Generative adversarial networks and Variational AutoEncoders

Serena Palazzo

Data Science and Machine Learning course

November 16, 2020

THE UNIVERSITY of EDINBURGH

Lesson from last lecture (checkpoint 5)

- Very good job with checkpoint 5!
- This afternoon we will do checkpoint 7
- Checkpoint 7 deadline **Friday November 20 2020**

Outline

- Generative models
  - Variational auto encoders (VAE)
  - Generative adversarial networks (GAN)
- Description of project 1

Generative models

Generative algorithms

- Generative algorithms are part of unsupervised learning techniques
- A generative model can learn to mimic any distribution of it.
- They are powerful because can learn to reproduce similar models in any domain
- some of these domains are:
    - Images
    - Music
    - Speech
    - Text
    - Videos

Discriminative vs generative algorithms

- **Discriminative models** aim to predict the label to which the data belongs.
  - it aims to map features to label
- **Generative models** on the other hand do the opposite: they aim to predict features given a certain label.

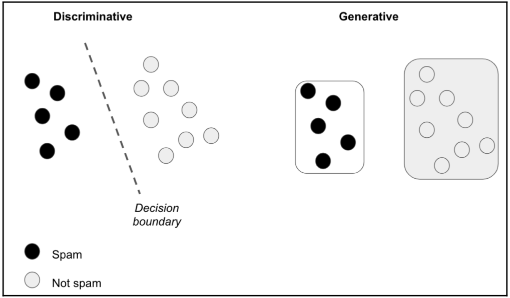
Discriminative vs generative: spam emails

- Let's consider a practical example in the context of whether an email is spam or not
- We can consider **x** the model feature
  - for example all the words in an email
- We can consider **y** the target variable
  - whether the email is actually spam
- The discriminative and generative models will aim to answer the following questions:
  - **Discriminative p(y|x)**: Given the input feature x what is the probability of the email being spam?
  - **Generative p(x|y)**: Given that the email is spam how likely are the input features to be x?

Discriminative vs generative: spam emails

In other words

- **Discriminative models** learns the boundary between classes
- **Generative models** learn to model the distribution of individual classes



Bayes Theorem

- The generative model learns to predict the joint probability with the help of the following Bayes theorem:
$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

- p(y|x) is a conditional probability, the likelihood of y, given x
- p(x|y) is a conditional probability, the likelihood of x, given y
- p(x) and p(y) are the probabilities of observing x and y independently of each other

Examples of discriminative and generative classifiers

Discriminative classifiers

- Logistic regression
- Nearest neighbors
- Support vector machines (SVMs)

Generative classifiers

- Naive Bayes
- Hidden Markow Models (HMMs)
- Bayesian networks

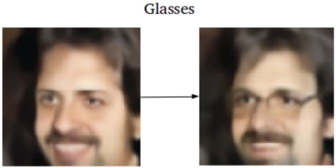
Variational Autoencoders (VAE)

Variational autoencoders

- Variational autoencoders (VAEs) differ from the standard autoencoders.
  - they describe an observation in a **probabilistic** rather than deterministic manner.
  - The output is therefore a probability distribution rather than a single value

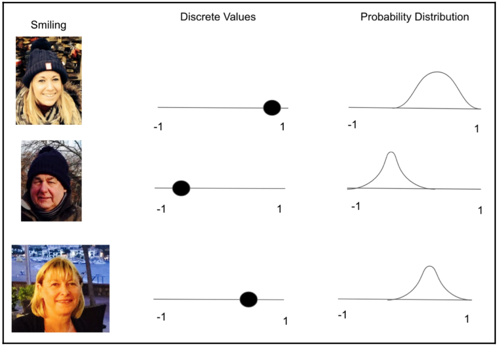
Standard autoencoders vs VAEs

- Standard autoencoders are useful only when we want to replicate data that was input into them $\rightarrow$ limited applications in real world
- VAEs are powerful **generative** models that can be applied to cases where we don't want to output data that is the same as the input data
  - For example are useful for exploring a specific variation of input data



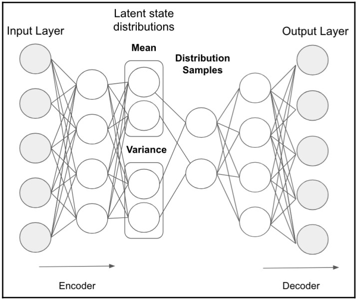
VAE in a real-world context

- Using VAEs allows to describe each attribute in probabilistic term.
  - each feature will be within a range of possible values



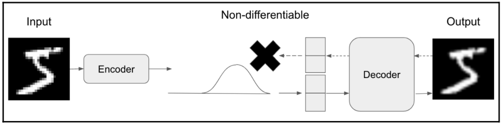
VAE: latent feature

- The distribution of each latent attribute is sampled from the image to generate the vector that is used as input for the decoder model
- It is assumed that the distribution of each feature is Gaussian.
  - therefore 2 vectors are output: one describes the mean and the other the variance distributions



Training VAEs

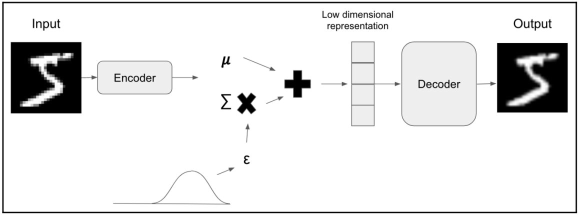
- When training a VAE it is necessary to calculate the relation of each parameter of the network wrt the overall loss, i.e. do backpropagation
- Standard Autoencoders use backpropagation, not easy for VAE since the sampling operation is not differentiable



Reparameterization trick: $z = \mu + \sigma\epsilon$

- Sample ϵ from a unit normal distribution, shift it by mean μ of the latent feature and scale it by the latent attributes' variance σ

↬ With the introduction of the reparameterization trick, we can now train the model via simple backpropagation



- Define sampling with reparameterization trick

```
def sample_z(args):
mu, sigma = args
batch = K.shape(mu)[0]
dim = K.int_shape(mu)[1]
eps = K.random_normal(shape=(batch, dim))
return mu + K.exp(sigma / 2) * eps
```

- We then use this with a Lambda to ensure that correct gradients are computed during the backwards pass based on our values for mu and sigma

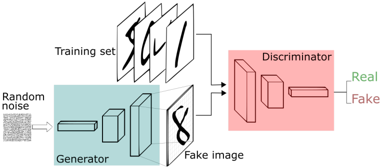
```
z = Lambda(sample_z, output_shape=(latent_dim, ), name='z')([mu, sigma])
```

Generative adversarial networks

Generative Adversial Networks (GANs)

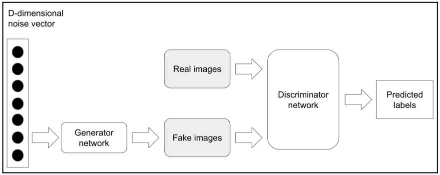
GANs are generative models that try to learn the model to generate the input distribution as realistically as possible.

- A GAN is an unsupervised learning technique and consists of two neural networks:
  - A generative model **Generator** (G) generates new data points from some random uniform distribution.
  - A discriminative model **Discriminator** (D) identifies fake data produced by Generator from the real data.



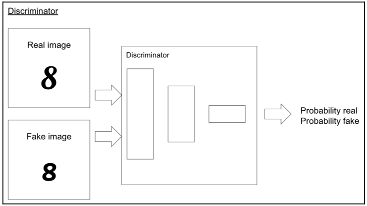
Steps that a GAN takes

1. Random numbers are fed into generator and an image is generated
2. The generated image is fed into the discriminator along with other images taken from the real dataset
3. The discriminator considers all of the images fed into it and returns a probability as to wheter it thinks the image is real or fake



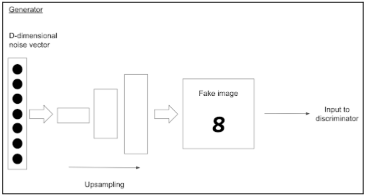
Discriminator network

- The discriminator network is simply a standard convolutional network that categorises images being feed to it.
- It performs downsampling and classifies the images in a binary way, labeling each image as real or fake



Generator network

- The generator network is essentially the reverse of a convolutional network.
- It takes the random noise and performs upsampling in order to output the image



MinMax Game on GANs

- **Generator network:** try to foll the discriminator by generating real-looking images
- **Discriminator network:** try to distinguish between real and fake images

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

- $D(\theta_d)$ wants to maximize objective such that D(x) is close to 1 (real) and $D(G(z))$ is close to 0 (fake)
- $G(\theta_d)$ wants to minimize objective sush that $D(G(z))$ is close to 1
  - discriminator is fooled into thinking generated G(z) is real

Example of Generator model in Keras

```
model = Sequential()
model.add(Dense(128*7*7,input_dim = 100, activation =
LeakyReLU(0.1)))
model.add(BatchNormalization())
model.add(Reshape((7,7,128)))
model.add(UpSampling2D())
model.add(Conv2D(64,(5,5), padding = 'same', activation =
LeakyReLU(0.1)))
model.add(BatchNormalization())
model.add(UpSampling2D())
model.add(Conv2D(1,(5,5), padding = 'same', activation = 'tanh'))
```

Example of Discriminator model in Keras

```
model = Sequential()
model.add(Conv2D(64,(5,5),padding = 'same',input_shape = (28,28,1) ,
activation = LeakyReLU(0.1) ))
model.add(Dropout(0.3))
model.add(Conv2D(128,(5,5),padding = 'same', activation =
LeakyReLU(0.1)))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(1,activation = 'sigmoid'))
```

Training GANs

Because a GAN contains two separetely trained networks, its training algorithm must address 2 complications:

- GANs must juggle two different kinds of training (generator and discriminator)
- GAN convergence is hard to identify

GAN training proceeds in alternating periods:

1. The discriminator trains for one or more epochs
2. The generator trains for one or more epochs
3. Repeat the 2 steps to continue to train the generator and discriminator networks

Training GANs

- We keep the generator constant during the discriminator training phase. This because the discriminator has to learn how to recognize the generator's flaws.
- We keep the discriminator constant during the generator training phase. Otherwise the generator would be trying to hit a moving target and might never converge.

Convergence

- As the generator improves with training, the discriminator performance get worse because it cannot easily tell the difference between real and fake
- If the generator succeeds perfectly, then the discriminator has a 50% accuracy
- This poses a problem for convergence of the GAN: the discriminator feedback gets less meaningful over time
- If the GAN continues training past the point where the discriminator is giving completely random feedback, then the generator starts to train on junk feedbacks and its own quality may **collapse**

Project 1

Project 1: Overview and Submission

- An input dataset will be given and you have to perform some studies
- Perform 3-4 studies on different approaches to classify particles.
- Describe your work in the Jupyter notebook in something like a report form
- When marking the TAs will look for: (rough weights)
    - Data exploration and preprocessing (10%)
    - Model selection (30%)
    - Performance evaluation (20%)
    - Discussion, style throughout (40%)
    - Report notebooks should be submitted by **Friday December 4**

- A brief particle physics background useful for the dataset that you will explore for project 1

The Standard Model

- The building blocks of matter are elementary particles.
- These particles are divided in two major types: quarks and leptons.
- The Standard Model also studies the interaction of these particles through fundamental forces (strong, weak and electromagnetic).
- For every type of particle there also exists a corresponding antiparticle.

Project 1: elementary particles

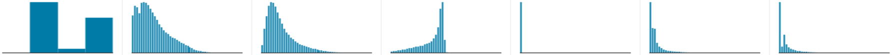
- **Quarks** are fundamental constituents of matter because they combine to form hadrons. There are six quarks paired in three groups: "up/down", "charm/strange" and "top/bottom". They are held together through strong forces.
- **Hadrons** They divide in Baryons and Mesons:
  - **Baryons** are made of three quarks. For example protons are made of (uud) quarks and neutrons are made of (udd) quarks.
  - **Mesons** contain one quark and one antiquark.
- **Leptons** have a 1/2 spin and do not undergo strong interactions. There are six leptons, three of which have an electrical charge. These are: electron, muon and tau. The three remaining are neutrinos. A positron is the antiparticle counterpart of an electron. It possess the same mass and spin but positive charge.

Inelastic scattering

- Is a process used to probe the inside structure of hadrons, in this case protons.
- An incident particle (photoelectron) collides with a target proton. The kinetic energy of the incident particle is not conserved after the collision.
- During inelastic scattering a proton can break up into its constituent quarks which then form a hadronic jet. The angles of the deflection gives information about the nature of the process.

Project 1: Dataset

- The dataset is a simulation of electron-proton inelastic scattering measured by a particle detector system. I
- In order to analyse these data is necessary to recall some concepts.

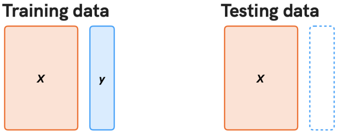


id	# p	# theta	# beta	# nphe	# ein	# eout
particle type	momentum (GeV/c)	theta (rad)	beta	number of photoelectrons	inner energy (GeV)	outer energy (GeV)
-11 – 2212	0.21 – 5.29	0.09 – 2.22	0.55 – 1.5	0 – 349	0 – 0.9	0 – 1.1
211	0.780041	1.08148	0.989962	0	0	0
211	0.260929	0.778892	0.90245	0	0	0
2212	0.773022	0.185953	0.642428	4	0.1019	0
211	0.476997	0.445561	0.951471	0	0	0
2212	2.12329	0.337332	0.908652	0	0.0343786	0.0492561

- Four particle types: positron (-11), pion (211), kaon (321), and proton (2212);
- six detector responses. Some detector responses are zero due to detector inefficiencies or incomplete geometric coverage

Project 1: Training and test data

- You will have two presplit datasets



- Training data has truth labels (y), testing data does not
- The problem is **multiclassification**
- You will have to classify particles type. Your **label** y will be the particle ID

Project 1: Studies that can be performed

- You can perform studies based on all the techniques that you learnt during the course.
- You are free to perform studies that may or may not work. It is important to report and document everything you try and w

Conclusions

- Introduced Generative models
  - Variational autoencoders
  - Generative adversarial networks
- Introduction to project 1

Thanks for your attention and enjoy checkpoint 7 this afternoon!