Approfondimento di Intelligenza Artificiale

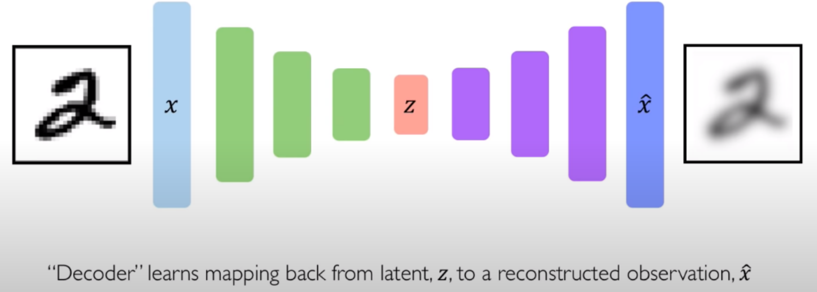
Tristano Munini

11 ottobre 2020

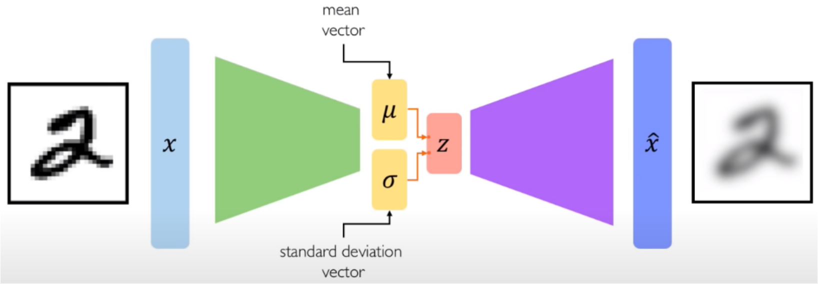
Introduzione

TODO

template

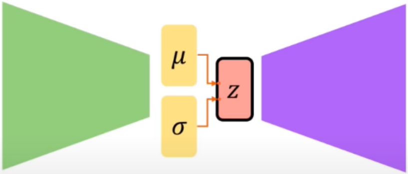

x
z
$\hat{x}$
"Decoder" learns mapping back from latent, z, to a reconstructed observation, $\hat{x}$

template



template

Reparametrizing the sampling layer



Key Idea:

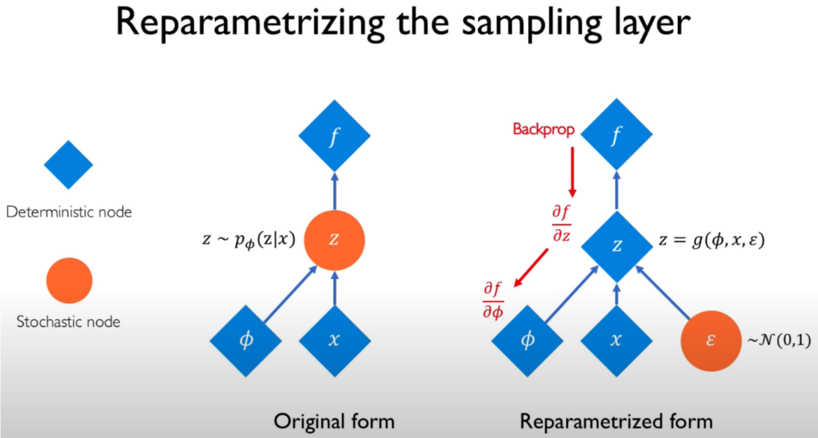
$$\cancel{z \sim \mathcal{N}(\mu, \sigma^2)}$$

Consider the sampled latent vector z as a sum of

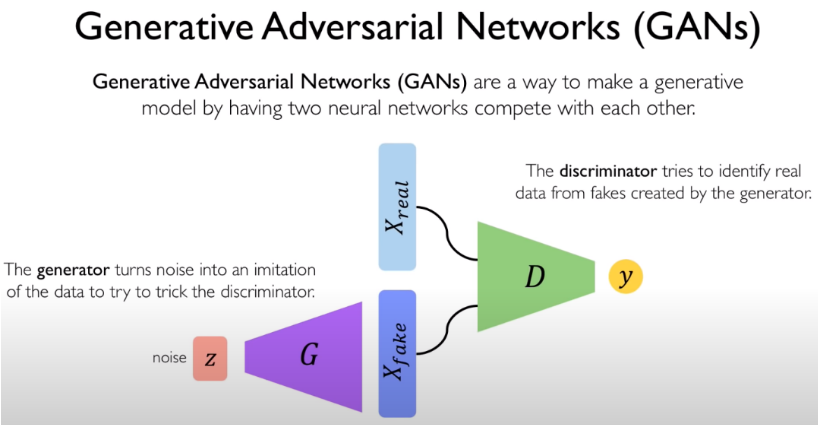
- a fixed μ vector,
- and fixed σ vector, scaled by random constants drawn from the prior distribution

$$\Rightarrow z = \mu + \sigma \odot \varepsilon$$

where $\varepsilon \sim \mathcal{N}(0,1)$


Reparametrizing the sampling layer
Deterministic node
Stochastic node
f
z ~ pϕ(z|x)
z
ϕ
x
Original form
Backprop
f
∂f/∂z
z
z = g(ϕ, x, ε)
∂f/∂ϕ
ϕ
x
ε
~N(0,1)
Reparametrized form

template


Generative Adversarial Networks (GANs)
Generative Adversarial Networks (GANs) are a way to make a generative model by having two neural networks compete with each other.
The discriminator tries to identify real data from fakes created by the generator.
The generator turns noise into an imitation of the data to try to trick the discriminator.
noise
z
G
X_{real}
X_{fake}
D
y

template

Total Reward (**Return**)

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

Discounted Total Reward (**Return**)

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

γ: discount factor; $0 < \gamma < 1$

template

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

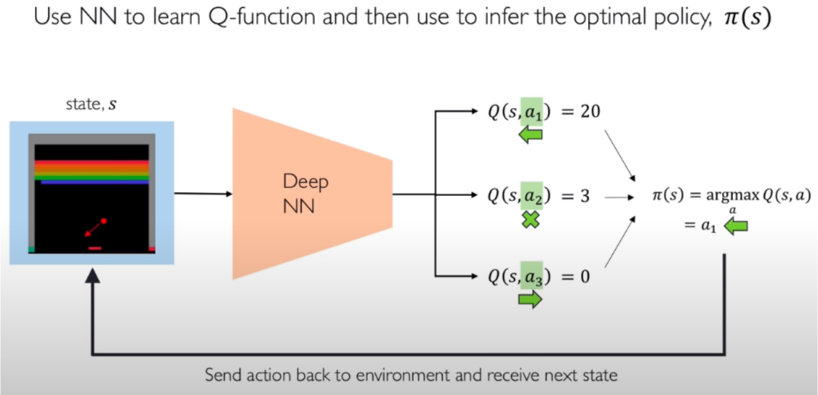
Total reward, R_t, is the discounted sum of all rewards obtained from time t

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

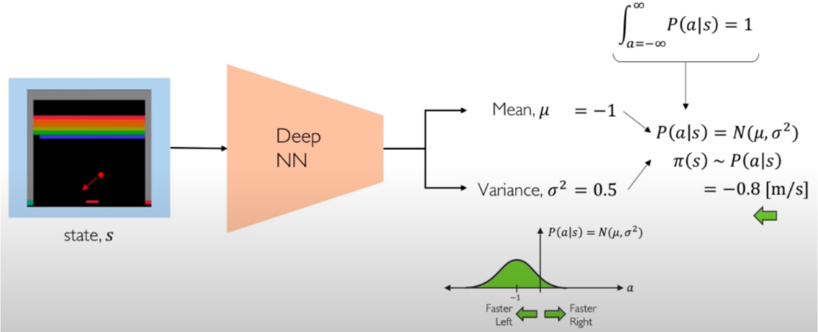
The Q-function captures the **expected total future reward** an agent in state, s, can receive by executing a certain action, a

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

template

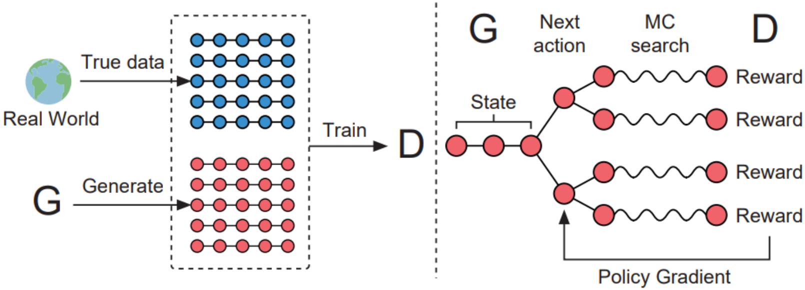

Use NN to learn Q-function and then use to infer the optimal policy, $\pi(s)$
state, s
Deep NN
$Q(s, a_1) = 20$
$Q(s, a_2) = 3$
$Q(s, a_3) = 0$
$\pi(s) = \operatorname{argmax}_a Q(s, a)$
$= a_1$
Send action back to environment and receive next state

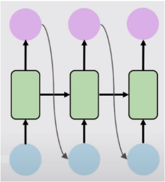
template



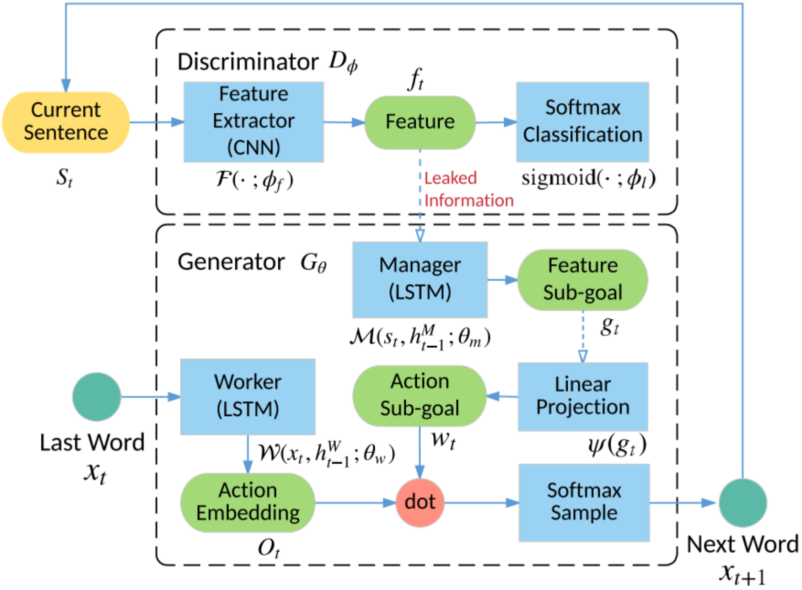
$\int_{a=-\infty}^{\infty} P(a|s) = 1$
Deep
NN
state, s
Mean, μ $= -1$
Variance, σ^2 $= 0.5$
$P(a|s) = N(\mu, \sigma^2)$
$\pi(s) \sim P(a|s)$
$= -0.8$ [m/s]
$P(a|s) = N(\mu, \sigma^2)$
a
-1
Faster
Left
Faster
Right

template


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

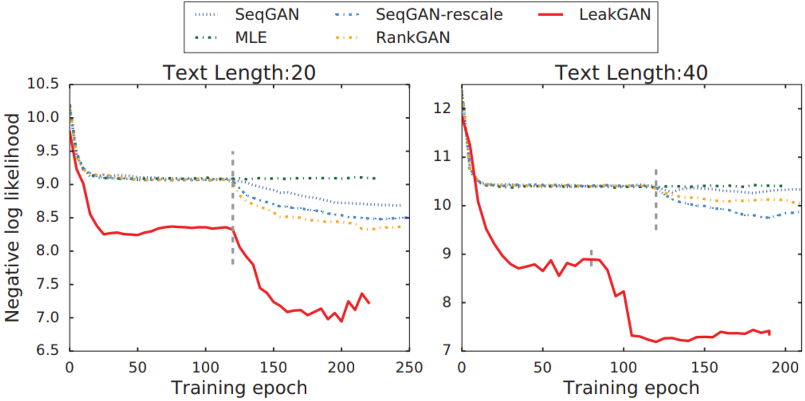


template



Discriminator D_ϕ
Current Sentence
s_t
Feature Extractor (CNN)
$\mathcal{F}(\cdot\,;\phi_f)$
f_t
Feature
Softmax Classification
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
Linear Projection
$\psi(g_t)$
Action Sub-goal
w_t
Worker (LSTM)
$\mathcal{W}(x_t, h^W_{t-1};\theta_w)$
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