**Функции активации**

* Rectified Linear Unit (Relu)

f(x) = max(0,x)

* SoftMax - нормализованная экспоненциальная функция
* Используется для представления вероятности
* Сумма все выходных значений нейронов равна 1

Пример программы

model - Sequential()

# Входной полносвязный слой, 800 нейронов, 784 входа в каждый кайрон

model. add (Dense (800, input\_dim 784, activation="relu"))

# выходной полносвязный слой 10 нейронов (по количеству рукописных цифр)

model.add(Dense (10, activation "softmax"))

model.compile(loss="categorical crommentropy", optimizer ="SQD", metrica=("accuracy”])

print(model. summary ())

**Метрики качества**

* Метрики под каждую задачу могут выбираться индивидуально.
* Можно создавать свои метрики.
* Есть ряд общепринятых и часто употребляемых метрик качества.

Для задач регрессии чаще всего применяются:

* среднее квадратичное отклонение (mean square error - MSE) – среднее квадратов всех отклонений предсказанных значений от истинных
* среднее абсолютное отклонение (mean absolute error - MAE) – среднее абсолютных значение всех отклонений
* Существует еще масса модификаций этих метрик (MSE, MAPE, MSLE), которые можно увидеть в документации

TensorFlow. https://www.tensorflow.org/api docs/python/tf/karas/metrics

Для задач классификации (классификация на два класса: отрицательный и положительный (класс 0 и класс 1):

* accuracy - полная точность

где TP - это количество верно классифицированных меток положительного класса

TM - это количество верно классифицированных меток отрицательного класса

FP - это количество неверно классифицированных меток положительного класса

FN - это количество неверно классифицированных меток отрицательного класса

* precision - точность для положительного класса
* recall
* Cross Entropy
* p - настоящая вероятность класса (0 либо 1)
* ai- вероятность класса i-го объекта, предсказанная нейронной сетью

**Функции потерь и оптимизаторы обучения**

* Функции потерь могут послужить
* MAE
* MSE;
* Cross Entropy и другие метрики.
* Полный список функций потерь можно увидеть в документации TensorFlow

https://www.tensorflow.org/api docs/python/tf/karas/losses

* За сам процесс распространения посчитанной ошибки обратно на все веса модели нейронной сети отвечают так называемые оптимизаторы:
* SGD - стохастический градиентный спуск;
* RMSprop - модификация градиентного спуска;
* Adam - модификация градиентного спуска.

Полный список данных функций можно увидеть в документации

TensorFlow https://www.tensorflow.org/api docs/python/tf/karas/optimizers

Пример программы

history = model.fit(x train, y train,

batch size=200,

epochs=100,

verbose=1)

predictions=model.predict(x\_train)

print(predictions(0])

print(np.argmax(predictions[0])) 🡪9

**Обобщение**

* Набор данных Fashion MNIST
* Изображение моделей одежды
* Нейронная сеть на KARAS
* Модель: Sequential
* Уровни сети: Dense
* Функции активации: Relu и SoftMax
* Этапы обучения нейронной сети
* Загрузка и подготовка данных
* Определение структуры нейронной сети
* Компиляция (оптимизатор, функция ошибки,
* метрика качества)
* Обучение нейронной сети
* Распознавание данных
* **Переобучение**
* При обучении нейросетей может возникнуть проблема **переобучения**.
* В этом случае сеть может научиться распознавать особенности выборки, а не данных.
* Это ведет к снижению **обобщающей способности нейросети.**
* Чтобы избежать переобучения, необходимо оценивать качество работы нейросети на тех данных, которые она **не видела в процессе обучения.**

Для обучения используются три набора данных:

**Обучающая выборка** - набор данных, который используется для обучения сети.

**Проверочная выборка** - набор данных, который используется в процессе обучения для подбора гиперпараметров сети.

**Тестовая выборка** - набор данных, который используется для оценки качества работы сети после завершения обучения.

Переобучение

#обучаем сеть

history = model.fit(x train, train,

batch size=200,

epochs=100,

validation split=0.2,

verbose=1)

#Оцениваем качество обучения сети на тестовых данных

scores = model.evaluate(x test, y test, verbose=1)

print("Доля верных ответов на тестовых данных, в процентах: ",

round(scores[1] \* 100, 4))

🡪Доля верных ответов на тестовых данных, в процентах: 86.39

* Параметры нейронной сети
* Веса входов в нейроны
* Гиперпараметры сети
* Количество слоев и нейронов в каждом слое
* Количество эпох обучения
* Размер мини-выборки для обучения
* Размер параметра скорости обучения
* Архитектура нейронной сети

Использование проверочной выборки

* Часть данных из обучающей выборки выделяется в **проверочную выборку**
* На каждой эпохи обучения рассчитывается ошибка на **обучающей и проверочной выборке**
* Увеличение ошибки на проверочной выборке говорит о **переобучении**

**Проблема переобучения**

* Нейронная сеть распознает особенности обучающей выборки, а не данных
* Необходима проверка работы сети на данных, не использованных для обучения

**Проверочная выборка**

* Используется в процессе обучения для подбора гиперпараметров
* Рост ошибки на проверочной выборке говорит о переобучении

**Тестовая выборка**

* Проверка качества работы сети после завершения обучения

**Улучшение качества обучения**

* Чтобы улучшить качество обучения сети нужно изменить следующие гиперпараметры:
* **количество эпох обучения,**
* **размер мини-выборки,**
* **количество нейронов во входном слое,**
* **количество скрытых слоев.**
* Для этого нужно провести серию экспериментов, в каждом из которых следует менять один из гиперпараметров, и анализировать, как изменилось качество работы сети.

**Количество эпох обучения**

Оценим влияние количества эпох обучения на качество обучения сети

Количество эпох задается в аргументе epochs метода model.fit:

History = model.fit(x\_train, y train,

batchsize=200,

epochs=100,

validation split=0.2,

verbose=1)

Следует выбрать количество эпох, при котором самая высокая доля верных ответов нейросети на тестовых данных

* **Количество нейронов входного слоя.**
* Следует изменить количество нейронов во входном слое и оценить, как оно влияет на качество обучения сети.
* Количество нейронов задается при создании входного слоя:

model.add (Dense (800, input\_dim=784, activation="relu"))

* Нужно выбрать значение, при котором самая высокая доля верных ответов нейросети на тестовых данных.

**Добавляем скрытый слой**

Следует добавить в нейронную сеть скрытый слой, чтобы она стала

глубокой:

model,add (Deny (800, input dim=784, activation "relu”))

model,add (Deny (600, activation "relu”))

model,add (Deny (10, activation "sofrmax”))

Следует выбрать наиболее подходящее количество нейронов скрытого слоя.