Keras
Custom Activation Functions
Adam Optimizer

Tóp. Esp. BD III
Christian Cardozo
Prof.: Geraldo Zimbrão

Sumário

Keras

- API de alto nível para redes neurais
- Python
- Suporta como Backend: TensorFlow, CNTK e Theano
- CPU e GPU

Keras - Instalação

- sudo pip3 install keras
- Pacotes opcionais:
    - cuDNN (Keras on GPU).
    - HDF5 and h5py (saving Keras models to disk).
    - graphviz and pydot (used by visualization utilities).

Keras - Instalação

- Por default o Keras vem habilitado para TensorFlow.
- Para mudar, editar o arquivo $HOME/.keras/keras.json

```
{ "image_data_format": "channels_last",
   "epsilon": 1e-07,
   "floatx": "float32",
   "backend": "tensorflow" }
```

- Mudar backend para o desejado: "theano", "tensorflow" ou "cntk"

Keras - Model

- Estrutura central
- Forma com que as layers se organizam
- Sequential Model
- Primeira camada recebe tamanho do input (apenas a primeira)
  - input_shape, input_dim, input_length
- Todas as layers possuem os métodos em comum:
  - get_weights, set_weights, get_config

Keras - Layers

- Dense
  - keras.layers.core.Dense(units, activation=None, use_bias=True, kernel_initializer='glorot_uniform', bias_initializer='zeros', kernel_regularizer=None, bias_regularizer=None, activity_regularizer=None, kernel_constraint=None, bias_constraint=None)
- Activation Layers
  - sigmoid, relu, tanh, softmax, leakyrelu, etc.

Keras - Layers

- Outras Layers no core:
  - Dropout, Flatten, etc
- Convolucionais:
  - Conv1D, Conv2D, ZeroPadding1D, etc
- Polling:
  - MaxPooling1D, MaxPooling2D, AveragePooling1D, etc.
- Outros tipos

Keras - Compile

- model.compile
- Recebe 3 parâmetros:
  - Otimizador: sgd, adam, adamax, etc.
  - Loss Function: mean_squared_error, categorical_crossentropy, etc
  - Lista de métricas: accuracy, categorical_accuracy, etc

Keras - Training

- Método fit
  - fit(self, x, y, batch_size=32, epochs=10, verbose=1, callbacks=None, validation_split=0.0, validation_data=None, shuffle=True, class_weight=None, sample_weight=None, initial_epoch=0)

Keras - Evaluate & Predict

- score = model.evaluate(X_test, y_test)
- classes = model.predict(X_test)

Keras - MNIST Regular Neural Net

- Arquivo: mnist.py
- 10 neurônios; 7840 parâmetros (pesos);
- Ativação: softmax
- Épocas: 10; batch_size: 100
- Gradiente descendente estocástico
- Acurácia no teste: 0.9027 em 1 minuto

Keras - MNIST CNN

- Arquivo: mnist_cnn.py
- Arquitetura:
  input->[conv2d->relu->pool2d]*2->fc->relu->dropout->fc->softmax
- Épocas: 40; batch_size: 50
- Filtros: 5x5 e 5x5
- # Canais de saída: 32 e 64
- Tamanho Pooling: 2x2 e 2x2
- Acurácia no teste: 0.9910 em 40 minutos

Keras - Funções de ativação personalizadas

- Criar uma function e registrar em Activation como custom object
- Camada Lambda
- Layer personalizada (possibilidade de parâmetros treináveis)
- Arquivo: mnist_act.py

Adam - Otimização

- Algoritmo que extende o de gradiente descendente estocástico
- Adam: A Method for Stochastic Optimization (2015) - https://arxiv.org/abs/1412.6980
- Adam = adaptive moment estimation

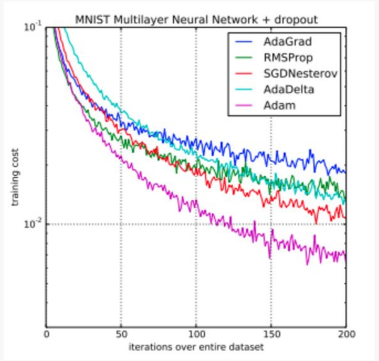
Adam - Otimização

- SGD: maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training.
- AdaGrad: that maintains a per-parameter learning rate.
- RMSProp: that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight.

Adam - Otimização

- Adam realizes the benefits of both AdaGrad and RMSProp
- Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance).

Adam - Otimização



Comparison of Adam to Other Optimization Algorithms Training a Multilayer Perceptron
Taken from Adam: A Method for Stochastic Optimization, 2015.

Adam - Otimização

- Parâmetros default normalmente utilizados:
    - TensorFlow: learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08.
    - Keras: lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0.
    - Blocks: learning_rate=0.002, beta1=0.9, beta2=0.999, epsilon=1e-08, decay_factor=1.
    - Lasagne: learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08
    - Caffe: learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08
    - MxNet: learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-8
    - Torch: learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-8

Referências

- https://keras.io/
- https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/