Введение в нейронные сети. Урок 2. Keras

GeekBrains

План вебинара

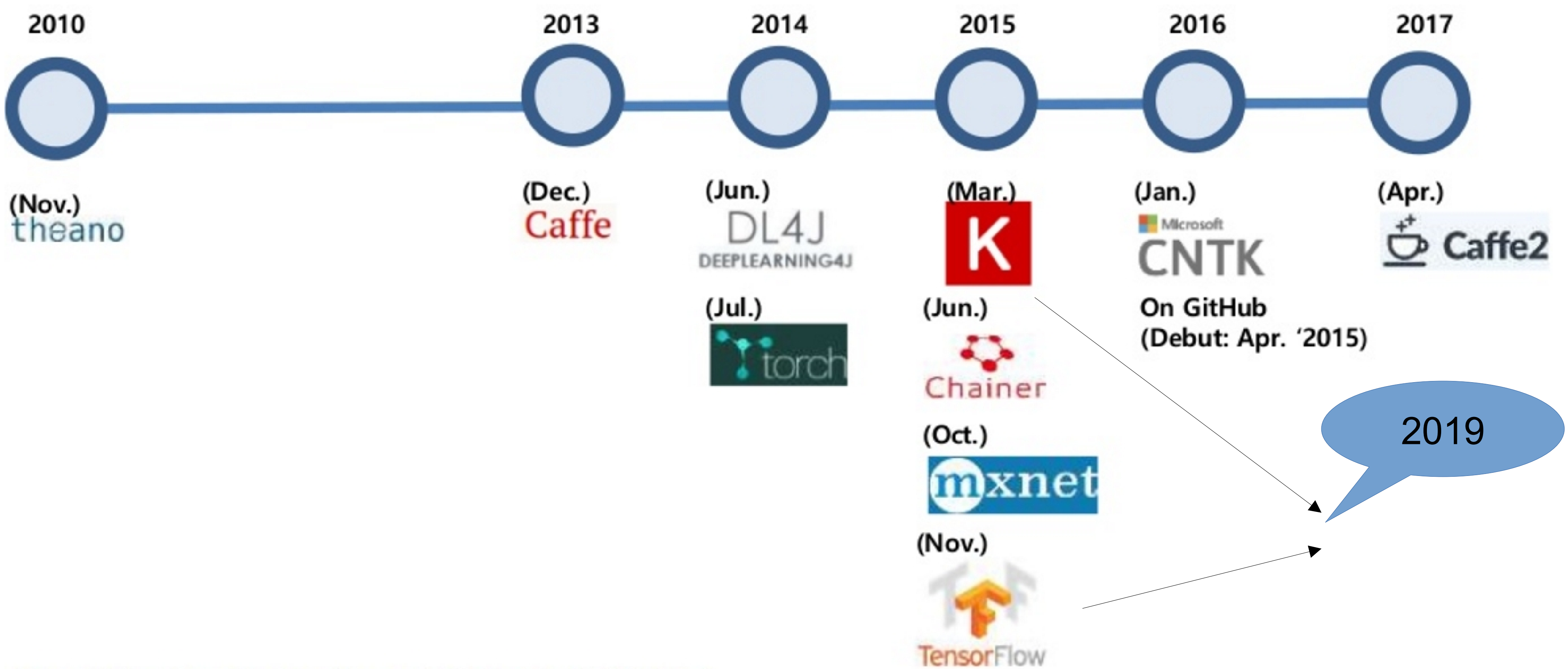
1. Инструменты для создания нейронных сетей.

2. Общие сведения о Keras

3. Синтаксис Keras

4. Практика

Инструменты для создания нейр. сетей



Общие сведения о Keras

Основы синтаксиса

GeekBrains

Deep Learning with Keras : : CHEAT SHEET

Keras
TensorFlow

Intro

Keras is a high-level neural networks API developed with a focus on enabling fast experimentation. It supports multiple back-ends, including TensorFlow, CNTK and Theano.

TensorFlow is a lower level mathematical library for building deep neural network architectures. The keras R package makes it easy to use Keras and TensorFlow in R.

Define
- Model
- Sequential model
- Multi-GPU model

→ **Compile**
- Optimiser
- Loss
- Metrics

→ **Fit**
- Batch size
- Epochs
- Validation split

→ **Evaluate**
- Evaluate
- Plot

→ **Predict**
- classes
- probability

https://keras.rstudio.com

https://www.manning.com/books/deep-learning-with-r

The "Hello, World!" of deep learning

INSTALLATION

The keras R package uses the Python keras library. You can install all the prerequisites directly from R.
https://keras.rstudio.com/reference/install_keras.html

```
library(keras)
install_keras()
```

See ?keras_install for GPU instructions

This installs the required libraries in an Anaconda environment or virtual environment 'r-tensorflow'.

Working with keras models

DEFINE A MODEL

keras_model() Keras Model

keras_model_sequential() Keras Model composed of a linear stack of layers

multi_gpu_model() Replicates a model on different GPUs

COMPILE A MODEL

compile(object, optimizer, loss, metrics = NULL)
Configure a Keras model for training

FIT A MODEL

fit(object, x = NULL, y = NULL, batch_size = NULL, epochs = 10, verbose = 1, callbacks = NULL, ...)
Train a Keras model for a fixed number of epochs (iterations)

fit_generator() Fits the model on data yielded batch-by-batch by a generator

train_on_batch() test_on_batch() Single gradient update or model evaluation over one batch of samples

EVALUATE A MODEL

evaluate(object, x = NULL, y = NULL, batch_size = NULL) Evaluate a Keras model

evaluate_generator() Evaluates the model on a data generator

PREDICT

predict() Generate predictions from a Keras model

predict_proba() and **predict_classes()**
Generates probability or class probability predictions for the input samples

predict_on_batch() Returns predictions for a single batch of samples

predict_generator() Generates predictions for the input samples from a data generator

OTHER MODEL OPERATIONS

summary() Print a summary of a Keras model

export_savedmodel() Export a saved model

get_layer() Retrieves a layer based on either its name (unique) or index

pop_layer() Remove the last layer in a model

save_model_hdf5(); load_model_hdf5() Save/Load models using HDF5 files

serialize_model(); unserialize_model()
Serialize a model to an R object

clone_model() Clone a model instance

freeze_weights(); unfreeze_weights()
Freeze and unfreeze weights

CORE LAYERS

layer_input() Input layer

layer_dense() Add a densely-connected NN layer to an output

layer_activation() Apply an activation function to an output

layer_dropout() Applies Dropout to the input

layer_reshape() Reshapes an output to a certain shape

layer_permute() Permute the dimensions of an input according to a given pattern

layer_repeat_vector() Repeats the input n times

layer_lambda(object, f) Wraps arbitrary expression as a layer

layer_activity_regularization() Layer that applies an update to the cost function based input activity

layer_masking() Masks a sequence by using a mask value to skip timesteps

layer_flatten() Flattens an input

TRAINING AN IMAGE RECOGNIZER ON MNIST DATA

```
# input layer: use MNIST images
mnist <- dataset_mnist()
x_train <- mnist$train$x;  y_train <- mnist$train$y
x_test <- mnist$test$x;  y_test <- mnist$test$y

# reshape and rescale
x_train <- array_reshape(x_train, c(nrow(x_train), 784))
x_test <- array_reshape(x_test, c(nrow(x_test), 784))
x_train <- x_train / 255;  x_test <- x_test / 255

y_train <- to_categorical(y_train, 10)
y_test <- to_categorical(y_test, 10)

# defining the model and layers
model <- keras_model_sequential()
model %>%
 layer_dense(units = 256, activation = 'relu',
             input_shape = c(784)) %>%
 layer_dropout(rate = 0.4) %>%
 layer_dense(units = 128, activation = 'relu') %>%
 layer_dense(units = 10, activation = 'softmax')

# compile (define loss and optimizer)
model %>% compile(
 loss = 'categorical_crossentropy',
 optimizer = optimizer_rmsprop(),
 metrics = c('accuracy')
)

# train (fit)
model %>% fit(
 x_train, y_train,
 epochs = 30, batch_size = 128,
 validation_split = 0.2
)
model %>% evaluate(x_test, y_test)
model %>% predict_classes(x_test)
```

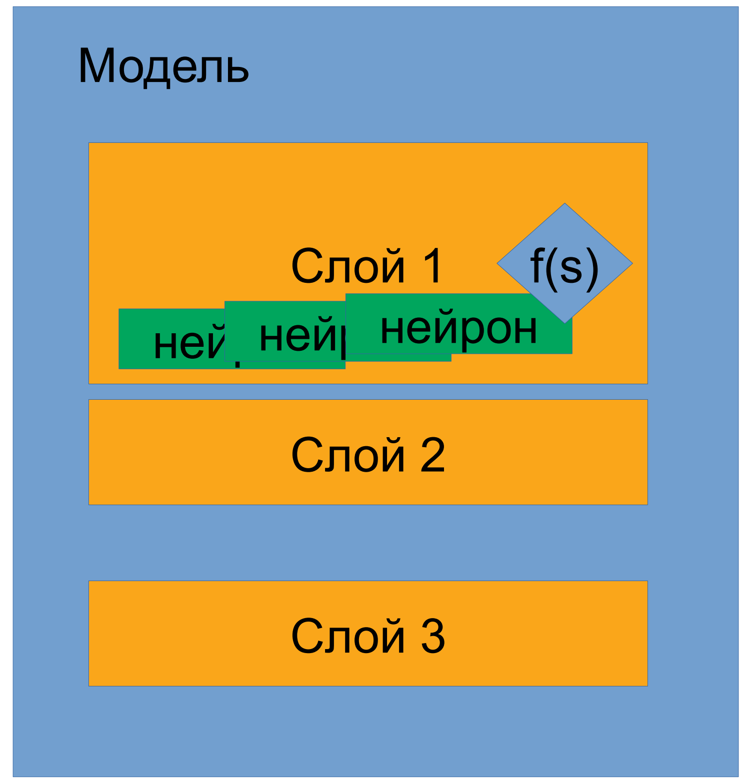
RStudio

Структура Keras

Models
- Sequential
- Functional API

Layers
- сверточные
- рекуррентные
- полносвязные
- служебные

Preprocessing
- utils
- обработка изображений
- обработка текстов

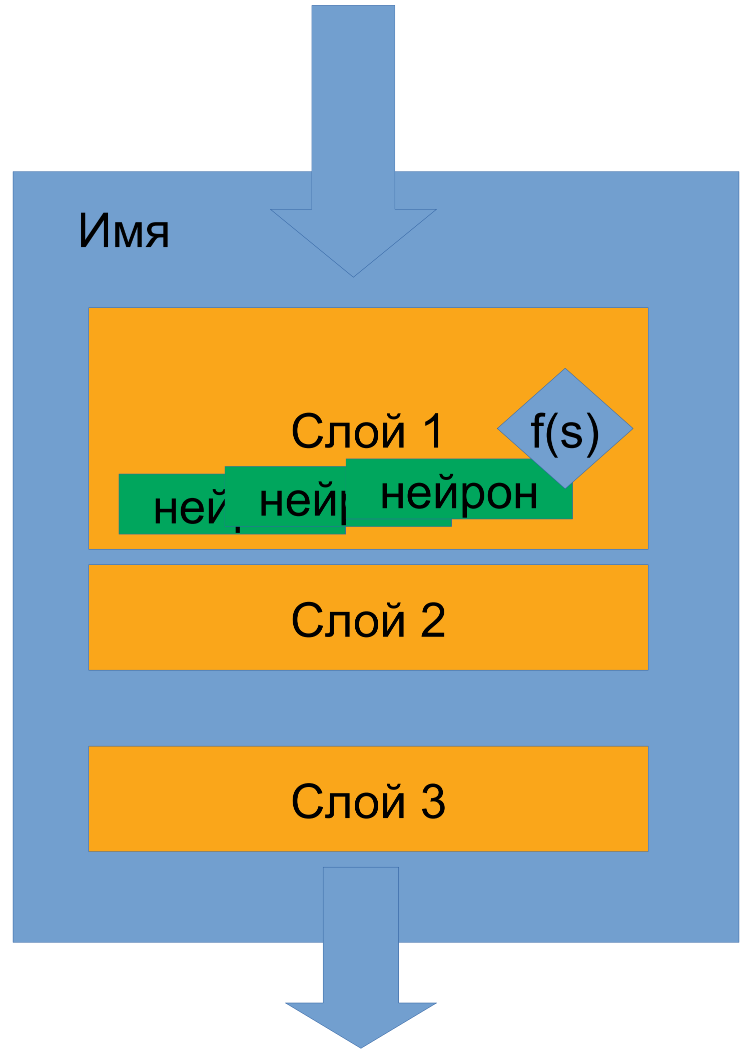


Models.Model

keras.Model()

inputs
outputs
name

Model.summary()
Model.get_layer(name=None, index=None)

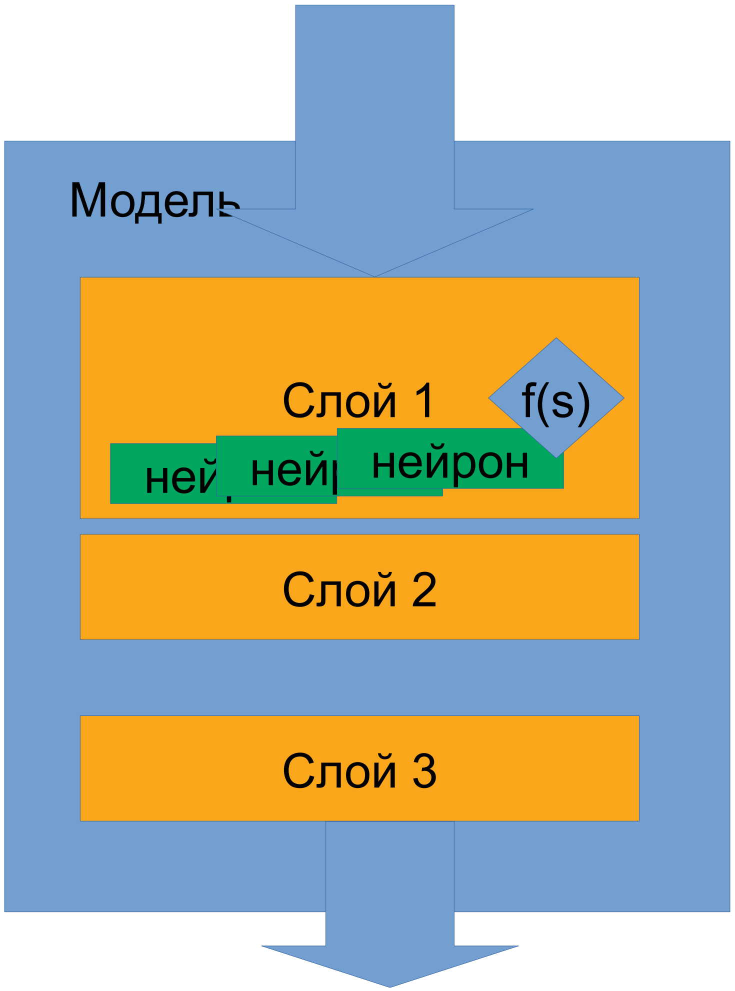


Models.Sequential

keras.Sequential()

layers,
Name

модель.add(<Слой>)

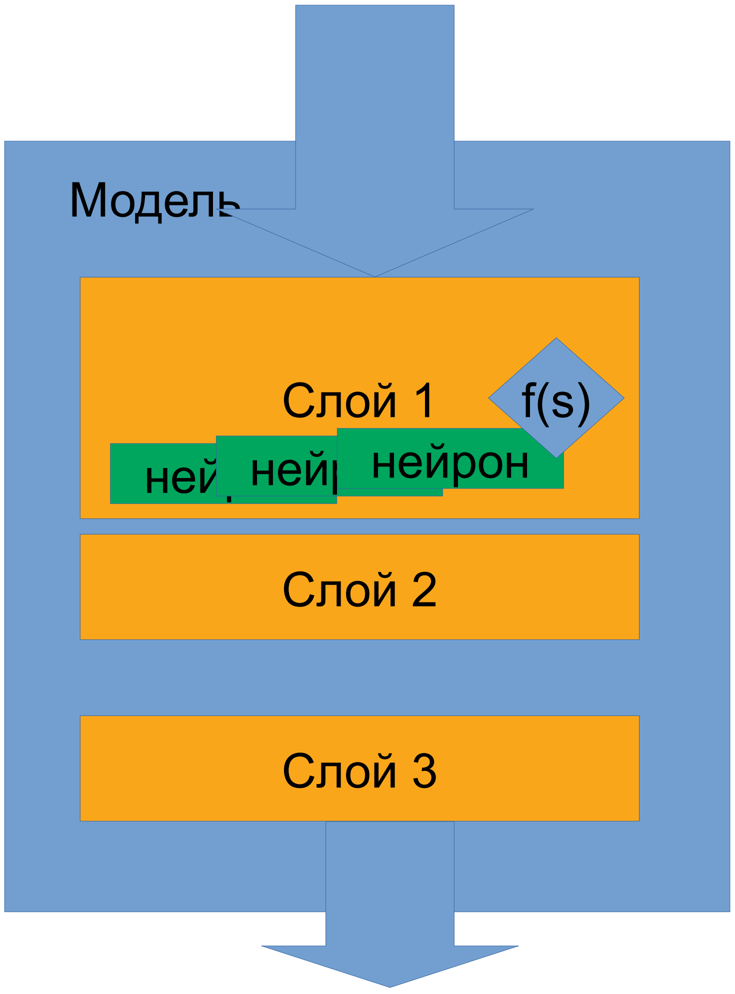


Models Model training API

```
Model.compile(
    optimizer="rmsprop",
    loss=None,
    metrics=None)

Model.fit(    x=None,   y=None,
    batch_size=None,
    epochs=1,
    validation_split=0.0)

Model.predict(  x)
```

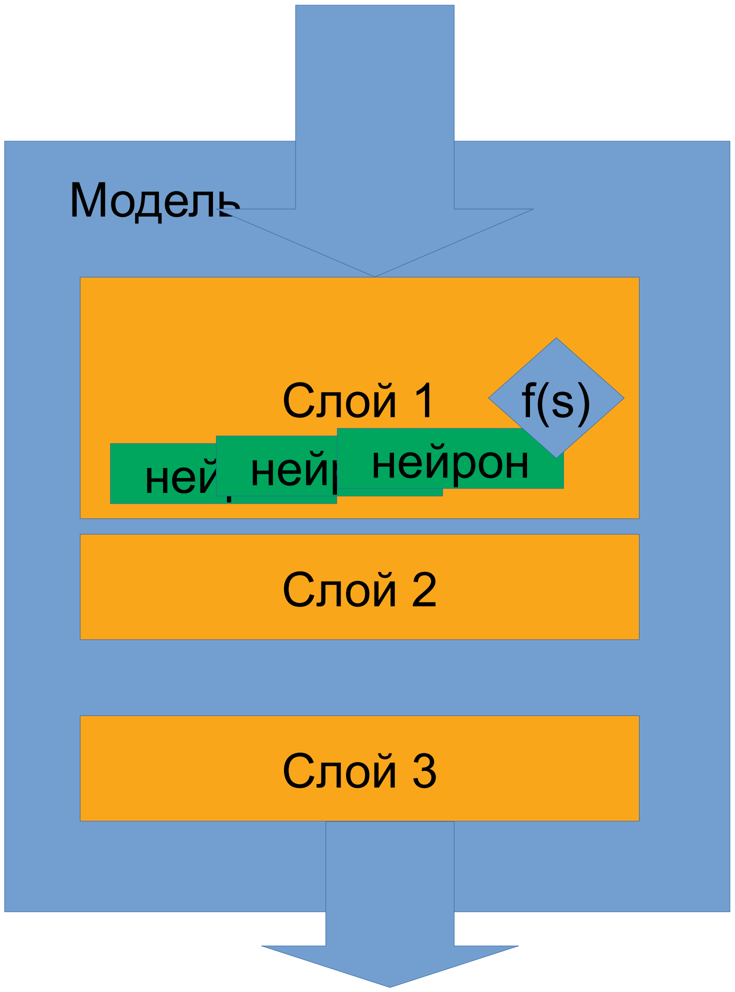


Models Model training API

```
Model.compile(
    optimizer="rmsprop",
    loss=None,
    metrics=None)
```

Optimizer:

- SGD
- RMSProp\
- Adam
- ...



Оптимизаторы

SGD

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

```
while True:
  dx = compute_gradient(x)
  x += learning_rate * dx
```

SGD+Momentum

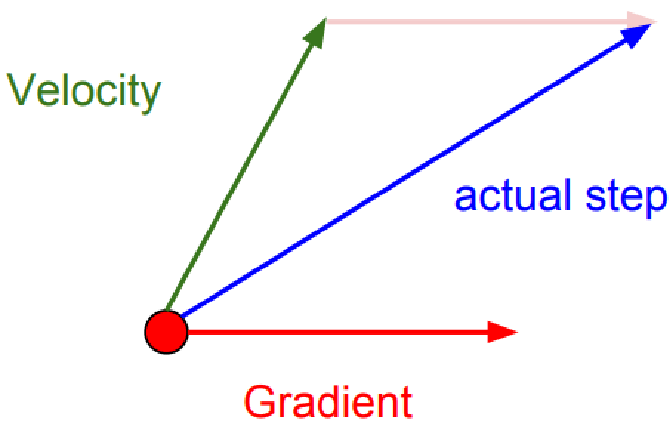
$$v_{t+1} = \rho v_t + \nabla f(x_t)$$
$$x_{t+1} = x_t - \alpha v_{t+1}$$

```
vx = 0
while True:
  dx = compute_gradient(x)
  vx = rho * vx + dx
  x += learning_rate * vx
```

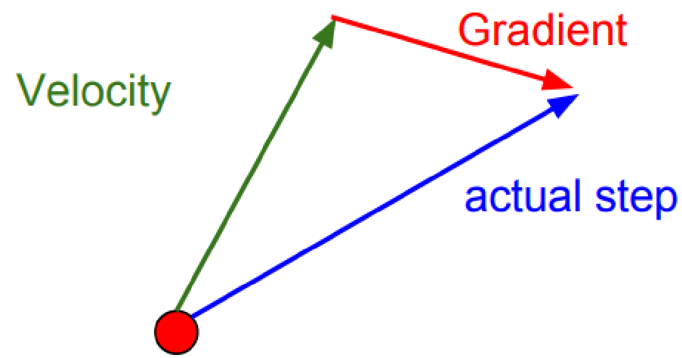
https://www.reg.ru/blog/stehnfordskij-kurs-lekciya-4-vvedenie-v-nejronnye-seti/

Оптимизаторы

Momentum update:



Nesterov Momentum



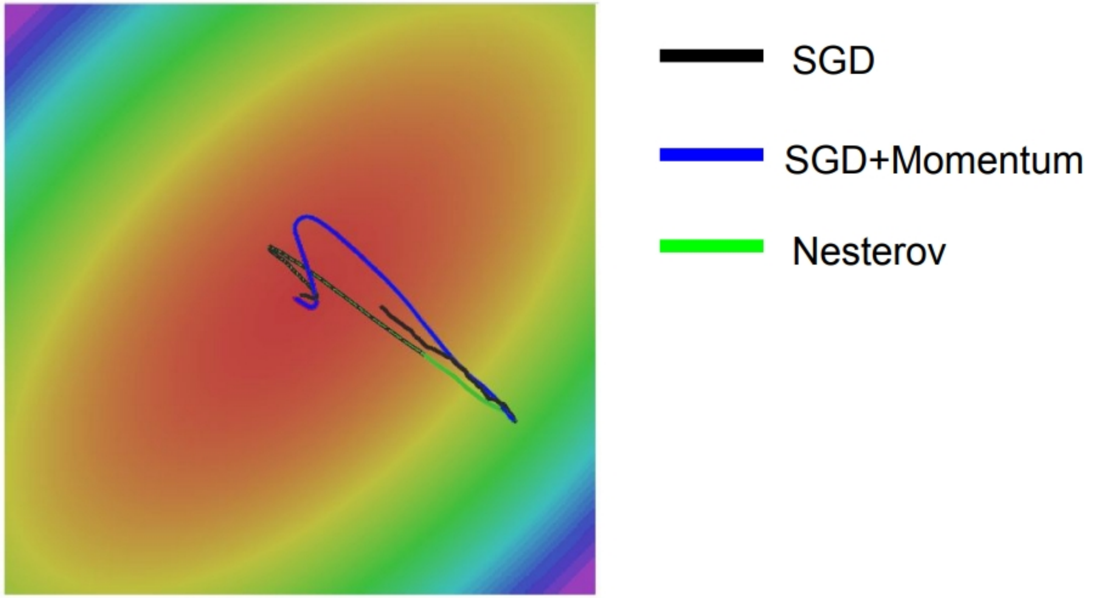
Сначала шаг, а потом адаптация из этого шага

Оптимизаторы

Nesterov Momentum

$$v_{t+1} = \rho v_t - \alpha \nabla f(x_t + \rho v_t)$$
$$x_{t+1} = x_t + v_{t+1}$$

```
dx = compute_gradient(x)
old_v = v
v = rho * v - learning_rate * dx
x += -rho * old_v + (1 + rho) * v
```



Оптимизаторы: Adam

$$a(X_j, W) = X_j W$$

1. W_0
2. $Q = \frac{1}{n}\sum_{j=1}^{n}(X_j W - Y_j)^2$
3. $X_j \in X$
4. $L_j = (X_j W - Y_j)^2,$

$$m = W^t \cdot \beta 1 + (1-\beta 1)\nabla L_j$$

$$l = \Delta W^t \cdot \beta 2 + (1-\beta 2) * (\nabla L_j)^2$$

5. $W^{t+1} = W^t - \frac{\eta}{\sqrt{\frac{l}{(1-\beta 2^t)}}} \frac{m}{(1-\beta 1^t)}$
6. $Q = \lambda e_j + (1-\lambda)Q$
7. $проверка \quad на \quad Q, \quad \Delta W : \langle true : end \quad false : к \quad п.3 \rangle$

Оптимизаторы

Что еще смотрим ?

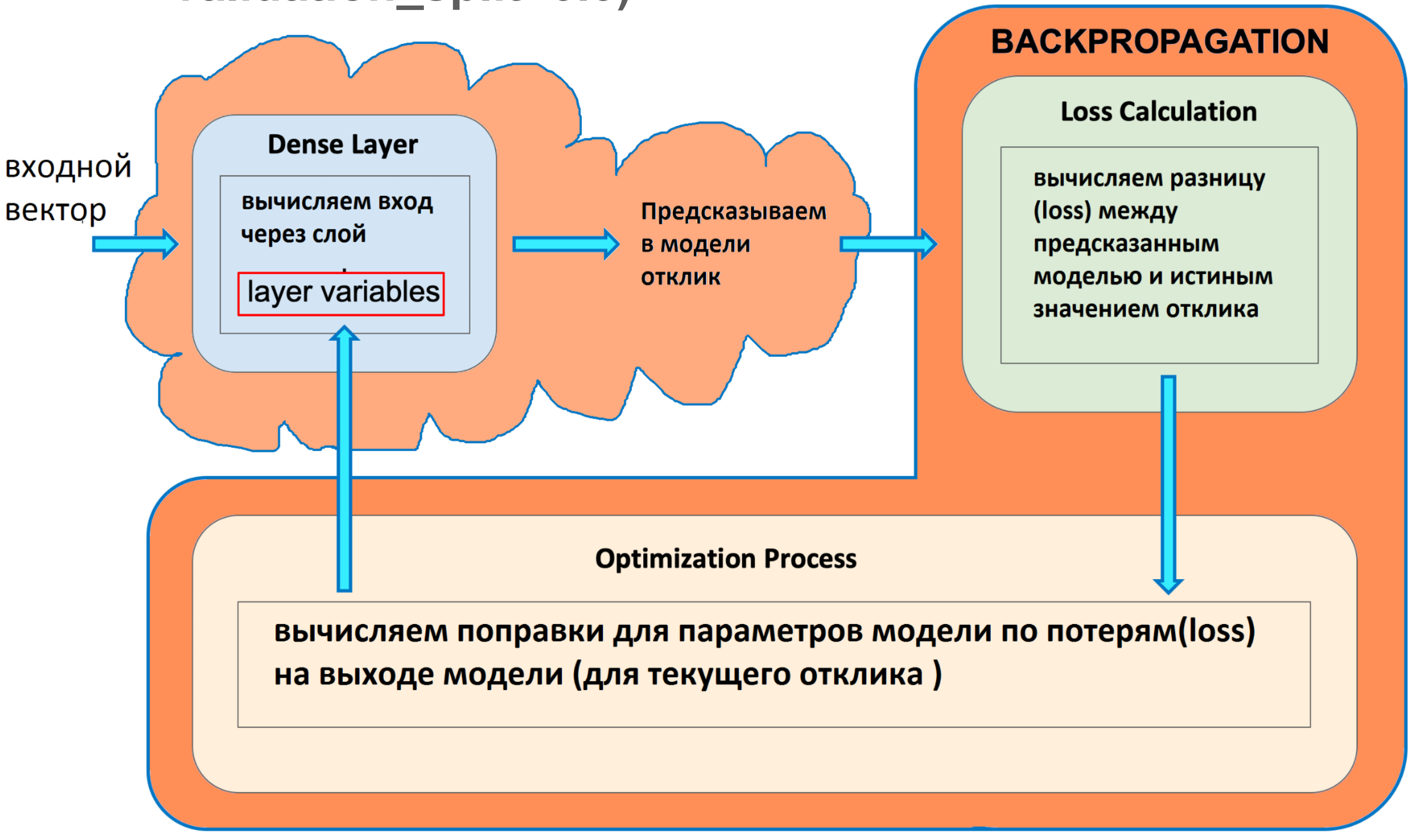
Методы оптимизации : https://www.reg.ru/blog/stehnfordskij-kurs-lekciya-4-vvedenie-v-nejronnye-seti/
Методы высоких порядков: BFGS
https://habr.com/ru/post/333356/

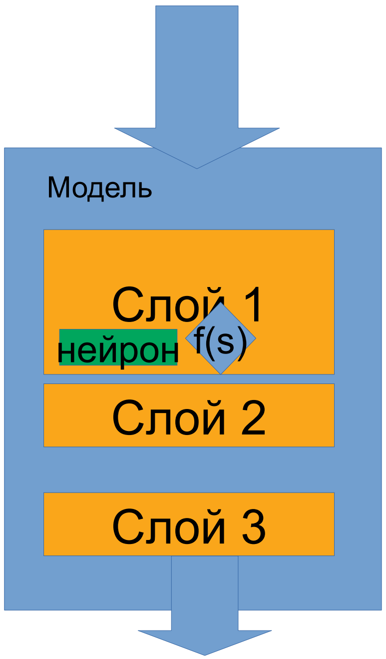
Резюме:

- Универсального алгоритма нет
- Параметры нужно подбирать
- Делаем несколько экспериментов в каждой конфигурации

Models Model training API

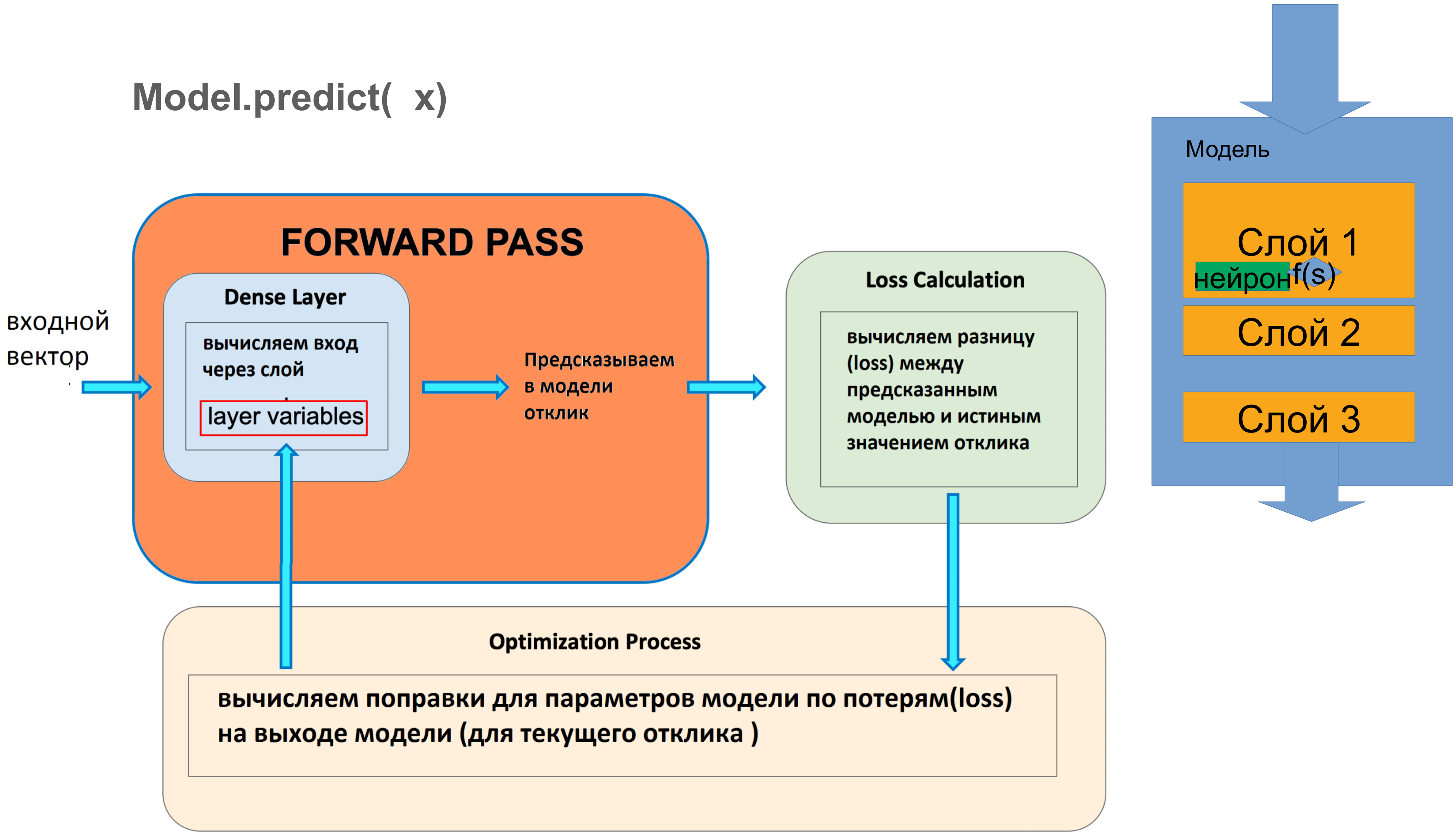
```
Model.fit(    x=None,   y=None,
    batch_size=None,
    epochs=1,
    validation_split=0.0)
```





Models Model training API

Model.predict(x)



Вопросы

Практическое задание

GeekBrains

Используем набор примеров fashion-MNIST

1. Опишите - какой результата получен в нейросети в зависимости от:
   - числа нейронов в слое(для 2-хслойной сети),
   - числа слоев (2, 3, 5, 10) при близких размерах сети (близкое число тренируемых парметров).
   - фиксируйте для тренировочного и тестового набора метрики accuracy.
2. Проверьте работу разных оптимизаторов (SGD, Adam, RMSProp) для одной из моделей п.1.Фиксируйте для тренировочного и тестового набора метрики accuracy.

3. Сделайте вывод - что помогло вам улучшить качество классификации в нейросети на тестовом наборе?

4. Для одного варианта сетей сформируйте матрицу ошибок по классам. Оцените качество модели по каждому классу отдельно (полнота , точность)

* Поработайте с документацией Keras. Найдите полезные команды не разобранные на уроке.