Aprendizagem profunda

Consultoria Especializada em Inteligência Artificial 2

Aula 3 – Começando a usar redes neurais

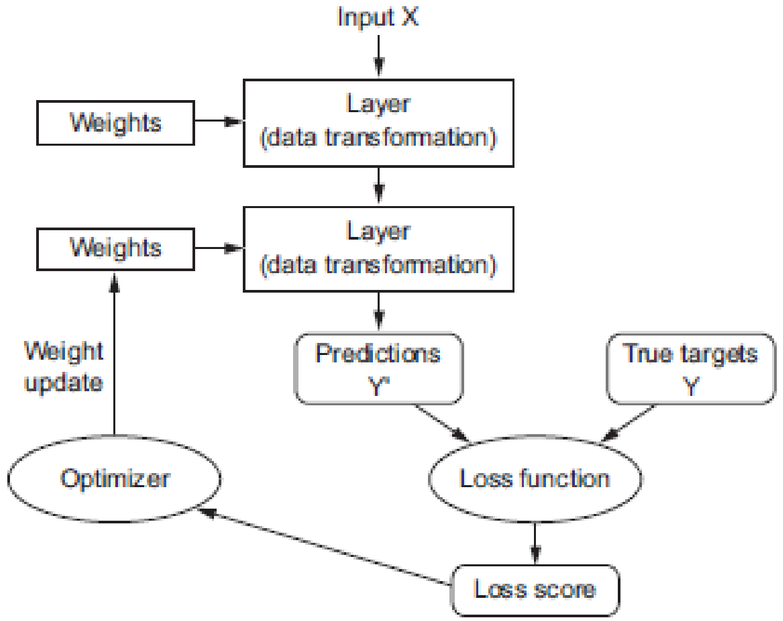
Ciência de Dados e Inteligência Artificial

PUC-SP

Anatomia de uma rede neural

1

Anatomia de uma rede neural



- **Camadas**, que são combinadas em uma **rede** (ou **modelo**)
- **Dados de entrada** e valores-alvo correspondentes
- **Função perda**, que define o sinal de *feedback* usado para o aprendizado
- **Otimizador**, que determina como o aprendizado ocorre

Camadas: as peças-chave da aprendizagem profunda

Via de regra, camadas possuem um **estado**, que são:

- os *pesos* associados à camada;
- um ou mais tensores aprendidos durante o SGD.

e que, em conjunto, contém o **conhecimento** da rede.

Diferentes camadas são apropriadas para formatos diferentes de tensores e tipos diferentes de dados.

Exemplos:

- Vector data `(samples, features)` são em geral processados por camadas **densamente conexas** (também chamadas de **inteiramente conexas** ou camadas **densas**) – a classe `Dense` no Keras.
- Sequence data `(samples, timesteps, features)` são em geral processados por camadas **recorrentes**, tais como `LSTM`.
- Imagens, armazenadas em tensores 4D, são usualmente processadas por camadas convolucionais 2D (`Conv2D`).

Compatibilidade entre camadas

```
from keras import layers
layer = layers.Dense(32, input_shape=(784,))
```

A camada acima só aceitará tensores 2D de entrada em que a primeira dimensão tenha tamanho 784. Essa camada retornará um tensor cuja primeira dimensão foi transformada para 32.

Portanto, essa camada só se conectará com outra camada que aguarda um vetor com tamanho 32. Quando se usa o Keras, porém, não precisamos nos preocupar com isso, pois as camadas já são construídas para serem compatíveis com a anterior.

Compatibilidade entre camadas

```
from keras import models
from keras import layers
model = models.Sequential()
model.add(layers.Dense(32, input_shape=(784,)))
model.add(layers.Dense(32))
```

A segunda camada não recebe o `input_shape` como argumento,

já que ela já sabe qual o `output_shape` da última camada.

Modelos: redes de camadas

Um modelo de aprendizagem profunda é um **grafo direcionado e acíclico** de camadas. A instância mais comum é uma pilha linear de camadas. Mas há outras bastante comuns também, tais como:

- Redes de dois ramos (*two-branch networks*)
- Redes *multihead*
- Blocos de incepção

A topologia de uma rede define um **espaço de hipóteses**. Ao escolher uma topologia, restringe-se o **espaço de possibilidades** a uma série específica de operações tensoriais que mapeiam entradas em saídas.

Só a prática pode te ajudar a encontrar a melhor topologia de uma rede.

Funções perda e otimizadores: chave para configurar o processo de aprendizagem

Função perda (**função objetivo**) – a quantidade que será minimizada durante o treinamento.

Otimizador – determina como a rede será atualizada baseada na função perda.

Uma rede neural que possui múltiplas saídas pode ter múltiplas funções peso (uma para cada saída). Mas a otimização deve-se basear num único escalar. Por isso as funções perdas são combinadas, caso sejam múltiplas.

Há alguns princípios na escolha da função perda, a depender do tipo de problema. Veremos isso adiante.

Introdução ao Keras

Características do Keras

CPU e GPU

Permite que o mesmo código execute em CPU ou GPU.

API amigável

Tornando possível o desenvolvimento rápido de modelos.

Suporte

Suporte integrado para redes convolucionais (para visão computacional) e redes recorrentes (para processamento de dados sequenciais)

Flexibilidade

Suporte para arquiteturas de rede arbitrárias.

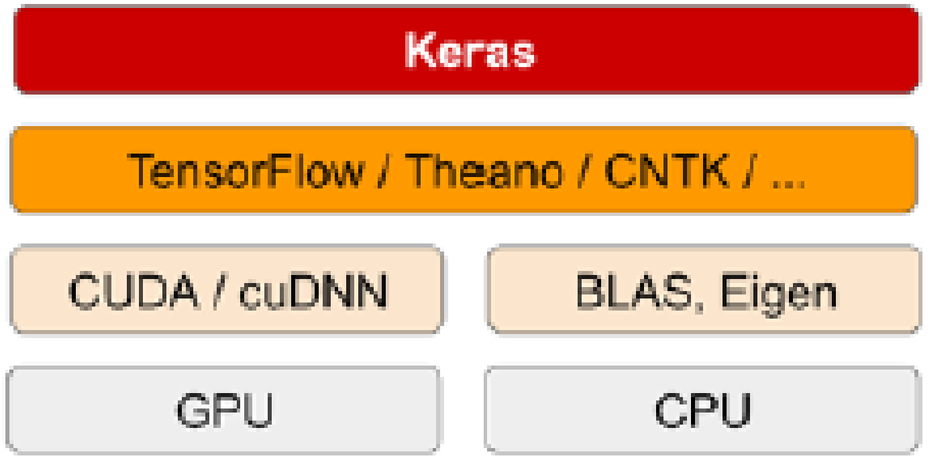
Keras: framework de *deep learning* para Python

Distribuição: Licença MIT permissiva (possibilita uso comercial gratuito)

Compatibilidade: Python 2.6 até versões atuais

Algumas empresas que usam: Google, Netflix, Uber, Yelp, ...

É uma biblioteca a nível de modelagem, capaz de usar diversas *backend engines.*



Desenvolvendo com Keras

Quatro passos:

1. Defina seus **dados de treinamento**: tensores de input e tensores de resultados esperados.
2. Defina uma rede de camadas (ou **modelo**) que mapeia seus inputs em outputs.
3. Configure o processo de treinamento escolhendo uma **função de perda**, um **otimizador** e algumas métricas para monitorar.
4. Chame a função `fit()` para iterar nos seus dados de treinamento até obter os pesos da rede neural.

Classificando resenhas de filmes: um exemplo de classificação binária

Classificação de resenhas de filmes

Base de dados: IMDB. 50k resenhas; 25k para treino e 25 para teste, cada conjunto contendo 50% de resenhas positivas e 50% de resenhas negativas.

A base já é integrada ao Keras e já vem pré-processada: as resenhas (sequências de palavras) foram transformadas em sequências de inteiros, em que cada inteiro representa uma palavra específica em um dicionário.

imdb.ipynb

Pontos para reter

Pré-processamento

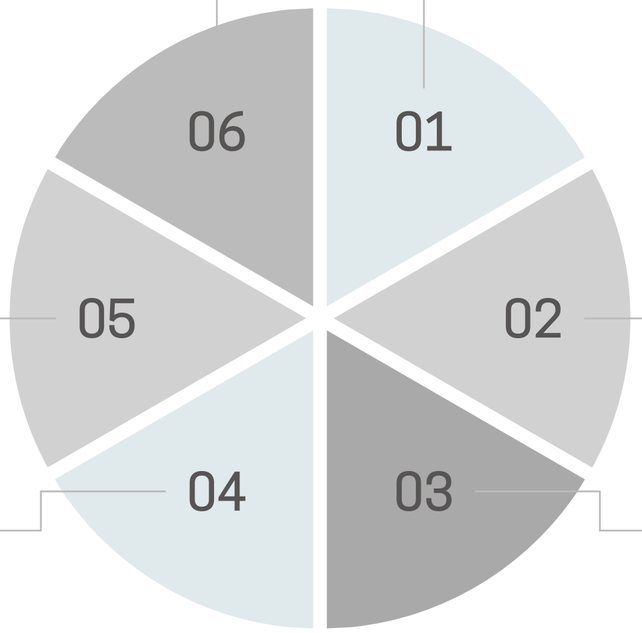
É preciso transformar os dados em tensores para alimentar uma rede neural. Sequências de palavras podem ser codificadas como vetores binários, mas há outras opções de codificação.

Dense e relu

Podem resolver uma ampla variedade de problemas (incluindo classificação de sentimentos).

Classificação binária

Sua rede deve terminar com uma camada Dense com uma unidade e ativação sigmoid: a saída da sua rede deve ser um escalar entre 0 e 1, codificando uma probabilidade.



Overfitting

À medida que melhoram nos dados de treinamento, as redes neurais eventualmente começam a fazer overfitting e acabam obtendo resultados cada vez piores em dados novos

Rmsprop

Geralmente é uma escolha boa o suficiente, qualquer que seja o seu problema. Ou seja: um problema a menos

Binary_crossentropy

Com uma saída sigmoid escalar em um problema de classificação binária, a função de perda que você deve usar é binary_crossentropy .

4

Classificando tópicos de notícias: um exemplo de classificação multiclasse

Classificação de tópicos de notícias

Base de dados: Reuters – conjunto de curtas notícias e seus tópicos, publicadas pela Reuters em 1986. Há 46 tópicos diferentes; alguns tópicos possuem mais notícias que outros, mas cada tópico possui pelo menos 10 amostras nos dados de treinamento.

A base também já é integrada ao Keras.

reuters.ipynb

Pontos para reter

Classificação multiclasse

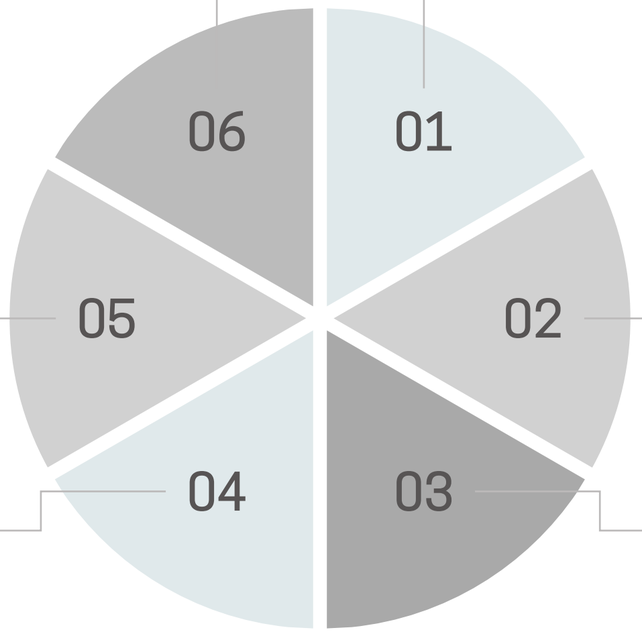
Sua rede deve terminar com uma camada **Dense** de tamanho N.

Softmax

Sua rede deve terminar com uma função de ativação **softmax** para que ela produza uma distribuição de probabilidade sobre as N classes de saída.

Categorial crossentropy

É quase sempre a função de perda que você deve usar para esses problemas. Ela minimiza a distância entre as distribuições de probabilidade produzidas pela rede e a verdadeira distribuição dos rótulos.



One-hot ou categorical encoding

Essa é uma maneira de codificar os rótulos, e deve ser usada junto à categorical_crossentropy como função de perda

Integer encoding

Essa é outra maneira de codificar os rótulos, e deve ser usada junto à sparse_categorical_crossentropy como função de perda

Evitar gargalos

Se você precisa classificar dados em um grande número de categorias, deve evitar criar gargalos de informação em sua rede

Predizendo preços de casas: um exemplo de regressão

Predizendo preços de casas

Em problemas de regressão, queremos predizer um valor contínuo, em vez de um valor discreto indicando um rótulo de classificação.

Base de dados: Boston Housing Price – "Preço médio das casas em um subúrbio de Boston na década de 1970, considerando dados sobre o subúrbio na época, como a taxa de criminalidade, a taxa de imposto local sobre propriedades, etc. Possui poucos dados: 506 (404 para treinamento e 102 para teste). Além disso, cada *feature* nos dados de entrada (por exemplo, a taxa de criminalidade) possui uma escala diferente. Ex., alguns valores são proporções, variando entre 0 e 1; outros variam entre 1 e 12, enquanto outros variam entre 0 e 100, e assim por diante.

boston_housing.ipynb

Validação cruzada k-fold

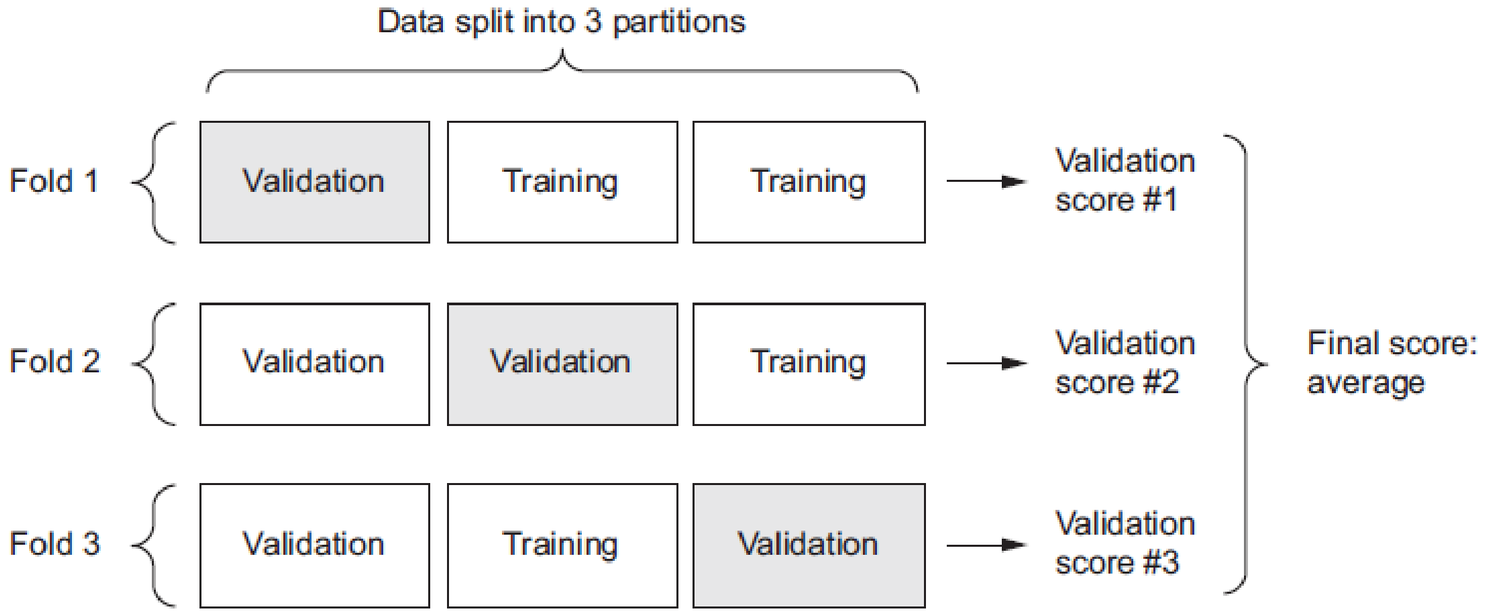
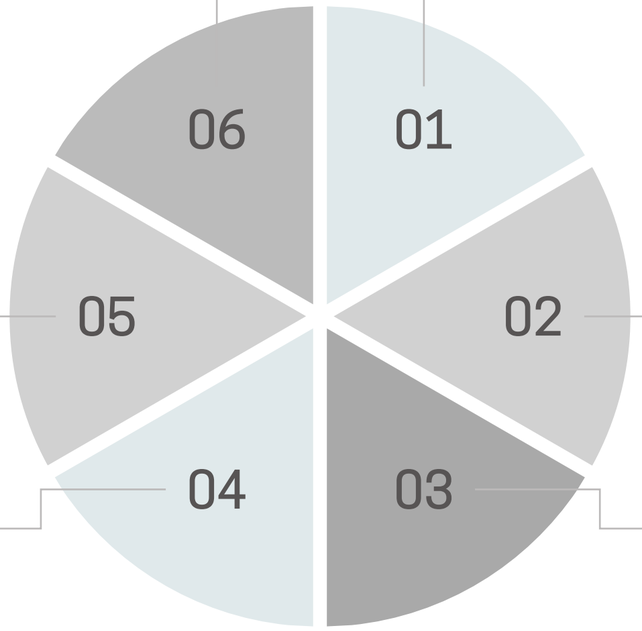


Figure 3.11 3-fold cross-validation

Pontos para reter



MSE (erro quadrático médio)

É a função de perda comumente usada em problemas de regressão.

MAE (erro médio absoluto)

É uma métrica comumente usada em problemas de regressão.

Pré-processamento

Quando as características nos dados de entrada têm valores em diferentes intervalos, cada característica deve ser escalonada.

Evitando overfitting

Quando há poucos dados de treinamento disponíveis, é preferível usar uma rede pequena com poucas camadas ocultas (geralmente apenas uma ou duas) para evitar *overfitting*.

K-fold

Quando há poucos dados disponíveis, usar a validação cruzada K-fold é uma ótima maneira de avaliar confiavelmente um modelo.

Questões?