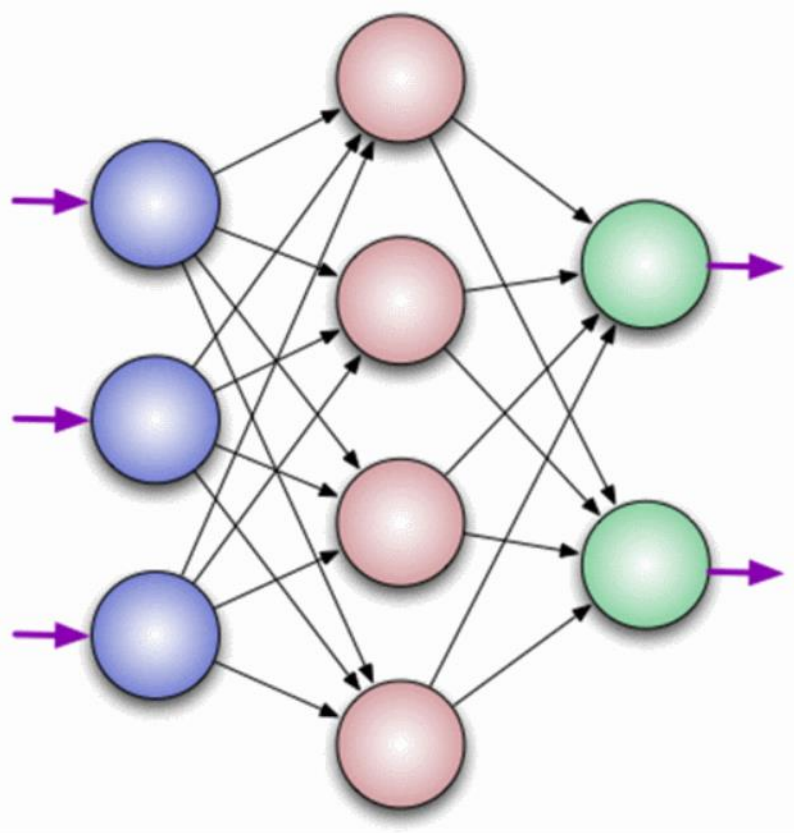
Современные нейросетевые технологии

МАЛЕВА ТВ. ЛЕКЦИЯ 1. МНОГОСЛОЙНЫЙ ПЕРСЕПТРОН

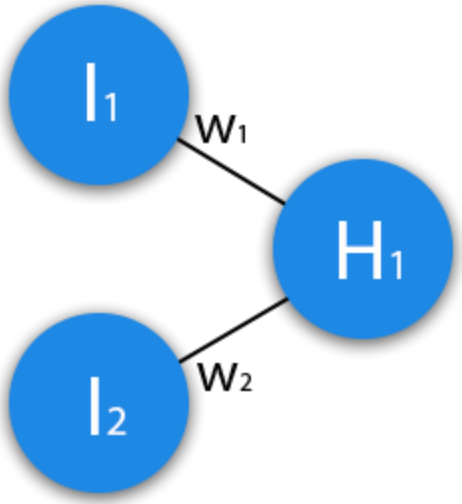
Нейронные сети



Входной, скрытый, выходной нейрон

Одз = [0,1], [-1,1], [0,1/n] - нормализация

Нейронные сети



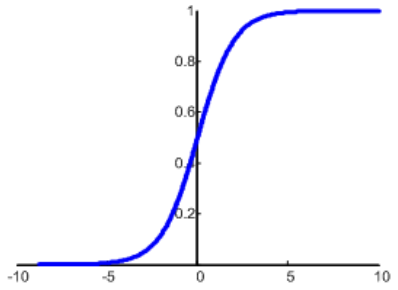
$$1) H_{1_{input}} = (I_1 * w_1) + (I_2 * w_2)$$

$$2) H_{1_{output}} = f_{activation}(H_{1_{input}})$$

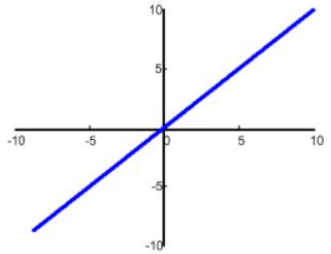
Синапс (веса)

Функция активации

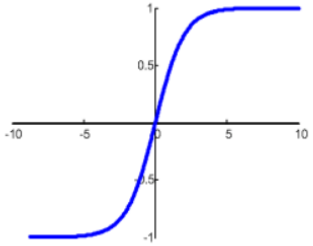
$$f(x) = \frac{1}{1+e^{-x}}$$



$$f(x) = x$$

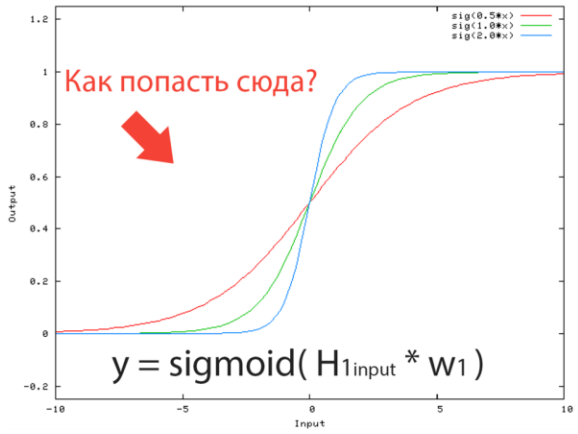


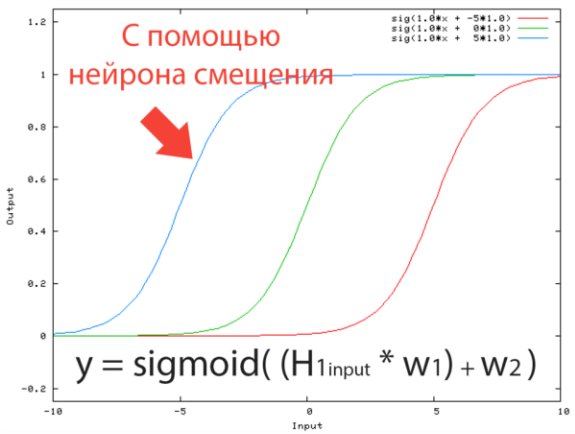
$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

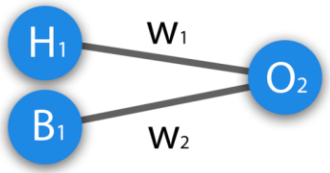


Функции активации

Нейрон смещения







1. вход и выход bias всегда равняются 1 и они никогда не имеют входных синапсов.
2. Нейроны смещения могут, либо присутствовать в нейронной сети по одному на слое, либо полностью отсутствовать
3. Соединения у нейронов смещения такие же, как у обычных нейронов — со всеми нейронами следующего уровня, за исключением того, что синапсов между двумя bias нейронами быть не может. Следовательно, их можно размещать на входном слое и всех скрытых слоях, но никак не на выходном слое
4. Нейрон смещения нужен для того, чтобы иметь возможность получать выходной результат, путем сдвига графика функции активации вправо или влево.

Нейронные сети. Потери

- Потери (loss, ошибка) — это процентная величина, отражающая расхождение между ожидаемым и полученным ответами.
- Потери формируются каждую эпоху и должны идти на спад.
- Mean Squared Error (MSE), Root MSE, Arctan. 3
- У Arctan, ошибка, почти всегда, будет больше, так как он работает по принципу: чем больше разница, тем больше ошибка.
- У Root MSE будет наименьшая ошибка, поэтому, чаще всего, используют MSE, которая сохраняет баланс в вычислении ошибки.

MSE

$$\frac{(i_1-a_1)^2+(i_2-a_2)^2+\ldots+(i_n-a_n)^2}{n}$$

Root MSE

$$\sqrt{\frac{(i_1-a_1)^2+(i_2-a_2)^2+\ldots+(i_n-a_n)^2}{n}}$$

Arctan

$$\frac{\arctan^2(i_1-a_1)+\ldots+\arctan^2(i_n-a_n)}{n}$$

Обучение с учителем. Метод градиентного спуска (SGD)

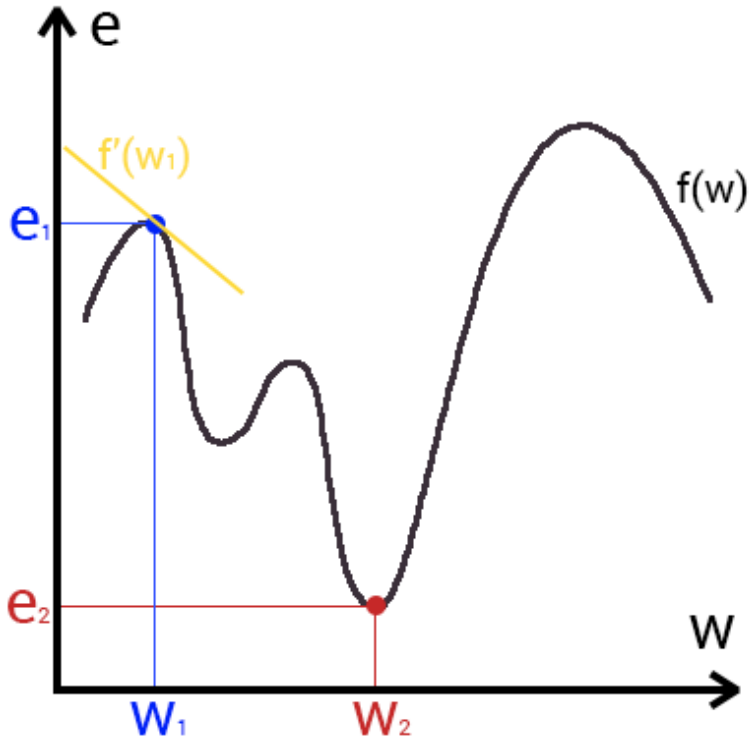
способ нахождения локального минимума или максимума функции с помощью движения вдоль градиента

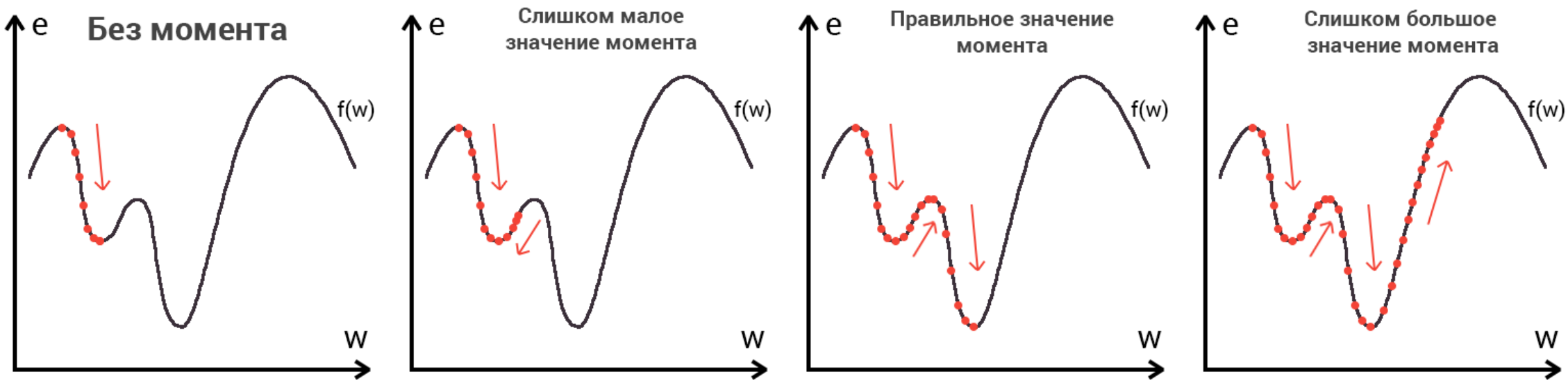
функция f(w) является зависимостью ошибки от выбранного веса.

глобальный минимум — точка (w2,e2)

у каждого веса в нейросети будет свой график и градиент и у каждого надо найти глобальный минимум.

Градиент — это вектор который определяет крутизну склона и указывает его направление относительно какой либо из точек на поверхности или графике. Чтобы найти градиент нужно взять производную от графика по данной точке (как это и показано на графике). Двигаясь по направлению этого градиента минимум будет плавно скатываться в низину.





ОБУЧЕНИЕ С УЧИТЕЛЕМ. МЕТОД ГРАДИЕНТНОГО СПУСКА

Обучение с учителем. Метод градиентного спуска

O – output

H – hidden

точка A это точка в начале синапса, а точка B на конце синапса

E — скорость обучения, α — момент

Стохастический метод — нашел Δw, сразу обнови соответствующий вес

Пакетный метод - мы суммируем Δw всех весов на текущей итерации и только потом обновляем все веса используя эту сумму. Один из самых важных плюсов такого подхода — это значительная экономия времени на вычисление, точность же в таком случае может сильно пострадать.

Мини-пакетный метод является золотой серединой и пытается совместить в себе плюсы обоих методов. Здесь принцип таков: в свободном порядке распределяем веса по группам и меняем их веса на сумму Δw всех весов в той или иной группе.

$$1)\ \delta_O = (OUT_{ideal} - OUT_{actual}) * f'(IN)$$

$$2)\ \delta_H = f'(IN) * \sum(w_i * \delta_i)$$

$$f'(IN) = \begin{cases} f_{sigmoid} = (1 - OUT) * OUT \\ f_{tangh} = 1 - OUT^2 \end{cases}$$

$$GRAD_B^A = \delta_B * OUT_A$$

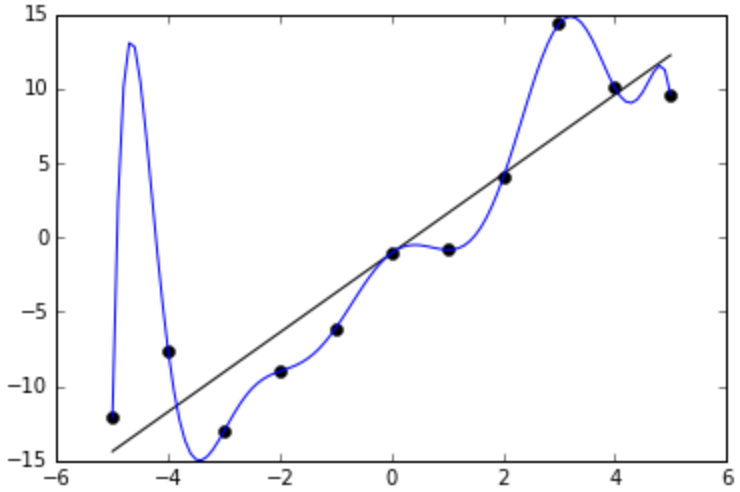
$$\Delta w_i = E * GRADw + \alpha * \Delta w_{i-1}$$

Гиперпараметры

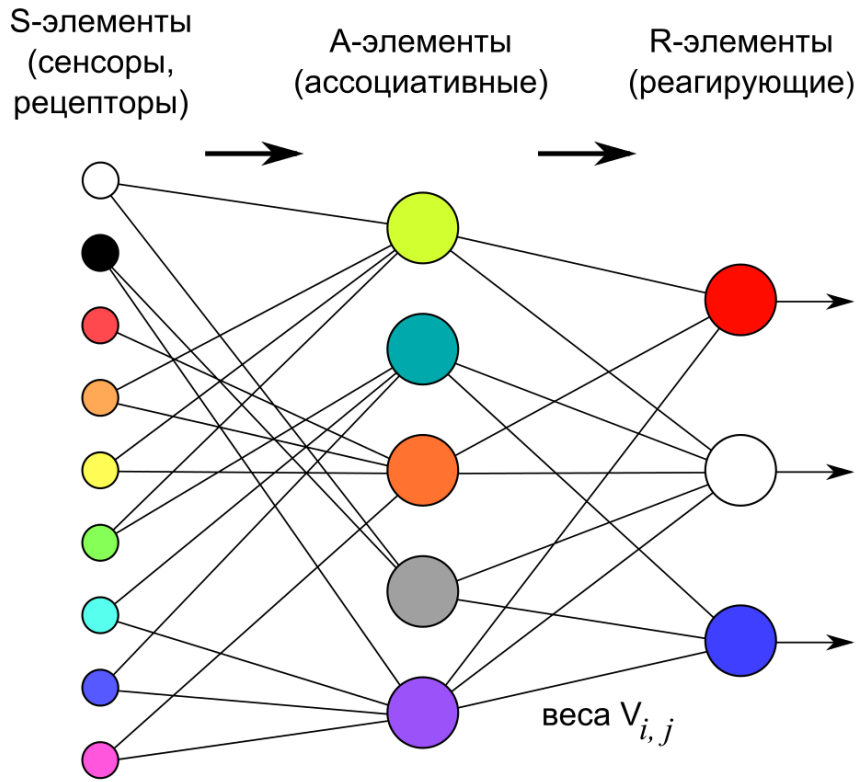
Гиперпараметры — это значения, которые нужно подбирать вручную и зачастую методом проб и ошибок.

- Момент и скорость обучения
- Количество скрытых слоев
- Количество нейронов в каждом слое
- Наличие или отсутствие нейронов смещения

Сходимость и переобучение



Персептрон (KERAS)




```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential()
model.add(layers.Dense(64, kernel_initializer='uniform', input_shape=(10,)))
model.add(layers.Activation('sigmoid'))

opt = keras.optimizers.Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer=opt)
```

Плотный слой/ Dense

```
layers.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,
    **kwargs
)
```

Плотный слой/ Dense

1. units: положительное целое число, размерность выходного пространства.
2. activation: функция активации для использования. Если вы ничего не укажете, активация не применяется (например, «линейная» активация :) a(x) = x.
3. **use_bias** : логическое значение, определяет, использует ли вектор смещения.
4. **kernel_initializer** : Инициализатор для матрицы весов.
5. **bias_initializer** : Инициализатор вектора смещения.
6. **kernel_regularizer** : функция регуляризатора, примененная к матрице весов.
7. **bias_regularizer** : функция регуляризатора, примененная к вектору смещения.
8. **activity_regularizer** : функция регуляризатора, применяемая к выходу слоя (его «активация»).
9. **kernel_constraint** : функция ограничения, применяемая к матрице весов.
10. **bias_constraint** : функция ограничения, применяемая к вектору смещения.
11. **Форма ввода**

Тензор ND с формой: (batch_size, ..., input_dim). Наиболее распространенной ситуацией является 2D-ввод с формой (batch_size, input_dim).

1. **Форма вывода**

Тензор ND с формой: (batch_size, ..., units). Например, для двумерного ввода с формой (batch_size, input_dim) вывод будет иметь форму (batch_size, units).

Dropout (прореживание)

layers.Dropout(rate, noise_shape=None, seed=None, **kwargs)

rate : Float от 0 до 1. Доля вводимых единиц, которую нужно отбросить.

noise_shape : 1D целочисленный тензор, представляющий форму двоичной маски исключения, которая будет **умножена** на вход. Например, если ваши входные данные имеют форму (batch_size, timesteps, features)и вы хотите, чтобы маска исключения была одинаковой для всех временных шагов, вы можете использовать noise_shape=(batch_size, 1, features).

seed : целое число Python для использования в качестве случайного числа.

Инициализация весов

1. Uniform (-0.05,0.05)
2. Random_normal (mean = 0, std = 0.05)
3. Zero

Функции активации

1. relu = max(x, 0)
2. sigmoid = 1 / (1 + exp(-x))
3. softmax = exp(x) / tf.reduce_sum(exp(x)) . преобразует вещественный вектор в вектор категориальных вероятностей. Элементы выходного вектора находятся в диапазоне (0, 1) и в сумме равны 1
4. softplus = log(exp(x) + 1)
5. softsign = x / (abs(x) + 1)
6. tanh = sinh(x)/cosh(x) = ((exp(x) - exp(-x))/(exp(x) + exp(-x))).
7. selu = if x > 0: return scale * x, if x < 0: return scale * alpha * (exp(x) - 1), где alpha и scale- предварительно определенные константы (alpha=1.67326324и scale=1.05070098).
8. Elu. Экспоненциальная линейная единица (ELU) alpha > 0: x если x > 0 и alpha * (exp(x) - 1) если x < 0 Гиперпараметр ELU alpha управляет значением, до которого насыщается ELU для отрицательных входных цепей. ELU уменьшают эффект исчезающего градиента.
9. Exp
10. tf.keras.layers.advanced_activations. К ним относятся PReLUи LeakyReLU

Функция потерь

- Среднеквадратическая ошибка mse (MeanSquaredError)

https://keras.io/api/losses/regression_losses/

- Бинарная перекрестная энтропия BinaryCrossentropy

https://keras.io/api/losses/probabilistic_losses/

- Категориальная перекрёстная энтропия CategoricalCrossentropy

https://keras.io/api/losses/probabilistic_losses/

Метрики качества

1. Верность - число правильных предсказаний / общее число меток
2. Точность - доля правильных ответов модели (Accuracy)
3. Полнота - доля обнаруженных истинных событий
4. https://keras.io/api/metrics/

Регуляризаторы

1. L1 - по сумме модулей весов (лассо)
2. L2 - по сумме квадратов весов (гребневая)
3. L1&L2
4. Min(loss +k*reg)
5. from tensorflow.keras import layers
6. from tensorflow.keras import regularizers
7.
7. layer = layers.Dense(
8. units=64,
9. kernel_regularizer=regularizers.l1_l2(l1=1e-5, l2=1e-4),
10. bias_regularizer=regularizers.l2(1e-4),
11. activity_regularizer=regularizers.l2(1e-5)
12.)

Классификация рукописных цифр

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential()
model.add(layers.Dense(64, kernel_initializer='uniform', input_shape=(10,)))
model.add(layers.Activation('sigmoid'))

opt = keras.optimizers.Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer=opt)
```

```
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.optimizers import SGD
from keras.utils import np_utils
np.random.seed(1671)

NB_EPOCH = 200
BATCH_SIZE = 128
VERBOSE = 1
NB_CLASSES = 10
OPTIMIZER = SGD()
N_HIDDEN = 128
VALIDATION_SPLIT = 0.2

(X_train, y_train), (X_test, y_test) = mnist.load_data()
RESHAPED = 784
X_train = X_train.reshape(60000, RESHAPED)
X_test = X_test.reshape(10000, RESHAPED)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

X_train /= 255
X_test /= 255

print(X_train.shape[0], 'train sapmples')
print(X_test.shape[0], 'test sapmples')

Y_train = np_utils.to_categorical(y_train, NB_CLASSES)
Y_test = np_utils.to_categorical(y_test, NB_CLASSES)
```

```
60000 train sapmples
10000 test sapmples
```

```
model = Sequential()
model.add(Dense(NB_CLASSES, input_shape=(RESHAPED,)))
model.add(Activation('softmax'))
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                7850
_________________________________________________________________
activation_1 (Activation)    (None, 10)                0
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss='categorical_crossentropy',optimizer = OPTIMIZER,metrics = ['accuracy'])
```

```
h = model.fit(X_train,Y_train,batch_size = 25, epochs = 10,verbose = VERBOSE,validation_split = 0.3)
```

```
Train on 42000 samples, validate on 18000 samples
Epoch 1/10
42000/42000 [==============================] - 2s 46us/step - loss: 0.8102 - accuracy: 0.8087 - val_loss:
val_accuracy: 0.8733
Epoch 2/10
42000/42000 [==============================] - 1s 28us/step - loss: 0.4733 - accuracy: 0.8777 - val_loss:
```


```
model.compile(loss='categorical_crossentropy',optimizer = OPTIMIZER,metrics = ['accuracy'])
```

```
h = model.fit(X_train,Y_train,batch_size = 25, epochs = 10,verbose = VERBOSE,validation_split = 0.3)
```


```
score = model.evaluate(X_test,Y_test,verbose = VERBOSE)
print(score[0],score[1])
```

```
10000/10000 [==============================] - 0s 17us/step
0.3107519558250904 0.9136999845504761
```

Задание

Задание 1 (для MNIST)

1. Добавить скрытые слои
2. Добавить Dropout
3. Сравнить точность вычислений обеих сеток

Задание 2

Взять произвольную функцию четырех (8,12) аргументов и аппроксимировать ее нейронной сетью. Какова оптимальная глубина сети для каждого количества аргументов, если функция линейная, квадратичная, трансцендентная, как связано количество аргументов и глубина сети?