**Keras**
is a high-level neural networks API,
written in Python and
capable of running on top of **TensorFlow**, CNTK, or Theano.

автор
**François Chollet**
Deep learning researcher at Google

Какие архитектуры поддерживает **Keras**?

feedforward networks
convolutional networks and recurrent networks,
as well as combinations of the two

**модули**

neural layers,
cost functions,
optimizers,
initialization schemes,
activation functions,
regularization schemes

**Keras Sequential**

**Последовательность шагов**
Описать архитектуру сети
Описать входные значения
Описать условия обучения (Compilation)
Обучить (несколько раз?)
Оценить качество модели
Применить

**Compilation**

- **An optimizer**. ( `rmsprop` or `adagrad` )
- **A loss function**. ( `categorical_crossentropy` or `mse` )
- **A list of metrics**. `metrics=['accuracy']`

```
# For a binary classification problem
model.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['accuracy'])

# For a mean squared error regression problem
model.compile(optimizer='rmsprop', loss='mse')
```

## Метод скорейшего спуска (градиентного спуска)

$$argmin\, F(x_{1,}\, x_{2,} \ldots x_k)$$

Выбираем начальную точку $(x_{1,0,}\, x_{2,0,}\, x_{k,0})$
Далее итеративно меняем эту точку по правилу

$$x_{1,i+1} = x_{1,i} - \lambda \frac{\partial F(x_{1,i}, x_{2,i}, x_{k,i})}{\partial x_{1,i}}$$
$$x_{2,i+1} = x_{2,i} - \lambda \frac{\partial F(x_{1,i}, x_{2,i}, x_{k,i})}{\partial x_{2,i}}$$
$$\dots$$
$$x_{k,i+1} = x_{k,i} - \lambda \frac{\partial F(x_{1,i}, x_{2,i}, x_{k,i})}{\partial x_{k,i}}$$

Правило остановки
- число итераций
- малое уменьшение функции

Зависимость от начальной точки (инициализация)

Особенности обучения нейронных сетей: начальное значение близко к нулю (кроме свободных слагаемых)

График зависимости критерия качества Q от номера итерации

Скорость обучения — рекомендованные значения 0.1, 0.001 или 0.0001

Методом проб и ошибок

Малое значение скорости обучения — долгое обучение.
Большое значение скорости обучения — риск того, что последовательность будет нестабильна. Овраги.

Входные значения стандартизируют. Иначе риск насыщения.
Не забыть стандартизировать и новые значения тоже!

Скорость обучения должна убывать.
Точнее - скорость обучения должна быть адаптивной.

Stochastic gradient descent (коррекция весов после каждого наблюдения)
SGD

**Улучшения метода скорейшего спуска**

Momentum
Nesterov  Momentum
Adam

**эпохи и батчи**

Для обучения может хватить <10 эпох

Когда закончится эпоха, порядок наблюдений меняют

Batch (коррекция весов после каждой эпохи/batch)

Mini-batch (коррекция весов после каждого Mini-batch)

При этом складывают градиенты или усредняют их

Насыщение

Взрыв  Exploding Gradients
gradient clipping

https://keras.io/initializers/

https://keras.io/optimizers/#usage-of-optimizers