

Липецкий государственный технический университет

Кафедра прикладной математики

## **КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ МАТЕМАТИЧЕСКИХ ИССЛЕДОВАНИЙ**

Лекция 2.4

### **Deep Learning в R**

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

Липецк - 2018

# Outline

---

4.1. Разделы машинного обучения

4.2. Оценка моделей машинного обучения

4.3. Обработка данных, конструирование признаков и обучение признаков

4.4. Обобщенный процесс решения задач машинного обучения#

## 4.1. Разделы машинного обучения

---

### Контролируемое обучение

- Наиболее распространенный случай
- Научить модель отображать исходные данные в известные целевые значения (аннотации) на наборе примеров (часто аннотированных людьми)
- Основную долю задач контролируемого обучения составляют классификация и регрессия, но есть и другие:
  - Генерация последовательностей — по заданной картинке предсказать заголовок, который ее описывает
  - Прогнозирование дерева синтаксиса — по имеющемуся предложению требуется спрогнозировать его разложение в дерево синтаксиса
  - Распознавание объектов — на имеющейся картинке требуется нарисовать рамки вокруг определенных объектов
  - Сегментирование изображений — по имеющейся картинке построить пиксельную маску для конкретного объекта

## 4.1. Разделы машинного обучения

---

### Неконтролируемое обучение

- Поиск интересных преобразований входных данных без помощи каких-либо целевых значений для нужд визуализации, сжатия или очистки данных от шумов или для лучшего понимания взаимосвязей в данных.
- Неконтролируемое обучение — это основа анализа данных, оно часто оказывается необходимым шагом на пути изучения набора данных перед применением методов контролируемого обучения.
- Хорошо известными примерами неконтролируемого обучения являются *понижение размерности* и *кластеризация*.

### Самоконтролируемое обучение

- Контролируется без использования меток, расставленных человеком, — самоконтролируемое обучение можно считать контролируемым обучением без участия людей. Метки генерируются из исходных данных, обычно с применением эвристических алгоритмов.
- Хорошо известным примером самоконтролируемого обучения могут служить автокодировщики, которые генерируют цели по исходным, немодифицированным данным.

## 4.1. Разделы машинного обучения

---

### Обучение с подкреплением

- В обучении с подкреплением *агент* получает информацию о своем окружении и учится выбирать действия, максимизирующие некоторую выгоду.
- В настоящее время обучение с подкреплением пока является областью исследований и не имеет существенных практических успехов помимо применения в играх.

## 4.2. Оценка моделей машинного обучения

---

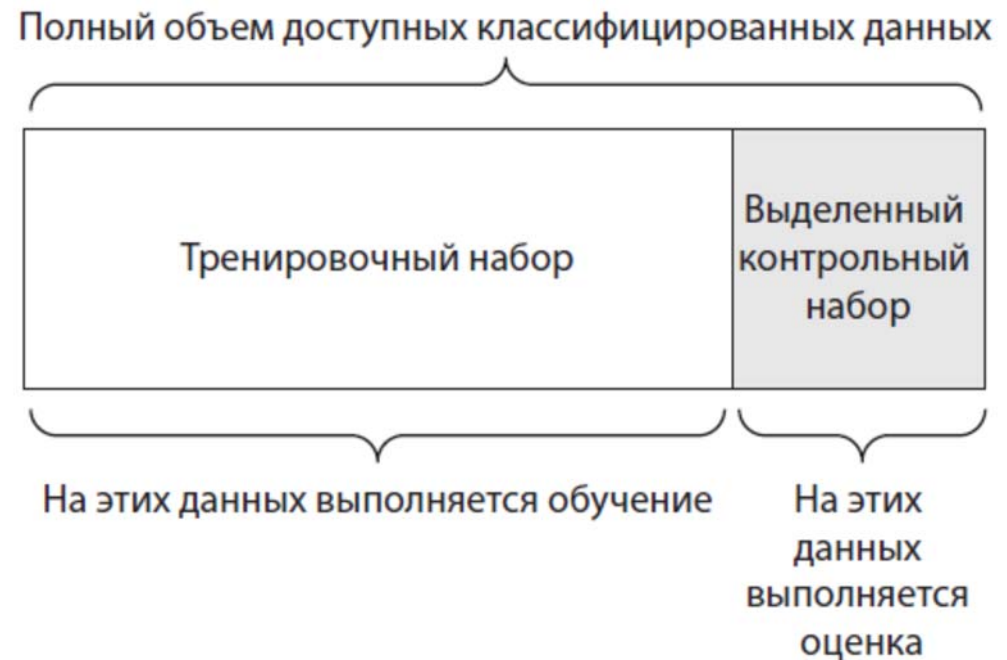
- Модели нельзя оценивать на тех же данных, на которых они обучались. Спустя всего несколько эпох возникает эффект переобучения.
- Цель машинного обучения состоит в создании обобщающих моделей, дающих качественный прогноз на данных, не участвовавших в обучении, а переобучение является главным препятствием к ее достижению.
- Оценка модели всегда сводится к делению доступных данных на три набора: тренировочный, проверочный и тестовый (или контрольный).
- Процесс конструирования модели всегда связан с настройкой ее параметров: например, с выбором количества слоев или изменением их размерности (такие настройки называют *гиперпараметрами* модели, чтобы отличать их от параметров — *весовых коэффициентов*).
- Модель не должна иметь доступа ни к какой информации из контрольного набора, даже косвенно.
- Проверка с простым расщеплением выборки (hold-out validation), перекрестная проверка по K блокам (K-fold validation) и итерационная проверка по K блокам с перемешиванием (iterated K-fold validation with shuffling).

## 4.2. Оценка моделей машинного обучения

---

### Проверка с простым расщеплением выборки

- Некоторая часть данных выделяется в контрольный набор. Обучение производится на оставшихся данных, а оценка качества — на контрольных.
- Для предотвращения утечек информации модель не должна настраиваться по результатам прогнозирования на контрольных данных, поэтому требуется также зарезервировать отдельный проверочный набор.



## 4.2. Оценка моделей машинного обучения

---

### Проверка с простым расщеплением выборки

```
indices <- sample(1:nrow(data), size = 0.80 * nrow(data))
evaluation_data <- data[-indices, ]
training_data <- data[indices, ]

model <- get_model()
model %>% train(training_data)
validation_score <- model %>% evaluate(validation_data)

model <- get_model()
model %>% train(data)
test_score <- model %>% evaluate(test_data)
```

Перемешивание данных нередко весьма желательно

← Определение проверочного набора

← Определение обучающего набора

Обучение модели на обучающих и оценка на проверочных данных

После настройки гиперпараметров часто желательно выполнить обучение окончательной модели на всех данных, не включенных в контрольный набор

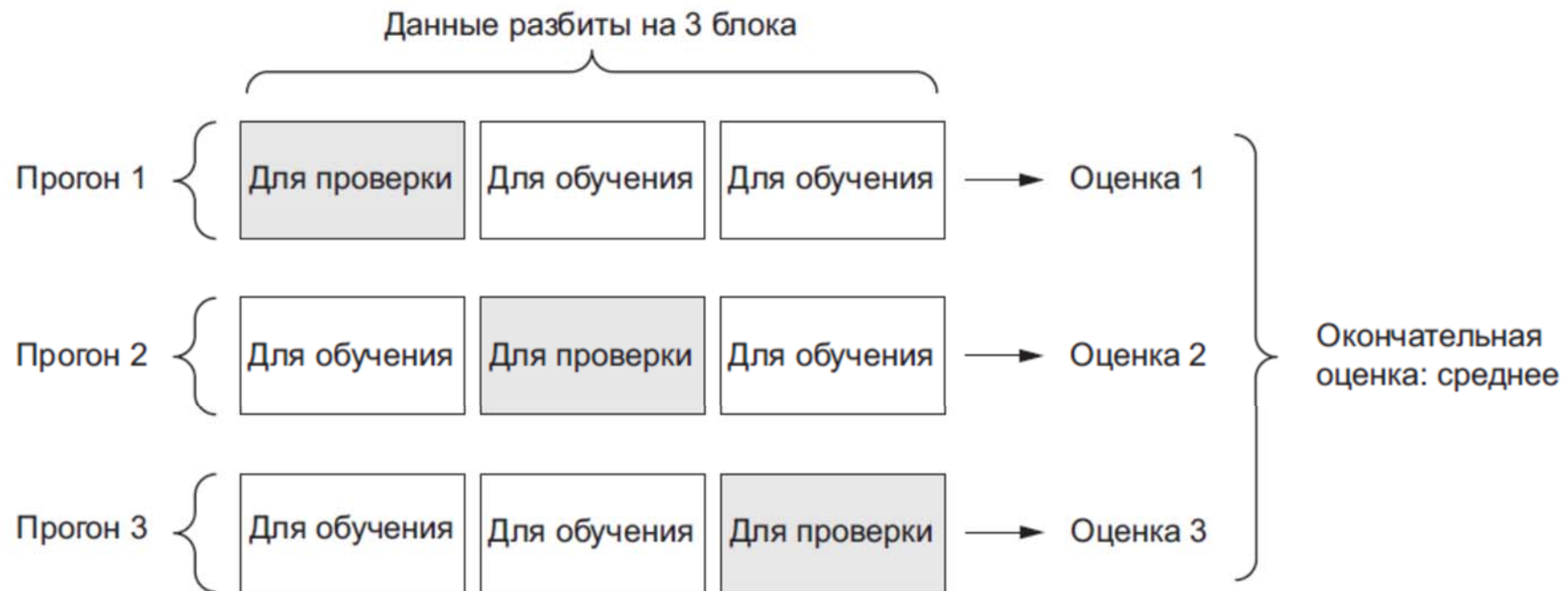
**НО! При небольшом объеме доступных данных проверочный и контрольный наборы могут содержать слишком мало образцов, чтобы считаться статистически репрезентативными.**



## 4.2. Оценка моделей машинного обучения

### Перекрестная проверка по $K$ блокам

- Данные разбиваются на  $K$  блоков равного размера.
- Для каждого блока  $i$  производится обучение модели на остальных  $K - 1$  блоках и оценка на блоке  $i$ .
- Окончательная оценка рассчитывается как среднее  $K$  промежуточных оценок.
- Этот метод может пригодиться, когда качество модели слишком сильно зависит от деления данных на тренировочный / контрольный наборы.



## 4.2. Оценка моделей машинного обучения

---

### Перекрестная проверка по K блокам

```
k <- 4
indices <- sample(1:nrow(data))
folds <- cut(indices, breaks = k, labels = FALSE)

validation_scores <- c()
for (i in 1:k) {

  validation_indices <- which(folds == i, arr.ind = TRUE)
  validation_data <- data[validation_indices,]
  training_data <- data[-validation_indices,]

  model <- get_model()
  model %>% train(training_data)
  results <- model %>% evaluate(validation_data)
  validation_scores <- c(validation_scores, results$accuracy)
}

validation_score <- mean(validation_scores)

model <- get_model()
model %>% train(data)
results <- model %>% evaluate(test_data)
```

Выбор блока данных для проверки

Использовать остальные данные для обучения

Создание совершенно новой (необученной) модели

Общая оценка: среднее оценок по K блокам

Обучение окончательной модели на всех данных, не вошедших в контрольный набор

## 4.2. Оценка моделей машинного обучения

---

### Итерационная проверка по $K$ блокам с перемешиванием

- Этот метод подходит для ситуаций, когда имеется относительно небольшой набор данных и требуется оценить модель максимально точно.
- Суть заключается в многократном применении перекрестной проверки по  $K$  блокам с перемешиванием данных перед каждым разделением на  $K$  блоков.
- Конечная оценка — среднее по оценкам, полученным в прогонах перекрестной проверки по  $K$  блокам.
- В конечном счете обучению и оценке подвергается  $P \times K$  моделей (где  $P$  — число итераций), что может быть очень затратным.

## 4.2. Оценка моделей машинного обучения

---

Выбирая протокол оценки, важно помнить:

- *О репрезентативности данных* — наборы тренировочных и контрольных данных должны быть репрезентативными для всего объема имеющихся данных.
- *О направлении оси времени* — пытаюсь предсказать будущее по прошлому (например, погоду на завтра, движение товаров и т. д.), нельзя производить перемешивание данных перед делением, потому что это создаст временную утечку: модель фактически будет обучаться по данным в будущем.
- *Об избыточности данных* — если некоторые образцы присутствуют в данных в нескольких экземплярах (частое явление в реальном мире), перемешивание и деление данных на тренировочный и проверочный наборы приведет к появлению избыточности между тренировочным и проверочным наборами. *Тренировочный и проверочный наборы не должны пересекаться.*

## 4.3. Обработка данных, конструирование признаков и обучение признаков

---

### Векторизация

- Все входы и цели в нейронной сети должны быть тензорами чисел с плавающей точкой (или, в особых случаях, тензорами целых чисел).

### Нормализация


- Чтобы упростить обучение сети, данные должны обладать следующими характеристиками:
  - *принимать небольшие значения* — как правило, значения должны находиться в диапазоне 0–1;
  - *быть однородными* — то есть все признаки должны принимать значения из примерно одного и того же диапазона.

```
mean <- apply(train_data, 2, mean)
std  <- apply(train_data, 2, sd)
```



Вычисление среднего и стандартного отклонения  
в обучающих данных

```
train_data <- scale(train_data, center = mean, scale = std)
test_data  <- scale(test_data, center = mean, scale = std)
```





Масштабирование обучающих и контрольных данных  
с использованием среднего и стандартного отклонения,  
вычисленных по обучающим данным

## 4.3. Обработка данных, конструирование признаков и обучение признаков

### Конструирование признаков

*Конструирование признаков* — это процесс использования собственных знаний о данных и алгоритме машинного обучения (в данном случае — нейронной сети), чтобы улучшить эффективность алгоритма применением predetermined преобразований к данным перед передачей их в модель. **Данные должны передаваться в модель в виде, облегчающем ее работу.**

|  |   |   |
|--|---|---|
| Исходные<br>данные: сетка<br>с пикселями                     |  |  |
| Более удачные<br>признаки:<br>координаты<br>стрелок          | $\{x1: 0.7,$<br>$y1: 0.7\}$<br>$\{x2: 0.5,$<br>$y2: 0.0\}$                        | $\{x1: 0.0,$<br>$y2: 1.0\}$<br>$\{x2: -0.38,$<br>$2: 0.32\}$                      |
| Еще более<br>удачные<br>признаки: углы<br>отклонения стрелок | theta1: 45<br>theta2: 0   | theta1: 90<br>theta2: 140   |

- Хорошие признаки позволяют решать задачи более элегантно и с меньшими затратами ресурсов. Хорошие признаки позволяют решать задачи, имея намного меньший объем исходных данных.
- Способность моделей глубокого обучения самостоятельно выделять признаки зависит от наличия большого объема исходных данных; если образцов всего несколько, то информационная ценность их признаков приобретает определяющее значение.

### 4.3. Обработка данных, конструирование признаков и обучение признаков

---

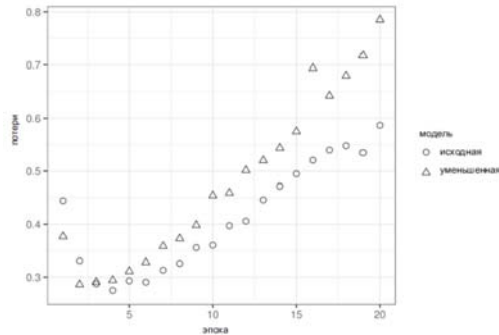
Основной проблемой машинного обучения является противоречие между *оптимизацией* и *общностью*.

Под оптимизацией понимается процесс настройки модели для получения максимального качества на тренировочных данных (обучение в машинном обучении), а под общностью — качество обученной модели на данных, которые она прежде не видела.

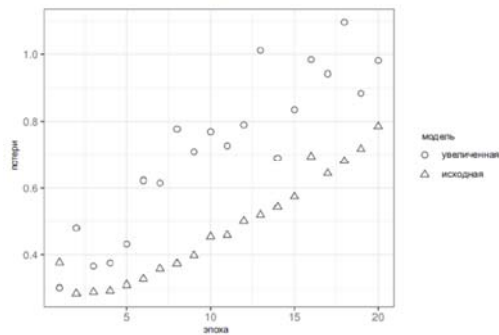
Лучший способ предотвратить изучение моделью специфических или нерелевантных шаблонов, имеющих место в тренировочных данных, — увеличить объем тренировочных данных. Борьба с переобучением таким способом называется *регуляризацией*.

### 4.3. Обработка данных, конструирование признаков и обучение признаков

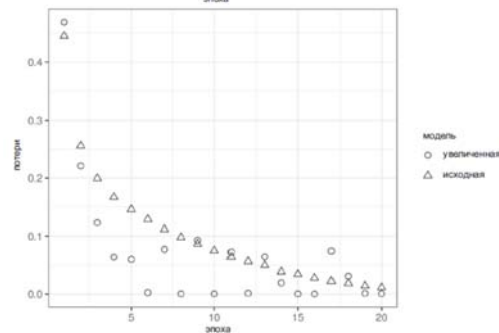
---



Влияние емкости модели на величину потерь на проверочных данных: попытка уменьшить модель



Влияние емкости модели на величину потерь на проверочных данных: попытка увеличить модель



Влияние емкости модели на величину потерь на обучающих данных: попытка увеличить модель



## 4.3. Обработка данных, конструирование признаков и обучение признаков

---

### Добавление регуляризации весов

Принцип бритвы Оккама: если какому-то явлению можно дать два объяснения, правильным, скорее всего, будет более простое — имеющее меньшее количество допущений. Простые модели менее склонны к переобучению.

Простая модель (в данном контексте) — это модель, в которой распределение значений параметров имеет меньшую энтропию (или модель с меньшим числом параметров).

- Типичный способ смягчения проблемы переобучения заключается в уменьшении сложности сети путем ограничения значений ее весовых коэффициентов, что делает их распределение более равномерным.

Этот прием называется **регуляризацией весов**, он реализуется добавлением в функцию потерь сети штрафа за увеличение весов и имеет две разновидности:

- L1-регуляризация (L1 regularization) — добавляемый штраф прямо пропорционален абсолютным значениям весовых коэффициентов (L1-норма весов).
- L2-регуляризация (L2 regularization) — добавляемый штраф пропорционален квадратам значений весовых коэффициентов (L2-норма весов).

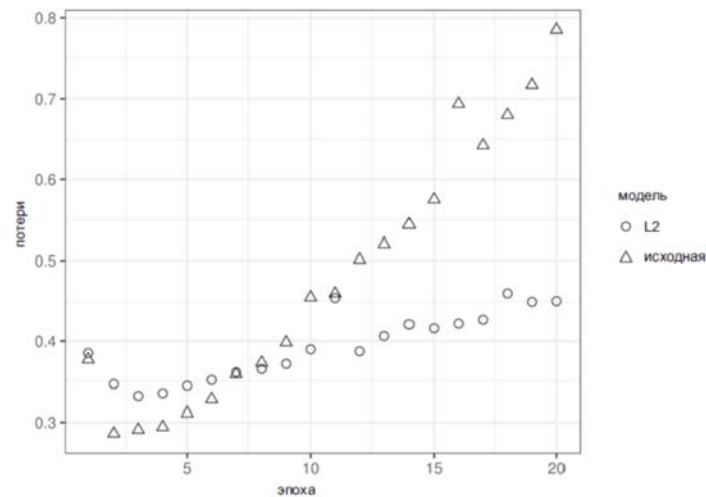
## 4.3. Обработка данных, конструирование признаков и обучение признаков

---

### Добавление регуляризации весов

```
model <- keras_model_sequential() %>%  
  layer_dense(units = 16, kernel_regularizer = regularizer_l2(0.001),  
              activation = "relu", input_shape = c(10000)) %>%  
  layer_dense(units = 16, kernel_regularizer = regularizer_l2(0.001),  
              activation = "relu") %>%  
  layer_dense(units = 1, activation = "sigmoid")
```

!!! так как штраф добавляется только на этапе обучения, величина потерь сети на этапе обучения будет намного выше, чем на этапе контроля.



### 4.3. Обработка данных, конструирование признаков и обучение признаков

---

#### Добавление прореживания

**Прореживание** (dropout) — один из наиболее эффективных и распространенных приемов регуляризации для нейронных сетей, разработанный Джеффом Хинтоном (Geoff Hinton) и его студентами в Университете Торонто. Прореживание, что применяется к уровню, заключается в удалении (присваивании нуля) случайно выбираемым признакам на этапе обучения.

Коэффициент прореживания — это доля обнуляемых признаков; обычно он выбирается в диапазоне от 0,2 до 0,5. На этапе тестирования прореживание не производится; вместо этого выходные значения уровня уменьшаются на коэффициент, равный коэффициенту прореживания, чтобы компенсировать разницу в активности признаков на этапах тестирования и обучения.

```
layer_output <- layer_output * sample(0:1, length(layer_output),
                                       replace = TRUE)

layer_output <- layer_output * 0.5

layer_output <- layer_output * sample(0:1, length(layer_output),
                                       replace = TRUE) ← На этапе обучения

layer_output <- layer_output / 0.5 ← Обратите внимание: в данном случае происходит
                                   увеличение, а не уменьшение значений.
```

## 4.3. Обработка данных, конструирование признаков и обучение признаков

---

### Добавление прореживания



```
layer_dropout(rate = 0.5)
model <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = "relu", input_shape = c(10000)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 1, activation = "sigmoid")
```

Наиболее распространенные способы ослабления проблемы переобучения нейронных сетей:

- увеличить объем обучающих данных;
- уменьшить емкость сети;
- добавить регуляризацию весов;
- добавить прореживание.

## 4.4. Обобщенный процесс решения задач машинного обучения

---

### Определение задачи и создание набора данных

- Какой вид будут иметь входные данные? Что требуется предсказать? Вы сможете обучить сеть предсказывать что-либо только при наличии тренировочных данных?#
  - К какому типу относится задача? Бинарная классификация? Многоклассовая классификация? Скалярная регрессия? Векторная регрессия? Многоклассовая, многозначная (нечеткая) классификация? Что-то иное, например, кластеризация, генерация или обучение с подкреплением? Идентификация типа задачи определит выбор архитектуры модели, функции потерь и т.д.
- ✓ гипотеза о том, что выходные данные можно предсказать по входным данным;
  - ✓ гипотеза о том, что доступные данные достаточно информативны для изучения отношений между входными и выходными данными

### Выбор меры успеха

Для задач симметричной классификации, когда каждый класс одинаково вероятен, часто используются такие показатели, как близость и площадь под кривой рабочей характеристики приемника. Для задач несимметричной классификации можно использовать точность и полноту. Для задач ранжирования или многозначной классификации можно использовать среднее математическое ожидание. Также нередко приходится определять собственную меру успеха.

## 4.4. Обобщенный процесс решения задач машинного обучения

---

Выбор протокола оценки

Предварительная подготовка данных

Разработка модели более совершенной, чем базовый случай

**Выбор функции активации для последнего уровня и функции потерь**

| Тип задачи                                | Функция активации для последнего слоя | Функция потерь              |
|---|---------------------------------------|-----------------------------|
| Бинарная классификация                    | sigmoid                               | binary_crossentropy         |
| Многоклассовая однозначная классификация  | softmax                               | categorical_crossentropy    |
| Многоклассовая многозначная классификация | sigmoid                               | binary_crossentropy         |
| Регрессия по произвольным значениям       | Нет                                   | mse                         |
| Регрессия по значениям между 0 и 1        | sigmoid                               | mse или binary_crossentropy |

Масштабирование по вертикали: разработка модели с переобучением

Регуляризация модели и настройка гиперпараметров