Липецкий государственный технический университет

Кафедра прикладной математики

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ МАТЕМАТИЧЕСКИХ ИССЛЕДОВАНИЙ

Лекция 13

Deep Learning в R

Составитель - Сысоев А.С., к.т.н., доцент

Липецк - 2022

Outline

14.1. Искусственный интеллект, машинное обучение и deep learning

14.2. Нейронные сети

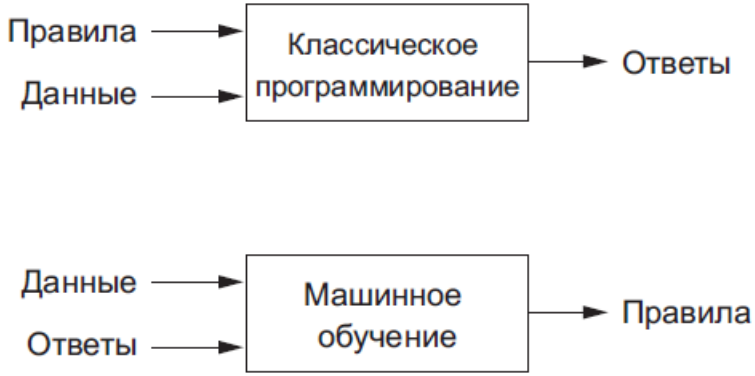
14.1. Искусственный интеллект, машинное обучение и deep learning

Искусственный интеллект. 1950-е гг. Попытка автоматизировать задачи, которые обычно решают люди.

Символический ИИ, 1950-1980-е гг.

Экспертные системы, 1980-е гг.

Машинное обучение. Алан Тьюринг, 1950-е гг. Может ли компьютер, не используя созданные наборы правил, изучить их, просто глядя на данные?



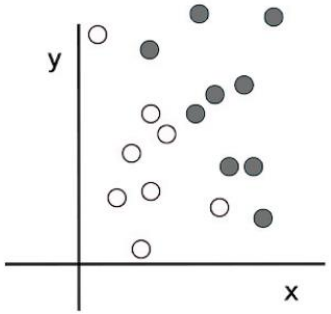
Глубинное обучение. Практически ориентированный раздел ИИ, содержит мало теоретических обоснований.

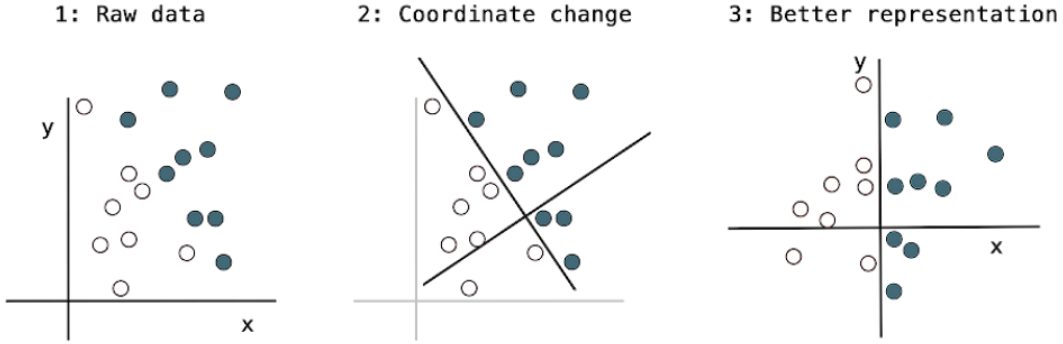
14.1. Искусственный интеллект, машинное обучение и deep learning

Для выполнения машинного обучения необходимо:

1. Входные данные
2. Примеры ожидаемых выходов, соотнесенных со входами
3. Способ «измерить» качество работы алгоритма

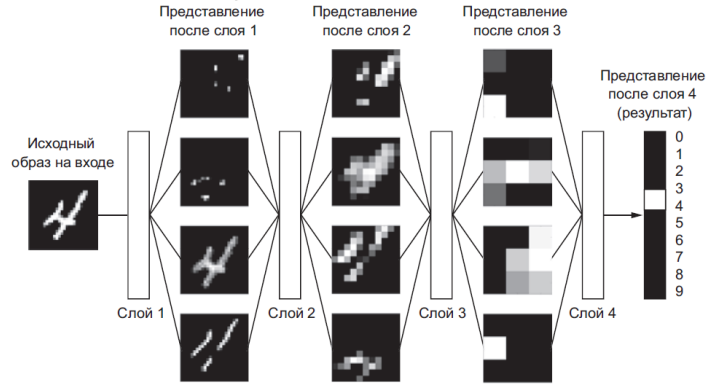
В ходе обучения происходит значительная трансформация данных (представление).



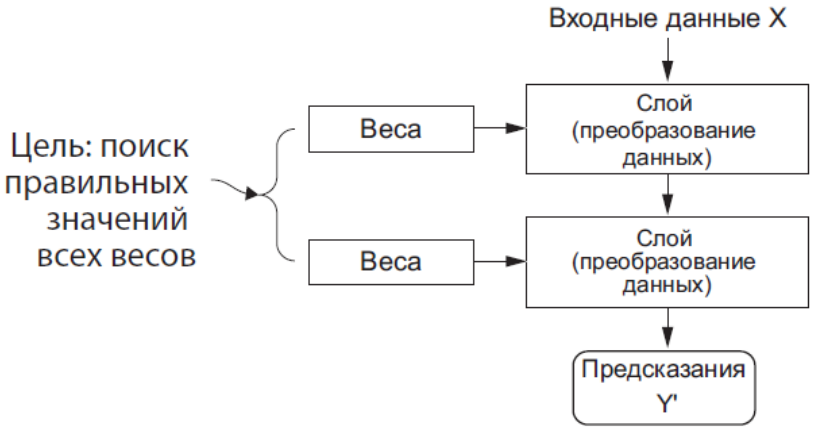


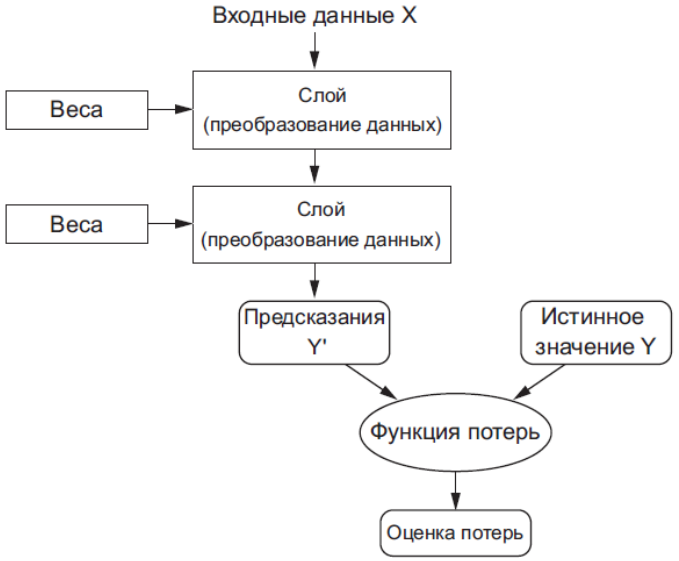
Глубинное обучение – не модель активности человеческого мозга! Этому нет подтверждений.

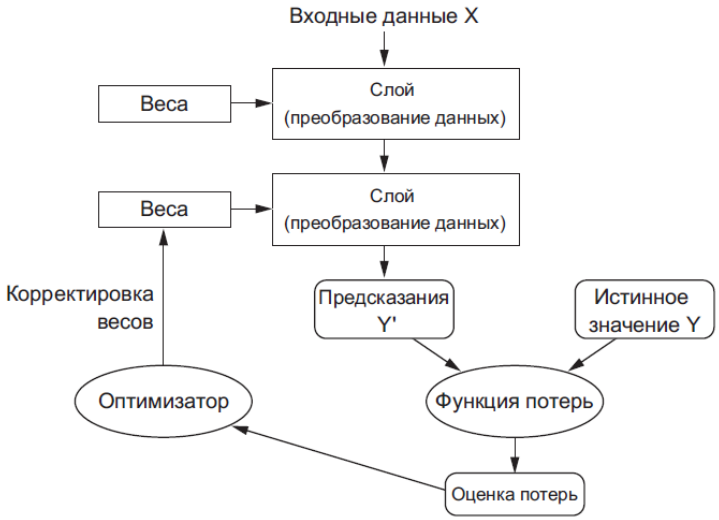
Глубинное обучение – это математический концепт выбора необходимого представления данных.



14.1. Искусственный интеллект, машинное обучение и deep learning



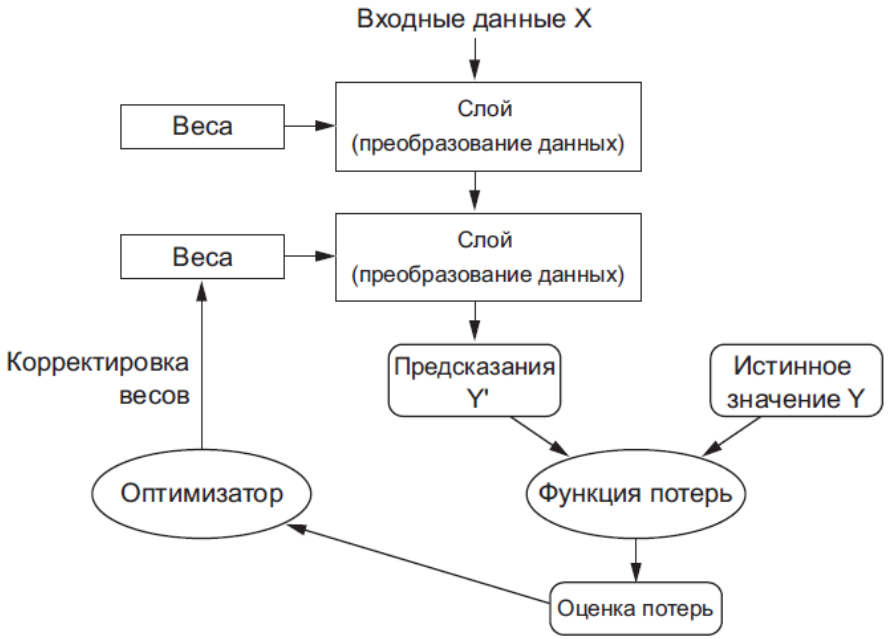




14.2. Нейронные сети

Обучение нейронных сетей сосредодточено вокруг следующих объектов:

- *слоев*, которые объединяются в сеть (или модель);
- *исходных данных* и соответствующих им *целей*;
- *функции потерь*, которая определяет сигнал обратной связи, используемый для обучения;
- *оптимизатора*, определяющего, как происходит обучение.



Слой — это модуль обработки данных, принимающий на входе и возвращающий на выходе один или несколько тензоров. Некоторые слои не сохраняют состояния, но чаще это не так: они имеют состояние веса слоя, один или несколько тензоров, обучаемых с применением алгоритма стохастического градиентного спуска, которые вместе хранят *знание сети*.

После того как мы определились с архитектурой сети, требуется также выбрать еще два параметра:

- *функцию потерь (целевую функцию)*, возвращающую количественную оценку, которая будет минимизироваться в процессе обучения. Представляет меру успеха в решении стоящей задачи;
- *оптимизатор*, определяющий, как будет изменяться сеть под воздействием функции потерь. Реализует конкретный вариант стохастического градиентного спуска (Stochastic Gradient Descent, SGD).

Набор данных IMDB. Множество 50 000 самых разных отзывов о фильмах в интернет-базе кинофильмов (Internet Movie Database). Набор разбит на 25 000 обучающих и 25 000 контрольных отзывов, каждый набор на 50 % состоит из отрицательных и на 50 % — из положительных отзывов.

Загрузка набора данных IMDB

```
library(keras)
imdb <- dataset_imdb(num_words = 10000)
c(c(train_data, train_labels), c(test_data, test_labels)) %<-% imdb
```

ОПЕРАТОР МНОЖЕСТВЕННОГО ПРИСВАИВАНИЯ (%<-%)

```
imdb <- dataset_imdb(num_words = 10000)
train_data <- imdb$train$x
train_labels <- imdb$train$y
test_data <- imdb$test$x
test_labels <- imdb$test$y
```

```
> str(train_data[[1]])
int [1:218] 1 14 22 16 43 530 973 1622 1385 65 ...

> train_labels[[1]]
[1] 1
```

декодирование одного из отзывов в последовательность слов на английском языке

word_index — это список именованных элементов, отображающий слова в целочисленные индексы

Получить обратное представление словаря, отображающее индексы в слова

Декодирование отзыва. Обратите внимание, что индексы смещены на 3, потому что индексы 0, 1 и 2 зарезервированы для слов «padding» (отступ), «start of sequence» (начало последовательности) и «unknown» (неизвестно)

```
word_index <- dataset_imdb_word_index()
reverse_word_index <- names(word_index)
names(reverse_word_index) <- word_index
decoded_review <- sapply(train_data[[1]], function(index) {
  word <- if (index >= 3) reverse_word_index[[as.character(index - 3)]]
  if (!is.null(word)) word else «?»
})
```

Кодирование последовательностей целых чисел в бинарную матрицу

```
vectorize_sequences <- function(sequences, dimension = 10000) {
  results <- matrix(0, nrow = length(sequences), ncol = dimension)
  for (i in 1:length(sequences))
    results[i, sequences[[i]]] <- 1
  results
}

x_train <- vectorize_sequences(train_data)
x_test <- vectorize_sequences(test_data)
```

Запись единицы в элемент с данным индексом

Создаст матрицу с формой (length(sequences), dimension)

```
> str(x_train[1,])
num [1:10000] 1 1 0 1 1 1 1 1 1 0 ...

y_train <- as.numeric(train_labels)
y_test <- as.numeric(test_labels)
```

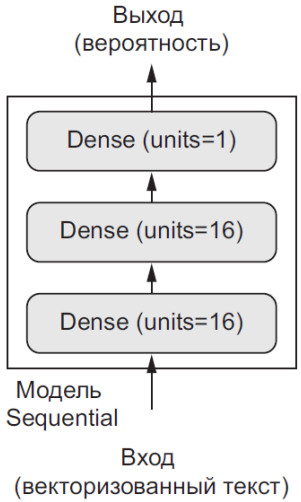
14.2. Нейронные сети

Входные данные представлены векторами, а метки — скалярами (единицами и нулями): это самый простой набор данных, какой можно встретить. С задачами этого вида прекрасно справляются сети, организованные как простой стек полносвязных уровней с операцией активации

```
relu: layer_dense(units = 16, activation = = "relu")
```

Два важных архитектурных решения:

- сколько слоев использовать;
- сколько скрытых единиц выбрать для каждого уровня




```
library(keras)

model <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = "relu", input_shape = c(10000)) %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")
```

14.2. Нейронные сети

Компиляция модели

```
model %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)
```

```
model %>% compile(
  optimizer = optimizer_rmsprop(lr=0.001),
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)
```

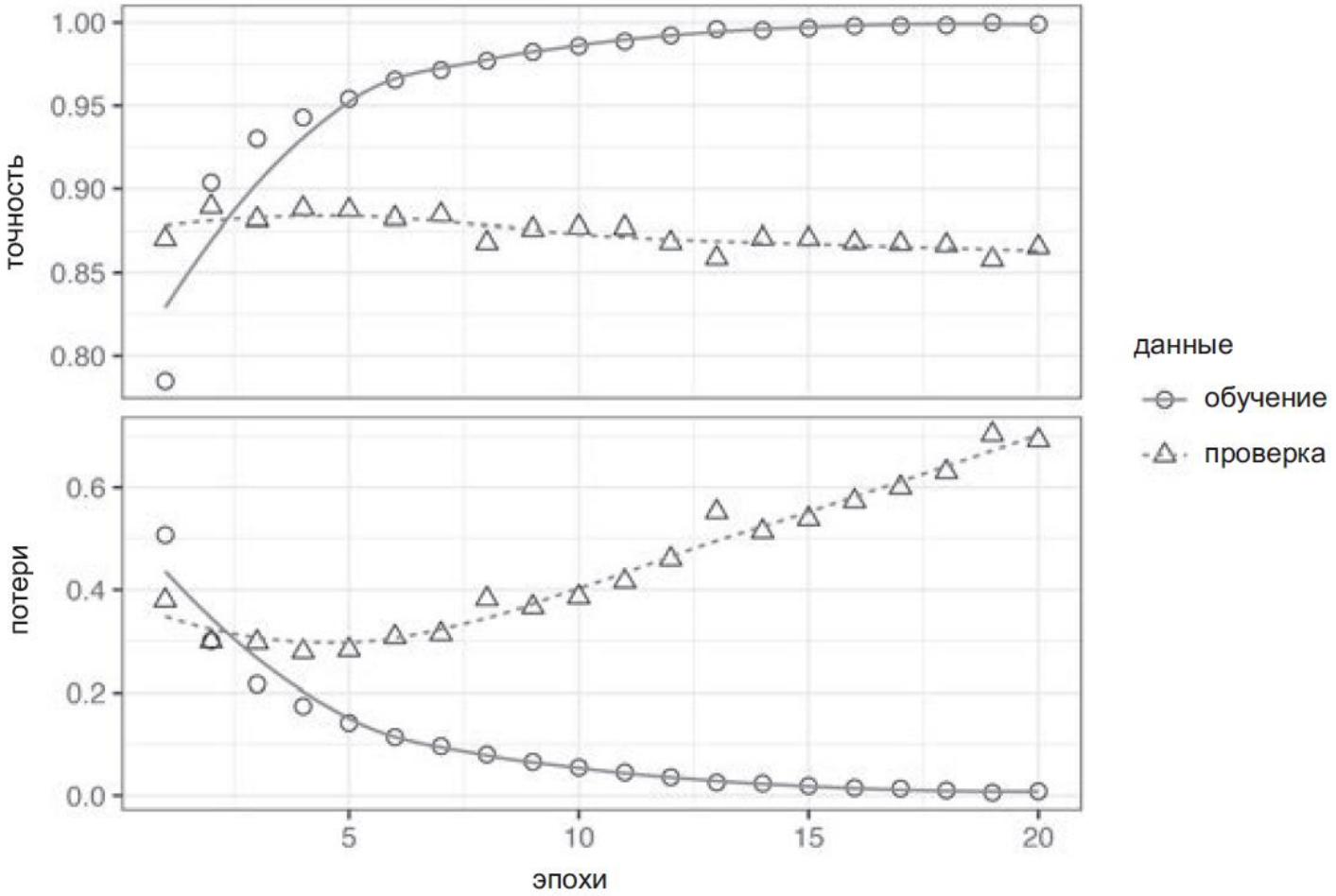
```
model %>% compile(
  optimizer = optimizer_rmsprop(lr = 0.001),
  loss = loss_binary_crossentropy,
  metrics = metric_binary_accuracy
)
```

14.2. Нейронные сети

Создание проверочного набора

```
val_indices <- 1:10000

x_val <- x_train[val_indices,]
partial_x_train <- x_train[-val_indices,]

y_val <- y_train[val_indices]
partial_y_train <- y_train[-val_indices]

model %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("accuracy")
)

history <- model %>% fit(
  partial_x_train,
  partial_y_train,
  epochs = 20,
  batch_size = 512,
  validation_data = list(x_val, y_val)
)
```

14.2. Нейронные сети



14.2. Нейронные сети

Использование обученной сети для предсказаний на новых данных

```
> model %>% predict(x_test[1:10,])
 [1,] 0.92306918
 [2,] 0.84061098
 [3,] 0.99952853
 [4,] 0.67913240
 [5,] 0.73874789
 [6,] 0.23108074
 [7,] 0.01230567
 [8,] 0.04898361
 [9,] 0.99017477
[10,] 0.72034937
```

- Обычно исходные данные приходится подвергать некоторой предварительной обработке, чтобы передать их в нейронную сеть в виде тензоров. Последовательности слов можно преобразовать в бинарные векторы.
- Стек полносвязных уровней с функцией активации relu способен решать широкий круг задач (включая классификацию эмоциональной окраски).
- В задаче бинарной классификации (с двумя выходными классами) одной единицей и функцией активации sigmoid: результатом сети должно быть скалярное значение в диапазоне между 0 и 1, представляющее вероятность.
- С таким скалярным результатом, получаемым с помощью сигмоидной функции, в задачах бинарной классификации следует использовать функцию потерь binary_crossentropy.
- В общем случае оптимизатор rmsprop является хорошим выбором для любых задач.
- По мере улучшения на обучающих данных нейронные сети рано или поздно начинают переобучаться, демонстрируя ухудшение результатов на данных, которые они прежде не видели. Поэтому всегда необходимо контролировать качество работы сети на данных не из обучающего набора.

Литература

Deep Learning with R [Francois Chollet, J. J. Allaire] 2017

Шолле Франсуаю. Глубокое обучение на R. — СПб.: Питер, 2018.