

AutoEncoder

Advanced Institute for Artificial Intelligence – AI2

<https://advancedinstitute.ai>

Agenda

- ☐ Aprendizagem não-supervisionada
- ☐ Redução de Dimensionalidade
- ☐ Autoencoders
- ☐ Autoencoders e Redes Convolucionais
- ☐ Autoencoders e Sequências
- ☐ Variational Autoencoders

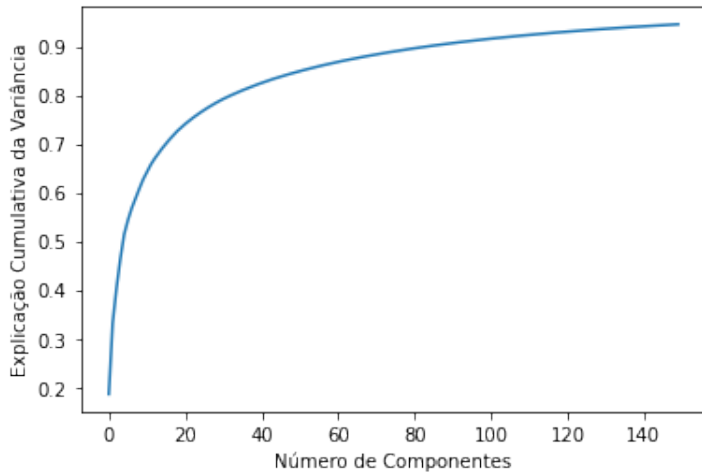
Aprendizagem não-supervisionada

☐ PCA - Principal Component Analysis

- Monta uma representação de um domínio com um número menos de características
- Em muitos domínios uma pequena parte das características é capaz de representar uma parte significativa da variância
- Essa transformação é útil para manipular as características do problema

AutoEncoder

Uma forma de entender o poder do uso do PCA é avaliar a explicação da variância acumulada



Redução de Dimensionalidade

- Trabalhar com uma quantidade menor de informação
 - Melhoria de desempenho e uso de recursos
- Melhorar o desempenho do classificador
- A partir dos componentes gerados é possível gerar a entrada novamente
 - Em geral vai ocorrer perdas e imprecisões nesse processo

Reconhecimento Facial

- ☐ Um exemplo de aplicação de aprendizagem de máquina muito explorado é o reconhecimento facial
- ☐ A Base dados eigenfaces representa cada face usando 2914 pixel (42x67)
- ☐ Todos os pixels são representativos para identificar uma face unicamente?

AutoEncoder

Exemplo de Redução de dimensionalidade e depois retornando a representação original da base eigenfaces



Redes autocodificadoras (autoencoders)

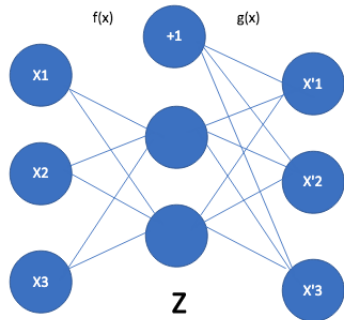
- ☐ Aprendizagem não-supervisionada
- ☐ Representacao concisa da entrada
 - Similar ao modelo PCA
 - Representacao permite reconstruir entrada
- ☐ Características encontradas podem ser usadas para posterior aprendizagem supervisionada
- ☐ A idéia do Autoencoder é reproduzir na saída a própria entrada

Ao reduzir a dimensionalidade da entrada é possível reconstruir a entrada com pouca quantidade de informação

- Alguns exemplos de problemas que podem ser investigados com autoencoders
 - Extração de características
 - Detecção de Outliers
 - Agrupamento
 - Compressão de dados
 - Recuperação de Informação faltante
 - Eliminar ruídos
 - Reconstruir parte faltante de uma sequência

- AutoEncoder não assume a premissa de linearidade
- AutoEncoder sem função de ativação se comporta como PCA
- Transformacoes sao aplicadas na entrada de acordo com dois tipos de funcoes
 - Funcao de extracao de caracteristicas (encoder) mapeia o conjunto de treinamento para uma representacao latente.
 - Funcao de reconstrucao (decoder) mapeia a representacao do espaço latente de volta ao espaço original

AutoEncoder



X representa a entrada

$f(x)=Z$ representa a transformação da entrada para espaço Latente

$g(z)=x$ transforma do espaço latente para entrada novamente

O objetivo é aprender as duas funções minimizando o erro de reconstrução

A entrada e o valor alvo são os mesmos

Espaço Latente subcompleto e sobrecompleto

- Quando o tamanho do espaço latente é menor que a entrada é chamado de subcompleto
- Quando o tamanho do espaço latente é maior que a entrada é chamado de sobrecompleto

Classificação quanto ao tamanho do espaço latente

- Espaço subcompleto
 - Adequado para compressão
 - Identificação de características relevantes na entrada (filtro de características)
- Espaço sobrecompleto
 - Interpolação simples do espaço de busca
 - Dificuldade em aprender características importantes da entrada

Uma forma simples de criar um Autoencoder é usar dois modelos sequencial combinados
Modelo encoder

```
1
2 latent_dim = 100
3
4 input_shape=784
5
6 encoder = Sequential(name="encoder")
7 encoder.add(Dense(128, input_shape=(input_shape,), activation="relu"))
8 encoder.add(Dense(latent_dim)) # Vetor Latente
```

Modelo decoder

```
1
2 decoder = Sequential(name="decoder")
3 decoder.add(Dense(128, activation="relu", input_shape=(latent_dim,) ))
4 decoder.add(Dense(input_shape, activation="sigmoid"))
```

Combinando encoder e decoder

Modelo com base na entrada combina encoder e decoder

```
1
2 input_img = Input(shape=(input_shape,))
3 z = encoder(input_img)
4 recons = decoder(z)
5 ae = Model(input_img, recons)
```


API para construção de redes neurais complexas

- ☐ Classe Sequential facilita a construção de rede Neural no formato de grafo com todas as camadas uma em cima da outra
- ☐ API Funcional permite construir grafos mais complexos
- ☐ Cada camada adicionada pode ter relação com outras camadas

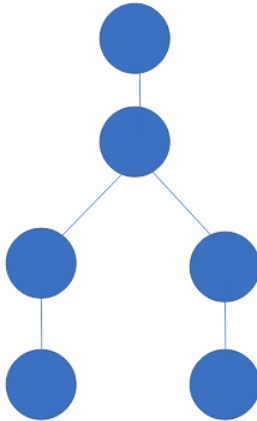
AutoEncoder

Comparação dos tipo de de grafos que é possível criar com cada API

Sequential



Functional



Após declarar uma camada, é possível colocar a relação com outra camada
No exemplo a seguir uma camada está exatamente abaixo da outra

```
1
2 input_img = Input(shape=(62, 47,1)) # adapt this if using 'channels_first
   ' image data format
3 #input_img = Reshape(60,44,1)(input_img2)
4 x = Conv2D(16, (3, 3), activation='relu', padding='same')(input_img)
5 x = MaxPooling2D((2, 2), padding='same')(x)
```

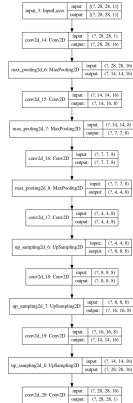
Exemplo de grafo em que uma camada abre para mais de uma camada
Duas camadas diferentes estão associadas a mesma camada chamada encoder

```
1
2 encoder = LSTM(100, activation='relu')(visible)
3 # decoder 1
4 decoder1 = RepeatVector(n_in)(encoder)
5 decoder1 = LSTM(100, activation='relu', return_sequences=True)(decoder1)
6 decoder1 = TimeDistributed(Dense(1))(decoder1)
7 # decoder 2
8 decoder2 = RepeatVector(n_out)(encoder)
9 decoder2 = LSTM(100, activation='relu', return_sequences=True)(decoder2)
10 decoder2 = TimeDistributed(Dense(1))(decoder2)
```

O Modelo de Autoencoder pode ser aplicado a qualquer tipo de Rede Neural

- Vamos avaliar alguns exemplos com Convolução e LSTM
- A figura a seguir mostra uma topologia de convolução para o Mnist
- Essa topologia pode ser adaptada para outros formatos de imagem, alterando o formato de entrada

AutoEncoder

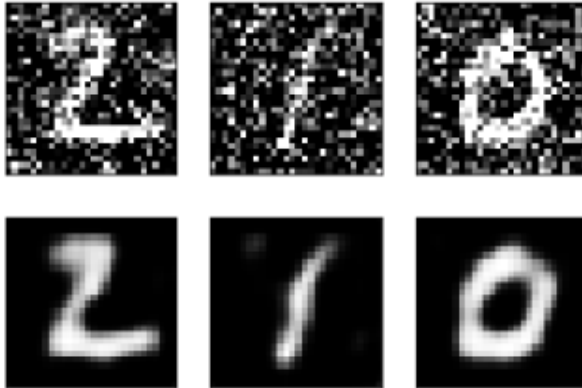


Uma ferramenta do tensorflow para monitorar treino de modelos é o Tensorboard

- ☐ O monitoramento do treino é feito associando a função fit um parametro callback que é um diretório
- ☐ Nesse diretório a informação de treinamento é armazenada e acessada pelo tensorboard em tempo real
- ☐ O tensorboard inicia uma interface gráfica que acompanha em tempo real a evolução do modelo

AutoEncoder

Exemplo de imagem com ruído e imagem tratada



- Uma aplicação prática dessa topologia é limpeza de ruído
- A rede é capaz de reproduzir como saída a própria entrada, considerando apenas o que está no espaço latente
- Dessa forma, a predição elimina os ruídos e pode completar partes faltantes da imagem

- ☐ O autoencoder pode ser usado para gerar como saída a própria entrada
- ☐ O espaço latente pode ser utilizado também como entrada para classificação, assim como o modelo PCA
- ☐ Para isso é necessário separar o encoder do decoder, para ter acesso direto ao espaço latente
- ☐ Para isso é necessário utilizarmos o recurso backend do keras, para ter mais flexibilidade quanto a definição das camadas

O recurso backend do Keras permite controlar de maneira mais detalhada a rede neural em Keras

```
1  
2 from keras import backend as K
```

Outra forma de uso do autoencoder é gerar sequências a partir de uma sequência fornecida

- ☐ Esse modelo é útil para tratar sequências, e também para fazer previsões
- ☐ É possível a partir do espaço latente treinar a geração de outra sequência e a previsão na mesma rede
- ☐ Nesse sentido, é possível gerar um autoencoder capaz de prever o próximo item em uma sequência

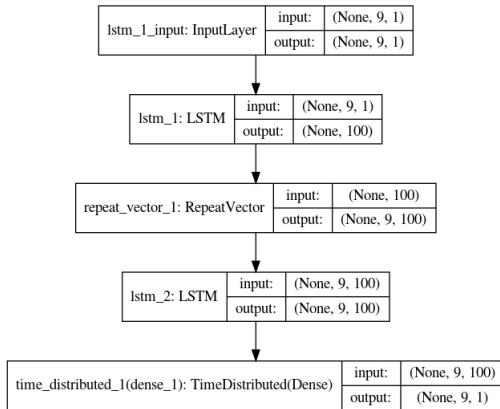
- Outro problema muito explorado combinando autoencoder e sequências é o chamado sequence-to-sequence
- Nesse cenário, o autoencoder aprender a gerar uma sequência a partir de uma sequência fornecida, porém a sequência de saída é diferente da entrada
- Esse modelo é muito usado para tradução, o modelo aprende a gerar uma sequência de palavras em uma língua para outra língua

Dois recursos que facilitam a construção de redes neurais para manipulação de sequências

- RepeatVector: repete a saída da camada anterior, aumentando a entrada para a camada seguinte
- TimeDistributed: aplica uma camada para cada item de uma sequência
 - Dessa forma uma entrada 2D se transforma em 3D
 - Por exemplo, uma entrada de 10 frames de imagens de 120x120 em canal RGB fica com formato (10,120,120,3)
 - Ao usar TimeDistributed a camada é aplicada a cada frame de modo independente

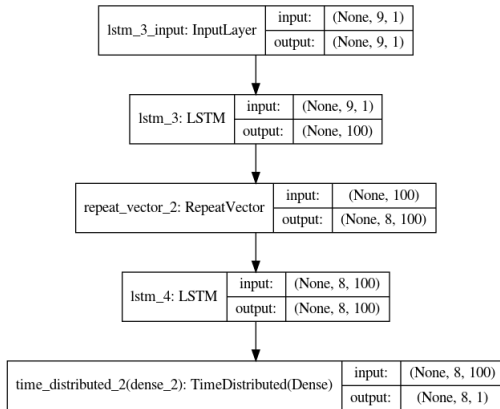
No exemplo a seguir a rede recebe uma sequência e aprende e gera uma nova sequência a partir do espaço latente

AutoEncoder



No exemplo a seguir a rede recebe uma sequência e realiza a predição de cada item a partir do espaço latente

- ☐ A saída da rede representa o próximo elemento de cada sequência de entrada
- ☐ Dessa forma, a entrada possui 9 elementos e a saída 8 elemento
- ☐ Nesse caso o Autoencoder aprender a gerar a saída a partir da entrada, sendo a saída diferente da entrada
- ☐ O efeito é o mesmo do modelo discriminador, porém implementa uma estratégia diferente



A rede neural a seguir combina as duas funcionalidades

