Approfondimento di Intelligenza Artificiale

Tristano Munini

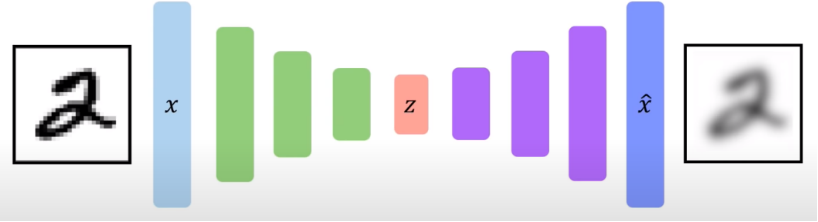
11 ottobre 2020

Indice

- AutoEncoders (AE) e Variational AutoEncoders (VAE)
- Generative Adversarial Network (GAN)
- Reinforcement Learning (RL)
  - QNN
  - Policy Gradient
- GAN + RL per generazione di testi
  - SeqGAN
  - LeakGAN

AutoEncoders

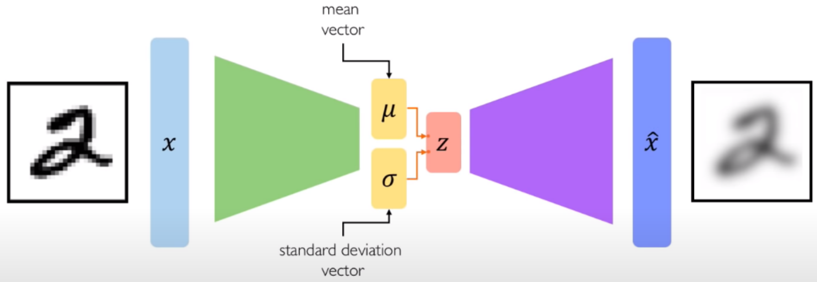
$x \in$ real data space
$z \in$ latent space
$\hat{x} \in$ real data space



The point of maximum compression, where z is produced, is called bottleneck of the network

Variational AutoEncoders

z is sampled from the distribution with mean vector μ and standard deviation vector σ.
We want an $\hat{x}$ that is similar to x but that has some differences.
We want a variation of x.



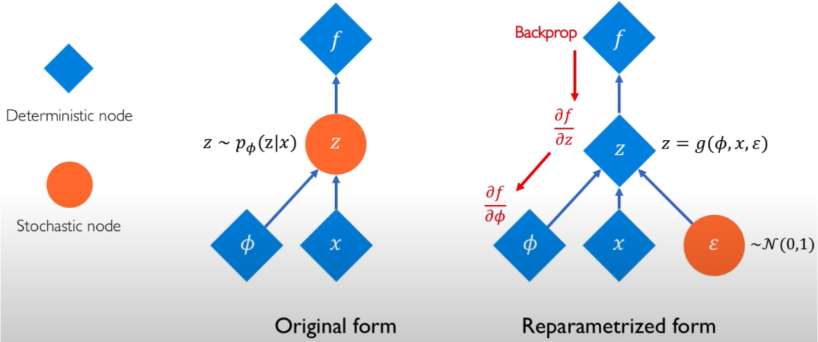
Reparametrization Trick

We can not use $z \sim \mathcal{N}(\mu, \sigma^2)$
So we use

$$z = \mu + \sigma \odot \varepsilon \quad \text{where } \varepsilon \sim \mathcal{N}(0, 1)$$

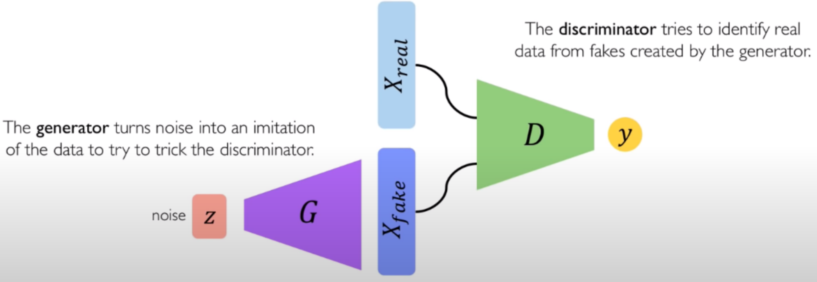
Where $\odot$ notes the element-wise multiplication
Now z is the sum of two fixed vectors, μ and σ, and a random constant ε used as a weight

Reparametrization Trick



Generative Adversarial Networks

GANs are a way to make a generative model by having two neural networks compete with each other



z can be sampled form $\mathcal{N}(0, 1)$ as in the VAEs

GAN Training

It is a min-max game

$$min_G\ max_D\ V(D,G) = \mathbb{E}_{x\sim P_{real(x)}}[logD(x)] + \mathbb{E}_{z\sim P_z(x)}[log(1 - D(G(z))]$$

Where $P_{real(x)}$ is the probability distribution of the real data and $P_z(x) = \mathcal{N}(0,1)$ in our case

GAN Problems

- Hard to converge to a good solution
- Vanishing gradient problem
- Model collapse
- Hard to find equilibrium

Reinforcement Learning

Total Reward (Return)

$$R_t = \sum_{i=t}^{\infty} r_i = r_t + r_{t+1} + \cdots + r_{t+n} + \ldots$$

Discounted Total Reward (Return)

$$R_t = \sum_{i=t}^{\infty} \gamma^i r_i = \gamma^t r_t + \gamma^{t+1} r_{t+1} + \cdots + \gamma^{t+n} r_{t+n} + \ldots$$

Quality Function

The quality function, also called Q-function, is the expected total future reward an agent in state s_t can receive by doing action a_t

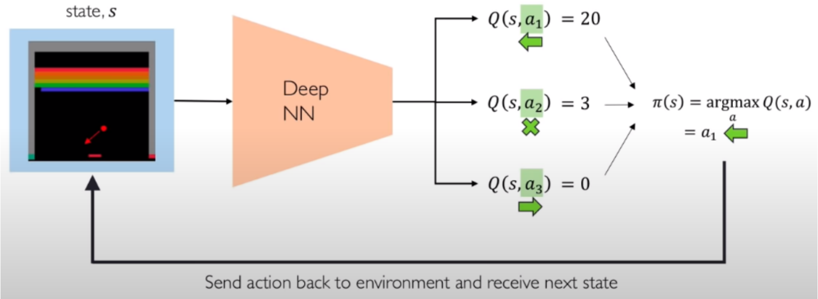
$$Q(s_t, a_t) = \mathbb{E}[R_t | s_t, a_t]$$

The policy that the agent follows is

$$\pi(s_t) = argmax_a Q(s_t, a)$$

QNN

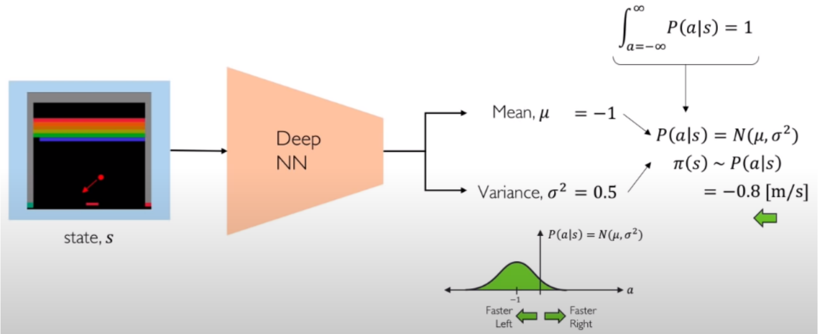
It may be hard to find manually a good Q-Function
We can use an NN to estimate $Q(s_t, a)$ for all possible actions a



This solution is limited to discrete action space

Policy Gradient

What if we need to use an agent within a continuous action space?



Policy Gradient

The loss used in Policy Gradient is

$$loss = -logP(a_t|s_t)R_t$$

so we weight the expected return (total reward in the future) with the probability of taking the action that gives this specific return

Remember that the network chooses the next action using $\pi(s) \sim P(a|s)$

N-Monte Carlo Rollouts

$$loss = -logP(a_t|s_t)R_t$$

R_t is a sum over infinity so we need to approximate it
To do so we can use the Monte Carlo Rollouts:

- use the current policy on current state $\pi(s_t)$ to calculate N possible outcomes to state s_{t+n} in the future
- sum all the N rewards r_i^k where $i = t, \ldots, t+n$ and $k = 1, \ldots, N$
- divide by N, so get the mean of the expected future reward from state s_t with action a_t

REINFORCE

The REINFORCE Algorithm:

1. from current state s_t execute several times $\pi(s_t)$ until termination or state s_{t+n}
2. calculate the mean of the expected future reward
3. use the just calculated R_t in the loss, and update the network
4. sample an a_t from the resulting distribution $\pi(s_t)$, and repeat from 1

Notice that step 1 and 4 are an explore and exploit step, respectively

GAN + RL for text

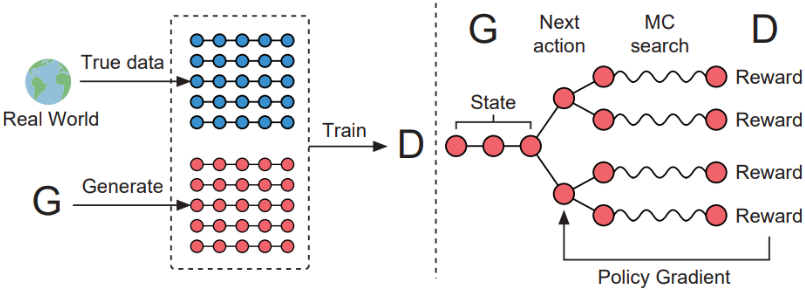
We want to generate long sequences of words (tokens) that resemble human generated sentences
To tackle this difficult problem we use an RNN-based GAN architecture with an RL approach for the Generator

The SeqGAN are the first kind of network presented and can generate sequences up to 20 tokens

The LeakGAN are based on the previous but are more robust and can generate really good results with sentences of 40 words

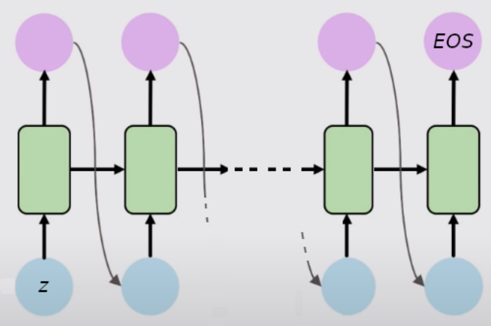
SeqGAN

The Sequential GAN architecture



SeqGAN

The schema of the RNN used inside the Generator of a SeqGAN



The first blue dot is a noise z and the last pink dot is a special "End Of Sequence" token
In the case of SeqGAN EOS is generated after 20 tokens

Loss Function for G

$$J(\theta) = \mathbb{E}[R_t|s_0, \theta] = \sum_{y_1 \in \mathbb{T}} G_\theta(y_1|s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1)$$

where G_θ is the generator with parameters θ and $Q_{D_\phi}^{G_\theta}(s, a)$ is the future reward gained by doing action a in state s following the policy given by G_θ

Note that $\mathbb{T}$ is the action space composed by all the legal tokens (words) and s_0 is the noise vector z

Loss Function for G

We need to estimate $Q_{D_\phi}^{G_\theta}(s_{t-1}, y_t)$
To do so we use an N-Monte Carlo Search

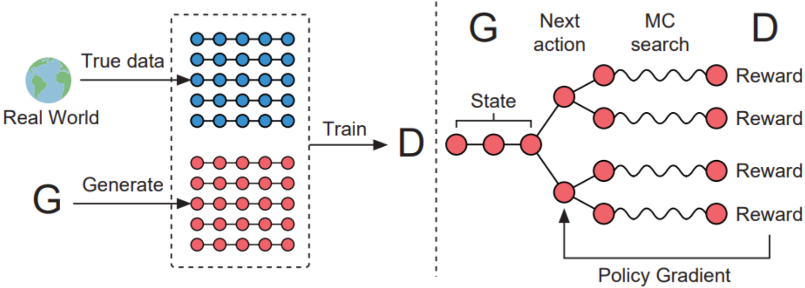
$$\{Y_{1:T}^1, \ldots, Y_{1:T}^N\} = MC(Y_{1:t}; N)$$

where $Y_{1:T}^N$ is a complete sequence composed by a fixed prefix $Y_{1:t}^N$ and a variable suffix $Y_{t+1:T}^N$

So we estimate

$$Q_{D_\phi}^{G_\theta}(s = Y_{1:t-1}, a = y_t) = \begin{cases} \frac{1}{N} \sum_{n=1}^{N} D_\phi(Y_{1:T}^n), \; Y_{1:T}^n \in MC(Y_{1:t}; N) \\ D_\phi(Y_{1:t}) \end{cases}$$

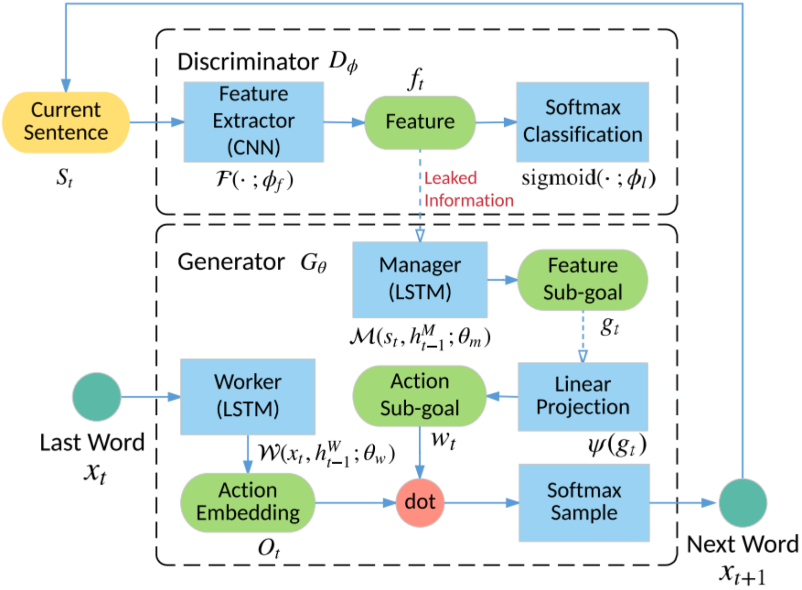
SeqGAN


True data
Real World
Generate
G
Train
D
G
Next
action
MC
search
D
State
Reward
Reward
Reward
Reward
Policy Gradient

SeqGAN Algorithm

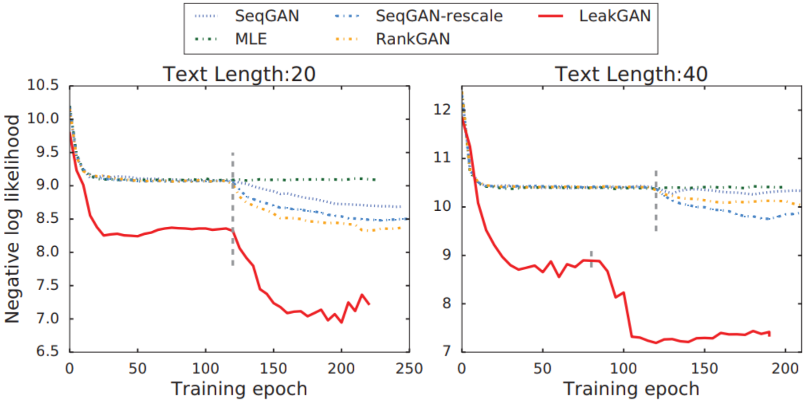
Initialize G_θ, D_ϕ with random weights θ, ϕ
Pre-train G_θ using MLE on real data
Generate negative samples using G_θ for training D_ϕ
Pre-train D_ϕ via minimizing the cross entropy
repeat
  for g-steps **do**
    Generate a sequence $Y_{1:T} = (y_1, \ldots, y_T) \sim G_\theta$
    for t in $1 : T$ **do**
      Compute $Q_{D_\phi}^{G_\theta}(s = Y_{1:T}; a = y_t)$
    end for
    Update generator parameters via policy gradient
  end for
  for d-steps **do**
    Use current G_θ to generate negative examples and combine with given positive examples
    Train D_ϕ for k epochs
  end for
until SeqGAN converges

template



Discriminator D_ϕ
Current Sentence
S_t
Feature Extractor (CNN)
$\mathcal{F}(\cdot\,;\phi_f)$
f_t
Feature
Softmax Classification
$\mathrm{sigmoid}(\cdot\,;\phi_l)$
Leaked Information
Generator G_θ
Manager (LSTM)
$\mathcal{M}(s_t, h^M_{t-1};\theta_m)$
Feature Sub-goal
g_t
Worker (LSTM)
$\mathcal{W}(x_t, h^W_{t-1};\theta_w)$
Action Sub-goal
w_t
Linear Projection
$\psi(g_t)$
Last Word
x_t
Action Embedding
O_t
dot
Softmax Sample
Next Word
x_{t+1}

template



SeqGAN
SeqGAN-rescale
LeakGAN
MLE
RankGAN
Text Length:20
Text Length:40
Negative log likelihood
10.5
10.0
9.5
9.0
8.5
8.0
7.5
7.0
6.5
0
50
100
150
200
250
Training epoch
12
11
10
9
8
7
0
50
100
150
200
Training epoch