteration. Updates proceed with ***fixed*** step $\boldsymbol{\epsilon}$:

$$\Delta w_{ji} = \boldsymbol{\epsilon} \frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \eta\left(\rho_{ji}^{+} - \rho_{ji}^{-}\right)$$

The ***learning rate*** $\eta = \frac{\boldsymbol{\epsilon}}{T}$ is ***inversely proportional*** to T in the ***Simulated Annealing cooling*** steps

Boltzmann Machines were an early theoretical break-through by ***J. Hinton*** and colleagues towards ***Generative AI***. However, they suffer from great complexity and slow convergence
Improvements: Restrictive Boltzmann Machines (RBM) & Deep Belief Networks (DBN)

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Discriminative & Generative Classification

Traditional Discriminative Model of Supervised Learning

It is based on ***conditional probabilities*** $P(y|x)$ directly estimated from (labeled) training sample data (e.g. ***Logistic Regression*** and ***Back-Propagation Algorithm***). A ***new*** input observable element x is assigned to target output y based on the highest $P(y|x)$ inferred during training and presumably ***generalizable*** for unseen sample elements

Generative Model

It is based on ***joint probabilities*** $P(x,y)$ as the relative frequency of joint appearances of observable x and target y. Conditional ***posterior*** probabilities $P(y|x) = \frac{P(x,y)}{P(x)} = \frac{P(x|y)P(y)}{P(x)}$ are evaluated via ***Bayesian*** reasoning. The ***evidence*** probability $P(x) = \sum_y P(x,y)$ results from sums of joint probabilities. Note that the other terms in ***Bayes*** formula are referred to as the ***prior*** probability $P(y)$ and ***likelihood*** $P(x|y)$. Pairs (x,y) can be ***generated*** or ***corrected*** according to $P(x,y)$ estimated during the ***training phase*** and ***generalized*** to reflect statistics of specific use cases

Example: $x \in \{1,2\}, y \in \{0,1\}$ (https://en.wikipedia.org/wiki/Generative_model)

$P(x,y)$	$y=0$	$y=1$
$x=1$	$1/2$	0
$x=2$	$1/6$	$2/6$

$\Rightarrow$

$P(y\|x)$	$y=0$	$y=1$
$x=1$	$\mathbf{1}$	0
$x=2$	$2/6$	$\mathbf{4/6}$

$P(x=1) = 1/2,\ P(x=2) = 3/6 = 1/2$

Used for cases of deficient datasets (e.g. gaps in images, noisy voice signals) requiring corrective actions based on inferred statistical correlations, e.g. ***Boltzmann Machine***

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

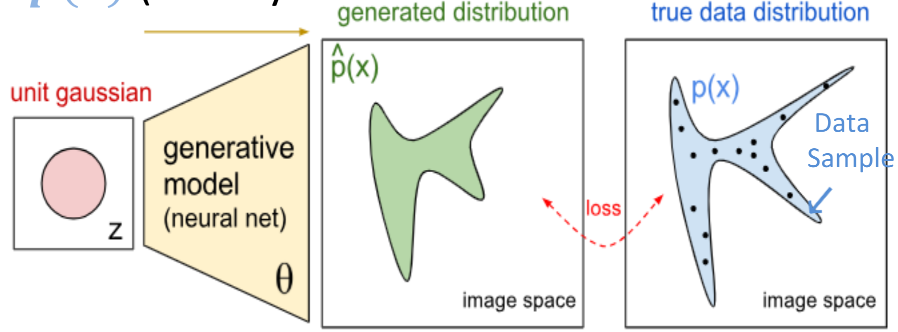
Statistical Aspects of Generative Models

https://openai.com/blog/generative-models/

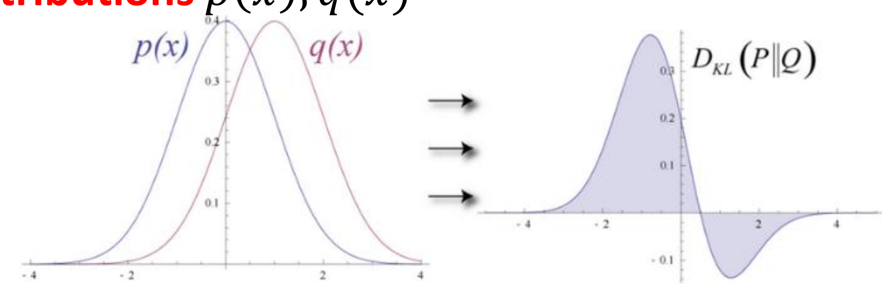
$p(x)$: Distribution of ***Training Sample*** with n elements $x \in \{x_1, x_2, \dots, x_n\}$

$\hat{p}_\theta(x)$: Distribution of ***Generated Sample*** at the ***output*** of a neural network with parameters θ and arbitrary ***Input Sample*** Z, e.g. 100 random numbers with ***Gaussian*** distribution,

Learning Phase: Tuning of parameters θ of the neural network based on ***Training Sample*** elements so that $\hat{p}_\theta(x) \to p(x)$ (usually as in the ***Kullback-Leibler*** divergence metric)



Similarity Metrics of Distributions $p(x), q(x)$



- **Divergence *Kullback-Leibler*** (***KL***) (1951):

  $$D_{\mathrm{KL}}(P||Q) = \sum_{x \in \mathcal{T}} P(x) \log \frac{Q(x)}{P(x)}$$

  e.g. applied in *Boltzmann Machine*

  https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence

- **Expectation-Maximization** (***EM***) **Algorithm** :

  Two successive steps (expectation – minimization) to determine ***latent*** parameters

  e.g. determination of percentages of random variables combined from *independent Gauss* samples

  https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Generative Adversarial Network - GAN

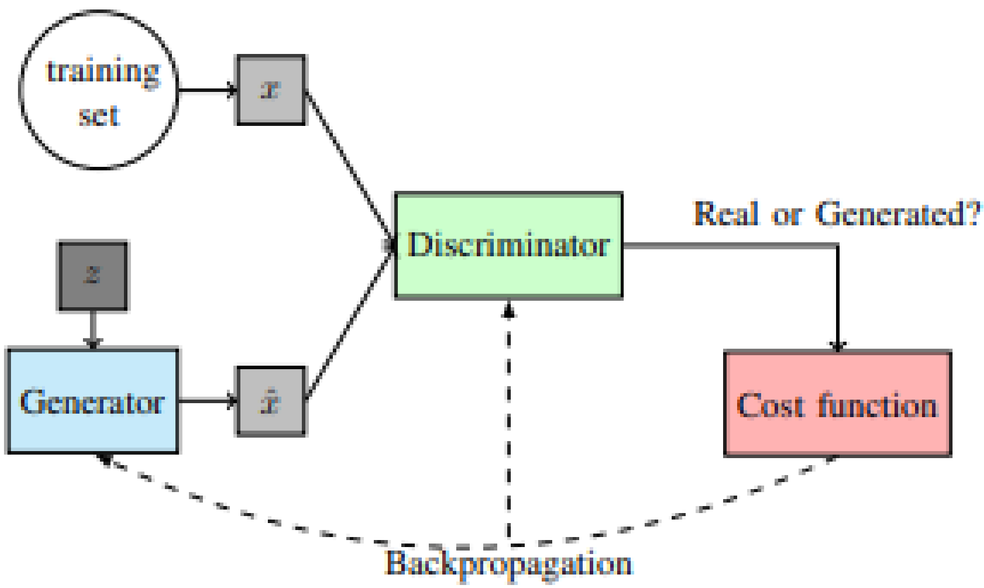
2014 Ian Goodfellow et.al. https://arxiv.org/pdf/1406.2661.pdf

Based on comparisons of outputs processed by two independent ***players*** in a ***zero-sum adversarial min-max game***: Divergence between ***Generated*** and ***Real*** sample elements

Learning is based on tuning of two deep ***Multilayer Perceptrons*** - ***MLP***:

- The **Generator** (**G**) with input ***latent random variables*** z (e.g. ***Gauss***) and output $G(z)$ that ***generates*** a virtual sample element $\hat{x}$, distributed according to $p_\theta(\hat{x})$
- The **Discriminator** (**D**) that classifies via supervised learning (backpropagation) the divergence between ***real*** elements $x \sim p(x)$ and virtual ***generated*** elements $\hat{x} \sim p_\theta(\hat{x})$

For as long as **D** senses the difference between x and $\hat{x}$ it **classifies** the output as ***Generated***. Then player **G** modifies its parameters θ and the game is repeated until **D** is deceived and **classifies** the output as ***Real***



Cost Functions (Loss) for D - G Game:

D: $max \left\{ \log D(x) + \log\left(1 - D\left(G(z)\right)\right) \right\}$

maximize probability $\hat{x}$ classified as fake

G: $min \left\{ \log\left(1 - D\left(G(z)\right)\right) \right\}$

minimize probability $\hat{x}$ classified as fake

Applications: Computer vision, virtual reality, computer graphics, interactive games, scientific simulations, ….

https://en.wikipedia.org/wiki/Generative_adversarial_network