Conditional Generation

Conditional Generation

General issue:

A neural network compute a single function.

Can we compute a family of functions instead?
(a function parametric w.r.t. given attributes)

For instance, in generative model, we would like to parametrize generation according to specific attributes

- generate a **given** digit
- generate the face of an **old** man **wearing glasses**
- generate a **red, sportiv car**
- ...

Two issues

- Integrate the condition inside the generative model
- Concrete handling of the condition
(mixing input and condition)

Conditional VAE (CVAE)

Both the decoder $Q(z|X)$ and the decoder $P(X|x)$ are now parametrized w.r.t. a given condition c: $Q(z|X,c)$ and $P(X|z,c)$.

What about the prior?

- We can still work with a single, condition independent prior (e.g. a normal gaussian)
  $\Rightarrow$ simpler, a little more burden on the decoder side
- We can also use a different - possibly learned - prior (e.g. a different Gaussian) for each condition
  $\Rightarrow$ slightly more complex; not clearly beneficial

Conditional VAE (CVAE)

Both the decoder $Q(z|X)$ and the decoder $P(X|x)$ are now parametrized w.r.t. a given condition c: $Q(z|X,c)$ and $P(X|z,c)$.

What about the prior?

- We can still work with a single, condition independent prior (e.g. a normal gaussian)
  $\Rightarrow$ simpler, a little more burden on the decoder side
- We can also use a different - possibly learned - prior (e.g. a different Gaussian) for each condition
  $\Rightarrow$ slightly more complex; not clearly beneficial

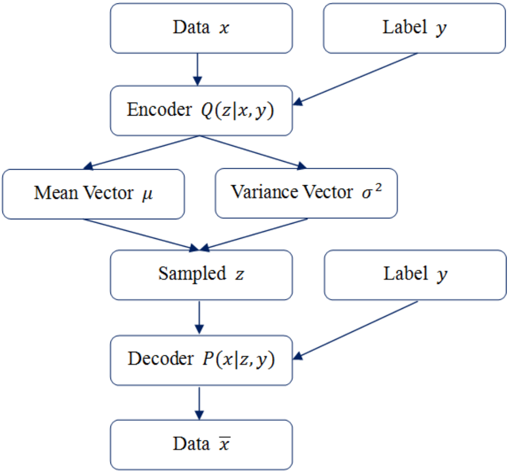
Conditional VAE (CVAE)

Both the decoder $Q(z|X)$ and the decoder $P(X|x)$ are now parametrized w.r.t. a given condition c: $Q(z|X,c)$ and $P(X|z,c)$.

What about the prior?

- We can still work with a single, condition independent prior (e.g. a normal gaussian)
  $\Rightarrow$ simpler, a little more burden on the decoder side
- We can also use a different - possibly learned - prior (e.g. a different Gaussian) for each condition
  $\Rightarrow$ slightly more complex; not clearly beneficial

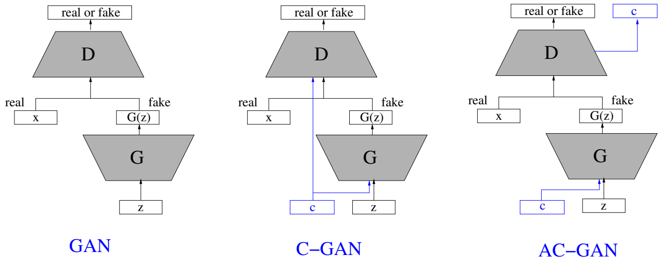
CVAE architecture



Conditional GANs

The generator takes in input the condition, in addition to the noise.

What about the discriminator?



- use the condition to discriminate fakes for real of the given class (Conditional GAN)
- try to classify w.r.t different conditions in addition to true/fake discrimination (Auxiliary Classifier GAN)

AC-GAN loss function

Notation:

- $p^*(x, c)$ is **true** image-condition joint distribution
- $p_\theta(x, c)$ is the joint distibution of **generated** data
- $q_\theta(c|x)$ is the **classifier**

In addition to the usual GAN objective, we also try to minimize the following quantities:

$$\underbrace{-\mathop{\mathbb{E}}_{p^*(x,c)} ln(q_\theta(c|x))}_{\textit{term } 1} \underbrace{-\mathop{\mathbb{E}}_{p_\theta(x,c)} ln(q_\theta(c|x))}_{\textit{term } 2}$$

term 1: the classifier should be consistent with the real distribution
term 2: the generator must create images easy to classify

AC-GAN vs InfoGan

AC-Gans are closely related to InfoGan.
In InfoGan, we only have the first term:

$$\underbrace{-\mathop{\mathbb{E}}_{p^*(x,c)} ln(q_\theta(c|x))}_{term\ 1} \underbrace{-\mathop{\mathbb{E}}_{p_\theta(x,c)} ln(q_\theta(c|x))}_{term\ 2}$$

The second term helps to generate images far from boundaries between classes, hence, likely more sharp.

But what if real images **are** close to boundaries?
Suggested reading: AC-GAN learns a biased distribution

Next arguments

Concrete handling of the condition

handling the condition

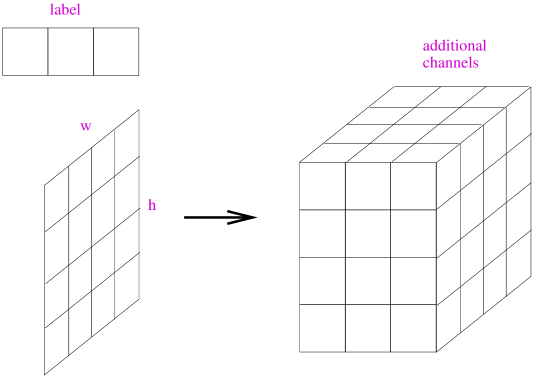
In conditional networks, we pass the label/condition as an additional input.
How is this input going to be processed?

If we need to add it to a dense layer, we just concatenate the label to the input.

If we need to add it to a convolutional layer, we have two basic ways:

- Vectorization
- Feature-wise Linear Modulation (FILM)

vectorization



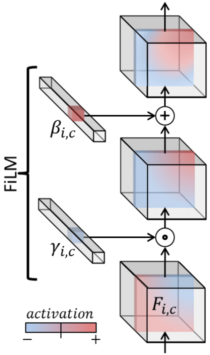
repeat the label (typically in categorical form) for every input neuron, and stack them as new channels

Feature-wise Linear Modulation

Idea: use the condition to give a different weight to each feature (each channel)

Use the condition to generate two vectors γ and β with size equal to the channels of the layer.

rescale layers by γ and add β



Less invasive than parametrizing the weights.