



```
# modifier_ob.type != "MESH" and modifier_ob.type != "LATTICE"
mirror_ob = modifier_ob # set to mirror_ob, hope the other
#
# mirror_ob.select = False
# modifier_ob = bpy.context.selected_objects[0]
else:
    #mirror_ob
    mirror_ob = bpy.context.active_object
    mirror_ob.select = False # pop modifier_ob from sel stack
    print("popped")

    #modifier_ob
    modifier_ob = bpy.context.selected_objects[0]
    print("Modifier object:" + str(modifier_ob.name))

    #modifier_ob.select=1

    print("mirror_ob",mirror_ob)
    print("modifier_ob",modifier_ob)

# put mirror modifier on modifier_ob
mirror_mod = modifier_ob.modifiers.new("mirror_mirror","MIRROR")

# set mirror object to mirror_ob
mirror_mod.mirror_object = mirror_ob

if _operation == "MIRROR_X":
    mirror_mod.use_x = True
    mirror_mod.use_y = False
    mirror_mod.use_z = False
    #add _operation == "MIRROR_Y":
    #mirror_mod.use_x = False
    #mirror_mod.use_y = True
    #mirror_mod.use_z = False
    #add _operation == "MIRROR_Z":
    #mirror_mod.use_x = False
    #mirror_mod.use_y = False
    #mirror_mod.use_z = True
```

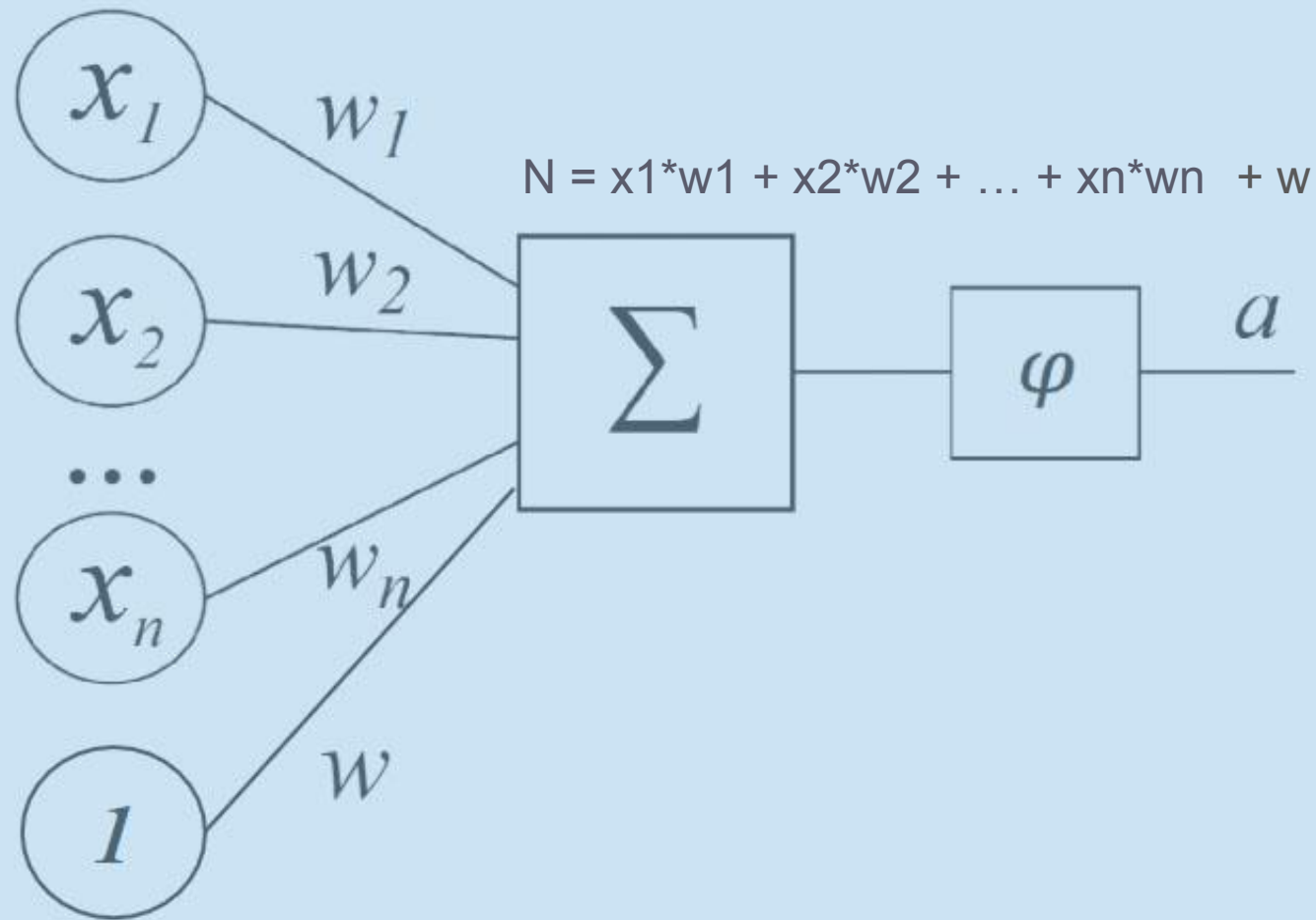
ОЦЕНКА КАЧЕСТВА ОБУЧЕНИЯ НЕЙРОСЕТИ

Bias

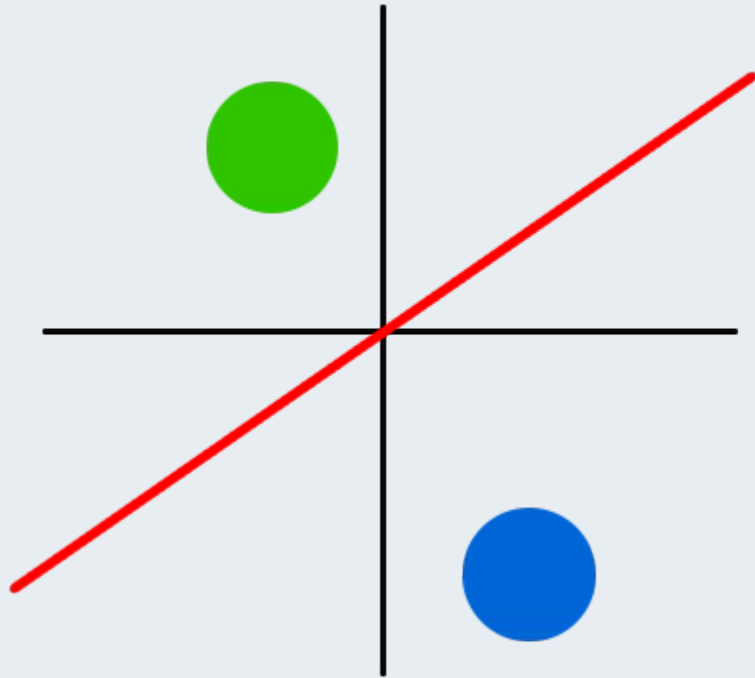
В НЕЙРОНАХ

МОДЕЛЬ

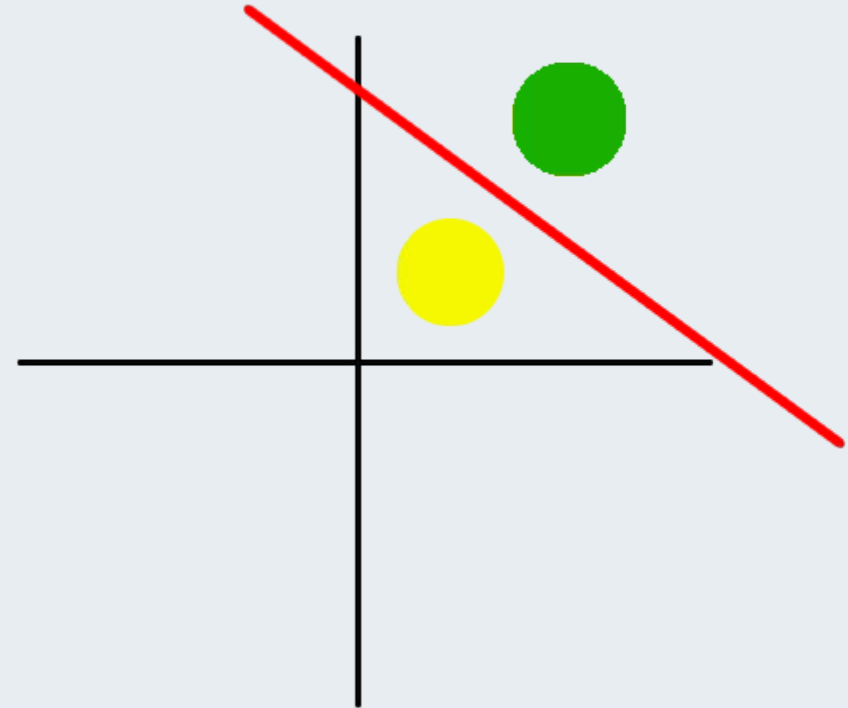
ИСКУССТВЕННОГО НЕЙРОНА



$$y = a * x$$



$$y = a * x + b$$





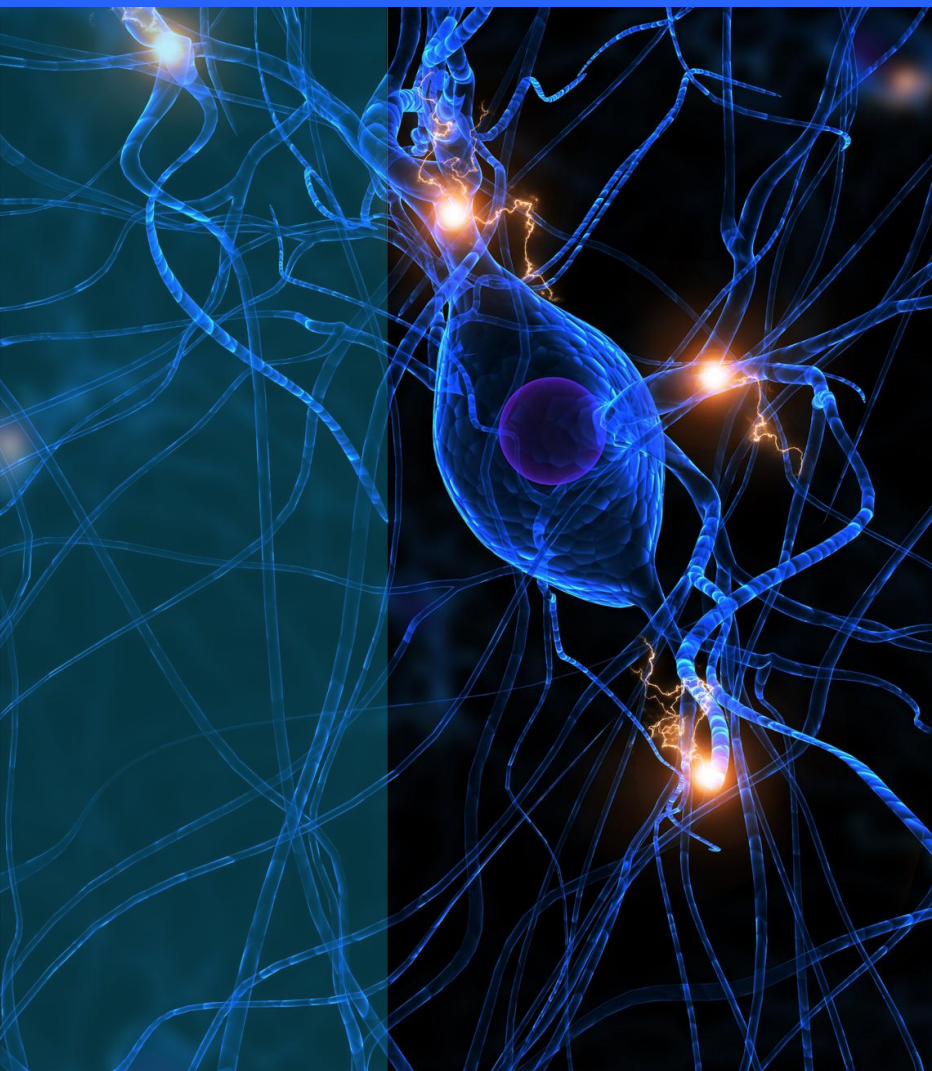
**ОБУЧАЮЩАЯ, ПРОВЕРОЧНАЯ
И ТЕСТОВАЯ ВЫБОРКИ**

Наборы данных для обучения



- **Обучающая выборка (training set)** – набор данных, который используется для обучения сети
- **Проверочная выборка (validation set)** – набор данных, который используется в процессе обучения для оценки качества обучения
- **Тестовая выборка (test set)** – набор данных, который используется для оценки качества работы сети после завершения обучения

Подбор гиперпараметров модели



Параметры модели

- Изменяются в процессе обучения
- Для нейронной сети – веса входов в нейроны

Гиперпараметры модели

- Не меняются в процессе обучения
- Влияют на конфигурацию модели и методы обучения
- Для нейронной сети – количество слоев, количество нейронов на слое, скорость обучения, размер мини-выборки

Подбор гиперпараметров

- Обучение на обучающей выборке, проверка на проверочной, изменение гиперпараметров
- Контрольная проверка на тестовой выборке



НАБОРЫ ДАННЫХ ДЛЯ ОБУЧЕНИЯ

Демонстрация использования обучающей,
проверочной и тестовой выборок в Keras

Схема обучения нейронной сети



Делим данные на три набора

- Обучающий, проверочный, тестовый

Обучаем модель на обучающем и проверочном наборе

- Для подбора гиперпараметров используем проверочный набор
- В процессе обучения мониторим качество на обучающем и проверочном наборе
- Если качество на обучающем наборе растет, а на проверочном падает – **началось переобучение**

Проверка обобщающей способности сети

- Оценка качества работы на тестовом наборе данных, которые сеть не видела в процессе обучения



ПЕРЕОБУЧЕНИЕ

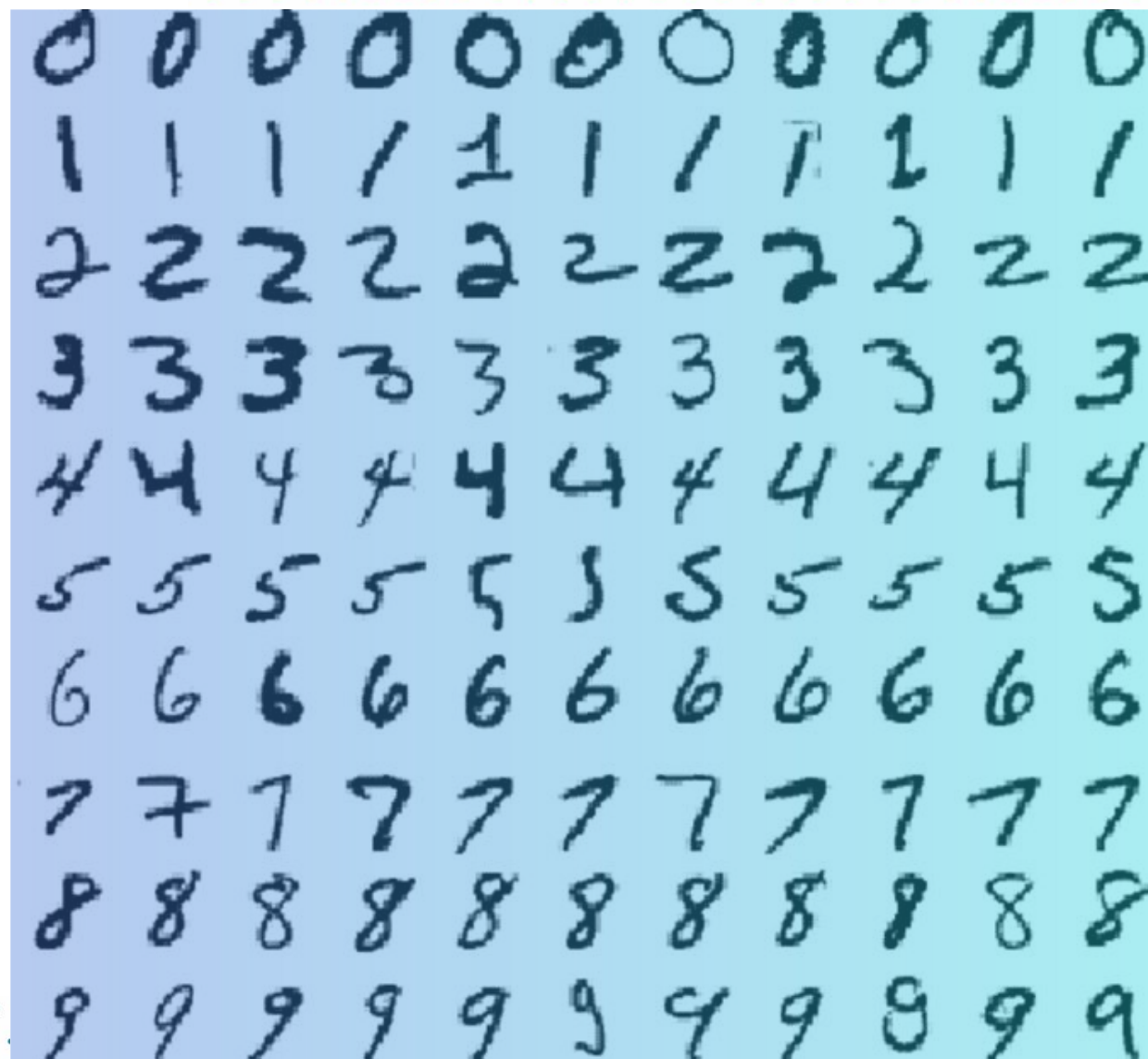
Точность распознавания



- Нейронная сеть распознает цифры, которые она видела, с точностью **100%**
- Это хорошо или плохо?

Проблема переобучения

Сеть может научиться
распознавать
особенности выборки,
а не данных



Причины переобучения



- Маленькая база
- Плохо собранная база
- Слишком сложная архитектура сети
- Разбалансировка базы

Как бороться с переобучением

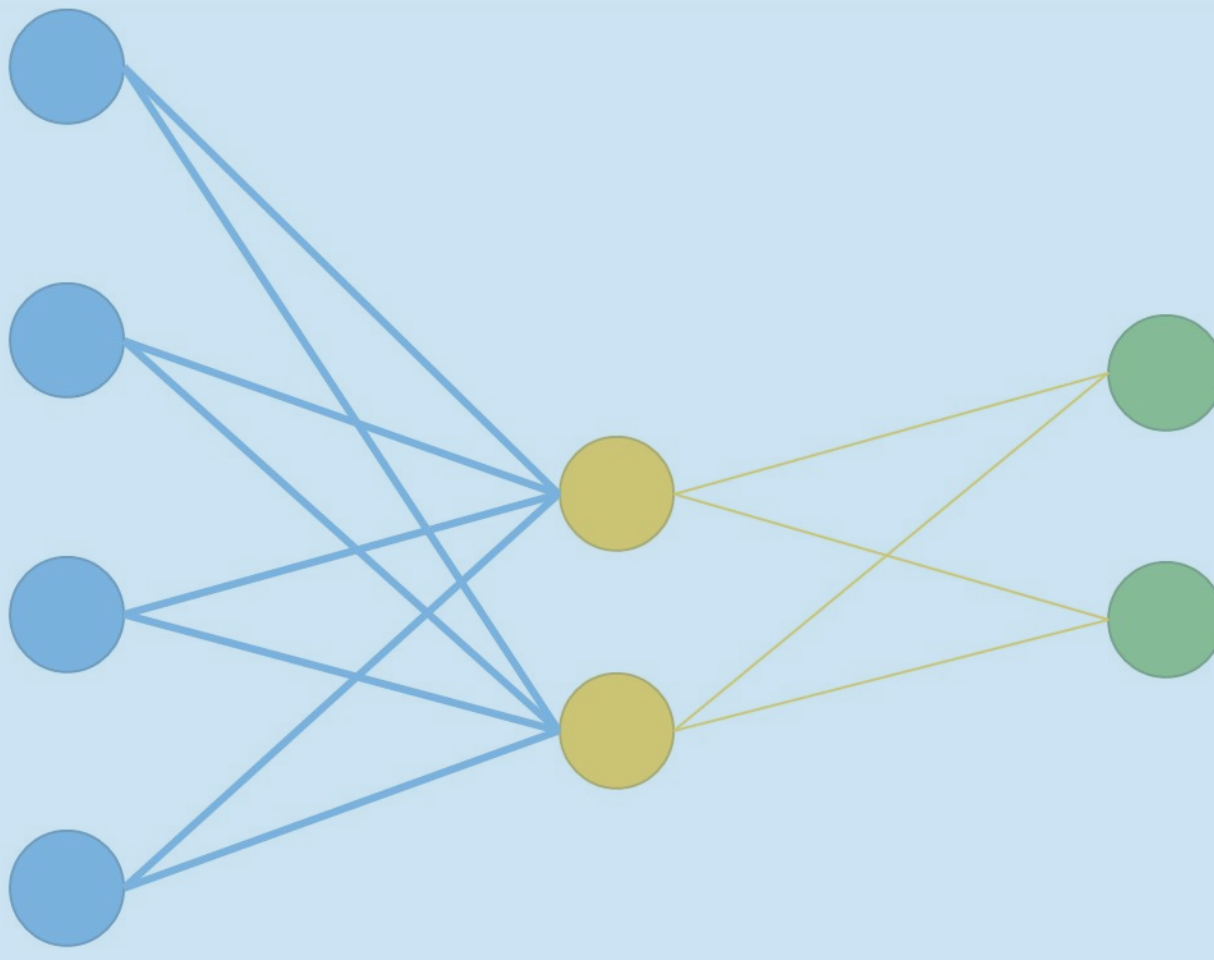


- Увеличивать базу
- Чистить базу
- Делать проще архитектуру сети
- Нормализовать данные
- Использовать Dropout
- Использовать BatchNormalization



СЛОЙ DROPOUT

СЛОЙ **Dropout**





Dropout в Keras

ВХОДНОЙ ПОЛНОСВЯЗНЫЙ СЛОЙ

```
model.add(Dense(800, input_dim=784,  
activation="relu"))
```

Слой Dropout

```
model.add(Dropout(0.5))
```

ВЫХОДНОЙ ПОЛНОСВЯЗНЫЙ СЛОЙ

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

Dropout: A Simple Way to Prevent Neural Networks from Overfitting.
<http://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf>



Слой BatchNormalization


Нормализация

Алгоритмы машинного обучения лучше работают с данными в нормализованном виде

Виды нормализации

- Данные в диапазоне от 0 до 1
- Данные со средним значением в 0 и стандартным отклонением 1





Пакетная нормализация в Keras

```
model = Sequential()
```

```
model.add(BatchNormalization(input_shape=(28, 28, 1)))
```

```
model.add(Conv2D(32, (3, 3), padding='same',  
activation='relu'))
```

```
model.add(Dropout(0.25))
```

```
model.add(BatchNormalization(input_shape=(28, 28, 1)))
```

```
model.add(Conv2D(32, (3, 3), padding='same',  
activation='relu'))
```

```
model.add(Dropout(0.25))
```


Предотвращение переобучения:

- Разделение данных на три набора: обучающий, проверочный, тестовый
- Слои Dropout и BatchNormalization
- Упрощение архитектуры сети
- Уменьшение шага обучения



СПАСИБО

ЗА ВНИМАНИЕ