Advanced Institute for Artificial Intelligence

AutoEncoder

Advanced Institute for Artificial Intelligence – AI2

https://advancedinstitute.ai

Agenda

Agenda

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

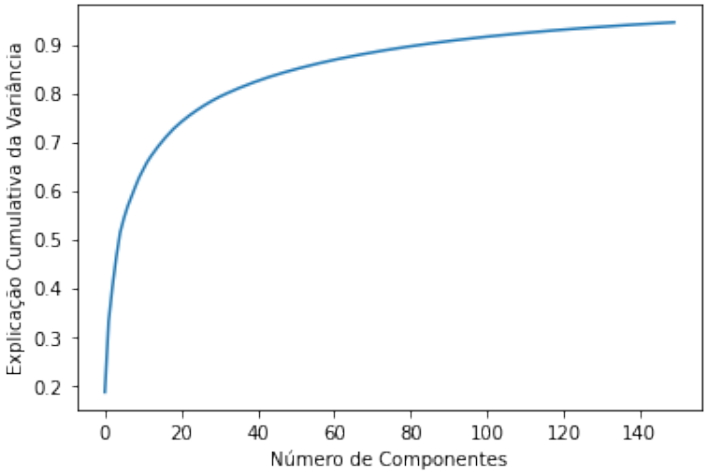
AutoEncoder

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



AutoEncoder

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

AutoEncoder

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

AutoEncoder

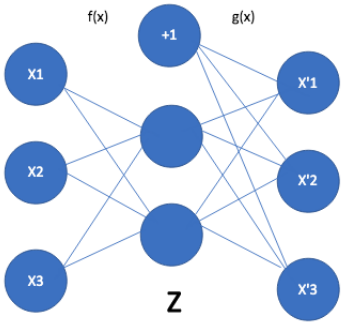
Ao reduzir a dimensionalidade da entrada é possível reconstruir a entrada com pouca quantidade de informação

- 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

- AutoEncoder não assume a premissa de linearidade
- AutoEncoder sem função de ativcação 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

AutoEncoder

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 menor que a entrada é chamado de sobrecompleto

AutoEncoder

Classificação quanto ao tamanho do espaço latente

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

AutoEncoder

Uma forma simples de criar um Autoencoder é usar dois modelos sequential 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
```

AutoEncoder

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"))
```

AutoEncoder

Combinando encoder e decoder

Modelo com base na entrada combina encoder e decoder

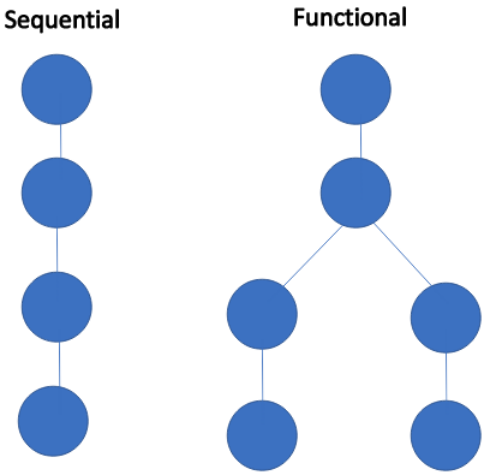
```
1
2  input_img = Input(shape=(input_shape,) )
3  z = encoder(input_img)
4  recons = decoder(z)
5  ae = Model(input_img, recons)
```

AutoEncoder

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

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



AutoEncoder

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)
```

AutoEncoder

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

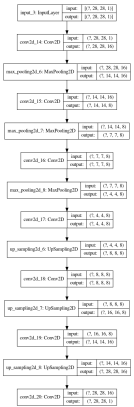
```
1
2  encoder = LSTM(100, activation='relu')(visible)
3  # decoder 1
4  decoder1 = RepeatVector(n_in)(encoder)
5  decoder1 = LSTM(100, activation='relu', return_sequences=True)(decoder1)
6  decoder1 = TimeDistributed(Dense(1))(decoder1)
7  # decoder 2
8  decoder2 = RepeatVector(n_out)(encoder)
9 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

AutoEncoder

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



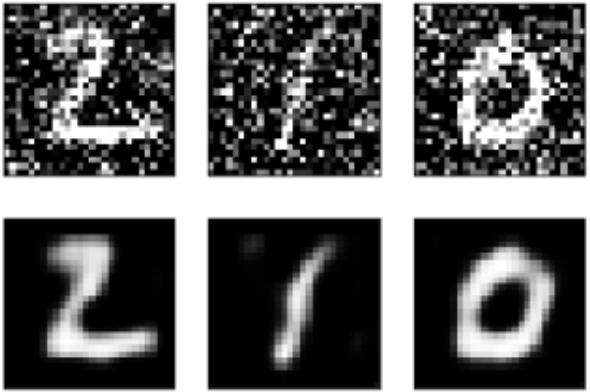
AutoEncoder

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



AutoEncoder

- 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

AutoEncoder

- 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

AutoEncoder

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

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

AutoEncoder

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 predições
- É possível a partir do espaço latente treinar a geração de outra sequência e a predição na mesma rede
- Nesse sentido, é possível gerar um autoencoder capaz de predizer o próximo item em uma sequência

AutoEncoder

- 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 lingua para outra lingua

AutoEncoder

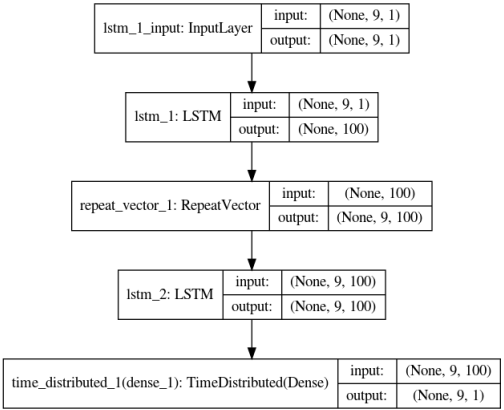
Dois recursos que facilitam a construção de redes neurais para manipulação de sequências

- 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

AutoEncoder

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

AutoEncoder

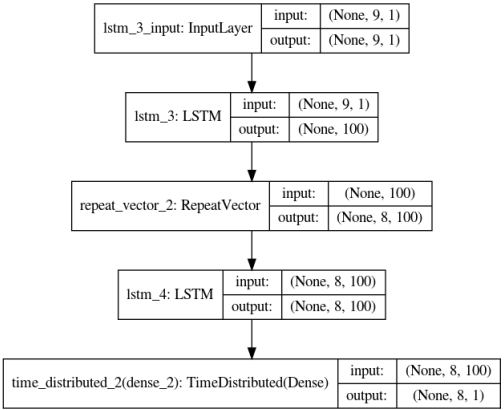


AutoEncoder

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

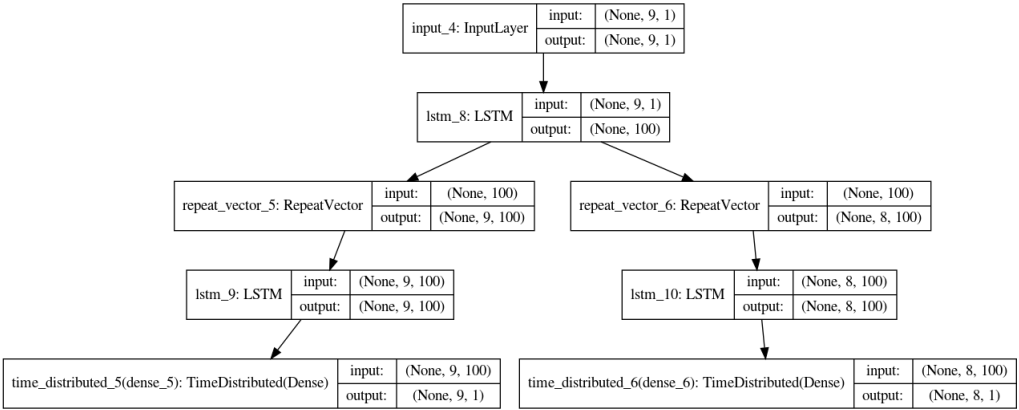
AutoEncoder



AutoEncoder

A rede neural a seguir combina as duas funcionalidades

AutoEncoder



AutoEncoder

- Com AE é possível gerar uma representação concisa da entrada
- O espaço latente possui o que é fundamental para gerar algo novo sem a entrada
- Sem uma organização mínima do espaço latente, é difícil criar uma regra para gerar algo, sem a entrada

AutoEncoder

- Um espaço latente estável poderia ser usado para gerar conteúdo novo, no lugar de gerar a entrada novamente
- A regularidade do espaço latente para autoencoders é um ponto difícil que depende da distribuição dos dados no espaço inicial, da dimensão do espaço latente e da arquitetura do codificador
- Nesse sentido, é difícil que com o espaço latente de um Autoencoder seja compatível com o processo generativo
- Sem uma regularizacao explícita, alguns pontos do espaço latente não possuem significado depois de serem decodificados

VAE (Variational AutoEncoder)

- representa uma ligação entre Autoencoder e modelo generativo
- autoencoder variacional pode ser definido como sendo um autoencoder cujo treinamento é regularizado para:
  - Evitar overfit
  - Garantir que o espaço latente tenha boas propriedades que possibilitem processos generativos

AutoEncoder

Para introduzir regularização do espaço latente é necessário modificar o processo de codificação-decodificação

- em vez de codificar uma entrada como um único ponto, a codificamos como uma distribuição no espaço latente
- Os valores gerados são compatíveis com uma distribuição de probabilidade dos valores da entrada
- A representação dos valores de cada entrada no espaço latente e a distribuição de probabilidades sáo ajustadas continuamente
- Dessa forma é possível que ao pegar um valor no espaço latente, seja possível gerar algum valor, sem possuir o valor original

AutoEncoder

O modelo é treinado da seguinte maneira:

- A entrada é codificada como distribuição no espaço latente
- Um ponto do espaço latente é amostrado a partir dessa distribuição
- O ponto amostrado é decodificado e o erro de reconstrução pode ser calculado
- O erro de reconstrução é retropropagado pela rede

AutoEncoder

- Na prática, as distribuições codificadas são escolhidas para serem normais, de modo que o codificador possa ser treinado para retornar a média e a matriz de covariância que descrevem esses gaussianas.
- A razão pela qual uma entrada é codificada como uma distribuição com alguma variação em vez de um único ponto é que torna possível expressar muito naturalmente a regularização do espaço latente:
- as distribuições retornadas pelo codificador são impostas para estarem próximas a uma distribuição normal padrão.
- Asseguramos desta forma uma regularização local e global do espaço latente (local por causa do controle de variância e global por causa do controle médio).

AutoEncoder

Assim, a função de perda que é minimizada ao treinar um VAE é composta por

- um "termo de reconstrução" (na camada final), que tende a tornar o esquema de codificação-decodificação o mais eficiente possível
- um "termo de regularização" (no camada latente), que tende a regularizar a organização do espaço latente, tornando as distribuições retornadas pelo codificador próximas a uma distribuição normal padrão
  - Esse termo de regularização é expresso como a divergência de Kulback-Leibler entre a distribuição retornada e uma gaussiana padrão
- A divergência de Kullback-Leibler entre duas distribuições gaussianas tem uma forma fechada que pode ser expressa diretamente em termos dos meios e das matrizes de covariância das duas distribuições.

AutoEncoder

- A regularidade esperada do espaço latente para possibilitar o processo generativo pode ser expressa por meio de duas propriedades principais:
- Continuidade (dois pontos próximos no espaço latente não deve fornecer dois conteúdos completamente diferentes uma vez decodificados)
- Completude (para uma distribuição escolhida , um ponto amostrado no espaço latente deve fornecer conteúdo "significativo" depois de decodificado).

AutoEncoder

- A geração da distribuição de probabilidades pode levar a dois cenários:
  - Distribuições com pequenas variações e portando dificil diferencias entre os elementos da entrada
  - distribuições com médias muito diferentes portanto muito distantes no espaço latente, nesse caso, pode encontrar elementos intermediarios a partir da representação da entrada
- Nesse sentido, o modelo precisa ser regularizado com base na matriz de covariância e na média das distribuições retornadas pelo codificador
- Uma forma de fazer isso é impor que a distribuição esteja próxima de uma distribuição normal padrão (centralizada e reduzida)

AutoEncoder

Esse modelo de Rede Neural precisa considerar dois aspectos que não são previstos nas camadas prontas do keras:

- mapear uma distribuição de probabilidades a partir da entrada e calcular a divergência KL dessa probabiblidade a partir de uma distribuição conhecida
- Para isso devemos :
  - Usar uma camada lambda para criar o espaço latente com base em uma distribuição de probabilidades
  - criar um novo loss com base na divergência KL
    - A idéia desse loss é forçar que a distribuição de probabilidades que está sendo criada a cada passada de época seja próxima de uma distribuição pré-definida
    - Esse loss penaliza o modelo quando a divergência é alta