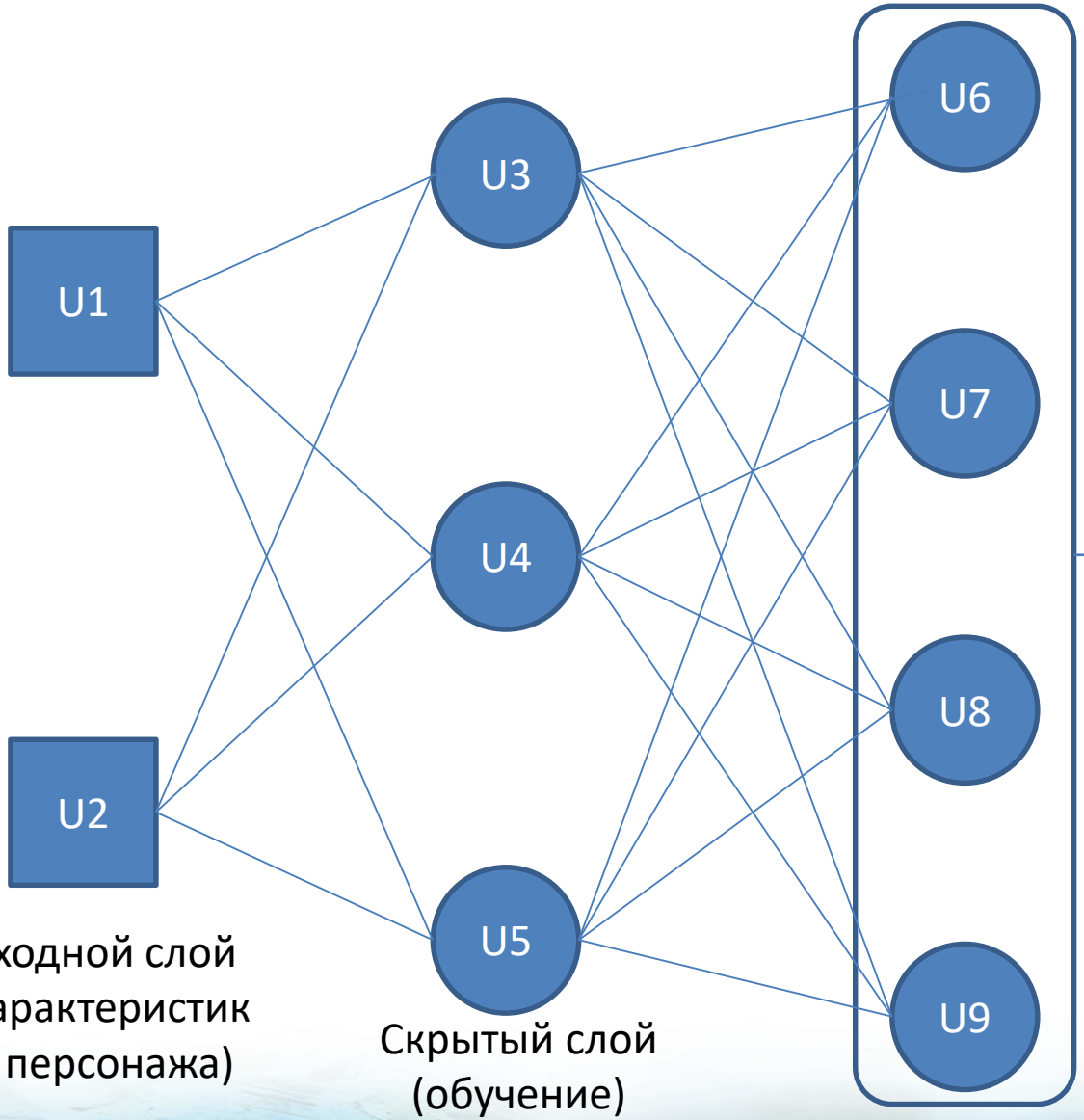
Машинное обучение и интеллектуальный анализ данных

Семинар 7
Лабораторная работа 5

Г.А. Ососков*, О.И. Стрельцова*, Д.И. Пряхина*,
Д.В. Подгайный*, А.В. Стадник*, Ю.А. Бутенко*
Государственный университет «Дубна»
***Лаборатория информационных технологий, ОИЯИ**
Дубна, Россия

Нейроконтроллер для персонажей компьютерных игр



Входной слой (характеристик и персонажа)

Скрытый слой (обучение)

Выходной слой

Выходная ячейка с наибольшей суммой весов является «победителем» и допускается к действию (активируется)

Постановка задачи

Расчет поведения искусственного интеллекта для компьютерных игр

Необходимо построить нейронную сеть для управления поведением персонажа компьютерной игры в зависимости от обстановки окружающей среды.

Исходные данные (data_train_lab5.csv и data_test_lab5.csv)

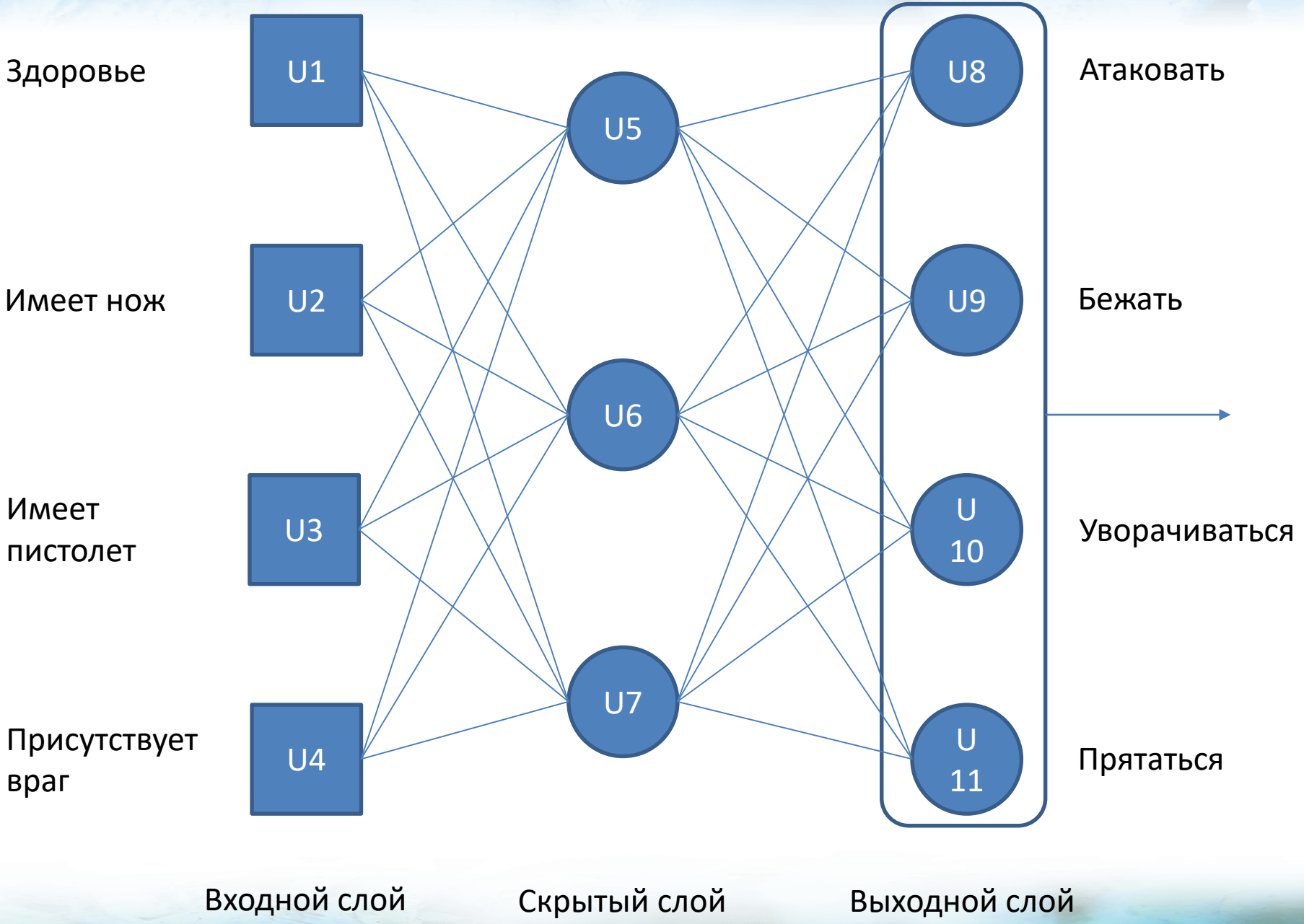
Обучающая и тестовые выборки представлены в виде таблиц:

- здоровье (значения от 0 (плохое) до 2 (среднее))
- имеет нож (1 – имеет, 0 - нет)
- имеет пистолет (1 – имеет, 0 - нет)
- присутствует враг (количество врагов)
- последний столбец показывает принадлежность классу, т.е. действие
    - уворачиваться
    - атаковать
    - прятаться
    - бежать

Пример: персонаж здоров, имеет пистолет и есть только один враг, то надо атаковать.

	Health	knife	gun	enemies	Beh
0	2	0	0	0	dodge
1	2	0	0	1	dodge
2	2	0	1	1	attack
3	2	0	1	2	attack
4	2	1	0	2	hide
5	2	1	0	1	attack
6	1	0	0	0	dodge
7	1	0	0	1	hide
8	1	0	1	1	attack
9	1	0	1	2	hide
10	1	1	0	2	hide
11	1	1	0	1	hide
12	0	0	0	0	dodge
13	0	0	0	1	hide
14	0	0	1	1	hide
15	0	0	1	2	run
16	0	1	0	2	run
17	0	1	0	1	hide

Архитектура нейронной сети



Алгоритм решения задачи

1. Загрузка данных (библиотека *pandas*)
2. Создание обучающей и тестовой выборок

```
X_train = dataT.iloc[:, 0:4]
print(X_train)
```

```
y_train = dataT.select_dtypes(include=[object])
print(y_train)
```

```
y_train.Beh.unique()
```

3. Преобразование данных для *Keras*

Преобразуем метки классов (возможные действия персонажа), представленные в виде строк, в набор чисел 0, 1, 2, 3

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()

train_labels2 = y_train.apply(le.fit_transform)

test_labels2 = y_test.apply(le.fit_transform)
```

```
print(train_labels2)
print(test_labels2)
```

Алгоритм решения задачи

Векторизовать метки можно одним из двух способов:
1) сохранить их в тензоре целых чисел,
2) использовать прямое кодирование.
Прямое кодирование (*onehot encoding*) широко используется для форматирования категорий и также называется кодированием категорий (*categorical encoding*).
В данном случае прямое кодирование меток заключается в конструировании вектора с нулевыми элементами со значением 1 в элементе, индекс которого соответствует индексу метки.
Преобразуем метки в тензорный объект, например, для 4 классов (с номерами 0, 1, 2, 3) метка класса 3 будет преобразована в массив: label3 -> [0, 0, 1, 0]

```
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models, utils
print (tf.__version__)
print (tf.keras.__version__)
```

```
from tensorflow.keras.utils import to_categorical
train_labels = to_categorical(train_labels2)
test_labels = to_categorical(test_labels2)
print(train_labels)
print(test_labels)
```

Алгоритм решения задачи

4. Построение модели нейронной сети

Модель в *Keras* можно описать двумя основными способами:

```
model = Sequential()
model.add(Dense(512, input_shape=(max_words,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

```
model = Sequential([
          Dense(512, input_shape=(max_words,)),
          Activation('relu'),
          Dropout(0.5),
          Dense(num_classes),
          Activation('softmax')
        ])
```

```
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import utils
```

```
model = models.Sequential()
model.add(layers.Dense(3, activation='relu', input_shape=(4,)))
model.add(layers.Dense(4, activation='softmax'))
```

```
from tensorflow.keras.utils import plot_model
plot_model(model, to_file='model.png', show_shapes=True)
```

Алгоритм решения задачи

Общая информация о созданной модели

```
model.summary()
```

Визуализация модели

```
from IPython.display import SVG
from tensorflow.python.keras.utils.vis_utils import model_to_dot
```

```
SVG(model_to_dot(model, show_shapes=True).create(prog='dot', format='svg'))
```

Компиляция модели

```
from tensorflow.keras.optimizers import SGD
opt=SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Обучение модели

```
history=model.fit(X_train, train_labels, validation_data=(X_test,test_labels), epochs=500,  batch_size=1)
```

Алгоритм решения задачи

Визуализация процесса обучения

```
import matplotlib.pyplot as plt

# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

Предсказание модели

```
ynew = model.predict_classes(X_test)
print(ynew)
print(test_labels2)
```

Содержание отчета

1. Постановка задачи.
2. Описание исходных данных.
3. Подготовка данных для работы с нейронной сетью.
4. Построение модели нейронной сети и ее визуализация (2 способа) по примеру.
5. Визуализация процесса обучения модели (см. графики).
6. Проверка модели на тестовом наборе данных. Выводы. Дополнить тестовую выборку 2-3 примерами, проверить модель, сделать выводы.
7. Построение моделей нейронной сети с разными функциями активации. Сравнение полученных результатов при проверке моделей на тестовых выборках. Выбор наилучшего варианта.
8. Исследование функций потерь и метрик по графикам в процессе обучения.
9. Список литературы.